John D. Lee

Interactive data visualization

To my son,

without whom I should have finished this book two years earlier

Contents

List of Tables

List of Figures

Preface

This book provides an introduction to ggplot2 for interactive data visualization. Its intent is to provide examples of common graphs and basic visualization principles.

Minard's plot shows the deaths of almost 300,000 troups as they march to Moscow demonstrates the horror of war, and is considered one of the best visualizations ever producced (**?**). Reasons why this graph is so effect is that it has a clear purpose, it answers important questions with a comples array of data that are presented in an understandable and aesthetically pleasing manner.

Why read this book

The aim of this book is help people make more graphs like Figure **??**). It links principles of graph design to examples that are implemented in R, particularly the popular graphic package ggplot2. The book provides a catalog of graphs and their design rationale organized around general questions that graphs are typically used to answer.

Structure of the book

Chapter **??** introduces R and the tidyverse functions and provides links for learning more about the basic capabilties of R. Chapters **??** - **??** each describe different types of graphs that answer questions regarding association, distribution, comparison, proportion, fluctionation, and connection. Chapter **??** briefly considers graphical elements in tables and Chapters **??** - **??** discuss interactive graphs and adjustments neeed for publication.

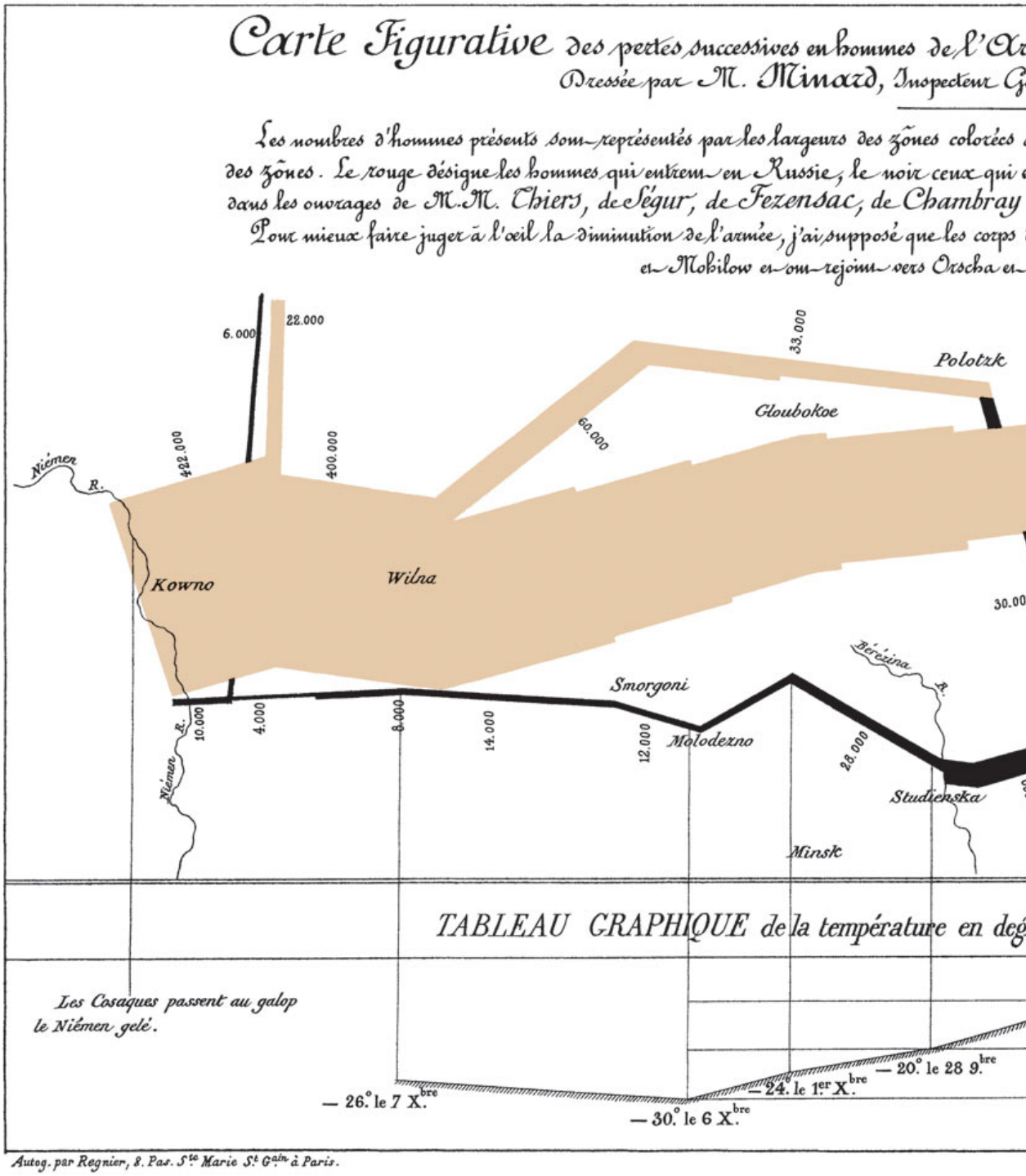


FIGURE 1: Minards visualization of Napoleon's disasterous march to Moscow

Software information and conventions

I used the **knitr** package (**?**) and the **bookdown** package (**?**) to compile the book. Most graphs have been created with **ggplot2** [@{Wickham2016a] and data manipulation is done with *dplyr*.

Acknowledgments

About the Author

John D. Lee is a professor in the Department of Industrial and Systems Engineering at the University of Wisconsin-Madison. He has investigated the issues of human-automation interaction, particularly trust inautomation, for over 20 years. More specifically, his research considers trust and acceptance, as well as issues of distraction and engagement. He helped to edit the Handbook of Cognitive Engineering, which focusses on human interaction with increasingly autonomous systems. He is also a co-author of a popular textbook: Designing for People: An introduction to human factors engineering (http://designing4people.com).

0

Introduction: Purposes, questions, and audiences

Information technology has brought large volumes of data that the promise of deeper understanding the challenges most central to our individual and collective existance.

Often this promise is not kept and data overwhelms rather than informs.

Well-crafted visualization can make data meaningful.

This book provides principles and examples for data visualization

Provides ways of addressing common challenges: remove legend (Section **??**) reorder categories of a bar chart or facets Creating Tufte inspired styles (Section **??**)

0.1 Seeing meaning rather than numbers

Figure **??** shows a typical report of a medical diagnostic test. The numerical summary shows the patient's values and the range of standard values. The information is there to show if a patient is dangerously outside the range, but a quick glance at the table might miss these indications. Even a careful reading of the table might miss warning signs, particularly if the critical information is in the trend that requires looking at a second table on another tab. Figure **??** shows the data relative to the high and low normal range and makes deviations much more apparent. The ghosted points show past results and roughly indicate trends.

```
## Warning: package 'bindrcpp' was built under R version
## 3.4.4
```

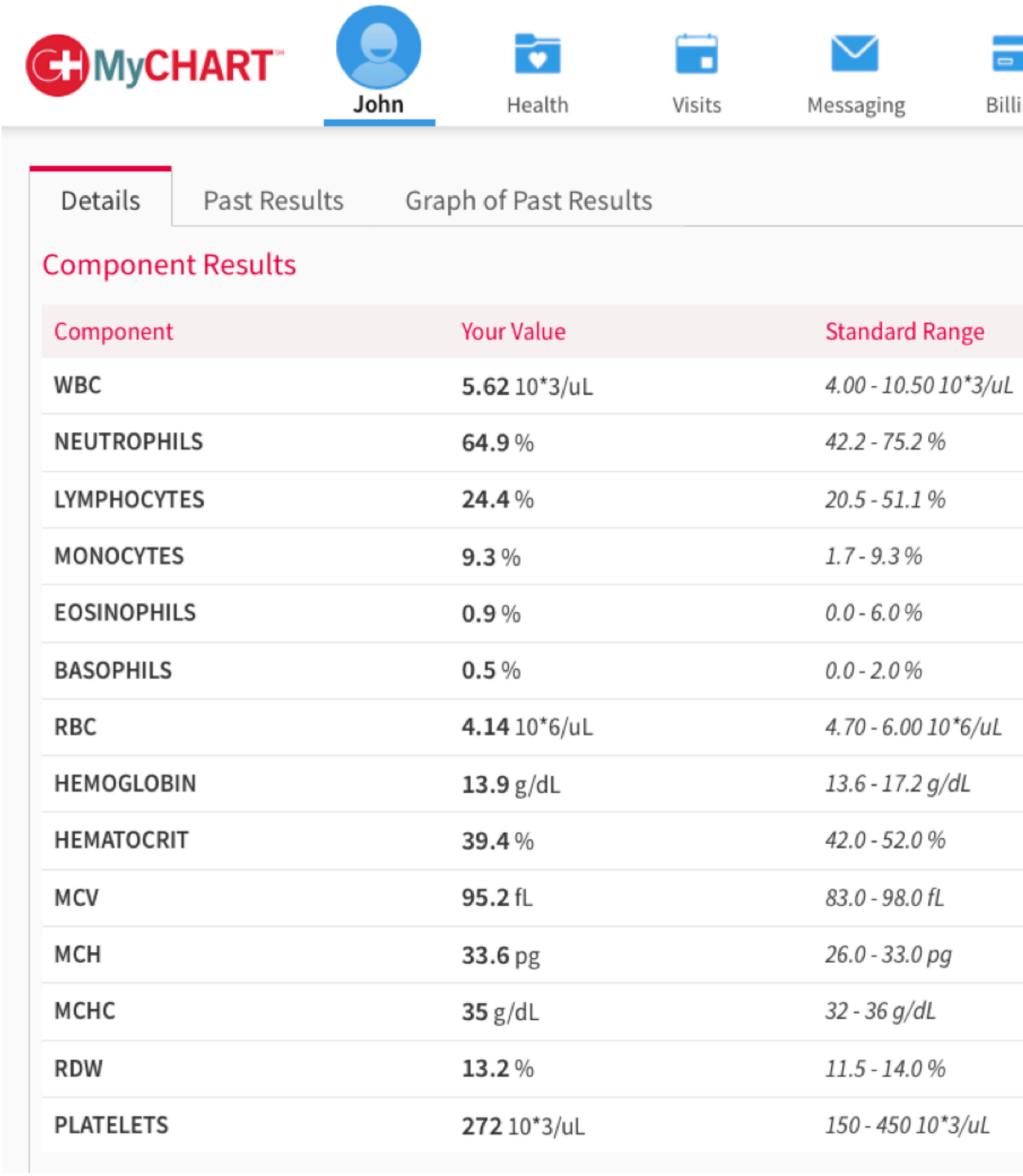


FIGURE 2: A typical report of a medical test makes finding deviations from the normal range difficult.

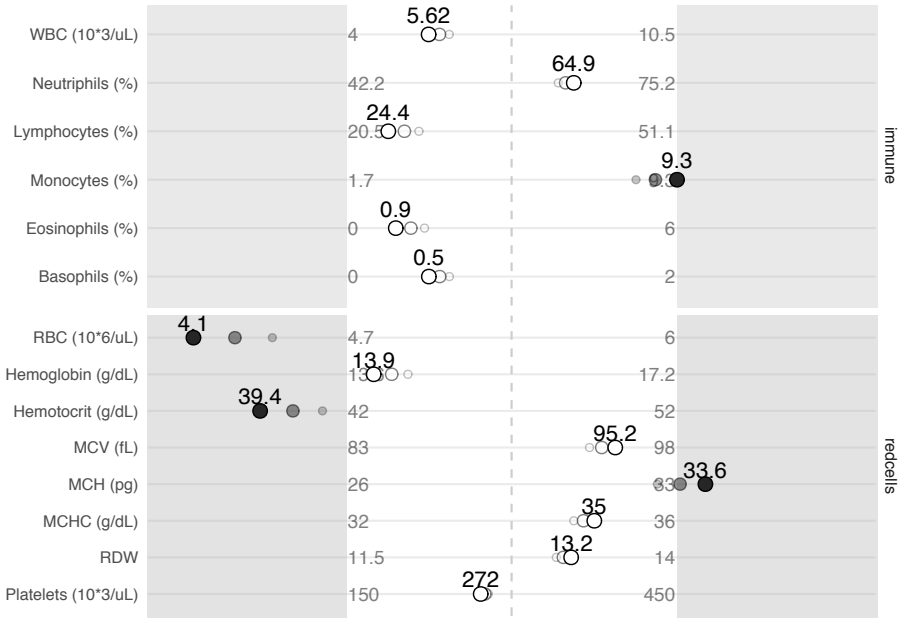


FIGURE 3: A visualization of the same results makes the deviations pop out.

0.2 Seeing more than summary statistics

The easy availability of sophisticated machine learning and statistical models makes algorithmic interpretation of data tempting. However, such interpretations can mislead, with similar outcomes produced by very different underlying data.

Figure **??** shows four distinct sets of data. The differences are obvious when graphed. One might expect that the typical summary statistics–mean, standard deviation, and correlation–would show equally stark differences. Table **??** shows this is not the case. Each data set has the same summary statistics.

0.3 Purpose and audience of visualizations

Explore, inform, and engage (**?**)

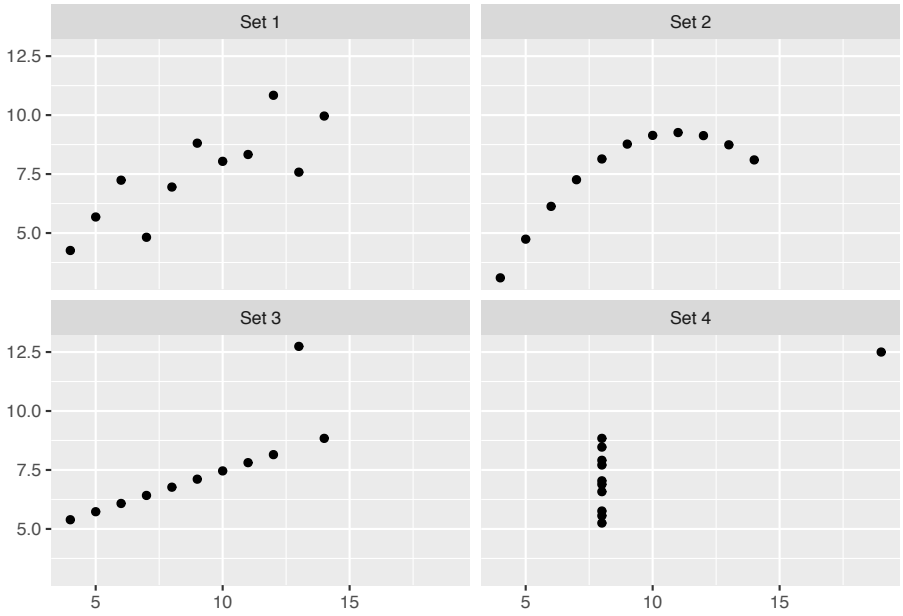


FIGURE 4: The Anscombe quartet and the limits of summary statistics.

TABLE 0.1: Four seemingly identical datasets that illustrate the limits of algorithmic interpretation.

Data set	Mean	Standard deviation	Correlation
Set 1	7.5	2.03	0.82
Set 2	7.5	2.03	0.82
Set 3	7.5	2.03	0.82
Set 4	7.5	2.03	0.82

Target audience: yourself, peers, scientists and engineers, public (NYTimes ref)

Explore: the answer is unknown and audience is likely yourself and peers involved in the research

Inform: the answer is known and the audience is likely a broader audience of scientists, engineers, or manager not direclty involved in the research.

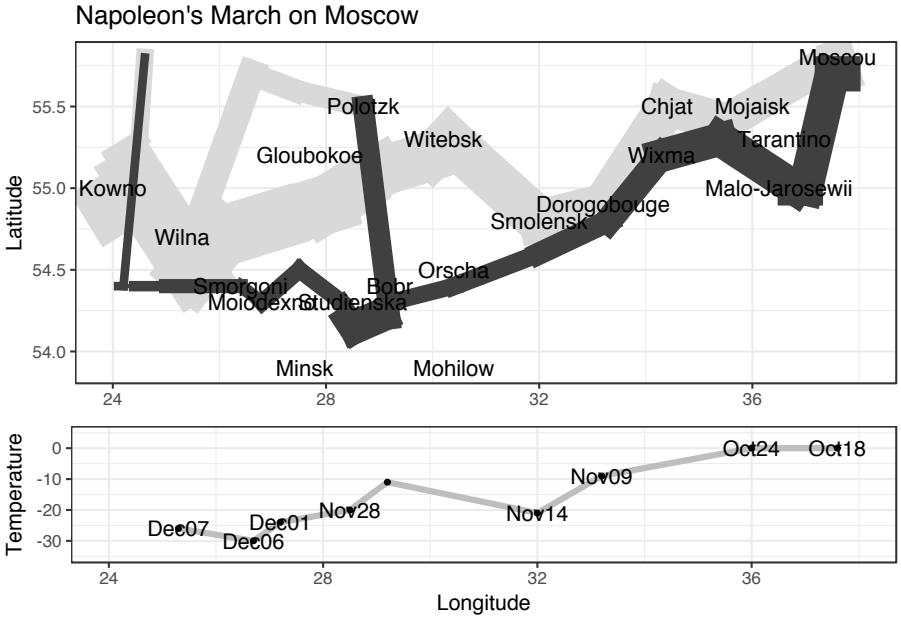
Engage: the answer is known and must be communicated in an entertaining way to those who may might neet to be drawn into readin the graph and may not be familiar with conventions of scientific visualization, such as box plots.

0.4 What question to answer?

0.5 Storytelling with graphics

- High-level principles for communication, such as "Show don't tell"
- Role of annotation in going beyond the data: direct attention and explain, as in Table **??**.

Based on https://www.rdocumentation.org/packages/HistData/versions/0.8-4/topics/Minard



0.6 Data

This book is not about data reduction and data wrangling. The tidyverse provides an intrgrated set of tools for data wrangling http://r4ds.had.co.nz. This book uses data from the following sources:

https://www.kaggle.com

https://www.data.gov Consumer complaint database, NTSB accident database

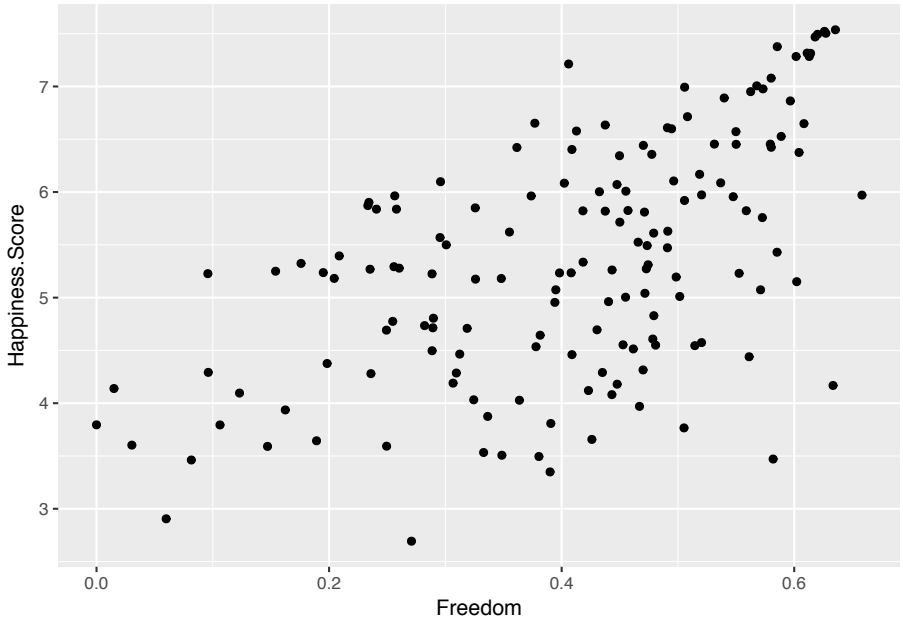
https://flowingdata.com/category/projects/data-underload/

http://www.wolframalpha.com

R packages: HistData, babynames

Yao about data and web scraping and the package

```
## Read data from website
# sports <- read_tsv("https://github.com/halhen/viz-pub/raw/master/sports-time-of-day/acti


## Happiness
#
happiness.df = read.csv("data/world-happiness-report/2017.csv")
ggplot(happiness.df, aes(Freedom, Happiness.Score)) + geom_point()
```



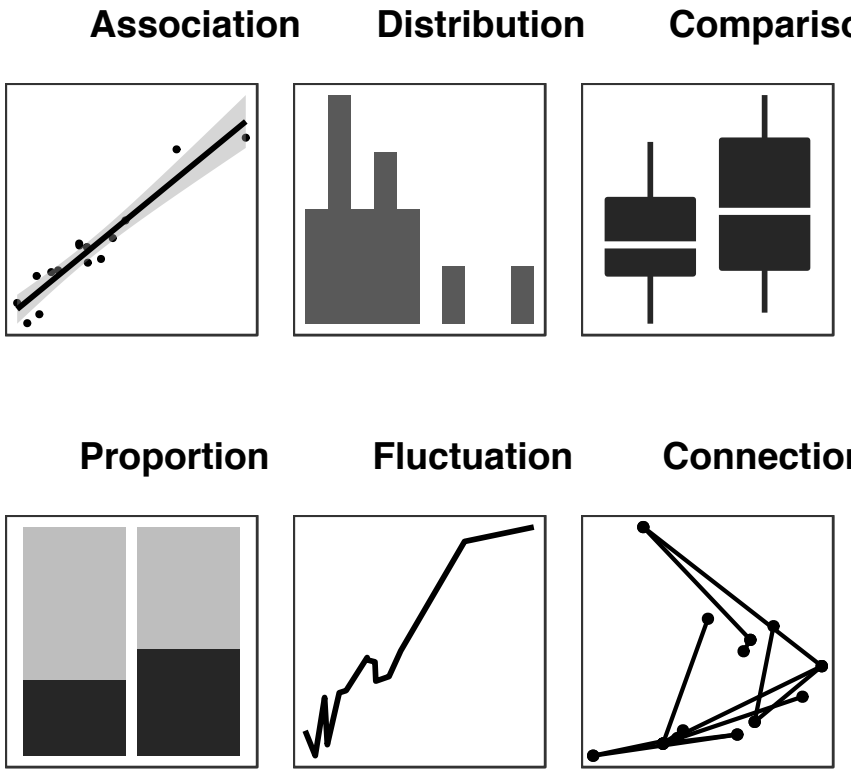

```
## Chocolate
# chocolate.df = read.csv("flavors_of_cacao.csv")
# chocolate.df$Cocoa.Percent = as.numeric(chocolate.df$Cocoa.Percent)
# ggplot(chocolate.df, aes(Cocoa.Percent, Rating)) + geom_point()
```

```
## Police
## 2535 observations, Age, gender, how armed, state, threat, body cameraAll factors
 police.df = read.csv("data/PoliceKillingsUS.csv")

# Canadian vehicle specifications: http://www.carsp.ca/research/resources/safety-sources/c
```

0

Visualization types and principles

```
## Warning: package 'igraph' was built under R version
## 3.4.4
```



0.7 Pairing questions and graph types

Graphs answer questions about data by showing relationships and making comparisons easier. Before creating a graph it is critical to specify the ques-

tions and comparisons of interest. Figure **??** shows common graphs and general questions they might answer. For example, in the upper left is a graph that shows the association between variables. This type of graph answers questions such as "how does X influence Y?", as in "does increasing the prices of gas reduce the amount of driving?". A scatter plot shows the strength and nature of this association. Each graph in Figure **??** is suited to a different question:

Graphs answer questions about data by showing relationships and making comparisons easier. Before creating a graph it is critical to specify the questions and comparisons of interest. Table **??** shows common graphs and general questions they might answer. For example, in the upper left is a graph that shows the association between variables. This type of graph answers questions such as how does X influence Y, as in "does increasing the prices of gas reduce the amount of driving?". A scatter plot shows the strength and nature of this association.

- Association: What influences an outcome?
- Distribution: What is the spread of the observations?
- Comparision: How does one condition differ from another?
- Proportion: What is the size of the components that make up the whole?
- Fluctuation: How does the do observations vary over time?
- Connection: How are the observations connnected over a map or network?

Combinations of questions, such as changes in distribution or proportion over time

Questions in terms of patterns vs precision

Graph type and familiarity, pie charts Scatter plot to 2-density, comparison to ranking, dotplot to violin or boxplot.

Graph types and volume of data.
More data requires abstraction. Some plots scale well others do not, overplotting one example of scaling challenges with increasingly large data.

More data points and more variables (e.g., time sequences, categories), organize chapters to move from few points and few variables to many (e.g., histogram to small multiple, to heatmap)

Types of data sets: Number of observations (independent, sequential) Number variables (nominal, ordinal, interval)

~50 observations and 5 nominal and 7 interval variables (mtcars, IIHS vehicle fatalities) ~50 observations and 1 nominal and 4 interval variables (iris) ~200 observations and 2 nominal and interval variables (10) (belts) ~50,000 observations an 10 nominal and interval variables (diamonds)

The examples for each type of graphs represent one of many possible representations. For example, the stacked bar chart addresses questions of proportion, but so can pie charts and 3-D pie charts. How do you choose between these alternatives? One consideration is to select display dimensions that make it

easy for people to make comparisons needed to answer the questions—identify effective mapping between data and display dimensions—which we turn to in the following section.

0.8 Percpetual processes to be supported: Comparison, Detection, Pattern identification

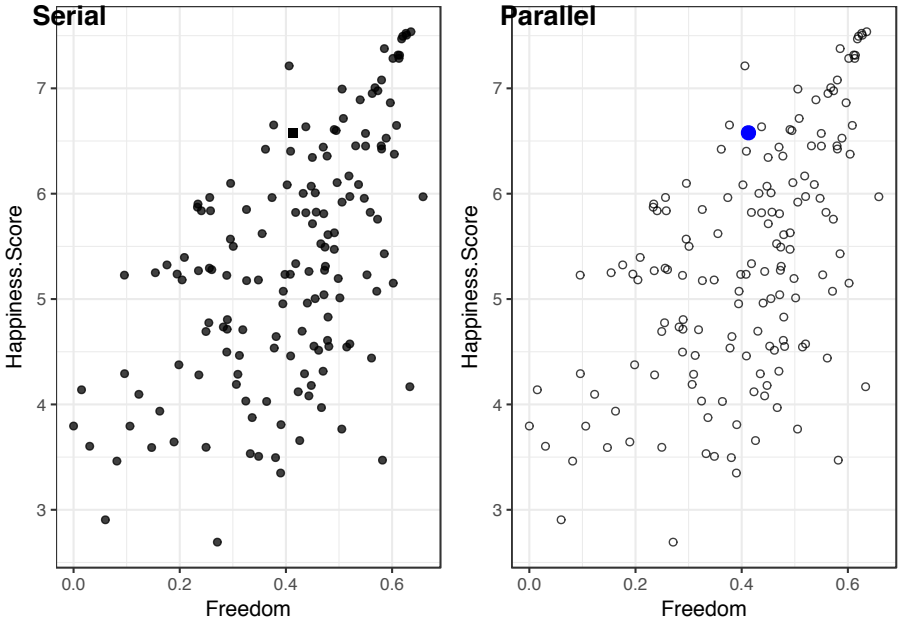
Attentional span Visual WM limits Preattentive cues Compatability Conventions and familiarity

0.8.1 Comparison

Differences between conditions, Compare to zero? Perceptual sensitivity Proximity compatibility principle (enable relative rather than absolute comparisons with reference lines and data ordering)

0.8.2 Detection

Outliers, deviations from assumptions Popout effects TODO Create figure to show cost of conjunctive search and benefit of redundant coding



0.8.3 Pattern identification

Associations, interactions, and changes over time Gestalt principles

Preattentive processing

TODO Create figure to show relative benefit of shape, intensity color for grouping

Grouping and gestalt Similarity Continuity Connection Proximity Enclosure Closure

TODO Figure ground showing data and summary vs summary and data

0.9 Principles from general to specific

(**?**), (**?**, **?**)

%ten guidelines (**?**)

0.9.1 Identify audience, story, and key relationships (Few)

0.9.2 Focus attention and organize reading

Group Prioritize Provide context Sequence

Be consistent, every difference should tell

0.9.3 Annotate to show cause and explain why

0.9.4 Concrete details engage and are memorable

Connect to the world

0.9.5 Enable comparisons and put data in context

(Tufte) Scatter plot: Data points with linear and loess models Category plot: Boxplot with individual data points Time series: Small multiples with grand mean

Estimation errors and effort proportional to the absolute difference from common baseline: reference lines provide a local baseline. TODO Show tall bars with mean reference line

0.9.6 Map types of variables to graph features

Mapping data to graph features (**?**)

For the purposes of display design, three different data types guide the choice of display dimensions: interval, ordinal, and nominal (**?**). Interval data include real or integer numbers (e.g., height and weight), ordinal data are categories that have a meaningful order (e.g., compact, mid-size, and full-size cars), and nominal data are categories that have no order (e.g., male, female). Each data type can be represented with one of several graph dimensions, such as color or position, but certain mapping support more accurate judgments.

Size of circle: map to radius or the area TODO create plot to show map to radius and area

TODO show good and bad mappings

color (**?**)

```
data = read.csv('data/DataAestheticMapping.csv')
```

```
data$Type = factor(data$Type, levels=c("Interval", "Ordinal", "Nominal"))

ggplot(data, aes(Type, reorder(Rank, -Rank), group = Aesthetic)) +
  geom_line(alpha = .4, size = 2.5) +
  geom_text(aes(label = Aesthetic), size = 5) +
  ylab("Rank") + xlab("Data type") +
  theme_bw()
```

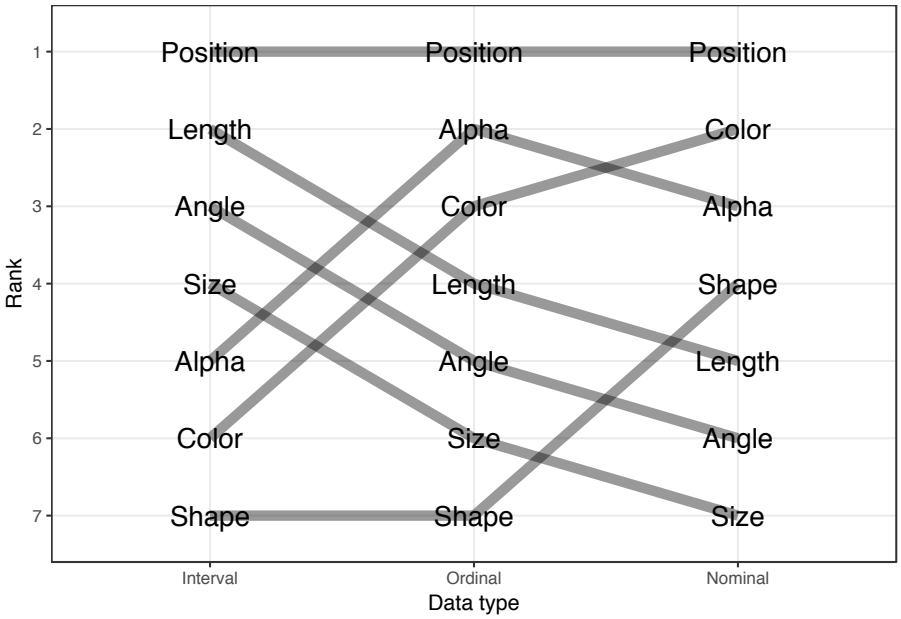


FIGURE 5: Aesthetic mapping.

%TODO figure for mapping types of data and graph dimensions (**?**)

Figure XX shows seven ways to code these data (**?**). For all three types of data, position, such as the horizontal or vertical placement of a point in a graph, support the most precise judgments. The other ways of coding information depend on the type of data: hue is a poor choice for interval data, but a good choice for nominal data, as is shape. Because shape and color have no natural mapping to magnitude, they are a poor choice for interval and ordinal data. Magnitude is best represented by position on a common scale, followed by position on an unaligned scale, length and then angle, followed by size (**?**, **?**). Because size and angles are relatively hard to judge, pie charts are not a good way to represent proportions.

Limits of absolute judgment underlie the effectiveness of coding data with various display dimensions. Coding nominal data with more than seven hues

will exceed people's ability and so they would not be able to reliably link lines on a graph to categories. Data presented on aligned scales, such as the bottom category in a stacked bar chart, can be judged very precisely, but the limits of absolute judgment make interpreting the upper categories more difficult. This means that the bottom category of a stacked bar chart should be chosen carefully. Generally, avoid placing data on unaligned scales. Instead, support relative judgments based on a common scale. The circular format of pie charts means that there are no aligned scales and is another reason why they are not as effective as stacked bar charts.

Because visualization involves multiple conceptual dimensions, a natural choice is to use three-dimensional Euclidian space. However, three-dimensional figures make accurate comparisons difficult due the ambiguity of rendering three dimensions on a two dimensional plane. Of all the ways to represent a quantity, the volume of a three-dimensional object leads to the most inaccurate judgments (**?**).

Another important conceptual dimension is time. Time, like space, is compatibly mapped to display dimension of position, often advancing from left (past) to right (future). Time can also be directly mapped to display time via animation. Animated graphs can be compelling, but they require working memory to track objects across the display and so severely limit the number of data points that can be compared. Interactive visualization described in Chapter 10 can give control with a slider and avoids this limit to some degree.

0.9.7 Ensure proximity compatibility

Proximity compatibility and legend: link to line, orientate to match orientation in graph, sequence to match sequence in graph

Visual attention must sometimes do a lot of work, traveling from place to place on the graph, and this effort can hinder graph interpretation. Hence, it is important to construct graphs so things that need to be compared (or integrated) are either close together in space or can be easily linked perceptually by a common visual code. This, of course, is a feature for the proximity compatibility principle (A3) and can apply to keeping legends close to the lines that they identify, rather than in remote captions or boxes. Similarly, when the slopes and intercepts of two lines need to be compared, keep them on the same panel of a graph rather than on separate panels. The problems of low proximity will be magnified as the graphs contain more information—more lines. Similarly, in a box plot with many categories people will be able to compare categories that are close to each other more precisely than those that are separated. You should order categories so that those to be compared are closest.

Proximity goes beyond physical distance. A line linking points on a timeline

can enhance proximity as can color and shape. Lines and color can be effective ways of making groups of points in a network diagram "closer", and easier to interpret as a group. Objects with identical colors tend to be associated together, even when they are spatially separated. Furthermore a unique color tends to stand out. It is also the case that space is compatibly mapped to space, so that visualization of geographic areas is best accomplished when the dimensions of rendered space correspond to the dimensions of displayed space–a map.

As with its application to other display designs, the proximity compatibility principle means that the visual proximity of elements of the graph need to correspond to the mental proximity needed to interpret this information. For graphs, this means the questions and comparisons the graph is intended to address should specify what is "close" in the graph.

0.9.8 Legibility and consistency

As with other types of displays, issues of legibility are again relevant. However, in addition to making lines and labels large enough to be readable, a second critical point relates to discriminability (P9). Too often, lines that have very different meanings are distinguished only by points that are highly confusable, as in the graph on the left of Figure XX. Here incorporating redundant coding of differences can be quite helpful. In modern graphics packages, color is often used to discriminate lines, but it is essential to use color coding redundantly with another salient cue. Why? As we noted in Chapter 4, not all viewers have good color vision, and a non-redundant colored graph printed from a black and white printer or photocopied may be useless.

0.9.9 Maximize data/ink ratio

(Tufte) * Maximize data to create rich representation, minimize extraneous non-data elements

- Minimize non-data elements: bar charts rather than 3-D pie

Annotate to integrate interpretation and data

Simplify to amplify content

Simplify content to amplify point

0.9.10 Manage clutter with grouping and layering

- Match data type to appropriate aesthetics (Cleveland) Only position good for all data types: Focus on 2d-plane and relative judgments Consider data type: size better than color for interval data

Graphs can easily become cluttered by presenting more lines and marks than the actual information they convey. As we know, clutter can be counterproductive , and this has led some to argue that the data-ink ratio should always be maximized (**?**); that is, the greatest amount of data should be presented with the smallest amount of ink. While adhering to this guideline is a valuable safeguard against the excessive ink of "chart junk" graphs, such as those that unnecessarily put a 2-D graph into 3-D perspective, the guideline of minimizing ink can however be counterproductive if carried too far. Thus, for example, the "minimalist" graph in center of Figure X, which maximizes data-ink ratio, gains little by its decluttering and loses a lot in its representation of the trend, compared to the line graph on the right Figure XX. The line graph contains an emergent feature—slope—which is not visible in the dot graph. The latter is also much more vulnerable to the conditions of poor viewing (or the misinterpretation caused by the dead bug on the page!).

%Figure 21. Space shuttle launches, temperature and O-ring damage. %TODO I love the example, but A more elaborated caption is needed to direct reader's attention to the problem.

In some cases, the poor data to ink ratio and prevalence of chart junk might create engaging graphics, other times it can be annoying, but in presenting engineering data it can undermine the quality of life and death decisions. Figure 20 shows the graphic used to support the launch decision associated with the disastrous flight of the Space Shuttle Challenger the data presented in this way makes it difficult to assess the effect of temperature on O-ring damage, which may have encouraged the managers to launch in cold weather (**?**).

Figure X shows that you can increase the data-to-ink ratio by reducing the "ink" devoted to non-data elements. Another way to increase the data-to-ink ratio is to include more data. More data can take the form of reference lines and multiple small graphs, as in Figure XX. More data can also take the form of directly plotting the raw data rather than summary data.

Figure **??** shows an extreme version, in which each data point represents one of approximately 693,000 trips reported in the 2009 travel survey XXcite FHWA2011. The horizontal axis indicates the duration and the vertical axis shows distance of each trip. The diagonal lines of constant speed place these data in context by showing very slow trips—those under the 3mph line—and very fast trips—those over the 90mph line. Histograms at the top and side show the distribution of trip duration and distance. The faint vertical and horizontal lines show the mean duration and distance. Like other visualizations

that include the raw data, this visualization shows what is behind the summary statistics, such as mean trip distance and duration.

Showing the underlying data has the benefit of providing a more complete representation, but it can also overwhelm people. Data can create clutter. One way to minimize clutter is by grouping and layering the data. In the case of Figure **??** this means making the individual data points small and faint.

0.10 Overview of examples

Simple, few variables, few observations and single graphical element to and complex, many observations to combitions of graphical elements

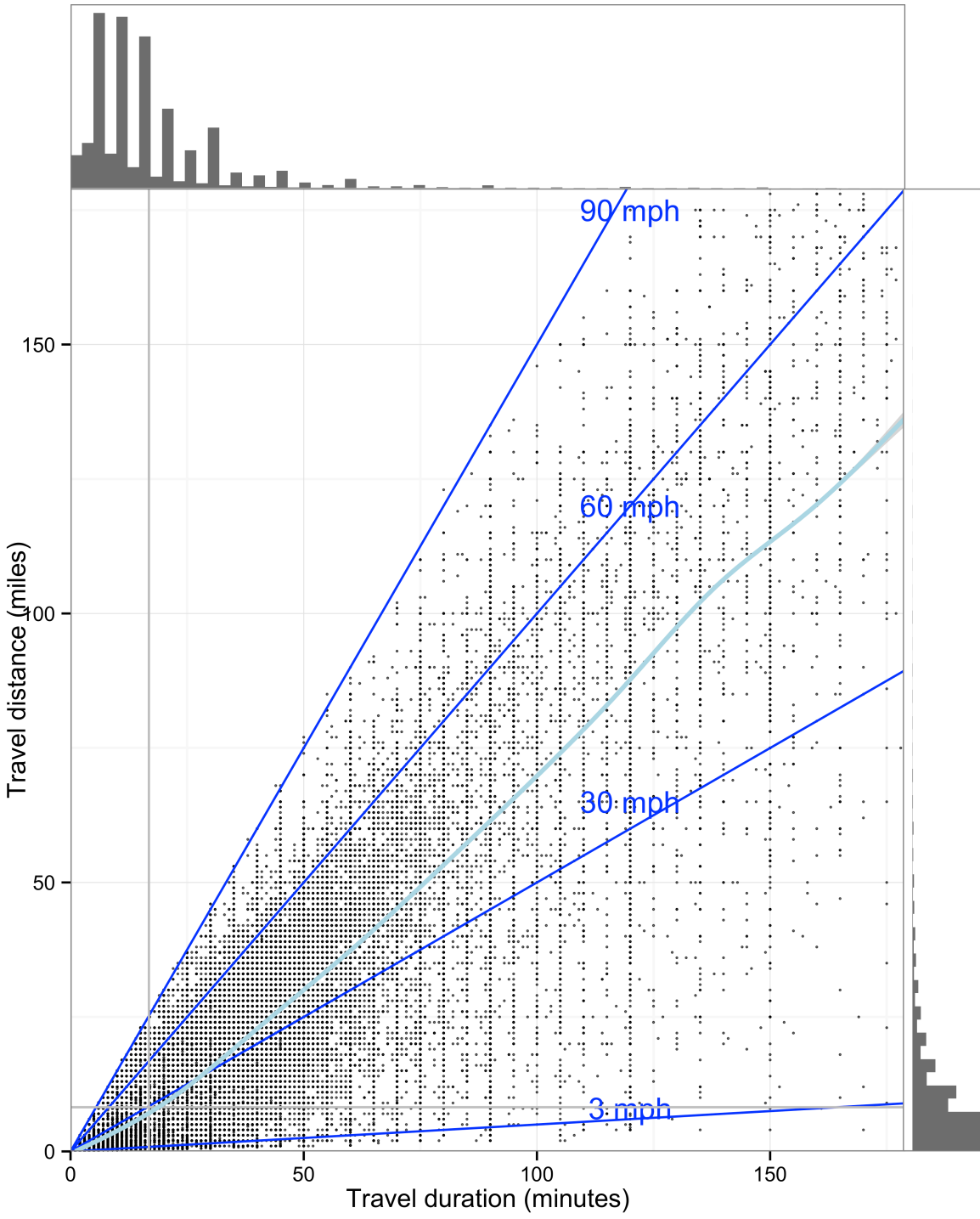


FIGURE 6: An example of extreme data-to-ink with over 693,000 data points

0

Association–scatterplots

0.11 Basic elements of the grammar of graphics

TABLE 0.2: Summary of ggplot element.

ggplot element	Description
Data	ggplot uses a dataframe as input (e.g., data = mtcars.data)
Geoms	geometric element (e.g., geom_point, geom_bar)
Mapping	links data variables to aesthetic dimensions (aes) (e.g., aes(x = cyl, y = mpg))
Setting	specifies value of aesthetic dimension directly (e.g., colour = "blue")
Layers	add components to base plot, most often geoms (additional layers added with "+")
Stats	statistical summary, such as density or count; each geom has a default statistic
Position	adjusts the location of the plotted geom
Annotations	text and graphical overlays
Coordinate system	Cartesian, polar or small multiple facets
Themes	sets of plot parameters (e.g., font, background)

0.12 Simple scatterplot

The simplest plot must include data, aesthetic mapping, and geom a geometric element. The data must be organized with each observation as a row and each variable as a column. For a scatterplot the geometric element is a point, and the aesthetic mapping links variables to properties of the geometric element. For a scatterplot this would be the x and y position of the point. the the point the x and y position must be specified, but other properties, such as color, size, alpha level, and shape, can be mapped to variables. These properties of the geometric element can also be set to specific values, such as specifying the color of the point.

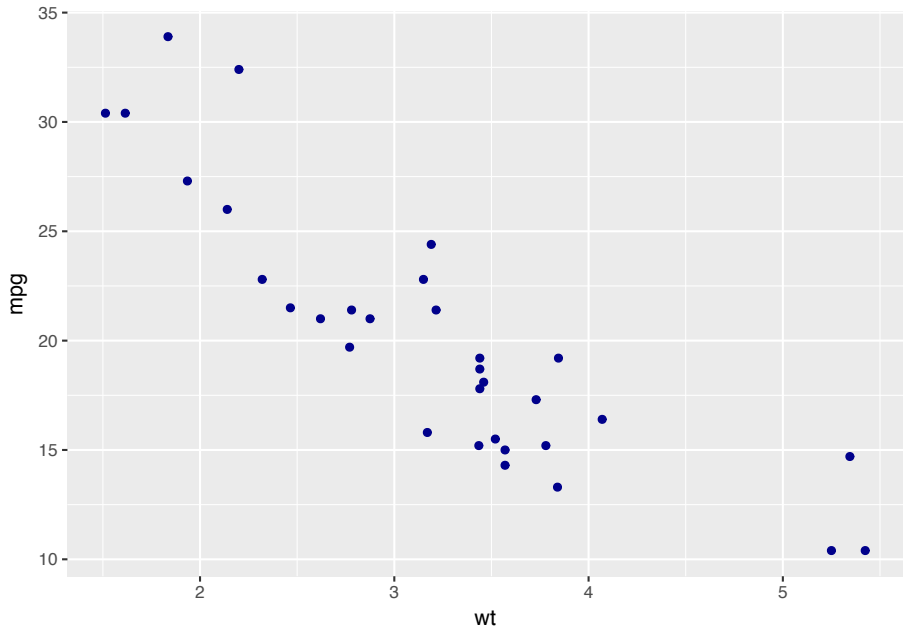
Figure **??** shows the ggplot2 code and associated scatterplot. The equation used to specify the plot implicitly specifies the values by their position, such

as data being identifed as following "ggplot(". The following speciifcations are equivalent:

" ggplot(data = mtcars.df, mapping = aes(x = wt, y = mpg)) + geom_point(colour = "darkblue")

ggplot(mtcars.df, aes(wt, mpg)) + geom_point(colour = "darkblue") "

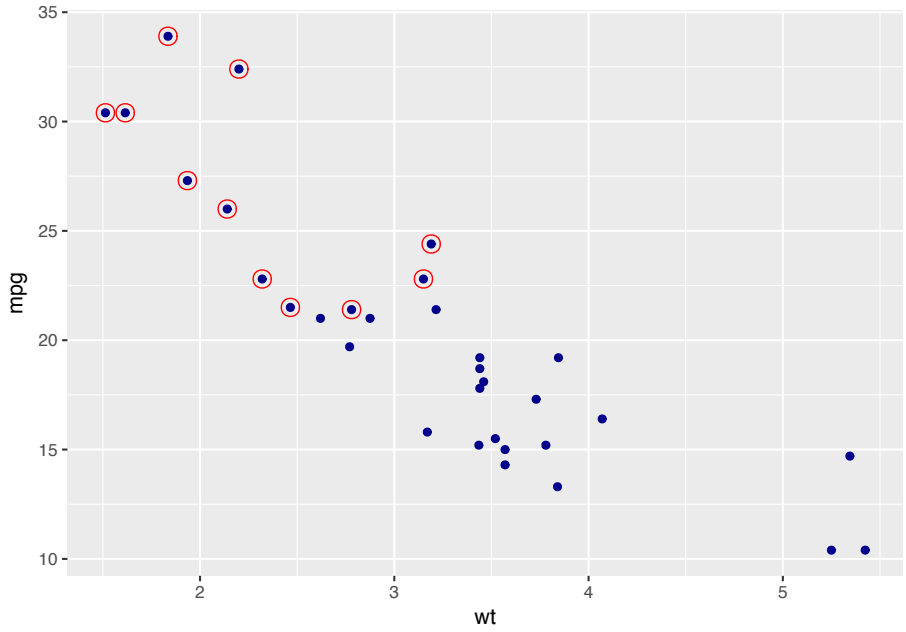
```
library(tidyverse)

mtcars.df = mtcars

ggplot(data = mtcars.df, mapping = aes(x = wt, y = mpg)) +
  geom_point(colour = "darkblue")
```



A powerful feature of ggplot2 is the abilty to add layers of geometric elements to a plot. Each layer can have its own data, mapping of aesthetic properties, and setting of aesthetic properties. The data and mapping specified in the base plot statement–"ggplot(data = mtcars.df, mapping = aes(x = wt, y = mpg))"–are global and apply to all layers, but can overridden by the any mappings specific to a layer. Figure **??** shows a layer of red circles based on a subset of the data.

```
fourcyl.mtcars.df = mtcars.df %>% filter(cyl==4)
```

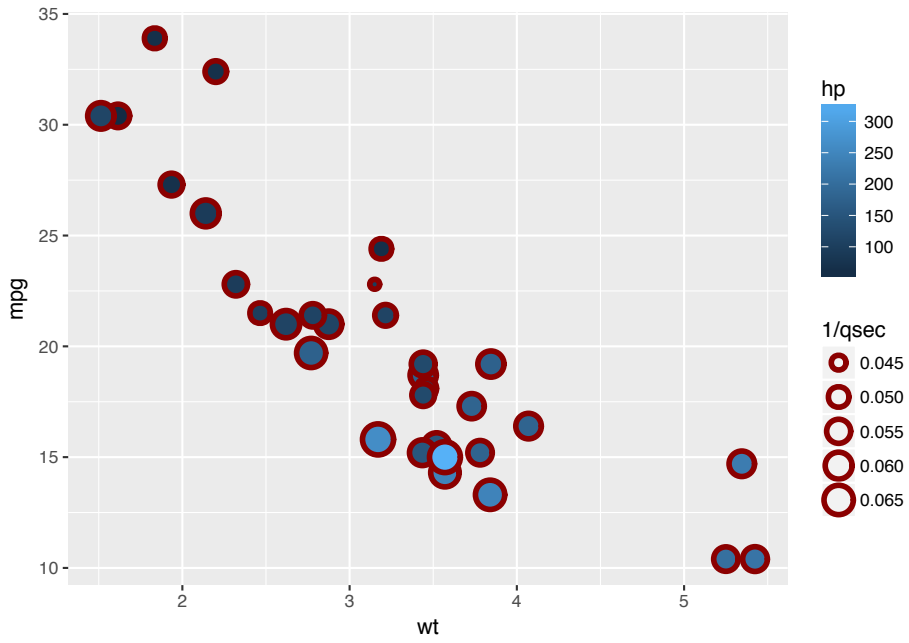
```
ggplot(data = mtcars.df, mapping = aes(x = wt, y = mpg)) +
  geom_point(colour = "darkblue") +
  geom_point(data = fourcyl.mtcars.df, colour = "red", shape = 21, size = 4)
```



0.13 Scatterplot with additional mappings

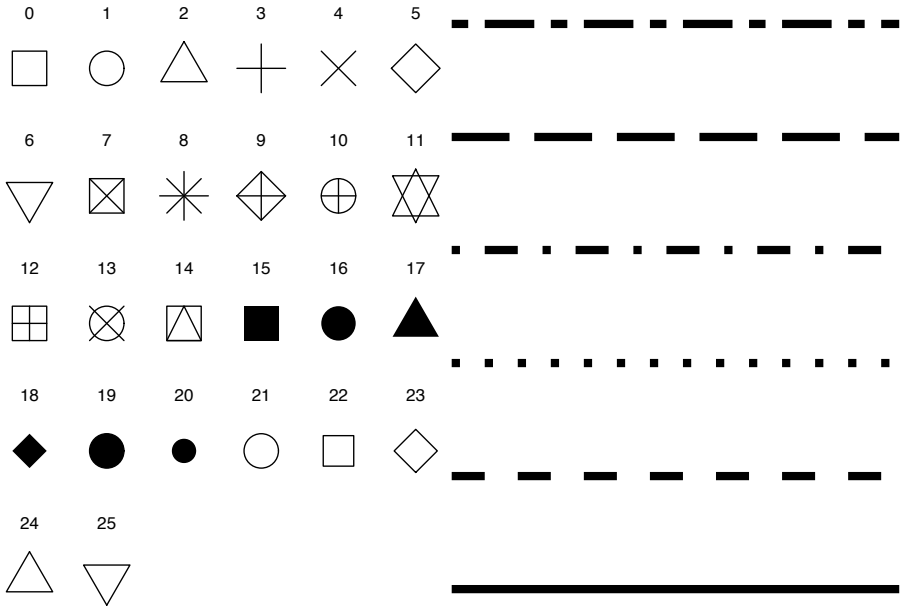
The scatterplot typically maps variables to the x and y position of the points, but ggplot allows for other mappings. Figure **??** shows mapping variables to the fill and size of the points. The shape, stroke, and color of the points are set to values they could also be mapped, which could quickly overload the graph. Note that only shapes 21-25 in Figure **??** can include fill and stroke, with the other symbols color determines the color of the whole symbol not just the border.

```
ggplot(data = mtcars, mapping = aes(x = wt, y = mpg, fill = hp, size = 1/qsec)) +
  geom_point(shape = 21, colour = "darkred", stroke = 2.)
```




```
## Warning in align_plots(plotlist = plots, align = align,
## axis = axis): Complex graphs cannot be vertically
## aligned unless axis parameter is set properly. Placing
## graphs unaligned.
```

```
## Warning in align_plots(plotlist = plots, align = align,
## axis = axis): Graphs cannot be horizontally aligned,
## unless axis parameter set. Placing graphs unaligned.
```



0.14 Scatterplot with linear and loess fit

The layers can include geometric elements beyond geom_point. Perhapts the most useful geoms to add to a scatterplot is a curve fit. Figure **??** shows a simple scatterplot with two cruve fits. The loess fit shows a smooth fit that indicates non-linar trends, and the blue line shows a linear regeression. The loess line highligths areas in the data that deviate from a linear relationship shown by the blue line. All three layers inherit the same x and y mapping from the ggplot base layer.

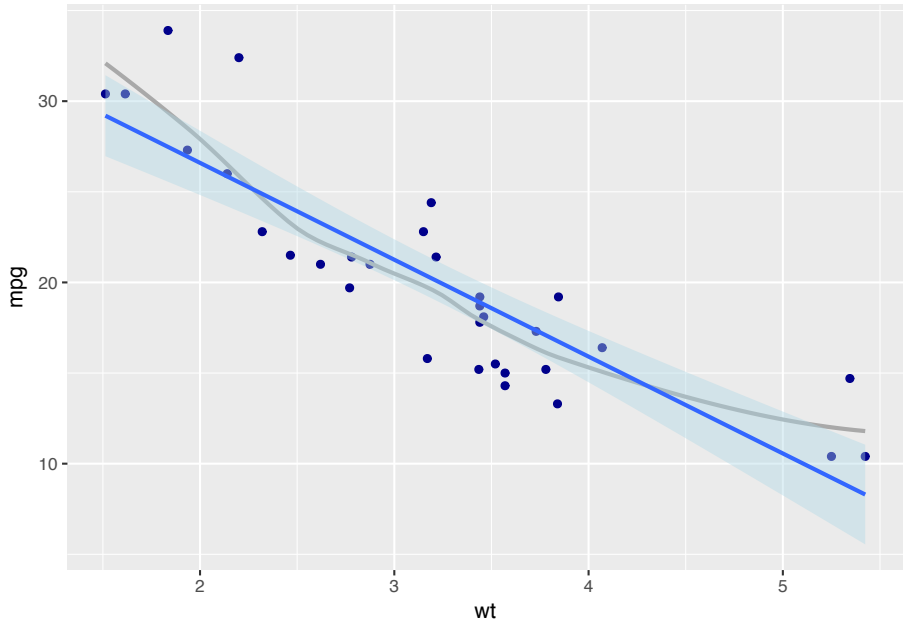
When building a plot each layer is placed on top of the preceding layer, such that the last layer lies on top of all the others. With Figure **??**, the points are on the bottom and the light blue line is on top of the gray loess line.

Table **??** shows the full set of possible geometric elements that can be used to create graphs, the following chapters describe many of these.

Note the smooth fit geoms include additional settings for the method and whether the line should include a standard error.

```
ggplot(data = mtcars.df, aes(x = wt, y = mpg)) +
  geom_point(colour = "darkblue") +
```

```
  geom_smooth(method = "loess", se = FALSE, colour = "darkgrey") +
  geom_smooth(method = "lm", fill = "lightblue")
```

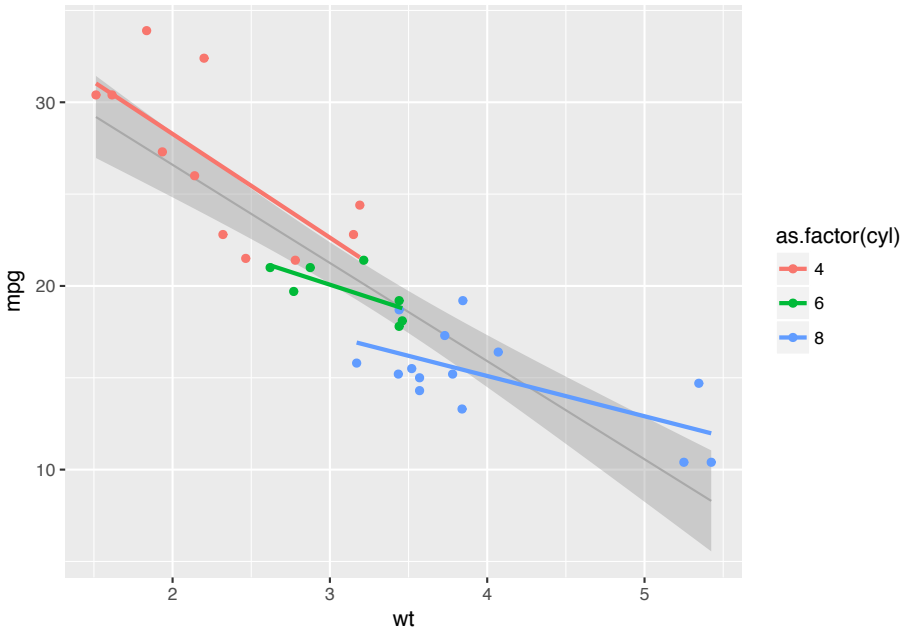



```
## Warning: package 'kableExtra' was built under R version
## 3.4.4
```

x
geom_abline
geom_area
geom_bar
geom_bin2d
geom_blank
geom_boxplot
geom_col
geom_contour
geom_count
geom_crossbar
geom_curve
geom_density
geom_density_2d
geom_density2d
geom_dotplot
geom_errorbar
geom_errorbarh
geom_freqpoly
geom_hex
geom_histogram
geom_hline
geom_jitter
geom_label
geom_line
geom_linerange
geom_map
geom_path
geom_point
geom_pointrange
geom_polygon
geom_qq
geom_quantile
geom_raster
geom_rect
geom_ribbon
geom_rug
geom_segment
geom_smooth
geom_spoke
geom_step
geom_text
geom_tile
geom_violin
geom_vline

0.15 Global and local regression

```
ggplot(data = mtcars.df, aes(x = wt, y = mpg)) +
    geom_smooth(method = lm, colour = "darkgrey", size = .5) +
  geom_point(aes(colour = as.factor(cyl))) +
  geom_smooth(aes(colour = as.factor(cyl)), method = lm, se = FALSE)
```



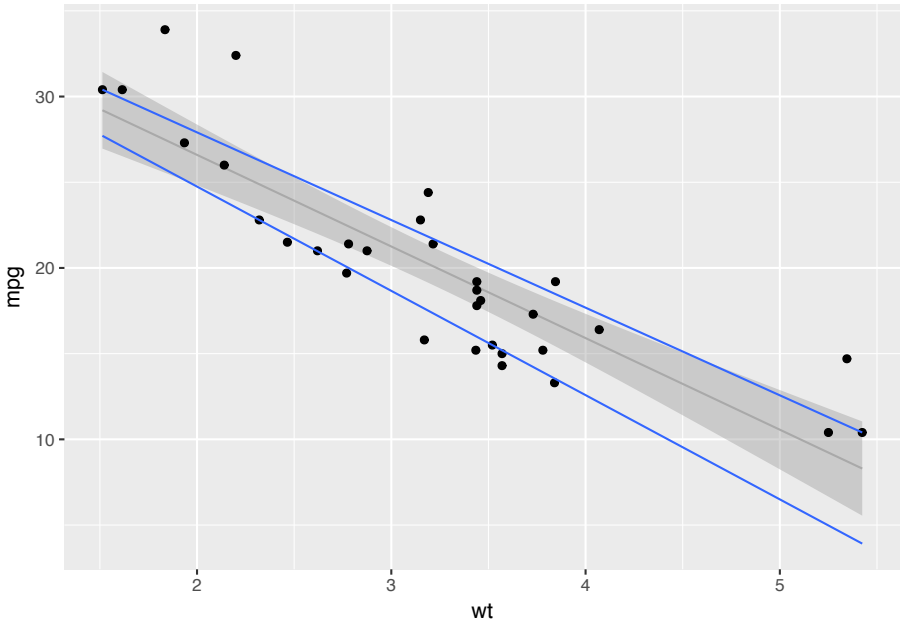
0.16 Quantile regression and other functional relationships

Often the question the graph is meant to answer is not about the central tendency, but about the likelihood of relatively extreme values, such as the 25th and 75th percentiles.

```
ggplot(data = mtcars.df, aes(x = wt, y = mpg)) +
    geom_smooth(method = lm, colour = "darkgrey", size = .5) +
```

```
  geom_point() +
  geom_quantile(quantiles = c(.25, .75))
```

```
## Loading required package: SparseM
##
## Attaching package: 'SparseM'
## The following object is masked from 'package:base':
##
##     backsolve
## Smoothing formula not specified. Using: y ~ x
```



0.17 Scatterplot with regression equation and marginal distributions

Scatterplot augmented with marginal distributions, regression equation, and Tufte-inspired range frame.

Marginal distributions show that a 1-D scatterplot is a histogram and that a 2-d histogram is a scatterplot. Chapter **??** describes such plots in detail.

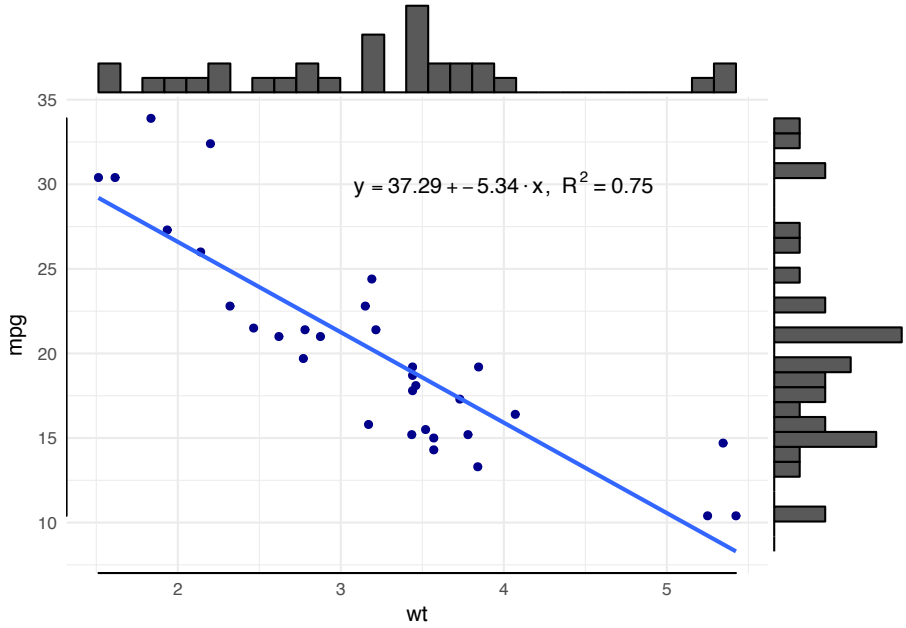
Chapter **??** shows how to add annotations, such as the equation.

Derived from: http://t-redactyl.io/blog/2016/05/creating-plots-in-r-using-ggplot2-part-11-linear-regression-plots.html

```
library(ggthemes)
library(ggExtra) # For marginal histograms
```

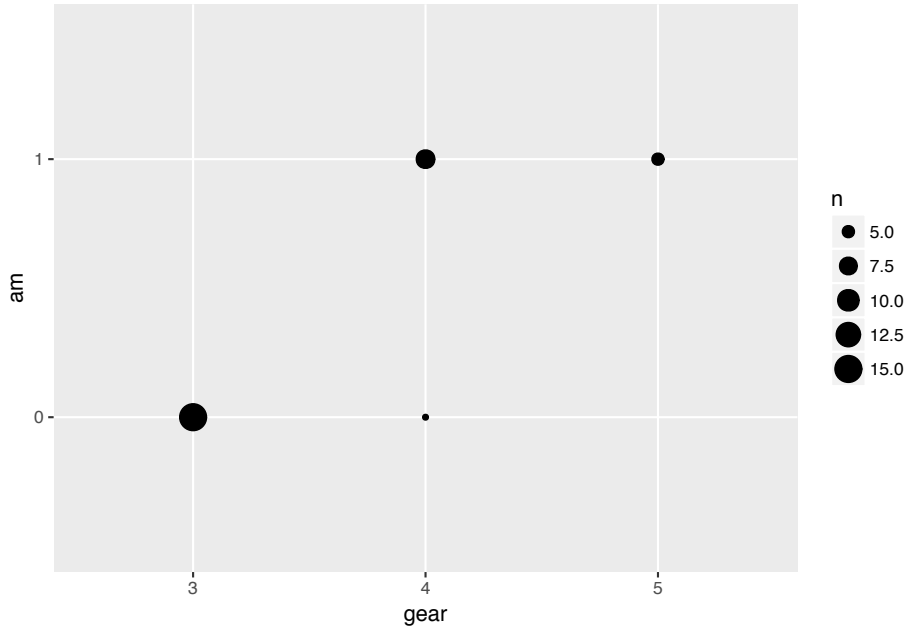
```
## Warning: package 'ggExtra' was built under R version
## 3.4.4
```

```
equation = function(x) {
  lm_coef <- list(a = round(coef(x)[1], digits = 2),
                  b = round(coef(x)[2], digits = 2),
                  r2 = round(summary(x)$r.squared, digits = 2));
  lm_eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(R)^2~"="~r2,lm_coef)
  as.character(as.expression(lm_eq));
}

mtcars.df = mtcars
fit = lm(mpg~wt, data = mtcars.df)

p = ggplot(mtcars.df, aes(x=wt, y=mpg)) +
  geom_point(colour = "darkblue") +
  geom_smooth(method=lm, se=FALSE) +
  annotate("text", x = 4, y = 30, label = equation(fit), parse = TRUE) +
  geom_rangeframe() + # Requires ggthemes
  theme_minimal()

p = ggMarginal(p, type = "histogram")
p
```

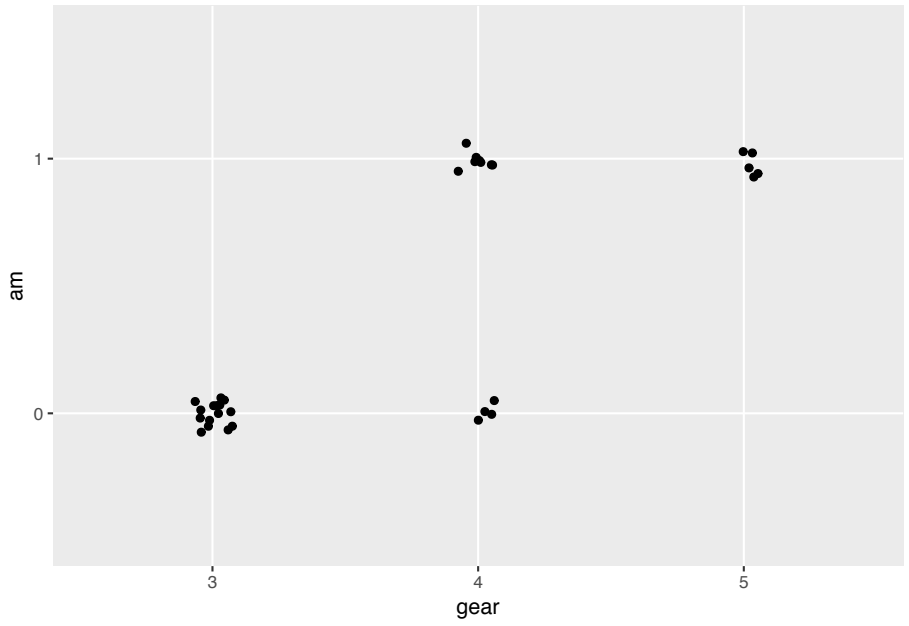


0.18 Categorical scatterplot

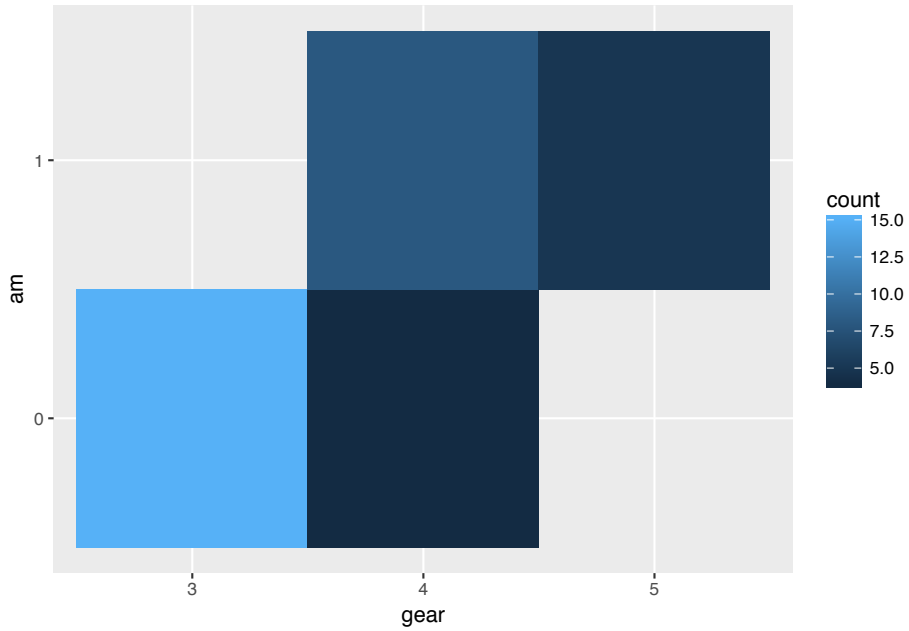
```
mtcars.df = mtcars
mtcars.df$gear = as.factor(mtcars.df$gear)
mtcars.df$am = as.factor(mtcars.df$am)

ggplot(mtcars.df, aes(gear, am)) +
  geom_count()
```




```
ggplot(mtcars.df, aes(gear, am)) +
 geom_jitter(width = 0.075, height = 0.075)
```



```
s.mtcars.df = mtcars.df %>% group_by(gear, am) %>% summarise(count = n())

ggplot(s.mtcars.df, aes(gear, am)) +
  geom_tile(aes(fill = count))
```

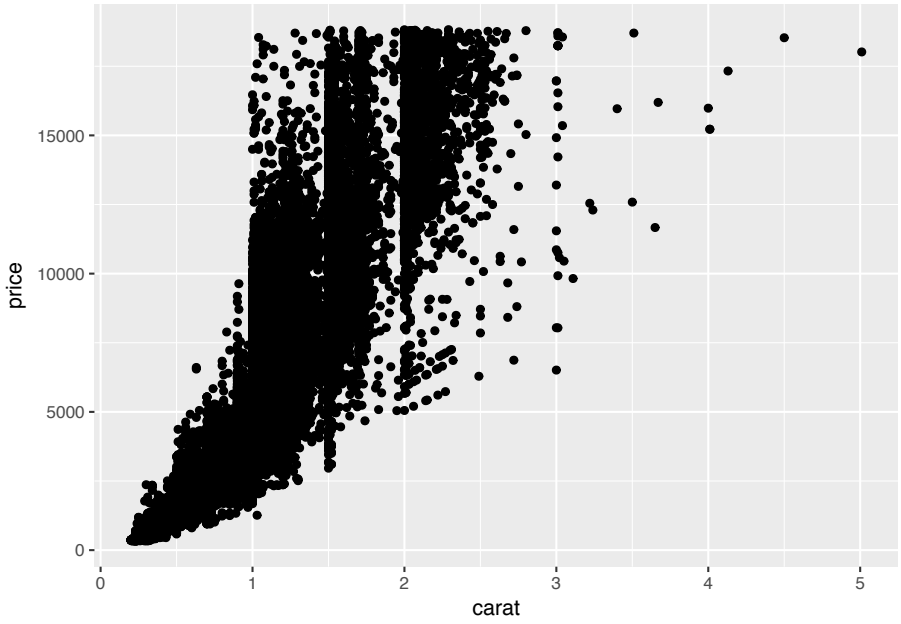


0.19 Table lens

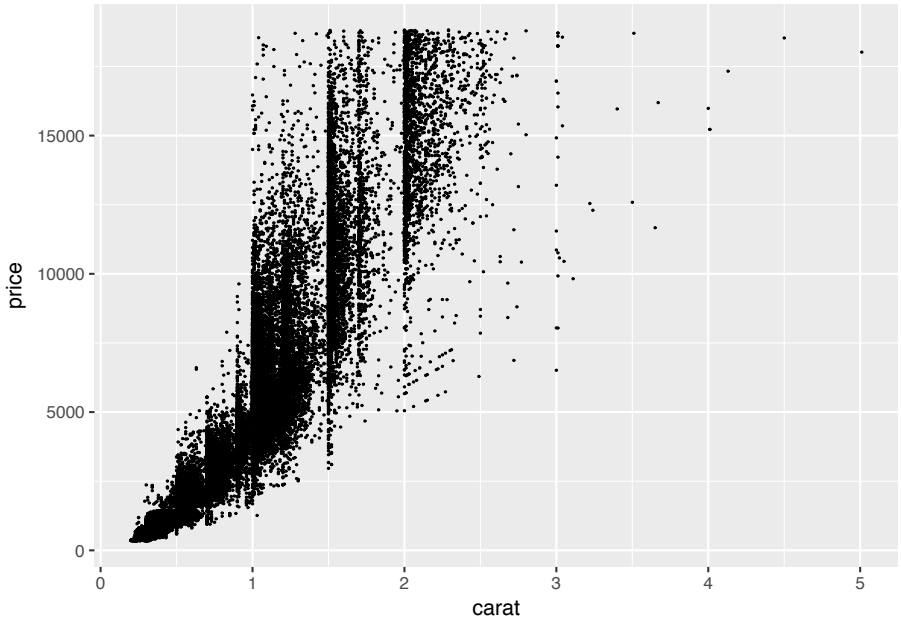
Table lens serves a similar purpose to the scatterplot but might be more familiar and focusses attention on individual variables and individual cases. Chapter **??** provides more detail on this technique.

0.20 Scatterplot with overplotting mitigation

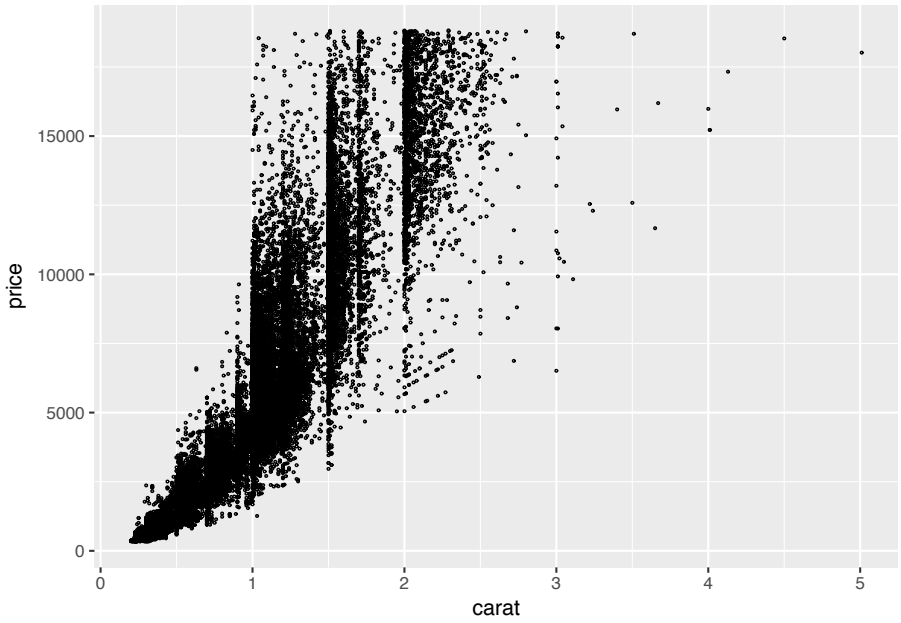
```
diamonds.df = diamonds

ggplot(diamonds.df, aes(carat, price)) +
  geom_point()
```



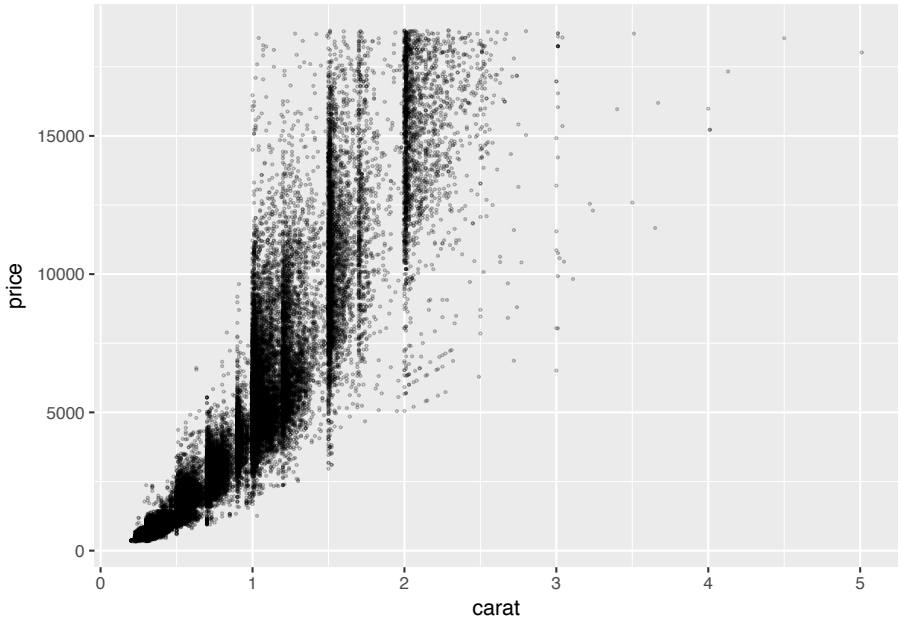

```
ggplot(diamonds.df, aes(carat, price)) +
  geom_point(size = .1)
```



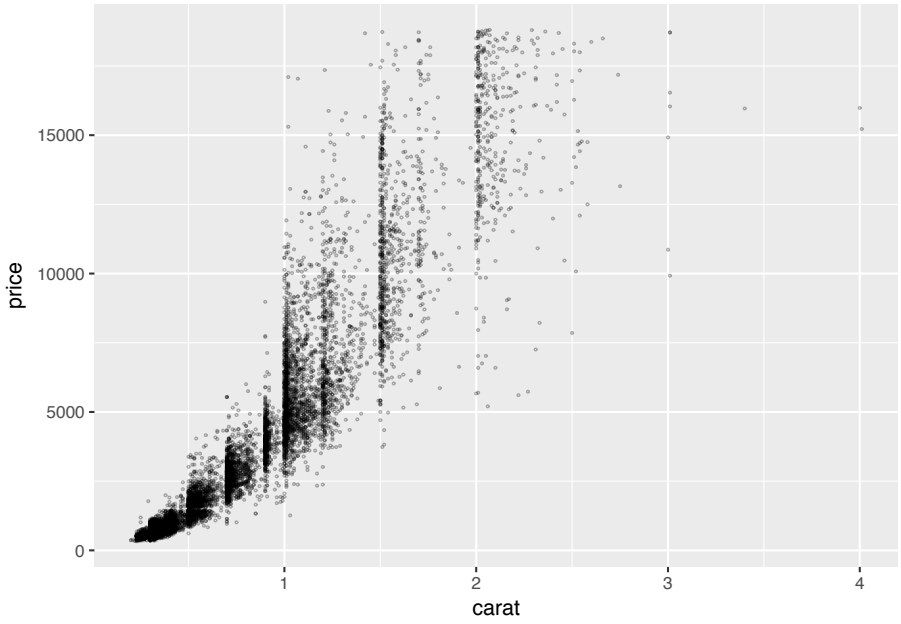

```
ggplot(diamonds.df, aes(carat, price)) +
  geom_point(size = .3, shape = 21)
```



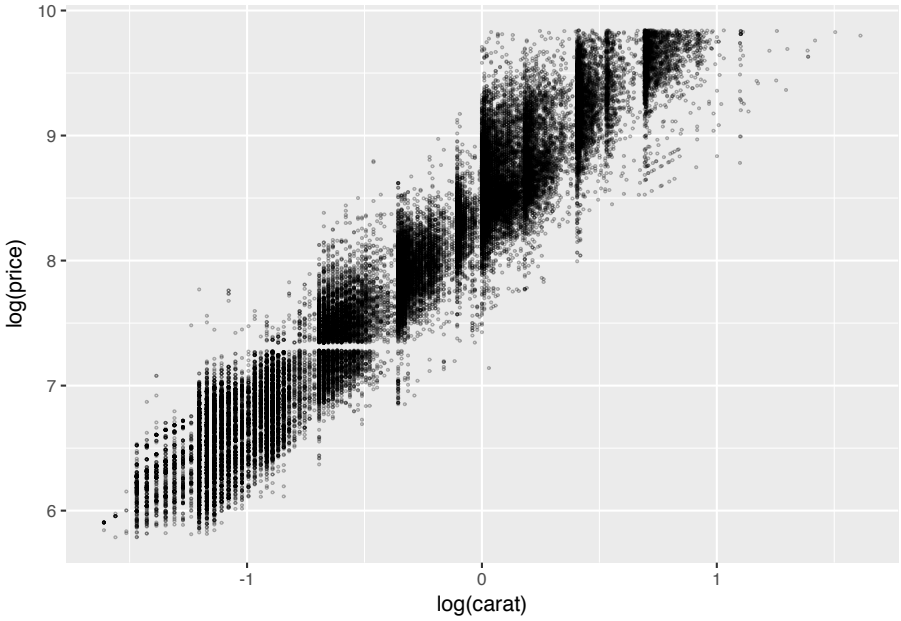
```
ggplot(diamonds.df, aes(carat, price)) +
  geom_point(size = .3, shape = 21, alpha = .3)
```



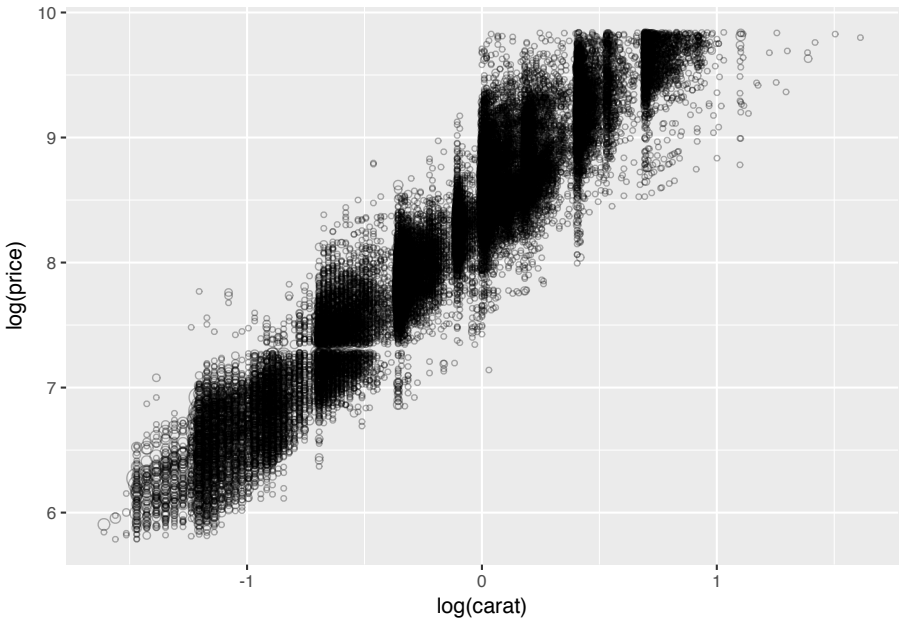

```
ggplot(diamonds.df %>% sample_n(10000), aes(carat, price)) +
  geom_point(size = .3, shape = 21, alpha = .3)
```



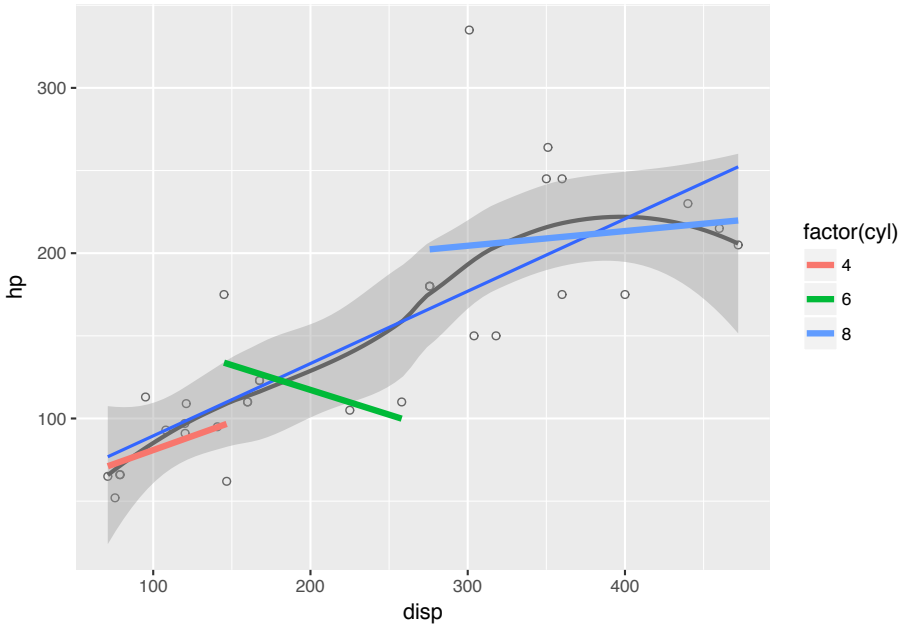

```
ggplot(diamonds.df, aes(log(carat), log(price))) +
  geom_point(size = .3, shape = 21, alpha = .3)
```



```
ggplot(diamonds.df, aes(log(carat), log(price))) +
  geom_count(show.legend=F, alpha =.3, shape =21)
```

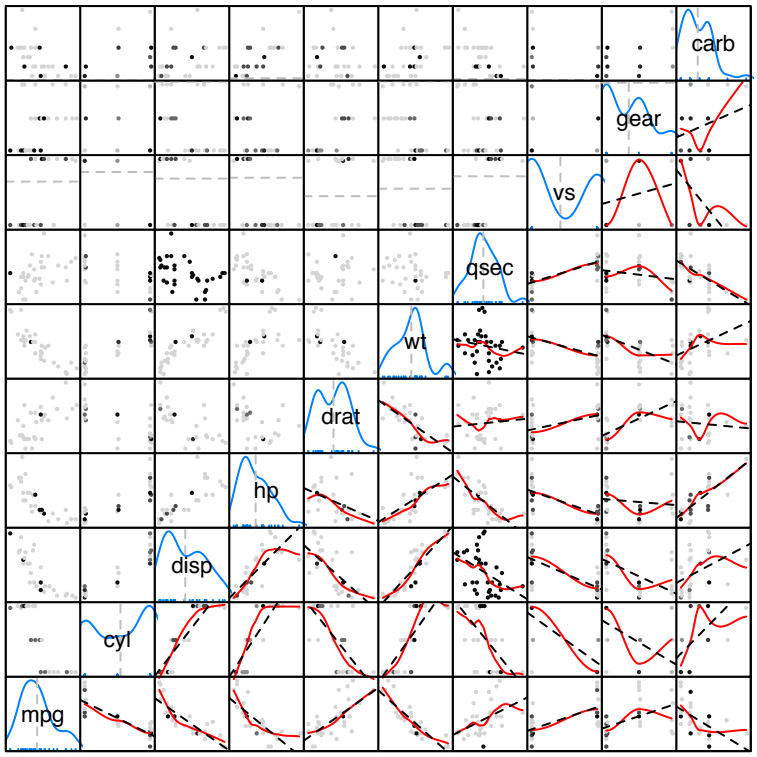



```
ggplot(data = mtcars.df, aes( x = disp, y = hp)) +
  geom_point(colour = "grey40", shape = 21)+
  geom_smooth(method = loess, colour = "grey40")+
  geom_smooth(method = lm, se = FALSE, size = .75) +
    geom_smooth(aes(colour = factor(cyl)),
        method = lm, se = FALSE, size = 1.5)
```



0.21 A matrix of scatterplots

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec"
##  [8] "vs"   "am"   "gear" "carb"
```



Scatter Plot Matrix

0

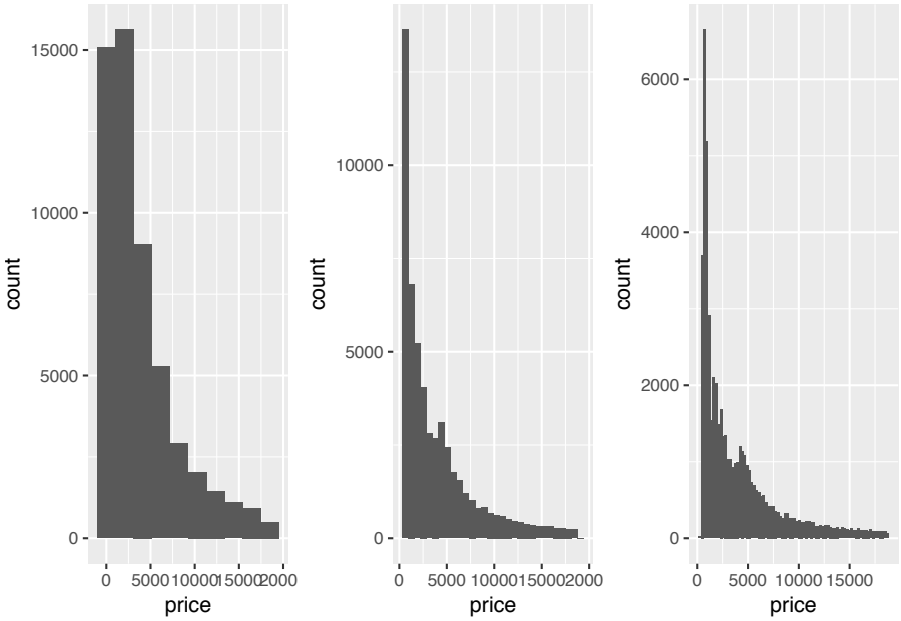
Distribution–histograms and density plots

```
library(tidyverse)
library(HistData)
```

Seeing the smooth and rough of data bins or binwidth, default number of bins is 30

0.22 Histograms and bin choice

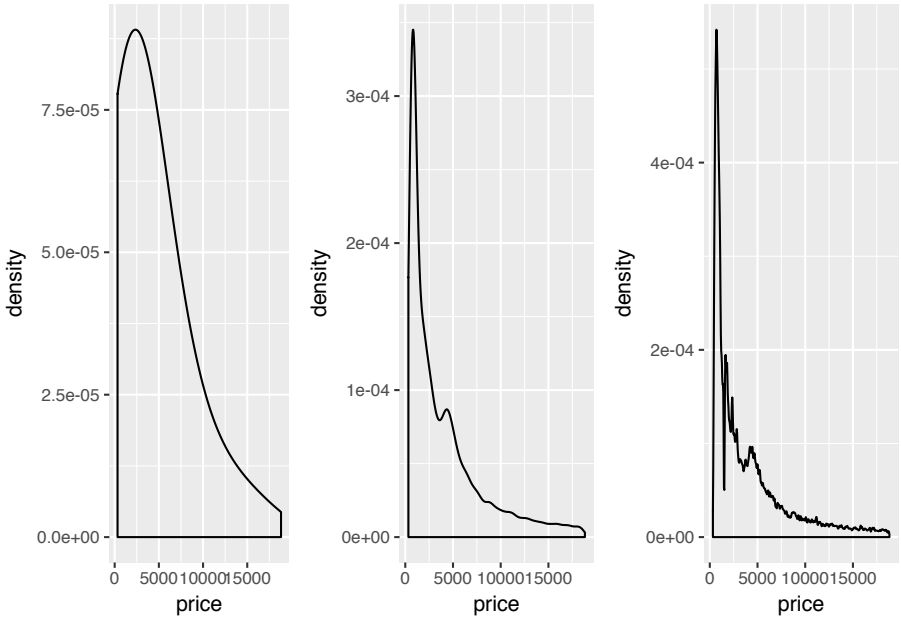
```
h10.plot = ggplot(data = diamonds.df, aes(price)) +
  geom_histogram(bins = 10)

h30.plot = ggplot(data = diamonds.df, aes(price)) +
  geom_histogram(bins = 30)

h80.plot = ggplot(data = diamonds.df, aes(price)) +
  geom_histogram(bins = 80)

ggarrange(h10.plot, h30.plot, h80.plot,
    nrow=1, ncol = 3, align = "h")
```



Density as abstraction and model

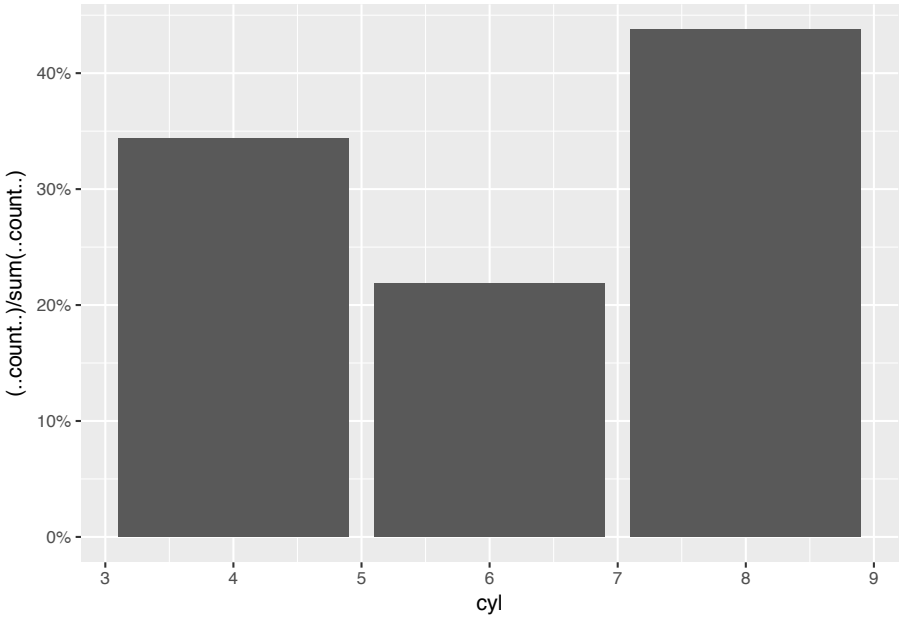
Adjust is a multiplier on the default kernel bandwidth and so 1 represents the default ## Density and kernel adjustment

```
a10.plot = ggplot(data = diamonds.df, aes(price)) +
  geom_density(adjust = 10)

a1.plot = ggplot(data = diamonds.df, aes(price)) +
  geom_density(adjust = 1)

a01.plot = ggplot(data = diamonds.df, aes(price)) +
  geom_density(adjust  = 0.1)

ggarrange(a10.plot, a1.plot, a01.plot,
    nrow=1, ncol = 3, align = "h")
```



0.23 Histogram percentage rather than count

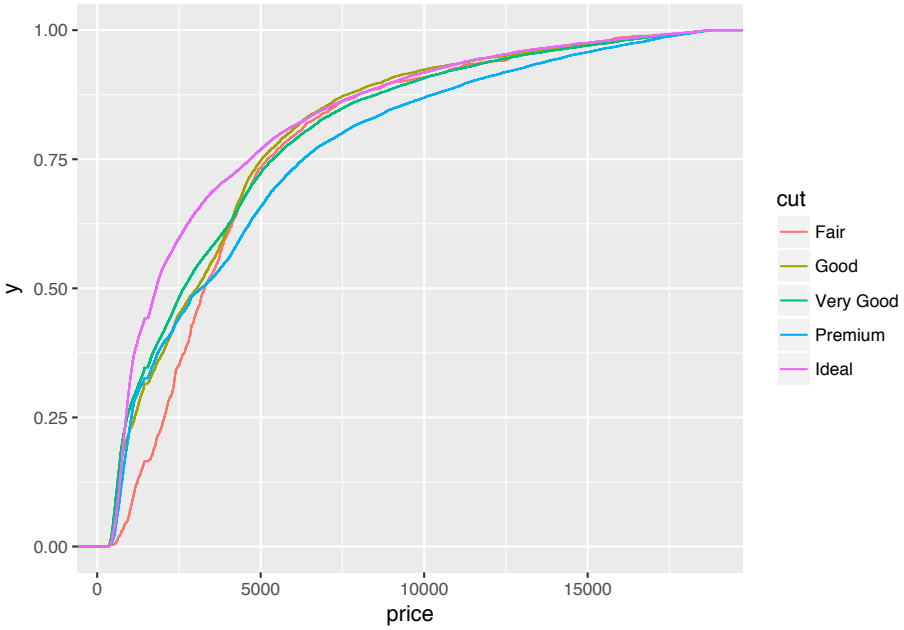
```
ggplot(mtcars.df , aes(x = cyl)) +
    geom_bar(aes(y = (..count..)/sum(..count..))) +
    scale_y_continuous(labels = percent)
```



0.24 Histogram, density overlay, and normal overlay

0.25 Cummulative density

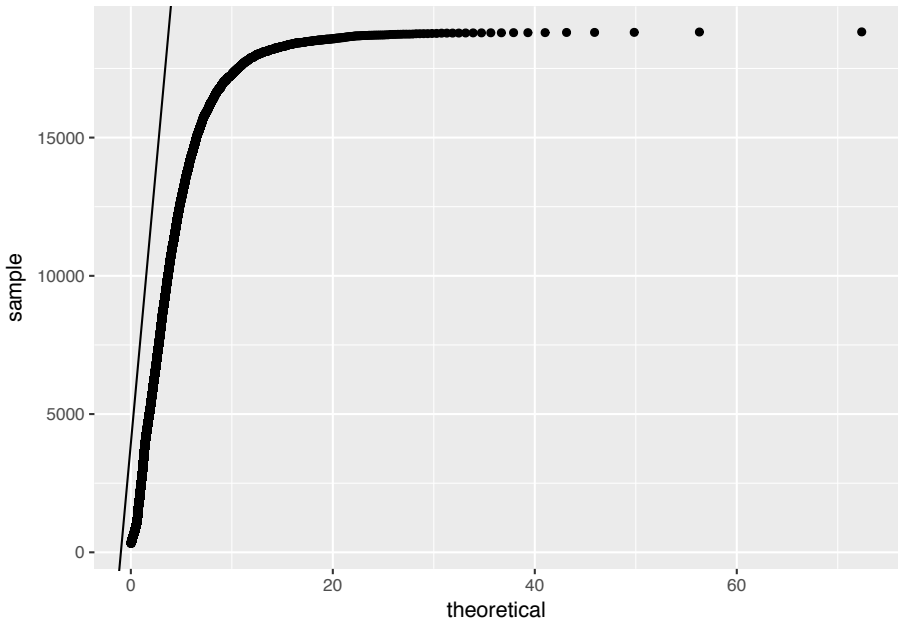
```
ggplot(diamonds.df, aes(price, colour = cut)) +
  stat_ecdf(geom = "step")
```



0.26 Quantile-quantle plot

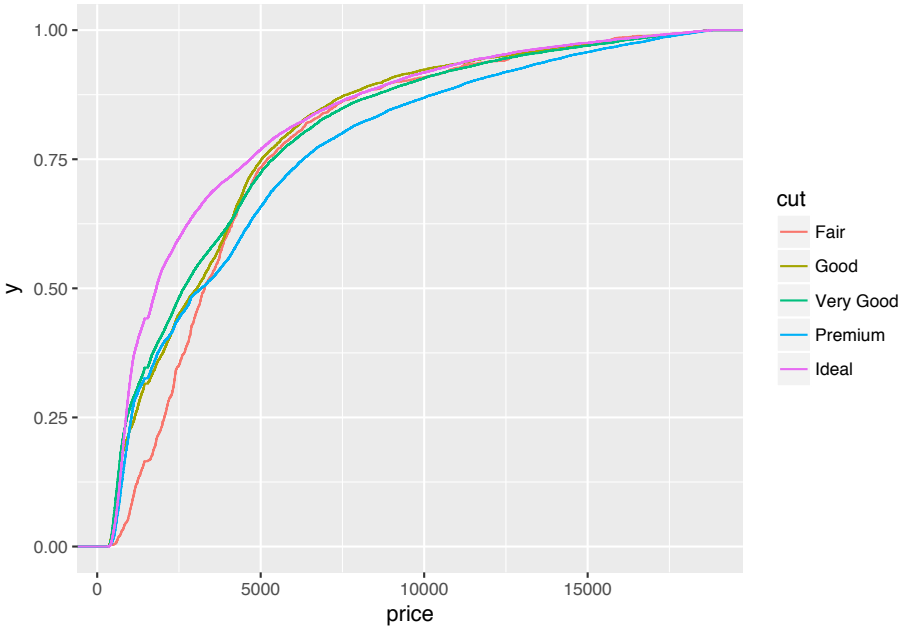
Plots quantiles of sample as a function of the quantiles of the theoretical distribution.

```
diamonds.df = diamonds

ggplot(diamonds.df, aes(sample = price)) +
  geom_qq(distribution = qlnorm) +
  geom_abline(intercept = mean(diamonds.df$price), slope = sd(diamonds.df$price))
```



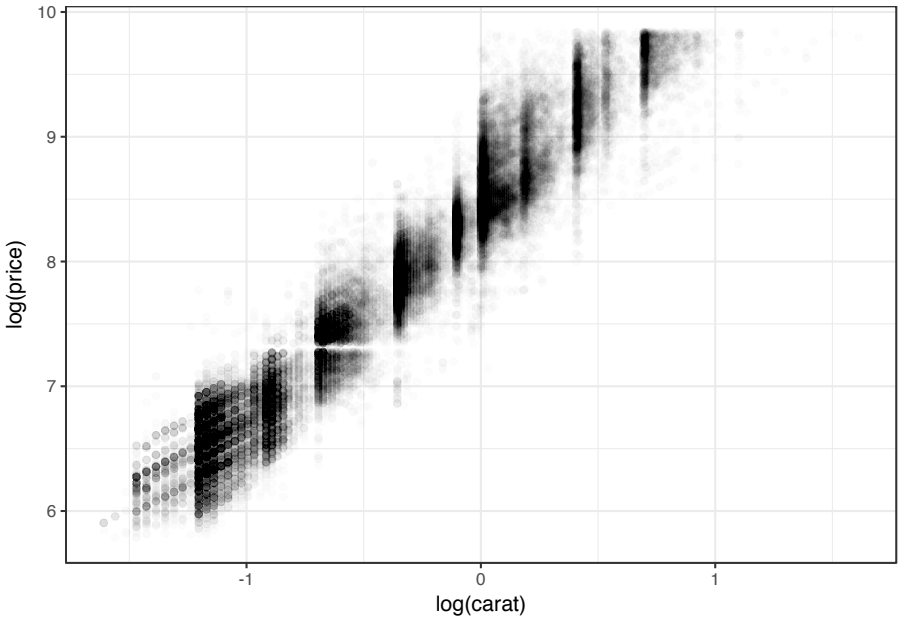
0.27 Cummulative density

```
ggplot(diamonds.df, aes(price, colour = cut)) +
  stat_ecdf(geom = "step")
```

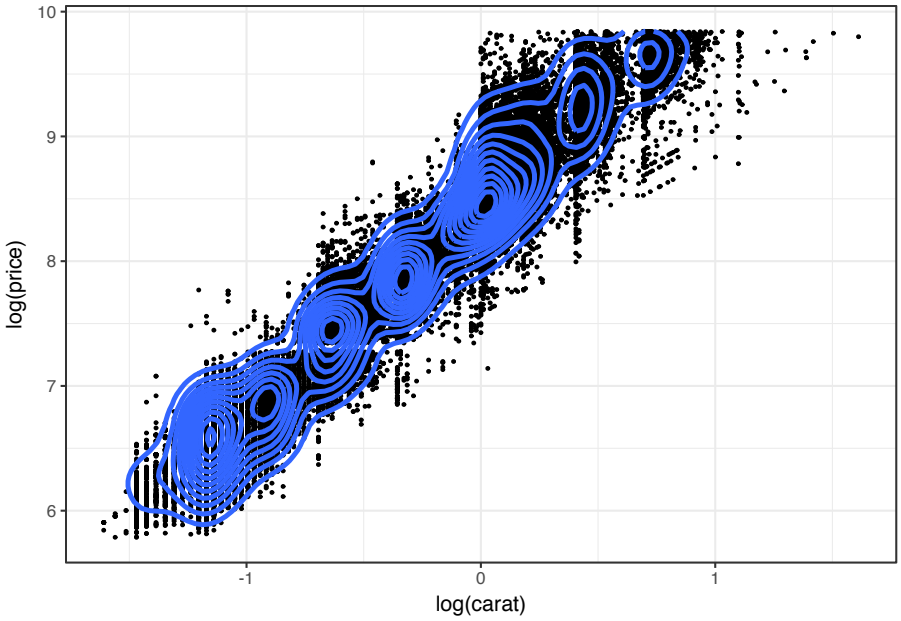


0.28 Distribution: 2-D distribution and overplotting revisited

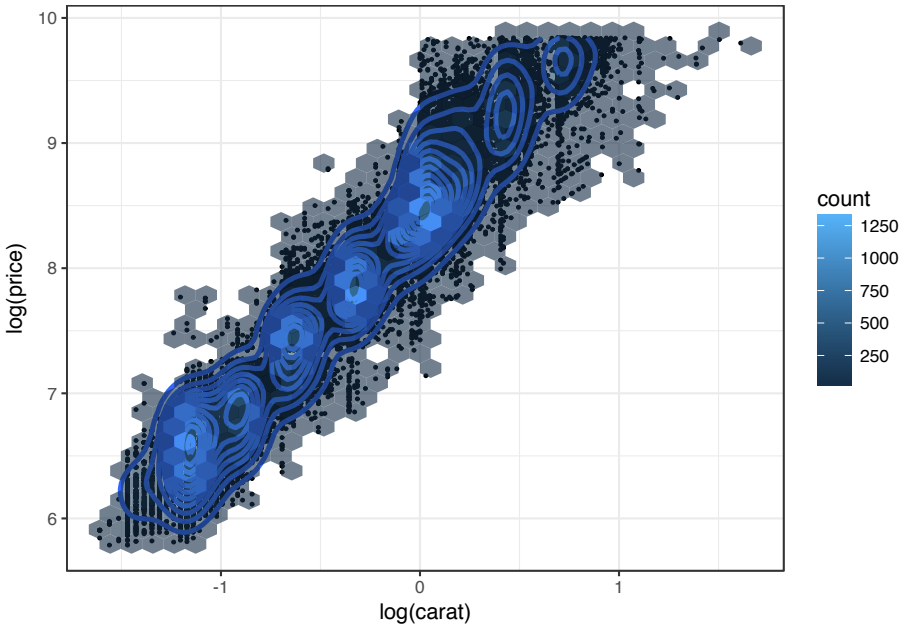
```
ggplot(diamonds, aes(log(carat), log(price)))+
  geom_point(alpha = .01)+
  theme_bw()
```



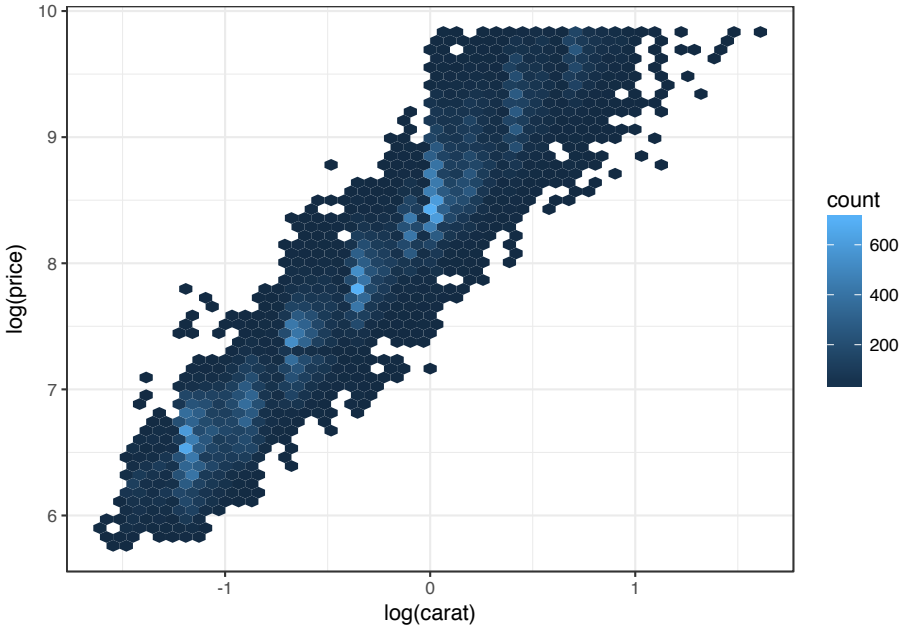

```
ggplot(diamonds, aes(log(carat), log(price)))+
  geom_point(size = .5)+
  geom_density2d(size=1.2)+
  theme_bw()
```



```
ggplot(diamonds, aes(log(carat), log(price)))+
  geom_point(size = .5)+
  geom_density2d(size=1.2)+
  geom_hex(alpha = .6) +
  theme_bw()
```



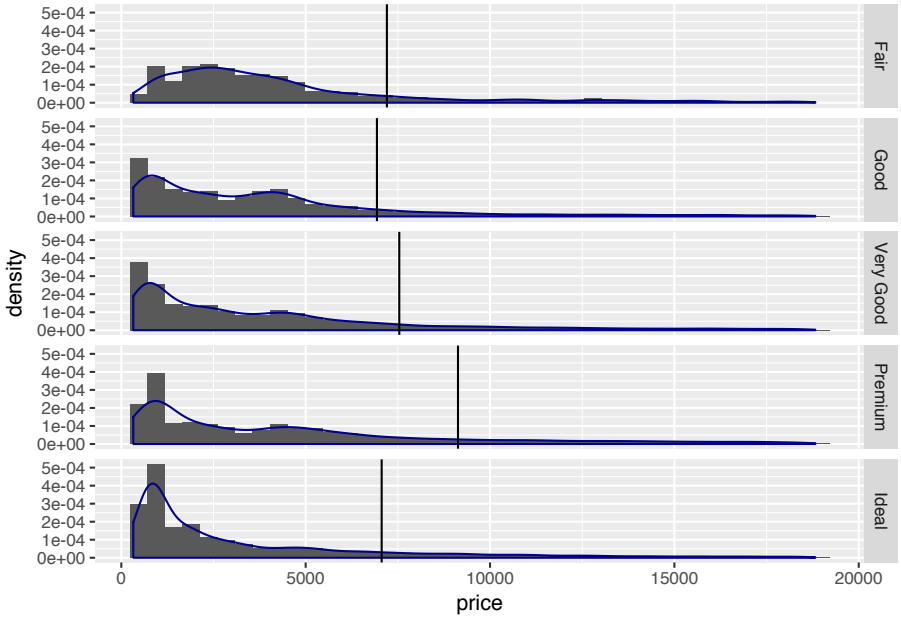

```
ggplot(diamonds, aes(log(carat), log(price)))+
  #geom_point( )+
  #geom_point(size = .5)+
  #geom_density2d(size=1.2)+
  geom_hex(bins = 50) +
  theme_bw()
```



0.29 Histogram with density and median reference line

TODO change to diversity data gender across job types

```
diamonds.df = diamonds

sum.diamonds.df = diamonds.df %>% group_by(cut) %>%
  summarise(q85 = quantile(price, 0.85))

ggplot(data = diamonds.df, aes(price)) +
  geom_histogram(aes(y = ..density..), bins = 40) +
  geom_density(colour = "darkblue") +
  geom_vline(data = sum.diamonds.df, aes(xintercept = q85)) +
  facet_grid(cut ~ .)
```



0.30 Ridge plot–An array of density plots

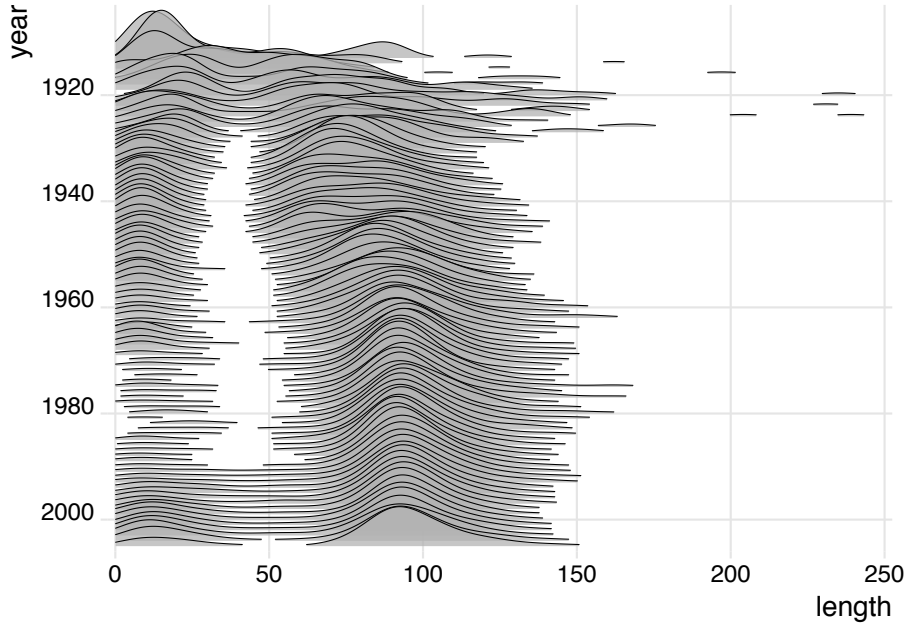
https://cran.r-project.org/web/packages/ggridges/vignettes/gallery.html

```
## Ridge plot
library(ggridges)
```

```
## Warning: package 'ggridges' was built under R version
## 3.4.4
```

```
library(ggplot2movies)

movies %>% filter(year>1912, length<250) %>%
ggplot(aes(x = length, y = year, group = year)) +
  geom_density_ridges(scale = 10, size = 0.25, rel_min_height = 0.03, alpha=.75) +
  scale_x_continuous(limits=c(0, 250), expand = c(0.01, 0)) +
  scale_y_reverse(breaks=c(2000, 1980, 1960, 1940, 1920, 1900), expand = c(0.01, 0)) +
  theme_ridges()
```

```
## Picking joint bandwidth of 6.89
```



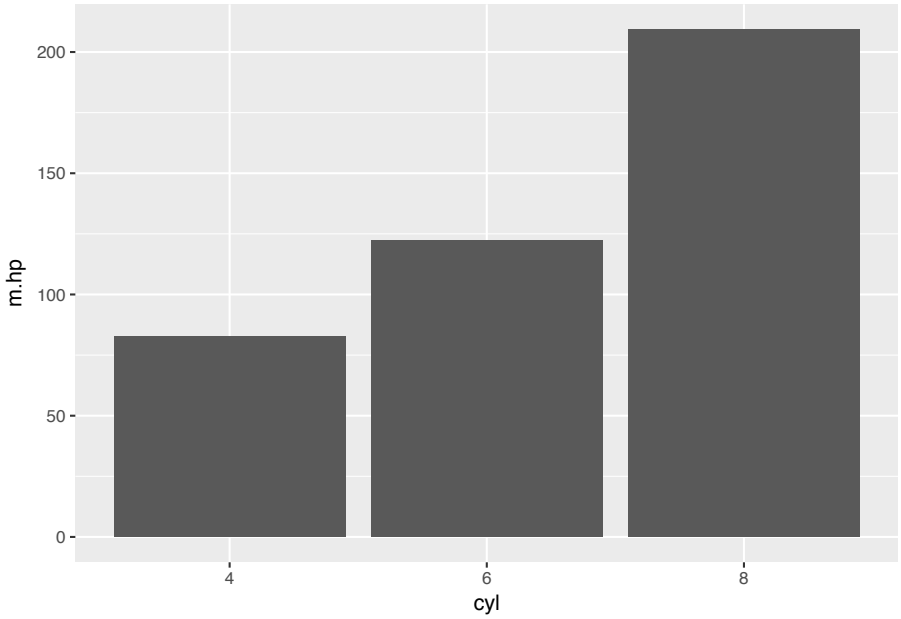
0

Comparison–barchart and boxplots

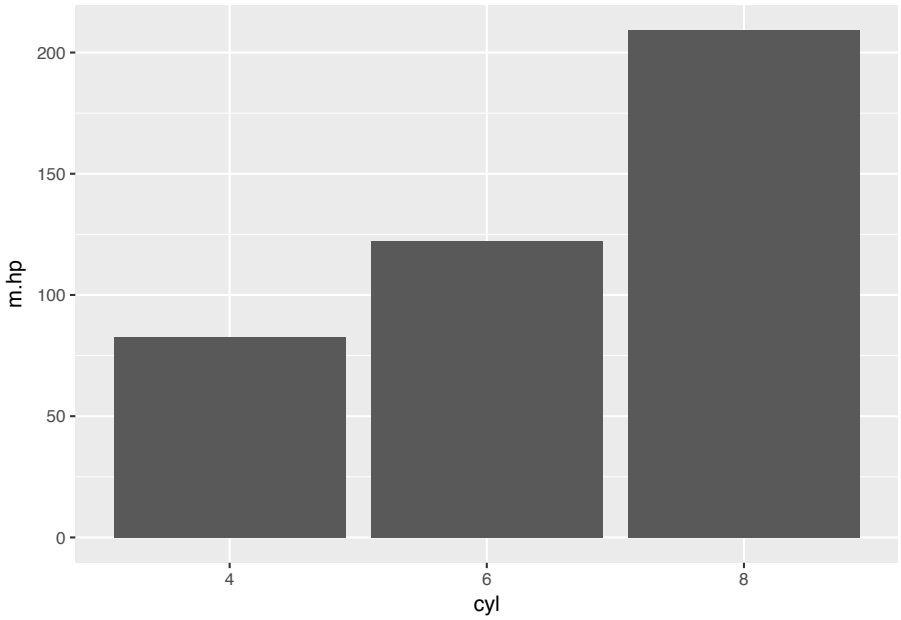
0.31 Graph considerations for communication: aggregation, abstraction, complexity

0.31.1 Simple bar chart

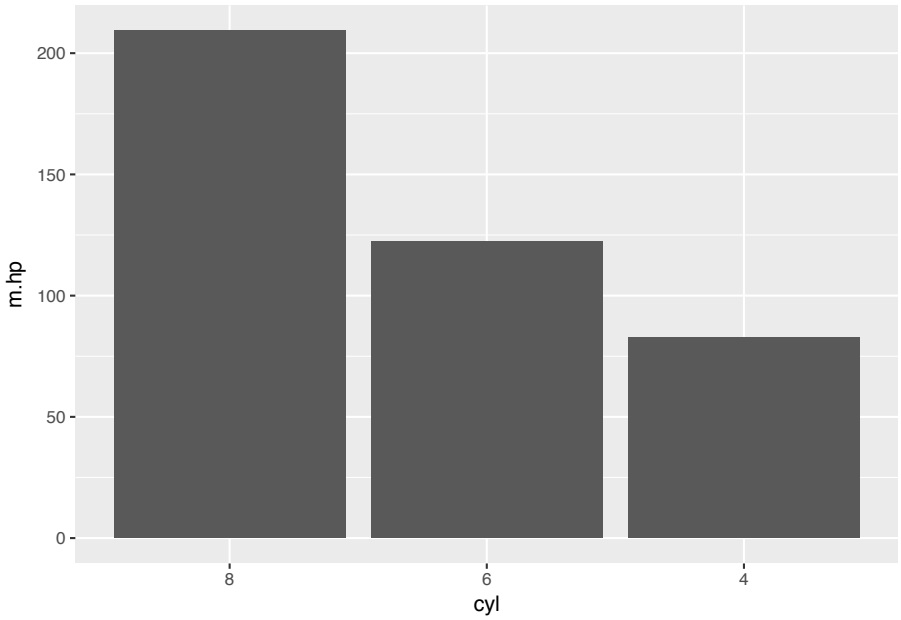
```
library(tidyr)
library(ggforce)
library(ggthemes)
library(ggstance)

mtcars.df = mtcars
mtcars.df = mtcars.df %>% mutate(cyl = as.factor(cyl))

s.mtcars.df = mtcars.df %>% group_by(cyl) %>% summarise(m.hp = mean(hp), se.hp= sd(hp)/n()


ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_bar(stat="identity")
```




```
ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_col()
```

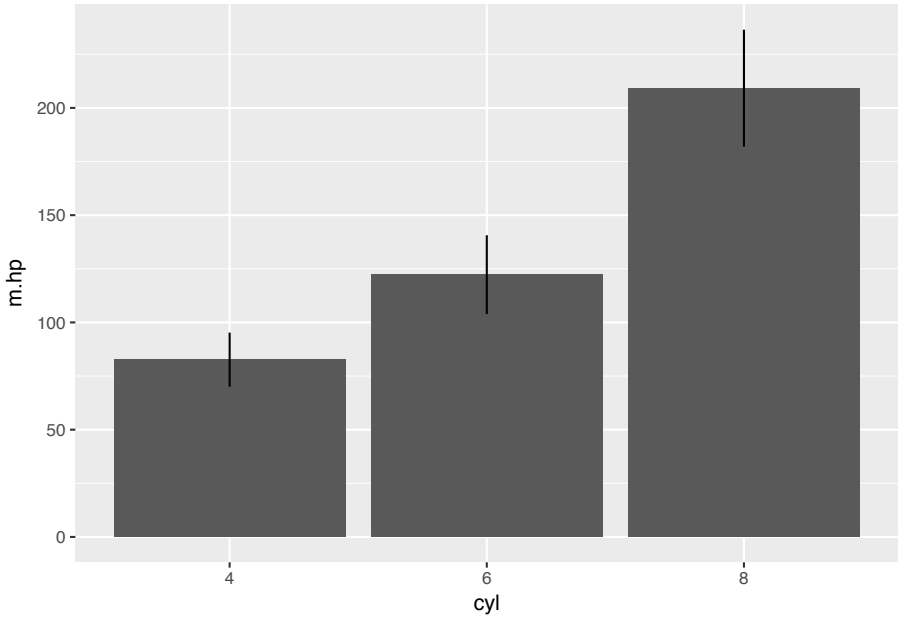


```
## Change order of bars
cyl.order <- c("8", "6", "4")
ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_col() +
  scale_x_discrete(limits = cyl.order)
```

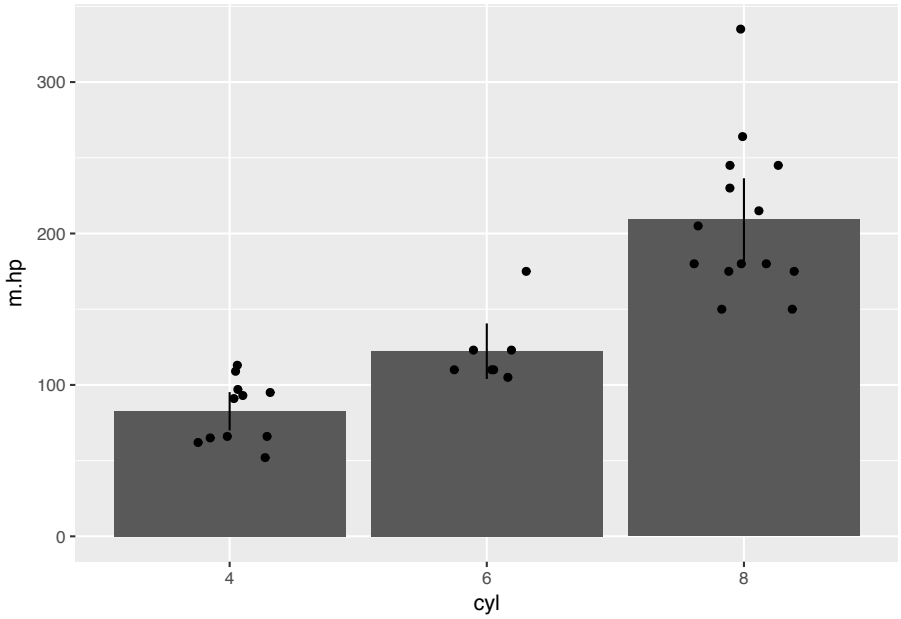


0.31.2 Bar chart with error bars

```
ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_bar(stat="identity")+
  geom_linerange(aes(ymin=m.hp-2*se.hp, ymax=m.hp+2*se.hp))
```



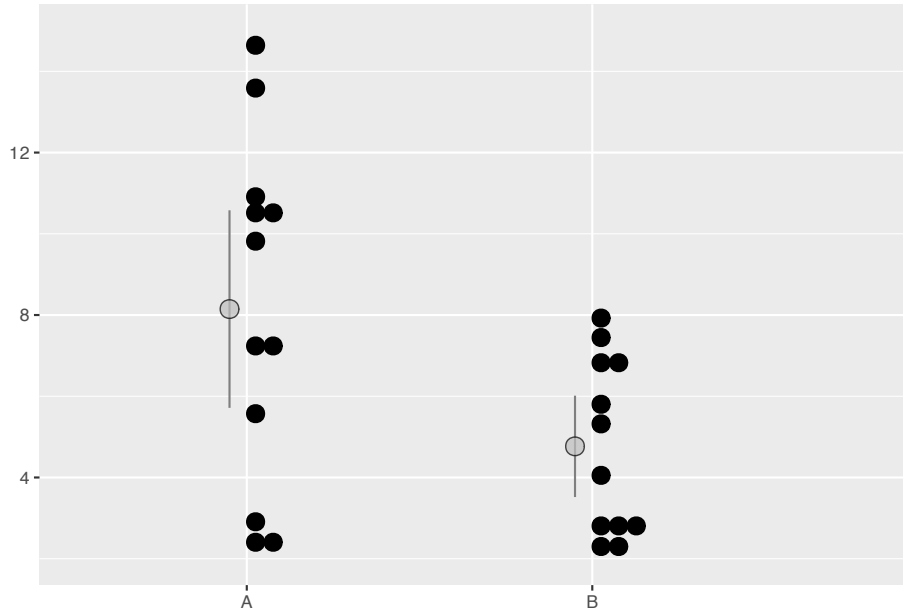

```
ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_bar(stat="identity")+
  geom_linerange(aes(ymin=m.hp-2*se.hp, ymax=m.hp+2*se.hp))+
  geom_point(data = mtcars.df, aes(cyl, hp), position = position_jitter(width = .2, height
```



0.31.3 dotplot and offset range plot

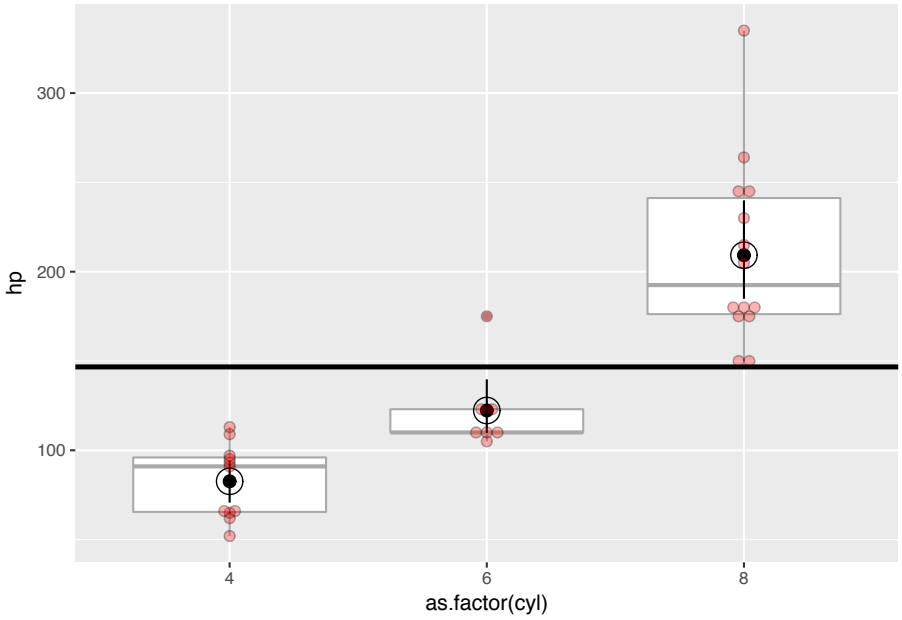
```
## Set seed and create data
set.seed(999)
df = data_frame(A = runif(12,1,17), B = runif(12, 2, 8))


l.df = gather(df, condition, value)
l.df$condition = as.factor(l.df$condition)
m.l.df = l.df %>% group_by(condition) %>% summarise(m.value = mean(value, na.rm=TRUE),
      n= sum(!is.na(value)), sd=sd(value, na.rm=TRUE), sde=sd(value, na.rm=TRUE)/n^.5,
       ci= 2*sde)
m.l.df$n.condition = as.numeric(m.l.df$condition)-.05


## Plot with offset for mean and error bar
ggplot()+
  geom_dotplot(data = filter(l.df, condition=="A"|condition=="B"),
               aes(condition, value), binaxis = "y", stackdir = "up")+
  geom_linerange(data = filter(m.l.df, condition=="A"|condition=="B"),
                 aes(n.condition, ymin=m.value-ci, ymax=m.value+ci), color="grey50") +
  geom_point(data = filter(m.l.df, condition=="A"|condition=="B"),
             aes(n.condition, y= m.value), shape = 21, size = 4, fill="grey", alpha=.7) +
  labs(x="", y="") +
  ylim(2, 15)
```

```
## `stat_bindot()` using `bins = 30`. Pick better value with `binwidth`.
```



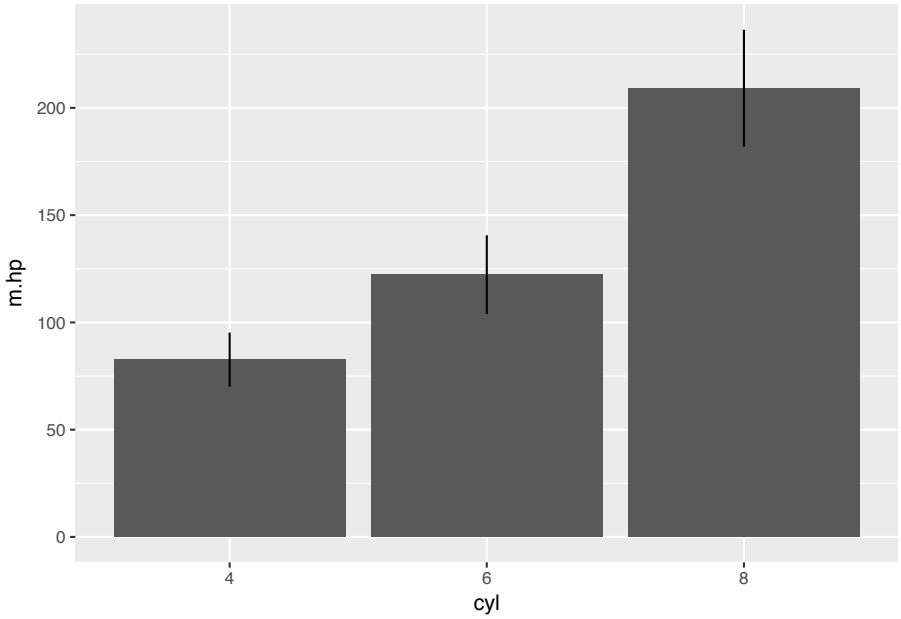
0.31.4 Statistical significance in context

```
ggplot(data = mtcars.df, aes(x = as.factor(cyl), y = hp)) +
  geom_boxplot(colour = "darkgrey") +
  geom_point(stat="summary", fun.y = "mean", size = 6, shape = 1) +
  geom_pointrange(stat="summary", fun.data = "mean_cl_boot") +
  geom_dotplot(binaxis = "y", stackdir = "center", binwidth = 1,
               dotsize = 6,alpha = .3, color = "black", fill = "red") +
  geom_hline(aes(yintercept = mean(hp)), size = 1.2)
```

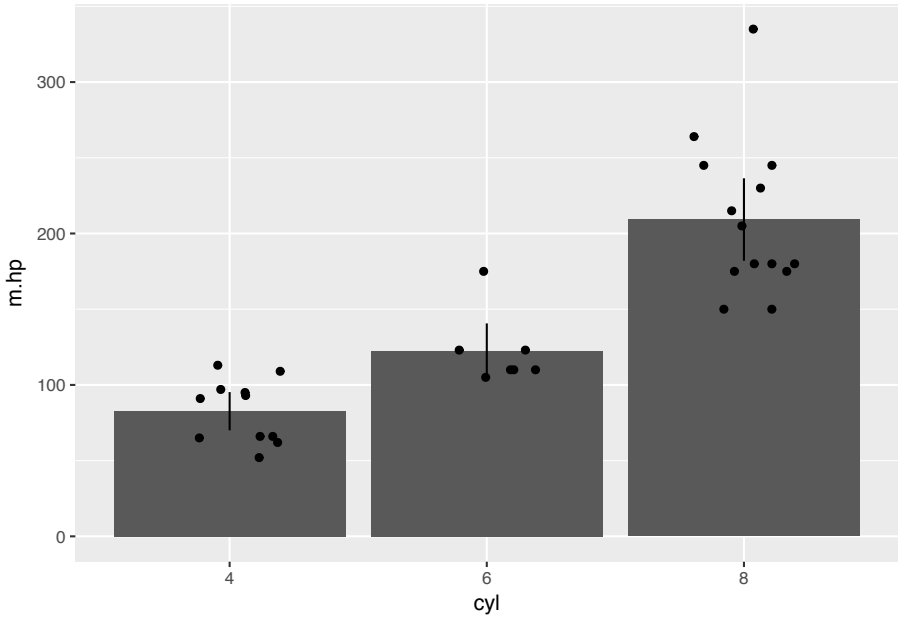


0.32 Comparing distributions, box, violyn, and sina plot

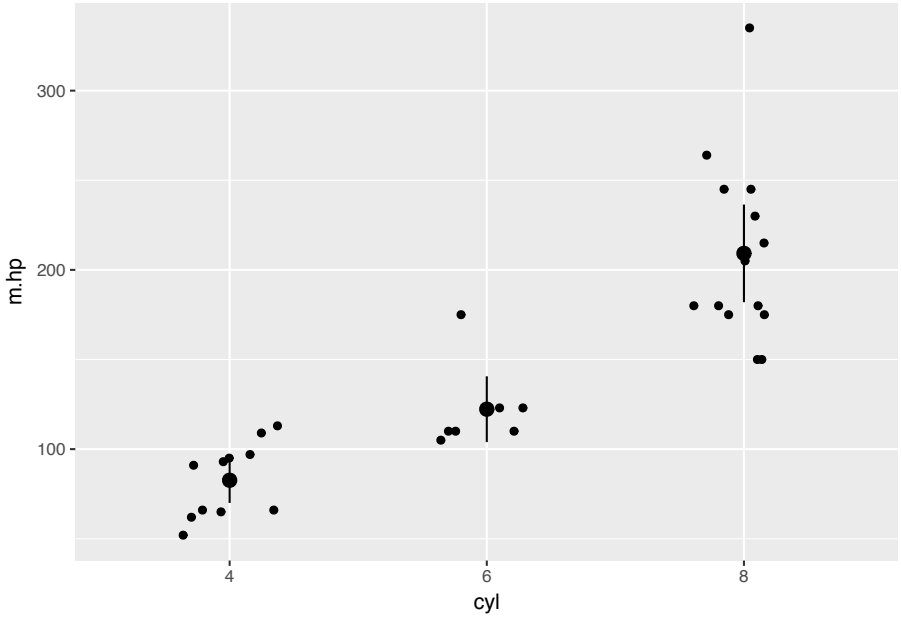
```
library(tidyr)

ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_col()+
  geom_linerange(aes(ymin=m.hp-2*se.hp, ymax=m.hp+2*se.hp))
```



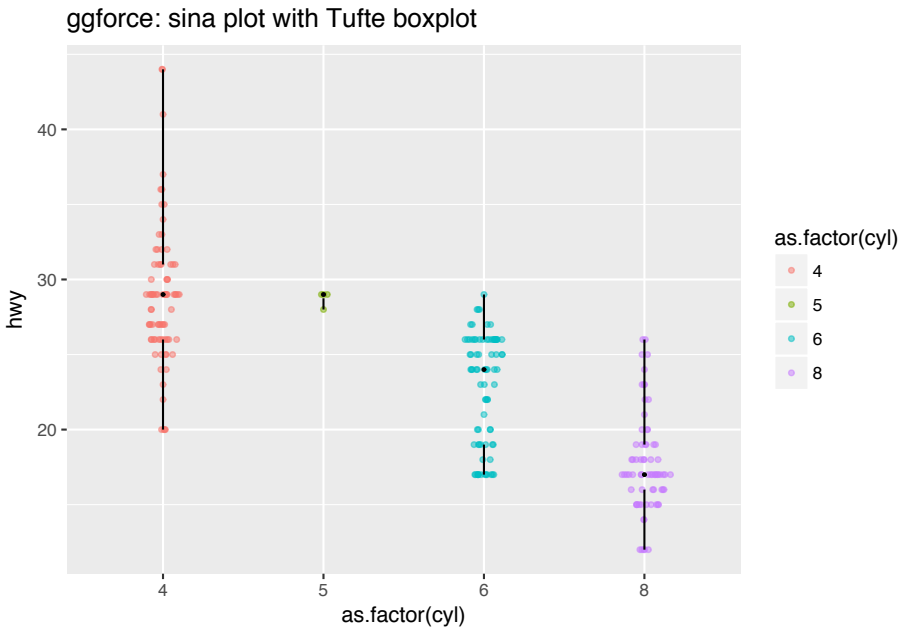

```
ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_col()+
  geom_linerange(aes(ymin=m.hp-2*se.hp, ymax=m.hp+2*se.hp))+
  geom_point(data = mtcars.df, aes(cyl, hp), position = position_jitter(width = .2, height
```



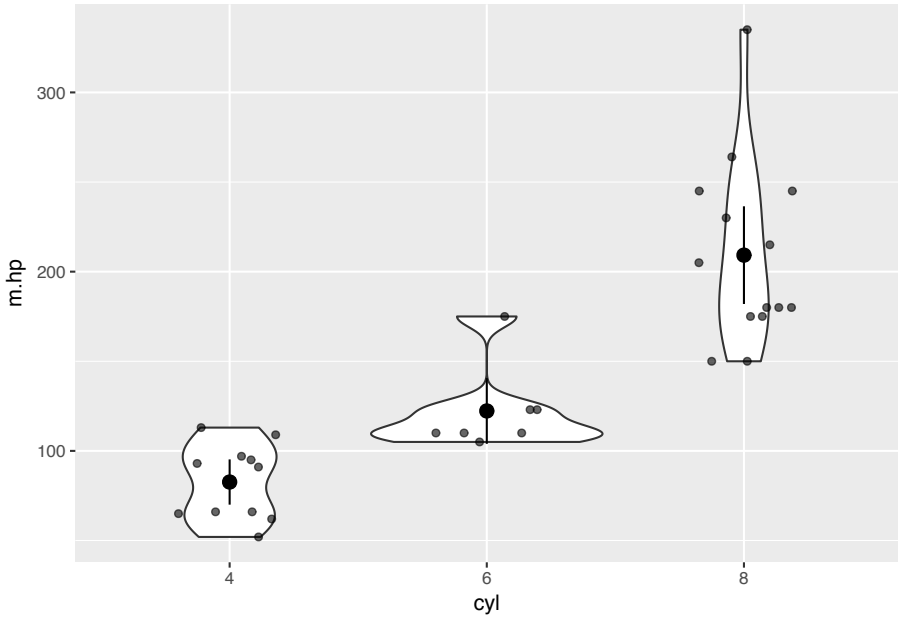
```
ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_point(stat="identity", size = 3)+
  geom_linerange(aes(ymin=m.hp-2*se.hp, ymax=m.hp+2*se.hp))+
  geom_point(data = mtcars.df, aes(cyl, hp), position = position_jitter(width = .2, height
```




```
## Sina plot
ggplot(mpg, aes(as.factor(cyl), hwy))+
 geom_sina(aes(color = as.factor(cyl)),size = 1, alpha =.5) +
 geom_tufteboxplot()+
  labs(title = "ggforce: sina plot with Tufte boxplot")
```

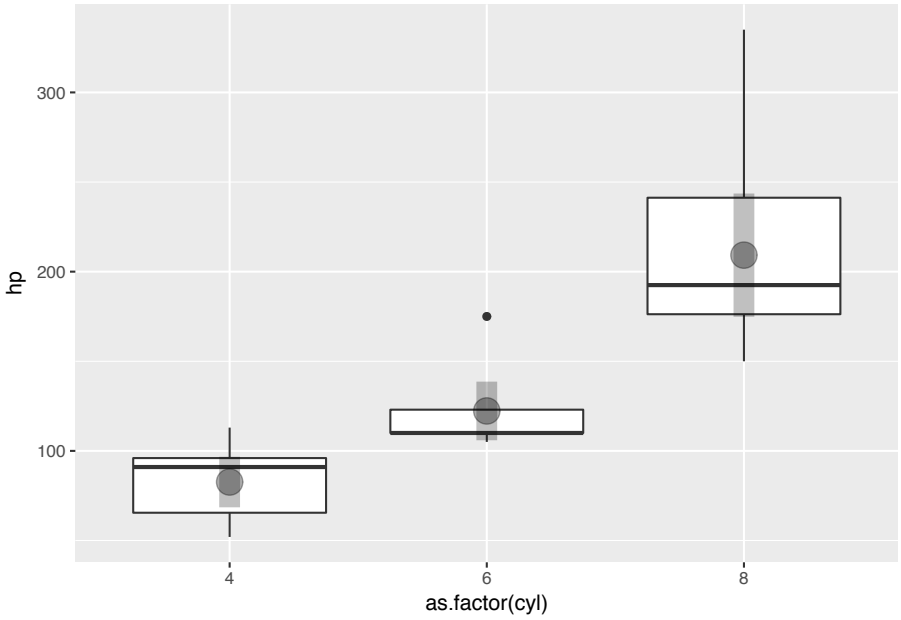



```
# ggplot(s.mtcars.df, aes(m.hp, m.mpg, colour =as.factor(cyl)))+
# geom_pointrangeh(aes(xmin= m.hp-se.hp, xmax = m.hp+se.hp))+
#   geom_pointrange(aes(ymin= m.mpg-se.mpg, ymax = m.mpg+se.mpg))+
#   labs(title = "ggstance: horizontal point range")


ggplot(data = s.mtcars.df, aes(x = cyl, y = m.hp)) +
  geom_violin(data= mtcars.df, aes(cyl, hp))+
  geom_point(stat="identity", size = 3)+
  geom_linerange(aes(ymin=m.hp-2*se.hp, ymax=m.hp+2*se.hp))+
  geom_point(data = mtcars.df, aes(cyl, hp),
             position = position_jitter(width = .2, height = 0), alpha =.6)
```



0.32.1 Compare empirical and theoretical distribution

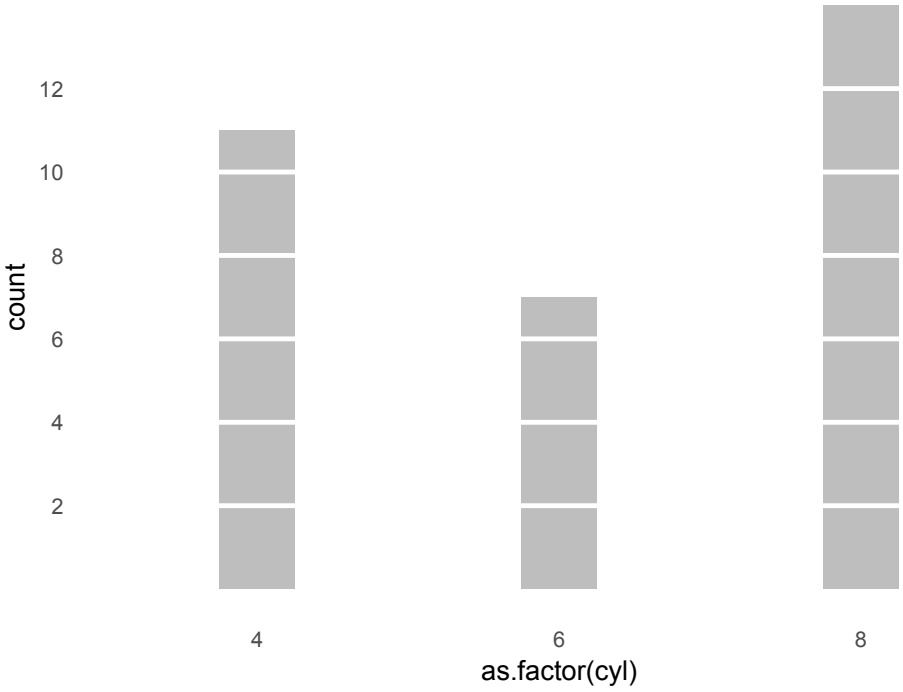
```
sum.mtcars.df = mtcars.df%>% group_by(cyl) %>%
    summarise(m.hp = mean(hp), sd.hp = sd(hp))

ggplot(mtcars.df) +
  geom_boxplot(aes(as.factor(cyl), hp)) +
  geom_linerange(data = sum.mtcars.df,
         aes(x = as.factor(cyl),
         ymin = m.hp + qnorm(.25)*sd.hp, ymax = m.hp + qnorm(.75)*sd.hp),
                  size = 5, alpha = .25) +
  geom_point(data = sum.mtcars.df,
          aes(as.factor(cyl), y= m.hp),size = 6, alpha = .33)
```



0.32.2 Tufte-inspired minimal bar chart

http://motioninsocial.com/tufte/

```
#TODO replace with better dataset with for more columns
library(ggthemes)
ggplot(mtcars.df, aes(x=as.factor(cyl))) +
  geom_bar(width=0.25, fill="gray") +
  scale_y_continuous(breaks=seq(2, 12, 2)) +
  geom_hline(yintercept=seq(2, 12, 2), colour="white", lwd=1) +
  theme_tufte(base_size=12, ticks=F, base_family = "Arial")
```

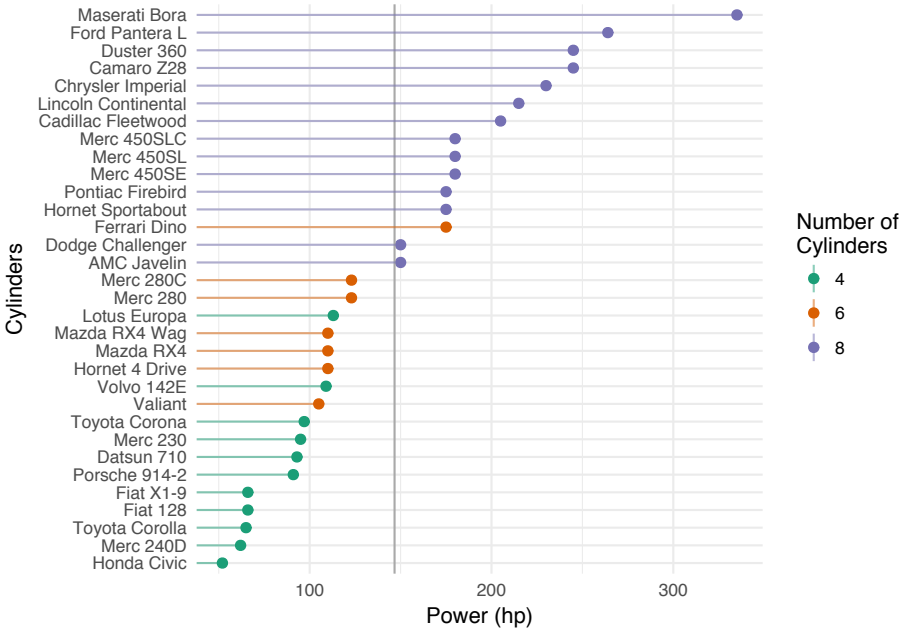


0.33 Comparing across many variables

0.33.1 Dot plots and reordering

Comparing many mean values

```
#TODO Change to mean sd catepilar plot
mtcars.df = mtcars
mtcars.df$name = rownames(mtcars.df)
ggplot(data = mtcars.df, aes(reorder(name, hp), y = hp, colour = as.factor(cyl))) +
  geom_point(size = 2) +
  geom_hline(aes(yintercept = mean(hp)), colour = "darkgrey") +
  geom_linerange(aes(ymin= -Inf, ymax= hp), alpha =.5) +
  coord_flip() +
  labs(x = "Cylinders", y = "Power (hp)") +
  scale_colour_brewer(name = "Number of \nCylinders", palette="Dark2") + # http://colorbre
  theme(legend.position = c(.75, .25)) +
  theme_minimal()
```



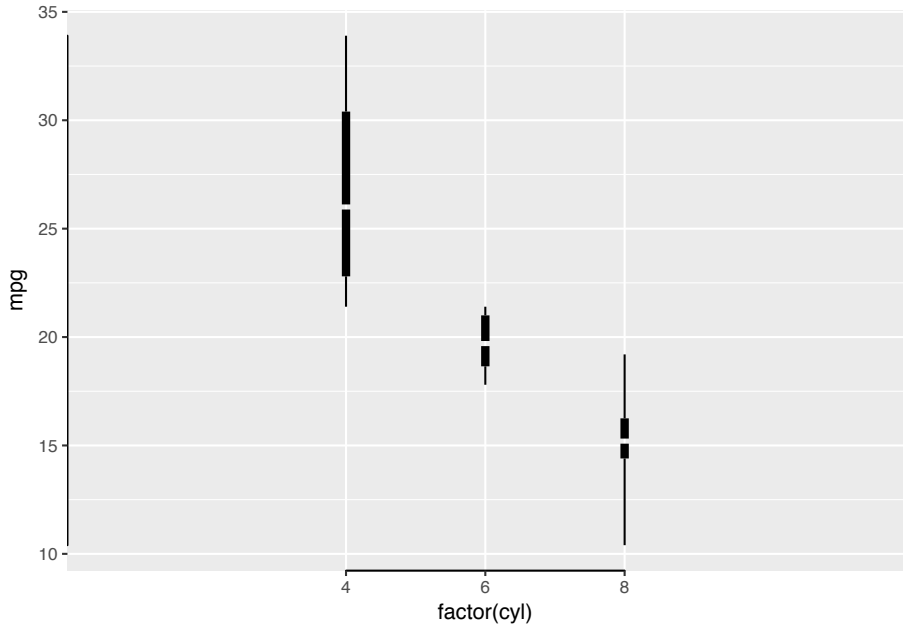
0.33.2 Point range on x and y

```
## Point range on x and y
library(ggstance)
s.mtcars.df = mtcars %>% group_by(cyl) %>%
  summarise(m.hp = mean(hp), se.hp = sd(hp)/n()^.5,
            m.mpg = mean(mpg), se.mpg = sd(mpg)/n()^.5)
```

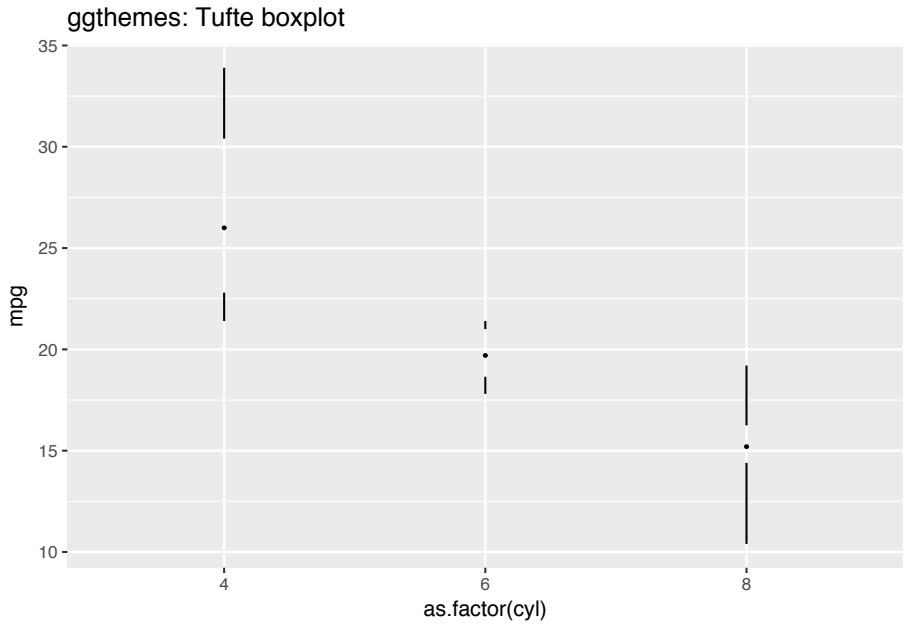
0.33.3 Tufte boxplot for many variables

```
library(ggthemes)

ggplot(mtcars, aes(factor(cyl), mpg)) +
  geom_tufteboxplot(median.type = "line", whisker.type = 'line', hoffset = 0, width = 4) +
  geom_rangeframe()
```

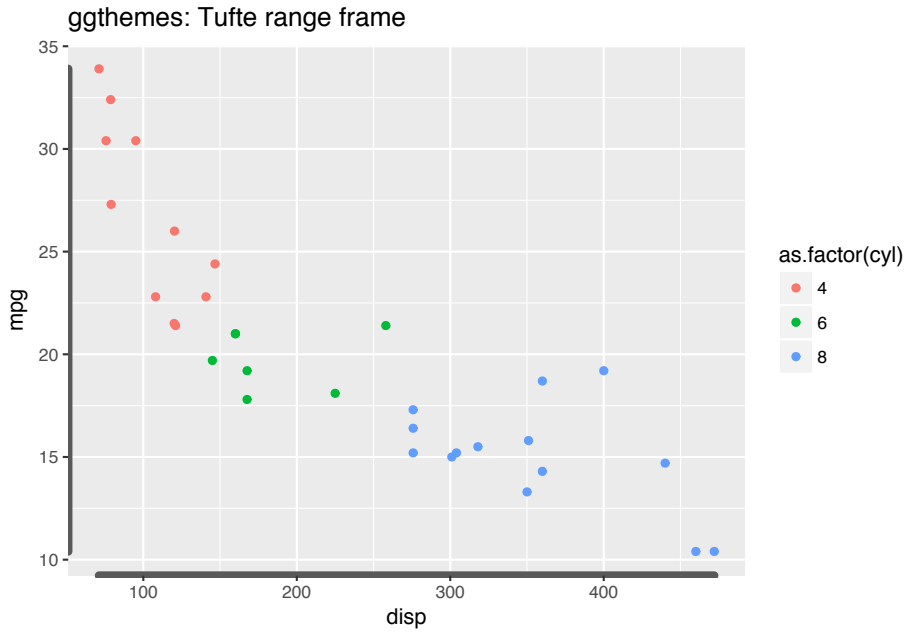
```
## Warning: position_dodge requires non-overlapping x
## intervals
```




```
## Tufte boxplot
ggplot(mtcars, aes(as.factor(cyl), mpg))+
  geom_tufteboxplot()+
  labs(title = "ggthemes: Tufte boxplot")
```



```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl)))+
geom_point()+
  geom_rangeframe(size = 2, colour = "grey35")+
  labs(title = "ggthemes: Tufte range frame")
```

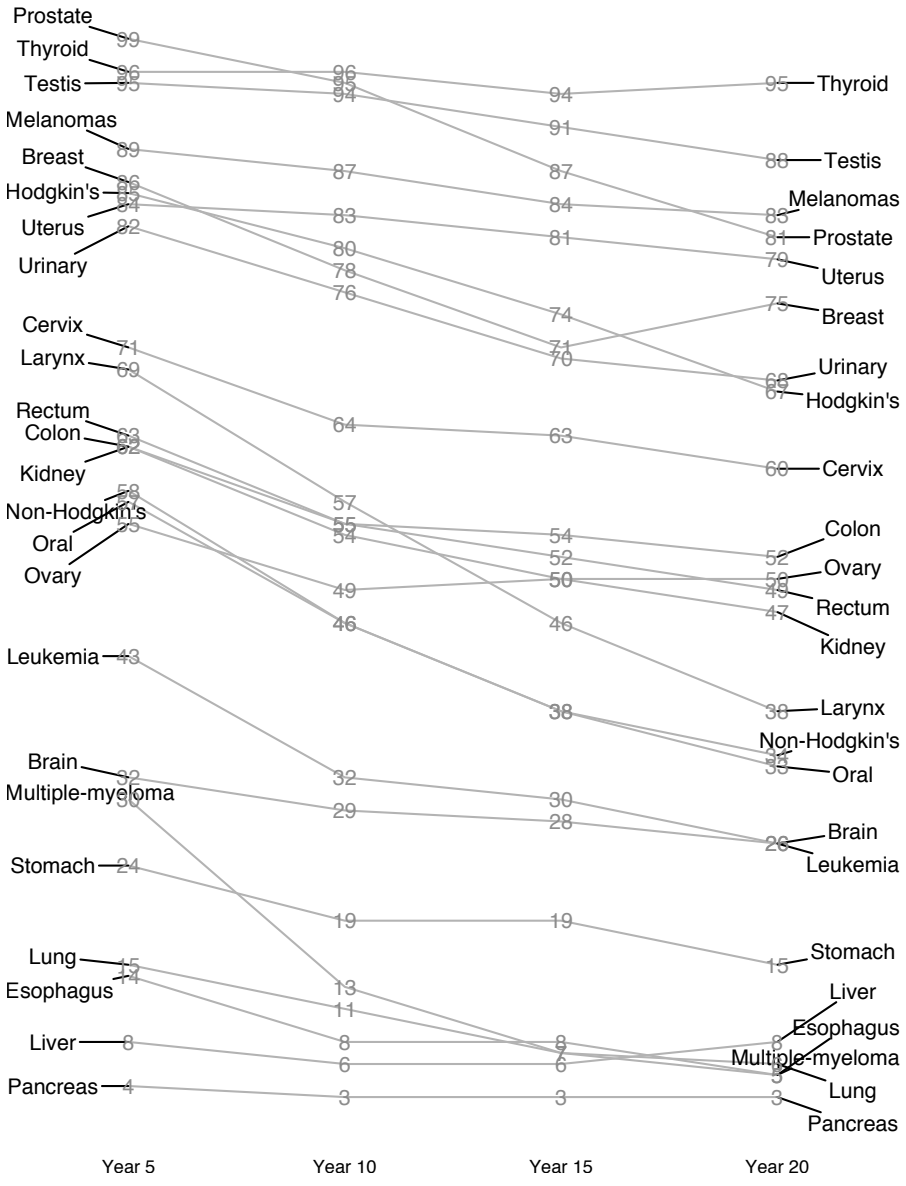


0.33.4 Tufte-inspired slope graphs

```
library(tidyverse)
library(ggrepel)

# https://github.com/leeper/slopegraph
cancer.df = read_csv("data/tufte-cancer-survival-data.csv")
```

```
## Parsed with column specification:
## cols(
##   Type = col_character(),
##   `Year 5` = col_double(),
##   `Year 10` = col_double(),
##   `Year 15` = col_double(),
##   `Year 20` = col_double()
## )
```

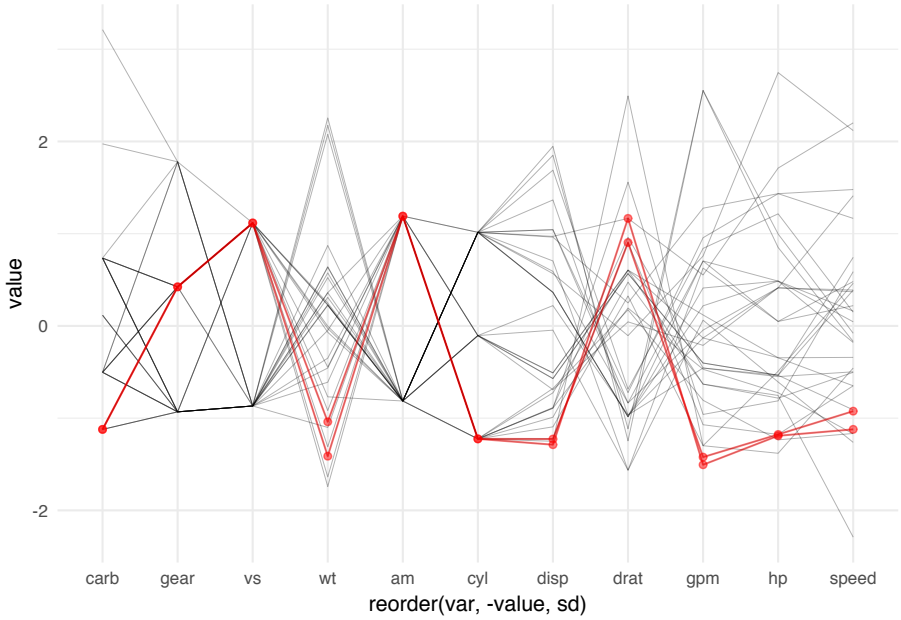
```
l.cancer.df = cancer.df %>% gather(key = year, value = rate, 2:5)

l.cancer.df$year = factor(l.cancer.df$year,
                          levels = c("Year 5", "Year 10", "Year 15", "Year 20"))

ggplot(l.cancer.df, aes(year, rate, group = Type))+
  geom_line(colour = "grey70") +
  geom_text_repel(data = l.cancer.df %>% filter(year == "Year 5"),
                  aes(label = Type), nudge_x = -.35, direction = "y",
                  point.padding = .02)+
  geom_text_repel(data = l.cancer.df %>% filter(year=="Year 20"),
                  aes(label = Type), nudge_x = .35,  direction = "y",
                  point.padding = .02)+
  geom_label(aes(label = rate), colour = "grey55", label.size = .02)+
  theme_void()+
  theme(axis.text = element_text(size = rel(.85)),
        axis.text.y=element_blank())
```



0.33.5 Parallel coordinate plot with similar items highlighted

Link to network for showing links in multivariate data

```
## [1] "Fiat 128"
## [1] "Toyota Corolla"
```



0.34 Gliphs: Chernof face and radar plots

Show patterns and outliers not precise comparisons

0

Proportion–Pie charts and pareto plots

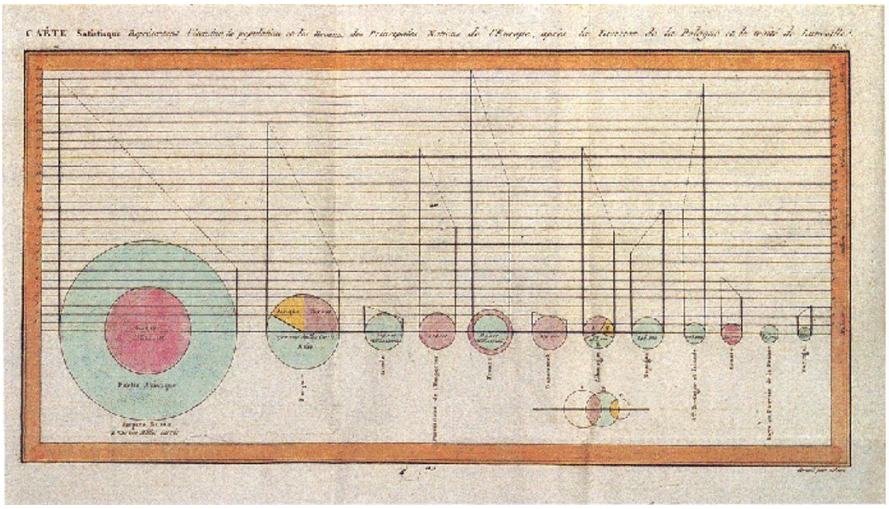
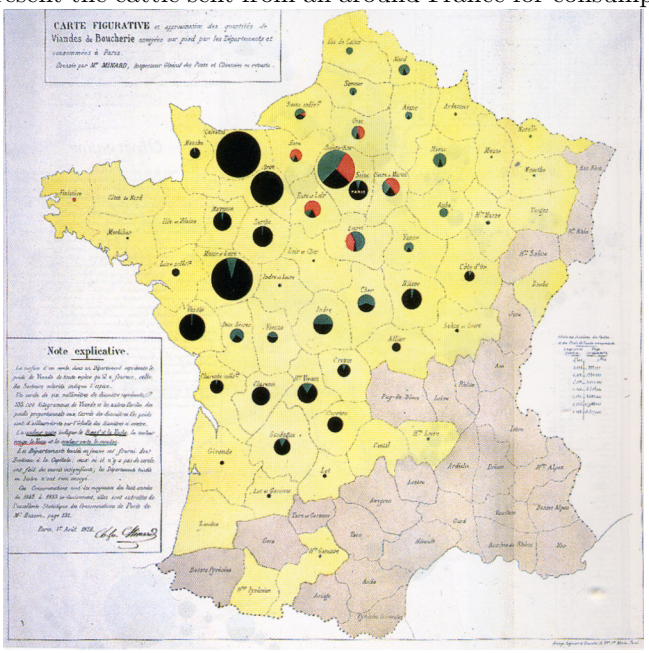


FIGURE 7: My picture

From wikipedia: “The French engineer Charles Joseph Minard was one of the first to use pie charts in 1858, in particular in maps. Minard’s map, 1858 used

pie charts to represent the cattle sent from all around France for consumption in Paris (1858)."



Other examples of Minard's work: https://cartographia.wordpress.com/category/charles-joseph-minard/

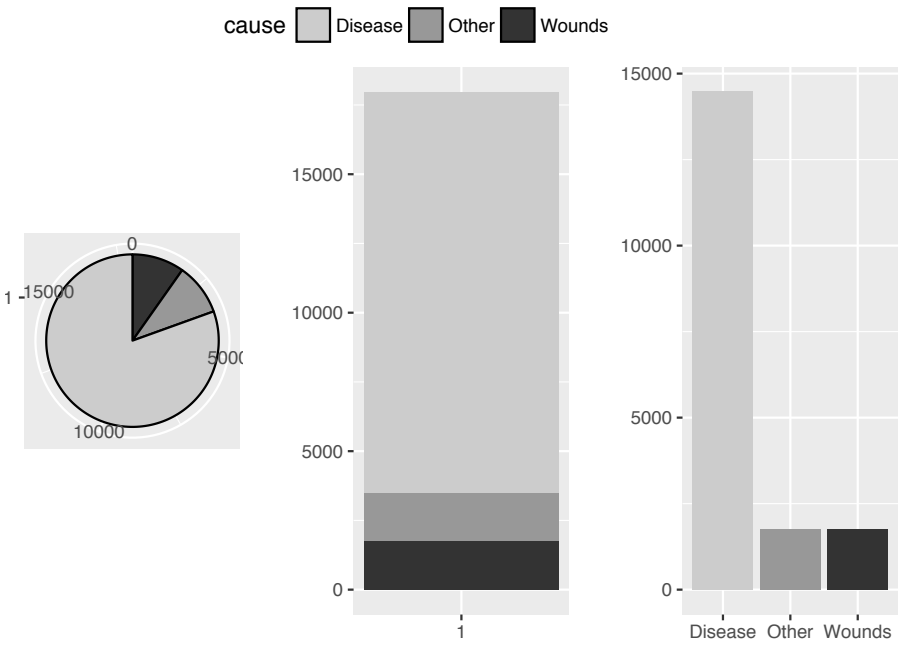
0.35 Pie and bar chart

Good pie chart: Few elements, directly labled, alpha for pie pieces

Small multiple for pie vs stacked bar likert ratings or time trends

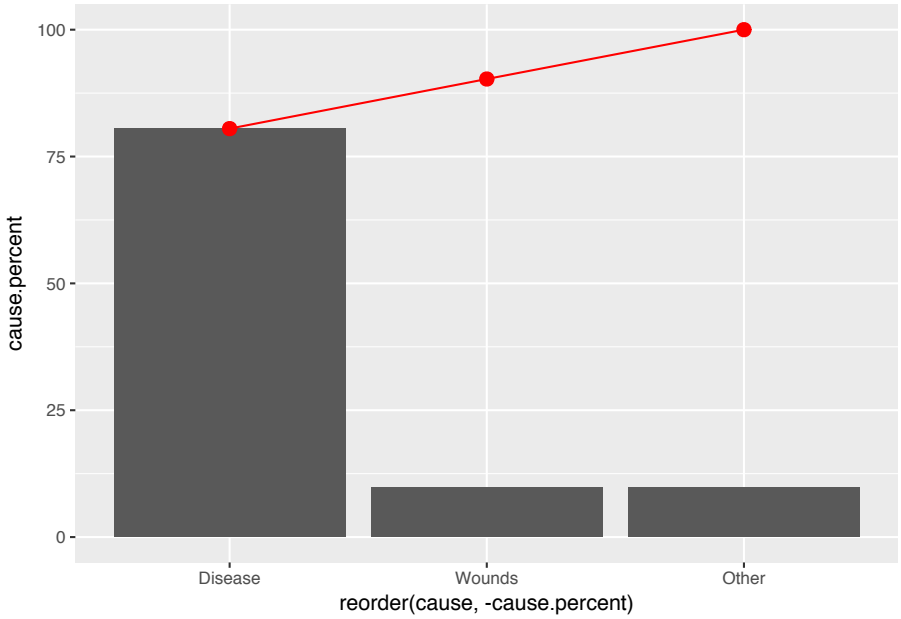
```
library(HistData)
library(tidyverse)

night.df = Nightingale %>%
  gather(key = cause, value = deaths, Disease:Other) %>%
  mutate(intervention = ordered( rep(c(rep('Before', 12), rep('After', 12)), 3), levels=c(
  group_by(intervention, Month, cause) %>%
  summarise(deaths = sum(deaths))

sum.night.df = night.df %>% group_by(cause) %>% summarise(deaths = sum(deaths))
```

```
# ## Statistic calculated internally
# ggplot(night.df, aes(cause, deaths)) +
#   geom_bar(stat="summary", fun.y = "sum")
#
# ## Same plot but with seperately calculated summary
# ggplot(sum.night.df) +
#   geom_bar(aes(reorder(cause, -cause.percent), cause.percent), stat = "identity")
#
# ## Horizontal bar
# ggplot(sum.night.df) +
#   geom_bar(aes(reorder(cause, -cause.percent), cause.percent), stat = "identity") +
#   coord_flip()



pie.plot = ggplot(sum.night.df, aes(x = factor(1), y = deaths, fill = cause)) +
  geom_bar(width = 1,  color="black", stat = "identity") +
  coord_polar(theta="y") +
  fill_palette(palette = "grey") +
  labs(x = "", y = "")

stacked.plot =  ggplot(sum.night.df, aes(x = factor(1), y=deaths, fill = cause))+
   geom_bar(stat = "identity", position = "stack") +
   fill_palette(palette = "grey") +
   labs(x = "", y = "")

dodged.plot =  ggplot(sum.night.df, aes(x =cause, y=deaths, fill = cause))+
  geom_bar(stat = "identity", position = "dodge") +
   fill_palette(palette = "grey") +
   labs(x = "", y = "")

deaths.plot = ggarrange(pie.plot, stacked.plot, dodged.plot,
                        nrow=1, ncol = 3, align = "hv", common.legend = TRUE)

deaths.plot
```

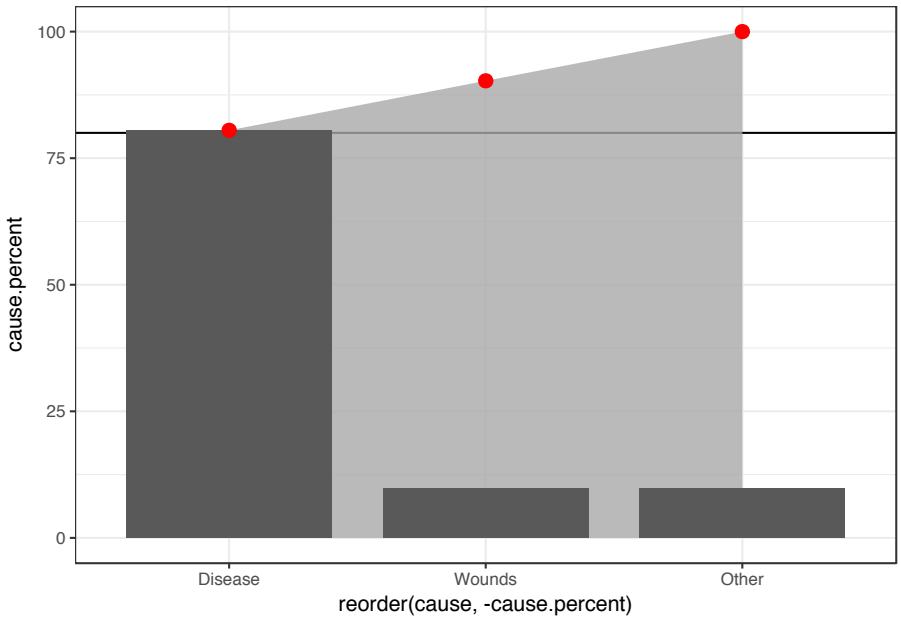


0.36 Pareto plot: Whole part and and ranking

```
## Calculate percent and cumulative percent

sum.night.df = night.df %>% ungroup() %>%
  mutate(total.deaths = sum(deaths)) %>% group_by(cause) %>%
  summarise(cause.percent = 100*sum(deaths)/max(total.deaths)) %>% ungroup() %>%
  arrange(-cause.percent) %>%
  mutate(cum.cause.percent = cumsum(cause.percent))


## Pareto plot: Individual and cummulative proportion
ggplot(sum.night.df) +
  geom_bar(aes(reorder(cause, -cause.percent), cause.percent), stat = "identity")+
  geom_point(aes(reorder(cause, -cause.percent), cum.cause.percent), colour = "red", size
  geom_line(aes(reorder(cause, -cause.percent), cum.cause.percent, group = 1), colour = "r
```



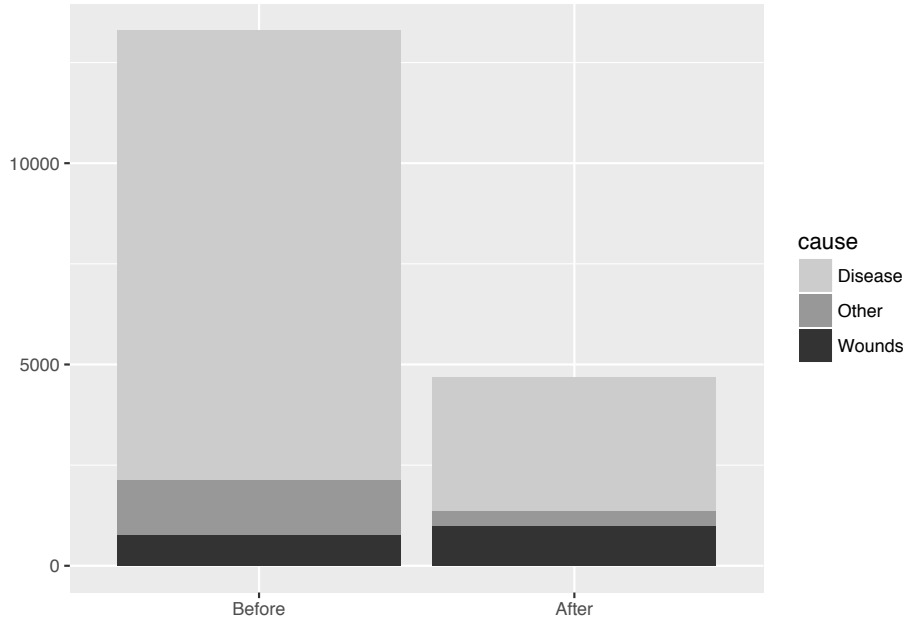

```
ggplot(data = sum.night.df) +
  geom_hline(yintercept = 80) +
  geom_ribbon(aes(reorder(cause, -cause.percent),
                  ymin = 0, ymax = cum.cause.percent, group = 1), fill = "darkgrey", alpha
  geom_bar(aes(reorder(cause, -cause.percent), cause.percent), stat = "identity", width =
  geom_point(aes(reorder(cause, -cause.percent), cum.cause.percent), size = 3, colour = "r
  theme_bw()
```



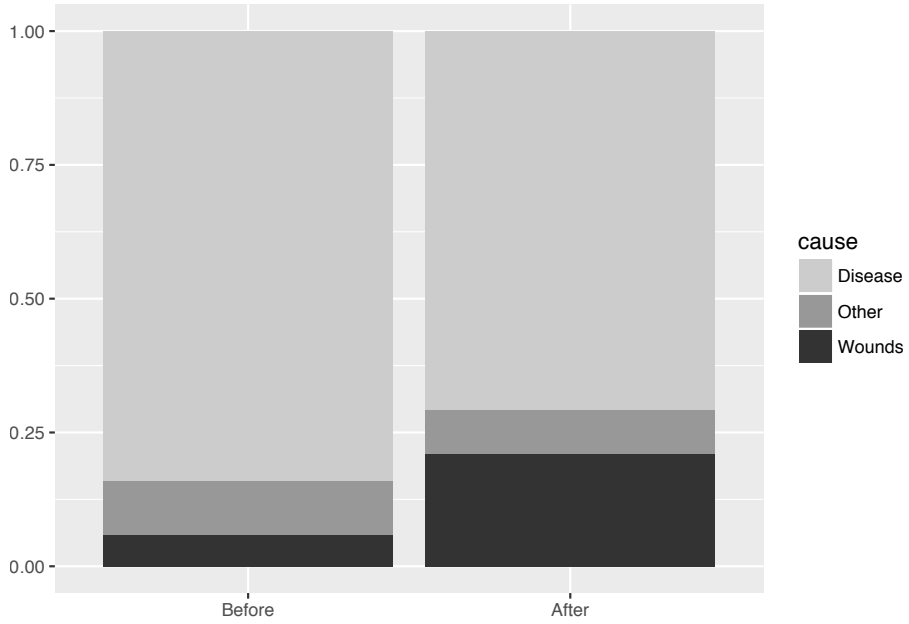

```
# ## Pareto plot
# mtcars.df = mtcars
# sum.mtcars.df = mtcars.df %>% ungroup() %>%
#   mutate(total.n = n()) %>% group_by(gear) %>%
#   summarise(gear.percent = 100*max(n())/max(total.n)) %>% ungroup() %>%
#   arrange(-gear.percent) %>%
#   mutate(cum.gear.percent = cumsum(gear.percent))

# ggplot(data = sum.mtcars.df) +
#   geom_hline(yintercept = 80) +
#   geom_ribbon(aes(reorder(gear, -gear.percent),
#                   ymin = 0, ymax = cum.gear.percent, group = 1), fill = "darkgrey", alph
#   geom_bar(aes(reorder(gear, -gear.percent), gear.percent), stat = "identity", width = .
#   geom_point(aes(reorder(gear, -gear.percent), cum.gear.percent), size = 3, colour = "re
```

0.37 Stacked bar chart

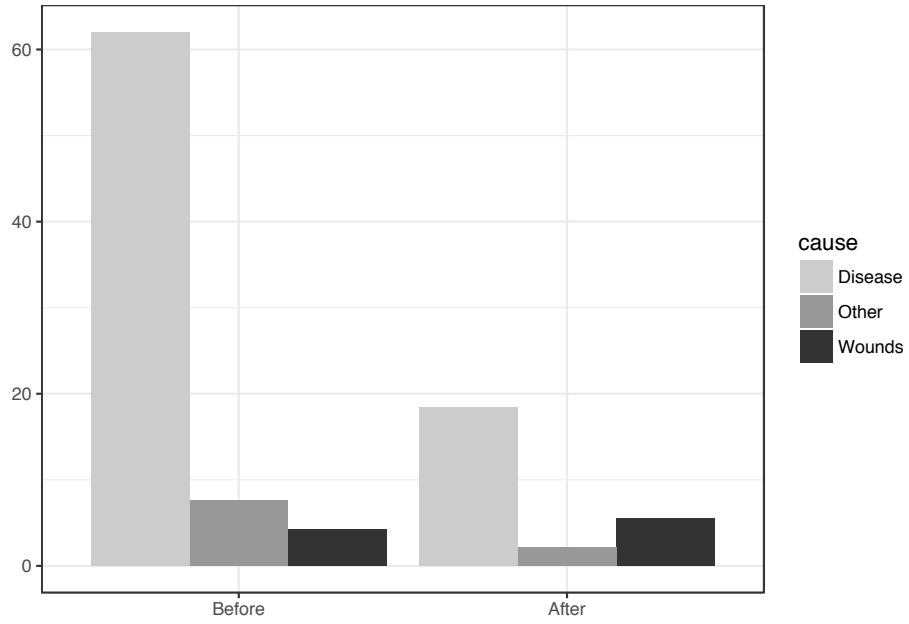
```
sum.night.df = night.df %>% ungroup() %>%
  mutate(total.deaths = sum(deaths)) %>% group_by(cause, intervention) %>%
  summarise(cause.percent = 100*sum(deaths)/max(total.deaths),
            deaths = sum(deaths))

## Stacked bar with count: Shows data directly
ggplot(sum.night.df, aes(intervention, deaths, fill = cause)) +
    geom_bar(position = "stack", stat = "identity") +
   fill_palette(palette = "grey") +
   labs(x = "", y = "")
```




```
## Stacked bar with proportion: Abstracts to proportion
ggplot(sum.night.df, aes(intervention, cause.percent, fill = cause))+
    geom_bar(position = "fill", stat = "identity") +
   fill_palette(palette = "grey") +
   labs(x = "", y = "")
```



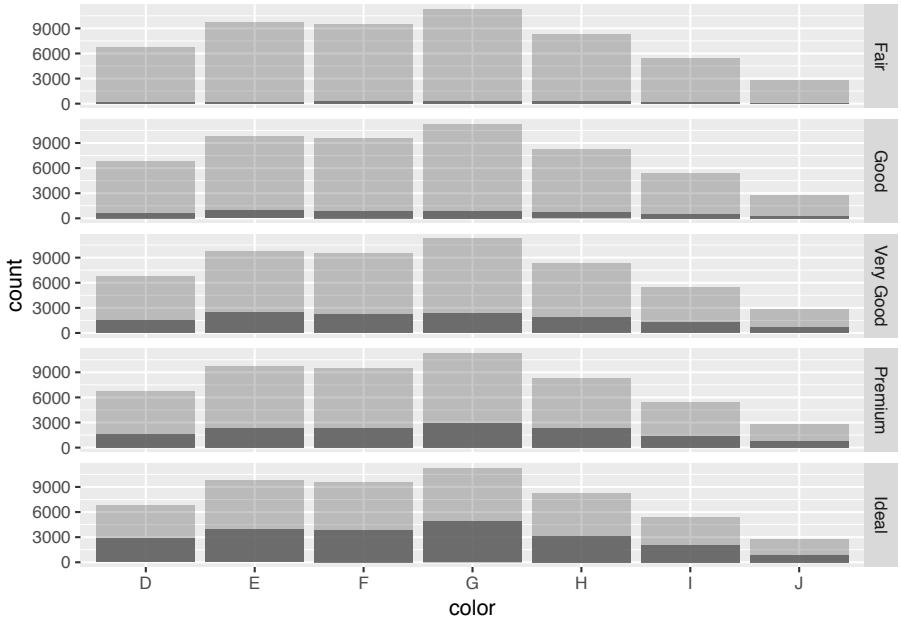

```
ggplot(sum.night.df, aes(intervention, cause.percent, fill = cause))+
    geom_bar(position = "dodge", stat = "identity") +
   fill_palette(palette = "grey") +
   labs(x = "", y = "")+theme_bw()
```



0.38 Faceted Bar chart with overall reference distribution

The grey bars in the background represent the overall distribution and provide a referende for each of the marginal distributions.

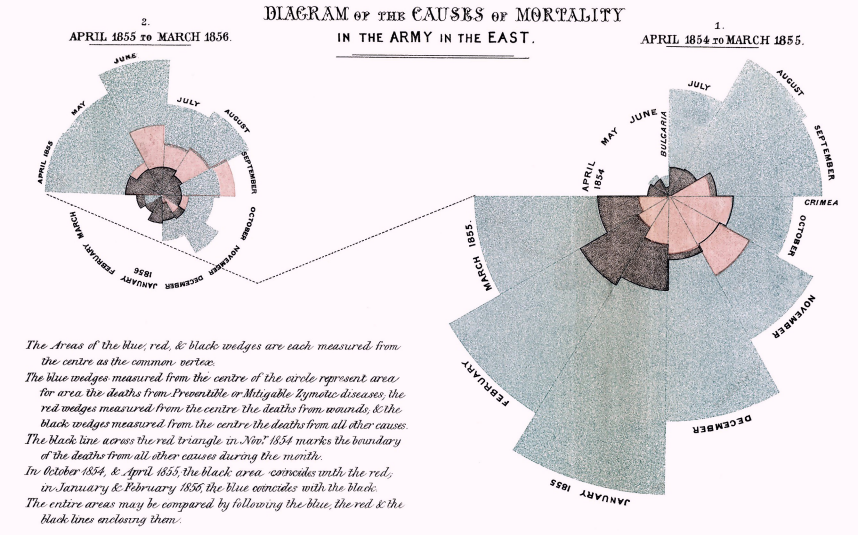
```
diamonds.df = diamonds

count.diamonds.df = diamonds.df %>% group_by(cut, color) %>% summarise(count = n()) %>%
  ungroup() %>% group_by(color) %>% mutate(color.count = sum(count))

ggplot(count.diamonds.df, aes(color, count)) +
  geom_bar(aes(color, color.count), stat = "identity", alpha =.33) +
  geom_bar(stat = "identity", alpha = .8) +
  facet_grid(cut~.)
```



0.39 Rose or Coxcomb plots

Nightingale produced a graph "Diagram of the Causes of Mortality in the Army in the East" that showed that most soldiers during the Crimean war died of disease rather than wounds. Improving hygiene in March of 1855 led to fewer disease related deaths.

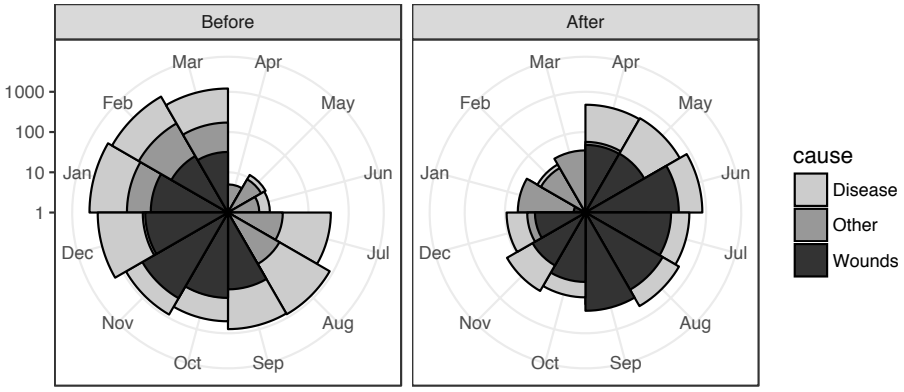
This "Diagram of the causes of mortality in the army in the East" was published in Notes on Matters Affecting the Health, Efficiency, and Hospital Administration of the British Army and sent to Queen Victoria in 1858.



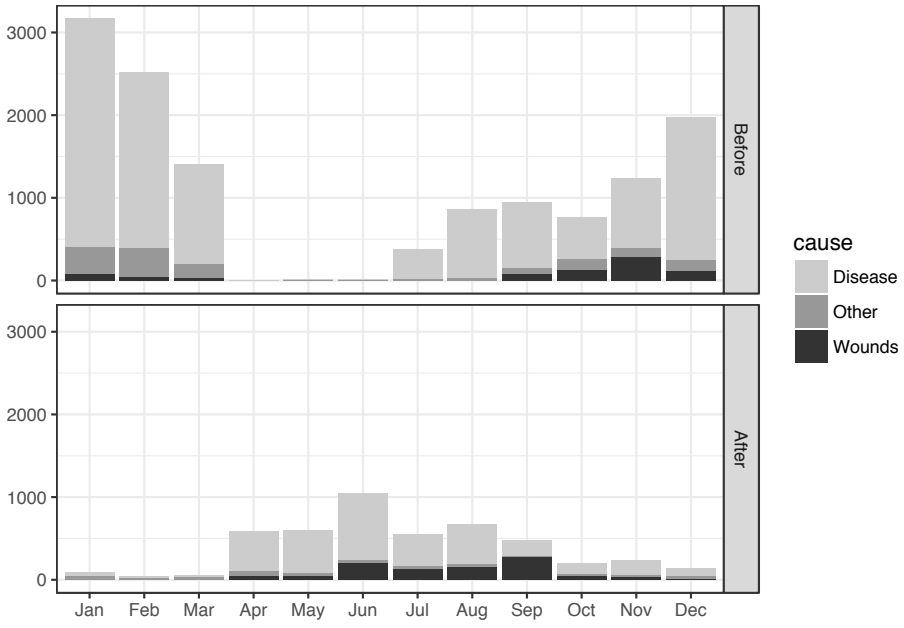
Coxcombe plot diminishes small values and requires square root transform

```
ggplot(night.df, aes(x = Month, y = deaths, fill = cause)) +
  geom_bar(width = 1, position = "identity", color="black", stat = "identity") +
  scale_y_log10() +
  coord_polar(start=3*pi/2)  +
  facet_grid(.~intervention)  +
  fill_palette(palette = "grey") +
  labs(x = "", y = "") +
  theme_bw()
```

```
## Warning: Transformation introduced infinite values in
## continuous y-axis
```



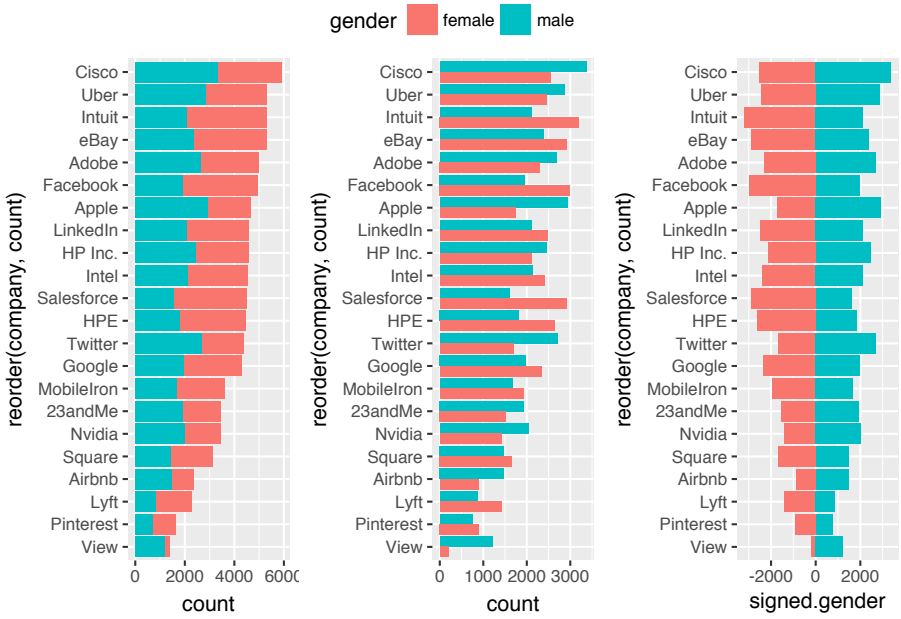

```
ggplot(night.df, aes(Month, deaths, fill = cause)) +
  geom_bar(stat = "identity") +
  facet_grid(intervention~.) +
   fill_palette(palette = "grey") +
   labs(x = "", y = "")+
  theme_bw()
```



0.40 Stacked, dodged and opposed bar chart

Comparison of many categories

- Stacked makes grouping easy
- Dodge makes comparison easy with common axis and relative judgment
- Opposing makes gender more apparent

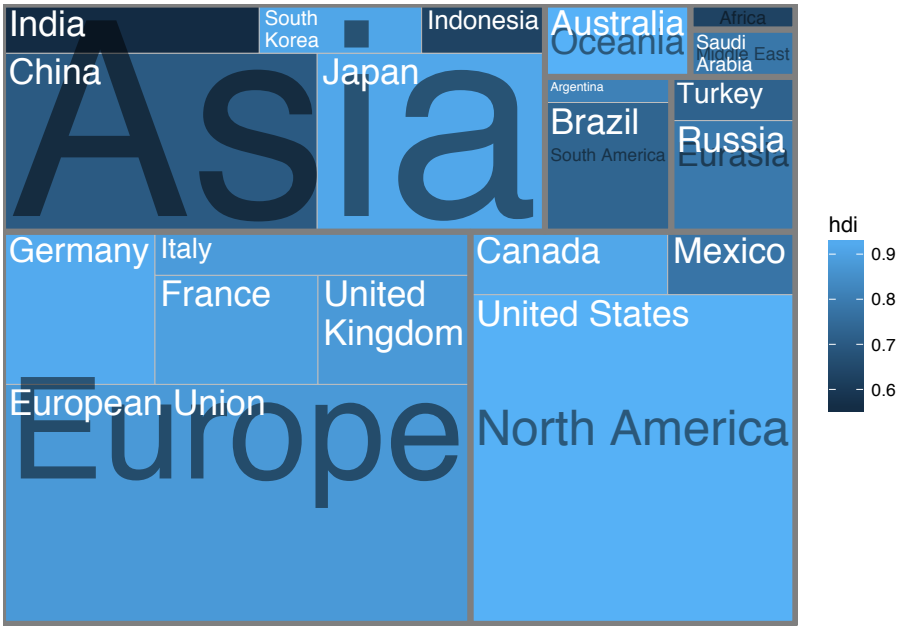


0.41 Treemaps for whole-part of hierarchy

Schneiderman

```
library(treemapify)

ggplot(G20, ggplot2::aes(area = gdp_mil_usd, fill = hdi, label = country, subgroup = regio
  treemapify::geom_treemap() +
  geom_treemap_subgroup_border() +
  geom_treemap_subgroup_text(place = "centre", grow = T, alpha = 0.5, colour =
```

```
                          "black", fontface = "italic", min.size = 0) +
  geom_treemap_text(colour = "white", place = "topleft", reflow = T)
```



0.42 Circle packing

The most valuable graphical dimensions of x and y position are wasted in this plot becaue they have no meaning, but it can be engaging

```
library(packcircles)
library(viridis)
```

```
## Warning: package 'viridis' was built under R version
## 3.4.4
```

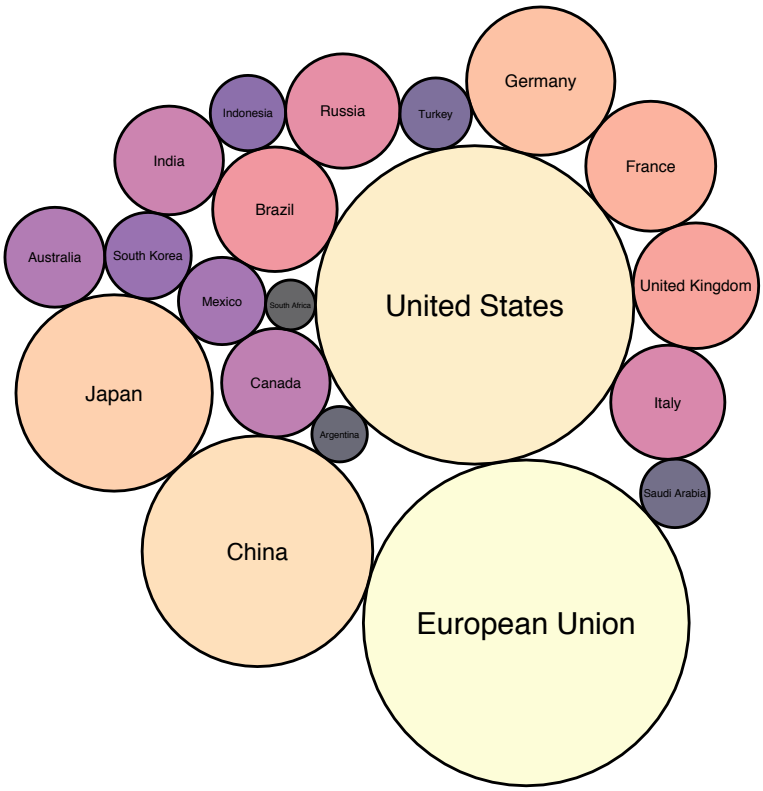
```
## Loading required package: viridisLite
```

```
##
## Attaching package: 'viridis'
```

```
## The following object is masked from 'package:scales':
##
```

```
##     viridis_pal
```

```
library(tidyverse)

# Show with radius vs area
packing <- circleProgressiveLayout(G20$gdp_mil_usd, sizetype='area')
G20.df = cbind(G20, packing)
layout = G20.df %>%
  dplyr::select(country, x, y, radius)

dat.pack <- circleLayoutVertices(layout, npoints=60, idcol = 1, xysizecols=2:4, sizetype =

dat.pack = left_join(dat.pack, G20.df, by = c("id" = "country"))

ggplot() +
  geom_polygon(data = dat.pack, aes(x.x, y.x, group = id, fill=as.factor(gdp_mil_usd)), co
  geom_text(data = G20.df, aes(x, y, size=gdp_mil_usd, label = country)) +
  scale_fill_manual(values = magma(nrow(G20.df))) +
  scale_size_continuous(range = c(1, 4)) +
  theme_void() +
  theme(legend.position="none") +
  coord_equal()
```


Germany
Indonesia
Russia
Turkey
India
France
Brazil
Australia
South Korea
United States
United Kingdom
Mexico
South Africa
Japan
Canada
Italy
Argentina
Saudi Arabia
China
European Union

0

Fluctuation–timelines

```
library(tidyverse)
```

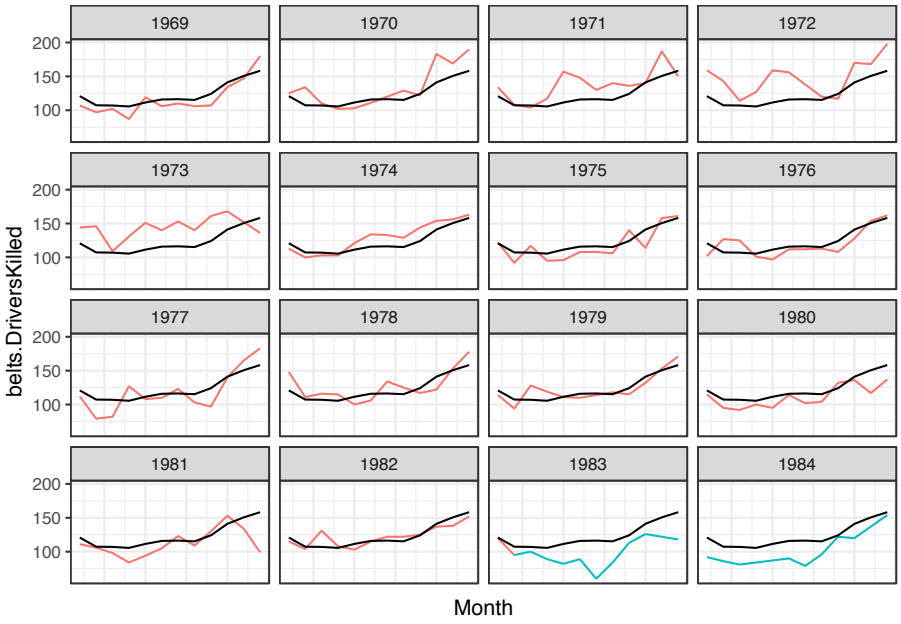
0.43 Multiple time series

two lines on one plot and problems faceting

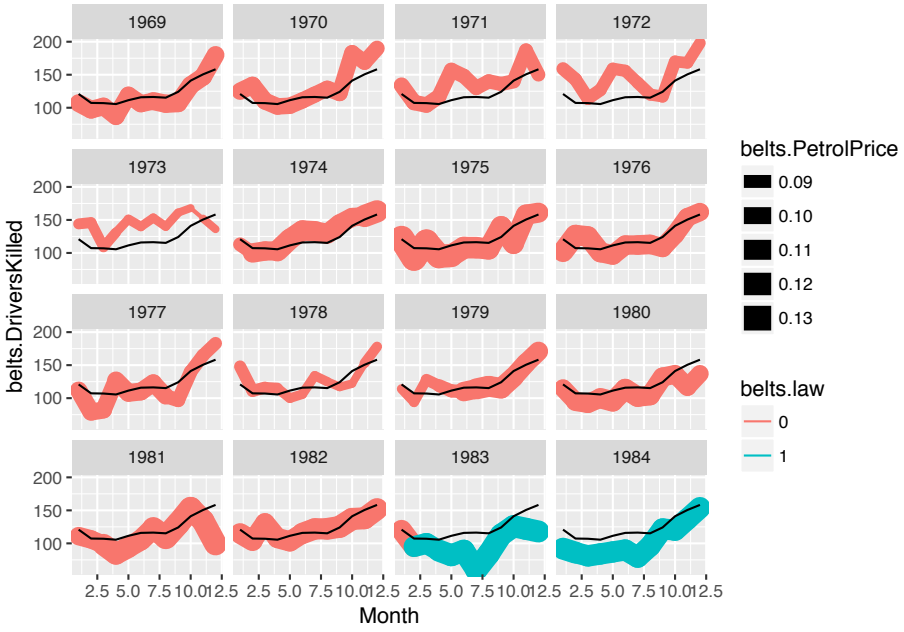
0.44 Time series with reference line

```
## Reference lines in time series
belts = Seatbelts
belts.df = as.data.frame(
  cbind(Year = round(trunc(time(belts)), 1),
        Month = cycle(belts),
        belts))

belts.df$belts.law = as.factor(belts.df$belts.law)
belts.DriversKilled.bymonth = belts.df %>% group_by(Month) %>%
  summarise(mean.DriversKilled = mean(belts.DriversKilled))

ggplot(belts.df, aes(x = Month, y = belts.DriversKilled)) +
  geom_line(aes(colour = belts.law, group = Year)) +
  geom_line(data = belts.DriversKilled.bymonth,
            aes(x = Month, y = mean.DriversKilled)) +
  facet_wrap(~Year, nrow= 4, ncol= 4) +
```

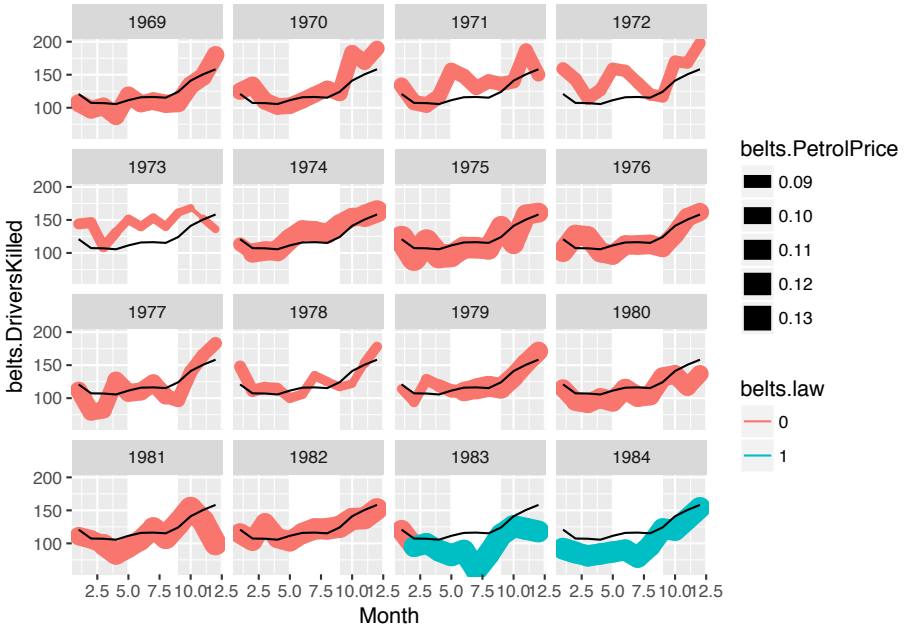
```
  theme_bw() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = "none")
```




```
ggplot(belts.df, aes(Month, belts.DriversKilled)) +
    geom_path(aes(colour = belts.law, size = belts.PetrolPrice, group = Year), lineend = "
    geom_line(data = belts.DriversKilled.bymonth,  aes(x = Month, y = mean.DriversKilled))
  facet_wrap(~Year, nrow= 4, ncol= 4) # Creates a matrix of graphs
```



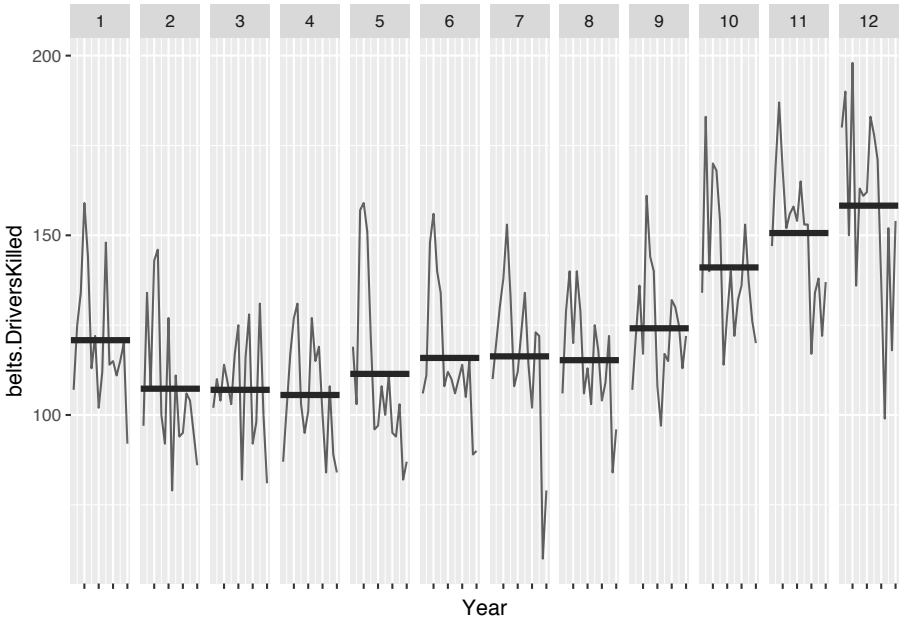

```
belts.df = belts.df %>% group_by(Year) %>% mutate(summer.s=Month[Month==5], summer.e=Month

ggplot(belts.df, aes(Month, belts.DriversKilled)) +
    geom_rect(aes(xmin=summer.s, xmax=summer.e, ymin=-Inf, ymax=+Inf),
                  fill = "white") +
  geom_path(aes(colour = belts.law, size = belts.PetrolPrice, group = Year), lineend = "ro
    geom_line(data = belts.DriversKilled.bymonth,  aes(x = Month, y = mean.DriversKilled))
  facet_wrap(~Year, nrow= 4, ncol= 4)
```



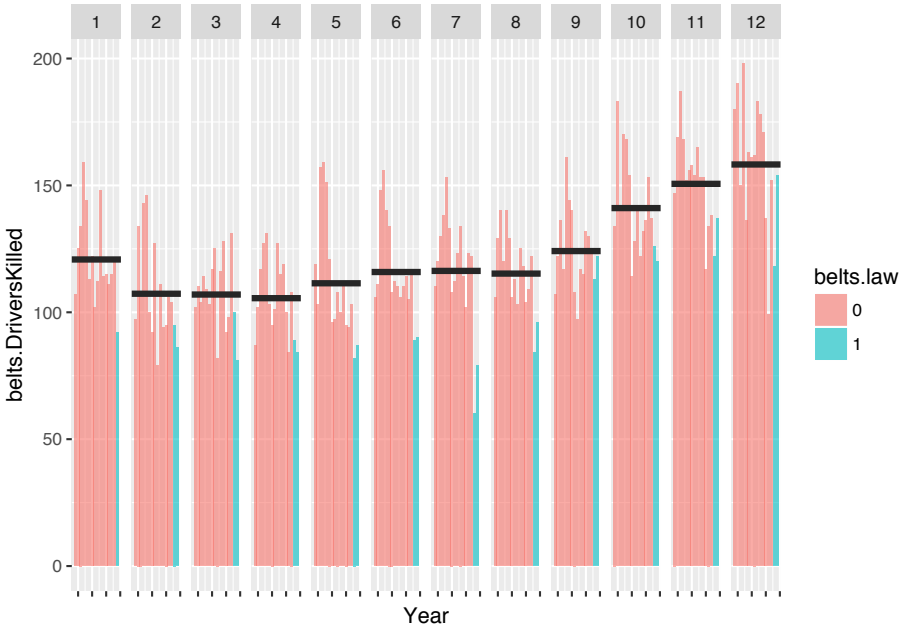
0.45 Cycle plot

Cycle plot make comparions between months easy. Showing bars rather than lines helps focus attenton the specitic years when the seatbelt law was enacted.

```
hline.df <- belts.df %>% group_by(Month) %>% summarize(m.killed = mean(belts.DriversKilled

ggplot() +
  geom_line(data = belts.df, aes(x = Year, y = belts.DriversKilled, group = Month), alpha
  geom_hline( data = hline.df, aes(yintercept = m.killed), colour = "grey15", size = 1.5)
  facet_grid(~Month) +
  theme(axis.text.x = element_blank())
```




```
ggplot() +
  geom_bar(data = belts.df, aes(x = Year, y = belts.DriversKilled, group = Month, fill = b
    stat = "identity") +
  geom_hline( data = hline.df, aes(yintercept = m.killed), colour = "grey15", size = 1.5)
  facet_grid(~Month) +
  theme(axis.text.x = element_blank())
```

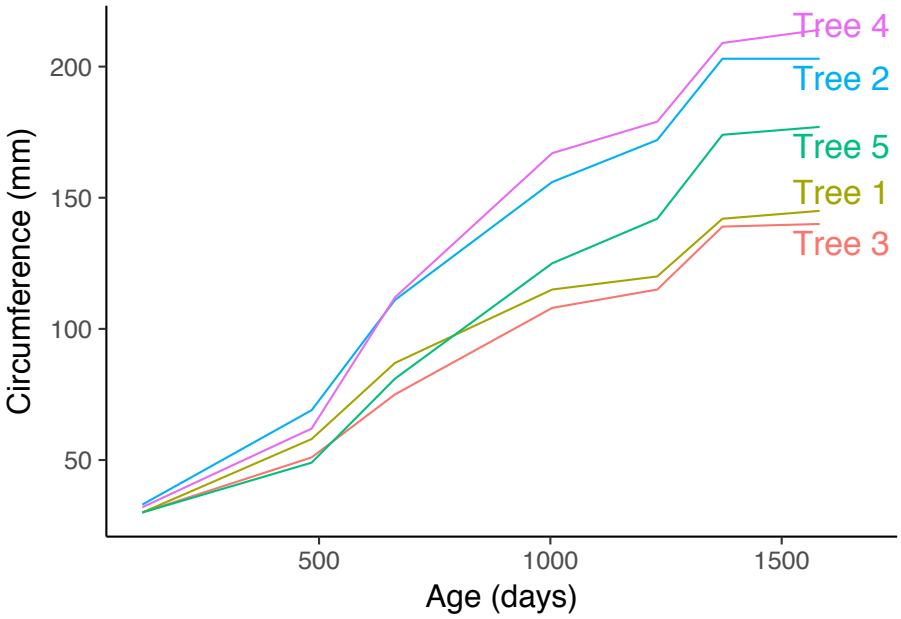


0.46 Time series with lines labeled

Based on example from: https://cran.r-project.org/web/packages/ggrepel/vignettes/ggrepel.html

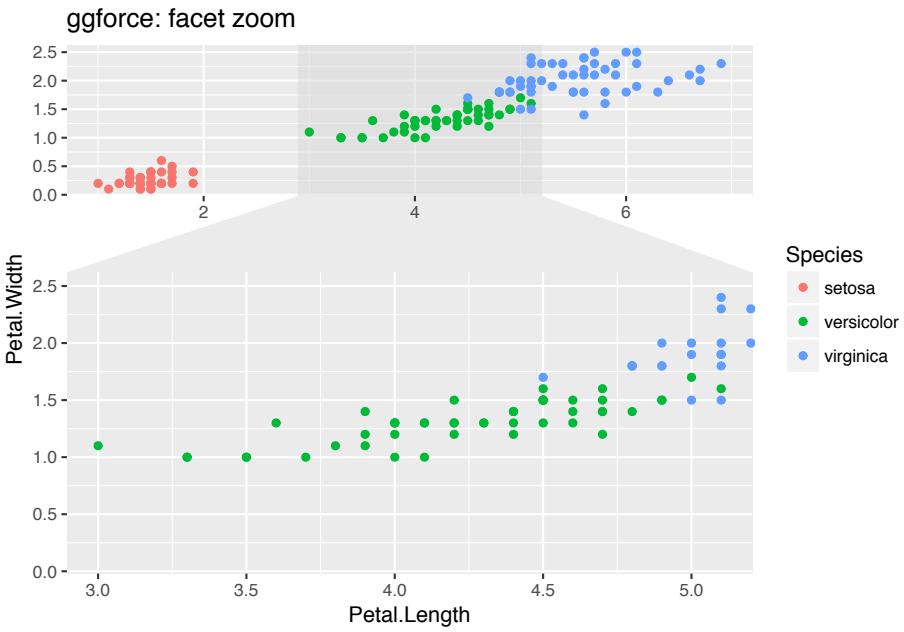
```
library(ggrepel)

## Lines with labels rather than legend
ggplot(Orange, aes(age, circumference, color = Tree)) +
  geom_line() +
  coord_cartesian(xlim = c(min(Orange$age), max(Orange$age) + 90)) +
  geom_text_repel(
    data = subset(Orange, age == max(age)),
    aes(label = paste("Tree", Tree)),
    size = 6,
    nudge_x = 45,
    segment.color = NA
  ) +
  theme_classic(base_size = 16) +
```

```
  theme(legend.position = "none") +
  labs(x = "Age (days)", y = "Circumference (mm)")
```



0.47 Faceted zoom

```
library(ggforce)
## Examples from: https://cran.r-project.org/web/packages/ggforce/vignettes/Visual_Guide.h

ggplot(iris, aes(Petal.Length, Petal.Width, colour = Species)) +
    geom_point() +
    facet_zoom(x = Species == "versicolor")+
    labs(title = "ggforce: facet zoom")
```



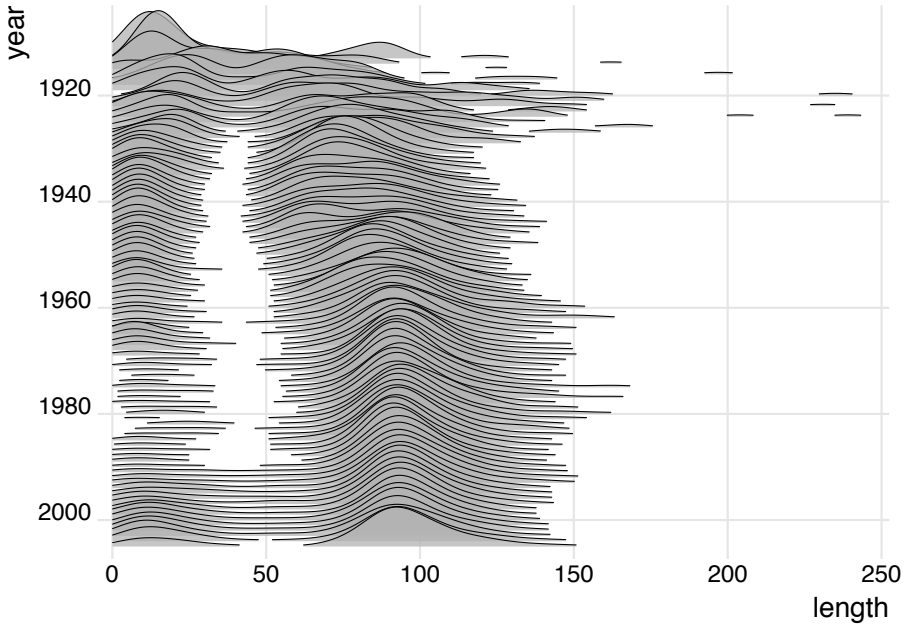
TODO Step graph

0.48 Ridge plot

https://cran.r-project.org/web/packages/ggridges/vignettes/gallery.html

```
## Ridge plot
library(ggridges)
library(ggplot2movies)

movies %>% filter(year>1912, length<250) %>%
ggplot(aes(x = length, y = year, group = year)) +
  geom_density_ridges(scale = 10, size = 0.25, rel_min_height = 0.03, alpha=.75) +
  scale_x_continuous(limits=c(0, 250), expand = c(0.01, 0)) +
  scale_y_reverse(breaks=c(2000, 1980, 1960, 1940, 1920, 1900), expand = c(0.01, 0)) +
  theme_ridges()
```

```
## Picking joint bandwidth of 6.89
```



0.49 Stacked area and line graphs

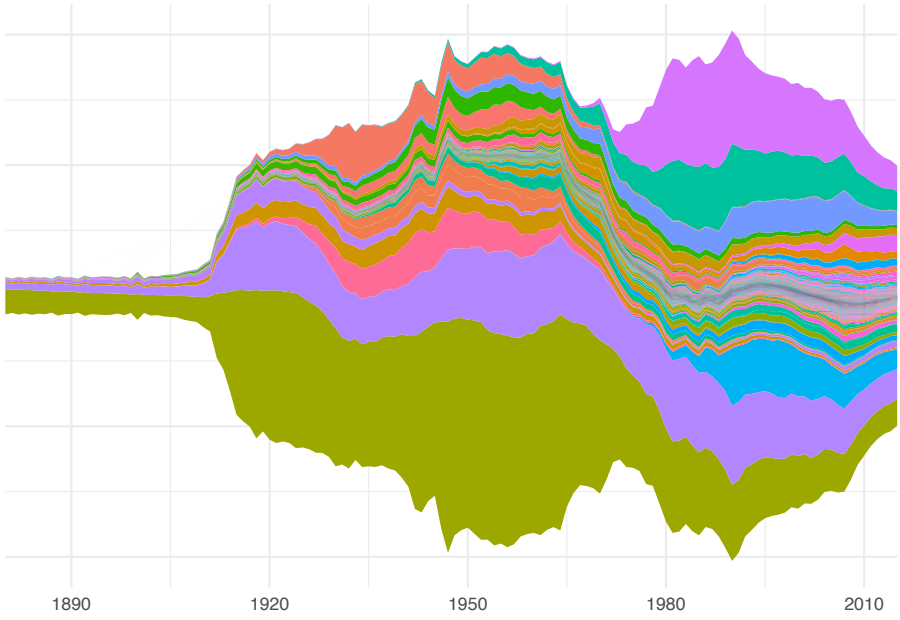
Challenges of comparing individual contributions, ease of seeing combined effect Stream plot

"Streamgraphs are a generalization of stacked area graphs where the baseline is free. By shifting the baseline, it is possible to minimize the change in slope (or wiggle) in individual series, thereby making it easier to perceive the thickness of any given layer across the data. Byron & Wattenberg describe several streamgraph algorithms in 'Stacked Graphs—Geometry & Aesthetics[1]'."[2]

"A steamgraph is a more aesthetically appealing version of a stacked area chart. It tries to highlight the changes in the data by placing the groups with the most variance on the edges, and the groups with the least variance towards the centre. This feature in conjunction with the centred alignment of each of the contributing areas makes it easier for the viewer to compare the contribution of any of the components across time."

[1] http://www.leebyron.com/else/streamgraph/
[2] Bostock. http://bl.ocks.org/mbostock/4060954

```
#devtools::install_github('Ather-Energy/ggTimeSeries')
library(babynames)
library(ggTimeSeries)

names.df = babynames %>%
  filter(grepl("^Jo", name)) %>%
  group_by(year, name) %>%
  tally(wt=n)
##TODO smooth sequence

ggplot(names.df, aes(year, y = nn, group = name, fill = name)) +
  stat_steamgraph() +
  labs(x="", y = "") +
  scale_x_continuous(expand = c(0, 0)) +
  theme_minimal() +
  theme(legend.position = "none",
        axis.text.y=element_blank())
```



0.50 Temporal heatmap

TODO replace with rain data for SEA from https://www.r-bloggers.com/ggplot2-time-series-heatmaps-revisited-in-the-tidyverse/

```
# The core idea is to transform the data such that one can
# plot "Value" as a function of "WeekOfMonth" versus "DayOfWeek"
# and facet this Year versus Month

xts_heatmap <- function(x){
  data.frame(Date=as.Date(index(x)), x[,1]) %>%
    setNames(c("Date","Value")) %>%
    dplyr::mutate(
      Year=lubridate::year(Date),
      Month=lubridate::month(Date),
      # I use factors here to get plot ordering in the right order
      # without worrying about locale
      MonthTag=factor(Month,levels=as.character(1:12),
                      labels=c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct"
      # week start on Monday in my world
      Wday=lubridate::wday(Date,week_start=1),
      # the rev reverse here is just for the plotting order
      WdayTag=factor(Wday,levels=rev(1:7),labels=rev(c("Mon","Tue","Wed","Thu","Fri","Sat"
      Week=as.numeric(format(Date,"%W"))
    ) %>%
    # ok here we group by year and month and then calculate the week of the month
    # we are currently in
    dplyr::group_by(Year,Month) %>%
    dplyr::mutate(Wmonth=1+Week-min(Week)) %>%
    dplyr::ungroup() %>%
    ggplot(aes(x=Wmonth, y=WdayTag, fill = Value)) +
    geom_tile(colour = "white") +
    facet_grid(Year~MonthTag) +
    scale_fill_gradient(low="red", high="yellow") +
    labs(x="Week of Month", y=NULL)
}

require(quantmod)
```

```
## Loading required package: quantmod
```

```
## Loading required package: xts
```

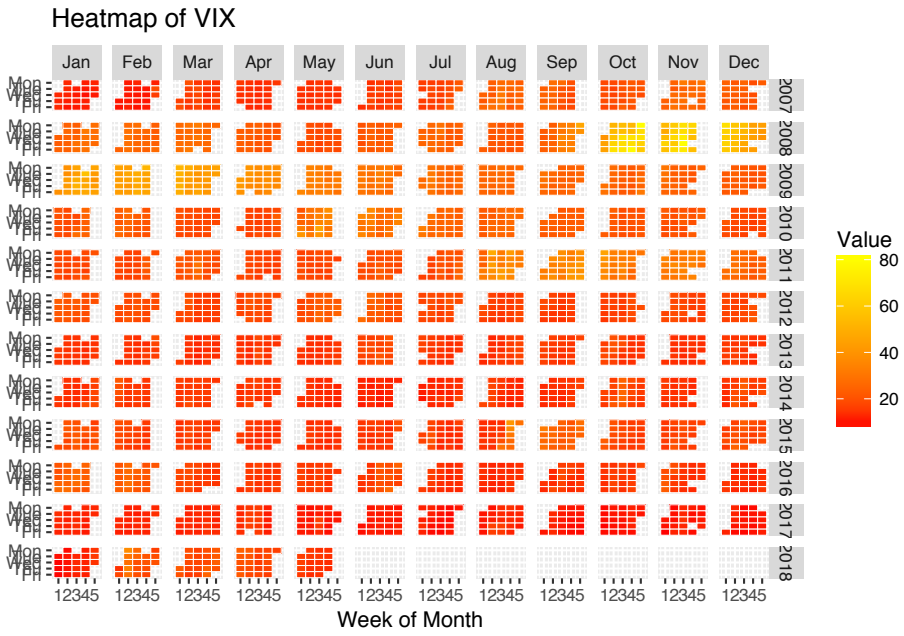
```
## Warning: package 'xts' was built under R version 3.4.4
```

```
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Attaching package: 'xts'
## The following objects are masked from 'package:dplyr':
##
##     first, last
## Loading required package: TTR
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```
# Download some Data, e.g. the CBOE VIX
quantmod::getSymbols("^VIX",src="yahoo")
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
##
## WARNING: There have been significant changes to Yahoo Finance data.
## Please see the Warning section of '?getSymbols.yahoo' for details.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.yahoo.warning"=FALSE).
## [1] "VIX"
```

```
# lets see
xts_heatmap(Cl(VIX)) + labs(title="Heatmap of VIX")
```


Heatmap of VIX
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
Value
80
60
40
20
12345
Week of Month

```
# ok
```

0

Connection–maps and network plots

Test of examples

```
library(tidyverse)
```

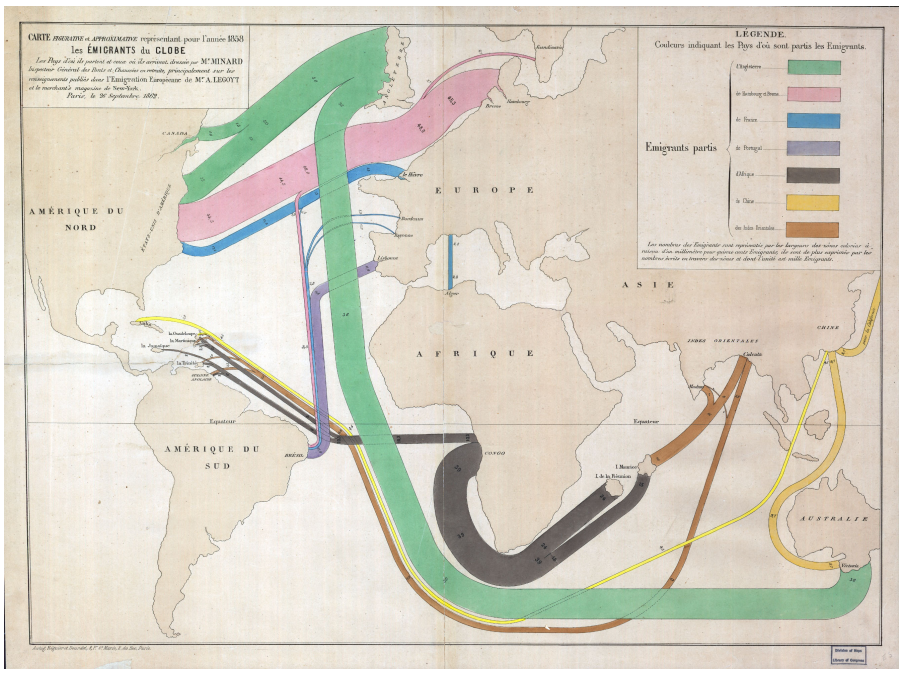


FIGURE 8: My picture

0.51 Maps

```
library(tidyverse)
library(ggalt)
```

0.51.1 Choropleth

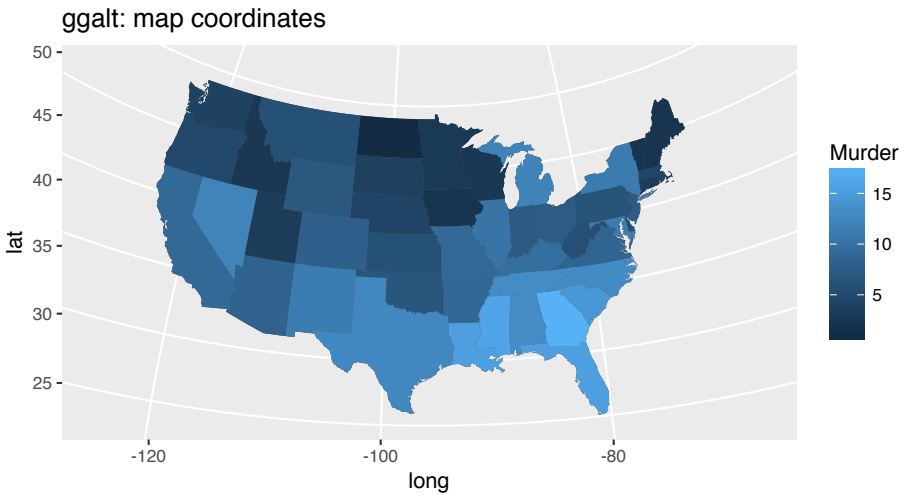
```
crimes.df <- data.frame(state = tolower(rownames(USArrests)), USArrests)

crimes.l.df <- gather(crimes.df, key = type, value = rate, -state)

states_map <- map_data("state")
```

```
## Warning: package 'maps' was built under R version 3.4.4
```

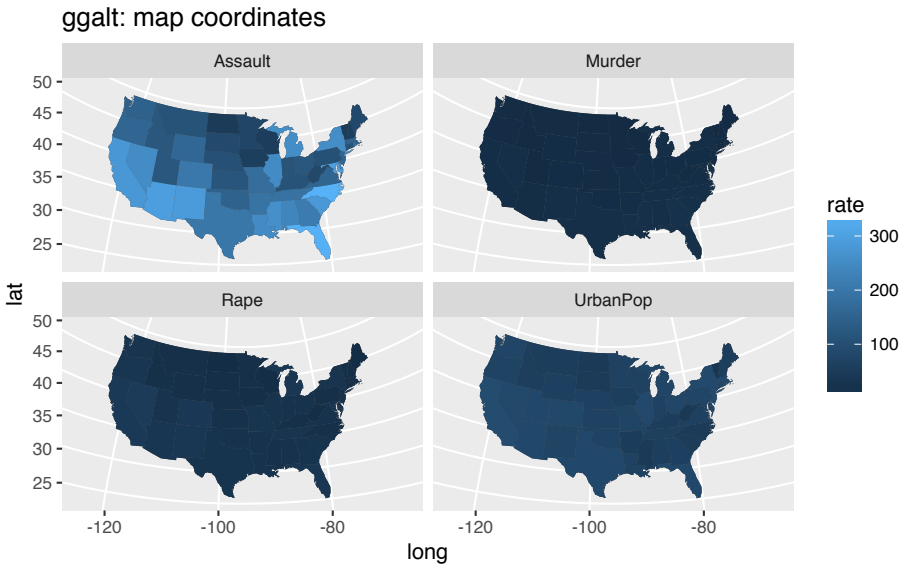
```
##
## Attaching package: 'maps'
```

```
## The following object is masked from 'package:purrr':
##
##     map
```

```
ggplot() +
    geom_cartogram(data=states_map, aes(long, lat, map_id = region), map=states_map) +
    geom_cartogram(data=crimes.df, aes(fill = Murder, map_id = state), map=states_map) +
    coord_map("polyconic")+
    labs(title = "ggalt: map coordinates")
```

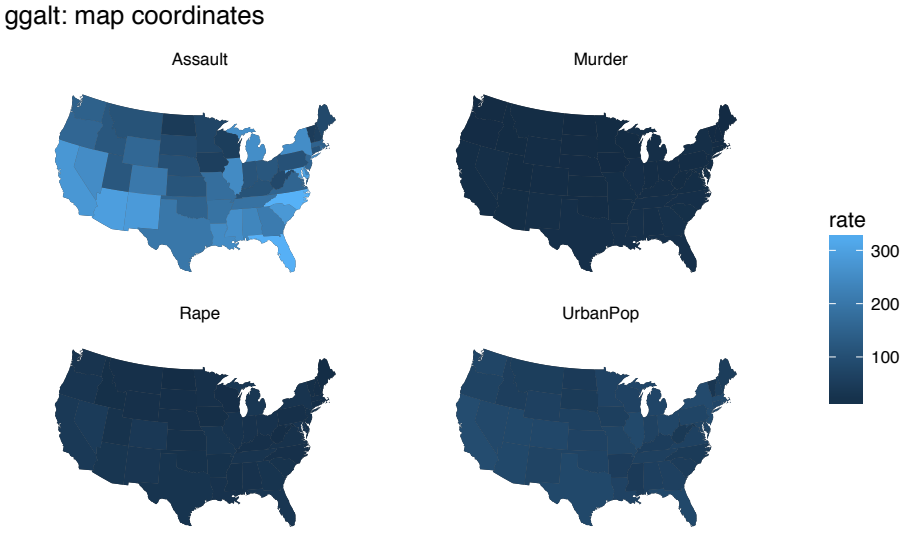



```
ggplot() +
    geom_cartogram( data=states_map, aes(long, lat, map_id=region), map = states_map) +
    geom_cartogram(data=crimes.l.df, aes(fill = rate, map_id=state), map = states_map) +
    coord_map("polyconic") +
    facet_wrap( ~ type) +
    labs(title = "ggalt: map coordinates")
```




```
ggplot() +
    geom_cartogram( data=states_map, aes(long, lat, map_id=region), map = states_map) +
```

```
    geom_cartogram(data=crimes.l.df, aes(fill = rate, map_id=state), map = states_map) +
    coord_map("polyconic") +
    facet_wrap( ~ type) +
    labs(title = "ggalt: map coordinates") +
    theme_void()
```



0.51.2 County map

```
## devtools::install_github("hrbrmstr/ggalt", force=TRUE) # To install development version
## Examples from https://github.com/hrbrmstr/ggalt

# TODO replace with poverty or health outcomes data
library(tidyverse)
library(stringr)
library(viridis)


# From https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html

## Read and clean data
unemp.df = read_csv("http://datasets.flowingdata.com/unemployment09.csv")
```

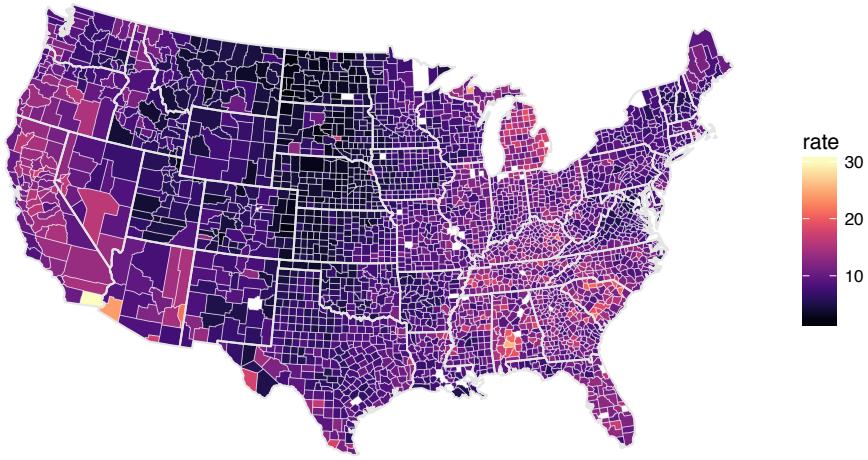
```
## Parsed with column specification:
## cols(
```

```
##   CN010010 = col_character(),
##   `01` = col_character(),
##   `001` = col_character(),
##   `Autauga County, AL` = col_character(),
##   `2009` = col_integer(),
##   `23,288` = col_number(),
##   `21,025` = col_number(),
##   `2,263` = col_number(),
##   `9.7` = col_double()
## )
```

```
save(unemp.df, file = "unemp")

load("unemp")

names(unemp.df) = c("id", "state_fips", "county_fips", "name", "year",
                    "--", "---", "---", "rate")
unemp.df$county = tolower(str_replace(unemp.df$name, " County, [A-Z]{2}", ""))

#unemp.df$county <- tolower(gsub(" County, [A-Z]{2}", "", unemp.df$name))
unemp.df$county = str_replace(unemp.df$county,"^(.*) parish, ..$","\\1")
unemp.df$state = str_replace(unemp.df$name, "^.*([A-Z]{2}).*$", "\\1")

## Use the maps package to convert maps data to a data frame
# "county" is a county map of the US
county.df <- map_data("county", projection = "albers", parameters = c(39, 45))

names(county.df) <- c("long", "lat", "group", "order", "state_name", "county")

state.df <- map_data("state", projection = "albers", parameters = c(39, 45))

## Replace state name with state abbreviations
county.df$state <- state.abb[match(county.df$state_name, tolower(state.name))]
county.df$state_name <- NULL

## Merge county and state shape information with unemployment data
choropleth.df <- merge(county.df, unemp.df, by = c("state", "county"))
#choropleth.df = county.df %>% inner_join(unemp.df, by = c("state", "county"))


choropleth.df = choropleth.df %>% dplyr::select(state:rate)
choropleth.df = choropleth.df %>% arrange(desc(order))

ggplot(choropleth.df, aes(long, lat, group = group)) +
```

```
  geom_polygon(aes(fill = rate), colour = alpha("white", 1/2), size = 0.05) +
  geom_polygon(data = state.df, colour = "grey90", fill = NA, size = 0.5) +
  coord_fixed() +
  theme_minimal() +
  ggtitle("US unemployment rate by county") +
  scale_fill_viridis(option="magma")+
  theme_void()
```

US unemployment rate by county



0.52 US county small multiples

0.53 World migration

TODO Add plot with migration data from kaggle

0.54 Networks

https://www.data-imaginist.com/2017/ggraph-introduction-layouts/

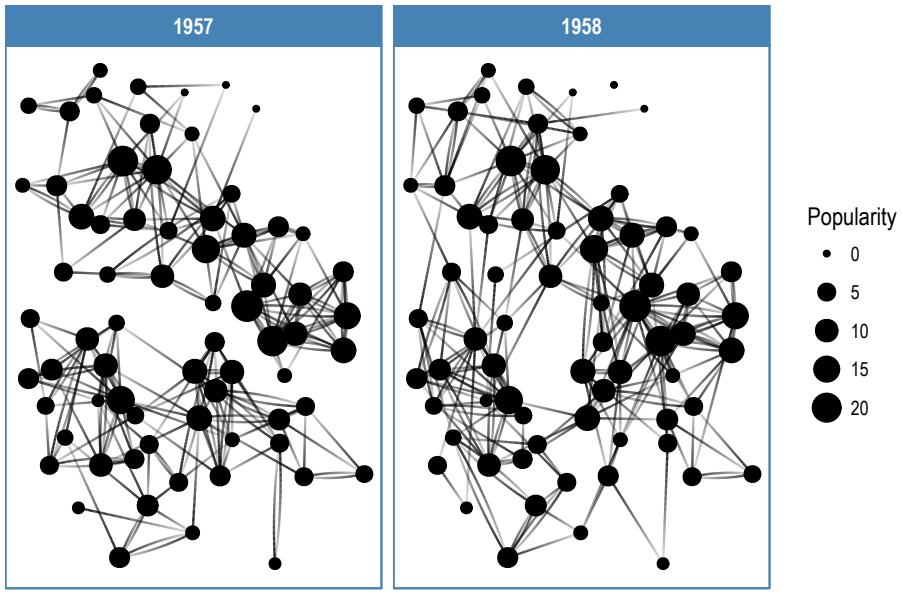
0.54.1 World migration network

Examples from: https://github.com/thomasp85/ggraph

```
library(ggraph) # ggplot extension
```

```
##
## Attaching package: 'ggraph'
## The following object is masked from 'package:treemapify':
##
##     geom_treemap
```

```
library(igraph) # For network calculations


# Graph of highschool friendships
graph <- graph_from_data_frame(highschool)
V(graph)$Popularity <- degree(graph, mode = 'in')

# Network faceted by year
ggraph(graph, layout = 'kk') +
    geom_edge_fan(aes(alpha = ..index..), show.legend = FALSE) +
    geom_node_point(aes(size = Popularity)) +
    facet_edges(~year) +
    theme_graph(foreground = 'steelblue', fg_text_colour = 'white')
```



0.54.2 Hierarchy

link to treemap

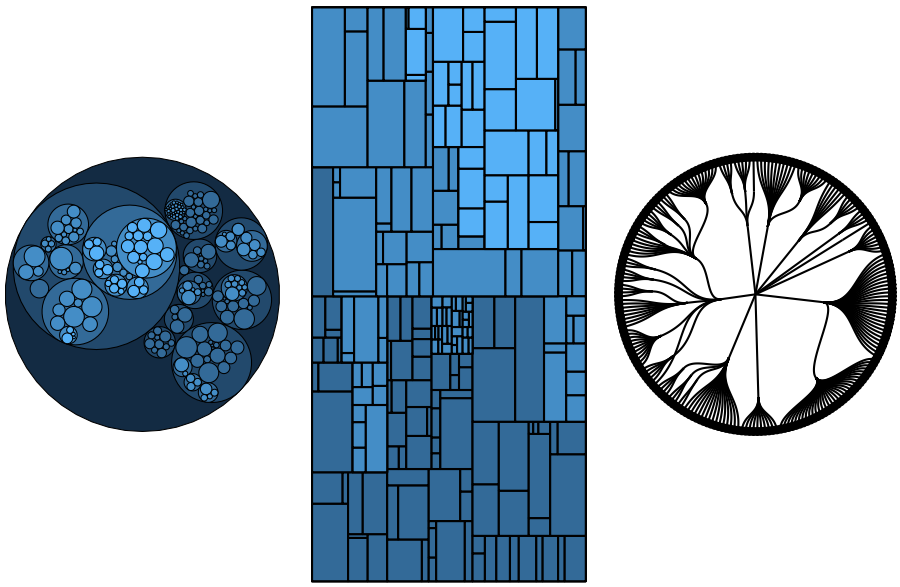
```
## TODO Convert to migration data
library(ggraph)
```

```
##
## Attaching package: 'ggraph'
```

```
## The following object is masked from 'package:treemapify':
##
##     geom_treemap
```

```
flare.df = ggraph::flare

graph <- graph_from_data_frame(flare.df$edges, vertices = flare.df$vertices)


circle.plot = ggraph(graph, 'circlepack', weight = 'size') +
    geom_node_circle(aes(fill = depth), size = 0.25, n = 50) +
    coord_fixed() +
    theme_graph() +
    theme(legend.position = "none", plot.margin=unit(c(0,0,0,0), "cm"))
```

```
## Data describe the class hiearchy of the Flare visualization library
tree.plot = ggraph(graph, layout = 'treemap', weight = 'size') +
    geom_node_tile(aes(fill = depth)) +
    theme_graph() +
    theme(legend.position = "none", plot.margin=unit(c(0,0,0,0), "cm"))



## Same basic data plotted as a circular tree
round_dendro.plot = ggraph(graph, layout = 'dendrogram', circular = TRUE) +
    geom_edge_diagonal() +
    geom_node_point(aes(filter = leaf)) +
    coord_fixed() +
    theme_graph() +
  theme(legend.position = "none", plot.margin=unit(c(0,0,0,0), "cm"))


ggarrange(circle.plot, tree.plot, round_dendro.plot,
nrow=1, ncol = 3, align = "hv")
```



0

Graphical tables–interactive tables, highlights, and sparklines

```
library(tidyverse)
```

```
library(DT)

datatable(mpg, options = list(pageLength = 5), filter = 'top')
```

0.55 Heatmap table

From https://rstudio.github.io/DT/010-style.html

```
df = as.data.frame(cbind(matrix(round(rnorm(50), 3), 10), sample(0:1, 10, TRUE)))

brks <- quantile(df, probs = seq(.05, .95, .05), na.rm = TRUE)
clrs <- round(seq(255, 40, length.out = length(brks) + 1), 0) %>%
  {paste0("rgb(255,", ., ",", ., ")")}
datatable(df) %>% formatStyle(names(df), backgroundColor = styleInterval(brks, clrs))
```

0.56 Table lens with a integrated bar

```
datatable(df) %>% formatStyle(names(df),
  background = styleColorBar(range(df), 'lightblue'),
```

```
  backgroundSize = '98% 88%',
  backgroundRepeat = 'no-repeat',
  backgroundPosition = 'center')
```

0.57 Sparkline

https://leonawicz.github.io/HtmlWidgetExamples/ex_dt_sparkline.html

0

Polishing and annotating graphs

```
library(tidyverse)
library(ggrepel)
mtcars.df = mtcars
```

All graphs should have axis labels with units Legends should have titles Titles are often included in the figure caption and not the graph Annotations can label points and tell a story Annotations can label lines better than legends

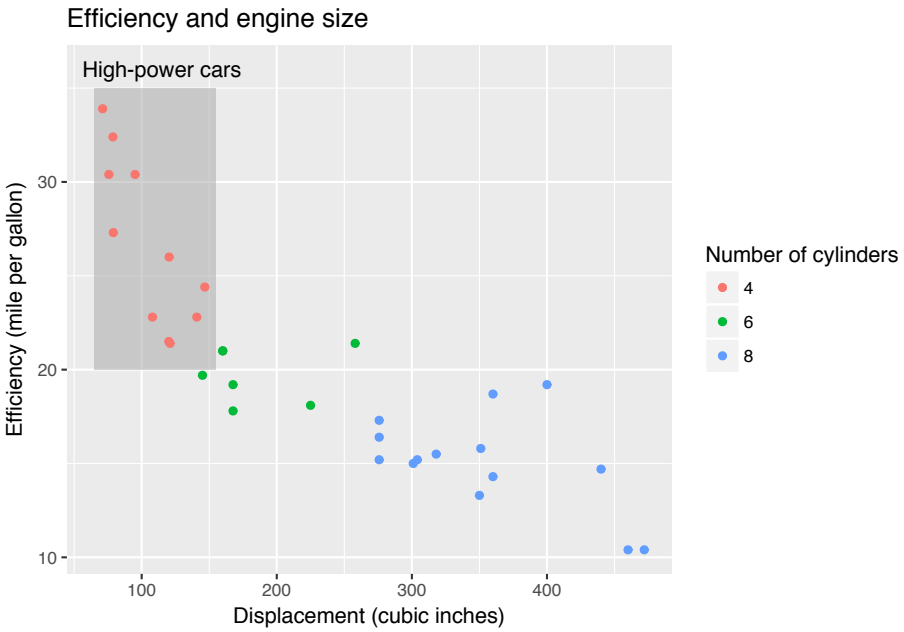
Labels and annotations added as another layer

Create a plot Add axis labels, with units Annotate to tell a story Select and appropriate color palette Place the legend inside the graph Change the theme to increase the data to ink ratio Check that changing the theme did not undo your legend placement Save as a 5X5 inch image in a format to minimize blur

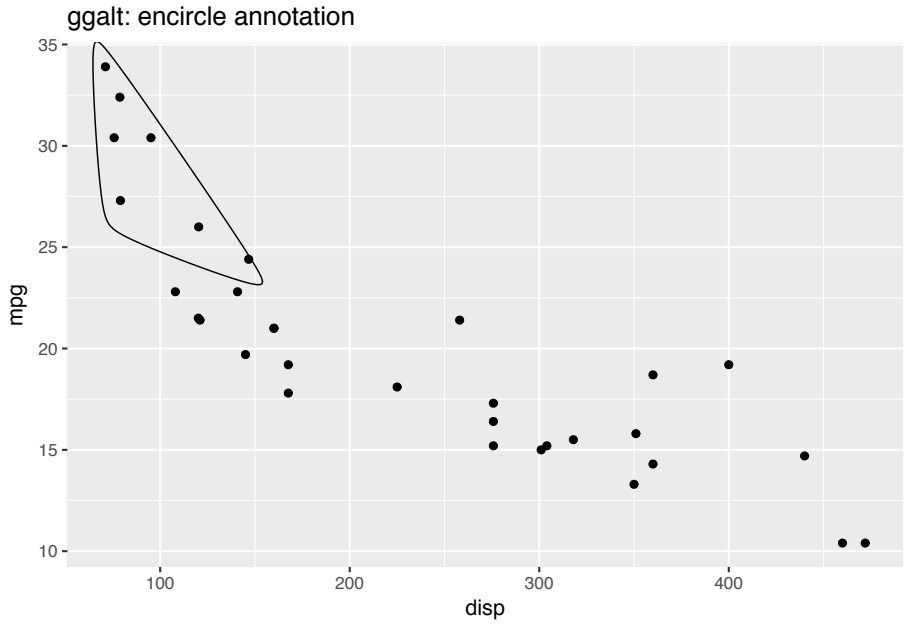
0.58 Labeling and annotation

0.58.1 Annotating data

```
## Text and rectangle annotation
ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
  annotate(geom = "rect", ymin = 20, ymax = 35, xmin = 65, xmax = 155, fill = "grey65", al
  annotate(geom = "text", x = 115, y = 36, label = "High-power cars" ) +
  geom_point()+
  labs(x = "Displacement (cubic inches)", y = "Efficiency (mile per gallon)",
       title = "Efficiency and engine size", colour = "Number of cylinders")
```



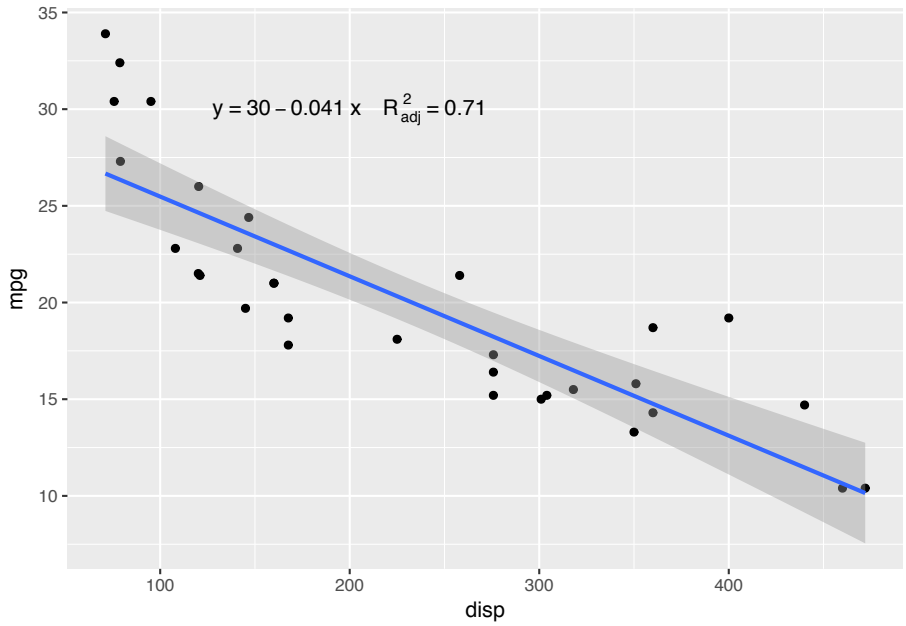

```
## Encircle pooints
library(ggalt)

 ggplot(mtcars, aes(disp, mpg))+
    geom_point()+
    geom_encircle(data=filter(mtcars, mpg>24),
                  s_shape=0.75, expand=0.05) +
   labs(title = "ggalt: encircle annotation")
```




```
#

library(ggpmisc) # For stat_poly_eq for equation annotation
```

```
## For news about 'ggpmisc', please, see http://www.r4photobiology.info/
## For on-line documentation see http://docs.r4photobiology.info/ggpmisc/
```

```
# https://cran.r-project.org/web/packages/ggpmisc/vignettes/user-guide-1.html

formula <- y ~ poly(x, 1, raw = TRUE)
regression.plot =
  ggplot(mtcars, aes(x=disp, y=mpg)) +
    geom_point() +
  geom_smooth(method = "lm", formula = formula) +
  stat_poly_eq(aes(label =  paste(..eq.label.., ..adj.rr.label.., sep = "~~~~")),
                   formula = formula, parse = TRUE, coef.digits = 2, label.x = 200, label.

regression.plot
```



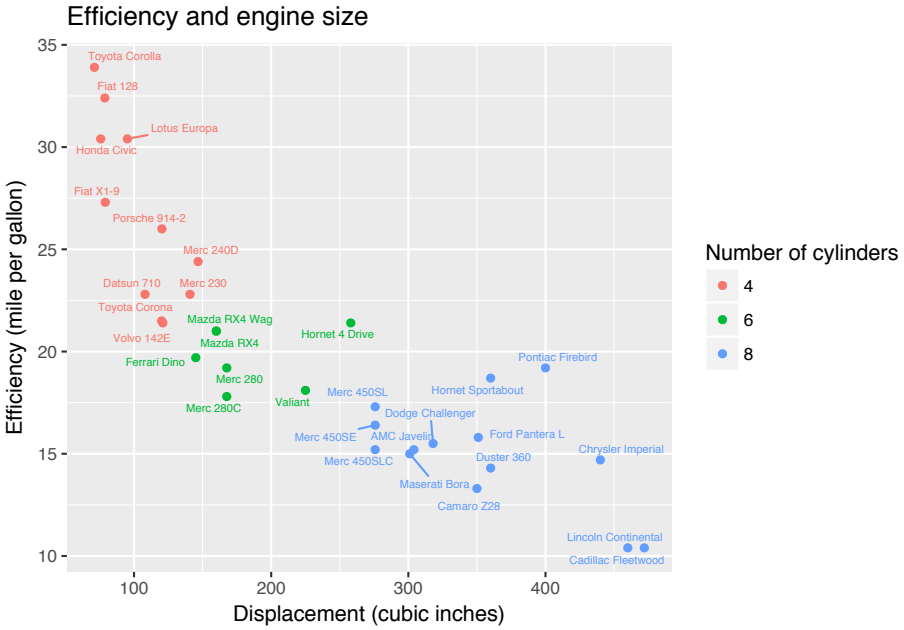

```
ggsave(file = "regression_annotation.pdf", plot = regression.plot)
```

```
## Saving 6.5 x 4.5 in image
```

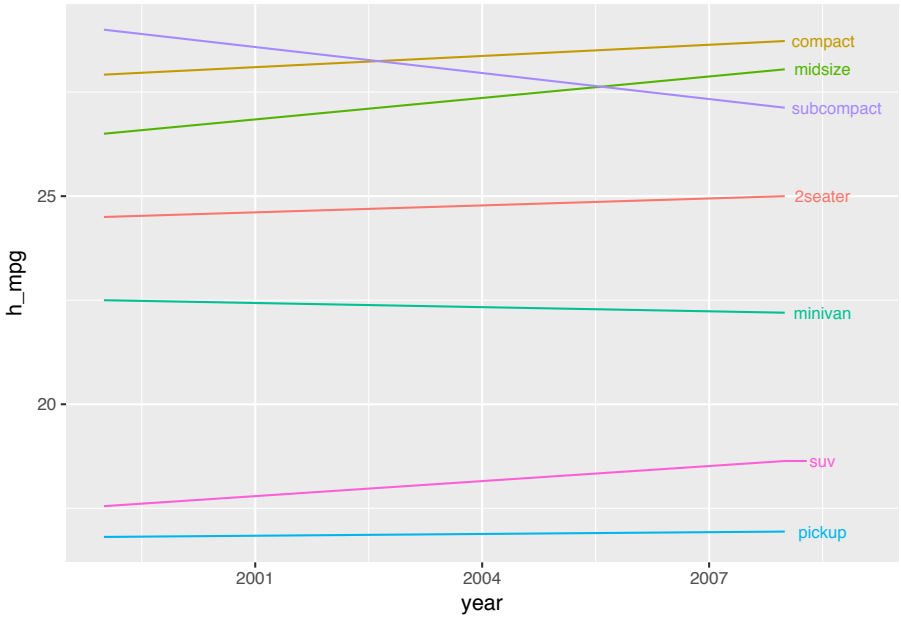
0.58.2 Annotating points and select points

library(ggrepel)

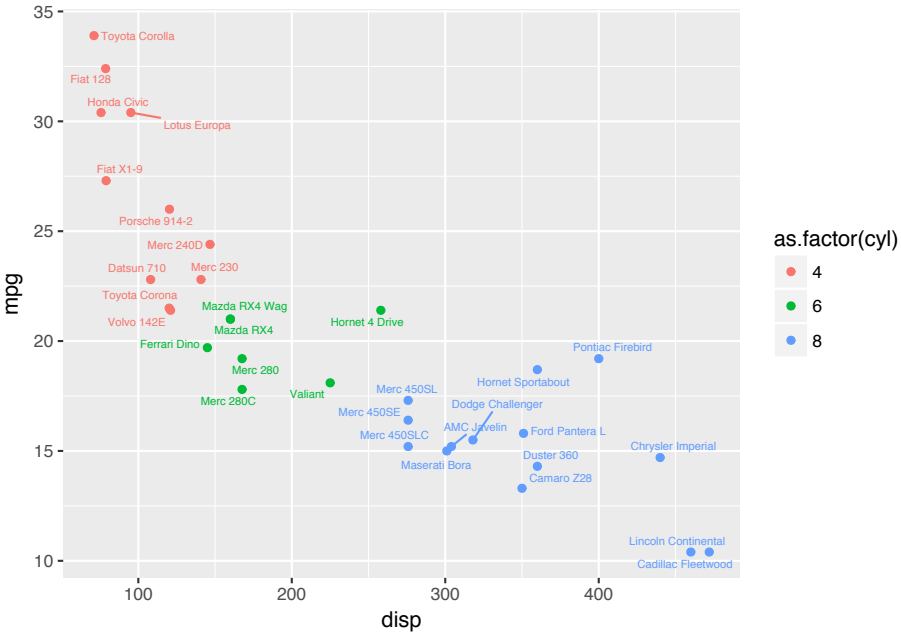
```
# Annotated points
## ggrepel package
# Add names of cars to dataframe
mtcars.df = mtcars
mtcars.df$name = row.names(mtcars.df)

ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
  geom_point()+
  geom_text_repel(aes(label = name), size = 2, show.legend = FALSE)+
  labs(x = "Displacement (cubic inches)", y = "Efficiency (mile per gallon)",
       title = "Efficiency and engine size", colour = "Number of cylinders")
```



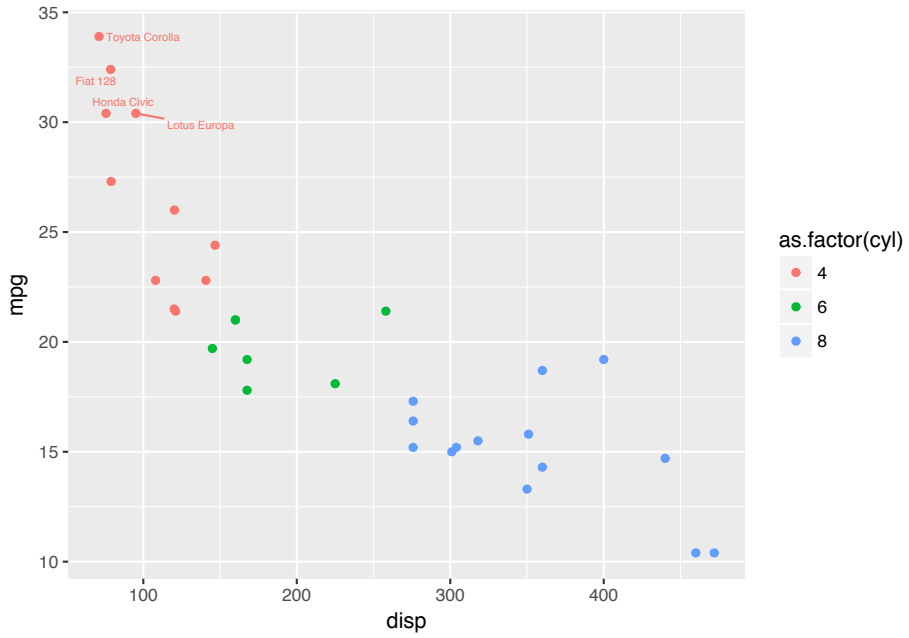

```
## Add annotations to lines rather than legend
s.mpg.df = mpg %>% group_by(year, class) %>% summarise(h_mpg = mean(hwy, na.rm = TRUE)) %>


ggplot(s.mpg.df, aes(year, h_mpg, colour = class))+geom_line()+
   coord_cartesian(xlim = c(min(s.mpg.df$year), max(s.mpg.df$year) + 1)) +
   geom_text_repel(
     data = filter(s.mpg.df, year == max(year)),
     aes(label = class),
     size = 3, nudge_x = .5) +
  guides(color=FALSE)
```




```
ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
  geom_point()+
  geom_text_repel(aes(label = name), size = 2, show.legend = FALSE)
```

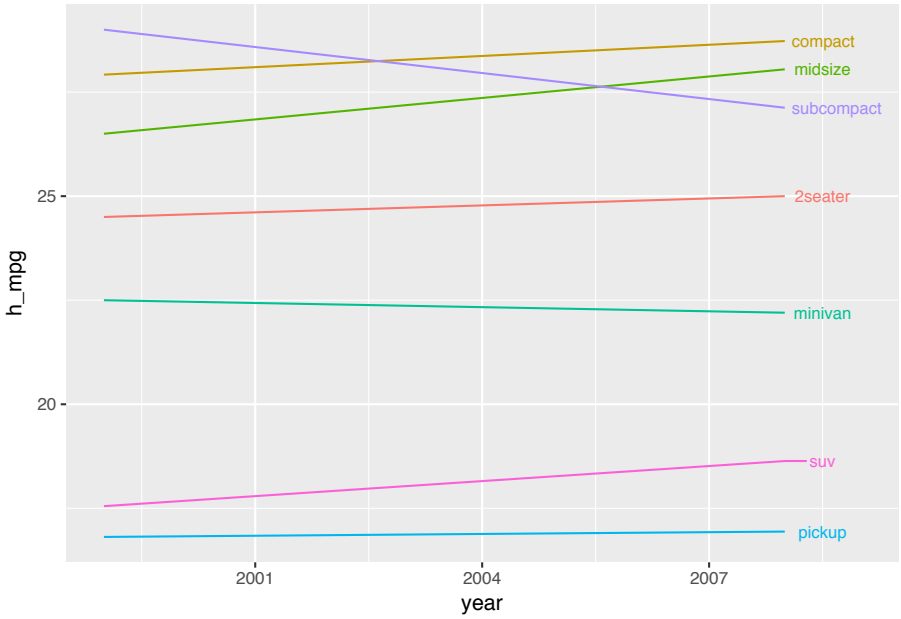


```
ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
  geom_point()+
  geom_text_repel(data = mtcars.df %>% filter(mpg>30), aes(label = name), size = 2, show.l
```

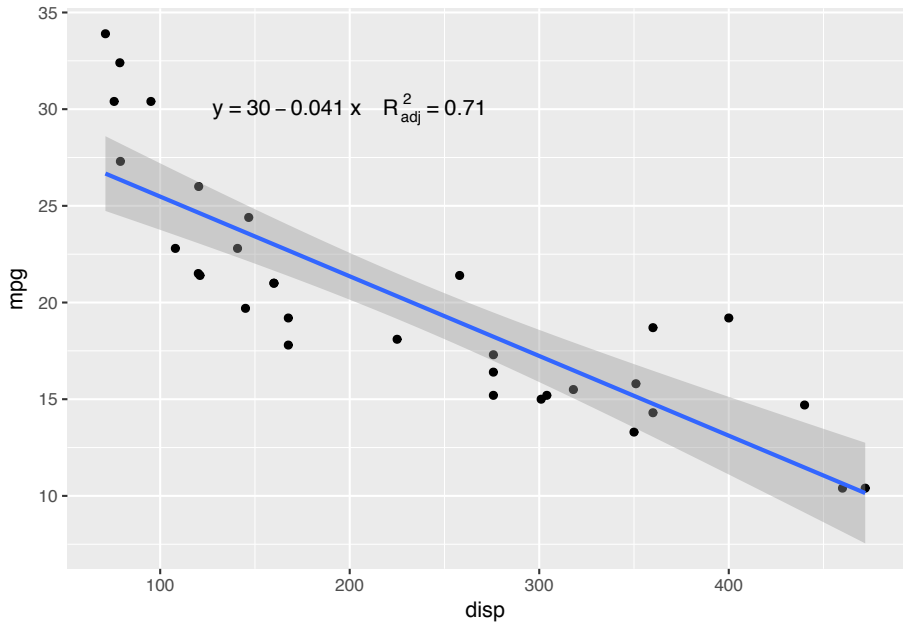


0.58.3 Labels on lines

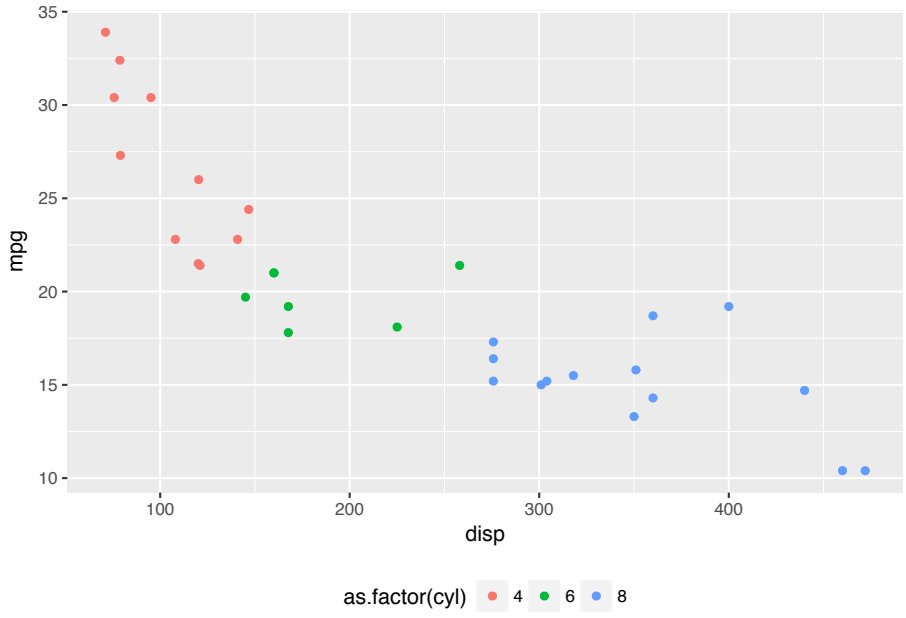
```
s.mpg.df = mpg %>% group_by(year, class) %>%
  summarise(h_mpg = mean(hwy, na.rm = TRUE)) %>%
  ungroup()

ggplot(s.mpg.df, aes(year, h_mpg, colour = class))+
    geom_line()+
    coord_cartesian(xlim = c(min(s.mpg.df$year), max(s.mpg.df$year) + 1)) +     geom_text_
        aes(label = class), size = 3, nudge_x = .5) +
    guides(color=FALSE)
```



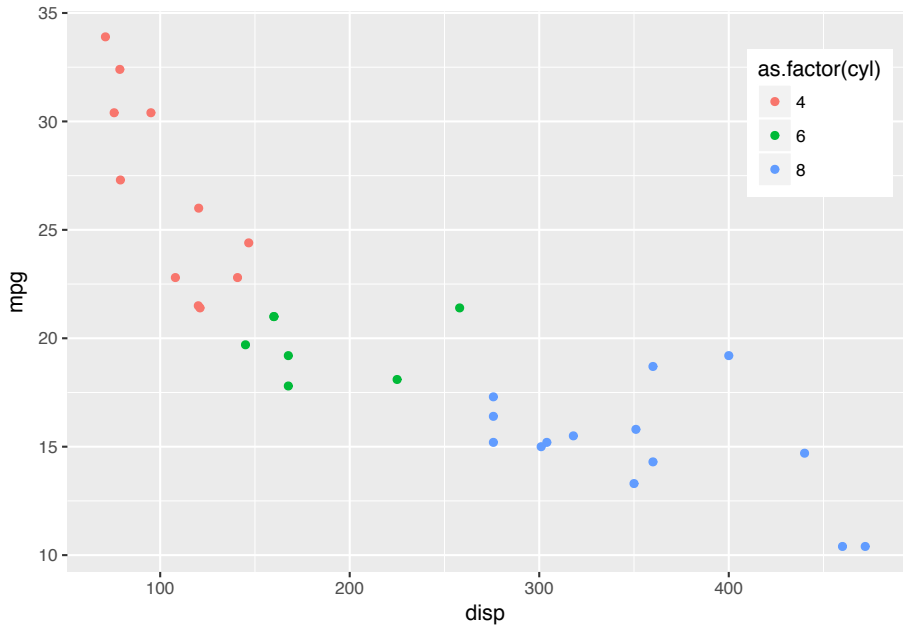

```
library(ggpmisc) # For stat_poly_eq for equation annotation
formula <- y ~ poly(x, 1, raw = TRUE)
ggplot(mtcars, aes(x=disp, y=mpg)) +
    geom_point() +
    geom_smooth(method = "lm", formula = formula) +     stat_poly_eq(aes(label =  paste(..
        sep = "~~~~")), formula = formula, parse = TRUE, coef.digits = 2, label.x = 200, l
```




```
# Place legend at bottom
ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point() +
theme(legend.position = "bottom")
```

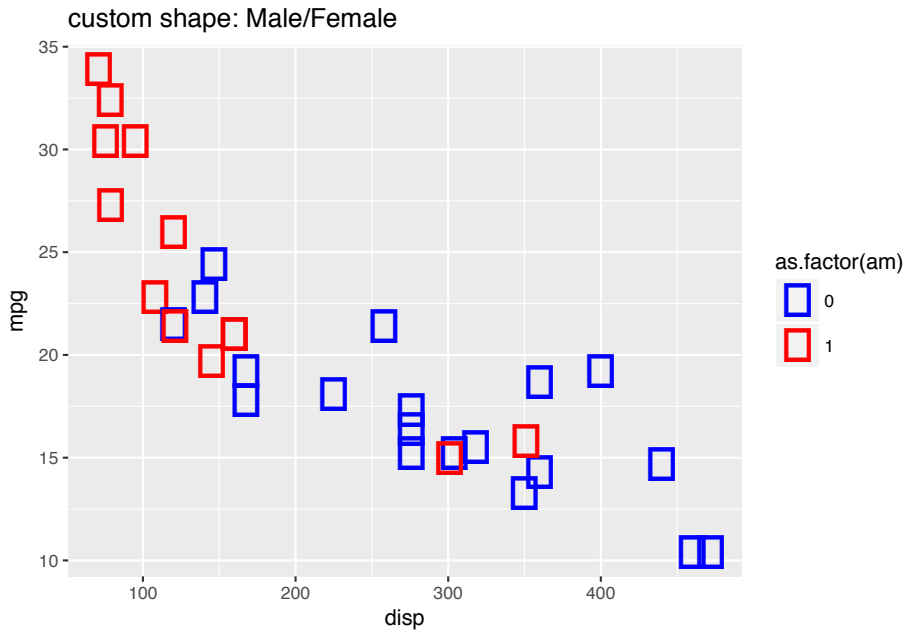


```
# Place legend in graph
ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point() +
theme(legend.position = c(0.9, 0.8))
```



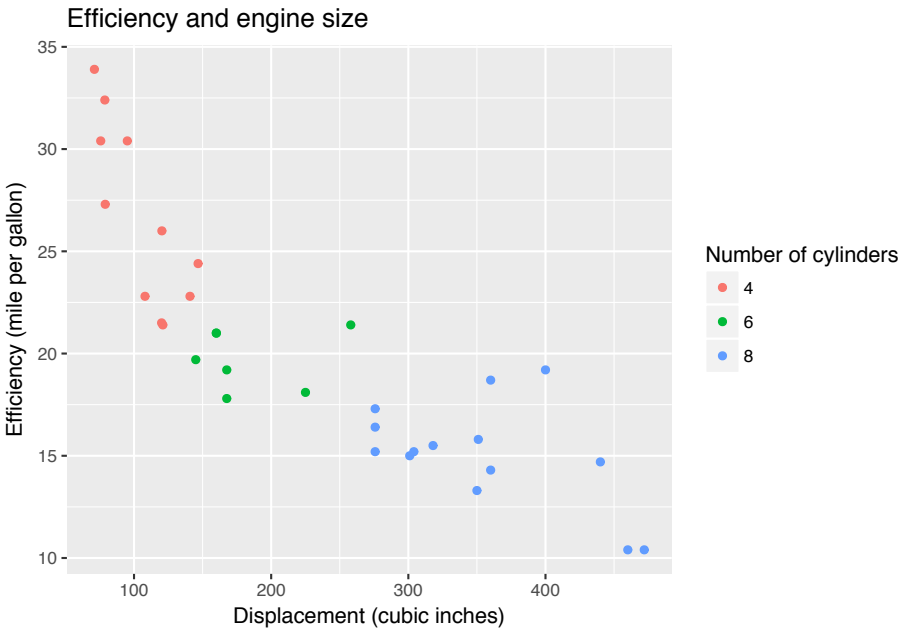
0.58.4 Meaningful symbols

```
ggplot(mtcars,
        aes(disp,  mpg, colour = as.factor(am), shape = as.factor(am))) +
  geom_point(size = 8) +
    scale_color_manual(values = c("0" = "blue", "1" = "red")) +
  scale_shape_manual(values = c("0" = "\u2642", "1" = "\u2640")) +
  labs(title = "custom shape: Male/Female")
```

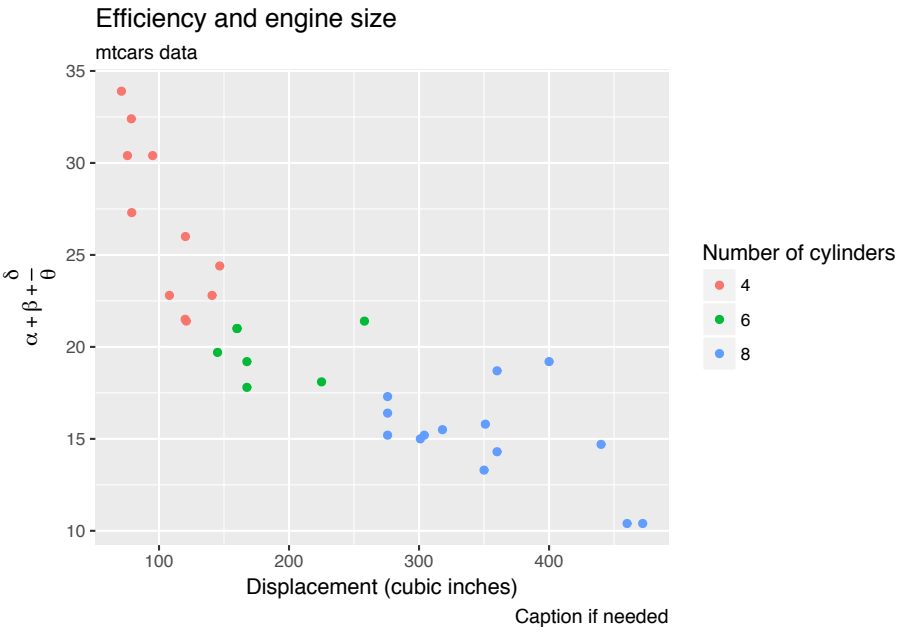


0.58.5 Adding and removing labels: title, axis labels, facet labels, and legends

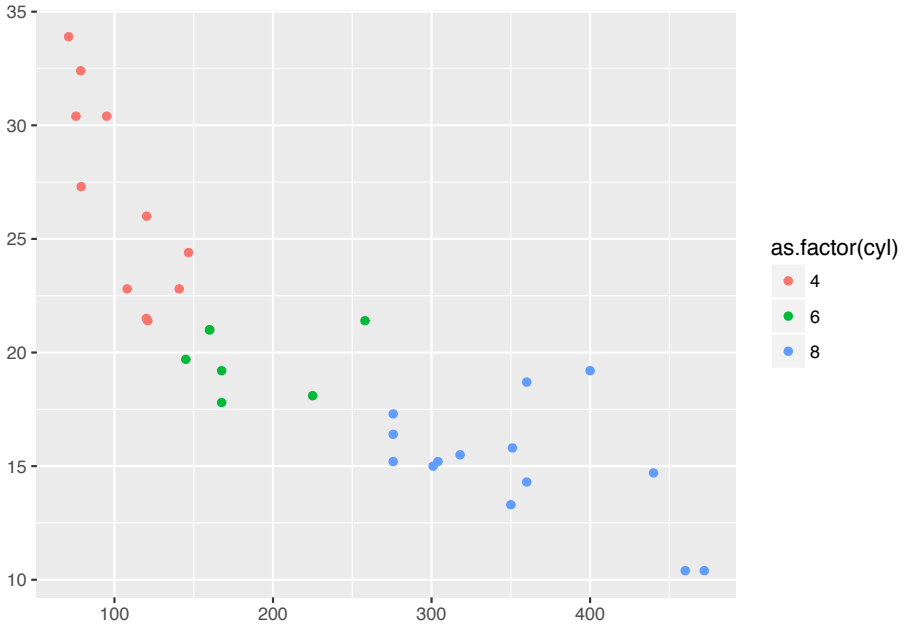
```
library(tidyverse)
library(ggalt)
library(ggrepel)

# Add names of cars to dataframe
mtcars.df = mtcars
mtcars.df$name = row.names(mtcars.df)

## Axis labels, title, and legend
ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
  geom_point()+
  labs(x = "Displacement (cubic inches)", y = "Efficiency (mile per gallon)",
       title = "Efficiency and engine size", colour = "Number of cylinders")
```



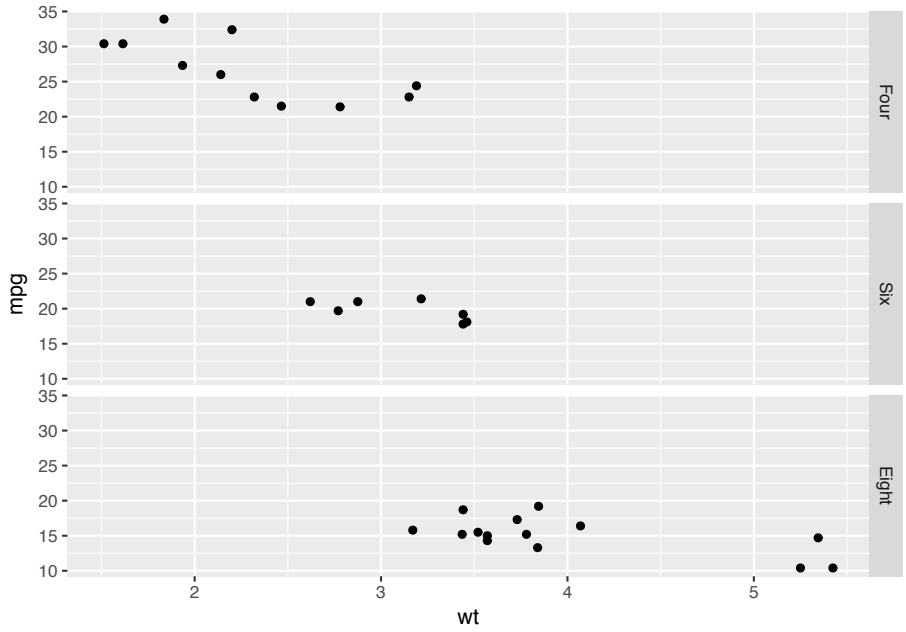

```
## Subtitle and caption
ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
  geom_point()+
  labs(x = "Displacement (cubic inches)", y = quote(alpha + beta + frac(delta, theta)),
       title = "Efficiency and engine size", subtitle = "mtcars data",
       caption = "Caption if needed",
       colour = "Number of cylinders")
```



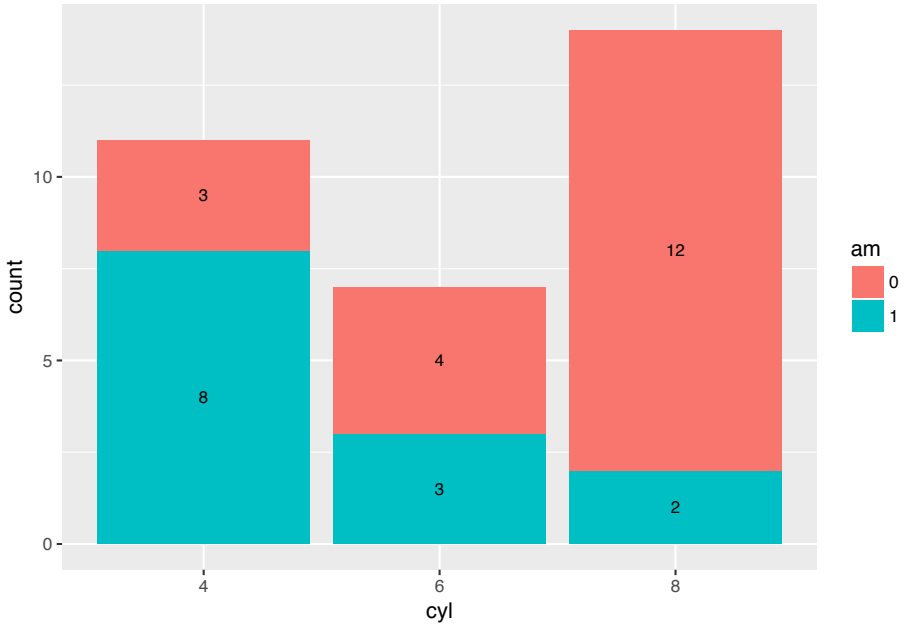

```
## Remove labels
ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
  geom_point()+
  labs(x = "", y = "")
```



```
## Facet labels
cyl_names <- as_labeller(c("4" = "Four", "6" = "Six", "8" = "Eight"))

 ggplot(mtcars.df, aes(wt, mpg))+
   geom_point() +
   facet_grid(cyl ~ ., labeller = as_labeller(cyl_names))
```

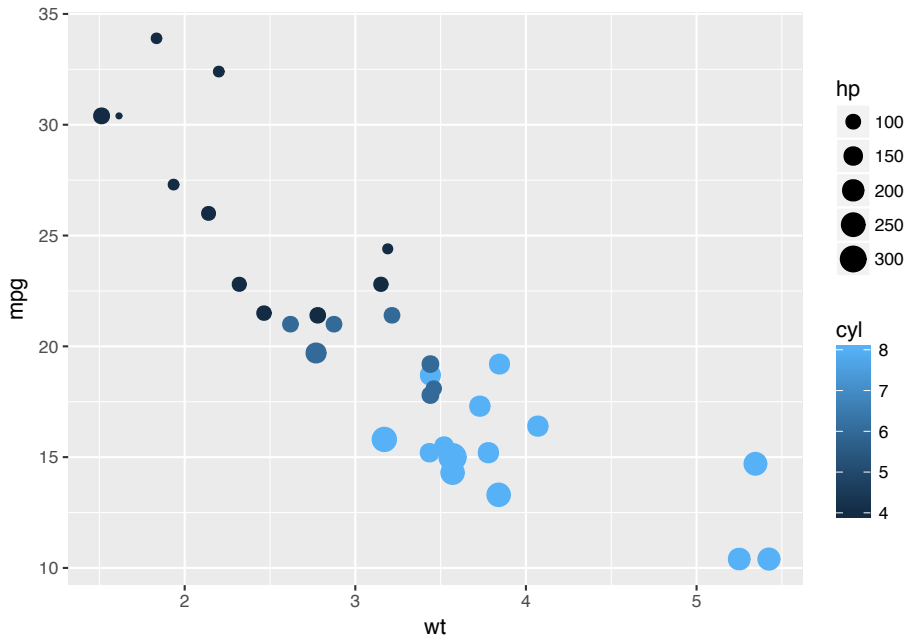



```
count.mtcars.df = mtcars.df %>%
  mutate(cyl = as.factor(cyl), am = as.factor(am)) %>%
  group_by(am, cyl) %>%
  summarise(count = n()) %>%
  mutate(label.pos = cumsum(count) - (0.1 * count))

ggplot(count.mtcars.df, aes(x = cyl, y = count, fill = am, label = count)) +
  geom_bar(stat = "identity") +
  geom_text(size = 3, position = position_stack(vjust = 0.5))
```

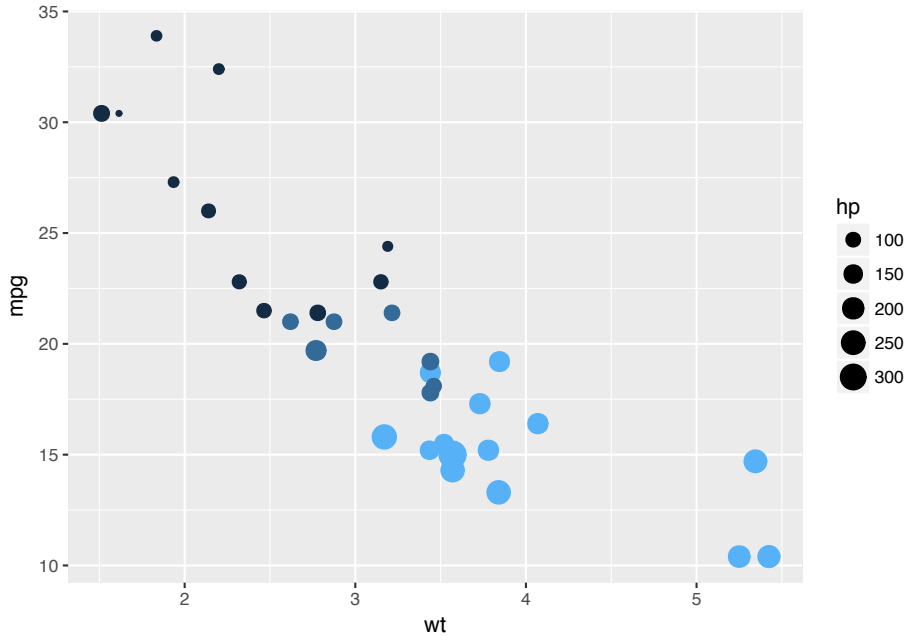


0.58.6 Remove and position one or more legends

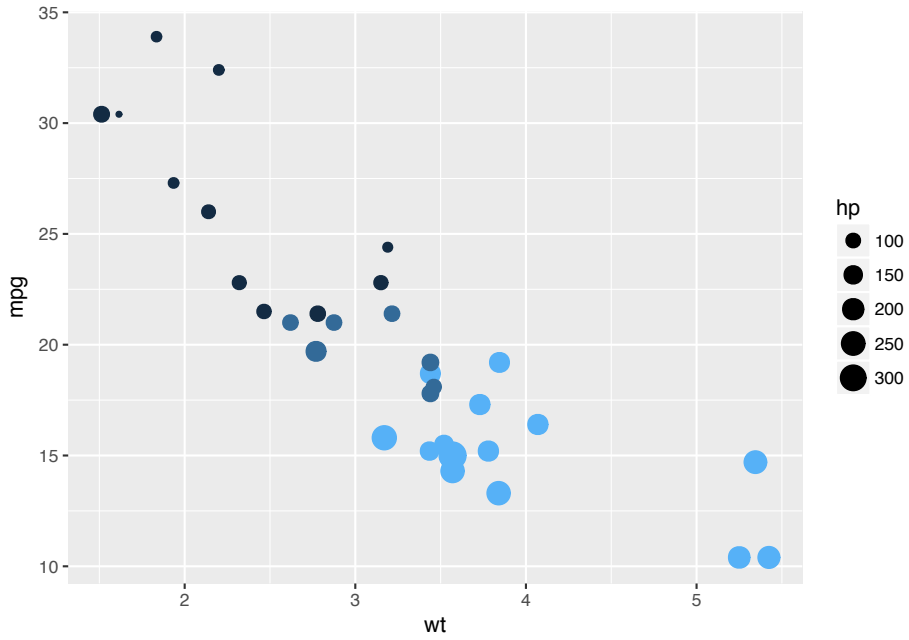
```
ggplot(mtcars.df, aes(wt, mpg, colour = cyl, size = hp))+
  geom_point()
```



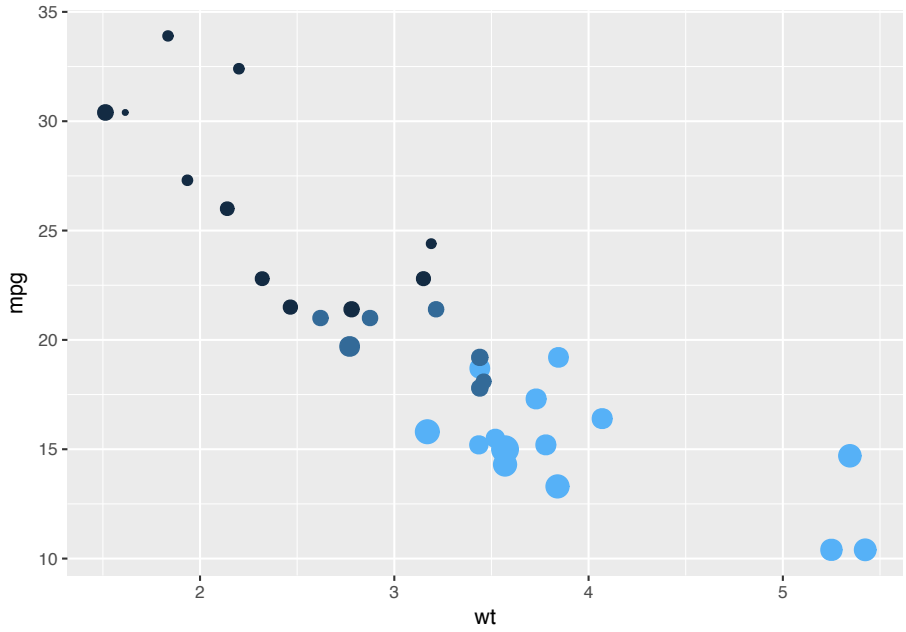

```
ggplot(mtcars.df, aes(wt, mpg, colour = cyl, size = hp))+
  geom_point()+
  guides(colour=FALSE)
```



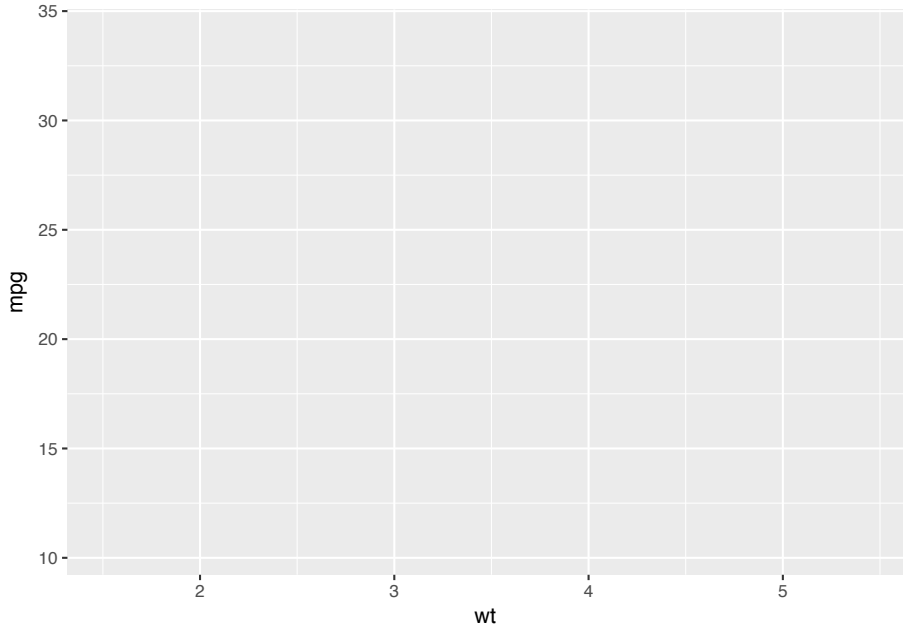
```
ggplot(mtcars.df, aes(wt, mpg, colour = cyl, size = hp))+
  geom_point()+
scale_colour_continuous(guide=FALSE)
```




```
ggplot(mtcars.df, aes(wt, mpg, colour = cyl, size = hp))+
  geom_point(show.legend=FALSE)
```

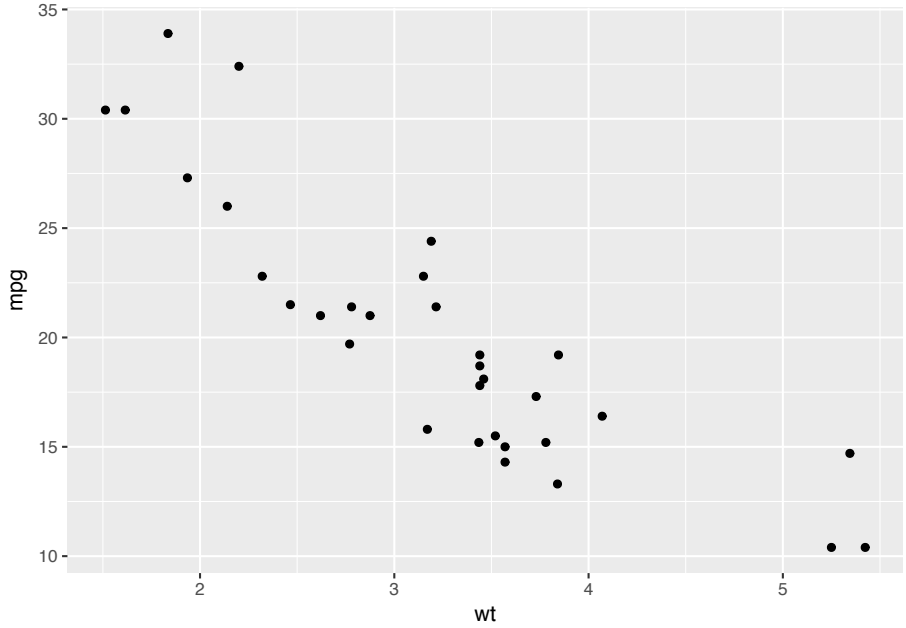



```
ggplot(mtcars.df, aes(wt, mpg, colour = cyl, size = hp))+
theme(legend.position="none") # All legends
```

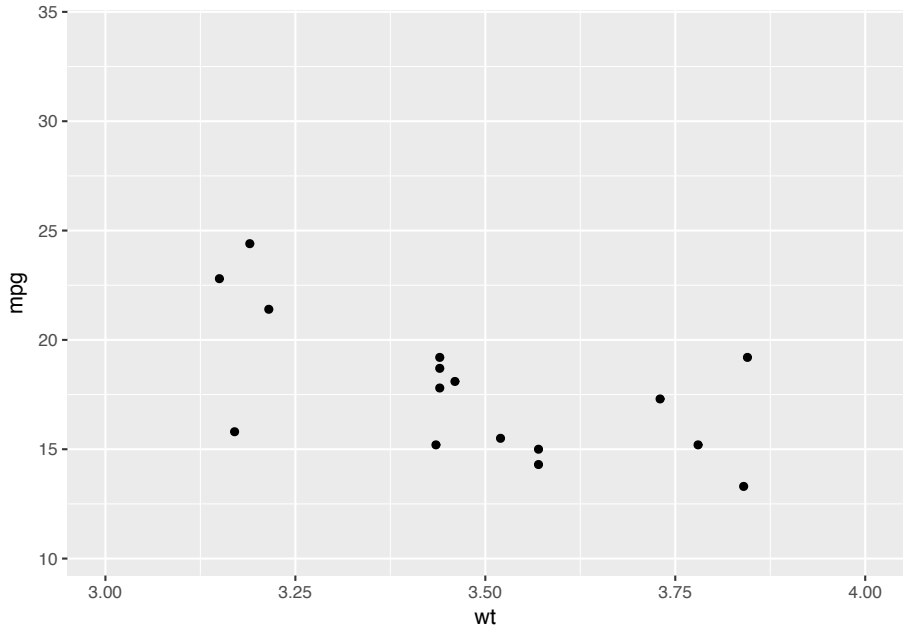


0.58.7 Setting limits on axis and position of graph

```
r ## Change axis limits to focus on part of data ggplot(mtcars.df,
aes(wt, mpg))+ geom_point()
```

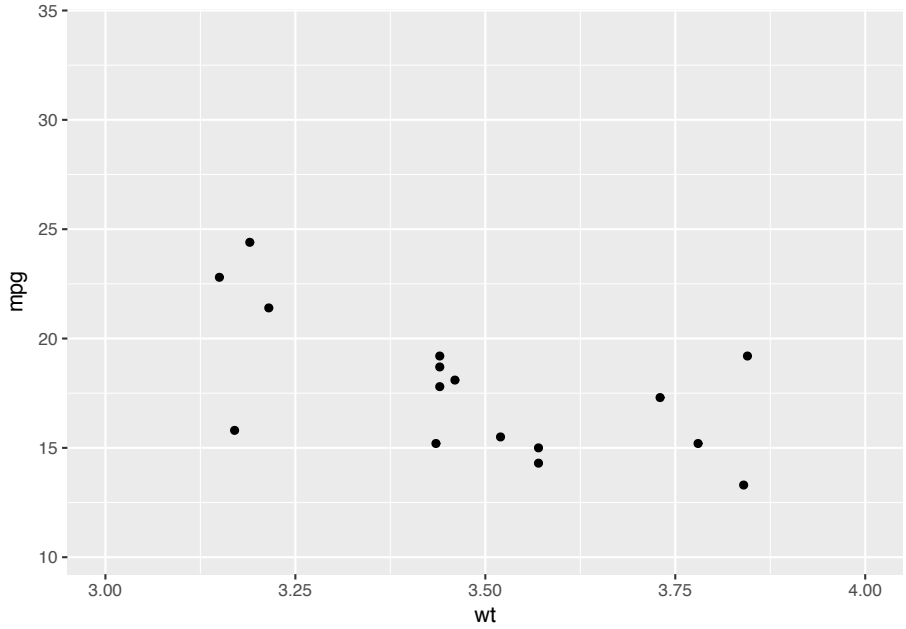



```
r ggplot(mtcars.df, aes(wt, mpg))+ geom_point()+ lims(x = c(3,
4))

## Warning: Removed 16 rows containing missing values ##
(geom_point).
```



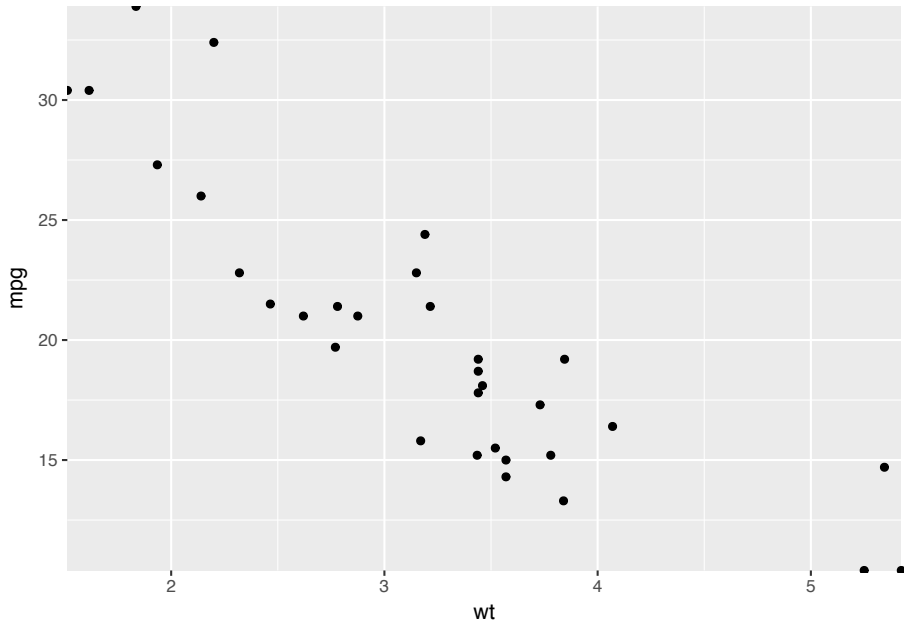

```
r      ggplot(mtcars.df,   aes(wt,   mpg))+      geom_point()+
scale_x_continuous(limits = c(3, 4))
```

```
## Warning: Removed 16 rows containing missing values   ##
(geom_point).
```

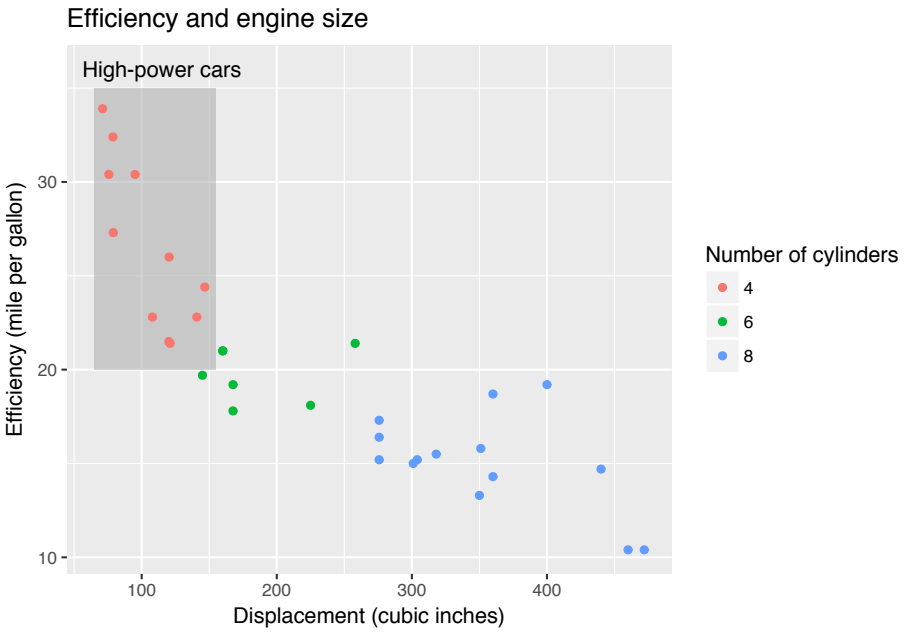


0.58.8 Adjust margin between points and edge of plot area

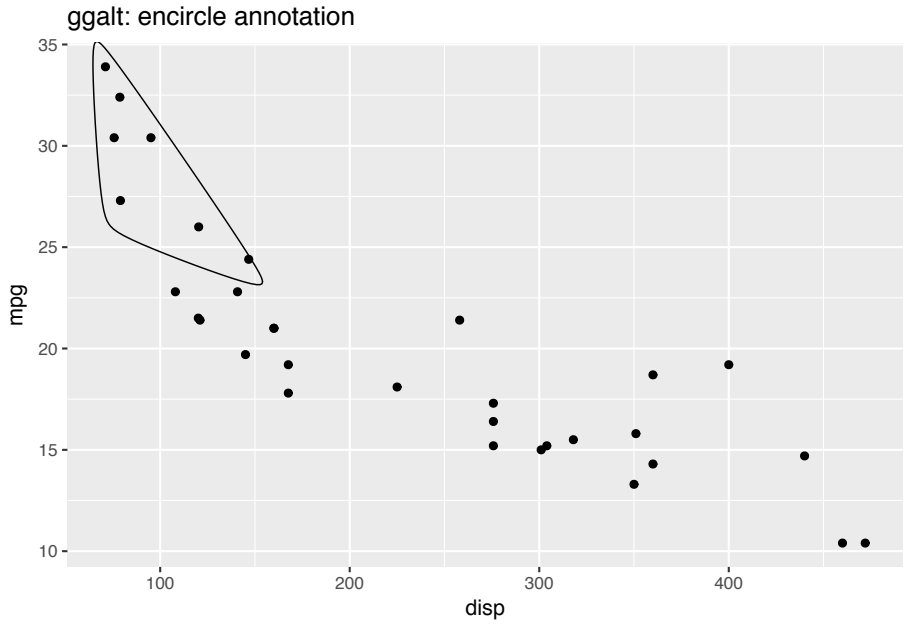
```
ggplot(mtcars.df, aes(wt, mpg))+
 geom_point() +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0))
```




```
ggplot(mtcars.df, aes(disp,  mpg, color = as.factor(cyl))) +
 annotate(geom = "rect",
    ymin = 20, ymax = 35, xmin = 65, xmax = 155, fill = "grey65", alpha = .5) +
 annotate(geom = "text", x = 115, y = 36, label = "High-power cars" )+
 geom_point() +
 labs(x = "Displacement (cubic inches)", y = "Efficiency (mile per gallon)",           ti
```




```
library(ggalt)
ggplot(mtcars, aes(disp, mpg))+
geom_point()+
geom_encircle(data=filter(mtcars, mpg>24), s_shape=0.75, expand=0.05) +
labs(title = "ggalt: encircle annotation")
```



0.59 Color

The palette of colors that is scaled to the data is set separately for "fill" and for "colour"

Color palettes that work for discrete variables (e.g., factors) may not work for continuous, numeric variables

Color palette design is complex and choice depends on

Number of categories in the data Ability to support black and white printing

Ability to support color-deficient vision

Size of the space being colored

Small areas, such as points, benefit from saturated color, and large areas, such a bars are better with less intense color.

Great resources to explain the basis of the Brewer, Viridis, and ptol palettes: http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3

https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html

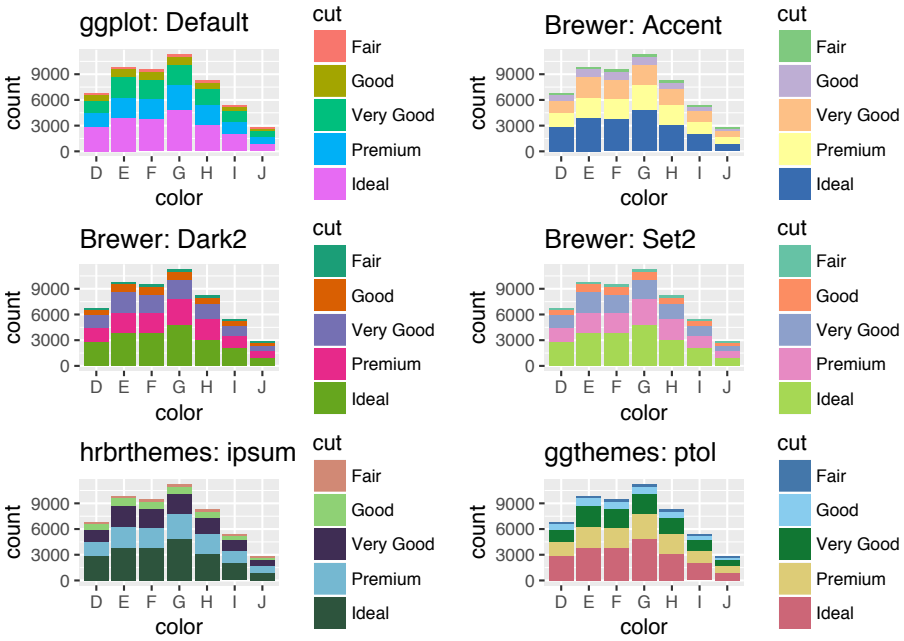
https://personal.sron.nl/~pault/

0.59.1 Resources to explain the basis form color choice

Brewer, Viridis, and ptol palettes http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3 https://cran.r-project.org/web/packages/viridis/vignettes/intro-to-viridis.html https://personal.sron.nl/~pault/

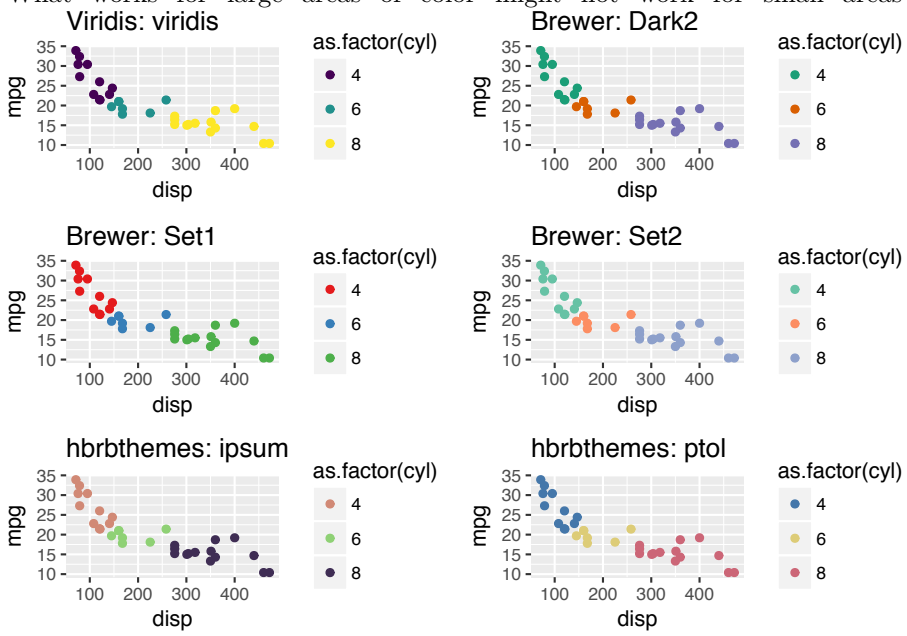
0.59.2 Large and small area colors

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are *required* to use these themes.
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
##       if Arial Narrow is not on your system, please see http://bit.ly/arialnarrow
```



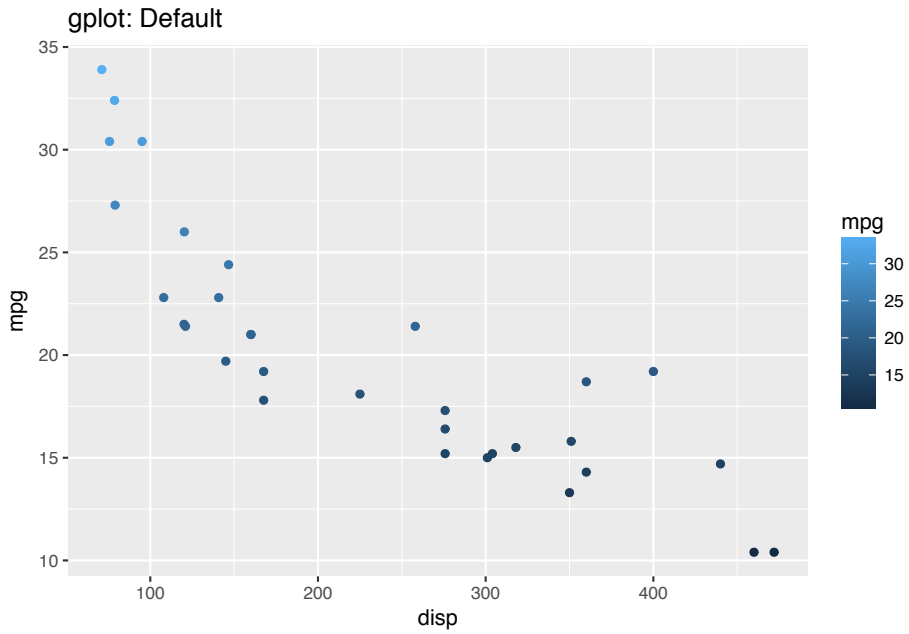
0.59.3 Color for large and small areas

What works for large areas of color might not work for small areas

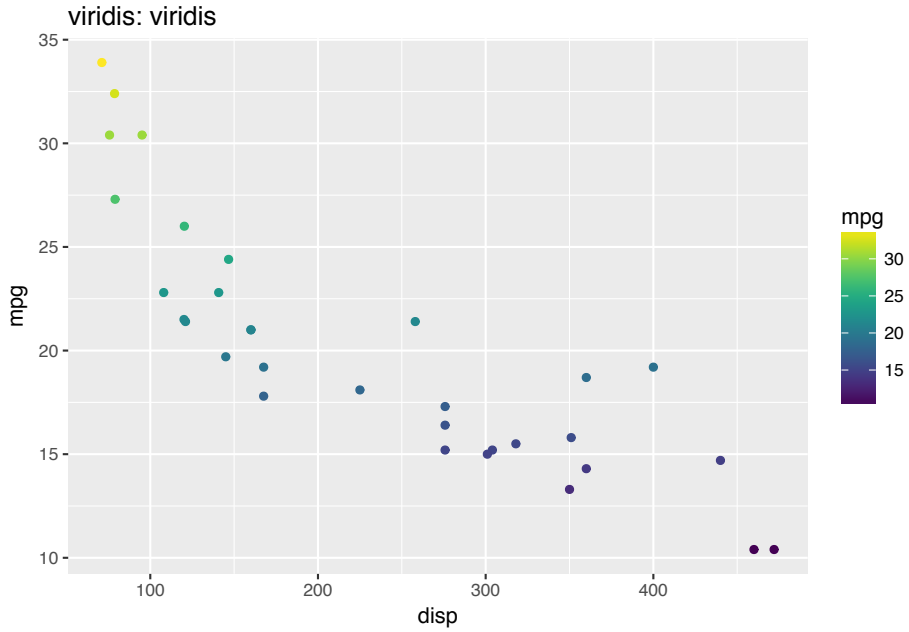


0.59.4 Continuous color scales

```
# Scales made for factors do not work with continuous variables
ggplot(mtcars, aes(disp,  mpg, color = mpg))+geom_point()+
  labs(title = "gplot: Default")
```



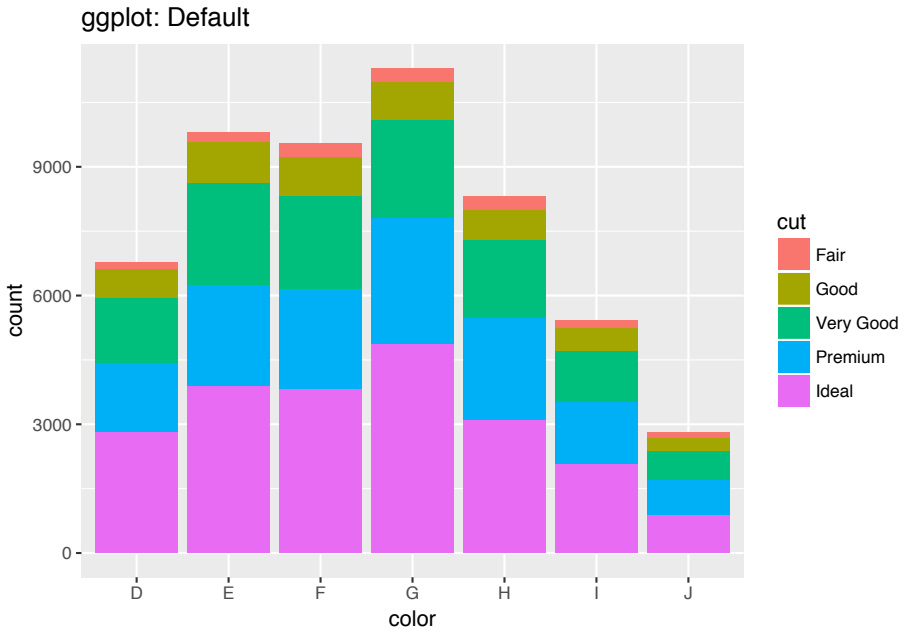

```
ggplot(mtcars, aes(disp,  mpg, color = mpg))+geom_point()+
  scale_color_viridis(discrete = FALSE) +
  labs(title = "viridis: viridis")
```



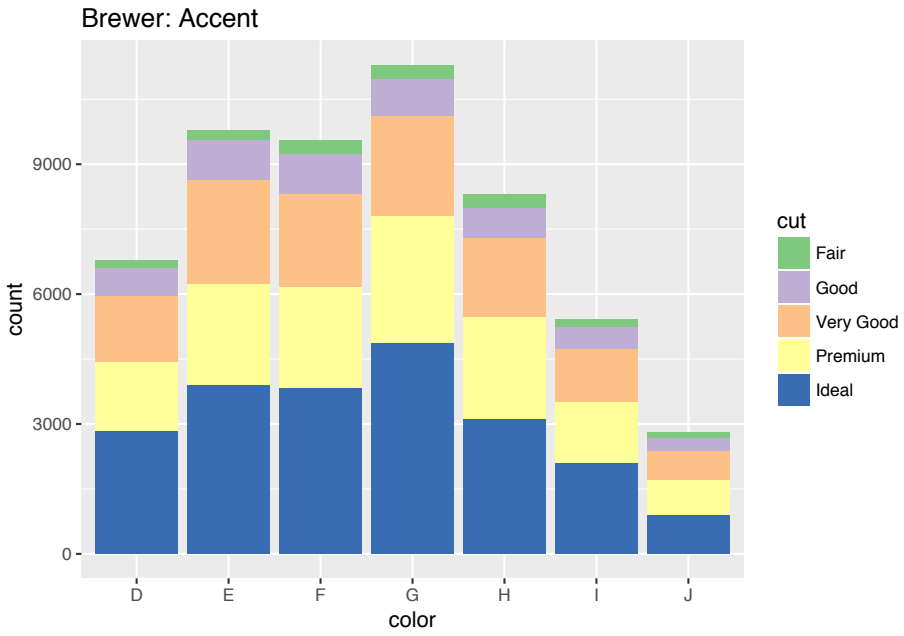
```
##TODO Add parula
```

0.59.5 Discrete color mappings

```
ggplot(diamonds, aes(x = color, fill = cut)) +
    geom_bar() +
  scale_fill_hue()+
  labs(title = "ggplot: Default")
```



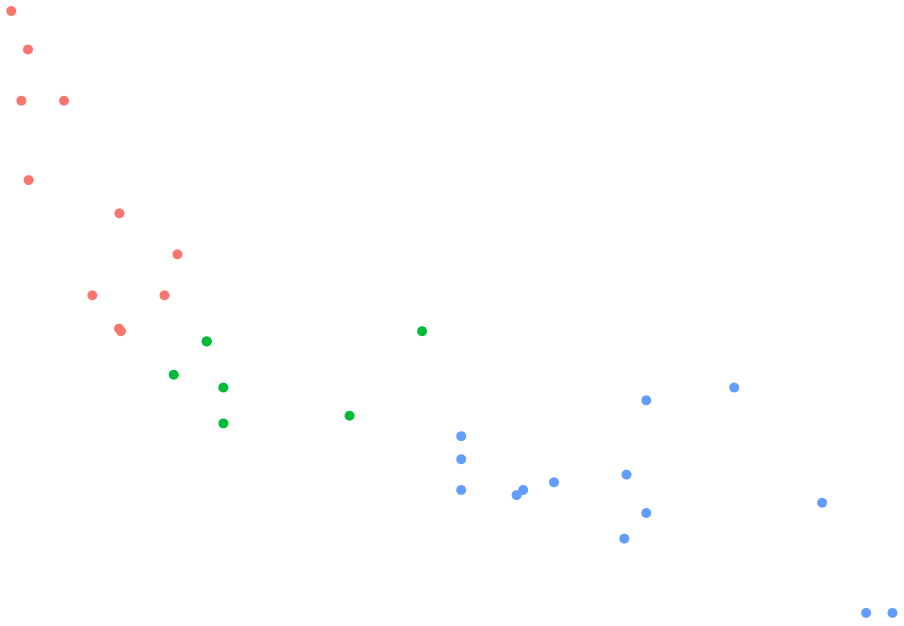

```
ggplot(diamonds, aes(x = color, fill = cut)) +  geom_bar() +
  scale_fill_brewer(palette = "Accent") +
  labs(title = "Brewer: Accent")
```



0.60 Themes and theme options

Turn off many theme elements

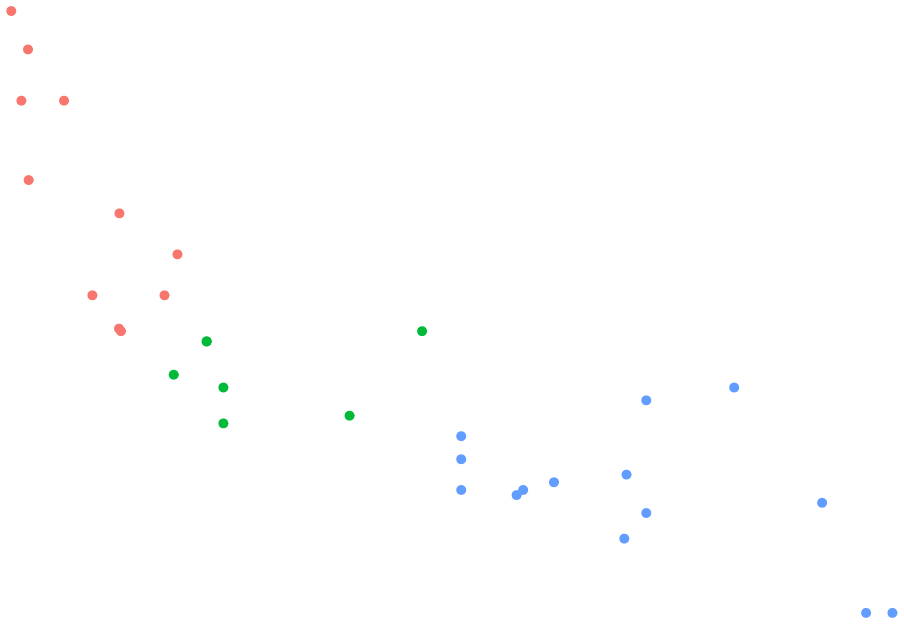
```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) +
geom_point() +
    theme(axis.line=element_blank(), axis.text.x=element_blank(),   axis.text.y=element_bl
```



0.60.1 Remove chart details

Useful for plotting graphs and networks and maps

```
ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point() +
  theme(axis.line=element_blank(),
      axis.text.x=element_blank(),
      axis.text.y=element_blank(),
      axis.ticks=element_blank(),
      axis.title.x=element_blank(),
      axis.title.y=element_blank(),
      legend.position="none",
      panel.background=element_blank(),
      panel.border=element_blank(),
      panel.grid.major=element_blank(),
      panel.grid.minor=element_blank(),
      plot.background=element_blank())
```



0.60.2 Predefined themes adjust many elements

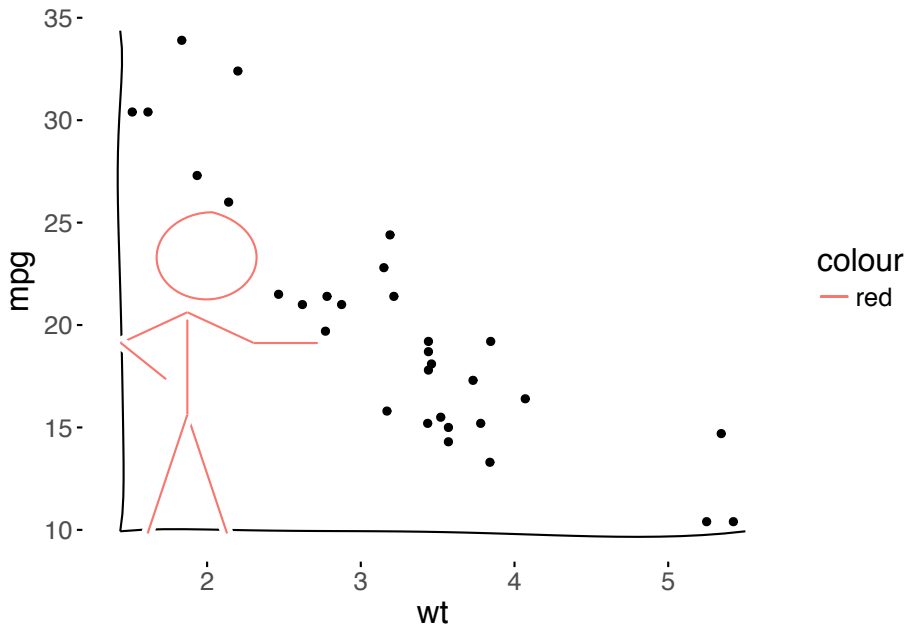
```
library(hrbrthemes) # Precise font and minimal grid
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are *required* to use these themes.
##       Please use hrbrthemes::import_roboto_condensed() to install Roboto Condensed and
##       if Arial Narrow is not on your system, please see http://bit.ly/arialnarrow
```

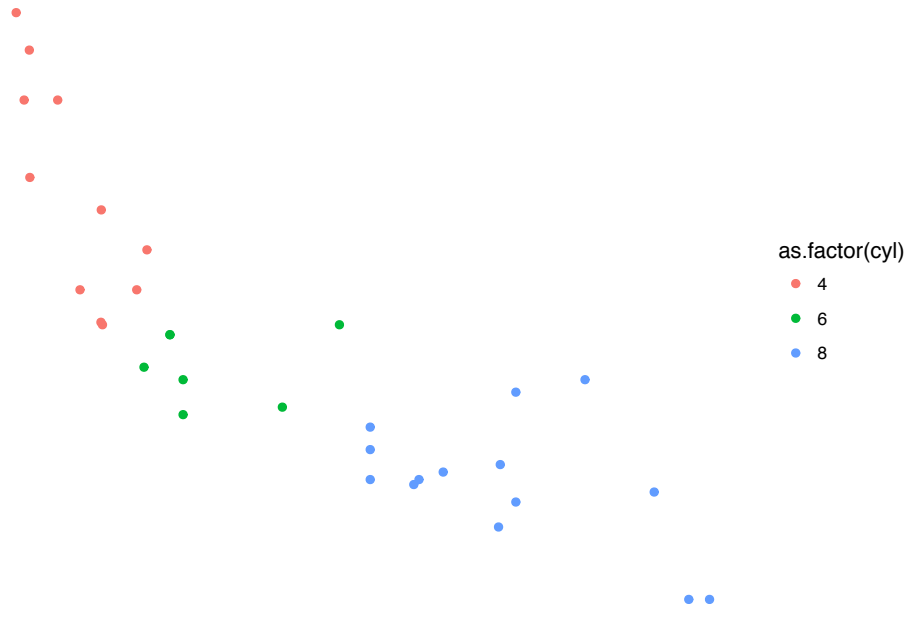
```
library(ggthemes) # Huge variety of themes including Tufte and Few
library(xkcd) # Plots in the xkcd comic style
```

```
## Loading required package: extrafont
## Registering fonts with R
```

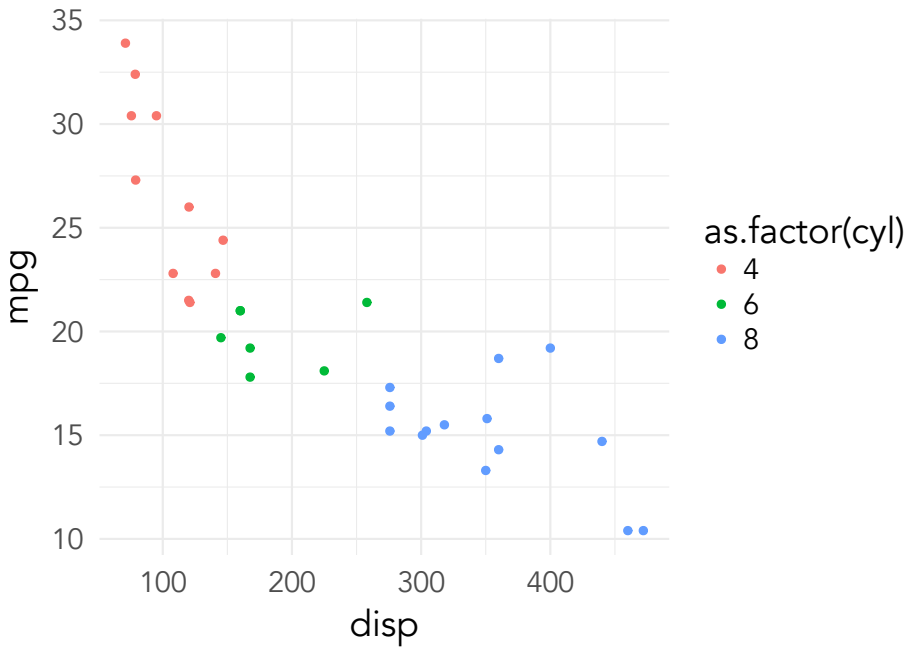
An engaging and fun theme ftp://200.236.31.7/CRAN/web/packages/xkcd/vignettes/xkcd-intro.pdf



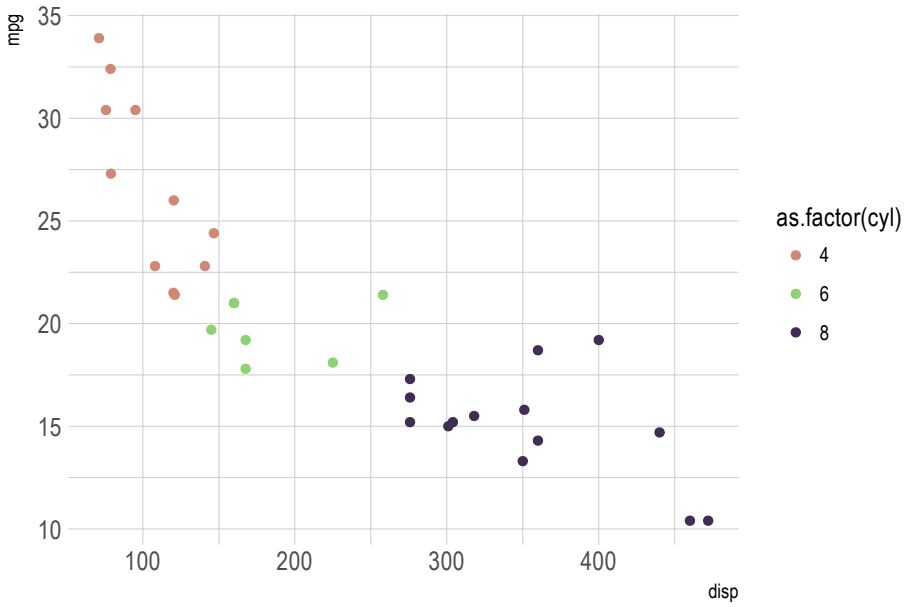

```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) +
    geom_point() +
    theme_void()
```



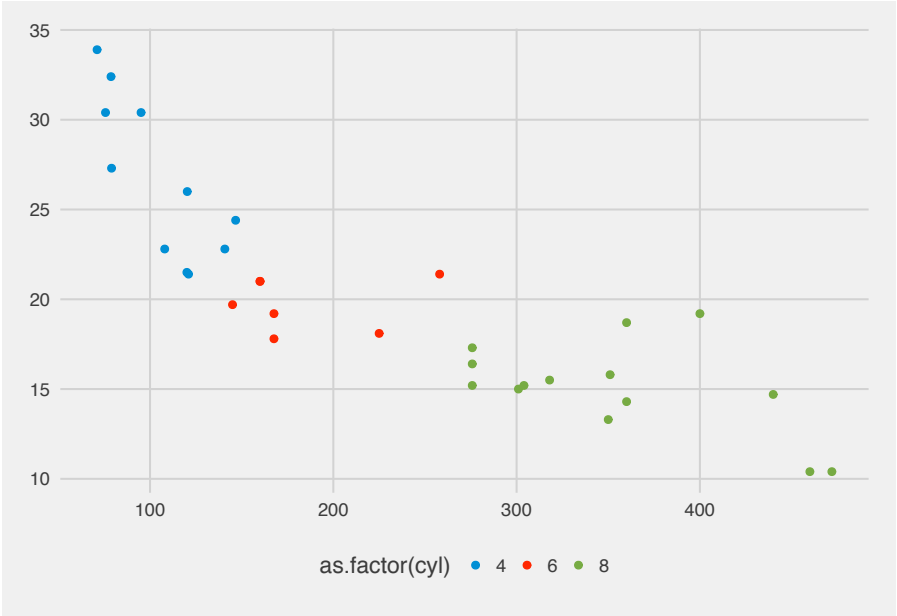
```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) +
    geom_point() +
  theme_minimal(base_size = 18, base_family = "Avenir")
```



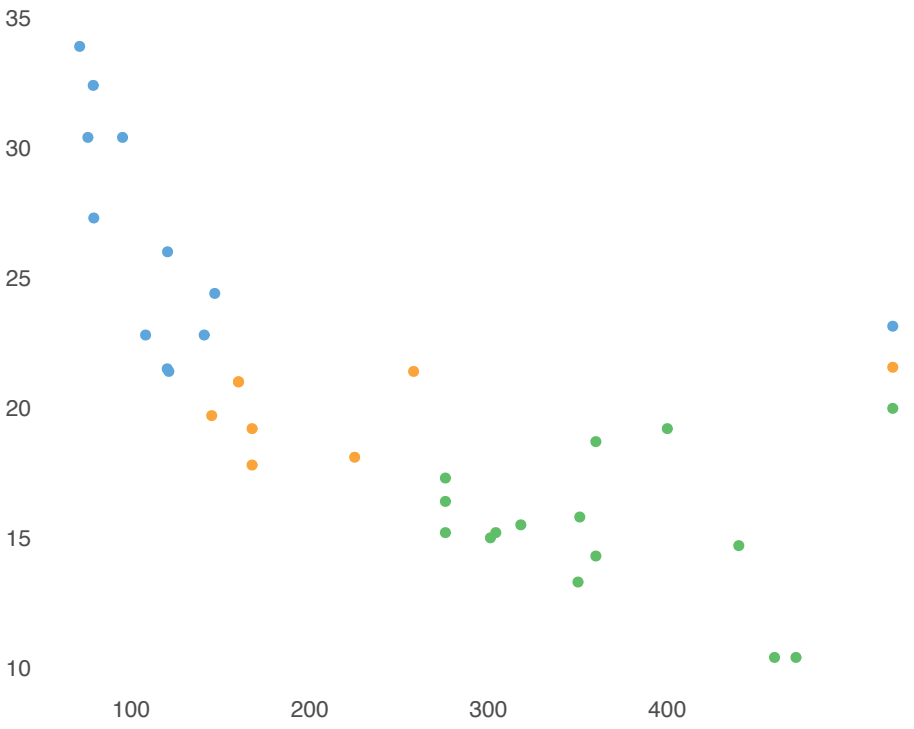

```
#ipsum, latin for neat

library(hrbrthemes)

ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) +
  geom_point() +
  scale_colour_ipsum() +
    theme_ipsum()
```



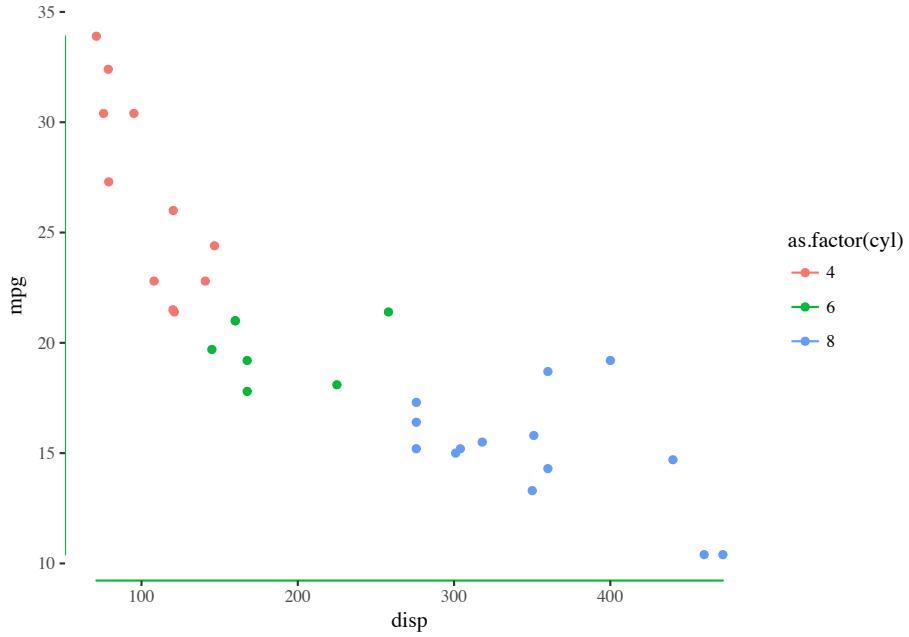

```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) +
  geom_point() +
  scale_color_fivethirtyeight() +
    theme_fivethirtyeight()
```



```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) +
  geom_point()+
  scale_color_few() +
    theme_few()
```

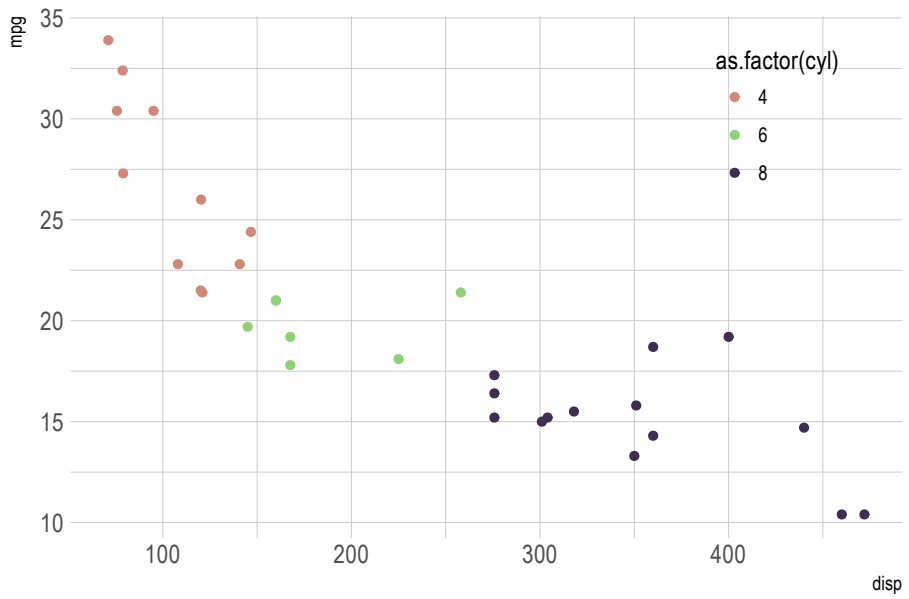



```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) +
  geom_point()+
    geom_rangeframe()+
    theme_tufte()
```

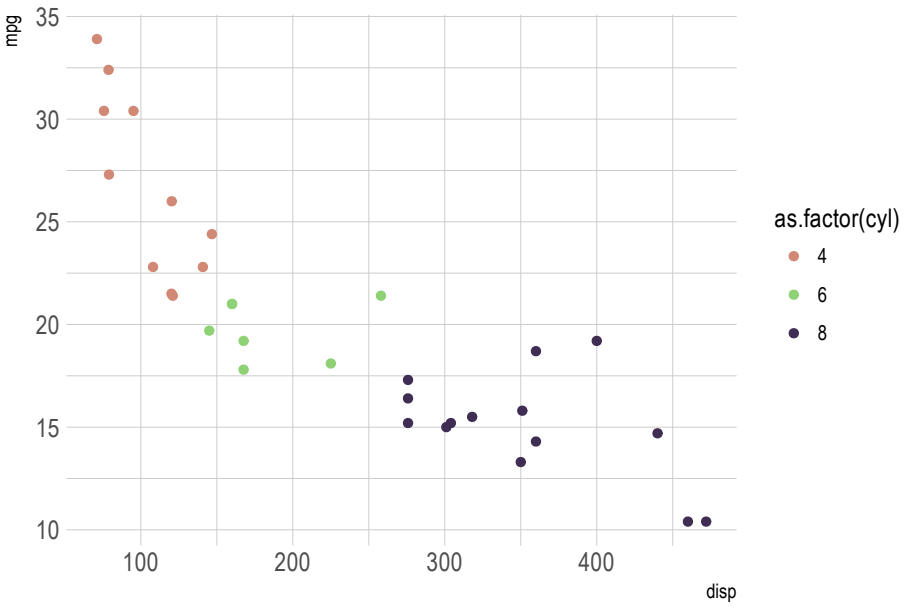


order of application matters

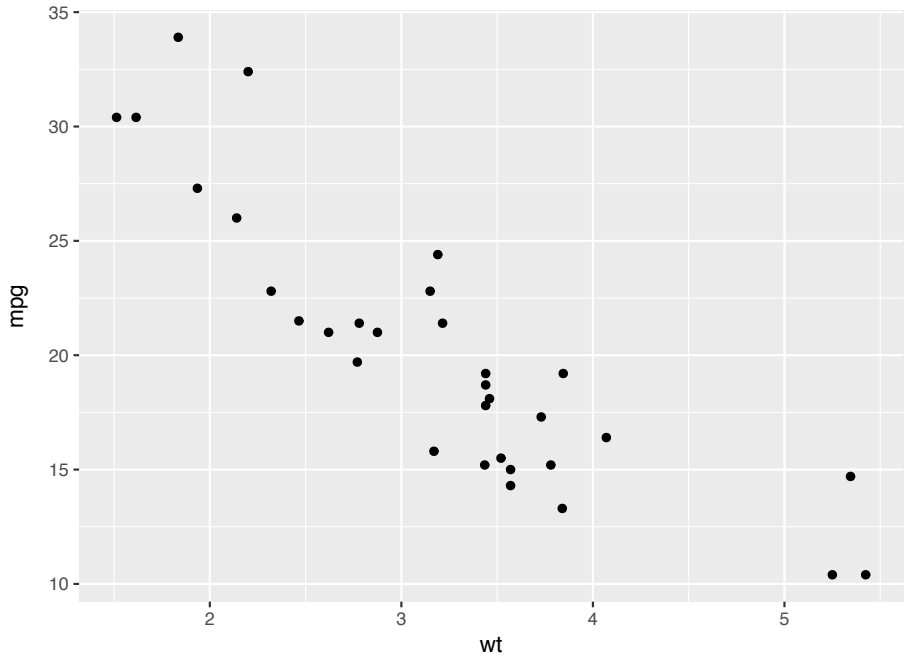
```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) + geom_point()+  scale_colour_ipsum
theme_ipsum() +
theme(legend.position = c(.85, .8))
```



```
ggplot(mtcars, aes(disp, mpg, color = as.factor(cyl))) + geom_point()+  scale_colour_ipsum
theme_ipsum() +
theme(legend.position = c(.85, .8))+
theme_ipsum()
```

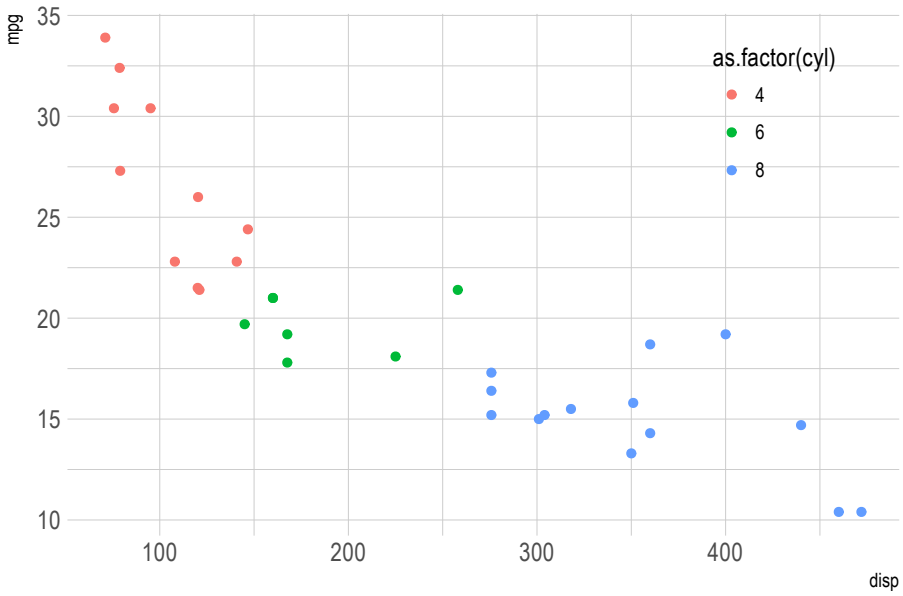



```
ggplot(mtcars.df, aes(wt, mpg)) +
  geom_point()+
  theme(axis.title.y = element_text(margin = margin(t = 10, r = 10, b = 80, l = 20)))
```

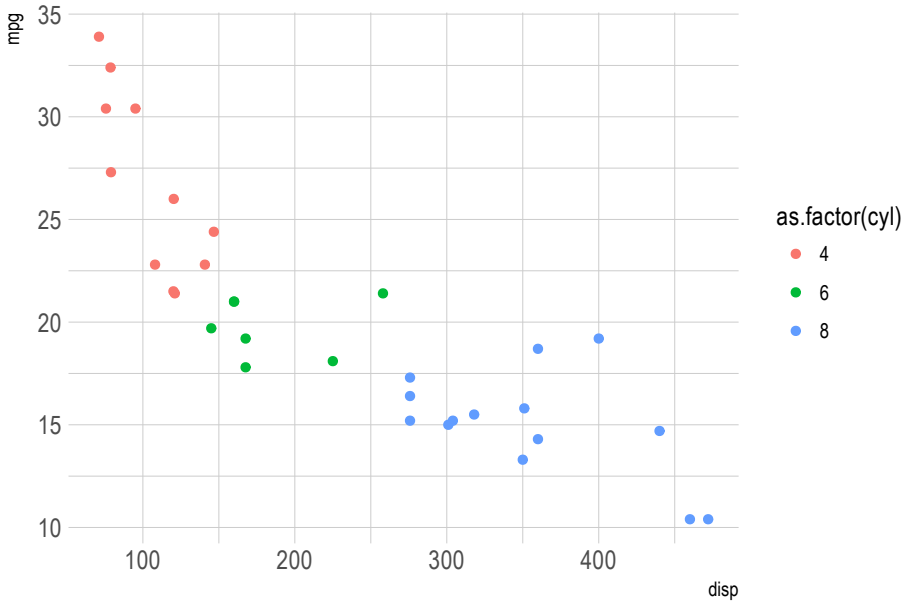


0.60.3 Ordering theme layers

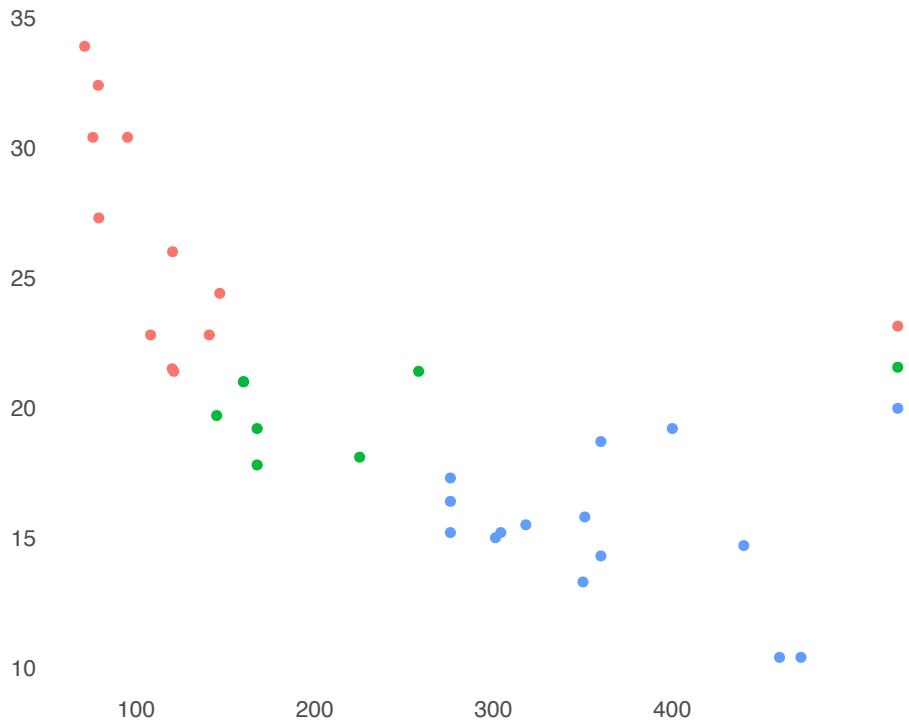
```
# Sets theme then adjusts legend
ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point()+
  theme_ipsum()+
  theme(legend.position = c(.85, .8))
```




```
# Adjusts legend, but the overrides with setting theme
ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point()+
  theme(legend.position = c(.85, .8))+
  theme_ipsum()
```

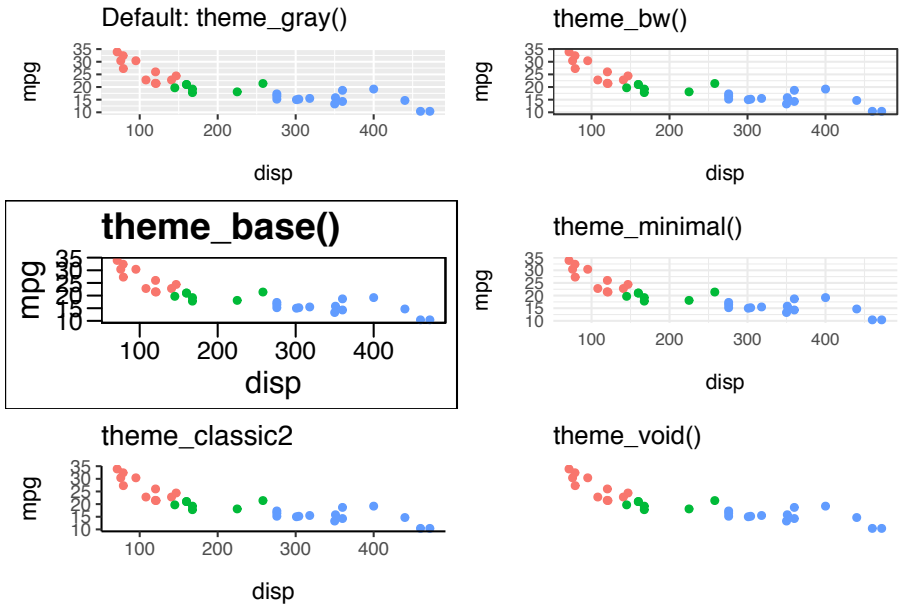


```
## To set for all plots
theme_set(theme_few())
ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point()
```




```
theme_set(theme_gray()) # Returns to default theme
```

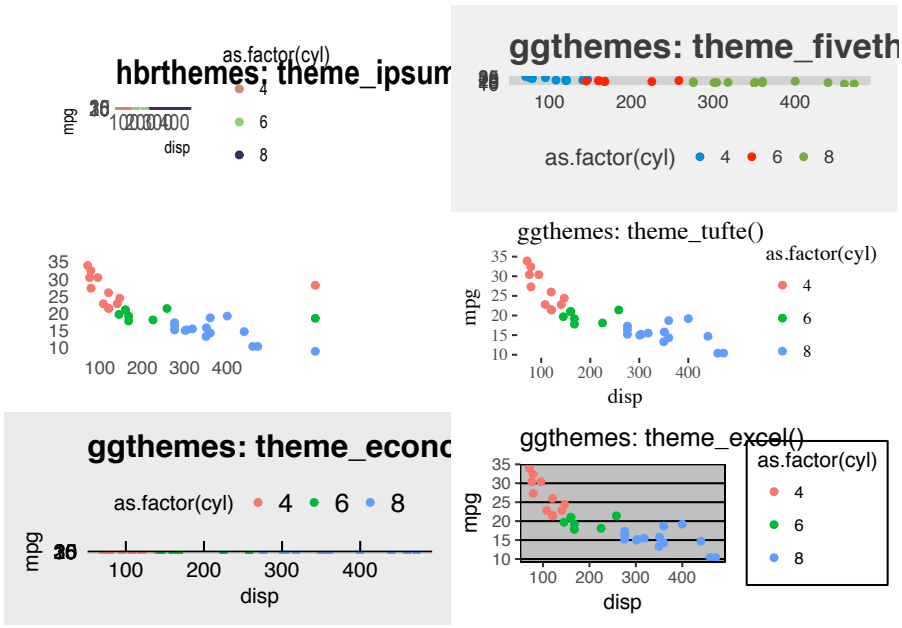
0.60.4 Pre-set theme options



0.60.5 Themes form other packages

```
## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database
```

```
## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in grid.Call(C_textBounds,
## as.graphicsAnnot(x$label), x$x, x$y, : font family
## 'Arial Narrow' not found in PostScript font database

## Warning in align_plots(plotlist = plots, align = align,
## axis = axis): Complex graphs cannot be vertically
## aligned unless axis parameter is set properly. Placing
## graphs unaligned.

## Warning in align_plots(plotlist = plots, align = align,
## axis = axis): Graphs cannot be horizontally aligned,
## unless axis parameter set. Placing graphs unaligned.
```



0.61 Saving and printing plots to include in documents

For formal documents save graphs and import. Do not cut and paste from RStudio Saving and importing provides consistent physical size, resolution, and aspect ratio: DO NOT re-scale in the document Vector formats (PDF, SVG)

Provide crisp images even when zoomed in and raster File size scales with number of data points Raster formats (PNG, TIFF)

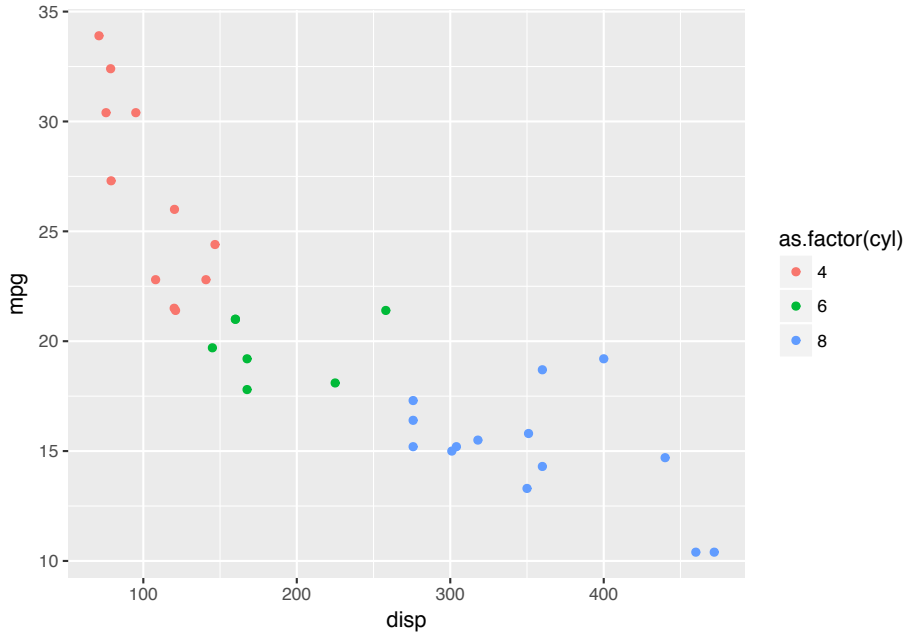
The dpi (dots per inch) defines the resolution of the image File size scales with dimensions of graph and dpi

0.61.1 PNG, JPEG, PDF, and SVG

```
mpg.plot = ggplot(mtcars,
    aes(disp,  mpg, color = as.factor(cyl))) +
    geom_point()
```

```
ggsave(filename = "mpg.png",
    device = "png",
    plot = mpg.plot, height = 4, width = 5, units = "in",
    dpi = 300)

ggsave(filename = "mpg.pdf", device = "pdf",
      plot = mpg.plot, height = 4, width = 5, units = "in")
```

```
library(svglite)
ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point()
```



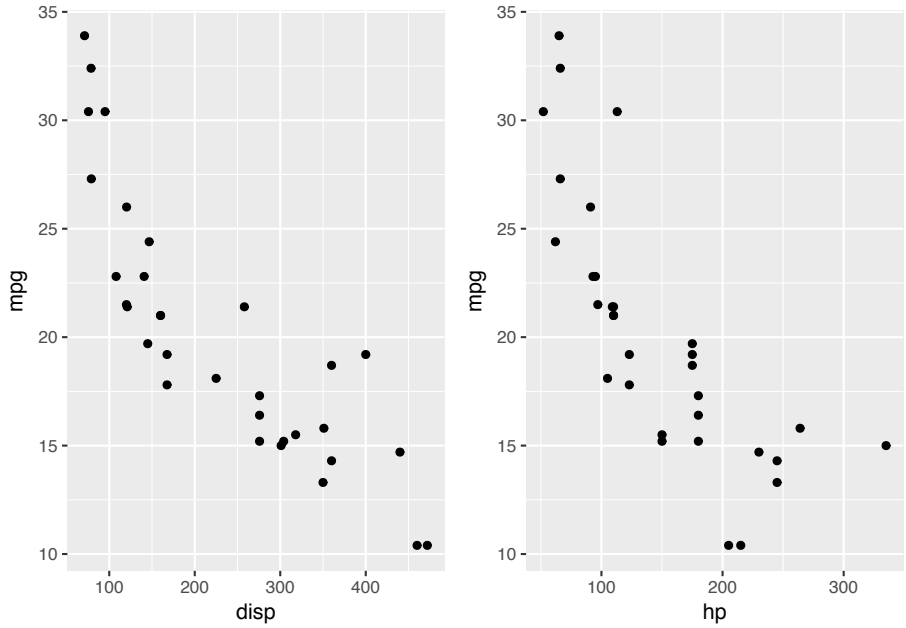

```
mpg.plot = ggplot(mtcars, aes(disp,  mpg, color = as.factor(cyl))) + geom_point()

## Possible formats:
#  "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf" (windo

## PNG--raster format that looks blurry at low resolution
ggsave(filename = "mpg.png", device = "png",
       plot = mpg.plot, height = 4, width = 5, units = "in", dpi = 300)

## PDF--vector format that remains sharp even when zoomed in
```

```
ggsave(filename = "mpg.pdf", device = "pdf",
       plot = mpg.plot, height = 4, width = 5, units = "in")

## SVG--vector format that remains sharp even when zoomed in
ggsave(filename = "mpg.svg", device = "svg",
       plot = mpg.plot, height = 4, width = 5, units = "in")


## Saving many plots into a single file
# Calculate the number of pages with 9 panels per page
n_pages <- ceiling(
  length(levels(diamonds$color)) * length(levels(diamonds$cut:diamonds$clarity)) / 9
)

pdf("multipage.pdf")
  for (i in seq_len(n_pages)) {
    p=  ggplot(diamonds, aes(carat, price)) +
          geom_point(alpha = 0.1) +
          facet_grid_paginate(color~cut:clarity, ncol = 3, nrow = 3, page = i)+
        labs(title = "ggforce: facet pagination")
    print(p)
}
dev.off()
```

```
## cairo_pdf
##         2
```

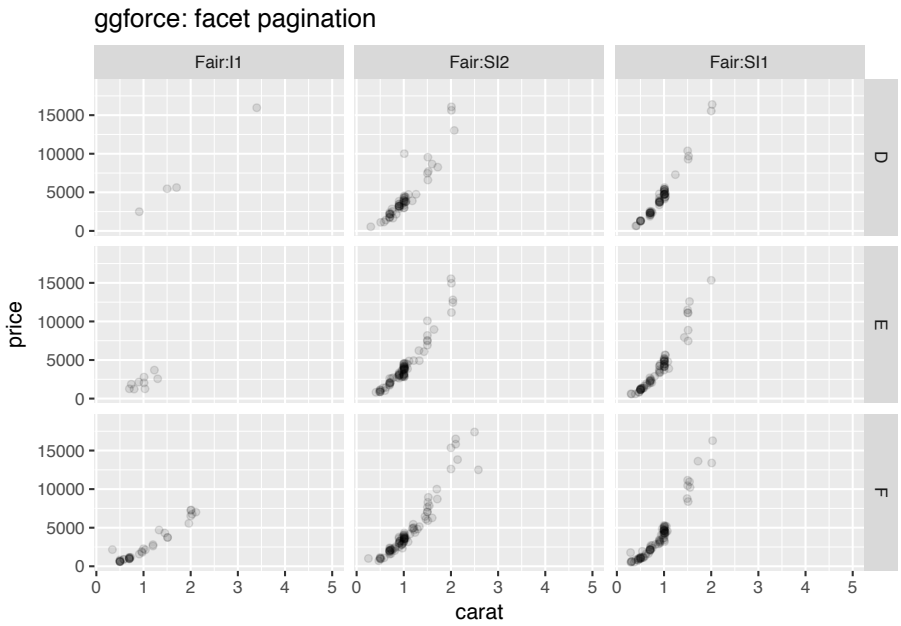
0.61.2 Combining multiple graphs for publication

```
library(ggpubr)

displ.plot = ggplot(mtcars, aes(disp, mpg)) + geom_point()
hp.plot = ggplot(mtcars, aes(hp, mpg)) + geom_point()

combined.plot = ggarrange(displ.plot, hp.plot, nrow=1, ncol = 2, align = "hv")
combined.plot
```



0.61.3 Faceted pagination

```
library(ggforce)
## Examples from: https://cran.r-project.org/web/packages/ggforce/vignettes/Visual_Guide.h


ggplot(diamonds, aes(carat, price)) +
  geom_point(alpha = 0.1) +
  facet_grid_paginate(color~cut:clarity, ncol = 3, nrow = 3, page = 1)+
  labs(title = "ggforce: facet pagination")
```


ggforce: facet pagination
Fair:I1
Fair:SI2
Fair:SI1
D
E
F
price
carat
0
5000
10000
15000
0
1
2
3
4
5

0

Interaction–zoom, annotate, highlight

```
library(tidyverse)
```

```
library(ggiraph)

mpg.plot =  ggplot(mpg, aes( x = displ, y = cty, color = hwy))+     geom_point_interactive

ggiraph(code =  print(mpg.plot))
```

0

Shiny–Advanced interactive graphics

```
library(tidyverse)
```

.1 Data sources

https://www.kaggle.com

.2 Visualization resources

Final commments at end.