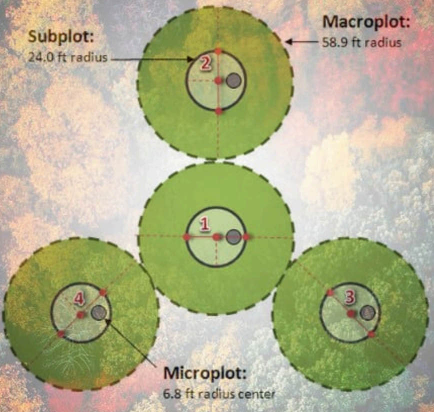
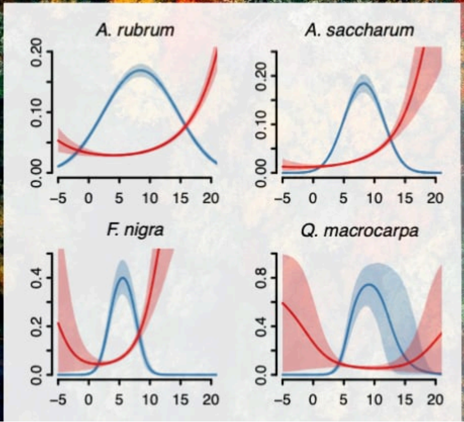
Lecture 2

Summary Statistics & Base Graphics

Introduction to R for Biologists - Lauren Talluto





Summary Statistics in R

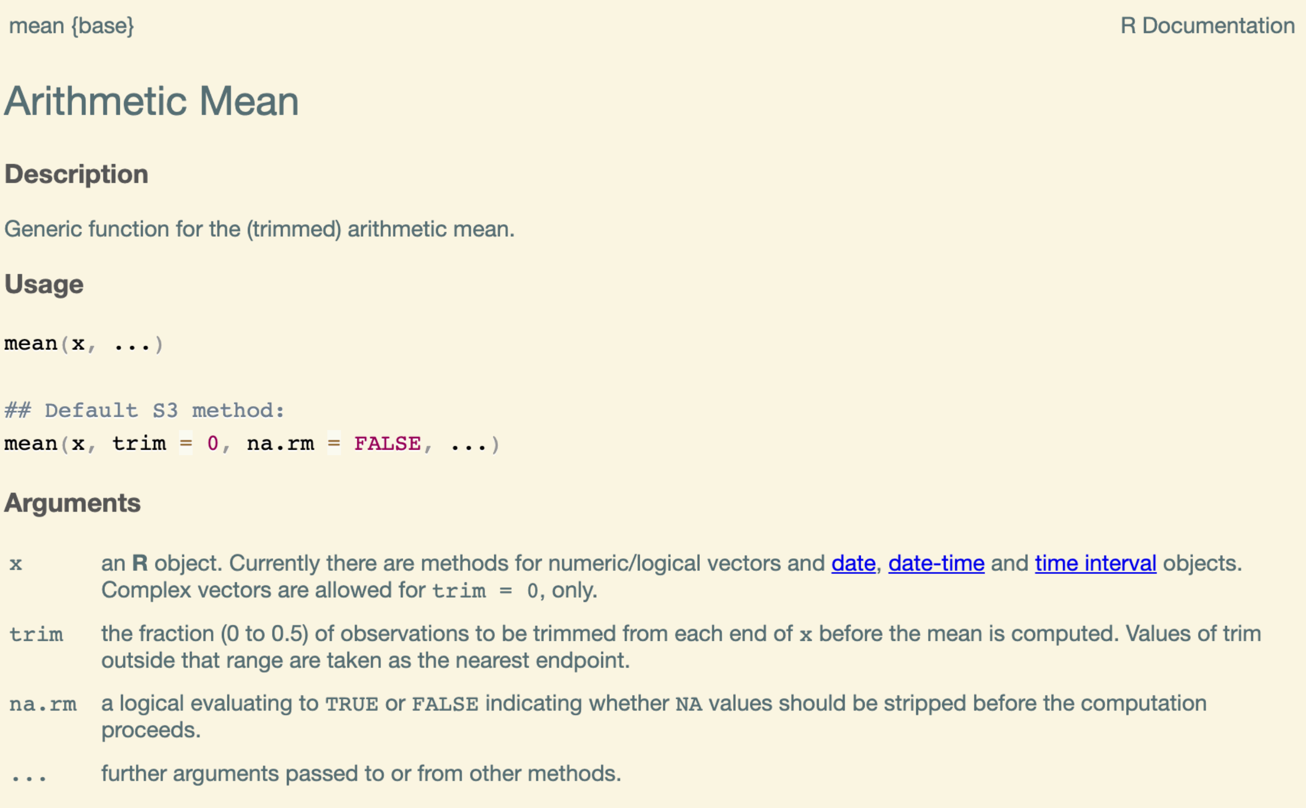
Helpful functions to try

```r
penguins = read.csv("data/penguins.csv")
y = penguins$body_mass_g

summary(y)

# minimum, maximum
range(y)
min(y)
max(y)

# central tendency
mean(y)
median(y)

# variability
var(y)
sd(y)
```

Dealing with missing values

The penguin dataset has some `NA` values.

Check the help functions: `?mean`. Do you see any option for dealing with NAs?



mean {base}　　R Documentation

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

`x`	an **R** object. Currently there are methods for numeric/logical vectors and date, date-time and time interval objects. Complex vectors are allowed for `trim = 0`, only.
`trim`	the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
`na.rm`	a logical evaluating to `TRUE` or `FALSE` indicating whether `NA` values should be stripped before the computation proceeds.
`...`	further arguments passed to or from other methods.

Removing missing values

You can use `subset()` with `complete.cases()` to remove ALL rows that have at least one missing value.

```
penguins = read.csv("data/penguins.csv")
nrow(penguins)
## [1] 344
```

```
head(penguins)
##   species    island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
## 1  Adelie Torgersen           39.1          18.7               181        3750
## 2  Adelie Torgersen           39.5          17.4               186        3800
## 3  Adelie Torgersen           40.3          18.0               195        3250
## 4  Adelie Torgersen             NA            NA                NA          NA
## 5  Adelie Torgersen           36.7          19.3               193        3450
## 6  Adelie Torgersen           39.3          20.6               190        3650
##      sex year
## 1   male 2007
## 2 female 2007
## 3 female 2007
## 4   <NA> 2007
## 5 female 2007
## 6   male 2007
```

Removing missing values

You can use `subset()` with `complete.cases()` to remove ALL rows that have at least one missing value.

```
penguins_no_na = subset(penguins, complete.cases(penguins))
nrow(penguins_no_na)
## [1] 333
```

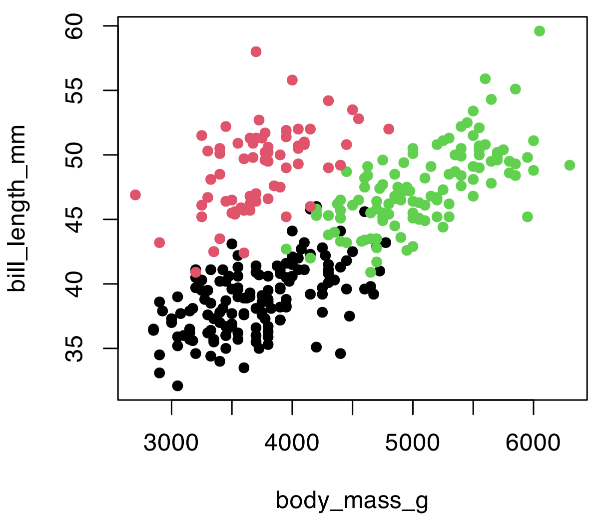
```
head(penguins_no_na)
##   species    island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
## 1  Adelie Torgersen           39.1          18.7               181        3750
## 2  Adelie Torgersen           39.5          17.4               186        3800
## 3  Adelie Torgersen           40.3          18.0               195        3250
## 5  Adelie Torgersen           36.7          19.3               193        3450
## 6  Adelie Torgersen           39.3          20.6               190        3650
## 7  Adelie Torgersen           38.9          17.8               181        3625
##      sex year
## 1   male 2007
## 2 female 2007
## 3 female 2007
## 5 female 2007
## 6   male 2007
## 7 female 2007
```

Dealing with partitioned data

Our data have meaningful partitions, especially by `species` and `sex`!

Our summary statistics will make more sense if we compute them seperately by species / sex.

```
with(penguins_no_na,
    plot(body_mass_g, bill_length_mm, col = factor(species), pch = 16)
)
```



Computing by partition, the slow way

```
mean(penguins_no_na$bill_length_mm[penguins_no_na$species == "Adelie"])
## [1] 38.82397
```

```
mean(penguins_no_na$bill_length_mm[penguins_no_na$species == "Chinstrap"])
## [1] 48.83382
```

```
mean(penguins_no_na$bill_length_mm[penguins_no_na$species == "Gentoo"])
## [1] 47.56807
```

Computing by partition, the smarter way

```r
tapply(penguins_no_na$bill_length_mm, # first the variable you want to summarise
       penguins_no_na$species, # then the partitioning variable
       mean) # then the function you want to use for summarising
##    Adelie Chinstrap    Gentoo
##  38.82397  48.83382  47.56807
```

Multiple partitions

```
tapply(penguins_no_na$bill_length_mm,
       penguins_no_na[, c('species', 'sex')],
       mean)
##             sex
## species        female     male
##   Adelie    37.25753 40.39041
##   Chinstrap 46.57353 51.09412
##   Gentoo    45.56379 49.47377
```

R base graphics

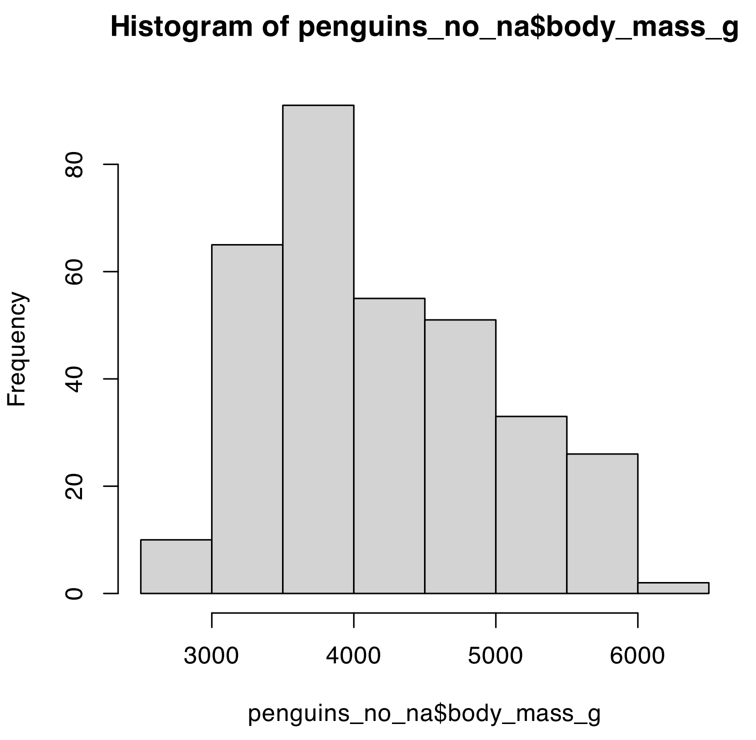
A brief summary

Histograms

A histogram shows the distribution of a single variable.

The default from `hist` is pretty ugly.

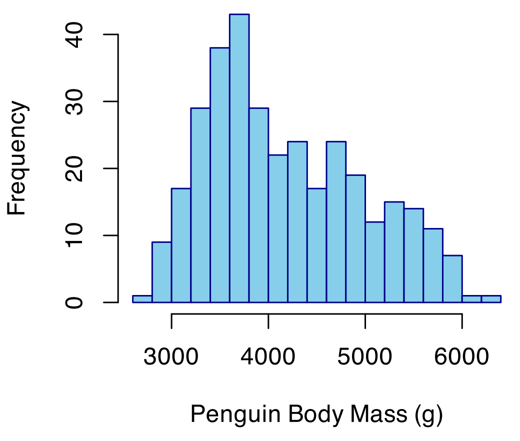
```
hist(penguins_no_na$body_mass_g)
```



Histograms

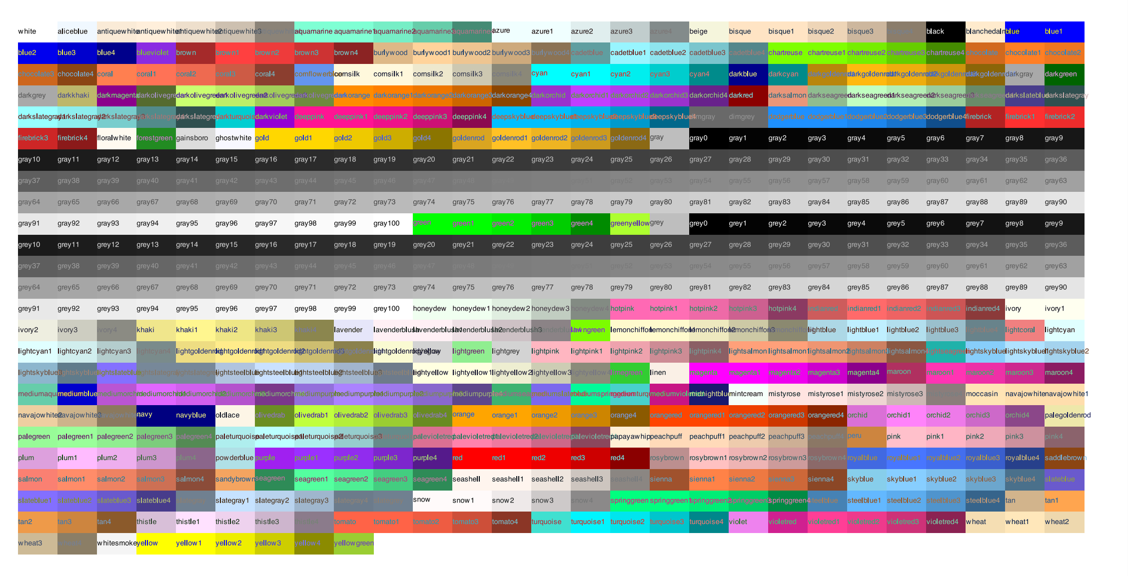
There are many options you can adjust.

```r
hist(penguins_no_na$body_mass_g,
     main = "", # disables the title
     xlab = "Penguin Body Mass (g)",
     breaks = 15, # adjust the number of bins
     col = 'skyblue', border = 'darkblue')
```



Colours - named colours

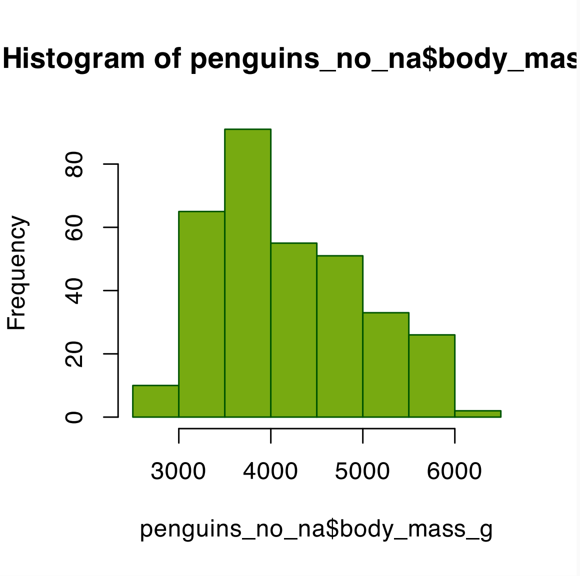
- R has 657 named colours: `col = 'rosybrown'`
- see the `colors()` function for the names



Colours - 24-bit colour

- R supports colours using the common HTML color coding: `#RRGGBB`
    - RR, GG, BB are the amounts of red, green, and blue
    - each ranges from `00` (none) to `FF` (most)
    - Colour pickers online help you translate a color in real life to a coded color

```
hist(penguins_no_na$body_mass_g, col = "#77aa11", border = "#005500")
```



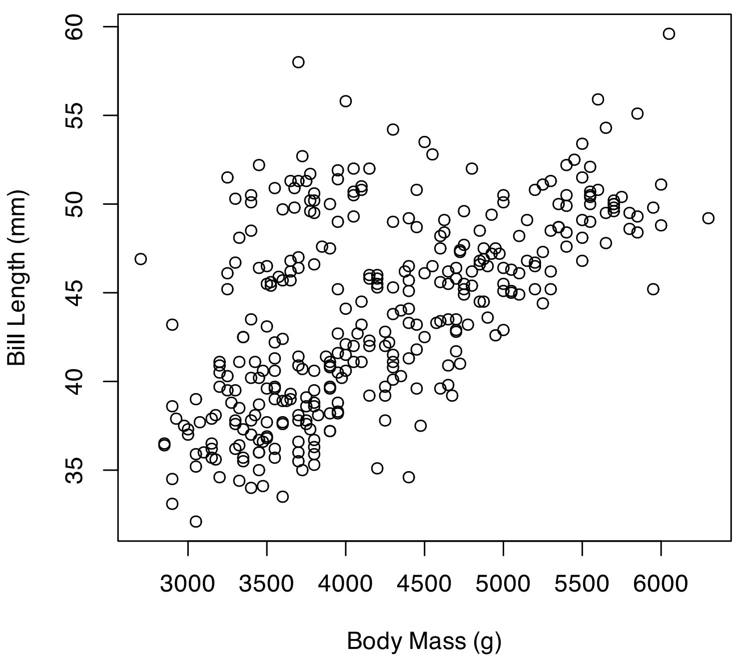
Colour advice

- Best practise: choose colours using reputable packages based in colour theory
    - scico (continuous data)
    - viridis (continuous data)
    - rcolorbrewer (Categorical data)
    - iWantHue (generates custom categorical palettes)

Scatterplots

For comparing how two variables *covary*

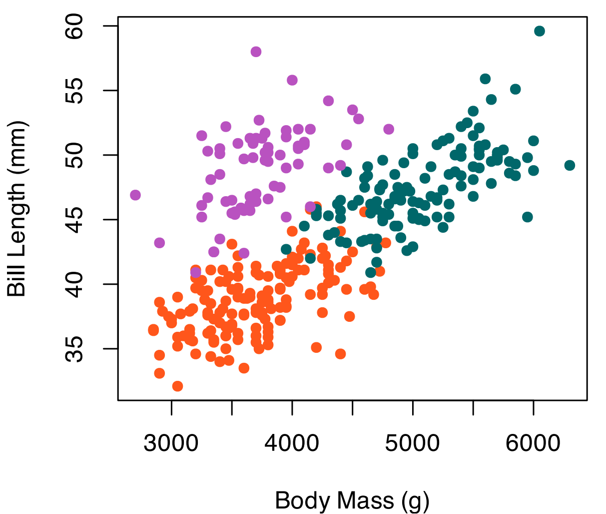
```
plot(penguins$body_mass_g, penguins$bill_length_mm,
     xlab = "Body Mass (g)", ylab = "Bill Length (mm)")
```



Scatterplots

I can create a variable to store the colour I want to use for each point.

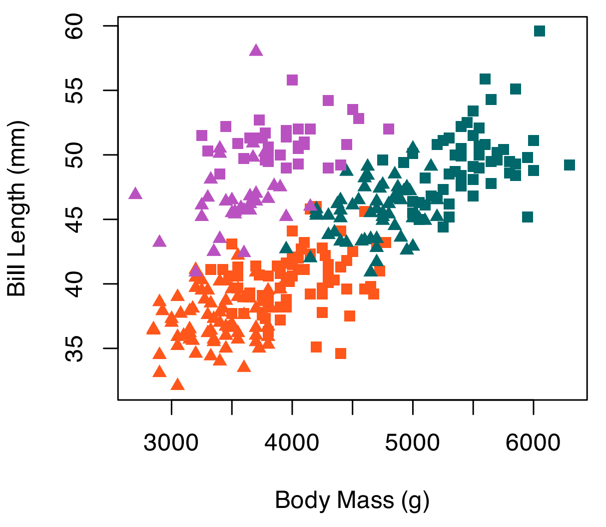
```
penguins$color[penguins$species == "Adelie"] = "#ff561b"
penguins$color[penguins$species == "Chinstrap"] = "#b952c0"
penguins$color[penguins$species == "Gentoo"] = "#00676a"


plot(penguins$body_mass_g, penguins$bill_length_mm,
     xlab = "Body Mass (g)", ylab = "Bill Length (mm)",
     col = penguins$color, pch = 16) # pch=16 uses a solid circle
```



Scatterplots

I can do the same for sex. `ifelse()` works great if you have 2 categories!

```r
penguins$symbol = ifelse(penguins$sex == "male", 15, 17)

plot(penguins$body_mass_g, penguins$bill_length_mm,
     xlab = "Body Mass (g)", ylab = "Bill Length (mm)",
     col = penguins$color, pch = penguins$symbol)
```



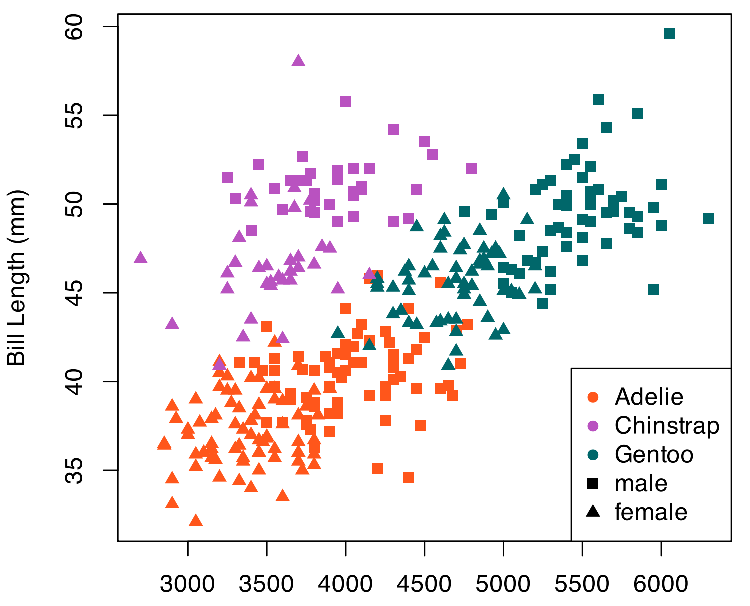
Annotations

You may *annotate* plots by running commands after you use the `plot()` command. Things to try:

```
legend() # create a legend
abline() # create a line if you know intercept/slope
lines() # more general function for adding lines
points() # add x-y points
text() # Add text annotations
mtext() # Add text to plot margins
```

Add a legend

```
plot(penguins$body_mass_g, penguins$bill_length_mm,
     xlab = "Body Mass (g)", ylab = "Bill Length (mm)",
     col = penguins$color, pch = penguins$symbol)

legend("bottomright",
       legend = c("Adelie", "Chinstrap", "Gentoo", "male", "female"),
       col = c("#ff561b", "#b952c0", "#00676a", "black", "black"),
       pch = c(16, 16, 16, 15, 17))
```

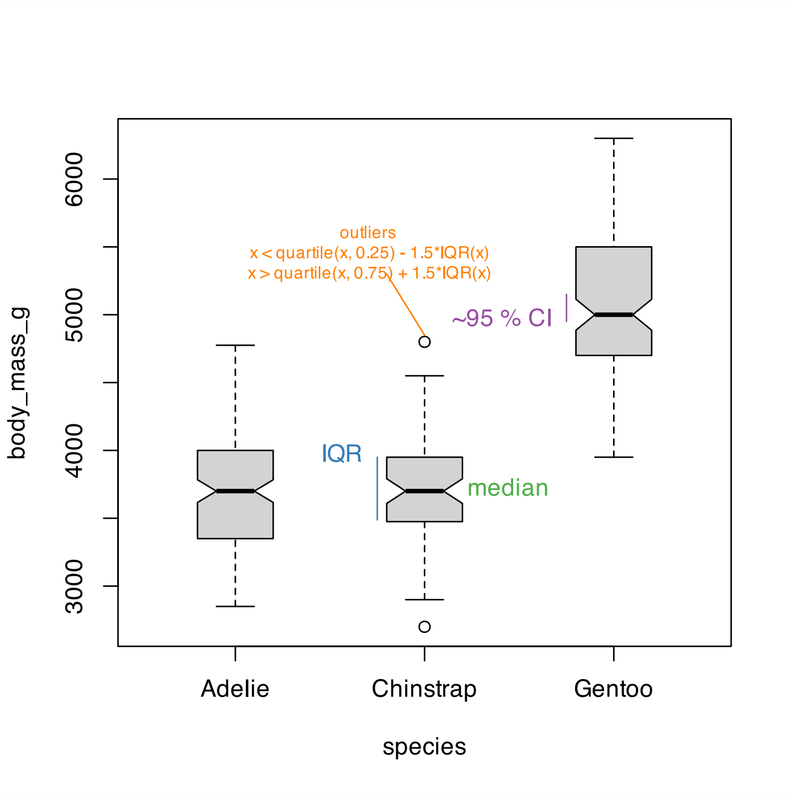


Boxplots: For summarizing across partitions

- **Boxplots** (sometimes **box-and-whisker** diagrams) summarize key statistics.
  - median
  - first and third quartiles (hinges)
  - approx. 95% confidence interval for median (notch)
  - min/max (or quartile + 1.5*IQR) (whiskers)
- They are very useful for comparing variables among groups.
- `y ~ group` is a special data type called a `formula`

Boxplots: For summarizing across partitions

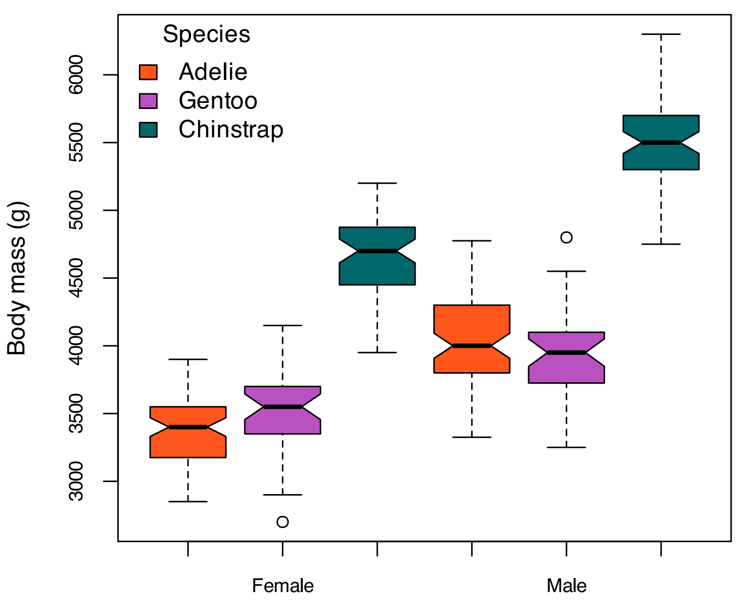
```
boxplot(body_mass_g ~ species, data = penguins, boxwex=0.4, notch = TRUE)
```



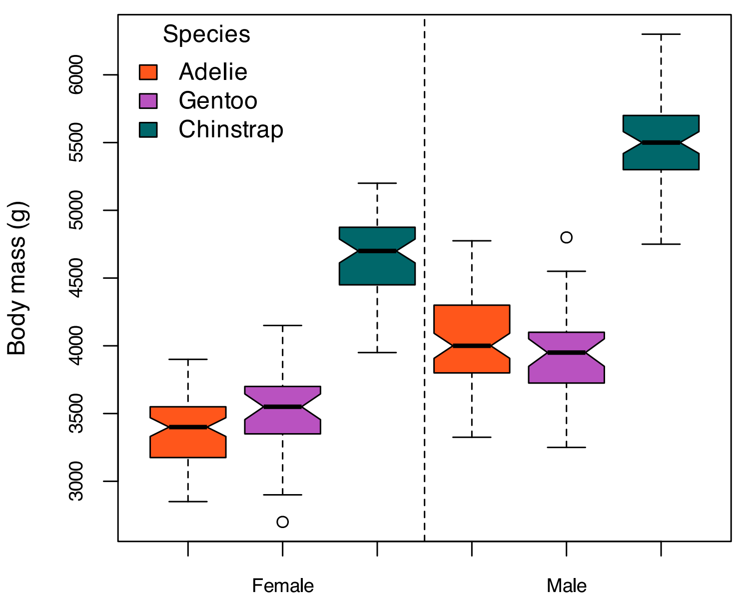
Making a boxplot

```r
cols = c("#ff561b", "#b952c0", "#00676a")
boxplot(body_mass_g ~ species+sex, data = penguins,
        # set the colours, repeated twice (female and male)
        col = rep(cols, 2),
        # axis label text size
        cex.axis = 0.8,
        # labels under the boxes
        names = c("", "Female", "", "", "Male", ""),
        # set axis titles
        xlab = "", ylab = "Body mass (g)",
        # disable the box around the plot
        bty = 'n',
        notch = TRUE
)
# add a legend to the plot
legend("topleft", legend = unique(penguins$species),
       title = "Species", fill = cols, bty = 'n')
```

Boxplots: For summarizing across partitions



Boxplots: For summarizing across partitions



Add a line with `abline()` to distinguish male vs female.

Or experiment with the `at =` argument.