WiMa-Praktikum II Stochastic

R Graphics

Arthur Allignol

arthur.allignol@uni-ulm.de

Table of Contents

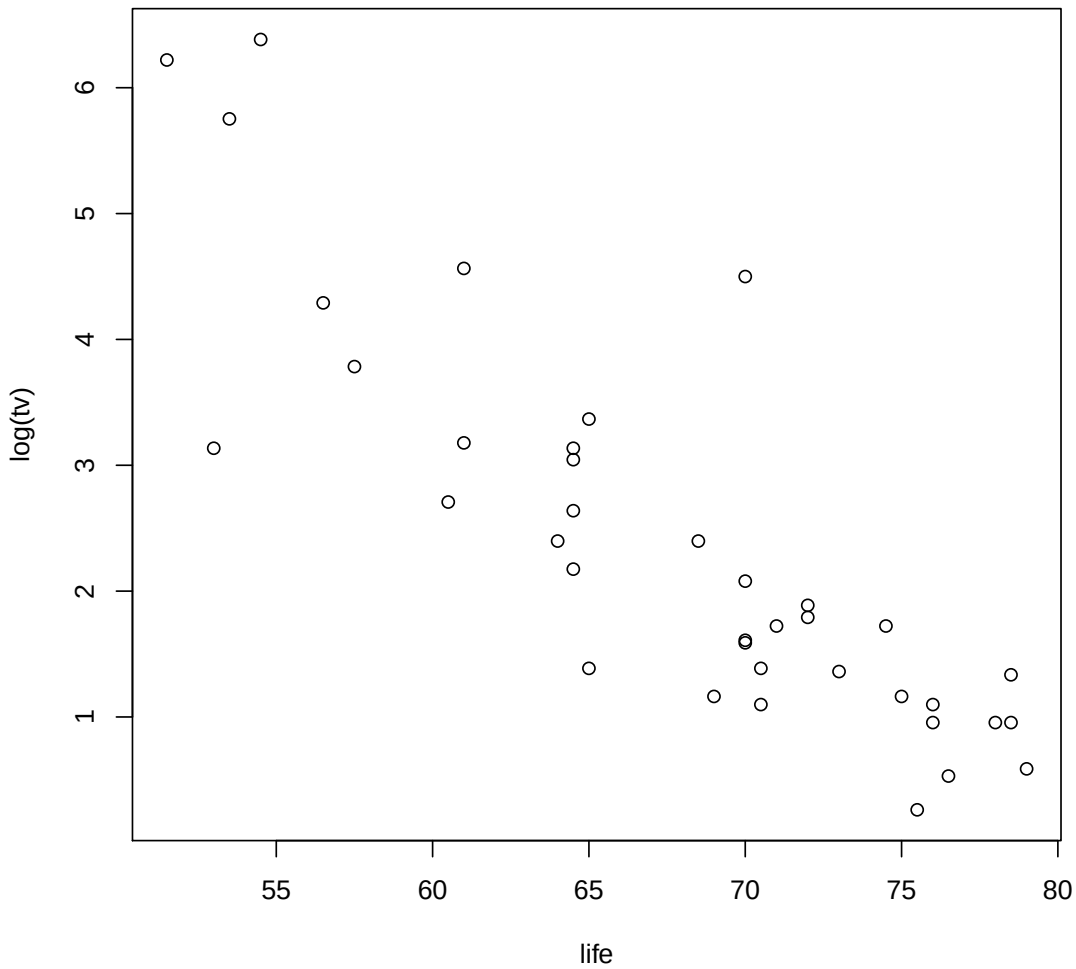
Introduction

Why do we need graphics?

- Data exploration
  - Look for trends and/or associations between variable
  - The first step before modelling
  - $\rightarrow$ quick and dirty graphs
- Check assumptions of statistical models
- These graphics are usually for you

Introduction

Data exploration



Introduction

Model checking

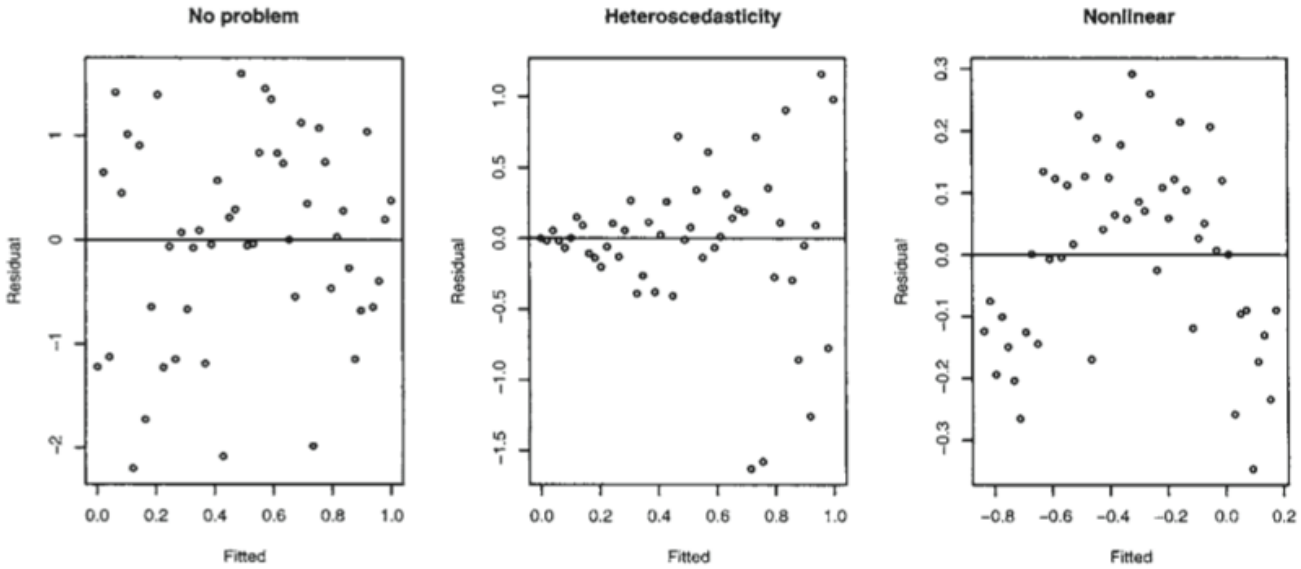


Figure 4.1 *Residuals vs. fitted plots—the first suggests no change to the current model while the second shows nonconstant variance and the third indicates some nonlinearity, which should prompt some change in the structural form of the model.*

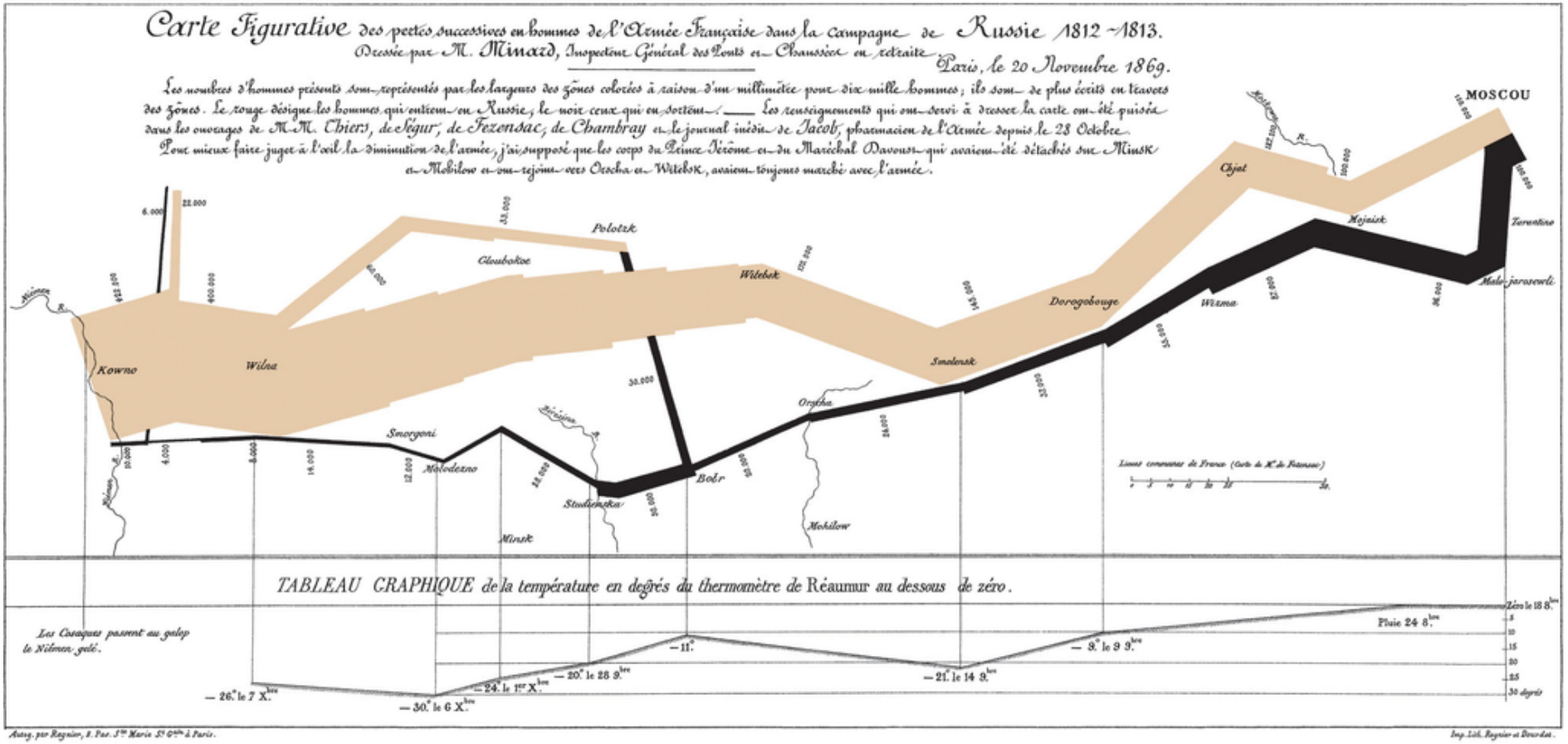
Introduction

Why do we need graphics?

- Information visualisation/Communication
  - Need a lot of polishing
  - Iteration is crucial
  - Think about where you present the graphics, e.g, colour, line thickness for a beamer presentation

Introduction

Minard's Flow Map



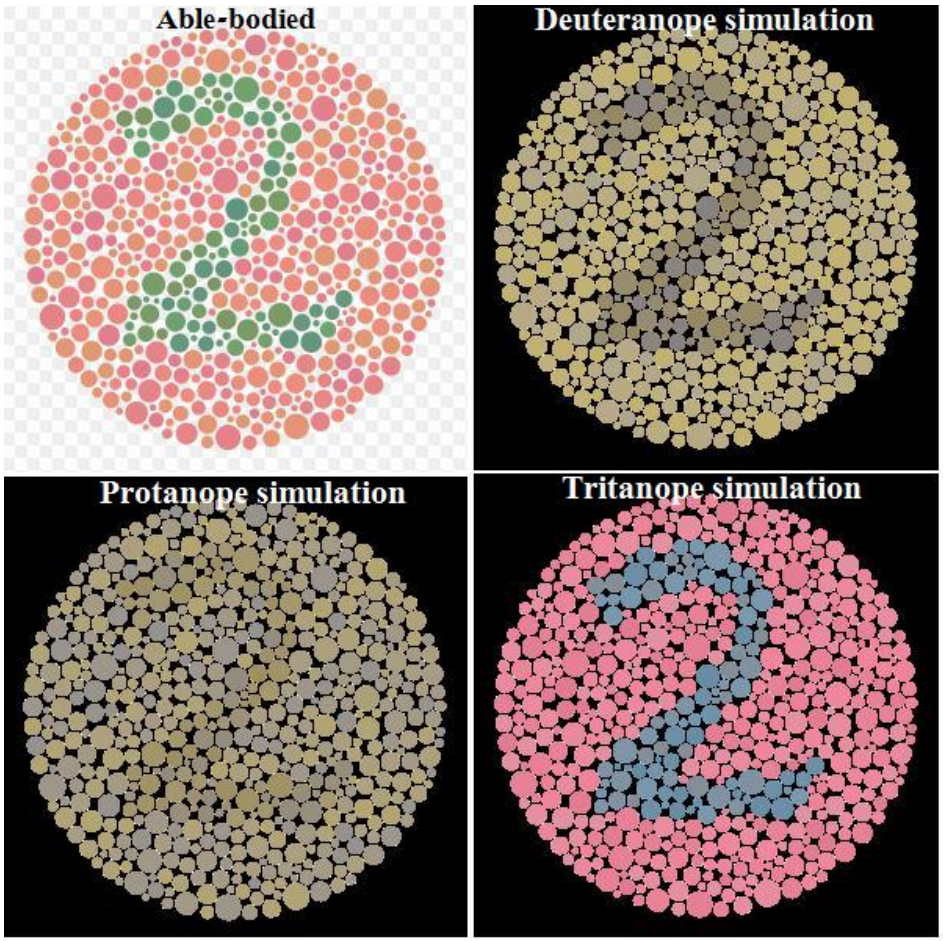
Why R for Graphics

It's the best for the price

Some Visualisation Tips

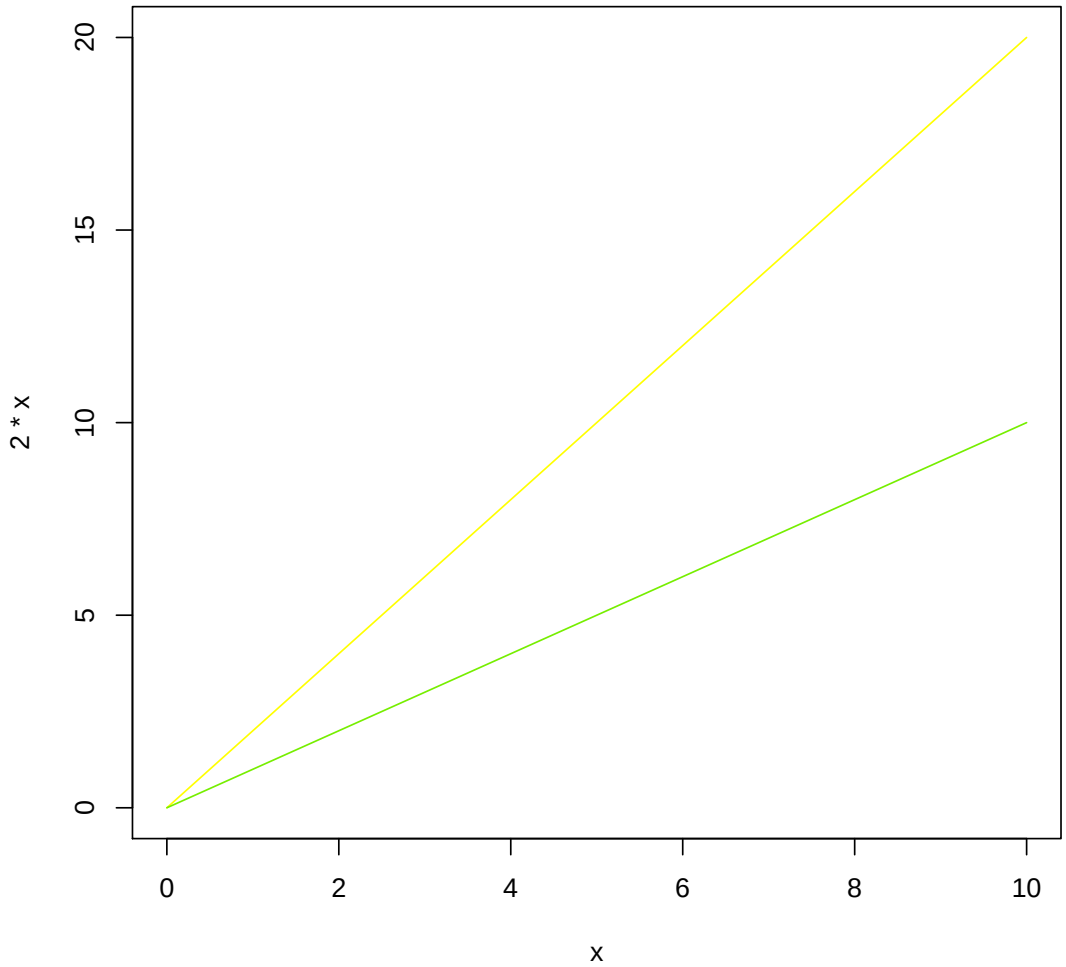
Colours

- 7-10% of people are red-green colour blind



Some Visualisation Tips

Colours



Some Visualisation Tips

Colours

Sequential palettes: for ordered data from low to high

Some Visualisation Tips

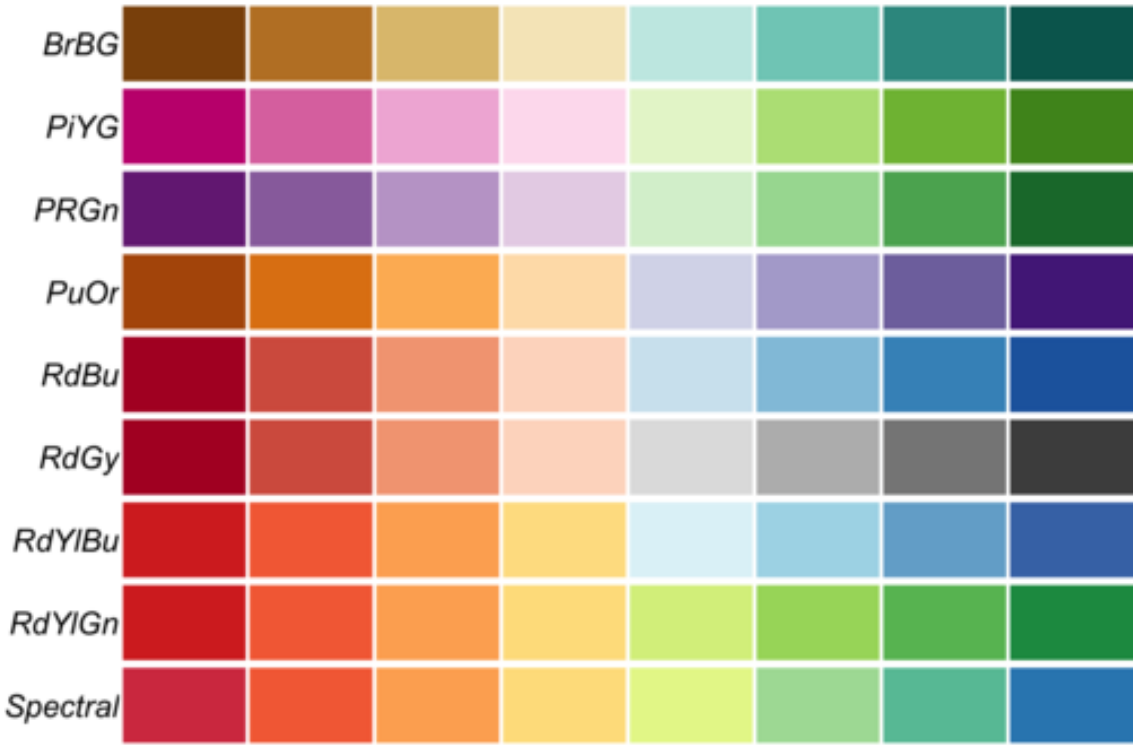
Colours

Qualitative palettes: suited to representing nominal or categorical data

Some Visualisation Tips

Colours

Diverging palettes: useful when the data has a natural, meaningful break-point (as with correlation values, which are spread around zero)



See `http://colorbrewer2.org/`
`http://tools.medialab.sciences-po.fr/iwanthue/`

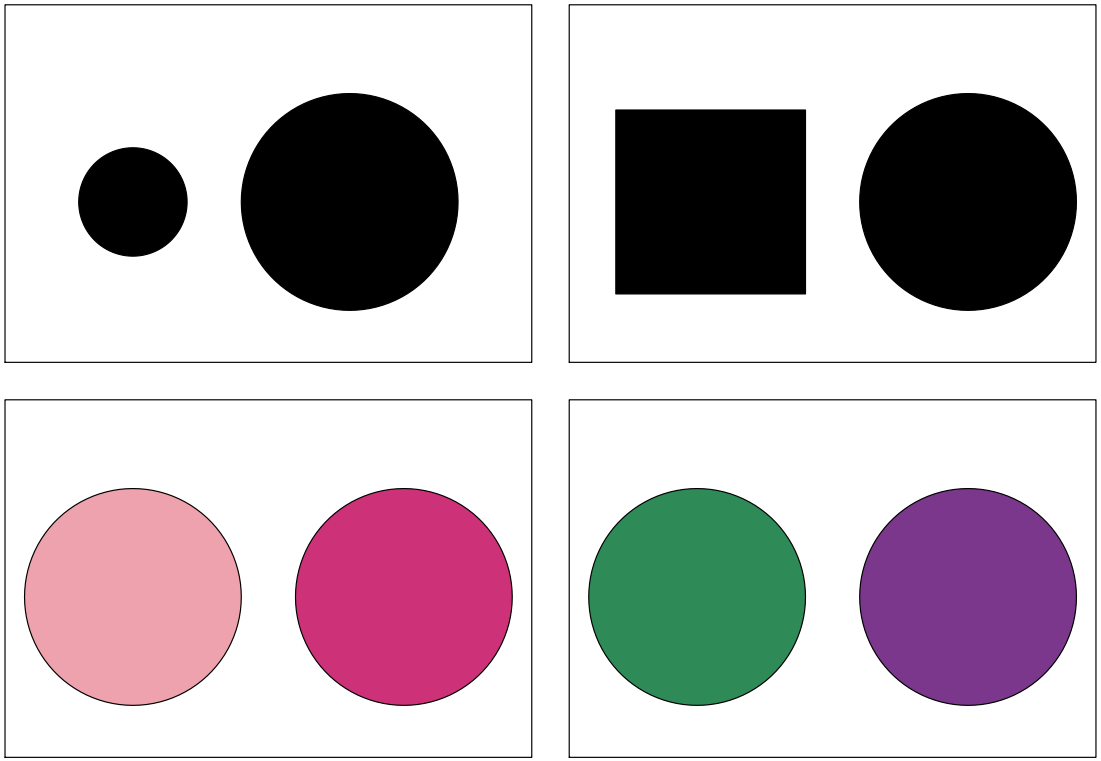
Some Visualisation Tips

Match perceptual and data topology

Some Visualisation Tips

Match perceptual and data topology

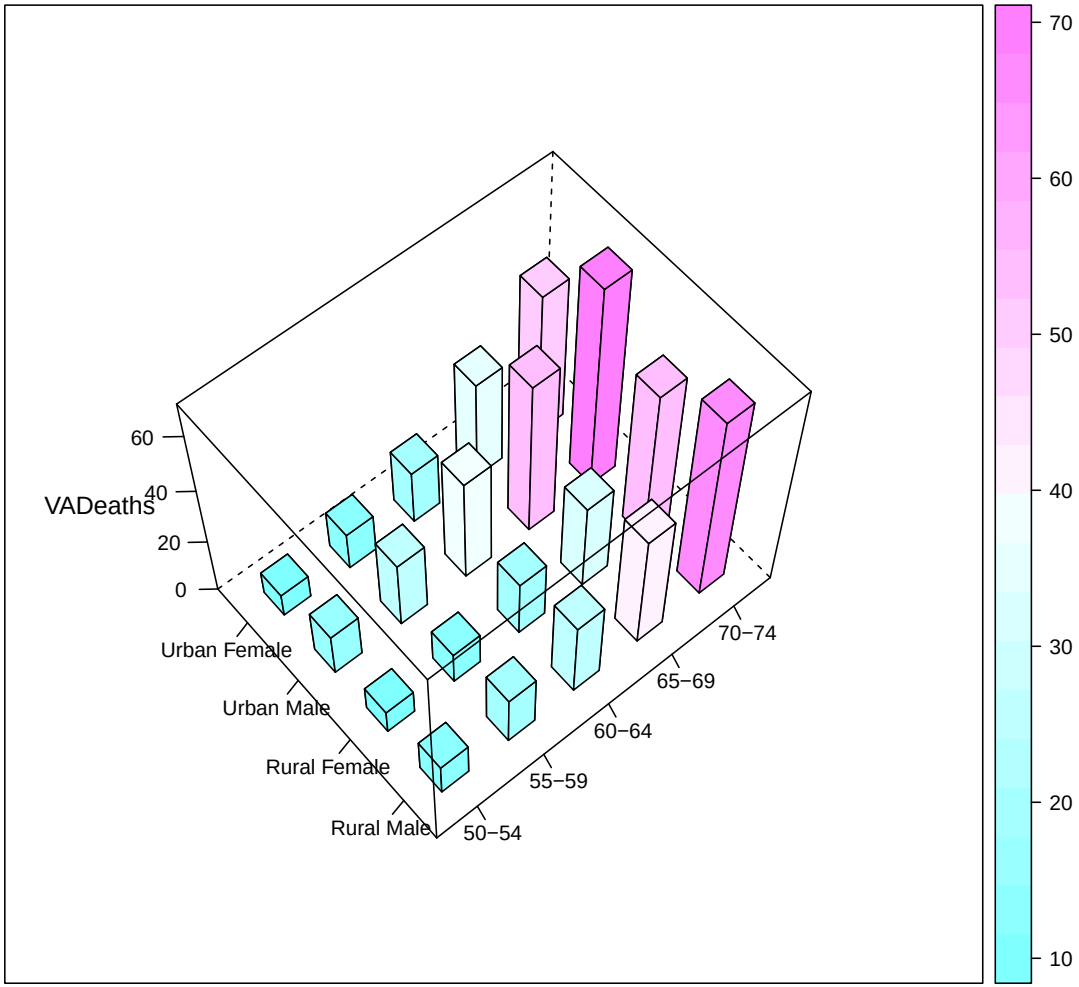
which represents the larger value?



Some Visualisation Tips

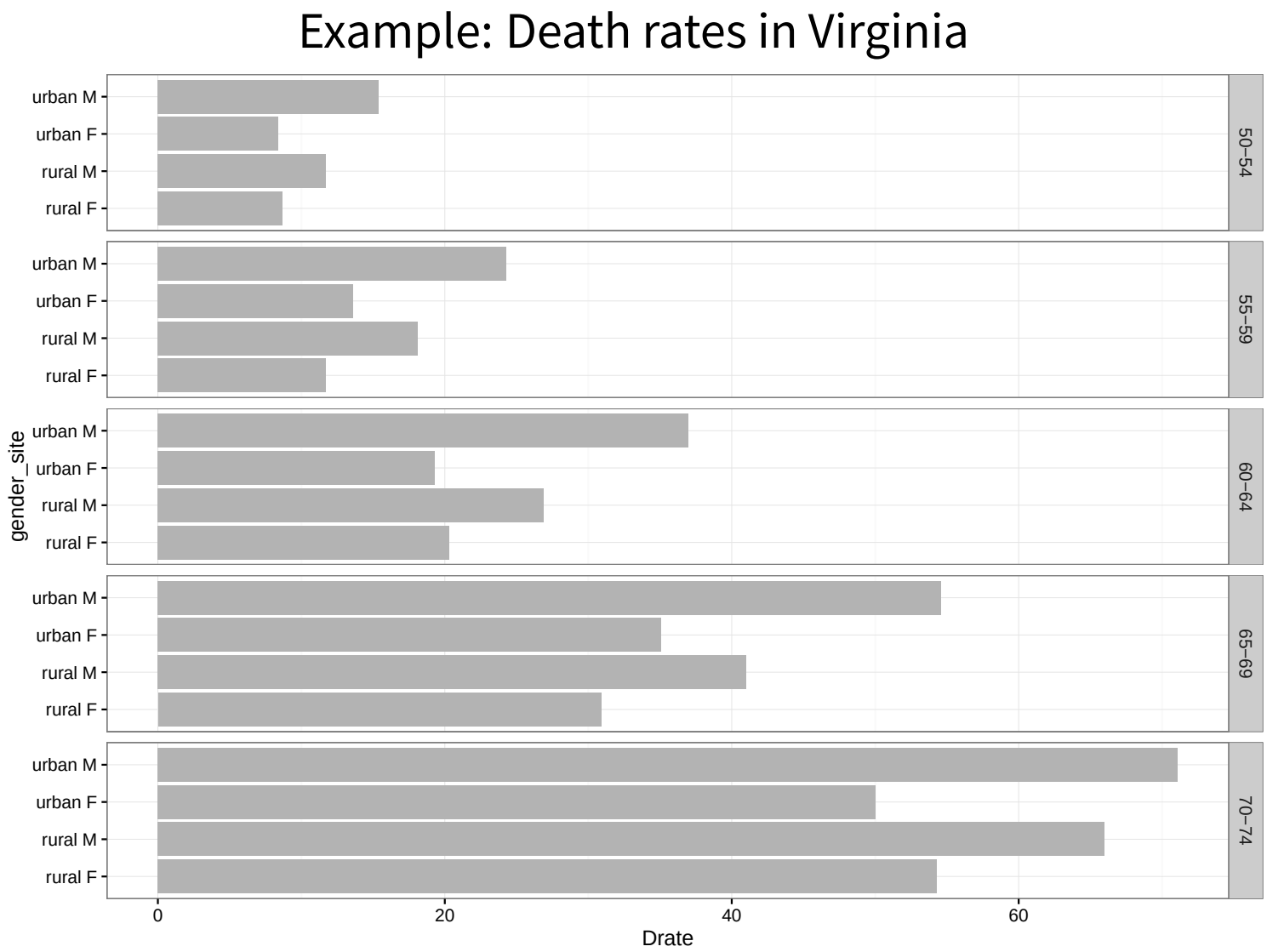
Avoid junk charts

Example: Death rates in Virginia



Some Visualisation Tips

Avoid junk charts



Some Visualisation Tips

- What you see on the screen is not what will appear on paper or on the projector
- Make important comparisons easy

Table of Contents

Base Graphics

- Graphs created using a series of *high level* and *low level* functions.
  - High level functions create new plots
  - Low level functions add information to an existing plot
- Customise graphs (line style, symbols, colours, ...) by specifying graphical parameters
  - using the `par()` function
  - Include graphic options as additional arguments to plotting functions

Base Graphics

High level functions

`plot()`	scatterplot, lineplot
`hist()`	Histogram
`boxplot()`	Boxplot
`qqplot(), qqline(), qqnorm()`	Quantile plots
`pairs()`	Scatter plot matrix
`symbols()`	Draw symbols on a plot
`dotchart(), barplot(), pie()`	Dot chart, barplot, pie chart
`curve()`	Draw a curve from a given function
`contour(), filled.contour()`	Contour plots

Base Graphics

Low level functions

`points()`	Add points to a plot
`lines()`	Add lines to a figure
`text()`	Add test in the plot region
`mtext()`	Insert text in the figure or outer margin
`title()`	Add figure title
`legend()`	Add a legend
`axis()`	Customise axis
`abline()`	Add vertical or horizontal line
`segments()`	Add line segment
...	

Base Graphics

Main options

See `?par` for the full list

`type`	1 character string denoting the plot type, e.g., `'n'` for none, `'l'` for line, `'s'` for step function, ...
`xlim, ylim`	x- and y-limit in the form `c(0, 100)`
`main`	Main title for the plot
`xlab, ylab`	x- and y-axis labels
`col`	A character string or a number specifying the colours
`pch`	Number referencing a plotting symbol or a character string
`cex`	A number that gives the character expansion of the plot symbols
`cex.axis, cex.lab`	character expansion for axis marks, labels
`lty`	Line type: 1 for solid, 2 for dashed, ...
`lwd`	Line thickness

Base Graphics

Multiple graphs

- Create a $n \times m$ grid of figures using `par()` with arguments `mfrow` or `mfcol`
  - `mfrow=c(n, m)` adds figures by row
  - `mfcol=c(n, m)` adds figures by column
- Create more complex arrangements using the `layout()` function
  - `mat`: a matrix object specifying the location of the next N figures on the output device
  - `widths` and `heights`: vectors that specify the relative widths and heights of the columns and rows, respectively
- `split.screen()` is used to create multiple plots and allows you to switch control between plot

Base Graphics

Math expression

- R is able to add LaTeX like expressions to R graphics
- Use `expression()` to add math expressions to a figure
- The function `bquote()` is used to add expressions and values. Terms inside `.()` are evaluated. The remaining terms are evaluated as math expressions
- See `?plotmath` for more details and examples

```
plot(1:10, type = "n", xlab = "", ylab = "",
     xaxt = "n", yaxt = "n")
text(5, 8, expression(alpha[1](t) == f^x), cex = 3)

theta <- 2
text(6, 4, bquote(hat(theta) == .(theta)), cex = 3)
```

Base Graphics

Math expression

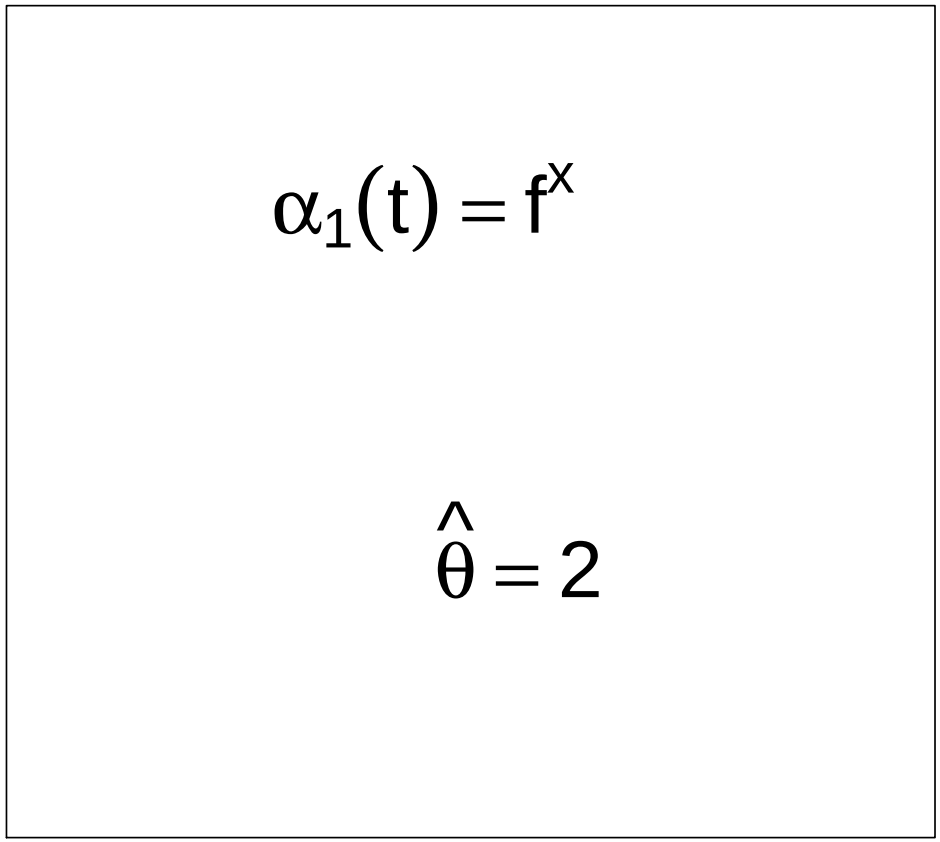


Table of Contents

The **ggplot2** Package

The **ggplot2** package is based on the *Grammar of Graphics* (Wilkinson)
The components of **ggplot2**'s grammar of graphics are

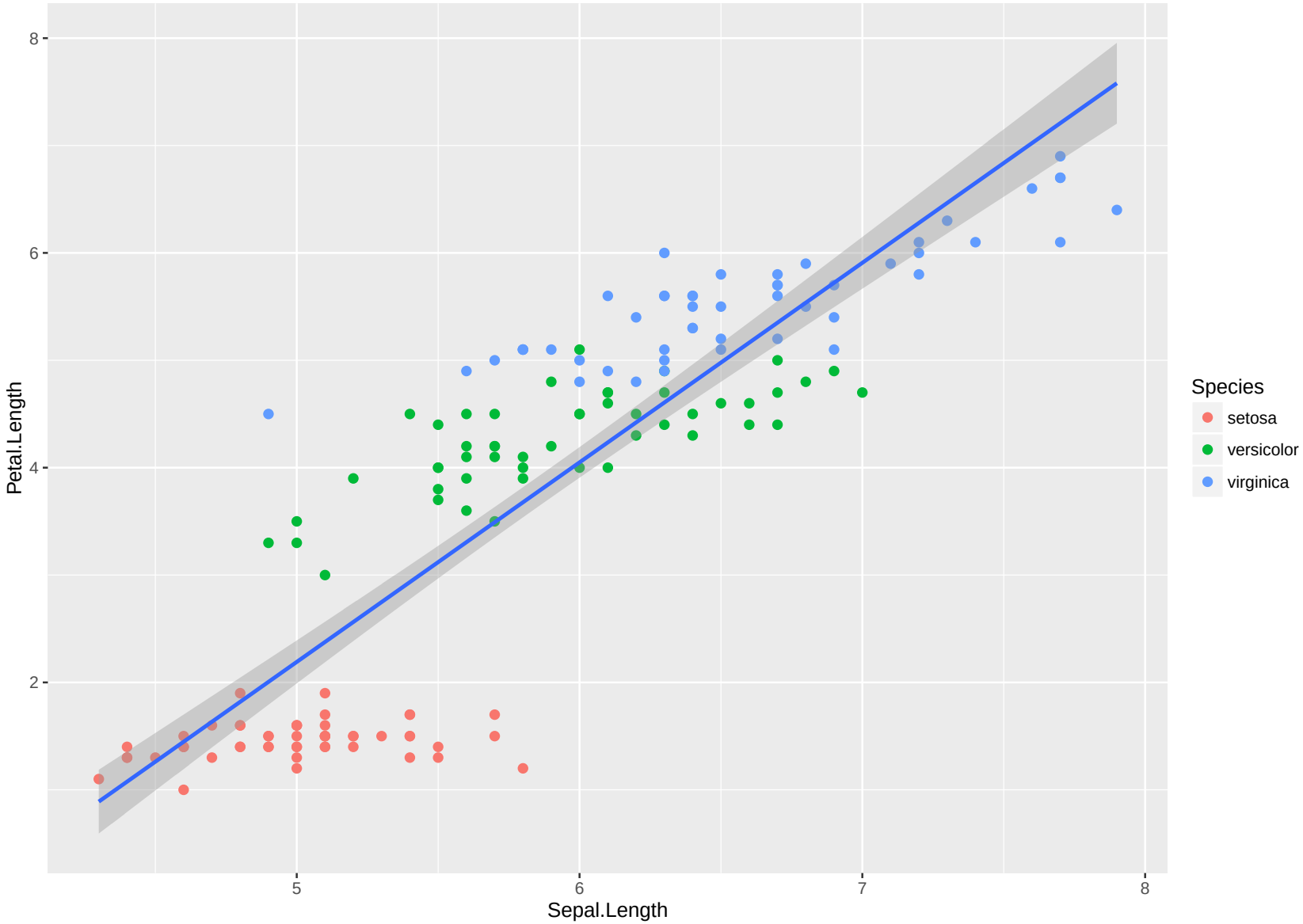
1. One or more layers
    1. Data
    2. *Aes*thetics mappings
    3. A statistical transformation
    4. A geometric object
    5. Position adjustment
2. A scale for each aesthetic
3. A coordinate system
4. A facet specification
5. Guides (legends)

Layers

- `ggplot` produces an object that is rendered into a plot
- This object consists of some layers
- Each layer may share some arguments with `ggplot` or gets its own input

```
p <- ggplot(data = iris, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point(aes(colour = Species), size = 2) +
  geom_smooth(method = "lm")
p
```

Layers


Petal.Length
8
6
4
2
5
6
7
8
Sepal.Length
Species
setosa
versicolor
virginica

Components of a layer

Data and aesthetic mappings

- The **data** are what we want to visualize
- The *aes*thetic mappings map the columns of the data.frame to x/y axis, colours, size, ...

```
p <- ggplot(data = iris, aes(x = Sepal.Length, y = Petal.Length))
```

Components of a layer

A statistical transformation

- A summary of the raw data input
- Somehow implicit in a lot of **ggplot** functions
- Example: identity, binning, smoothing, boxplot, ...

```
p + geom_point(aes(colour = Species)) +
  geom_smooth(method = "lm")
```

Components of a layer

A geometric object

- The type of plot to be created, e.g., points, lines, polygon, ...
- See functions `geom_point`, `geom_line`, `geom_hist`, ...

```
p + geom_point(aes(colour = Species)) +
  geom_smooth(method = "lm")
```

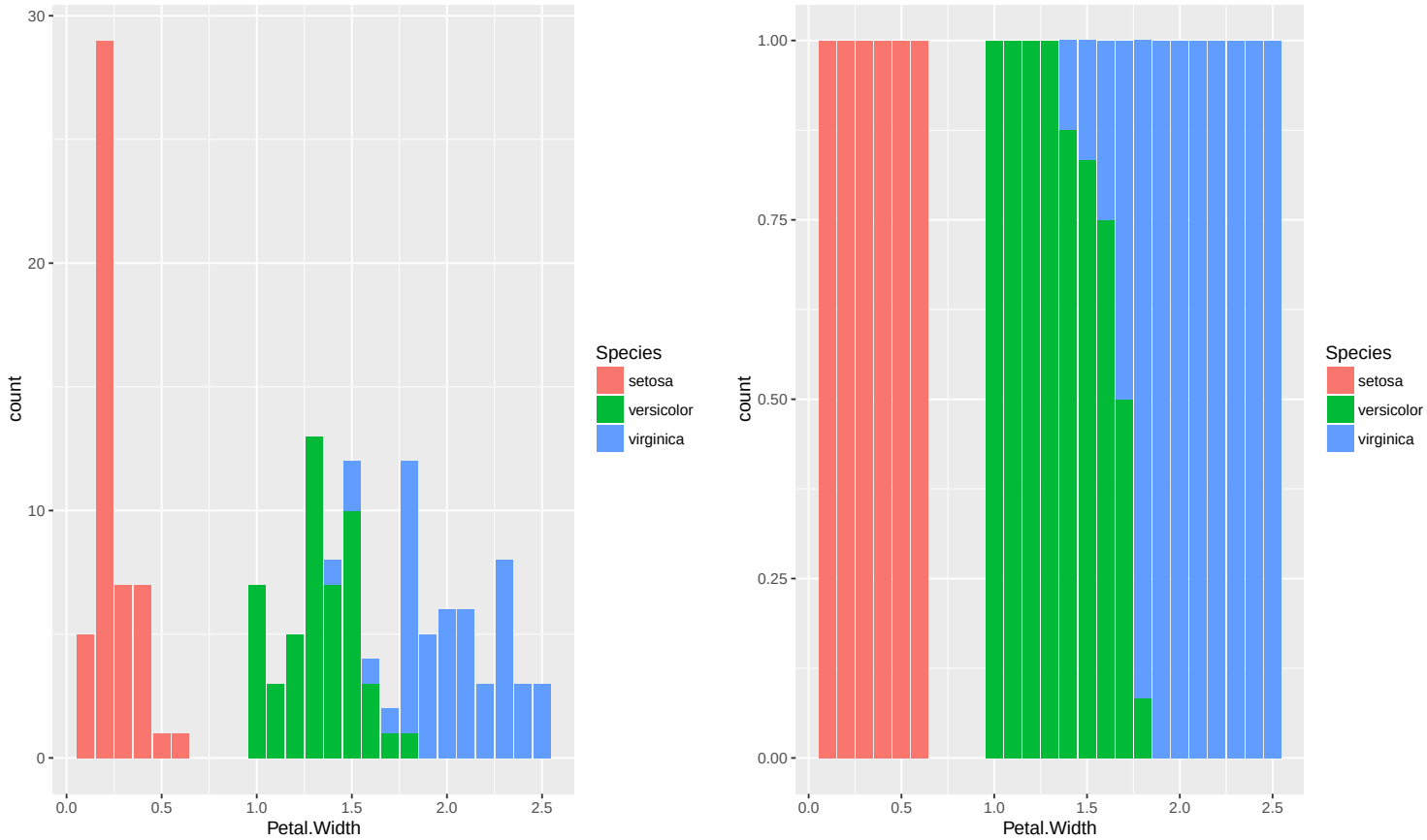
Components of a layer

Position

- Determines how different parts of a layer are positioned relative to each other, e.g, identity, jitter, dodge

```
g1 <- ggplot(iris, aes(x = Petal.Width, fill = Species)) + geom_bar()
g2 <- ggplot(iris, aes(x = Petal.Width, fill = Species)) +
  geom_bar(position = "fill")
grid.arrange(g1, g2, ncol = 2)
```

Components of a layer



Other elements

Scaling How each input value maps to the specified aesthetic, e.g., logarithmic, continuous, ordinal, limits, labels

Coordinates How positions of things are mapped to positions on the screen

Facets Allows arranging different graphs in a grid/panel

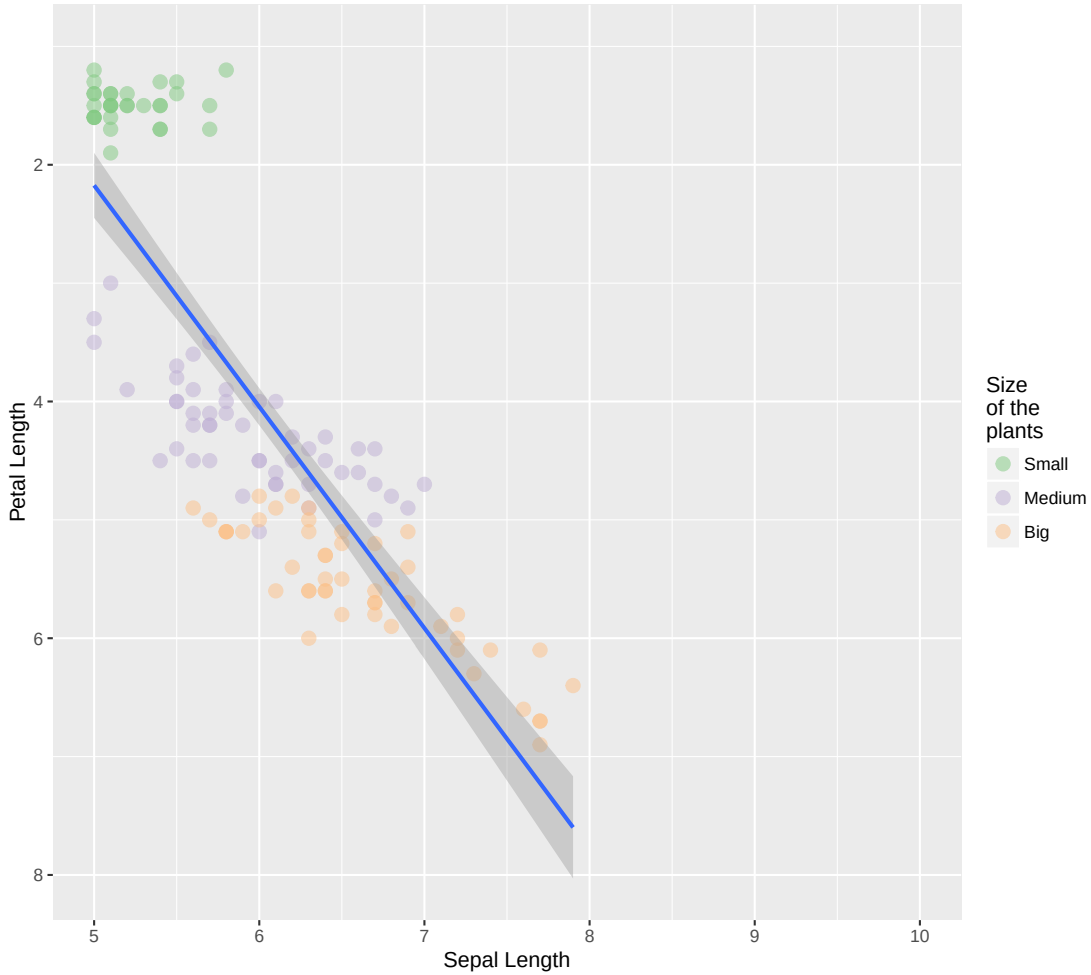
Example

```
p <- ggplot(data = iris, aes(x = Sepal.Length, y = Petal.Length)) +
  geom_point(aes(colour = Species), size = 3, alpha = .5) +
  geom_smooth(method = "lm") +
  scale_x_continuous("Sepal Length", limits = c(5, 10),
                     breaks = 5:10) +
  scale_y_reverse("Petal Length") +
  scale_colour_brewer("Size\nof the\nplants", type = "qual",
                      labels = c("Small", "Medium", "Big"))
p
```

See all the options and geoms at
`http://docs.ggplot2.org/current/index.html`

Example

```
## Warning:  Removed 22 rows containing non-finite values (stat_smooth).
## Warning:  Removed 22 rows containing missing values (geom_point).
```



Example

```
p + facet_grid(. ~ Species)
```

```
## Warning:  Removed 22 rows containing non-finite values (stat_smooth).
## Warning:  Removed 22 rows containing missing values (geom_point).
```