Introduction to Statistics and R

Computing with and plotting of data tables

Eric Stemmler

Khovd University

24.03.2021

Section 1

Recap: Accessing data tables

Recap: Accessing data tables

- Accessing columns from a `data.table`
  - Single columns: e.g. `dt$colA`
  - Multiple columns: e.g. `dt[, c("colA", colB")]`
- Accessing rows/ subsetting a data.table
  - By row number: e.g. `dt[1:3]` returns first 3 rows
  - By variable values: e.g. `dt[colA > 3 & colB < 2]`

Section 2

Aggregating grouped data

Calculating measures of columns

```
library(data.table)
dt <- fread("age-guessing.csv")
dt[, mean(card_1)]
```

```
## [1] -2.3
```

```
dt[, median(card_1)]
```

```
## [1] -1.5
```

```
dt[, var(card_1)]
```

```
## [1] 24.01111
```

```
dt[, var(card_1) + var(card_2)]
```

```
## [1] 120.2444
```

Using factors

- How to categorize data? Use factors!
- factors are categories. Example: "mammals"
  - Horse
  - Sheep
  - Cow
  - Goat
- different categories in a factor are called "levels"
- In R we say: factor `mammals` has levels `Horse`, `Sheep`, `Cow`, `Goat`

Using factors

mammal	age	weight
horse	2	716.26
horse	3	546.82
horse	2	452.83
sheep	2	94.58
sheep	3	100.56
sheep	3	91.08
cow	3	747.53
cow	2	587.70
cow	3	588.39
goat	1	69.63
goat	2	79.12
goat	2	26.48

Using factors

Example: Create a factor in R

```
my_factor <- factor(x = c("horse", "sheep", "cow", "goat"))
class(my_factor)
```

```
## [1] "factor"
```

```
my_factor
```

```
## [1] horse sheep cow   goat
## Levels: cow goat horse sheep
```

Using factors

Use function repeat function `rep()` to create multiple entries of categories: Parameter `each` defines how many times *each* item is repeated

```
# use rep() to avoid too much typing:
my_categories <- rep(x = c("horse", "sheep", "cow", "goat"),
    each = 3L)
my_categories
```

```
##  [1] "horse" "horse" "horse" "sheep" "sheep" "sheep"
##  [7] "cow"   "cow"   "cow"   "goat"  "goat"  "goat"
```

Using factors

... or using parameter `times` to repeat the whole sequence:

```
# use rep() to avoid too much typing:
my_categories <- rep(x = c("horse", "sheep", "cow", "goat"),
    times = 3L)
my_categories
```

```
##  [1] "horse" "sheep" "cow"   "goat"  "horse" "sheep"
##  [7] "cow"   "goat"  "horse" "sheep" "cow"   "goat"
```

Using factors

... and then convert to factor

```
my_categories <- factor(my_categories)
my_categories
```

```
##  [1] horse sheep cow   goat  horse sheep cow   goat
##  [9] horse sheep cow   goat
## Levels: cow goat horse sheep
```

Note: Levels are sorted alphabetically

Calculating measures per groups

Calculating e.g. the mean of column `weight`:

```
dt[, mean(weight)]
```

```
## [1] 341.7474
```

. . . not very meaningful, since these are different mammals. How to calculate for each mammal type?:

```
dt[, mean(weight), by = mammal]
```

```
##    mammal        V1
## 1:  horse 571.97033
## 2:  sheep  95.40519
## 3:    cow 641.20540
## 4:   goat  58.40853
```

Calculating measures per groups

Using `by`: Do anything you do in `rows` on `columns` for each different value in `group` separately

```
dt[rows, columns, by = group]
```

Calculating measures per groups

Calculating e.g. the mean of column `weight`:

```
dt[, mean(weight)]
```

```
## [1] 341.7474
```

. . . not very meaningful, since these are different mammals. How to calculate for each mammal type?:

```
dt[, mean(weight), by = mammal]
```

```
##    mammal        V1
## 1:  horse 571.97033
## 2:  sheep  95.40519
## 3:    cow 641.20540
## 4:   goat  58.40853
```

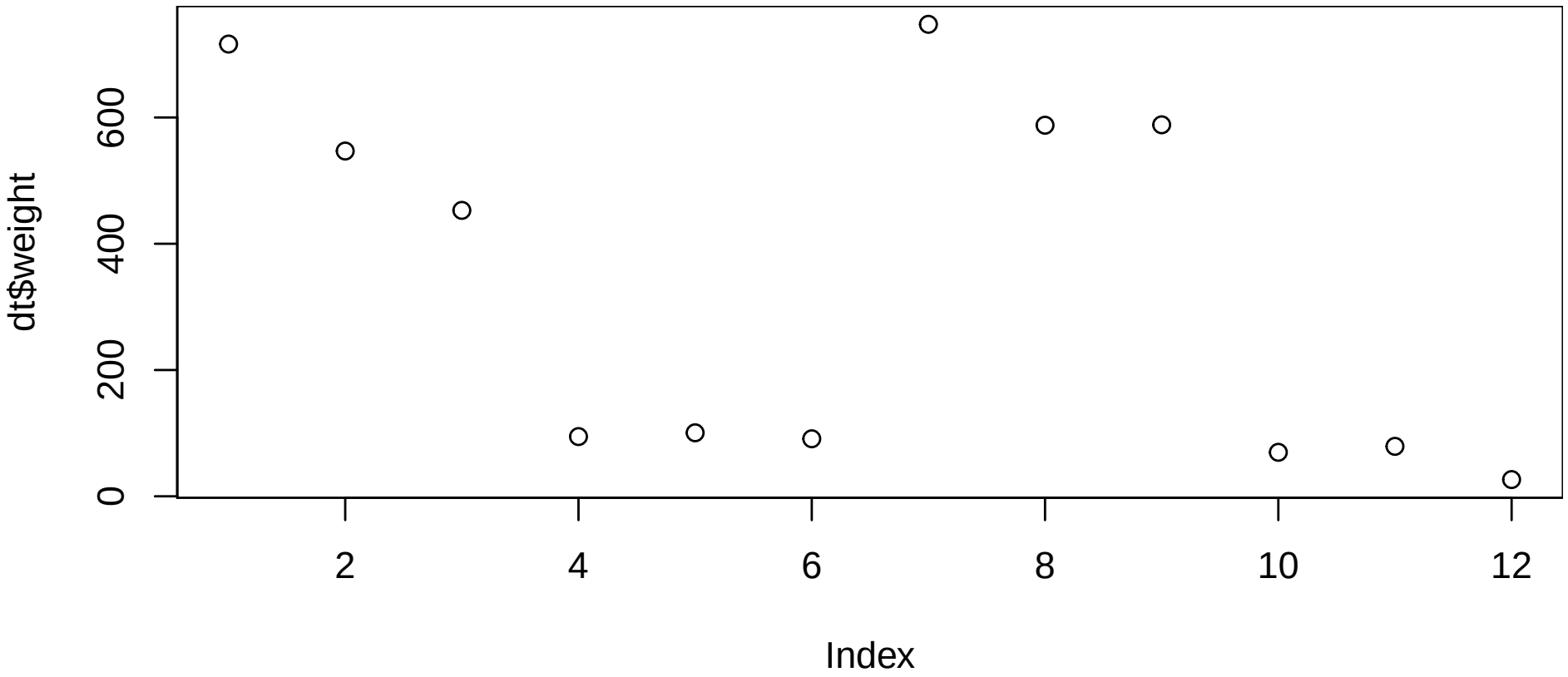
Section 3

Data Visualization: Plotting

Plotting in R

Scatterplot: Plot values from **one** or two columns

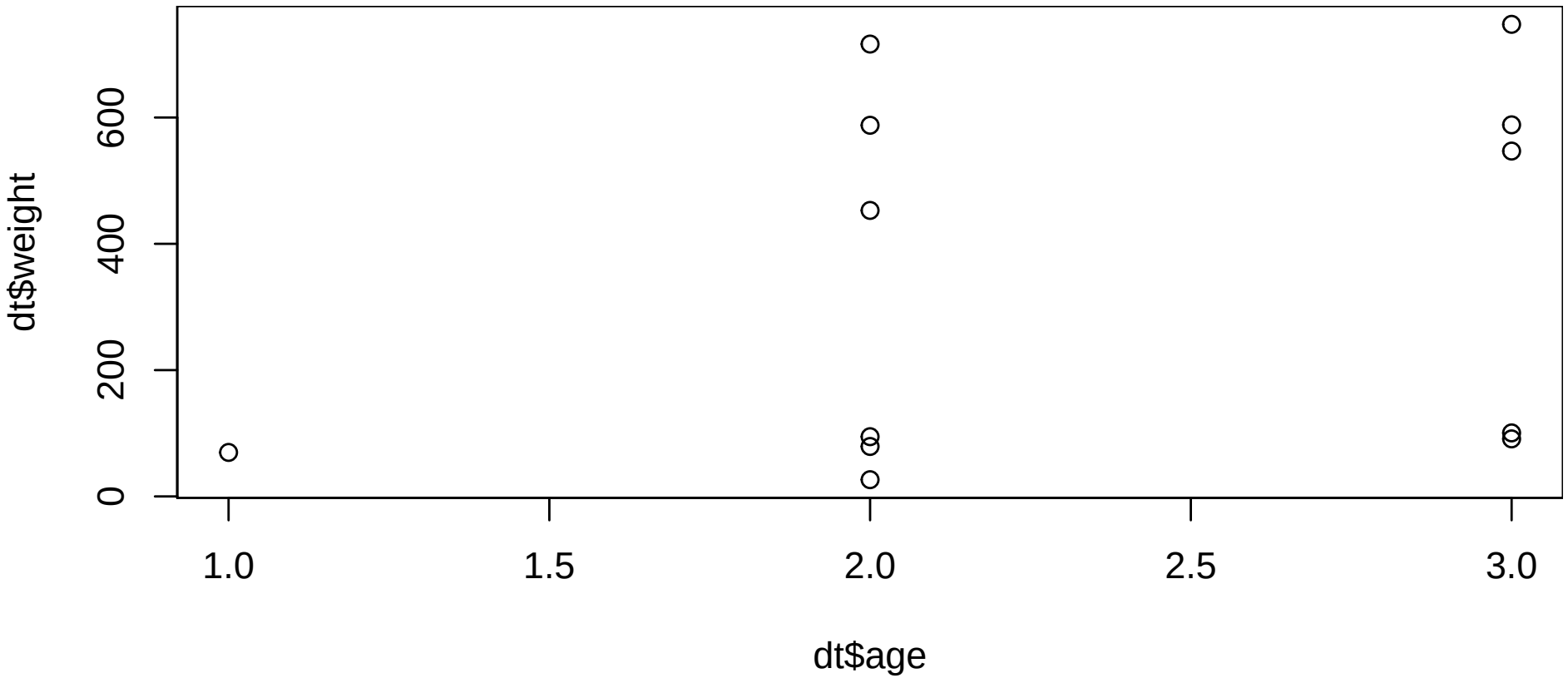
```
plot(dt$weight)
```



Plotting in R

Scatterplot: Plot values from one or **two** columns

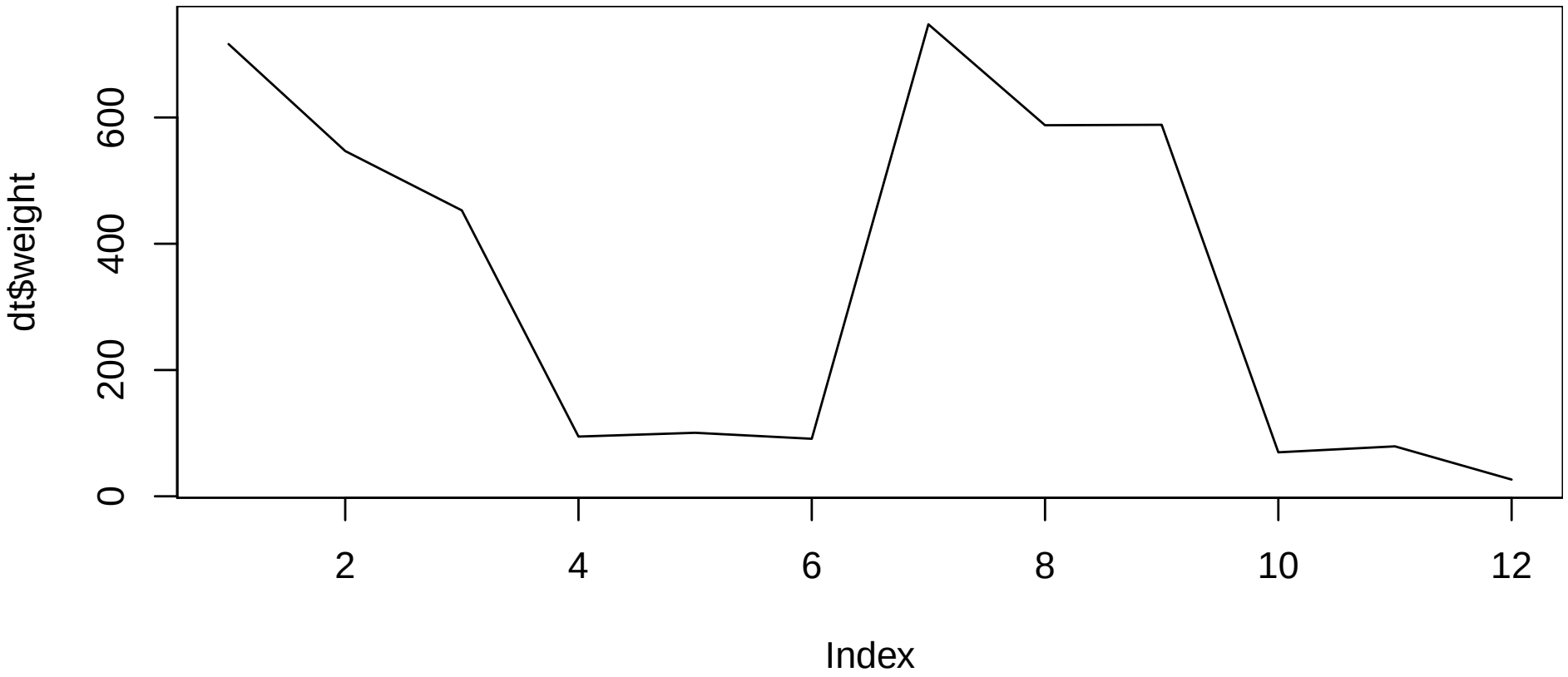
```
plot(y = dt$weight, x = dt$age)
```



Different plot types

Line plot: Like scatterplot where points are connected by lines

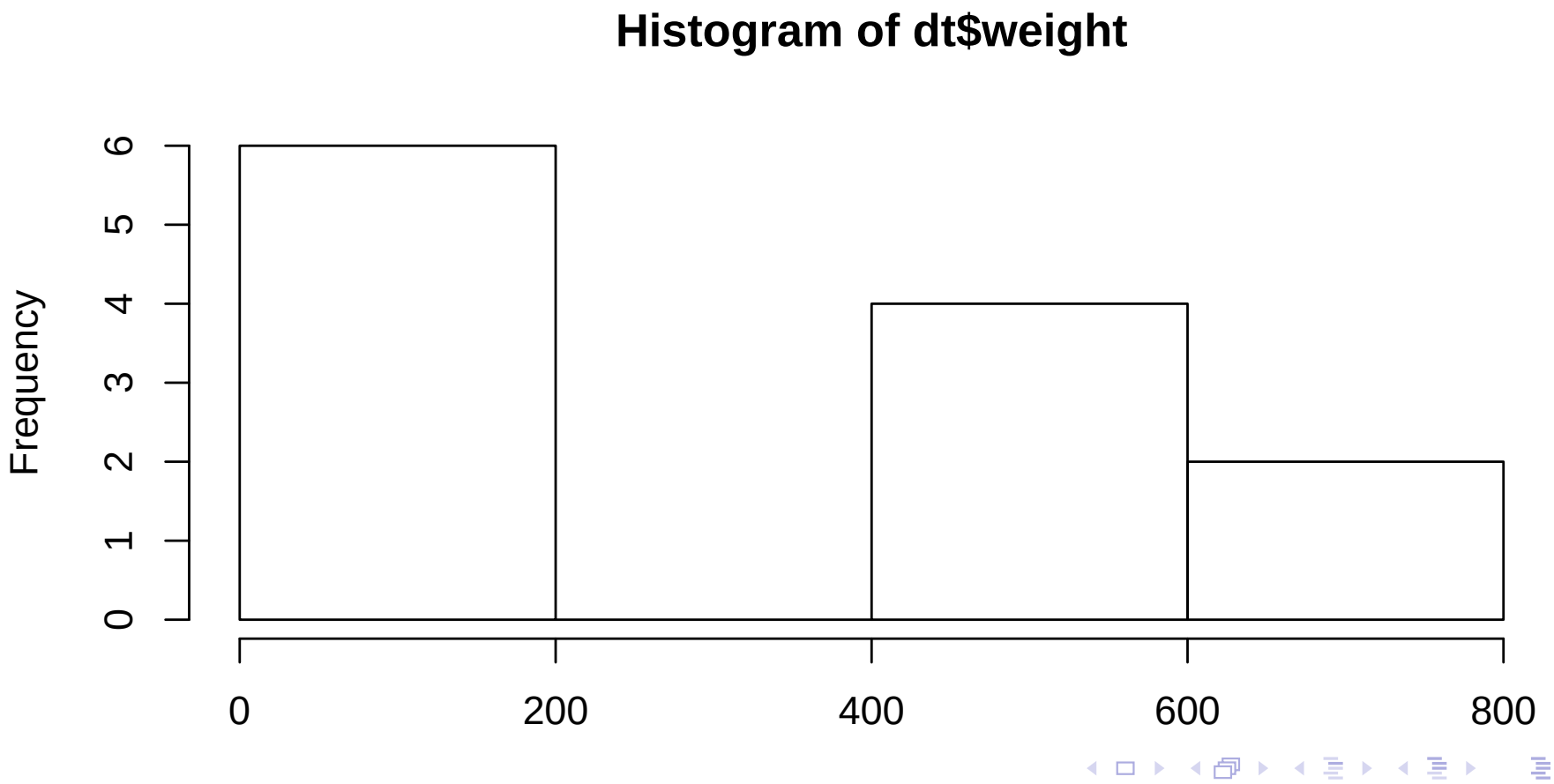
```
plot(dt$weight, type = "l")
```



Different plot types

Histogram

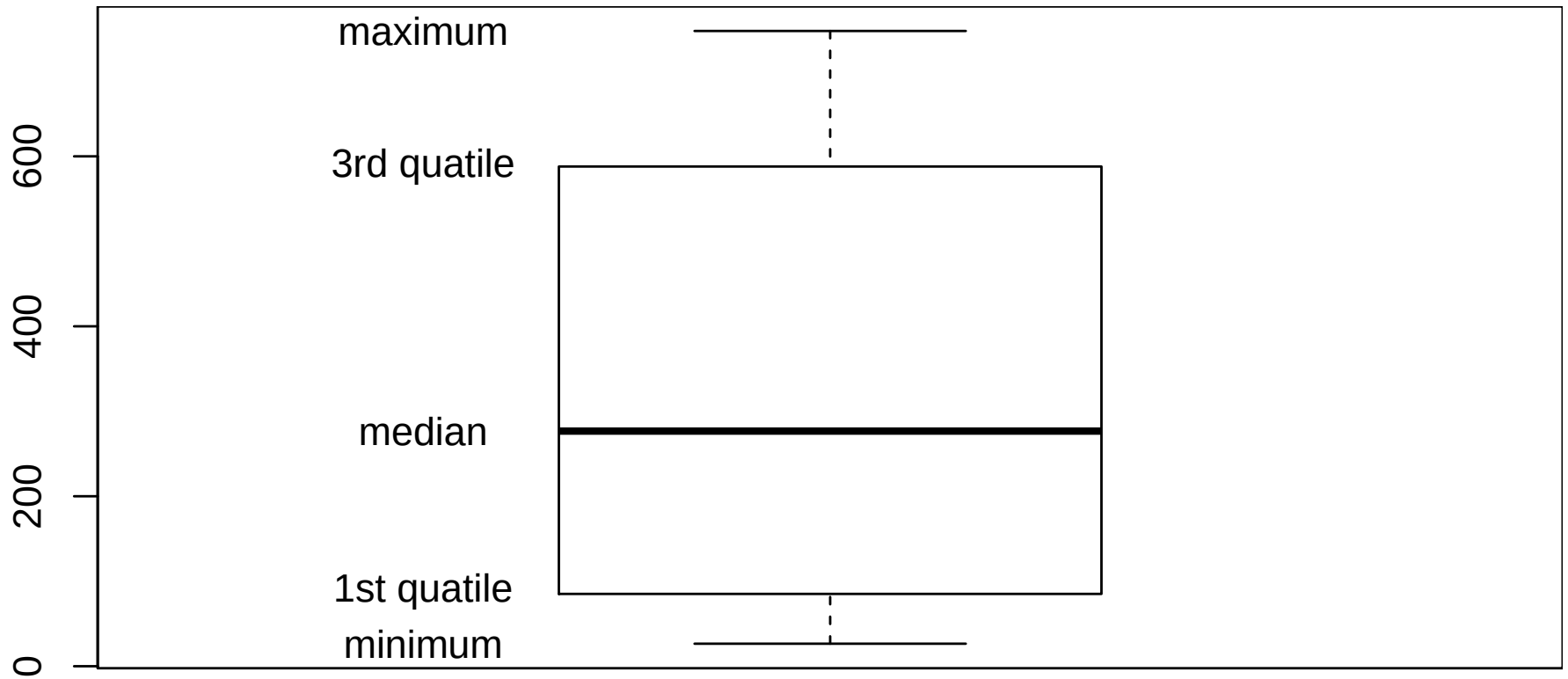
```
hist(dt$weight)
```



Different plot types

Box-wishker plot

```
boxplot(dt$weight)
```



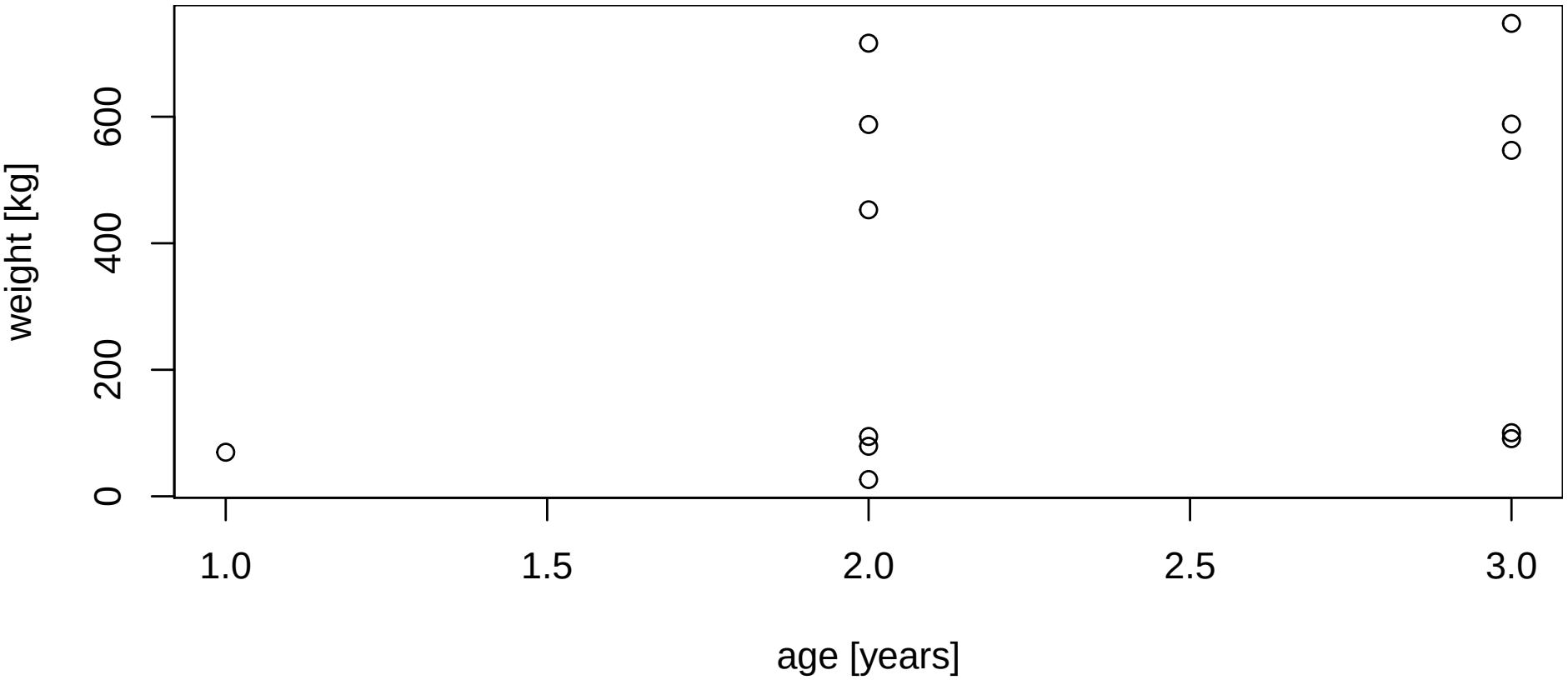
Subsection 1

Styling a plot

Titles and lables

Scatterplot: Changing **axis labels**

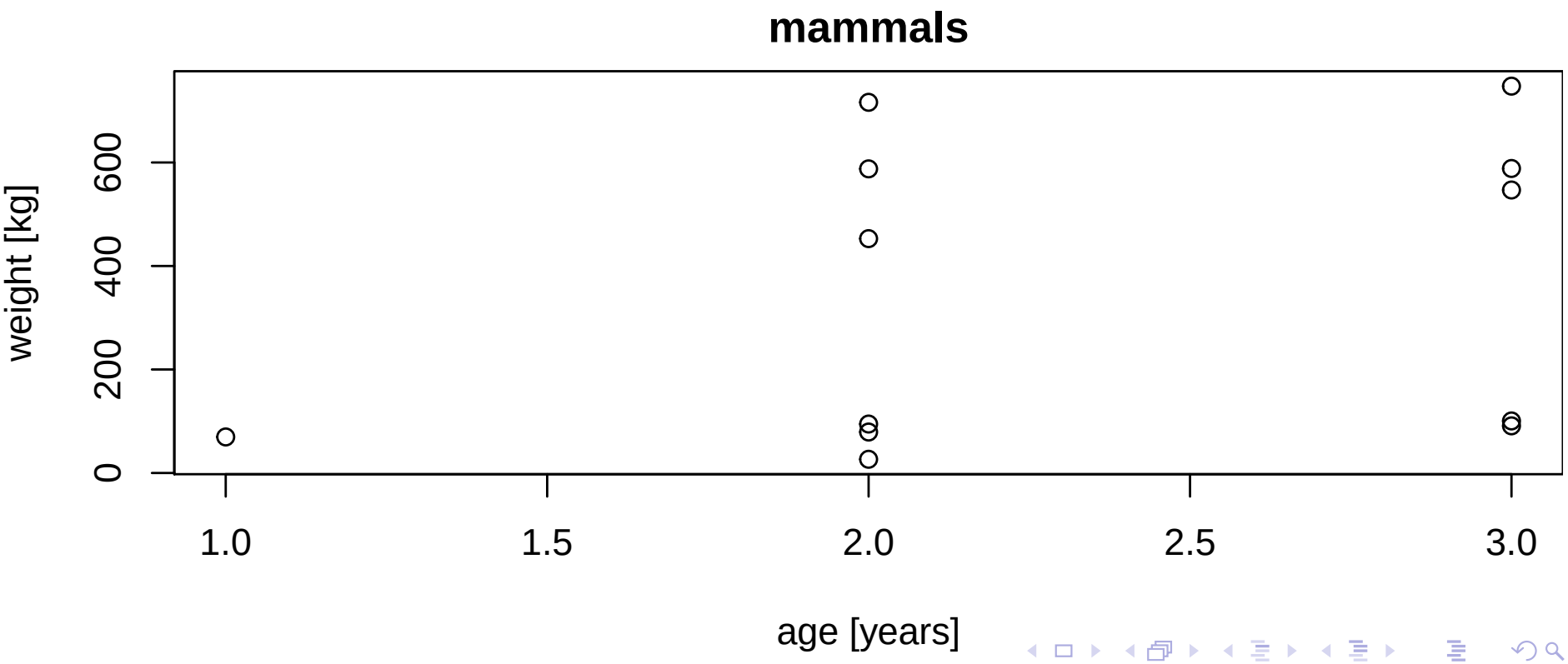
```
plot(y = dt$weight, x = dt$age,
     xlab = "age [years]", ylab = "weight [kg]")
```



Titles and lables

Scatterplot: adding a **title**

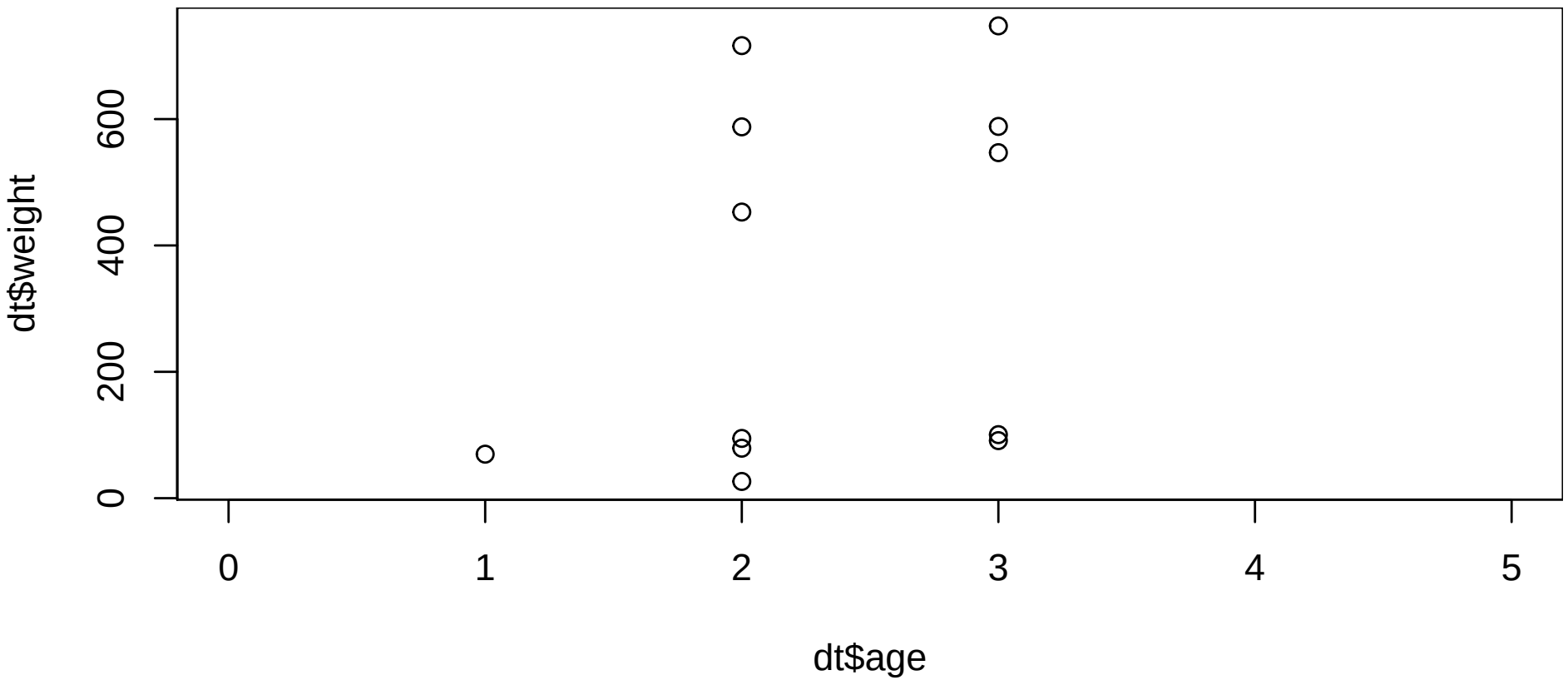
```
plot(y = dt$weight, x = dt$age,
     xlab = "age [years]", ylab = "weight [kg]",
     main = "mammals")
```



Adjusting the axis

Scatterplot: Changing **axis limits**

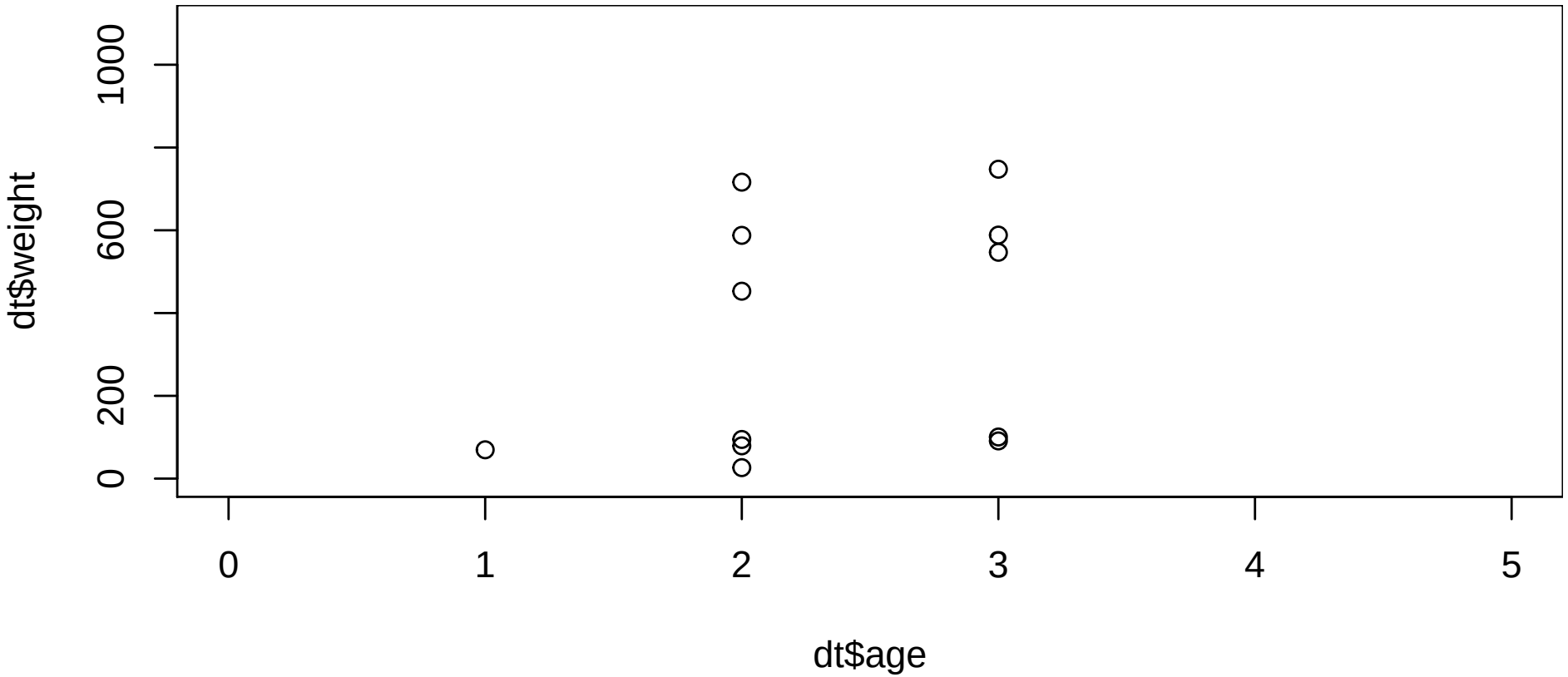
```
plot(y = dt$weight, x = dt$age,
     xlim = c(0, 5))
```



Adjusting the axis

Scatterplot: Changing **axis limits**

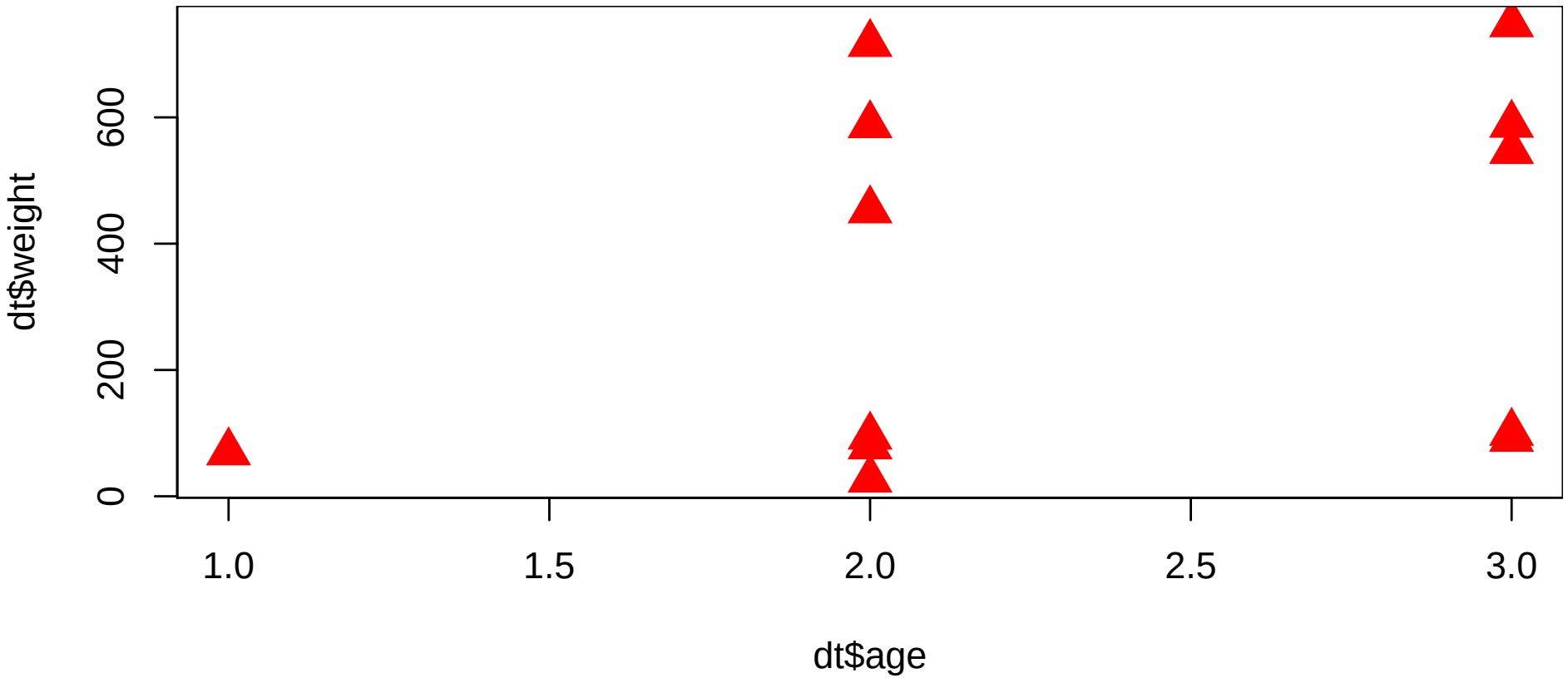
```
plot(y = dt$weight, x = dt$age,
     xlim = c(0, 5), ylim = c(0, 1100))
```



Color, shape and size

Scatterplot: Changing **color**, **shape** and **size**

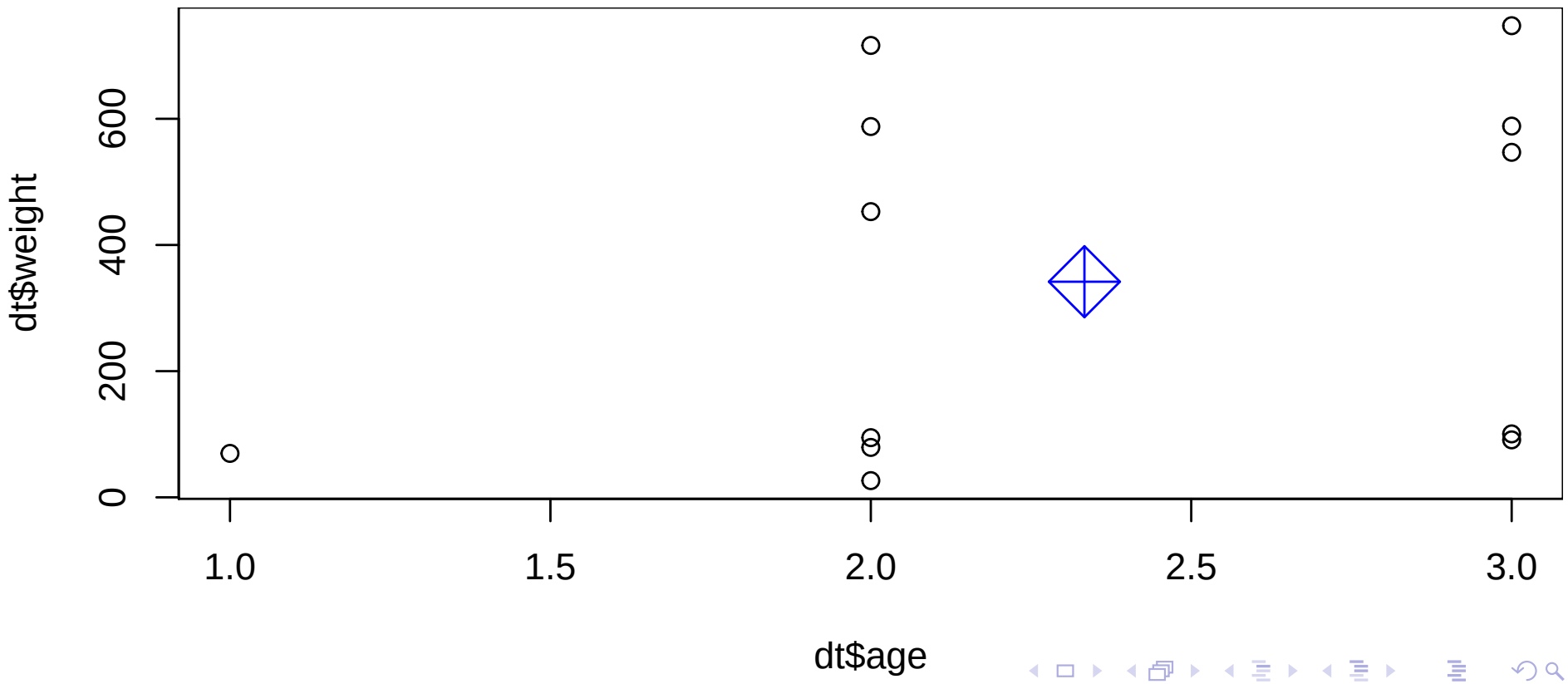
```
plot(y = dt$weight, x = dt$age,
     col = "red", pch = 17, cex = 2)
```



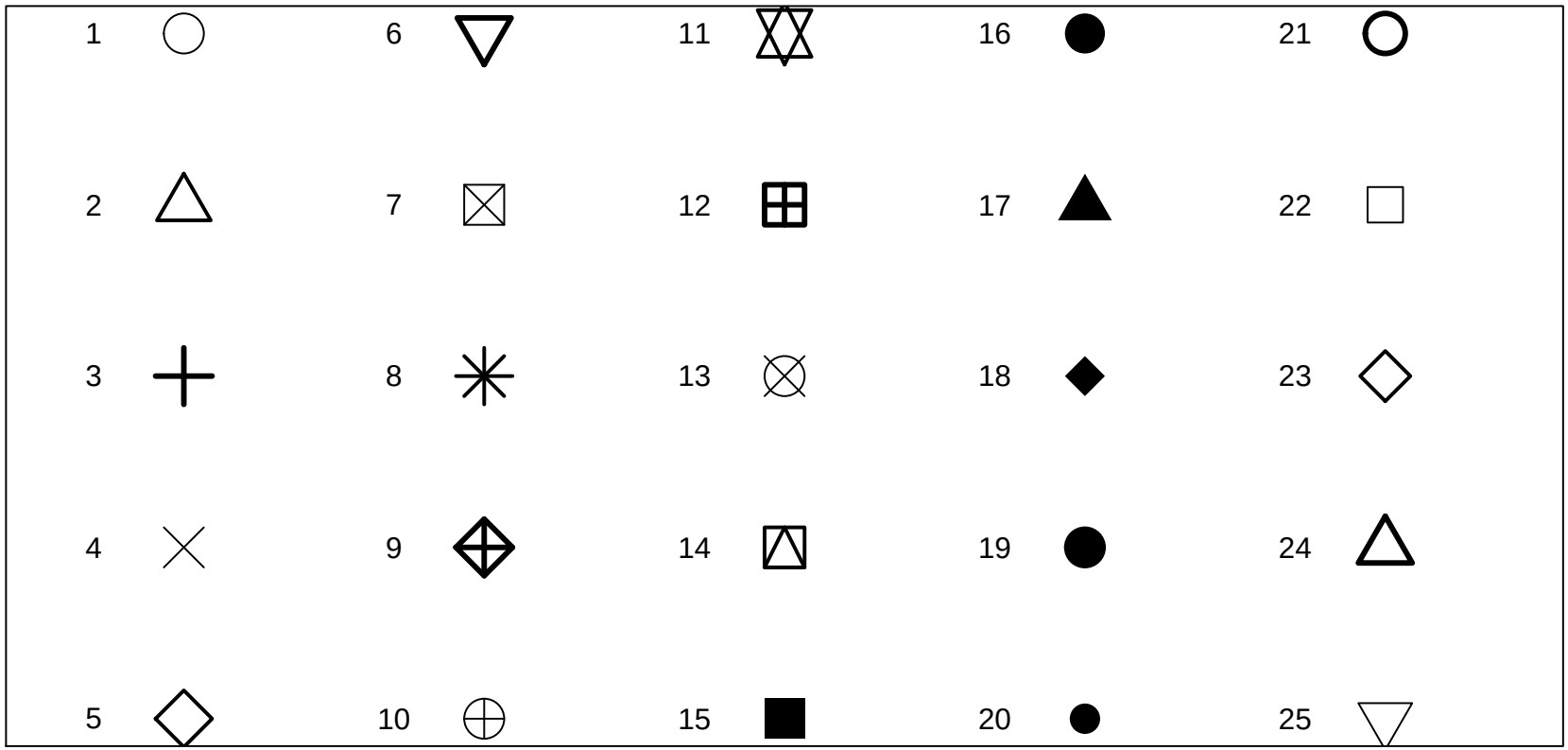
Color, shape and size

Scatterplot: Adding points

```
plot(y = dt$weight, x = dt$age)
points(y = mean(dt$weight), x = mean(dt$age),
       col = "blue", pch = 9, cex = 3)
```



Color, shape and size



Section 4

Summary

What functions did we learn?

- `mean()`, `median()`, `var()`: aggregation functions
- `rep()`: repeat input
- `factor()`: create variable of categories
- `plot()`: scatterplot
- `plot(..., type = 'l')`: line plot
- `points()`: add points to plot
- `hist()`: histogram
- `boxplot()`: draw a box-wishkers plot

Section 5

Exercises

Exercises

Use of the `plot` function using terrestial ecology data:

1. In Chapter 16 of Zuur et al. (2009), a study is presented analysing numbers of amphibians killed along a road in Portugal using generalised additive mixed modelling techniques. In this exercise, we use the `plot` command to visualise a segment of the data. Open the file `Amphibian_road_Kills.xls`, prepare a spreadsheet, and import the data into R. Download: http://highstat.com/Books/Book3/MoreData.zip
2. The variable, `TOT_N`, is the number of dead animals at a sampling site, `OLIVE` is the number of olive groves at a sampling site, and `D Park` is the distance from each sampling point to the nearby natural park. Create a plot of `TOT_N` versus `D_park`. Use appropriate labels.

Alain Zuur, Elena N Ieno, and Erik Meesters. *A Beginner's Guide to R*. Springer Science & Business Media, 2009.