Financial Engineering Analytics: A Practice Manual Using R

DRAFT 1.3

William G. Foote

2018-01-08

Contents

Chapter 1

Introduction to Financial Analytics

> Science alone of all the subjects contains within itself the lesson of the danger of belief in the infallibility of the greatest teachers of the preceding generation. - Richard Feynman

This book is designed to provide students, analysts, and practitioners (the collective "we" and "us") with approaches to analyze various types of financial data sets, and to make meaningful decisions based on statistics obtained from the data. The book covers various areas in the financial industry, from analyzing credit data (credit card receivables), to studying global relations between macroeconomic events, to managing risk and return in multi-asset portfolios. The topics in the book employ a wide range of techniques including non-linear estimation, portfolio analytics, risk measurement, extreme value analysis, forecasting and predictive techniques, and financial modeling.

1.1 Analytics

By its very nature the science of data analytics is disruptive. That means, among many other things, that much attention should be paid to the scale and range of invalid, as yet not understood, outlying, and emerging trends. This is as true within the finance domain of knowledge as any other.

Throughout the book, we will learn the core of ideas of programming software development to implement financial analyses (functions, objects, data structures, flow control, input and output, debugging, logical design and abstraction) through writing code. We will learn how to set up stochastic simulations, manage data analyses, employ numerical optimization algorithms, diagnose their limitations, and work with and filter large data sets. Since code is also an important form of communication among analysts, we will learn how to comment and organize code, as well as document work product.

1.2 Chapter Outline

Here is an outline of topics covered by chapter.

2. R Warm-Ups for Finance. `R` computations, data structures, financial, probability, and statistics calculations, visualization. Documentation with `R Markdown`.

3. More R Warm-Ups. Functions, loops, control bootstrapping, simulation, and more visualization.

4. Stylized Facts of Financial Markets. Data from FRED, Yahoo, and other sources. Empirical characteristics of economic and financial time series. Boostrapping confidence intervals.

5. Term Structure of Interest Rates. Bond pricing, forward and yield curves. Estimating Non-linear regression splines. Applications.

6. Market Risk. Quantile (i.e., Value at Risk) and coherent (i.e., Expected Shortfall) risk measures. **7. Credit Risk.** Hazard rate models, Markov transition probabilities Risk measures, Laplace simulation with FFT.

8. Operational Risk and Extreme Finance. Generate frequency and severity of operational loss distributions. Estimating operational risk distribution parameters. Simulating loss distributions.

9. Measuring Volatility. Measuring volatility. GARCH estimation. GARCH simulation. Measuring Value at Risk (VaR) and Expected Shortfall (ES).

10. Portfolio Optimization. Combining risk management with portfolio allocations. Optimizing allocations. Simulating the efficient frontier.

11. Aggregating Enterprise Risks. Enterprise risk management analytics and application. Workflow to build an online application. Introduction to `Shiny` and `ShinyDashboard`. Building a simple app. Using `R Markdown` and `Shiny`.

1.3 Setting Up R for Analytics

1.3.1 Why this Appendix?

The general aim of this appendix is to situate the software platform `R` as part of your learning of statistics, operational research, and data analytics that accompanies nearly every domain of knowledge, from epidemiology to financial engineering. The specific aim of this appendix is to provide detailed instructions on how to install `R` an integrated development environment (IDE), `RStudio`, and a documentation system `R Markdown` on a personal computing platform (also known as your personal computer). This will enable us to learn the statistical concepts usually included in an analytics course with explanations and examples aimed at

the appropriate level. This appendix purposely does not attempt to teach you about R's many fundamental and advanced features.

1.3.2 Some useful R resources

There are many R books useful for managing implementation of models in this course. Three useful R books include:

1. Paul Teetor, *The R Cookbook*
2. Phil Spector, *Data Manipulation with R*
3. Norman Matloff, *The Art of R Programming: A Tour of Statistical Software Design*
4. John Taveras, *R for Excel Users* at https://www.rforexcelusers.com/book/.

The first one will serve as our R textbook. The other books are extremely valuable reference works. You will ultimately need all three (and whatever else you can get your hands on) in your professional work. John Taveras's book is an excellent bridge and compendium of Excel and R practices.

Much is available in books, e-books, and online for free. This is an extensive online user community that links expert and novice modelers globally.

1. The standard start-up is at CRAN http://cran.r-project.org/manuals.html. A script in the appendix can be dropped into a workspace and played with easily.
2. Julian Faraway's https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf is a fairly complete course on regression where you can imbibe deeply of the many ways to use R in statistics.
3. Along econometrics lines is Grant Farnsworth's https://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf.
4. Winston Chang's http://www.cookbook-r.com/ and Hadley Wickham's example at http://ggplot2.org/ are online graphics resources.
5. Stack Overflow is a programming user community with an R thread at http://stackoverflow.com/questions/tagged/r. The odds are that if you have a problem, error, or question, it has already been asked, and answered, on this site.
6. For using R Markdown there is a short reference at https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf. Cosma Shalizi has a much more extensive manual at http://www.stat.cmu.edu/~cshalizi/rmarkdown/.

1.3.3 Install R on your computer

Directions exist at the R website, http://cran.r-project.org/ for installing R. There are several twotorials, including some on installation that can be helpful at http://www.twotorials.com/.

Here are more explicit instructions that tell you what to do.

Download the software from the CRAN website. There is only one file that you need to obtain (a different file depending on the operating system). Running this file begins the installation process which is straight-forward in most, if not all, systems.

- Download `R` from the web. Go the `R` home page at http://cran.r-project.org/.
- If you have **Windows** (95 or later), then perform these actions. Click on the link `Windows (95 and later)`, then click on the link called `base`, and finally click on the most recent executable version. After the download is complete, double-click on the downloaded file and follow the on screen installation instructions. If you have a 64 bit system, use the 64 bit version of `R`.
- If you have **Macintosh** (OS X), then perform these actions. Click on the link `MacOS (System 8.6 to 9.1 and MacOS X)`, then click on the most recent package which begins the download. When given a choice to unstuff or save, choose save and save it on your desktop. Double-click on the downloaded file. Your Mac will unstuff the downloaded file and create an `R` folder. Inside this folder, there are many files including one with the `R` logo. You may drag a copy of this to your panel and then drag the whole `R` folder to your `Applications` folder (located on the hard drive). After completing this, you can drag the original downloaded file to your trash bin.

1.3.4 Install `RStudio`

Every software platform has a graphical user interface ("GUI" for short). One of the more popular GUIs, and the one used exclusively in this course, is provided by `RStudio` at http://www.rstudio.com. `RStudio` is a freely distributed integrated development environment (IDE) for `R`. It includes a console to execute code, a syntax-highlighting editor that supports direct code execution, as well as tools for plotting, reviewing code history, debugging code, and managing workspaces. In the following steps you will navigate to the `RStudio` website where you can download `R` and `RStudio`. These steps assume you have a `Windows` or `Mac OSX` operating system.

1. Click on https://www.rstudio.com/products/RStudio/ and navigate down to the `Download Desktop` button and click.
2. Click on the `Download` button for the `RStudio Desktop Personal License` choice.
3. Navigate to the sentence: "RStudio requires R 2.11.1+. If you don't already have R, download it *here*." If you have not downloaded `R` (or want to again), click on `here`. You will be directed to the https://cran.rstudio.com/ website in a new browser tab.

- In the `CRAN` site, click on `Download R for Windows`, or `Download R for (MAC) OS X` depending on the computer you use. This action sends you to a new webpage in the site.
- Click on `base`. This action takes you to the download page itself.

4. If you have `Windows`

- Click on `Download R 3.3.2 for Windows (62 megabytes, 32/64 bit)` (as of 11/8/2016; other version numbers may appear later than this date). A Windows installer in an over 70 MB `R-3.3.2-win.exe` file will download through your browser.
- In the Chrome browser, the installation-executable file will reside in a tray at the bottom of the browser. Click on the up arrow to the right of the file name and click `Open` in the list box. Follow the many instructions and accept default values throughout.
- Use the default `Core` and `32-Bit` files if you have a Windows 32-bit Operating System. You may want to use `64-Bit` files if that is your operating system architecture. You can check this out by going to the `Control Panel`, then `System and Security`, then `System`, and look up the `System Type:`. It may read for example `32-bit Operating System`.
- Click `Next` to accept defaults. Click `Next` again to accept placing `R` in the startup menu folder. Click `Next` again to use the `R` icon and alter and create registries. At this point the installer extracts files, creates shortcuts, and completes the installation.
- Click `Finish` to finish.

4. If you have a `MAC OS X`

- Click on `Download R 3.3.2 for MACs (62 megabytes, 32/64 bit)` (as of 11/8/2016; other version numbers may appear later than this date). A Windows installer in an over 70 MB `R-3.3.2-win.exe` file will download through your browser.
- When given a choice to unstuff or save, choose save and save it on your desktop. Double-click on the downloaded file. Your Mac will unstuff the downloaded file and create an `R` folder. Inside this folder, there are many files including one with the `R` logo.
- Inside the `R` folder drag a copy of `R` logo file to your panel and then drag the whole `R` folder to your `Applications` folder (located on the hard drive).

5. Now go back to `RStudio` browser tab. Click on `RStudio 1.0.44 - Windows Vista/7/8/10` or `RStudio 1.0.44 - MAC OS X` to download `RStudio`. Executiable files will download. Follow the directions exactly, and similarly, to the ones above.

1.3.5 Install `R Markdown`

Click on `RStudio` in your tray or start up menu. Be sure you are connected to the Internet. A console panel will appear. At the console prompt `>` type

```
install.packages("rmarkdown")
```

- This action will install the `RMarkdown` package. This package will enable you to construct documentation for your work in the course. Assignments will be documented using RMarkdown for submission to the learning management system.

- This extremely helpful web page, http://rmarkdown.rstudio.com/gallery.html, is a portal to several examples of `R Markdown` source files that can be loaded into `RStudio`, modified, and used with other content for your own work.

1.3.6 Install LaTex

`R Markdown` uses a text rendering system called `LaTeX` to render text, including mathematical and graphical content.

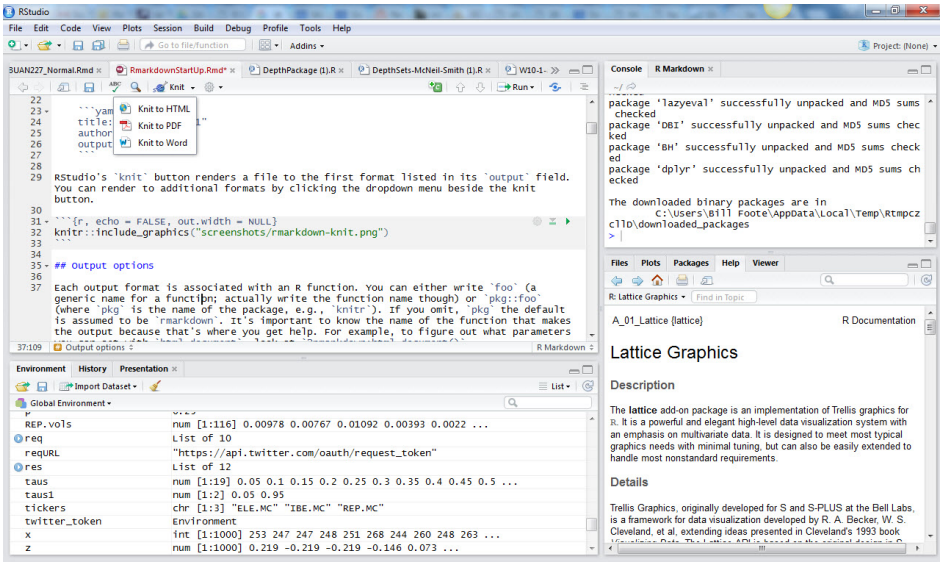
1. Install the `MikTeX` document rendering system for `Windows` or `MacTeX` document rendering system for `Mac OS X`.

- For `Windows`, navigate to the https://miktex.org/download page and go to the **64- or 32- bit** installer. Click on the appropriate `Download` button and follow the directions. **Be very sure you select the *COMPLETE* installation**. Frequently Asked Questions (FAQ) can be found at https://docs.miktex.org/faq/. If you have `RStudio` already running, you will have to restart your session.

- For `MAC OS X`, navigate to the http://www.tug.org/mactex/ page and download the `MacTeX` system and follow the directions. This distribution requires Mac OS 10.5 Leopard or higher and runs on Intel or PowerPC processors. **Be very sure you select the *FULL* installation**. Frequently Asked Questions (FAQ) can be found at https://docs.miktex.org/faq/. If you have `RStudio` already running, you will have to restart your session. FAQ can be found at http://www.tug.org/mactex/faq/index.html.

1.3.7 R Markdown

Open `RStudio` and see something like this screenshot…



- You can modify the position and content of the four panes by selecting `View > Panes > Pane Options`.
- Under `File > New File > Rmarkdown` a dialog box invites you to open document, presentation, Shiny, and other files. Upon choosing `documents` you may open up a new file. Under `File > Save As` save the untitle file in an appropriate directory. The `R Markdown` file extension `Rmd` will appear in the file name in your directory.
- When creating a new `Rmarkdown` file, `RStudio` deposits a template that shows you how to use the markdown approach. You can generate a document by clicking on `knit` in the icon ribbon attached to the file name tab in the script pane. If you do not see `knit`, then you might need to install and load the `knitr` package with the following statements in the `R` console. You might need also to restart your RStudio session.

```
install.packages("knitr")
library(knitr)
```

The `Rmd` file contains three types of content:

1. An (optional) **YAML header** surrounded by `---` on the top and the bottom of `YAML` statements. `YAML` is "Yet Another Markdown (or up) Language". Here is an example from this document:

```
---
title: "Setting Up R for Analytics"
author: "Bill Foote"
date: "November 11, 2016"
output: pdf_document
---
```

2. **Chunks** of R code surrounded by "‘ (find this key usually with the ~ symbol).
3. Text mixed with text formatting like `# heading` and `_italics_` and mathematical formulae like `$z = \frac{(\bar x-\mu_0)}{s/\sqrt{n}}$` which will render

$$z = \frac{(\bar{x} - \mu_0)}{s/\sqrt{n}}$$

.

When you open an `.Rmd` file, `RStudio` provides an interface where code, code output, and text documentation are interleaved. You can run each code chunk by clicking the `Run` icon (it looks like a play button at the top of the chunk), or by pressing `Cmd/Ctrl + Shift + Enter`. `RStudio` executes the code and displays the results in the console with the code.

You can write mathematical formulae in an `R Markdown` document as well. For example, here is a formula for net present value.

```
$$
NPV = \sum_{t=0}^{T} \frac{NCF_t}{(1+WACC)^t}
$$
```

This script will render

$$NPV = \sum_{t=0}^{T} \frac{NCF_t}{(1+WACC)^t}$$

- Here are examples of common in file text formatting in `R Markdown`.

```
Text formatting
------------------------------------------------------------

*italic*  or _italic_
**bold**   __bold__
`code`
superscript^2^ and subscript~2~

Headings
------------------------------------------------------------

# 1st Level Header

## 2nd Level Header

### 3rd Level Header

Lists
------------------------------------------------------------

*   Bulleted list item 1

*   Item 2
```

```
    * Item 2a

    * Item 2b

1.  Numbered list item 1

1.  Item 2. The numbers are incremented automatically in the output.

Links and images
--------------------------------------------------------------

<http://example.com>

[linked phrase](http://example.com)

![optional caption text](path/to/img.png)

Tables
--------------------------------------------------------------

First Header  | Second Header
------------- | -------------
Content Cell  | Content Cell
Content Cell  | Content Cell

Math
--------------------------------------------------------------

$\frac{\mu}{\sigma^2}$

\[\frac{\mu}{\sigma^2}]
```

More information can be found at the R Markdown web site.

1.3.8 jaRgon

(directly copied from Patrick Burns at http://www.burns-stat.com/documents/tutorials/impatient-r/jargon/, and annotated a bit, for educational use only.)

- atomic vector

An object that contains only one form of data. The atomic modes are: *logical*, *numeric*, *complex* and *character*.

- attach

The act of adding an item to the search list. You usually attach a package with the `require` function, you attach saved files and objects with the `attach` function.

- data frame

A rectangular data object where each column may be a different type of data. Conceptually a generalization of a matrix, but implemented entirely differently.

- factor

A data object that represents categorical data. It is possible (and often unfortunate) to confuse a factor with a character vector.

- global environment

The first location on the search list, and the place where objects that you create reside. See **search list**.

- list

A type of object with possibly multiple components where each component may be an arbitrary object, including a list.

- matrix

A rectangular data object where all cells have the same data type. Conceptually a specialization of a data frame, but implemented entirely differently. This object has rows and columns.

- package

A collection of `R` objects in a special format that includes help files and such. Most packages primarily or exclusively contain functions, but some packages exclusively contain datasets.

- search list

The collection of locations that `R` searches for objects when it is evaluating a command.

1.4 Nomenclature

Here is a table of symbols used throughout the text.

Symbol	Name	Usage
α	alpha	significance level
β	beta	regression coefficient, scale parameter
γ	gamma	autocorrelation
Γ	Gamma	Gamma function
δ	delta	delta
Δ	Delta	change
ϵ	espilon 1	is element of
ε	epsilon 2	error term
ξ	ksi	shape parameter
λ	lambda	hazard rate
μ	mu	mean of a distribution
ν	nu	number of degrees of freedom
π	pi	probability of occurrence
ρ	rho	correlation coefficient
σ	sigma	standard deviation
Σ	sum sigma	arithmetic sum
τ	tau	quantile

Chapter 2

R Warm-ups in Finance

2.1 Learning outcomes

By the end of this chapter you should be able to:

1. Use `R` to store and operate (arithmetic and logic) on arrays, vectors, and matrices
2. Apply `R` vectors to the calculation of net present value. ## Imagine This

You work for the division president of an aerospace company that makes testing equipment for high-tech manufacturers. Orders arrive "lumpy" at best as some quarters are big producers, others stretch the company's credit revolver.

The president, call her Nancy, found a new way to make money: lease equipment. This would help finance operations and smooth cash flow. You had a follow-on idea: build a captive finance company to finance and insure the manufactured product for customers.

"Nice ideas," Nancy quips. "Show me how we can make money."

For starters we want to borrow low and sell (leases) high! How? We can define three salient factors:

1. The "money factor"

- Sets the monthly payments
- Depends on the length and frequency of payment of the lease
- Also depends on the value of money at monthly forward borrowing rates

2. Residual value of the equipment

- Uncertain based on competitors' innovations, demand for manufacturers' products, etc.
- Uncertain based on quality of equipment at end of lease (is there a secondary market?)

3. Portfolio of leases

- By maturity

- By equipment class
- By customer segment

This simple vignette about leasing introduces us to the complexities and challenges of financial analytics. We see that cash flows evolve over time and may depend on various uncertain factors. The time series of cash flows and their underlying factors in turn have distributions of potential outcomes and also may be related to one another. Groups of cashflows may also be related to one another structurally through portfolios. These are simply combinations of sets of cashflows and, of course, their underlying and uncertain factors. Sets of cash flows and their factors may be delineated according to customer segments, classification of equipment, and introducing a timeing element, by their maturity.

2.1.1 Modeling process

Throughout financial analytics there is a modeling process we can deploy. Let's begin the modeling process by identifying leasing cash flow components and financial considerations we might use to begin to build reasonable scenarios. In ourleasing scenario here is a concordance of cash flow elements with models that will be illustrated throughout this book:

1. Lease payments

- Chapter 5: Term structure of interest rates
- Chapter 7: Credit risk of customers
- Chapter 4: Impact of economy on customers' market value

2. Residual cashflow

- Chapter 8: Operational risk
- Chapter 11: Aggregating risks

3. Borrowing costs

- Chapter 5: Term structure of interest rates
- Chapter 7: Our own credit risk

4. Collateral

- Chapter 10: Portfolio optimization
- Chapter 6: Market risk

5. Regulatory issues

- Chapter 8: Operational risk

6. Shareholder tolerance for risk

- Chapter 6: Market risk
- Chapter 9: Hedging
- Chapter 11: Capital requirements

In this first Chapter we will review some aspects of R programming to whet our appetite for further work in finance and risk. The first steps here will prepare the way for us to tackle the many issues encountered with financial time series, portfolios of risk and return factors, market, credit, and operational risk measurement, the aggregation of risk and return, and the fundamental measurement of volatility.

2.1.2 Chapter overview

In this first chapter we will

1. Introduce R and calculations, arrays, text handling, graphics
2. Review basic finance and statistics content
3. Use introductory R calculations in financial and statistical examples
4. Extend introductory calculations with further examples

2.1.3 What is R?

R is software for interacting with data along a variety of user generated paths. With R you can create sophisticated (even interactive) graphs, you can carry out statistical and operational research analyses, and you can create and run simulations. R is also a programming language with an extensive set of built-in functions. With increasing experience, you can extend the language and write your own code to build your own financial analytical tools. Advanced users can even incorporate functions written in other languages, such as C, C++, and Fortran.

The current version of R derives from the S language. S has been around for more than twenty years and has been with extrensive use in statistics and finance, first as S and then as the commercially available S-PLUS. R is an open source implementation of the S language that is now a viable alternative to S-PLUS. A core team of statisticians and many other contributors work to update and improve R and to make versions that run well under all of the most popular operating systems. Importantly, R is a free, high-quality statistical software that will be useful as you learn financial analytics even though it is also a first-rate tool for professional statisticians, operational researchers, and financial analysts and engineers.[1]But see this post on a truly big data language APL: https://scottlocklin.wordpress.com/2013/07/28/ruins-of-forgotten-empires-apl-languages/)

2.1.4 R for analytics

There are several reasons that make R an excellent choice of software for an analytics course. Some benefits of using R include:

[1](

- R is free and available online. R is open-source and runs on UNIX, Windows, and Macintosh operating systems.
- R has a well-documented, context-based, help system enhanced by a wide, and deep, ranging user community globally and across several disciplines.
- R has excellent native static graphing capabilities. Interactive dynamic graphics are evolving along with the ability to embed analytics into online applications. With R you can build dashboards and websites to communicate results dynamically with consumers of the analytics you generate.
- Practitioners can easily migrate to the commercially supported S-Plus program, if commercial software is required. S and S-Plus are the immediate ancestors of the R programming environment. Cloud computing is now available with large data implementations.
- R's language has a powerful, easy-to-learn syntax with many built-in statistical and operational research functions. Just as important are the extensive web-scraping, text structuring, object class construction, and the extensible functional programming aspects of the language. A formal language definition is being developed. This will yield more standardization and better control of the language in future versions.
- R is a computer programming language. For programmers it will feel more familiar than for others, for example Excel users. R requires array thinking and object relationships that are not necessarily native, but indeed are possible, in an Excel spreadsheet environment. In many ways, the Excel style and R style of environments complement one another.
- Even though it is not necessarily the simplest software to use, the basics are easy enough to master, so that learning to use R need not interfere with learning the statistical, operational research, data, and domain-specific concepts encountered in an analytics-focused course.

There is at least one drawback.

- The primary hurdle to using R is that most existing documentation and plethora of packages are written for an audience that is knowledgable about statistics and operational research and has experience with other statistical computing programs. In contrast, this course intends to make R accessible to you, especially those who are new to both statistical concepts and statistical computing.

2.1.5 Hot and cold running resources

Much is available in books, e-books, and online for free. This is an extensive online comm-Chaptery that links expert and novice modelers globally.

The standard start-up is at CRAN http://cran.r-project.org/manuals.html. A script in the appendix can be dropped into a workspace and played with easily. Other resources include

- Julian Faraway's https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf complete course on regression where you can imbibe deeply of the many ways to use R in statistics.
- Along econometrics lines is Grant Farnsworth's https://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf.
- Winston Chang's http://www.cookbook-r.com/ and Hadley Wickham's example at http://ggplot2.org/ are terrific online graphics resources.

2.2 Tickling the Ivories

Or if you paint and draw, the 2-minute pose will warm you up. In the RStudio console panel (in the SW pane of the IDE) play with these by typing these statements at the > symbol:

```
1 + (1:5)
```

```
## [1] 2 3 4 5 6
```

This will produce a vector from 2 to 6.

We can use alt- (hold alt and hyphen keys down simultaneously) to produce <-, and assign data to an new object. This is a from R's predecessor James Chamber's S (ATT Bell Labs) that was ported from the single keystroke ← in Ken Iverson's APL (IBM), where it is reserved as a binary logical operator. We can now also use = to assign variables in R. But, also a holdover from APL, we will continue to use = only for assignments within functions. [Glad we got that over!]

Now let's try these expressions.

```
x <- 1 + (1:5)
sum(x)
```

```
## [1] 20
```

```
prod(x)
```

```
## [1] 720
```

These actions assign the results of a calculation to a variable x and then sum and multiply the elements. x is stored in the active workspace. You can verify that by typing ls() in the console to list the objects in the workspace. Type in these statements as well.

```
ls()
```

```
## [1] "x"
```

```
length(x)
```

```
## [1] 5
```

```
x[1:length(x)]
```

```
## [1] 2 3 4 5 6
```

```
x[6:8]
```

```
## [1] NA NA NA
```

```
x[6:8] <- 7:9
x/0
```

```
## [1] Inf Inf Inf Inf Inf Inf Inf Inf
```

x has length of 5 and we use that to index all of the current elements of `x`. Trying to access elements 6 to 8 produces `na` because they do not exist yet. Appending 7 to 9 will fill the spaces. Dividing by 0 produces `inf`.

```
(x1 <- x - 2)
```

```
## [1] 0 1 2 3 4 5 6 7
```

```
x1
```

```
## [1] 0 1 2 3 4 5 6 7
```

```
x/x1
```

```
## [1]      Inf 3.000000 2.000000 1.666667 1.500000 1.400000 1.333333 1.285714
```

Putting parentheses around an expression is the same as printing out the result of the expression. Element-wise division (multiplication, addition, subtraction) produces `inf` as the first element.

2.2.1 Try this exercise

Suppose we have a gargleblaster machine that produces free cash flows of $10 million each year for 8 years. The machine will be scrapped and currently you believe you can get $5 million at the end of year 8 as salvage value. The forward curve of interest rates for the next 1 to 8 years is 0.06, 0.07, 0.05, 0.09, 0.09, 0.08, 0.08, 0.08.

1. What is the value of $1 received at the end of each of the next 8 years? Use this script to begin the modeling process. Describe each calculation.

```
rates <- c(0.06, 0.07, 0.05, 0.09, 0.09,
    0.08, 0.08, 0.08)
t <- seq(1, 8)
(pv.1 <- sum(1/(1 + rates)^t))
```

2. What is the present value of salvage? Salvage would be at element 8 of an 8-element cash flow vector, and thus would use the eighth forward rate, `rate[8]`, and `t` would be

8 as well. Eliminate the sum in the above script. Make a variable called `salvage` and assign salvage value to this variable. Use this variable in place of the `1` in the above script for `pv.1`. Call the new present value `pv.salvage`.

3. What is the present value of the gargleblaster machine? Type in these statements. The `rep` function makes an `8` element cash flow vector. We change the value of the 8th element of the cash flow vector to include salvage. Now use the `pv.1` statement above and substitute `cashflow` for `1`. You will have your result.

```
cashflow <- rep(10, 8)
cashflow[8] <- cashflow[8] + salvage
```

Some results follow. The present value of $1 is The present value of a $1 is this mathemetical formula.

$$PV = \sum_{t=1}^{8} \frac{1}{(1+r)^t}$$

This mathematical expression can be translated into `R` this way

```
rates <- c(0.06, 0.07, 0.05, 0.09, 0.09,
    0.08, 0.08, 0.08)
t <- seq(1, 8)
(1/(1 + rates)^t)
```

```
## [1] 0.9433962 0.8734387 0.8638376 0.7084252 0.6499314 0.6301696 0.5834904
## [8] 0.5402689
```

```
(pv.1 <- sum(1/(1 + rates)^t))
```

```
## [1] 5.792958
```

We define `rates` as a vector using the `c()` concatenation function. We then define a sequence of 8 time indices `t` starting with 1. The present value of a $1 is sum of the vector element-by-element calculation of the date by date discounts $1/(1+r)^t$.

The present value of salvage is the discounted salvage that is expected to occur at, and in this illustration only at, year 8.

$$PV_{salvage} = \frac{salvage}{(1+r)^8}$$

Translated into `R` we have

```
salvage <- 5
(pv.salvage <- salvage/(1 + rates[8])^8)
```

```
## [1] 2.701344
```

The present value of the gargleblaster machine is the present value of cashflows from operations from year 1 to year 8 plus the present value of salvage received in year 8. Salvage by

definition is realized at the of the life of the operational cashflows upon disposition of the asset, here at year 8.

$$PV_{total} = \sum_{t=1}^{8} \frac{cashflow_t}{(1+r)^t} + \frac{salvage}{(1+r)^8}$$

This expression translates into R this way:

```
cashflow <- rep(10, 8)
cashflow[8] <- cashflow[8] + salvage
(pv.machine <- sum(cashflow/(1 + rates)^t))
```

```
## [1] 60.63092
```

The rep or "repeat" function creates cash flows of $10 for each of 8 years. We adjust the year 8 cash flow to reflect salvage so that $cashflow_8 = 10 + salvage$. The [8] indexes the eighth element of the cashflow vector.

2.3 Building Some Character

Let's type these expressions into the console at the > prompt:

```
x[length(x) + 1] <- "end"
x[length(x) + 1] <- "end"
x.char <- x[-length(x)]
x <- as.numeric(x.char[-length(x.char)])
str(x)
```

```
##  num [1:8] 2 3 4 5 6 7 8 9
```

We have appended the string "end" to the end of x, twice.

- We use the - negative operator to eliminate it.
- By inserting a string of characters into a numeric vector we have forced R to transform all numerical values to characters.
- To keep things straight we called the character version x.char.
- In the end we convert x.char back to numbers that we check with the str(ucture) function.

We will use this procedure to build data tables (we will call these "data frames") when comparing distributions of variables such as stock returns.

Here's a useful set of statements for coding and classifying variables. Type these statements into the console.

```
set.seed(1016)
n.sim <- 10
x <- rnorm(n.sim)
y <- x/(rchisq(x^2, df = 3))^0.5
```

We did a lot of R here. First, we set a random seed to reproduce the same results every time we run this simulaton. Then, we store the number of simulations in n.sim and produced two new variables with normal and a weirder looking distribution (almost a Student's t distribution). Invoking help will display help with distributions in the SE pane of the RStudio IDE.

```
z <- c(x, y)
indicator <- rep(c("normal", "abnormal"),
    each = length(x))
xy.df <- data.frame(Variates = z, Distributions = indicator)
```

Next we concatenate the two variables into a new variable z. We built into the variable indicator the classifier to indicate which is x and which is y. But let's visualize what we want. (Paint in words here.) We want a column the first n.sim elements of which are x and the second are y. We then want a column the first n.sim elements of which are indicated by the character string "normal", and the second n.sim elements by "abnormal".

The rep function replicates the concatenation of "normal" and "abnormal" 10 times (the length(x)). The each feature concatenates 10 replications of "normal" to 10 replications of "abnormal". We concatenate the variates into xy with the c() function.

We can see the first 5 components of the data frame components using the $ subsetting notation as below.

```
str(xy.df)
```

```
## 'data.frame':    20 obs. of  2 variables:
##  $ Variates     : num  0.777 1.373 1.303 0.148 -1.825 ...
##  $ Distributions: Factor w/ 2 levels "abnormal","normal": 2 2 2 2 2 2 2 2 2 2 ...
```

```
head(xy.df$Variates, n = 5)
```

```
## [1]  0.7773788  1.3733067  1.3025762  0.1482796 -1.8251426
```

```
head(xy.df$Distributions, n = 5)
```

```
## [1] normal normal normal normal normal
## Levels: abnormal normal
```

The str call returns the two vectors inside of xy. One is numeric and the other is a "factor" with two levels. R and many of the routines in R will interpret these as zeros and ones in developing indicator and dummy variables for regressions and filtering.

2.3.1 The plot thickens

We will want to see our handiwork, so load the ggplot2 library using install.packages("ggplot2"). [2]Visit Hadley Wickham's examples at http://ggplot2.org/.)

[2](

This plotting package requires data frames. A "data frame" simply put is a list of vectors and arrays with names. An example of a data frame in Excel is just the worksheet. There are columns with names in the first row, followed by several rows of data in each column.

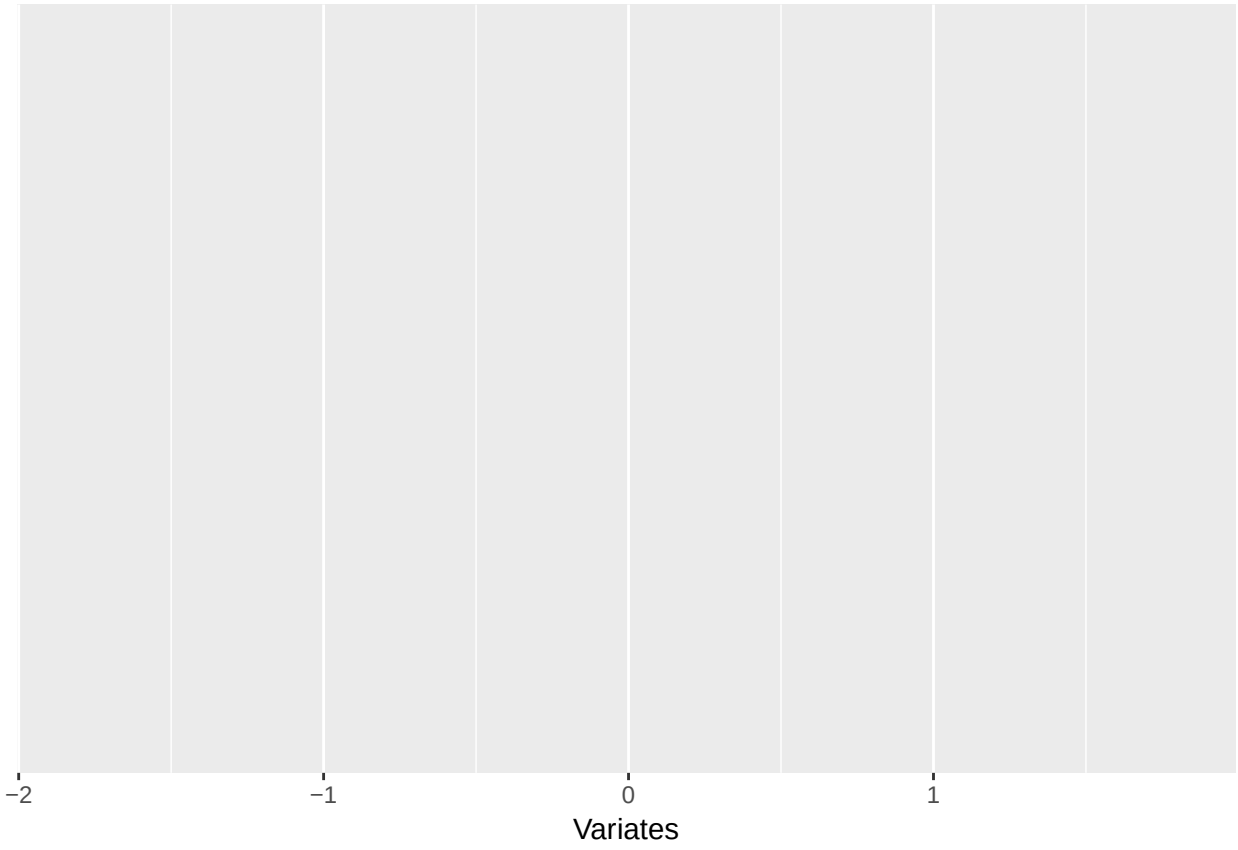
Here we have defined a data frame `xy.df`. All of the `x` and `y` variates are put into one part of the frame, and the distribution indicator into another. For all of this to work in a plot the two arrays must be of the same length. Thus we use the common `n.sim` and `length(x)` to insure this when we computed the series. We always examine the data, here using the `head` and `tail` functions.

Type `help(ggplot)` into the console for details. The `ggplot2` graphics package embodies Hadley Wickham's "grammar of graphics" we can review at http://ggplot2.org. Hadley Wickham has a very useful presentation with numerous examples at http://ggplot2.org/resources/2007-past-present-future.pdf.

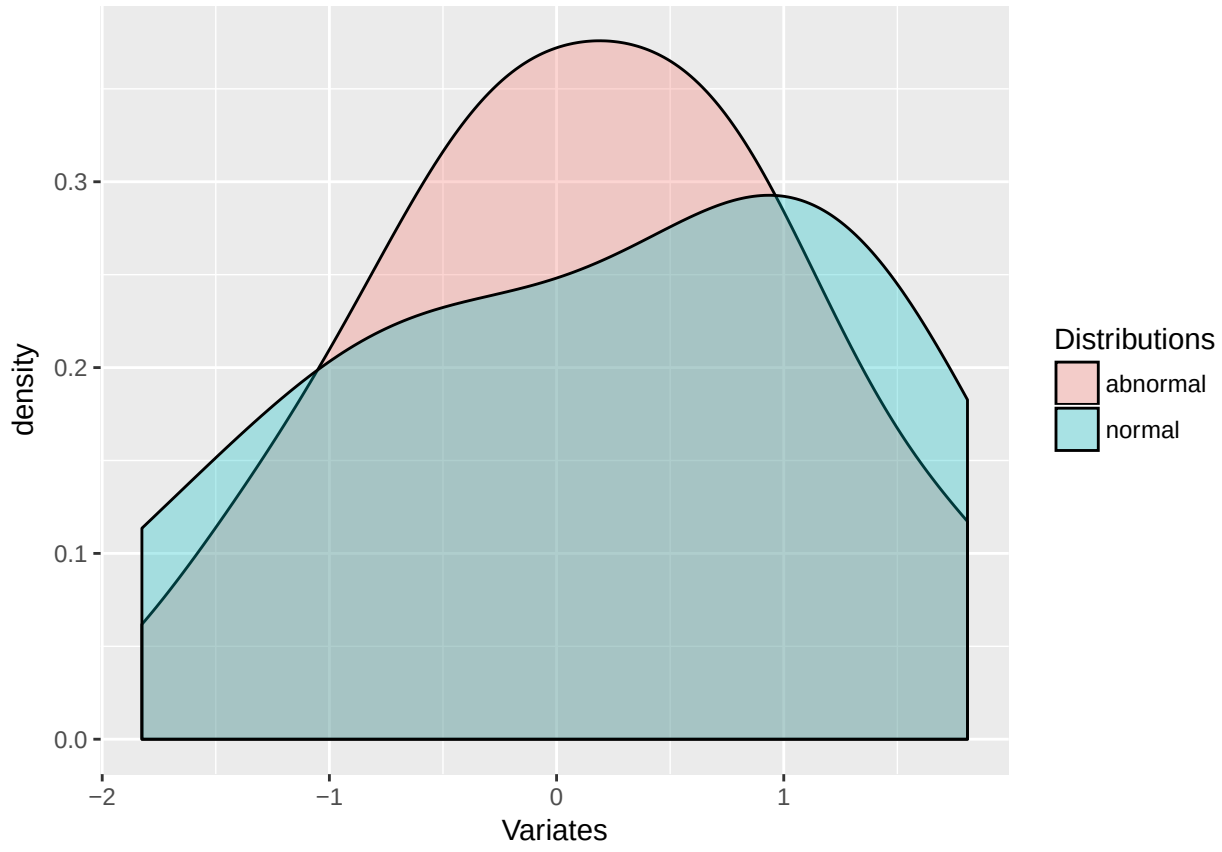
As mentioned above, the package uses data frames to process graphics. A lot of packages other than `ggplot2`, including the base `stats` package, require data frames.

We load the library first. The next statement sets up the blank but all too ready canvas (it will be empty!) on which a density plot can be rendered.

```
library(ggplot2)
ggplot(xy.df, aes(x = Variates, fill = Distributions))
```



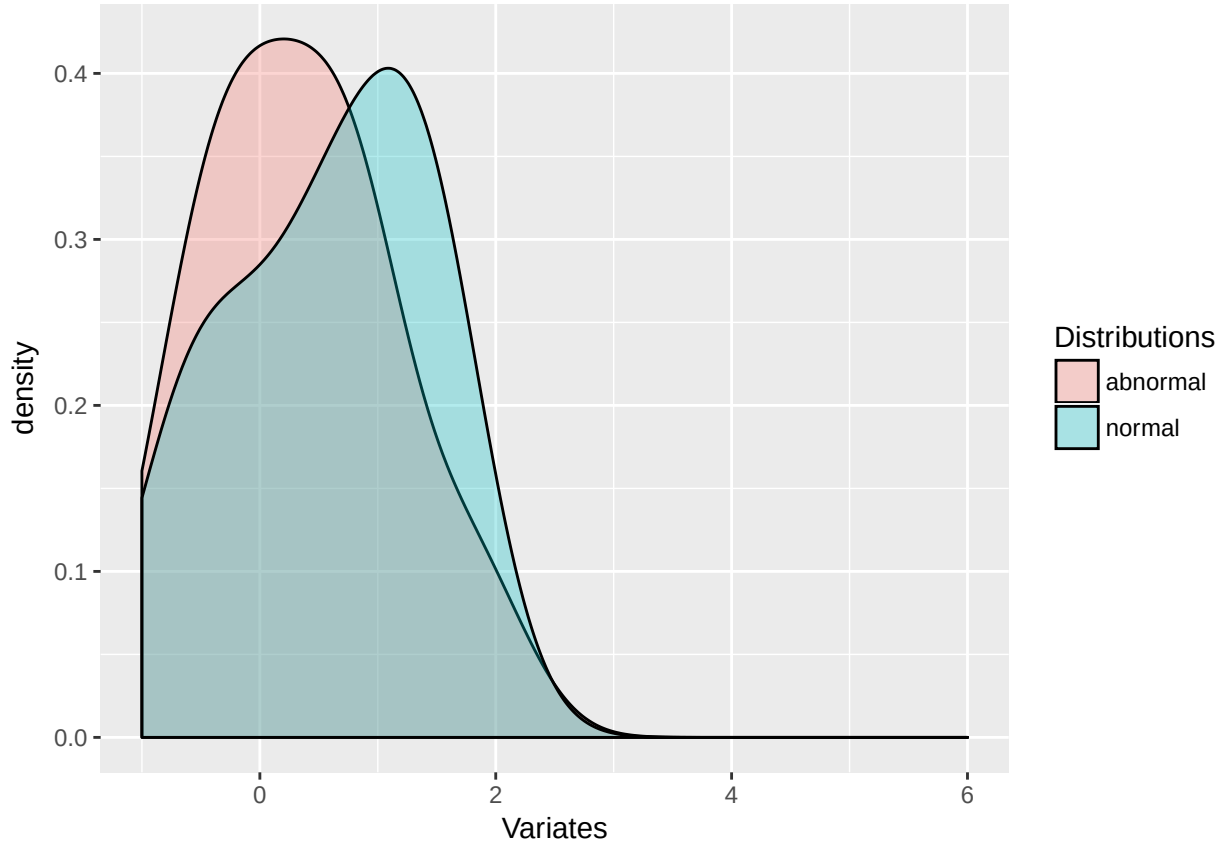
The data frame name `xy.df` is first followed by the aesthetics mapping of data. The next statement inserts a geometrical element, here a density curve, which has a transparency parameter aesthetic `alpha`.



2.3.2 Try this example

Zoom in with `xlim` and lower x-axis and upper x-axis limits using the following statement:

```
ggplot(xy.df, aes(x = Variates, fill = Distributions)) +
    geom_density(alpha = 0.3) + xlim(-1,
    6)
```



Now we are getting to extreme finance by visualizing the tail of this distribution.

2.4 Arrays and You

Arrays have rows and columns and are akin to tables. All of Excel's worksheets are organized into cells that are tables with columns and rows. Data frames are more akin to tables in data bases. Here are some simple matrix arrays and functions. We start by making a mistake:

```
(A.error <- matrix(1:11, ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9    1
```

The `matrix()` function takes as input here the sequence of numbers from 1 to 11. It then tries to put these 11 elements into a 4 column array with 3 rows. It is missing a number as the error points out. To make a 4 column array out of 11 numbers it needs a twelth number to complete the third row. We then type in these statements

```
(A.row <- matrix(1:12, ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
(A.col <- matrix(1:12, ncol = 4, byrow = FALSE))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

In `A` we take 12 integers in a row and specify they be organized into 4 columns, and in `R` this is by row. In the next statement we see that `A.col` and column binding `cbind()` are equivalent.

```
(R <- rbind(1:4, 5:8, 9:12))  # Concatenate rows
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

```
(C <- cbind(1:3, 4:6, 7:9, 10:12))  # concatenate columns
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

```
A.col == C
```

```
##      [,1] [,2] [,3] [,4]
## [1,] TRUE TRUE TRUE TRUE
## [2,] TRUE TRUE TRUE TRUE
## [3,] TRUE TRUE TRUE TRUE
```

Using the `outer` product allows us to operate on matrix elements, first picking the minimum, then the maximum of each row. The `pmin` and `pmax` compare rows element by element. If you used `min` and `max` you would get the minimum and maximum of the whole matrix.

```
(A.min <- outer(3:6/4, 3:6/4, FUN = pmin))  #
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 0.75 0.75 0.75 0.75
## [2,] 0.75 1.00 1.00 1.00
## [3,] 0.75 1.00 1.25 1.25
## [4,] 0.75 1.00 1.25 1.50
```

```
(A.max <- outer(3:6/4, 3:6/4, FUN = pmax))  #
```

```
##      [,1] [,2] [,3] [,4]
## [1,] 0.75 1.00 1.25  1.5
## [2,] 1.00 1.00 1.25  1.5
## [3,] 1.25 1.25 1.25  1.5
## [4,] 1.50 1.50 1.50  1.5
```

We build a symmetrical matrix and replace the diagonal with 1. `A.sym` looks like a correlation matrix. Here all we were doing is playing with shaping data.

```
(A.sym <- A.max - A.min - 0.5)
```

```
##       [,1]  [,2]  [,3]  [,4]
## [1,] -0.50 -0.25  0.00  0.25
## [2,] -0.25 -0.50 -0.25  0.00
## [3,]  0.00 -0.25 -0.50 -0.25
## [4,]  0.25  0.00 -0.25 -0.50
```

```
diag(A.sym) <- 1
A.sym
```

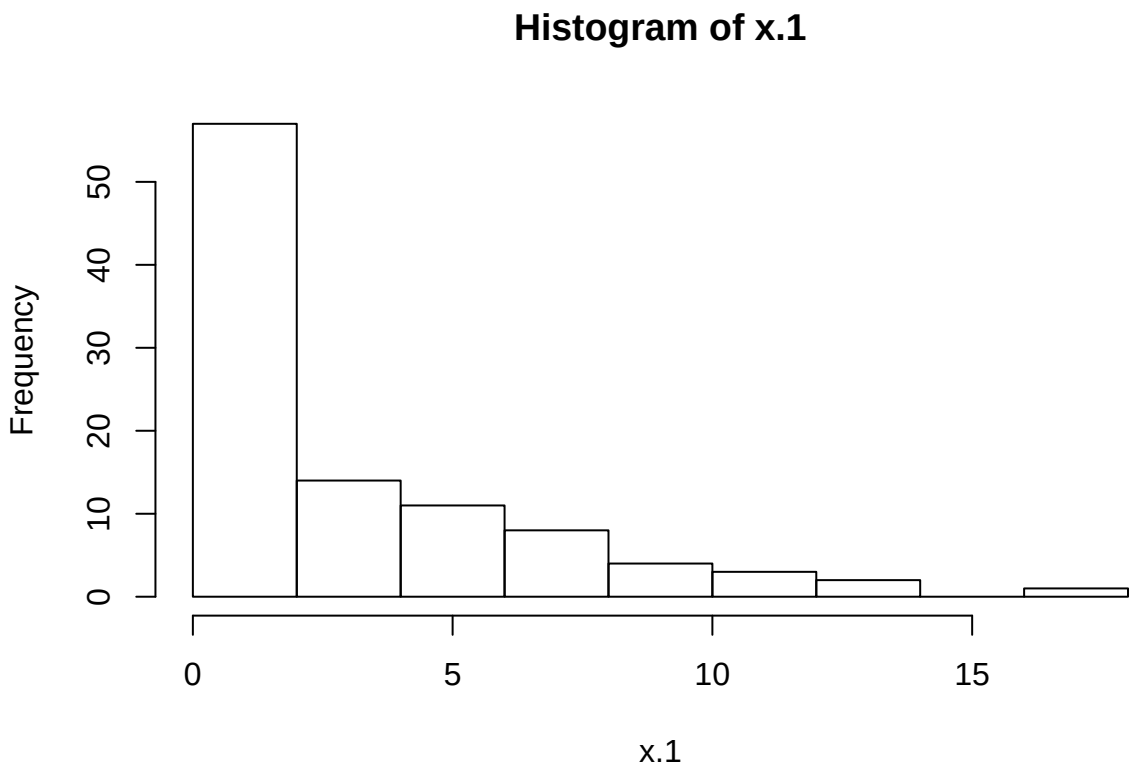
```
##       [,1]  [,2]  [,3]  [,4]
## [1,]  1.00 -0.25  0.00  0.25
## [2,] -0.25  1.00 -0.25  0.00
## [3,]  0.00 -0.25  1.00 -0.25
## [4,]  0.25  0.00 -0.25  1.00
```

2.4.1 Try this exercise

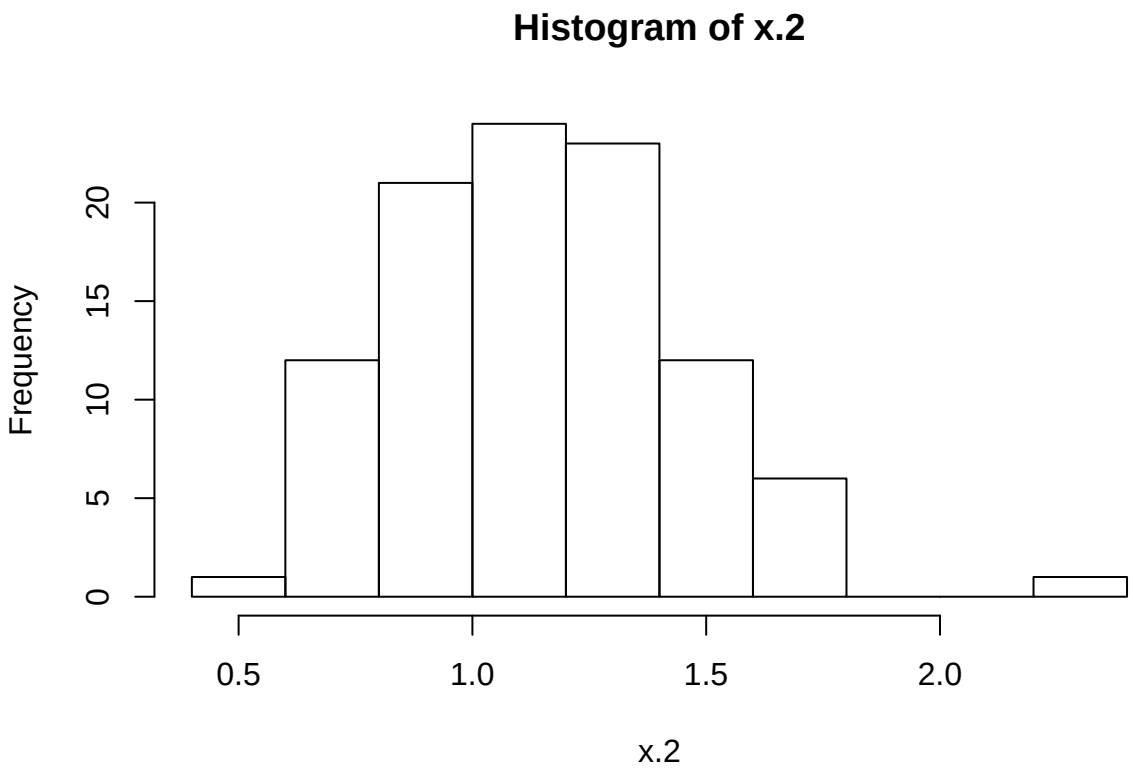
The `inner` product `%*%` cross-multiplies successive elements of a row with the successive elements of a column. If there are two rows with 5 columns, there must be a matrix at least with 1 column that has 5 rows in it.

Let's run these statements.

```
n.sim <- 100
x.1 <- rgamma(n.sim, 0.5, 0.2)
x.2 <- rlnorm(n.sim, 0.15, 0.25)
hist(x.1)
```




```
hist(x.2)
```




```
X <- cbind(x.1, x.2)
```

`rgamma` allows us to generate `n.sim` versions of the gamma distribution with scale parameter `0.5` and shape parameter `0.2`. `rlnorm` is a popular financial return distribution with mean `0.15` and standard deviation `0.25`. We can call up `??distributions` to get detailed information. Let's plot the histograms of each simulated random variate using `hist()`.

The `cbind` function binds into matrix columns the row arrays `x.1` and `x.2`. These might be simulations of operational and financial losses. The `X` matrix could look like the "design" matrix for a regression.

Let's simulate a response vector, say equity, and call it `y` and look at its histogram.

```
y <- 1.5 * x.1 + 0.8 * x.2 + rnorm(n.sim,
    4.2, 5.03)
```

Now we have a model for y:

$$y = X\beta + \varepsilon$$

where y is a 100×1 (rows $\times$ columns) vector, X is a 100×2 matrix, β is a 2×1 vector, and ϵ is a 100×1 vector of disturbances (a.k.a., "errors").

Multiplying out the matrix term $X\beta$ we have

$$y = \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

where y, x_1, x_2, and ε are all vectors with 100 rows for simulated observations.

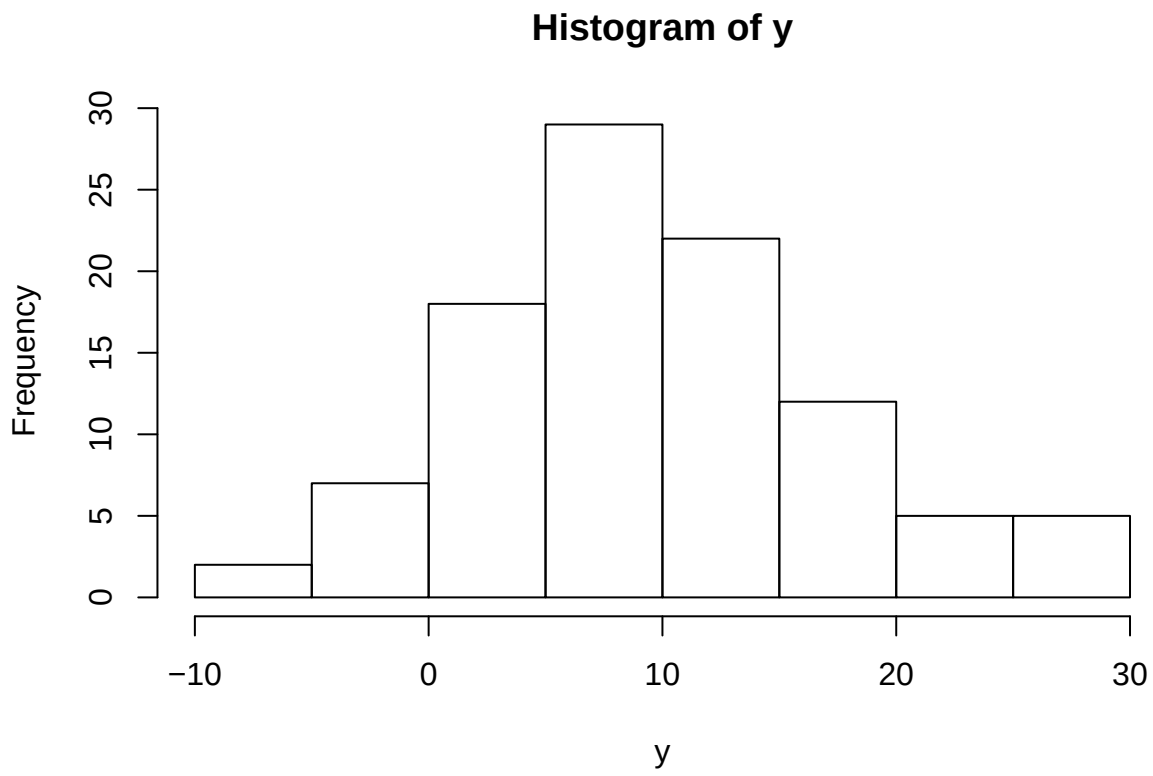
If we look for β to minimize the sum of squared ε we would find that the solution is

$$\hat{\beta} = (X^T X)^{-1} X^T y.$$

Where $\hat{\beta}$ is read as "beta hat".

The result y with its `hist()` is

```
hist(y)
```



The rubber meets the road here as we compute $\hat{\beta}$.

```
X <- cbind(x.1, x.2)
XTX.inverse <- solve(t(X) %*% X)
(beta.hat <- XTX.inverse %*% t(X) %*%
    y)
```

```
##         [,1]
## x.1 1.660246
```

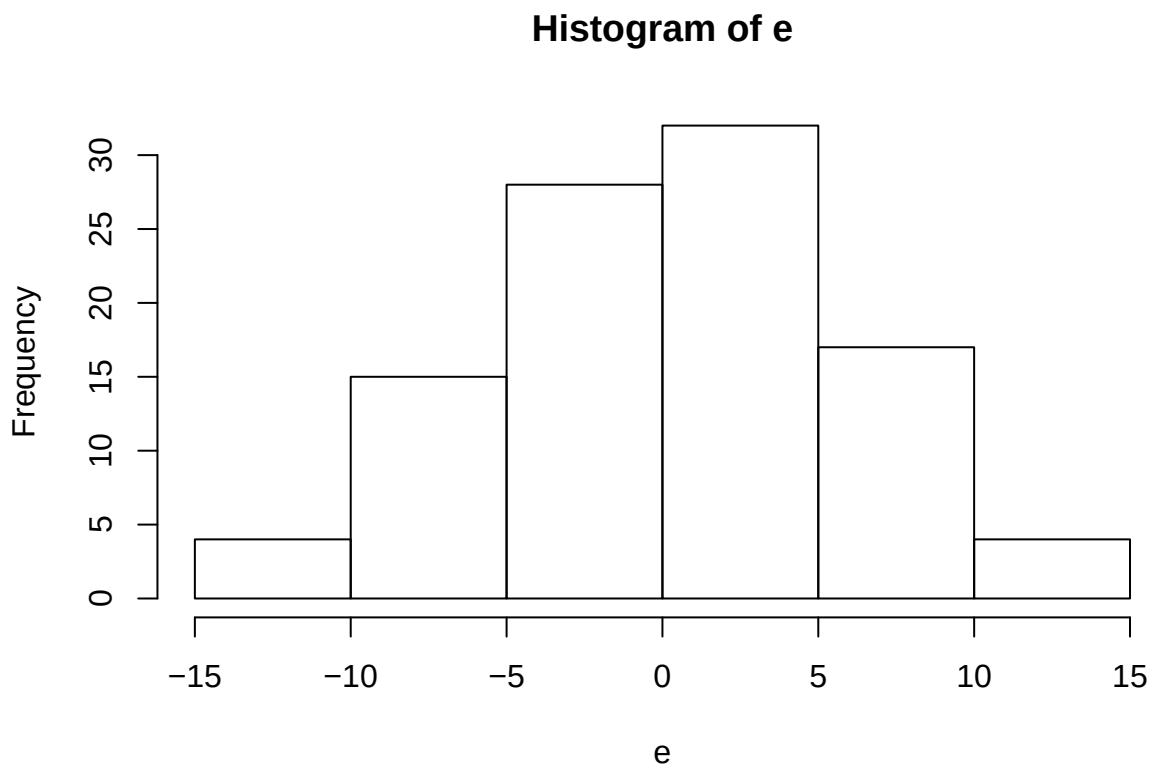
```
## x.2 3.900737
```

The `beta.hat` coefficients are much different than our model for `y`. Why? Because of the innovation, error, disturbance term `rnorm(n.sim, 1, 2)` we added to the `1.5*x.1 + 0.8 * x.2` terms.

Now for the estimated ε where we use the matrix inner product `%*%`. We need to be sure to *pre*-multiply `beta.hat` with `X`!

```
e <- y - X %*% beta.hat
```

```
hist(e)
```



We see that the "residuals" are almost centered at `0`.

2.4.2 More about residuals

For no charge at all let's calculate the sum of squared errors in matrix talk, along with the number of obervations `n` and degrees of freedom `n - k`, all to get the standard error of the regression `e.se`. Mathematically we are computing

$$\sigma_\varepsilon = \sqrt{\sum_{i=1}^{N} \frac{\varepsilon_i^2}{n-k}}$$

```
(e.sse <- t(e) %*% e)
```

```
##          [,1]
## [1,] 3003.548
```

```
(n <- dim(X)[1])
```

```
## [1] 100
```

```
(k <- nrow(beta.hat))
```

```
## [1] 2
```

```
(e.se <- (e.sse/(n - k))^0.5)
```

```
##          [,1]
## [1,] 5.536104
```

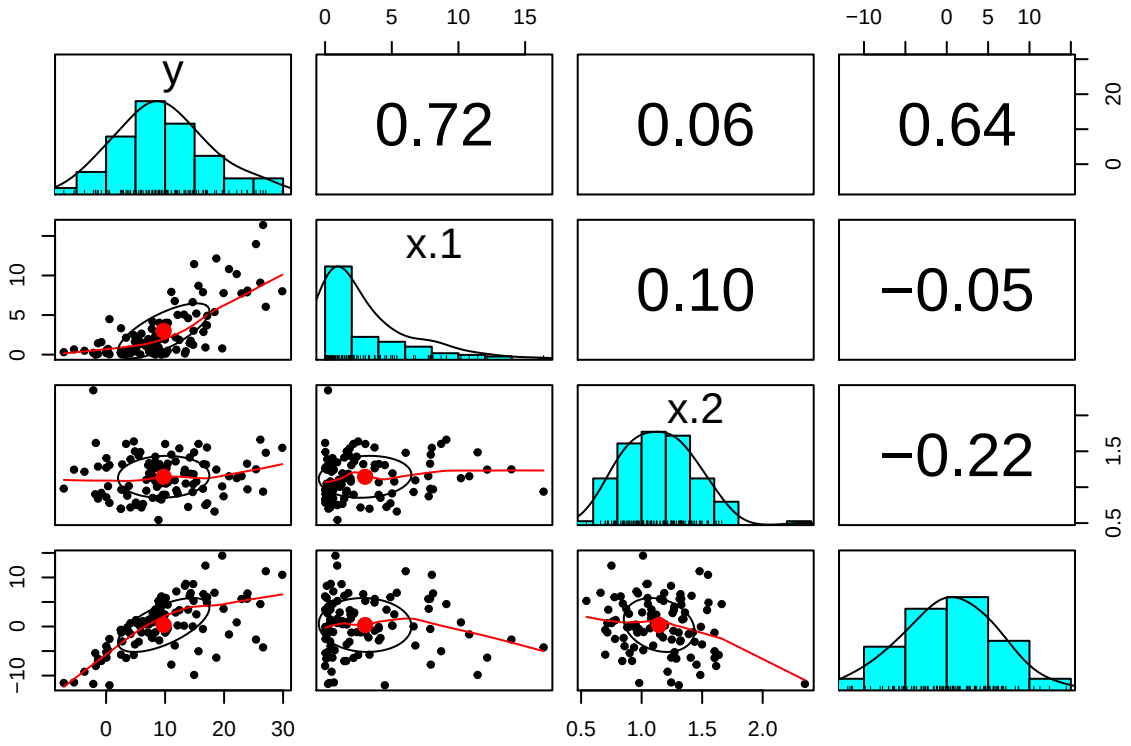
The statement `dim(X)[1]` returns the first of two dimensions of the matrix `X`.

Finally, again for no charge at all, lets load library `psych` (use `install.packages("psych")` as needed). We will use `pairs.panels()` for a pretty picture of our work in this try out. First column bind `cbind()` the `y`, `X`, and `e` arrays to create a data frame for `pairs.panel()`.

```
library(psych)
all <- cbind(y, X, e)
```

We then invoke the `pairs.panels()` function using the `all` array we just created. The result is a scatterplot matrix with histograms of each variate down the diagonal. The lower triangle of the matrix is populated with scatterplots. The upper triangle of the matrix has correlations depicted with increasing font sizes for higher correlations.

```
pairs.panels(all)
```



We will use this tool again and again to explore the multivariate relationships among our data.

2.5 More Array Work

We show off some more array operations in the following statements.

```
nrow(A.min)
```

```
## [1] 4
```

```
ncol(A.min)
```

```
## [1] 4
```

```
dim(A.min)
```

```
## [1] 4 4
```

We calculate the number of rows and columns first. We then see that these exactly correspond to the two element vector produced by `dim`. Next we enter these statements into the console.

```
rowSums(A.min)
```

```
## [1] 3.00 3.75 4.25 4.50
```

```
colSums(A.min)
```

```
## [1] 3.00 3.75 4.25 4.50
```

```
apply(A.min, 1, sum)
```

```
## [1] 3.00 3.75 4.25 4.50
```

```
apply(A.min, 2, sum)
```

```
## [1] 3.00 3.75 4.25 4.50
```

We also calculate the sums of each row and each column. Alternatively we can use the `apply` function on the first dimension (rows) and then on the second dimension (columns) of the matrix. Some matrix multiplications follow below.

```
(A.inner <- A.sym %*% t(A.min[, 1:dim(A.min)[2]]))
```

```
##       [,1]   [,2]   [,3]  [,4]
## [1,] 0.750 0.7500 0.8125 0.875
## [2,] 0.375 0.5625 0.5000 0.500
## [3,] 0.375 0.5000 0.6875 0.625
## [4,] 0.750 0.9375 1.1250 1.375
```

Starting from the inner circle of embedded parentheses we pull every row (the `[,col]` piece) for columns from the first to the second dimension of the `dim()` of `A.min`. We then transpose (row for column) the elements of `A.min` and cross left multiply in an inner product this transposed matrix with `A.sym`.

We have already deployed very useful matrix operation, the inverse. The `R` function `solve()` provides the answer to the question: what two matrices, when multiplied by one another, produces the identity matrix? The identity matrix is a matrix of all ones down the diagonal and zeros elsewhere.

```
(A.inner.invert <- solve(A.inner))
```

```
##               [,1]      [,2]      [,3]      [,4]
## [1,]  4.952381e+00 -3.047619 -1.142857 -1.523810
## [2,] -2.285714e+00  6.857143 -2.285714  0.000000
## [3,]  1.268826e-16 -2.285714  6.857143 -2.285714
## [4,] -1.142857e+00 -1.142857 -3.428571  3.428571
```

Now we use our inverse with the original matrix we inverted.

```
(A.result <- A.inner %*% A.inner.invert)
```

```
##              [,1]          [,2]          [,3]         [,4]
## [1,] 1.000000e+00  0.000000e+00  0.000000e+00 0.000000e+00
## [2,] 2.220446e-16  1.000000e+00  0.000000e+00 0.000000e+00
```

```
## [3,] 1.110223e-16  1.110223e-16  1.000000e+00 4.440892e-16
## [4,] 2.220446e-16 -4.440892e-16 -8.881784e-16 1.000000e+00
```

When we cross multiply `A.inner` with its inverse, we should, and do, get the identity matrix that is a matrix of ones in the diagonal and zeros in the off-diagonal elements.

2.6 Summary

We covered very general data manipulation in `R` including arithmetical operations, vectors and matrices, their formation and operations, and data frames. We used data frames as inputs to plotting functions. We also built a matrix-based linear regression model and a present value calculator.

2.7 Further Reading

This introductory chapter covers material from Teetor, chapters 1, 2, 5, 6. Present value, salvage, and other valuation topics can be found in Brealey et al. under `present value` in the index of any of several editions.

2.8 Practice Set

2.8.1 Purpose, Process, Product

These practice sets will repeat various `R` features in this chapter. Specifically we will practice defining vectors, matrices (arrays), and data frames and their use in present value, growth, future value calculations, We will build on this basic practice with the computation of ordinary lease squares coefficients and plots using `ggplot2`. We will summarize our findings in debrief documented with an `R markdown` file and output.

2.8.2 `R Markdown` set up

1. Open a new `R Markdown` pdf document file and save it with file name `MYName-FIN654-PS01` to your working directory. The `Rmd` file extension will automatically be appended to the file name. Create a new folder called `data` in this working directory and deposit the `.csv` file for practice set #2 to this directory.
2. Modify the `YAML` header in the `Rmd` file to reflect the name of this practice set, your name, and date.
3. Replace the `R Markdown` example in the new file with the following script.

```
## Practice set 1: present value
(INSERT results here)
## Practice set 2: regression
(Insert results here)
```

4. Click `knit` in the `Rstudio` command bar to produce the `pdf` document.

2.8.3 Set A

2.8.3.1 Problem

We work for a mutual fund that is legally required to fair value the stock of unlisted companies it owns. Your fund is about to purchase shares of InUrCorner, a U.S. based company, that provides internet-of-things legal services.

- We sampled several companies with business plans similar to InUrCorner and find that the average weighted average cost of capital is 18%.
- InUrCorner sales is $80 million and projected to growth at 50% per year for the next 3 years and 15% per year thereafter.
- Cost of services provided as a percent of sales is currently 75% and projected to be flat for the foreseeable future.
- Depreciation is also constant at 5% of net fixed assets (gross fixed asset minus accumulated depreciation), as are taxes (all-in) at 35% of taxable profits.
- Discussions with InUrCorner management indicate that the company will need an increase in working capital at the rate of 15% each year and an increase in fixed assets at the rate of 10% of sales each year. Currently working capital is $10, net fixed assets is $90, and accumulated depreciation is $15.

2.8.3.2 Questions

1. Let's project `sales`, `cost`, increments to net fixed assets `NFA`, increments to working capital `WC`, `depreciation`, `tax`, and free cash flow `FCF` for the next 4 years. We will use a table to report the projection.

Let's use this code to build and display a table.

```
# Form table of results
table.names <- c("Sales", "Cost", "Working Capital (incr.)",
    "Net Fixed Assets (incr.)", "Free                                Cash Flow")
# Assign projection labels
table.year <- year  # Assign projection years
table.data <- rbind(sales, cost, WC.incr,
    NFA.incr, FCF)  # Layer projections
```

```
rownames(table.data) <- table.names  # Replace rows with projection labels
colnames(table.data) <- table.year  # Replace columns with projection years
knitr::kable(table.data)  # Display a readable table
```

2. Modify the assumptions by +/- 10% and report the results.

2.8.4 Set B

2.8.4.1 Problem

We work for a healthcare insurer and our management is interested in understanding the relationship between input admission and outpatient rates as drivers of expenses, payroll, and employment. We gathered a sample of 200 hospitals in a test market in this data set.

```
x.data <- read.csv("data/hospitals.csv")
```

2.8.4.2 Questions

1. Build a table that explores this data set variable by variable and relationships among variables.
2. Investigate the influence of admission and outpatient rates on expenses and payroll. First, form these arrays.

Next, compute the regression coefficients.

Finally, compute the regression statistics.

3. Use this code to investigate further the relationship among predicted expenses and the drivers, admissions and outpatients.

```
require(reshape2)
require(ggplot2)
actual <- y
predicted <- X %*% beta.hat
residual <- actual - predicted
results <- data.frame(actual = actual,
    predicted = predicted, residual = residual)
# Insert comment here
min_xy <- min(min(results$actual), min(results$predicted))
max_xy <- max(max(results$actual), max(results$predicted))
# Insert comment here
plot.melt <- melt(results, id.vars = "predicted")
# Insert comment here
plot.data <- rbind(plot.melt, data.frame(predicted = c(min_xy,
```

```
    max_xy), variable = c("actual", "actual"),
    value = c(max_xy, min_xy)))
# Insert comment here
p <- ggplot(plot, aes(x = predicted,
    y = value)) + geom_point(size = 2.5) +
    theme_bw()
p <- p + facet_wrap(~variable, scales = "free")
p
```

2.8.5 Practice Set Debrief

1. List the R skills needed to complete these practice labs.
2. What are the packages used to compute and graph results. Explain each of them.
3. How well did the results begin to answer the business questions posed at the beginning of each practice lab?

2.9 Project

2.9.1 Purpose

This project will allow us to practice various R features using live data to support a decision regarding the provision of captive financing to customers at the beginning of this chapter. We will focus on translating regression statistics into R, plotting results, and interpreting ordinary least squares regression outcomes.

2.9.2 Problem

As we researched how to provide captive financing and insurance for our customers we found that we needed to understand the relationships among lending rates and various terms and conditions of typical equipment financing contracts.

We will focus on one question:

> *What is the influence of terms and conditions on the lending rate of fully committed commercial loans with maturities greater than one year?*

2.9.3 Data

The data set `commloan.csv` contains data from the St. Louis Federal Reserve Bank's FRED website we will use to get some high level insights. The quarterly data extends from the first quarter of 2003 to the second quarter of 2016 and aggregates a survey administered by the St. Louis Fed. There are several time series included. Each is by the time that pricing terms Were set and by commitment, with maturities more than 365 Days from a survey of all commercial banks. Here are the definitions.

Variable	Description	Units of Measure
rate	Weighted-Average Effective Loan Rate	percent
prepay	Percent of Value of Loans Subject to Prepayment Penalty	percent
maturity	Weighted-Average Maturity/Repricing Interval in Days	days
size	Average Loan Size	thousands USD
volume	Total Value of Loans	millions USD

2.9.4 Work Flow

1. Prepare the data.

- Visit the FRED website. Include any information on the site to enhance the interpretation of results.
- Use `read.csv` to read the data into `R`. Be sure to set the project's working directory where the data directory resides. Use `na.omit()` to clean the data.

```
# setwd('C:/Users/Bill
# Foote/bookdown/bookdown-demo-master')
# the project directory
x.data <- read.csv("data/commloans.csv")
x.data <- na.omit(x.data)
```

- Assign the data to a variable called `x.data`. Examine the first and last 5 entries. Run a `summary` of the data set.
- What anomalies appear based on these procedures?

2. Explore the data.

- Let's plot the time series data using this code:

```
require(ggplot2)
require(reshape2)
# Use melt() from reshape2 to build
```

```
# data frame with data as id and
# values of variables
x.melted <- melt(x.data[, c(1:4)], id = "date")
ggplot(data = x.melted, aes(x = date,
    y = value)) + geom_point() + facet_wrap(~variable,
    scales = "free_x")
```

- Describe the data frame that `melt()` produces.
- Let's load the `psych` library and produce a scatterplot matrix. Interpret this exploration.

3. Analyze the data.

- Let's regress `rate` on the rest of the variables in `x.data`. To do this we form a matrix of independent variables (predictor or explanatory variables) in the matrix `X` and a separate vector `y` for the dependent (response) variable `rate`. We recall that the `1` vector will produce a constant intercept in the regression model.

```
y <- as.vector(x.data[, "rate"])
X <- as.matrix(cbind(1, x.data[, c("prepaypenalty",
    "maturity", "size", "volume")]))
head(y)
head(X)
```

- Explain the code used to form `y` and `X`.
- Calculate the $\hat{\beta}$ coefficients and interpret their meaning.
- Calculate actual and predicted `rates` and plot using this code.

```
# Insert comment here
require(reshape2)
require(ggplot2)
actual <- y
predicted <- X %*% beta.hat
residual <- actual - predicted
results <- data.frame(actual = actual,
    predicted = predicted, residual = residual)
# Insert comment here
min_xy <- min(min(results$actual), min(results$predicted))
max_xy <- max(max(results$actual), max(results$predicted))
# Insert comment here
plot.melt <- melt(results, id.vars = "predicted")
# Insert comment here
plot.data <- rbind(plot.melt, data.frame(predicted = c(min_xy,
    max_xy), variable = c("actual", "actual"),
    value = c(max_xy, min_xy)))
```

```
# Insert comment here
p <- ggplot(plot, aes(x = predicted,
    y = value)) + geom_point(size = 2.5) +
    theme_bw()
p <- p + facet_wrap(~variable, scales = "free")
p
```

- Insert explanatory comments into the code chunk to document the work flow for this plot.
- Interpret the graphs of actual and residual versus predicted values of `rate`.
- Calculate the standard error of the residuals, Interpret its meaning.

4. Interpret and present results.

- We will produce an `R Markdown` document with code chunks to document and interpret our results.
- The format will introduce the problem to be analyzed, with sections that discuss the data to be used, and which follow the work flow we have defined.

2.9.5 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.
- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.
- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.

- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

2.10 References

Brealey, Richard, stewart Myers, and Franklyn Allen. 2016. Principles of Corporate Finance, revised edition. New York: McGraw-Hill.

Teetor, Paul. 2011. R Cookbook. Sebastopol, CA: O'Reilly.

Chapter 3

R Data Modeling

3.1 Imagine This

Your project team is assigned to work with the accounts receivable team. Specifically, accounts receivable is about to review a portfolio of customers from a potential acquisition. Managers would like to query data provided through due diligence during the acquisition process. Some of the questions include:

1. What is the income risk across applicant pools?
2. Are there differences in applicant income?
3. Does age matter?
4. Is there a pattern of family dependents across applicant pools?
5. How much income per dependent?

The analytics team will process the data, review its quality, and help accounts receivable answer these questions.

In this chapter we will build approaches to manage such queries, including pivot tables, lookups, and the creation of new metrics from existing data. We will expand this assortment of skills into the writing of functions, such as net present value and internal rate of return, more plotting of data, working with time series data, and fitting data to probability distributions.

3.2 Pivot tables and Vertical Lookups

These are, mythically at least, two of the most-used Excel features. Pivot tables are the slice and dice machine we use to partition data sets. Lookups allow us to relate one data table to another. We will explore these tools in `R`, here made easier and less apt to crash on large data sets. We start with some definitions.

3.2.1 Some definitions

The pivot table is a data summarization tool that can automatically sort, count, total, or give the average of the data stored in one table or spreadsheet, displaying the results in a second table showing the summarized data. This tool transforms a flat table of fields with rows of data into a table with grouped row values and column header values. The specification of grouped row values and column headers can rotate the flat table's data rows into the intersection of the row and column labels.

"V" or "vertical" stands for the looking up of a value in a column. This feature allows the analyst to find approximate and exact matches between the look up value and a table value in a vertical column assigned to the look up value. A `HLOOKUP` function does the same lookup but for a specified row instead of a column.

3.2.2 Pivot and Parry

Let's return to the Credit Card Applicant business questions:

1. What is the income risk across applicant pools?
2. Are there differences in applicant income?
3. Does age matter?
4. Is there a pattern of dependents across applicant pools?
5. How much income per dependent?

The first step in building an analysis of the data relative to these questions is to understand the required dimensions of the data that apply to the questions. Here we would scan the table column names in the data base and look for

1. Card status
2. Ownership
3. Employment

```
CreditCard <- read.csv("data/CreditCard.csv")
str(CreditCard)
```

```
## 'data.frame':    1319 obs. of  13 variables:
##  $ card       : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ reports    : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ age        : num  37.7 33.2 33.7 30.5 32.2 ...
##  $ income     : num  4.52 2.42 4.5 2.54 9.79 ...
##  $ share      : num  0.03327 0.00522 0.00416 0.06521 0.06705 ...
##  $ expenditure: num  124.98 9.85 15 137.87 546.5 ...
##  $ owner      : Factor w/ 2 levels "no","yes": 2 1 2 1 2 1 1 2 2 1 ...
##  $ selfemp    : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ dependents : int  3 3 4 0 2 0 2 0 0 0 ...
##  $ months     : int  54 34 58 25 64 54 7 77 97 65 ...
```

```
##  $ majorcards : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ active     : int  12 13 5 7 5 1 5 3 6 18 ...
##  $ state      : Factor w/ 3 levels "CT","NJ","NY": 3 3 3 3 3 3 3 3 3 3 ...
```

The `str()` function allows us to see all of the objects in `CreditCard`. Next lext look at the data itself inside this object using `head` (for the beginning of the data).

```
head(CreditCard, 3)
```

```
##   card reports      age income       share expenditure owner selfemp
## 1  yes       0 37.66667   4.52 0.033269910  124.983300   yes      no
## 2  yes       0 33.25000   2.42 0.005216942    9.854167    no      no
## 3  yes       0 33.66667   4.50 0.004155556   15.000000   yes      no
##   dependents months majorcards active state
## 1          3     54          1     12    NY
## 2          3     34          1     13    NY
## 3          4     58          1      5    NY
```

Knowing the structure and a sample of the data, we can build a summary of the data and review the minimum, maximum, and quartiles in each of `CreditCard`'s columns of data.

```
summary(CreditCard)
```

```
##   card         reports            age              income
##  no : 296   Min.   : 0.0000   Min.   : 0.1667   Min.   : 0.210
##  yes:1023   1st Qu.: 0.0000   1st Qu.:25.4167   1st Qu.: 2.244
##             Median : 0.0000   Median :31.2500   Median : 2.900
##             Mean   : 0.4564   Mean   :33.2131   Mean   : 3.365
##             3rd Qu.: 0.0000   3rd Qu.:39.4167   3rd Qu.: 4.000
##             Max.   :14.0000   Max.   :83.5000   Max.   :13.500
##      share            expenditure       owner     selfemp
##  Min.   :0.0001091   Min.   :   0.000   no :738   no :1228
##  1st Qu.:0.0023159   1st Qu.:   4.583   yes:581   yes:  91
##  Median :0.0388272   Median : 101.298
##  Mean   :0.0687322   Mean   : 185.057
##  3rd Qu.:0.0936168   3rd Qu.: 249.036
##  Max.   :0.9063205   Max.   :3099.505
##    dependents         months         majorcards         active
##  Min.   :0.0000   Min.   :  0.00   Min.   :0.0000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 12.00   1st Qu.:1.0000   1st Qu.: 2.000
##  Median :1.0000   Median : 30.00   Median :1.0000   Median : 6.000
##  Mean   :0.9939   Mean   : 55.27   Mean   :0.8173   Mean   : 6.997
##  3rd Qu.:2.0000   3rd Qu.: 72.00   3rd Qu.:1.0000   3rd Qu.:11.000
##  Max.   :6.0000   Max.   :540.00   Max.   :1.0000   Max.   :46.000
##  state
##  CT:442
##  NJ:472
```

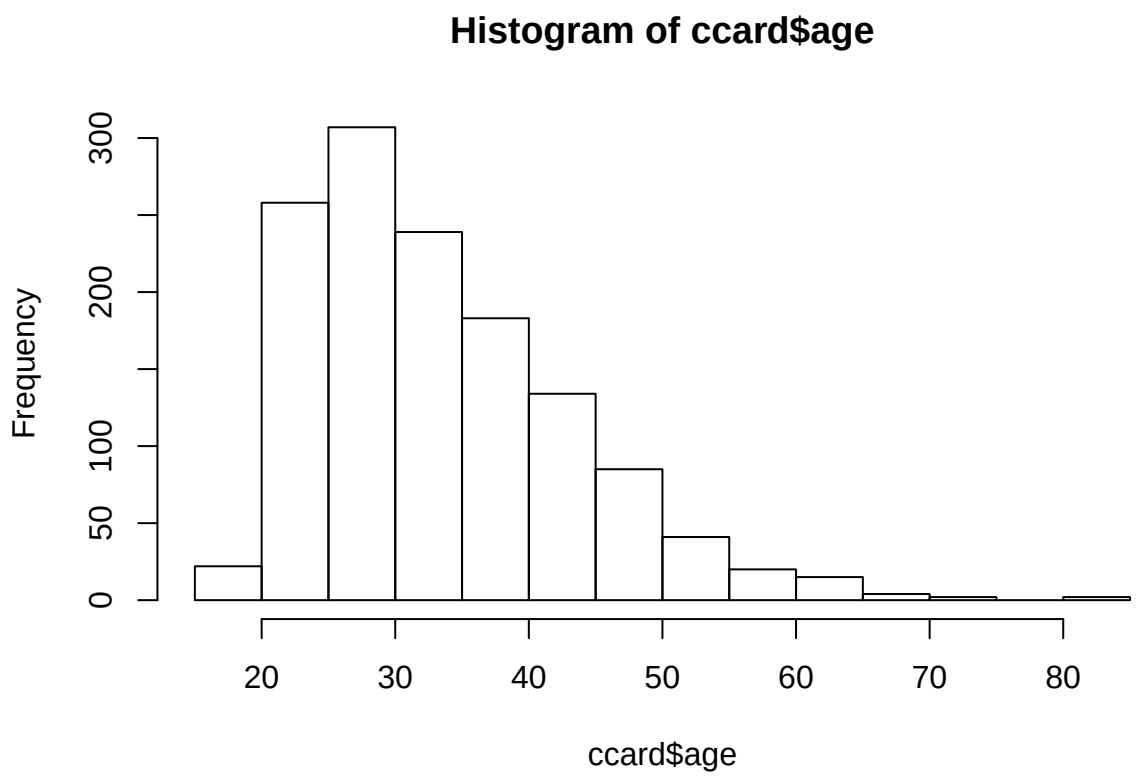
```
##  NY:405
##
##
##
```

We immediately see an age minimum of 0.2. Either this is an anomaly, or outright error, or there is an application not quite a year old!. Let's filter the data for ages greater than 18 to be safe.

```
ccard <- CreditCard[CreditCard$age >=
    18, ]
```

In the filter, the comma means keep data on applicants only at or in excess of 18 years of age. When we leave the column empty, it means apply this filter across all columns. We next review the distribution of ages of applicants to be sure our filter does the job properly. The function `hist()` builds a simple frequency histogram to visualize this data.

```
hist(ccard$age)
```



3.2.3 Try this exercise

What is the basic design of this inquiry?

1. Business questions?
2. Dimensions?
3. Taxonomy and metrics?

To answer 1 and 2 we have business questions along the lines of indicator variables:

- Card issued (`card`)
- Own or rent (`owner`)
- Self-employed or not (`selfemp`)

For 3 our basic taxonomy is:

1. For each card issued…in New York
2. …and for each owner…
3. …who is employed…
4. What are the range of income, average dependents, age, and income per dependent?

Here is the basic 3 step pivot table design. We should check if we have installed the `dplyr` package into the `R` environment.

```
# install.packages('dplyr') if not
# already
require(dplyr)
## 1: filter to keep three states.
pivot.table <- filter(ccard, state %in%
    "NY")
## 2: set up data frame for by-group
## processing.
pivot.table <- group_by(pivot.table,
    card, owner, selfemp)
## 3: calculate the three summary
## metrics
options(dplyr.width = Inf)  ## to display all columns
pivot.table <- summarise(pivot.table,
    income.cv = sd(income)/mean(income),
    age.avg = mean(age), income.per.dependent = sum(income)/sum(dependents))
```

We then visualize results in a table. Here we use `knitr`, which is a package that powers `rmarkdown`. The function `kable()` is short for "knitr table."

```
knitr::kable(pivot.table)
```

card	owner	selfemp	income.cv	age.avg	income.per.dependent
no	no	no	0.4941859	31.91936	3.645848
no	no	yes	0.5652634	26.38542	2.852000
no	yes	no	0.3756274	36.01786	2.157589
no	yes	yes	NaN	53.33333	Inf
yes	no	no	0.3298633	28.09311	5.313677
yes	no	yes	0.4367858	37.45238	7.062500
yes	yes	no	0.5519888	36.79503	3.154476
yes	yes	yes	0.5032180	41.91667	3.194547

3.2.4 Now to VLOOKUP

Let's start with a different data set. We load this IBRD (World Bank) data that has

- The variable `life.expectancy` is the average life expectancy for each country from 2009 through 2014.
- The variable `sanitation` is the percentage of population with direct access to sanitation facilities.

```
le <- read.csv("data/life_expectancy.csv",
    header = TRUE, stringsAsFactors = FALSE)
sa <- read.csv("data/sanitation_.csv",
    header = TRUE, stringsAsFactors = FALSE)
```

Always we look at the first few records.

```
head(le)
```

```
##                country years.life.expectancy.avg
## 1          Afghanistan                   46.62135
## 2              Albania                   71.11807
## 3              Algeria                   61.81652
## 4               Angola                   41.65847
## 5 Antigua and Barbuda                   69.81219
## 6           Arab World                   60.93432
```

```
head(sa)
```

```
##           country sanitation.avg
## 1     Afghanistan       25.39600
## 2         Albania       85.36154
## 3         Algeria       84.21538
## 4 American Samoa       61.73077
## 5         Andorra      100.00000
## 6          Angola       36.00769
```

The job here is to join sanitation data with life expectancy data, by country. In Excel we

would typically use a `VLOOKUP(country, sanitation, 2, FALSE)` statement.

1. In this statement `country` is the value to be looked up, for example, "Australia".
2. The variable `sanitation` is the range of the sanitation lookup table of two columns of country and sanitation data, for example, `B2:C104` in Excel.
3. The `2` is the second column of the sanitation lookup table, for example column `C`.
4. `FALSE` means don't find an exact match.

In `R` we can use the `merge()` function.

```
life.sanitation <- merge(le[, c("country",
    "years.life.expectancy.avg")], sa[,
    c("country", "sanitation.avg")])
```

The whole range of countries is populated by the lookup.

```
head(life.sanitation, 3)
```

```
##       country years.life.expectancy.avg sanitation.avg
## 1 Afghanistan                   46.62135       25.39600
## 2     Albania                   71.11807       85.36154
## 3     Algeria                   61.81652       84.21538
```

3.2.5 Try this exercise

We will load yet another data set on house prices. Suppose we work for a housing developer like Toll Brothers (NYSE: TOL) and want to allocate resources to marketing and financing the building of luxury homes in major US metropolitan areas. We have data for one test market.

```
hprice <- read.csv("data/hprice.csv")
```

Let's look at the available data:

```
summary(hprice)
```

```
##        ID              Price             SqFt         Bedrooms
##  Min.   :  1.00   Min.   : 69100   Min.   :1450   Min.   :2.000
##  1st Qu.: 32.75   1st Qu.:111325   1st Qu.:1880   1st Qu.:3.000
##  Median : 64.50   Median :125950   Median :2000   Median :3.000
##  Mean   : 64.50   Mean   :130427   Mean   :2001   Mean   :3.023
##  3rd Qu.: 96.25   3rd Qu.:148250   3rd Qu.:2140   3rd Qu.:3.000
##  Max.   :128.00   Max.   :211200   Max.   :2590   Max.   :5.000
##    Bathrooms         Offers      Brick    Neighborhood
##  Min.   :2.000   Min.   :1.000   No :86   East :45
##  1st Qu.:2.000   1st Qu.:2.000   Yes:42   North:44
##  Median :2.000   Median :3.000            West :39
##  Mean   :2.445   Mean   :2.578
```

```
##  3rd Qu.:3.000   3rd Qu.:3.000
##  Max.   :4.000   Max.   :6.000
```

Our business questions include:

1. What are the most valuable (higher price) neighborhoods?
2. What housing characteristics maintain the most housing value?

First, where and what are the most valuable houses? One way to answer this is to build a pivot table. Next we pivot the data and build metrics into the query. We will use the `mean()` and standard deviation `sd()` functions to help answer our questions.

```
require(dplyr)
## 1: filter to those houses with
## fairly high prices
pivot.table <- filter(hprice, Price >
    99999)
## 2: set up data frame for by-group
## processing
pivot.table <- group_by(pivot.table,
    Brick, Neighborhood)
## 3: calculate the summary metrics
options(dplyr.width = Inf)  ## to display all columns
pivot.table <- summarise(pivot.table,
    Price.avg = mean(Price), Price.cv = sd(Price)/mean(Price),
    SqFt.avg = mean(SqFt), Price.per.SqFt = mean(Price)/mean(SqFt))
```

Then we visualize in a table.

```
knitr::kable(pivot.table)
```

Brick	Neighborhood	Price.avg	Price.cv	SqFt.avg	Price.per.SqFt
No	East	121095.7	0.1251510	2019.565	59.96125
No	North	115307.1	0.0939797	1958.214	58.88382
No	West	148230.4	0.0912350	2073.478	71.48878
Yes	East	135468.4	0.0977973	2031.053	66.69863
Yes	North	118457.1	0.1308498	1857.143	63.78462
Yes	West	175200.0	0.0930105	2091.250	83.77764

Based on this data set from one metropolitan area, the most valuable properties (fetching the highest average price and price per square foot) are made of brick in the West neighborhood. Brick or not, the West neighborhood also seems have the lowest relative variation in price.

Now for something different: functions.

3.3 Why Functions?

We will encapsulate several operations into a reusable storage device called a function. The usual suspects and candidates for the use of functions are:

- Data structures rack together related values into one object.
- Functions group related commands into one object.

In both cases the logic and coding is easier to understand, easier to work with, easier to build into larger things, and less prone to breaches of plain-old stubby finger breaches of operational safety and security.

For example, here is an Excel look-alike NPV function. We enter this into a code-chunk in an `R markdown` file or directly into the console to store the function into the current `R` environment. Once that is done, we now have a new function we can use like any other function.

```
## Net Present Value function Inputs:
## vector of rates (rates) with 0 as
## the first rate for time 0, vector
## of cash flows (cashflows) Outputs:
## scalar net present value
NPV.1 <- function(rates, cashflows) {
    NPV <- sum(cashflows/(1 + rates)^(seq_along(cashflows) -
        1))
    return(NPV)
}
```

The structure of a function has these parts:

1. A header describes the function along with inputs and outputs. Here we use comment characters `#` to describe and document the function.
2. A definition names the function and identify the interface of inputs and outputs to the programming environment. The name is like a variable and is assigned to `function()`, where inputs are defined.
3. Code statements take the inputs from the definition and program the tasks, logic, and decisions in the function's work flow into output.
4. An output statement releases the function's results for use in other code statements outside of the function's "mini-verse." We use the formal `return()` function to identify the output that the function will produce. If we did not use `return()`, then `R` will us the last assigned variable as the output of the function.

In this example We generate data internal to the function:

- We use `seq_along` to generate time index of cashflows.
- We must subtract 1 from this sequence as starting cashflow is time 0.
- We generate a net present value directly in one line of code.

Our functions get used just like the built-in ones, for example, `mean()`. Let's define `rates` and `cashflows` as vector inputs to the `NPV.1()` function and run this code.

```
rates <- c(0, 0.08, 0.06, 0.04)  ## first rate is always 0.00
cashflows <- c(-100, 200, 300, 10)
NPV.1(rates, cashflows)
```

```
## [1] 361.0741
```

We go back to the declaration and look at the parts:

```
## Net Present Value function Inputs:
## vector of rates (rates) with 0 as
## the first rate for time 0, vector
## of cash flows (cashflows) Outputs:
## scalar net present value
NPV.1 <- function(rates, cashflows) {
    NPV <- sum(cashflows/(1 + rates)^(seq_along(cashflows) -
        1))
    return(NPV)
}
```

Interfaces refer to these components:

- **inputs** or **arguments**
- **outputs** or **return value**
- Calls other functions `sum`, `seq_along()`, operators `/`, `+`, `^` and `-` .

We can also call other functions we've written. We use `return()` to explicitly say what the output is. This is simply good documentation. Alternately, a function will return the last evaluation.

Comments, that is, lines that begin with `#`, are not required by `R`, but are always a good and welcome idea that provide a terse description of purpose and direction. Initial comments should also include a listing of inputs, also called "arguments," and outputs.

3.3.1 What should be a function?

Functions should be written for code we are going to re-run, especially if it will be re-run with changes in inputs. They can also be code chunks we keep highlighting and hitting return on. We often write functions for code chunkswhich are small parts of bigger analyses.

In the next redition of `irr.1` we improve the code with named and default arguments.

```
## Internal Rate of Return (IRR)
## function Inputs: vector of cash
## flows (cashflows), scalar
## interations (maxiter) Outputs:
```

```
## scalar net present value
IRR.1 <- function(cashflows, maxiter = 1000) {
    t <- seq_along(cashflows) - 1
    ## rate will eventually converge to
    ## IRR
    f <- function(rate) (sum(cashflows/(1 +
        rate)^t))
    ## use uniroot function to solve for
    ## root (IRR = rate) of f = 0 c(-1,1)
    ## bounds solution for only positive
    ## or negative rates select the root
    ## estimate
    return(uniroot(f, c(-1, 1), maxiter = maxiter)$root)
}
```

Here the *default argument* is `maxiter` which controls the number of iterations. At our peril we can eliminate this argument if we want. This illustrates yet another need for functions: we can put error and exception logic to handle somtimes fatal issues our calculations might present.

Here are the cashflows for a 3% coupon bond bought at a hefty premium.

```
cashflows <- c(-150, 3, 3, 3, 3, 3, 3,
    3, 103)
IRR.1(cashflows)
```

```
## [1] -0.02554088
```

```
IRR.1(cashflows, maxiter = 100)
```

```
## [1] -0.02554088
```

We get a negative IRR or yield to maturity on this net present value = 0 calculation.

3.3.2 Shooting trouble

Problem: We see "odd"" behavior when arguments aren't as we expect.

```
NPV.1(c(0.1, 0.05), c(-10, 5, 6, 100))
```

```
## [1] 86.10434
```

We do get a result, but…

- What does it mean?
- What rates correspond with what cashflows?

Here the function calculates a net present value. But the analyst entered two rates for four cash flows.

Solution: We put sanity checks into the code.

- Let's use the `stopifnot(some logical statment)` is TRUE.

```
## Net Present Value function Inputs:
## vector of rates (rates) with 0 as
## the first rate for time 0, vector
## of cash flows (cashflows), length
## of rates must equal length of
## cashflows Outputs: scalar net
## present value
NPV.2 <- function(rates, cashflows) {
    stopifnot(length(rates) == length(cashflows))
    NPV <- sum(cashflows/(1 + rates)^(seq_along(cashflows) -
        1))
    return(NPV)
}
```

Here are some thoughts about `stopifnot TRUE` error handling

- Arguments to `stopifnot()` are a series of logical expressions which should all be TRUE.
- Execution halts, with error message, at *first* FALSE.

```
NPV.2(c(0.1, 0.05), c(-10, 5, 6, 100))
```

Hit (not too hard!) the `Escape` key on your keyboard, This will take you out of `Browse[1]>` mode and back to the console prompt `>`.

3.3.3 What the function can see and do

Each function has its own environment. Names here will override names in the global environment. The function's internal environment starts with the named arguments. Assignments inside the function only change the internal environment. If a name is not defined in the function, the function will look for this name in the environment the function gets called from.

3.3.4 Try this ...

Your company is running a 100 million pound sterling project in the EU. You must post 25% collateral in a Landesbank using only high-quality government securities. You find a high-quality gilt fund that will pay 1.5% (coupon rate) annually for three years.

Some questions for analysis

1. How much would you pay for this collateral if the rate curve (yield to maturity of cash flows) is (from next year on...)

```
rates <- c(-0.001, 0.002, 0.01)
```

2. Suppose a bond dealer asks for 130% of notional collateral value for this bond. What is the yield on this transaction (IRR)? Would you buy it?
3. What is the return on this collateral if you terminate the project in one year and liquidate the collateral (i.e., sell it for cash) if the yield shifts down by 0.005? This is a "parallel" shift, which is finance for: "take each rate and deduct 0.005."

To get at these requirements we will build rates and cash flows across the 3-year time frame, remembering our previous work.

```
(rates <- c(0, rates))
```

```
## [1]  0.000 -0.001  0.002  0.010
```

```
collateral.periods <- 3
collateral.rate <- 0.25
collateral.notional <- collateral.rate *
    100
coupon.rate <- 0.015
cashflows <- rep(coupon.rate * collateral.notional,
    collateral.periods)
cashflows[collateral.periods] <- collateral.notional +
    cashflows[collateral.periods]
(cashflows <- c(0, cashflows))
```

```
## [1]  0.000  0.375  0.375 25.375
```

What just happened?

1. We appended a `0` to the rate schedule so we can use the `NPV.2` function.
2. We then parameterized the term sheet (terms of the collateral transaction),
3. We used `rep()` to form coupon cash flows.
4. Then we added notional value repayment to the last cash flow.

Now we can calculate the present value of the bond using `NPV.2`.

```
(Value.0 <- NPV.2(rates, cashflows))
```

```
## [1] 25.3776
```

The answer is 25.378 million pounds sterling or `Value.0 / collateral.notional` times the notional value.

The yield to maturity averages the forward rates across the bond cash flows. A "forward rate" is the rate per period we would expect to earn each period. This is one interpretation of the Internal Rate of Return ("IRR").

```
cashflows.IRR <- cashflows
collateral.ask <- 130
```

```
cashflows.IRR[1] <- -(collateral.ask/100) *
    collateral.notional
## mind the negative sign!
(collateral.IRR.1 <- IRR.1(cashflows.IRR))
```

```
## [1] -0.07112366
```

You end up paying over 7% per annum for the privilege of owning this bond! You call up the European Central Bank, report this overly hefty haircut on your project. You send out a request for proposal to other bond dealers. They come back with an average asking price of 109 (109% of notional).

```
cashflows.IRR <- cashflows
collateral.ask <- 109
cashflows.IRR[1] <- -(collateral.ask/100) *
    collateral.notional
(collateral.IRR.1 <- IRR.1(cashflows.IRR))
```

```
## [1] -0.01415712
```

That's more like it: about 140 basis points (1.41% x 100 basis points per percentage) cost (negative sign).

Let's unwind the project, and the collateral transaction, in 1 year. Let's suppose the yield curve in 1 year has parallel shifted down by 0.005.

```
rate.shift <- -0.005
rates.1 <- c(0, rates[-2]) + rate.shift
cashflows.1 <- c(0, cashflows[-2])
(Value.1 <- NPV.2(rates.1, cashflows.1))
```

```
## [1] 25.37541
```

```
(collateral.return.1 <- Value.1/(-cashflows.IRR[1]) -
    1)
```

```
## [1] -0.0687923
```

This results ooks much more than a break-even return on the collateral transation:

```
(collateral.gainloss <- collateral.notional *
    collateral.return.1) * 1e+06
```

```
## [1] -1719807
```

```
## adjust for millions of euros
```

That's probably someone's salary…(in pounds sterling).

3.3.5 Mind the Interface!

Interfaces mark out a controlled inner environment for our code;

- They allow us to interact with the rest of the system only at the interface.
- Arguments explicitly give the function all the information the function needs to operate and reduces the risk of confusion and error.

- There are exceptions such as true universals like π.
- Likewise, output should only be through the return value.

Let's build (parametric) distributions next.

3.4 Making distributions

As always, let's load some data, this time from the Bureau of Labor Statistics (BLS) and load the export-import price index whose description and graph are at http://data.bls.gov/timeseries/EIUIR?output_view=pct_1mth. We look up the symbols "EIUIR" and "EIUIR100" and download a text file that we then convert to a comma separated variable or `csv` file in Excel. We deposit the `csv` file in a directory and read it into a variable called `EIUIR`.

```
require(xts)
require(zoo)
EIUIR <- read.csv("data/EIUIR.csv")
head(EIUIR)
```

```
##         Date Value
## 1 2006-01-01 113.7
## 2 2006-02-01 112.8
## 3 2006-03-01 112.7
## 4 2006-04-01 115.1
## 5 2006-05-01 117.2
## 6 2006-06-01 117.3
```

```
xmprice <- na.omit(EIUIR)  ## to clean up any missing data
str(xmprice)
```

```
## 'data.frame':    131 obs. of  2 variables:
##  $ Date : Factor w/ 131 levels "2006-01-01","2006-02-01",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ Value: num  114 113 113 115 117 ...
```

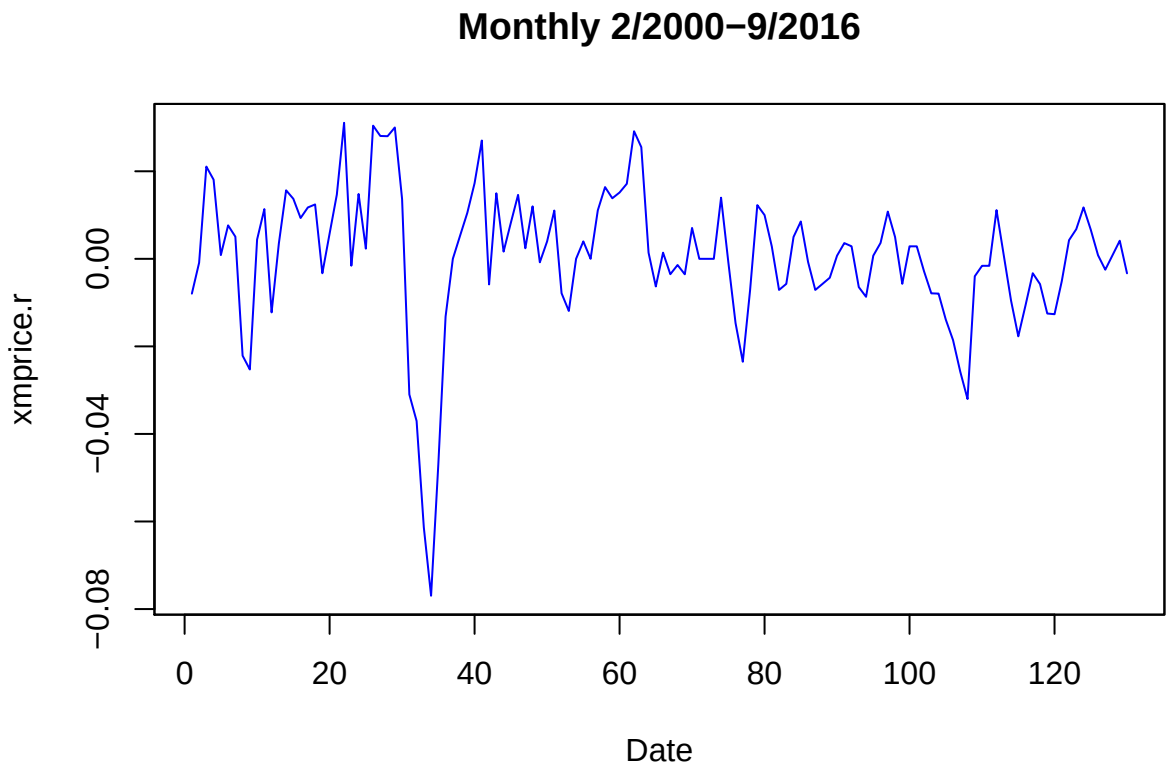
We might have to have installed separatetly the `xts` and `zoo` packages that handle time series data explicitly. The `str()` function indicates that the `Value` column in the data frame contains the export price series. We then compute the natural logarithm of prices and

calculate the differences to get rates of growth from month to month. A simple plot reveals aspects of the data series to explore.

```
xmprice.r <- as.zoo(na.omit((diff(log(xmprice$Value)))))  ## compute rates
head(xmprice.r)
```

```
##             1             2             3             4             5
## -0.0079470617 -0.0008869180  0.0210718947  0.0180805614  0.0008528785
##             6
##  0.0076433493
```

```
plot(xmprice.r, type = "l", col = "blue",
    xlab = "Date", main = "Monthly 2/2000-9/2016")
```



We further transform the data exploring the absolute value of price rates. This is a first stab at understanding the clustering of volatility in financial-economic time series, a topic to which we will return.

```
xmprice.r.df <- data.frame(xmprice.r,
    Date = index(xmprice.r), Rate = xmprice.r[,
        1], Rate.abs <- abs(xmprice.r[,
        1]))
head(xmprice.r.df)
```

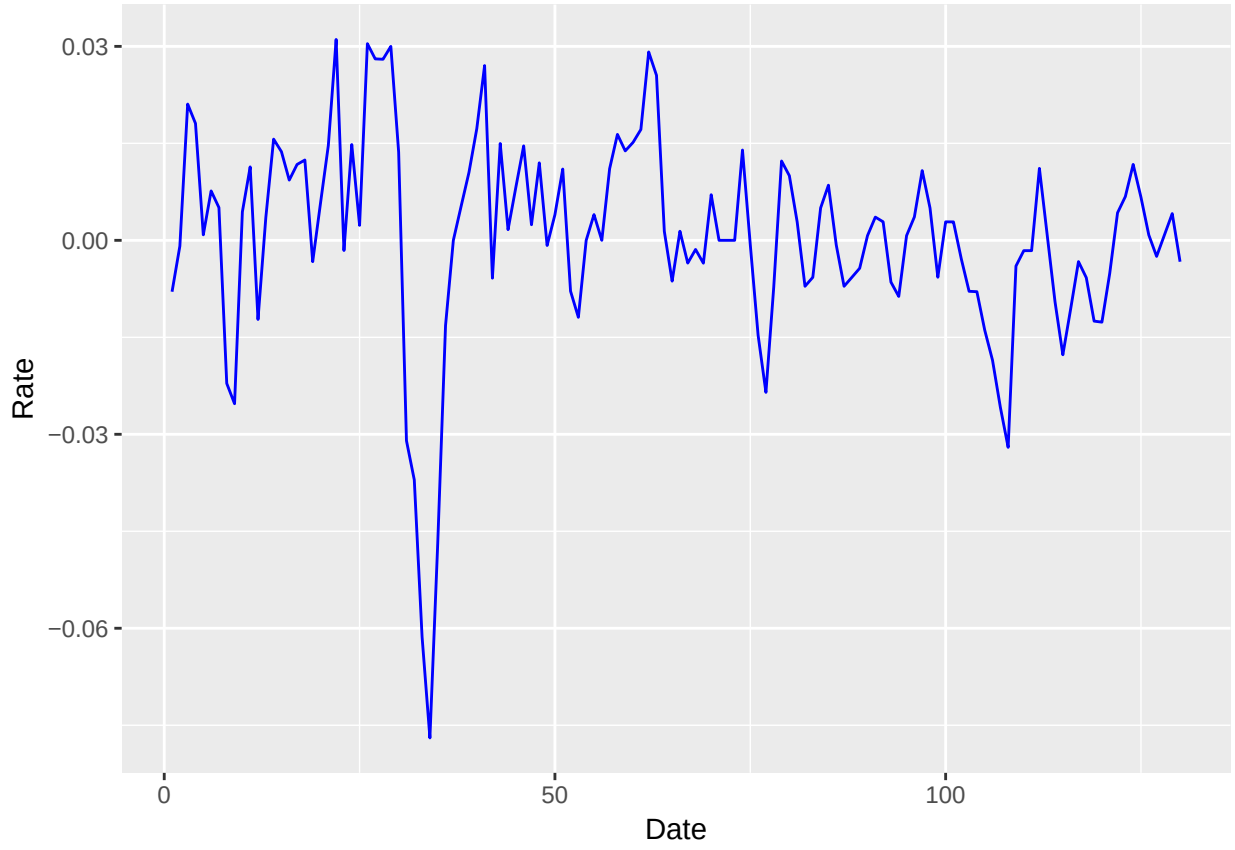
```
##       xmprice.r Date          Rate Rate.abs....abs.xmprice.r...1..
## 1 -0.0079470617    1 -0.0079470617                    0.0079470617
## 2 -0.0008869180    2 -0.0008869180                    0.0008869180
## 3  0.0210718947    3  0.0210718947                    0.0210718947
## 4  0.0180805614    4  0.0180805614                    0.0180805614
## 5  0.0008528785    5  0.0008528785                    0.0008528785
## 6  0.0076433493    6  0.0076433493                    0.0076433493
```

```
str(xmprice.r.df)
```

```
## 'data.frame':    130 obs. of  4 variables:
##  $ xmprice.r                     : num  -0.007947 -0.000887 0.021072 0.018081 0.0008
##  $ Date                          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Rate                          : num  -0.007947 -0.000887 0.021072 0.018081 0.0008
##  $ Rate.abs....abs.xmprice.r...1..: num  0.007947 0.000887 0.021072 0.018081 0.000853
```

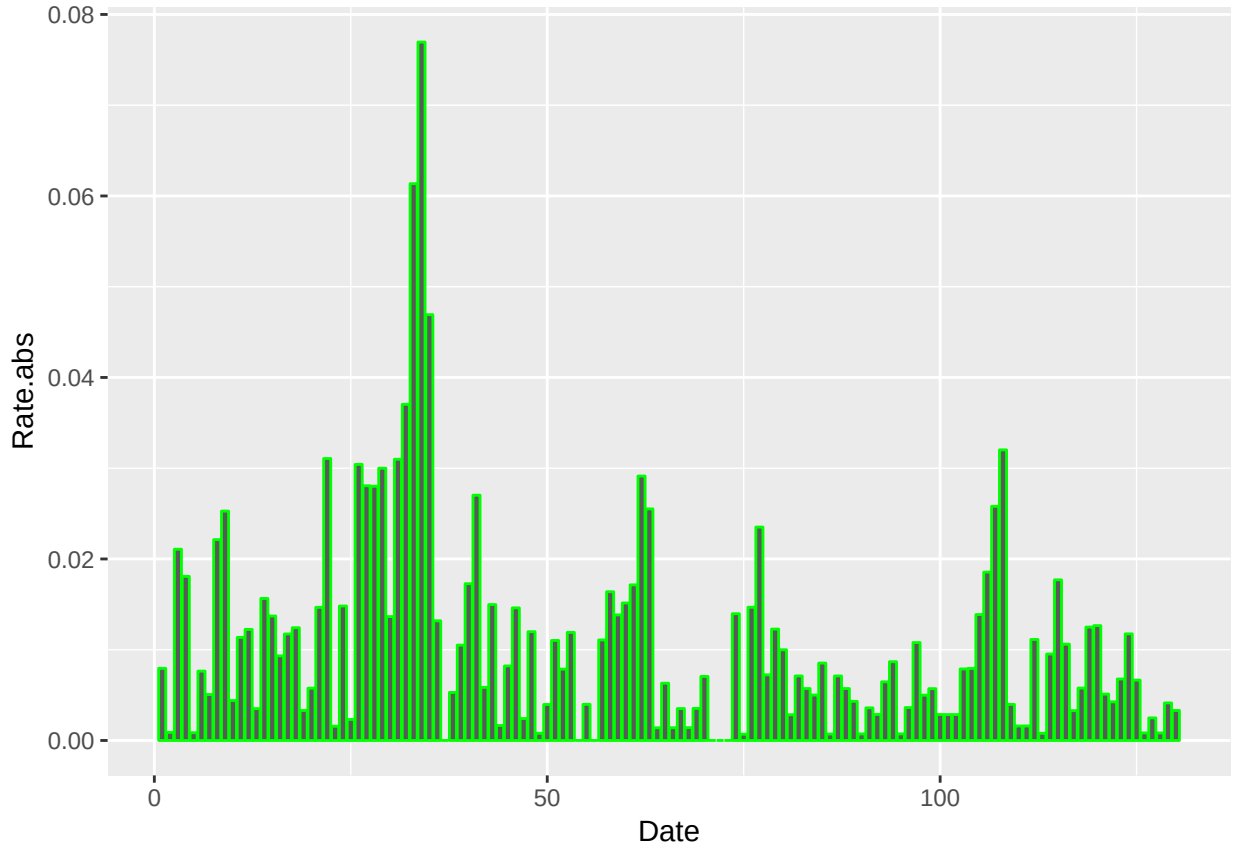
We can achieve a "prettier" plot with the `ggplot2` package. In the `ggplot` statements we use `aes`, "aesthetics", to pick `x` (horizontal) and `y` (vertical) axes. The added (`+`) `geom_line` is the geometrical method that builds the line plot.

```
require(ggplot2)
ggplot(xmprice.r.df, aes(x = Date, y = Rate)) +
    geom_line(colour = "blue")
```



Let's try a bar graph of the absolute value of price rates. We use `geom_bar` to build this picture.

```
require(ggplot2)
ggplot(xmprice.r.df, aes(x = Date, y = Rate.abs)) +
    geom_bar(stat = "identity", colour = "green")
```



3.4.1 Try this exercise

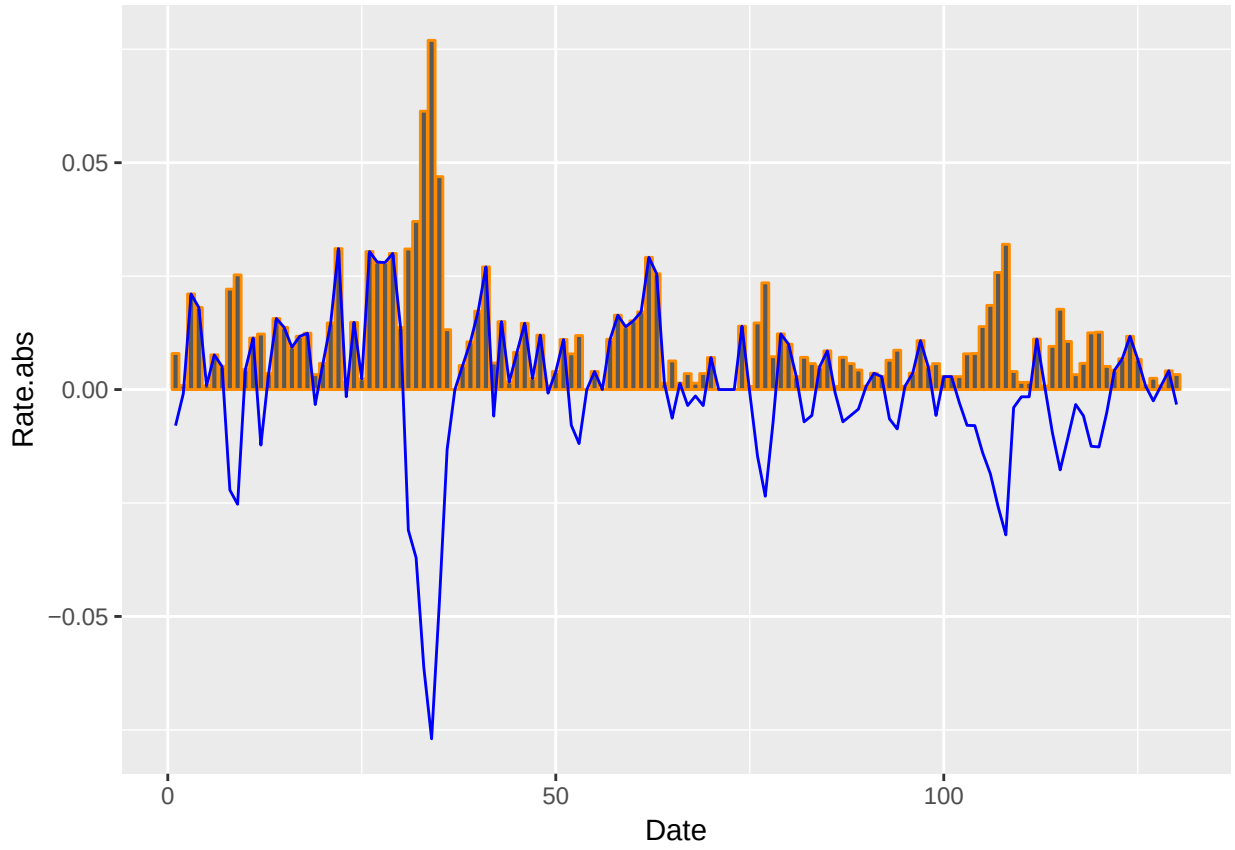
Let's overlay returns (`geom_line`) and their absolute value `geom_bar`.

- `ggplot` declares the canvas using the price data frame.
- `aes` establishes the data series to be used to generate pictures.
- `geom_bar` builds a bar chart.
- `geom_line` overlays the bar chart with a line chart.

By examining this chart, what business questions about your Univeral Export-Import Ltd supply chain could this help answer? Why is this helpful?

```
require(ggplot2)
ggplot(xmprice.r.df, aes(Date, Rate.abs)) +
    geom_bar(stat = "identity", colour = "darkorange") +
```

```
    geom_line(data = xmprice.r.df, aes(Date,
        Rate), colour = "blue")
```



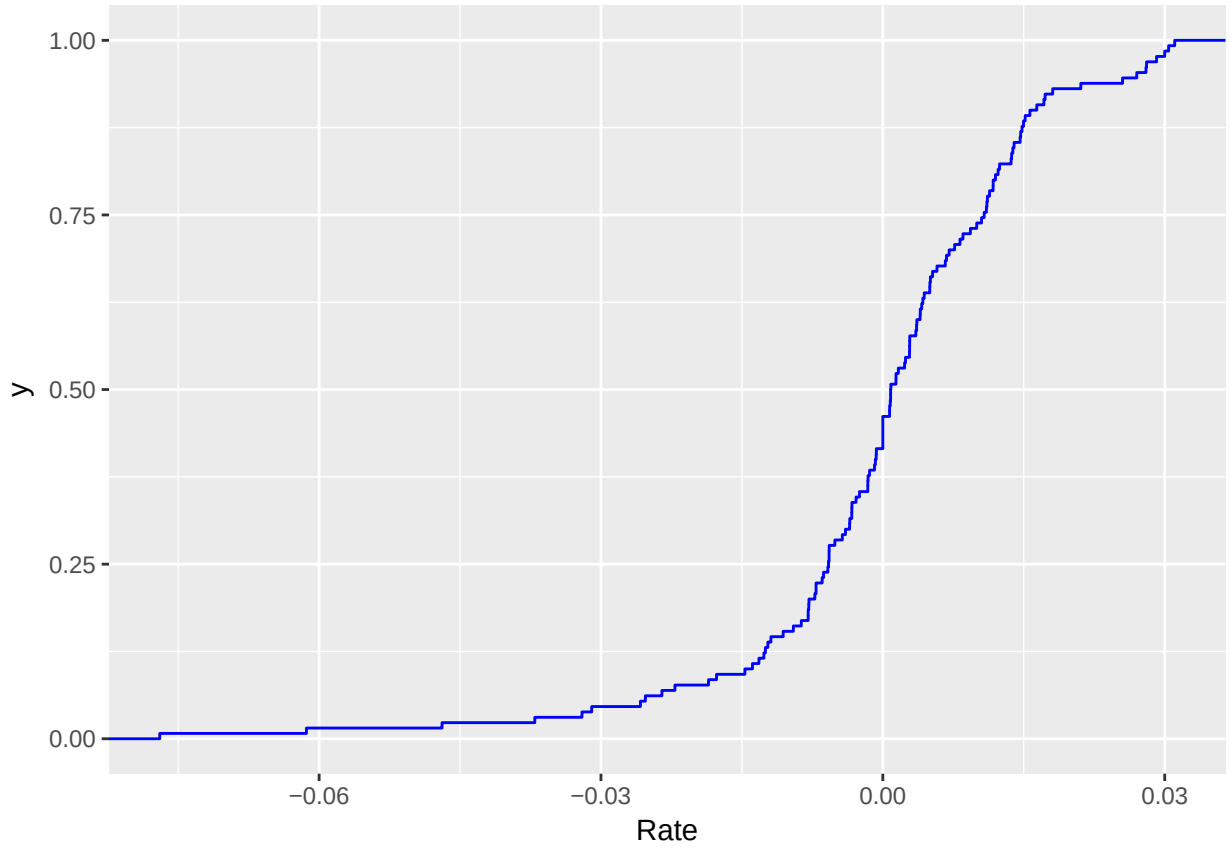
The plot goes a long way to answering the question: When supply and demand tightens, does price volatility cluster?

1. If we are selling, we would experience strong swings in demand and thus in revenue at the customer fulfillment end of the chain.
2. If we are buying, we would experience strong swings in cost and input product utilization at the procurement end of the chain.
3. For the financial implications: we would have a tough time making the earnings we forecast to the market.

3.4.2 Picture this

We import goods as input to our manufacturing process. We might want to know the odds that a very high export-import rate might occur. We can plot a cumulative distribution function (*cdf* or *CDF*) call. we can build this plot using the `stat_ecdf()` function in `ggplot2`.

```
require(ggplot2)
ggplot(xmprice.r.df, aes(Rate)) + stat_ecdf(colour = "blue")
```



3.4.3 Try another exercise

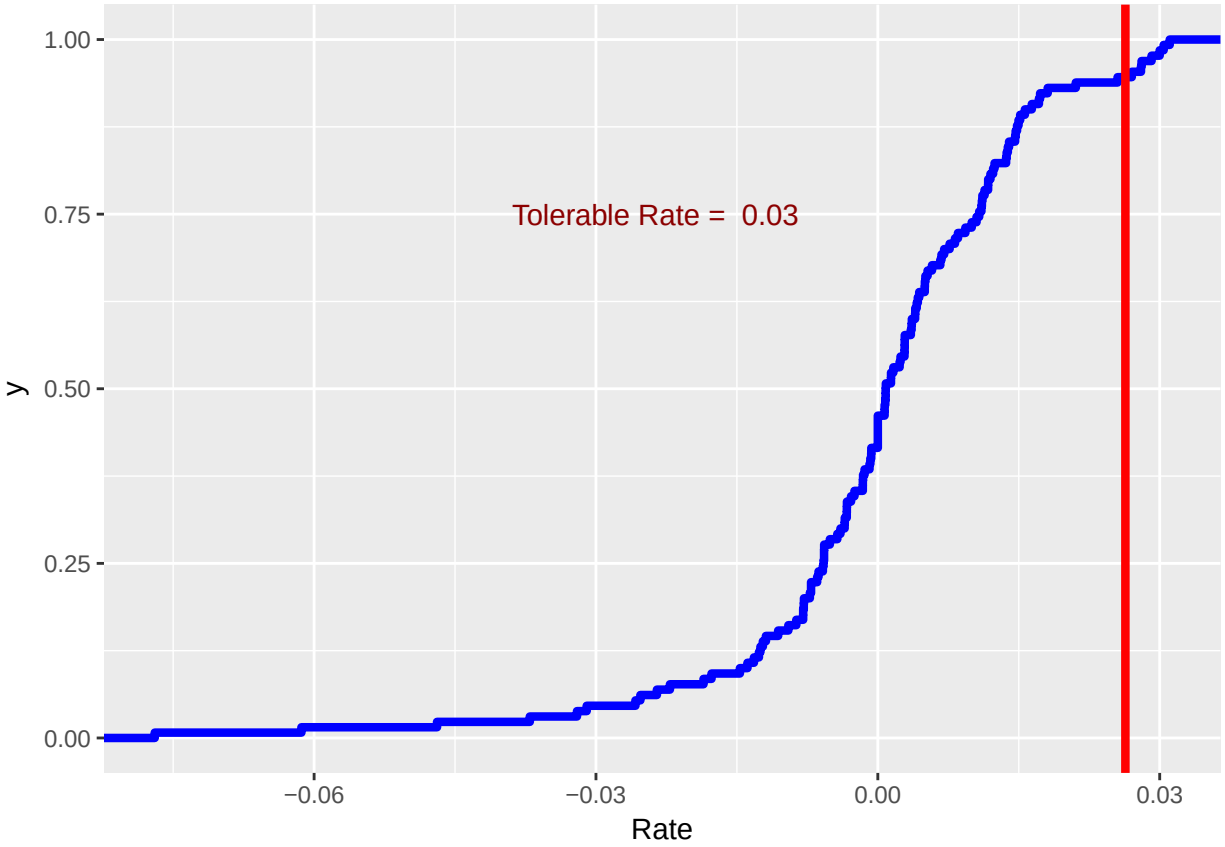
1. Suppose the procurement team's delegation of authority remit states: "Procurement may approve input invoices when there is only a 5% chance that prices will rise any higher than the price rate associated with that tolerance. If input prices do rise higher than the tolerable rate, you must get divisional approval."
2. Plot a vertical line to indicate the maximum tolerable rate for procurement using the BLS EIUR data from 2000 to the present.

- Use `r.tol <- quantile(xmprice.r.df$Rate, 0.95)` to find the tolerable rate.
- Use `+ geom_vline(xintercept = r.tol)` in the CDF plot.

We can implement these requirements with the following code.

```
require(ggplot2)
r.tol.pct <- 0.95
r.tol <- quantile(xmprice.r.df$Rate,
    r.tol.pct)
```

```
r.tol.label <- paste("Tolerable Rate = ",
    round(r.tol, 2))
ggplot(xmprice.r.df, aes(Rate)) + stat_ecdf(colour = "blue",
    size = 1.5) + geom_vline(xintercept = r.tol,
    colour = "red", size = 1.5) + annotate("text",
    x = r.tol - 0.05, y = 0.75, label = r.tol.label,
    colour = "darkred")
```



This may be a little more than we bargained for originally. We used the `paste` and `round` (to two, `2`, decimal places) functions to make a label. We made much thicker lines (`size = 1.5`). At 2% we drew a line with `geom_vline()` and annotated the line with text.

Now that we have *made* some distributions out of live data, let's estimate the parameters of specific distributions that might be fit to that data.

3.5 Optimization

The optimization we will conduct here helps us to find the distribution that best fits the data. We will use results from optimization to simulate that data to help us make decisions *prospectively*.

There are many distributions in R: `?distributions` will tell you all about them.

1. If `name` is the name of a distribution (e.g., `norm` for "normal"), then

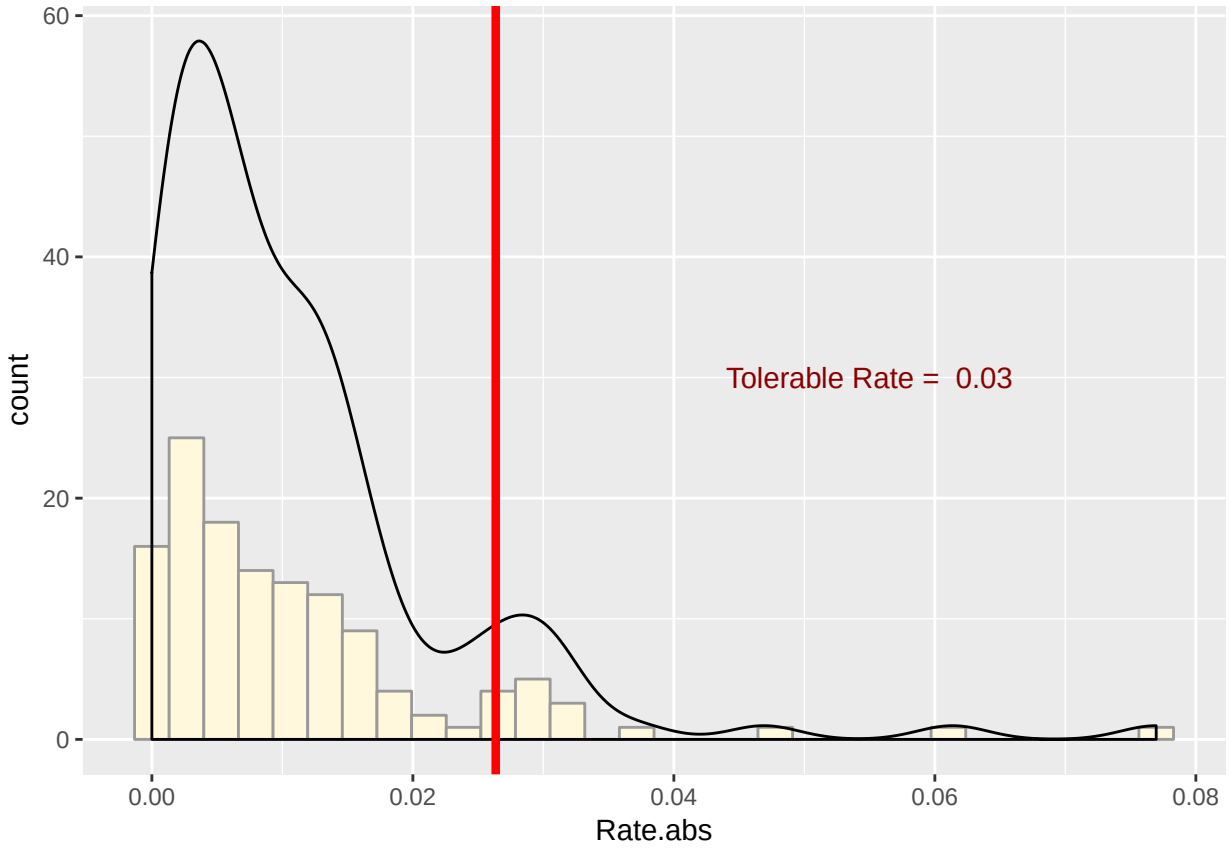
- dname = the probability *d*ensity (if continuous) or probability mass function of `name` (pdf or pmf), think "histogram"
- pname = the cumulative *p*robability function (CDF), think "s-curve"
- qname = the *q*uantile function (inverse to CDF), "think tolerance line"
- rname = draw *r*andom numbers from `name` (first argument always the number of draws), think whatever you want…it's kind of random

2. And ways to write your own (like the `pareto` distribution we use in finance)

3.5.1 Try this exercise

Suppose the `EIUR` price series is the *benchmark* in several import contracts you write as the procurement officer of your organization. Your concern is with volatility. Thus you think that you need to simulate the size of the price rates, whatever direction they go in. Draw the histogram of the absolute value of price rates.

```
require(ggplot2)
r.tol <- quantile(xmprice.r.df$Rate,
    0.95)
r.tol.label <- paste("Tolerable Rate = ",
    round(r.tol, 2))
ggplot(xmprice.r.df, aes(Rate.abs)) +
    geom_histogram(fill = "cornsilk",
        colour = "grey60") + geom_density() +
    geom_vline(xintercept = r.tol, colour = "red",
        size = 1.5) + annotate("text",
    x = 0.055, y = 30, label = r.tol.label,
    colour = "darkred")
```



This series is right-skewed and thickly-tailed. We will use this function to pull several of the statistics calculations together.

```
## r_moments function INPUTS: r vector
## OUTPUTS: list of scalars (mean, sd,
## median, skewness, kurtosis)
data_moments <- function(data) {
    require(moments)
    mean.r <- mean(data)
    sd.r <- sd(data)
    median.r <- median(data)
    skewness.r <- skewness(data)
    kurtosis.r <- kurtosis(data)
    result <- data.frame(mean = mean.r,
        std_dev = sd.r, median = median.r,
        skewness = skewness.r, kurtosis = kurtosis.r)
    return(result)
}
```

We might need to `install.packages("moments")` to make this function operate. We then run these sentences.

```
ans <- data_moments(xmprice.r.df$Rate.abs)
ans <- round(ans, 4)
knitr::kable(ans)
```

mean	std_dev	median	skewness	kurtosis
0.0109	0.0117	0.0074	2.523	12.1545

As we visually surmised, the series is right-skewed and very thickly tailed. This may indicate that the `gamma` and `pareto` functions may help us describe these series and prepare us for simulations, estimation, and inference. We will make liberal use of the `fitdistr` function from `MASS` and come back to this moments function.

3.6 Estimate until morale improves...

We will try one method that works often enough in practice, Method of Moments ("MM" or, more affectionately, "MOM"). This estimation technique finds the distribution parameters such that the moments of the data match the moments of the distribution. Other methods include:

- `fitdistr`: Let the opaque box do the job for you; look at the package `MASS` which uses the "maximum likelihood" approach in the `fitdistr` estimating function (like `lm` for regression).
- `fitdistrplus`: For the more adventurous analyst, this package contains several methods, including MM, to get the job done.

uppose we believe that absolute price rates somehow follow a `gamma` distribution. You can look up this distribution easily enough in Wikipedia's good article on the subject. Behind managerial scenes, we can model the loss with `gamma` severity function that allows for skewness and "heavy" tails. We can specify the `gamma` distribution with by shape, α, and scale, β, parameters. We will find in operational loss analysis that this distribution is especially useful for time-sensitive losses.

It turns out We can specify the shape and scale parameters using the mean, μ, and standard deviation, σ of the random severities, X. The scale parameter is

$$\beta = \sigma^2/\mu,$$

and shape parameter,

$$\alpha = \mu^2/\sigma^2.$$

The distribution itself is defined as

$$f(x; alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)},$$

where, to have a complete statement,

$$\Gamma(x) = \int_0^\infty x^{t-1} e^{-x} dx.$$

Let's finally implement into R.

First, we will load a cost sample and calculate moments and gamma parameters:

```
cost <- read.csv("data/cost.csv")
cost <- cost$x
cost.moments <- data_moments(cost)
cost.mean <- cost.moments$mean
cost.sd <- cost.moments$std_dev
(cost.shape <- cost.mean^2/cost.sd^2)
```

```
## [1] 19.06531
```

```
(cost.scale <- cost.sd^2/cost.mean)
```

```
## [1] 0.5575862
```

```
gamma.start <- c(cost.shape, cost.scale)
```

When performing these calculations, be sure that the function `data_moments` is loaded into the workspace.

Second, we can use `fitdistr` from the `Mass` package to estimate the `gamma` parameters `alpha` and `beta`.

```
require(MASS)
fit.gamma.cost <- fitdistr(cost, "gamma")
fit.gamma.cost
```

```
##      shape         rate
##   20.2998092    1.9095724
##  ( 2.3729250) ( 0.2259942)
```

Third, we construct the ratio of estimates to the standard error of estimates. This computes the number of standard deviations away from zero the estimates are. If they are "far" enough away from zero, we have reason to reject the null hypothesis that the estimates are no different from zero.

```
(cost.t <- fit.gamma.cost$estimate/fit.gamma.cost$sd)
```

```
##    shape     rate
## 8.554762 8.449652
```

```
knitr::kable(cost.t)
```

shape	8.554762
rate	8.449652

Nice…but we also note that the scale parameter is `fit.gamma.cost$estimate[2] / gamma.start[2]` times the moment estimates above.

3.6.1 Try this exercise

Let's use the export-input price series rates and the `t` distribution instead of the `gamma`.

First, we calculate the moments (mean, etc.).

```
rate <- xmprice.r.df$Rate
rate.moments <- data_moments(rate)
(rate.mean <- rate.moments$mean)
```

```
## [1] 0.0004595748
```

```
(rate.sd <- rate.moments$std_dev)
```

```
## [1] 0.01602021
```

Second, we use `fitdistr` from the `Mass` package to estimate the parameters of the `t` distribution.

```
fit.t.rate <- fitdistr(rate, "t", hessian = TRUE)
fit.t.rate
```

```
##         m             s            df
##   0.001791738   0.009833018   2.888000806
##  (0.001059206) (0.001131725) (0.843729312)
```

Third, we infer if we did a good job or not. The null hypothesis is that the parameters are no different than zero (H_0). We calculate t statistics to approximate the mapping of parameter estimates to a dimensionless scale that will compute the number of standard deviations from the null hypothesis that the parameters are just zero and of no further use.

```
##        m        s       df
## 1.691586 8.688522 3.422900
```

Nice…but that location parameter is a bit low relative to moment estimate. What else can we do? Simulate the estimated results and see if, at least, skewness and kurtosis line up with the moments.

3.7 Summary

We used our newly found ability to write functions and built insightful pictures of distributions. We also ran nonlinear (gamma and t-distributions are indeed very nonlinear) regressions using a package and the method of moments. All of this to answer critical business questions.

More specifically we waded into:

- Excel look alike processes: Pivot tables and VLOOKUP
- Excel look alike functions
- Graphics to get insights into distributions
- Estimating parameters of distribution
- Goodness of fit

3.8 Further Reading

Teetor's various chapters have much to guide us in the writing of functions and the building of expressions. Present value and internal rate of return can be found in Brealey et al. Use of `ggplot2` in this chapter relies heavily on Chang (2014).

3.9 Practice Sets

These practice sets reference materials developed in this chapter. We will explore new problems and data with models, `R` packages, tables, and plots worked out already in the chapter.

3.9.1 Set A

In this set we will build a data set using filters and `if` and `diff` statements. We will then answer some questions using plots and a pivot table report. We will then review a function to house our approach in case we would like to run the same analysis on other data sets.

3.9.1.1 Problem

Supply chain managers at our company continue to note we have a significant exposure to heating oil prices (Heating Oil No. 2, or HO2), specifically New York Harbor. The exposure hits the variable cost of producing several products. When HO2 is volatile, so is earnings. Our company has missed earnings forecasts for five straight quarters. To get a handle on Brent we download this data set and review some basic aspects of the prices.

```
# Read in data
HO2 <- read.csv("data/nyhh02.csv", header = T,
    stringsAsFactors = F)
# stringsAsFactors sets dates as
# character type
head(HO2)
HO2 <- na.omit(HO2)  ## to clean up any missing data
str(HO2)  # review the structure of the data so far
```

3.9.1.2 Questions

1. What is the nature of HO2 returns? We want to reflect the ups and downs of price movements, something of prime interest to management. First, we calculate percentage changes as log returns. Our interest is in the ups and downs. To look at that we use `if` and `else` statements to define a new column called `direction`. We will build a data frame to house this analysis.

```
# Construct expanded data frame
return <- as.numeric(diff(log(HO2$DHOILNYH))) *
    100
size <- as.numeric(abs(return))  # size is indicator of volatility
direction <- ifelse(return > 0, "up",
    ifelse(return < 0, "down", "same"))  # another indicator of volatility
date <- as.Date(HO2$DATE[-1], "%m/%d/%Y")  # length of DATE is length of return +1: omi
price <- as.numeric(HO2$DHOILNYH[-1])  # length of DHOILNYH is length of return +1: omi
HO2.df <- na.omit(data.frame(date = date,
    price = price, return = return, size = size,
    direction = direction))  # clean up data frame by omitting NAs
str(HO2.df)
```

We can plot with the `ggplot2` package. In the `ggplot` statements we use `aes`, "aesthetics", to pick `x` (horizontal) and `y` (vertical) axes. Use `group =1` to ensure that all data is plotted. The added (`+`) `geom_line` is the geometrical method that builds the line plot.

Let's try a bar graph of the absolute value of price rates. We use `geom_bar` to build this picture.

Now let's build an overlay of `return` on `size`.

2. Let's dig deeper and compute mean, standard deviation, etc. Load the `data_moments()` function. Run the function using the `HO2.df$return` subset and write a `knitr::kable()` report.

3. Let's pivot `size` and `return` on `direction`. What is the average and range of returns by direction? How often might we view positive or negative movements in HO2?

3.9.2 Set B

We will use the data from the previous set to investigate the distribution of returns we generated. This will entail fitting the data to some parametric distributions as well as plotting and building supporting data frames.

3.9.2.1 Problem

We want to further characterize the distribution of up and down movements visually. Also we would like to repeat the analysis periodically for inclusion in management reports.

3.9.2.2 Questions

1. How can we show the differences in the shape of ups and downs in HO2, especially given our tolerance for risk? Let's use the `HO2.df` data frame with `ggplot2` and the cumulative relative frequency function `stat_ecdf`.
2. How can we regularly, and reliably, analyze HO2 price movements? For this requirement, let's write a function similar to `data_moments`.

Let's test `HO2_movement()`.

Morale: more work today (build the function) means less work tomorrow (write yet another report).

3. Suppose we wanted to simulate future movements in HO2 returns. What distribution might we use to run those scenarios? Here, let's use the `MASS` package's `fitdistr()` function to find the optimal fit of the HO2 data to a parametric distribution.

3.9.3 Practice Set Debrief

1. List the R skills needed to complete these practice sets.
2. What are the packages used to compute and graph results. Explain each of them.
3. How well did the results begin to answer the business questions posed at the beginning of each practice set?

3.10 Project

3.10.1 Background

Your company uses natural gas as a major input to recycle otherwise non-recoverable waste. The only thing that prevents the operation from being a little better than break-even is volatile natural gas prices. In its annual review, management requires information and analysis of recycling operations with a view to making decisions about outsourcing contracts. These contracts typically have three year tenors.

Since management will be making do or outsource decisions over a three year forward span, analysts will build models that characterize historical movements in natural gas prices, the

volatility of those prices, and probability distributions to simulate future natural gas scenarios.

3.10.2 Data

In a preliminary analysis, you gather data from FRED on daily natural gas prices. You whill use this data to characterize historical natural gas price movements and construct provisional probability distributions for eventual generation of forward scenarios.

3.10.3 Workflow

1. Data collection. Collect, clean, and review data definitions, and data transformations of price into returns. Use tables and graphs to report results.
2. Analysis.

- Group prices into up, same (no movement), and down movements using percent change in daily price as the criterion.
- Build a table of summary statistics that pivots the data and computes metrics.
- Graph the cumulative probability of an up, same, and down group of historical returns.
- Estimate Student's t distribution parameters for up, same, and down movements in natural gas returns.

3. Observations and Recommendations.

- Summarize the data, its characteristics, and applicability to attend to the problem being solved for management.
- Discuss key take-aways from analytical results that are relevant to the decisions that managers will make.
- Produce an `R Markdown` document with code chunks to document and interpret results.
- The format will introduce the problem to be analyzed, with sections that discuss the data used, and which follow the work flow.

3.10.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.
- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.

- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.
- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.
- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

3.11 References

Brealey

Chang

Teetor

knitr

xts

zoo

BLS

Chapter 4

Macrofinancial Data Analysis

4.1 Imagine This

Your US-based company just landed a contract worth more than 20 percent of your company's current revenue in Spain. Now that everyone has recovered from this coup, your management wants you to

1. Retrieve and begin to analyze data about the Spanish economy
2. Compare and contrast Spanish stock market and government-issued debt value versus the United States and several other countries
3. Begin to generate economic scenarios based on political events that may, or may not, happen in Spain

Up to this point we had reviewed several ways to manipulate data in `R`. We then reviewed some basic finance and statistics concepts in `R`. We also got some idea of the financial analytics workflow.

1. What decision(s) are we making?
2. What are the key business questions we need to support this decision?
3. What data do we need?
4. What tools do we need to analyze the data?
5. How do we communicate answers to inform the decision?

4.1.1 Working an example

Let's use this workflow to motivate our work in this chapter.

1. Let's identify a decision at work (e.g., investment in a new machine, financing a building, acquisition of customers, hiring talent, locating manufacturing).

2. For this decision we will list three business questions you need to inform the decision we chose.

3. Now we consider data we might need to answer one of those questions and choose from this set:

- Macroeconomic data: GDP, inflation, wages, population
- Financial data: stock market prices, bond prices, exchange rates, commodity prices

Here is the example using the scenario that started this chapter.

1. Our decision is **supply a new market segment**

- Product: voltage devices with supporting software
- Geography: Spain
- Customers: major buyers at Iberdrola, Repsol, and Endesa

2. We pose three business questions:

- How would the performance of these companies affect the size and timing of orders?
- How would the value of their products affect the value of our business with these companies?
- We are a US functional currency firm (see FAS 52), so how would we manage the repatriation of accounts receivable from Spain?

3. Some data and analysis to inform the decision could include

- Customer stock prices: volatility and correlation
- Oil prices: volatility
- USD/EUR exchange rates: volatility
- All together: correlations among these indicators

4.1.2 How we will proceed

This chapter will develop styliZed facts of the market. These continue to be learned the hard Way: financial data is not independent, it possesses volatile volatility, and has extremes.

- Financial stock, bond, commodity…you name it…have highly interdependent relationships.
- Volatility is rarely constant and often has a structure (mean reversion) and is dependent on the past.
- Past shocks persist and may or may not dampen (rock in a pool).
- Extreme events are likely to happen with other extreme events.
- Negative returns are more likely than positive returns (left skew).

4.2 Building the Stylized Facts

Examples from the 70s, 80s, and 90s have multiple intersecting global events influencing decision makers. We will load some computational help and some data from

Brent, format dates, and create a time series object (package `zoo' will be needed by packagesfBasicsandevir'):

```
library(fBasics)
library(evir)
library(qrmdata)
library(zoo)
data(OIL_Brent)
str(OIL_Brent)
```

```
## An 'xts' object on 1987-05-20/2015-12-28 containing:
##   Data: num [1:7258, 1] 18.6 18.4 18.6 18.6 18.6 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "OIL_Brent"
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

We will compute rates of change for Brent oil prices next.

```
Brent.price <- as.zoo(OIL_Brent)
str(Brent.price)
```

```
## 'zoo' series from 1987-05-20 to 2015-12-28
##   Data: num [1:7258, 1] 18.6 18.4 18.6 18.6 18.6 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr "OIL_Brent"
##   Index:  Date[1:7258], format: "1987-05-20" "1987-05-21" "1987-05-22" "1987-05-25" "
```

```
Brent.return <- diff(log(Brent.price))[-1] *
    100
colnames(Brent.return) <- "Brent.return"
head(Brent.return, n = 5)
```

```
##            Brent.return
## 1987-05-22    0.5405419
## 1987-05-25    0.2691792
## 1987-05-26    0.1611604
## 1987-05-27   -0.1611604
## 1987-05-28    0.0000000
```

```
tail(Brent.return, n = 5)
```

```
##            Brent.return
## 2015-12-21   -3.9394831
## 2015-12-22   -0.2266290
```

```
## 2015-12-23    1.4919348
## 2015-12-24    3.9177726
## 2015-12-28   -0.3768511
```

Let's look at this data with box plots and autocorrelation functions. Box plots will show minimum to maximum with the mean in the middle of the box. Autocorrelation plots will reveal how persistent the returns are over time.

We run these statements.

```
boxplot(as.vector(Brent.return), title = FALSE,
    main = "Brent Daily % Change", col = "blue",
    cex = 0.5, pch = 19)
skewness(Brent.return)
kurtosis(Brent.return)
```

This time series plot shows lots of return clustering and spikes, especially negative ones.

Performing some "eyeball econometrics" these clusters seem to occur around - The oil embargo of the '70s - The height of the new interest rate regime of Paul Volcker at the Fed - "Black Monday" stock market crash in 1987 - Gulf I - Barings and other derivatives business collapses in the '90s

2. Let's look at the likelihood of positive versus negative returns. We might want to review skewness and kurtosis definitions and ranges to help us.

Now to look at persistence:

```
acf(coredata(Brent.return), main = "Brent Daily Autocorrelogram",
    lag.max = 20, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
pacf(coredata(Brent.return), main = "Brent Daily Partial Autocorrelogram",
    lag.max = 20, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
```

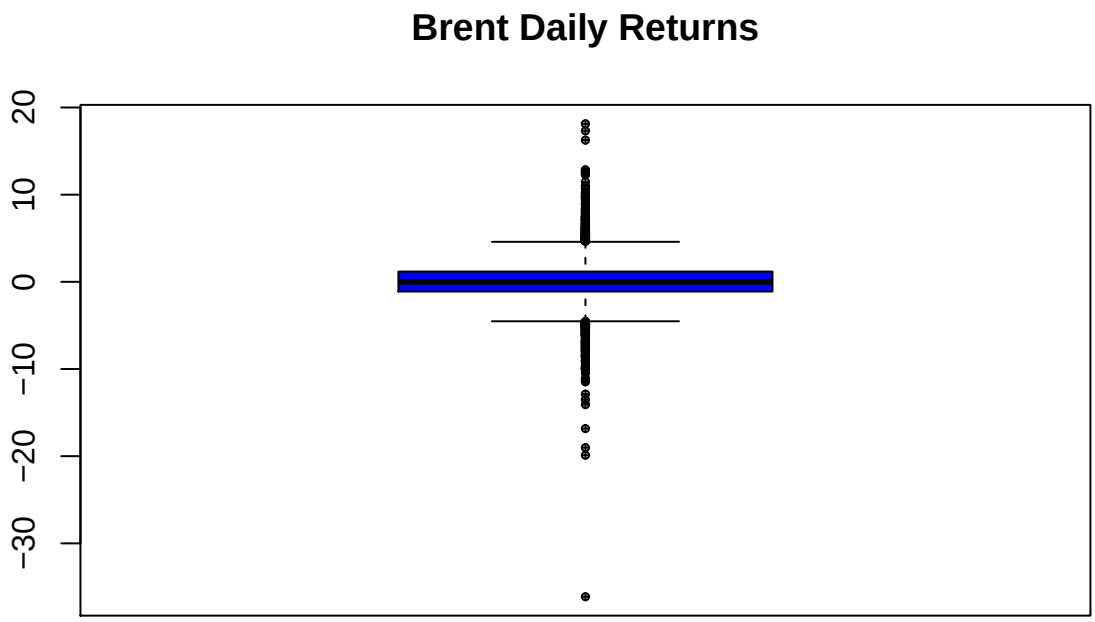
Confidence intervals are the red dashed lines. ACF at lag 6 means the correlation of current Brent returns with returns 6 trading days ago, including any correlations from trading day 1 through 6. PACF is simpler: it is the raw correlation between day 0 and day 6. ACF starts at lag 0 (today); PACF starts at lag 1 (yesterday).

3. How many trading days in a typical week or in a month? Comment on the spikes (blue lines that grow over or under the red dashed lines).

4. How thick is that tail?

Here is a first look:

```
boxplot(as.vector(Brent.return), title = FALSE,
    main = "Brent Daily Returns", col = "blue",
    cex = 0.5, pch = 10)
```



… with some basic stats to back up the eyeball econometrics in the box plot:

```
skewness(Brent.return)
```

```
## [1] -0.6210447
## attr(,"method")
## [1] "moment"
```

```
kurtosis(Brent.return)
```

```
## [1] 14.62226
## attr(,"method")
## [1] "excess"
```

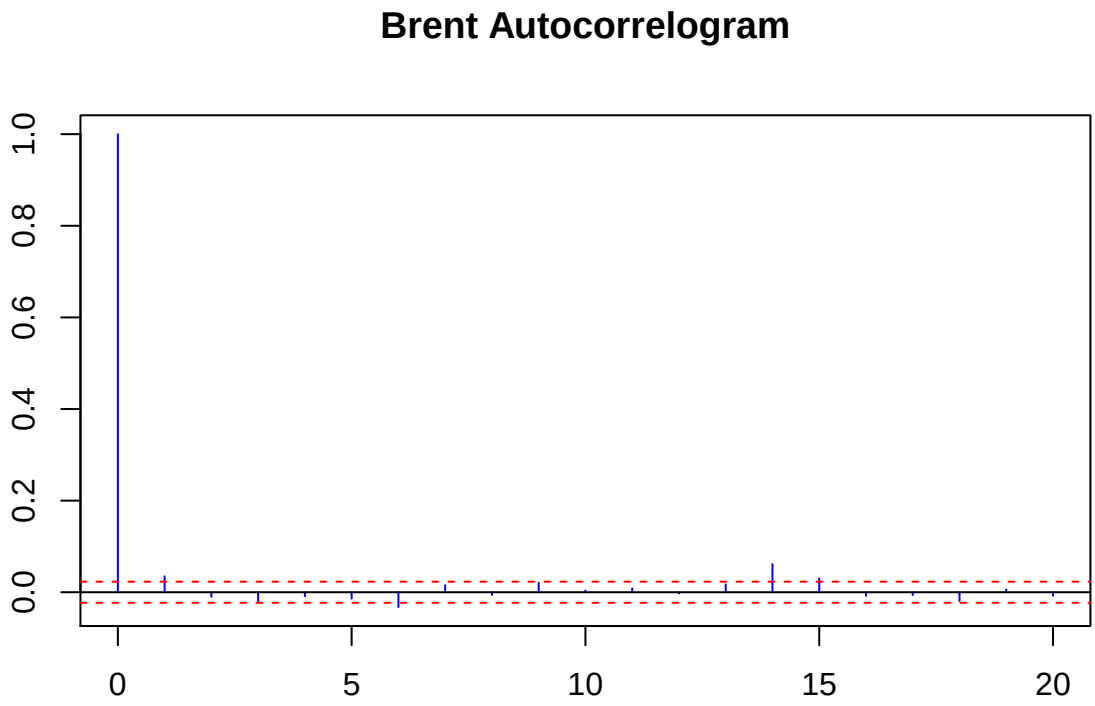
- A negative skew means there are more observations less than the median than greater.
- This high a kurtosis means a pretty heavy tail, especially in negative returns. That means they have happened more often than positive returns.
- A preponderance of negative returns frequently happening spells trouble for anyone owning these assets.

4.2.1 Implications

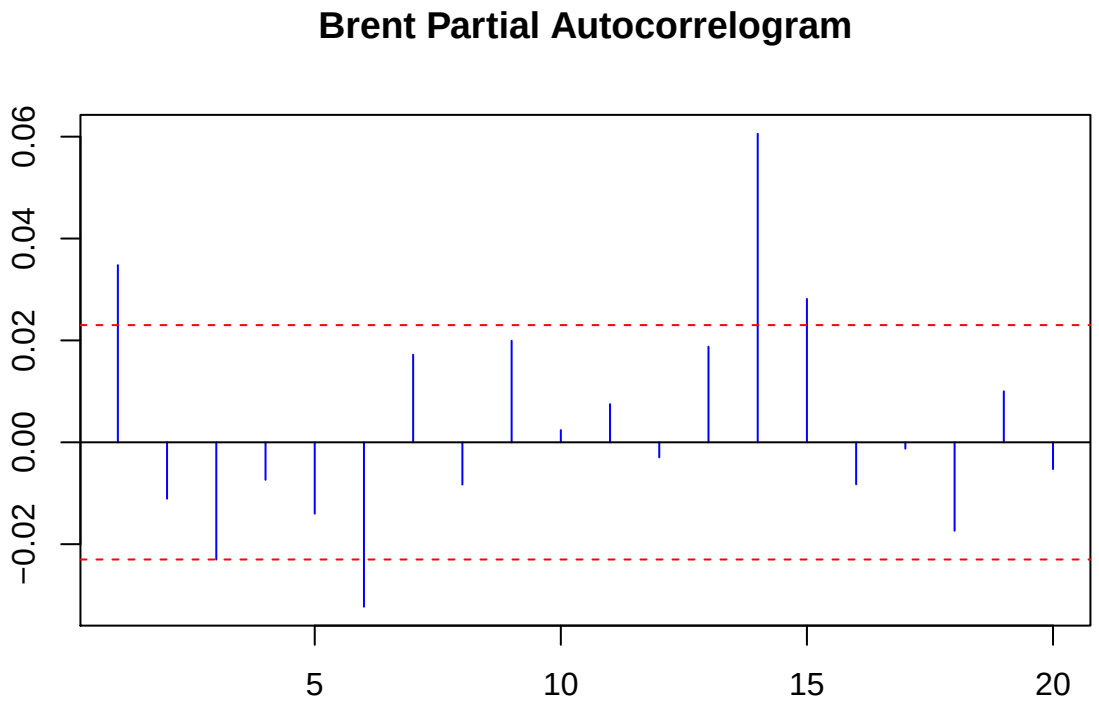
- We should recommend that management budget for the body of the distribution from the mean and out to positive levels.
- At the same time management should build a comprehensive playbook for the strong possibility that bad tail events frequently happen and might happen again (and why shouldn't they?).

3. Now for something really interesting

```
acf(coredata(Brent.return), main = "Brent Autocorrelogram",
    lag.max = 20, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
```




```
pacf(coredata(Brent.return), main = "Brent Partial Autocorrelogram",
    lag.max = 20, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
```



On average there are 5 days in the trading week and 20 in the trading month.

Some further thoughts:

- There seems to be positive weekly and negative monthly cycles.
- On a weekly basis negative rates (5 trading days ago) are followed by negative rates (today) and vice-versa with positive rates.
- On a monthly basis negative rates (20 days ago) are followed by positive rates (today).
- There is memory in the markets: positive correlation at least weekly up to a month ago reinforces the strong and frequently occurring negative rates (negative skew and leptokurtotic, a.k.a. heavy tails).
- Run the PACF for 60 days to see a 40-day negative correlation as well.

4.2.2 Now for somthing really interesting...again

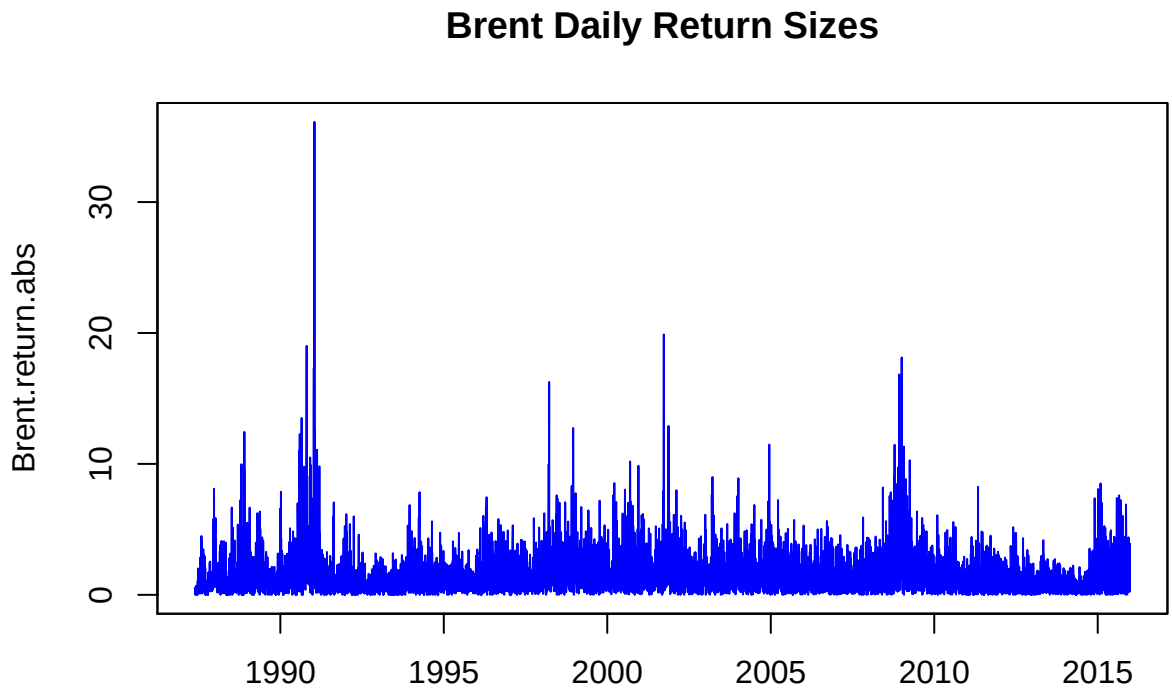
Let's look just at the size of the Brent returns. The absolute value of the returns (think of oil and countries entering and leaving the EU!) can signal contagion, herd mentality, and simply very large margin calls (and the collateral to back it all up!). Let's run this code:

```
Brent.return.abs <- abs(Brent.return)
## Trading position size matters
Brent.return.tail <- tail(Brent.return.abs[order(Brent.return.abs)],
```

```
    100)[1]
## Take just the first of the 100
## observations and pick the first
index <- which(Brent.return.abs > Brent.return.tail,
    arr.ind = TRUE)
## Build an index of those sizes that
## exceed the heavy tail threshold
Brent.return.abs.tail <- timeSeries(rep(0,
    length(Brent.return)), charvec = time(Brent.return))
## just a lot of zeros we will fill up
## next
Brent.return.abs.tail[index, 1] <- Brent.return.abs[index]
## A Phew! is in order
```

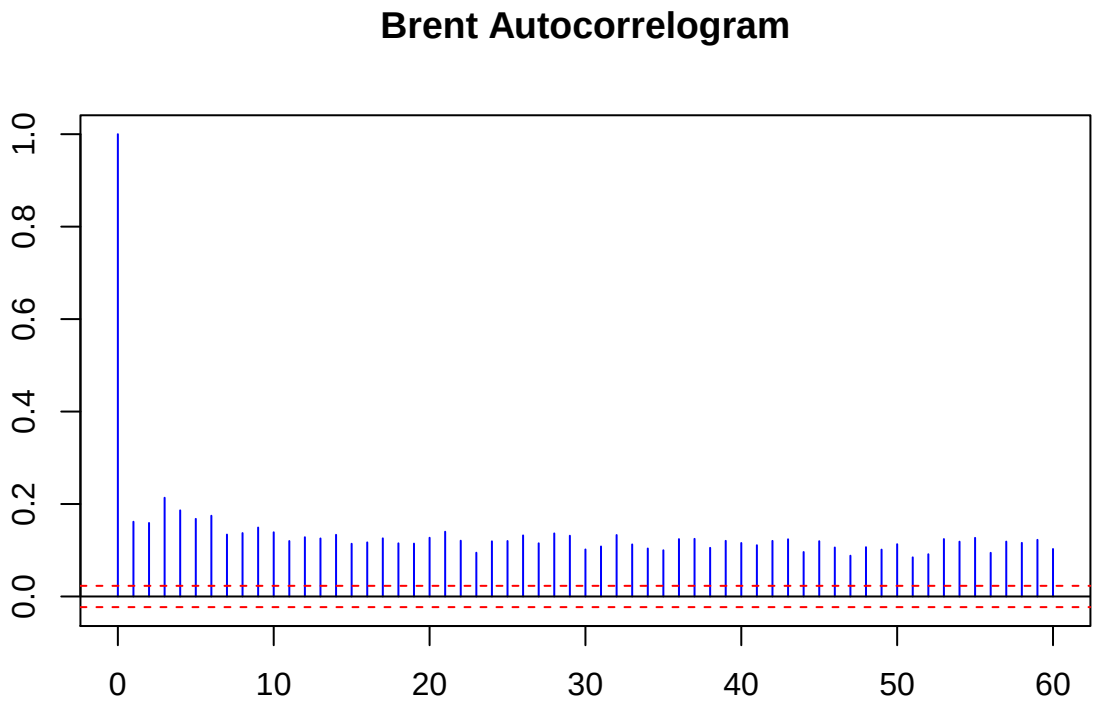
What did we do? Let's run some plots next.

```
plot(Brent.return.abs, xlab = "", main = "Brent Daily Return Sizes",
    col = "blue")
```

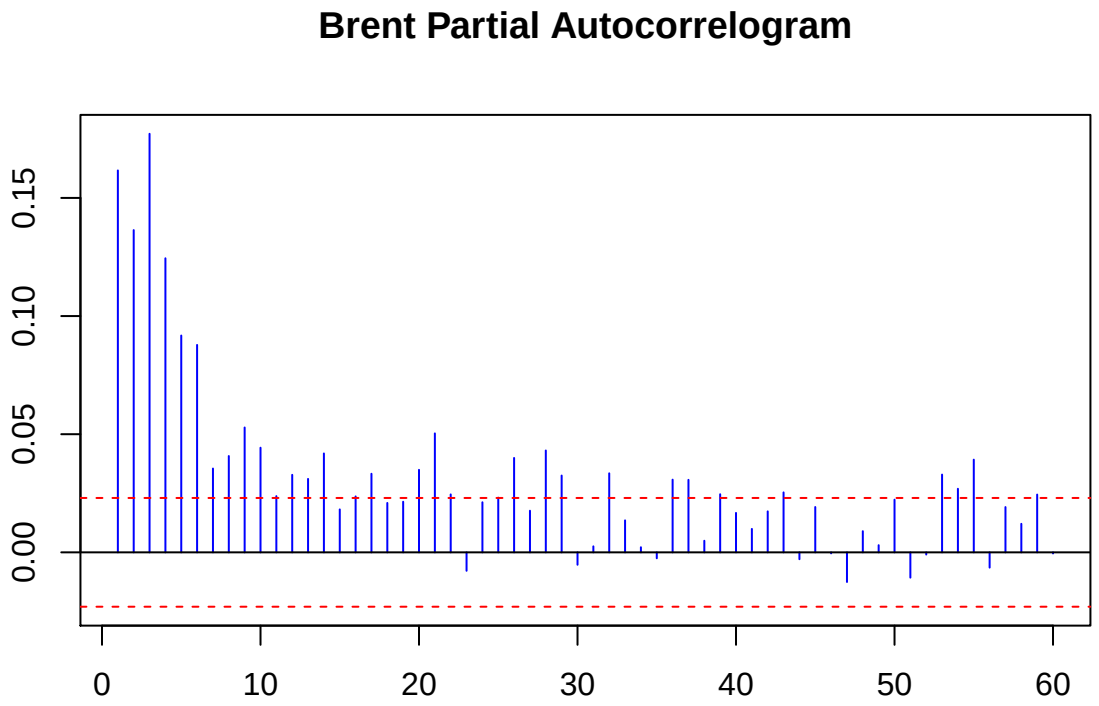


We see lots of return volatility – just in the pure size along. These are correlated with financial innovations from the '80s and '90s, as well as Gulf 1, Gulf 2, Great Recession, and its 9/11 antecedents.

```
acf(coredata(Brent.return.abs), main = "Brent Autocorrelogram",
    lag.max = 60, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
```




```
pacf(coredata(Brent.return.abs), main = "Brent Partial Autocorrelogram",
    lag.max = 60, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
```



There is *Volatility Clustering* galore. Strong persistent lags of absolute movements in returns evidenced by the `ACF` plot. There is evidence of dampening with after shocks past trading 10 days 10 ago. Monthly volatility affects today's performance.

Some of this volatility arises from the way Brent is traded. It is lifted through well-heads in the North Sea. It then is scheduled for loading onto ships and loads are then bid, along with routes to destination. It takes about five days to load crude and another five to unload. At each partial loading and unloading, the crude is re-priced. Then there is the voyage lag itself, where paper claims to wet crude create further pricing, and volatility.

Next we explore the relationships among financial variables.

4.3 Getting Caught in the Cross-Current

Now our job is to ask the really important questions around connectivity. Suppose we are banking our investments in certain sectors of an economy, with its GDP, financial capability, employment, exports and imports, and so on.

- How will we decide to contract for goods and services, segment vendors, segment customers, based on these interactions?
- How do we construct out portfolio of business opportunities?

- How do we identify insurgent and relational risks and build a playbook to manage these?
- How will changes in one sector's factors (say, finance, political will) affect factors in another?

We will now stretch our univariate analysis a bit and look at *cross-correlations* to help us get the ground truth around these relationships, and *begin* to answer some of these business questions in a more specific context.

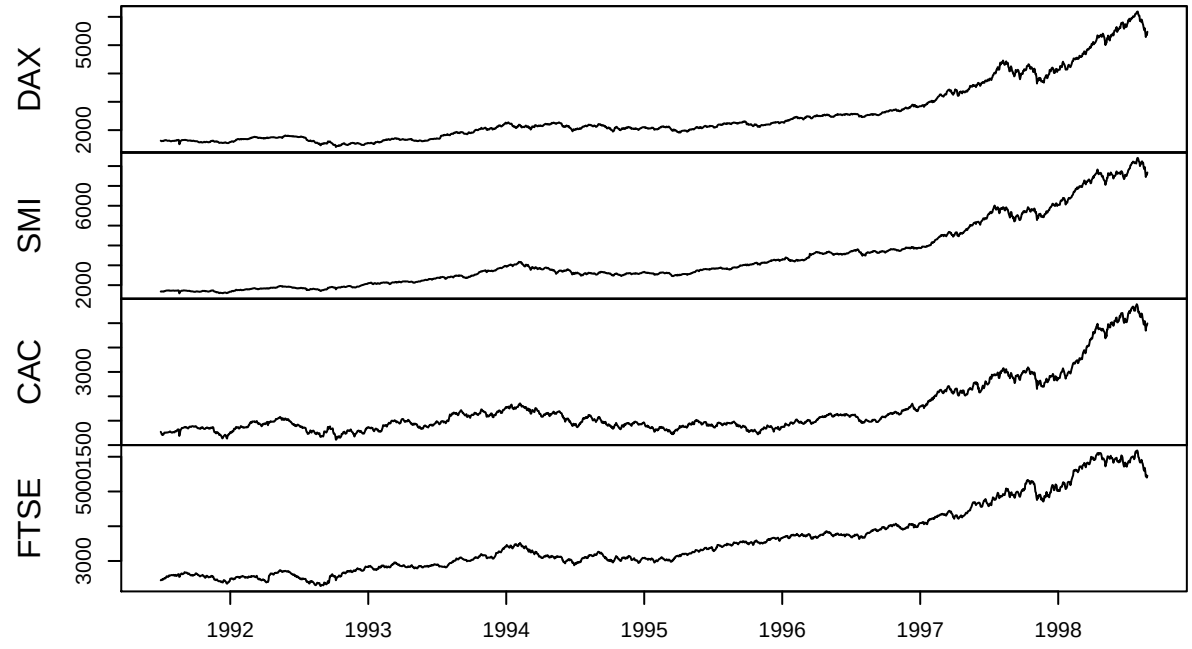
Let's load the `zoo` and `qrmdata` libraries first and look at the `EuroStoxx50` data set. Here we can imagine we are rebuilding our brand and footprint in the European Union and United Kingdom. Our customers might be the companies based in these countries as our target market.

- The data: 4 stock exchange indices across Europe (and the United Kingdom)
- This will allow us to profile the forward capabilities of these companies across their economies.
- Again we will look at returns data using the `diff(log(data))[-1]` formula.

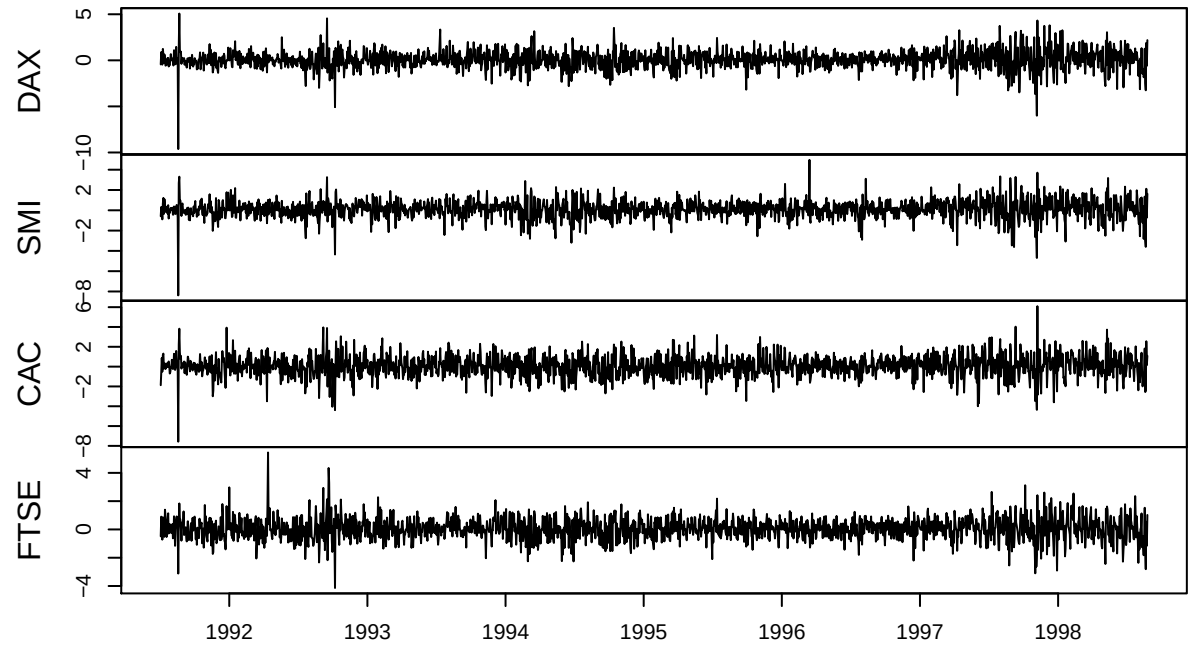
```
require(zoo)
require(qrmdata)
require(xts)
data("EuStockMarkets")
EuStockMarkets.price <- as.zoo(EuStockMarkets)
EuStockMarkets.return <- diff(log(EuStockMarkets.price))[-1] *
    100
```

We then plot price levels and returns.

```
plot(EuStockMarkets.price, xlab = " ",
    main = " ")
```



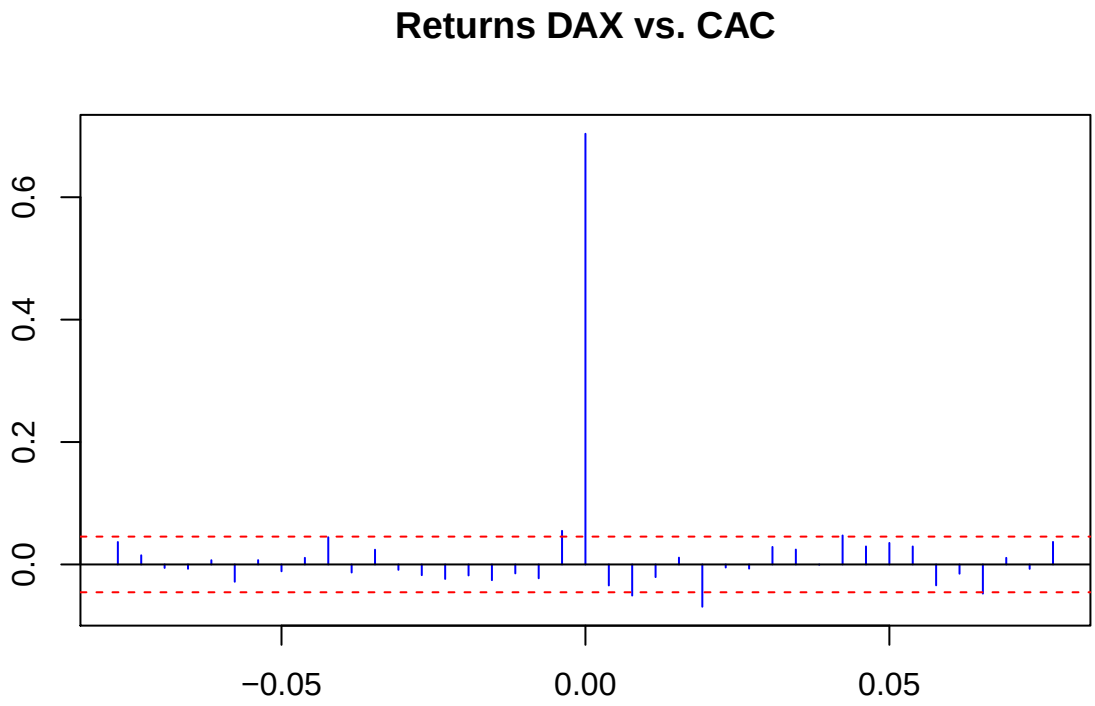

```
plot(EuStockMarkets.return, xlab = " ",
    main = " ")
```



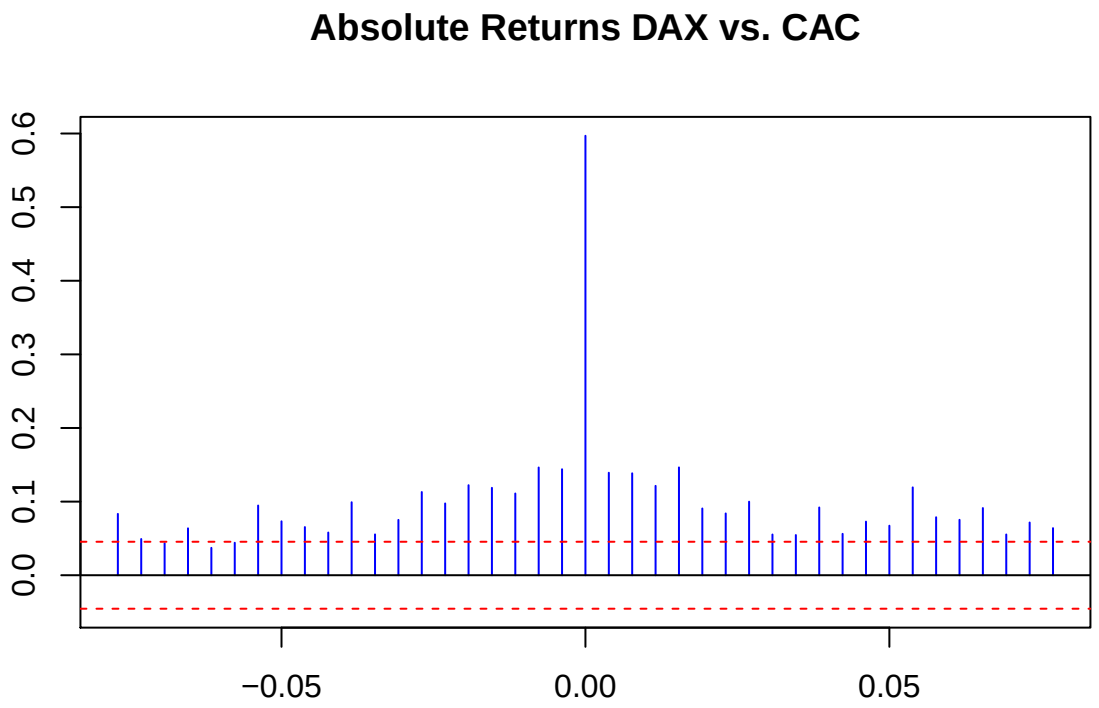
We see much the same thing as Brent oil with volatility clustering and heavily weighted tails.

Let's then look at cross-correlations among one pair of these indices to see how they are related across time (lags) for returns and the absolute value of returns. THe function `ccf` will aid us tremendously.

```
ccf(EuStockMarkets.return[, 1], EuStockMarkets.return[,
    2], main = "Returns DAX vs. CAC",
    lag.max = 20, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
```




```
ccf(abs(EuStockMarkets.return[, 1]),
    abs(EuStockMarkets.return[, 2]),
    main = "Absolute Returns DAX vs. CAC",
    lag.max = 20, ylab = "", xlab = "",
    col = "blue", ci.col = "red")
```



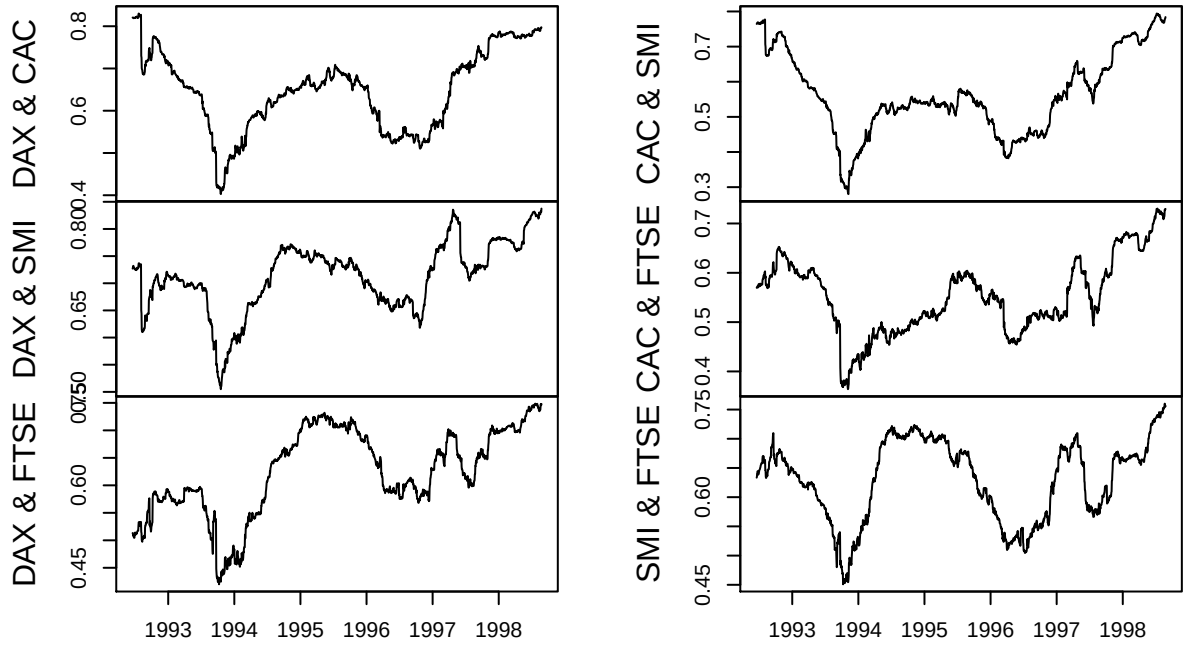
We see some small raw correlations across time with raw returns. More revealing, we see volatility of correlation clustering using return sizes. We can conduct one more experiment: a rolling correlation using this function:

```
corr.rolling <- function(x) {
    dim <- ncol(x)
    corr.r <- cor(x)[lower.tri(diag(dim),
        diag = FALSE)]
    return(corr.r)
}
```

We then embed our rolling correlation function, `corr.rolling`, into the function `rollapply` (look this one up using `??rollapply` at the console). The question we need to answer is: What is the history of correlations, and from the history, the pattern of correlations in the UK and EU stock markets? If there is a "history" with a "pattern," then we have to manage the risk that conducting business in one country will definitely affect business in another. The implication is that bad things will be followed by more bad things more often than good things. The implication compounds a similar implication across markets.

```
corr.returns <- rollapply(EuStockMarkets.return,
    width = 250, corr.rolling, align = "right",
    by.column = FALSE)
colnames(corr.returns) <- c("DAX & CAC",
```

```
    "DAX & SMI", "DAX & FTSE", "CAC & SMI",
    "CAC & FTSE", "SMI & FTSE")
plot(corr.returns, xlab = "", main = "")
```



Again we observe the volatility clustering from bunching up of the the absolute sizes of returns. Economic performance is certainly subject here to the same dynamics we saw for a single financial variable such as Brent.

Let's redo some of the work we just did using another set of techniques. This time we are using the "Fisher" transformation. Look up Fisher in Wikipedia and in your reference texts.

- How can the Fisher Transformation possibly help us answer our business questions?
- For three Spanish companies, Iberdrola, Endesa, and Repsol, replicate the Brent and EU stock market experiments above with absolute sizes and tails. Here we already have "series" covered.

First, the Fisher transformation is a smoothing routine that helps us tabilize the volitility of a variate. It does this by pulling some of the shockiness (i.e., outliers and aberrant noise) out of the original time series. In a phrase, it helps us see the forest (or the wood) for the trees.

We now replicate the Brent and EU stock exchange experiments. We again load some packages and get some data using `quantmod`'s `getSymbols` off the Madrid stock exchange to

match our initial working example of Iberian companies on account. Then compute returns and merge into a master file.

```
require(xts)
require(qrmdata)
require(quantreg)
require(quantmod)
require(matrixStats)

tickers <- c("ELE.MC", "IBE.MC", "REP.MC")
getSymbols(tickers)
```

```
## [1] "ELE.MC" "IBE.MC" "REP.MC"
```

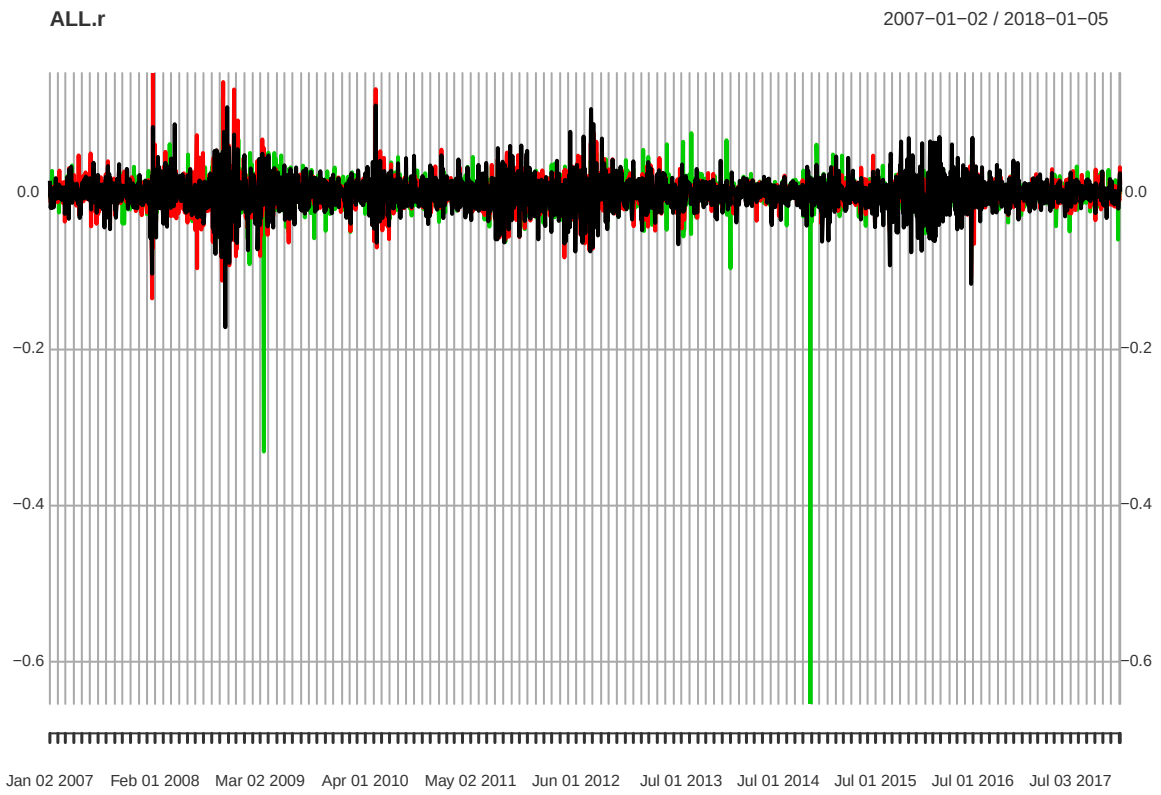
```
REP.r <- na.omit(diff(log(REP.MC[, 4]))[-1])
IBE.r <- na.omit(diff(log(IBE.MC[, 4]))[-1])
ELE.r <- na.omit(diff(log(ELE.MC[, 4]))[-1])  # clean up missing values

ALL.r <- na.omit(merge(REP = REP.r, IBE = IBE.r,
    ELE = ELE.r, all = FALSE))
```

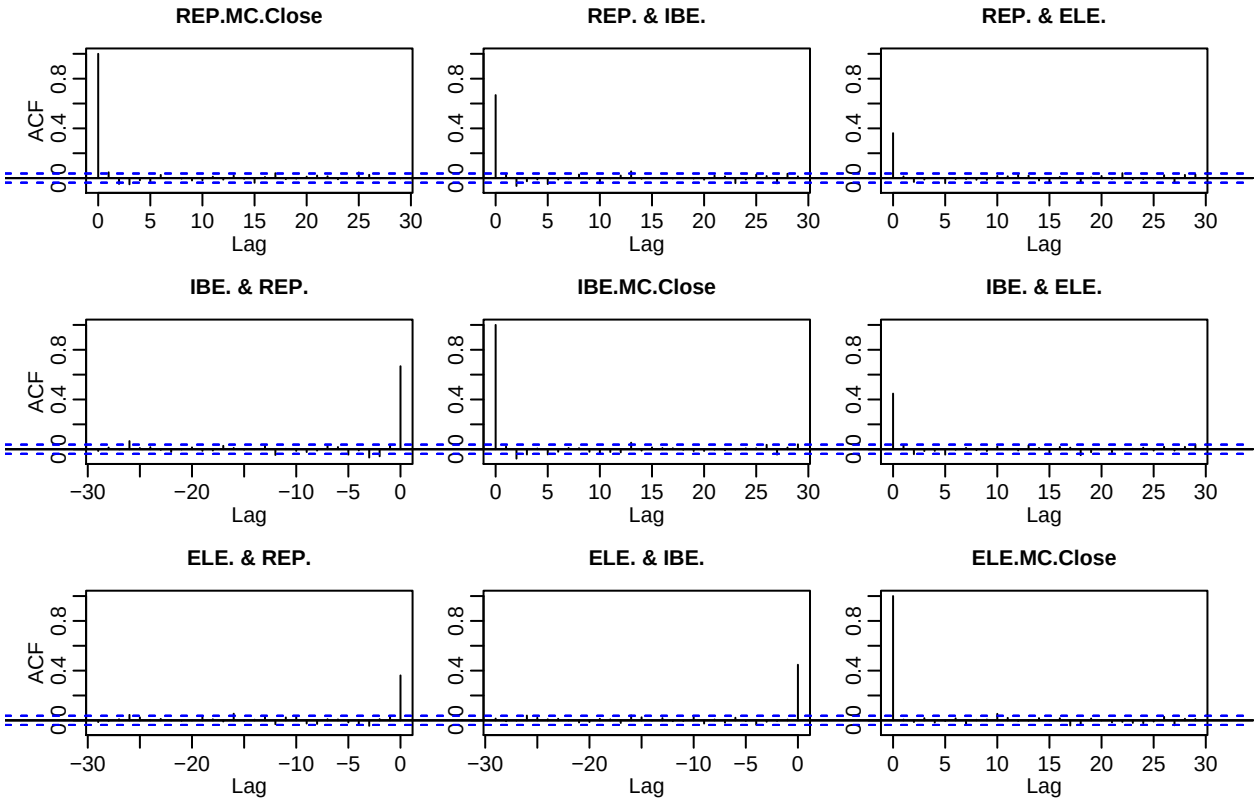
Next we plot the returns and their absolute values, acf and pacf, all like we did in Brent. Again we see

1. The persistence of returns
2. The importance of return size
3. Clustering of volatility

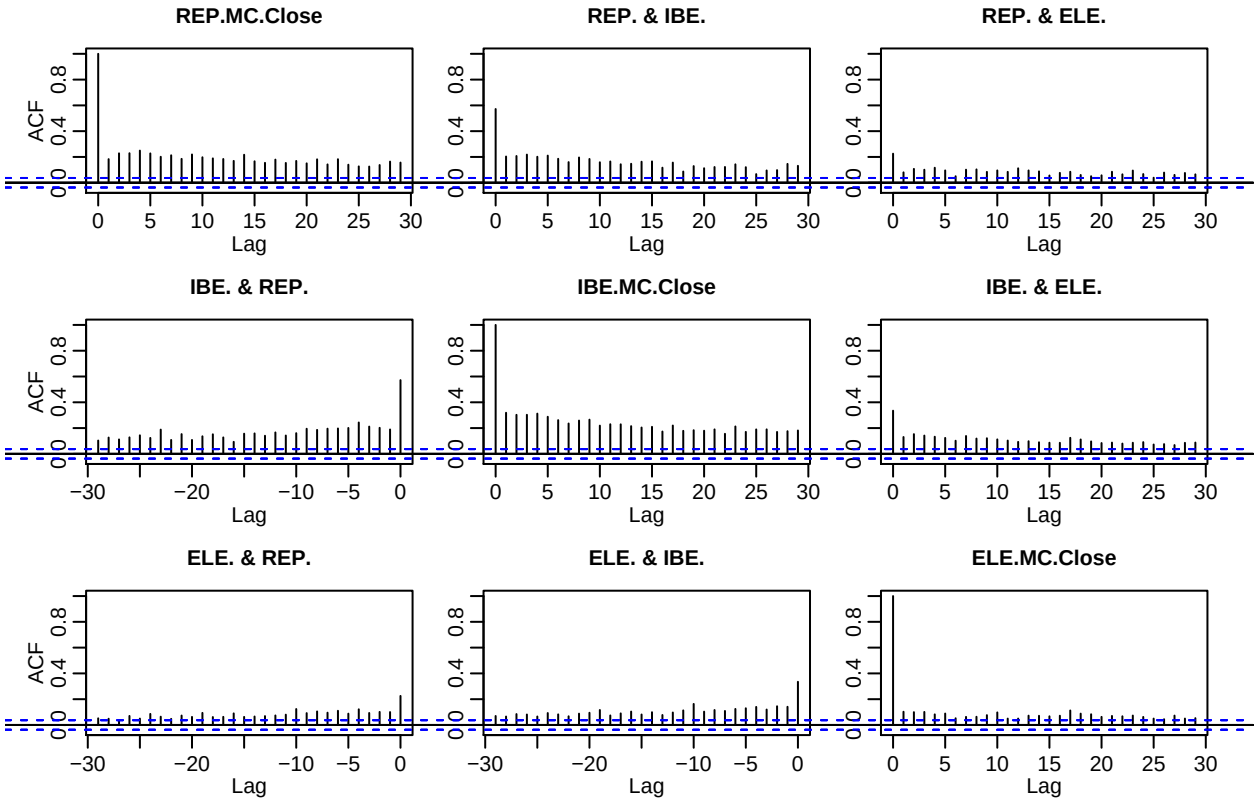
```
plot(ALL.r)
```



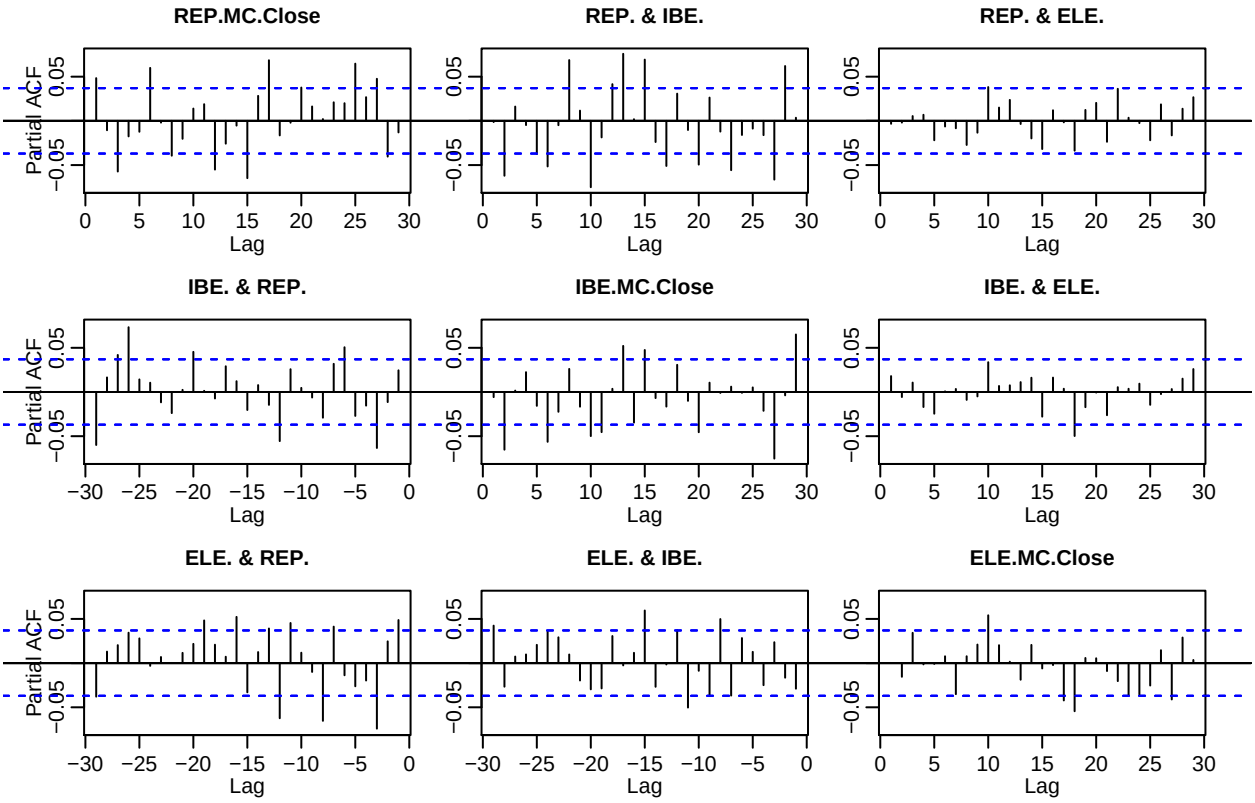

```
par(mfrow = c(2, 1))
acf(ALL.r)
```



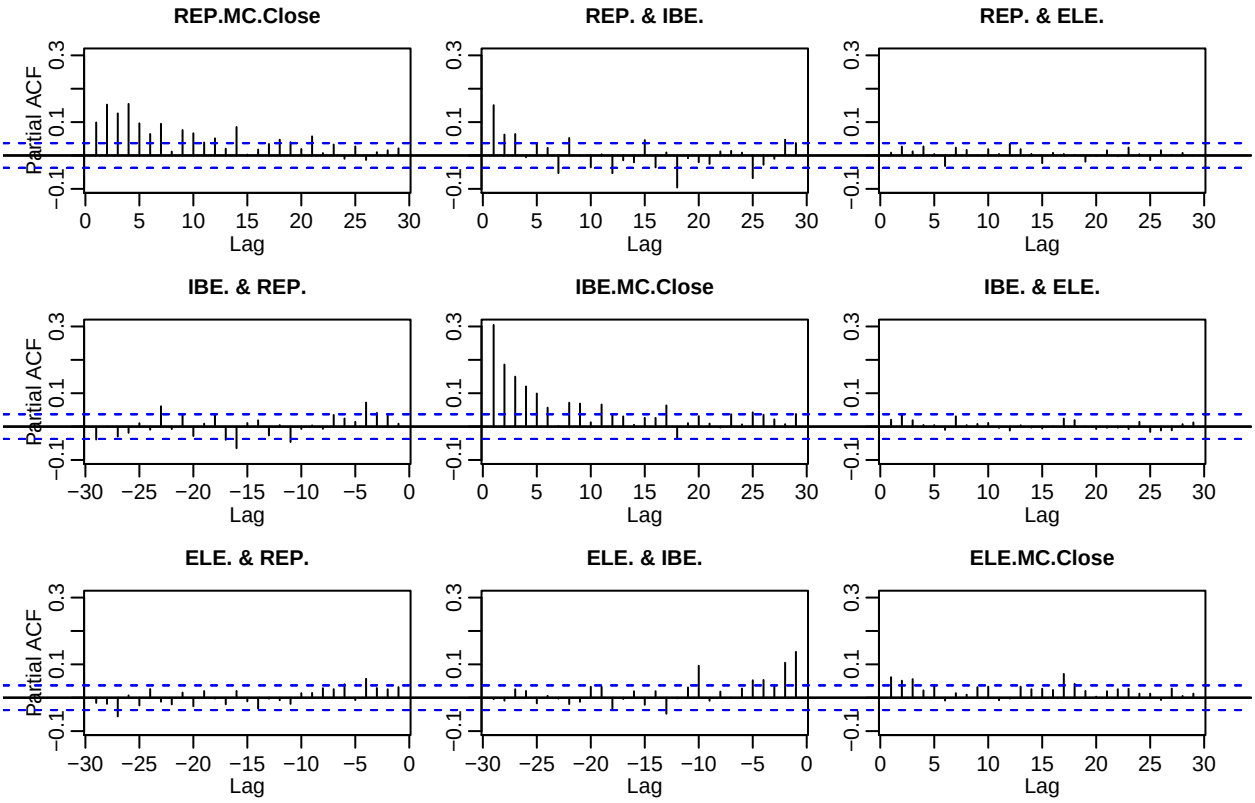

```
par(mfrow = c(2, 1))
acf(abs(ALL.r))
```




```
par(mfrow = c(2, 1))
pacf(ALL.r)
```




```
par(mfrow = c(2, 1))
pacf(abs(ALL.r))
```



Let's examine the correlation structure of markets where we can observe

1. The relationship between correlation and volatility
2. How quantile regression gets us to an understanding of high stress (high and low quantile) episodes

```
R.corr <- apply.monthly(ALL.r, FUN = cor)
R.vols <- apply.monthly(ALL.r, FUN = colSds)  ## from MatrixStats
head(R.corr, 3)
```

```
##            [,1]      [,2]        [,3]      [,4] [,5]       [,6]
## 2007-01-31    1 0.3613554 -0.27540757 0.3613554    1 0.10413800
## 2007-02-28    1 0.5661814 -0.09855544 0.5661814    1 0.10760477
## 2007-03-30    1 0.4500982 -0.08874664 0.4500982    1 0.08538064
##                  [,7]       [,8] [,9]
## 2007-01-31 -0.27540757 0.10413800    1
## 2007-02-28 -0.09855544 0.10760477    1
## 2007-03-30 -0.08874664 0.08538064    1
```

```
head(R.vols, 3)
```

```
##            REP.MC.Close IBE.MC.Close ELE.MC.Close
## 2007-01-31  0.009787963  0.007892759  0.009777426
## 2007-02-28  0.009181099  0.014571945  0.007674848
```

```
## 2007-03-30  0.015317331  0.012719792  0.010919155
```

```
R.corr.1 <- matrix(R.corr[1, ], nrow = 3,
    ncol = 3, byrow = FALSE)
rownames(R.corr.1) <- tickers
colnames(R.corr.1) <- tickers
head(R.corr.1)
```

```
##            ELE.MC    IBE.MC     REP.MC
## ELE.MC  1.0000000 0.3613554 -0.2754076
## IBE.MC  0.3613554 1.0000000  0.1041380
## REP.MC -0.2754076 0.1041380  1.0000000
```

```
R.corr <- R.corr[, c(2, 3, 6)]
colnames(R.corr) <- c("ELE.IBE", "ELE.REP",
    "IBE.REP")
colnames(R.vols) <- c("ELE.vols", "IBE.vols",
    "REP.vols")
head(R.corr, 3)
```

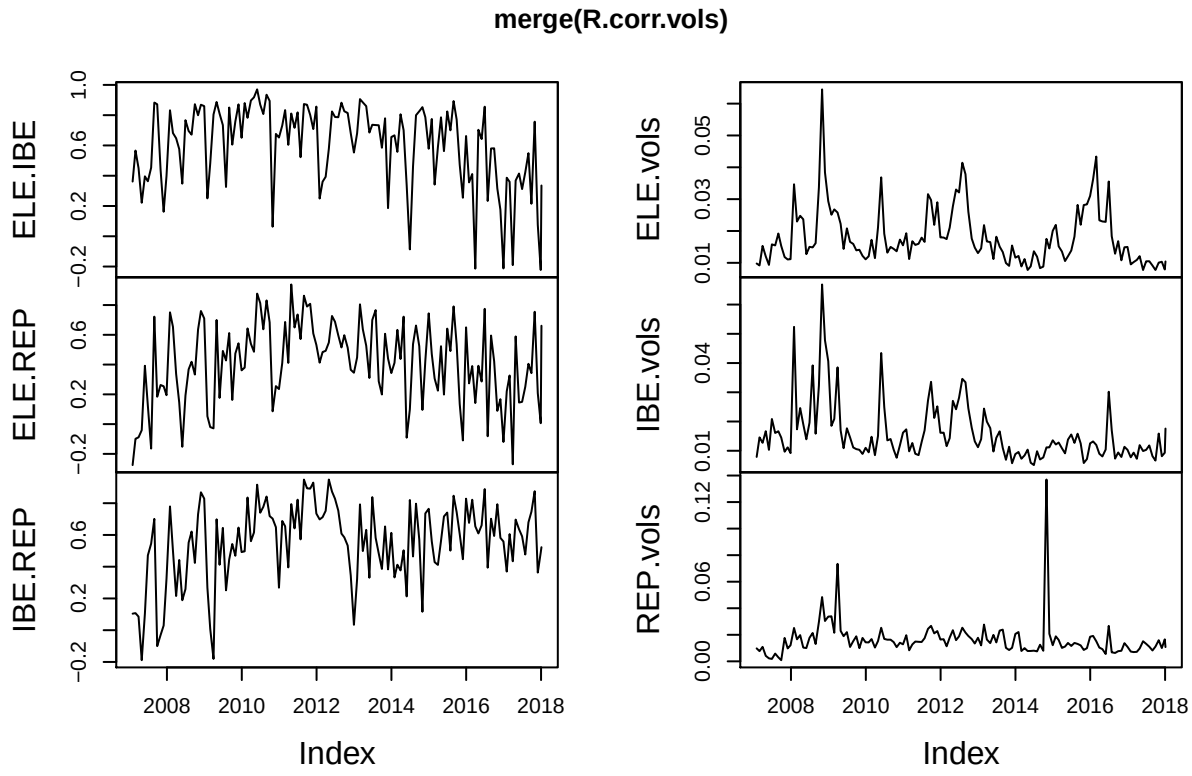
```
##              ELE.IBE     ELE.REP    IBE.REP
## 2007-01-31 0.3613554 -0.27540757 0.10413800
## 2007-02-28 0.5661814 -0.09855544 0.10760477
## 2007-03-30 0.4500982 -0.08874664 0.08538064
```

```
head(R.vols, 3)
```

```
##               ELE.vols    IBE.vols    REP.vols
## 2007-01-31 0.009787963 0.007892759 0.009777426
## 2007-02-28 0.009181099 0.014571945 0.007674848
## 2007-03-30 0.015317331 0.012719792 0.010919155
```

```
R.corr.vols <- merge(R.corr, R.vols)
```

```
plot.zoo(merge(R.corr.vols))
```




```
ELE.vols <- as.numeric(R.corr.vols[,
    "ELE.vols"])
IBE.vols <- as.numeric(R.vols[, "IBE.vols"])
REP.vols <- as.numeric(R.vols[, "REP.vols"])
length(ELE.vols)
```

```
## [1] 133
```

```
fisher <- function(r) {
    0.5 * log((1 + r)/(1 - r))
}
rho.fisher <- matrix(fisher(as.numeric(R.corr.vols[,
    1:3])), nrow = length(ELE.vols),
    ncol = 3, byrow = FALSE)
```

4.3.1 On to quantiles

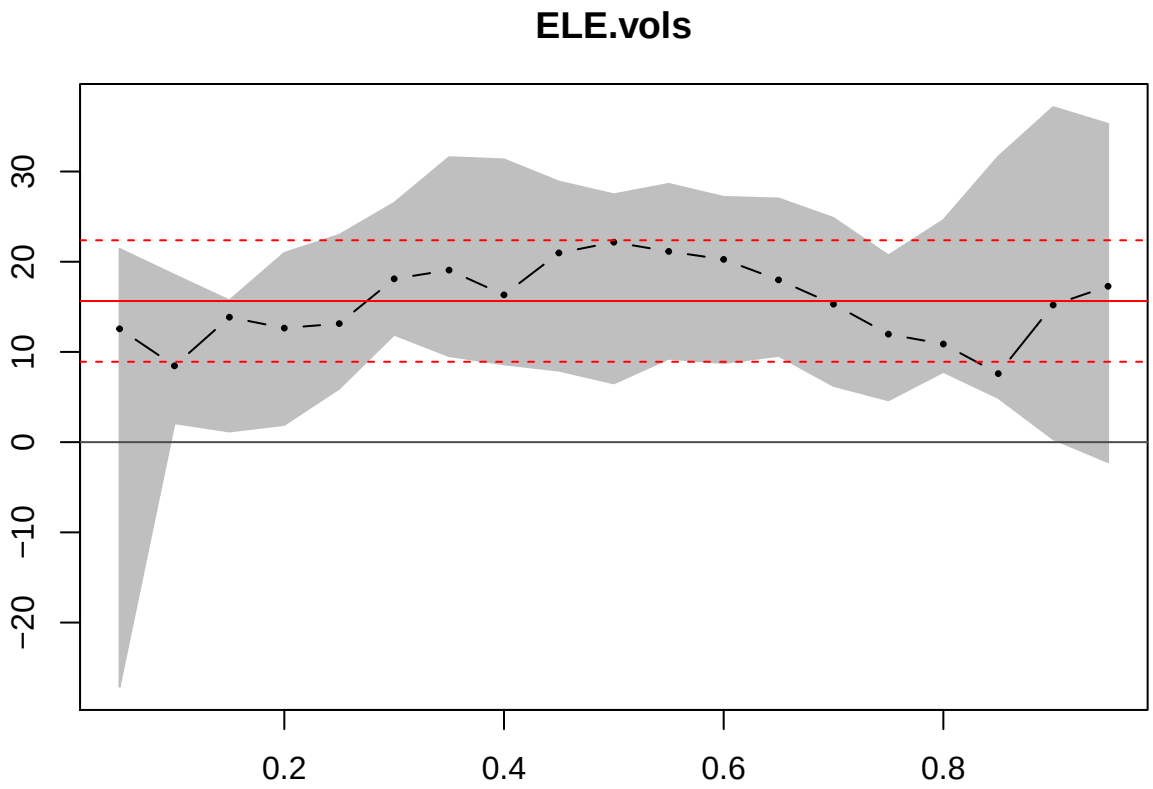
Here is the quantile regression part of the package. Quantile regression finds the average relationship between dependent and independent variables just like ordinary least squares with one exception. Instead of centering the regression on the arithmetic mean of the dependent variable, quantile regression centers the regression on a specified quantile of the dependent

variable. So instead of using the arithemetic average of the rolling correlations, we now use the 10th quantile, or the median, which is the 50th quantile as our reference. This makes great intuitive sense since we have already established that the series we deal with here are thick tailed, skewed, and certainly not normally distributed.

Here is how we use the `quantreg` package.

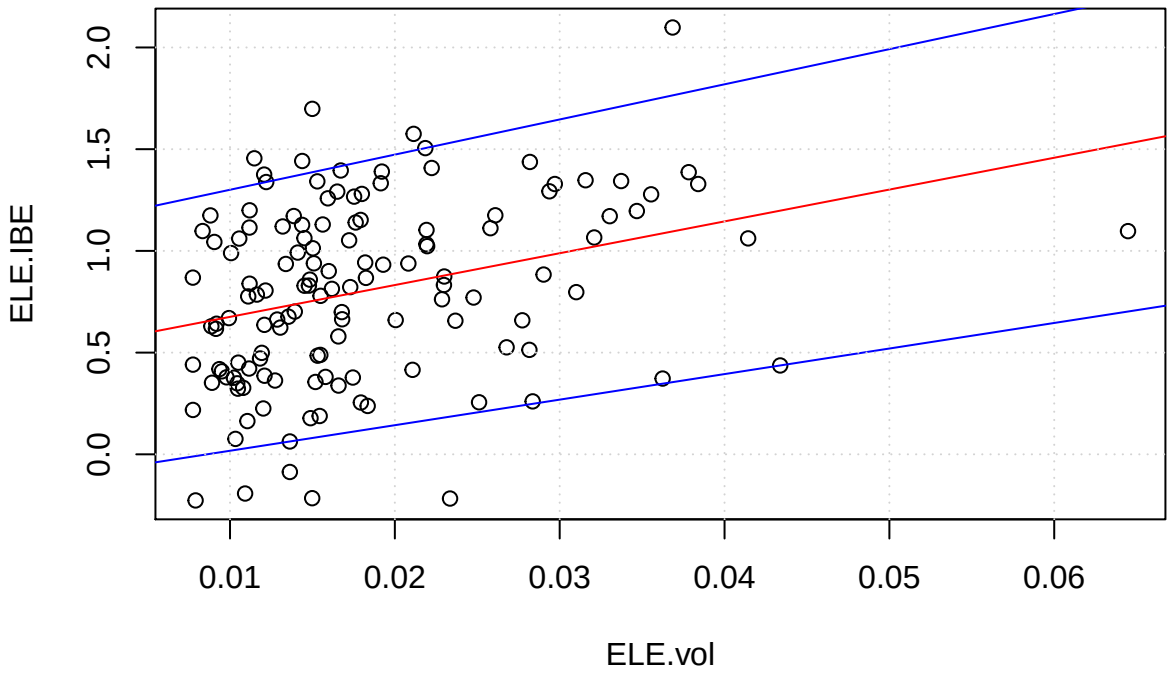
1. We set `taus` as the quantiles of interest.
2. We run the quantile regression using the `quantreg` package and a call to the `rq` function.
3. We can overlay the quantile regression results onto the standard linear model regression.
4. We can sensitize our analysis with the range of upper and lower bounds on the parameter estimates of the relationship between correlation and volatility. This sensitivity analysis is really a confidence interval based on quantile regressions.

```
require(quantreg)
taus <- seq(0.05, 0.95, 0.05)
fit.rq.ELE.IBE <- rq(rho.fisher[, 1] ~
    ELE.vols, tau = taus)
fit.lm.ELE.IBE <- lm(rho.fisher[, 1] ~
    ELE.vols)
plot(summary(fit.rq.ELE.IBE), parm = "ELE.vols")
```



Here we build the estimations and plot the upper and lower bounds.

```
taus1 <- c(0.05, 0.95)  ## fit the confidence interval (CI)
plot(ELE.vols, rho.fisher[, 1], xlab = "ELE.vol",
    ylab = "ELE.IBE")
abline(fit.lm.ELE.IBE, col = "red")
for (i in 1:length(taus1)) {
    ## these lines will be the CI
    abline(rq(rho.fisher[, 1] ~ ELE.vols,
        tau = taus1[i]), col = "blue")
}
grid()
```



Quantile regression helps us to see the upper and lower bounds. Relationships between high-stress periods and correlation are abundant. These markets simply reflect normal buying behaviors across many types of exchanges: buying food at Safeway or Whole Foods, buying collateral to insure a project, selling off illiquid assets.

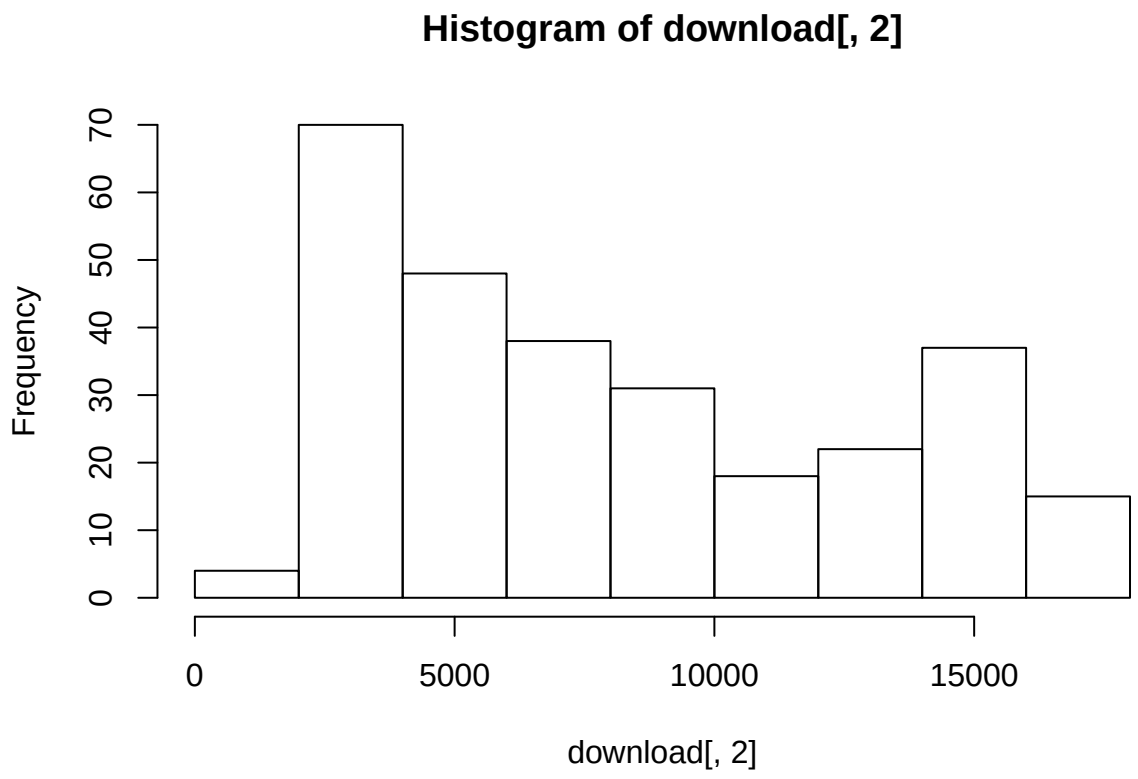
4.4 Time is on our Side

Off to another important variable, the level and growth rate of Gross National Product. Let's start with some US Gross National Product (GNP) data from the St. Louis Fed's open data website ("FRED"). We access https://fred.stlouisfed.org/series/GNPC96 to download real GNP in chained 1996 dollars. Saving this as a CSV file wwe then read the saved file into our `R` workspace.

```
name <- "GNP"
download <- read.csv("data/GNPC96.csv")
```

Look at the data:

```
hist(download[, 2])
```




```
summary(download[, 2])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1943    3808    6659    8035   12385   17353
```

We then create a raw time series object using the `ts` function where rownames are dates, select some data, and calculate growth rates. This will allow us and plotting functions to use the dates to index the data. Again we make use of the `diff(log(data))` vector calculation.

```
GNP <- ts(download[1:85, 2], start = c(1995,
    1), freq = 4)
GNP.rate <- 100 * diff(log(GNP))  # In percentage terms
str(GNP)
```

```
##  Time-Series [1:85] from 1995 to 2016: 1947 1945 1943 1974 2004 ...
```

```
head(GNP)
```

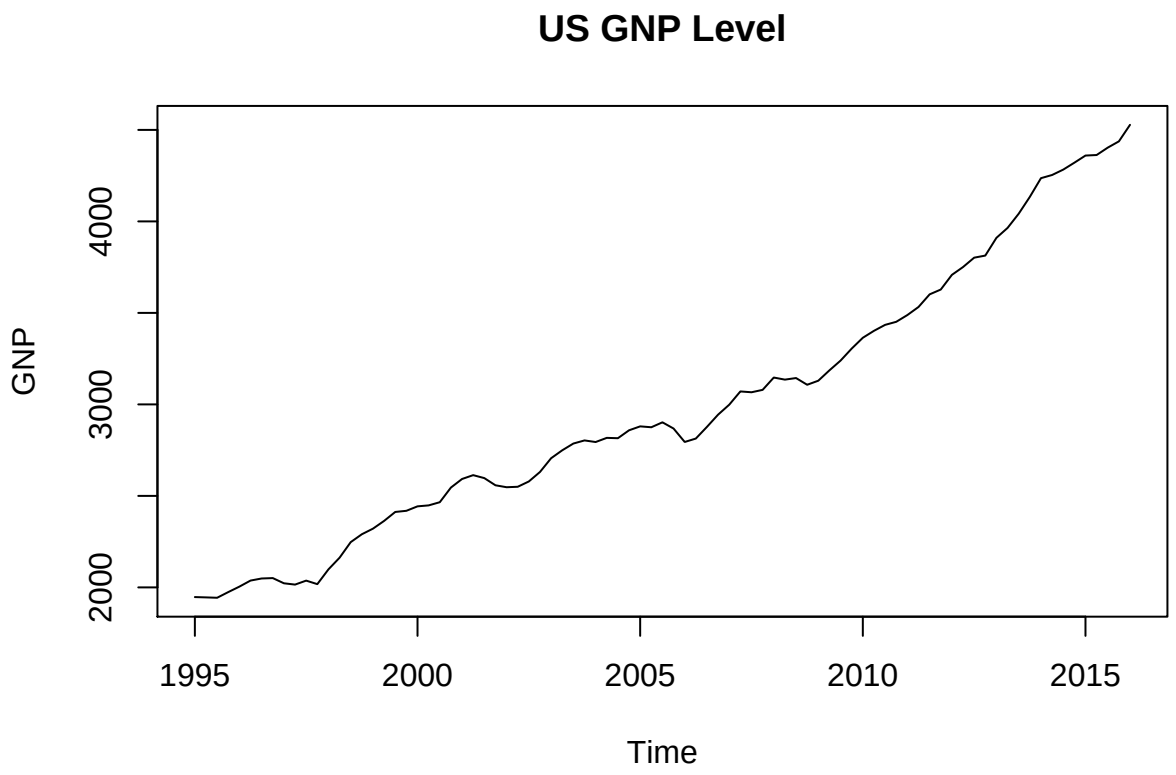
```
## [1] 1947.003 1945.311 1943.290 1974.312 2004.218 2037.215
```

```
head(GNP.rate)
```

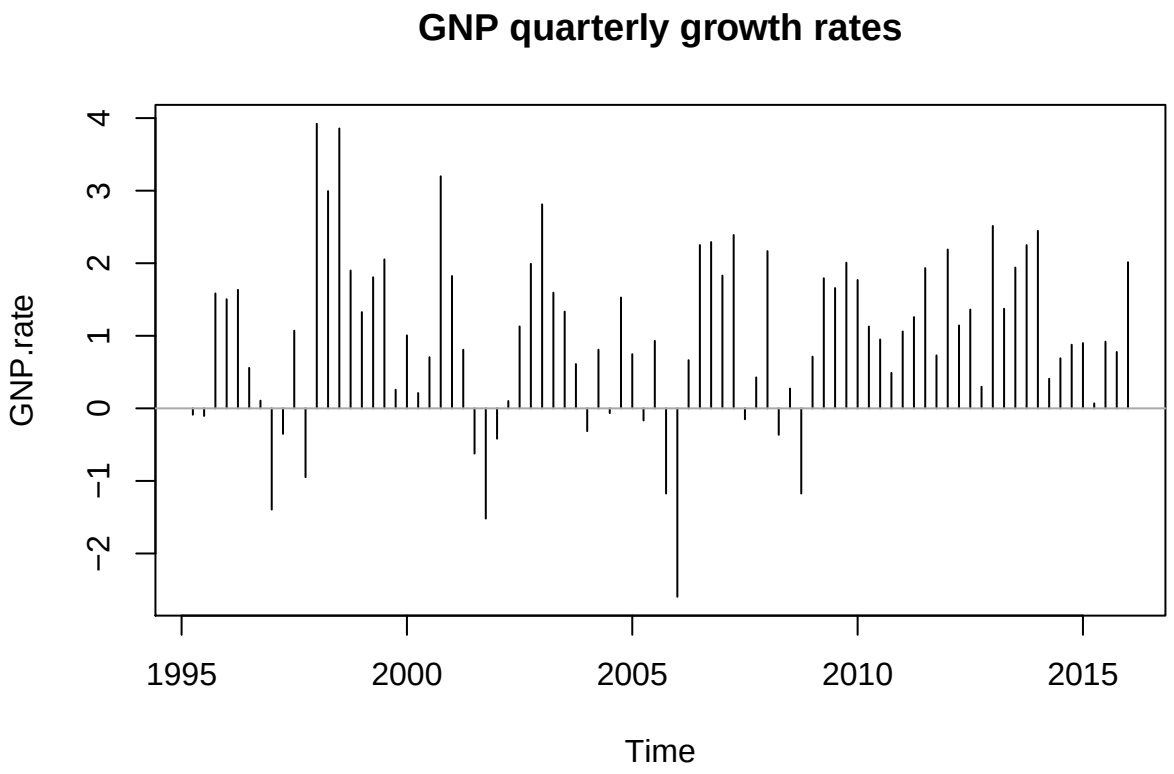
```
## [1] -0.08694058 -0.10394485  1.58375702  1.50339765  1.63297193  0.55749065
```

Let's plot the GNP level and rate and comment on the patterns.

```
plot(GNP, type = "l", main = "US GNP Level")
```




```
plot(GNP.rate, type = "h", main = "GNP quarterly growth rates")
abline(h = 0, col = "darkgray")
```

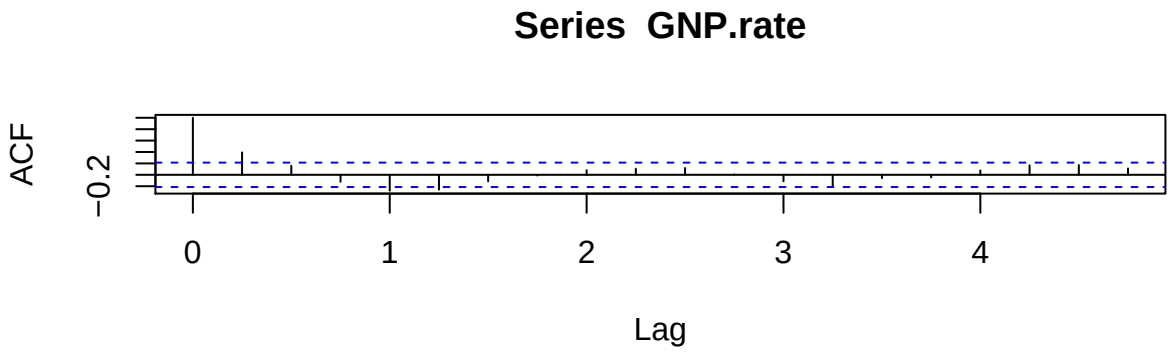


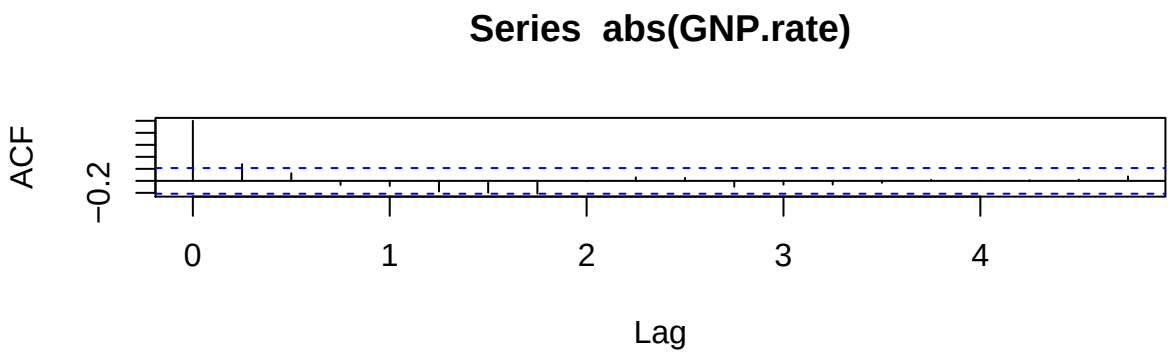
We see a phenomenon called "nonstationarity." The probability distribution (think `hist()`) would seem to change over time (many versions of a `hist()`). This means that the standard deviation and mean change as well (and higher moments such as skewness and kurtosis). There is trend in the level and simply dampened sinusoidal in the rate. In a nutshell we observe several distributions mixed together in this series. This will occur again in the term structure of interest rates where we will use splines and their knots to get at parameterizing the various distributions lurking just beneath the ebb and flow of the data.

4.4.1 Forecasting GNP

As always let's look at ACF and PACF:

```
par(mfrow = c(2, 1))  ##stacked up and down
acf(GNP.rate)
acf(abs(GNP.rate))
```






```
par(mfrow = c(1, 1))  ##default setting
```

What do we think is going on? There are several significant autocorrelations within the last 4 quarters. Partial autocorrelation also indicates some possible relationship 8 quarters back.

Let's use `R`'s time series estimation and prediction tool `arima`. In this world we think there is a regression that looks like this:

$$x_t = a_0 + a_1 x_{t-1} ... a_p x_{t-p} + b_1 \varepsilon_{t-1} + ... + b_q \varepsilon_{t-q}$$

where x_t is a first, $d = 1$, differenced level of a variable, here GNP. There are p lags of the rate itself and q lags of residuals. We officially call this an *Autoregressive Integrated Moving Average* process of order (p, d, q), or `ARIMA(p,d,q)` for short.

Estimation is quick and easy.

```
fit.rate <- arima(GNP.rate, order = c(2,
    1, 1))
```

The order is 2 lags of rates, 1 further difference (already differenced once when we calculated `diff(log(GNP))`), and 1 lag of residuals. Let's diagnose the results with `tsdiag()`. What are the results?

```
fit.rate
```

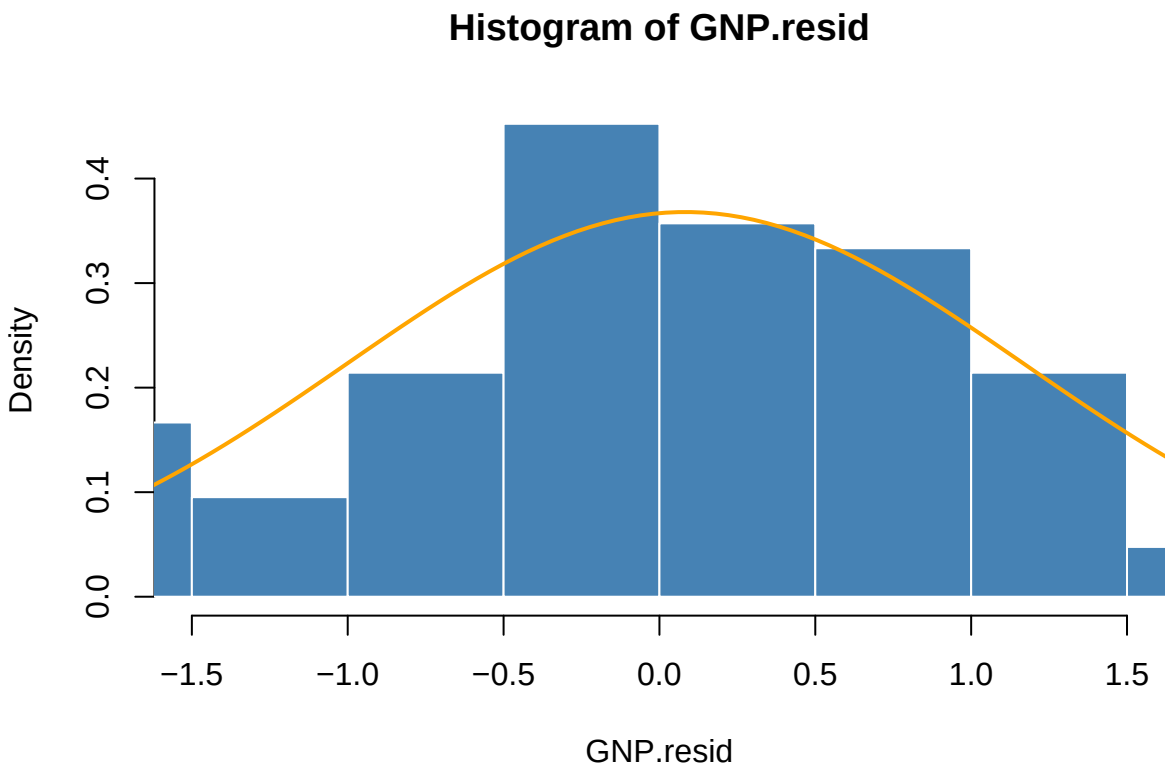
```
##
## Call:
## arima(x = GNP.rate, order = c(2, 1, 1))
##
## Coefficients:
##          ar1     ar2     ma1
##       0.4062  0.0170  -1.000
## s.e.  0.1106  0.1107   0.038
##
## sigma^2 estimated as 1.182:  log likelihood = -126.49,  aic = 260.98
```

Let's take out the moving average term and compare:

```
fit.rate.2 <- arima(GNP.rate, order = c(2,
    0, 0))
fit.rate.2
```

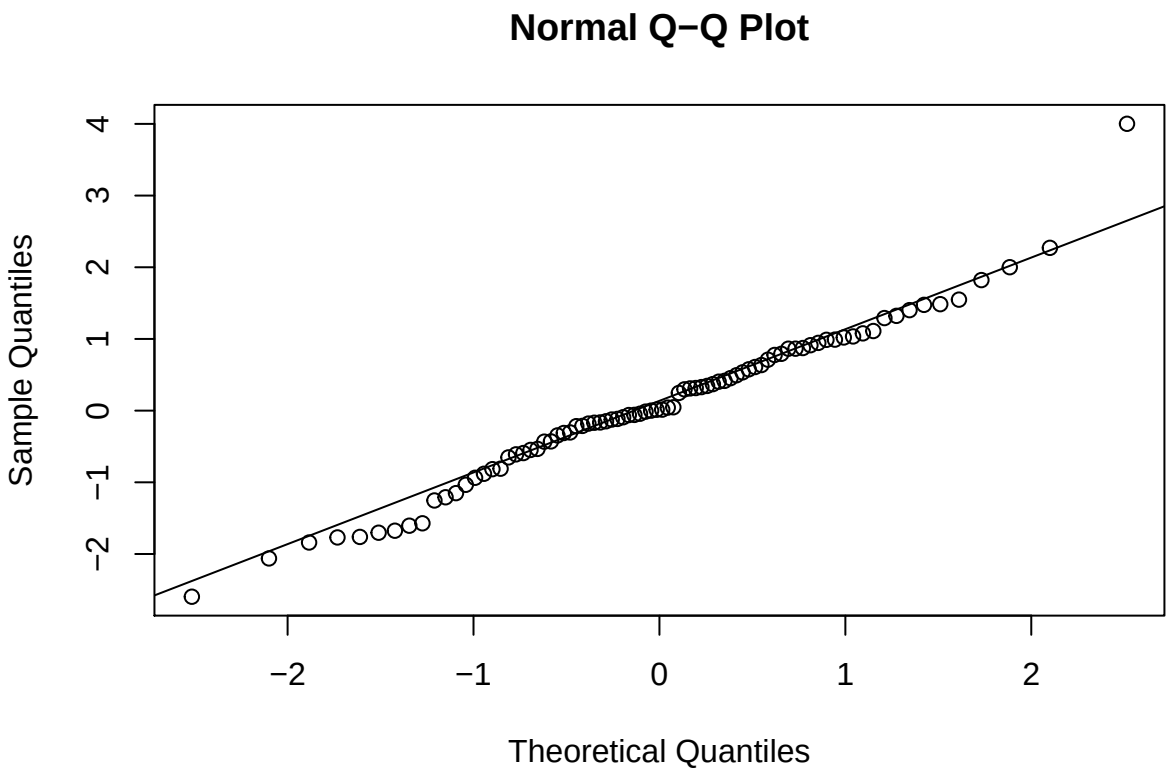
```
##
## Call:
## arima(x = GNP.rate, order = c(2, 0, 0))
##
## Coefficients:
##          ar1     ar2  intercept
##       0.3939  0.0049     1.0039
## s.e.  0.1092  0.1093     0.1946
##
## sigma^2 estimated as 1.168:  log likelihood = -125.8,  aic = 259.59
```

We examine the residuals next.



The `qqnorm` function plots actual quantiles against theoretical normal distributions of the quantiles. A line through the scatterplot will reveal deviations of actual quantiles from the normal ones. Those deviations are the key to understanding tail behavior, and thus the potential influence of outliers, on our understanding of the data.

```
qqnorm(GNP.resid)
qqline(GNP.resid)
```



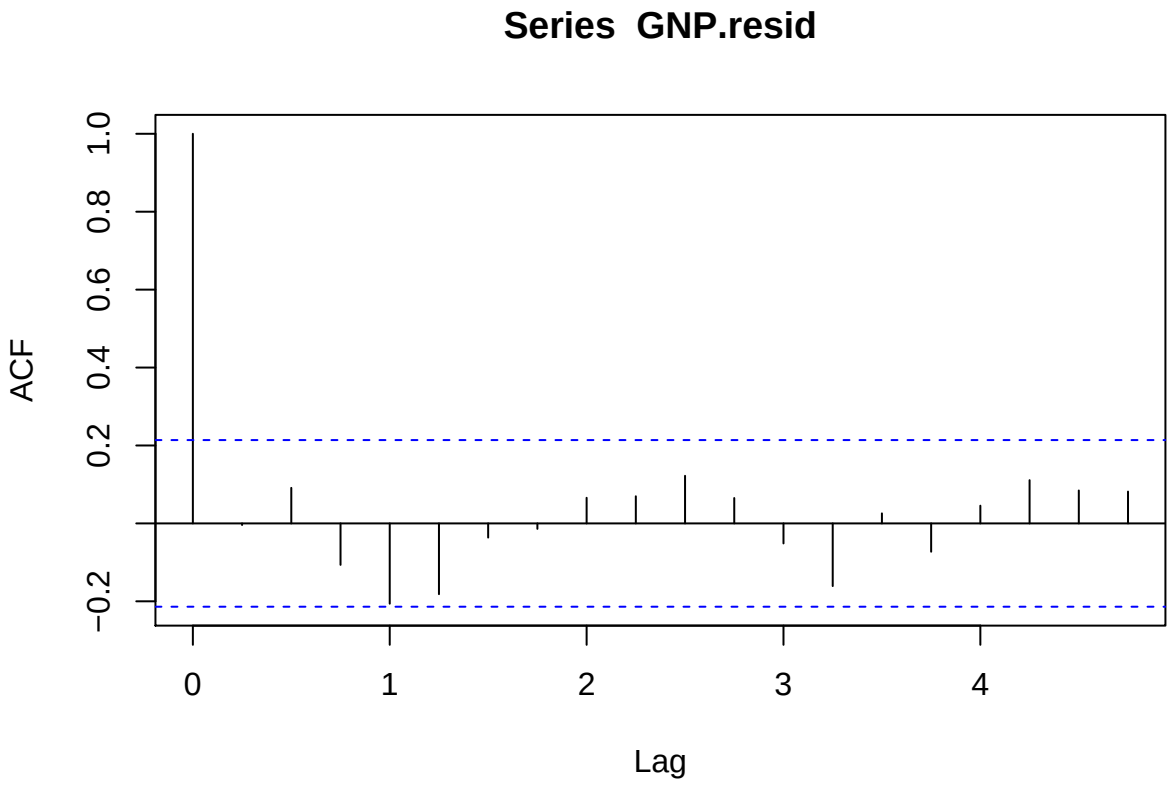
Some ways to interpret the qq-chart include

1. The diagonal line is the normal distribution quantile line.
2. Deviations of actual quantiles from the normal quantile line mean nonnormal.
3. Especially deviations at either (or both) end of the line spell thick tails and lots more "shape" than the normal distribution allows.

4.4.2 Residuals again

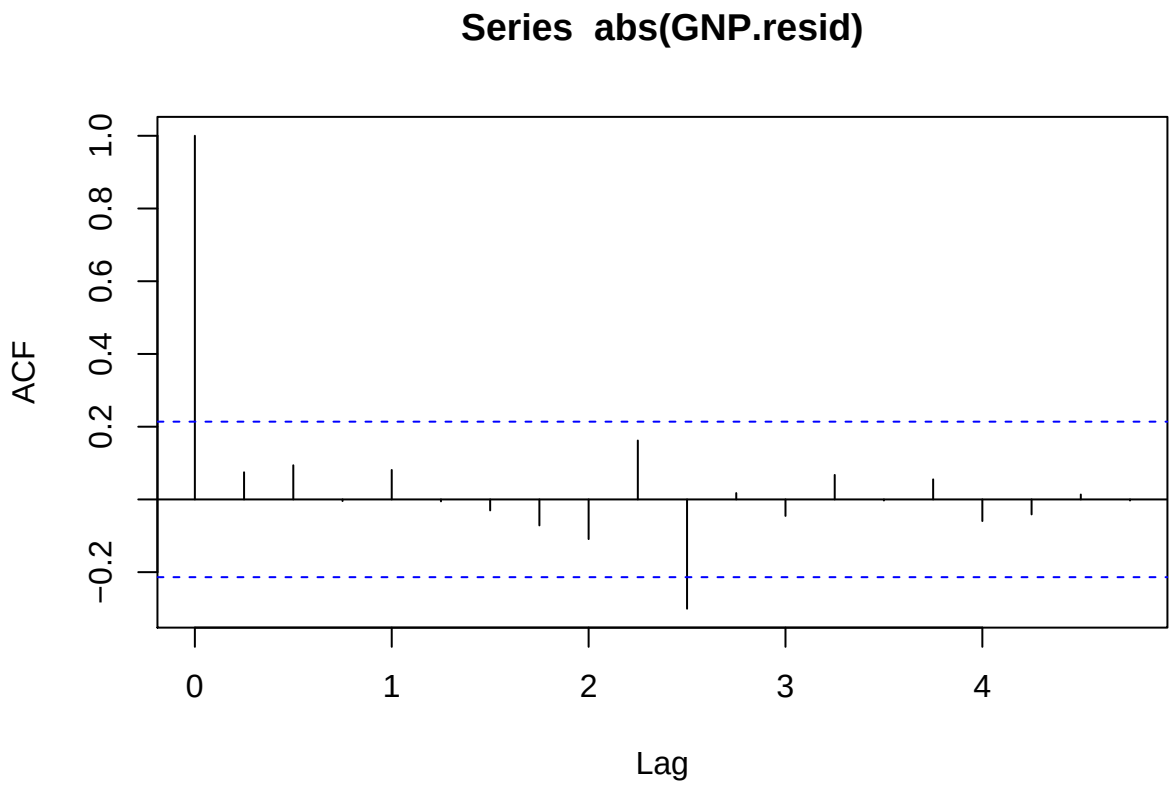
How can we begin to diagnose the GNP residuals? Let's use the ACF and the `moments` package to calculate `skewness` and `kurtosis`. We find that the series is very thick tailed and serially correlated as evidenced by the usual statistical suspects. But no volatility clustering.

```
acf(GNP.resid)
```



Now let's look at the absolute values of growth (i.e., GNP growth sizes). This will help us understand the time series aspects of the volatility of the GNP residuals.

```
acf(abs(GNP.resid))
```



...and compute tail statistics.

```
require(moments)
skewness(GNP.resid)
```

```
## [1] 0.2283692
## attr(,"method")
## [1] "moment"
```

```
kurtosis(GNP.resid)
```

```
## [1] 1.037626
## attr(,"method")
## [1] "excess"
```

The residuals are positively skewed and not so thick tailed, as the normal distribution has by definition a kurtosis equal to `3.00`. By the by: Where's the forecast?

```
(GNP.pred <- predict(fit.rate, n.ahead = 8))
```

```
## $pred
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2016          1.410045 1.185815 1.084467
## 2017 1.039485 1.019489 1.010602 1.006652
```

```
## 2018 1.004896
##
## $se
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2016          1.093704 1.185259 1.204803
## 2017 1.209512 1.210836 1.211272 1.211437
## 2018 1.211504
```

Now for something really interesting, yet another rendering of the notorious Efficient Markets Hypothesis.

4.5 Give it the Boot

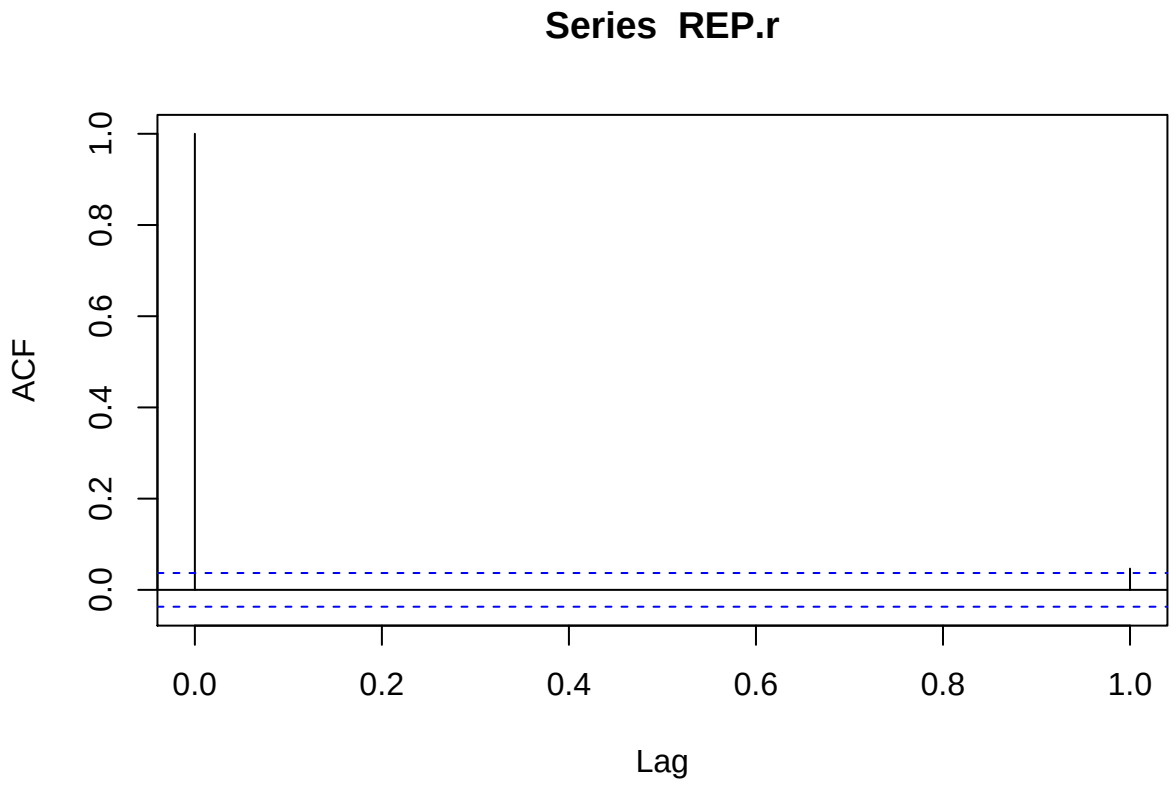
Our goal is to infer the significance of a statistical relationship among variates. However, we do not have access to, or a "formula" does not exist, that allows us to compute the sample standard deviation of the mean estimator.

- The context is just how dependent is today's stock return on yesterday's?
- We want to use the distribution of real-world returns data, without needing assumptions about normality.
- The null hypothesis H_0 is lack of dependence (i.e., an efficient market). The alternative hypothesis H_1 is that today's returns depend on past returns, on average.

Our strategy is to change the data repeatedly, and re-estimate a relationship. The data is sampled using the `replicate` function, and the sample ACF is computed. This gives us the distribution of the coefficient of the ACF under the null hypotheses, $H0$: independence, while using the empirical distribution of the returns data.

Let's use the Repsol returns and pull the 1st autocorrelation from the sample with this simple code,

```
acf(REP.r, 1)
```



There is not much for us to see, barely a blip, but there is a correlation over the 95% line. Let's further test this idea.

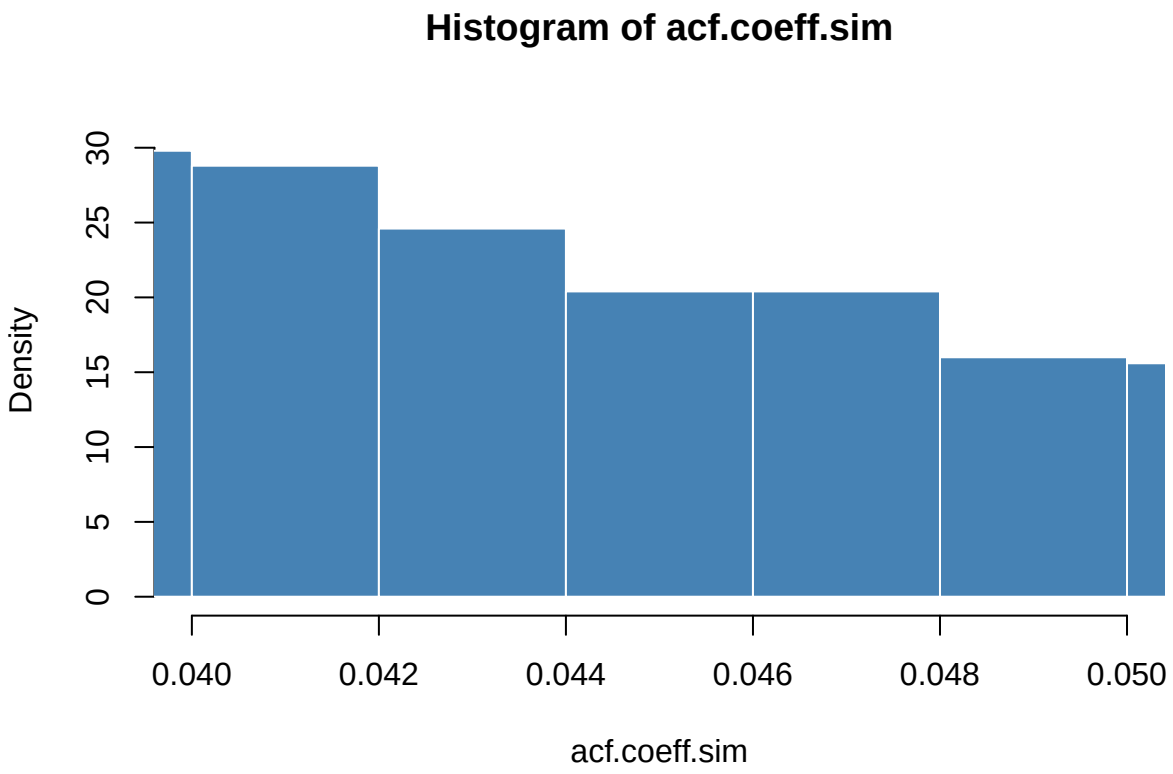
- We obtain 2500 draws from the distribution of the first autocorrelation using the `replicate` function.
- We operate under the null hypothesis of independence, assuming rational markets (i.e, rational markets is a "maintained hypothesis").

```
set.seed(1016)
acf.coeff.sim <- replicate(2500, acf(sample(REP.r,
    size = 2500, replace = FALSE), lag = 2,
    plot = FALSE)$acf[2])
summary(acf.coeff.sim)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.02038   0.02689   0.03653   0.03630   0.04568   0.08768
```

Here is a plot of the distribution of the sample means of the one lag correlation between successive returns.

```
hist(acf.coeff.sim, probability = TRUE,
    breaks = "FD", xlim = c(0.04, 0.05),
    col = "steelblue", border = "white")
```



4.5.1 Try this exercise

We will nvestigate tolerances of 5% and 1% from both ends of the distribution of the 1-lag acf coefficient using these statements. That was a mouthful! When we think of inference, we first identify a parameter of interest, and its estimator. That parameter is the coefficient of correlation between the current return and its 1-period lag. We estimate this parameter using the history of returns. If the parameter is significantly, and probably, not equal to zero, then we would have reason to believe there is "pattern" in the "history."

```
## At 95% tolerance level
quantile(acf.coeff.sim, probs = c(0.025,
    0.975))
```

```
##        2.5%       97.5%
## 0.006454092 0.064821915
```

```
## At 99% tolerance level
quantile(acf.coeff.sim, probs = c(0.005,
    0.995))
```

```
##         0.5%        99.5%
## -0.004856163  0.074229547
```

```
## And the
(t.sim <- mean(acf.coeff.sim)/sd(acf.coeff.sim))
```

```
## [1] 2.464299
```

```
(1 - pt(t.sim, df = 2))
```

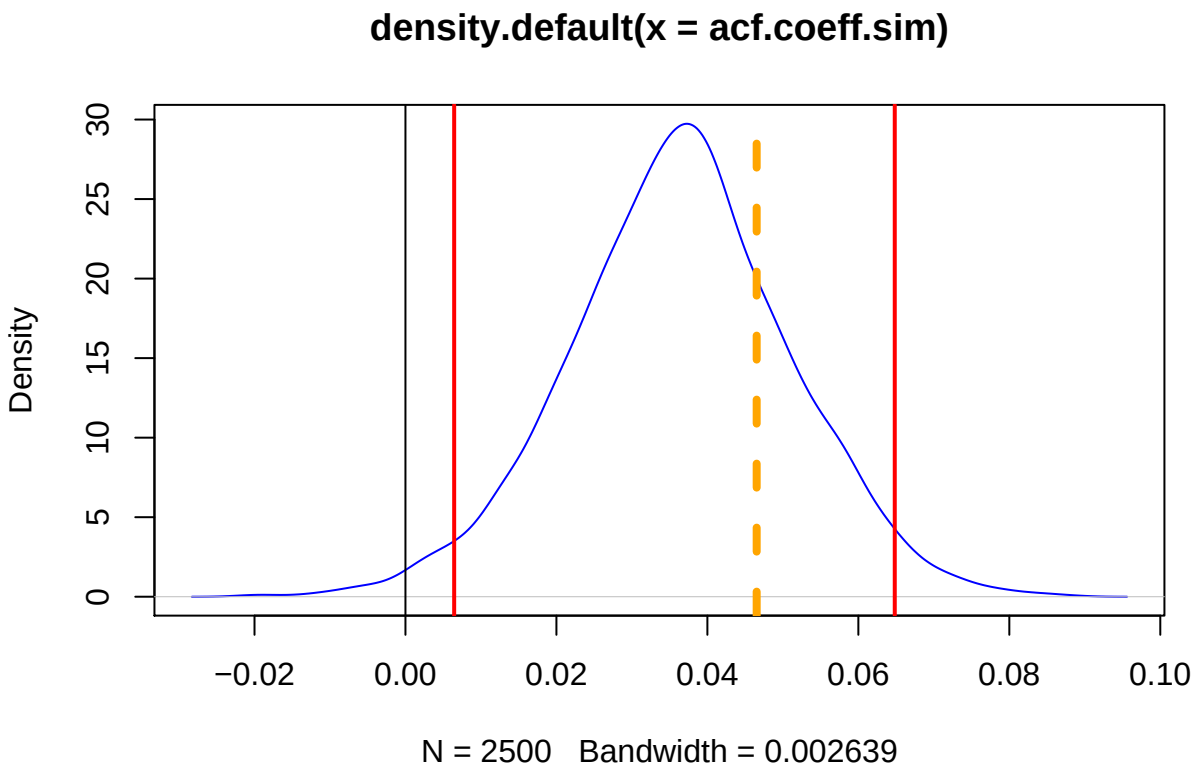
```
## [1] 0.06633726
```

Here are some highly preliminary and provisional answers to ponder.

1. Quantile values are very narrow…
2. How narrow (feeling like rejecting the null hypothesis)?
3. Thet-stat is huge, but…
4. …no buts!, the probability that we would be wrong to reject the null hypothesis is very small.

Here we plot the simulated density and lower and upper quantiles, along with the estimate of the lag-1 coefficient:

```
plot(density(acf.coeff.sim), col = "blue")
abline(v = 0)
abline(v = quantile(acf.coeff.sim, probs = c(0.025,
    0.975)), lwd = 2, col = "red")
abline(v = acf(REP.r, 1, plot = FALSE)$acf[2],
    lty = 2, lwd = 4, col = "orange")
```



Can we reject the null hypothesis that the coefficient = 0? Is the market "efficient"?

1. Reject the null hypothesis since there is a less than 0.02% chance that the coefficient is zero.
2. Read [Fama(2013, p. 365-367)]https://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2013/fama-lecture.pdf for a diagnosis.
3. If the model is correct (ACF lag-1) then the previous day's return can predict today's return according to our analysis. Thus the market would seem to be inefficient.
4. This means we might be able to create a profitable trading strategy that makes use of the little bit of correlation we found to be significant (net of the costs of trading).

4.6 Summary

We explored time series data using ACF, PACF, and CCF. We showed how to pull data from Yahoo! and FRED. We characterized several stylized facts of financial returns and inferred behavior using a rolling correlation regression on volatility. We then supplemented the ordinary least square regression confidence intervals using the entire distribution of the data with quantile regression. We also built Using bootstrapping techniques we simulated coefficient inference to check the efficient markets hypothesis. This, along with the quantile regression technique, allows us to examine risk tolerance from an inference point of view.

4.7 Further Reading

In this chapter we touch on the voluminous topic of time series analysis. Ruppert et al. in chapters 12, 13, 14, and 15 explore the basics, as in this chapter, as well as far more advanced topics such as GARCH and cointegration. We will explore GARCH in a later chapter as well. McNeil et al. in their chapter 1 surveys the perspective of risk, all of helps to yield the so-called stylized fact of financial data in chapter 5 and a more formal treatment of time series topics in chapter 4.

4.8 Practice Laboratory

4.8.1 Practice laboratory #1

4.8.1.1 Problem

4.8.1.2 Questions

4.8.2 Practice laboratory #2

4.8.2.1 Problem

4.8.2.2 Questions

4.9 Project

4.9.1 Background

4.9.2 Data

4.9.3 Workflow

4.9.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.

- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.
- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.
- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

4.10 References

McNeill, Alexander J., Rudiger Frey, and Paul Embrechts. 2015. Quantitative Risk Management: Concepts, Techniques and Tools. Revised Edition. Princeton: Princeton University Press.

Ruppert, David and David S. Matteson. 2015. Statistics and Data Analysis for Financial Engineering with R Examples, Second Edition. New York: Springer.

Chapter 5

Term Structure and Splines

5.1 Imagine This

Our organization is about to expand its operations into the European Union (EU). To do that we will have to raise several hundred million dollars of collateral to back employees, customers, and our supply chain in the EU. Negative interest rates abound in the markets as political events and central bankers vie for control of the euro and its relationship to major currencies. Our task is to help the Chief Financial Officer understand how the term structure of interest rates might change and thus impact the amount and pricing of the collateral our organization is about to raise.

We might ask ourselves:

1. What is the term structure of interest rates?
2. Why would the term structure matter to the CFO?

"Term" refers to the maturity of a debt instrument, the "loan" we get to buy a house. "Term structure" is the schedule of interest rates posted for each maturity. The "term structure" is also known as the "yield curve." If the maturity on your loan (or your company's bonds) rises, a higher yield might just convince investors (like you or TIAA/CREF or other cash-rich investor) to wait longer for a return of the principal they lend to you. Higher yields mean also that you would have to price the bond at a higher coupon rate. This means more income is paid out to lenders and bondholders than to management, staff, and shareholders. The consequence is cash flow volatility.

Our objective is to conceive a model of bond prices that reflects the underlying dynamics of the term structure of interest rates. Such a model requires us to formulate forward rates of return on bonds of various maturities, and average these rates across the time to maturity in a yield calculation. We will then need to understand how to interpolate and extrapolate rates across maturities. With that tool in hand, we can price bonds, collateral, and build discount rates to evaluate future expected cash flows.

To do all of this we will employ "regression splines" to interpolate and extrapolate values of

forward rates from the term structure. The regression spline will reflect a financial model of the term structure applied to the estimation of actual term structures. We will

- Start with statistical definitions and financial models of bond prices.
- Move into the working example of US Treasury STRIPs (zero-coupon bonds) and explore the possibilities.
- Build a financially informed model of the term structure of forward empirical rates.
- Estimate the model with nonlinear least squares.
- Compare and contrast two competing model specifications.

In the end we will have these tools:

- Extensible quadratic spline model of the forward curve;
- `R` skills to explore data with plots and translate theoretical forward curve model into an estimable model; and
- Applications of the tools to the problem of managing the value of collateral.

5.2 The Bond

Our analysis begins with understanding the cash flows and valuation of a simple bond. A typical bond is a financial instrument that pays fixed cash flows for the maturity of the bond with a repayment of the principal (notional or face value) of the bond at maturity. In symbols,

$$V = \sum_{t=0}^{mT} \frac{cP}{(1+y/m)^t} + \frac{P}{(1+y/m)^{mT}},$$

where V is the present value of the bond, T is the number of years to maturity, m is the number of periods cash flows occur per year, y is the bond yield per year, c is the coupon rate per year, and P is the bond's principal (notional or face value).

Using the idea of an annuity that pays $(c/m)P$ per period for mT periods and P at maturity period mT we get a nice formula for the present value sum of coupon payments:

$$V = (c/m)P\left(\frac{1}{y/m} - \frac{1}{(y/m)(1+y/m)^{mT}}\right) + \frac{P}{(1+y/m)^{mT}}.$$

From a financial engineering point of view this formulation is the same as constructing a position that is

- long a perpetuity that pays a unit of currency starting the next compounding period and
- short another perpetuity that starts to pay a unit of currency at the maturity of the bond plus one compounding period.

Our typical bond pays coupons twice a year, so $m = 2$. If the bond's maturity in years is 10 years, then $mT = 10 \times 2 = 20$ compounding periods. We will assume that there is no accrual of interest as of the bond's valuation date for the moment.

```
c <- 0.05
P <- 100
y <- c(0.04, 0.05, 0.06)
m <- 2
T <- 10
(V <- (c/m) * P * (1/(y/m) - 1/((y/m) *
    (1 + (y/m))^(m * T))) + P/(1 + (y/m))^(m *
    T))
```

```
## [1] 108.17572 100.00000  92.56126
```

5.2.1 A quick example

1. If the coupon rate is greater than the yield, why is the price greater than par value?
2. Negative interest rates abound, so set `y <- c(-0.02, -0.01, 0.00, .01, .02)` and recalculate the potential bond values.

One answer might be:

The bond pays out at a rate greater than what is required in the market. Buyers pay more than par to make up the difference.

Here are more results. We assign variables to assumptions in the question and then calculate the

```
c <- 0.05
P <- 100
y <- c(-0.02, -0.01, 0, 0.01, 0.02)
m <- 2
T <- 10
(V <- (c/m) * P * (1/(y/m) - 1/((y/m) *
    (1 + (y/m))^(m * T))) + P/(1 + (y/m))^(m *
    T))
```

```
## [1] 177.9215 163.2689      NaN 137.9748 127.0683
```

Why a `NAN`? Because we are dividing by '0'! By the way, these are some hefty premia. Check out this article on negative yields... http://www.ft.com/cms/s/0/312f0a8c-0094-11e6-ac98-3c15a1aa2e62.html

The yield in this model is an average of the forward rates the markets used to price future cash flows *during future periods of time*. How can we incorporate a changing forward rate into a model of bond prices? Our strategy is to use the past as a guide for the future and

calibrate rates to a curve of rates versus maturities. By using a regression spline model of the term structure we can develop rates at any maturity. Then we can use the interpolated (even extrapolated) rates as building blocks in the valuation of bonds.

5.2.2 What's a spline?

A *spline* is a function that is constructed piece-wise from polynomial functions. But imagine the polynomials are pieces of the term structure of interest rates. Each term structure segment is marked off by a set of price and maturity pairs. Whole sections can be marked off by a *knot* at a location in the term structure paired data. Knots are most commonly placed at quantiles to put more knots where data is clustered close together. A different polynomial function is estimated for each range and domain of data between each knot: this is the spline. Now we will use some finance to build a polynomial regression spline out of US Treasury zero-coupon data.

In general a polynomial is an expression that looks like this:

$$f(x) = a_0x^0 + a_1x^1 + a_2x^2 + ... + a_px^p,$$

where the a's are constant coefficients, and $x^0 = 1$ and $x^1 = x$.

- If $p = 0$, then we have a *constant* function.
- If $p = 1$, we have a *linear* function.
- If $p = 2$, we have a *quadratic* function.
- If $p = 3$, we have a *cubic* function, and so on…

In our term structure work We will be using a *cubic* function to build a regression spline.

5.2.3 Back to the Bond

Suppose investors give an issuer today the present value, the bond "price" P_T of receiving back the 100% of the face value of a zero-coupon bond at maturity year (or fraction thereof), T.

The price of a zero coupon bond, quoted in terms of the percentage of face value, is this expression for discrete compounding at rate y_T^d (say, monthly you get a percentage of last month's balance):

$$P_T = \frac{100}{(1 + y^d(T))^T}.$$

Suppose, with the help of Jacob Bernoulli and Leonhardt Euler, we run this experiment:

1. Pay today at the beginning of the year P to receive \$1 at the end of one year AND interest y is compounded only once in that year. Thus at the end of the year we receive $P + yP$. But by contract this amount is \$1, so that

$$P + yP = P(1 + y) = 1$$

and solving for P

$$P = \frac{1}{(1+y)^1}$$

where we raised $(1 + y)$ to the first power to emphasize that one compounding period was modeled.

2. Now suppose that we can receive interest twice a year at an annual interest rate of y. Then at the end of the first half of the year we receive half of the interest $y/2$ so that we have in our contract account

$$P\left(1 + \frac{y}{2}\right)$$

We then can let this stay in the account this amount will also earn $y/2$ interest to the end of the contract year so that

$$P\left(1 + \frac{y}{2}\right) + \frac{y}{2}\left[P\left(1 + \frac{y}{2}\right)\right] = P\left(1 + \frac{y}{2}\right)\left(1 + \frac{y}{2}\right) = P\left(1 + \frac{y}{2}\right)^2$$

Setting

$$P\left(1 + \frac{y}{2}\right)^2 = \$1$$

we can solve for P, the present value of receiving \$1 at the end of the contract year when we have two compounding periods or

$$P = \frac{\$1}{\left(1 + \frac{y}{2}\right)^2}$$

3. We can, again with Jacob Bernoulli's help, more generally state for m compounding periods

$$P = \frac{\$1}{\left(1 + \frac{y}{m}\right)^m}$$

4. Now let's suppose $y = 100\%$ interest and $m = 365 \times 24 \times 60 = 525600$ compounding periods (minute by minute compounding), then

$$P = \frac{\$1}{\left(1 + \frac{1}{525600}\right)^{5\,25600}}$$

```
(m <- 365 * 24 * 60)
```

```
## [1] 525600
```

```
(y <- 1)
```

```
## [1] 1
```

```
(P <- 1/(1 + (y/m))^m)
```

```
## [1] 0.3678798
```

This translates into continuous compounding (you get a percentage of the last nanosecond's balance at the end of this nanosecond…as the nanoseconds get ever smaller…) as

$$P_T(\theta) = 100exp(-y_T T).$$

This expression is the present value of receiving a cash flow of 100% of face value at maturity. If the bond has coupons, we can consider each of the coupon payments as a mini-zero bond. Taking the sum of the present value of each of the mini-zeros gives us the value of the bond, now seen as a portfolio of mini-zeros.

The yield y_T the rate from date 0 to date T, maturity. It covers the stream of rates for each intermediate maturity from 0 to T. Suppose we define forward rates $r(t, \theta)$, where each t is one of the intermediate maturity dates between time 0 and maturity T, and θ contains all of the information we need about the shape of r across maturities. We can estimate the forward curve from bond prices $P(T)$ of the Tth maturity with

$$-\frac{\Delta log(P(T_i))}{\Delta T_i} = -\frac{log(P(T_i)) - log(P(T_{i-1}))}{T_i - T_{i-1}}.$$

The Δ stands for the difference in one price or maturity i and the previous price and maturity $i-1$. The $log()$ function is the natural logarithm. An example will follow.

The *yield* is then the *average of the forward rates* from date 0 to date T of a zero bond. We use the integral, which amounts to a cumulative sum, to compute this average:

$$y_T(\theta) = T^{-1} \int_0^T r(t, \theta) dt.$$

The numerator is the cumulative sum of the forward rates for each maturity up to the last maturity T. In this expression, rdt is the forward rate across a small movement in maturity. The denominator is the number of maturity years T.

5.2.4 An example to clarify

Load these lines of `R` into the RStudio console:

```
maturity <- c(1, 5, 10, 20, 30)  # in years
price <- c(99, 98, 96, 93, 89)  # in percentage of face value
```

A. Now let's experiment on these zero-coupon prices with their respective maturities:

1. Calculate the `log(price)/100`. Then find the forward rates using using

```
(forward <- -diff(log(price))/diff(maturity))
```

2. Compare `log(price)` with `price`.
3. What does the forward rate formula indicate? What would we use it for?

B. Find the yield-to-maturity curve and recover the bond prices using

```
(forward.initial <- -log(price[1]/100))
(forward.integrate <- c(forward.initial,
    forward.initial + cumsum(forward *
        diff(maturity))))
# a rolling integration of rates
# across maturities
(price <- 100 * exp(-forward.integrate))
# present value of receiving 100% of
# face value
```

1. What is the interpretation of the `forward.integrate` vector?
2. What happened to the first bond price?
3. Did we recover the original prices?

Some results follow.

For question A we ran the forward rates. Let's run the natural logarithm of `price`:

```
(forward <- -diff(log(price/100))/diff(maturity))
```

```
## [1] 0.002538093 0.004123857 0.003174870 0.004396312
```

```
(-log(price/100))
```

```
## [1] 0.01005034 0.02020271 0.04082199 0.07257069 0.11653382
```

`-log(price/100)` seems to give us the yield-to-maturity directly. But do look at `-diff(log(price))` next:

```
(-diff(log(price/100)))
```

```
## [1] 0.01015237 0.02061929 0.03174870 0.04396312
```

These look like rates, because they are. They are the continuous time version of a percentage change, or growth, rate from one maturity to another maturity. We can use these numerical results to motivate an interpretation: interest rates are simply rates of change of present values relative to how much time they cover in maturity.

Now let's calculate

```
(-diff(price/100)/(price[-length(price)]/100))
```

```
## [1] 0.01010101 0.02040816 0.03125000 0.04301075
```

These are discrete percentage changes that are similar, but not quite the same, as the continuous (using `log()`) version.. Note the use of the indexing of price to eliminate the last price, since what we want to compute is:

$$\frac{P(T_i) - P(T_{i-1})}{P(T_{i-1})}$$

5.2.5 Rolling the integration

Running the code for question B we get:

```
forward.initial <- -log(price[1]/100)
(forward.integrate <- c(forward.initial,
    forward.initial + cumsum(forward *
        diff(maturity))))
```

```
## [1] 0.01005034 0.02020271 0.04082199 0.07257069 0.11653382
```

```
# a rolling integration of rates
# across maturities
(price <- 100 * exp(-forward.integrate))
```

```
## [1] 99 98 96 93 89
```

```
# present value of receiving 100% of
# face value
```

The rolling "integration" is an accumulative process of adding more forward rates as the maturity advances, thus a cumulative sum or in `R` a `cumsum()` is deployed.

1. Yields are the accumulation of forward rates. Thus the use of the cumulative sum as a discrete version of the integral. Rates add up (instead of multiply: nice feature) when we use `log` and `exp` to do our pricing.
2. The first forward rate is just the discount rate on the 1-year maturity bond stored in `forward.initial`.
3. All bond prices are recovered by inverting the process of producing forwards from prices and converting to yields and back to prices.

5.2.6 And now some more about that bond yield

We restate the definition of the price of a zero-coupon bond. The price of a zero-coupon bond quoted in percentage of face value is this expression for discrete compounding (say,

monthly you get a percentage of last month's balance):

$$P_T = \frac{100}{(1 + y^d(T))^T}$$

This translates into continuous compounding (you get a percentage of the last nanosecond's balance at the end of this nanosecond...) as

$$P_T(\theta) = 100exp(-y_T T) = 100exp(\int_0^T r(t, \theta)dt)$$

This is the present value of receiving a cash flow of 100% of face value at maturity. If the bond has coupons we can consider each of the coupon payments as a mini-zero bond.

The term with the $\int$ symbol is a nanosecond by nanosecond way of summing the forward rates across the maturity of the bond. Equating the two statements and solving for y_T, the continuously compounded yield we get:

$$P_T = \frac{100}{(1 + y(T)_d)^T} = 100exp(-y_T T).$$

Rearranging with some creative algebra: 100 drops out, and remembering that $exp(-y_T t) = 1/exp(y_T T)$, we have:

$$exp(y_T T) = (1 + y(T)_d)^T.$$

Then, taking logarithms of both sizes we get

$$exp(y_T T)^{-T} = (1 + y(T)_d)^{T-T} = (1 + y(T)_d).$$

Using the facts that $log(exp(x)) = x$ and $log(x^T) = Tlog(x)$ (very convenient):

$$log(exp(y_T T)^{-T}) = y_T = log(1 + y(T)_d).$$

We now calculate discrete and continuous time versions of the yield:

```
y.T.d <- 0.2499  # Usury rate in NYS; d = discrete, T = maturity in years
(y.T <- log(1 + y.T.d))
```

```
## [1] 0.2230635
```

```
(y.T.d <- exp(y.T) - 1)
```

```
## [1] 0.2499
```

We note that the continuous time yield is always less than the discrete rate because there are so many more continuous compounding periods. This result leaps back to Jacob Bernoulli's (17xx) derivation.

5.3 Forward Rate Parameters

Now we attend to the heretofore mysterious θ, embedded in the forward rate sequence r. We suppose the forward rate is composed of a long-term constant, θ_0; a sloping term with parameter θ_1; a more shapely, even "humped" term with parameter θ_2; and so on for p polynomial terms:

$$r(t, \theta) = \theta_0 + \theta_1 t + \theta_2 t^2 + ... + \theta_p t^p$$

We recall (perhaps not too painfully!) from calculus that when we integrate any variable to a power (the antiderivative), we raise the variable to the next power (and divide the term by that next power). Here we integrate the forward rate to get

$$\int_0^T r(t, \theta) dt = \theta_0 T + \theta_1 \frac{T^2}{2} + \theta_2 \frac{T^3}{3} + ... + \theta_p \frac{T^{p+1}}{p+1}.$$

This is equivalent, after a fashion, to calculating the cumulative sum of forward rates across small increments in time dt from today at maturity $t = 0$ to the maturity of the term at $t = T$. We then can estimate the yield curve (and then the zero-coupon bond price) using the integrated forward rates divided by the maturity T to get the yield as

$$y_T(\theta) = \theta_0 + \theta_1 \frac{T}{2} + \theta_2 \frac{T^2}{3} + ... + \theta_p \frac{T^p}{p+1}.$$

Before we go any further we will procure some term structure data to see more clearly what we have just calculated.

5.3.1 Term Structure Data

Here is some (very old) data from US Treasury STRIPS (a.k.a. for "Separate Trading of Registered Interest and Principal of Securities"). The data set will prices and maturities for what are known as "zero-coupon" bonds, that is, bonds that only pay out the face value of the bond at maturity. There is more information about STRIPS at https://www.newyorkfed.org/aboutthefed/fedpoint/fed42.html.

```
# The data is in a directory called
# data that is a sub-directory of the
# directory from which this code
```

```
# executes.
dat <- read.table("data/strips_dec95.txt",
    header = TRUE)
```

What does the data look like? Anything else? We run this code chunk to get a preliminary view of this data.

```
head(dat, n = 3)
```

```
##        T  price
## 1 0.1260 99.393
## 2 0.6219 96.924
## 3 1.1260 94.511
```

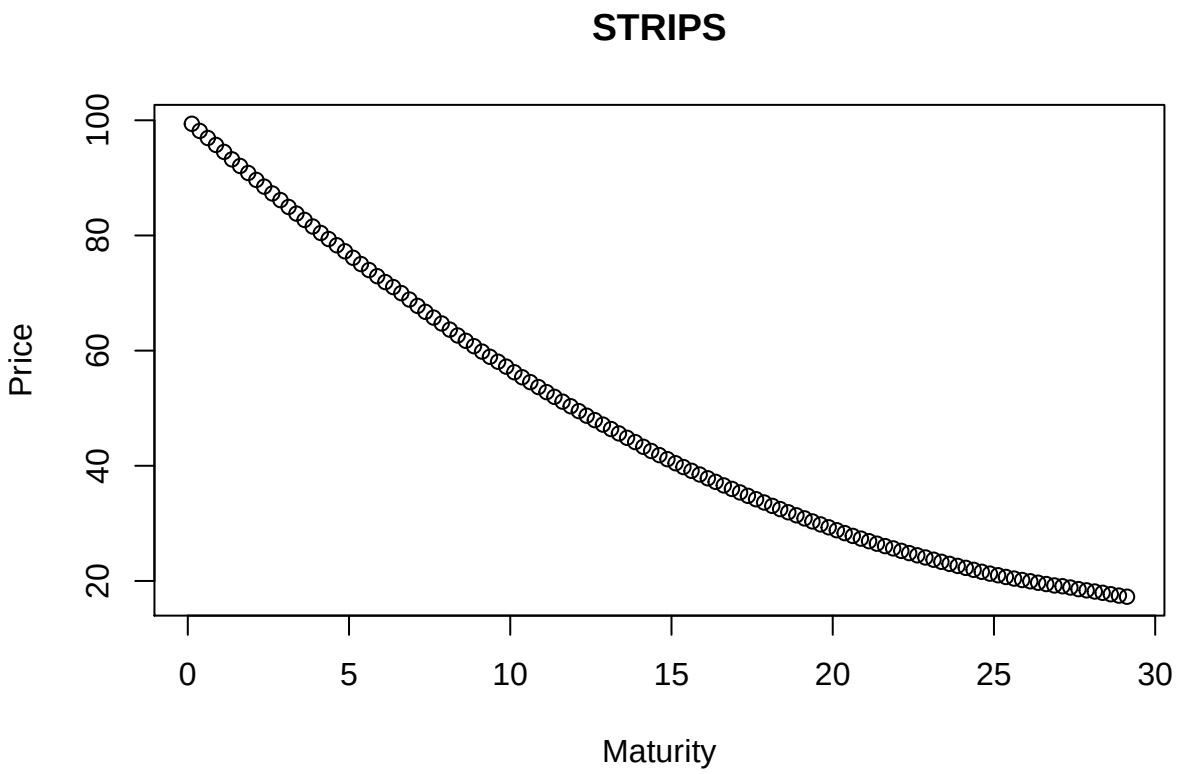
```
names(dat)
```

```
## [1] "T"     "price"
```

```
dat <- dat[order(dat$T), ]
```

We read in a table of text with a header, look at the first few observations with `head()`, and order the data by maturity T_i. A simple plot is in order now.

```
plot(dat$T, dat$price, main = "STRIPS",
    xlab = "Maturity", ylab = "Price")
```



5.3.2 The empirical forward curve

We estimate the empirical forward curve using:

$$-\frac{\Delta log(P(T_i))}{\Delta T_i} = -\frac{log(P(T_i)) - log(P(T_{i-1}))}{T_i - T_{i-1}},$$

where P is the bond price and T_i is ith maturity,

```
t <- seq(0, 30, length = 100)
emp <- -diff(log(dat$price))/diff(dat$T)
```

The equation is translated into `R` with the `diff` function. We will use vector `t` later when we plot our models of the forward curve. Definitely view it from the console and check out `??seq` for more information about this function.

5.3.3 Try this exercise

Let's plot the empirical forward curves using this line of code:

```
plot(dat$T[2:length(dat$T)], emp, ylim = c(0.025,
    0.075), xlab = "maturity", ylab = "empirical forward rate",
    type = "b", cex = 0.75, lwd = 2,
    main = "US Treasury STRIPs - 1995")
```

Try to answer these questions before moving on:

1. What exactly will `dat$T[2:length(dat$T)]` do when executed?
2. What effect will `ylim`, `lwd`, `xlab`, `ylab`, `type`, `cex`, `main` have on the plot?
3. Is there an break in the curve? Write the plot command to zoom in on maturities from 10 to 20 years.

We get these results:

1. What exactly will `dat$T[2:length(dat$T)]` do when executed?

```
length(dat$T)
```

```
## [1] 117
```

```
head(dat$T[2:length(dat$T)])
```

```
## [1] 0.3699 0.6219 0.8740 1.1260 1.3699 1.6219
```

We retrieve the `T` vector from the `dat` data frame using `dat$T`. Then `length` returns the number of maturities in the `dat` data frame. `dat$T[2:length(dat$T)]` truncates the first observation. Why? Because we differenced the prices to get forward rates and we need to align the forward rates with their respective maturities T_i.

5.3.4 Plot parameters

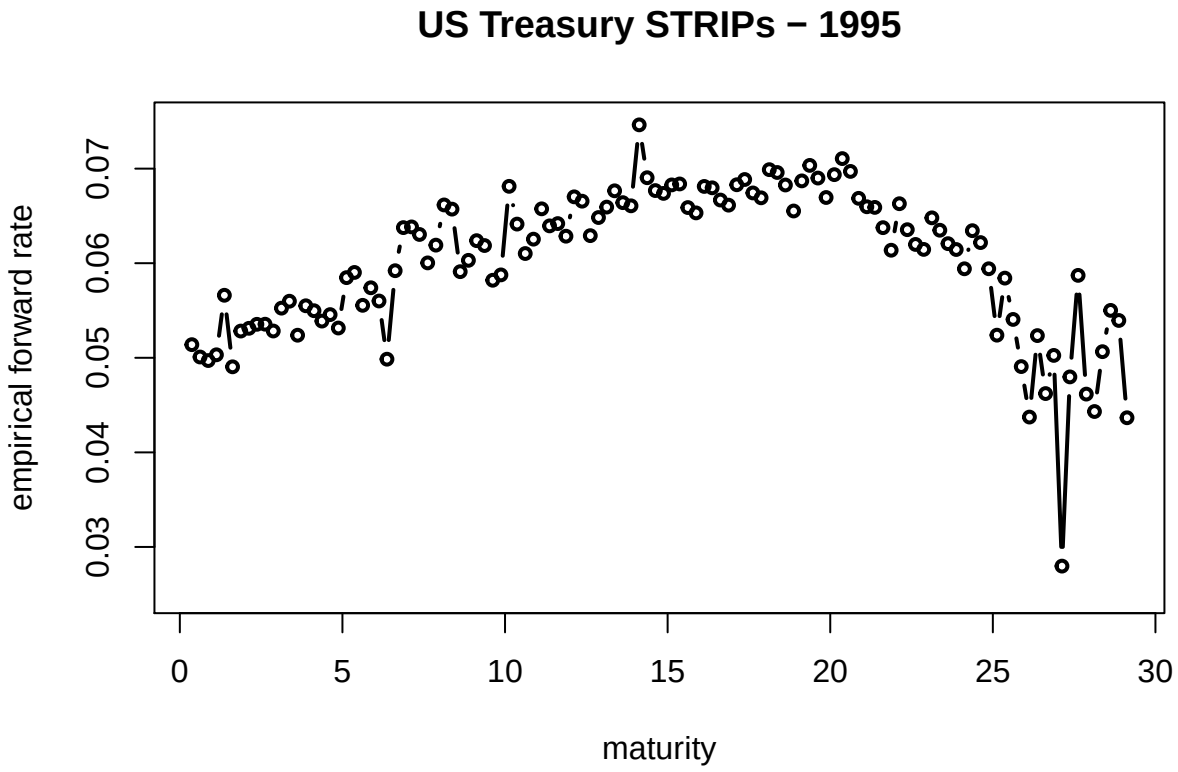
Here is the list:

- `ylim` zooms in the y-axis data range
- `lwd` changes the line width
- `xlab` specifies the x-axis label
- `ylab` specifies the y-axis label
- `type` specifies the line
- `cex` changes the scaling of text and symbols
- `main` specifies the plot title.

We can go to http://www.statmethods.net/advgraphs/parameters.html to find more information and the answer to the zoom-in question.

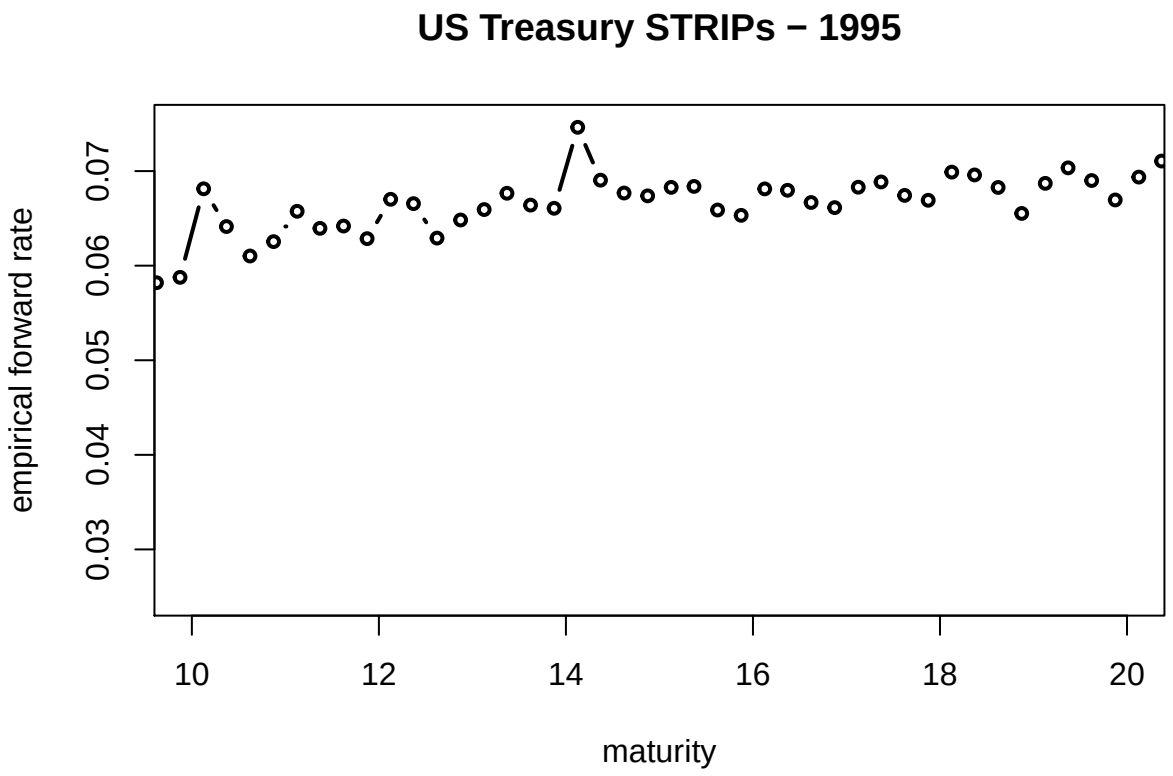
5.3.5 The plot itself

```
plot(dat$T[2:length(dat$T)], emp, ylim = c(0.025,
    0.075), xlab = "maturity", ylab = "empirical forward rate",
    type = "b", cex = 0.75, lwd = 2,
    main = "US Treasury STRIPS - 1995")
```



Now the zoom in:

```
plot(dat$T[2:length(dat$T)], emp, xlim = c(10,
    20), ylim = c(0.025, 0.075), xlab = "maturity",
    ylab = "empirical forward rate",
    type = "b", cex = 0.75, lwd = 2,
    pin = c(3, 2), main = "US Treasury STRIPS - 1995")
```



At $T_i = 14$ there appears to be an outlier. More importantly there is a break at $T_i = 15$. This is a natural *knot*. Thus the possibility of a need for a *spline*.

5.4 Back to Our Story

Let's add a kink in the yield curve that allows two different quadratic functions: one before the kink, and one after the kink. Let the kink be a knot k at $T_i = 15$. We evaluate a knot as 0 if the maturities $T_i - k < 0$, and equal $T - k$ if $T - k > 0$. We write this as $(T - k)_+$. Let's now drop the knot into our integral:

$$\int_0^T r(t, \theta)dt = \theta_0 T + \theta_1 \frac{T^2}{2} + \theta_2 \frac{T^3}{3} + \theta_2 \frac{(T - 15)_+^3}{3}$$

We can divide by T to get the yield. But to calculate the bond price we have to multiply the yield by the bond maturity T, so the bond price is then:

$$P_T(\theta) = 100exp[-(\theta_0 T + \theta_1 \frac{T^2}{2} + \theta_2 \frac{T^3}{3} + \theta_3 \frac{(T - 15)_+^3}{3})]$$

After all of that setup now we move on to estimate the thetas.

5.4.1 Gee, that's very nonlinear of you...

Yes, the bond price is nonlinear in the θ parameters. Our statistical job now is to find a set of θ such that the difference between the actual bond prices P and our clever model of bond prices (long equation that ended the last slide) is very small in the sense of the sum of squared differences ("errors"). We thus find θ that minimizes

$$\sum_{i=0}^{N}[(P(T_i) - P(T_i, \theta)]^2$$

To find the best set of θ we will resort to a numerical search using the `R` function `nls`, for nonlinear least squares.

5.4.2 Try this exervise as we get down to business

Back to the data: we now find the θs. The logical expression `(T>k)` is 1 if `TRUE` and 0 if `FALSE`. We put the `R` version of the bond price into the `nls` function, along with a specification of the data frame `dat` and starting values.

Run these statements to compute the (nonlinear) regression of the term structure:

```
fit.spline <- nls(price ~ 100 * exp(-theta_0 *
    T - (theta_1 * T^2)/2 - (theta_2 *
    T^3)/3 - (T > 15) * (theta_3 * (T -
    15)^3)/3), data = dat, start = list(theta_0 = 0.047,
    theta_1 = 0.0024, theta_2 = 0, theta_3 = -7e-05))
```

5.4.3 Just a couple of questions:

1. What are the dependent and independent variables?
2. Which parameter measures the sensitivity of forward rates to the *knot*?

The dependent variable is `price`. The independent variables are inside the exp() operator and include powers of `T`, the maturities of the bonds. The `theta_3` parameter exposes the bond price to movements in maturity around the knot at $T = 15$ years.

Let's look at our handiwork using `kable` from the `knitr` package:

```
library(knitr)
kable(summary(fit.spline)$coefficients,
    digits = 4)
```

	Estimate	Std. Error	t value	Pr(>|t|)
theta_0	0.0495	1e-04	536.5180	0
theta_1	0.0016	0e+00	51.5117	0
theta_2	0.0000	0e+00	-13.6152	0
theta_3	-0.0002	0e+00	-30.6419	0

```
(sigma <- (summary(fit.spline)$sigma)^0.5)
```

```
## [1] 0.2582649
```

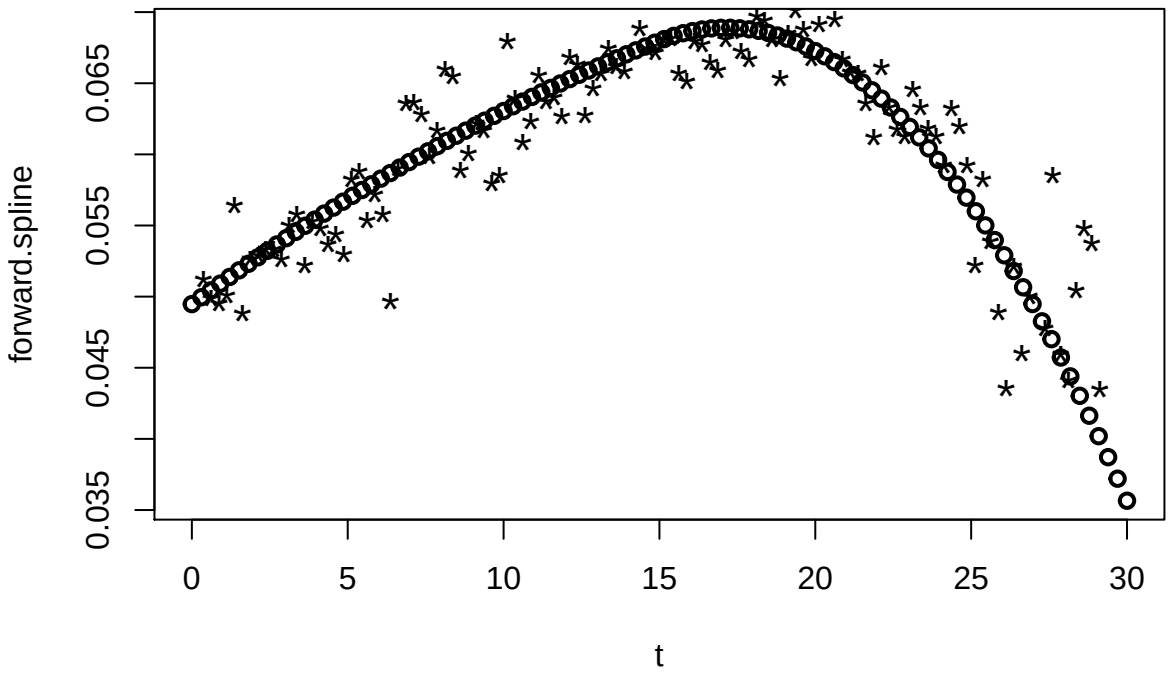
All coefficients are significant and we have a standard error to compare with other models.

5.4.4 Build a spline

Let's now produce a plot of our results using a sequence of maturities `T`. Our first task is to parse the coefficients from the `nls()` spline fit and build the spline prediction. Here we construct the forward rate spline across `T` maturities.

```
coef.spline <- summary(fit.spline)$coef[,
    1]
forward.spline <- coef.spline[1] + (coef.spline[2] *
    t)
+(coef.spline[3] * t^2)
+(t > k) * (coef.spline[4] * (t - 15)^2)
```

Second, pull the coefficients from a `summary()` of the `fit.spline` object for the plot.



5.5 A Summary Exercise

Let's try these tasks to bring this analysis together.

1. Compare the quadratic spline we just constructed with a pure quadratic polynomial. Simply take the knot out of the `nls` formula and rerun. Remember that "quadratic" refers to the polynomial degree p of the assumed structure of the forward rate $r(t, \theta)$.
2. Plot the data against the quadratic spline and the quadratic polynomial.
3. Interpret financially the terms in θ_0, θ_1, and θ_2.

5.5.1 Compare

Run this code: remembering that a "cubic" term represents the integration of the "quadratic" term in

$$r(t, \theta) = \theta_0 + \theta_1 t + \theta_2 t^2$$

```
fit.quad <- nls(price ~ 100 * exp(-theta_0 *
    T - (theta_1 * T^2)/2 - (theta_2 *
    T^3)/3), data = dat, start = list(theta_0 = 0.047,
    theta_1 = 0.0024, theta_2 = 0))
```

This estimate gives us *one* quadratic function through the cloud of zero-coupon price data.

```
knitr::kable(summary(fit.quad)$coefficients,
    digits = 4)
```

	Estimate	Std. Error	t value	Pr(>|t|)
theta_0	0.0475	2e-04	239.0168	0
theta_1	0.0024	1e-04	46.5464	0
theta_2	-0.0001	0e+00	-33.0016	0

All conveniently significant.

```
(sigma <- (summary(fit.quad)$sigma)^0.5)
```

```
## [1] 0.4498518
```

The pure quadratic model produces a higher standard deviation of error than the quadratic spline.

5.5.2 Plot

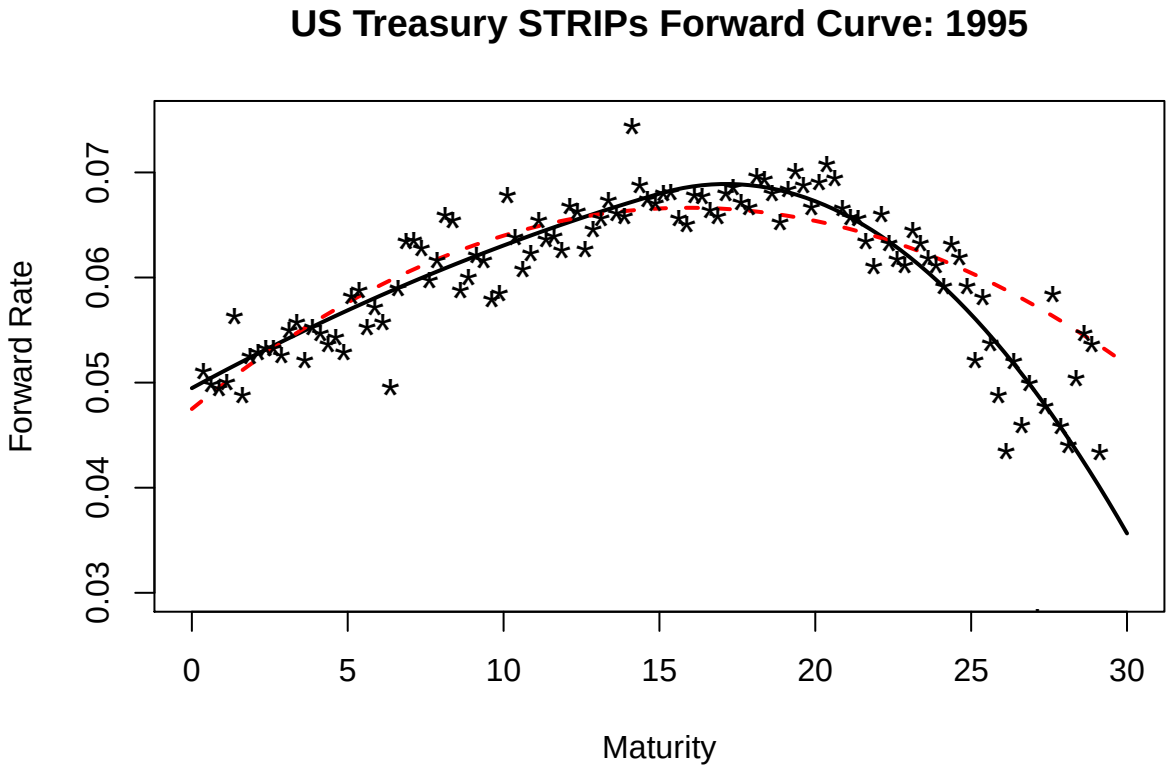
We will run this code to set up the data for a plot. First some calculations based on the estimations we just performed.

```
coef.spline <- summary(fit.spline)$coef[,
    1]
forward.spline <- coef.spline[1] + (coef.spline[2] *
    t)
+(coef.spline[3] * t^2)
+(t > 15) * (coef.spline[4] * (t - 15)^2)
coef.quad <- summary(fit.quad)$coef[,
    1]
forward.quad <- coef.quad[1] + (coef.quad[2] *
    t) + (coef.quad[3] * t^2)
```

Here is the plot itself.

```
plot(t, forward.spline, type = "l", lwd = 2,
    ylim = c(0.03, 0.075), xlab = "Maturity",
    ylab = "Forward Rate", main = "US Treasury STRIPs Forward Curve: 1995")
lines(t, forward.quad, lty = 2, lwd = 2,
    col = "red")
```

```
points(dat$T[2:length(dat$T)], emp, pch = "*",
    cex = 1.5)
```



We can use "eyeball" econometrics to see what data the quadratic forward curve misses!

5.5.3 Interpret

There seem to be three components in the forward curve:

- Constant
- Slope
- Curved (affectionately called "humped")

Our interpretation follows what we might visualize:

- θ_0 is independent of maturity and thus represents the long-run average forward rate.
- θ_1 helps to measure the average sensitivity of forward rates to a change in maturity.
- θ_2 helps to measure the maturity risk of the forward curve for this instrument.

The pure quadratic forward curve seems to dramatically underfit the maturities higher than 15 years. Using a knot at the right maturity adds a boost to the reduction of error in this regression. That means that predictions of future potential term structures will apt to be more accurate than the null hypothesis of no knot.

5.6 Just one more thing

Here is the infamous **So What?!**, especially after all of the work we just did.

1. Suppose we just bought a 10 year maturity zero-coupon bond to satisfy collateral requirements for workers' compensation in the (great) State of New York.
2. The forward rate has been estimated as:

$$r(t) = 0.001 + 0.002t - 0.0003(t-7)_+$$

3. In 6 months we then exit all business in New York State, have no employees that can claim workers' compensation, sell the 10 year maturity zero-coupon bond. The forward curve is now

$$r(t) = 0.001 + 0.0025t - 0.0004(t-7)_+$$

4. How much would we gain or lose on this transaction at our exit?

5.6.1 Some bond maths to (re)consider

Let's recall the following:

1. The forward rate is the rate of change of the yield-to-maturity
2. This means we integrate (i.e., take the cumulative sum of) forward rates to get the yield
3. The cumulative sum would then be some maturity times the components of the yield curve adjusted for the slope of the forward curve (the terms in t).
4. This adjustment is just one-half (1/2) of the slope term.

Here are some calculations to set up today's yield curve and the curve 6 months out.

```
maturity.now <- 10
maturity.6m <- 9.5
(yield.now <- 0.001 * maturity.now +
    0.002 * maturity.now^2/2 - 3e-04 *
    (maturity.now > 7)^2/2)
```

```
## [1] 0.10985
```

```
(yield.6m <- 0.001 * maturity.6m + 0.0025 *
    maturity.6m^2/2 - 4e-04 * (maturity.6m >
    7)^2/2)
```

```
## [1] 0.1221125
```

Using these yields we can compute the bond prices for today and for 6 months out as well.

```
(bond.price.now <- exp(-yield.now))
```

```
## [1] 0.8959685
```

```
(bond.price.6m <- exp(-yield.6m))
```

```
## [1] 0.8850488
```

5.6.2 Exit

Our exit transaction is long today's version of the bond and short the 6 month version. This translates into a 6 month return that calculates

$$R_{6m} = \frac{P(T-0.5) - P(T)}{P(T)} = \frac{P(T-0.5)}{P(T)} - 1$$

which in R is

```
(return = bond.price.6m/bond.price.now -
    1)
```

```
## [1] -0.01218762
```

```
2 * return  ### annualized return
```

```
## [1] -0.02437524
```

It appears that we lost something in this exit from New York state as the return is negative.

5.7 Summary

This chapter covers the fundamentals of bond mathematics: prices, yields, forward curves. Using this background built two models of the forward curve and then implemented these models in R with live data. In the process We also learned something about the nonlinear least squares method and some more R programming to visualize results.

5.8 Further Reading

Ruppert and Matteson, pp. 19-43, introduces fixed income (bonds) including yield to maturity, continuous discounting, forward rates, and bond prices. From pp. 271-281 we can learn about nonlinear regression and the use of quadratic terms in term structure estimations. From pp. 645-664 we can learn about regression splines and their application to various analytical questions. McNeill et al. have similar bond mathematics discussions from pp. 329-337. They use the `termstr` package from CRAN to illustrate term structure models.

5.9 Practice Laboratory

5.9.1 Practice laboratory #1

5.9.1.1 Problem

5.9.1.2 Questions

5.9.2 Practice laboratory #2

5.9.2.1 Problem

5.9.2.2 Questions

5.10 Project

5.10.1 Background

5.10.2 Data

5.10.3 Workflow

5.10.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.
- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.
- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or

procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.

- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

5.11 References

McNeill, Alexander J., Rudiger Frey, and Paul Embrechts. 2015. Quantitative Risk Management: Concepts, Techniques and Tools. Revised Edition. Princeton: Princeton University Press.

Ruppert, David and David S. Matteson. 2015. Statistics and Data Analysis for Financial Engineering with R Examples, Second Edition. New York: Springer.

Chapter 6

Market Risk

6.1 Imagine This

Suppose a division in our company buys electricity to make steel. We know of two very volatile factors in this process:

1. The price of steel at the revenue end, and the other is
2. The price of electricity at the cost end.

To model the joint volatility of the power-steel spread We can use electricity and steel prices straight from the commodity markets. We can also use stock market indexes or company stock prices from electricity producers and transmitters and from a steel products company to proxy for commodities. Using company proxies gives us a broader view of these commodities than just the traded pure play in them.

In this chapter we will

1. Measure risks using historical and parametric approaches
2. Interpret results relative to business decisions
3. Visualize market risk

6.2 What is Market Risk?

Market risk for financial markets is the impact of unanticipated price changes on the value of an organization's position in instruments, commodities, and other contracts. In commodity markets there is sometimes considered a more physical type of market risk called volumetric risk that relates to the delivery of the commodity to a buyer. This risk might be triggered by a complex contract such as a CDO or a spark spread tolling agreement in power and energy markets. Here we will assume that volumetric changes are physical in the sense that a electricity system operator governs a physical process such as idling an electric generation plant.

A "position" is the physical holding of an asset or commodity, such as a share of Apple stock, or an ounce of gold. A long position means that the market participant possesses a positive amount of the asset or commodity. A short position means that the market participant does not possess an asset or commodity. The "position" is considered exogenous to the price stochastic process. This implies that changes in position do not affect the liquidity of the market relative to that position.

6.2.1 Try this exercise

Suppose you are in charge of the project to manage the contracting for electricity and steel at your speciality steel company. Your company and industry have traditionally used tolling agreements to manage the steel-power spread, both from a power input and a steel output point of view.

1. Look up a tolling agreement and summarize its main components.
2. What are the input and output decisions in this kind of agreement.

Here are some results

1. In the electric power to steel tolling agreement a steel buyer supplies power to a steel plant and receives from the plant supplier an amount of steel based on an assumed power-to-steel transformation rate at an agreed cost.

- Prices of power and steel
- Transformation rate
- Agree cost

2. Decisions include

- Buying an amount of power (MWh)
- Selling an amount of steel (tons)
- Scheduling plant operations

Decisions will thus depend on the prices of electricity and steel, the customer and vendor segments served, the technology that determines the steel (tons) / Power (MWh) transformation rate, start-up, idle, and shut-down timing and costs and overall plant production costs.

6.3 History Speaks

To get the basic idea of risk measures across we develop the *value at risk* and *expected shortfall* metrics from the historical simulated distributions of risk factors. Given these risk factors we combine them into a portfolio and calculate their losses. Finally with the loss distribution in hand we can compute the risk measures.

We will use a purely, non-parametric historical simulation approach in this section. This means that we will not need to compute means, standard deviations, correlations, or other statistical estimators, also known as parameters.

First we need to get some data. We will use throughout these computations several libraries:

1. `mvtnorm` builds multivariate normal (Gaussian) simulations and
2. `QRM` estimates Student-t and generalized pareto distribution (GPD) simulation.
3. We will hold off on these parametric approaches till later and start with historical simulation.
4. The `psych` library helps us to explore the interactions among data through scatter plots and histograms.
5. The `ggplot2` library allows us to build complex vizualizations that will aid the generation of further insights.

We read in the `csv` file from the working directory. This file contains dates and several risk factors. In this setup we will use RWE stock prices will stand in for electricity price risk factor input and THYSSEN stock prices for the steel price risk factor.

```
# Download the data
data.all <- read.csv("data/eurostock.csv",
    stringsAsFactors = FALSE)
## This will convert string dates to
## date objects below
str(data.all)  ## Check the structure and look for dates
```

```
## 'data.frame':    6147 obs. of  24 variables:
##  $ X              : chr  "1973-01-01" "1973-01-02" "1973-01-03" "1973-01-04" ...
##  $ ALLIANZ.HLDG.  : num  156 156 161 162 164 ...
##  $ COMMERZBANK    : num  147 147 149 152 152 ...
##  $ DRESDNER.BANK  : num  18.4 18.4 18.8 18.9 18.9 ...
##  $ BMW            : num  104 109 110 111 109 ...
##  $ SCHERING       : num  36.9 37.4 37.8 37.9 37.4 ...
##  $ BASF           : num  15 15.4 15.6 15.8 15.8 ...
##  $ BAYER          : num  12.2 11.9 12.1 12.7 12.7 ...
##  $ BAYERISCHE.VBK.: num  23.5 22.9 23.4 23.7 23.9 ...
##  $ BAYER.HYPBK.   : num  23.4 23.2 23.3 23.5 23.4 ...
##  $ DEGUSSA        : num  203 207 208 210 214 ...
##  $ DEUTSCHE.BANK  : num  22.3 22.5 22.9 23 23.3 ...
##  $ CONTINENTAL    : num  8.54 8.83 8.78 8.83 8.73 8.82 8.74 8.73 8.74 8.74 ...
##  $ VOLKSWAGEN     : num  134 140 145 144 140 ...
##  $ DAIMLER.BENZ   : num  17 17.6 17.8 17.8 17.7 ...
##  $ HOECHST        : num  13.8 13.8 14.2 14.3 14.2 ...
##  $ SIEMENS        : num  20.8 21.1 21.3 21.4 21.5 ...
##  $ KARSTADT       : num  360 360 362 369 368 ...
##  $ LINDE          : num  136 137 140 142 144 ...
##  $ THYSSEN        : num  67.5 68.4 67.5 71.6 71.2 ...
```

```
##  $ MANNESMANN     : num  85 86.5 87.8 88.7 88.6 ...
##  $ MAN            : num  118 119 125 125 127 ...
##  $ RWE            : num  11.7 11.9 12 11.9 12 ...
##  $ INDEX          : num  536 545 552 556 557 ...
```

The next thing we must do is transform the data set into a time series object. The way we do that is to make the dates into row names so that dates are the index for the two risk factors. Making dates an index allows us to easily filter the data.

```
str(row.names <- data.all$X)  ## We find that the first field X contains dates
```

```
##  chr [1:6147] "1973-01-01" "1973-01-02" "1973-01-03" "1973-01-04" ...
```

```
date <- as.Date(row.names)  ## convert string dates to date objects
str(date)  ##Always look at structure to be sure
```

```
##  Date[1:6147], format: "1973-01-01" "1973-01-02" "1973-01-03" "1973-01-04" "1973-01-0
```

```
rownames(data.all) <- date
head(data.all)
```

```
##                     X ALLIANZ.HLDG. COMMERZBANK DRESDNER.BANK    BMW
## 1973-01-01 1973-01-01        155.51       147.41          18.40 103.97
## 1973-01-02 1973-01-02        155.51       147.41          18.40 109.05
## 1973-01-03 1973-01-03        160.58       149.14          18.80 109.83
## 1973-01-04 1973-01-04        162.27       152.05          18.91 110.81
## 1973-01-05 1973-01-05        164.30       152.05          18.89 109.44
## 1973-01-08 1973-01-08        164.30       152.25          18.99 109.05
##            SCHERING  BASF BAYER BAYERISCHE.VBK. BAYER.HYPBK. DEGUSSA
## 1973-01-01    36.88 14.96 12.24           23.47        23.40  203.46
## 1973-01-02    37.44 15.43 11.95           22.92        23.22  206.85
## 1973-01-03    37.79 15.61 12.10           23.45        23.34  208.21
## 1973-01-04    37.86 15.85 12.71           23.66        23.49  210.11
## 1973-01-05    37.44 15.75 12.74           23.87        23.40  214.31
## 1973-01-08    37.79 15.80 12.74           24.07        23.46  216.68
##            DEUTSCHE.BANK CONTINENTAL VOLKSWAGEN DAIMLER.BENZ HOECHST
## 1973-01-01         22.29        8.54     134.06        16.97   13.77
## 1973-01-02         22.50        8.83     140.00        17.59   13.77
## 1973-01-03         22.86        8.78     144.53        17.79   14.22
## 1973-01-04         23.04        8.83     144.04        17.81   14.32
## 1973-01-05         23.29        8.73     139.92        17.73   14.23
## 1973-01-08         23.18        8.82     143.77        17.70   14.19
##            SIEMENS KARSTADT  LINDE THYSSEN MANNESMANN    MAN   RWE  INDEX
## 1973-01-01   20.76   359.55 135.95   67.47      84.97 117.92 11.68 536.36
## 1973-01-02   21.06   359.96 136.89   68.41      86.51 118.78 11.87 545.43
## 1973-01-03   21.29   361.99 139.59   67.47      87.75 124.95 12.03 552.46
## 1973-01-04   21.44   369.32 142.21   71.62      88.71 124.95 11.95 556.14
```

```
## 1973-01-05   21.48   368.50 143.71   71.24      88.63 127.29 12.03 557.44
## 1973-01-08   21.48   366.88 143.77   70.77      89.01 125.34 11.91 555.51
```

```
tail(data.all)  ##And always look at data
```

```
##                     X ALLIANZ.HLDG. COMMERZBANK DRESDNER.BANK   BMW
## 1996-07-16 1996-07-16          2550       323.0         38.15 843.5
## 1996-07-17 1996-07-17          2572       331.0         38.35 845.2
## 1996-07-18 1996-07-18          2619       335.0         39.60 844.0
## 1996-07-19 1996-07-19          2678       336.8         39.50 847.5
## 1996-07-22 1996-07-22          2632       336.8         39.00 844.0
## 1996-07-23 1996-07-23          2622       337.5         39.20 844.0
##            SCHERING  BASF BAYER BAYERISCHE.VBK. BAYER.HYPBK. DEGUSSA
## 1996-07-16    101.0 41.00 50.45           47.45        40.20   498.0
## 1996-07-17    102.5 41.87 50.92           48.08        40.55   503.2
## 1996-07-18    101.2 41.86 52.00           49.05        41.48   507.5
## 1996-07-19    102.9 42.10 51.85           49.48        41.92   506.0
## 1996-07-22    101.8 40.70 50.60           49.40        41.40   501.0
## 1996-07-23    102.0 40.15 50.25           49.88        41.55   499.0
##            DEUTSCHE.BANK CONTINENTAL VOLKSWAGEN DAIMLER.BENZ HOECHST
## 1996-07-16         72.10       23.00     531.00        78.45   49.85
## 1996-07-17         72.86       23.63     539.00        79.30   50.30
## 1996-07-18         74.30       24.11     528.50        78.00   50.50
## 1996-07-19         74.90       24.18     531.00        78.25   50.70
## 1996-07-22         73.60       24.10     522.25        77.48   49.20
## 1996-07-23         73.70       24.15     515.00        77.35   48.35
##            SIEMENS KARSTADT LINDE THYSSEN MANNESMANN   MAN   RWE   INDEX
## 1996-07-16   78.75    544.0   923   274.0      536.0 373.0 54.20 2469.79
## 1996-07-17   78.79    554.0   925   273.1      542.0 374.5 54.40 2497.19
## 1996-07-18   77.61    543.0   920   271.0      536.7 369.0 55.00 2506.22
## 1996-07-19   77.24    543.0   932   271.9      535.3 369.5 54.33 2520.19
## 1996-07-22   76.49    540.0   931   268.1      529.5 364.0 52.90 2482.40
## 1996-07-23   76.90    539.5   935   265.5      530.5 360.0 53.15 2475.07
```

With this machinery in hand we can subset the data by starting and ending date as well as the choice of `RWE` and `THYSSEN`.

```
# Subset the data using a start and
# end date
start.date <- "1975-06-02"
end.date <- "1990-12-30"
## First column looks for filtered
## dates, second and third columns
## pull out prices
price <- data.all[start.date <= date &
    date <= end.date, c("RWE", "THYSSEN")]
```

```
## We add a check to ensure that price
## is a matrix and that ncol will work
if (!is.matrix(price)) price <- rbind(price,
    deparse.level = 0L)
str(price)
```

```
## 'data.frame':    4065 obs. of  2 variables:
##  $ RWE    : num  8.96 9.2 9.16 9.2 9.36 9.24 9.12 9.08 9.04 8.99 ...
##  $ THYSSEN: num  69.8 70.8 69.8 68.9 68.8 ...
```

```
head(price)  ## show the beginning
```

```
##              RWE THYSSEN
## 1975-06-02 8.96   69.82
## 1975-06-03 9.20   70.77
## 1975-06-04 9.16   69.82
## 1975-06-05 9.20   68.88
## 1975-06-06 9.36   68.79
## 1975-06-09 9.24   67.94
```

```
tail(price)  ## and the end
```

```
##              RWE THYSSEN
## 1990-12-21 36.36   187.5
## 1990-12-24 36.36   187.5
## 1990-12-25 36.36   187.5
## 1990-12-26 36.36   187.5
## 1990-12-27 36.28   186.5
## 1990-12-28 35.75   184.5
```

The code before the `str`, `head`, and `tail` filters the price data by start and end dates. We could also perform this head and tail work using the following code.

```
(end.idx <- dim(price)[1])
```

```
## [1] 4065
```

```
(price.2 <- rbind(price[1:5, ], price[(end.idx -
    4):end.idx, ]))
```

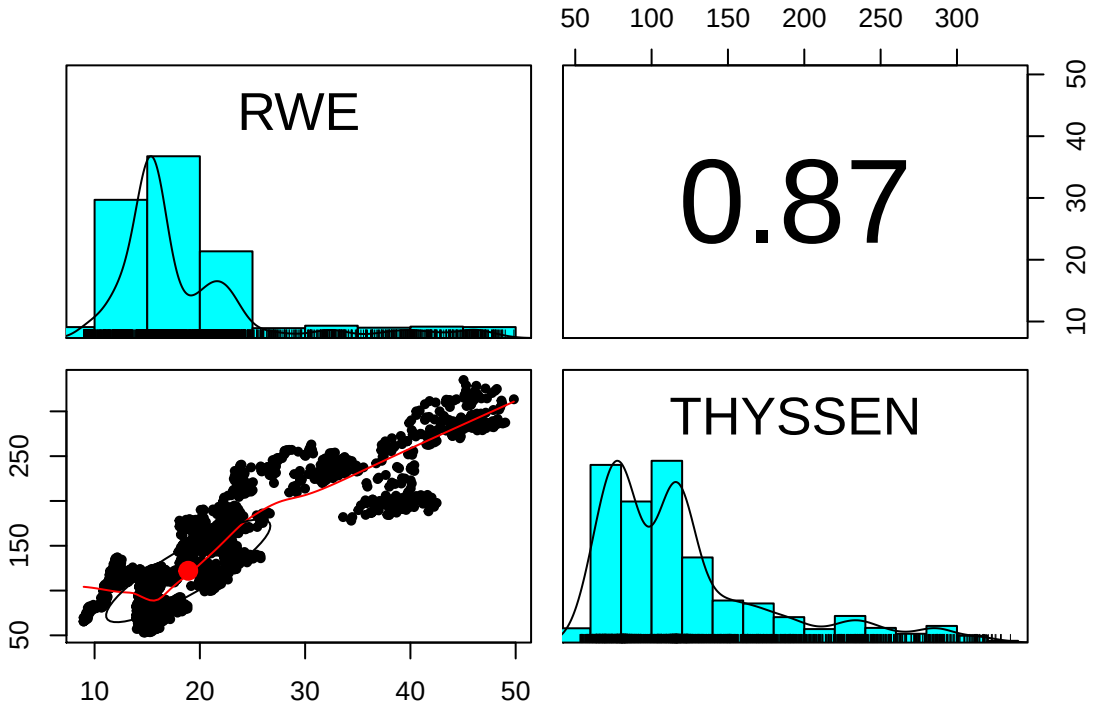
```
##              RWE THYSSEN
## 1975-06-02  8.96   69.82
## 1975-06-03  9.20   70.77
## 1975-06-04  9.16   69.82
## 1975-06-05  9.20   68.88
## 1975-06-06  9.36   68.79
## 1990-12-24 36.36  187.50
## 1990-12-25 36.36  187.50
```

```
## 1990-12-26 36.36  187.50
## 1990-12-27 36.28  186.50
## 1990-12-28 35.75  184.50
```

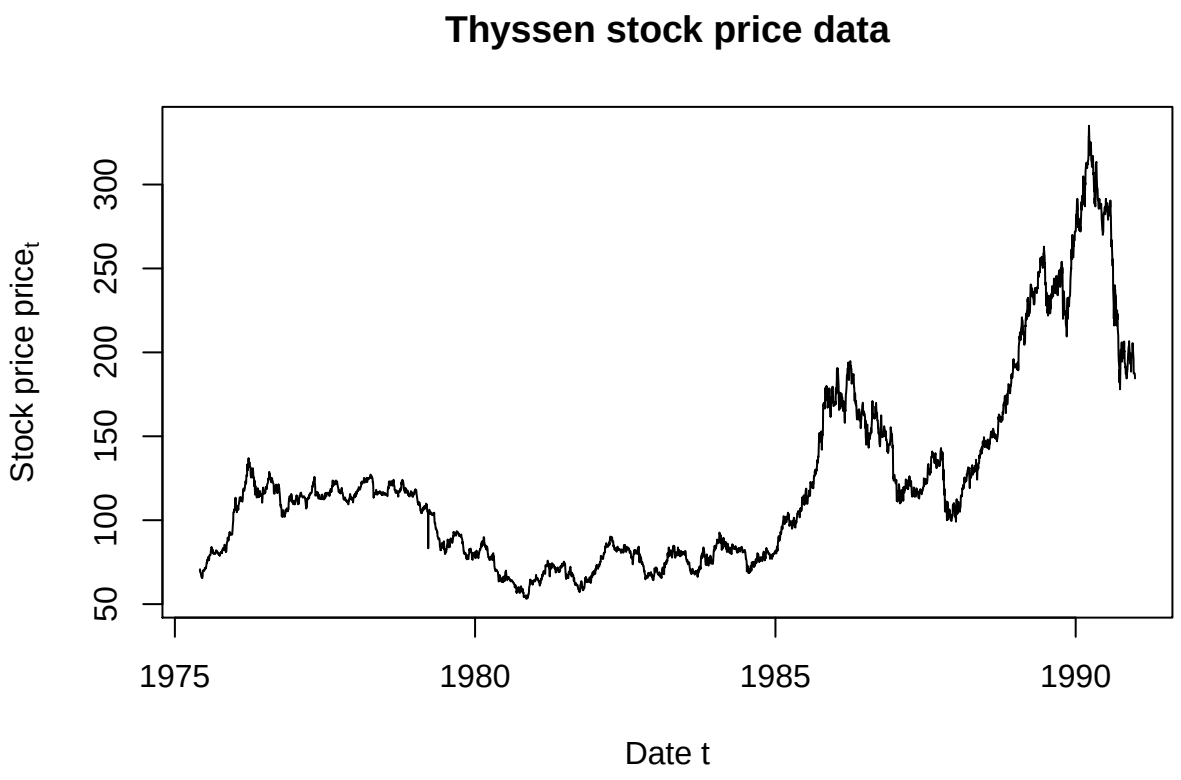
6.4 $ Try this exercise

Now let's really explore this data. The library `psych` has a prefabricated scatter plot histogram matrix we can use. With this composite plot we can examine historical relationships between the two risk factors as well as the shape of the risk factors themselves. We can also use this device to look at dependent simulations. After the scatter plots, we then look at the time series plots of the two factors.

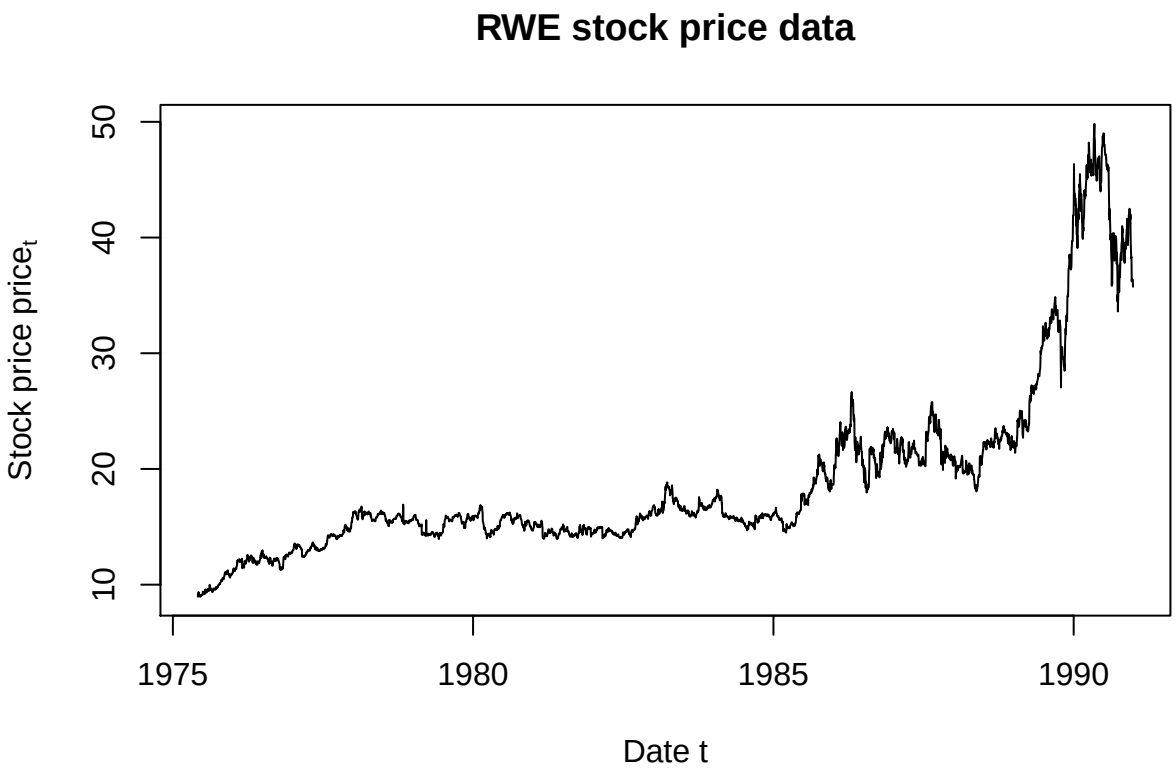
```
#Use scatter plots of the two price series along with their histograms to examine the
library(psych)
pairs.panels(price)
```




```
price.rownames <- rownames(price)
plot(as.Date(price.rownames), price[,"THYSSEN"], type="l",
     main="Thyssen stock price data", ## title
     xlab="Date t", ## x-axis label
     ylab=expression(Stock~price~price[t])) ## y-axis label
```




```
plot(as.Date(price.rownames), price[,"RWE"], type="l",
     main="RWE stock price data", ## title
     xlab="Date t", ## x-axis label
     ylab=expression(Stock~price~price[t])) ## y-axis label
```



The `pairs.panel` plot displays a matrix of interactions between `RWE` and `THYSSEN`. Price levels are interesting but, as we have seen, are not stable predictors. Let's transform them to returns next.

6.5 Now to the Matter at Hand

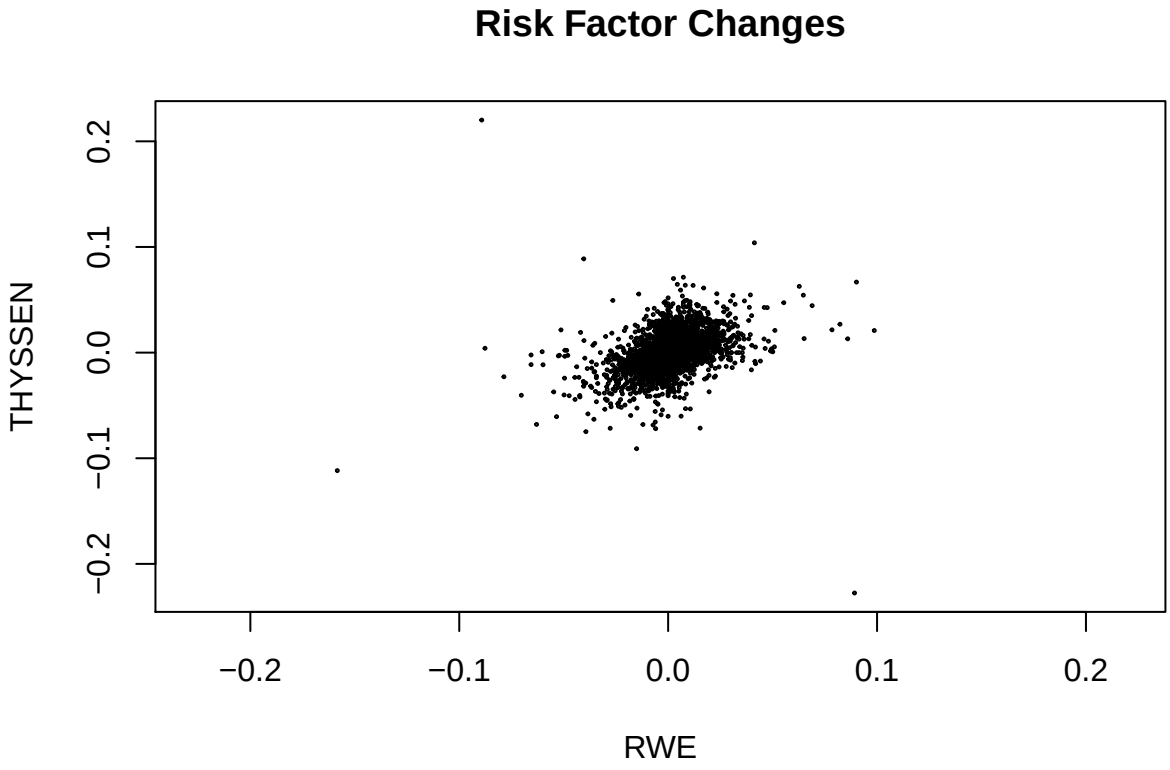
Now to the matter at hand: *value at risk* and *expected shortfall*. These two measures are based on the quantiles of losses attributable to risk factors. Value at risk is the quantile at an α level of tolerance. Expected shortfall is the mean of the distribution beyond the value at risk threshold.

To get losses attributable to market risk factors we compute log price differences (also called log price relatives). These can be interpreted as returns, or simply as percentage changes, in the risk factor prices. A plot lets us examine the results.

```
# Here we can compute two items
# together: log price differences,
# and their range (to bound a plot)
return.range <- range(return.series <- apply(log(price),
    2, diff))  ## compute log-returns and range
return.range
```

```
## [1] -0.2275282  0.2201375
```

```
plot(return.series, xlim = return.range,
    ylim = return.range, main = "Risk Factor Changes",
    cex = 0.2)
```



Using the returns we can now compute loss. Weights are defined as the value of the positions in each risk factor. We can compute this as the notional times the last price. Remember we are talking about an input, electricity, and an output, steel. We form the margin:

$$Margin = price_{steel} \times tons - price_{power} \times [(rate_{MWh/tons} \times tons],$$

where the last term is the power to steel transformation rate that converts power prices $ per MWh to $ per ton.

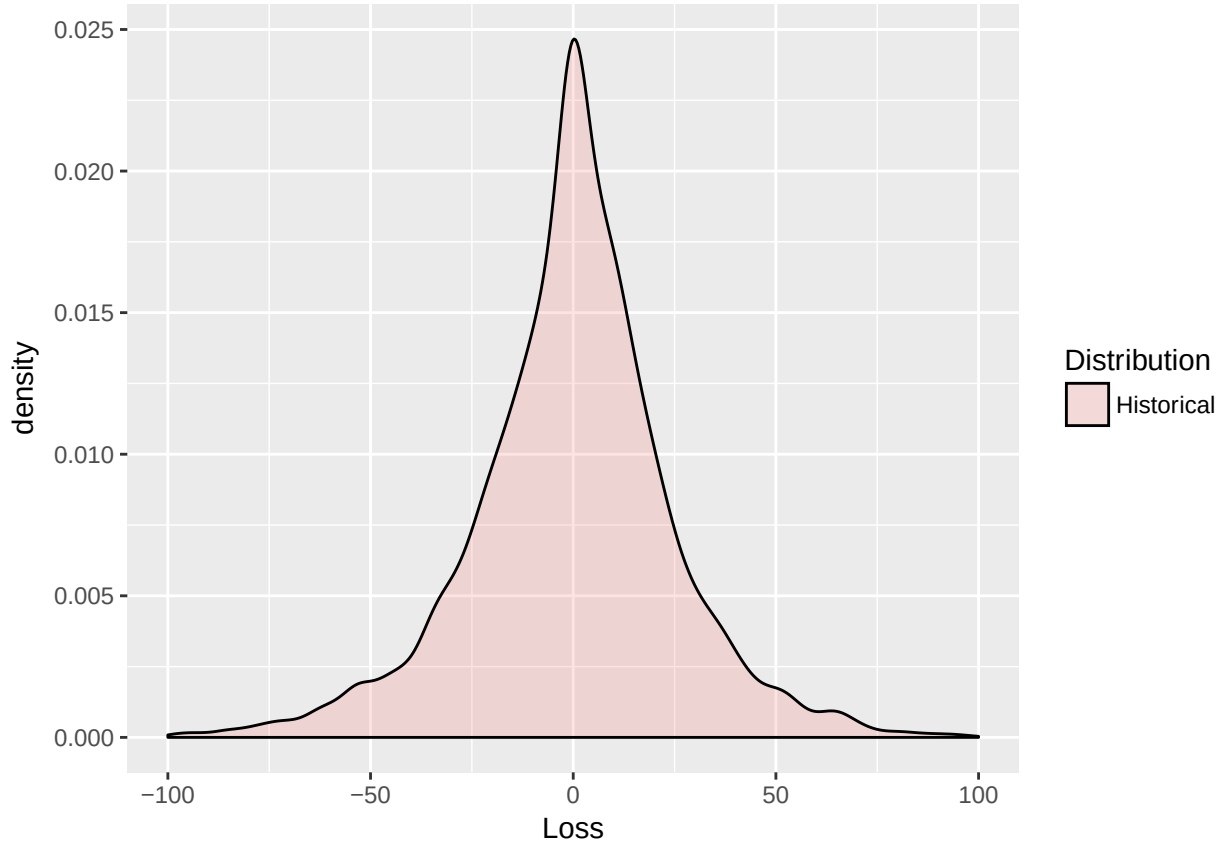
We convert prices to share prices and tons to equivalent values in terms of the number of shares. The naturally short position in power is equivalent to a negative number of shares (in the square brackets). The naturally long position in steel is equivalent to a positive number of shares. By naturally short we mean that power is an input, incurs a cost, and is demanded by the plant, and supplied by a third party. By naturally long we mean that steel is an output, earns a revenue, and demanded by a third party.

```
## Get last prices
price.last <- as.numeric(tail(price,
    n = 1))
## Specify the positions
position.rf <- c(-30, 10)
## And compute the position weights
w <- position.rf * price.last
## Fan these across the length and
## breadth of the risk factor series
weights.rf <- matrix(w, nrow = nrow(return.series),
    ncol = ncol(return.series), byrow = TRUE)
# We need to compute exp(x) - 1 for
# very small x: expm1 accomplishes
# this
loss.rf <- -rowSums(expm1(return.series) *
    weights.rf)
summary(loss.rf)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -545.8825  -13.3084    0.0000   -0.2467   13.1330  475.5576
```

We can visualize the data using this `ggplot2` routine which begins with the definition of a data frame.

```
loss.rf.df <- data.frame(Loss = loss.rf,
    Distribution = rep("Historical",
        each = length(loss.rf)))
require(ggplot2)
ggplot(loss.rf.df, aes(x = Loss, fill = Distribution)) +
    geom_density(alpha = 0.2) + xlim(-100,
    100)
```



The plot reveals some interesting deep and shallow outliers. The distribution is definitely very peaked. We use the base function `expm1` that computes the natural exponent of returns all minus 1.

$$e^r - 1$$

Some of these returns, or percentage price changes if you will, are very close to zero. High precision arithmetic is needed to get accurate calculations. The function `expm1` does this well.

Now we can get to estimating value at risk (VaR) and expected shortfal (ES). We set the tolerance level α, for example, equal to 95%. This would mean that a decision maker would not tolerate loss in more than 5% of all risk scenarios.

We define the VaR as the quantile for probability $\alpha \in (0, 1)$, as

$$VaR_\alpha(X) = inf\{x \in R : F(x) \geq \alpha\},$$

which means find the greatest lower bound of loss x (what the symbol inf = *infimum* means in English), such that the cumulative probability of x is greater than or equal to α.

Using the VaR_α definition we can also define ES as

$$ES_\alpha = E[X|X \geq VaR_\alpha],$$

where ES is "expected shortfall" and E is the expectation operator, also known as the "mean." Again, in English, the expected shortfall is the average of all losses greater than the loss at a VaR associated with probability α, and $ES \geq VaR$.

6.5.1 Try this example

1. Let's run the following lines of code.
2. We look up the `quantile` function in `R` and see that it matches the calculation for `VaR.hist`.
3. Using `VaR` we then calculate `ES` by only looking for losses greater than `VaR`.
4. We also look closely at the text annotations we can achieve in `ggplot2`.

Here is the code:

First the computations of VaR and ES:

```
# Simple Value at Risk
alpha.tolerance <- 0.99
(VaR.hist <- quantile(loss.rf, probs = alpha.tolerance,
    names = FALSE))
```

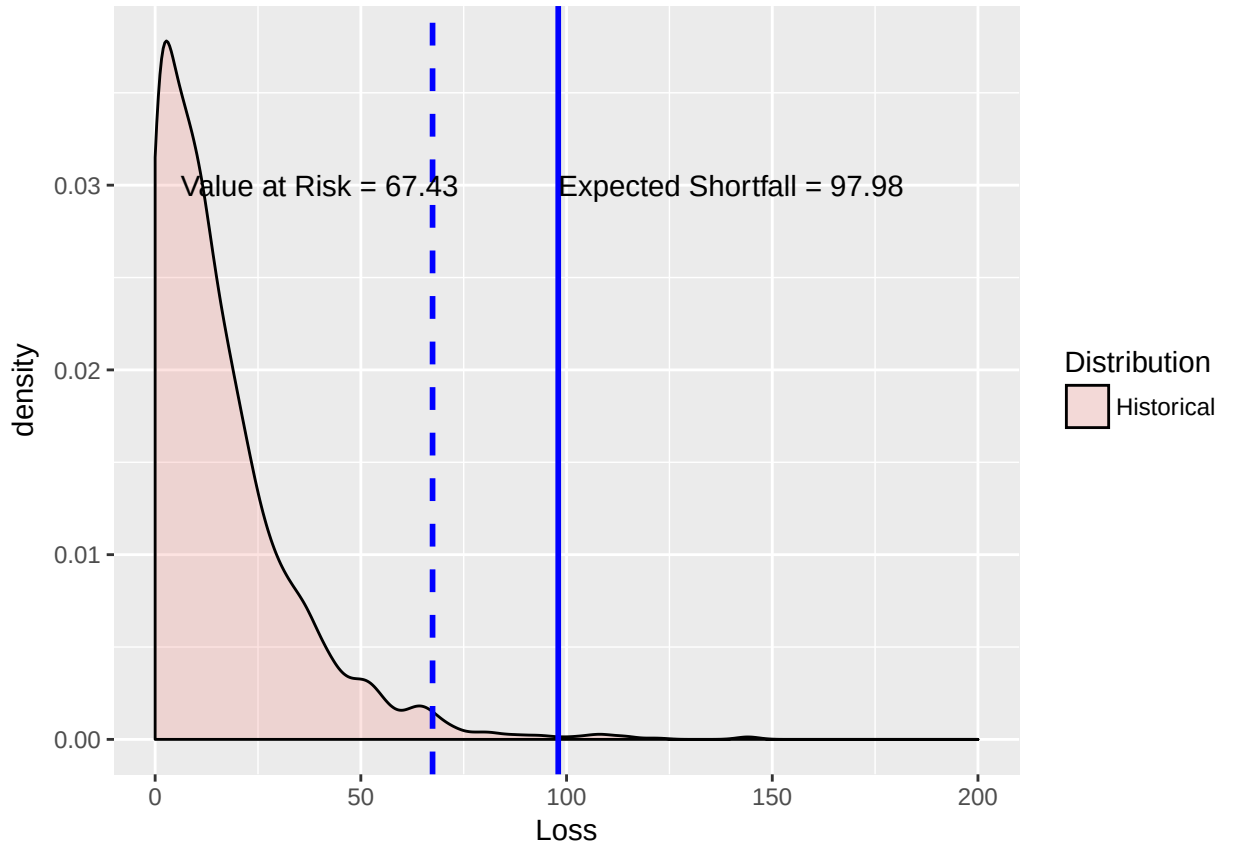
```
## [1] 67.43459
```

```
# Just as simple Expected shortfall
(ES.hist <- mean(loss.rf[loss.rf > VaR.hist]))
```

```
## [1] 97.97649
```

Next we set up the text and plotting environment.

```
VaR.text <- paste("Value at Risk =",
    round(VaR.hist, 2))
ES.text <- paste("Expected Shortfall =",
    round(ES.hist, 2))
ggplot(loss.rf.df, aes(x = Loss, fill = Distribution)) +
    geom_density(alpha = 0.2) + geom_vline(aes(xintercept = VaR.hist),
    linetype = "dashed", size = 1, color = "blue") +
    geom_vline(aes(xintercept = ES.hist),
        size = 1, color = "blue") + xlim(0,
    200) + annotate("text", x = 40, y = 0.03,
    label = VaR.text) + annotate("text",
    x = 140, y = 0.03, label = ES.text)
```

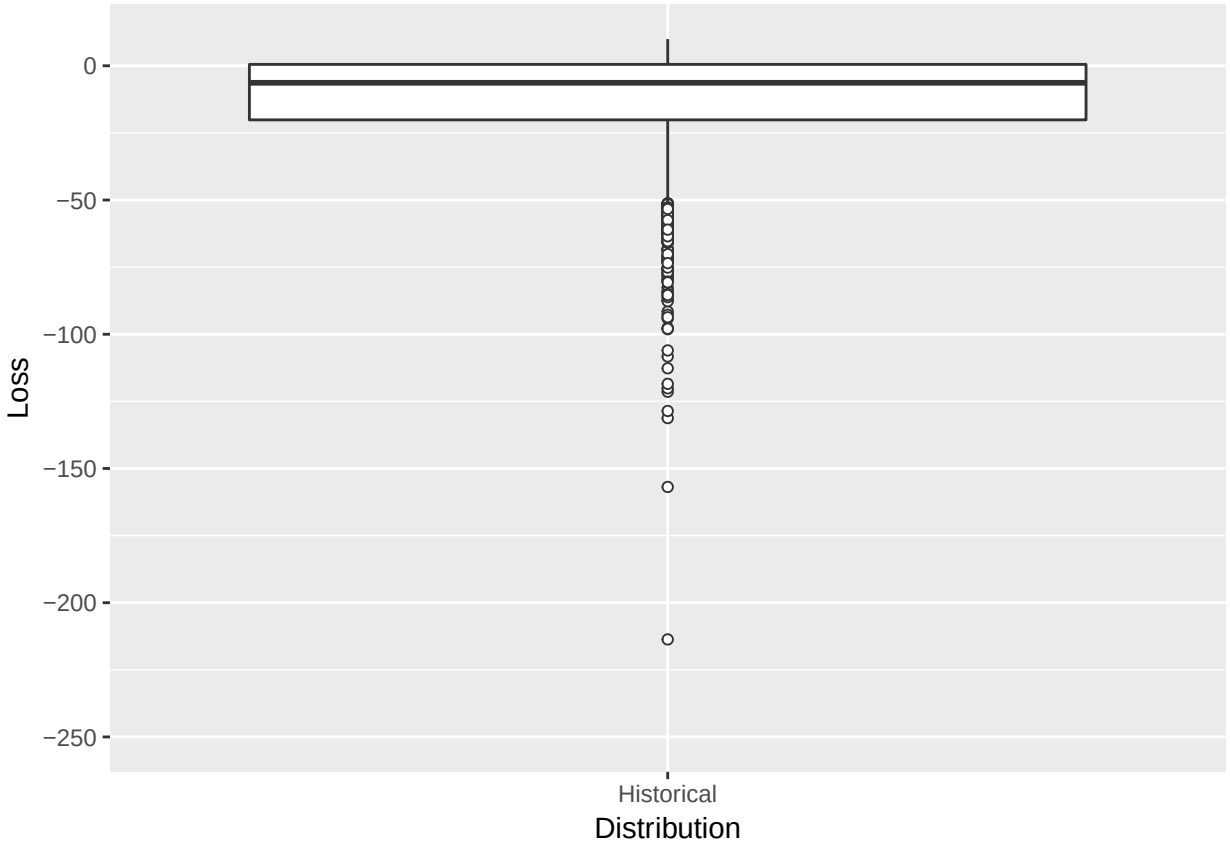


We see that `ES` is much bigger than `VaR` but also much less than the maximum historical loss.

> One note: `VaR` is computed as a pre-event indicator beyond a loss of `0` in this example. Many applications of this metric center loss at the median loss. Thus, loss would be computed as gross loss minus the median (50th percentile of loss).

A box plot might also help us visualize the results without resorting to a probability distribution function.

```
ggplot(loss.rf.df, aes(x = Distribution,
    y = Loss)) + geom_boxplot(outlier.size = 1.5,
    outlier.shape = 21) + ylim(-250,
    10)
```



This box plot might look better with more than one distribution. So far we simply let history speak for itself. We did not assume anything at all about the shape of the data. We just used the empirical record be the shape. In what follows let's start to put some different shapes into the loss potential of our tolling agreement.

6.6 Carl Friedrich Gauss, I Presume…

What we just did was the classic historical simulation technique for computing tail risk measures. Historical simulation is a "nonparametric" technique, since there is no estimation of parameters conditional on a distribution. Only history, unadorned, informs risk measurement. Now we shift gears into the parametric work of Gauss: Gaussian, Generalized Pareto, and as an exercise Gossett's (Student's t) distributions.

Carl Friedrich Gauss is often credited with the discovery of the normal distribution. So we tack his name often enough to the normal distribution. This distribution has a crucial role in quantitative risk and finance. It is often the basis for most derivative pricing models and for simulation of risk factors in general. It does not exhibit thick tails, and definitely is not skewed or peaked. This distribution definitely does not describe volatility clustering we observe in most financial and commodity time series. Nevertheless, it is otherwise ubiquitous, if only as a benchmark (like "perfect competition" or "efficient markets").

With just a little of math here, we can define the Gaussian (normal) distribution function. If x is a uniformly distributed random variable, then

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

is the probability density function of the normally distributed x with mean μ and standard deviation σ.

"Halfway"" between the normal Gaussian distribution and Student's t is the chi-square, χ^2, distribution. We define χ^2 as the distribution of the sum of the squared normal random variables x with density function and k degrees of freedom for $x > 0$:

$$f(x) = \frac{x^{(k/2-1)} e^{-x/2}}{2^{k/2}\Gamma(\frac{k}{2})}$$

and 0 otherwise. The "degrees of freedom" are the number of normal distributions used to create a chi-square variate.

Now on to Student's t distribution which is defined in terms of the Gaussian and chi-square distributions as the ratio of a Gaussian random variate to the square root of a chi-squared random variate. Student (a pseudonym for William Sealy Gossett) will have thicker tails but also the same symmetry as the normal curve. (Lookup this curve in Wikipedia among other references.)

Here is a quick comparison of the standard Gaussian and the Student's t distributions. The functions `rnorm` and `rt` generate Gaussian and Student's t variates, respectively. The functions `qnorm` and `qt` compute the distance from the mean (probability = 50%) for a given probability, here stored in `alpha.tolerance`.

```
library(mvtnorm)  ## Allows us to generate Gaussian and Student-t variates
library(ggplot2)
set.seed(1016)
n.sim <- 1000
z <- rnorm(n.sim)
t <- rt(n.sim, df = 5)
alpha.tolerance <- 0.95
(z.threshold <- qnorm(alpha.tolerance))
```

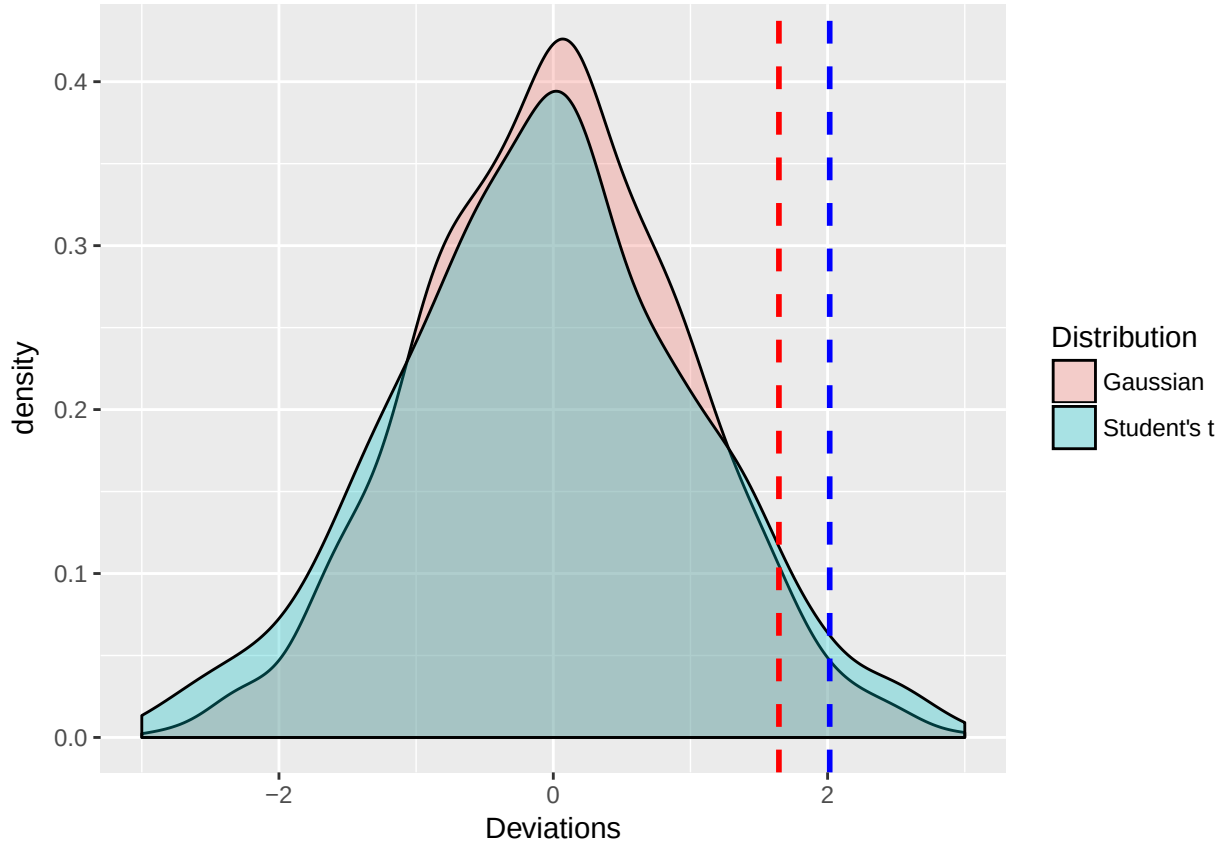
```
## [1] 1.644854
```

```
(t.threshold <- qt(alpha.tolerance, df = 5))
```

```
## [1] 2.015048
```

Now we make a data frame and plot with `ggplot`:

```
zt.df <- data.frame(Deviations = c(z,
    t), Distribution = rep(c("Gaussian",
    "Student's t"), each = n.sim))
ggplot(zt.df, aes(x = Deviations, fill = Distribution)) +
    geom_density(alpha = 0.3) + geom_vline(aes(xintercept = z.threshold),
    color = "red", linetype = "dashed",
    size = 1) + geom_vline(aes(xintercept = t.threshold),
    color = "blue", linetype = "dashed",
    size = 1) + xlim(-3, 3)
```

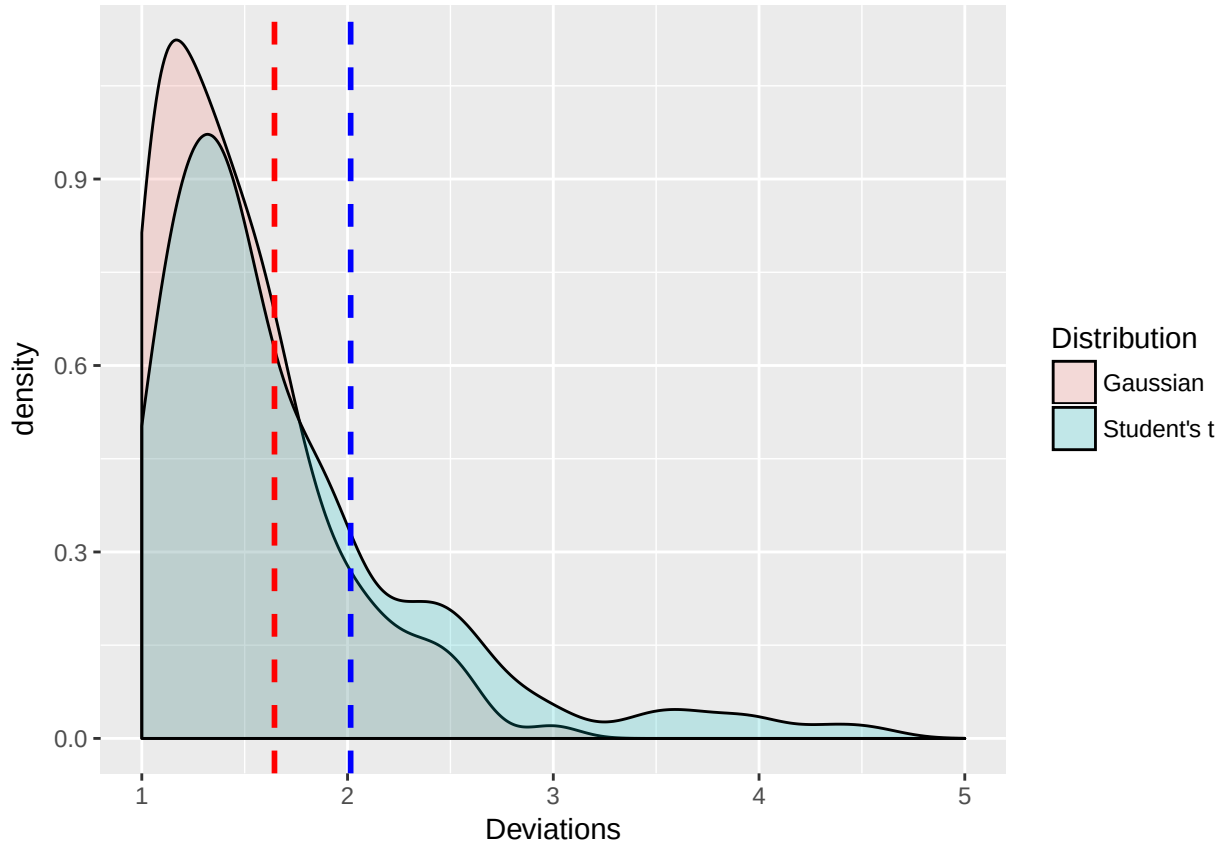


The `ggplots2` library allows us to control several aspects of the histogram including fill, borders, vertical lines, colors, and line types and thickness. The plot requires a data frame where we have indicated the type of distribution using a replication of character strings. We see the two distributions are nearly the same in appearance. But the Student's t tail is indeed thicker in the tail as the blue `t` density overtakes the red `z` density. This is numerically evident as the `t.threshold` is $>$ than the `z.threshold` for a cumulative probability of 95%, the 95th quantile.

6.6.1 Try this example

Let's zoom in on the right tail of the distribution with the `xlim` facet.

```
ggplot(zt.df, aes(x = Deviations, fill = Distribution)) +
    geom_density(alpha = 0.2) + geom_vline(aes(xintercept = z.threshold),
    color = "red", linetype = "dashed",
    size = 1) + geom_vline(aes(xintercept = t.threshold),
    color = "blue", linetype = "dashed",
    size = 1) + xlim(1, 5)
```



Interesting digression! But not really not too far off the mark. The thresholds are the same with two standard risk measures, scaled for particular risk factors and positions. We have simulated two different *values at risk*.

6.7 Back to the Future

Let's remember where the returns (as changes) in each risk factor come from. Also, we will extract the last price for use below.

```
# Again computing returns as changes
# in the risk factors
return.series <- apply(log(price), 2,
    diff)  ## compute risk-factor changes
price.last <- as.numeric(tail(price,
    n = 1))  ## reserve last price
```

Again to emphasize what constitutes this data, we specify the notional exposure. These are number of shares of stock, number of $1 million contracts of futures, or volumetric contract sizes, e.g., MMBtus or boe. All of these work for us given the that price is dimensioned relative to the notional dimension.

So if the risk factors are oil and natural gas prices, then we should use a common volumetric equivalent such as Btu (energy content) or boe (barrel of oil equivalent for volume). Position weights are then calculated as position times the last available price.

First, we can set the weights directly and a little more simply than before since we do not need to simulate historically.

```
## Specify the positions
position.rf <- c(-30, 10)  ## As before
## And compute the position weights
## directly again as before
(w <- position.rf * price.last)
```

```
## [1] -1072.5  1845.0
```

Second, we estimate the mean vector and the variance-covariance matrix, the two major inputs to the simulation of normal risk factor changes. Here we use a purely parametric approach.

```
mu.hat <- colMeans(return.series)  ## Mean vector mu; estimated = hat
Sigma.hat <- var(return.series)  ## Variance-covariance matrix Sigma
(loss.mean <- -sum(w * mu.hat))  ## Mean loss
```

```
## [1] -0.07596846
```

```
(loss.stdev <- sqrt(t(w) %*% Sigma.hat %*%
    w))  ## Standard deviation of loss
```

```
##          [,1]
## [1,] 28.4431
```

Third, we set the level of risk tolerance α. Then let's calculate `VaR` and `ES`:

```
# Compute VaR and ES and return
alpha.tolerance <- 0.95
q.alpha <- qnorm(alpha.tolerance)
(VaR.varcov <- loss.mean + loss.stdev *
```

```
    q.alpha)
```

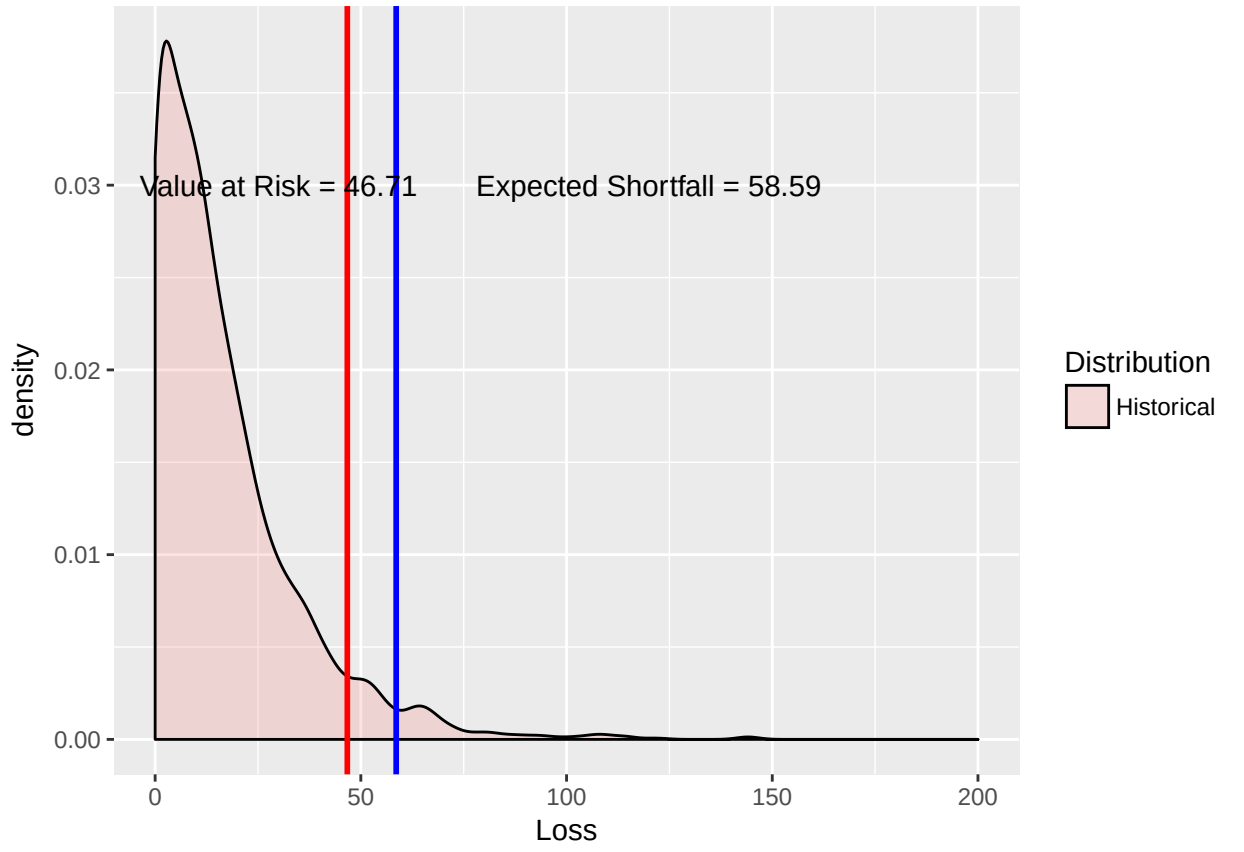
```
##          [,1]
## [1,] 46.70877
```

```
(ES.varcov <- loss.mean + loss.stdev *
    dnorm(q.alpha)/(1 - alpha.tolerance))
```

```
##          [,1]
## [1,] 58.59398
```

and plot

```
VaR.text <- paste("Value at Risk =",
    round(VaR.varcov, 2))
ES.text <- paste("Expected Shortfall =",
    round(ES.varcov, 2))
ggplot(loss.rf.df, aes(x = Loss, fill = Distribution)) +
    geom_density(alpha = 0.2) + geom_vline(aes(xintercept = VaR.varcov),
    colour = "red", size = 1) + geom_vline(aes(xintercept = ES.varcov),
    colour = "blue", size = 1) + xlim(0,
    200) + annotate("text", x = 30, y = 0.03,
    label = VaR.text) + annotate("text",
    x = 120, y = 0.03, label = ES.text)
```



6.8 Try this example

Suppose it takes less electricity to make steel than we thought above. We can model this by changing the positions to `(-20, 10)`. Let's redo steps 1, 2, and 3 (this begs for a function).

First, we can set the weights directly a little more simply than before since we do not need to simulate historically.

```
## Specify the positions
position.rf <- c(-20, 10)  ## As before
## And compute the position weights
## directly again as before
(w <- position.rf * price.last)
```

```
## [1] -715 1845
```

Second, estimate the mean vector and the variance-covariance matrix, the two major inputs to the simulation of normal risk factor changes. Here we use a purely parametric approach.

```
mu.hat <- colMeans(return.series)  ## Mean vector mu; estimated = hat
Sigma.hat <- var(return.series)  ## Variance-covariance matrix Sigma
(loss.mean <- -sum(w * mu.hat))  ## Mean loss
```

```
## [1] -0.1976962
```

```
(loss.stdev <- sqrt(t(w) %*% Sigma.hat %*%
    w))  ## Standard deviation of loss
```

```
##          [,1]
## [1,] 28.53755
```

Third, set the level of risk tolerance α. Then calculate `VaR` and `ES`:

```
# Compute VaR and ES and return
alpha.tolerance <- 0.95
q.alpha <- qnorm(alpha.tolerance)
(VaR.varcov <- loss.mean + loss.stdev *
    q.alpha)
```

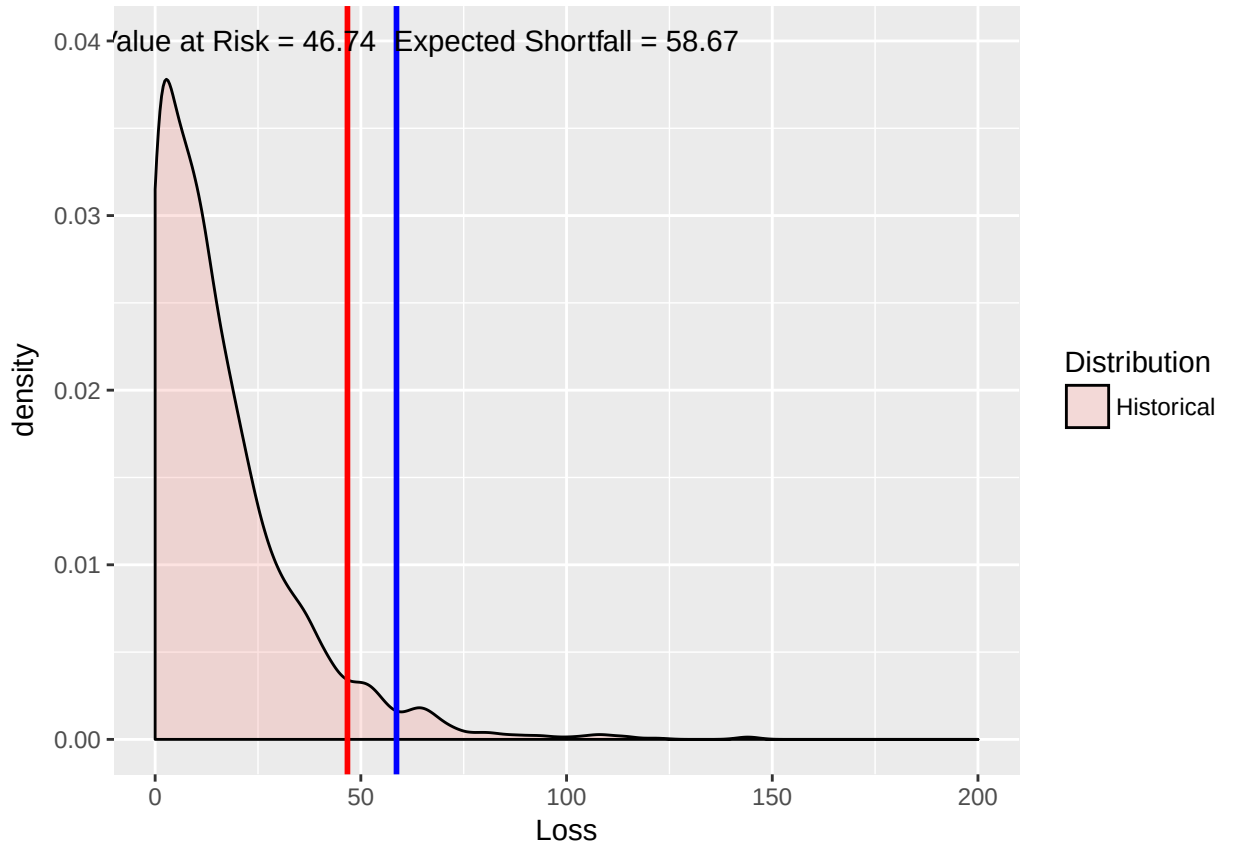
```
##          [,1]
## [1,] 46.7424
```

```
(ES.varcov <- loss.mean + loss.stdev *
    dnorm(q.alpha)/(1 - alpha.tolerance))
```

```
##          [,1]
## [1,] 58.66708
```

... and plot

```
VaR.text <- paste("Value at Risk =",
    round(VaR.varcov, 2))
ES.text <- paste("Expected Shortfall =",
    round(ES.varcov, 2))
ggplot(loss.rf.df, aes(x = Loss, fill = Distribution)) +
    geom_density(alpha = 0.2) + geom_vline(aes(xintercept = VaR.varcov),
    colour = "red", size = 1) + geom_vline(aes(xintercept = ES.varcov),
    colour = "blue", size = 1) + xlim(0,
    200) + annotate("text", x = 20, y = 0.04,
    label = VaR.text) + annotate("text",
    x = 100, y = 0.04, label = ES.text)
```



Aesthetics may overtake us here as we really shoulf change the `x` and `y` `annotate` coordinates to fit on the graph properly.

So ends the story of the main method used for years and embodied in the famous 4:15 (pm, that is) risk report at JP Morgan. Also we remember the loss that we simulate here is an operating income loss, which after taxes and other adjustments, and, say, a one-year horizon, means a loss of additions to retained earnings. Book equity drops and so will market capitalization on average.

6.9 Let's Go to Extremes

All along we have been stylizing financial returns, including commodities and exchange rates, as skewed and with thick tails. We next go on to investigate these tails further using an extreme tail distribution called the Generalized Pareto Distribution (GPD). For very high thresholds, such as value at risk and expected shortfall, GPD not only well describes behavior in excess of the threshold, but the mean excess over the threshold is linear in the threshold. From this we get more intuition around the use of expected shortfall as a coherent risk measure. In recent years markets well exceeded all Gaussian and Student's t thresholds.

For a random variate x, this distribution is defined for shape parameters $\xi \geq 0$ as:

$$g(x; \xi \geq 0) = 1 - (1 + x\xi/\beta)^{-1/\xi}$$

and when the shape parameter $\xi = 0$, the GPD becomes the exponential distribution dependent only on the scale parameter β:

$$g(x; \xi = 0) = 1 - exp(-x/\beta).$$

There is one reason for GPD's notoriety. If u is an upper (very high) threshold, then the excess of threshold function for the GPD is

$$e(u) = \frac{\beta + \xi u}{1 - \xi}.$$

This simple measure is *linear* in thresholds. It will allow us to visualize where rare events begin (see McNeil, Embrechts, and Frei (2015, chapter 5)). We will come back to this property when we look at operational loss data in a few chapters.

Let's use the `QRM` library to help us find the optimal fit of losses to the parameters. The `fit.GPD` function will do this for us.

```
library(QRM)
u <- quantile(loss.rf, alpha.tolerance,
    names = FALSE)
fit <- fit.GPD(loss.rf, threshold = u)  ## Fit GPD to the excesses
(xi.hat <- fit$par.ests[["xi"]])  ## fitted xi
```

```
## [1] 0.1928437
```

```
(beta.hat <- fit$par.ests[["beta"]])  ## fitted beta
```

```
## [1] 15.89524
```

Now for the closed form (no random variate simulation!) using the McNeil, Embrechts, and Frei (2015, chapter 5) formulae:

```
## Pull out the losses over the
## threshold and compute excess over
## the threshold
loss.excess <- loss.rf[loss.rf > u] -
    u  ## compute the excesses over u
n.relative.excess <- length(loss.excess)/length(loss.rf)  ## = N_u/n
(VaR.gpd <- u + (beta.hat/xi.hat) * (((1 -
    alpha.tolerance)/n.relative.excess)^(-xi.hat) -
    1))
```

```
## [1] 40.40925
```

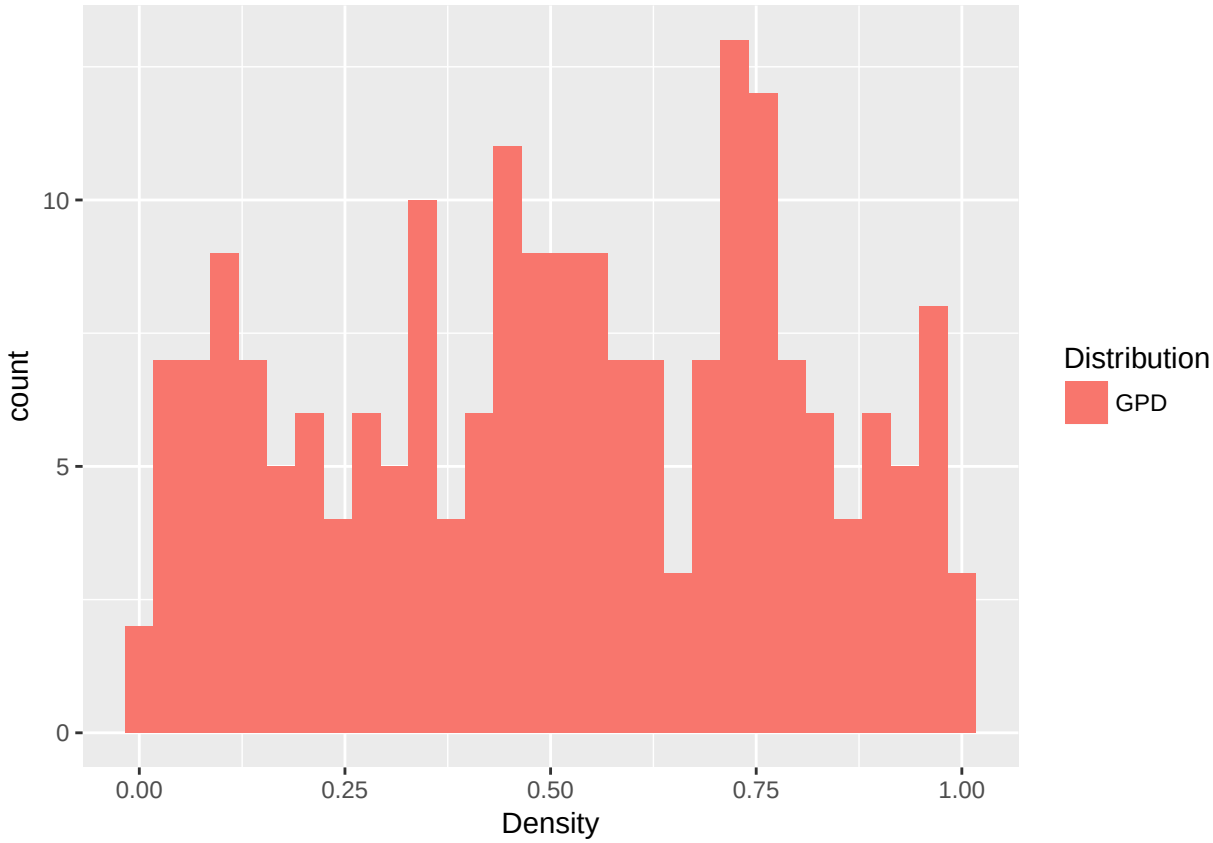
```
(ES.gpd <- (VaR.gpd + beta.hat - xi.hat *
    u)/(1 - xi.hat))
```

```
## [1] 60.11707
```

6.9.1 Try this example

How good a fit to the data have we found? This plot should look roughly uniform since the GPD excess loss function is a linear function of thresholds `u`.

```
gpd.density <- pGPD(loss.excess, xi = xi.hat,
    beta = beta.hat)
gpd.density.df <- data.frame(Density = gpd.density,
    Distribution = rep("GPD", each = length(gpd.density)))  ## This should be U[0,1]
ggplot(gpd.density.df, aes(x = Density,
    fill = Distribution)) + geom_histogram()
```

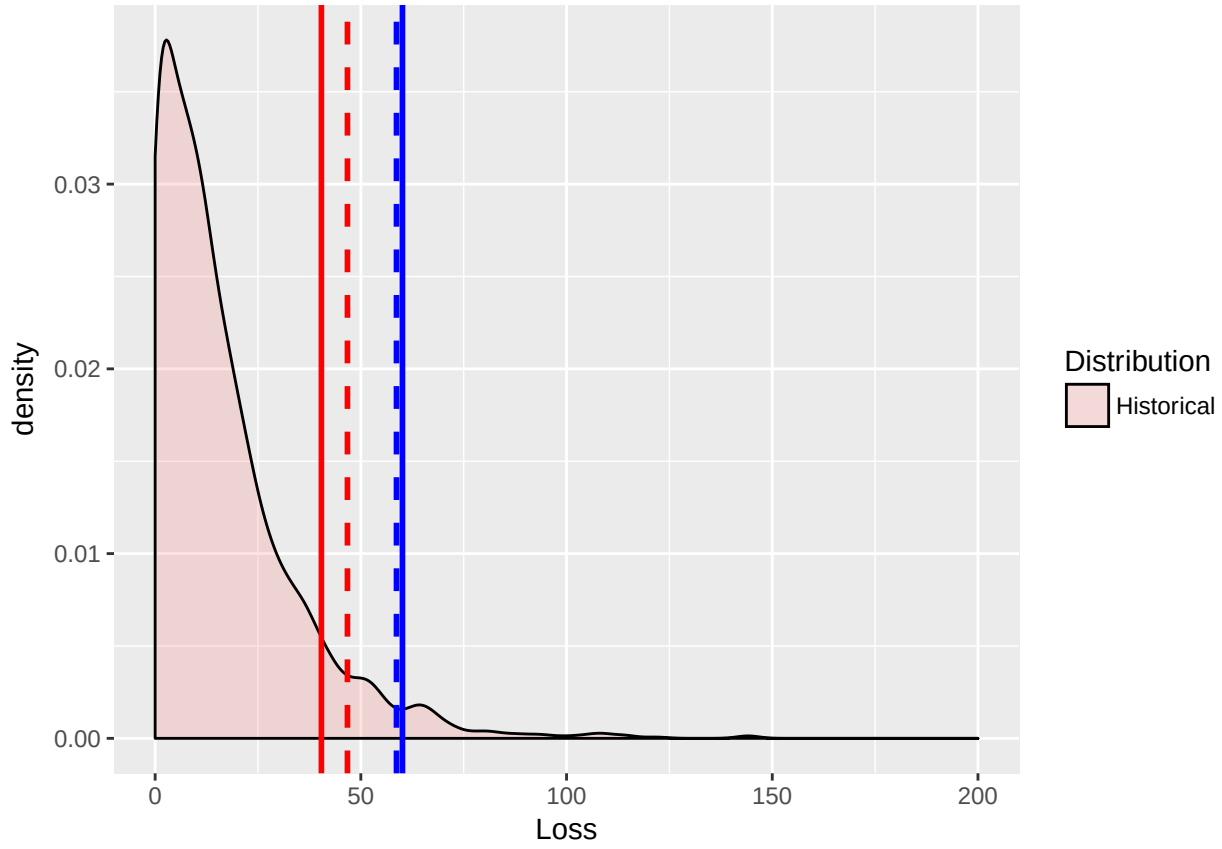


And it does look "uniform" enough (in a statistical sort of way as we perform eyeball econometrics again!).

6.10 All Together Now

Let's graph the historical simulation, variance-covariance and GPD results together.

```
loss.plot <- ggplot(loss.rf.df, aes(x = Loss,
    fill = Distribution)) + geom_density(alpha = 0.2)
loss.plot <- loss.plot + geom_vline(aes(xintercept = VaR.varcov),
    colour = "red", linetype = "dashed",
    size = 1)
loss.plot <- loss.plot + geom_vline(aes(xintercept = ES.varcov),
    colour = "blue", linetype = "dashed",
    size = 1)
loss.plot <- loss.plot + geom_vline(aes(xintercept = VaR.gpd),
    colour = "red", size = 1)
loss.plot <- loss.plot + geom_vline(aes(xintercept = ES.gpd),
    colour = "blue", size = 1)
loss.plot <- loss.plot + xlim(0, 200)
loss.plot
```



That was a lot. We will need our "mean over excess" knowledge when we get to operational risk. Actually we will be able to apply that to these risk measures for any kind of risk. But we will save ourselves for operational risk later. Someone might even annotate the graph…

6.11 Summary

Filtering - Math to `R` translation - Graphics - Normal and GPD distributions - VaR and ES - Loss distributions and mean over loss

6.12 Further Reading

6.13 Practice Laboratory

6.13.1 Practice laboratory #1

6.13.1.1 Problem

6.13.1.2 Questions

6.13.2 Practice laboratory #2

6.13.2.1 Problem

6.13.2.2 Questions

6.14 Project

6.14.1 Background

6.14.2 Data

6.14.3 Workflow

6.14.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.

- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.
- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.
- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

6.15 References

McNeill, Alexander J., Rudiger Frey, and Paul Embrechts. 2015. Quantitative Risk Management: Concepts, Techniques and Tools. Revised Edition. Princeton: Princeton University Press.

Ruppert, David and David S. Matteson. 2015. Statistics and Data Analysis for Financial Engineering with R Examples, Second Edition. New York: Springer.

Chapter 7

Credit Risk

7.1 Imagine This

Our company is trying to penetrate a new market. To do so it acquires several smaller competitors for your products and services. As we acquire the companies , we also acquire their customers…and now our customers' ability to pay us.

- Not only that, but yowe have also taken your NewCo's supply chain. Your company also has to contend with the credit worthiness of NewCo's vendors. If they default you don't get supplied, you can't produce, you can't fulfill your customers, they walk.
- Your CFO has handed you the job of organizing your accounts receivable, understanding your customers' paying patterns, and more importantly their defaulting patterns.

Some initial questions come to mind:

1. What are the key business questions you should ask about your customers' paying / defaulting patterns?
2. What systematic approach might you use to manage customer and counterparty credit risk?

Some ideas to answer these questions can be

1. Key business questions might be

- What customers and counterparties default more often than others?
- If customer default what can we recover?
- What is the total exposure that we experience at any given point in time?
- How far can we go with customers that might default?

2. Managing credit risk

- Set up a scoring system to accept new customers and track existing customers
- Monitor transitions of customers from one rating notch to another
- Build in early warning indicators of customer and counterparty default

- Build a playbook to manage the otherwise insurgent and unanticipated credit events that can overtake your customers and counterparties

Topics we got to in the last several chapters:

- Explored stylized fact of financial market data
- Learned just how insidious volatility really is
- Acquired new tools like `acf`, `pacf`, `ccf` to explore time series.

In this chapter on credit risk We Will Use actual transaction and credit migration data to examine relationships among default and explanations of credit-worthiness. We will also simulate default probabilities using markov chains. With this technology in hand we can then begin to Understand hazard rates and the probability of transitioning from one credit state to another. Then we have a first stab a predicting default and generating loss distributions for portfolios of credit exposures, as in the the Newco example of the accounts receivable credit portfolio.

7.2 New customers!

Not so fast! Let's load the credit profiles of our newly acquired customers. Here is what was collected these past few years:

```
firm.profile <- read.csv("data/creditfirmprofile.csv")
head(firm.profile)
```

```
##   id start.year default  wcTA  reTA ebitTA mktcapTL   sTA
## 1  1       2010       0 0.501 0.307  0.043    0.956 0.335
## 2  1       2011       0 0.550 0.320  0.050    1.060 0.330
## 3  1       2012       0 0.450 0.230  0.030    0.800 0.250
## 4  1       2013       0 0.310 0.190  0.030    0.390 0.250
## 5  1       2014       0 0.450 0.220  0.030    0.790 0.280
## 6  1       2015       0 0.460 0.220  0.030    1.290 0.320
```

Recorded for each of several years from 2006 through 2015 each firm's (customer's) indicator as to whether they defaulted or not (1 or 0).

```
summary(firm.profile)
```

```
##        id          start.year      default           wcTA        
##  Min.   :  1.0   Min.   :2006   Min.   :0.000   Min.   :-2.2400  
##  1st Qu.:192.0   1st Qu.:2009   1st Qu.:0.000   1st Qu.: 0.0300  
##  Median :358.0   Median :2011   Median :0.000   Median : 0.1200  
##  Mean   :356.3   Mean   :2011   Mean   :0.018   Mean   : 0.1426  
##  3rd Qu.:521.0   3rd Qu.:2013   3rd Qu.:0.000   3rd Qu.: 0.2400  
##  Max.   :830.0   Max.   :2015   Max.   :1.000   Max.   : 0.7700  
##       reTA             ebitTA           mktcapTL           sTA        
```

```
##  Min.   :-3.3100   Min.   :-0.59000   Min.   : 0.020   Min.   :0.0400
##  1st Qu.: 0.0900   1st Qu.: 0.04000   1st Qu.: 0.620   1st Qu.:0.1700
##  Median : 0.2200   Median : 0.05000   Median : 1.140   Median :0.2600
##  Mean   : 0.2104   Mean   : 0.05181   Mean   : 1.954   Mean   :0.3036
##  3rd Qu.: 0.3700   3rd Qu.: 0.07000   3rd Qu.: 2.240   3rd Qu.:0.3700
##  Max.   : 1.6400   Max.   : 0.20000   Max.   :60.610   Max.   :5.0100
```

Several risk factors can contribute to credit risk:

1. Working Capital risk is measured by the Working Capital / Total Assets ratio `wcTA`. When this ratio is zero, current assets are matched by current liabilities. When positive (negative), current assets are greater (lesser) than current liabilities. The risk is that there are very large current assets of low quality to feed revenue cash flow. Or the risk is that there are high current liabilities balances creating a hole in the cash conversion cycle and thus a possibility of low than expected cash flow.

2. Internal Funding risk is measured by the Retained Earnings / Total Assets ratio `reTA`. Retained Earnings measures the amount of net income plowed back into the organization. High ratios signal strong capability to fund projects with internally generated cash flow. The risk is that if the organization faces extreme changes in the market place, there is not enough internally generated funding to manage the change.

3. Asset Profitability risk is measured by EBIT / Total Assets ratio. This is the return on assets used to value the organization. The risk is that

- EBIT is too low or even negative and thus the assets are not productively reaching revenue markets or efficiently using the supply change, or both, all resulting in too low a cash flow to sustain operations and investor expectations, and

- This metric falls short of investors minimum required returns, and thus investors' expectations are dashed to the ground, they sell your stock, and with supply and demand simplicity your stock price falls, along with your equity-based compensation.

4. Capital Structure risk is measured by the Market Value of Equity / Total Liabilities ratio `mktcapTL`. If this ratio deviates from industry norms, or if too low, then shareholder claims to cash flow, and thus control of funding for responding to market changes will be impaired. The risk is similar to Internal Fund Risk but carries the additional market perception that the organization is unwilling or unable to manage change.

5. Asset Efficiency risk is measured by the Sales / Total Assets ratio `sTA`. If too low then the organization risks two things: being able to support sales with assets, and the overburden of unproductive assets unable to support new projects through additions to retained earnings or in meeting liability committments.

Let's also load customer credit migration data. This data records the start rating, end rating and timing for each of 830 customers as their business, and the recession, affected them.

```
firm.migration <- read.csv("data/creditmigration.csv")
head(firm.migration)
```

```
##   id start.date start.rating end.date end.rating time start.year end.year
## 1  1      40541            6    42366          6 1825       2010     2015
## 2  2      41412            6    42366          6  954       2013     2015
## 3  3      40174            5    40479          6  305       2009     2010
## 4  3      40479            6    40905          5  426       2010     2011
## 5  3      40905            5    42366          5 1461       2011     2015
## 6  4      40905            5    41056          6  151       2011     2012
```

Notice that the dates are given in number of days from January 1, 1900. Ratings are numerical.

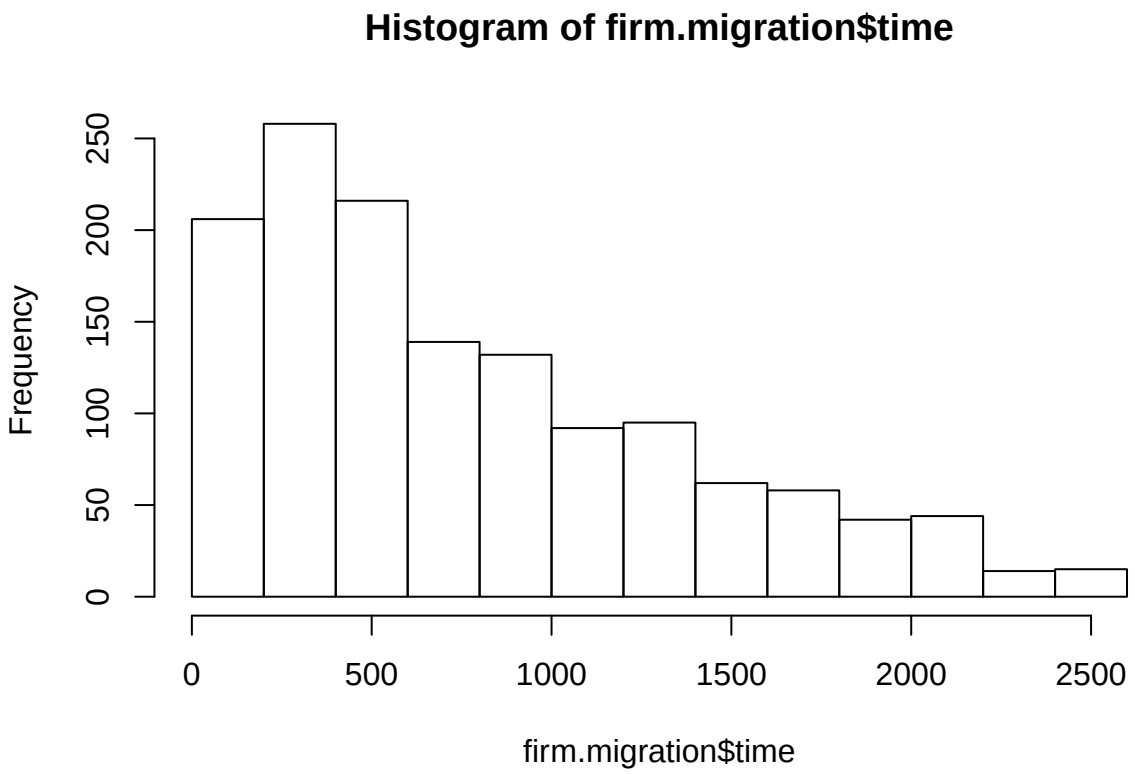
```
summary(firm.migration)
```

```
##        id          start.date     start.rating      end.date
##  Min.   :  1.0   Min.   :39951   Min.   :1.000   Min.   :39960
##  1st Qu.:230.0   1st Qu.:40418   1st Qu.:3.000   1st Qu.:41170
##  Median :430.0   Median :40905   Median :4.000   Median :42143
##  Mean   :421.8   Mean   :40973   Mean   :4.041   Mean   :41765
##  3rd Qu.:631.0   3rd Qu.:41421   3rd Qu.:5.000   3rd Qu.:42366
##  Max.   :830.0   Max.   :42366   Max.   :8.000   Max.   :42366
##  NA's   :1709    NA's   :1709    NA's   :1709    NA's   :1709
##    end.rating         time          start.year      end.year
##  Min.   :1.000   Min.   :   0.0   Min.   :2009   Min.   :2009
##  1st Qu.:3.000   1st Qu.: 344.0   1st Qu.:2010   1st Qu.:2012
##  Median :4.000   Median : 631.0   Median :2011   Median :2015
##  Mean   :4.121   Mean   : 791.5   Mean   :2012   Mean   :2014
##  3rd Qu.:5.000   3rd Qu.:1157.0   3rd Qu.:2013   3rd Qu.:2015
##  Max.   :8.000   Max.   :2415.0   Max.   :2015   Max.   :2015
##  NA's   :1709    NA's   :1709     NA's   :1709   NA's   :1709
```

```
firm.migration <- na.omit(firm.migration)
## firm.migration$start.rating <-
## levels(firm.migration$start.rating)
## firm.migration$end.rating <-
## levels(firm.migration$end.rating)
firm.migration$time <- as.numeric(firm.migration$time)
```

An interesting metric is in `firm.migration$time`. This field has records of the difference between `end.date` and `start.date` in days between start ratings and end ratings.

```
hist(firm.migration$time)
```



Let's now merge the two credit files by starting year. This will ("inner") join the data so we can see what customer conditions might be consistent with a rating change and rating dwell time. Two keys are `id` and `start.year`. The resulting data set will have records less than or equal to (if a perfect match) the maximum number of records (rows) in any input data set.

```
firm.credit <- merge(firm.profile, firm.migration,
    by = c("id", "start.year"))
head(firm.credit)
```

```
##    id start.year default  wcTA  reTA ebitTA mktcapTL   sTA start.date
## 1   1       2010       0 0.501 0.307  0.043    0.956 0.335      40541
## 2  10       2014       0 0.330 0.130  0.030    0.330 0.140      42001
## 3 100       2010       0 0.170 0.240  0.080    5.590 0.270      40510
## 4 100       2013       0 0.140 0.260  0.020    2.860 0.260      41635
## 5 106       2013       0 0.040 0.140  0.060    0.410 0.320      41605
## 6 106       2013       0 0.040 0.140  0.060    0.410 0.320      41421
##   start.rating end.date end.rating time end.year
## 1            6    42366          6 1825     2015
## 2            5    42366          5  365     2015
## 3            4    41635          3 1125     2013
## 4            3    42366          3  731     2015
```

```
## 5            5    42366          5  761     2015
## 6            6    41605          5  184     2013
```

```
dim(firm.credit)
```

```
## [1] 714  14
```

7.2.1 Try this exercise

The shape of `firm.migration$time` suggests a `gamma` or an `exponential` function. But before we go off on that goose chase, let's look at the inner-joined data to see potential differences in rating. Let's reuse this code from previous work:

```
library(dplyr)

## 1: filter to keep one state. Not
## needed (yet...)
pvt.table <- firm.credit  ## filter(firm.credit, xxx %in% 'NY')

## 2: set up data frame for by-group
## processing.
pvt.table <- group_by(pvt.table, default,
    end.rating)

## 3: calculate the three summary
## metrics
options(dplyr.width = Inf)  ## to display all columns
pvt.table <- summarise(pvt.table, time.avg = mean(time)/365,
    ebitTA.avg = mean(ebitTA), sTA.avg = mean(sTA))
```

Now we display the results in a nice table

```
knitr::kable(pvt.table)
```

default	end.rating	time.avg	ebitTA.avg	sTA.avg
0	1	3.021005	0.0491667	0.2633333
0	2	2.583007	0.0529762	0.3250000
0	3	2.468776	0.0520103	0.2904124
0	4	2.106548	0.0455495	0.2655495
0	5	1.912186	0.0501042	0.2883333
0	6	1.697821	0.0504886	0.3041477
0	7	1.672251	0.0494286	0.3308571
0	8	2.467123	0.0591667	0.3208333
1	1	2.413699	-0.0100000	0.4100000
1	2	2.497717	0.0216667	0.2683333
1	3	2.613699	0.0200000	0.2400000
1	6	2.242466	0.0350000	0.1150000
1	7	3.002740	0.0500000	0.3000000

Defaulting (`default` = 1)firms have very low EBIT returns on Total Assets as well as low Sales to Total Assets… as expected. They also spent a lot of time (in 365 day years) in rating 7 – equivalent to a "C" rating at S&P.

Now let's use the credit migration data to understand the probability of default as well as the probabilities of being in other ratings or migrating from one rating to another.

7.3 It Depends

Most interesting examples in probability have a little dependence added in: "If it rained yesterday, what is the probability it rains today?" We can use this idea to generate weather patterns and probabilities for some time in the future.

- In market risk, we can use this idea to generate the persistence of consumption spending, inflation, and the impact on zero coupon bond yields.
- In credit, dependence can be seen in credit migration: if an account receivable was A rated this year, what are the odds this receivable be A rated next year?

We will use a mathematical representation of these language statements to help us understand the dynamics of probabalistic change we so often observe in financial variables such as credit default.

7.3.1 Enter A.A Markov

Suppose we have a sequence of T observations, $\{X_t\}_1^T$, that are dependent. In a time series, what happens next can depend on what happened before:

$$p(X_1, X_2, ..., X_T) = p(X_1)p(X_2|X_1)...p(X_t|X_{t-1}, ..., X_1)$$

Here $p(x|y)$ is probability that an event x (e.g., default) occurs whenever (the vertical bar `|`) event y occurs. This is the statement of conditional probability. Event x depends in some way on the occurrence of y.

With markov dependence each outcome `only` depends on the one that came before.

$$p(X_1, X_2, ..., X_T) = p(X_1) \prod_{s=2}^{T} p(X_s|X_{s-1})$$

We have already encountered this when we used the functions `acf` and `ccf` to explore very similar dependencies in macro-financial time series data. In those cases we explored the dependency of today's returns on yesterday's returns. Markov dependence is equivalent to an `AR(1)` process (today = some part of yesterday plus some noise).

To generate a markov chain, we need to do three things:

1. Set up the conditional distribution.
2. Draw the initial state of the chain.
3. For every additional draw, use the previous draw to inform the new one.

Now we are a position to develop a very simple (but oftentimes useful) credit model:

- If a receivable's issuer (a customer) last month was investment grade, this month's chance of also being investment grade is 80%.
- If a receivable's issuer last month was **not** investment grade, this month's chance of being investment grade is 20%.

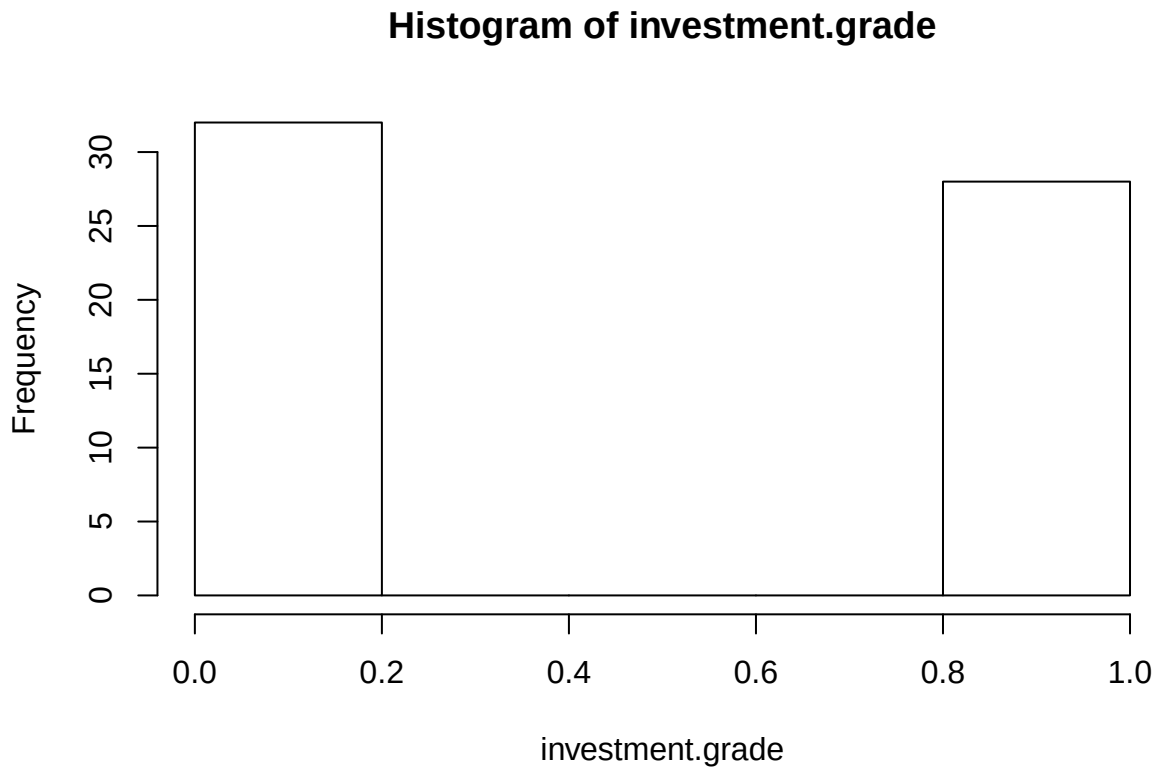
7.3.2 Try this exercise

- We will simulate monthly for 5 years. Here we parameterize even the years and calculate the number of months in the simulation. We set up a dummy `investment.grade` variable with `NA` entries into which we deposit 60 dependent coin tosses using the binomial distribution. The probability of success (state = "Investment Grade") is overall 80% and is composed of a long run 20% (across the 60 months) plus a short run 60% (of the previous month). Again the similarity to an autoregressive process here with lags at 1 month and 60 months.

Then,

1. We will run the following code.
2. We should look up `rbinom(n, size, prob)` (coin toss random draws) to see the syntax.
3. We will interpret what happens when we set the `long.run` rate to 0% and the `short.run` rate to 80%.

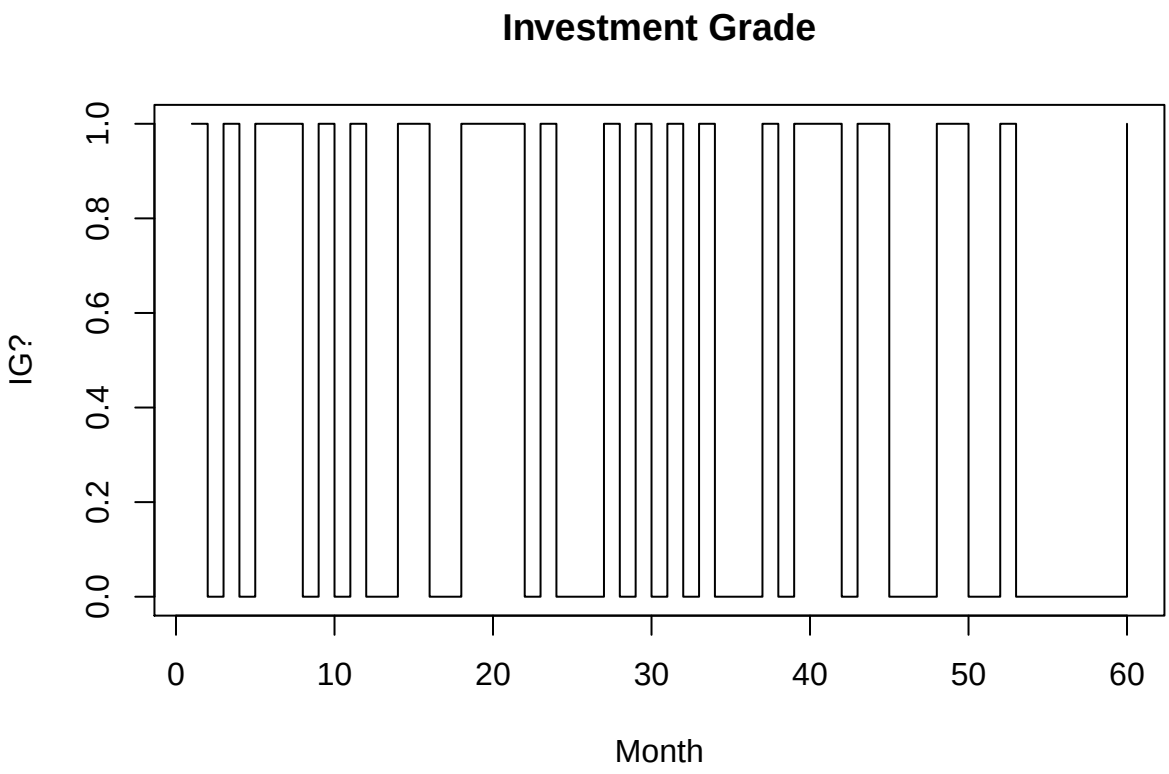
```
N.years <- 5
N.months <- N.years * 12
investment.grade <- rep(NA, N.months)
investment.grade[1] <- 1
long.run <- 0.5
short.run <- 0
for (month in 2:N.months) {
    investment.grade[month] <- rbinom(1,
        1, long.run + short.run * investment.grade[month -
            1])
}
hist(investment.grade)
```



The `for (month in 2:N.months)` loop says "For each month starting at month 2, perform the tasks in the curly brackets (`{}`) until, and including, `N.months`"

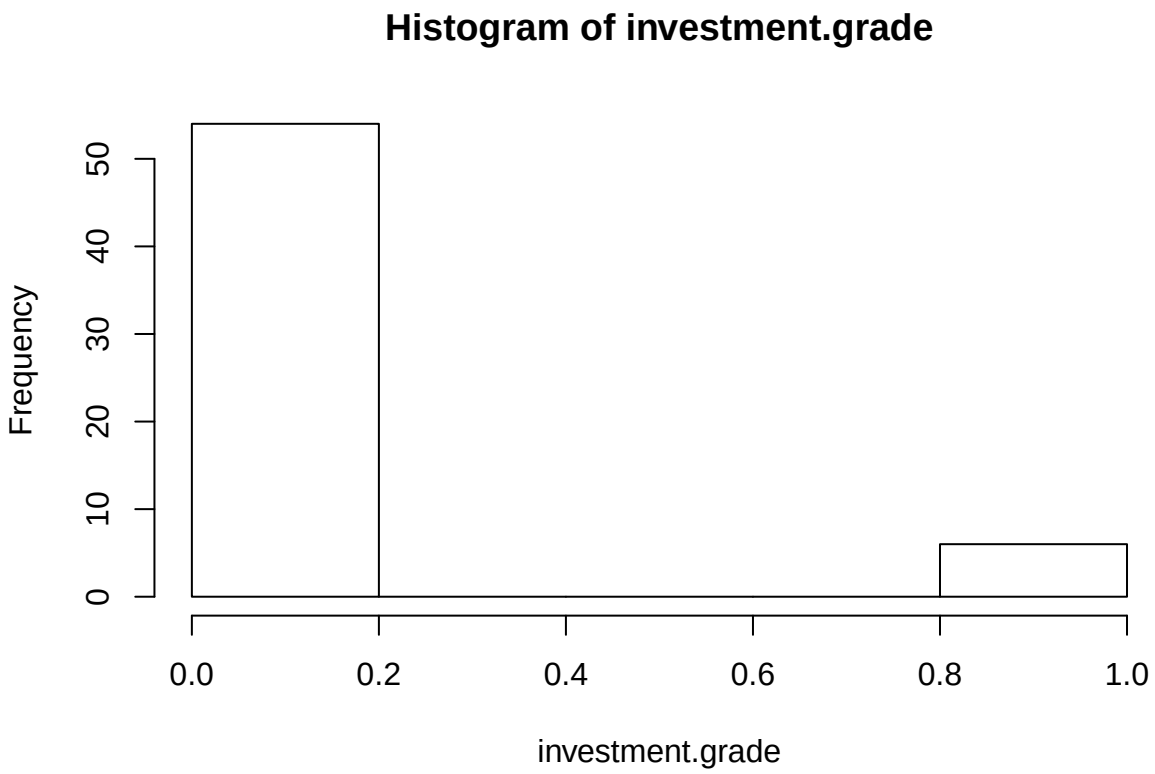
Almost evenly ditributed probabilities occur. We plot the results.

```
plot(investment.grade, main = "Investment Grade",
    xlab = "Month", ylab = "IG?", ty = "s")
```



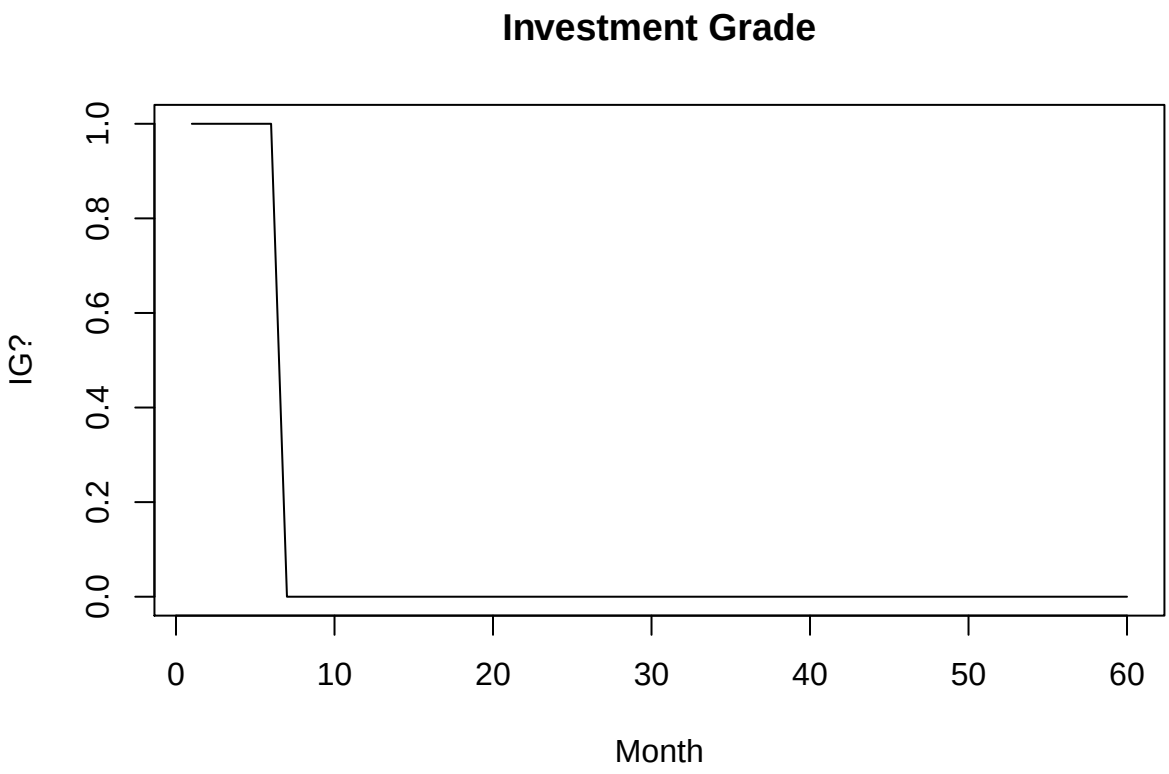
Now to look at a scenario with long-run = 0.0, and short-run = 0.80

```
N.years <- 5
N.months <- N.years * 12
investment.grade <- rep(NA, N.months)
investment.grade[1] <- 1
long.run <- 0  ## changed from base scenario 0.5
short.run <- 0.8  ## changed from base scenario 0.0
for (month in 2:N.months) {
    investment.grade[month] <- rbinom(1,
        1, long.run + short.run * investment.grade[month -
            1])
}
hist(investment.grade)
```



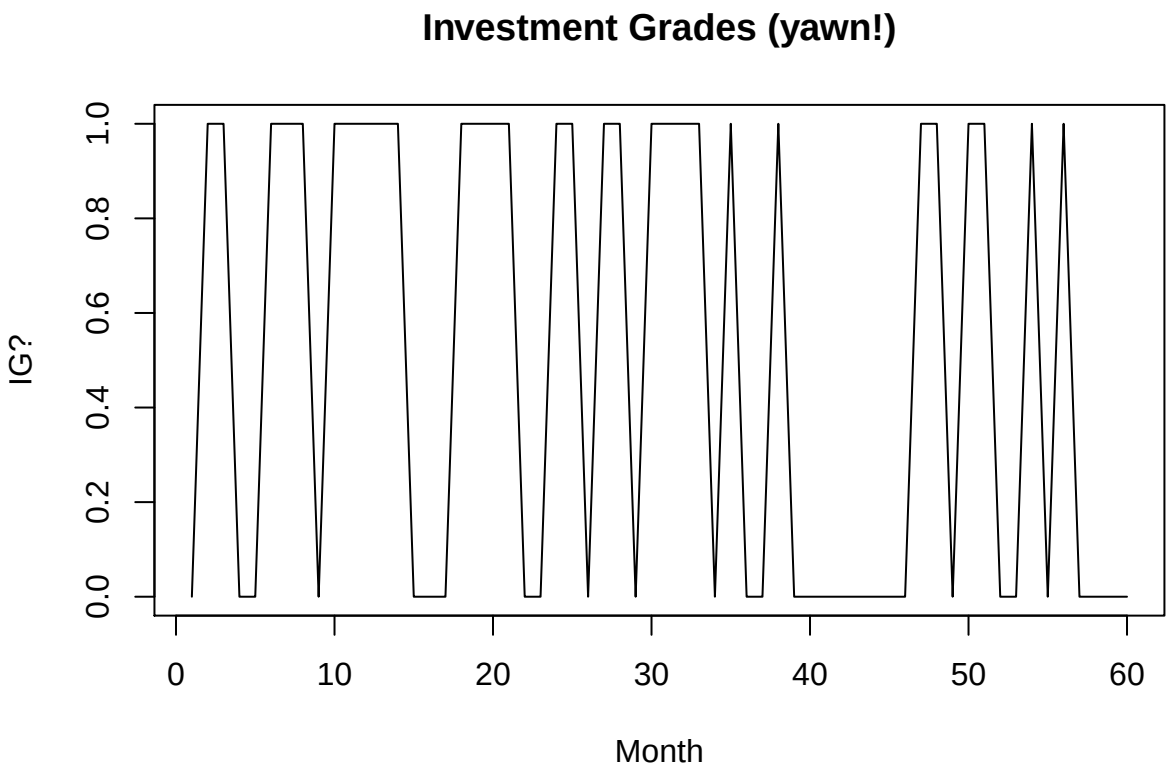
The result is much different now with more probability concentrated in lower end of investment grade scale. The next plot the up and down transitions between investment grade and not-investment grade using lines to connect up to down transitions. Now this looks more like a `bull` and `bear` graph.

```
plot(investment.grade, main = "Investment Grade",
    xlab = "Month", ylab = "IG?", ty = "l")
```



And we see how different these transitions are from a simple coin toss credit model (independence, not dependence). We could just set the long.run rate to 50% (a truly unbiased coin) and rerun, or simply run the following.

```
toss.grade <- rbinom(N.months, 1, 0.5)
plot(toss.grade, main = "Investment Grades (yawn!)",
    xlab = "Month", ylab = "IG?", ty = "l")
```



In our crude credit model transitions are represented as a matrix: Q_{ij} is $P(X_t = j|X_{t-1} = i)$ where, i is the start state ("Investment Grade"), and j is the end state ("not-Investment Grade"). Here is a transition matrix to encapsulate this data.

```
(transition.matrix <- matrix(c(0.8, 0.2,
    0.2, 0.8), nrow = 2))
```

```
##      [,1] [,2]
## [1,]  0.8  0.2
## [2,]  0.2  0.8
```

This function will nicely simulate this and more general random Markov chains.

```
rmarkovchain <- function(n.sim, transition.matrix,
    start = sample(1:nrow(transition.matrix),
        1)) {
    result <- rep(NA, n.sim)
    result[1] <- start
    for (t in 2:n.sim) result[t] <- sample(ncol(transition.matrix),
        1, prob = transition.matrix[result[t -
            1], ])
    return(result)
}
```

7.3.3 Try this next exercise

Let's run a 1000 trial markov chain with the 2-state `transition.matrix` and save to a variable called 'markov.sim'. Then we will use the `table` function to calculate how many 1's and 2' are simulated.

Many trials (and tribulations…) follow from this code.

```
markov.sim <- rmarkovchain(1000, transition.matrix)
head(markov.sim)
```

```
## [1] 2 1 1 1 1 2
```

Then we tabulate the contingency table.

```
ones <- which(markov.sim[-1000] == 1)
twos <- which(markov.sim[-1000] == 2)
state.one <- signif(table(markov.sim[ones +
    1])/length(ones), 3)
state.two <- signif(table(markov.sim[twos +
    1])/length(twos), 3)
(transition.matrix.sim <- rbind(state.one,
    state.two))
```

```
##                1     2
## state.one 0.817 0.183
## state.two 0.220 0.780
```

The result is pretty close to `transition.matrix`. The Law of large numbers would say we converge to these values. The function `which()` sets up two indexes to find where the 1's and 2's are in `markov.sim`. The function `signif()` with `3` means use 3 significant digits. The the function `table()` tabulates the number of one states and two states simulated.

Next let's develop an approach to estimating transition probabilities using observed (or these could be simulated… if we are not very sure of future trends) rates of credit migration from one rating to another.

7.4 Generating Some Hazards

Let's set up a more realistic situation. Suppose we have annual hazard rates λ for each of 4 in-house credit ratings for a portfolio of accounts receivable. We can use a pivot table to get these rates. Suppose that we know the number of accounts that transition from one rating (start state) to another rating (end state) in a unit of time (here a year). We define the hazard rate λ as the number of accounts that migrate from one rating to another rating (per year) divided by the number of all the accounts that year at that starting rating.

Next we suppose we have N ratings which in Markov-speak are states of being in a credit

rating. Transition probabilities are summarized by a N row $\times$ N column "generator" matrix $\Lambda = [\lambda_{ij}]$. Here the row index is $i = 1...N$ starting credit rating states and the column index is $j = 1...N$ ending credit rating states for a customer.

Over any small (and getting ever smaller…) time step of duration dt the probability of a transition from rating i to j is given approximately by $\lambda_{ij}dt$. The probability of moving to any other rating from the starting rating is $\Sigma_{i \neq j}\lambda_{ij}$ so that staying at rating i is given by $1 - \Sigma_{i \neq j}\lambda_{ij}$ all along the diagonal of the matrix.

A little more formally, and usefully, we can define for transitions from a starting state rating i to an ending state rating j over an interval of time $s = [0, t]$, the hazard rate for $i \neq j$,

$$\lambda_{ij} = \frac{N_{ij}}{\int_0^t Y_i(s)ds}$$

where N_{ij} is the number of accounts that made the transition from one rating i to another, different rating j, and Y_i is the number of accounts rated in the starting state rating i at the beginning of the time interval $(t, t + dt)$.

The `int` sign is again (remember the term structure of interest rates) the cumulative sum of the product of the number of accounts times the time an account is in a transition from one credit rating state to another. Using this formula, the math to generate ratings migration probabilities is fairly straightforward with a couple of kinks.

The probability of transitions is $P = exp(\Lambda)$. We can think of probabilities as discounts. They range from 0 to 1 like the present value of \$1. $\lambda_{ij}dt$ is just like a forward interest rate in form as it is the change in probability (present value) from one date to another dt. We will use the `expm` package to calculate the matrix exponent.

First suppose we already know the hazard rates for each starting state rating. We will assign the `D.rating` hazard rate as a 0 throughout as this is the last state, that is, there is no transition from this rating to another rating. Then create a λ matrix by concatenating the rows of hazard rates, and see that the diagonals are zero.

By definition, if an account stays in a rating, the diagonal must be the negative of the row sum of this matrix, where we use the apply function on the first, the row, dimension of the matrix. We put the negative row sum into the diagonal and now we have a proper hazard, or also called generator, matrix.

Now, for the last step here, we raise the hazard matrix to the exponent power. The result is the probability transition matrix.

```
require(Matrix)
rating.names <- c("A1", "A2", "A3", "B1",
    "B2", "B3", "C", "D")
lambda <- structure(c(-0.08179, 0.00663,
    6e-04, 0.00035, 0.00029, 0, 0.000815871122518043,
    0, 0.07493, -0.093, 0.02029, 0.002,
```

```
    0.00088, 0.00076, 0, 0, 0.00515,
    0.08159, -0.08697, 0.04178, 0.00441,
    0.00229, 0.00408, 0, 0.00114404022385915,
    0.00361106546371299, 0.0615488360753306,
    -0.110345216529572, 0.0534942615312562,
    0.00343192628444715, 0.0057110978576263,
    0, 0.000572020111929575, 0.000585578183304809,
    0.00321930968833733, 0.0588046608464803,
    -0.176844582534647, 0.0531948574089309,
    0.0138698090828067, 0, 0, 0.000487981819420674,
    0.0012004205617529, 0.00615719390039617,
    0.108360170794083, -0.190567240072496,
    0.134618735215477, 0, 0, 9.75963638841349e-05,
    0.000109129141977537, 0.000622637585433321,
    0.00666228898191469, 0.102671794676377,
    -0.828109189355814, 0, 0, 0, 0, 0.000622637585433321,
    0.00274329546314134, 0.0282180605610099,
    0.669014320464795, 0), dim = c(8,
    8), dimnames = list(rating.names,
    rating.names))
# write.csv(lambda,'data/lambdahat.csv')
# # To save this work lambda <-
# read.csv('data/lambdahat.csv')
rownames(lambda) <- c("A1", "A2", "A3",
    "B1", "B2", "B3", "C", "D")
colnames(lambda) <- c("A1", "A2", "A3",
    "B1", "B2", "B3", "C", "D")
apply(lambda, 1, sum)
```

```
##            A1            A2            A3            B1            B2
##  6.060336e-06  2.221830e-06 -2.304533e-06 -8.086612e-06 -4.565764e-06
##            B3             C             D
## -6.011417e-07  6.443874e-07  0.000000e+00
```

```
dimnames(lambda) <- list(rating.names,
    rating.names)
lambda.diag <- -apply(lambda, 1, sum)  ## this the is rate of staying in a state
diag(lambda) <- lambda.diag  ## this pops lambda.diag into the diagonal of lambda
apply(lambda, 1, sum)  ## should add up to zero
```

```
##        A1        A2        A3        B1        B2        B3         C
## 0.0817900 0.0930000 0.0869700 0.1103452 0.1768446 0.1905672 0.8281092
##         D
## 0.0000000
```

```
P <- expm(lambda)
apply(P, 1, sum)  ## Should be ones
```

```
##       A1       A2       A3       B1       B2       B3        C        D
## 1.085734 1.097323 1.092018 1.118836 1.195059 1.238993 1.844956 1.000000
```

We behold the generator matrix and its associated transition probabilities:

```
signif(lambda, 6)
```

```
##               A1           A2          A3          B1          B2
## A1 -6.06034e-06  7.49300e-02 5.15000e-03 1.14404e-03 5.72020e-04
## A2  6.63000e-03 -2.22183e-06 8.15900e-02 3.61107e-03 5.85578e-04
## A3  6.00000e-04  2.02900e-02 2.30453e-06 6.15488e-02 3.21931e-03
## B1  3.50000e-04  2.00000e-03 4.17800e-02 8.08661e-06 5.88047e-02
## B2  2.90000e-04  8.80000e-04 4.41000e-03 5.34943e-02 4.56576e-06
## B3  0.00000e+00  7.60000e-04 2.29000e-03 3.43193e-03 5.31949e-02
## C   8.15871e-04  0.00000e+00 4.08000e-03 5.71110e-03 1.38698e-02
## D   0.00000e+00  0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00
##              B3           C           D
## A1 0.00000e+00  0.00000e+00 0.000000000
## A2 4.87982e-04  9.75964e-05 0.000000000
## A3 1.20042e-03  1.09129e-04 0.000000000
## B1 6.15719e-03  6.22638e-04 0.000622638
## B2 1.08360e-01  6.66229e-03 0.002743300
## B3 6.01142e-07  1.02672e-01 0.028218100
## C  1.34619e-01 -6.44387e-07 0.669014000
## D  0.00000e+00  0.00000e+00 0.000000000
```

```
signif(P, 4)
```

```
## 8 x 8 Matrix of class "dgeMatrix"
##           A1        A2       A3       B1        B2        B3         C
## A1 1.000e+00 0.0750100 0.008240 0.001518 0.0006474 6.084e-05 8.586e-06
## A2 6.658e-03 1.0010000 0.081750 0.006152 0.0008909 6.040e-04 1.352e-04
## A3 6.794e-04 0.0203900 1.002000 0.061750 0.0050820 1.650e-03 2.206e-04
## B1 3.796e-04 0.0024710 0.042060 1.003000 0.0591600 9.455e-03 1.251e-03
## B2 3.089e-04 0.0010410 0.005727 0.053960 1.0050000 1.094e-01 1.228e-02
## B3 5.404e-05 0.0008185 0.002756 0.005259 0.0542200 1.010e+00 1.032e-01
## C  8.228e-04 0.0001392 0.004417 0.006528 0.0176900 1.358e-01 1.007e+00
## D  0.000e+00 0.0000000 0.000000 0.000000 0.0000000 0.000e+00 0.000e+00
##            D
## A1 3.617e-06
## A2 5.139e-05
## A3 1.065e-04
## B1 1.164e-03
```

```
## B2 7.781e-03
## B3 6.283e-02
## C  6.725e-01
## D  1.000e+00
```

The last row of P are all zeros until the last entry, the diagonal entry. It means if you are in "D" it is certain you stay in "D".

Digging in we look at the non-defaulting rating distributions with thresholds for the "C" rating.

First, we order the transition probabilities from worst (D) to best (A). Then take out the worst (D).

```
(P.reverse <- P[4:1, 4:1])  ## Reorder from worst to best
```

```
## 4 x 4 Matrix of class "dgeMatrix"
##             B1          A3          A2           A1
## B1 1.002899527 0.042057508 0.002470551 0.0003795657
## A3 0.061752300 1.002134622 0.020393062 0.0006794168
## A2 0.006152205 0.081749969 1.001081413 0.0066579619
## A1 0.001518019 0.008239507 0.075010753 1.0002449919
```

```
(P.reverse <- P.reverse[-1, ])  ## Eliminate the D state transitions
```

```
## 3 x 4 Matrix of class "dgeMatrix"
##             B1          A3         A2           A1
## A3 0.061752300 1.002134622 0.02039306 0.0006794168
## A2 0.006152205 0.081749969 1.00108141 0.0066579619
## A1 0.001518019 0.008239507 0.07501075 1.0002449919
```

h Second, we compute cumulative probabilities for the C rating in the first row now. Then compute cumulative probabilities.

```
(C.probs <- P.reverse[1, ])
```

```
##           B1           A3           A2           A1
## 0.0617523000 1.0021346225 0.0203930620 0.0006794168
```

```
(C.cumprobs <- pmin(0.999, pmax(0.001,
    cumsum(C.probs))))
```

```
## [1] 0.0617523 0.9990000 0.9990000 0.9990000
```

Third, let's interpret our results through an exercise.

7.4.1 Try this exercise

Now that we have gone this far let's run the following and answer some questions about what we did.

```
rating.df <- 16  ## Thinking that there are 3 non-defaulting ratings being spliced toget
(thresholds <- qt(C.cumprobs, rating.df))
threshold.plot <- data.frame(Deviation = seq(from = -5,
    to = 15, length = 100), Density = dt(seq(from = -5,
    to = 5, length = 100), df = rating.df))
require(ggplot2)
ggplot(threshold.plot, aes(x = Deviation,
    y = Density)) + geom_line(colour = "blue",
    size = 1.5) + geom_vline(xintercept = thresholds,
    colour = "red", size = 0.75) + geom_hline(yintercept = 0)
```

Here are the standard deviations for each transition from C to B to A:

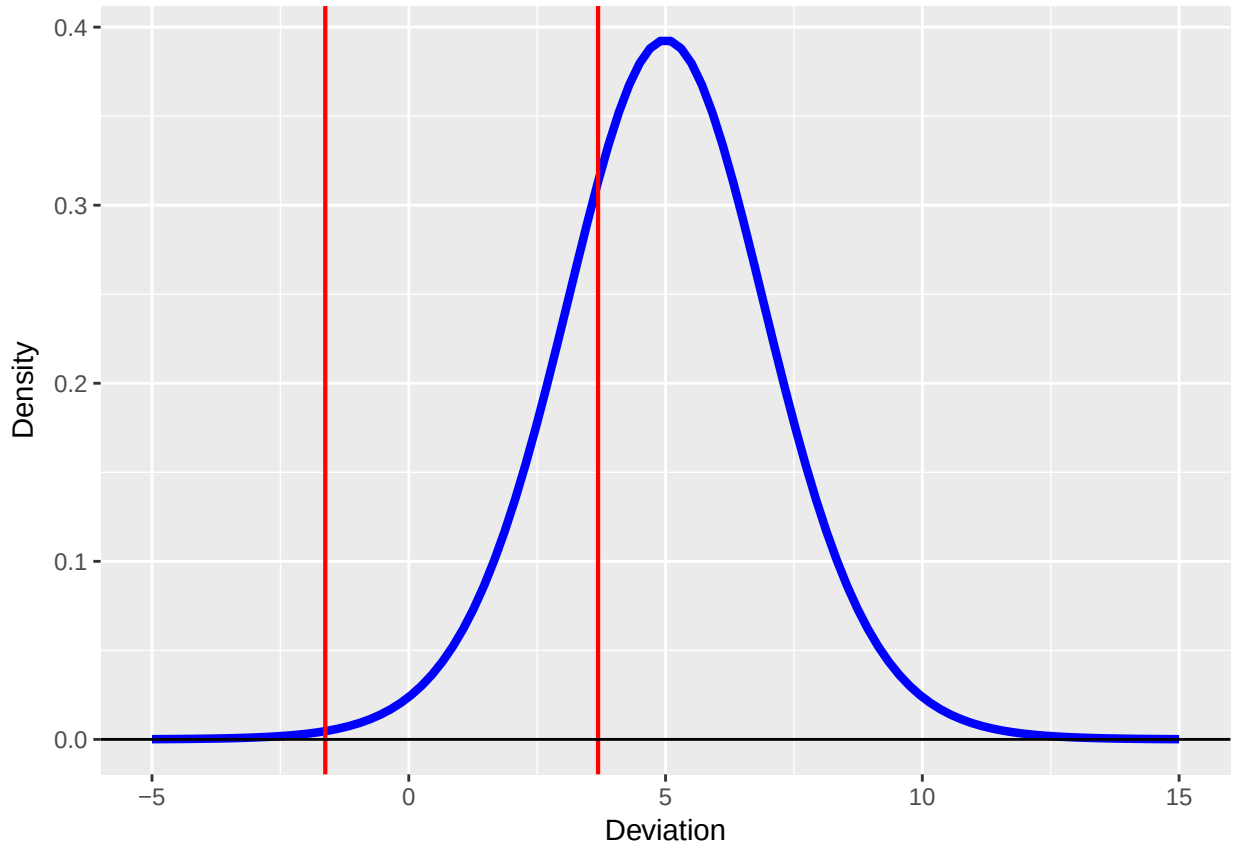
```
rating.df <- 16  ## Thinking that there are 3 non-defaulting ratings being spliced toget
(thresholds <- qt(C.cumprobs, rating.df))
```

```
## [1] -1.625890  3.686155  3.686155  3.686155
```

We compute thresholds using a fairly thick-tailed quantiles of Gossett' Student's t-distribution (only 2 degrees of freedom). These thresholds are what we have been seeking: they tell us when a customer's credit is suspect, needs to be further examined, or mitigated.

Here is the plot, where we first define a data frame suited to the task.

```
threshold.plot <- data.frame(Deviation = seq(from = -5,
    to = 15, length = 100), Density = dt(seq(from = -5,
    to = 5, length = 100), df = rating.df))
require(ggplot2)
ggplot(threshold.plot, aes(x = Deviation,
    y = Density)) + geom_line(colour = "blue",
    size = 1.5) + geom_vline(xintercept = thresholds,
    colour = "red", size = 0.75) + geom_hline(yintercept = 0)
```

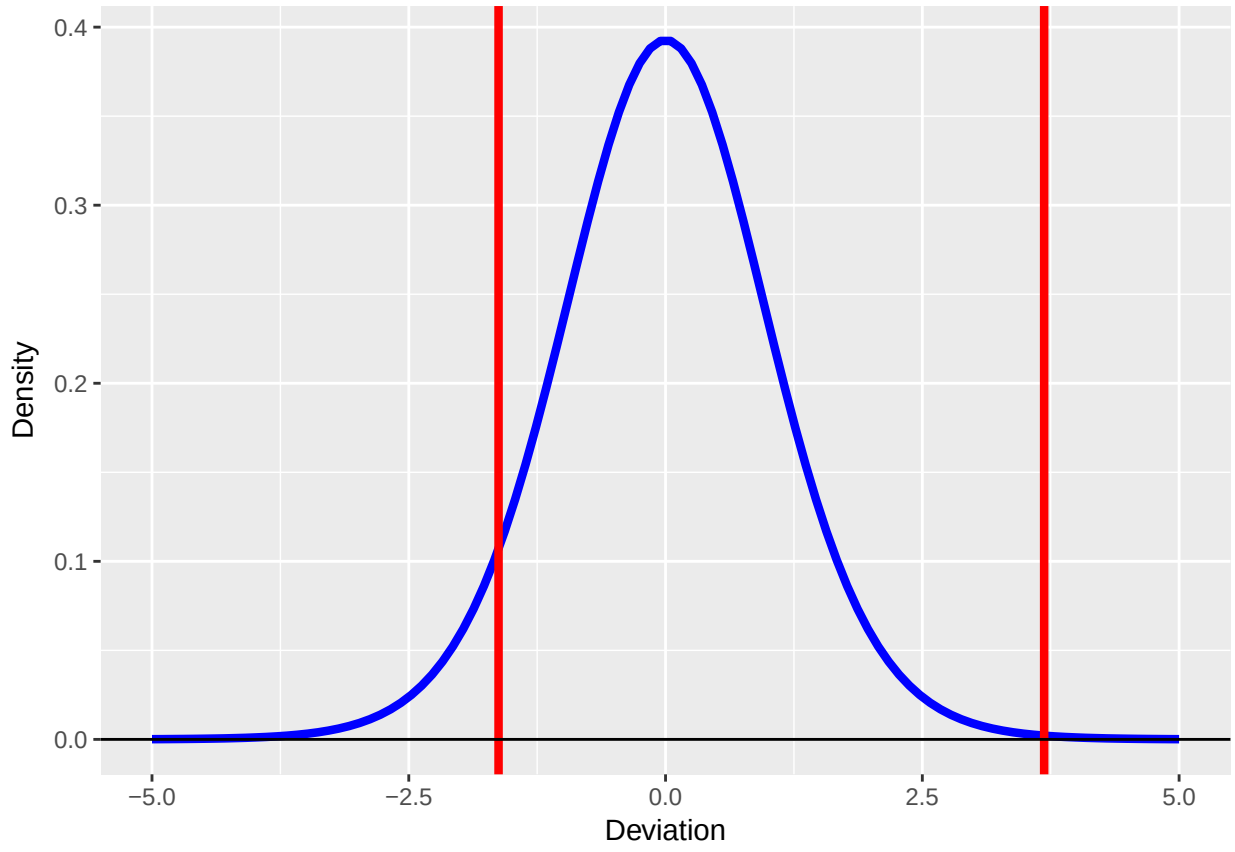



```
rating.df <- 16  ## Thinking that there are 3 non-defaulting ratings being spliced toget
(thresholds <- qt(C.cumprobs, rating.df))
threshold.plot <- data.frame(Deviation = seq(from = -5,
    to = 5, length = 100), Density = dt(seq(from = -5,
    to = 5, length = 100), df = rating.df))
require(ggplot2)
ggplot(threshold.plot, aes(x = Deviation,
    y = Density)) + geom_line(colour = "blue",
    size = 1.5) + geom_vline(xintercept = thresholds,
    colour = "red", size = 1.5) + geom_hline(yintercept = 0)
```

Again the t-distribution standard deviation thresholds:

```
## [1] -1.625890  3.686155  3.686155  3.686155
```

```
## [1] -1.625890  3.686155  3.686155  3.686155
```

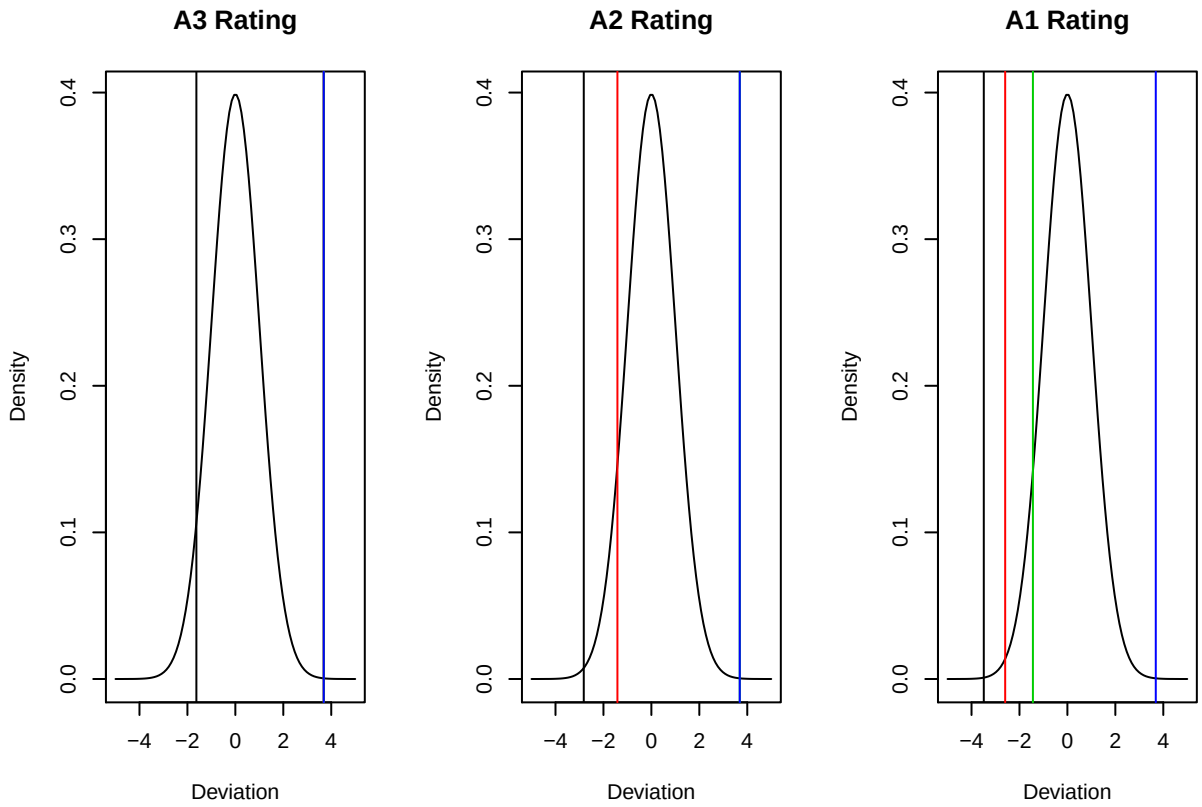


The `dt` in the plot is the Student's t density function from -5 to +5 for 100 ticks.

Now that this works for one rating let's plot all of the ratings row by row using apply with the cusum function (on the first dimension). We will use the Student's t distribution to generate thresholds.

```
## Being sure we transpose this next
## result
cumprobs <- t(apply(P.reverse, 1, function(v) {
    pmin(0.999, pmax(0.001, cumsum(v)))
}))
## v holds the values of of the
## elements of each row (remember this
## is 'row-wise') we are accumulating
## a sum
rating.df <- 16
thresholds <- qt(cumprobs, rating.df)  ##Use Student's t
plot.parm <- par(mfrow = c(1, 3))  ## Building a 1 row, 3 column panel
for (j in 1:nrow(thresholds)) {
    plot(seq(from = -5, to = 5, length = 100),
        dnorm(seq(from = -5, to = 5,
            length = 100)), type = "l",
```

```
        xlab = "Deviation", ylab = "Density",
        main = paste(rownames(thresholds)[j],
            "Rating"))
    abline(v = thresholds[j, ], col = 1:length(thresholds[j,
        ]))
}
```




```
par(plot.parm)
```

The A ratings have a much lower loss threshold as we would expect than C or B.

7.5 Now for the Future

Decision makers now want to use this model to look into the future. Using the hazard rates to develop policies for our accounts receivable, and ultimately customer and counterparty (e.g., vendors) relationships. Let's build a rating event simulation to dig into how these rates might occur in the wild future.

Let's use transaction data to estimate an eight rating hazard rate to simulate the data we might have seen in the first place.

```
## Number of rating categories
n.cat <- dim(lambda)[1]
## Sum of migration rates from each
## non-default rating
(rates.sum <- -diag(lambda)[-n.cat])
```

```
##            A1            A2            A3            B1            B2
##  6.060336e-06  2.221830e-06 -2.304533e-06 -8.086612e-06 -4.565764e-06
##            B3             C
## -6.011417e-07  6.443874e-07
```

```
## Names of non-default ratings
names.nondf <- names(rates.sum)
## probabilities of transition (off
## diagonal)
(transition.probs <- lambda[-n.cat, ]/rates.sum)
```

```
##            A1         A2         A3          B1           B2          B3
## A1   -1.00000 12364.0014   849.7879    188.7751     94.38753      0.0000
## A2 2984.02625    -1.0000 36721.9761   1625.2661    263.55666    219.6306
## A3 -260.35648 -8804.3884    -1.0000 -26707.7307  -1396.94691   -520.8955
## B1  -43.28141  -247.3224 -5166.5643     -1.0000  -7271.85403   -761.4059
## B2  -63.51620  -192.7388  -965.8843 -11716.3871     -1.00000 -23733.1944
## B3    0.00000 -1264.2609 -3809.4178  -5709.0135 -88489.70993     -1.0000
## C  1266.11897     0.0000  6331.5948   8862.8328  21524.02249 208909.6300
##                C            D
## A1       0.00000       0.0000
## A2      43.92611       0.0000
## A3     -47.35413       0.0000
## B1     -76.99610     -76.9961
## B2   -1459.18375    -600.8404
## B3 -170794.65519  -46940.7780
## C       -1.00000 1038217.5553
```

```
## matrix rows sum to zero
apply(transition.probs, 1, sum)
```

```
##         A1         A2         A3         B1         B2         B3
##   13495.95   41857.38  -37738.67  -13645.42  -38732.75 -317008.84
##          C
## 1285110.75
```

A last row would have been the D rating absorbing row of 3 zeros and a 1.

Let's create a miniverse of 20 years of rating migration data. This will help us understand the many possibilities for accounts receivable transitions, even through we do not have the direct data to do so. Within the 20 years we allow for up to 5 transitions per account. All

of this is deposited into a data frame that will have as columns (fields in a database) an ID, a start time for the transition state, and end time for the state, followed by the starting transition and the ending transition. These fields will allow us to compute how long an account dwells in a transition state (that is, a rating).

We generate a random portfolio of 100 accounts and calculate how long it takes for them to transition from one credit rating (transition start state) to another (transition end state). We also build into the simulation fractions of accounts in each of the non-defaulting notches. These fractions serve as the first step in estimating the probability of being in a rating notch.

```
set.seed(1016)
n.firms <- 100
fractions <- c(0.1, 0.1, 0.1, 0.1, 0.1,
    0.1, 0.1)  ## equal number of transitions of each rating
initial.ratings <- sample(names.nondf,
    size = n.firms, replace = TRUE, prob = fractions)
table(initial.ratings)
```

```
## initial.ratings
## A1 A2 A3 B1 B2 B3  C
## 14 18 11 10 21 13 13
```

The `table` function tells us how many firms out of a 100 are in each non-default rating on average over the simulation.

Now we set up parameters for 20 years of data across the 100 firms. This will help us understand the many possibilities for accounts receivable transitions, even through we might not have enough direct data to do so. Within the 20 years we allow for up to 5 transitions per account. All of this is deposited into a data frame that will have as columns (fields in a database) an ID, a start time for the transition state, and end time for the state, followed by the starting transition and the ending transition. These fields will allow us to compute how long an account dwells in a transition state (that is, a rating).

We begin the simulation of events by creating an outer loop of firms, 100 in all in our simulated accounts receivable portfolio. We are allowing no more than 5 credit rating transitions as well.

```
n.years <- 10
max.row <- n.firms * 5
output.Transitions <- data.frame(id = numeric(max.row),
    start.time = numeric(max.row), end.time = numeric(max.row),
    start.rating = rep("", max.row),
    end.rating = rep("", max.row), stringsAsFactors = FALSE)
```

The `for` loop simulates start and end times for each rating for each firm (up to so many transitions across so many years). Within the loop we calculate how long the firm is in each rating, until it isn't. It is this time in transition (the "spell" or "dwell time") that is really the core of the simulation and our understanding of hazard rate methodology. We will model

this dwell time as an exponential function using `rexp()` to sample a time increment. Other functions we could use might be the Weibull or double-exponential for this purpose. At the end of this routine we write the results to file.

```
count <- 0
for (i in 1:n.firms) {
    end.time <- 0
    end.rating <- initial.ratings[i]
    while ((end.time < n.years) & (end.rating !=
        "D") & !is.na(end.time)) {
        count <- count + 1
        start.time <- end.time
        start.rating <- end.rating
        end.time <- start.time + rexp(n = 1,
            rate = rates.sum[start.rating])
        if ((end.time <= n.years) & !is.na(end.time)) {
            pvals <- transition.probs[start.rating,
                names.nondf != start.rating]
            end.rating <- sample(names(pvals),
                size = 1, prob = pvals)
        }
        if ((end.time > n.years) & !is.na(end.time)) {
            end.time <- n.years
        }
        output.Transitions[count, ] <- list(i,
            start.time, end.time, start.rating,
            end.rating)
    }
}
write.csv(output.Transitions, "data/creditmigration.csv")
```

We configure all of the columns of our output data framw. We also compute the time in a rating state and add to the original `output.Transitions` data frame.

```
## output.Transitions <-
## output.Transitions[1:count,]
output.Transitions <- read.csv("data/creditmigration.csv")
output.Transitions$start.rating <- as.factor(output.Transitions$start.rating)
output.Transitions$end.rating <- as.factor(output.Transitions$end.rating)
## output.Transitions$time <-
## output.Transitions$end.time-output.Transitions$start.time
transition.Events <- output.Transitions
```

And then we look at the output.

```
head(output.Transitions)
```

```
##   id start.date start.rating end.date end.rating time start.year end.year
## 1  1      40541            6    42366          6 1825       2010     2015
## 2  2      41412            6    42366          6  954       2013     2015
## 3  3      40174            5    40479          6  305       2009     2010
## 4  3      40479            6    40905          5  426       2010     2011
## 5  3      40905            5    42366          5 1461       2011     2015
## 6  4      40905            5    41056          6  151       2011     2012
```

Let's inspect our handiwork further

```
head(transition.Events, n = 5)
```

```
##   id start.date start.rating end.date end.rating time start.year end.year
## 1  1      40541            6    42366          6 1825       2010     2015
## 2  2      41412            6    42366          6  954       2013     2015
## 3  3      40174            5    40479          6  305       2009     2010
## 4  3      40479            6    40905          5  426       2010     2011
## 5  3      40905            5    42366          5 1461       2011     2015
```

```
summary(transition.Events)
```

```
##        id          start.date     start.rating     end.date
##  Min.   :  1.0   Min.   :39951   3      : 351   Min.   :39960
##  1st Qu.:230.0   1st Qu.:40418   4      : 348   1st Qu.:41170
##  Median :430.0   Median :40905   5      : 212   Median :42143
##  Mean   :421.8   Mean   :40973   6      : 189   Mean   :41765
##  3rd Qu.:631.0   3rd Qu.:41421   2      : 179   3rd Qu.:42366
##  Max.   :830.0   Max.   :42366   (Other):  94   Max.   :42366
##  NA's   :1709    NA's   :1709    NA's   :1709   NA's   :1709
##    end.rating        time          start.year      end.year
##  3      : 333   Min.   :   0.0   Min.   :2009   Min.   :2009
##  4      : 332   1st Qu.: 344.0   1st Qu.:2010   1st Qu.:2012
##  5      : 209   Median : 631.0   Median :2011   Median :2015
##  6      : 190   Mean   : 791.5   Mean   :2012   Mean   :2014
##  2      : 177   3rd Qu.:1157.0   3rd Qu.:2013   3rd Qu.:2015
##  (Other): 132   Max.   :2415.0   Max.   :2015   Max.   :2015
##  NA's   :1709   NA's   :1709     NA's   :1709   NA's   :1709
```

```
dim(transition.Events)
```

```
## [1] 3082    8
```

We may want to save this simulated data to our working directory.

```
summary(transition.Events)
```

```
##        id          start.date     start.rating     end.date
##  Min.   :  1.0   Min.   :39951   3      : 351   Min.   :39960
##  1st Qu.:230.0   1st Qu.:40418   4      : 348   1st Qu.:41170
```

```
##  Median :430.0   Median :40905   5       : 212   Median :42143
##  Mean   :421.8   Mean   :40973   6       : 189   Mean   :41765
##  3rd Qu.:631.0   3rd Qu.:41421   2       : 179   3rd Qu.:42366
##  Max.   :830.0   Max.   :42366   (Other):  94   Max.   :42366
##  NA's   :1709    NA's   :1709    NA's   :1709   NA's   :1709
##    end.rating        time          start.year      end.year
##  3      : 333   Min.   :   0.0   Min.   :2009   Min.   :2009
##  4      : 332   1st Qu.: 344.0   1st Qu.:2010   1st Qu.:2012
##  5      : 209   Median : 631.0   Median :2011   Median :2015
##  6      : 190   Mean   : 791.5   Mean   :2012   Mean   :2014
##  2      : 177   3rd Qu.:1157.0   3rd Qu.:2013   3rd Qu.:2015
##  (Other): 132   Max.   :2415.0   Max.   :2015   Max.   :2015
##  NA's   :1709   NA's   :1709     NA's   :1709   NA's   :1709
```

```
dim(transition.Events)
```

```
## [1] 3082    8
```

```
## write.csv(transition.Events,'data/transitionevents.csv')
## to save this work
```

7.6 Build We Must

Now let's go through the work flow to build a hazard rate estimation model from data. Even though this is simulated data, we can use this resampling approach to examine the components of the hazard rate structure across this rating system.

Our first step is to tabulate the count of transitions from ratings to one another and to self with enough columns to fill the ending state ratings. Let's recall the notation of the hazard rate calculation:

$$\lambda_{ij} = \frac{N_{ij}}{\int_0^t Y_i(s)ds}$$

Now we begin to estimate:

```
(Nij.table <- table(transition.Events$start.rating,
    transition.Events$end.rating))
```

```
##
##       1   2   3   4   5   6   7   8
##   1  17   2   1   0   0   0   0   0
##   2   9 146  24   0   0   0   0   0
##   3   0  29 271  49   2   0   0   0
##   4   0   0  35 251  51   8   3   0
```

```
##   5   0   0   2  32 125  50   3   0
##   6   0   0   0   0  28 116  40   5
##   7   0   0   0   0   2  14  33  12
##   8   0   0   0   0   1   2   3   7
```

```
(RiskSet <- by(transition.Events$time,
    transition.Events$start.rating, sum))
```

```
## transition.Events$start.rating: 1
## [1] 25683
## ------------------------------------------------------------
## transition.Events$start.rating: 2
## [1] 172285
## ------------------------------------------------------------
## transition.Events$start.rating: 3
## [1] 318679
## ------------------------------------------------------------
## transition.Events$start.rating: 4
## [1] 279219
## ------------------------------------------------------------
## transition.Events$start.rating: 5
## [1] 129886
## ------------------------------------------------------------
## transition.Events$start.rating: 6
## [1] 120696
## ------------------------------------------------------------
## transition.Events$start.rating: 7
## [1] 29737
## ------------------------------------------------------------
## transition.Events$start.rating: 8
## [1] 10577
```

```
I.levels <- levels(transition.Events$start.rating)
J.levels <- levels(transition.Events$end.rating)
(Ni.matrix <- matrix(nrow = length(I.levels),
    ncol = length(J.levels), as.numeric(RiskSet),
    byrow = FALSE))
```

```
##        [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]   [,8]
## [1,]  25683  25683  25683  25683  25683  25683  25683  25683
## [2,] 172285 172285 172285 172285 172285 172285 172285 172285
## [3,] 318679 318679 318679 318679 318679 318679 318679 318679
## [4,] 279219 279219 279219 279219 279219 279219 279219 279219
## [5,] 129886 129886 129886 129886 129886 129886 129886 129886
## [6,] 120696 120696 120696 120696 120696 120696 120696 120696
## [7,]  29737  29737  29737  29737  29737  29737  29737  29737
```

```
## [8,]  10577  10577  10577  10577  10577  10577  10577  10577
```

The `Nij.table` tabulates the count of transitions from start to end. The `Ni.matrix` gives us the row sums of transition times ("spell" or "dwelling time") for each starting state.

Now we can estimate a simple hazard rate matrix. This looks just like the formula now.

```
(lambda.hat <- Nij.table/Ni.matrix)
```

```
##
##                1            2            3            4            5
##   1 6.619164e-04 7.787252e-05 3.893626e-05 0.000000e+00 0.000000e+00
##   2 5.223902e-05 8.474330e-04 1.393041e-04 0.000000e+00 0.000000e+00
##   3 0.000000e+00 9.100066e-05 8.503855e-04 1.537597e-04 6.275908e-06
##   4 0.000000e+00 0.000000e+00 1.253496e-04 8.989360e-04 1.826523e-04
##   5 0.000000e+00 0.000000e+00 1.539812e-05 2.463699e-04 9.623824e-04
##   6 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 2.319878e-04
##   7 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 6.725628e-05
##   8 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 9.454477e-05
##
##                6            7            8
##   1 0.000000e+00 0.000000e+00 0.000000e+00
##   2 0.000000e+00 0.000000e+00 0.000000e+00
##   3 0.000000e+00 0.000000e+00 0.000000e+00
##   4 2.865135e-05 1.074425e-05 0.000000e+00
##   5 3.849530e-04 2.309718e-05 0.000000e+00
##   6 9.610923e-04 3.314111e-04 4.142639e-05
##   7 4.707940e-04 1.109729e-03 4.035377e-04
##   8 1.890895e-04 2.836343e-04 6.618134e-04
```

hThe diagonal of a generator matrix is the negative of the sum of off-diagonal elements row by row.

```
## Add default row and correct
## diagonal
lambda.hat <- lambda.hat[-8, ]
lambda.hat.diag <- rep(0, dim(lambda.hat)[2])
lambda.hat <- rbind(lambda.hat, lambda.hat.diag)
diag(lambda.hat) <- lambda.hat.diag
rowsums <- apply(lambda.hat, 1, sum)
diag(lambda.hat) <- -rowsums
## check for valid generator
apply(lambda.hat, 1, sum)
```

```
##                1                2                3                4
##    0.000000e+00     0.000000e+00    -2.371692e-20     8.470329e-21
##                5                6                7 lambda.hat.diag
##   -5.082198e-20    -4.065758e-20     0.000000e+00     0.000000e+00
```

```
lambda.hat
```

```
##                             1             2             3             4
## 1                -1.168088e-04  7.787252e-05  3.893626e-05  0.0000000000
## 2                 5.223902e-05 -1.915431e-04  1.393041e-04  0.0000000000
## 3                 0.000000e+00  9.100066e-05 -2.510363e-04  0.0001537597
## 4                 0.000000e+00  0.000000e+00  1.253496e-04 -0.0003473976
## 5                 0.000000e+00  0.000000e+00  1.539812e-05  0.0002463699
## 6                 0.000000e+00  0.000000e+00  0.000000e+00  0.0000000000
## 7                 0.000000e+00  0.000000e+00  0.000000e+00  0.0000000000
## lambda.hat.diag   0.000000e+00  0.000000e+00  0.000000e+00  0.0000000000
##                             5             6             7             8
## 1                 0.000000e+00  0.000000e+00  0.000000e+00 0.000000e+00
## 2                 0.000000e+00  0.000000e+00  0.000000e+00 0.000000e+00
## 3                 6.275908e-06  0.000000e+00  0.000000e+00 0.000000e+00
## 4                 1.826523e-04  2.865135e-05  1.074425e-05 0.000000e+00
## 5                -6.698181e-04  3.849530e-04  2.309718e-05 0.000000e+00
## 6                 2.319878e-04 -6.048253e-04  3.314111e-04 4.142639e-05
## 7                 6.725628e-05  4.707940e-04 -9.415879e-04 4.035377e-04
## lambda.hat.diag   0.000000e+00  0.000000e+00  0.000000e+00 0.000000e+00
```

```
dim(lambda.hat)
```

```
## [1] 8 8
```

That last statement calculates the row sums of the `lambda.hat` matrix. If the sums are zero, then we correctly placed the diagonals.

Now we generate the transition matrix using the matrix exponent function. Verify how close the simulation is (or not) to the original transition probabilities using `P - P.hat`.

```
## The matrix exponential Annual
## transition probabilities
(P.hat <- as.matrix(expm(lambda.hat)))
```

```
##                            1            2            3            4
## 1                9.998832e-01 7.786229e-05 3.893452e-05 2.992989e-09
## 2                5.223097e-05 9.998085e-01 1.392743e-04 1.070695e-08
## 3                2.376450e-09 9.098053e-05 9.997490e-01 1.537145e-04
## 4                9.929226e-14 5.701991e-09 1.253135e-04 9.996527e-01
## 5                1.220226e-14 7.008280e-10 1.540647e-05 2.462458e-04
## 6                7.075944e-19 5.418359e-14 1.786430e-09 2.856297e-08
## 7                2.051897e-19 1.571320e-14 5.181150e-10 8.284062e-09
## lambda.hat.diag 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##                            5            6            7            8
## 1                1.223315e-10 4.427252e-14 1.166200e-14 1.635000e-18
## 2                4.376216e-10 1.583824e-13 4.172013e-14 5.849220e-18
```

```
## 3               6.287056e-06 3.411128e-09 8.985134e-10 1.679627e-13
## 4               1.825635e-04 2.867537e-05 1.074419e-05 2.761640e-09
## 5               9.993305e-01 3.847167e-04 2.314364e-05 1.263689e-08
## 6               2.318511e-04 9.993955e-01 3.311577e-04 4.148070e-05
## 7               6.725669e-05 4.704430e-04 9.990589e-01 4.033575e-04
## lambda.hat.diag 0.000000e+00 0.000000e+00 0.000000e+00 1.000000e+00
```

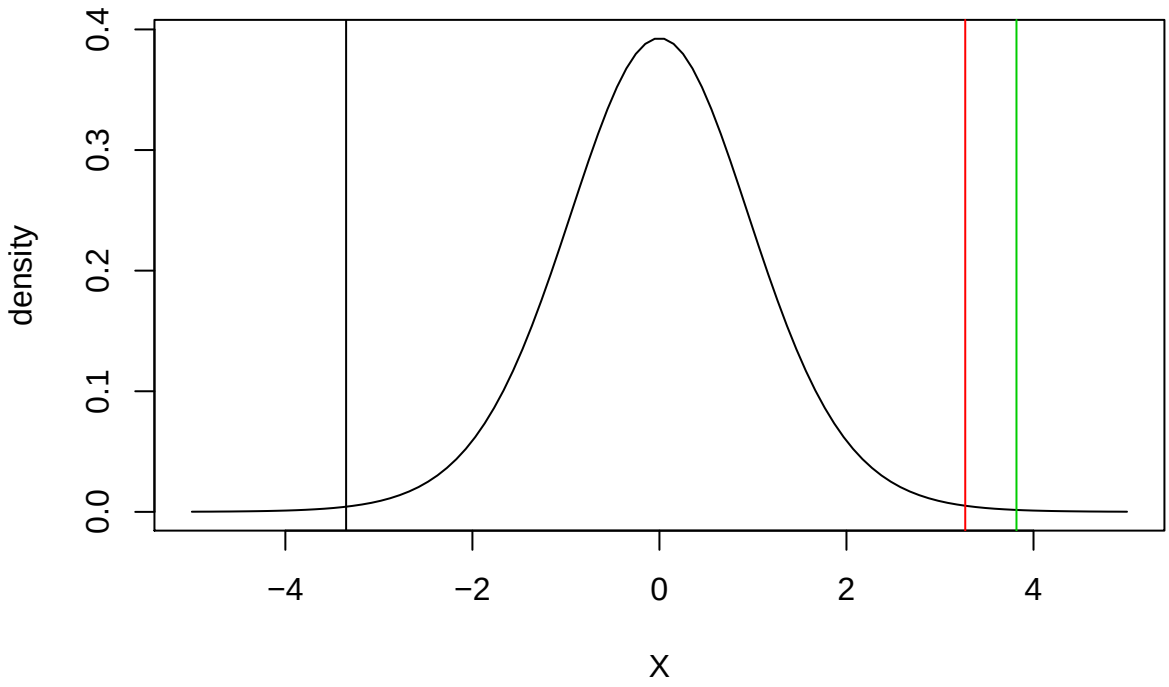
Most of our accounts will most probably stay in their initial ratings.

7.6.1 What does that mean?

Now let's replot the non-defaulting state distributions with thresholds using the simulation of the transition probabilities.

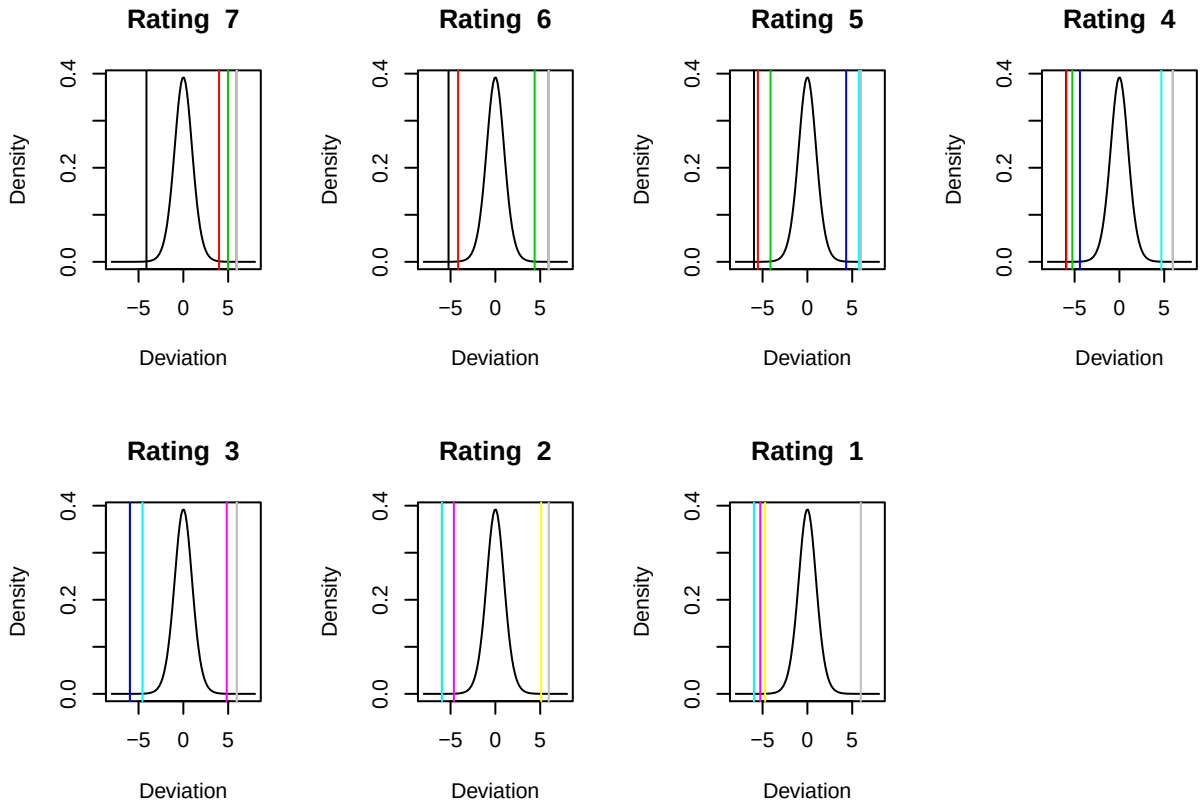
```
P.reverse <- P.hat[8:1, 8:1]  ## Use P.hat now
P.reverse <- P.reverse[-1, ]  ##without D state transitions
## select the C rating transition
## probabilities
C.probs <- P.reverse[1, ]
C.cumprobs <- cumsum(C.probs)
thresholds <- qnorm(C.cumprobs)

plot(seq(from = -5, to = 5, length = 100),
    dt(seq(from = -5, to = 5, length = 100),
        df = 16), type = "l", xlab = "X",
    ylab = "density")
abline(v = thresholds, col = 1:length(thresholds))
```



We look at all of the ratings row by row using apply with the cusum function.

```
cum.probs <- t(apply(P.reverse, 1, function(v) {
    pmin(0.99999, pmax(1e-05, cumsum(v)))
}))
thresholds <- qt(cum.probs, 16)  ##Use Student-t with 16 degrees of freedom
opa <- par(mfrow = c(2, 4))
for (j in 1:nrow(thresholds)) {
    plot(seq(from = -8, to = 8, length = 100),
        dt(seq(from = -8, to = 8, length = 100),
            16), type = "l", xlab = "Deviation",
        ylab = "Density", main = paste("Rating ",
            rownames(thresholds)[j]))
    abline(v = thresholds[j, ], col = 1:length(thresholds[j,
        ]))
}
par(opa)
```



`cum.probs` is adjusted for 0 and 1 as these might produce `NaN`s and stop the action. Notice the use of `pmin` and `pmax` to perform element by element (*p*arallel minimum and maximum) operations.

This just goes to show it is hard to be rated C. It is the riskiest of all.

7.7 Now for the Finale

We will now use a technique that will can be used with any risk category. The question on the table is: how can we generate a loss distribution for credit risk with so much variation across ratings?

- A loss distribution is composed of two elements, frequency and severity.
- Frequency asks the question how often and related to that question, how likely.
- Severity asks how big a loss.

For operational risk frequency will be modeled by a Poisson distribution with an average arrival rate of any loss above a threshold. Severity will be modeled using Beta, Gamma, Exponential, Weibull, Normal, Log-Normal, Student-t, or extreme value distributions. For credit risk we can model some further components: loss given default (i.e., recovery) and exposure at default for severity, and probability of default for frequency. By the way our

transition probabilities are counting distirubtion and have as their basis the Poisson distirbution. We have used both dwelling times AND the computation of hazards with dependent structures to model the transition probabilities.

Let's look at our top 20 customers by revenue and suppose we have these exposures. Probabilities of default are derived from the transition probabilities we just calculated.

```
## Specify portfolio characteristics
n.accounts <- 20
exposure.accounts <- c(5, 5, 5, 5, 10,
    10, 10, 10, 20, 20, 20, 20, 30, 30,
    30, 30, 40, 40, 40, 40)
probability.accounts <- c(rep(0.1, 10),
    rep(0.05, 10))
```

7.8 Enter Laplace

A hugely useful tool for finance and risk is the Laplace transform. Let's formally define this as the integral (again think cumulative sum).

$$L(f(t)) = \int_0^\infty e^{-st} f(t) dt = f(s)$$

where $f(t)$ is a monotonic, piecewise differentiable function, say the cash flow from an asset, or a cumulative density function . To make this "real" for us we can calculate (or look up on a standard table of transforms)

$$L\{1\} = \int_0^\infty e^{-st} 1 dt = \frac{1}{s}$$

If 1 is a cash flow today $t = 0$, then $L\{1\}$ can be interpreted as the present value of \$1 at s rate of return in perpetuity. Laplace transforms are thus to a financial economist a present value calculation. They map the time series of cash flows, returns, exposures, into rates of return.

In our context we are trying to meld receivables account exposures, the rate of recovery if a default occurs, and the probability of default we worked so hard to calculate using the Markov chain probabilities.

For our purposes we need to calculate for m exposures the Laplace transform of the sum of losses convolved with probabilities of default:

$$\sum_0^m (pd_i \times lgd_i \times S_i)$$

where pd is the probability of default, lgd is the loss given default, and S is exposure in monetary units. In what follows `lgd` is typically one minus the recovery rate in the event of default. Here we assume perfect recovery, even our attorney's fees.

This function effectively computes the cumulative loss given the probability of default, raw account exposures, and the loss given default.

```
laplace.transform <- function(t, pd,
    exposure, lgd = rep(1, length(exposure))) {
    output <- rep(NA, length(t))
    for (i in 1:length(t)) output[i] <- exp(sum(log(1 -
        pd * (1 - exp(-exposure * lgd *
            t[i])))))
    output
}
```

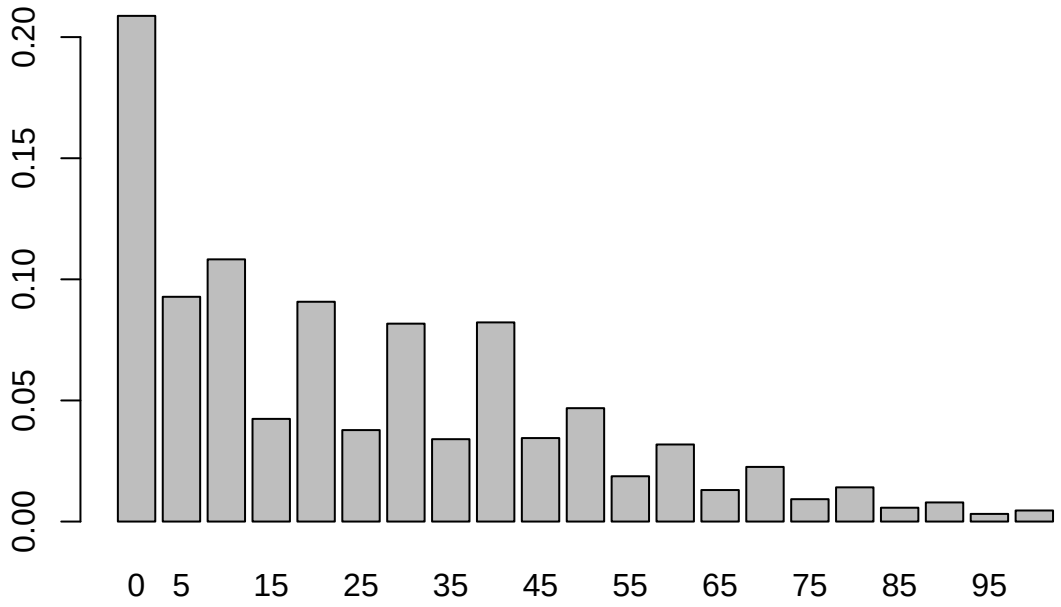
7.8.1 It's technical...

We can evaluate the Laplace Transform at $s = i$ (where $i = sqrt{-}1$, the imaginary number) to produce the loss distribution's characteristic function. The loss distribution's characteristic function encapsulates all of the information about loss: means, standard deviations, skewness, kurtosis, quantiles,..., all of it. When we use the characteristic function we can then calculate the Fast Fourier Transform of the loss distribution characteristic function to recover the loss distribution itself. - This is a fast alternative to Monte Carlo simulation. - Note below that we must divide the FFT output by the number of exposures (plus 1 to get the factor of 2 necessary for efficient operation of the FFT).

```
N <- sum(exposure.accounts) + 1  ## Exposure sum as a multiple of two
t <- 2 * pi * (0:(N - 1))/N  ## Setting up a grid of t's
loss.transform <- laplace.transform(-t *
    (0+1i), probability.accounts, exposure.accounts)  ## 1i is the imaginary number
loss.fft <- round(Re(fft(loss.transform)),
    digits = 20)  ## Back to Real numbers
sum(loss.fft)
```

```
## [1] 421
```

```
loss.probs <- loss.fft/N
loss.probs.1 <- loss.probs[(0:20) * 5 +
    1]
loss.q <- quantile(loss.probs.1, 0.99)
loss.es <- loss.probs.1[loss.probs.1 >
    loss.q]
barplot(loss.probs.1, names.arg = paste((0:20) *
    5))
```




```
(VaR <- loss.q * N)
```

```
##      99%
## 79.42721
```

```
(ES <- loss.es * N)
```

```
## [1] 87.89076
```

We can use this same technique when we try to aggregate across all exposures in the organization. The last two statements using the `quantile` function calculate the amount of capital we need to cover at least a 1% loss on this portfolio of accounts receivable. The barplot provides a rough and ready histogram of the loss distribution.

7.9 Summary

We covered a lot of credit risk maths: A. Markov, transition probabilities, hazard rates, M. Laplace. We just built a rating model that produced data driven risk thresholds. We used these probabilities to generate an aggregate credit loss distribution.

7.10 Further Reading

Much of the `R` code in generating hazard-based transition probabilities derives from <

7.11 Practice Laboratory

7.11.1 Practice laboratory #1

7.11.1.1 Problem

7.11.1.2 Questions

7.11.2 Practice laboratory #2

7.11.2.1 Problem

7.11.2.2 Questions

7.12 Project

7.12.1 Background

7.12.2 Data

7.12.3 Workflow

7.12.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.
- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.

- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.
- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

7.13 References

Chapter 8

Operational Risk and Extreme Finance

8.1 Imagine This

Remember that company we just acquired? Not only is customer creditworthiness apt to cost us another $80 million, but our walk-through of distribution, call-center, and production facilities had a raft of negatively impacting issues with health and safety, environmental, and intellectual property all located in places rife with fraud and corruption.

Not only that, but recent (5 years ago!) hurricane damage has still not been rectified. Our major insurance carrier has also just pulled out of underwriting your fire-related losses. To seemingly top it all off, three VP's of regional manufacturing have just been indicted for padding vendor accounts for kickbacks. To add insult to injury, employee turnover rates are over 40%.

Previously we studied the stylized facts of financial variables, market risk, and credit risk. A common theme runs through this data and outcomes: thick tails, high Value at Risk and Expected Shortfall. Rare events can have strong and persistent effects. Then there is the probability of contagion that allows one bad market to infect other markets.

Now we need to consider more formally the impact of extreme, but rare events on financial performance. To do this we will start off with modeling potential loss in two dimensions, frequency of loss and severity. Frequency of losses, modeled with a Poisson process. The Poisson process is at the heart of Markov credit risks. Each transition probability stems from an average hazard rate of defaults in a unit of time and a segment of borrowers. Here we will use a simplified sampling of frequencies of events drawn from a Poisson process similar to the one we used in credit risk measurement.

Severity is defined as currency based loss. In credit this is the unrecoverable face value, accrued interest, and administrative expenses of a defaulted credit instrument such as a loan. In operational risk this loss could be the value of customers lost due to a cyber

breach, or suppliers failing to deliver critical goods and services resulting in unmet demand, or reputation with investors leading to a flight of capital. We will typically use gamma or generalized Pareto distributions to help us model severity. In some equity markets, the log-normal distribution is prevailing practice.

- Mean Excess Loss from reliability and vulnerability analysis
- Historical data
- VaR and ES again

8.2 What is Operational Risk?

This is a third major category of risk that includes everything from the possibility of process failure, to fraud, to natural (and our own homemade) disaster, errors, omissions, in other words, having lots of really bad days.

In a nutshell we measure operational risk using `frequency` and `severity`. Again there is analogy to credit risk where the frequency is the probability of default and severity is recoverable exposure. Here we think of the probability of loss as how often a loss could occur, its `likelihood`. Infused into `likelihood` are opinions about "how often," "how fast," "how long" (before detection, response, correction), "how remote" (or "imminent"), among other potential suspects.

On the other hand We think of `severity` as the monetary loss itself. This loss is beyond a `threshold`, and thus the focus on "extreme" value methods of dealing with distributions. The "extreme" will be the tail of the distribution where probably but rare events with large magnitudes roam.

In typical experience in most organizations, there is no good time series or cross-sections of data on which to rely for measurements of operational risk. For example, the emerging and immanent risks of botnets wreaking havoc in our computer operating systems have really only a little loss data available for analysis. The problem with this data is that only certain aspects of the data are disclosed, they are supremely idiosyncratic, that is, applicable to highly specialized environments, and often seem far to much an estimate of enterprise-wide loss. Where we do have some data, we have established ways of working with severity distributions that make will sense of the shape of the data.

8.2.1 A working example

Suppose management, after much discussion and challenge by subject matter experts, determines that

1. The median loss due to a vulnerability of a known cyber breach is $60 million
2. With an average (or median) frequency of 25%,
3. All in a 12 month cycle.

4. Management also thinks that the variation in their estimates is about $20 million.

We sometimes call this the "volatility of the estimate" as it expresses the degree of managerial uncertainty about the size of the loss.

1. How would we quantify this risk?
2. How would we manage this risk?

Here are some steps management might take to measure and mitigate this operational risk:

1. Develop a cyber risk taxonomy, For example look at NIST and ISO for guidance.
2. Develop clear definitions of frequency and severity in terms of how often and how much. Calibrate to a metric like operational income.
3. Develop range of operational scenarios and score their frequency and severity of impact on operating income.
4. Aggregate and report ranges of median and quantile potential loss.
5. Develop mitigation scenarios and their value. Subtract this value from potential loss and call it retained risk.
6. Given the organization's risk tolerance, determine if more mitigation needs to be done.

This last step puts a "value" on operational risk scenarios using a rate of error that, if exceeded, would put the organization at ongoing risk that management is not willing to tolerate. This "significance level" in data analytics is often called "alpha" or α. An $\alpha = 5\%$ indicates that management, in a time frame such as one year, is not willing to be wrong more than 1 in 20 times about its experience of operational risk. Management believes that they are 95% $(1 - \alpha)$ confident in their measurement and management of risk. It is this sort of thinking that is at the heart of statistically based confidence intervals and hypothesis testing.

8.3 How Much?

Managers can use several probability distributions to express their preferences for risk of loss. A popular distribution is the `gamma` severity function. This function allows for skewness and "heavy"" tails. Skewness means that loss is attenuated and spread away from the mean or median, stretching occurrences further into the tail of the loss distribution.

The gamma disstribution is fully specified by shape, α, and scale, β, parameters. The shape parameter is a dimensionless number that affects the size and skewness of the tails. The scale parameter is a measure of central tendency "scaled" by the standard deviation of the loss distribution. Risk specialists find this distribution to be especially useful for time-sensitive losses.

We can specify the α shape and β scale parameters using the mean, μ, and standard deviation, σ of the random severities, X as

$$\beta = \mu/\sigma^2,$$

and

$$\alpha = \mu\beta$$

and this is the same as

$$\alpha = \mu^2/\sigma^2$$

Here β will have dimension of $LOSS^{-1}$, while α is dimensionless. The probability that a loss will be exactly x, conditional on α and β is

$$f(x|alpha, \beta) = \frac{\beta^\alpha x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)}$$

Where

$$\Gamma(x) = \int_0^\infty x^{t-1} e^{-x} dx$$

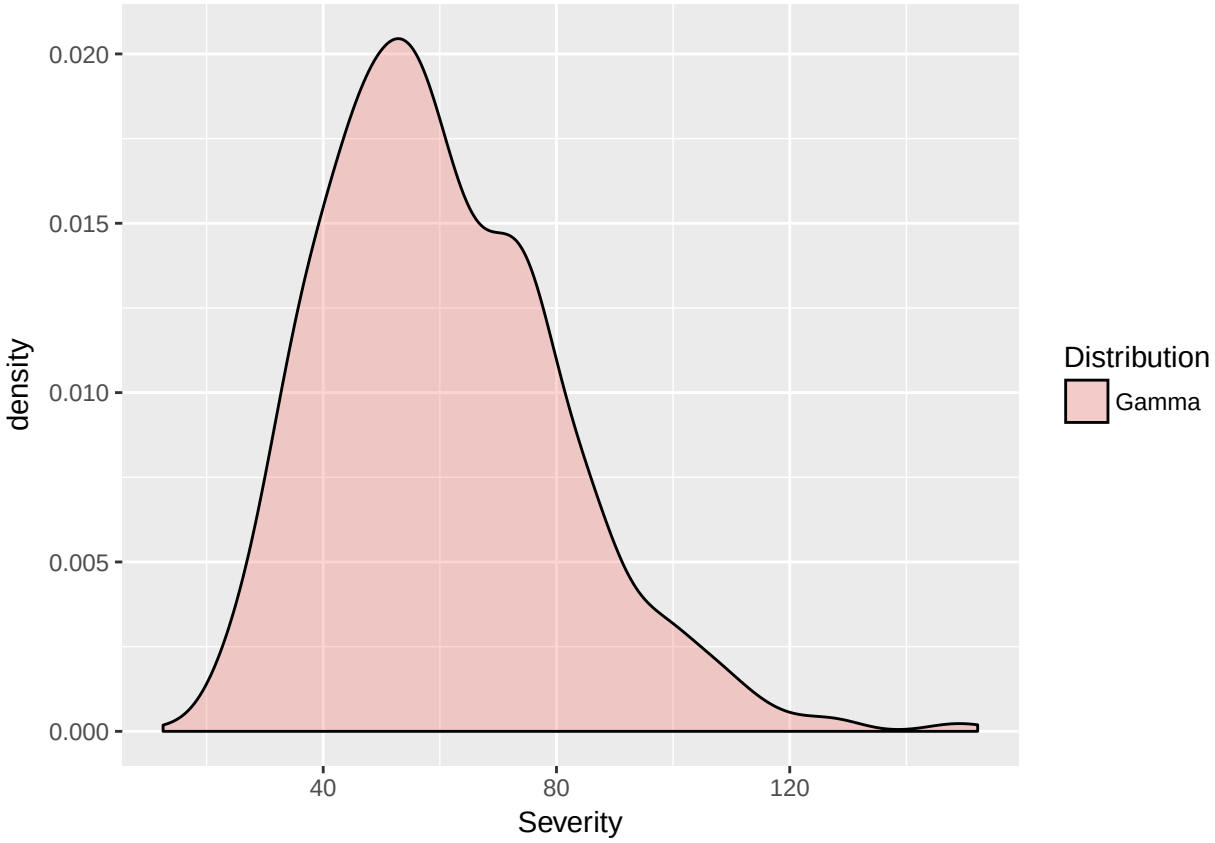
Enough of the math, although very useful for term structure interpolations, and transforming beasts of expressions into something tractable. Let's finally implement into R.

```
set.seed(1004)
n.sim <- 1000
mu <- 60  ## the mean
sigma <- 20  ## management's view of how certain they think their estimates are
sigma.sq <- sigma^2
beta <- mu/sigma.sq
alpha <- beta * mu
severity <- rgamma(n.sim, alpha, beta)
summary(severity)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   12.54   45.44   57.48   60.30   73.47  152.23
```

The distribution is dispersed from a low of 15 to a high of over 120 million dollars. We might check with management that this low and high (and intermediate levels) are reasonable. Let's graph the distribution. Here we form a data frame in `gamma.sev` to create factor values for the `ggplot` fill.

```
require(ggplot2)
gamma.sev <- data.frame(Severity = severity,
    Distribution = rep("Gamma", each = n.sim))
ggplot(gamma.sev, aes(x = Severity, fill = Distribution)) +
    geom_density(alpha = 0.3)
```



8.3.1 Severity thresholds

We can add thresholds based on value at risk (`VaR`) and expected shortfall `ES`. Here we calculate `VaR.sev` as the quantile for a confidence level of $1 - \alpha$.

```
alpha.tolerance <- 0.05
(VaR.sev <- quantile(severity, 1 - alpha.tolerance))
(ES.sev <- mean(gamma.sev$Severity[gamma.sev$Severity >
    VaR.sev]))
```

Plot them onto the distribution using the `geom_vline` feature in `ggplot2`. What is the relationship between `VaR` and `ES`?

```
ggplot(gamma.sev, aes(x = Severity, fill = Distribution)) +
    geom_density(alpha = 0.3) + geom_vline(xintercept = VaR.sev,
    color = "red") + geom_vline(xintercept = ES.sev,
    color = "blue")
```

Here are the results of these two exercises.

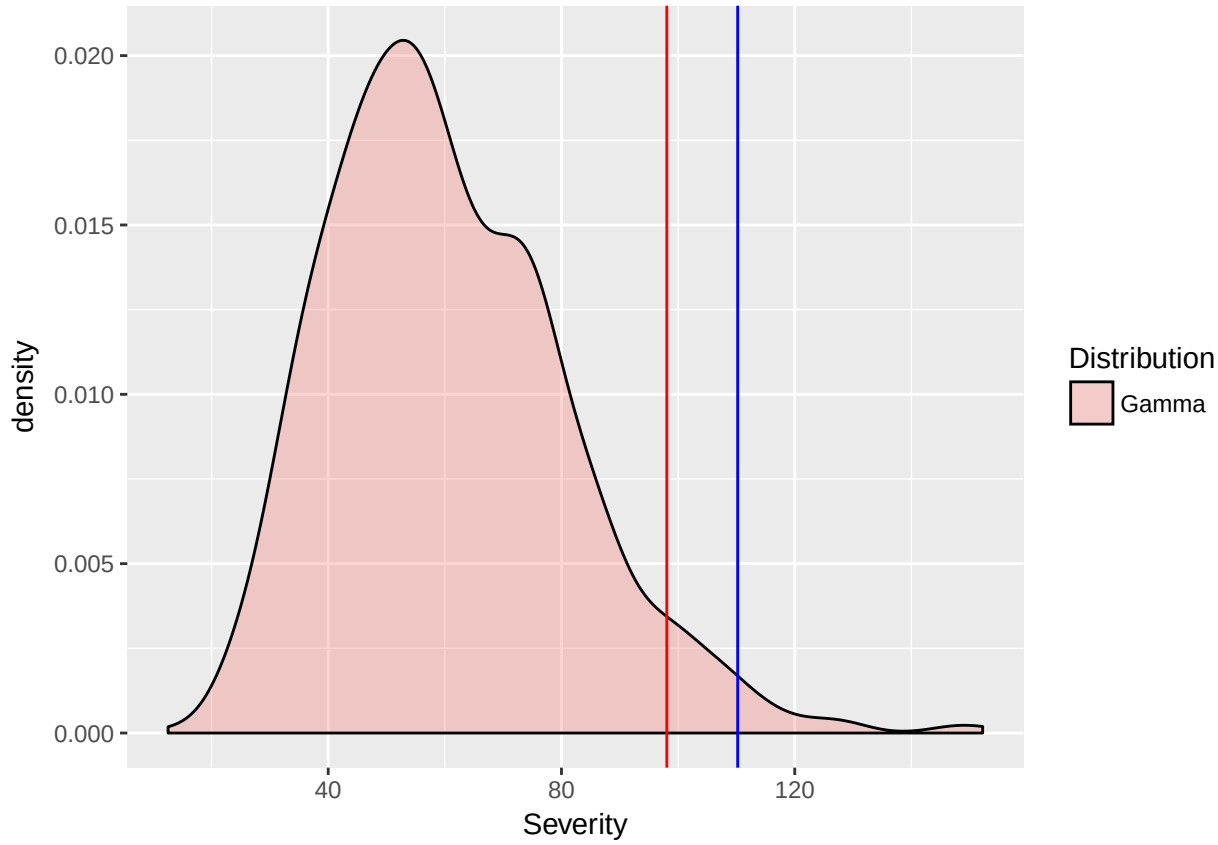
```
alpha.tolerance <- 0.95
(VaR.sev <- quantile(severity, alpha.tolerance))
```

```
##      95%
## 98.05779
```

```
(ES.sev <- mean(gamma.sev$Severity[gamma.sev$Severity >
    VaR.sev]))
```

```
## [1] 110.2134
```

```
ggplot(gamma.sev, aes(x = Severity, fill = Distribution)) +
    geom_density(alpha = 0.3) + geom_vline(xintercept = VaR.sev,
    color = "red") + geom_vline(xintercept = ES.sev,
    color = "blue")
```



What is the relationship between `Var` and `ES`? As it should be, the VaR is less than the ES. For this risk of potential cyber breach vulnerability, management might think of setting aside capital proportional to `ES.sev - quantile(severity, 0.50))`, that is, the difference between the expected shortfall and the median potential loss. The modifier "proportional" needs some further specification. We have to consider the likelihood of such a risk event.

8.4 How Often?

We usually model the frequency of a loss event with a `poisson` distribution. This is a very basic counting distribution defined by the rate of arrival of an event, λ in a unit of time. This rate is jus what we struggled to estimate for credit rating transitions.

Again for completeness sake and as a reference, we define the probability of exactly x risk events arriving at a rate of λ as

$$Probability[x|\lambda] = \frac{\lambda^x e^{-\lambda}}{x!}$$

Management might try to estimate the count of events in a unit of time using hazard rate models (same λ as in credit migration transitions). Here we simulate management's view with $\lambda = 0.25$. Our interpretation of this rating is that management believes that a vulnerability event will happen in the next 12 months once every 3 months (that is once every four months).

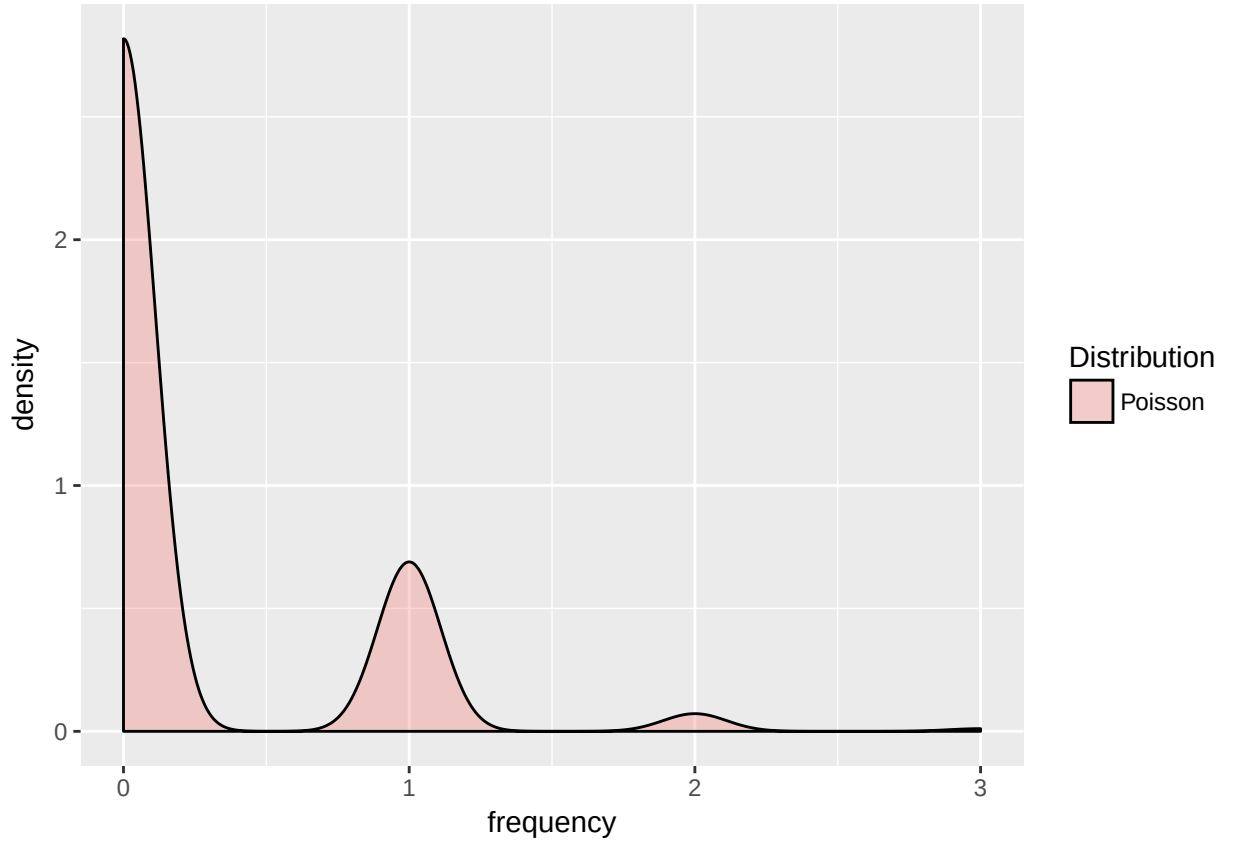
We simulate by using `n.sim` trials drawn with the `rpois()` function conditional on $\lambda = 0.25$.

```
n.sim <- 1000
lambda <- 0.25
frequency <- rpois(n.sim, lambda)
summary(frequency)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   0.241   0.000   3.000
```

`summary` is not very informative so we also plot this distribution using this code:

```
poisson.freq <- data.frame(Frequency = frequency,
    Distribution = rep("Poisson", each = n.sim))
ggplot(poisson.freq, aes(x = frequency,
    fill = Distribution)) + geom_density(alpha = 0.3)
```



This shows a smooth version of the discrete event count: more likely zero, then one event, then two , then three events, all of which devides the 12 months into four, 3-month event intervals.

8.5 How Much Potential Loss?

Now we get to the nub of the matter, how much potentially can we lose? This means we need to combine frequency with severity. We need to "convolve" the frequency and severity distributions together to form one loss distribution. "To convolve" means that for each simulated cyber dollar loss scenario we see whether the scenario occurs at all using the many `poisson` frequency scenarios.

This convolution can be accomplished in the following code:

```
loss <- rpois(n.sim, severity * lambda)
summary(loss)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   10.00   14.00   15.02   19.00   44.00
```

Some notes are in order:

1. This code takes each of the `gamma` distributed severities (`n.sim` of them) asks how often one of them will occur (`lambda`).
2. It then simulates the frequency of these risk events scaled by the `gamma` severities of the loss size.
3. The result is a *convolution* of all `n.sim` of the severities with all `n.sim` of the frequencies. We are literally combining each scenario with every other scenario.

Let's visualize our handiwork and frame up the data. Then calculate the risk measures for the potential loss as

```
loss.rf <- data.frame(Loss = loss, Distribution = rep("Potential Loss",
    each = n.sim))
(VaR.loss.rf <- quantile(loss.rf$Loss,
    1 - alpha.tolerance))
```

```
## 5%
##  6
```

```
(ES.loss.rf <- mean(loss.rf$Loss[loss.rf$Loss >
    VaR.loss.rf]))
```

```
## [1] 15.65462
```

Again `VaR` is a quantile and `ES` is the mean of a filter on the convolved losses in excess of the `VaR` quantile.

8.5.1 Try this exercise

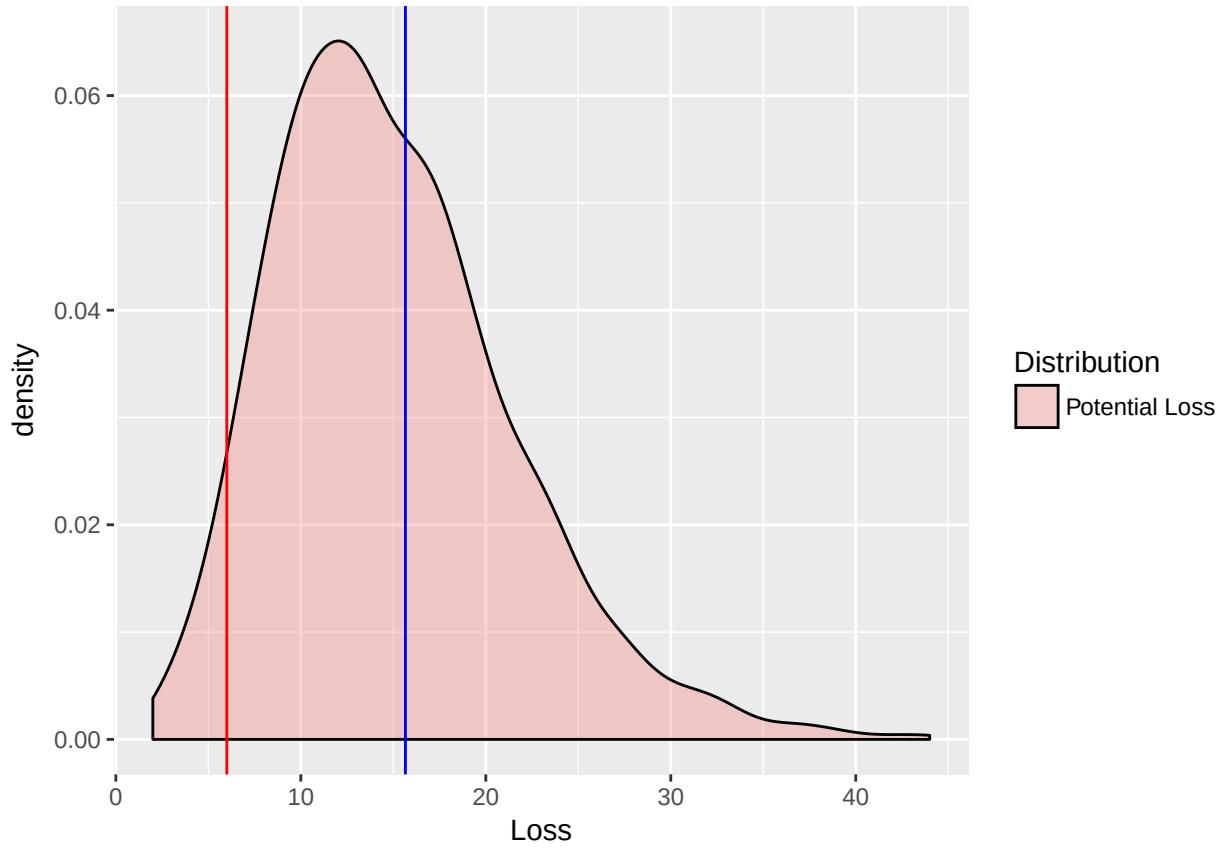
1. Plot the loss function and risk measures with

```
ggplot(loss.rf, aes(x = Loss, fill = Distribution)) +
    geom_density(alpha = 0.3) + geom_vline(xintercept = VaR.loss.rf,
    color = "red") + geom_vline(xintercept = ES.loss.rf,
    color = "blue")
```

2. What would you advise management about how much capital might be needed to underwrite these losses?

Some results using this code:

```
ggplot(loss.rf, aes(x = Loss, fill = Distribution)) +
    geom_density(alpha = 0.3) + geom_vline(xintercept = VaR.loss.rf,
    color = "red") + geom_vline(xintercept = ES.loss.rf,
    color = "blue")
```



Management should consider different risk tolerances first. Then the organization can decide on the difference between the VaR (or ES) and the median loss.

```
loss.rf <- data.frame(Loss = loss, Distribution = rep("Potential Loss",
    each = n.sim))
(VaR.loss.rf <- quantile(loss.rf$Loss,
    1 - alpha.tolerance))
```

```
## 5%
##  6
```

```
(ES.loss.rf <- mean(loss.rf$Loss[loss.rf$Loss >
    VaR.loss.rf]))
```

```
## [1] 15.65462
```

```
(Reserve.loss.rf <- ES.loss.rf - quantile(loss.rf$Loss,
    0.5))
```

```
##      50%
## 1.654623
```

If this were a bank, then managers would have calculated the capital requirement (almost a la Basle III).

8.6 We Have History

Now suppose we have some history of losses. In this context, we use an extreme tail distribution called the Generalized Pareto Distribution (GPD) to estimate historical loss parameters. This distribution models the the tails of any other distribution. We would have split a `poisson-gamma` loss distribution into two parts, a body, and a tail. The tail would begin at a specified threshold.

The GPD is especially well-known for a single property: for very high thresholds, GPD not only well describes the behavior in excess of the threshold, but the mean excess over the threshold is linear in the threshold. From this property we get more intuition around the use of Expected Shortfall as a coherent risk measure. In recent years we well exceeded all Gaussian and Student's t thresholds in distressed markets.

For a random variate x, the GPD distribution is defined for The scale parameter β and shape parameter $\xi \geq 0$ as:

$$g(x; \xi \geq 0) = 1 - (1 + x\xi/\beta)^{-1/\xi}$$

and when the shape parameter $\xi = 0$, the GPD becomes the exponential distribution dependent only on the scale parameter β:

$$g(x; \xi = 0) = 1 - exp(-x/\beta)$$

Now for the infamous property. If u is an upper (very high) threshold, then the excess of threshold function for the GPD is

$$e(u) = \frac{\beta + \xi u}{1 - \xi}$$

This simple measure is linear in thresholds. It will allow us to visualize where rare events begin (See McNeil, Embrechts, Frei (2015, Chapter 5)). We will use as a threshold the exceedance that begins to make the excesses linear.

8.7 Fire Losses

Now to get to some data. The first thing we do is load the well researched Danish fire claims data set in millions of Danish Kroner collected from 1980 to 1990 as a time series object. Then we will plot the Mean Excesses (of thresholds). These are simply the mean of $e(u)$, a function of the parameters of the GPD.[1]We will follow very closely the code and methodology used by McNeil et al. currently on the `qrmtutorial` site at http://www.qrmtutorial.org/r-code)

[1](

```
library(QRM)
## Load Danish fire loss data and look
## at structure and content
data(danish)
str(danish)
```

```
## Time Series:
##  Name:               object
## Data Matrix:
##  Dimension:          2167 1
##  Column Names:       FIRE.LOSSES
##  Row Names:          1980-01-03  ...  1990-12-31
## Positions:
##  Start:              1980-01-03
##  End:                1990-12-31
## With:
##  Format:             %Y-%m-%d
##  FinCenter:          GMT
##  Units:              FIRE.LOSSES
##  Title:              Time Series Object
##  Documentation:      Wed Mar 21 09:48:58 2012
```

```
head(danish, n = 3)
```

```
## GMT
##            FIRE.LOSSES
## 1980-01-03    1.683748
## 1980-01-04    2.093704
## 1980-01-05    1.732581
```

```
tail(danish, n = 3)
```

```
## GMT
##            FIRE.LOSSES
## 1990-12-30    4.867987
## 1990-12-30    1.072607
## 1990-12-31    4.125413
```

```
## Set Danish to a numeric series
## object if a time series
if (is.timeSeries(danish)) danish <- series(danish)
danish <- as.numeric(danish)
```

We review the dataset.

```
summary(danish)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   1.000   1.321   1.778   3.385   2.967 263.250
```

Our next task is to sort out the losses and rank order unique losses with this function.

```
n.danish <- length(danish)
## Sort and rank order unique losses
rank.series <- function(x, na.last = TRUE) {
    ranks <- sort.list(sort.list(x, na.last = na.last))
    if (is.na(na.last))
        x <- x[!is.na(x)]
    for (i in unique(x[duplicated(x)])) {
        which <- x == i & !is.na(x)
        ranks[which] <- max(ranks[which])
    }
    ranks
}
```

Now we use the `rank.series` function to create the mean excess function point by point through the sorted series of loss data. In the end we want to cumulatively sum data in a series of successive thresholds.

```
danish.sorted <- sort(danish)  ## From low to high
## Create sequence of high to low of
## indices for successive thresholds
n.excess <- unique(floor(length(danish.sorted) -
    rank.series(danish.sorted)))
points <- unique(danish.sorted)  ## Just the unique losses
n.points <- length(points)
## Take out last index and the last
## data point
n.excess <- n.excess[-n.points]
points <- points[-n.points]
## Compute cumulative sum series of
## losses
excess <- cumsum(rev(danish.sorted))[n.excess] -
    n.excess * points
excess.mean <- excess/n.excess  ## Finally the mean excess loss series
```

So much happened here. Let's spell this out again:

1. Sort the data
2. Construct indices for successive thresholds
3. With just unique losses throw out the last data point, and its index
4. Now for the whole point of this exercise: calculate the cumulative sum of losses by threshold (the `[n.excess]` operator) in excess of the cumulative threshold `n.excess*points`
5. Then take the average to get the mean excess loss series.

8.7.1 Try this exercise

1. Build a data frame with components `Excess.Mean`, `Thresholds`, and type of `Distribution`.

```
library(ggplot2)
omit.points <- 3
excess.mean.df <- data.frame(Excess.Mean = excess.mean[1:(n.points -
    omit.points)], Thresholds = points[1:(n.points -
    omit.points)], Distribution = rep("GPD",
    each = length(excess.mean[1:(n.points -
        omit.points)])))
```

2. plot the mean excess series against the sorted points. In this case we will omit the last 3 mean excess data points.

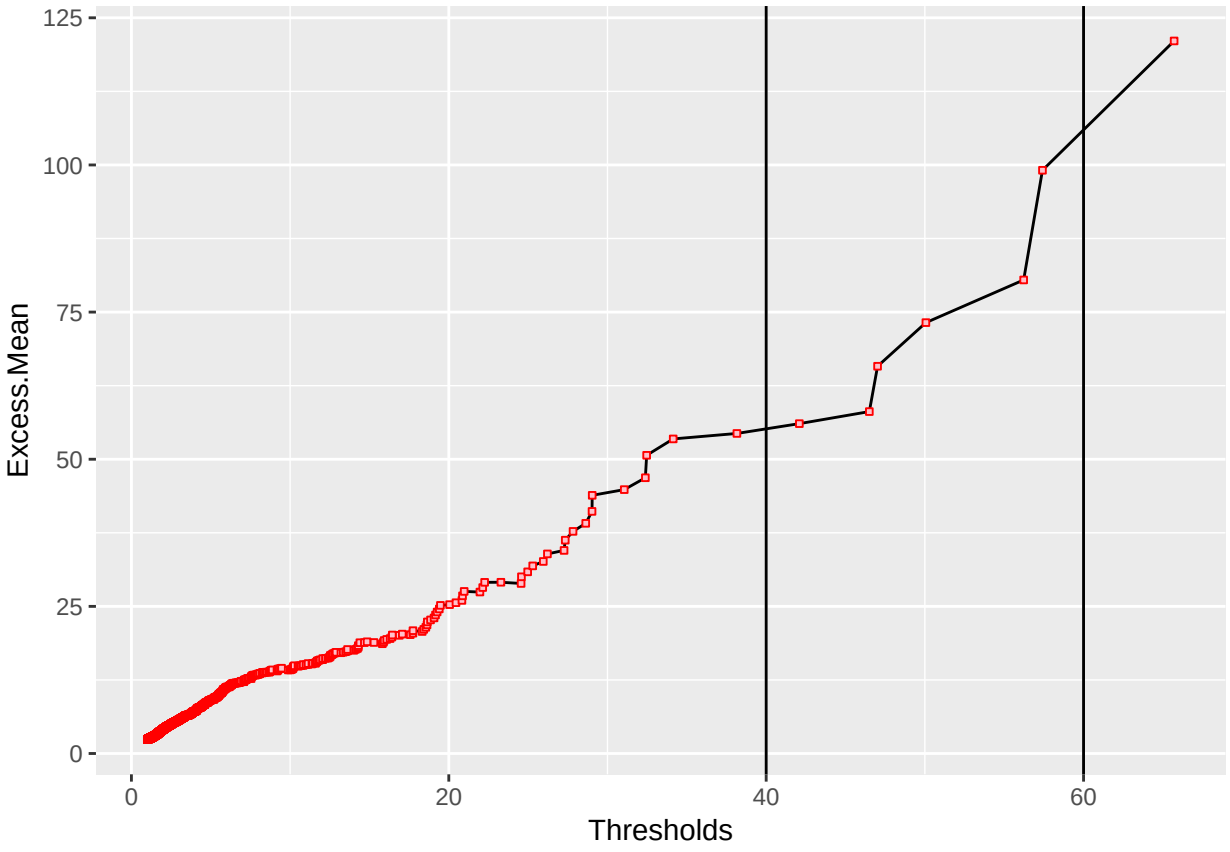
```
## Mean Excess Plot
ggplot(excess.mean.df, aes(x = Thresholds,
    y = Excess.Mean)) + geom_line() +
    geom_point(size = 1, shape = 22,
        colour = "red", fill = "pink") +
    geom_vline(xintercept = 40) + geom_vline(xintercept = 60)
## Plot density
ggplot(excess.mean.df, aes(x = Excess.Mean,
    fill = Distribution)) + geom_density() +
    xlim(0, 75)
```

Some results using this code:

```
library(ggplot2)
omit.points <- 3
excess.mean.df <- data.frame(Excess.Mean = excess.mean[1:(n.points -
    omit.points)], Thresholds = points[1:(n.points -
    omit.points)], Distribution = rep("GPD",
    each = length(excess.mean[1:(n.points -
        omit.points)])))
```

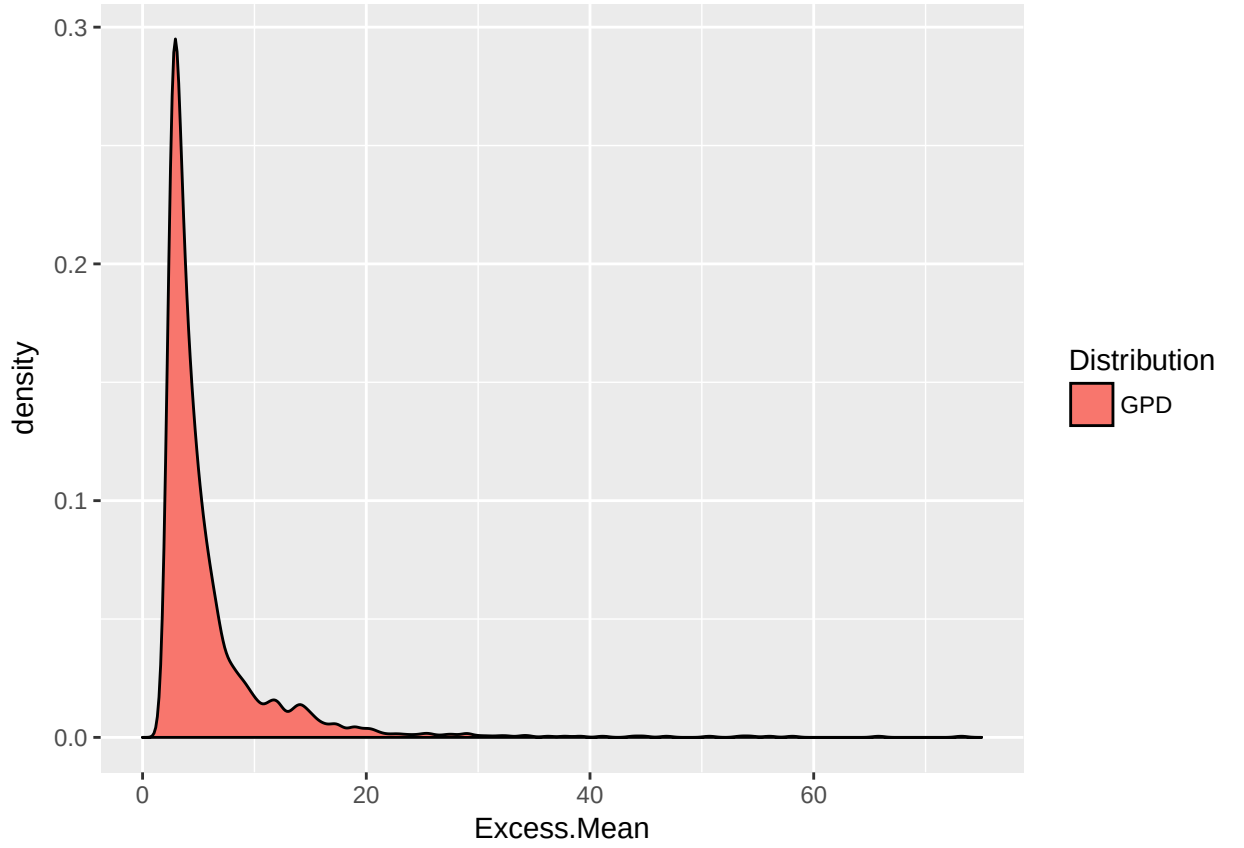
We can interpret this data by using the `[1:(n.points - omit.points)]` throughout to subset the mean excess series, the thresholds, and an indicator of the type of distribution.

```
## Mean Excess Plot
ggplot(excess.mean.df, aes(x = Thresholds,
    y = Excess.Mean)) + geom_line() +
    geom_point(size = 1, shape = 22,
        colour = "red", fill = "pink") +
    geom_vline(xintercept = 40) + geom_vline(xintercept = 60)
```



This plot shows that the mean of any excess over each and every point grows linearly with each threshold. As the mean excesses get larger, they also become more sparse, almost like outliers. They are the rare events we might want to mitigate. We might look at the vertical lines at 40 and 60 million kroner to possibly design retention and limits for a contract to insure against experiencing loss in this region.

```
## Plot density
ggplot(excess.mean.df, aes(x = Excess.Mean,
    fill = Distribution)) + geom_density() +
    xlim(0, 75)
```



This density plot confirms the pattern of the mean excess plot. The mean excess distribution is understood by engineers as the mean residual life of a physical component. In insurance on the other hand, if the random variable X is a model of insurance losses, like the `danish` data set, then the conditional mean $E(X - u|X > u)$ is the expected claim payment per loss given that the loss has exceeded the deductible of u. In this interpretation, the conditional mean $E(X - t|X > t)$ is called the mean excess loss function.

8.8 Estimating the Extremes

Now we borrow some Generalized Pareto Distribution code from our previous work in market risk and apply it to the `danish` data set.

```
## library(QRM)
alpha.tolerance <- 0.95
u <- quantile(danish, alpha.tolerance,
    names = FALSE)
fit.danish <- fit.GPD(danish, threshold = u)  ## Fit GPD to the excesses
(xi.hat.danish <- fit.danish$par.ses[["xi"]])  ## fitted xi
```

```
## [1] 0.1351274
```

```
(beta.hat.danish <- fit.danish$par.ses[["beta"]])  ## fitted beta
```

```
## [1] 1.117815
```

Let's put these estimates to good use. Here are the closed form calculations for value at risk and expected shortfall (no random variate simulation!) using McNeil, Embrechts, Frey (2015, chapter 5) formulae:

```
## Pull out the losses over the
## threshold and compute excess over
## the threshold
loss.excess <- danish[danish > u] - u  ## compute the excesses over u
n.relative.excess <- length(loss.excess)/length(danish)  ## = N_u/n
(VaR.gpd <- u + (beta.hat.danish/xi.hat.danish) *
    (((1 - alpha.tolerance)/n.relative.excess)^(-xi.hat.danish) -
        1))
```

```
## [1] 9.979336
```

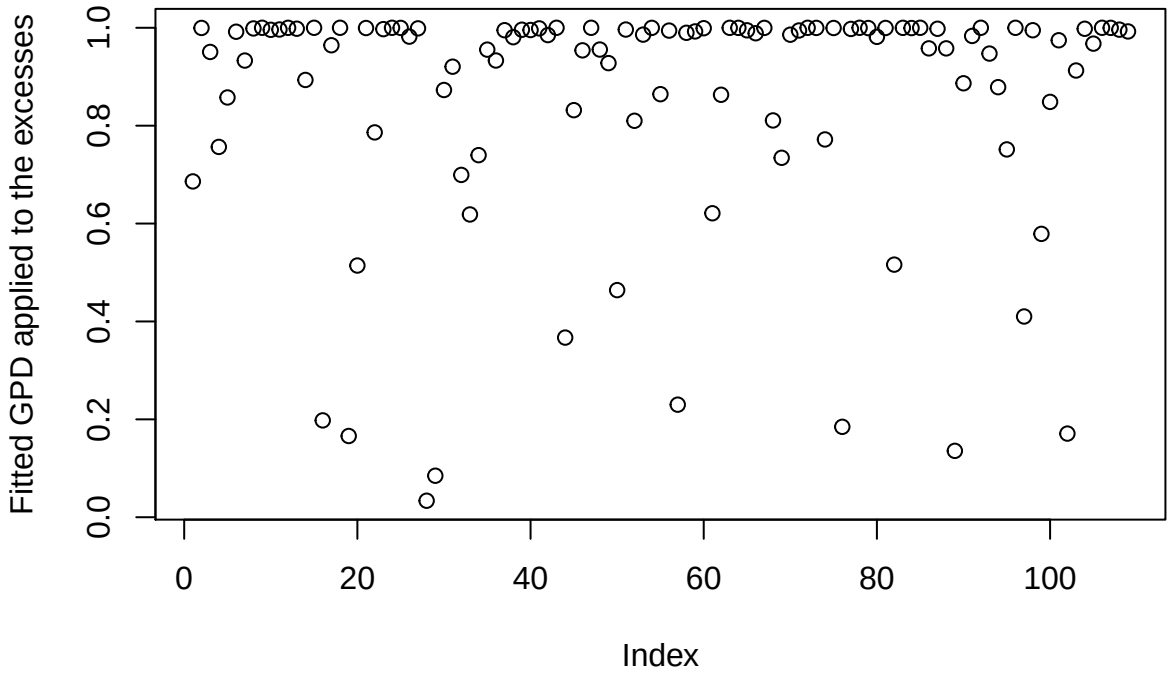
```
(ES.gpd <- (VaR.gpd + beta.hat.danish -
    xi.hat.danish * u)/(1 - xi.hat.danish))
```

```
## [1] 11.27284
```

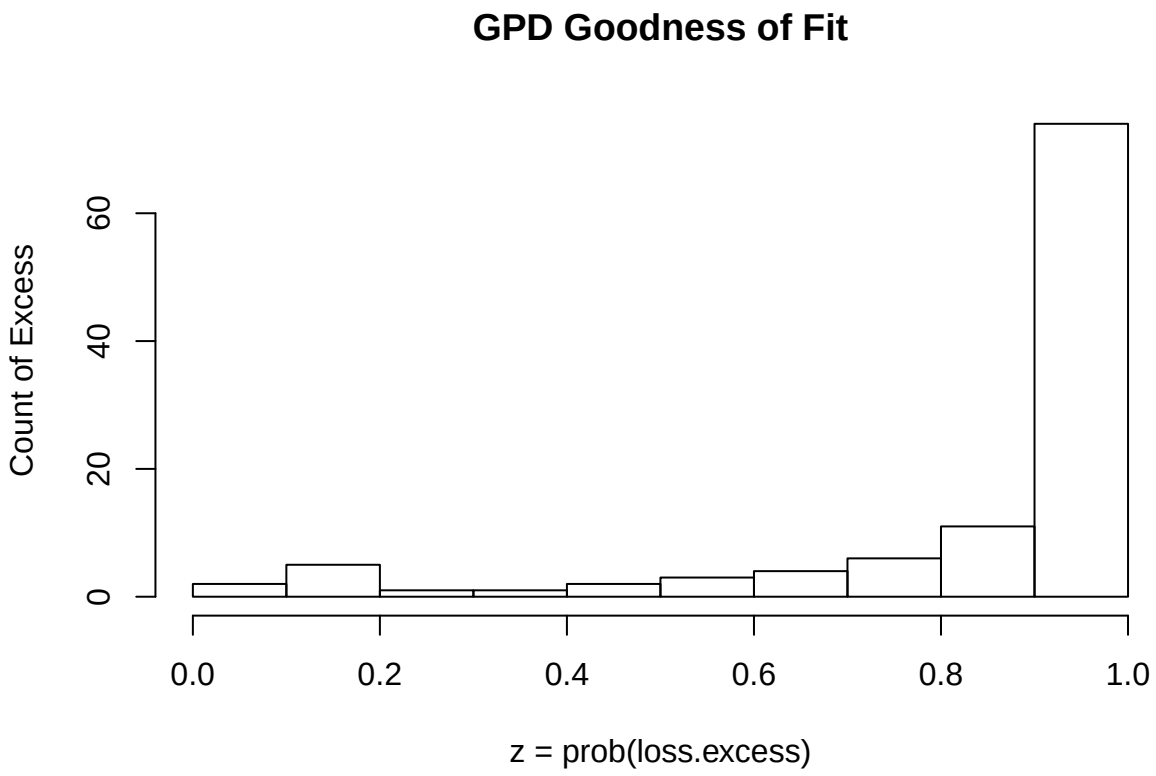
Using these risk measures we can begin to have a discussion around the size of reserves, the mitigation of losses, and the impact of this risk on budgeting for the allocation of risk tolerance relative to other risks.

How good a fit? Let's run this code next.

```
z <- pGPD(loss.excess, xi = xi.hat.danish,
    beta = beta.hat.danish)  ## should be U[0,1]
plot(z, ylab = "Fitted GPD applied to the excesses")  ## looks fine
```




```
hist(z, xlab = "z = prob(loss.excess)",
    ylab = "Count of Excess", main = "GPD Goodness of Fit")
```



And it is as most of the excesses are in the tail!

8.8.1 Try this exercise

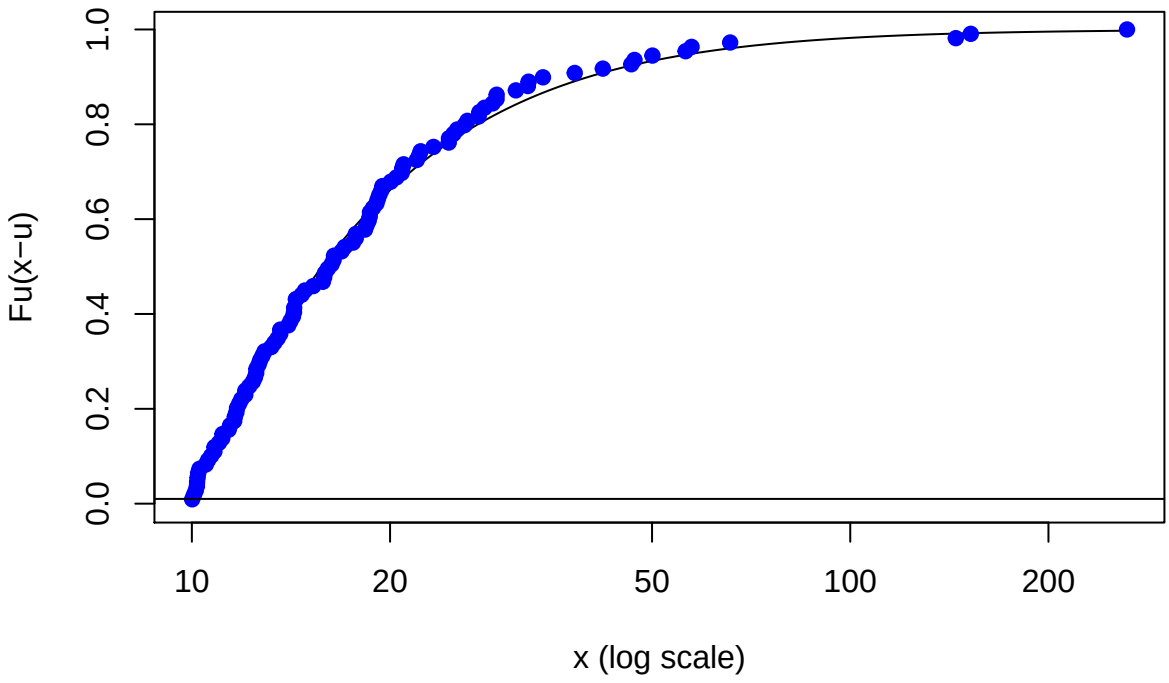
Let's try other thresholds to sensitize ourselves to the GPD fit and coefficients.

```
## Fit GPD model above 10
u <- 10
fit.danish <- fit.GPD(danish, threshold = u)
(RM.danish <- RiskMeasures(fit.danish,
    c(0.99, 0.995)))
```

```
##          p quantile    sfall
## [1,] 0.990 27.28640 58.22848
## [2,] 0.995 40.16646 83.83326
```

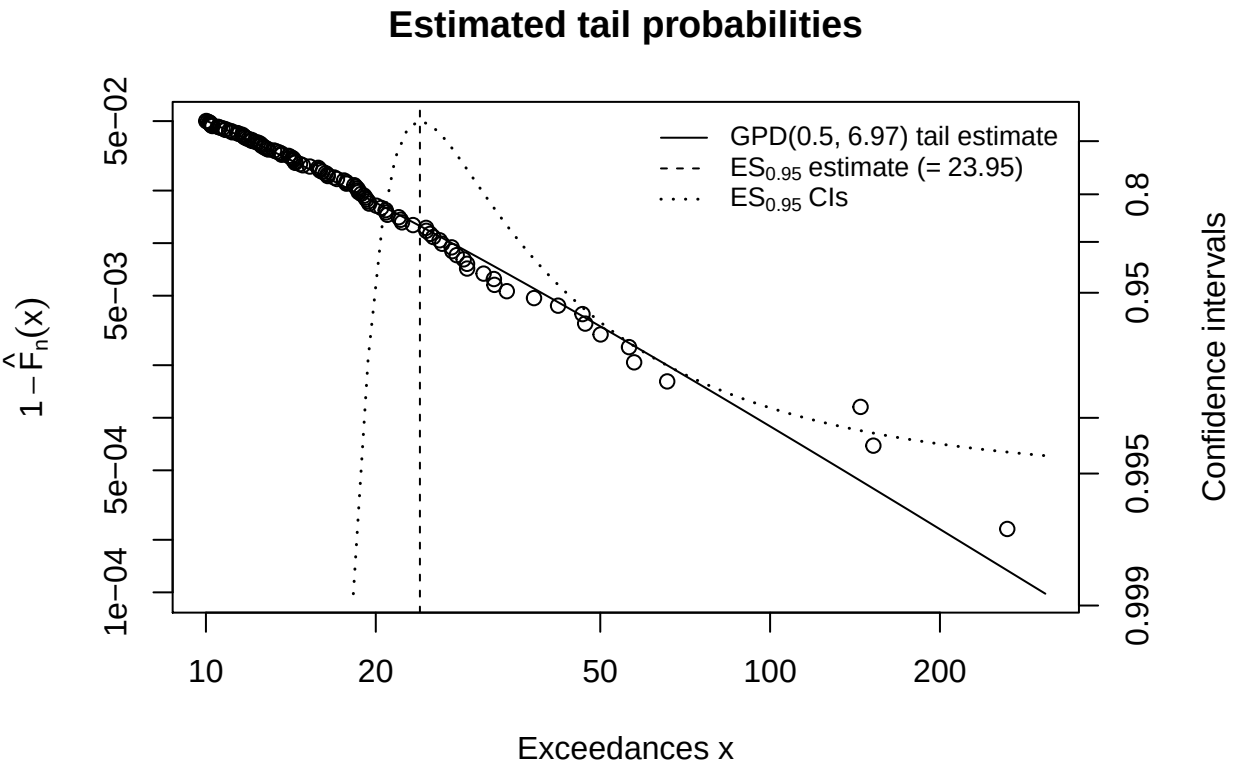
Now let's look at the whole picture with these exceedances. On the vertical axis we have the probabilities that you are in the tail … the extreme loss of this operation. On the vertical axis we have the exceedances (loss over the threshold `u`). The horizontal line is our risk tolerance (this would be `h` percent of the time we don't want to see a loss…)

```
plotFittedGPDvsEmpiricalExcesses(danish,
    threshold = u)
abline(h = 0.01)
```



Let's use the `showRM` function in the `QRM` package to calculate the expected shortfall confidence intervals. The initial `alpha.tolerance` is `1 - h` risk tolerance. We are only looking at the expected shortfall and could also have specified value at risk. What we really want to see is the range of `ES` as we wander into the deep end of the pool with high exceedances. The `BFGS` is described technically here: `https://en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_a`

```
alpha.tolerance <- 0.05
showRM(fit.danish, alpha = 1 - alpha.tolerance,
    RM = "ES", method = "BFGS")
```



`CI` stands for "confidence interval" and we answer this question:

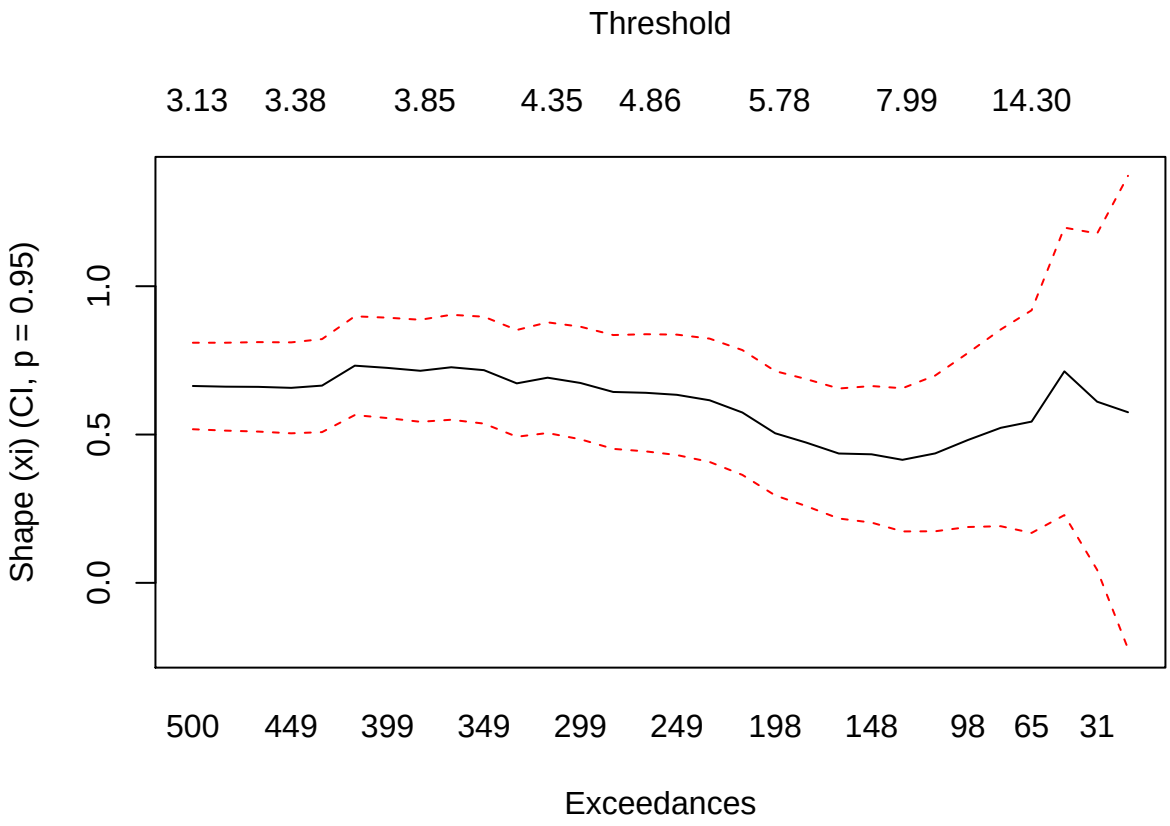
> For each level of tolerance for loss (vertical axis) what is the practical lower and upper limit on the expected shortfall of 23.95 million?

Practically speaking how much risk would managers be willing to retain (lower bound) and how much would you underwrite (upper - lower bounds)?

8.8.2 The shape of things to come

Let's now plot the GPD shape parameter as a function of the changing threshold.

```
## Effect of changing threshold on xi
xiplot(danish)
```



What does `xi` tell us? Mainly information about the relationship between our risk measures. The ratio of `VaR` to 'ES' is $(1-\xi)^{-1}$ if $0 \leq \xi \geq 1$. How bent the tail will be: higher ξ means a heavier tail, and a higher frequency of very large losses.

Again the upper and lower bounds help us diagnose what is happening to our exceedances. The Middle line is the shape parameter at variour thresholds and their corresponding exceedances. Dashed red lines are the "river banks" bounding the upper and lower edges of the tail (by threshold) distribution of the estimated risk measure. Fairly stable?

8.8.3 Example

We can run this code to throw the two popular risk measures together.

```
## Fit GPD model above 20
mod2 <- fit.GPD(danish, threshold = 20)
mod2$par.ests
```

```
##        xi      beta
## 0.6844366 9.6341385
```

```
mod2$par.ses
```

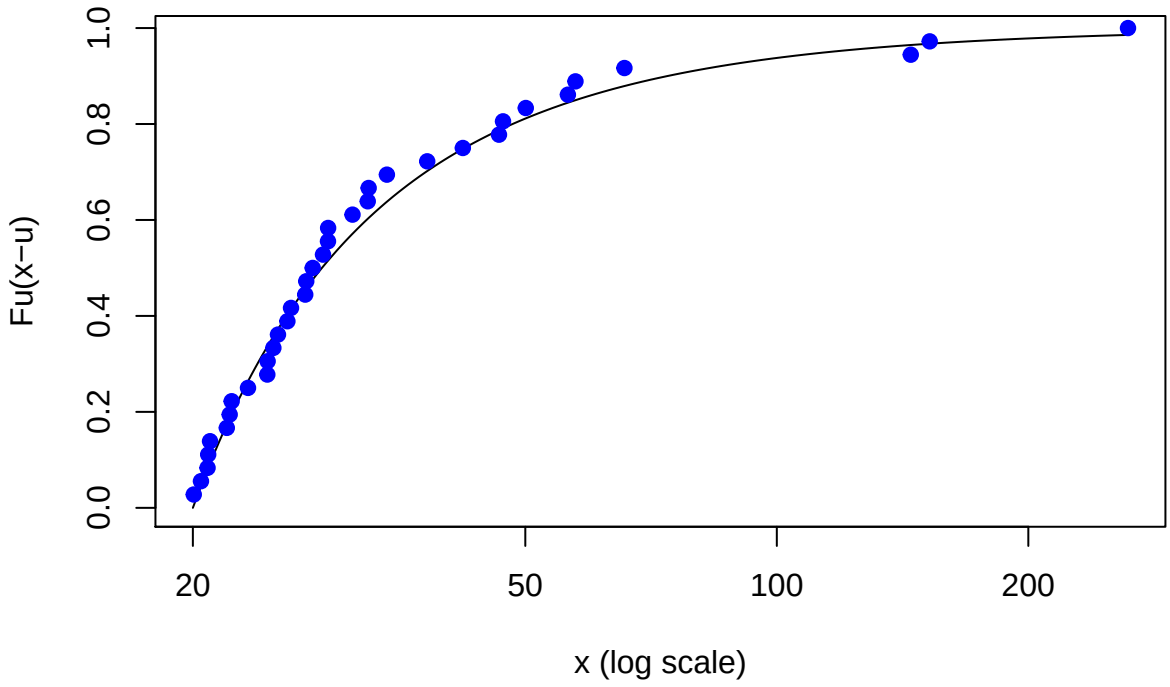
```
##        xi      beta
```

```
## 0.2752081 2.8976652
```

```
mod2$par.ests/mod2$par.ses
```

```
##       xi     beta
## 2.486978 3.324794
```

```
plotFittedGPDvsEmpiricalExcesses(danish,
    threshold = 20)
```



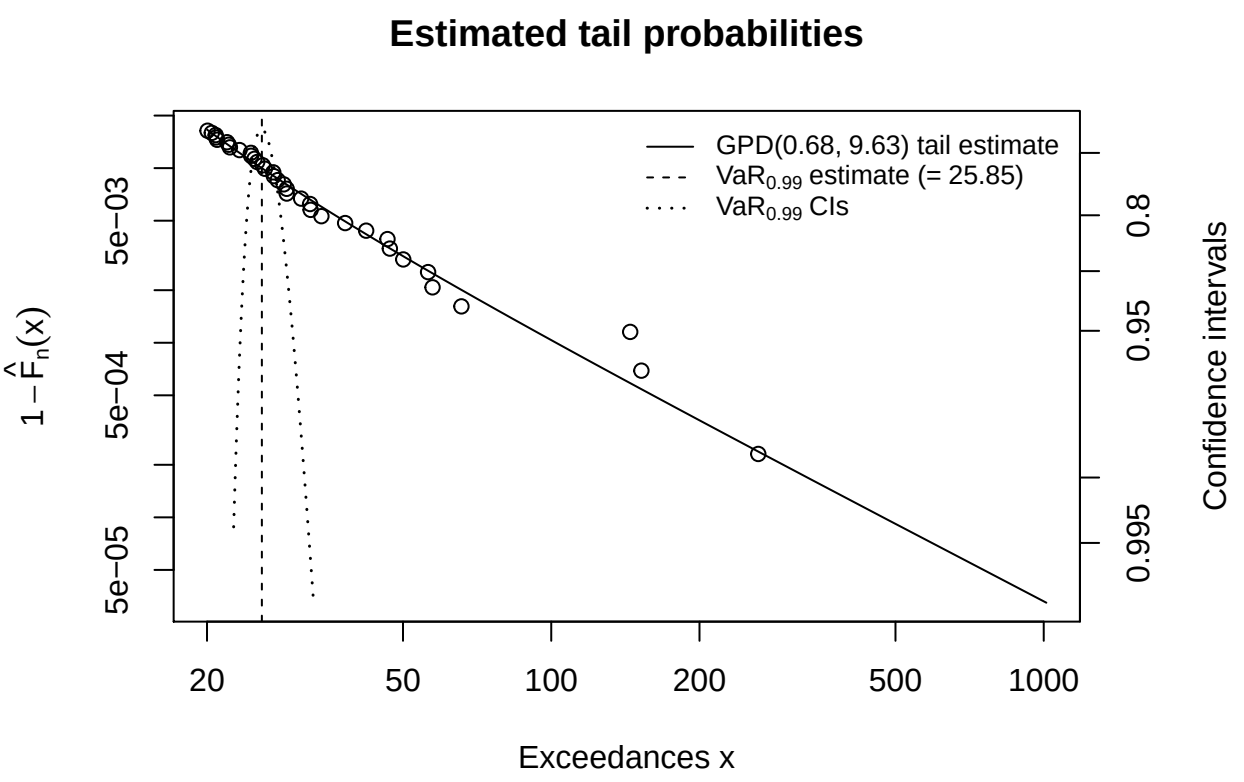

```
(RMs2 <- RiskMeasures(mod2, c(0.99, 0.995)))
```

```
##          p quantile     sfall
## [1,] 0.990 25.84720  69.05935
## [2,] 0.995 37.94207 107.38721
```

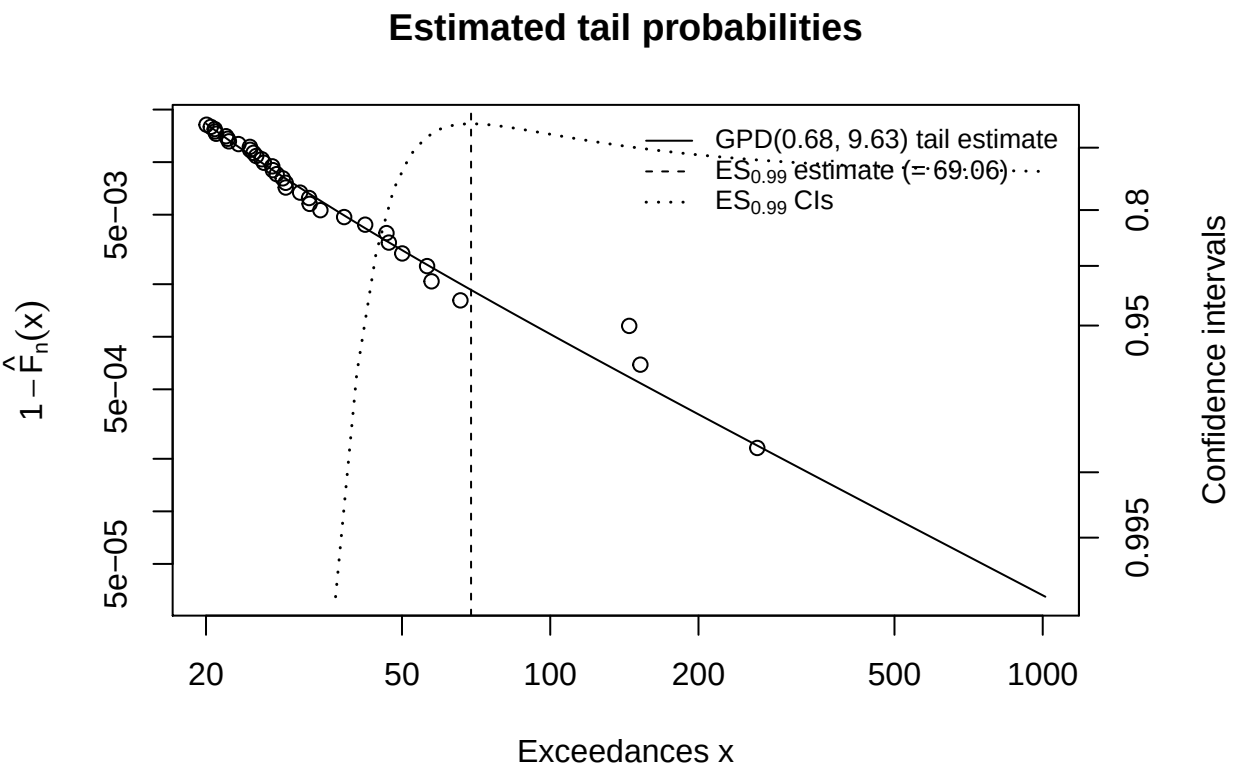
```
RMs2
```

```
##          p quantile     sfall
## [1,] 0.990 25.84720  69.05935
## [2,] 0.995 37.94207 107.38721
```

```
plotTail(mod2)
showRM(mod2, alpha = 0.99, RM = "VaR",
    method = "BFGS")
```




```
showRM(mod2, alpha = 0.99, RM = "ES",
    method = "BFGS")
```



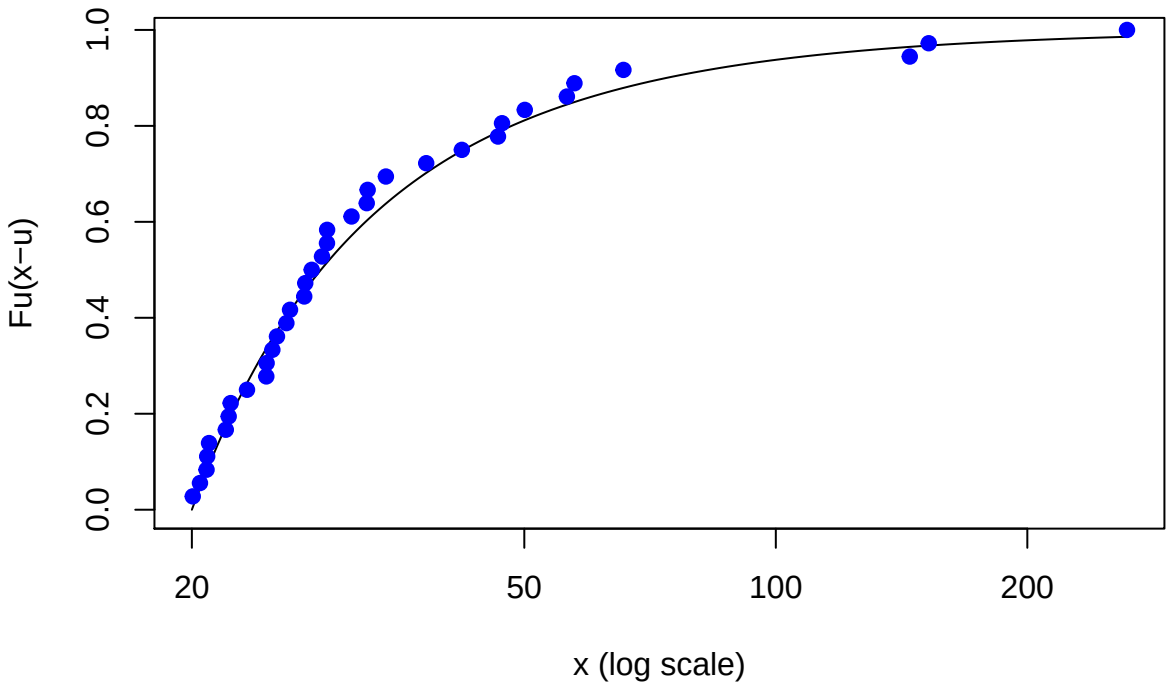
We are just doubling the threshold. Here is the workflow:

1. Interpret the fit with `mod2$par.ests/mod2$par.ses`
2. Interpret the empiral excesses for this threshold.
3. Compute the risk measures.
4. Relate risk tolerance to exceedance.
5. Compare value at risk with expected shortfall.

Some results follow

Run this code first. This is the cumulative probability of an exceedance over a threshold, here 20.

```
plotFittedGPDvsEmpiricalExcesses(danish,
    threshold = 20)
```



Let's interpret across various threshold regions:

1. From 20 to about 40, fairly dense and linear
2. From about 40 to 60, less dense and a more bent slope (ξ is bigger than for lower threshold)
3. Big and less frequent outliers from about 60 to well over 200.

Some managerial implications to spike the discussion could include:

1. Budget for loss in region 1
2. Buy insurance for region 2
3. Consider some loss reserves for region 3

```
mod2 <- fit.GPD(danish, threshold = 20)
mod2$par.ests
```

```
##        xi      beta
## 0.6844366 9.6341385
```

```
mod2$par.ses
```

```
##        xi      beta
## 0.2752081 2.8976652
```

```
(t.value <- mod2$par.ests/mod2$par.ses)
```

```
##       xi     beta
## 2.486978 3.324794
```

The ratio of the parameter estimates to the standard errors of those estimates gives up an idea of the rejection of the hypothesis that the parameters are no different than zero. In R we can do this:

```
(p.value = dt(t.value, df = length(danish) -
    2))
```

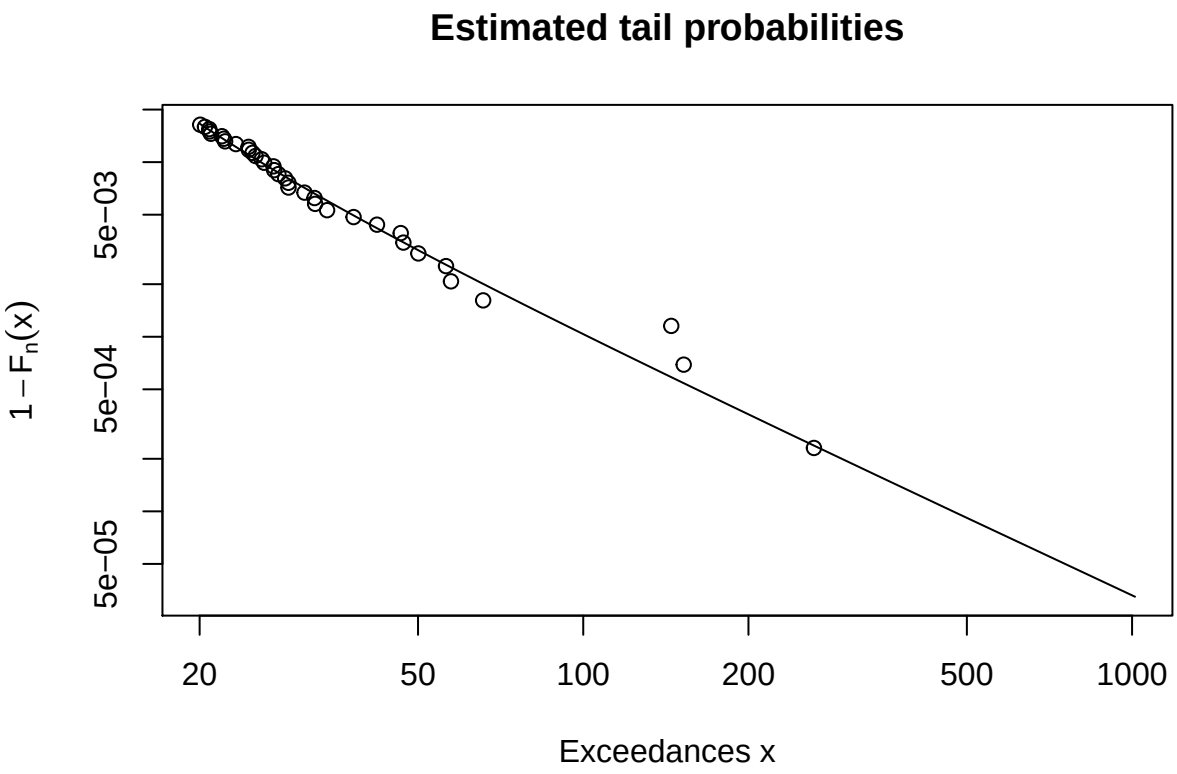
```
##          xi        beta
## 0.018158771 0.001604948
```

Here `df` is the degrees of freedom for `2` estimated parameters. The`p.value`s are very low, meaning there is a very small chance that the estimates are zero. Various risk measures and tail plots can elucidate more interpretation. Here we can use two confidence levels.

```
(RMs2 <- RiskMeasures(mod2, c(0.99, 0.995)))
```

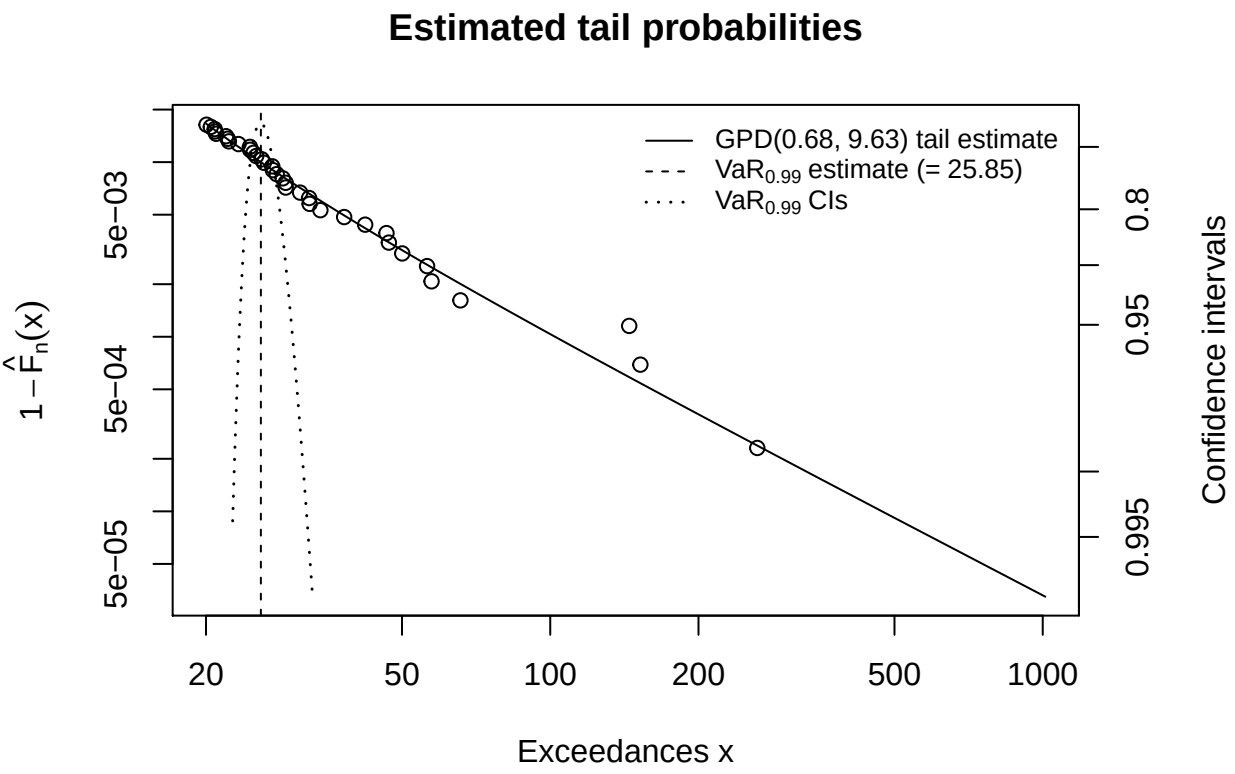
```
##          p quantile     sfall
## [1,] 0.990 25.84720  69.05935
## [2,] 0.995 37.94207 107.38721
```

```
plotTail(mod2)
```

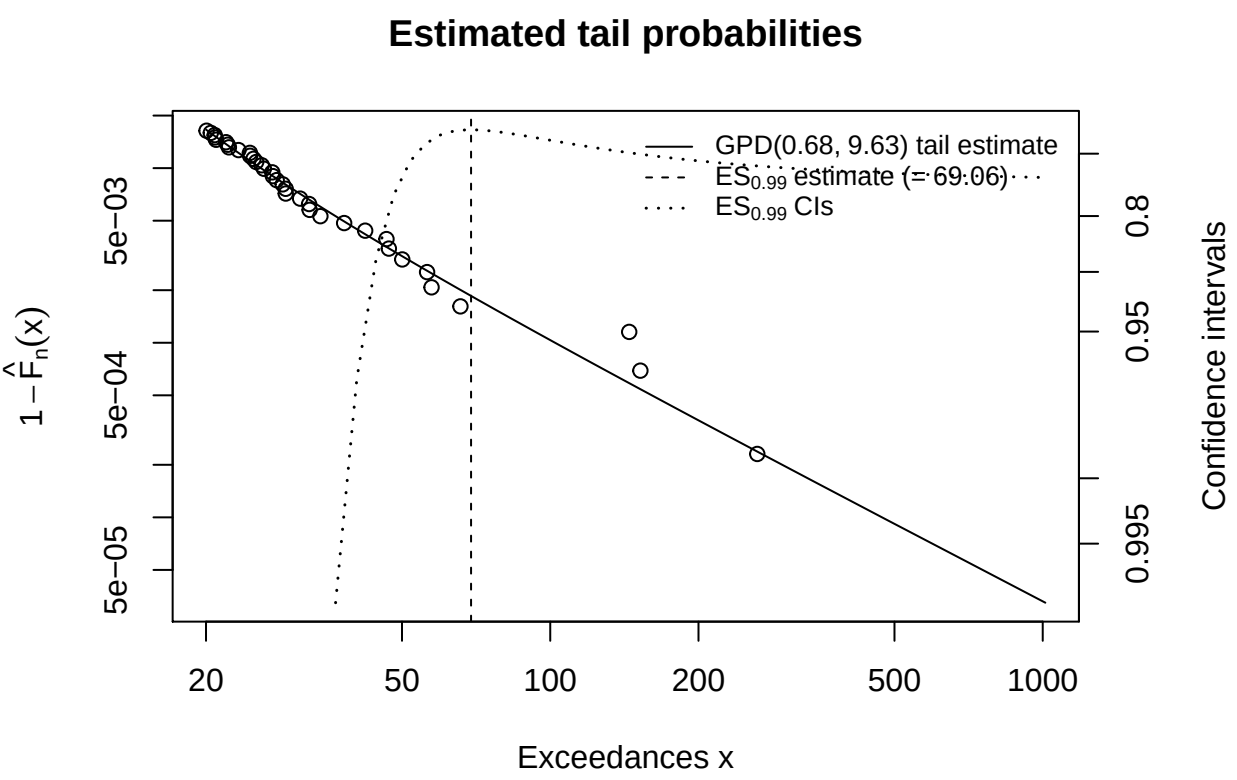


A comparison for the `mod2` run is shown here.

```
showRM(mod2, alpha = 0.99, RM = "VaR",
    method = "BFGS")
```




```
showRM(mod2, alpha = 0.99, RM = "ES",
    method = "BFGS")
```



8.9 Summary

Operational risk is "extreme" finance with (of course) extreme value distributions, methods from reliability and vulnerability analysis thrown in for good measure, and numerous regaultory capital regulations. We just built both simulation and estimation models that produced data driven risk thresholds of an operational nature. That nature means:

- Heavy tail loss severity distributions,
- Often with frequencies that may vary considerably over time, and
- Random losses over time

Application of robuswt risk measures is an ongoing exercise, especially as we learn more about loss and its impact on operations and strategy.

8.10 Further Reading

Much of the operational risk methodologies have been developed around insurance, bank supervision, and reliability engineering. Useful references are the textbooks of Cruz (2004) and Panjer (2006). The material in this chapter derives from these and prominently from

McNeil et al. (2015, chapters 5 and 13) with `R` code from their `QRM`, `qrmtools`, and `evir` packages on `CRAN`.

8.11 Practice Laboratory

8.11.1 Practice laboratory #1

8.11.1.1 Problem

8.11.1.2 Questions

8.11.2 Practice laboratory #2

8.11.2.1 Problem

8.11.2.2 Questions

8.12 Project

8.12.1 Background

8.12.2 Data

8.12.3 Workflow

8.12.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.
- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.

- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.
- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

8.13 References

Cruz, M.G. 2004. Operational Risk Modelling and Analysis: Theory and Practice. London: Risk Books.

McNeill, Alexander J., Rudiger Frey, and Paul Embrechts. 2015. Quantitative Risk Management: Concepts, Techniques and Tools. Revised Edition. Princeton: Princeton University Press.

Panjer, H.H. 2006. Operational Risk: Modeling Analytics. New York: Wiley.

Chapter 9

Meauring Volatility

9.1 Imagine this

- Your company owns several facilities for the manufacture and distribution of polysyllabic acid, a compound used in several industries (and of course quite fictional!).
- Inputs to the manufacturing and distribution processes include various petroleum products and natural gas. Price swings can be quite volatile, and you know that Brent crude exhibits volatility clustering.
- Working capital management is always a challenge, and recent movements in the dollar against the euro and pound sterling have impacted cash conversion severely.
- Your CFO, with a memory of having worked at Metallgesellschaft in the 1990's, is being pushed by the board to hedge the input price risk using futures and over-the-counter (OTC) instruments.

The board is concerned because the company has missed its earnings estimate for the fifth straight quarter in a row. Shareholders are not pleased. The culprits seem to be a volatile cost of goods sold coupled with large swings in some revenue segments in the United Kingdom (UK) and the European Union (EU).Your CFO has handed you the job of organizing the company's efforts to understand the limits your exposure can extend. The analysis will be used to develop policy guidelines for managing customer and vendor relationships.

9.1.1 Example

1. What are the key business questions you should ask about energy pricing patterns?
2. What systematic approach might you use to manage input volatility?

Here are some ideas.

1. Key business questions might be

- Which input prices and exchange rates are more volatile than others and when?
- Are price movements correlated?

- In times of market stress how volatile can they be?
- Are there hedging instruments we can deploy or third parties we can use to mitigate pricing risk?

2. Managing volatility

- Set up an input monitoring system to know what inputs affect what components and costs of running the manufacturing and distribution processes.
- Monitor price movements and characteristics and measure severity of impact on key operating income components by process.
- Build in early warning indicators of intolerable swings in prices.
- Build a playbook to manage the otherwise insurgent and unanticipated retained risk of input price volatility in manufacturing and distribution processes.

Topics we got to in previous chapters:

- Explored stylized fact of financial market data
- Learned just how insidious volatility really is
- Acquired new tools like `acf`, `pacf`, `ccf` to explore time series
- Analyzed market, credit, and operational risk

In this chapter we will

- Remember the stylized facts and use a fix for volatility clustering
- Fit AR-GARCH models
- Simulate volatility from the AR-GARCH model
- Measure the risks of various exposures

9.2 What is All the Fuss About?

We have already looked at volatility clustering. ARCH models are one way to model this phenomenon.

ARCH stands for

- *A*uto*r*egressive (lags in volatility)
- *C*onditional (any new values depend on others)
- *H*eteroscedasticity (Greek for varying volatility, here time-varying)

These models are especially useful for financial time series that exhibit periods of large return movements alongside intermittent periods of relative calm price changes.

An experiment is definitely in order.

The AR+ARCH model can be specified starting with $z(t)$ standard normal variables and initial (we will overwrite this in the simulation) volatility series $\sigma(t)^2 = z(t)^2$. We then condition these variates with the square of their variances $\varepsilon(t) = (sigma^2)^{1/2} z(t)$. Then we first compute for each date $t = 1...n$,

$$\varepsilon(t) = (sigma^2)^{1/2} z(t)$$

Then, using this conditional error term we compute the autoregression (with lag 1 and centered at the mean μ)

$$y(t) = \mu + \phi(y(t-1) - \mu) + \varepsilon(t)$$

Now we are ready to compute the new variance term.

```
n <-  10500 ## lots of trials
z <- rnorm(n) ## sample standard normal distribution variates
e <-  z ## store variates
y <-  z ## store again in a different place
sig2 <-  z^2 ## create volatility series
omega <-  1 ## base variance
alpha <-  0.55 ## Markov dependence on previous variance
phi <-  0.8 ## mMarkov dependence on previous period
mu <-  0.1 ## average return
omega/(1-alpha) ; sqrt(omega/(1-alpha))
```

```
## [1] 2.222222
```

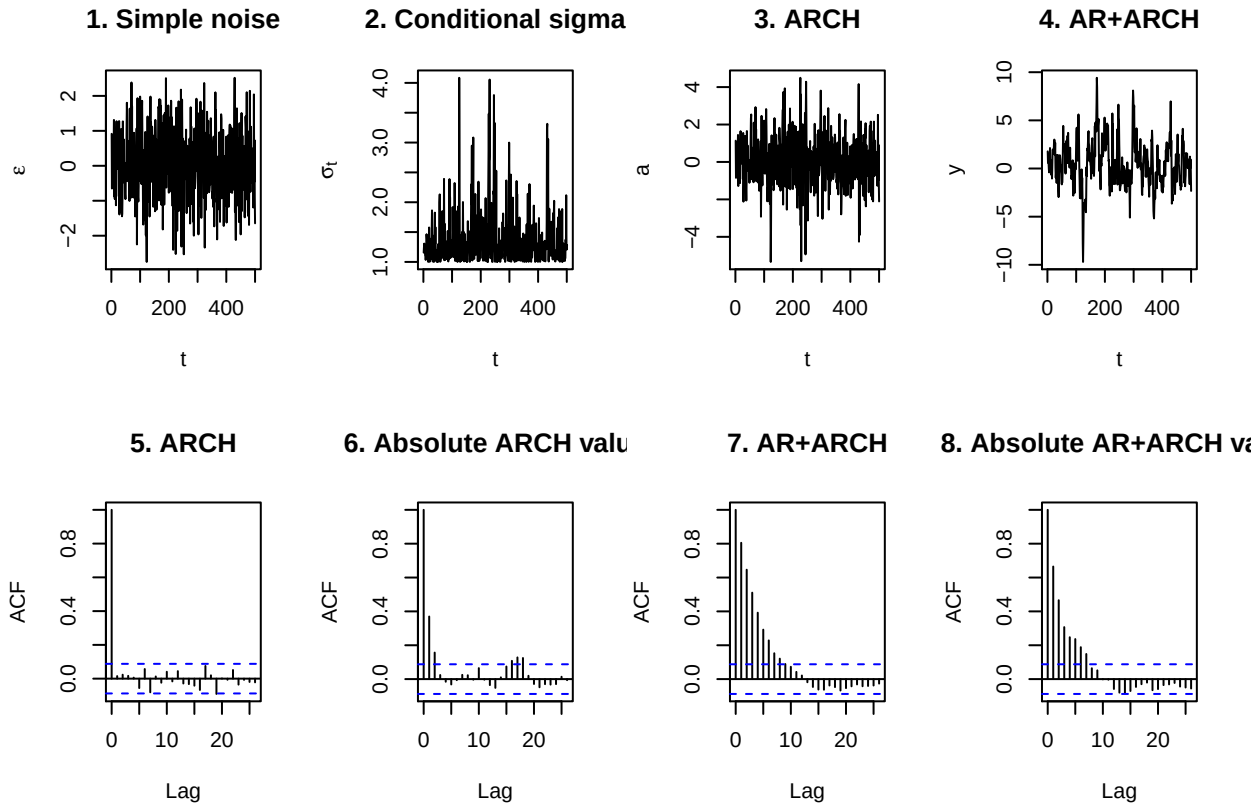
```
## [1] 1.490712
```

```
set.seed("1012")
for (t in 2:n) ## Because of lag start at second date
{
  e[t] <- sqrt(sig2[t])*z[t]          ## 1. e is conditional on sig
  y[t] <-  mu + phi*(y[t-1]-mu) + e[t] ## 2. generate returns
  sig2[t+1] <-  omega + alpha * e[t]^2 ## 3. generate new sigma^2 to feed 1.
}
```

A plot is a little than instructive.

```
par(mfrow = c(2, 4))
plot(z[10001:n], type = "l", xlab = "t",
    ylab = expression(epsilon), main = "1. Simple noise")
plot(sqrt(sig2[10000:n]), type = "l",
    xlab = "t", ylab = expression(sigma[t]),
    main = "2. Conditional sigma")
plot(e[10001:n], type = "l", xlab = "t",
    ylab = "a", main = "3. ARCH")
plot(y[10001:n], type = "l", xlab = "t",
    ylab = "y", main = "4. AR+ARCH")
acf(e[10001:n], main = "5. ARCH")
acf(abs(e[10001:n]), main = "6. Absolute ARCH value")
```

```
acf(y[10001:n], main = "7. AR+ARCH")
acf(abs(y[10001:n]), main = "8. Absolute AR+ARCH value")
```




```
##
```

What do we see?

1. Large outlying peaks in the conditional standard deviation
2. Showing up as well in the ARCH plot
3. AR adds the clustering as returns attempt to revert to the long run mean of $\mu = 10\%$.
4. Patterns reminiscent of clustering occur with thick and numerous lags in the `acf` plots. There is persistence of large movements both up and down.

Why does it matter?

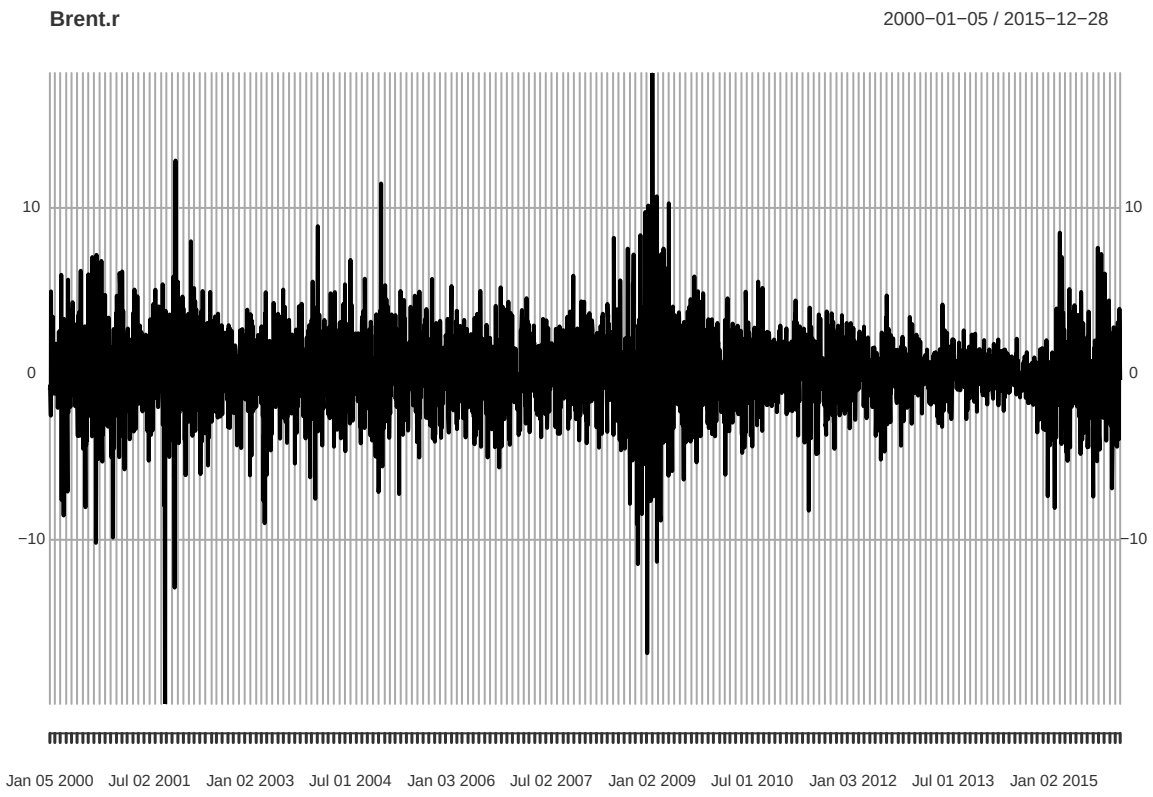
- Revenue received from customer contracts priced with volatility clustering will act like the prices: when in a downward spiral, that spiral will amplify more than when prices try to trend upward.
- The same will happen with the value of inventory and the costs of inputs.
- All of this adds up to volatile EBITDA (Earnings Before Interest and Tax adding in non-cash Depreciation and Amortization), missed earnings targets, shareholders selling, the stock price dropping, and equity-based compensation falling.

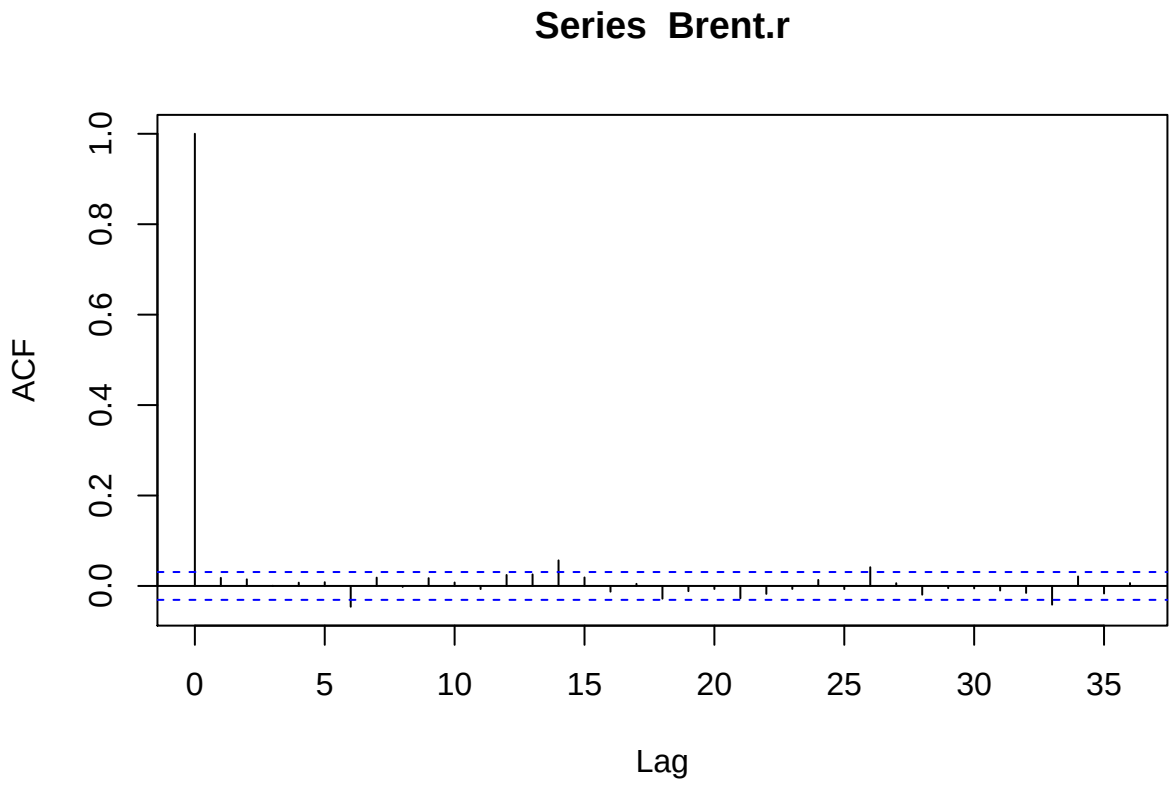
9.3 Lock and Load...

We have more than one way to estimate the parameters of the AR-ARCH process. Essentially we are running yet another "regression." Let's first load some data to tackle the CFO's questions around exposures in the UK, EU, and in the oil market.

```
require(rugarch)
require(qrmdata)
require(xts)
## The exchange rate data was obtained
## from OANDA (http://www.oanda.com/)
## on 2016-01-03
data("EUR_USD")
data("GBP_USD")
## The Brent data was obtained from
## Federal Reserve Economic Data
## (FRED) via Quandl on 2016-01-03
data("OIL_Brent")
data.1 <- na.omit(merge(EUR_USD, GBP_USD,
    OIL_Brent))
P <- data.1
R <- na.omit(diff(log(P)) * 100)
names.R <- c("EUR.USD", "GBP.USD", "OIL.Brent")
colnames(R) <- names.R
Brent.p <- data.1[, 3]
Brent.r <- R[, 3]  ## Pull out just the Brent pieces
```

Then we plot the data, transformations, and autocorrelations.


Brent.r
2000−01−05 / 2015−12−28
10
0
−10
10
0
−10
Jan 05 2000
Jul 02 2001
Jan 02 2003
Jul 01 2004
Jan 03 2006
Jul 02 2007
Jan 02 2009
Jul 01 2010
Jan 03 2012
Jul 01 2013
Jan 02 2015




```
##
##  Box-Ljung test
##
## data:  Brent.r
## X-squared = 32.272, df = 14, p-value = 0.003664
```

The p-value is small enough to more than reject the null hypothesis that the 14-day lag is not significantly different from zero.

9.4 It is Fitting...

Our first mechanical task is to specify the ARMA-GARCH model. First we specify.

1. Use the `ugarchspec` function to specify a plain vanilla `sGarch` model.
2. `garchOrder = c(1,1)` means we are using the first lags of residuals squared and variance or (with ω, "omega," the average variance, σ_t^2), here of Brent returns):

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_{t-1}\sigma_{t-1}^2.$$

3. Use `armaOrder = c(1,0)` to specify the mean Brent return model with long run average μ

$$r_t = \mu + \psi_1 y_{t-1} + \psi_2 \varepsilon_{t-1}.$$

4. Include `means` as in the equations above.
5. Specify the distribution as `norm` for normally distributed innovations ε_t. We will also compare this fit with the `std` Student's t-distribution innovations using the Akaike Information Criterion (AIC).
6. Fit the data to the model using `ugarchfit`.

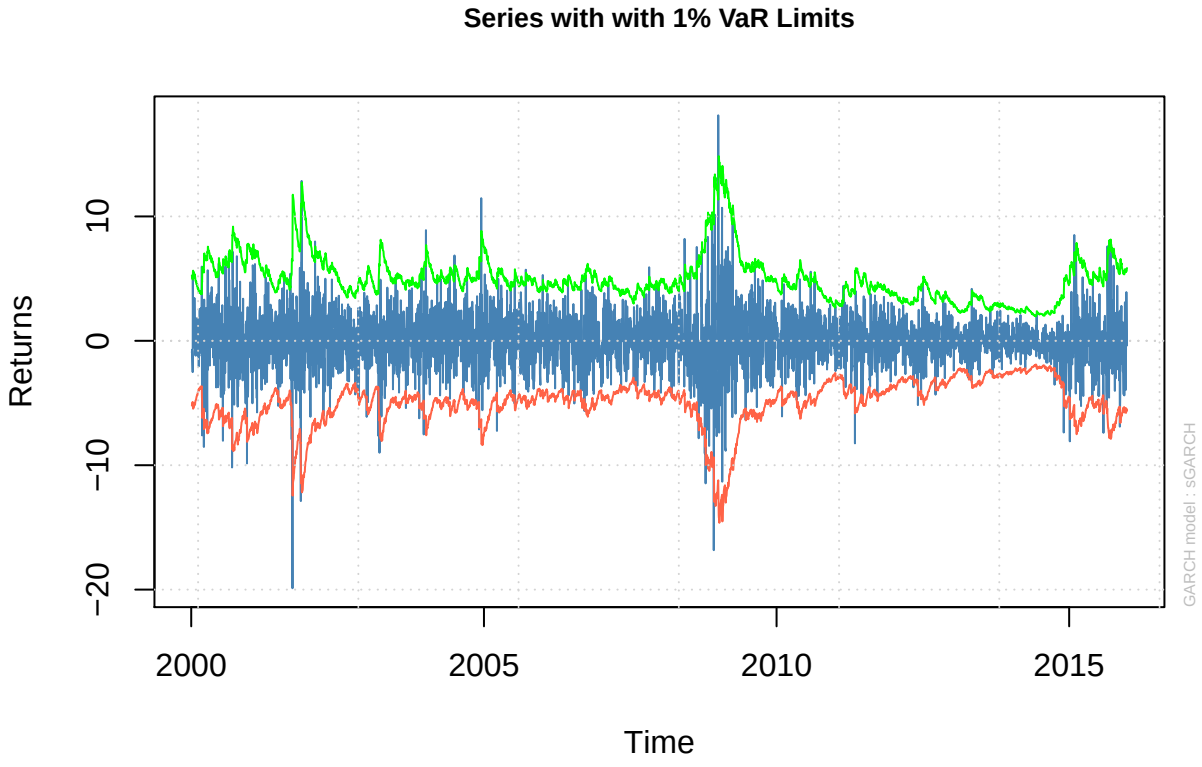
```
AR.GARCH.Brent.Norm.spec <- ugarchspec(variance.model = list(model = "sGARCH",
    garchOrder = c(1, 1)), mean.model = list(armaOrder = c(1,
    0), include.mean = TRUE), distribution.model = "norm")
fit.Brent.norm <- ugarchfit(spec = AR.GARCH.Brent.Norm.spec,
    data = Brent.r)
```

Let's look at the conditional quantiles from this model, otherwise known as the VaR limits, nominally set at 99%.

```
## First the series with conditional
## quantiles
plot(fit.Brent.norm, which = 2)
```

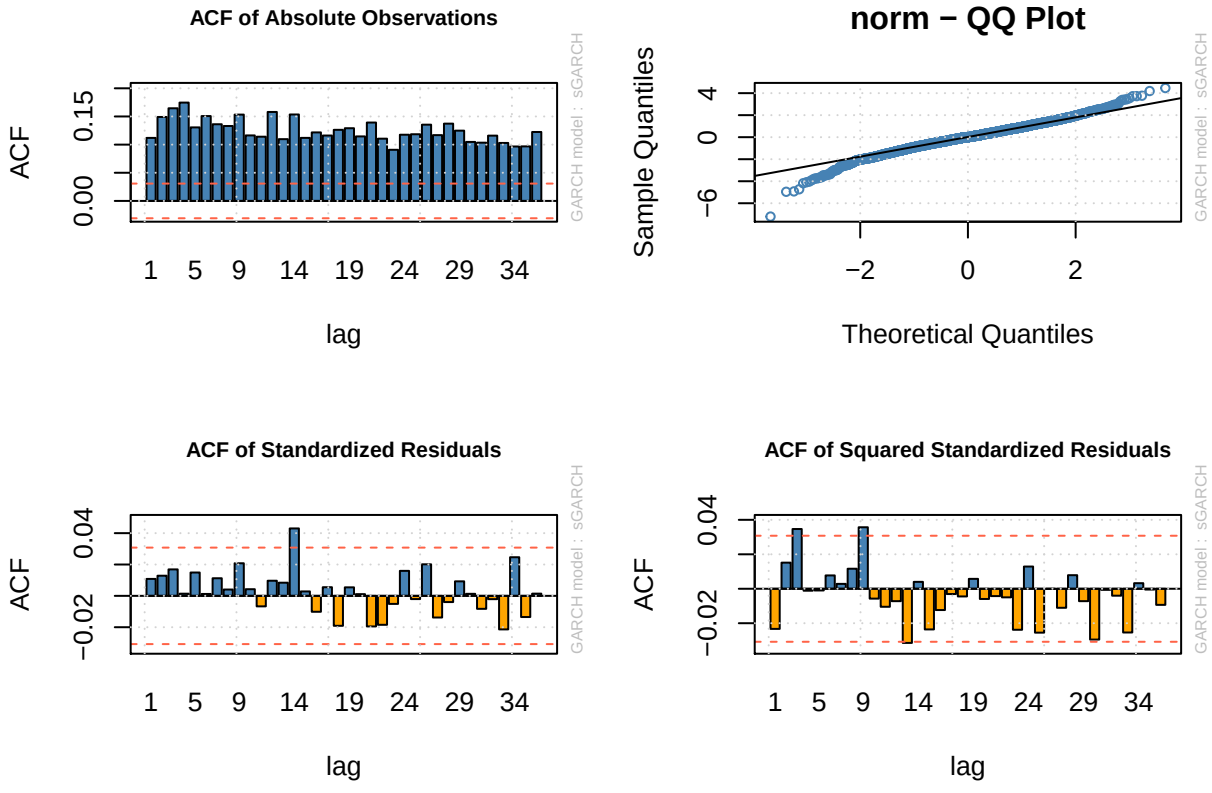
We might think about hedging the very highs and very lows.

```
##
## please wait...calculating quantiles...
```



Now let's generate a panel of plots.

```
par(mfrow = c(2, 2))
## acf of absolute data - shows serial
## correlation
plot(fit.Brent.norm, which = 6)
## QQplot of data - shows
## leptokurtosis of standardized
## rediduals - normal assumption not
## supported
plot(fit.Brent.norm, which = 9)
## acf of standardized residuals -
## shows AR dynamics do a reasonable
## job of explaining conditional mean
plot(fit.Brent.norm, which = 10)
## acf of squared standardized
## residuals - shows GARCH dynamics do
## a reasonable job of explaining
## conditional sd
plot(fit.Brent.norm, which = 11)
```

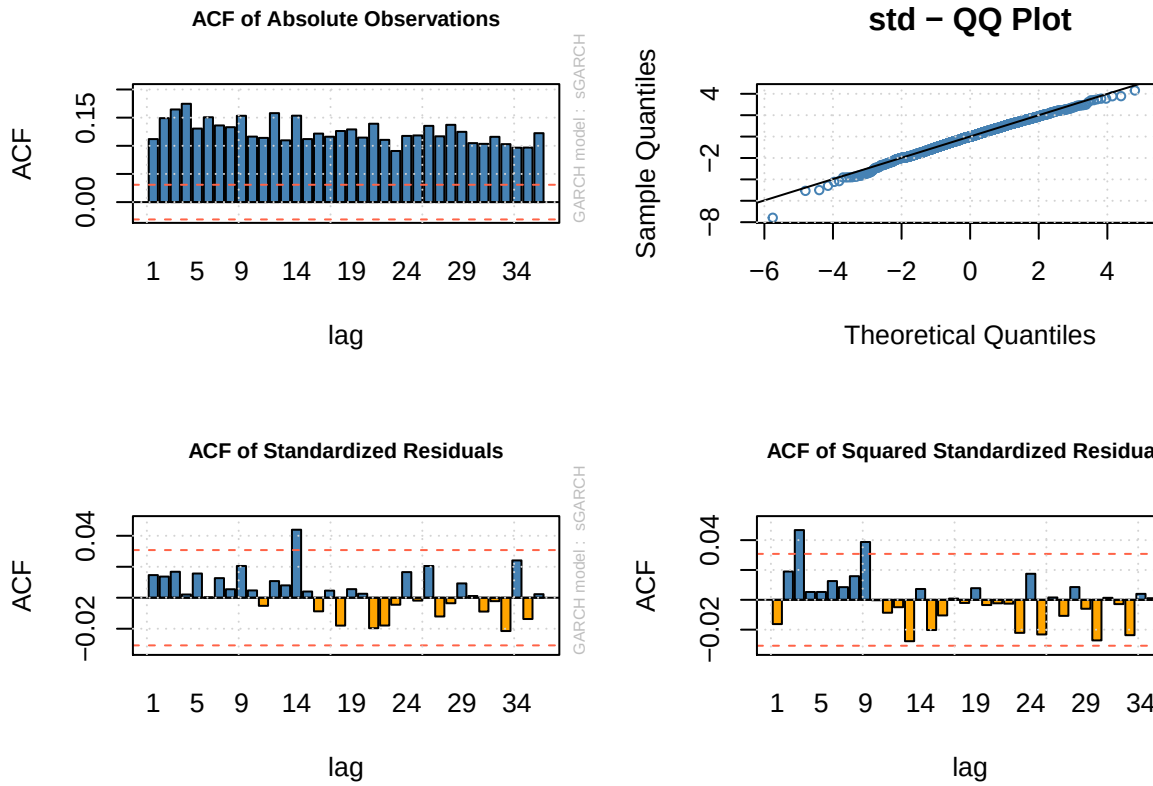


```
par(mfrow = c(1, 1))
```

9.4.1 Example

Let's redo the GARCH estimation, now using the possibly more realistic thicker tails of the Student's t-distribution for the ε innovations. Here we just replace `norm` with `std` in the `distribution.model =` statement in the `ugarchspec` function.

```
## Fit an AR(1)-GARCH(1,1) model with
## student innovations
AR.GARCH.Brent.T.spec <- ugarchspec(variance.model = list(model = "sGARCH",
    garchOrder = c(1, 1)), mean.model = list(armaOrder = c(1,
    0), include.mean = TRUE), distribution.model = "std")
fit.Brent.t <- ugarchfit(spec = AR.GARCH.Brent.T.spec,
    data = Brent.r)
par(mfrow = c(2, 2))
plot(fit.Brent.t, which = 6)
plot(fit.Brent.t, which = 9)
plot(fit.Brent.t, which = 10)
plot(fit.Brent.t, which = 11)
par(mfrow = c(1, 1))
```



Here are some results

1. ACF of absolute observations indicates much volatility clustering.
2. These are significantly dampened by the AR-ARCH estimation with almost bounded standardized residuals (residual / standard error).
3. More evidence of this comes from the ACF of the squared standardized residuals.
4. It appears that this AR-GARCH specification and Student's t-distributed innovations captures most of the movement in volatility for Brent.

Which model? - Use the Akaike Information Criterion (AIC) to measure information leakage from a model. - AIC measures the mount of information used in a model specified by a log likelihood function. - Likelihood: probability that you will observe the Brent returns given the parameters estimated by (in this case) the GARCH(1,1) model with normal or t-distributed innovations. - Smallest information leakage (smallest AIC) is the model for us.

Compute 5. Using normally distributed innovations produces a model with AIC = 4.2471. 6. Using Student's t-distributed innovations produces a model with AIC = 4.2062. 7. GARCH(1,1) with Student's t-distributed innovations is more likely to have less information leakage than the GARCH(1,1) with normally distributed innovations.

Here are some common results we can pull from the fit model:

```
coef(fit.Brent.t)
```

```
##          mu         ar1       omega      alpha1       beta1       shape
## 0.04018002 0.01727725 0.01087721 0.03816097 0.96074399 7.03778415
```

Coefficients include:

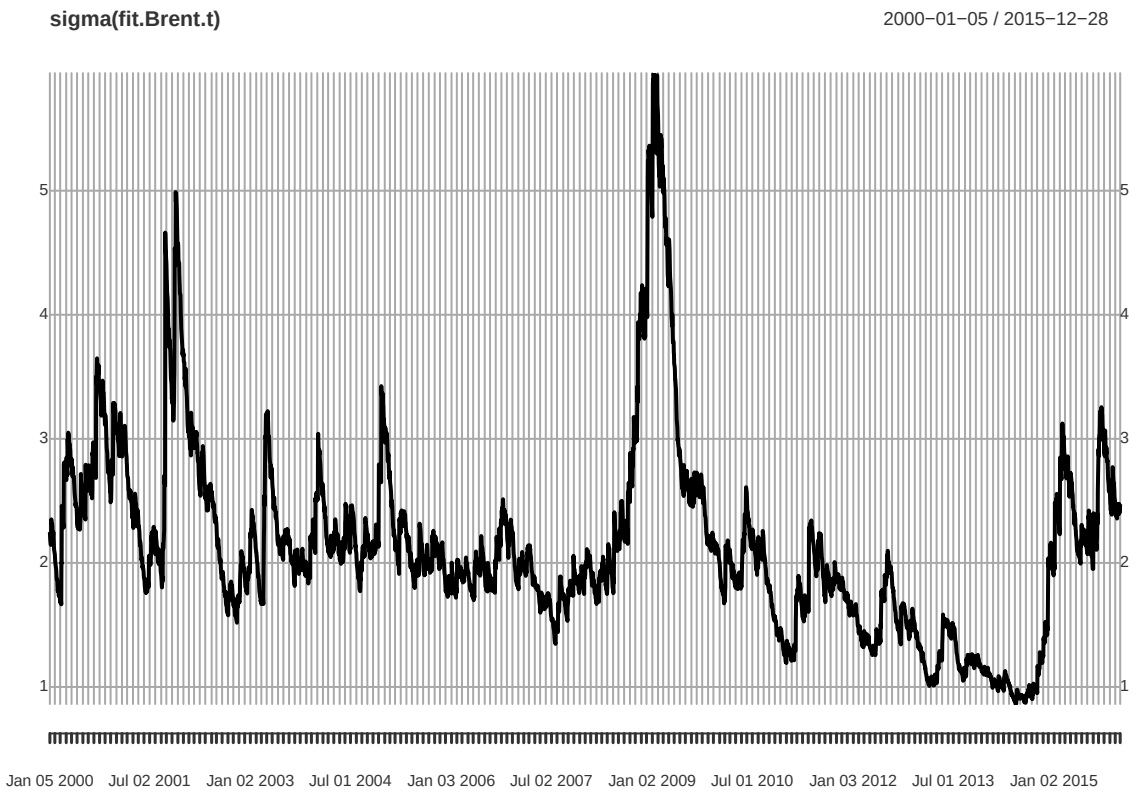
- `mu` is the long run average Brent return.
- `ar1` is the impact of one day lagged return on today's return.
- `omega` is the long run variance of Brent return.
- `alpha1` is the impact of lagged squared variance on today's return.
- `beta1` is the impact of lagged squared residuals on today's Brent return.
- `shape` is the degrees of freedom of the Student's t-distribution and the bigger this is, the thicker the tail.

Let's plot the star of this show: time-varying volatility.

```
coef(fit.Brent.t)
```

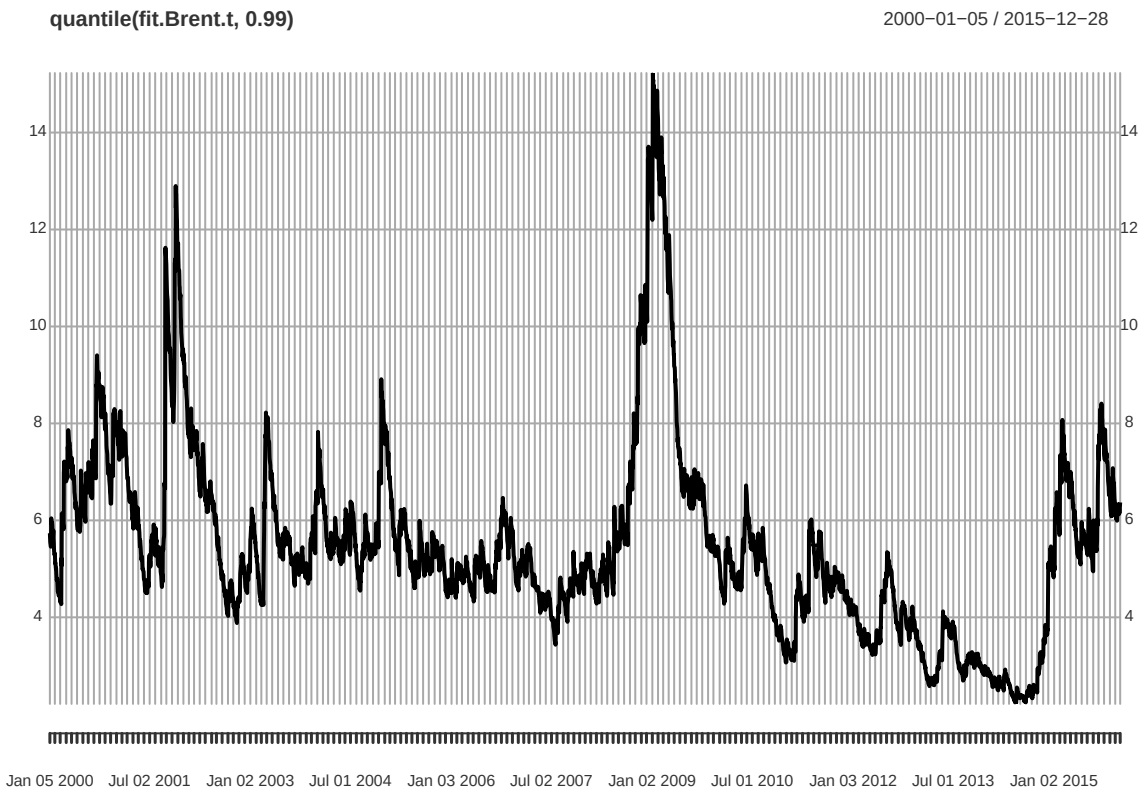
```
##          mu         ar1       omega      alpha1       beta1       shape
## 0.04018002 0.01727725 0.01087721 0.03816097 0.96074399 7.03778415
```

```
plot(sigma(fit.Brent.t))
```



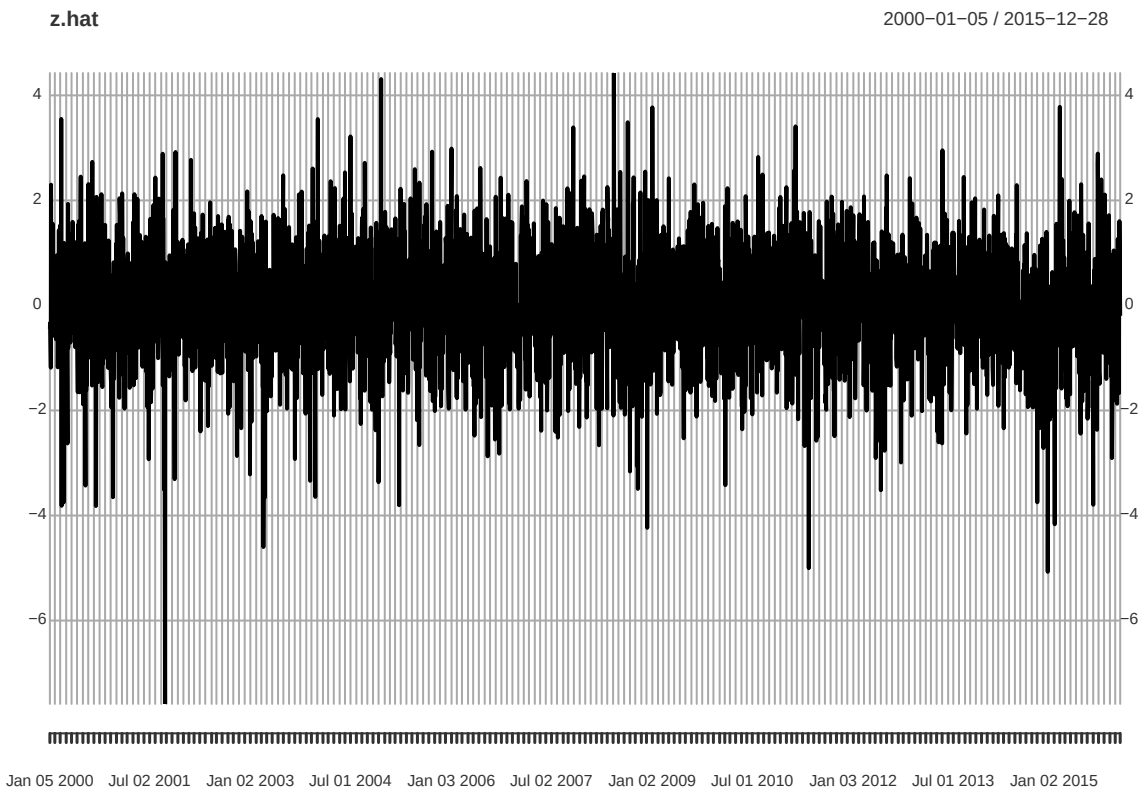
Here's the other reason for going through this exercise: we can look at any Brent volatility range we like.

```
plot(quantile(fit.Brent.t, 0.99))
```

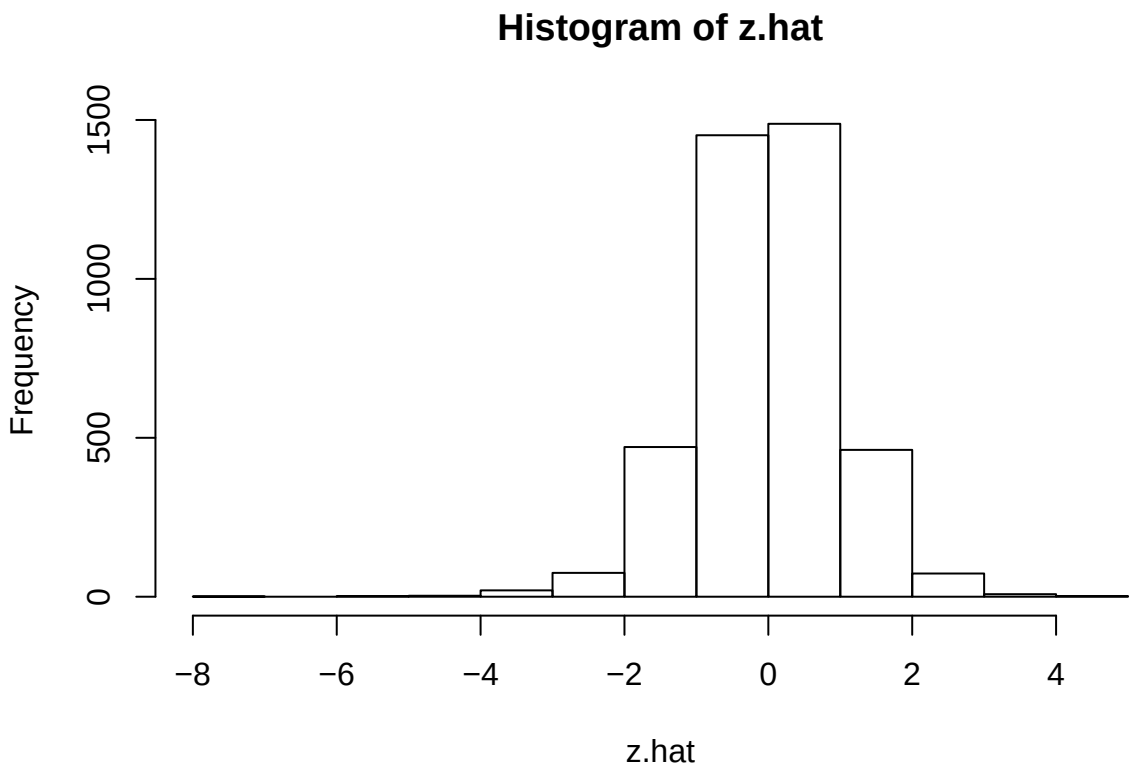


Next we plot and test the residuals:

```
z.hat <- residuals(fit.Brent.t, standardize = TRUE)
plot(z.hat)
```


z.hat
2000−01−05 / 2015−12−28
4
2
0
−2
−4
−6
Jan 05 2000
Jul 02 2001
Jan 02 2003
Jul 01 2004
Jan 03 2006
Jul 02 2007
Jan 02 2009
Jul 01 2010
Jan 03 2012
Jul 01 2013
Jan 02 2015

```
hist(z.hat)
```




```
mean(z.hat)
```

```
## [1] -0.0181139
```

```
var(z.hat)
```

```
##           [,1]
## [1,] 1.000682
```

```
require(moments)
skewness(z.hat)
```

```
## [1] -0.3207327
## attr(,"method")
## [1] "moment"
```

```
kurtosis(z.hat)
```

```
## [1] 2.048561
## attr(,"method")
## [1] "excess"
```

```
shapiro.test(as.numeric(z.hat))
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  as.numeric(z.hat)
## W = 0.98439, p-value < 2.2e-16
```

```
jarque.test(as.numeric(z.hat))
```

```
##
##  Jarque-Bera Normality Test
##
## data:  as.numeric(z.hat)
## JB = 780.73, p-value < 2.2e-16
## alternative hypothesis: greater
```

What do we see?

- Left skewed.
- Thick tailed.
- Potentially large losses can occur with ever larger losses in the offing.
- More negative than positive.
- Both standard tests indicate rejection of the null hypothesis that the series is normally distributed.

9.5 Simulate... again until Morale Improves...

1. Specify the AR-GARCH process using the parameters from the fit.Brent.t results.
2. Generate 2000 paths.

```
require(rugarch)
GARCHspec <- ugarchspec(variance.model = list(model = "sGARCH",
    garchOrder = c(1, 1)), mean.model = list(armaOrder = c(1,
    0), include.mean = TRUE), distribution.model = "std",
    fixed.pars = list(mu = 0.04, ar1 = 0.0173,
        omega = 0.0109, alpha1 = 0.0382,
        beta1 = 0.9607, shape = 7.0377))
GARCHspec
```

```
##
## *---------------------------------*
## *       GARCH Model Spec          *
## *---------------------------------*
##
## Conditional Variance Dynamics
## ----------------------------------
## GARCH Model      : sGARCH(1,1)
```

```
## Variance Targeting   : FALSE
##
## Conditional Mean Dynamics
## ------------------------------------
## Mean Model       : ARFIMA(1,0,0)
## Include Mean     : TRUE
## GARCH-in-Mean        : FALSE
##
## Conditional Distribution
## ------------------------------------
## Distribution :  std
## Includes Skew    :  FALSE
## Includes Shape   :  TRUE
## Includes Lambda  :  FALSE
```

```
## Generate two realizations of length
## 2000
path <- ugarchpath(GARCHspec, n.sim = 2000,
    n.start = 50, m.sim = 2)
```

There is a special plotting function for "uGARCHpath" objects.

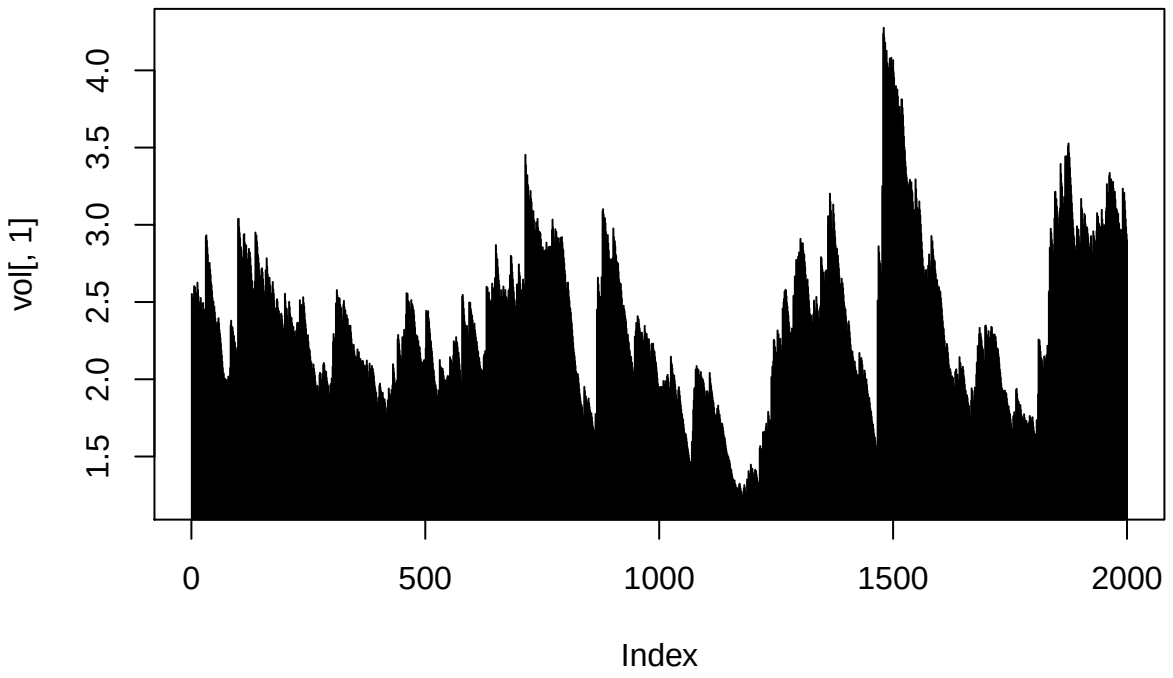
```
plot(path, which = 1)
plot(path, which = 2)
plot(path, which = 3)
plot(path, which = 4)
## How to see the documentation of the
## plot function showMethods('plot')
## getMethod('plot',c(x='GPDTAILS',
## y='missing'))
```

There is also an extraction function for the volatility.

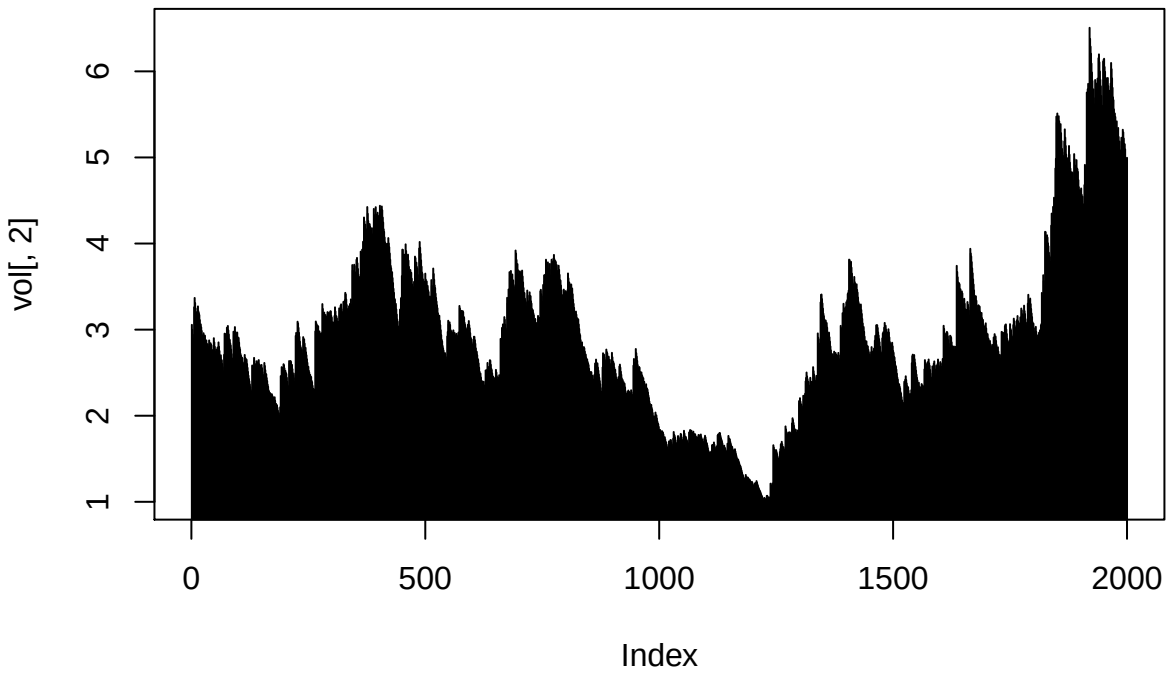
```
vol <- sigma(path)
head(vol)
```

```
##         [,1]     [,2]
## T+1 2.552394 3.055851
## T+2 2.522779 2.997590
## T+3 2.476002 2.941198
## T+4 2.435407 2.895049
## T+5 2.538984 2.839528
## T+6 2.602648 3.259610
```

```
plot(vol[, 1], type = "h")
```




```
plot(vol[, 2], type = "h")
```



Here is a little more background on the classes used.

```
series <- path@path
## series is a simple list
class(series)
```

```
## [1] "list"
```

```
names(series)
```

```
## [1] "sigmaSim"  "seriesSim" "residSim"
```

```
## the actual simulated data are in
## the matrix/vector called
## 'seriesSim'
X <- series$seriesSim
head(X)
```

```
##             [,1]        [,2]
## [1,] -1.5147777  0.32902835
## [2,] -0.3625537 -0.39313263
## [3,]  0.9288261 -1.21795218
## [4,]  4.4490588 -0.03232866
## [5,] -3.7458270 -8.62620734
```

```
## [6,]  1.8274273 -5.53845040
```

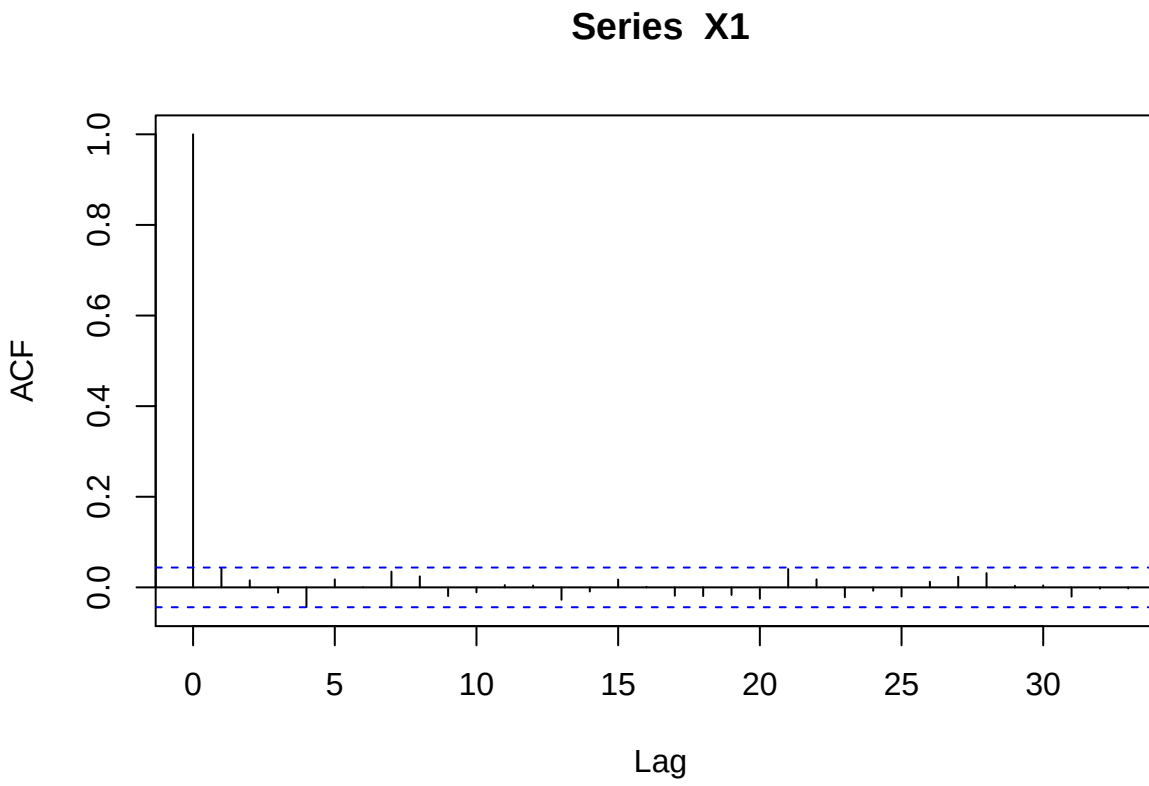
9.5.1 Example

Does the simulated series conform to stylized facts?

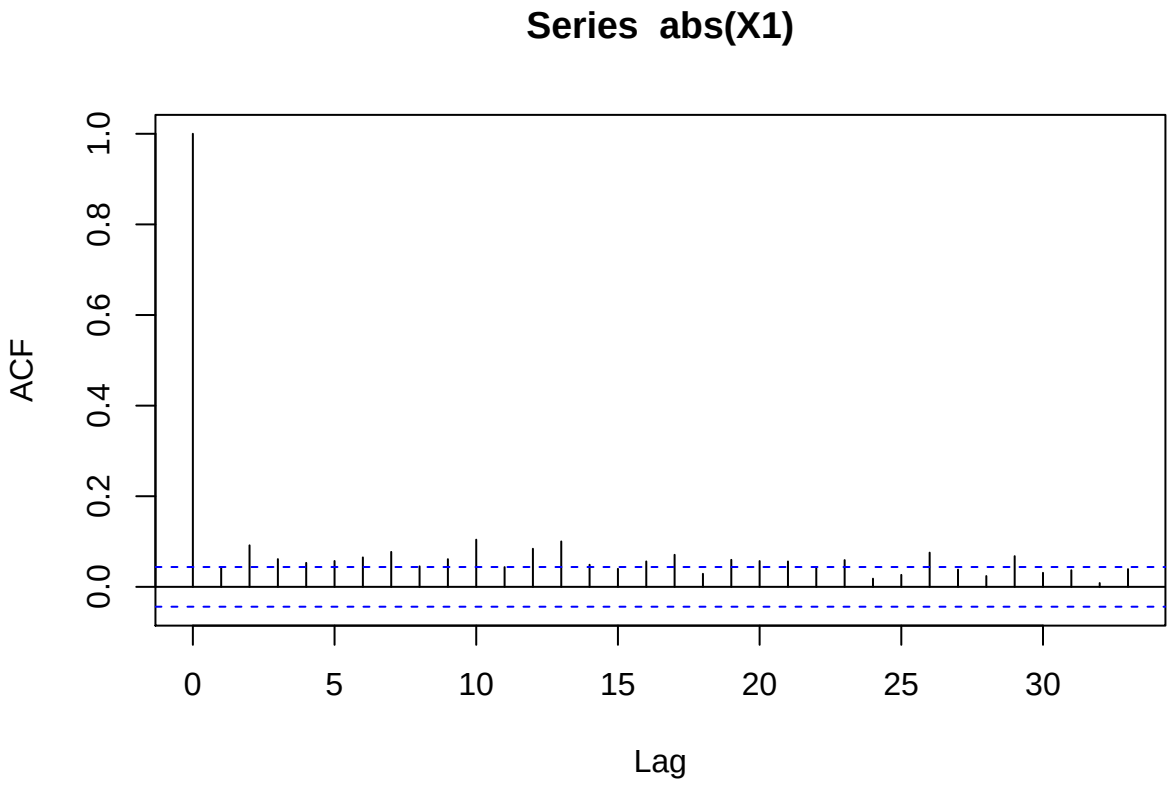
```
X1 <- X[, 1]
acf(X1)
acf(abs(X1))
qqnorm(X1)
qqline(X1, col = 2)
shapiro.test(X1)
```

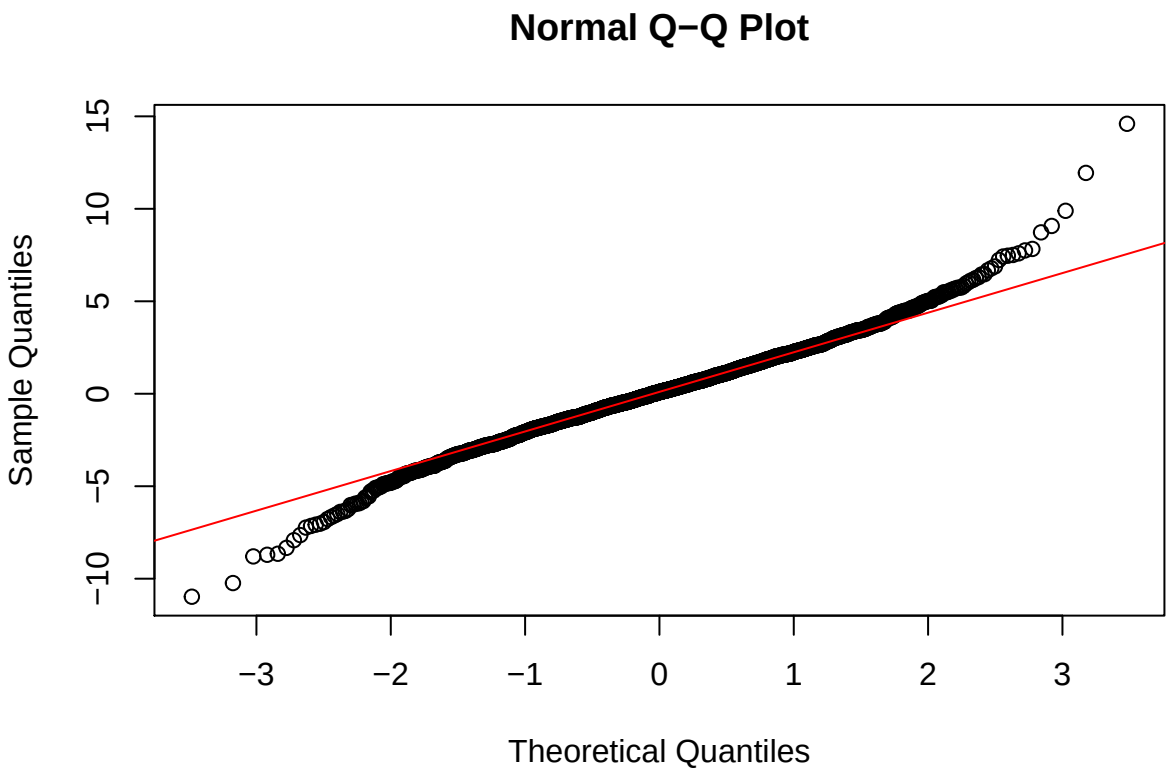
Remember the stylized facts?

1. Volatility clustering.
2. If it's bad, it gets worse more often.
3. If it's good, it get better less often.
4. High stress means high volatility.



Here are some results


Series abs(X1)
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
25
30
Lag




```
##
##  Shapiro-Wilk normality test
##
## data:  X1
## W = 0.98524, p-value = 1.66e-13
```

Shapiro-Wilk test - Null hypothesis: normally distributed. - Reject null if p-value is small enough. - Must verify with a QQ plot of empirical versus theoretical quantiles.

The stylized facts perdure.

9.6 Now for Something Really Interesting

We go from univariate GARCH to multivariate GARCH…and use the most recent technique to make it into the fray:

- The Dynamic Conditional Correlation of Nobel Laureate Robert Engle.
- In the GARCH model we just did, individual assets follow their own univariate GARCH process: they now have time-varying volatilities.
- Engle figured out how to make the correlations among asset return also time-varying.

Why? - What if we have a portfolio, like the accounts receivable that might face variations

in exchange rates and in Brent oil. - We would need to know the joint volatilities and dependences of these three factors as they contribute to overall accounts receivable volatility. - We would use these conditional variances at least to model option prices on instruments to manage currency and commodity risk.

```
require(rmgarch)
garch11.spec <- ugarchspec(mean.model = list(armaOrder = c(0,
    0)), variance.model = list(garchOrder = c(1,
    1), model = "sGARCH"), distribution.model = "std")
dcc.garch11.spec = dccspec(uspec = multispec(replicate(3,
    garch11.spec)), dccOrder = c(1, 1),
    distribution = "mvt")
```

Look at `dcc.garch11.spec`

```
dcc.garch11.spec
```

```
##
## *--------------------------------*
## *       DCC GARCH Spec           *
## *--------------------------------*
## Model           :  DCC(1,1)
## Estimation      :  2-step
## Distribution    :  mvt
## No. Parameters  :  21
## No. Series      :  3
```

Now for the fit (takes more than 27.39 seconds on my laptop…)

```
dcc.fit <- dccfit(dcc.garch11.spec, data = R)
```

Now let's get some results:

```
##
## *---------------------------------*
## *          DCC GARCH Fit          *
## *---------------------------------*
##
## Distribution         :  mvt
## Model                :  DCC(1,1)
## No. Parameters       :  21
## [VAR GARCH DCC UncQ] : [0+15+3+3]
## No. Series           :  3
## No. Obs.             :  4057
## Log-Likelihood       :  -12820.82
## Av.Log-Likelihood    :  -3.16
##
## Optimal Parameters
```

```
## ------------------------------------
##                     Estimate  Std. Error    t value Pr(>|t|)
## [EUR.USD].mu        0.006996    0.007195    0.97238 0.330861
## [EUR.USD].omega     0.000540    0.000288    1.87540 0.060738
## [EUR.USD].alpha1    0.036643    0.001590   23.04978 0.000000
## [EUR.USD].beta1     0.962357    0.000397 2426.49736 0.000000
## [EUR.USD].shape     9.344066    1.192132    7.83811 0.000000
## [GBP.USD].mu        0.006424    0.006386    1.00594 0.314447
## [GBP.USD].omega     0.000873    0.000327    2.67334 0.007510
## [GBP.USD].alpha1    0.038292    0.002217   17.27004 0.000000
## [GBP.USD].beta1     0.958481    0.000555 1727.86868 0.000000
## [GBP.USD].shape    10.481272    1.534457    6.83061 0.000000
## [OIL.Brent].mu      0.040479    0.026696    1.51627 0.129450
## [OIL.Brent].omega   0.010779    0.004342    2.48228 0.013055
## [OIL.Brent].alpha1  0.037986    0.001941   19.57467 0.000000
## [OIL.Brent].beta1   0.960927    0.000454 2118.80489 0.000000
## [OIL.Brent].shape   7.040287    0.729837    9.64639 0.000000
## [Joint]dcca1        0.009915    0.002821    3.51469 0.000440
## [Joint]dccb1        0.987616    0.004386  225.15202 0.000000
## [Joint]mshape       9.732509    0.652707   14.91100 0.000000
##
## Information Criteria
## --------------------
##
## Akaike       6.3307
## Bayes        6.3633
## Shibata      6.3306
## Hannan-Quinn 6.3423
##
##
## Elapsed time : 11.9331
```

- The mean models of each series (EUR.USD, GPB.USD, OIL.Brent) are overwhelmed by the preponderance of time-varying volatility, correlation, and shape (degrees of freedom since we used the Student's t-distribution).
- The joint conditional covariance (relative of correlation) parameters are also significantly different from zero.

Using all of the information from the fit, we now forecast. These are the numbers we would use to simulate hedging instruments or portfolio VaR or ES.Let's plot the time-varying sigma first.

```
# dcc.fcst <- dccforecast(dcc.fit,
# n.ahead = 100) plot(dcc.fcst, which
# = 2)
```

9.6.1 Example

Look at VaR and ES for the three risk factors given both conditional volatility and correlation.

Here are some results. First, compute, then plot.

```
dcc.residuals <- residuals(dcc.fit)
(Brent.dcc.var <- quantile(dcc.residuals$OIL.Brent,
    c(0.01, 0.05, 0.5, 0.95, 0.99)))
```

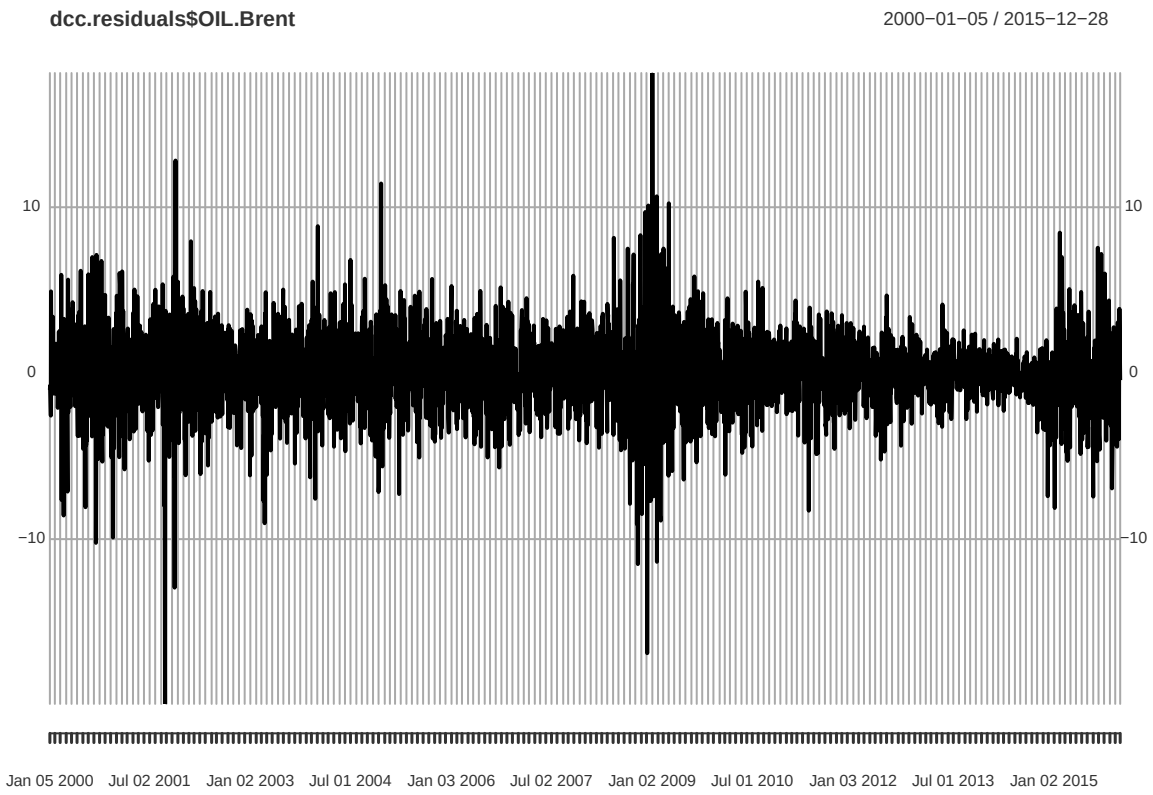
```
##           1%           5%          50%          95%          99%
## -6.137269958 -3.677130793 -0.004439644  3.391312753  5.896992710
```

```
(GBP.dcc.var <- quantile(dcc.residuals$GBP.USD,
    c(0.01, 0.05, 0.5, 0.95, 0.99)))
```

```
##            1%            5%           50%           95%           99%
## -1.3393119939 -0.8235076255 -0.0003271163  0.7659725631  1.2465945013
```

```
(EUR.dcc.var <- quantile(dcc.residuals$EUR.USD,
    c(0.01, 0.05, 0.5, 0.95, 0.99)))
```

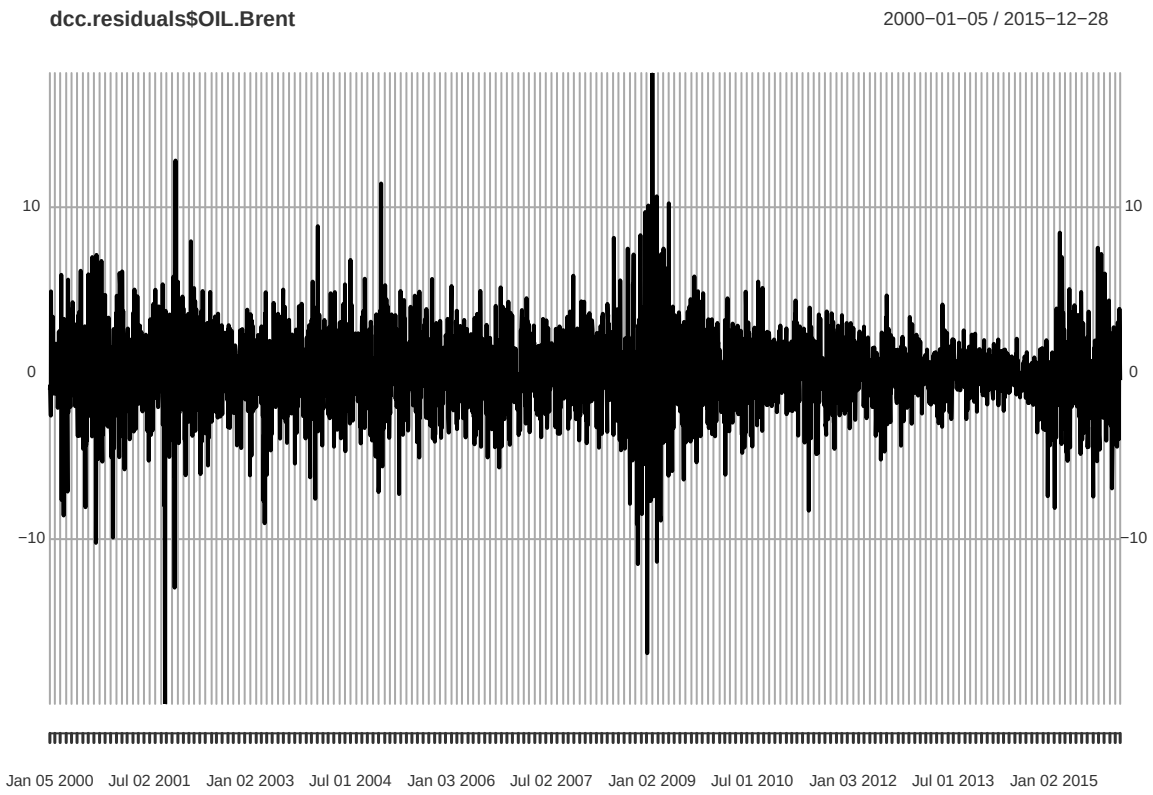
```
##           1%           5%          50%          95%          99%
## -1.520666396 -0.980794376  0.006889539  0.904772045  1.493169076
```



What do we see?

1. A bit more heavily weighted in the negative part of the distributions.
2. Exchange rates are about the same as one another in this profile.
3. Brent is shocky at best: large moves either way.
4. If you use Brent contingent inputs (costs) in your production process you are naturally short Brent and would experience losses at the rate of 500% about 1% of the time.
5. If you use Brent contingent outputs (revenue) in your customer and distribution processes you are naturally long Brent and could experience over 600% losses about 1% of the time.

```
plot(dcc.residuals$OIL.Brent)
```



9.7 Just One More Thing

Back to Brent. Let's refit using the new volatility models and innovation distributions to capture asymmetry and thick tails.

```
Brent.spec <- ugarchspec(variance.model = list(model = "gjrGARCH",
    garchOrder = c(1, 1)), mean.model = list(armaOrder = c(1,
    1), include.mean = TRUE), distribution.model = "nig")
```

Here we experiment with a new GARCH model: the gjr stands for Glosten, Jagannathan, and Runkle (1993), who proposed a volatility model that puts a knot into the works:

$$\sigma_t^2 = \omega + \alpha\sigma_{t-1}^2 + \beta_1\varepsilon_{t-1}^2 + \beta_2\varepsilon_{t-1}^2 I_{t-1}$$

where $I_{t-1} = 1$ when $\varepsilon_{t-1} > 0$ and 0 otherwise, the "knot."

We also experiment with a new distribution: the negative inverse gamma. Thick tails abound…

```
fit.Brent <- ugarchfit(spec = Brent.spec,
    data = R$OIL.Brent, solver.control = list(trace = 0))
```

Another 10 seconds (or so) of our lives will fit this model.

```
fit.Brent
```

```
##
## *---------------------------------*
## *          GARCH Model Fit        *
## *---------------------------------*
##
## Conditional Variance Dynamics
## -----------------------------------
## GARCH Model  : gjrGARCH(1,1)
## Mean Model   : ARFIMA(1,0,1)
## Distribution : nig
##
## Optimal Parameters
## ------------------------------------
##         Estimate  Std. Error     t value Pr(>|t|)
## mu     -0.040275    0.027883 -1.4445e+00 0.148608
## ar1     0.996072    0.001900  5.2430e+02 0.000000
## ma1    -0.989719    0.000005 -1.8786e+05 0.000000
## omega   0.006346    0.003427  1.8517e+00 0.064071
## alpha1  0.009670    0.003841  2.5178e+00 0.011808
## beta1   0.968206    0.001237  7.8286e+02 0.000000
## gamma1  0.042773    0.007183  5.9547e+00 0.000000
## skew   -0.120184    0.032059 -3.7488e+00 0.000178
## shape   2.362890    0.351494  6.7224e+00 0.000000
##
## Robust Standard Errors:
##         Estimate  Std. Error     t value Pr(>|t|)
## mu     -0.040275    0.030871 -1.3046e+00 0.192023
## ar1     0.996072    0.002107  4.7283e+02 0.000000
## ma1    -0.989719    0.000005 -1.8363e+05 0.000000
## omega   0.006346    0.003388  1.8729e+00 0.061086
## alpha1  0.009670    0.004565  2.1184e+00 0.034143
## beta1   0.968206    0.000352  2.7485e+03 0.000000
## gamma1  0.042773    0.008503  5.0300e+00 0.000000
## skew   -0.120184    0.033155 -3.6249e+00 0.000289
## shape   2.362890    0.405910  5.8212e+00 0.000000
##
## LogLikelihood : -8508.439
##
## Information Criteria
## ------------------------------------
##
## Akaike       4.1989
```

```
## Bayes        4.2129
## Shibata      4.1989
## Hannan-Quinn 4.2038
##
## Weighted Ljung-Box Test on Standardized Residuals
## ------------------------------------
##                         statistic p-value
## Lag[1]                      1.856  0.1730
## Lag[2*(p+q)+(p+q)-1][5]     2.196  0.9090
## Lag[4*(p+q)+(p+q)-1][9]     2.659  0.9354
## d.o.f=2
## H0 : No serial correlation
##
## Weighted Ljung-Box Test on Standardized Squared Residuals
## ------------------------------------
##                         statistic  p-value
## Lag[1]                     0.5109 0.474739
## Lag[2*(p+q)+(p+q)-1][5]    9.3918 0.013167
## Lag[4*(p+q)+(p+q)-1][9]   13.2753 0.009209
## d.o.f=2
##
## Weighted ARCH LM Tests
## ------------------------------------
##             Statistic Shape Scale  P-Value
## ARCH Lag[3]     10.26 0.500 2.000 0.001360
## ARCH Lag[5]     10.41 1.440 1.667 0.005216
## ARCH Lag[7]     11.06 2.315 1.543 0.010371
##
## Nyblom stability test
## ------------------------------------
## Joint Statistic:  2.5309
## Individual Statistics:
## mu     0.91051
## ar1    0.07050
## ma1    0.06321
## omega  0.70755
## alpha1 0.22126
## beta1  0.28137
## gamma1 0.17746
## skew   0.25115
## shape  0.16545
##
## Asymptotic Critical Values (10% 5% 1%)
## Joint Statistic:          2.1 2.32 2.82
## Individual Statistic:     0.35 0.47 0.75
```

```
##
## Sign Bias Test
## ------------------------------------
##                    t-value    prob sig
## Sign Bias           1.1836 0.23663
## Negative Sign Bias  0.7703 0.44119
## Positive Sign Bias  1.8249 0.06809   *
## Joint Effect        9.8802 0.01961  **
##
##
## Adjusted Pearson Goodness-of-Fit Test:
## ------------------------------------
##   group statistic p-value(g-1)
## 1    20     27.42      0.09520
## 2    30     46.32      0.02183
## 3    40     58.50      0.02311
## 4    50     70.37      0.02431
##
##
## Elapsed time : 7.077968
```

9.7.1 Example (last one!)

Let's run this code and recall what the `evir` package does for us.

```
require(evir)
Brent.resid <- abs(residuals(fit.Brent))
gpdfit.Brent <- gpd(Brent.resid, threshold = quantile(Brent.r,
    0.9))
(Brent.risk <- riskmeasures(gpdfit.Brent,
    c(0.9, 0.95, 0.975, 0.99, 0.999)))
```

We can interpret the results...and use the `tailplot()` function as well.

```
require(evir)
Brent.resid <- abs(residuals(fit.Brent))
gpdfit.Brent <- gpd(Brent.resid, threshold = quantile(Brent.r,
    0.9))
(Brent.risk <- riskmeasures(gpdfit.Brent,
    c(0.9, 0.95, 0.975, 0.99, 0.999)))
```

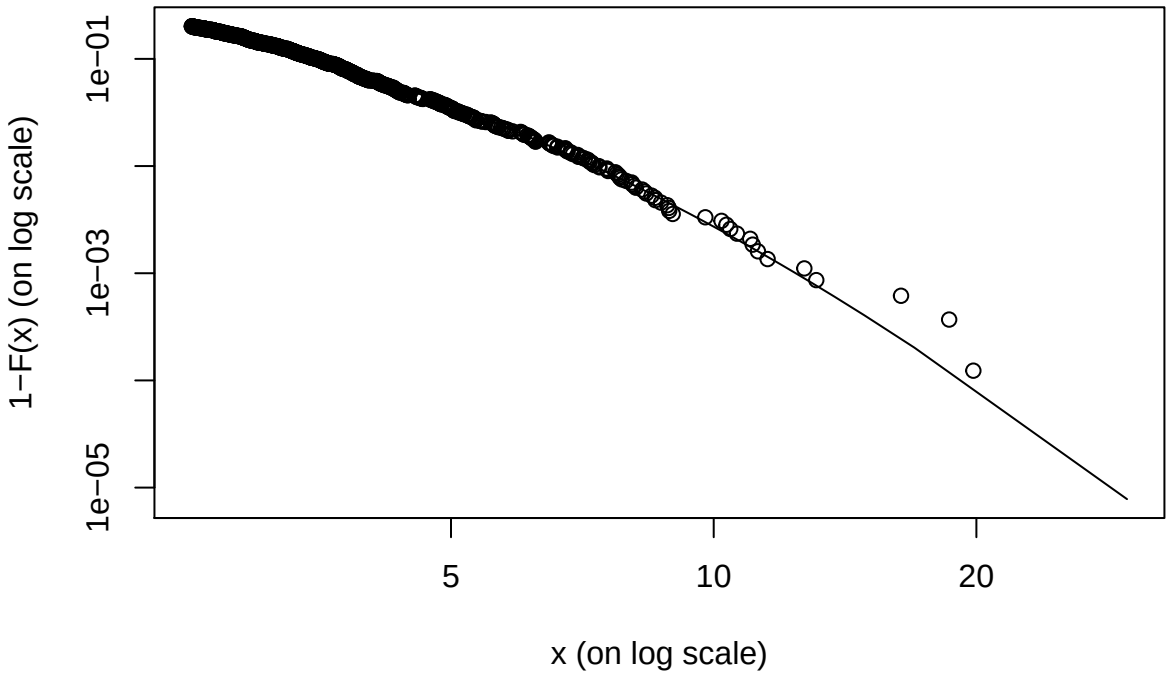
```
##          p  quantile     sfall
## [1,] 0.900  3.478474  5.110320
## [2,] 0.950  4.509217  6.293461
## [3,] 0.975  5.636221  7.587096
```

```
## [4,] 0.990  7.289163  9.484430
## [5,] 0.999 12.415553 15.368772
```

What does this mean? 1. `1 - p` gives us levels of tolerance 2. `quantile` gives us the value at risk (VaR) 3. `sfall` reports the expected short fall (ES)

From the `evir` package here is the tail plot.

```
tailplot(gpdfit.Brent)
```



What does this mean?

- The results show much thick and volatile tail activity event with the AR-GARCH treatment.
- We could well go back to market and operational risk sections to understand mean excess value (beyond thresholds) and the confidence intervals for VaR and ES.
- For accounts receivable mitigation strategies might be to have excess risk hedges provided through reinsurance and total return swaps.
- Credit risk analysis of customers is critical: frequent updates of Brent exposed customers will help to detect early on problems that might admit of some solution.

9.8 Summary

- Lots more `R` practice
- Univariate GARCH
- Multivariate GARCH
- Fitting models
- Simulating volatility and correlation
- …and why it might all matter: answering a critical business question of how much volatility do we have to manage.

9.9 Further Reading

9.10 Practice Laboratory

9.10.1 Practice laboratory #1

9.10.1.1 Problem

9.10.1.2 Questions

9.10.2 Practice laboratory #2

9.10.2.1 Problem

9.10.2.2 Questions

9.11 Project

9.11.1 Background

9.11.2 Data

9.11.3 Workflow

9.11.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.
- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.
- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.
- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

9.12 References

Chapter 10

Portfolio Analytics

> The first stage starts with observation and experience and ends with beliefs about the future performances of available securities. The second stage starts with the relevant beliefs about future performances and ends with the choice of portfolio.
> *Harry Markowitz*

10.1 Imagine This

- You are trying to negotiate a new energy contract across 10 different facilities and 20 gas and oil suppliers.
- Your colleague is managing accounts receivable across 38 different currencies in spot and forward markets.
- Another colleague manages collateral for workers' compensation funding in all 50 states.
- Yet another colleague manages 5 fund managers for a health insurer.

Portfolios are everywhere! We can conceive of every margin as a long position in revenue and a short position in costs. In all case of interest at least some notion of the the mean (central tendency) and standard deviation (scale and diffusion) of a portfolio metric (e.g., return) will be traded off. Operational and financial constraints will narrow the possible choices to achieve performance (mean = "mu" = μ) and risk (standard deviation = "sigma" = σ) goals.

10.1.1 Our "working example"

This will we Working Capital which is Receivables + Inventory - Payables. The CFO needs answers around why it is so big and always seems to bulge when the economic fortunes of our customers are in peril of deteriorating. She knows that there are three culprits: the euro rate, the Sterling rate, and Brent crude. She commissions you and your team to figure out

the ultimate combination of these factors that contributes to a $100 million working capital position with a volatility of over $25 million this past year.

10.1.2 This chapter

The goal of this chapter is to introduce portfolio analytics and incorporate our knowledge of statistics, optimization, and financial objects into the analysis.

10.2 Let's Walk Before We Run

Suppose management wants to achieve a targeted value at risk on new contracts. The value at risk (VaR) is the α quantile of portfolio value where α ("alpha") is the organization's tolerance for risk. While VaR is the maximum amount of tolerable loss more loss is possible.

Now suppose Management is considering a $1 billion contract with two Iberian companies and one United Kingdom-based (UK) company working in Spain. The deal is constrained by having $100 million in reserves are available to cover any losses. The Board wants some comfort that no more than a 10 % age loss (the average "return" μ) would occur. The Board has set the organization's tolerance for risk at 5%. This means in this case a maximum of 5% of the time could losses exceed 10%. To keep things simple at first we assume that losses are normally distributed.

Let's perform a "back of the envelope" analysis. We let R stand for returns, so that $-R$ is a loss. Our management team wants

$$Prob(R < -0.10) = 0.05,$$

that is, the probability of a loss worse than 10 % is no more than 5 %.

Let's now do some algebra and first let w be the "weight" invested in the risky contract. The rest of the proportion of the contract, $1 - w$, is in high quality collateral assets like treasury bonds. The weight w can take on values from 0% (0.0) to 100% (1.0).

- No collateral means $w = 1$
- No contract means $w = 0$.

The average return, μ, on the contract is

$$\mu = w(0.1) + (1 - w)(0.02).$$

This is the weighted *average* return of the contract at 10% and collateral at 2%.

The average level of risk in this model is given by the standard deviation of this combination of risky contract and default-free collateral.

Management currently believes that a 25% standard deviation of return on the contract, "sigma" or σ, is reasonable. Collateral is not "risky" in this scenario and thus its standard deviation is zero (and by definition is not correlated with the contract return).

$$\sigma = w^2(0.25)^2 + (1-w)^2(0.0).$$

We now try to figure out what w is. More precisely, we solve for the percentage of total assets as investment in this contract, that will make losses happen no greater than 5% of the time.

We form a normalizing "z-score" to help us. We shift the largest possible loss to average return to a deviation around the 10% loss, then divide by σ to scale losses to the number of standard deviations around the maximum tolerable loss.

$$z = \frac{-0.1 - \mu}{\sigma}.$$

The z the ratio of potential deviation of loss from the mean maximum loss per unit of risk. It is dimensionless and represents the number of standard deviations of loss around the maximum tolerable loss. Our job is to find w such that the z score under the normal distribution cannot exceed 5%.

$$Prob(R < -0.10) = Normal(z(w)) = 0.05,$$

where $Normal$ is the cumulative normal distribution (you might know this as =Norm.S.Dist() in Excel or `qnorm()` in `R`), with mean of zero and standard deviation of one.

Using our models of μ and σ we get

$$z = \frac{-0.1 - 0.1w - 0.02(1-w)}{0.25w}$$

After combining constant terms and terms in w and putting this into the target probability:

$$z = Normal\left[\frac{-0.12 - 0.12w}{0.25w}\right] = 0.05.$$

Finally, we solve for w in a few more steps.

1. We invert the normal distribution on both sides of the equation. On the left hand side we are left with the z score as a function of w, the percentage of all wealth in the risk contract.

$$Inverse\,Normal\left[Normal\left(\frac{-0.12 - 0.12w}{0.25w}\right)\right] = Inverse\,Normal(0.05)$$

   We can calculate $Inverse\,Normal(0.05)$ using `R`

```
qnorm(0.05)
```

```
## [1] -1.644854
```

or in Excel with `=NORM.S.INV(0.05)`. Each of these takes as input the probability under the normal distribution and calculates the z score associated with the probability.

2. We find that `qnorm(0.05)` is 1.64. Loss cannot exceed 1.64 times the portfolio standard deviation in the direction of loss ("negative" or less than the mean) using a one-tail interval. We justify a one-tail interval since we, and the Board, are only interested in loss. Inserting this value we get

$$\left(\frac{-0.12 - 0.12w}{0.25w}\right) = NormalInverse(0.05) = -1.64$$

Multiplying each side by $0.25w$, combining terms in w and dividing by the coefficient of that last combined w we get

$$w = \frac{-0.12}{0.25(-1.64) + 0.12} = 0.42.$$

In `R`:

```
0.413793103
```

```
## [1] 0.4137931
```

Implications?

- 42% of portfolio value = risky contract value.
- Portfolio value = \$1 billion / 0.42 = \$2.38 billion.
- Collateral value = \$2.38 billion - \$1 billion = \$1.38 billion or 68% of portfolio value.

We just found the notorious "tangency" portfolio. This portfolio, when combined with a risk-free (really "default-free" asset), will yield the best mix of risky and risk-free assets. "Best" here is in the sense of not violating the organization's risk tolerance policy.

A way to find the tangency portfolio in any situation is then to

1. Find the optimal combination of risky assets, the tangency portfolio.
2. Then find the optimal mix of tangency assets and the risk-free asset.

In our example working capital's "risk-free" asset is the cash account and the process of getting there is the cash-conversion cycle.

10.3 All In

Now that we have our basic procedure, let's complicate this problem with many risky assets. The basic solution will be choosing weights to minimize the portfolio risk given risk-adjusted return targets. This is the Markowitz (1952) portfolio solution. For this task we need to define a matrix version of the portfolio allocation problem. Our three risky "assets" will be the euro/USD and GBP/USD exchange rates and Brent crude.

This is all about the normally distributed universe. Let's define more precisely

First, the return matrix R for N assets across T sample periods and the subscript indicates the row (observation) and column (asset):

$$\begin{bmatrix} R_{11} & \dots & R_{1N} \\ \dots & \dots & \dots \\ R_{1T} & \dots & R_{TN} \end{bmatrix}$$

Then, the mean return vector μ is the arithmetic average of each column of R

$$\begin{bmatrix} \mu_1 \\ \dots \\ \mu_N \end{bmatrix},$$

after exchanging rows for columns (transpose).

10.3.1 Try this exercise

First let's be sure the `qrmdata` package is installed. We require this package and the daily data in it. Then we will look up the `apply` function to see how we can compute row averages.

```
require(qrmdata)
require(xts)
## The exchange rate data was obtained
## from OANDA (http://www.oanda.com/)
## on 2016-01-03
data("EUR_USD")
data("GBP_USD")
## The Brent data was obtained from
## Federal Reserve Economic Data
## (FRED) via Quandl on 2016-01-03
data("OIL_Brent")
data.1 <- na.omit(merge(EUR_USD, GBP_USD,
    OIL_Brent))
R <- na.omit(diff(log(data.1)) * 100)
```

```
names.R <- c("EUR.USD", "GBP.USD", "OIL.Brent")
colnames(R) <- names.R
```

First we compute the mean return.

```
(mean.R <- apply(R, 2, mean))
```

```
##      EUR.USD      GBP.USD    OIL.Brent
##  0.001538585 -0.002283062  0.010774203
```

Now some questions for us to consider:

1. Let's look at a `summary` of a few columns. Is there anything odd or curious?
2. What does the `2` indicate in the `apply` function.
3. What is Brent crude's annualized mean "return"?

Some immediate results ensue.

```
summary(R)
```

```
##       Index                EUR.USD             GBP.USD
##  Min.   :2000-01-05   Min.   :-2.522684   Min.   :-4.648461
##  1st Qu.:2003-12-18   1st Qu.:-0.308317   1st Qu.:-0.277715
##  Median :2007-12-05   Median : 0.013886   Median : 0.006097
##  Mean   :2007-12-19   Mean   : 0.001539   Mean   :-0.002283
##  3rd Qu.:2011-12-19   3rd Qu.: 0.322014   3rd Qu.: 0.286959
##  Max.   :2015-12-28   Max.   : 3.463777   Max.   : 3.140629
##    OIL.Brent
##  Min.   :-19.89065
##  1st Qu.: -1.15322
##  Median :  0.03604
##  Mean   :  0.01077
##  3rd Qu.:  1.24927
##  Max.   : 18.12974
```

Means are much less than medians. There are juge maximum and minimum returns. We can also look at `acf` and `ccf`, absolute returns, run GARCH models, and so on to hone our exploratory analysis.

We ook up `??apply` and read that the `2` indicates that we are calculating the mean for the second dimension of the data matrix, namely, the assets.

Brent crude's annualized mean return is calculated on a 252 average days traded in a year basis as:

```
(1 + mean.R[3]/100)^252 - 1
```

```
##  OIL.Brent
## 0.02752144
```

Some folks use 253 days. But this is all a back of the envelope computation.

10.3.2 Let's keep moving on...

So, what is the context? We have working capital with three main drivers of risk and return: two exchange rates and a commodity price. Over time we will want to know how these factors act and interact to produce EBITDA returns on Assets. Here EBITDA is Earnings Before Interest and Tax adding back in non-cash Depreciation and Amortization.

> Then, how does that combination compare with today's reality and especially answer the CFO's question of what to do about the millstone of working capital around the neck of EBITDA?

Given this context, and the data we found earlier on, we then calculate the variance-covariance matrix. The diagonals of this matrix are the variances, so that the square root of the diagonal will yield standard deviations. The off-diagonals can be converted to correlations as needed.

```
(mean.R <- apply(R, 2, mean))
```

```
##      EUR.USD      GBP.USD    OIL.Brent
##  0.001538585 -0.002283062  0.010774203
```

```
(cov.R <- cov(R))
```

```
##             EUR.USD   GBP.USD OIL.Brent
## EUR.USD   0.3341046 0.1939273 0.1630795
## GBP.USD   0.1939273 0.2538908 0.1809121
## OIL.Brent 0.1630795 0.1809121 5.0572328
```

```
(sd.R <- sqrt(diag(cov.R)))  ## remember these are in daily percentages
```

```
##   EUR.USD   GBP.USD OIL.Brent
## 0.5780178 0.5038758 2.2488292
```

Now for some programming (quadratic that is...) but first let's get more mathematical about the statement of the problem we are about to solve. In a mathematical nutshell we are formally (more tractable version to follow...) solving the problem of minimizing working capital factors risk, subject to target returns and a budget that says it all has to add up to our working capital position. We define weights as percentages of the total working capital position. Thus the weights need to add up to one.

$$
\begin{gathered}
min_w w_T \Sigma w \\
subject\,to \\
1^T w = 1 \\
w^T \mu = \mu_0
\end{gathered}
$$

where

- w are the weights in each instrument.
- Σ is the variance-covariance matrix we just estimated, `cov.R`.
- 1 is a vector of ones's with length equal to the number of instruments.
- μ are the mean returns we just estimated, `mean.R`.
- μ_0 is the target portfolio return.
- T is the matrix transpose.
- min_w means to find weights w that minimizes portfolio risk.

The expression $w_T \Sigma w$ is our measure of portfolio risk and is a quadratic form that looks like this for two instruments:

$$\begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

Multiplied out we get the following quadratic formula for portfolio variance:

$$\sigma_P^2 = w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + w_1 w_2 \sigma_{12} + w_2 w_1 \sigma_{21}$$

and because $\sigma_{12} = \sigma_{21}$ this reduces a bit to

$$\sigma_P^2 = w_1^2\sigma_1^2 + w_2^2\sigma_2^2 + 2w_1 w_2 \sigma_{12}$$

Tedious? Definitely. But useful to explain the components of portfolio risk

1. Two dashes of own asset risk $w_1^2\sigma_1^2 + w_2^2\sigma_2^2$, and
2. Two dashes of relational risk $2w_1 w_2 \sigma_{12}$

When $\sigma_{12} < 1$ we have *diversification.*

10.3.3 Try this exercise

Suppose we have two commodities (New York Harbor No. 2 Oil and Henry Hub Natural Gas) feeding a production process (Electricity Generation).These are the weights in this process:

$$w = \{w_{oil} = -.5, w_{ng} = -.5, w_{ele} = 1.0\}$$

The percentage changes in terms of the prices of these commodities are given by:

$$\mu = \{\mu_{oil} = 0.12, \mu_{ng} = -0.09, \mu_{ele} = 0.15\}.$$

Standard deviations are

$$\sigma = \{\sigma_{oil} = 0.20, \sigma_{ng} = 0.15, \sigma_{ele} = 0.40\}$$

The correlation matrix is

$$\rho = \begin{bmatrix} 1.0 & 0.2 & 0.6 \\ 0.2 & 1.0 & 0.4 \\ 0.6 & 0.4 & 1.0 \end{bmatrix}$$

Using the formula

$$\Sigma = (\sigma\sigma^T)\rho$$

we can calculate the variance-covariance matrix Σ using our `R` knowledge of arrays. [Hint: `t()` is the transpose of an array so that σ^T is `t(sigma)`.] We can also calculate the portfolio mean return and portfolio standard deviation.

Let's Use this `R` code to put all of this into action.

```
sigma <- c(0.2, 0.15, 0.4)
rho = c(1, 0.2, 0.6, 0.2, 1, 0.4, 0.6,
    0.4, 1)
(rho <- matrix(rho, nrow = 3, ncol = 3))
```

```
##      [,1] [,2] [,3]
## [1,]  1.0  0.2  0.6
## [2,]  0.2  1.0  0.4
## [3,]  0.6  0.4  1.0
```

```
(Sigma2 <- (sigma %*% t(sigma)) * rho)
```

```
##       [,1]   [,2]  [,3]
## [1,] 0.040 0.0060 0.048
## [2,] 0.006 0.0225 0.024
## [3,] 0.048 0.0240 0.160
```

The diagonals are the squared standard deviations. Next we tackle the portfolio average rate.

```
w <- c(-0.5, -0.5, 1)
mu <- c(0.12, -0.09, 0.15)
(mu.P <- t(w) %*% mu)
```

```
##       [,1]
## [1,] 0.135
```

Now we compute the portfolio average level of "risk".

```
(Sigma.P <- (t(w) %*% Sigma2 %*% w))^0.5
```

```
##           [,1]
## [1,] 0.3265348
```

What does all of this mean?

- Running a power-generating plant (or refinery, or distribution chain, ...) over time financially really means generating a spark spread: the margin between costs of inputs

natural gas and oil (the negative or short position) and revenue from the output
- The average spark spread rate of return for this plant is 10.67%. The spark spread is the difference between the price of electricity and the price of input fuels on a MWh basis (megawatt-hour).
- The standard deviation of the spark spread is 32.65%.

Our next job is to use these mechanics about portfolio means, standard deviations, and correlations to find the best set of weights that minimizes portfolio risk while attempting to achieve a target level of return.

10.4 Optimizing a Portfolio

To perform the optimization task we turn to the `quadprog` quadratic programming package (yes, parabolas are indeed very useful). We worked out a two-asset example that showed us clearly that the objective function has squared terms (and interactive product terms too). These are the tell-tale signs that mark the portfolio variance as quadratic. It is all in the weights.

After all of our wrangling above it is useful to define our portfolio optimization problem again here:

$$\begin{gathered} min_{(w)} w_T \Sigma w \\ subject\,to \\ 1^T w = 1 \\ w^T \mu = \mu_0 \end{gathered}$$

On the other hand here is what `quadprog` does in its native format.

$$\begin{gathered} min_d - d^T x + \tfrac{1}{2} x^T D x \\ subject\,to \\ A_{neq}^T x \geq b_{neq} \\ A_{eq}^T x = b_{eq} \end{gathered}$$

Now we need to transform these equations into portfolio paramters to solve our portfolio problem. We do this by setting

$$A_{eq}^T = \begin{bmatrix} 1^T \\ \mu^T \end{bmatrix}$$

This gives us a stack of equality constraints that looks like:

$$\begin{bmatrix} 1^T w \\ \mu^T w \end{bmatrix} = \begin{bmatrix} 1 \\ \mu_0 \end{bmatrix}$$

We will allow short positions, like the spark spread experiment above. This means we will not yet impose inequality constraints like $w \geq 0$.

Here is the setup code

```
library(quadprog)
Amat <- cbind(rep(1, 3), mean.R)  ## set the equality constraints matrix
mu.P <- seq(min(mean.R - 5e-04), max(mean.R +
    5e-04), length = 300)  ## set of 300 possible target portfolio returns
sigma.P <- mu.P  ## set up storage for std dev's of portfolio returns
weights <- matrix(0, nrow = 300, ncol = ncol(R))  ## storage for portfolio weights
colnames(weights) <- names.R
```

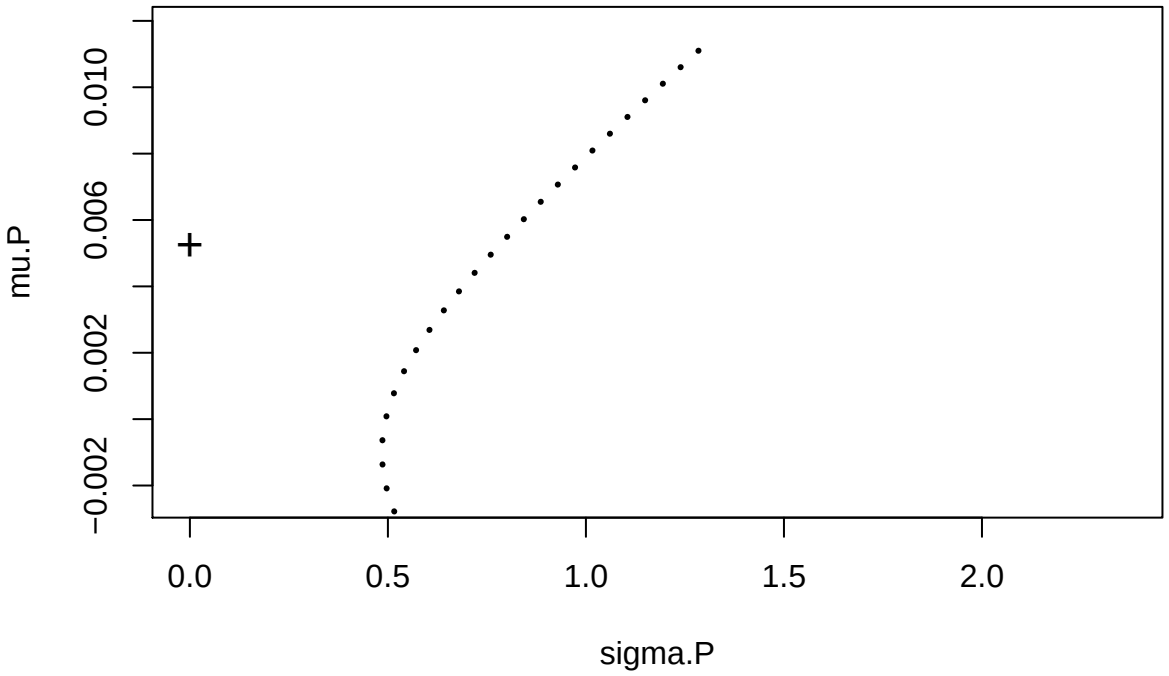
Next we build the "efficient frontier." This curve (a parabola…) traces optimal combinations of risk and return. For each combination there is an underlying set of weights calculated in successive optimizations, one for each target μ. In effect this is a very specialized sensitivity analysis.

```
for (i in 1:length(mu.P)) {
    bvec = c(1, mu.P[i])  ## constraint vector
    result = solve.QP(Dmat = 2 * cov.R,
        dvec = rep(0, 3), Amat = Amat,
        bvec = bvec, meq = 2)
    sigma.P[i] = sqrt(result$value)
    weights[i, ] = result$solution
}
```

First, we plot all of the portfolio combinations.

```
par(mfrow = c(1, 1))
plot(sigma.P, mu.P, type = "l", xlim = c(0,
    max(sd.R) * 1.1), ylim = c(0, max(mean.R) *
    1.1), lty = 3, lwd = 3)  ##  plot
## the efficient frontier (and
## inefficient portfolios below the
## min var portfolio)
mu.free = 1.3/253  ## input value of risk-free interest rate
points(0, mu.free, cex = 1, pch = "+")  ## show risk-free asset
```

Then we plot the point on the graph that represents the so-called risk-free (actually more like



default-free) asset.
Finally we deploy William Sharpe's ratio.

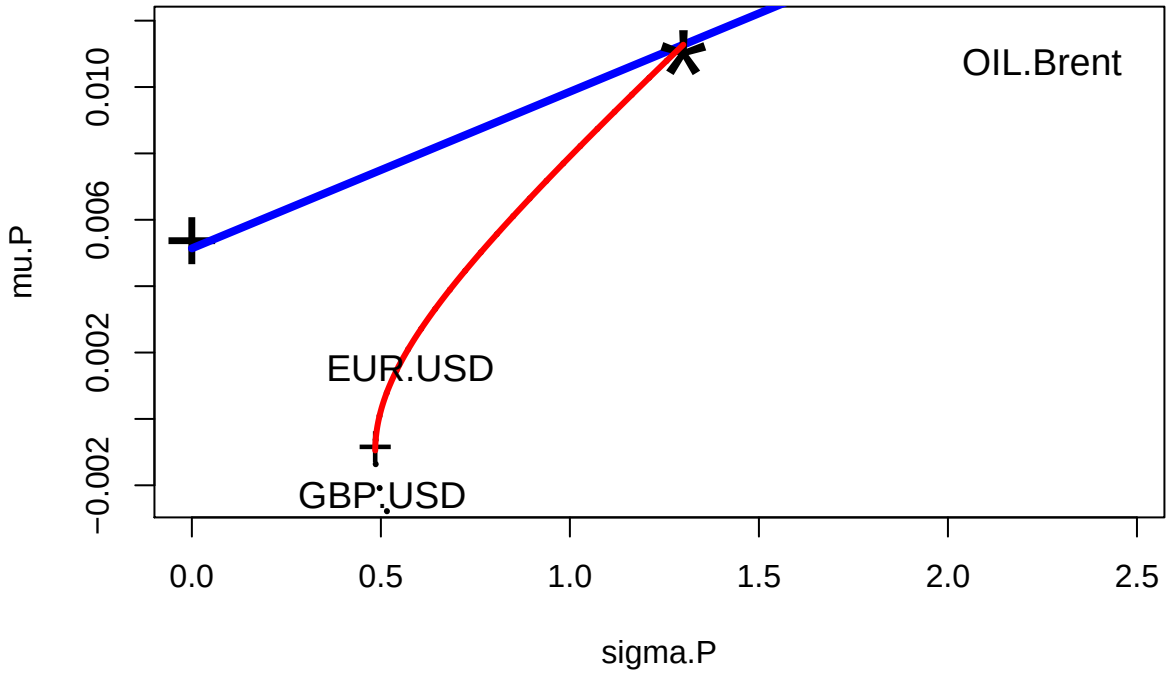
- This number is the amount of portfolio premium per unit of risk (the "price" of risk) across all combinations of portfolio assets on the efficient frontier. Its maximum is the best combination for the risk in terms of returns.
- We figure out where (the index `ind`) the return to risk is along the frontier, record the weights associated with this unique point in risk-return space, and
- We find where (the index `ind2`) the minimum variance portfolio is.
- We then plot the "efficient frontier": the efficient frontier will extend from the minimum variance portfolio (a "+" will mark the spot) up and out (in red). Anything else below this line is "inefficient" in the sense you get less and less return for mo

Here is the code we just built up.

```
sharpe = (mu.P - mu.free)/sigma.P  ## compute Sharpe's ratios
ind = (sharpe == max(sharpe))  ## Find maximum Sharpe's ratio
options(digits = 3)
lines(c(0, 2), mu.free + c(0, 2) * (mu.P[ind] -
    mu.free)/sigma.P[ind], lwd = 4, lty = 1,
    col = "blue")
## show line of optimal portfolios
points(sigma.P[ind], mu.P[ind], cex = 4,
    pch = "*")  ## show tangency portfolio
```

```
ind2 = (sigma.P == min(sigma.P))  ## find the minimum variance portfolio
points(sigma.P[ind2], mu.P[ind2], cex = 2,
    pch = "+")  ## show min var portfolio
ind3 = (mu.P > mu.P[ind2])  ## finally the efficient frontier
lines(sigma.P[ind3], mu.P[ind3], type = "l",
    xlim = c(0, max(sd.R) * 1.1), ylim = c(min(mean.R) *
        1.05, max(mean.R) * 1.1), lwd = 3,
    col = "red")  ##  plot the efficient frontier
text(sd.R[1], mean.R[1], "EUR.USD", cex = 1.15)
text(sd.R[2], mean.R[2], "GBP.USD", cex = 1.15)
text(sd.R[3], mean.R[3], "OIL_Brent",
    cex = 1.15)
```

Altogether we have these results.



The weights for the tangency portfolio ("*") are in

```
weights[ind, ]
```

```
##    EUR.USD    GBP.USD OIL.Brent
##      2.500     -1.807      0.306
```

```
sum(weights[ind, ])
```

```
## [1] 1
```

For a given notional amount in your portfolio, go long (buy) 250.% of that position in euros traded against USD, go short (sell) 180.7% of your aggregate position in euros traded against USD, and go long 30.6% in Brent.

This means in the working capital accounts:

1. $250 million should be denominated in euros
2. Net of a short (payables?) position of $180 million
3. With another $30 million priced in Brent crude.

If our working capital is $100 million in euros, -$200 in sterling, and $200 exposed to Brent, we might think of ways to bring this more into line with the optimal positions we just derived, by changing contract terms and using swaps and other derivative instruments.

10.4.1 Try this exercise

In this scenario we don't allow short positions (negative weights). This means we impose the inequality constraint:

$$w \geq 0$$

Further,

- We modify the `Amat` to `Amat` to `cbind(rep(1,3),mean.R,diag(1,nrow=3))`.
- We set the target return vector`mu.P` to `seq(min(mean.R)+.0001, max(mean.R)-.0001, length=300)`.
- We also set the righthand-side vector `bvec` to `c(1,mu.P[i],rep(0,3))`.

Let's watch what happens using these questions as a guide.

1. Are the tangency portfolio and minimum variance portfolio weights different?
2. Explain how the constraint matrix and target return vector are different from the first run where w was allowed to be negative.

Here is the new setup code where we no longer allow for short positions.

```
library(quadprog)
Amat <- cbind(rep(1, 3), mean.R, diag(1,
    nrow = 3))  ## set the equality ND inequality constraints matrix
mu.P <- seq(min(mean.R) + 1e-04, max(mean.R) -
    1e-04, length = 300)  ## set of 300 possible target portfolio returns
sigma.P <- mu.P  ## set up storage for std dev's of portfolio returns
weights <- matrix(0, nrow = 300, ncol = 3)  ## storage for portfolio weights
```

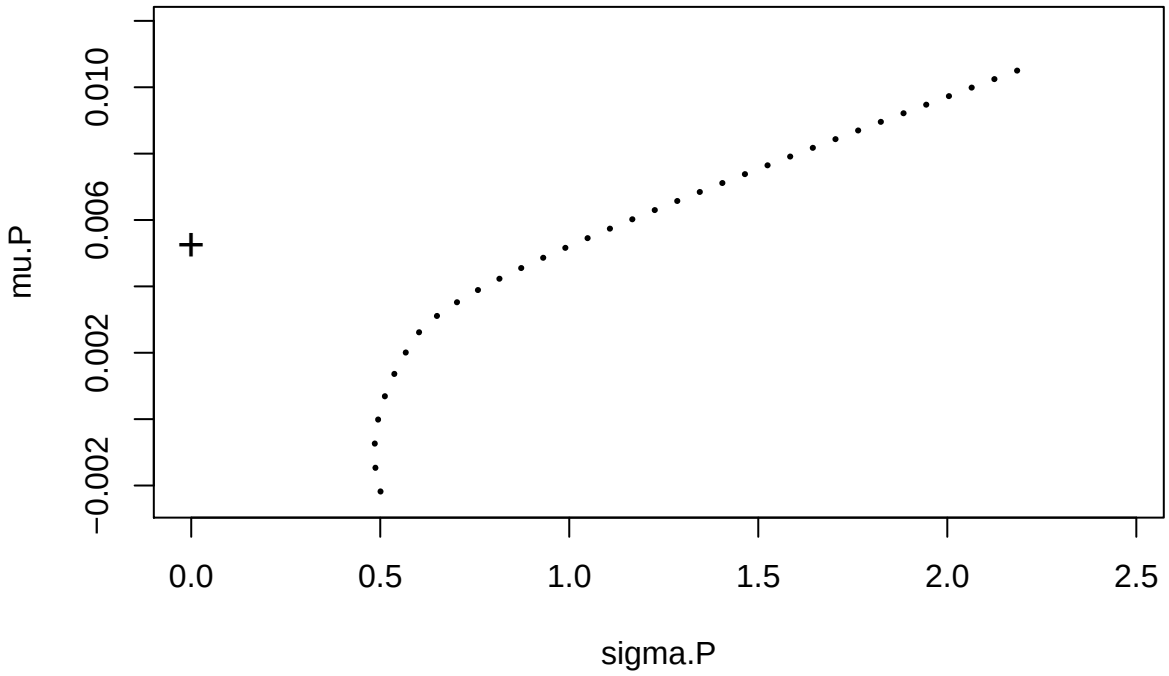
Next we build the "efficient frontier." All of this code is as before.

```
for (i in 1:length(mu.P)) {
    bvec <- c(1, mu.P[i], rep(0, 3))  ## constraint vector with no short positions
    result <- solve.QP(Dmat = 2 * cov.R,
        dvec = rep(0, 3), Amat = Amat,
        bvec = bvec, meq = 2)
    sigma.P[i] <- sqrt(result$value)
    weights[i, ] <- result$solution
}
```

Then we plot, again the same as before.

1. Plot all of the portfolio combinations.
2. Plot the point on the graph that represents the so-called risk-free (actually more like default-free) asset.

```
par(mfrow = c(1, 1))
plot(sigma.P, mu.P, type = "l", xlim = c(0,
    max(sd.R) * 1.1), ylim = c(min(mean.R) *
    1.05, max(mean.R) * 1.1), lty = 3,
    lwd = 3)  ##  plot the efficient frontier (and inefficient portfolios
## below the min var portfolio)
mu.free <- 1.3/253  ## input value of risk-free interest rate
points(0, mu.free, cex = 1.5, pch = "+")  ## show risk-free asset
```



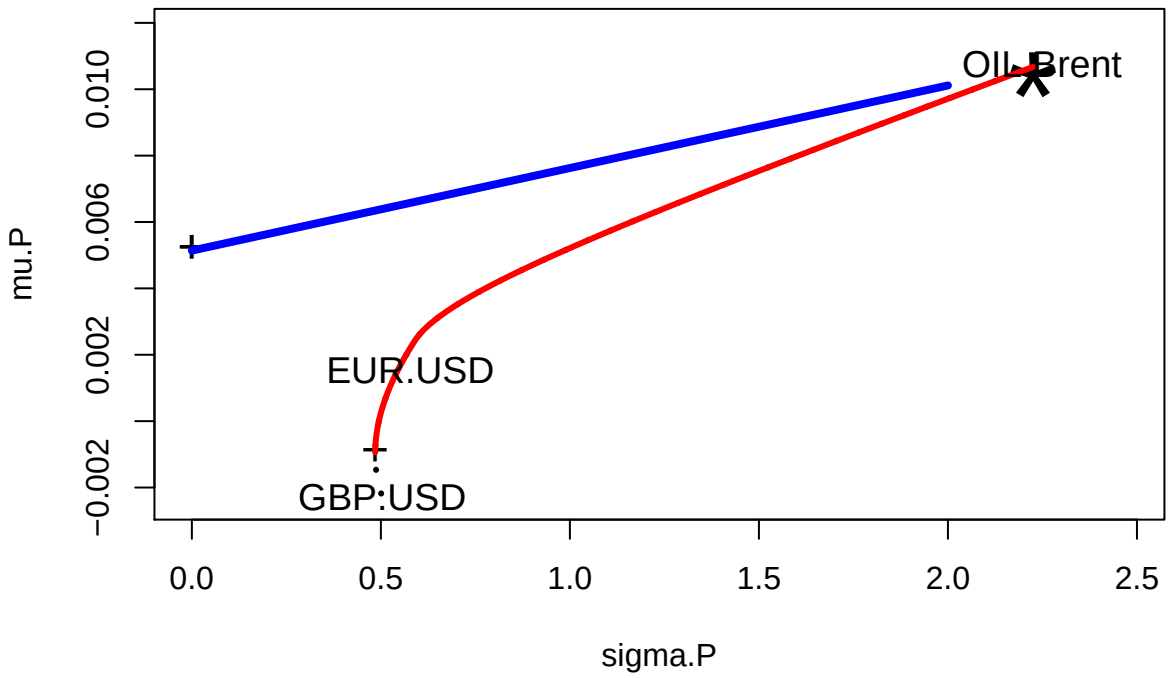
And...
Now for William Sharpe's ratio, again as before, where

- This number is the amount of portfolio premium per unit of risk (the "price" of risk) across all combinations of portfolio assets on the efficient frontier. Its maximum is the best combination for the risk in terms of returns.
- We figure out where (the index `ind`) the return to risk is along the frontier, record the weights associated with this unique point in risk-return space, and
- Find where (the index `ind2`) the minimum variance portfolio is.
- Plot the "efficient frontier": the efficient frontier will extend from the minimum variance portfolio (a "+" will mark the spot) up and out (in red). Anything else below this line is "inefficient" in the sense you get less and less return for more and m

Here is the code (again) for using Sharpe's ratio.

```
par(mfrow = c(1, 1))
plot(sigma.P, mu.P, type = "l", xlim = c(0,
    max(sd.R) * 1.1), ylim = c(min(mean.R) *
    1.05, max(mean.R) * 1.1), lty = 3,
    lwd = 3)  ##  plot the efficient frontier (and inefficient portfolios
## below the min var portfolio)
mu.free <- 1.3/253  ## input value of risk-free interest rate
points(0, mu.free, cex = 1.5, pch = "+")  ## show risk-free asset
mu.free <- 1.3/253  ## input value of risk-free interest rate
```

```
points(0, mu.free, cex = 1.5, pch = "+")  ## show risk-free asset
sharpe = (mu.P - mu.free)/sigma.P  ## compute Sharpe's ratios
ind = (sharpe == max(sharpe))  ## Find maximum Sharpe's ratio
options(digits = 3)
lines(c(0, 2), mu.free + c(0, 2) * (mu.P[ind] -
    mu.free)/sigma.P[ind], lwd = 4, lty = 1,
    col = "blue")
## show line of optimal portfolios
points(sigma.P[ind], mu.P[ind], cex = 4,
    pch = "*")  ## show tangency portfolio
ind2 = (sigma.P == min(sigma.P))  ## find the minimum variance portfolio
points(sigma.P[ind2], mu.P[ind2], cex = 1.5,
    pch = "+")  ## show min var portfolio
ind3 = (mu.P > mu.P[ind2])
lines(sigma.P[ind3], mu.P[ind3], type = "l",
    xlim = c(0, max(sd.R) * 1.1), ylim = c(min(mean.R) *
        1.05, max(mean.R) * 1.1), lwd = 3,
    col = "red")  ##  plot the efficient frontier
text(sd.R[1], mean.R[1], "EUR.USD", cex = 1.15)
text(sd.R[2], mean.R[2], "GBP.USD", cex = 1.15)
text(sd.R[3], mean.R[3], "OIL.Brent",
    cex = 1.15)
```




```
Amat
```

```
##                mean.R
## EUR.USD   1  0.00154 1 0 0
## GBP.USD   1 -0.00228 0 1 0
## OIL.Brent 1  0.01077 0 0 1
```

```
bvec
```

```
## [1] 1.0000 0.0107 0.0000 0.0000 0.0000
```

Here

1. `bvec` changes for each of the three assets. Here we see one of them.
2. The short position `bvec` has three zeros appended to it.
3. The `Amat` constraint matrix has the identity matrix appended to it to represent $w_i = 0$ in the formulation of the inequality constraints parsed by `quantprog`.
4. The tangency of the line from the risk-free rate to the maximum Sharpe ratio point on the efficient frontier does not change.

The weights are

```
## [1] 0.0108 0.0000 0.9892
```

The picture radically changes:

1. Long working capital position with only a $1 million euro exposure.
2. No pounding sterling exposure at all.
3. A huge $99 million Brent exposure.

10.5 Summary

We learned portfolio maths and finance: building feasible combinations of risk and return called the efficient frontier, figured out in 1952 by Harry Markowitz. We also looked at a simple example of tolerance for loss to imply the amount of collateral (risk-free asset) to hold. Using that idea and a ruler we drew a line to the efficient frontier to discover the best portfolio of exposures for a hypothetical working capital position: the one that maximizes the return for the risk, the ratio that William Sharpe figured out in 1966.

10.6 Further Reading

McNeil et al. has sections on . Ruppert et al. has a chapter on portfolio selection upon which much of the `R` programming in this chapter is based. Bassett et al. has a way forward for more risk management minded analytics that uses quantile regression techniques.

10.7 Practice Laboratory

10.7.1 Practice laboratory #1

10.7.1.1 Problem

10.7.1.2 Questions

10.7.2 Practice laboratory #2

10.7.2.1 Problem

10.7.2.2 Questions

10.8 Project

10.8.1 Background

10.8.2 Data

10.8.3 Workflow

10.8.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.
- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.
- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.
- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or

procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.

- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.
- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.
- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.
- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

10.9 References

Bassett, Gilbert W., Jr., Roger Koenker, and Gregory Kordas. 2004. Pessimistic Portfolio Allocation and Choquet Expected Utility. Journal of Financial Econometrics (2004) 2 (4): 477-492.

Markowitz, Harry. 1952. Portfolio Selection. The Journal of Finance, Vol. 7, No. 1. (March, 1952), pp. 77-91.

McNeill, Alexander J., Rudiger Frey, and Paul Embrechts. 2015. Quantitative Risk Management: Concepts, Techniques and Tools. Revised Edition. Princeton: Princeton University Press.

Ruppert, David and David S. Matteson. 2015. Statistics and Data Analysis for Financial Engineering with R Examples, Second Edition. New York: Springer.

Sharpe, William F. 1966. "Mutual Fund Performance." Journal of Business, January 1966, pp. 119-138.

Chapter 11

Aggregating Enterprise Risk

11.1 The Problem with Enterprise Risk

International Mulch & Compost Company[1]IM&C os a ficitious company dreamt up and used by Brealey and Myers.) makes and distributes an emerging energy source made from guano and prairie grass briquets. IM&C is about to go IPO. Corporate policy dictates that management must assess risks to equity annually and whether a circumstance dictates. Such a circumstance is an IPO.

Management knows of at least three material risks:

- Customers defect so there is uncertainty in revenue growth.
- Suppliers stop competing on price, quantity, and quality so there is uncertainty in variable expense.
- There are major compliance breaches which impact fixed expense.

No one knows much about these risks from history because this company is the first in its market to produce this very innovative product from bio-engineered guano. Very abundant prairie grass grows alongside every highway in North America. Management does have considerable experience in marketing, production, and operations. IM&C ponders its SEC disclosure for the IPO where it will report its view of material risks. One question management knows *someone* will ask is how likely is it that the net operating margin will fall below, say, indicated earnings of $400 million. IM&C thinks it needs to know how much capital is involved in this risk venture.

11.2 Let's make copulas

Our problem is:

[1](

1. We have three major risk factors and each has their own distribution.
2. We also know that they are somehow correlated.
3. How can we aggregate the three into one risk measure that is tangible, and preserve the correlation?

11.2.1 We do this from scratch.

Our first task is to generate multivariate normal variates that are correlated with one another. Here we relate three standard normal random variables together.A standard normal random variable has a mean, $\mu = 0$, and variance, $\sigma^2 = 1$. The variable `sigma` in the code below is the *correlation* matrix.

```
library(mvtnorm)
set.seed(1016)
n.risks <- 3  ## Number of risk factors
m <- n.risks
n.sim <- 1000
sigma <- matrix(c(1, 0.4, 0.2, 0.4, 1,
    -0.8, 0.2, -0.8, 1), nrow = 3)
z <- rmvnorm(n.sim, mean = rep(0, nrow(sigma)),
    sigma = sigma, method = "svd")
```

In the `rmvnorm` function `svd` stands for the "singular value decomposition" that allows us to fan the correlations across the `z` values.

11.2.2 Example

Let's use the `.panels` feature in the `psych` library to look at the variates so far. We also calculate two kinds of correlations, *Spearman* and *Pearson*.

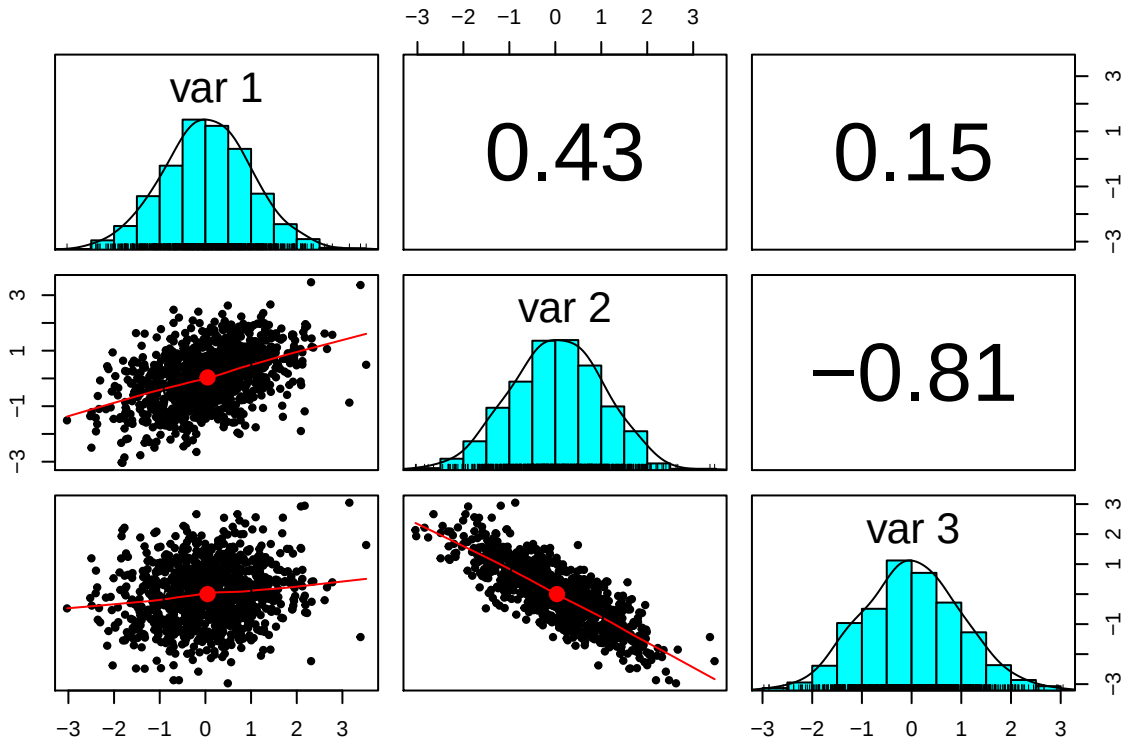
```
library(psych)
cor(z, method = "spearman")  ## Textbook calculation
```

```
##       [,1]   [,2]   [,3]
## [1,] 1.000  0.408  0.139
## [2,] 0.408  1.000 -0.801
## [3,] 0.139 -0.801  1.000
```

```
cor(z, method = "pearson")  ## Rank order calculation
```

```
##       [,1]   [,2]   [,3]
## [1,] 1.000  0.426  0.152
## [2,] 0.426  1.000 -0.811
## [3,] 0.152 -0.811  1.000
```

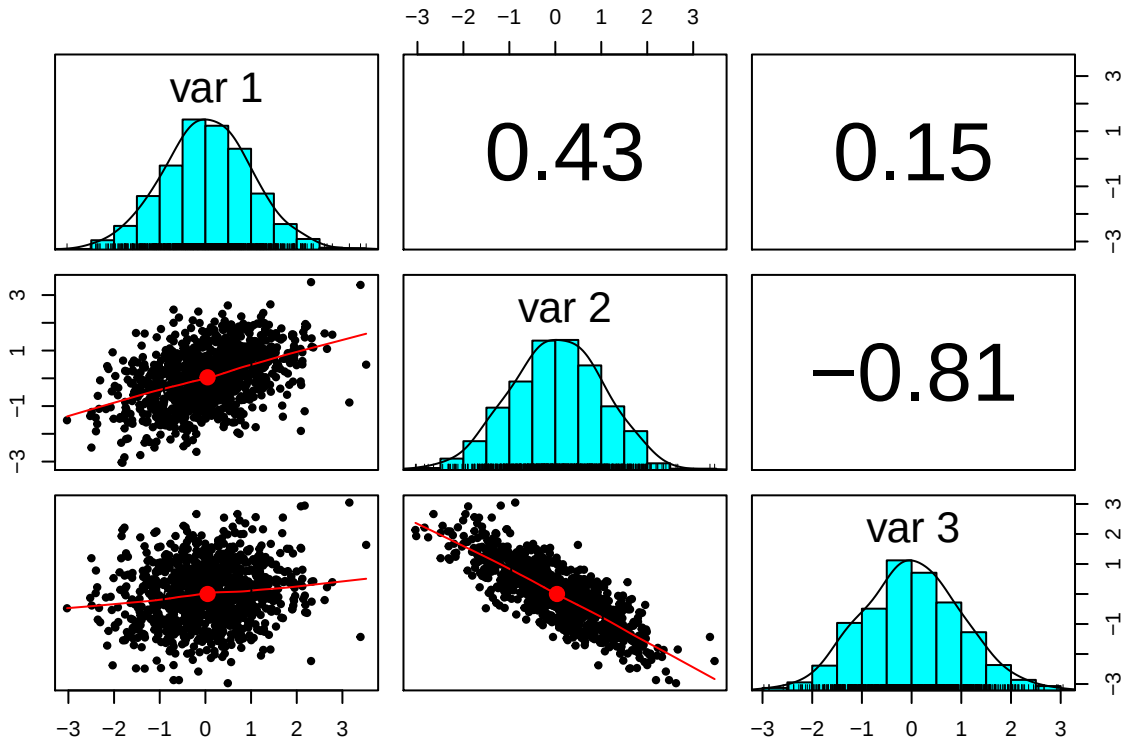
```
pairs.panels(z)
```



Here is the result.

```
##        [,1]   [,2]   [,3]
## [1,] 1.000  0.408  0.139
## [2,] 0.408  1.000 -0.801
## [3,] 0.139 -0.801  1.000
```

```
##        [,1]   [,2]   [,3]
## [1,] 1.000  0.426  0.152
## [2,] 0.426  1.000 -0.811
## [3,] 0.152 -0.811  1.000
```



Notice how close the correlations are to the ones we specified in `sigma`.

11.3 Sklar's in the house...

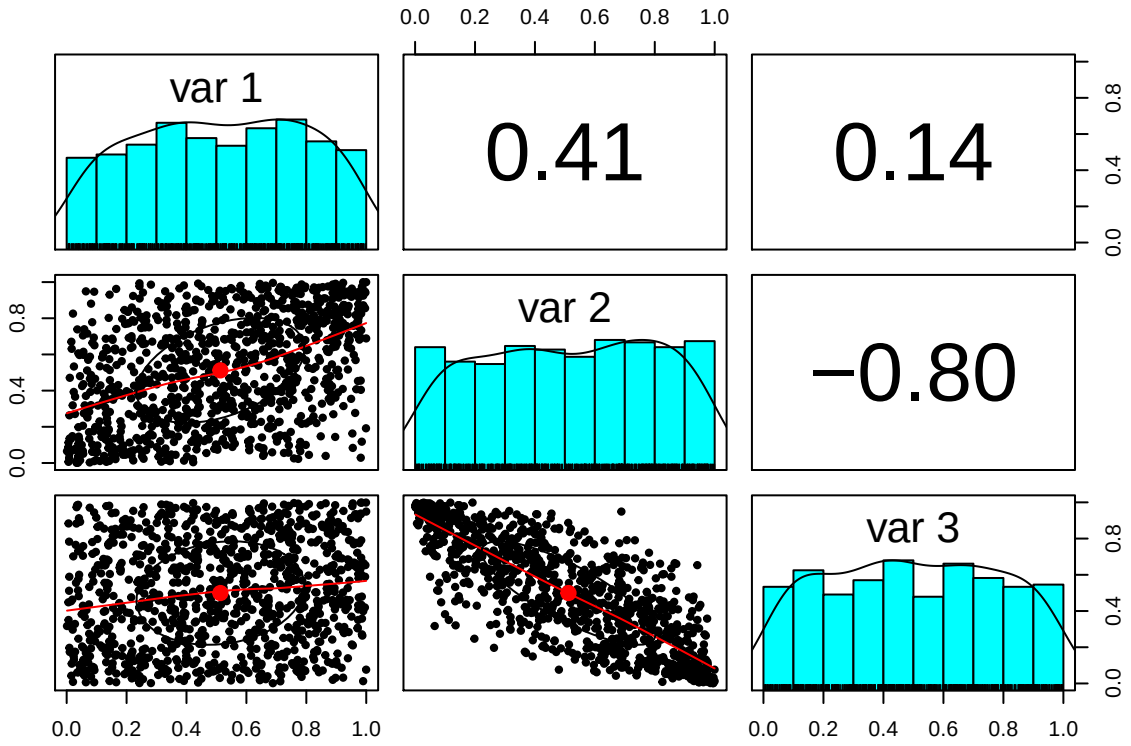
Next we use a result from mathematical probability called

Sklar's theorem (1959):

- If x is a random variable with distribution F,
- then $F(x)$ is uniformly distributed in the interval $[0, 1]$.

Let's translate this idea into `R` and look at the resulting interactions.

```
require(psych)
u <- pnorm(z)
pairs.panels(u)
```



We see that the Gaussian (normal) distribution has been reshaped into a uniform distribution, just as Sklar predicted. The idea around this theorem is the same as around the number 1. We can multiply any real number by one and get the real number back. This is an identity operation. (Please remember we are not trying to be mathematicians! My apologies to the mathematical community.) In a somewhat analogous way, the uniform distribution serves a role as an distribution identity operator.When we operate on the uniformly distributed random numbers with a distribution, we get back that distribution. But in this case the identity distribution has structure in it (correlations) that the new distribution inherits.

A 3-D plot looks more interesting. In the Rstudio graphics device window we can the roll the cube around to see into the relationships among the random variables. Try this at home for an interactive experience.

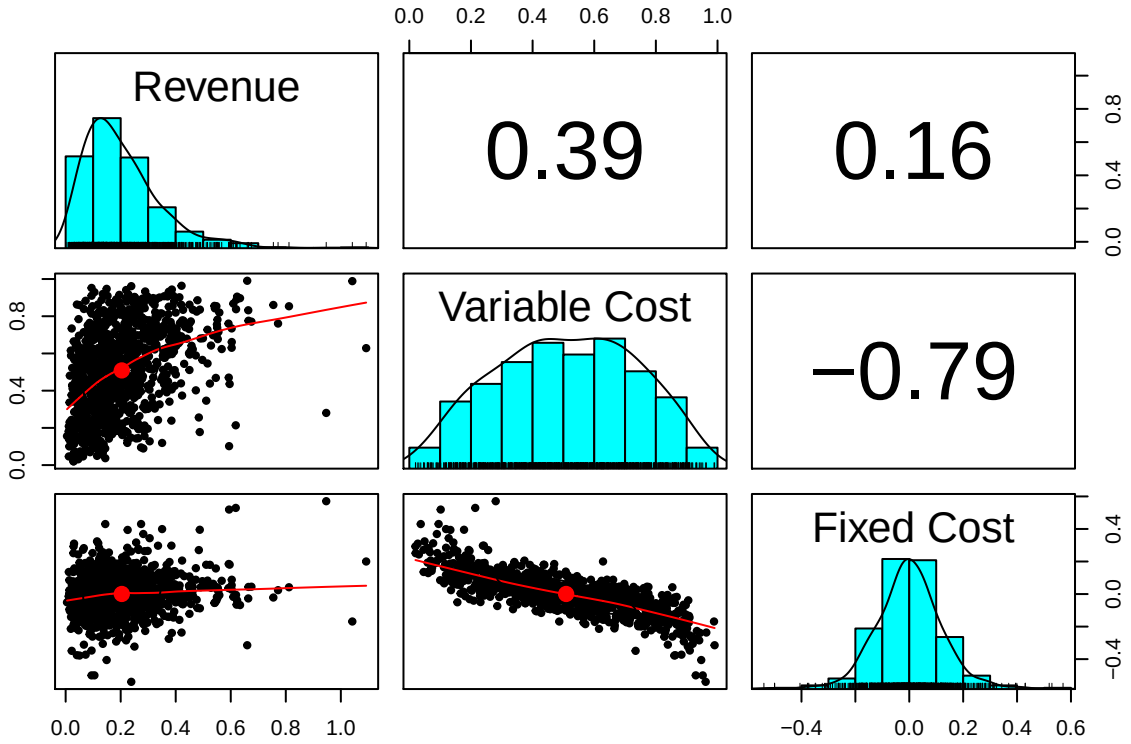
```
library(rgl)
plot3d(u[, 1], u[, 2], u[, 3], pch = 20,
    col = "orange")
```

Now, we only need to select the marginal probabilities of the risks we are assessing and apply them to the dependently related 'u' variates. Suppose the marginal probability distributions for revenue growth is `gamma`, variable expense ratio is `beta`, and the fixed expense ratio is Student's t distributed with these parameters:

```
x1 <- qgamma(u[, 1], shape = 2, scale = 1)
x2 <- qbeta(u[, 2], 2, 2)
x3 <- qt(u[, 3], df = 5)
```

Nice outliers! Starting from a multivariate normal distribution we created dependent uniform variates. Using the dependent uniform variates we created dependent distributions of our choosing.

```
factors.df <- cbind(x1/10, x2, x3/10)
colnames(factors.df) <- c("Revenue",
    "Variable Cost", "Fixed Cost")
pairs.panels(factors.df)
```




```
cor(factors.df, method = "spearman")
```

```
##                Revenue Variable Cost Fixed Cost
## Revenue          1.000         0.408      0.139
## Variable Cost    0.408         1.000     -0.801
## Fixed Cost       0.139        -0.801      1.000
```

11.4 Analyze that...

Now to use all of this simulation to project revenue, expense, and margin.

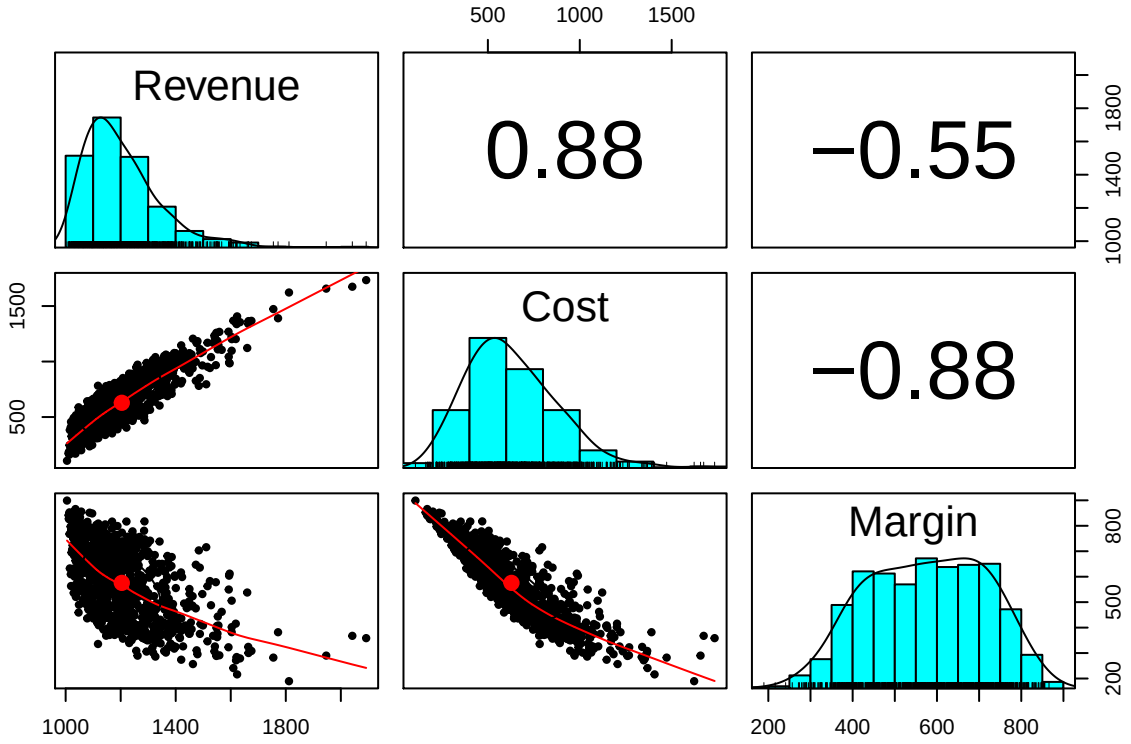
```
revenue <- 1000 * (1 + factors.df[, 1])
variable.cost <- revenue * factors.df[,
    2]
fixed.cost <- revenue * factors.df[,
    3]
total.cost <- variable.cost + fixed.cost
operating.margin <- revenue - variable.cost -
    fixed.cost
analysis <- cbind(revenue, total.cost,
    operating.margin)
colnames(analysis) <- c("Revenue", "Cost",
    "Margin")
```

11.4.1 Example

Run `pairs.panels` using the `analysis` data frame. What do you see?

Here's the result.

```
pairs.panels(analysis)
```



What do we see?

1. Variable and fixed cost aggregate into a distribution that is right-skewed.
2. Margin has a high density across a broad range of potential outcomes.
3. An increase (decrease) in cost will probably result in an increase (decrease) in revenue.
4. Revenue and margin also seem to be counter cyclical, a non-intuitive result, but one that makes sense only by looking at the negative correlation between cost and margin.

11.5 Risk measures

We are not yet done. The whole point of this analysis is to get consistent and coherent measures of risk to a consumer of the analysis, namely, the decision maker who is the CFO in this case. We define the value at risk, VaR, as the α quantile of the performance metric of interest. Higher α means lower risk tolerance. Here is the relationship:

$$Q(x, \alpha) = F(x; Prob[X] > \alpha).$$

The metric x in this case is margin. Expected Shortfall, ES, is then the mean of the margin beyond VaR. The parameter α is the level of organizational risk tolerance. If $\alpha = 0.99$, then the organization would want risk capital to cover a potential loss of VaR, and more conservatively, ES. The organization is even more conservative the higher the α.

We purloin the R code from the market risk material here:

```
### Simple Value at Risk
expected.margin <- 400
## Center margin loss on expected
## margin
loss.rf <- -(expected.margin - operating.margin)
## Assign metric of interest to
## reusable code
summary(loss.rf)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    -211      63     180     176     290     499
```

```
## Always review a key variable's
## content
alpha.tolerance <- 0.99
## Very intolerant!  Remember that
## putting a variable assignment in
## parentheses also prints the result
(VaR.hat <- quantile(loss.rf, probs = alpha.tolerance,
    names = FALSE))
```

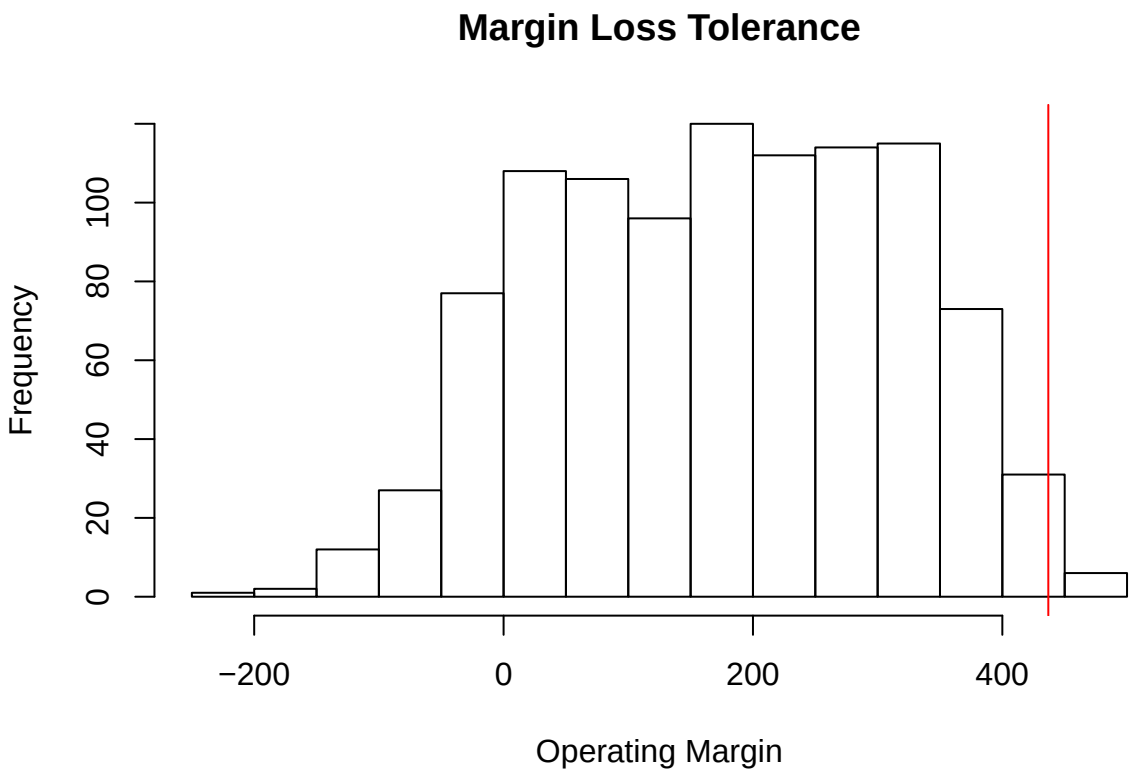
```
## [1] 437
```

```
### Just as simple Expected shortfall
(ES.hat <- mean(loss.rf[loss.rf > VaR.hat]))
```

```
## [1] 456
```

Let's plot the results.

```
hist(loss.rf, xlab = "Operating Margin",
    ylab = "Frequency", main = "Margin Loss Tolerance")
abline(v = VaR.hat, col = "red")
```



Sklar provides us with a way to join together any set of distributions. It transforms correlated variates into a uniform distribution. The uniform distribution takes on the role of the number 1 in algebra. Anything multiplied by 1 returns itself. In a very loose way, the uniform distribution is the identity distribution, just like one is the identity term in algebra. So that whenever we operate on the uniform distribution we get back the same distribution – but this time with correlation.

The rub is the starting point. Here we used the Gaussian (normal) distribution. This is not a very thickly tailed distribution, and it can be shown that extreme events are not dependent on one another using this distribution. This is NOT a useful feature ultimately. So, analysts use more thickly tailed distributions such as the Student-t and the generalized Pareto distribution (GPD) to get dependency far out into the tails. This is nearly perfect for risk managers and decision makers.

11.5.1 Example

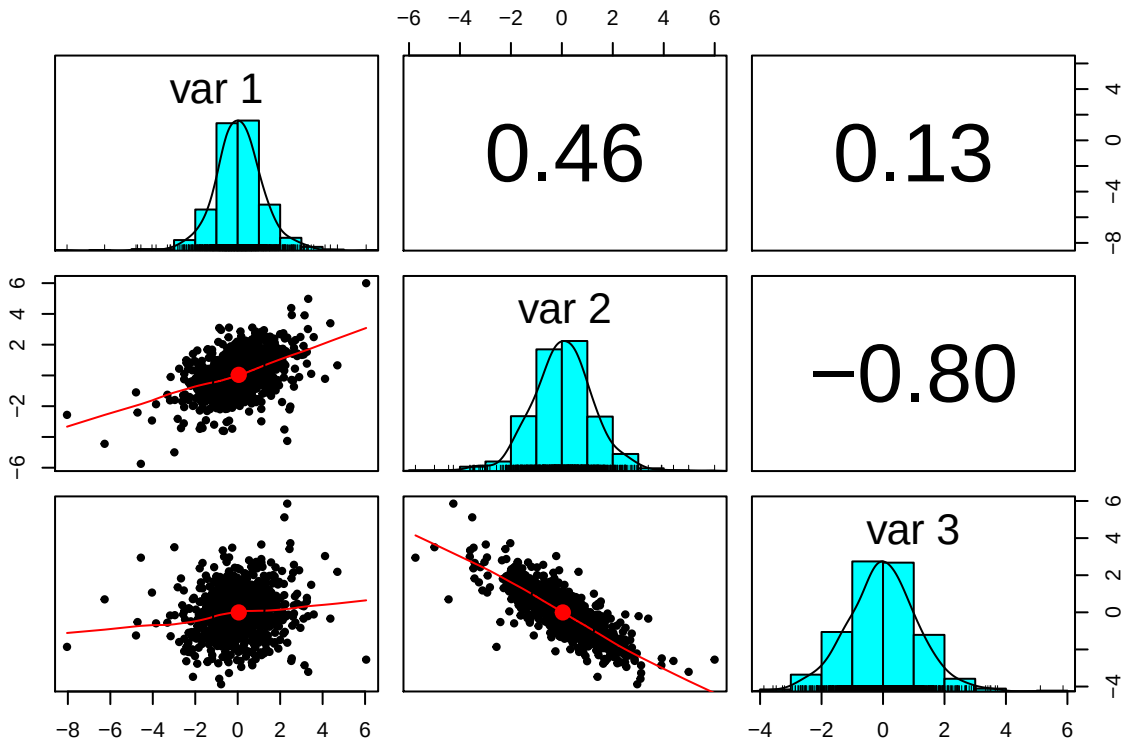
Let's use this R code to modify the copula-making machine we just built. Instead of `rmvnorm` we will use `rmvt` to generate the correlated risk factors. This is called a **t-copula**.

```
library(mvtnorm)
library(psych)
```

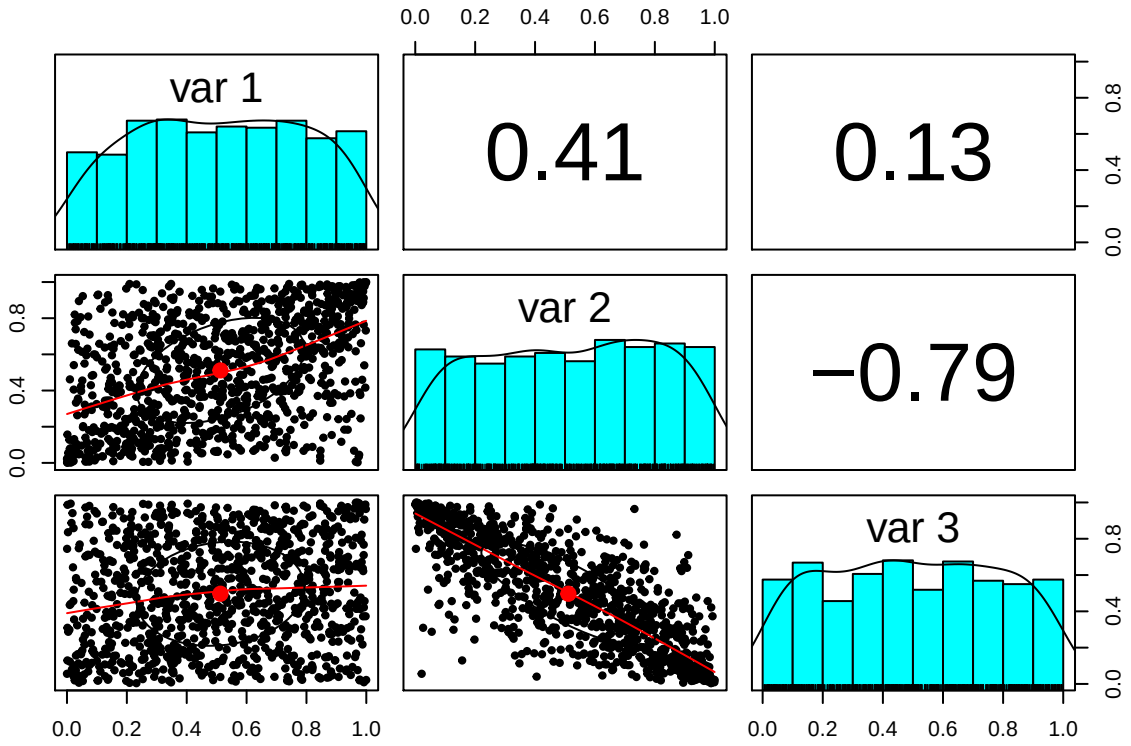
```
set.seed(1016)  ## Freezes the random seed to reproduce results exactly
n.risks <- 3  ## Number of risk factors
m <- n.risks
n.sim <- 1000
sigma <- matrix(c(1, 0.4, 0.2, 0.4, 1,
    -0.8, 0.2, -0.8, 1), nrow = m)
z <- rmvt(n.sim, delta = rep(0, nrow(sigma)),
    sigma = sigma, df = 6, type = "shifted")
```

Here are the results of our experiment. Let's go through the paces. First we look at the `z` variates we simulated using the multivariate Student's t-distribution.

```
pairs.panels(z)
```



We then run the uniform distribution generator (with correlation structure).



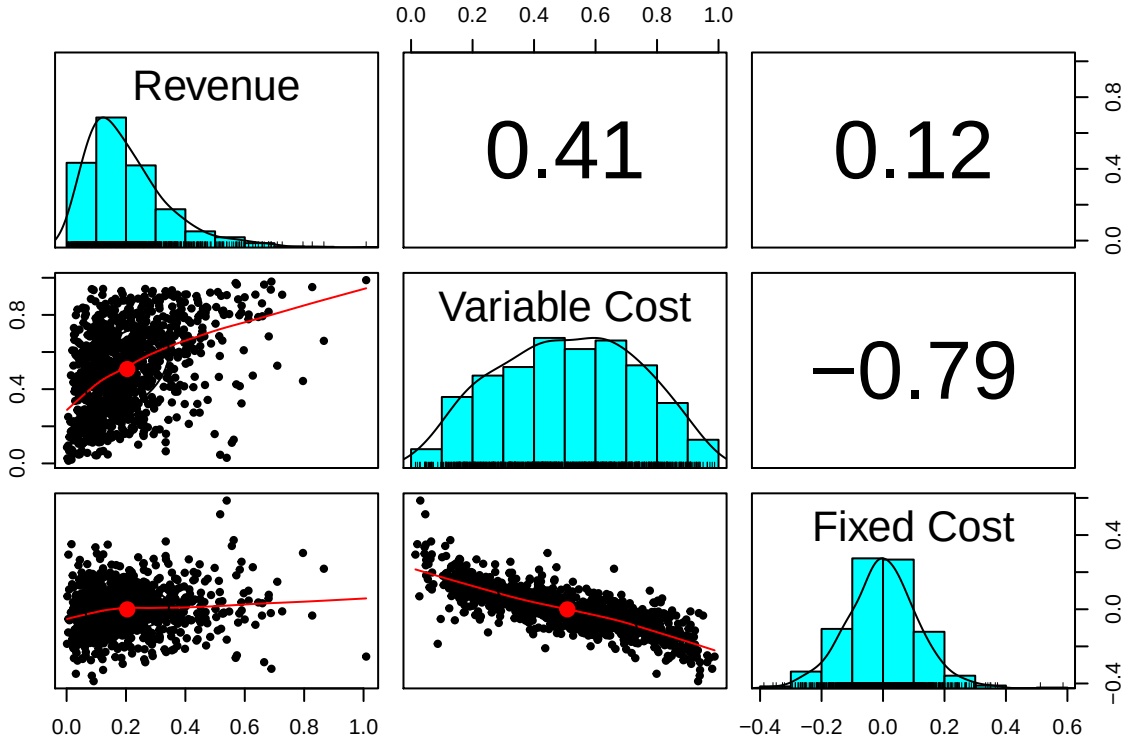
Now, we only need to select the marginal probabilities of the risks we are assessing and apply them to the dependently related 'u' variates. Again suppose the marginal probability for revenue growth is `gamma`, for the variable expense ratio is `beta`, and fixed expense ratio is Student's t distributed with these parameters:

```
x1 <- qgamma(u[, 1], shape = 2, scale = 1)
x2 <- qbeta(u[, 2], 2, 2)
x3 <- qt(u[, 3], df = 6)
```

Starting from a multivariate Student's t-distribution we created dependent uniform variates. Using the dependent uniform variates we created dependent distributions of our choosing.

Next we combine the series into a data frame and review the scatterplot matrix.

```
factors.df <- cbind(x1/10, x2, x3/10)
colnames(factors.df) <- c("Revenue",
    "Variable Cost", "Fixed Cost")
pairs.panels(factors.df)
```



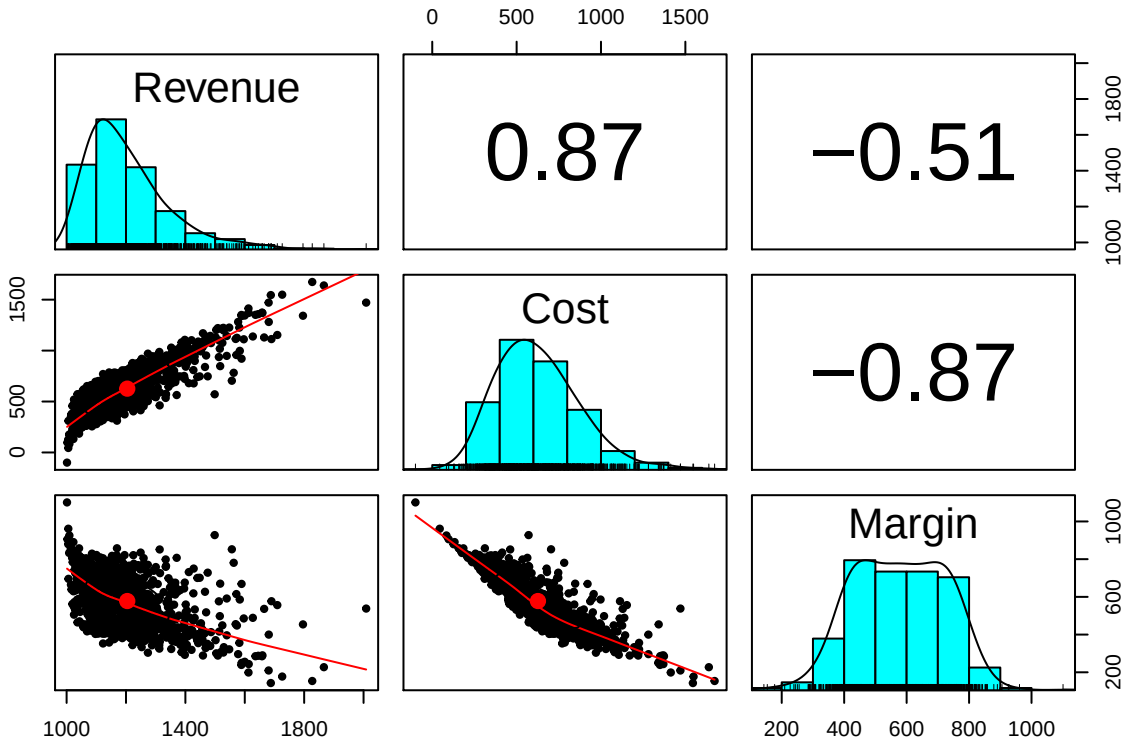

```
## cor(df,meth='spearman') could also
## be run to verify the pairs.panels()
```

Again, we have nice outliers! (We could run the `qqplot` to see this). Now to use all of this to project revenue, expense, and margin.

```
revenue <- 1000 * (1 + factors.df[, 1])
variable.cost <- revenue * factors.df[,
    2]
fixed.cost <- revenue * factors.df[,
    3]
total.cost <- variable.cost + fixed.cost
operating.margin <- revenue - variable.cost -
    fixed.cost
analysis.t <- cbind(revenue, total.cost,
    operating.margin)
colnames(analysis.t) <- c("Revenue",
    "Cost", "Margin")
```

And again here is the scatterplot matrix.

```
pairs.panels(analysis.t)
```



We can…

1. Experiment with different degrees of freedom to sensitive ourselves to the random numbers generated.
2. Parameterize correlations. This means assign correlations to a variable and place that variable into the `sigma` matrix. This might get into trouble with an error. It would mean we would have to reassign the correlation. The mathematical problem is finding a **positive definite** variance-covariance matrix.
3. How different are the value at risk and expected shortfall measures between the use of the Gaussian (normal) copula and the t-copula? Why should a decision maker care?

All of that experimentation begs for an interactive decision tool.

11.6 Let's build an app …

The application (the "app") will be housed in an `R` script that contain four architectural layers.

1. Analytics

2. User Interface (UI)
3. Server
4. Application generator

11.6.1 Analytics

1. Libraries used in app processes
2. Function that wraps analytical script
3. Inputs from UI layer to server layer
4. Outputs from server layer to UI layer

11.6.2 UI

1. Slide bars for user to input range of parameters
2. Plots to display results
3. Text to report results

11.6.3 Server

1. Run analytics with inputs from the UI and from a simulation function
2. Generate outputs for UI

11.6.4 Application generator

Here we run application function with UI and Server inputs

11.7 The simulation function

The `risk.sim` function is a wrapper that pulls all of the risk aggregation together. In our scenario we vary the correlation coefficients. `Shiny` calls these `input` and this is what is given to `risk.sim` through the `ui` to `risk.sim` by way of the `server`. `risk.sim` then outputs the results into result called `analysis.t`. This is fetched by the `server` and rendered in the app.

```
library(shiny)
require(mvtnorm)
require(psych)
risk.sim <- function(input) {
    ## Begin enterprise risk simulation
    set.seed(1016)  ## Freezes the random seed to reproduce results exactly
```

```
    n.risks <- 3  ## Number of risk factors
    m <- n.risks
    n.sim <- 1000  ## pull slider settings into the sigma correlation matrix
    sigma <- matrix(c(1, input[1], input[2],
        input[1], 1, input[3], input[2],
        input[3], 1), nrow = m)
    z <- rmvt(n.sim, delta = rep(0, nrow(sigma)),
        sigma = sigma, df = 6, type = "shifted")
    u <- pt(z, df = 6)
    x1 <- qgamma(u[, 1], shape = 2, scale = 1)
    x2 <- qbeta(u[, 2], 2, 2)
    x3 <- qt(u[, 3], df = 6)
    factors.df <- cbind(x1/10, x2, x3/10)
    colnames(factors.df) <- c("Revenue",
        "Variable Cost", "Fixed Cost")
    revenue <- 1000 * (1 + factors.df[,
        1])
    variable.cost <- revenue * factors.df[,
        2]
    fixed.cost <- revenue * factors.df[,
        3]
    total.cost <- variable.cost + fixed.cost
    operating.margin <- revenue - variable.cost -
        fixed.cost
    analysis.t <- cbind(revenue, total.cost,
        operating.margin)
    colnames(analysis.t) <- c("Revenue",
        "Cost", "Margin")
    return(analysis.t)
}
```

11.8 The UI

Here is a mock-up of the screen we will implement in Shiny.

Here is what the Shiny UI code looks like:

```
ui <- fluidPage(titlePanel("Enterprise Risk Analytics"),
    sidebarLayout(sidebarPanel(sliderInput(inputId = "cor.1",
        label = "Set the Revenue - Variable Cost Correlation",
        value = 0.5, min = 0.1, max = 0.9),
        sliderInput(inputId = "cor.2",
            label = "Set the Revenue - Variable Cost Correlation",
```

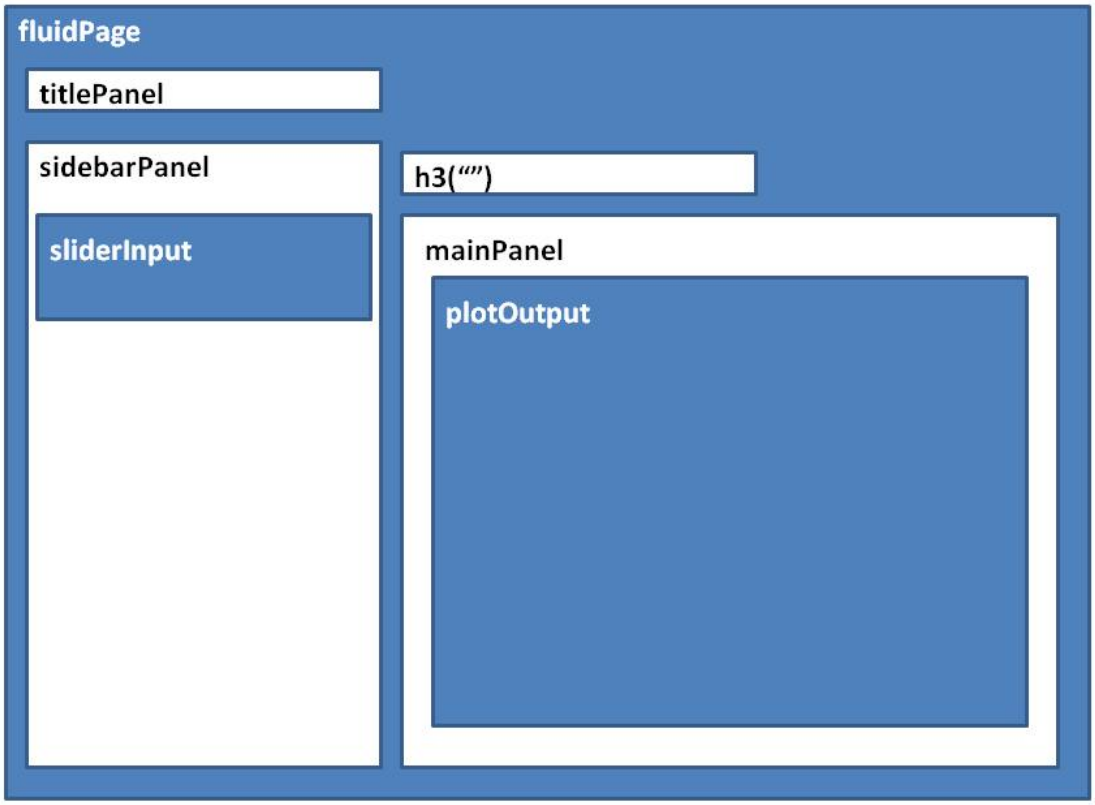


Figure 11.1: UI Design

```
            value = 0.5, min = 0.1, max = 0.9),
        sliderInput(inputId = "cor.3",
            label = "Set the Variable - Fixed Cost Correlation",
            value = 0.5, min = 0.1, max = 0.9)),
        mainPanel(plotOutput("pairs.1"))))
```

11.9 The server

- The Shiny server is a function
- The function gets inputs from the UI
- Generates outputs that are sent back to the UI

```
server <- function(input, output) {
    output$pairs.1 <- renderPlot({
        analysis.t <- risk.sim(c(input$cor.1,
            input$cor.2, input$cor.3))
        pairs.panels(analysis.t)
    })
}
```

11.10 Run the app

This function call the Shiny application process with inputs `ui` and `server`.

```
shinyApp(ui = ui, server = server)
```

Here is what you see when you run the app in the script window of Rstudio.

11.11 What else could we do?

- Build tabs for various components of the analysis
- Use tables to summarize metrics (e.g., VaR, ES)
- Whatever else the consumer of this analysis would need

11.12 Summary

- More and more `R`, finance, risk, statistics, probability
- Multivariate simulation of risk factors
- Math to `R` translation

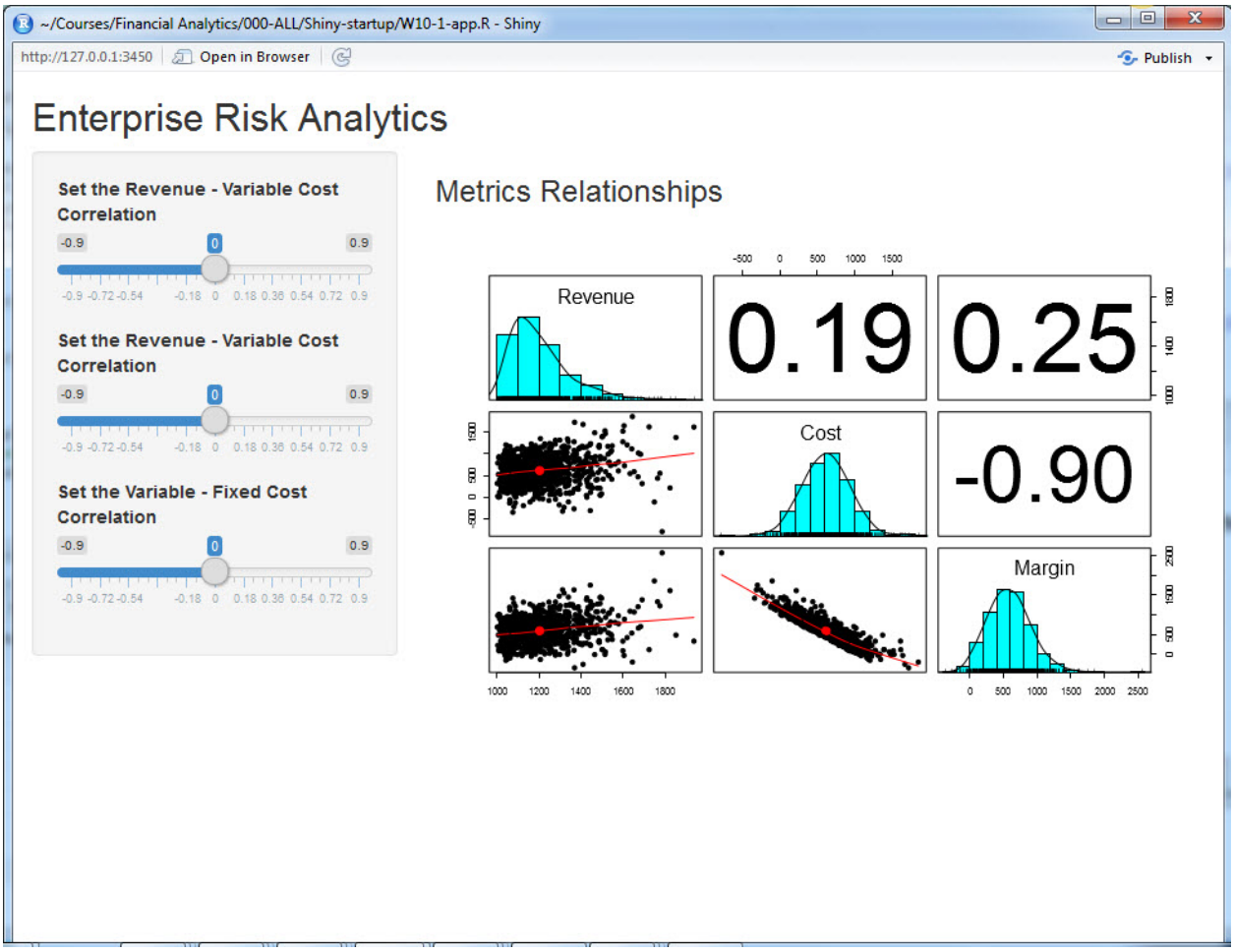


Figure 11.2: ERM Application Screenshot

- Graphics
- Normal, t, gamma, and beta distributions
- VaR and ES
- Aggregation of multiple risk factors
- Introduction to `Shiny` and application development

11.13 Further Reading

11.14 Practice Laboratory

11.14.1 Practice laboratory #1

11.14.1.1 Problem

11.14.1.2 Questions

11.14.2 Practice laboratory #2

11.14.2.1 Problem

11.14.2.2 Questions

11.15 Project

11.15.1 Background

11.15.2 Data

11.15.3 Workflow

11.15.4 Assessment

We will use the following rubric to assess our performance in producing analytic work product for the decision maker.

- **Words:** The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences, logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.

- **Numbers:** All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.

- **Pictures:** All figures and tables shown are relevant to the argument for ultimate conclusions. Figures and tables are easy to read, with informative captions, titles, axis labels and legends, and are placed near the relevant pieces of text.

- **Code:** The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the `R Markdown` file `knits` to `pdf_document` output, or other output agreed with the instructor.

- **Modeling:** Model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.

- **Inference:** The actual estimation and simulation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.

- **Conclusions:** The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers (for example, "if X, then Y , but if A, then B, else C") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.

- **Sources:** All sources used, whether in conversation, print, online, or otherwise are listed and acknowledged where they used in code, words, pictures, and any other components of the analysis.

11.16 References

11.17 R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

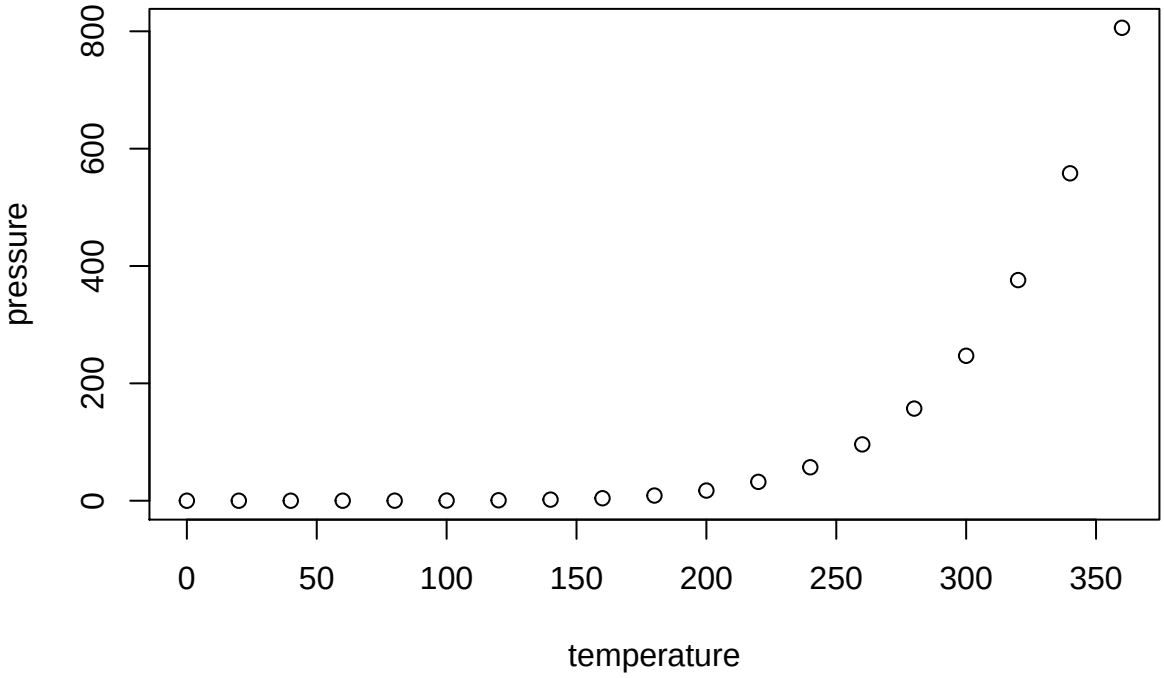
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2
##  1st Qu.:12.0   1st Qu.: 26
##  Median :15.0   Median : 36
##  Mean   :15.4   Mean   : 43
##  3rd Qu.:19.0   3rd Qu.: 56
##  Max.   :25.0   Max.   :120
```

11.18 Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Bibliography