Towards visualizing data

Gerko Vink & Erik-Jan van Kesteren

Statistical Programming in R

This lecture

- Data manipulation
- Basic analysis (correlation & t-test)
- Pipes

We use the following packages

```
library(MASS)     # for the cats data
library(dplyr)    # data manipulation
library(haven)    # in/exporting data
library(magrittr) # pipes
library(ggplot2)  # Plotting device
```

loading packages

```
library(mice)     # missing data
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

Data manipulation

The cats data

```
head(cats)
```

```
##   Sex Bwt Hwt
## 1   F 2.0 7.0
## 2   F 2.0 7.4
## 3   F 2.0 9.5
## 4   F 2.1 7.2
## 5   F 2.1 7.3
## 6   F 2.1 7.6
```

```
str(cats)
```

```
## 'data.frame':    144 obs. of  3 variables:
##  $ Sex: Factor w/ 2 levels "F","M": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Bwt: num  2 2 2 2.1 2.1 2.1 2.1 2.1 2.1 2.1 ...
##  $ Hwt: num  7 7.4 9.5 7.2 7.3 7.6 8.1 8.2 8.3 8.5 ...
```

How to get only Female cats?

```
fem.cats <- cats[cats$Sex == "F", ]
dim(fem.cats)
```

```
## [1] 47  3
```

```
head(fem.cats)
```

```
##   Sex Bwt Hwt
## 1   F 2.0 7.0
## 2   F 2.0 7.4
## 3   F 2.0 9.5
## 4   F 2.1 7.2
## 5   F 2.1 7.3
## 6   F 2.1 7.6
```

How to get only *heavy* cats?

```
heavy.cats <- cats[cats$Bwt > 3, ]
dim(heavy.cats)
```

```
## [1] 36  3
```

```
head(heavy.cats)
```

```
##     Sex Bwt  Hwt
## 109   M 3.1  9.9
## 110   M 3.1 11.5
## 111   M 3.1 12.1
## 112   M 3.1 12.5
## 113   M 3.1 13.0
## 114   M 3.1 14.3
```

How to get only *heavy* cats?

```
heavy.cats <- subset(cats, Bwt > 3)
dim(heavy.cats)
```

```
## [1] 36  3
```

```
head(heavy.cats)
```

```
##     Sex Bwt  Hwt
## 109   M 3.1  9.9
## 110   M 3.1 11.5
## 111   M 3.1 12.1
## 112   M 3.1 12.5
## 113   M 3.1 13.0
## 114   M 3.1 14.3
```

more flexible: **dplyr**

```
filter(cats, Bwt > 2, Bwt < 2.2, Sex == "F")
```

```
##   Sex Bwt Hwt
## 1   F 2.1 7.2
## 2   F 2.1 7.3
## 3   F 2.1 7.6
## 4   F 2.1 8.1
## 5   F 2.1 8.2
## 6   F 2.1 8.3
## 7   F 2.1 8.5
## 8   F 2.1 8.7
## 9   F 2.1 9.8
```

Working with factors

```
class(cats$Sex)
```

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

```
levels(cats$Sex)
```

```
## [1] "F" "M"
```

Working with factors

```
levels(cats$Sex) <- c("Female", "Male")
table(cats$Sex)
```

```
##
## Female   Male
##     47     97
```

```
head(cats)
```

```
##      Sex Bwt Hwt
## 1 Female 2.0 7.0
## 2 Female 2.0 7.4
## 3 Female 2.0 9.5
## 4 Female 2.1 7.2
## 5 Female 2.1 7.3
## 6 Female 2.1 7.6
```

Releveling factors

```
lm(Hwt ~ Bwt + Sex, data = cats)
```

```
##
## Call:
## lm(formula = Hwt ~ Bwt + Sex, data = cats)
##
## Coefficients:
## (Intercept)          Bwt      SexMale
##     -0.4150       4.0758      -0.0821
```

```
cats$Sex <- relevel(cats$Sex, ref = "Male")
lm(Hwt ~ Bwt + Sex, data = cats)
```

```
##
## Call:
## lm(formula = Hwt ~ Bwt + Sex, data = cats)
##
## Coefficients:
## (Intercept)          Bwt    SexFemale
##     -0.4970       4.0758       0.0821
```

Sorting

```
order(cats$Bwt)
```

```
##   [1]   1   2   3  48  49   4   5   6   7   8   9  10  11  12  50  13  14  15
##  [19]  16  17  18  51  52  53  54  55  56  57  58  19  20  21  22  23  24  25
##  [37]  26  27  28  29  30  59  31  32  33  34  60  61  62  63  64  35  36  65
##  [55]  66  67  68  69  70  71  72  37  38  39  73  74  75  76  77  78  40  41
##  [73]  42  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  43
##  [91]  44  45  95  96  97  98  99  46  47 100 101 102 103 104 105 106 107 108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
```

```
sorted.cats <- cats[order(cats$Bwt), ]
head(sorted.cats)
```

```
##       Sex Bwt Hwt
## 1  Female 2.0 7.0
## 2  Female 2.0 7.4
## 3  Female 2.0 9.5
## 48   Male 2.0 6.5
## 49   Male 2.0 6.5
## 4  Female 2.1 7.2
```

Better sorting: `arrange()`

```
sorted.cats2 <- arrange(cats, Bwt)
head(sorted.cats)
```

```
##       Sex Bwt Hwt
## 1  Female 2.0 7.0
## 2  Female 2.0 7.4
## 3  Female 2.0 9.5
## 48   Male 2.0 6.5
## 49   Male 2.0 6.5
## 4  Female 2.1 7.2
```

```
head(sorted.cats2)
```

```
##      Sex Bwt Hwt
## 1 Female 2.0 7.0
## 2 Female 2.0 7.4
## 3 Female 2.0 9.5
## 4   Male 2.0 6.5
## 5   Male 2.0 6.5
## 6 Female 2.1 7.2
```

Better sorting: `arrange()`

```
cats.2 <- arrange(cats, Bwt, desc(Hwt))
head(cats.2, n = 10)
```

```
##       Sex Bwt  Hwt
## 1  Female 2.0  9.5
## 2  Female 2.0  7.4
## 3  Female 2.0  7.0
## 4    Male 2.0  6.5
## 5    Male 2.0  6.5
## 6    Male 2.1 10.1
## 7  Female 2.1  9.8
## 8  Female 2.1  8.7
## 9  Female 2.1  8.5
## 10 Female 2.1  8.3
```

Better sorting: `arrange()`

```
cats.2 <- arrange(cats, desc(Hwt), Bwt)
head(cats.2, n = 10)
```

```
##     Sex Bwt  Hwt
## 1  Male 3.9 20.5
## 2  Male 3.5 17.2
## 3  Male 3.8 16.8
## 4  Male 3.5 15.7
## 5  Male 3.5 15.6
## 6  Male 3.3 15.4
## 7  Male 3.6 15.0
## 8  Male 3.3 14.9
## 9  Male 3.6 14.8
## 10 Male 3.8 14.8
```

Basic analysis

Correlation

```
cor(cats[, -1])
```

```
##           Bwt       Hwt
## Bwt 1.0000000 0.8041274
## Hwt 0.8041274 1.0000000
```

With `[, -1]` we exclude the first column

Correlation

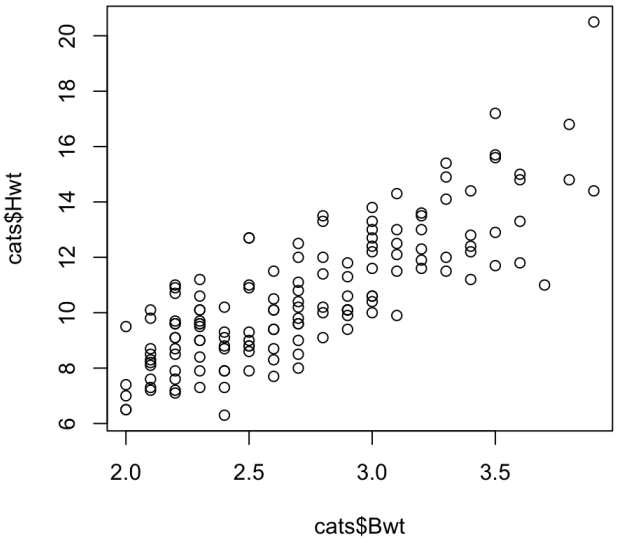
```
cor.test(cats$Bwt, cats$Hwt)
```

```
##
##  Pearson's product-moment correlation
##
## data:  cats$Bwt and cats$Hwt
## t = 16.119, df = 142, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7375682 0.8552122
## sample estimates:
##       cor
## 0.8041274
```

What do we conclude?

Correlation

```
plot(cats$Bwt, cats$Hwt)
```



T-test

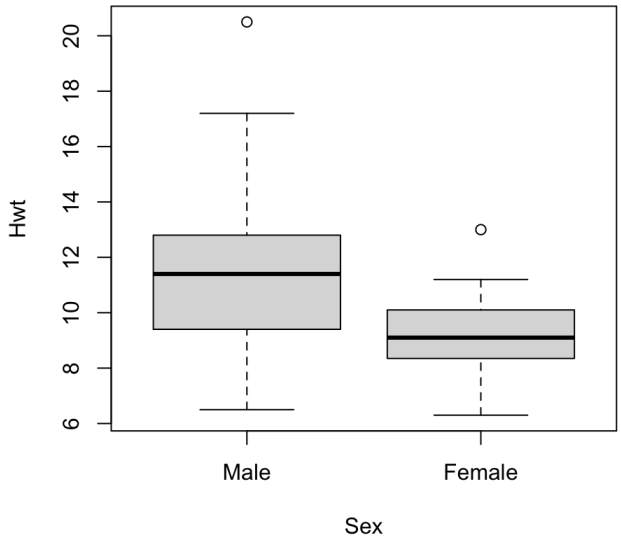
Test the null hypothesis that the difference in mean heart weight between male and female cats is 0

```
t.test(formula = Hwt ~ Sex, data = cats)
```

```
##
##  Welch Two Sample t-test
##
## data:  Hwt by Sex
## t = 6.5179, df = 140.61, p-value = 1.186e-09
## alternative hypothesis: true difference in means between group Male and group Female is not equal to 0
## 95 percent confidence interval:
##  1.477352 2.763753
## sample estimates:
##   mean in group Male mean in group Female
##            11.322680             9.202128
```

T-test

```
plot(formula = Hwt ~ Sex, data = cats)
```



Pipes

This is a pipe:

```
boys <-
  read_sav("boys.sav") %>%
  head()
```

It effectively replaces `head(read_sav("boys.sav"))`.

Why are pipes useful?

Let's assume that we want to load data, change a variable, filter cases and select columns. Without a pipe, this would look like

```
boys  <- read_sav("boys.sav")
boys2 <- transform(boys, hgt = hgt / 100)
boys3 <- filter(boys2, age > 15)
boys4 <- subset(boys3, select = c(hgt, wgt, bmi))
```

With the pipe:

```
boys <-
  read_sav("boys.sav") %>%
  transform(hgt = hgt/100) %>%
  filter(age > 15) %>%
  subset(select = c(hgt, wgt, bmi))
```

Benefit: a single object in memory that is easy to interpret

With pipes

Your code becomes more readable:

- data operations are structured from left-to-right and not from in-to-out
- nested function calls are avoided
- local variables and copied objects are avoided
- easy to add steps in the sequence

The boys data

```
boys %>% head()
```

```
##     hgt  wgt   bmi
## 1 1.880 91.6 25.91
## 2 1.806 58.0 17.78
## 3 1.855 62.7 18.22
## 4 1.785 54.1 16.97
## 5 1.725 64.0 21.50
## 6 1.781 74.5 23.48
```

What do pipes do:

- `f(x)` becomes `x %>% f()`

```
rnorm(10) %>% mean()
```

```
## [1] 0.5604495
```

always as the 1st argument! :)

- `f(x, y)` becomes `x %>% f(y)`

```
boys %>% cor(use = "pairwise.complete.obs")
```

```
##           hgt       wgt       bmi
## hgt 1.0000000 0.6100784 0.1758781
## wgt 0.6100784 1.0000000 0.8841304
## bmi 0.1758781 0.8841304 1.0000000
```

- `h(g(f(x)))` becomes `x %>% f %>% g %>% h`

```
boys %>% subset(select = wgt) %>% na.omit() %>% max()
```

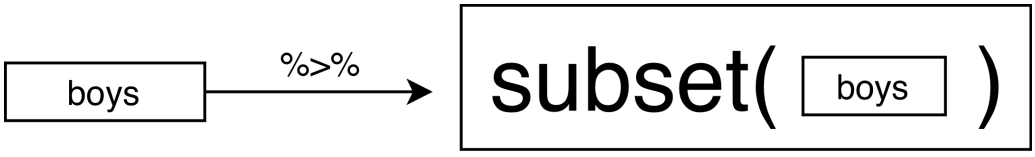
```
## [1] 117.4
```

Don't use it!!
Don't remove every row w/ NA!

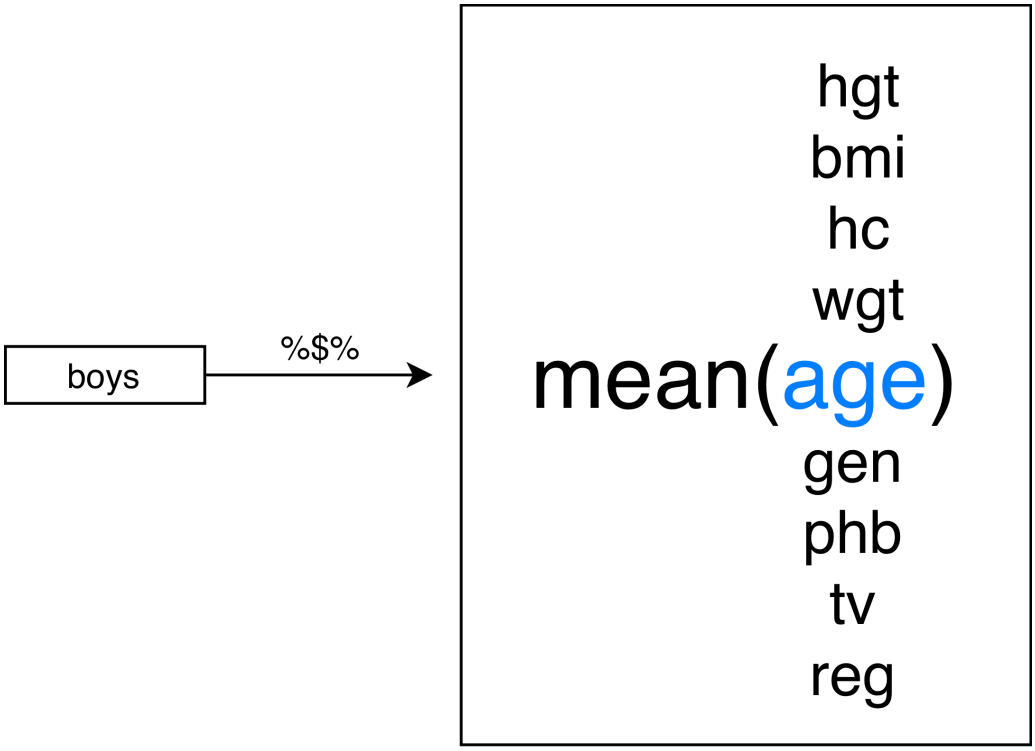
More pipe stuff

new pipe: |>

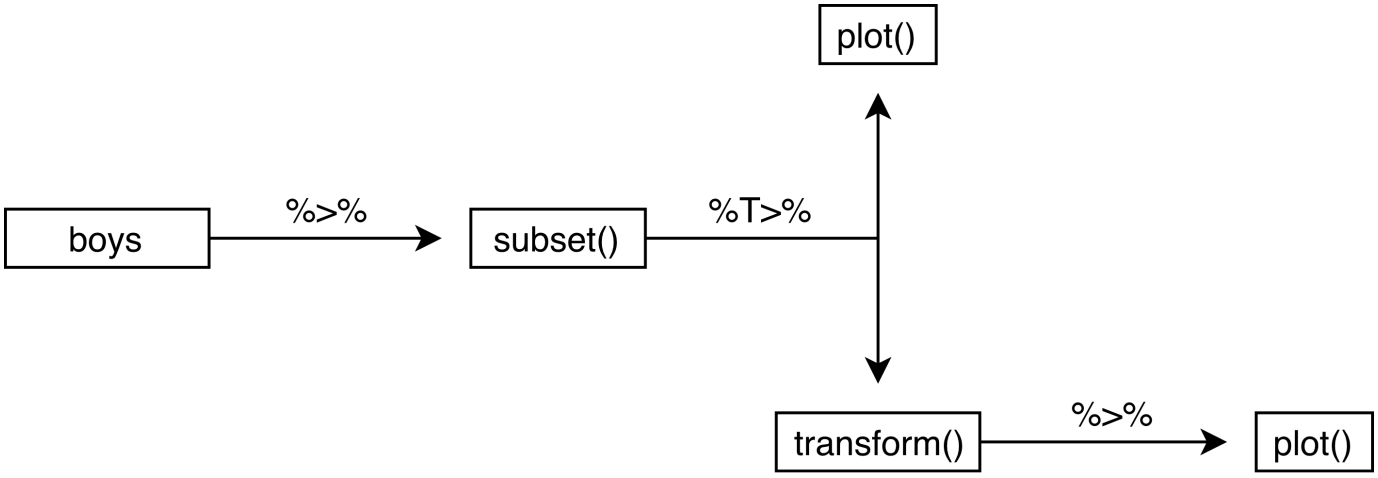
The standard %>% pipe



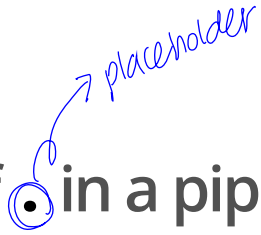
The %$% pipe



The %T>% pipe



The role of . in a pipe



In `a %>% b(arg1, arg2, arg3)`, `a` will become `arg1`. With `.` we can change this.

```
set.seed(123)
1:5 %>%
  mean() %>%
  rnorm(10)
```

```
## [1]  9.439524  9.769823 11.558708
```

VS

```
set.seed(123)
1:5 %>%
  mean() %>%
  rnorm(n = 10, mean = .)
```

```
##  [1] 2.439524 2.769823 4.558708 3.070508 3.129288 4.715065 3.460916 1.734939
##  [9] 2.313147 2.554338
```

The `.` can be used as a placeholder in the pipe.

Placeholder example

Remember: `sample()` takes a random sample from a vector

```
sample(x = c(1, 1, 2, 3, 5, 8), size = 2)
```

```
## [1] 1 3
```

Sample 3 positions from the alphabet and show the position and the letter

```
set.seed(123)
1:26 %>%
  sample(3) %>%
  paste(., LETTERS[.])
```

```
## [1] "15 O" "19 S" "14 N"
```

Debugging pipelines

If you don't know what's going on, run each statement separately!

```
set.seed(123)
1:26
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## [26] 26
```

```
set.seed(123)
1:26 %>%
  sample(3)
```

```
## [1] 15 19 14
```

```
set.seed(123)
1:26 %>%
  sample(3) %>%
  paste(., LETTERS[.])
```

```
## [1] "15 O" "19 S" "14 N"
```

Performing a t-test in a pipe

```
cats %$%
  t.test(Hwt ~ Sex)
```

you don't need "data=."

```
##
##  Welch Two Sample t-test
##
## data:  Hwt by Sex
## t = 6.5179, df = 140.61, p-value = 1.186e-09
## alternative hypothesis: true difference in means between group Male and group Female is not equal to 0
## 95 percent confidence interval:
##  1.477352 2.763753
## sample estimates:
##   mean in group Male mean in group Female
##            11.322680             9.202128
```

is the same as

```
t.test(Hwt ~ Sex, data = cats)
```

Storing a t-test from a pipe

```
cats.test <-
  cats %$%
  t.test(Bwt ~ Sex)

cats.test
```

```
##
##  Welch Two Sample t-test
##
## data:  Bwt by Sex
## t = 8.7095, df = 136.84, p-value = 8.831e-15
## alternative hypothesis: true difference in means between group Male and group Female is not equal to 0
## 95 percent confidence interval:
##  0.4177242 0.6631268
## sample estimates:
##   mean in group Male mean in group Female
##             2.900000             2.359574
```

Overwriting with a pipe

directly assigns the outcome

```
boys %<>%
  arrange(desc(bmi))
head(boys)
```

```
##     hgt   wgt   bmi
## 1 1.923 117.4 31.74
## 2 1.740  94.9 31.34
## 3 1.825 102.0 30.62
## 4 1.943 113.0 29.93
## 5 1.809  94.4 28.84
## 6 1.808  93.8 28.69
```

Beware

Be aware of the different interpretation of "."

```
boys %$%
  lm(bmi ~ .*., data = .)
```

```
##
## Call:
## lm(formula = bmi ~ . * ., data = .)
##
## Coefficients:
## (Intercept)          hgt          wgt      hgt:wgt
##     11.5686      -6.3493       0.7470      -0.2442
```

Visualization

Plotting

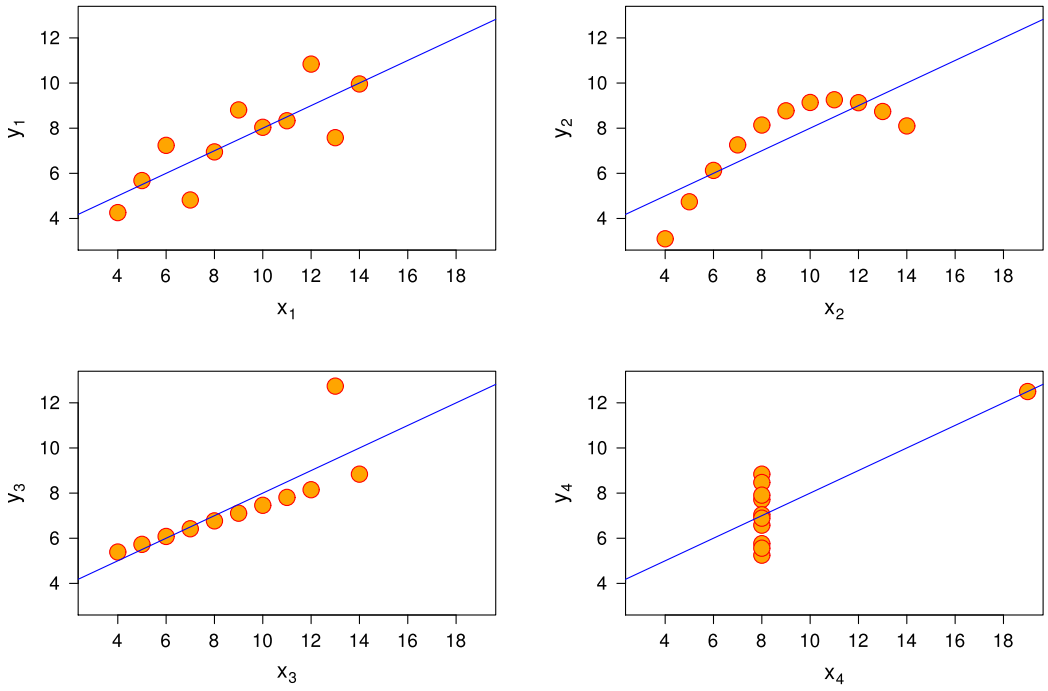
- `hist()`: histogram
- `plot()`: R's plotting device
- `barplot()`: bar plot function
- `boxplot()`: box plot function
- `density()`: function that calculates the density
- `ggplot()`: ggplot's plotting device

Why visualise?

- We can process a lot of information quickly with our eyes
- Plots give us information about
  - Distribution / shape
  - Irregularities
  - Assumptions
  - Intuitions
- Summary statistics, correlations, parameters, model tests, *p*-values do not tell the whole story

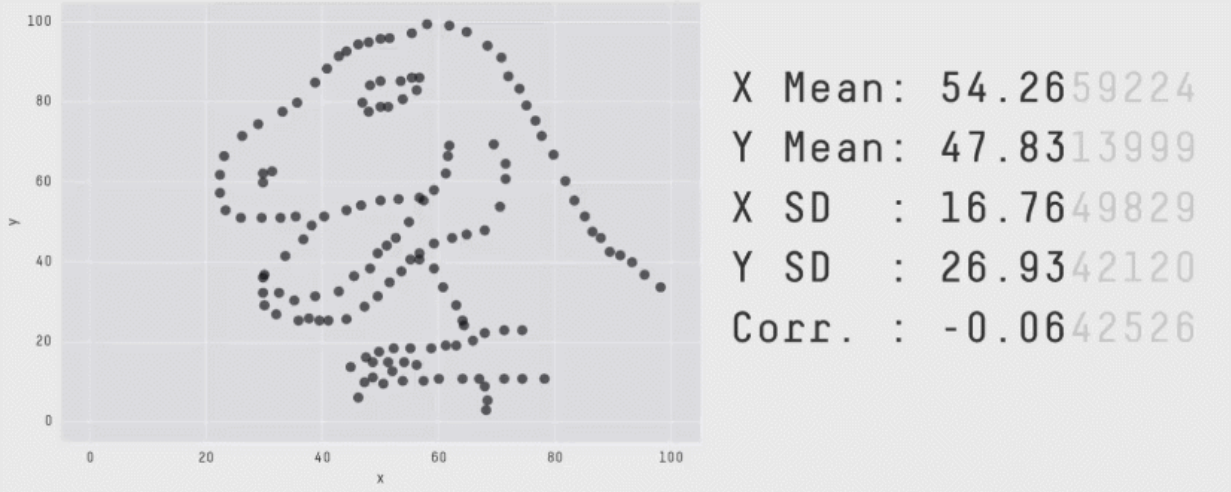
ALWAYS plot your data!

Why visualise?



Source: Anscombe, F. J. (1973). "Graphs in Statistical Analysis". American Statistician. 27 (1): 17–21.

Why visualise?

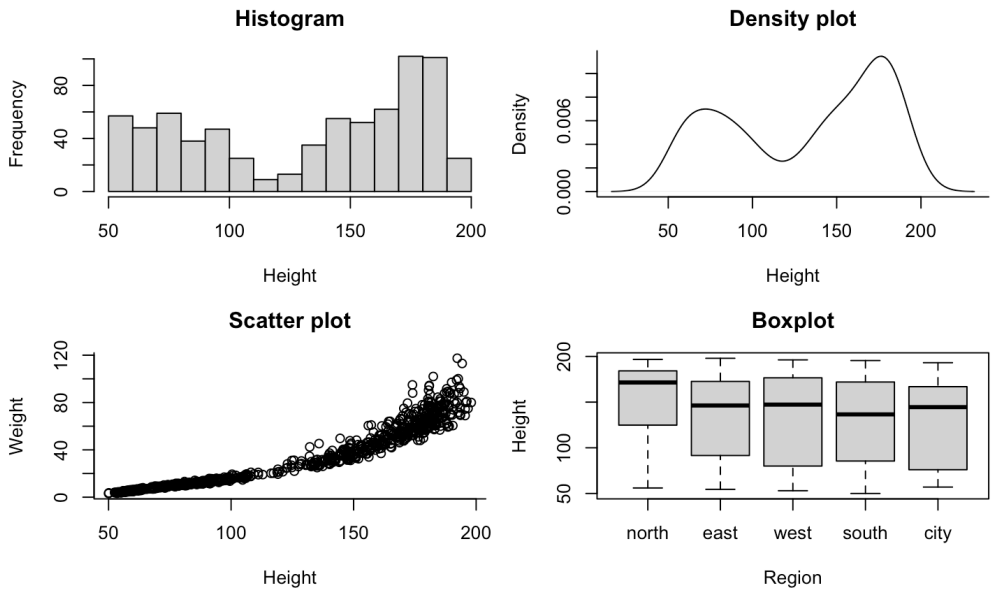


Source: https://www.autodeskresearch.com/publications/samestats

What we will do

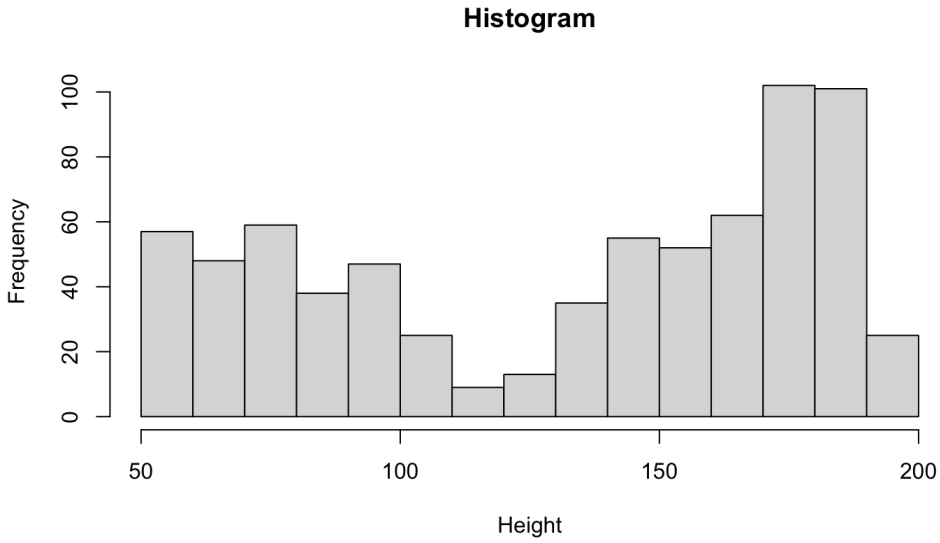
- A few plots in `base` graphics in `R`
- Plotting with `ggplot2` graphics

Plots


Histogram
Frequency
0
40
80
50
100
150
200
Height
Density plot
Density
0.000
0.006
50
100
150
200
Height
Scatter plot
Weight
0
40
80
120
50
100
150
200
Height
Boxplot
Height
50
100
200
north
east
west
south
city
Region

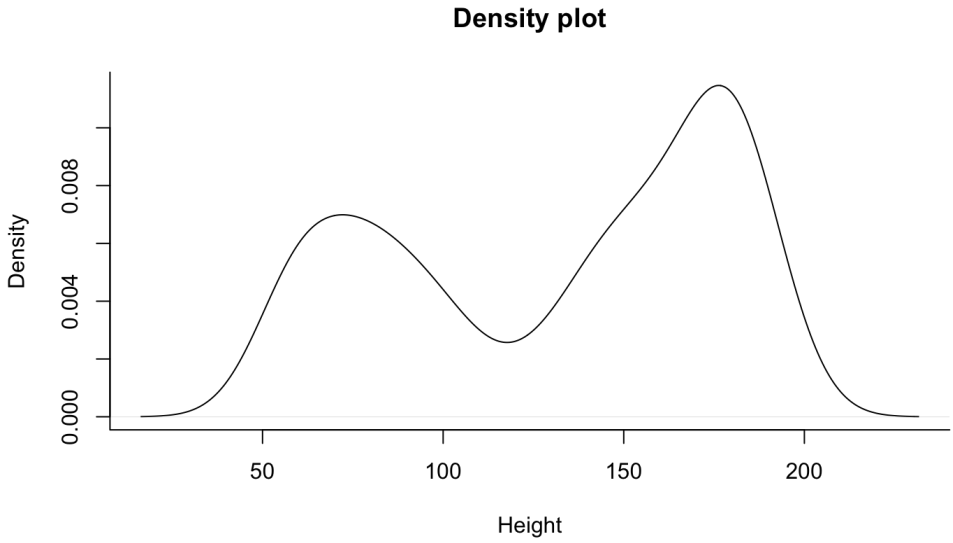
Histogram

```
hist(boys$hgt, main = "Histogram", xlab = "Height")
```



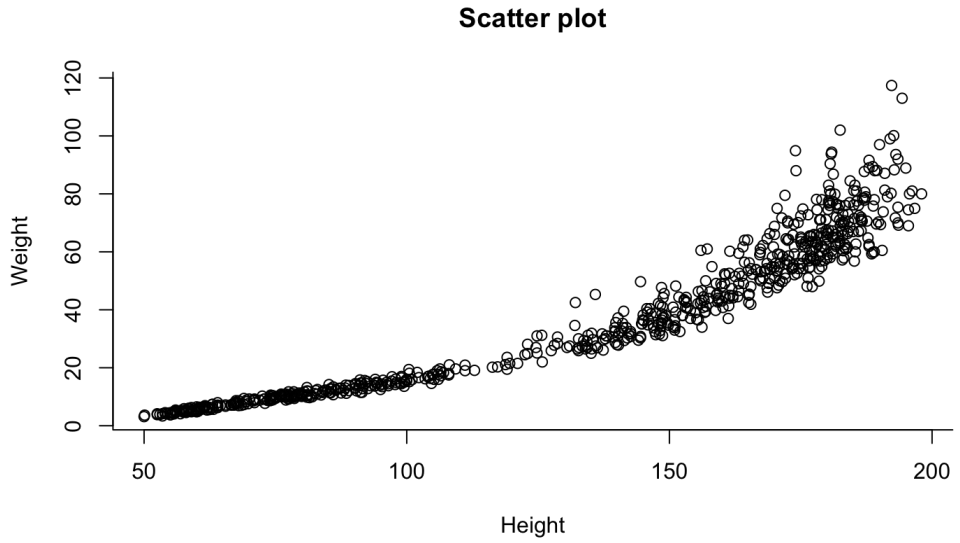
Density

```
dens <- density(boys$hgt, na.rm = TRUE)
plot(dens, main = "Density plot", xlab = "Height", bty = "L")
```



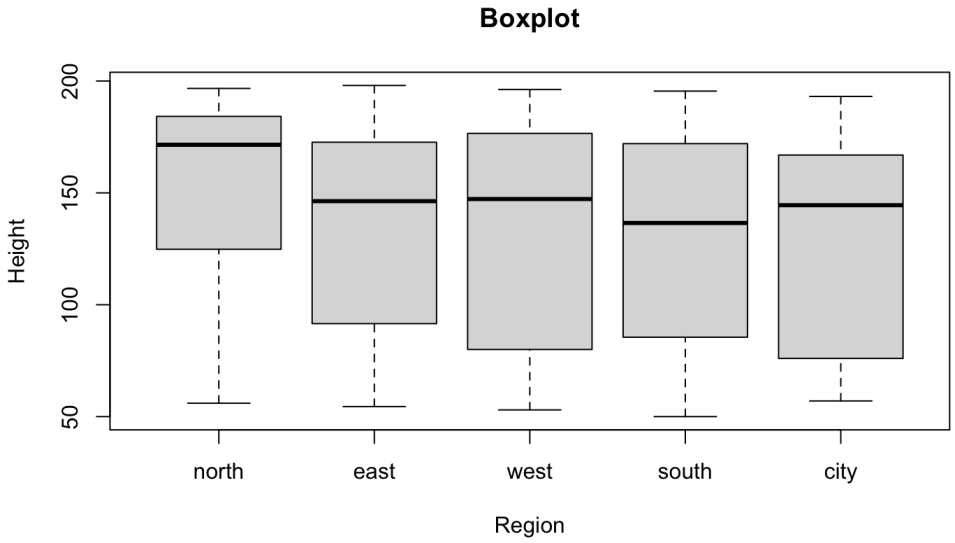
Scatter plot

```
plot(x = boys$hgt, y = boys$wgt, main = "Scatter plot",
     xlab = "Height", ylab = "Weight", bty = "L")
```



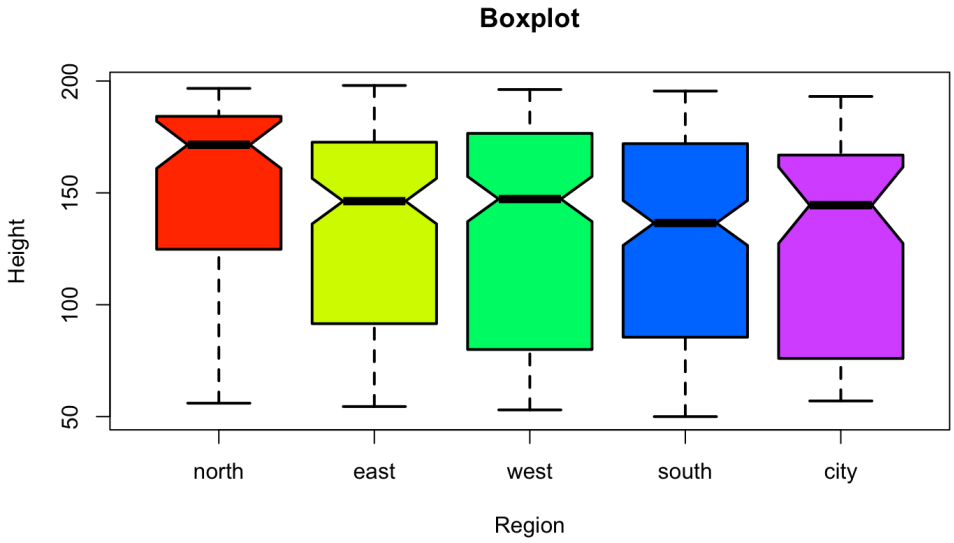
Box plot

```
boxplot(boys$hgt ~ boys$reg, main = "Boxplot",
        xlab = "Region", ylab = "Height")
```



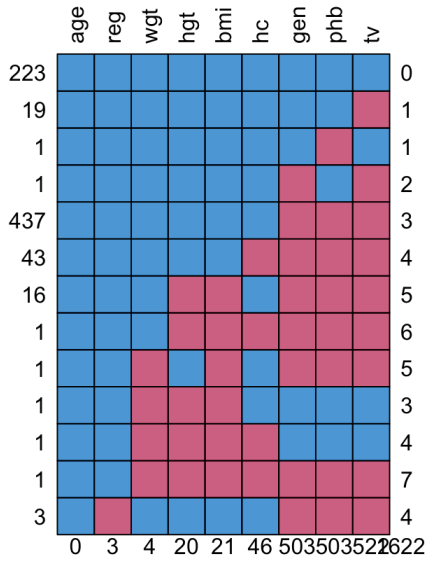
Box plot II

```
boxplot(hgt ~ reg, boys,  main = "Boxplot", xlab = "Region",
        ylab = "Height", lwd = 2, notch = TRUE, col = rainbow(5))
```



A lot can be done in base R!

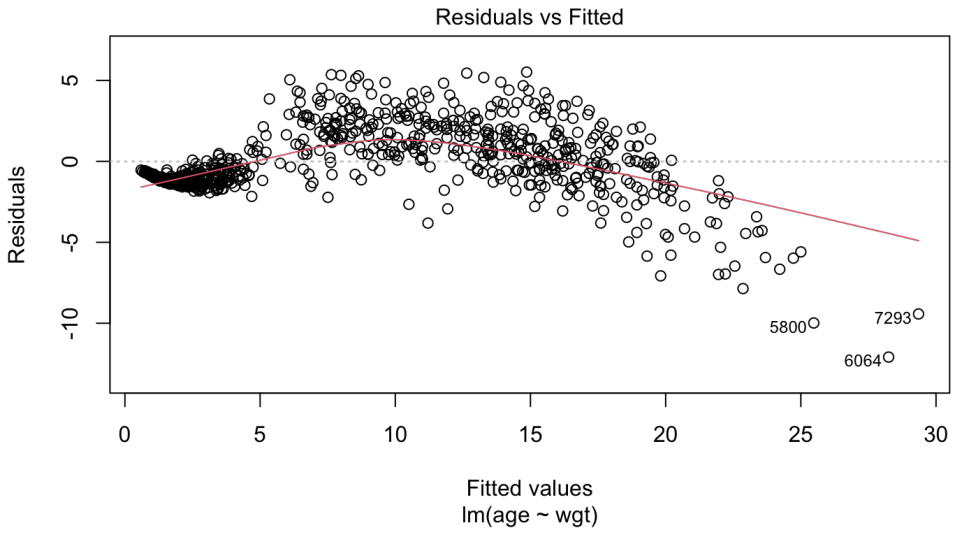
```
boys %>% md.pattern(rotate.names = TRUE) # from mice
```




```
##     age reg wgt hgt bmi hc gen phb  tv
## 223   1   1   1   1   1  1   1   1   1    0
## 19    1   1   1   1   1  1   1   1   0    1
## 1     1   1   1   1   1  1   1   0   1    1
## 1     1   1   1   1   1  1   0   1   0    2
## 437   1   1   1   1   1  1   0   0   0    3
## 43    1   1   1   1   1  0   0   0   0    4
## 16    1   1   1   0   0  1   0   0   0    5
## 1     1   1   1   0   0  0   0   0   0    6
## 1     1   1   0   1   0  1   0   0   0    5
## 1     1   1   0   0   0  1   1   1   1    3
## 1     1   1   0   0   0  0   1   1   1    4
## 1     1   1   0   0   0  0   0   0   0    7
## 3     1   0   1   1   1  1   0   0   0    4
##       0   3   4  20  21 46 503 503 522 1622
```

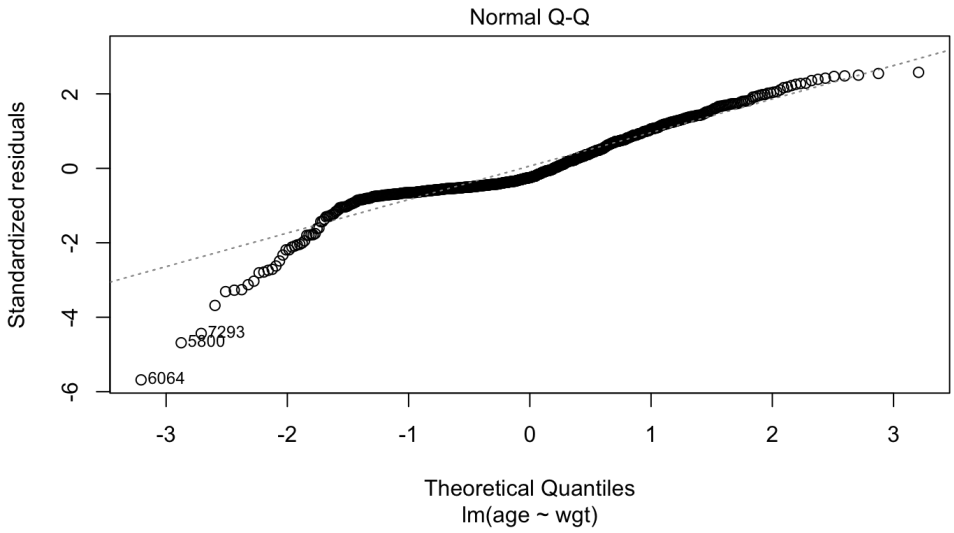
Many R classes have a `plot()` method

```
result <- lm(age~wgt, boys)
plot(result, which = 1)
```



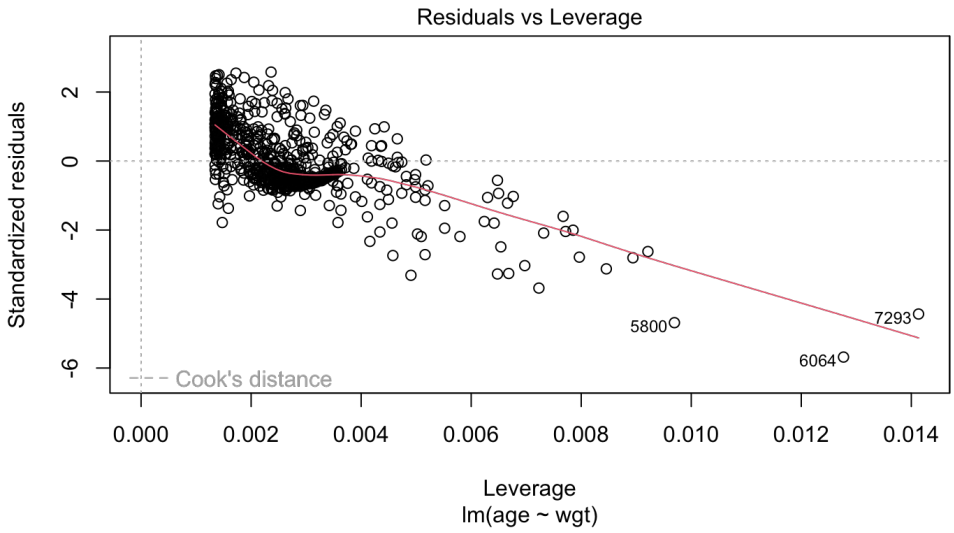
Many R classes have a `plot()` method

```
result <- lm(age~wgt, boys)
plot(result, which = 2)
```



Many R classes have a `plot()` method

```
result <- lm(age~wgt, boys)
plot(result, which = 5)
```



Neat! But what if we want more control?

ggplot2

What is `ggplot2`?

Layered plotting based on the book **The Grammer of Graphics** by Leland Wilkinsons.

With `ggplot2` you

1. provide the *data*
2. define how to map variables to *aesthetics*
3. state which *geometric object* to display
4. (optional) edit the overall *theme* of the plot

`ggplot2` then takes care of the details

An example: scatterplot

1: Provide the data

```
boys %>%
  ggplot()
```

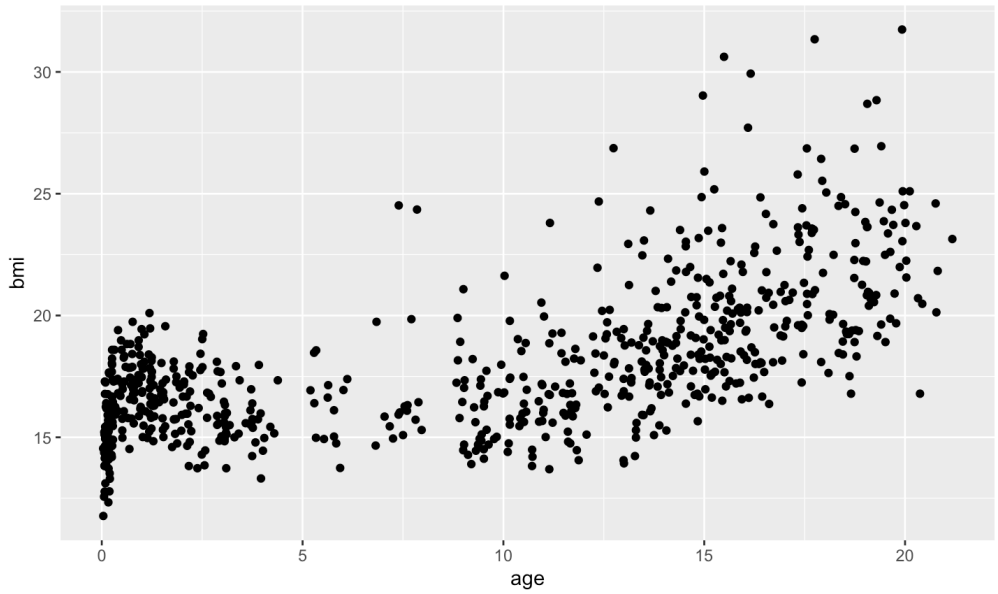
2: map variable to aesthetics

```
boys %>%
  ggplot(aes(x = age, y = bmi))
```

3: state which geometric object to display

```
boys %>%
  ggplot(aes(x = age, y = bmi)) +
  geom_point()
```

An example: scatterplot



Why this syntax?

Create the plot

```
gg <-
  boys %>%
  ggplot(aes(x = age, y = bmi)) +
  geom_point(col = "dark green")
```

Add another layer (smooth fit line)

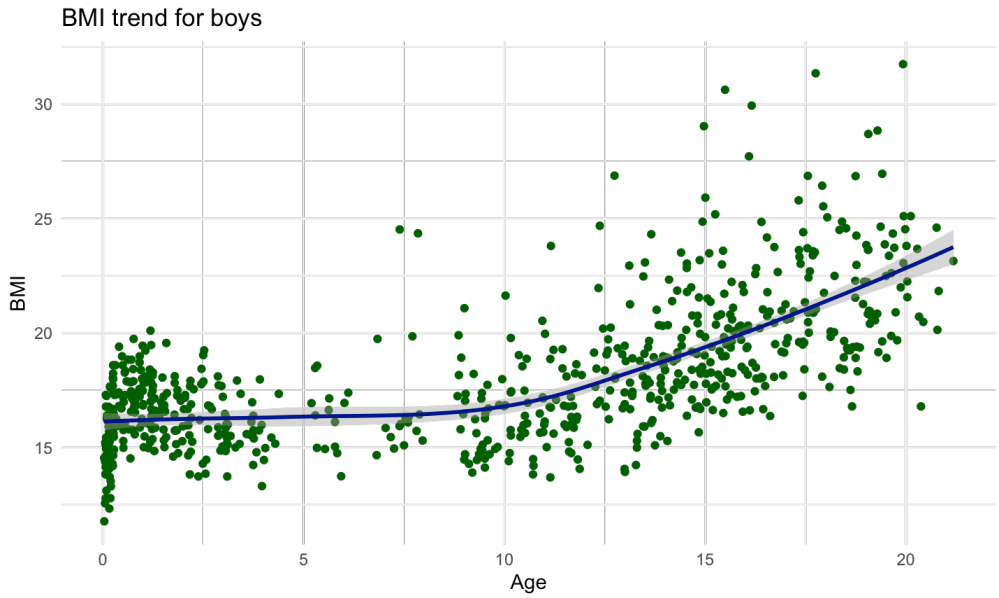
```
gg <- gg +
  geom_smooth(col = "dark blue")
```

Give it some labels and a nice look

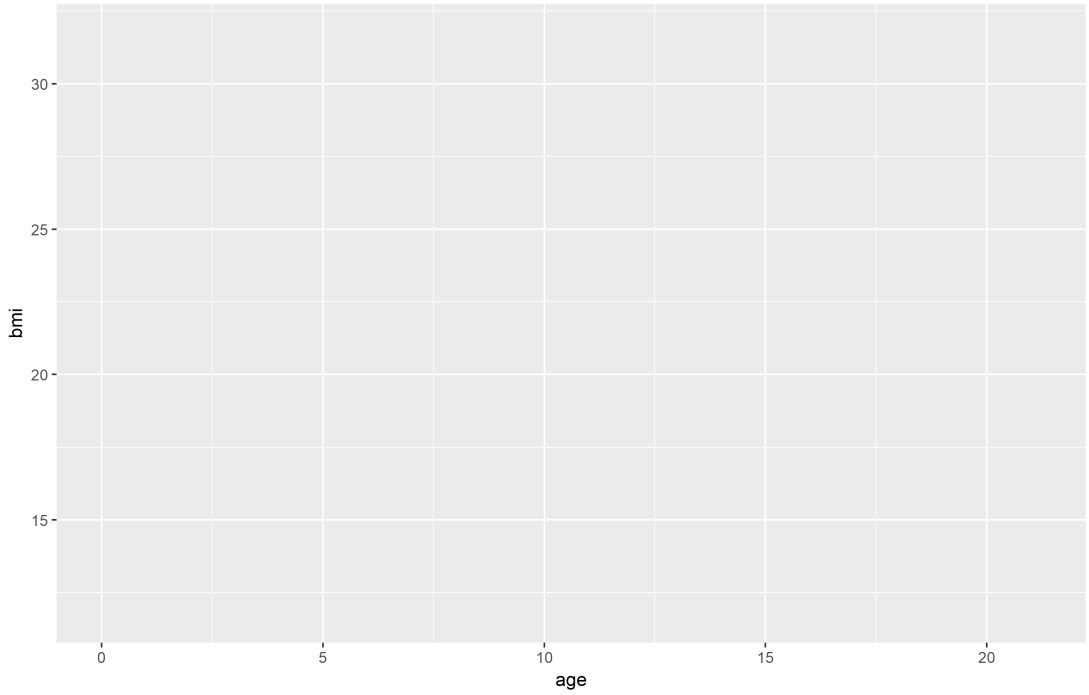
```
gg <- gg +
  labs(x = "Age", y = "BMI", title = "BMI trend for boys") +
  theme_minimal()
```

Why this syntax?

```
plot(gg)
```



Why this syntax?



Aesthetics

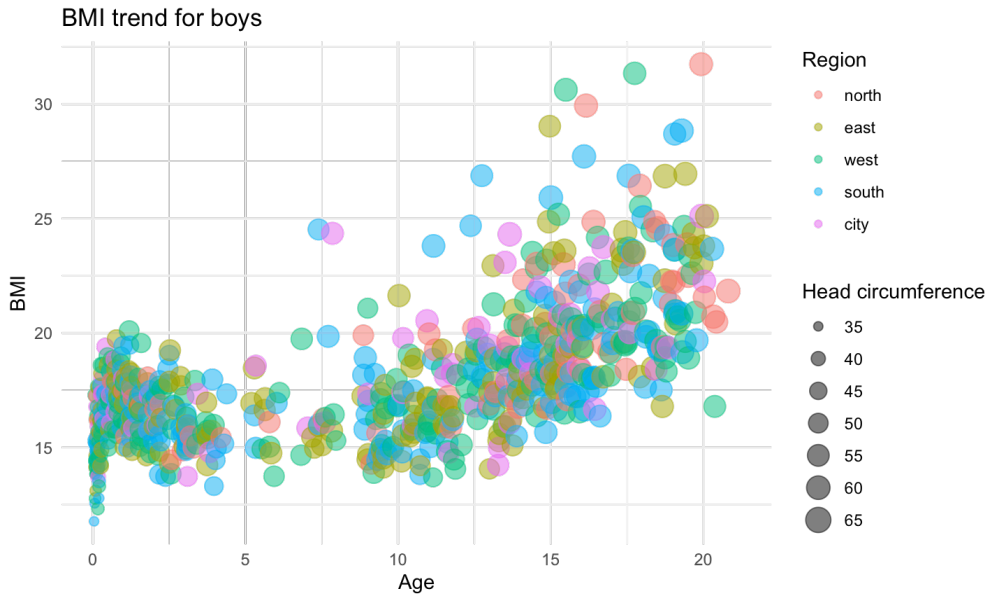
- x
- y
- size
- colour
- fill
- opacity (alpha)
- linetype
- …

Aesthetics

```
gg <-
  boys %>%
  filter(!is.na(reg)) %>%

  ggplot(aes(x      = age,
             y      = bmi,
             size   = hc,
             colour = reg)) +

  geom_point(alpha = 0.5) +

  labs(title  = "BMI trend for boys",
       x      = "Age",
       y      = "BMI",
       size   = "Head circumference",
       colour = "Region") +
  theme_minimal()
```

Aesthetics

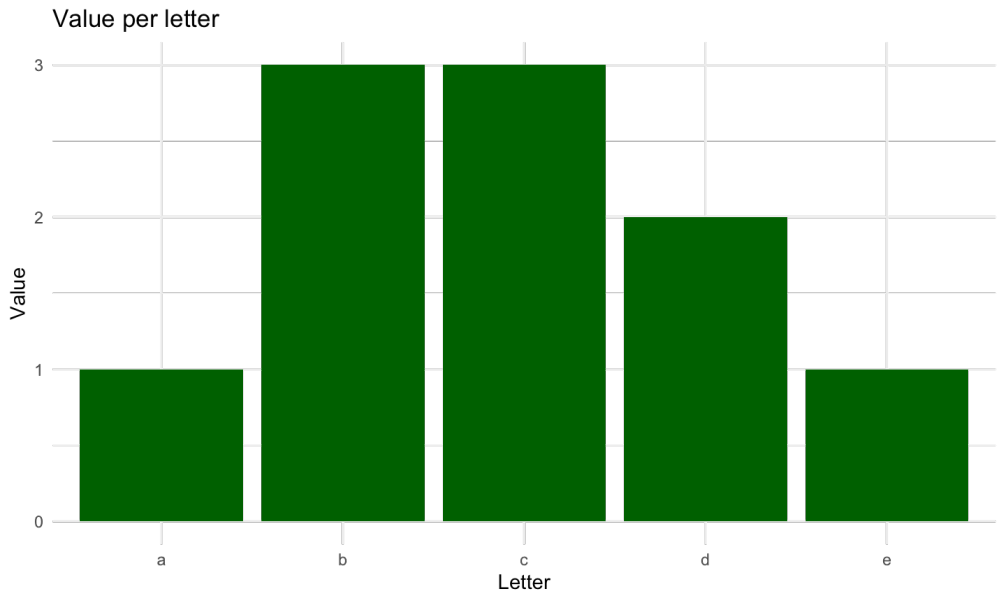
```
plot(gg)
```



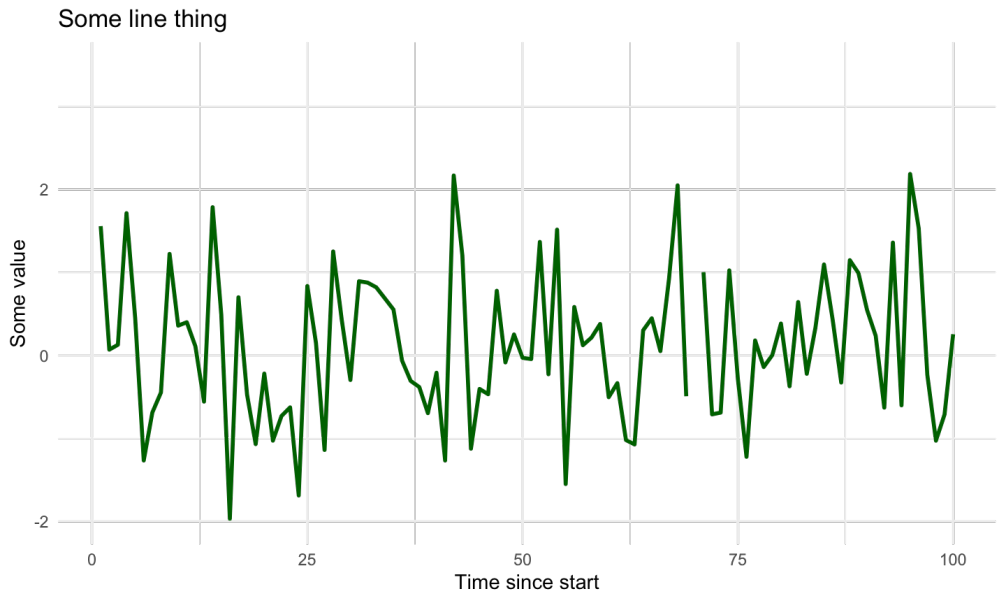
Geoms

- geom_point
- geom_bar
- geom_line
- geom_smooth
- geom_histogram
- geom_boxplot
- geom_density

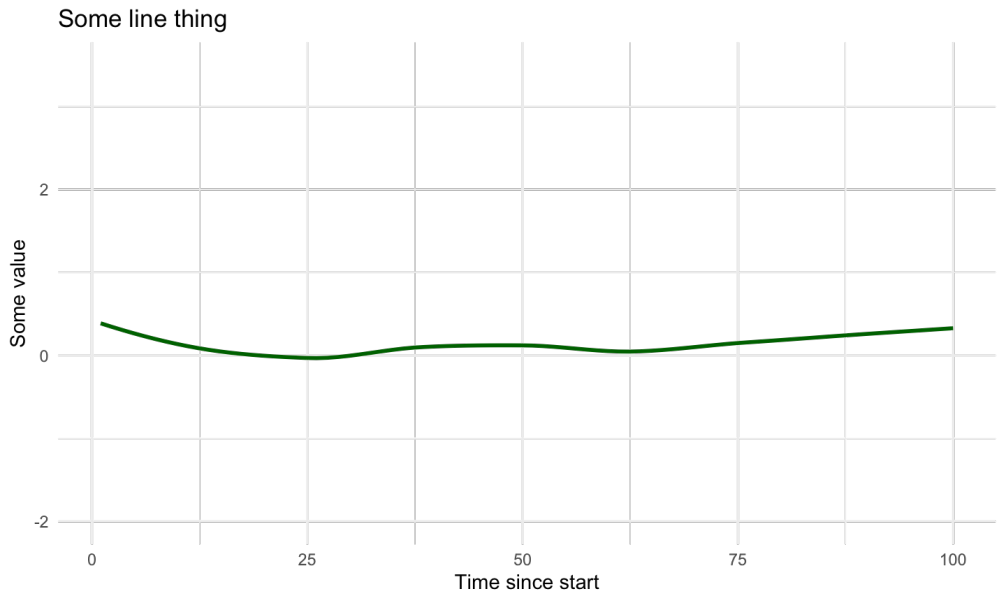
Geoms: Bar



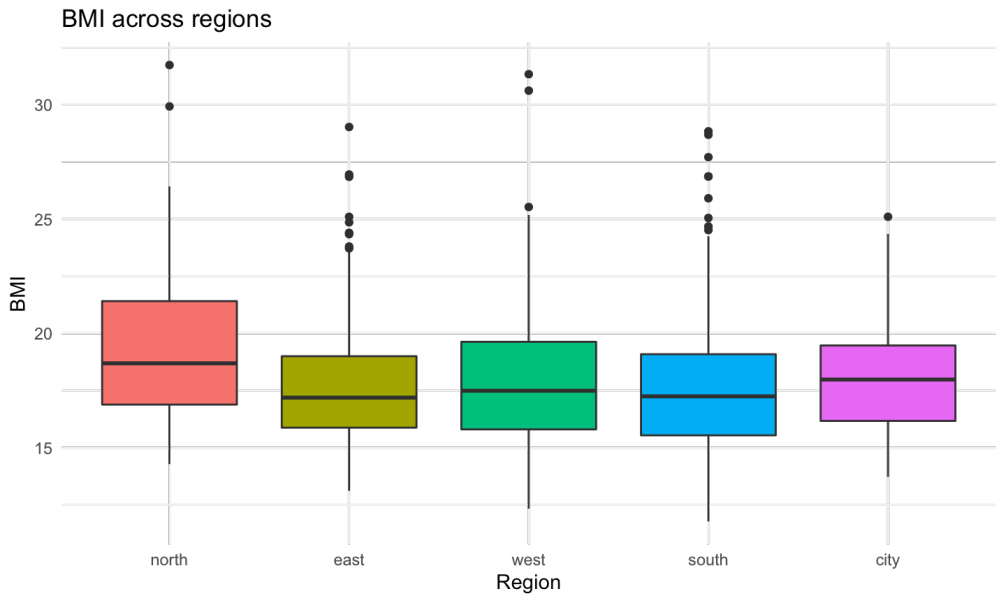
Geoms: Line



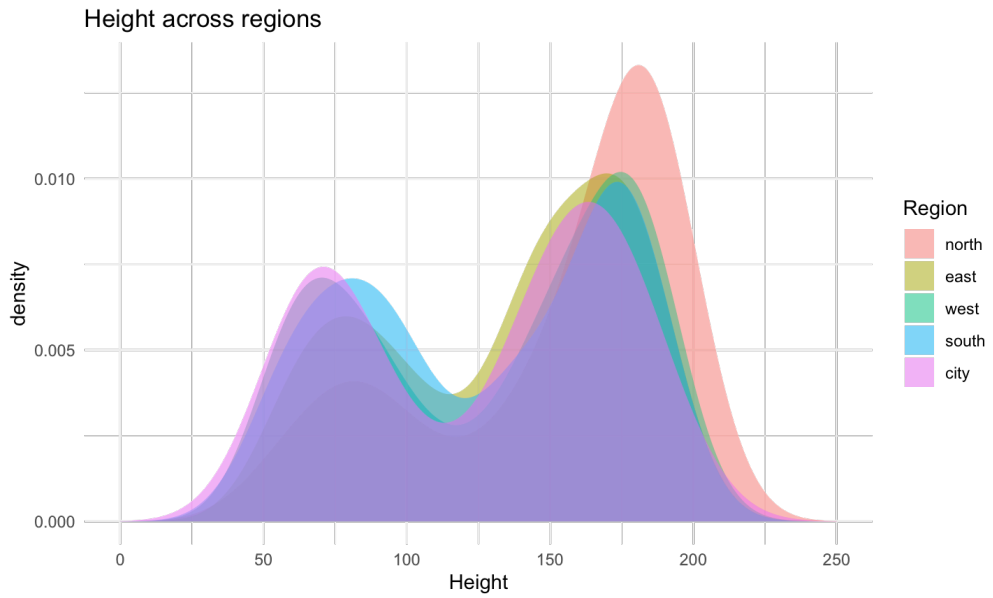
Geoms: Smooth



Geoms: Boxplot



Geoms: Density



Helpful link in RStudio