University of BRISTOL

Data visualisation techniques using R

Osama Mahmoud

Website: http://osmahmoud.com
E-mail: o.mahmoud@bristol.ac.uk

30 April 2019

Goals

In this session, we aim to cover:

- What the R base graphics and their types are.
- How to produce base plots in R.
- How to save your generated graphics.
- How to use `ggplot2` in R to produce advanced graphics.
- What the building-blocks of `ggplot2` graphs are.

Contents

Types of R graphics

- Base graphics.
- Grid graphics.
- Lattice graphics.
- ggplot2 graphics.

Base graphics

Installing the R package: BristolVis

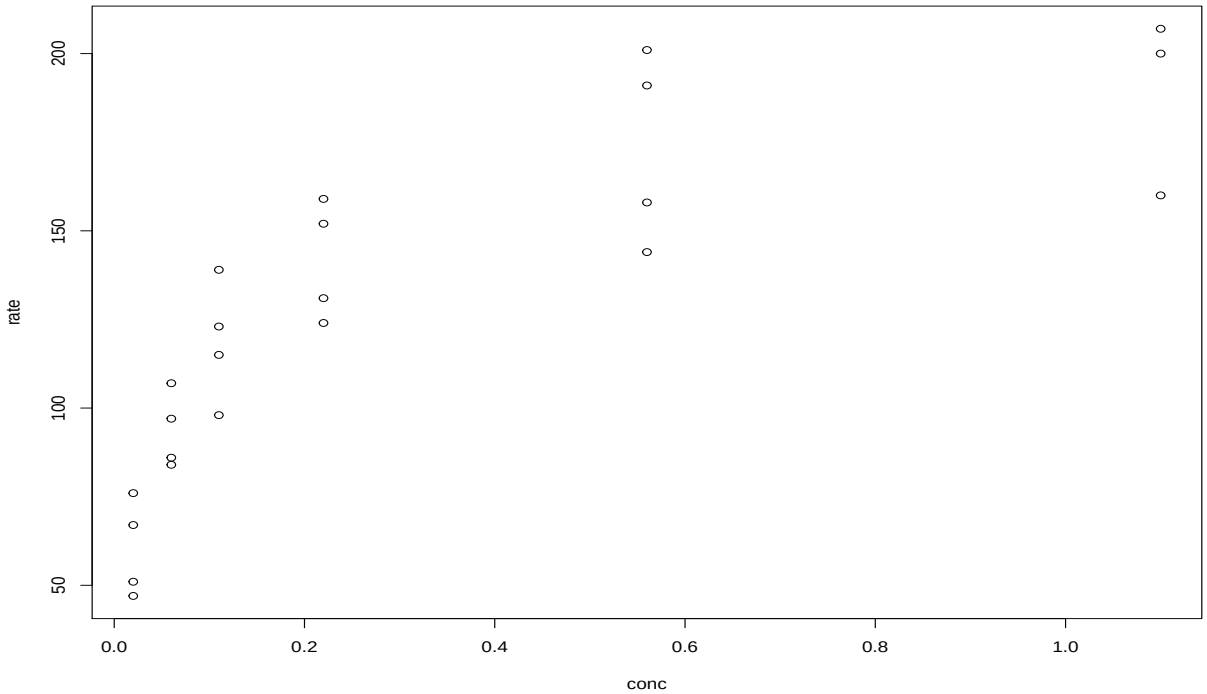
```
> install.packages("drat")

> drat::addRepo("statcourses")

> install.packages("BristolVis", type="source")
```

Overview of base graphics

- Graphics can be easily created in R and subsequently included in Word, Power Point, LATEX, etc.
- Let's plot a simple graphic:

```
> plot(rate ~ conc, data = Puromycin)
```

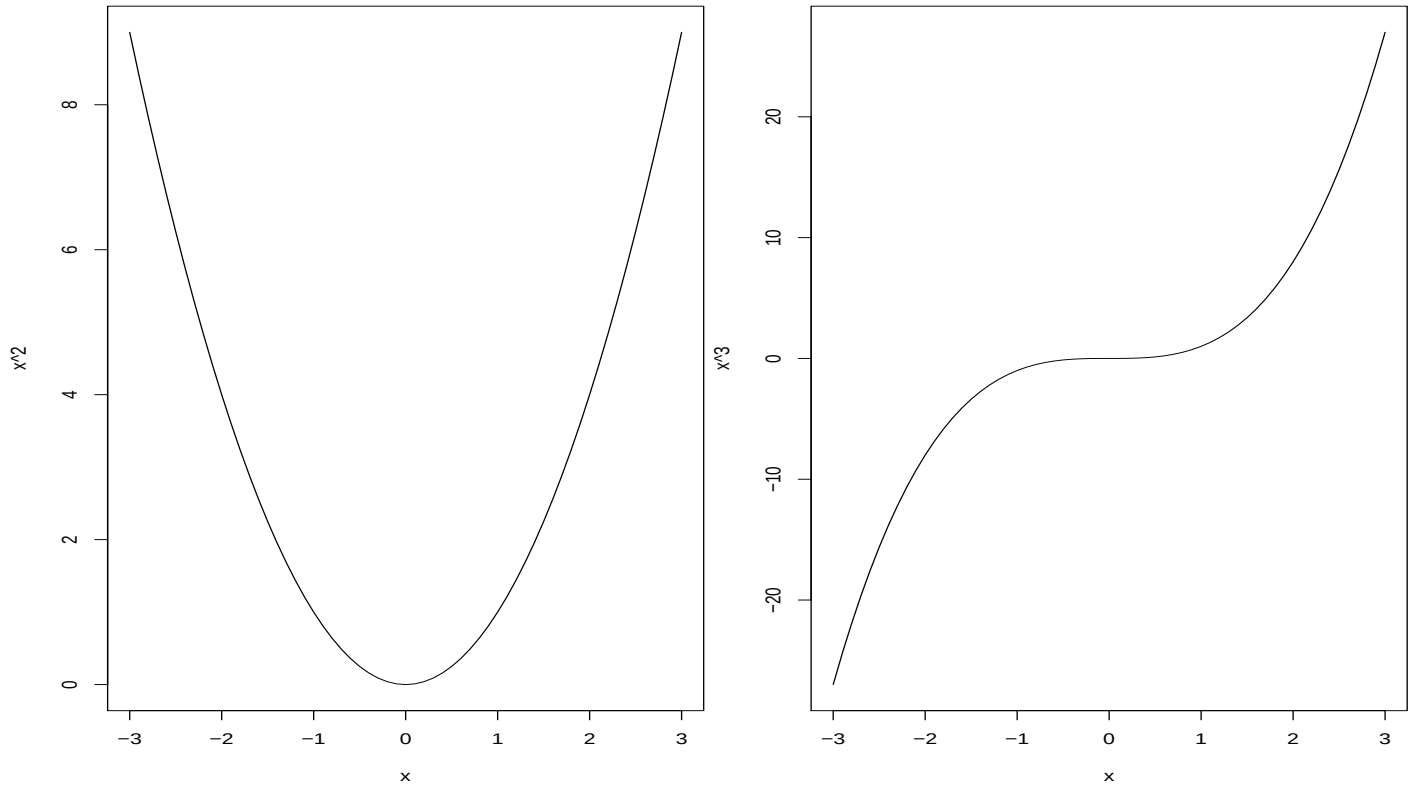
This implicitly opens a new graphics window.



Overview of base graphics

- Change settings of active graphic window

```
> par(mfrow = c(1,2), mar = c(4,4,0,0) + 0.1)
> x <- seq(-3,3, 0.1)
> plot(x, x^2, type = 'l')
> plot(x, x^3, type = 'l')
```



Overview of base graphics

- Close active graphics window:

```
> dev.off()
```

- Close all graphics windows:

```
> graphics.off()
```

Histograms

Histograms

Simple histogram (next slide left)

```
> data(birthweight, package = "BristolVis")
> hist(birthweight$weightgain)
```

- with a fixed set of "bars": (`breaks` give number of cut-points)

```
> hist(birthweight$weightgain, breaks = 3)
```

- with a defined scale for the y-axis:

```
> hist(birthweight$weightgain, breaks = 3, ylim = c(0,
10))
```

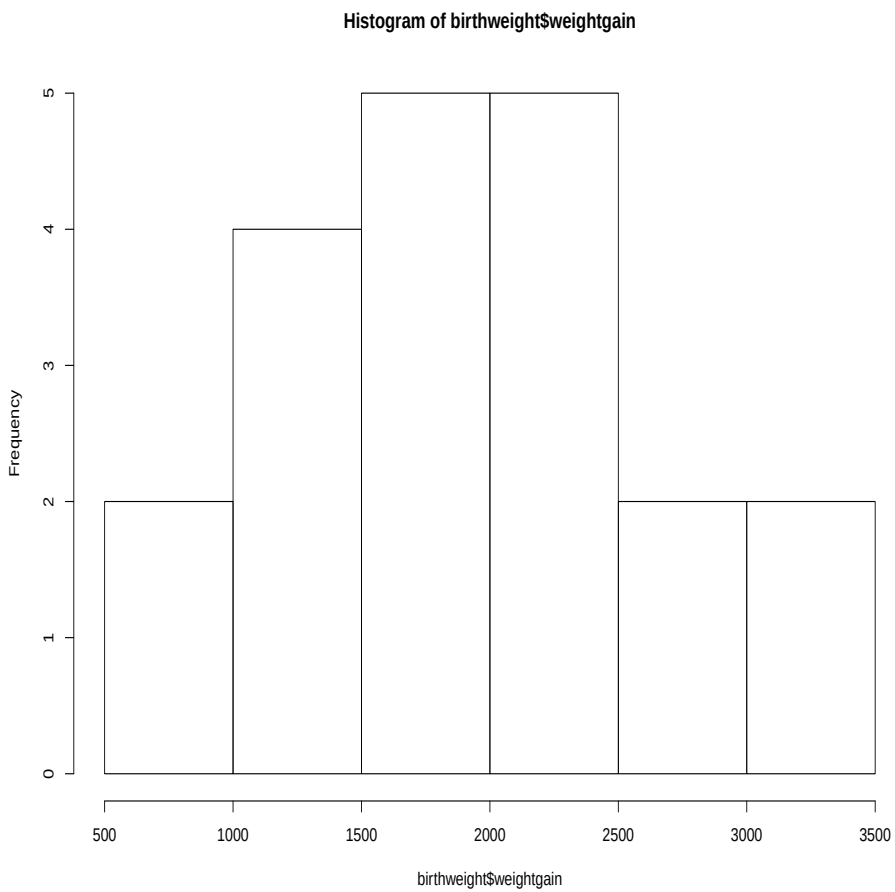
- with colored bars:

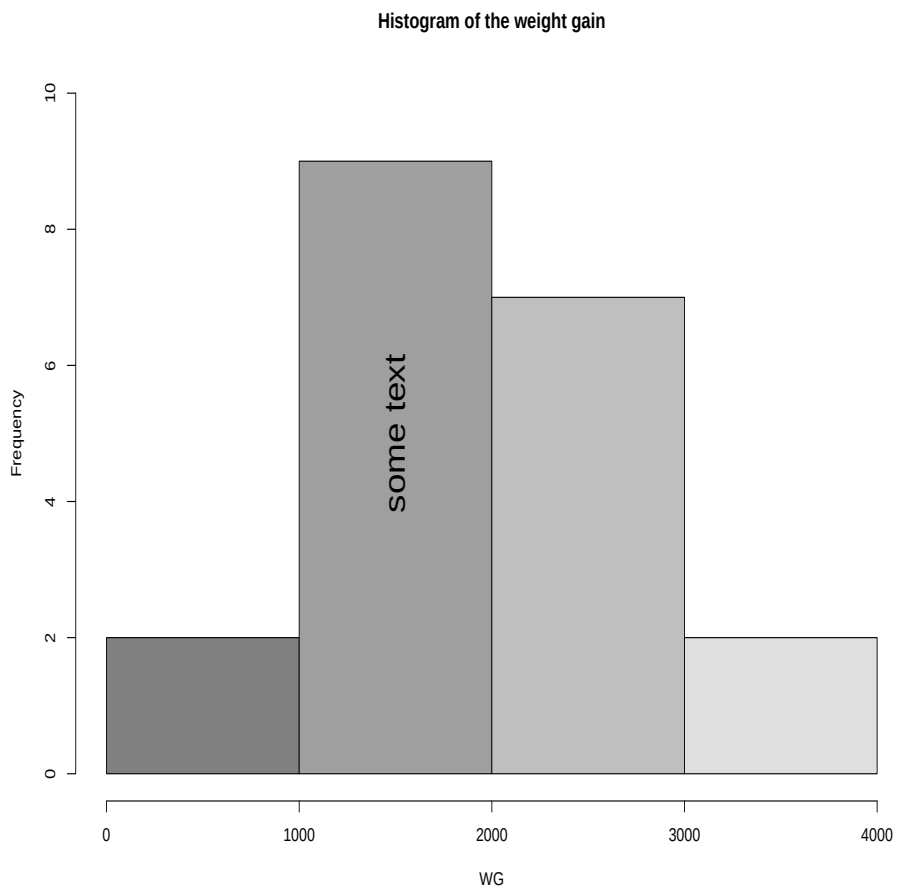
```
> hist(birthweight$weightgain, breaks = 3, ylim = c(0,
10), col = gray(4:8 / 8))
```

with arbitrary text, vertical and double text size (next slide right)

```
> text(1500, 5, "some text", srt = 90, cex = 2)
```

Histograms


Histogram of birthweight$weightgain
Frequency
0
1
2
3
4
5
500
1000
1500
2000
2500
3000
3500
birthweight$weightgain


Histogram of the weight gain
Frequency
0
2
4
6
8
10
some text
0
1000
2000
3000
4000
WG

Bar plots

Bar plots

Simple bar plot (next slide left)

```
> data(med, package = "BristolVis")
> treatment = table(med$treatment)
> barplot(treatment, ylim = c(0, 500))
```

- stacked bar plot:

```
> health_treat = table(med$health, med$treatment)
> barplot(health_treat, ylim = c(0, 500))
```

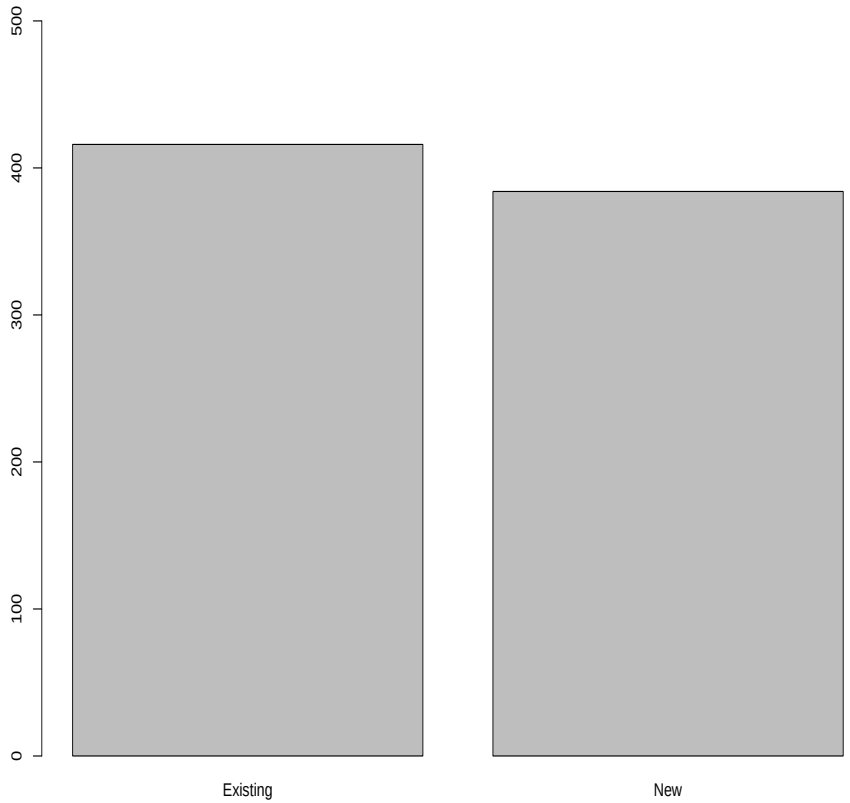
- with juxtaposed bars:

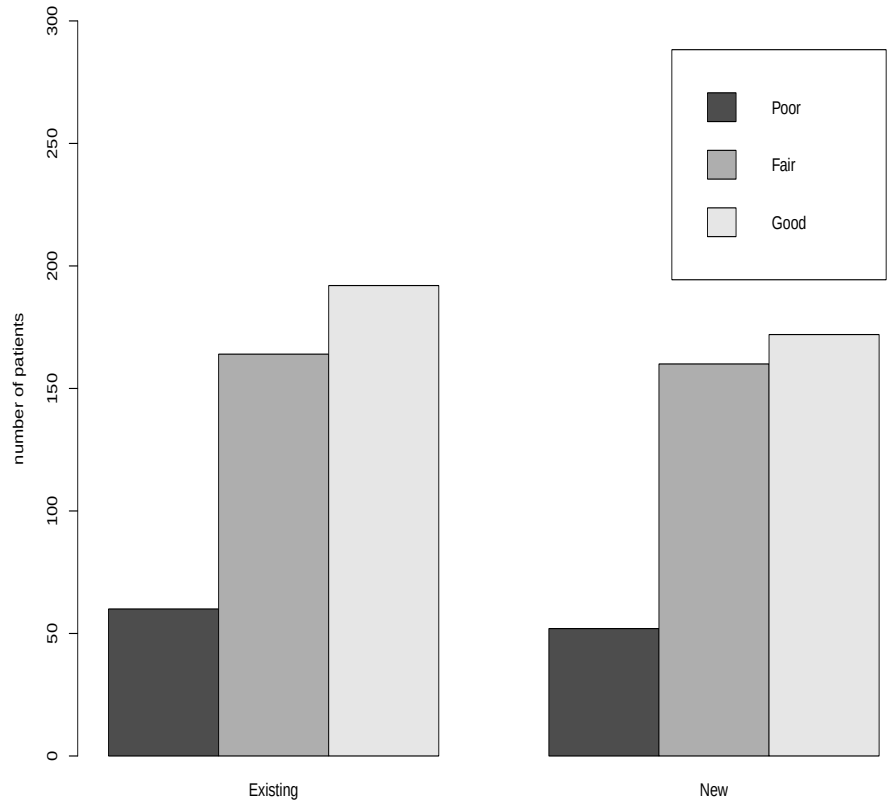
```
barplot(health_treat, ylim = c(0, 300), beside = TRUE)
```

with legend (next slide right)

```
> barplot(health_treat, ylim = c(0, 300), beside = TRUE,
legend.text = rownames(health_treat))
```

Bar plots


0
100
200
300
400
500
Existing
New


number of patients
0
50
100
150
200
250
300
Poor
Fair
Good
Existing
New

Box plots

Box plots

Simple box plot (next slide left)

```
> boxplot(birthweight$weight)
```

- grouped by sex:

```
> boxplot(weight ~ sex, data = birthweight)
```

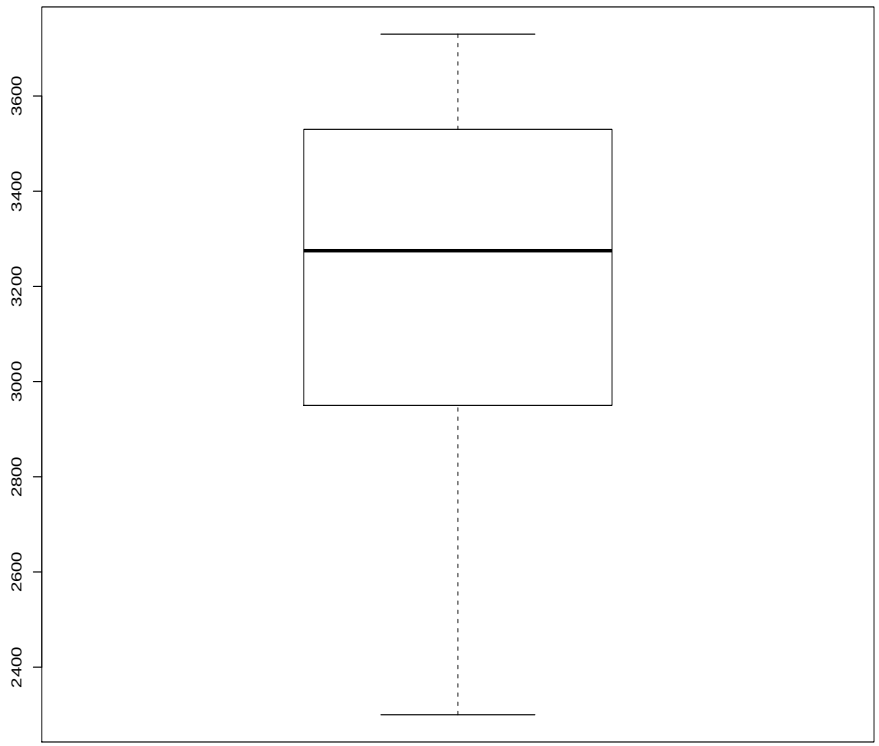
with new labels for sex, main title and a defined y-scale (next slide right)

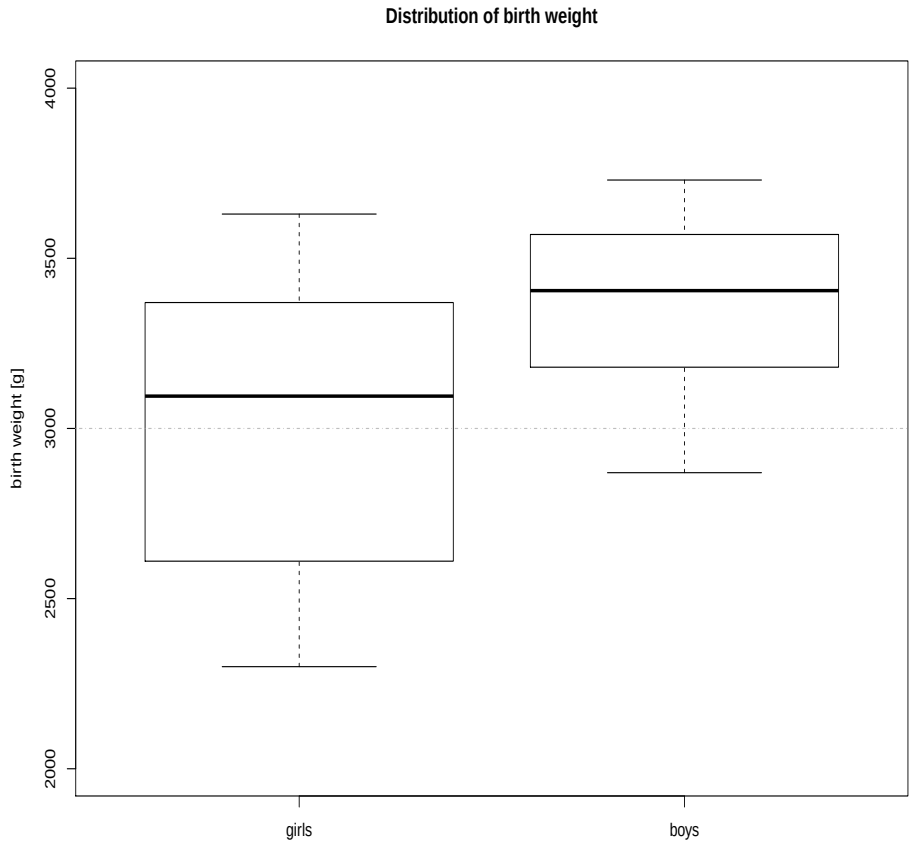
```
> boxplot(weight ~ sex, data = birthweight, names =
c("girls", "boys"), ylab = "birth weight [g]", ylim =
c(2000, 4000), main = "Distribution of birth weight")
```

- with additional line:

```
> abline(h = 3000, col = "gray", lty = 4)
```

Box plots


2400
2600
2800
3000
3200
3400
3600


Distribution of birth weight
birth weight [g]
2000
2500
3000
3500
4000
girls
boys

Scatter plots

Scatter plots

Simple scatter plot (next slide left)

```
> plot(weightgain ~ weight, data = birthweight)
```

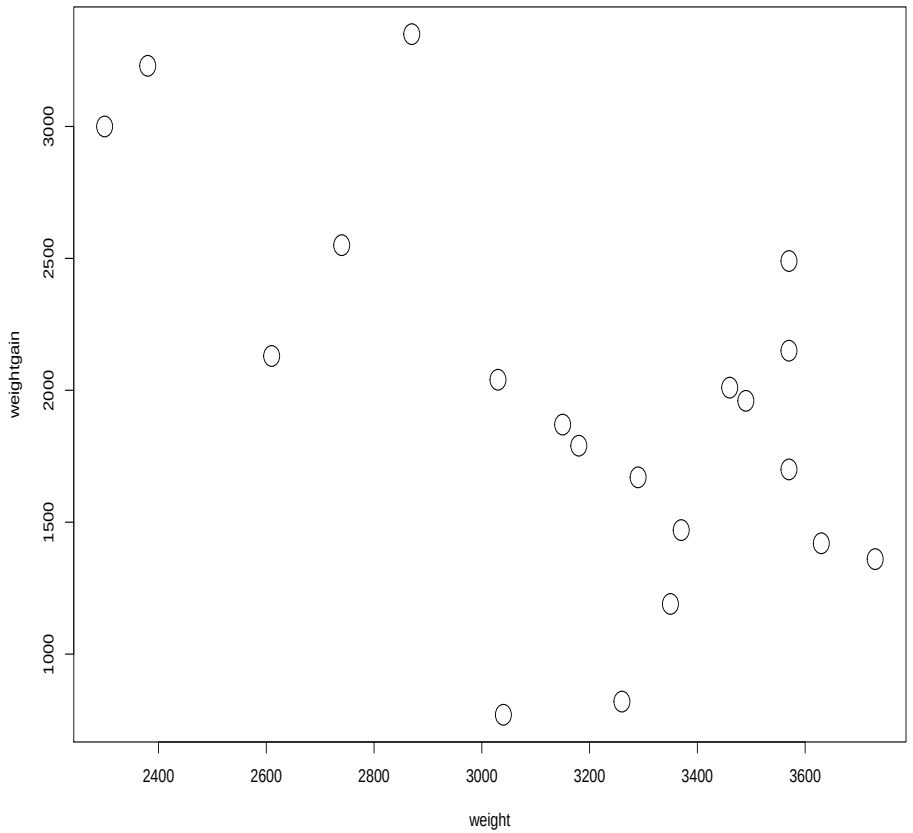
- for a subset:

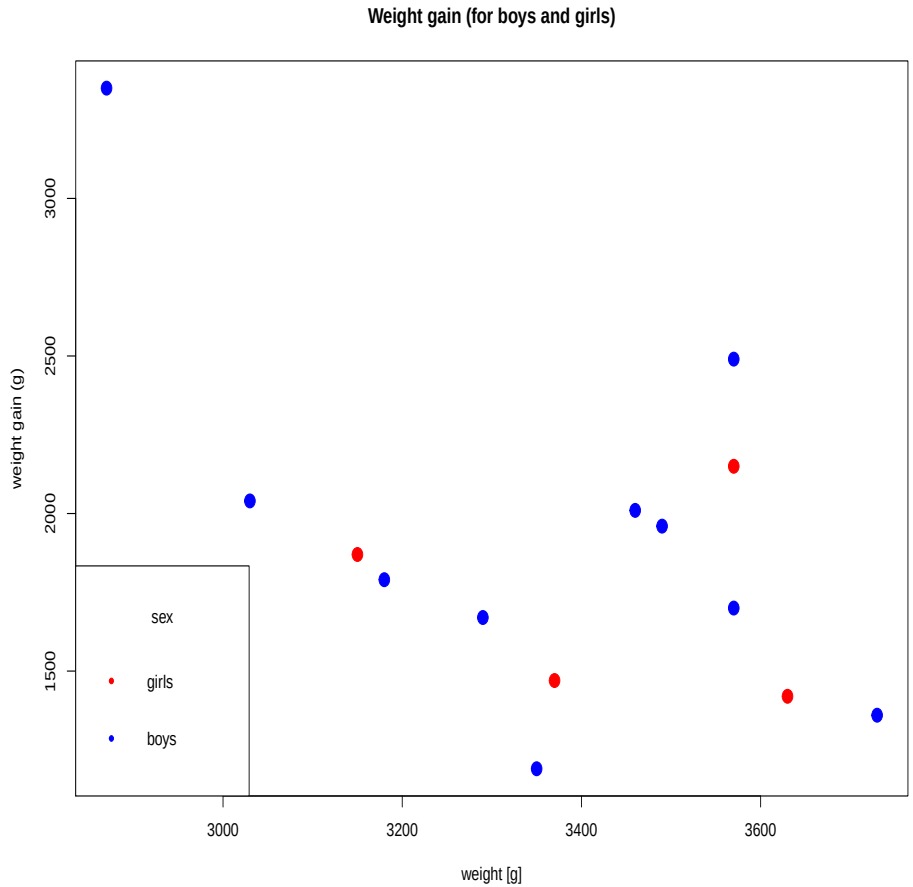
```
> plot(weightgain ~ weight, data = birthweight, subset =
(sex == "F"))
```

with colored solid points by sex and add a legend (next slide right)

```
> plot(weightgain ~ weight, data = birthweight, subset =
(sex == "M"), col = "blue", pch = 20)
> points(weightgain ~ weight, data = birthweight, subset =
(sex == "F"), col = "red", pch = 20) # add girls data
> legend("bottomleft", title = "sex", legend = c("girls",
"boys"), col = c("red","blue"), pch = 20) # add legend
```

Scatter plots





Examples of the graphic library

An overview of available demos for the built-in R library (graphics):

```
> demo(graphics)
```

Saving plots

Saving plots

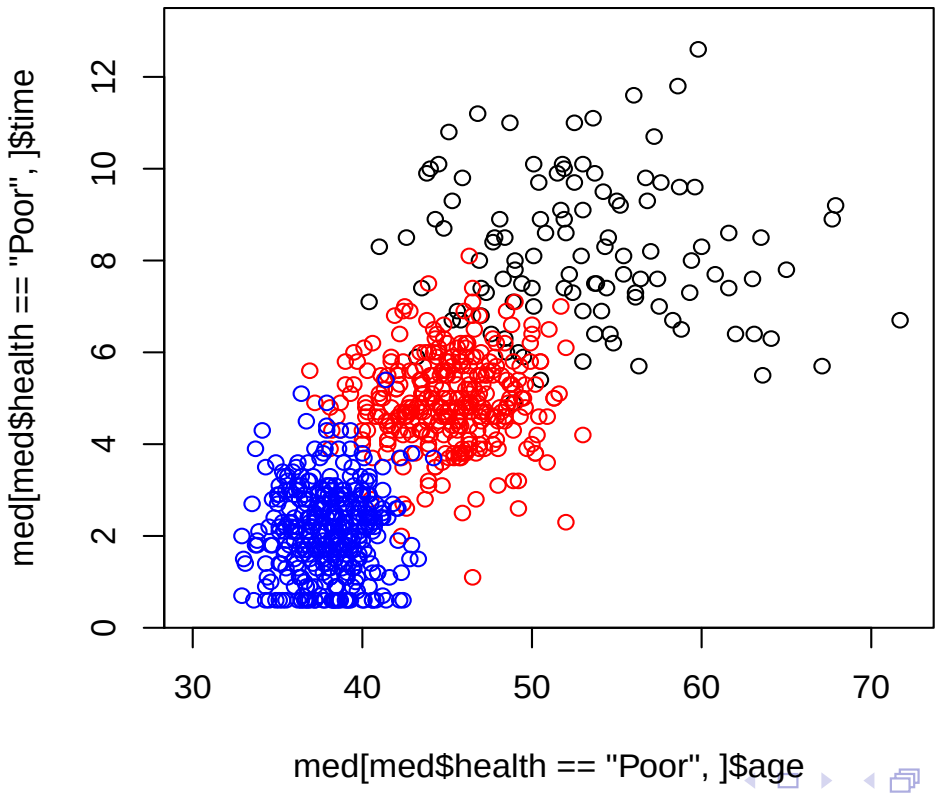
- Graphics can be saved directly from within R using different devices, e.g. postscript(), pdf(), bmp(), (see `?Devices`)
- For publications vector graphics such as PDFs or postscript files are preferable to pixel graphics such as jpg, bmp, etc.

```
> pdf("Fig1.pdf", width = 10, height = 7)
> boxplot(weight ~ sex, data = birthweight)
> dev.off()
```

Advanced Graphics Using ggplot2

Remember: Using base graphics

```
> plot(med[med$health=="Poor",]$age, med[med$health=="Poor",]$time,
xlim=c(30,72), ylim=c(0.5,13))
> points(med[med$health=="Fair",]$age, med[med$health=="Fair",]$time,
col=2)
> points(med[med$health=="Good",]$age, med[med$health=="Good",]$time,
col=4)
```



Remember: Using base graphics

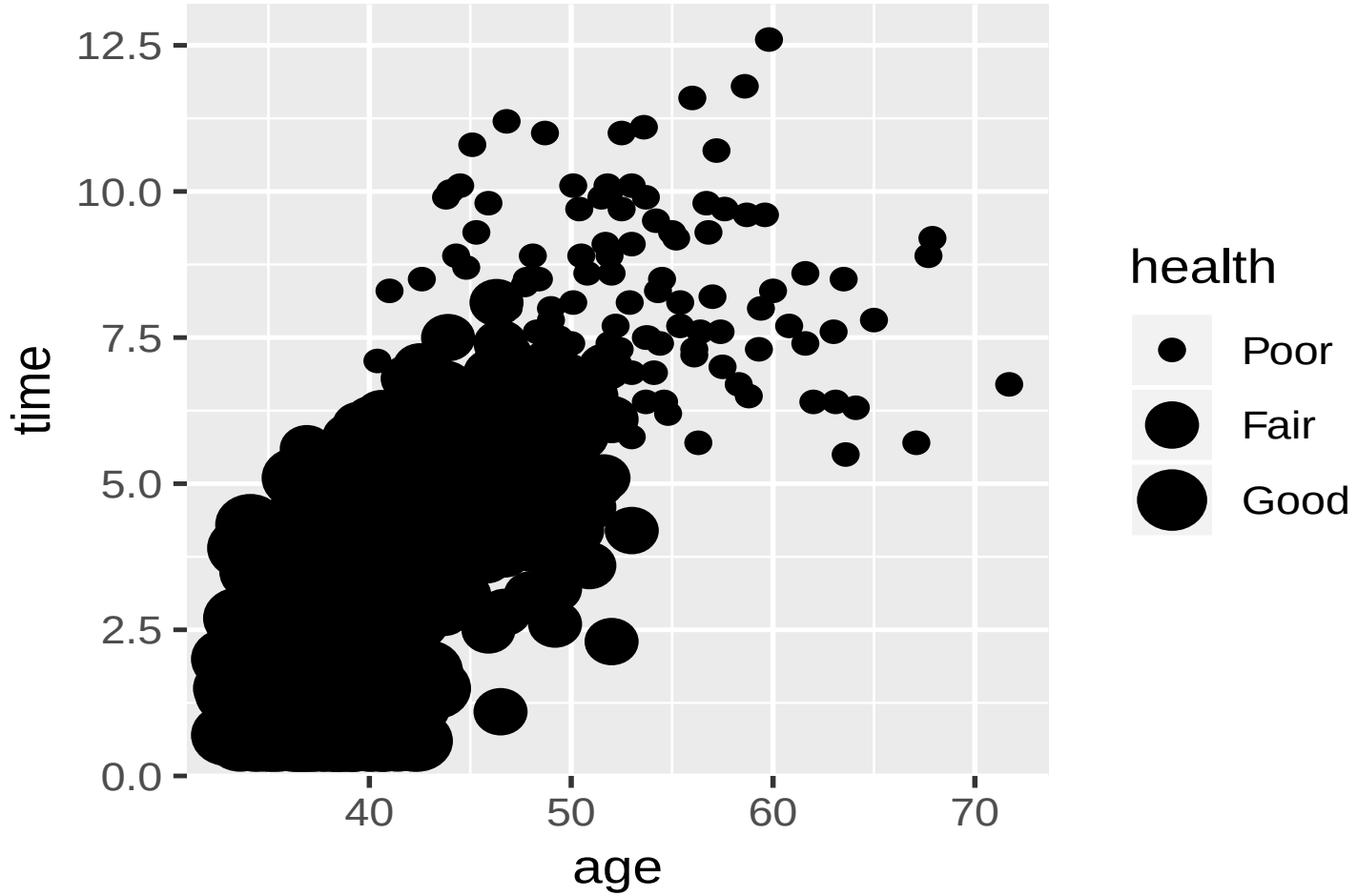
- We had to manually set the scales using the `xlim` and `ylim` parameters.
- We had not created a legend. We would need to use the `legend` function to create one.
- The default axis labels were terrible!

Equivalent graphics using ggplot2

```
> library(ggplot2)
> g = ggplot(data=med, aes(x=age, y=time))
> g + geom_point(aes(colour=health))
```

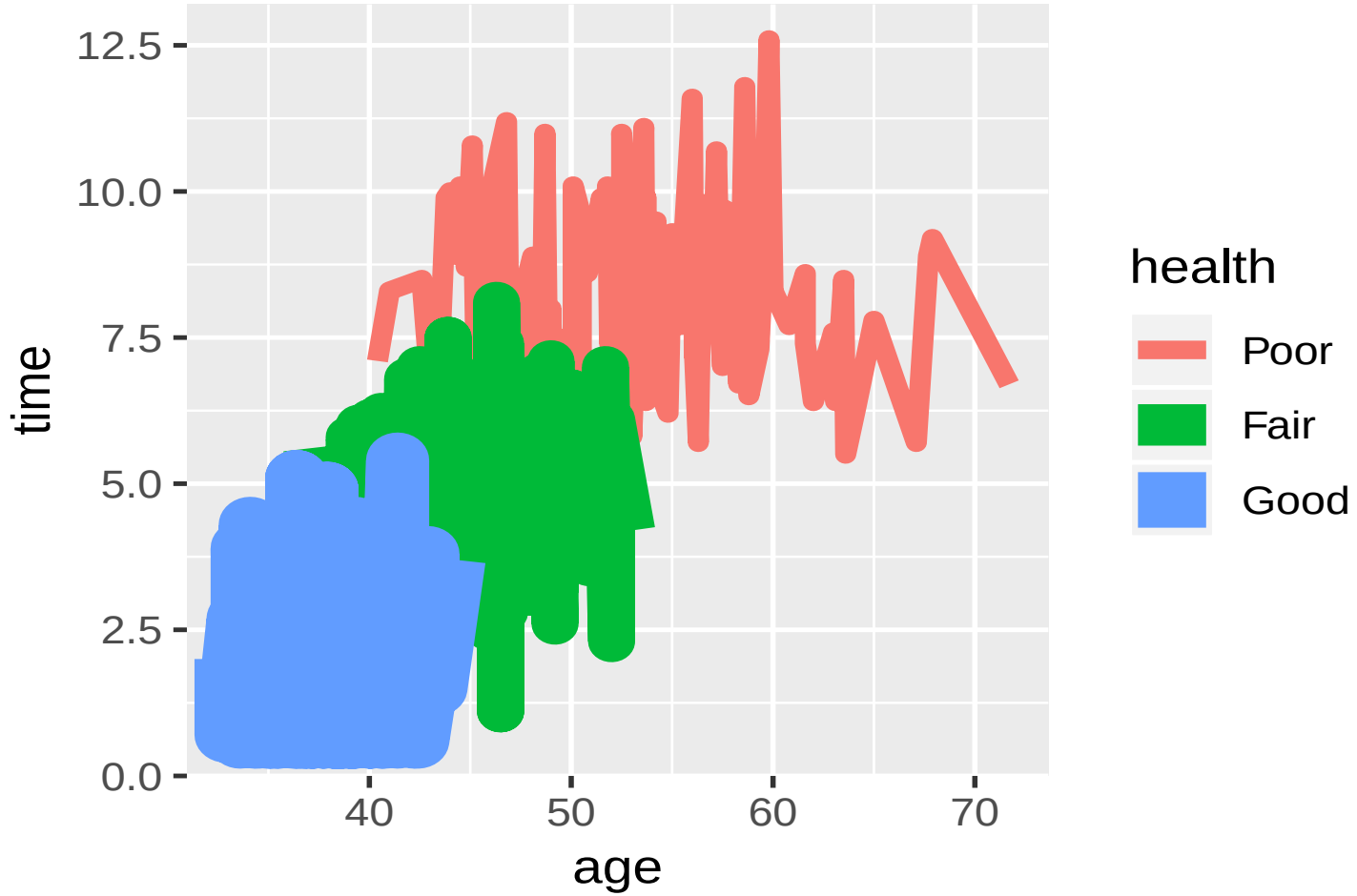
Factor of point sizes

```
g + geom_point(aes(size=health))
```



Example of a line chart

```
g + geom_line(aes(colour=health, size = health)))
```



Geometric objects

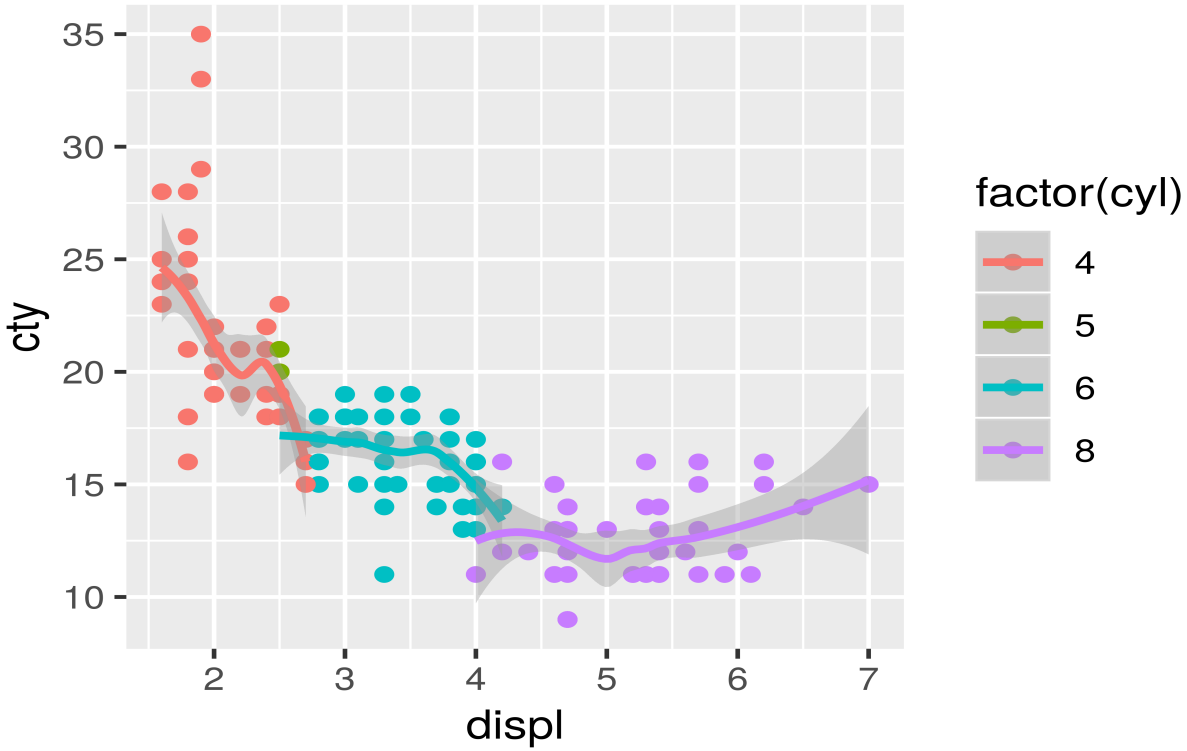
Points, bars and lines are examples of geom's. Some useful standard geoms and their equivalent base graphic counter part:

Plot Name	Geom	Base graphic
Bar chart	bar	`barplot`
Box-and-whisker	boxplot	`boxplot`
Histogram	histogram	`hist`
Line plot	line	`plot` and `lines`
Scatter plot	point	`plot` and `points`

More complicated functions

The idea of graphical layers, enables constructing more complicated functions, e.g.:

```
p = ggplot(mpg, aes(x = displ, y = cty)) +
geom_point(aes(colour=factor(cyl))) +
stat_smooth(aes(colour=factor(cyl)))
```



Understanding ggplot2 philosophy

Each `ggplot` command adds iteratively layers. A single layer may comprise of four elements:

- an aesthetic and data mapping;
- a geometric object (`geom`);
- a statistical transformation (`stat`);
- a position adjustment, i.e. how should overlapped objects be handled.

Understanding ggplot2 philosophy

For example, the command:

```
g + geom_point(aes(colour=health))
```

actually calls (in the background) the command:

```
g + layer(data = med, #inherited
mapping = aes(color=health), #x and y are inherited.
stat = "identity",
geom = "point",
position = "identity",
params = list(na.rm=FALSE))
```

Plot building-blocks

Initial plot object

An initial ggplot object, can be setup using the `ggplot()` function which has two arguments:

- `data` (*takes a data frame*)
- an aesthetic `mapping` (*creates default aesthetic attributes*)

```
g = ggplot(data=mpg, mapping=aes(x=displ, y=cty,
colour=factor(cyl)))
```

Or equivelently,

```
g = ggplot(mpg, aes(displ, cty, colour=factor(cyl)))
```

doesn't actually produce anything to be displayed, it just sets the intial plot object. We need to add layers for that to happen.

The geom_ functions

- The `geom_` functions perform the actual rendering in a plot, e.g. a line geom will create a line plot and a point geom creates a scatter plot.
- Each geom has a list of aesthetics that it accepts such as `x` , `y` , `colour` and `size`.
- However, some geoms have unique elements. For example, the `geom_errorbar` requires arguments ymax and ymin.
- The full list of aesthetics can be displayed by:

```
> ggplot2:::.all_aesthetics
```

The geom_ functions

- This table gives names and descriptions of some commonly used geoms:

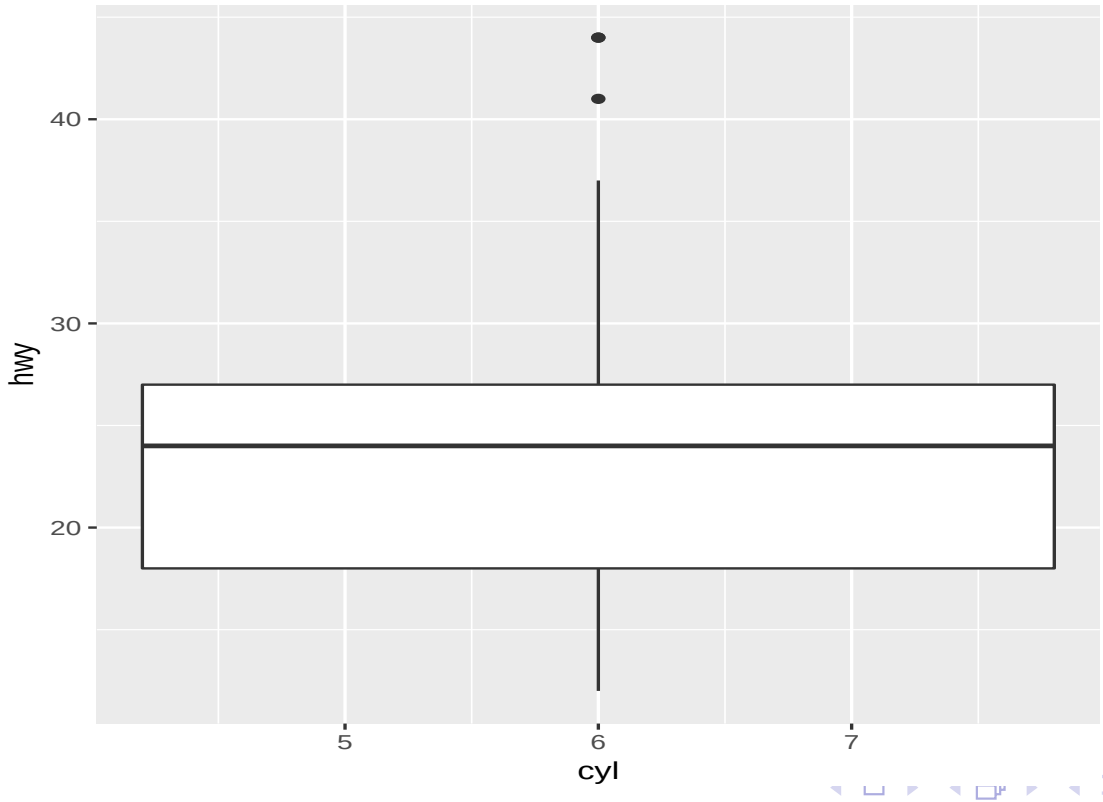
Name	Description
`abline`	Line, specified by slope and intercept
`boxplot`	Box and whiskers plot
`density`	Kernel density plot
`histogram`	Histograms
`jitter`	Individual points are jittered to avoid overlap
`step`	Connect observations by stairs

Combining geoms

I will show how to combine more that one `geom` function to produce a bit more complex plots. If we consider the `mpg` data set, a base ggplot object:

`g = ggplot(mpg, aes(x=factor(cyl), y=hwy))` will do nothing.

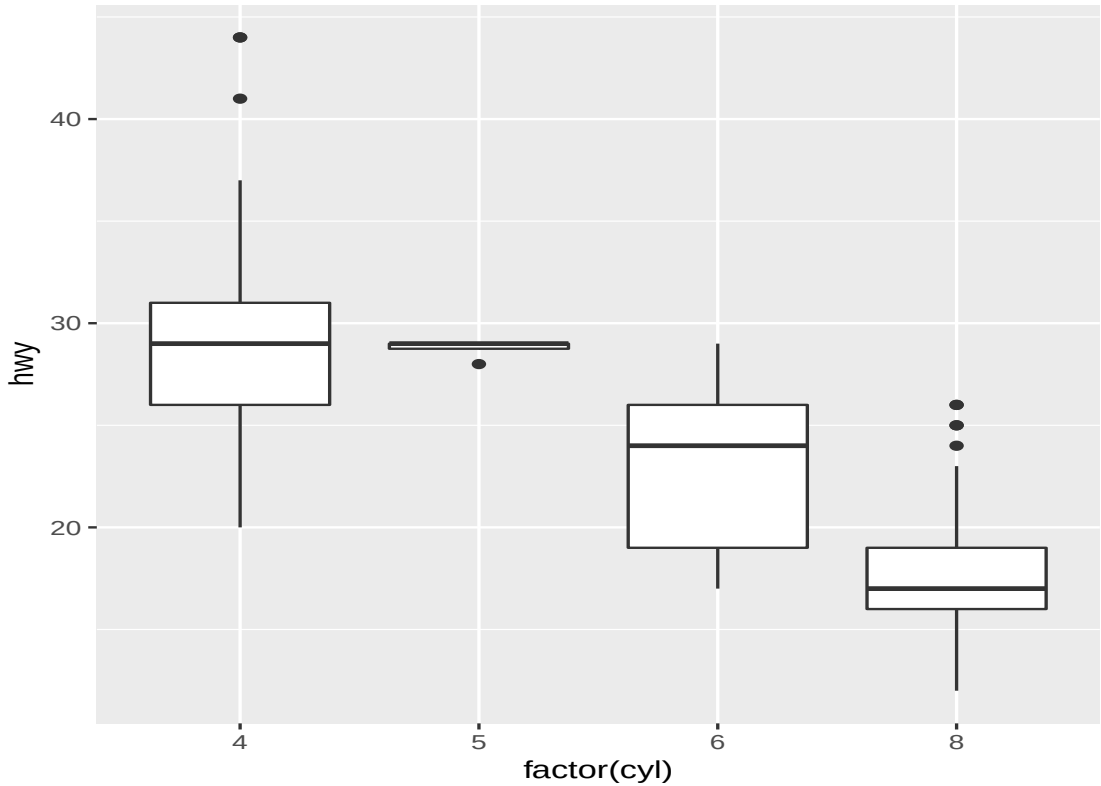
Now we'll create a boxplot: `(g1 = g + geom_boxplot())`



Combining geoms

Previous figure was a boxplot of all the `mpg` data, a more useful plot would be to have individual boxplots conditional on number of cylinders:

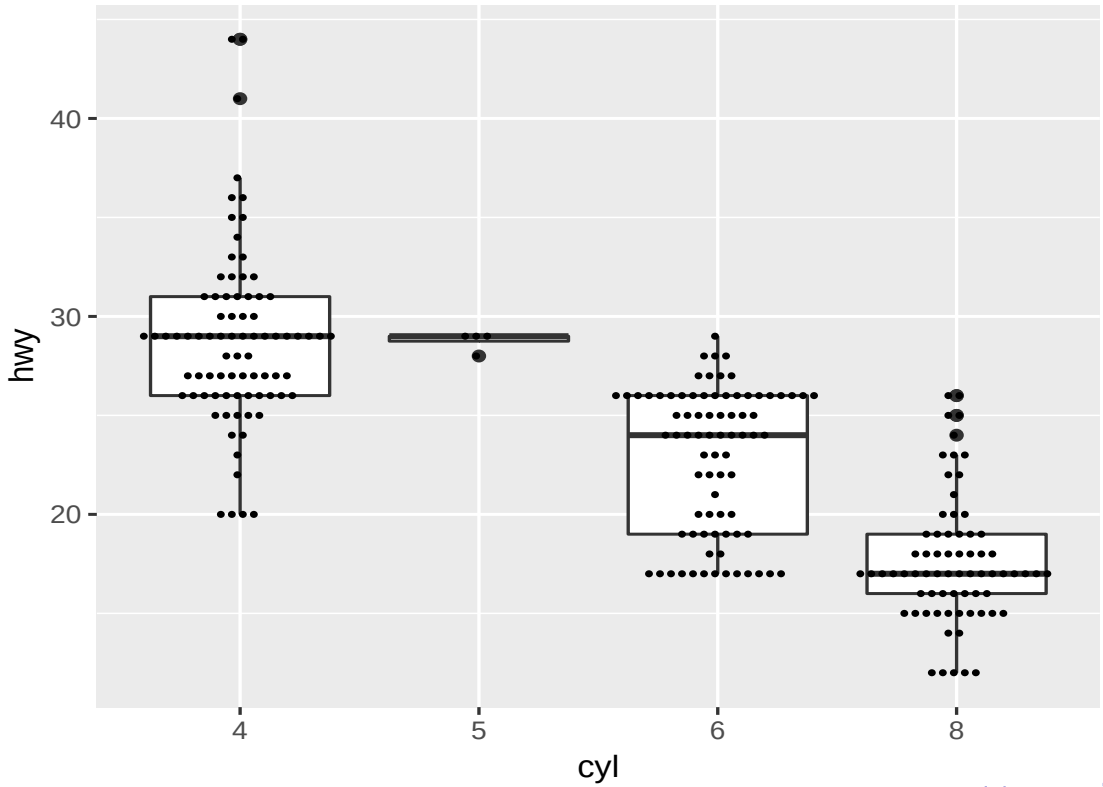
```
(g2 = g + geom_boxplot(aes(x=factor(cyl), group=cyl)))
```



Combining geoms

We are not restricted to a single geom. When data sets are reasonably small, it is useful to display the data on top of the boxplots:

```
(g3 = g2 + geom_dotplot(aes(x=factor(cyl), group=cyl),
binaxis="y", stackdir="center", binwidth=0.25,
stackratio=2))
```



Standard plots

There are a few standard geom 's that are particular useful:

- geom_line : a line plot.
- geom_boxplot : produces a boxplot.
- geom_point : a scatter plot.
- geom_dotplot: a dot plot.
- geom_bar : produces a standard barplot that counts the x values.
- geom_text : adds labels to specified points (as `geom_point` but draw labels rather than points).
- geom_raster: Similar to `levelplot` (heatmap).

Useful links

Choice of graphic colours

Names of colours in various formats:
http://colorbrewer2.org/

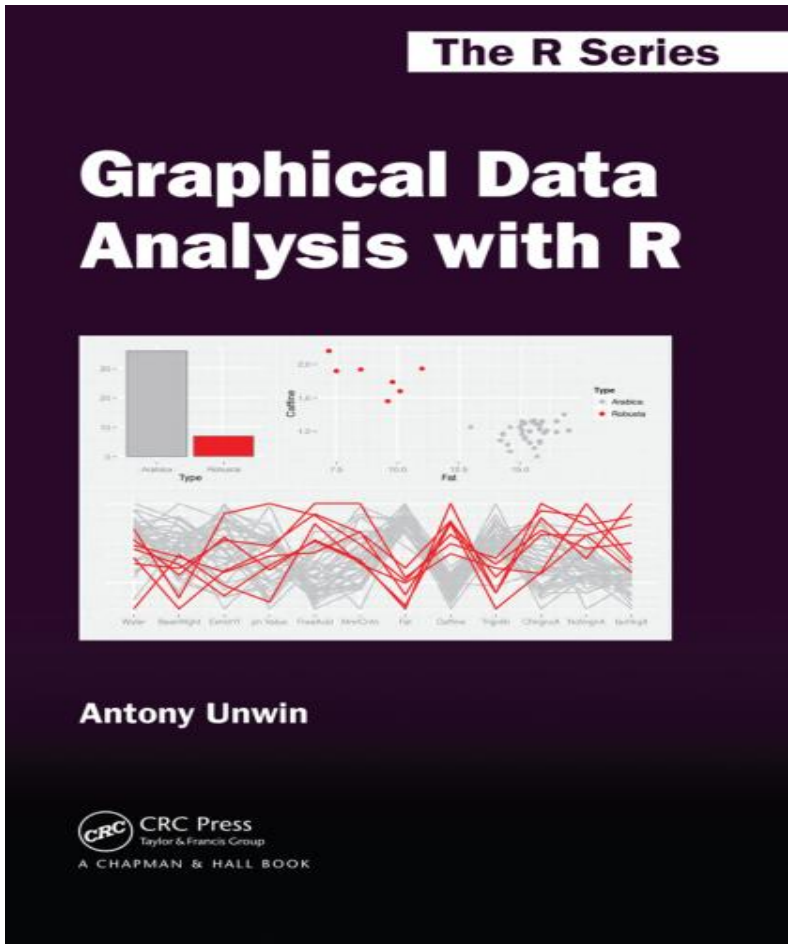
R Course: Shiny web-page

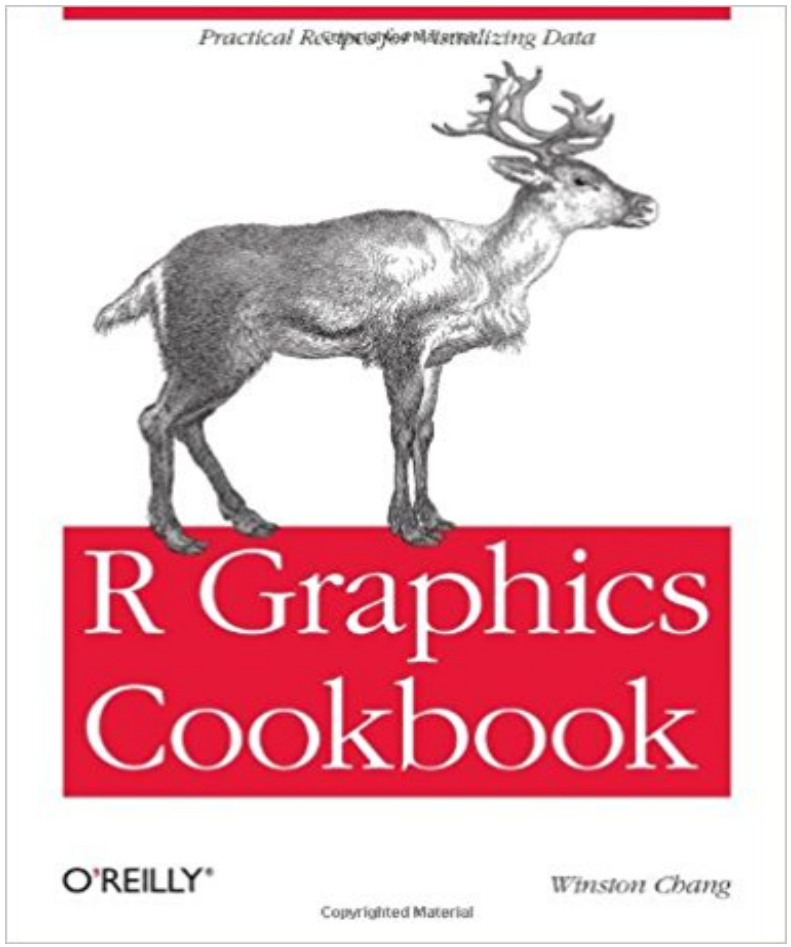
The RVis tool for learners of data visualisation using R:
http://bristol-medical-stat.bristol.ac.uk:3838/RVis/

R package: BristolVis web-page

The BristolVis package that associated with these materials:
https://github.com/statcourses/BristolVis

References


The R Series
Graphical Data Analysis with R
Antony Unwin
CRC Press
Taylor & Francis Group
A CHAPMAN & HALL BOOK


Practical Recipes for Visualizing Data
R Graphics Cookbook
O'REILLY
Winston Chang
Copyrighted Material

Thank You