Data Visualization with ggplot2 :: CHEAT SHEET

Basics

\textbf{ggplot2} is based on the grammar of graphics, the idea that you can build every graph from the same components: a \textit{data} set, a \textit{coordinate system}, and \textit{geoms}—visual marks that represent data points.

\begin{align*}
\text{data} & \quad \text{geom} \quad \text{coordinate} \quad \text{system} \\
+ & = \quad + \quad = \quad =
\end{align*}

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and \textit{x} and \textit{y} locations.

\begin{align*}
\text{data} & \quad \text{geom} \quad \text{coordinate} \quad \text{system} \\
+ & = \quad + \quad = \quad =
\end{align*}

Complete the template below to build a graph.

\textbf{ggplot} (\texttt{data = \textit{Complete the template below to build a graph}})

\begin{align*}
\textbf{COORDINATE_FUNCTION} + \\
\textbf{GEOMETRIC_FUNCTION} (\texttt{mapping = aes(\ldots)}) + \\
\textbf{AESTHETIC_FUNCTION} + \\
\textbf{THEME_FUNCTION}
\end{align*}

\texttt{ggplot(data = \texttt{mpg}, \texttt{aes(x = cty, y = hwy))}} Begins a plot that you finish by adding layers to. Add one \textit{geom} function per layer.

\texttt{ggplot(data = \texttt{mpg}, \texttt{aes(x = cty, y = hwy)})} Creates a complete plot with given data, \textit{geom}, and mappings. Supplies many useful defaults.

\texttt{last_plot()} Returns the last plot

\texttt{ggsave("plot.png", width = 5, height = 5)} Saves last plot as 5' x 5' file named “plot.png” in working directory.

Matches file type to file extension.

Geoms

Use a \textit{geom} function to represent data points, use the \textit{geom}'s aesthetic properties to represent variables. Each function returns a layer.

\begin{tabular}{|c|c|c|c|}
\hline
\textbf{GRAPHICAL PRIMITIVES} & \textbf{a, b} & \textbf{c, d} & \textbf{e, f} \\
\hline
\textit{a} & \texttt{ggplot(economics, aes(date, unemploy))} & \texttt{ggplot(mpg, aes(cty, hwy))} & \texttt{ggplot(diamonds, aes(carat, price))} \\
\hline
\textit{b} & \texttt{ggplot(seals, aes(x = long, y = lat))} & \texttt{geom_col() \quad geom_violin()} & \texttt{geom_map(pg = \texttt{map}, \texttt{aes(fill = \	extit{murder}}))} \\
\hline
\hline
\textbf{ONE VARIABLE} & \textbf{continuous} & \textbf{continuous} & \textbf{continuous} \\
\hline
\textit{c} & \texttt{ggplot(mpg, aes(hwy))} & \texttt{ggplot(mpg, aes(class, hwy))} & \texttt{ggplot(diamonds, aes(carat, price))} \\
\hline
\textit{d} & \texttt{ggplot(mpg, aes(fl))} & \texttt{ggplot(diamonds, aes(carat, price))} & \texttt{geom_hex()} \\
\hline
\hline
\textbf{THREE VARIABLES} \\
\hline
\textit{g} & \texttt{geom_map(pg = \texttt{map}, \texttt{id = state}, \texttt{map = map})} & \texttt{geom_tile()} & \texttt{geom_tile()} \\
\hline
\hline
\textbf{LINE SEGMENTS} \\
\hline
\textit{b} & \texttt{geom_abline(\texttt{aes(intercept = 0, slope = 1))} & \texttt{geom_vline()} & \texttt{geom_vline()} \\
\hline
\textit{a} & \texttt{geom_hline(\texttt{aes(yintercept = lat))} & \texttt{geom_segment()} & \texttt{geom_segment()} \\
\hline
\textbf{INTERPOLATION} \\
\hline
\textit{c} & \texttt{geom_interpolate(pg = \texttt{map}, \texttt{aes(x = long, y = lat))} & \texttt{geom_interpolate()} & \texttt{geom_interpolate()} \\
\hline
\end{tabular}
Stats

A stat builds new variables to plot (e.g., count, prop).

Use `stat` syntax to map stat variables to aesthetics.

### GENERAL PURPOSE SCALES

Use with most aesthetics

- `scale_*_discrete()`: map discrete variables to visual ones

- `scale_*_identity()`: map cont'ed variables to visual ones

- `scale_*_manual()`: use manually chosen visual ones

#### COLOR AND FILL SCALES (DISCRETE)

- `n = d + stat_bar()`: basic bar chart

- `n + scale_fill_manual(values = c("myblue", "myyellow", "myred"))`: custom color scale

### COLOR AND FILL SCALES (CONTINUOUS)

- `o = c + geom_dotplot(aes(x = ..stat(count).., y = ..y..))`: dot plot

- `o + scale_fill_distiller(palette = "RColorBrewer\textregistered display.brewer.all\textregistered")`: RColorBrewer palette

### SHAPE AND SIZE SCALES

- `p = e + geom_point(aes(shape = ..prop.., size = ..prop..))`: shape and size

- `p + scale_shape_manual(values = c(1, 3, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45))`: shape scale

- `p + scale_size_manual(values = c(3, 7))`: size scale

### Coordinate Systems

- `r <- d + geom_bar()`: basic bar chart

- `r + coord_cartesian(xlim = c(0, 5))`: custom coordinate system

- `r + coord_fixed(ratio = 1/2)`: fixed aspect ratio

- `r + coord_flip()`: flip x and y axes

### Position Adjustments

Each position adjustment can be recast as a function with a different name.

- `s = ggplot(mpg, aes(fl, fill = drv)) + geom_bar(position = "fill")`: fill position

- `s + coord_cartesian(xlim = c(0, 50), ylim = c(0, 100))`: limits

- `s + facet_wrap(~ fl)`: facets

### Themes

- `r + theme_bw()`: black and white

- `r + theme_classic()`: classic theme

- `r + theme_minimal()`: minimal theme

### Faceting

- `t + facet_grid(. ~ fl)`: facets

- `t + facet_wrap(~ drv)`: wrap facets

### Labels

- `n + labs(x = "New x axis label", y = "New y axis label")`: labels

- `n + theme(legend.position = "bottom")`: legend position

### Legends

- `n + scale_fill_manual(name = "Title", values = c("A", "B", "C", "D"))`: legend

### Zooming

- `t + coord_cartesian(xlim = c(0, 100), ylim = c(0, 100))`: zoom

- `t + coord_fixed()`: zoom without clipping