Coding Grace - ggplot2 Workshop 2016

About ggplot2

- The ggplot2 package, created by Hadley Wickham, offers a powerful graphics language for creating elegant and complex plots. Its popularity in the R community has exploded in recent years.
- Originally based on Leland Wilkinson's The Grammar of Graphics, ggplot2 allows you to create graphs that represent both univariate and multivariate numerical and categorical data in a straightforward manner.

Benefits of ggplot2

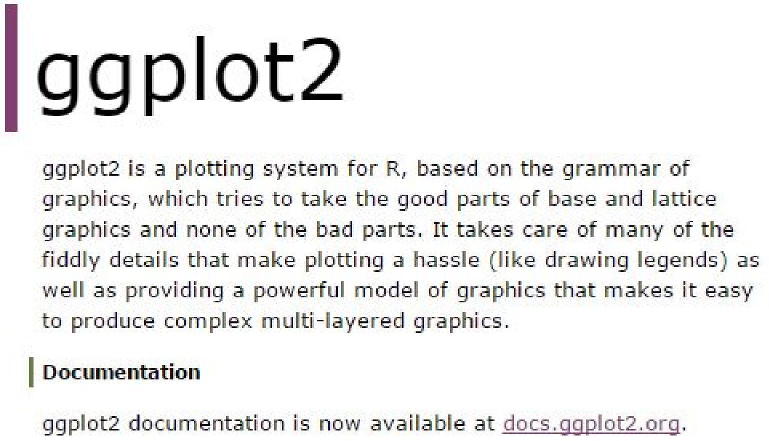
1. ggplot is easy to learn [*1*]
2. ggplot is fun
3. ggplot is powerful [*2*]

[1] Lots of learning resources, mainly intended for the R environment, that can applied to Python also.

[2] Less code required to compute high-level publication quality plot

(source: www.yhat.com)

Hadley Wickham (Chief Data Scientist, RStudio)



ggplot2

ggplot2 is a plotting system for R, based on the grammar of graphics, which tries to take the good parts of base and lattice graphics and none of the bad parts. It takes care of many of the fiddly details that make plotting a hassle (like drawing legends) as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

Documentation

ggplot2 documentation is now available at docs.ggplot2.org.

website: www.had.co.nz

Use R!

Hadley Wickham

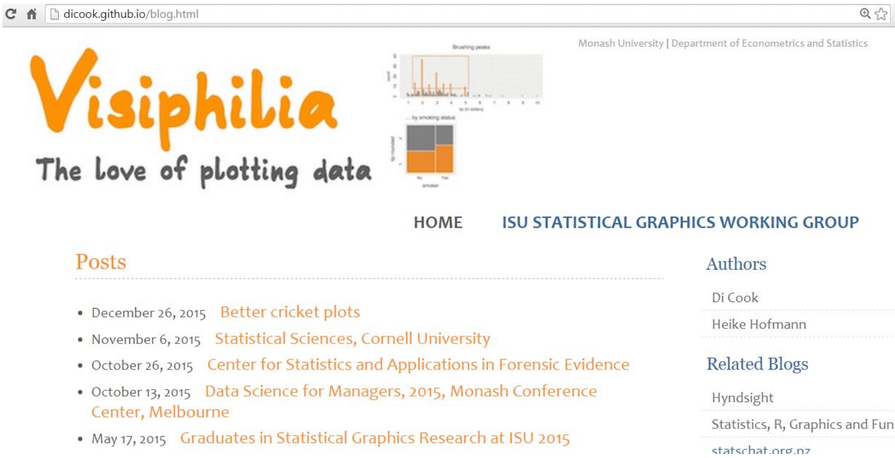
ggplot2

Elegant Graphics for Data Analysis

Springer

ggplot2: Elegant Graphics for Data Analysis

Prof. Dianne Cook (Monash University)



dicook.github.io/blog.html

Monash University | Department of Econometrics and Statistics

Visiphilia

The love of plotting data

HOME   ISU STATISTICAL GRAPHICS WORKING GROUP

Posts

- December 26, 2015 Better cricket plots
- November 6, 2015 Statistical Sciences, Cornell University
- October 26, 2015 Center for Statistics and Applications in Forensic Evidence
- October 13, 2015 Data Science for Managers, 2015, Monash Conference Center, Melbourne
- May 17, 2015 Graduates in Statistical Graphics Research at ISU 2015

Authors

- Di Cook
- Heike Hofmann

Related Blogs

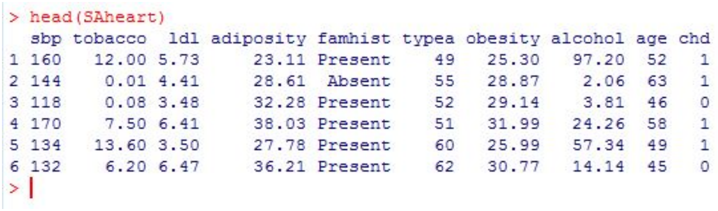
- Hyndsight
- Statistics, R, Graphics and Fun
- statschat.org.nz

1. Installing ggplot2

```
install.packages("ggplot2")

library(ggplot2)
```

N.B. Not installed automatically with R.

Data

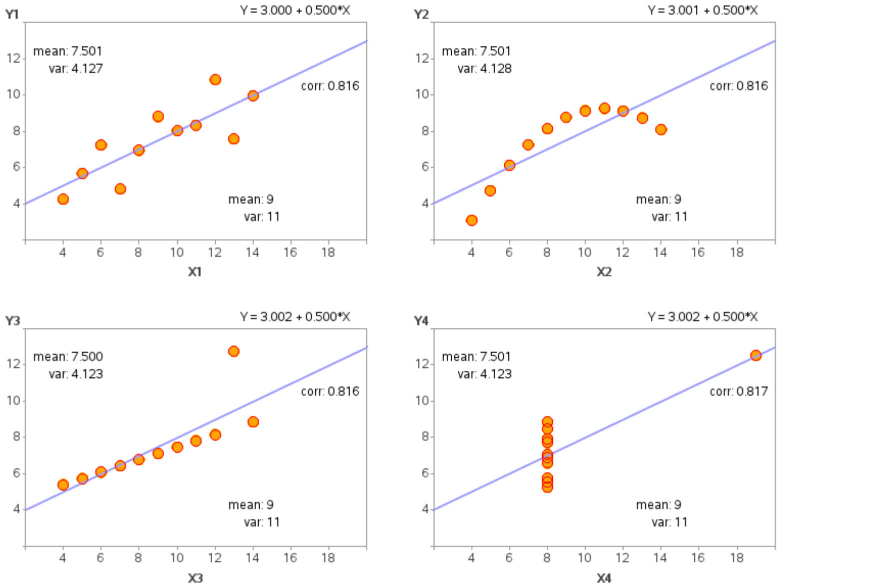


```
> head(SAheart)
  sbp tobacco  ldl adiposity famhist typea obesity alcohol age chd
1 160   12.00 5.73     23.11 Present    49   25.30   97.20  52   1
2 144    0.01 4.41     28.61  Absent    55   28.87    2.06  63   1
3 118    0.08 3.48     32.28 Present    52   29.14    3.81  46   0
4 170    7.50 6.41     38.03 Present    51   31.99   24.26  58   1
5 134   13.60 3.50     27.78 Present    60   25.99   57.34  49   1
6 132    6.20 6.47     36.21 Present    62   30.77   14.14  45   0
> 
```

- If you're planning on using ggplot2, it's best to keep your data in `data.frame` .
- Think of a `data.frame` as a tabular data object.

Anscombe Quartet

- In 1973, the statistician Francis Anscombe demonstrated the importance of graphing data.
- The Anscombe's Quartet shows how four sets of data with identical simple summary statistics can vary considerably when graphed.



(Source: Wikipedia)

Try this out!

```
ggplot(aes(x=x1, y=y1), data=anscombe2)
 + facet_wrap(group,scales=fixed)
 + geom_point()
 + stat_smooth(method="lm")
```

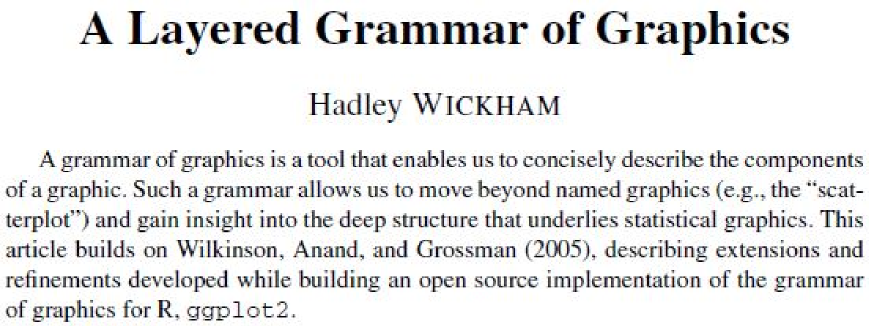
ggplot - Inbuilt Data Sets

Data sets included in ggplot2 and used in examples

diamonds	Prices of 50,000 round cut diamonds
faithfuld	2d density estimate of Old Faithful data
midwest	Midwest demographics.
mpg	Fuel economy data from 1999 and 2008 for 38 popular models of car
msleep	An updated and expanded version of the mammals sleep dataset.
txhousing	Housing sales in Texas.

- The main command is `ggplot()`.
- The name comes from "**grammar of graphics**", a book by Leland Wilkinson
- A very "high-level" approach to data visualization.

> *A grammar of graphics is a tool that enables us to concisely describe the components of a graphic. Such a grammar allows us to move beyond named graphics (e.g., the scatterplot) and gain insight into the deep structure that underlies statistical graphics.*



A Layered Grammar of Graphics

Hadley WICKHAM

A grammar of graphics is a tool that enables us to concisely describe the components of a graphic. Such a grammar allows us to move beyond named graphics (e.g., the "scatterplot") and gain insight into the deep structure that underlies statistical graphics. This article builds on Wilkinson, Anand, and Grossman (2005), describing extensions and refinements developed while building an open source implementation of the grammar of graphics for R, `ggplot2`.

Hadley Wickham.
A layered grammar of graphics.
Journal of Computational and Graphical Statistics,
vol. 19, no. 1, pp. 328, 2010.

Basic Premise

- Making plots is a very repetive: draw this line, add these colored points, then add these, etc.
- Instead of re-using the same code over and over, `ggplot` implements them using a high-level but very expressive API.
- The result is less time spent creating your charts, and more time interpreting what they mean.

(From ggplot documentation)

Basic Premise

- `ggplot` is not a good fit for people trying to make highly customized data visualizations.
- *(Compare this to high level "Bokeh" plots)*
- While you can make some very intricate, great looking plots, `ggplot` sacrifices highly customization in favour of general doing "what you'd expect".

(From ggplot documentation)

A **plot** is made up of multiple layers.

A **layer** consists of **data**, a set of **mappings** between variables and aesthetics, a **geom**etric object and a **stat**istical transformation

Scales control the details of the mapping.

All components are independent and reusable.

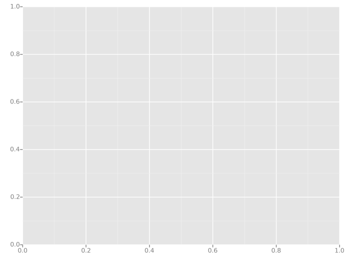
Layers

Layers

- **ggplot** lets you combine or add different types of visualization components (or layers) together.
- The command `ggplot()` does not actually create any plot, rather it prepares a "blank canvas" for further plotting .
- (We will introduce *geoms* shortly.)

Start with a blank canvas.

```
# meat data set (meat2.csv)

p <- ggplot(aes(x=date, y= beef), data=meat2)
p
```

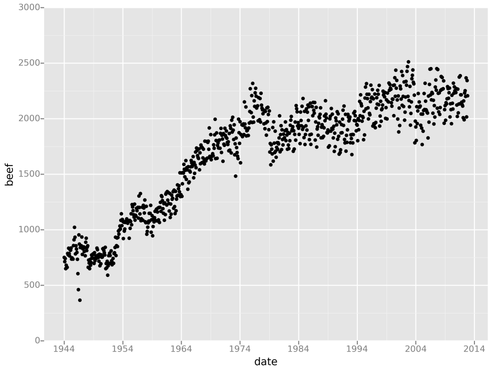


Lets try these exercises out with different data sets (both inbuilt)

```
p1 <- ggplot(aes(x=wt, y=mpg),
       data=mtcars)

p2 <- ggplot(aes(x=depth, y=carat),
               data=diamonds)
```

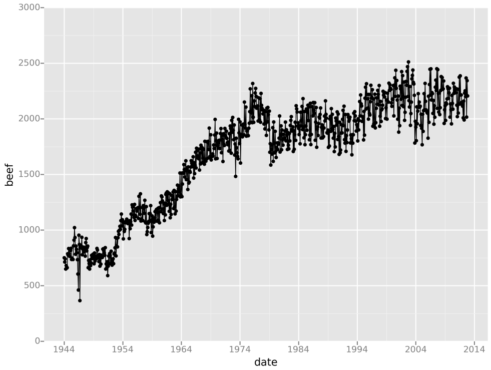
Add some points.

```
p + geom_point()
```



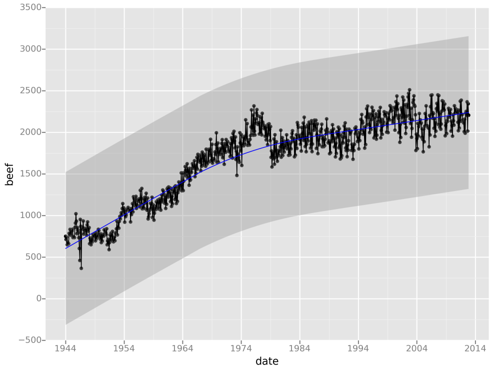
Add a line.

```
p + geom_point() + geom_line()
```



Add a trendline.

```
p + geom_point() + geom_line() +
stat_smooth(color="blue")
```



qplot

More!

```
p + ggtitle("Plot Title")
  + xlab(" X Axis Label")
  + ylab(" Y Axis Label")
```

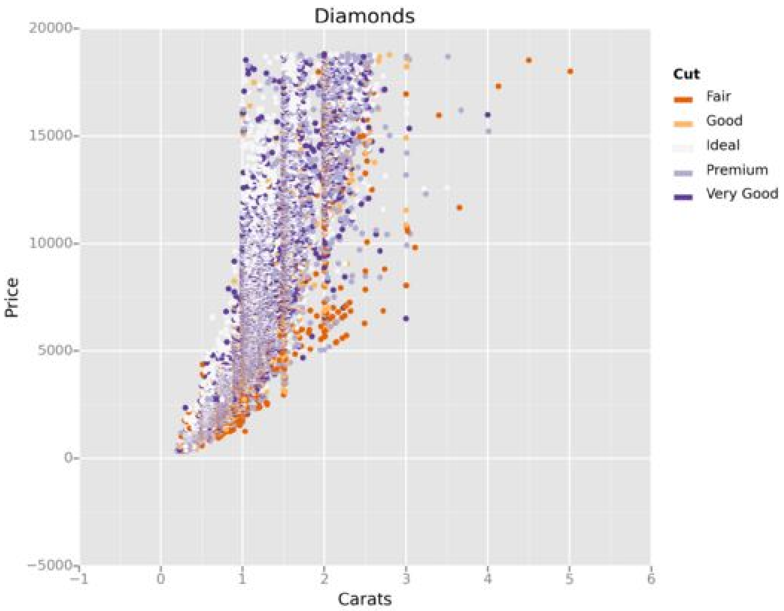
Aesthetics

Rule of Thumb

- The `aes` argument is for aesthetics
- Essentially it identifies which variables are being used, and in what order.
- The first two variables are the “X” and “Y” variable.
- Anymore variables after that are typically grouping variables (categorical).

Aesthetics

```
ggplot(diamonds, aes(x=carat,
  y=price, color=cut))+
  geom_point()
```



Diamonds
Price
Carats
20000
15000
10000
5000
0
−5000
−1
0
1
2
3
4
5
6
Cut
Fair
Good
Ideal
Premium
Very Good

geoms

Geometric objects (geoms) are the visual representations of (subsets of) observations.

- **Univariate** - single numeric variable
- **Bivariate** - two numeric variable
- **Multivariate** - Multiple variables

geoms for ggplot

Here are some of the geoms currently available.

geom_abline	geom_histogram	geom_pointrange
geom_area	geom_hline	geom_rect
geom_bar	geom_jitter	geom_smooth
geom_blank	geom_line	geom_step
geom_boxplot	geom_linerange	geom_text
geom_density	geom_path	geom_tile
geom_dotplot	geom_point	geom_vline

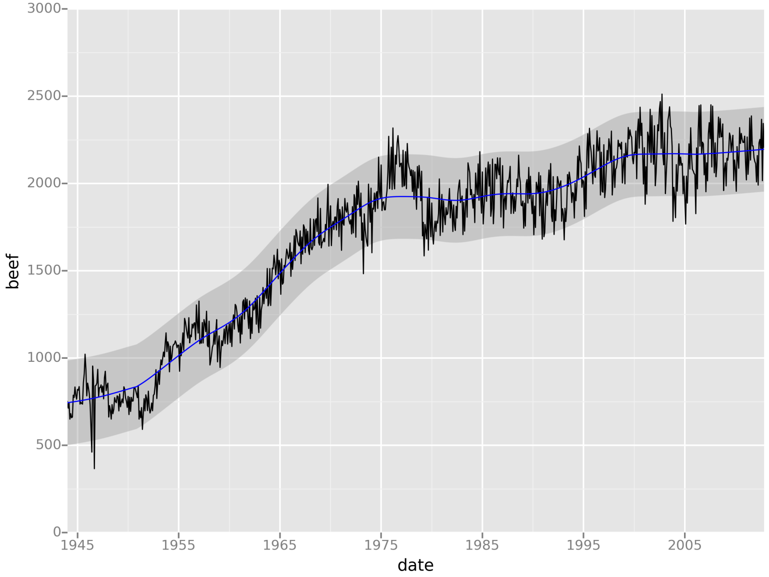
stats

- *Stats* apply statistical transformations that are used to summarise the data, and allows a huge range of possibilities.
- `Stat_smooth` is a nice stat to illustrate the principles, which fits a line and a shaded band to indicate some specified level of uncertainty, as shown in the following example which fits a linear regression line.

stats

```
ggplot(aes(x=date y=beef), data=meat2) +
geom_line() +
stat_smooth(colour='blue', span=0.2)
```

- Try this out for other data sets



3000
2500
2000
1500
1000
500
0
beef
1945
1955
1965
1975
1985
1995
2005
date

stats for ggplot

These are the stats currently available for ggplot in python.

stat_abline	stat_hline
stat_bar	stat_identity
stat_bin	stat_smooth
stat_bin2d	stat_summary
stat_density	stat_vline
stat_function	

qplot

quickplot

- `qplot()` is the basic plotting function in the ggplot package, designed for quick inspections of the data.
- The functionality is not as expansive as with using “`ggplot()`”.
- For the most part, the same code works for both.

ggplot2 - faceting

Facetting

- One of the most useful techniques in data visualization is rendering groups of data alongside each other, making it easy to compare the groups.
- With ggplot2, one way to do this is by mapping a discrete variable to an aesthetic, like x position, color, or shape.
- Another way of doing this is to create a subplot for each group and draw the subplots side by side.
- ggplot2 has two ways to do this: `facet_grid()` and `facet_wrap()`.

Faceting

The faceting approach supported by ggplot partitions a plot into a matrix of panels. Each panel shows a different subset of the data. There are two faceting approaches:

- `facet_wrap("cell")` - univariate: create a 1-d strip of panels, based on one factor, and wrap the strip into a 2-d matrix
- `facet_grid("row","col")` - (usually) bivariate: create a 2-d matrix of panels, based on two factors

Faceting

Quick Example of Faceting

Suppose `cyl` and `drv` are two categorical variables in the **mpg** data frame

```
qplot(.....) + facet_grid(cyl,drv)
```

Faceting

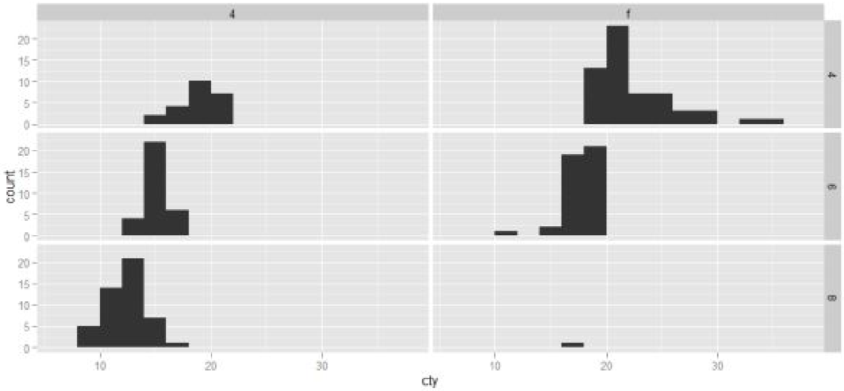


Figure: Grid Faceting

Faceting

Facet Wrap

- An alternative to grid facetting is a wrapped ribbon of plots
- `facet_wrap` generates a long ribbon of plots, and wraps it into 2d.

ggplot2 - faceting

Facet Grid
With `facet_grid()`, you can specify a variable to split the data into vertical subpanels, and another variable to split it into horizontal subpanels

```
# The base plot
p <- ggplot(diamonds,
 aes(x = price, y = depth)) + geom_point()

# Faceted by color
# vertically arranged subpanels
p + facet_grid(color ~ .)
```

ggplot2 - faceting

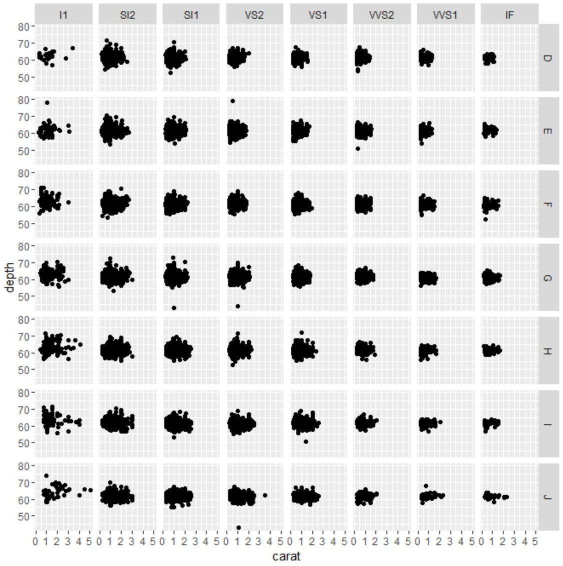
```
# Faceted by clarity
# in horizontally arranged subpanels

p + facet_grid(. ~ clarity)
```

ggplot2 - faceting

Putting Both Together

```
# Split by color (vertical) and clarity (horizontal)
p + facet_grid(color ~ clarity)
```

ggplot2 - faceting



ggplot2 - faceting

Facet Wraps

- `facet_wrap()` creates and labels a plot for every level of a factor which is passed to it.
- The primary argument takes the form of a one sided formula: ∼`Factor`.
- this will then efficiently "wrap these plots into a 2d grid.

```
p + facet_wrap(~clarity)
```

ggplot2 - faceting

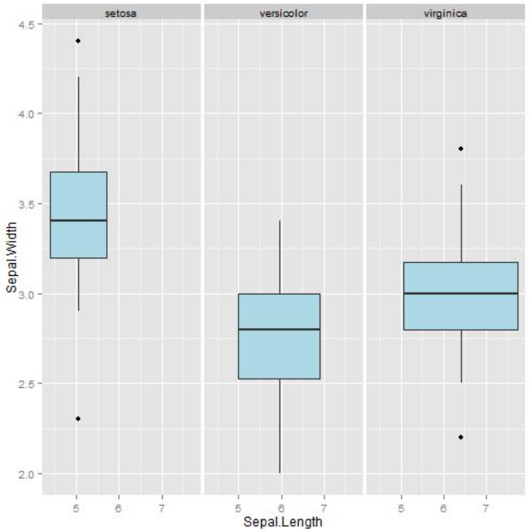
```
# These will have the same result:
# 2 rows and 4 cols

p + facet_wrap(~clarity, nrow = 2)

p + facet_wrap(~class, ncol = 4)

#Try some variants of this
```

ggplot2 - faceting

Facet Scales

- Usually you will want all of your facets to have the same x and y scales.
- If you're plotting the same data in each facet, having free scales on each of the facets will ruin comparability across facets.
- However, sometimes it will be appropriate to have free scales. You can do this by passing `scales = "free"` to `facet_wrap()`.

```
p + facet_wrap(~clarity, scales = "free")
```


setosa
versicolor
virginica
Sepal.Width
4.5
4.0
3.5
3.0
2.5
2.0
5
6
7
Sepal.Length

Scales and Themes

- ggplot2 provides a large number of scale functions to control aspects of a graphic including axes and legends
- theme functions allow us to control the overall style of the graphic

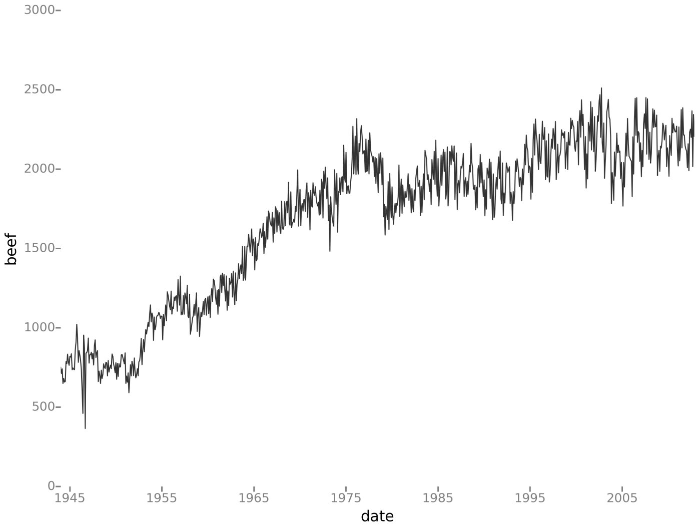
Scales

- A scale determines how an attribute of the data is mapped into an aesthetic property of a `geom` (e.g., the geom's position along the x axis, or a geom's fill color in a color space).
- The colours and shapes used in the chart can be manually adjusted if you dont like the defaults.

Themes

```
ggplot(aes(x=date, y=beef), meat2) +
geom_line() +
theme_bw()
```

Themes


3000
2500
2000
1500
1000
500
0
beef
1945
1955
1965
1975
1985
1995
2005
date

Themes

Try out the following themes:

- `theme_538`
- `theme_bw`
- `theme_gray`
- `theme_matplotlib`
- `theme_seaborn`
- `theme_xkcd`