# Assumptions and inference

Gerko Vink

Fundamental Techniques in Data Science with R

# Last week

Last week we started with linear modeling, but time did not permit us to finish the lecture.
Today we start with the leftover part of last-week's lecture.

# Today

- Assumptions
- Generating data
- Statistical inference

# Packages and functions that we use

```
library(mvtnorm)  # Multivariate Normal tools
library(dplyr)
library(magrittr)
library(ggplot2)
library(mice)
```

- `set.seed()`: Fixing the Random Number Generator seed

# Assumptions

# The key assumptions

There are four key assumptions about the use of linear regression models.

In short, we assume

- The outcome to have a **linear relation** with the predictors and the predictor relations to be **additive**.
  - the expected value for the outcome is a straight-line function of each predictor, given that the others are fixed.
  - the slope of each line does not depend on the values of the other predictors
  - the effects of the predictors on the expected value are additive

$$y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

- The residuals are statistically **independent**
- The residual **variance is constant**
  - accross the expected values
  - across any of the predictors
- The residuals are **normally distributed** with mean $\mu_\epsilon = 0$

# A simple example

$$y = \alpha + \beta X + \epsilon$$

and

$$\hat{y} = \alpha + \beta X$$

As a result, the following components in linear modeling

- outcome $y$
- predicted outcome $\hat{y}$
- predictor $X$
- residual $\epsilon$

can also be seen as columns in a data set.

# As a data set

As a data set, this would be the result for `lm(y1 ~ x1, data = anscombe)`

Show 10 entries

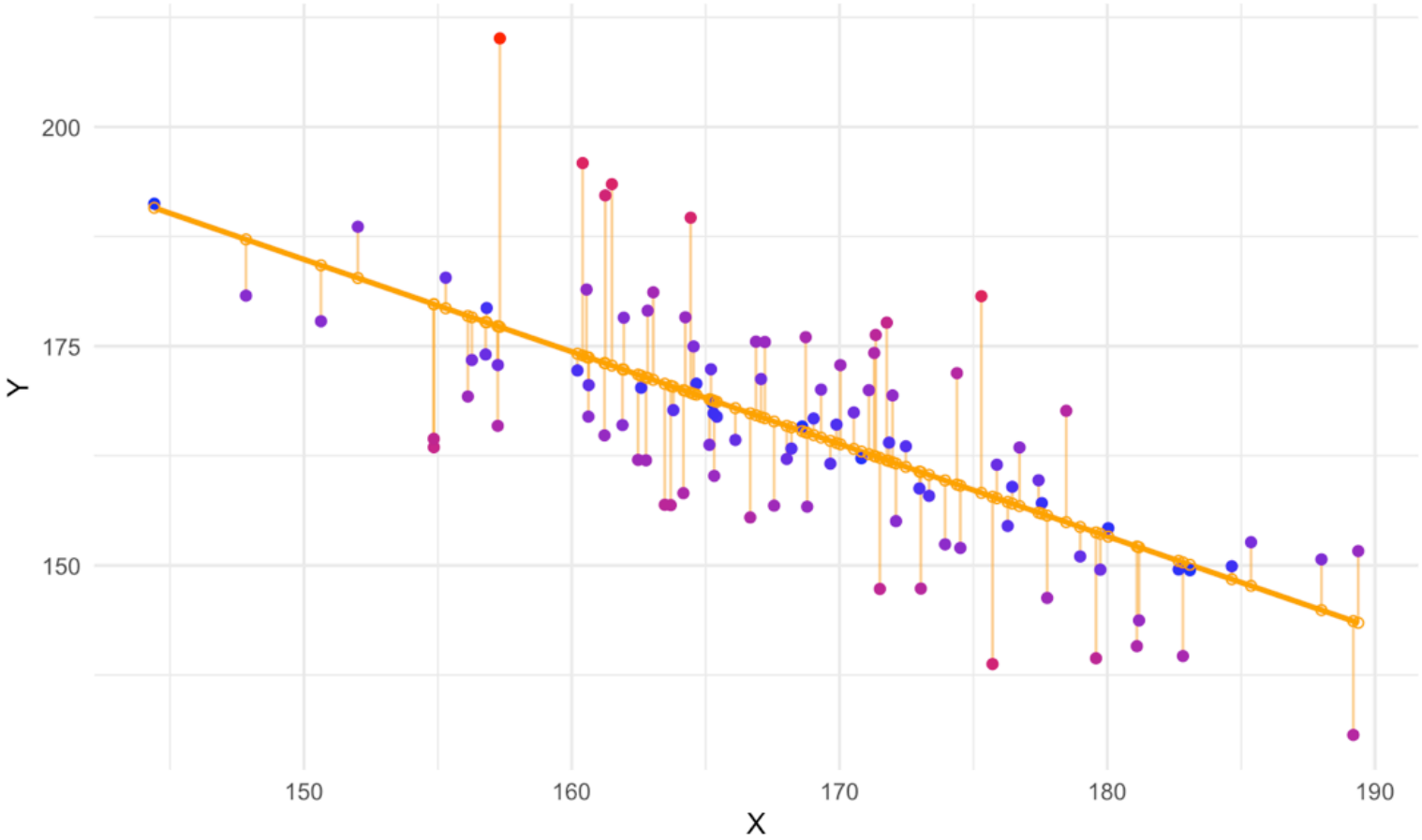
Search:

| | y1 | y1.hat | residuals | x1 |
|---|---|---|---|---|
| 1 | 8.04 | 8.001 | 0.038999999999999 | 10 |
| 2 | 6.95 | 7.00081818181818 | -0.0508181818181821 | 8 |
| 3 | 7.58 | 9.50127272727273 | -1.92127272727273 | 13 |
| 4 | 8.81 | 7.50090909090909 | 1.30909090909091 | 9 |
| 5 | 8.33 | 8.50109090909091 | -0.171090909090909 | 11 |
| 6 | 9.96 | 10.0013636363636 | -0.0413636363636357 | 14 |
| 7 | 7.24 | 6.00063636363636 | 1.23936363636364 | 6 |
| 8 | 4.26 | 5.00045454545455 | -0.740454545454545 | 4 |
| 9 | 10.84 | 9.00118181818182 | 1.83881818181818 | 12 |
| 10 | 4.82 | 6.50072727272727 | -1.68072727272727 | 7 |

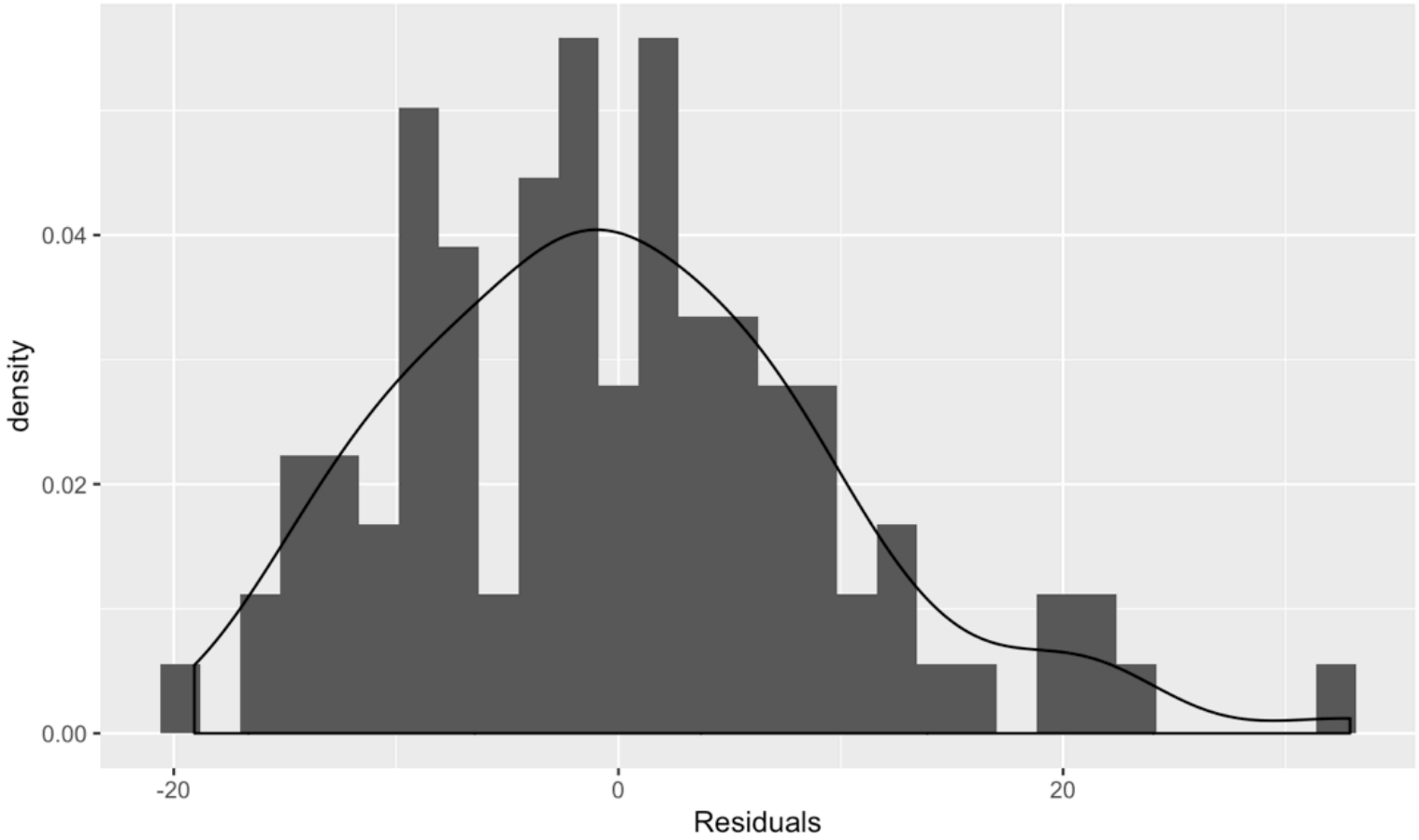
Showing 1 to 10 of 11 entries

Previous 1 2 Next

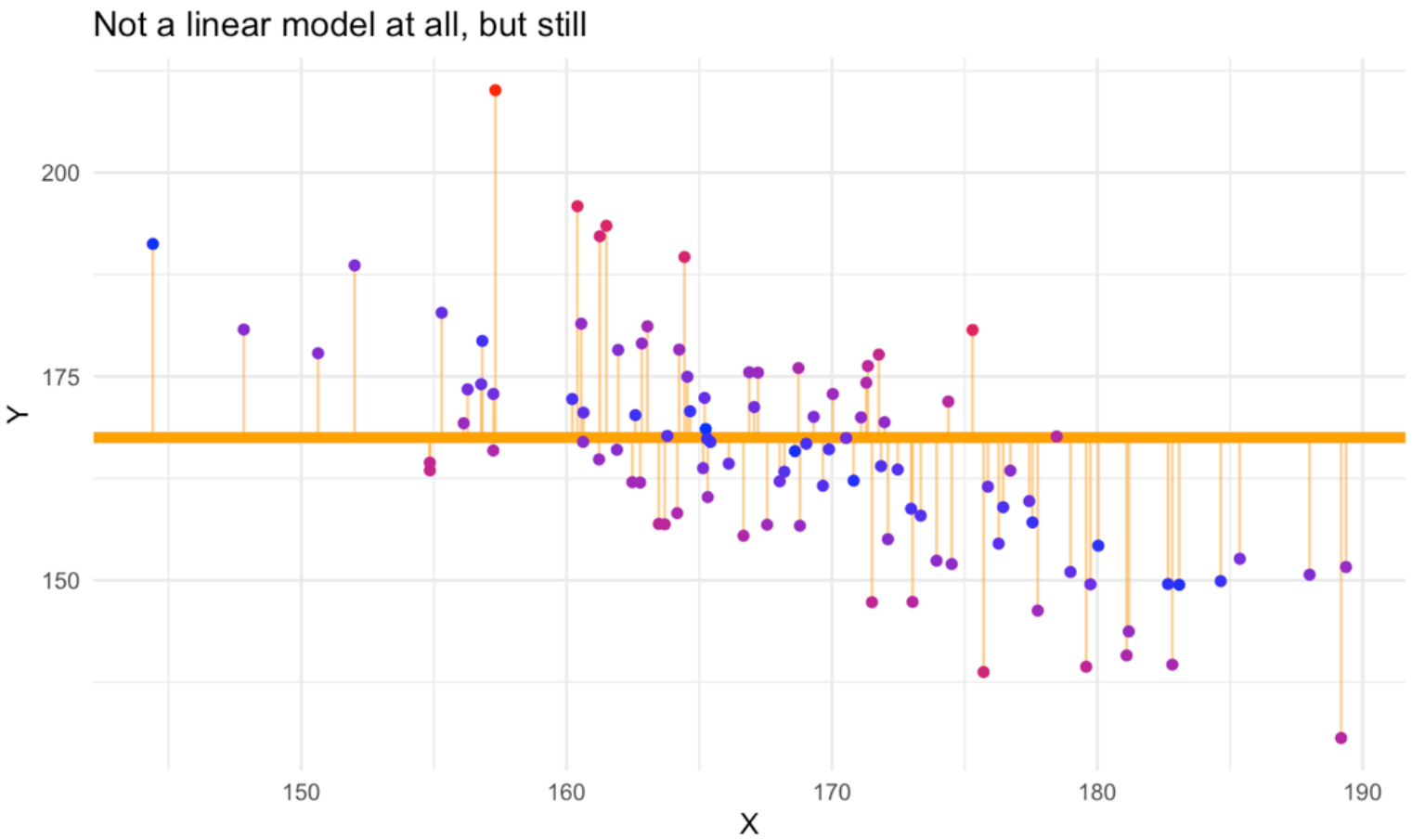
# An example



# Residual variance

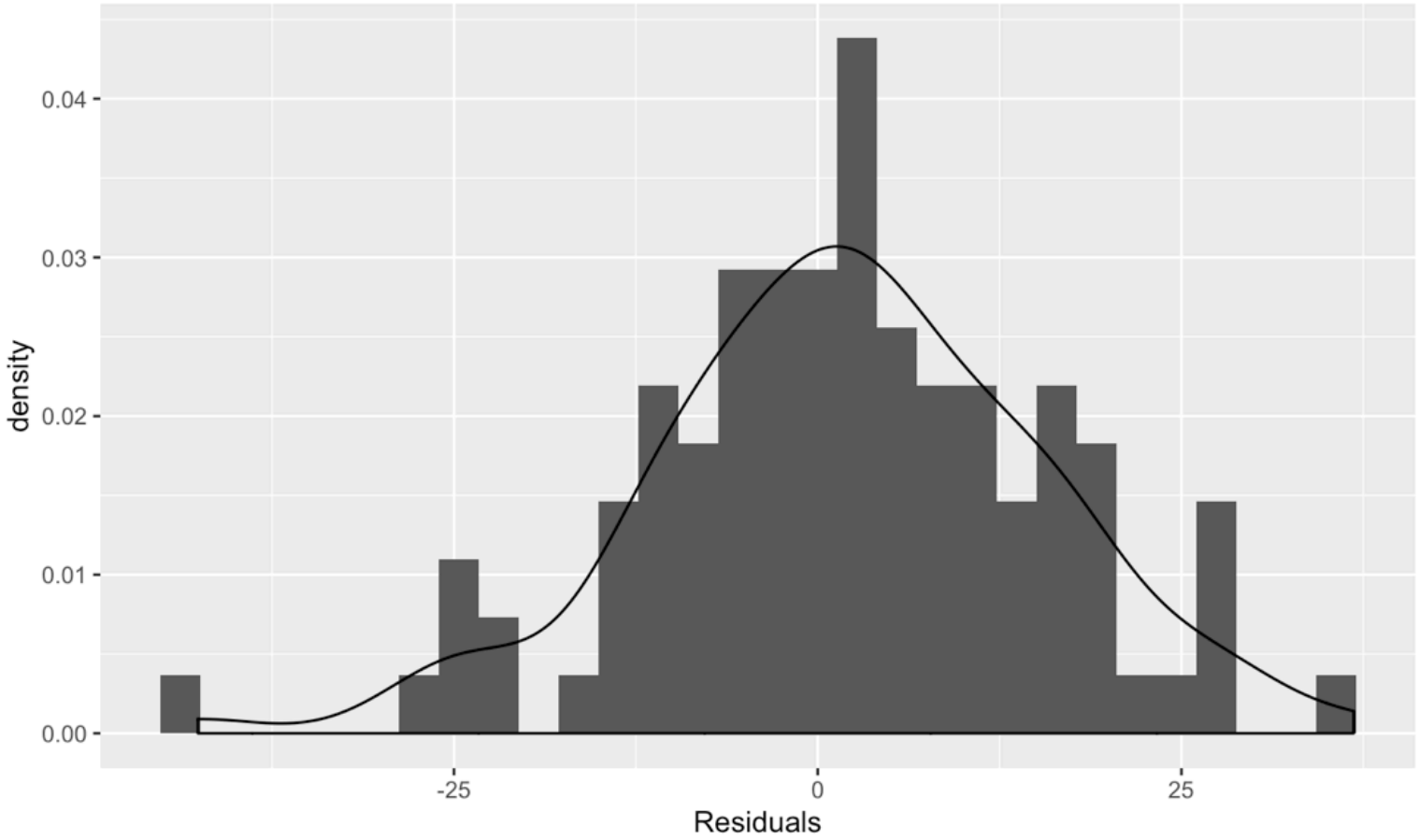



# Violated assumptions #1

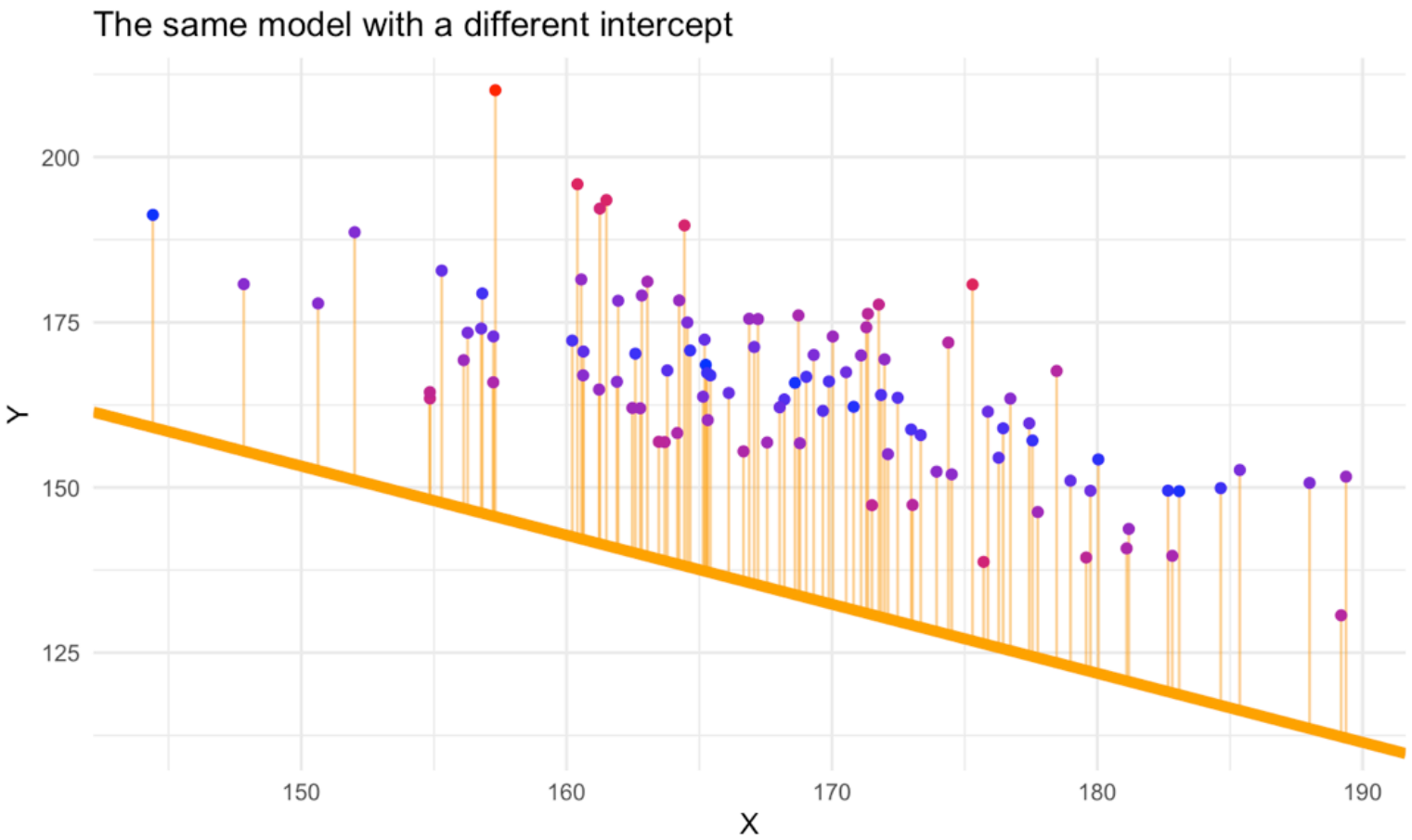



Here the residuals are not independent, and the residual variance is not constant!

# Residual variance

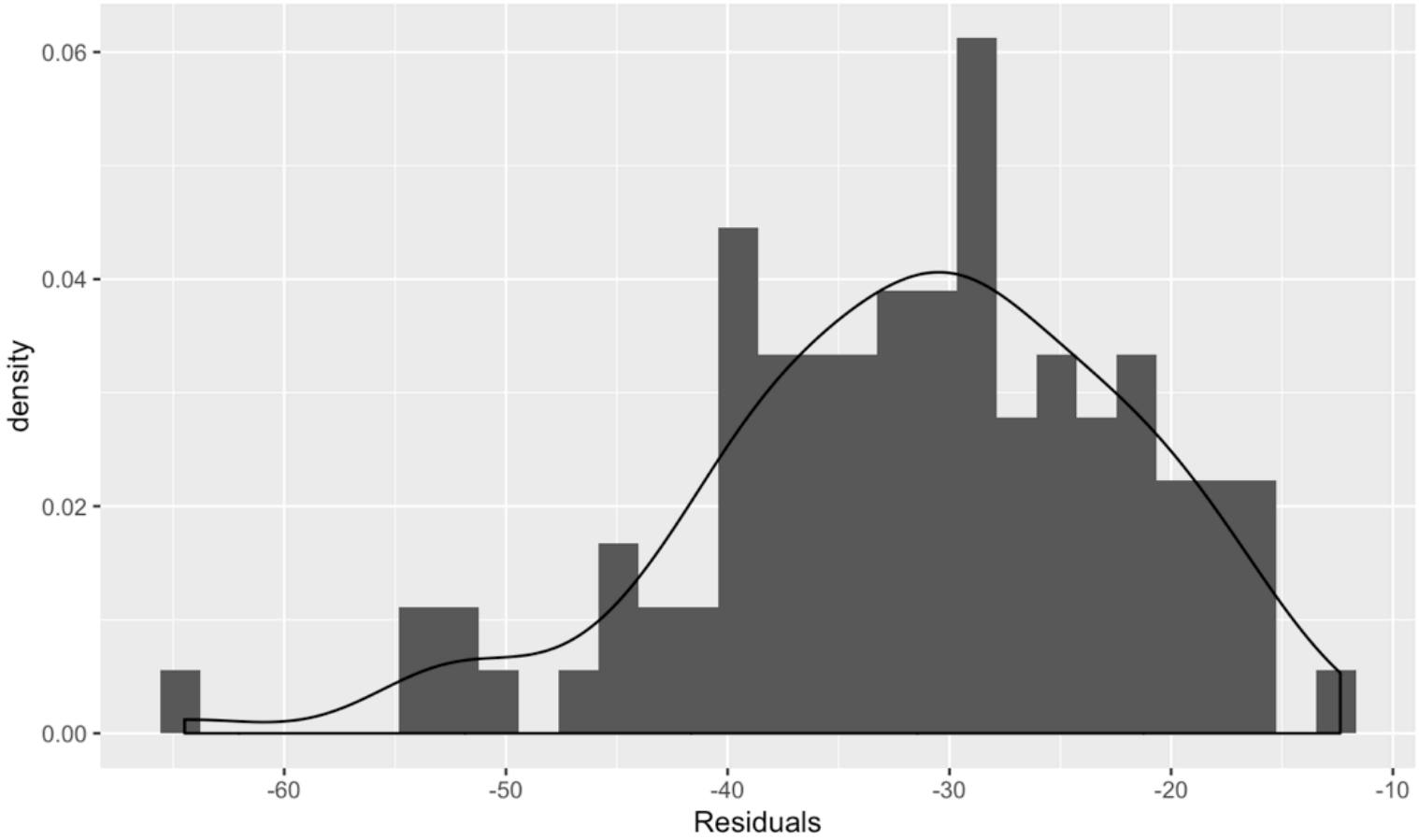


# Violated assumptions #2

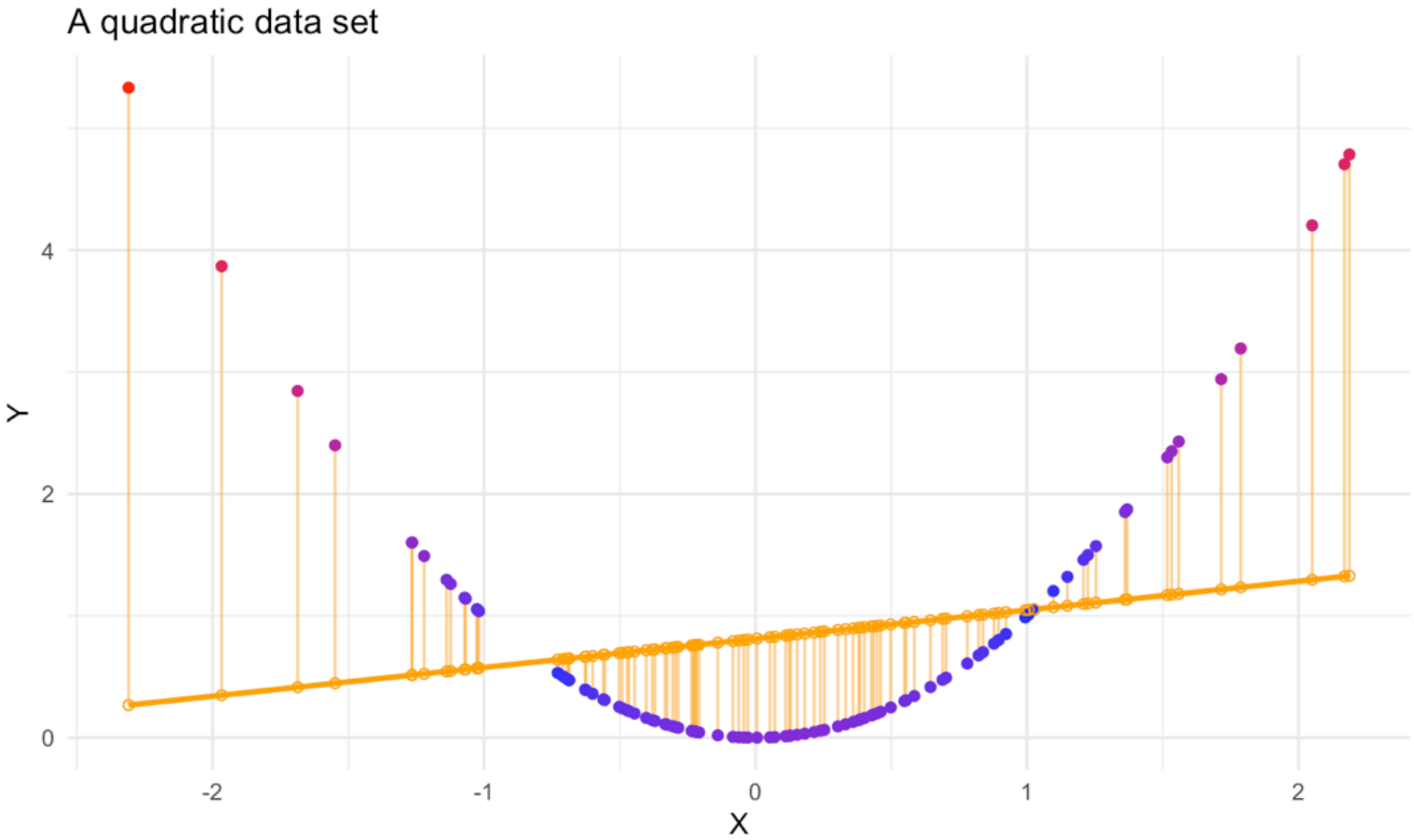



Here the residuals do not have mean zero.

# Residual variance

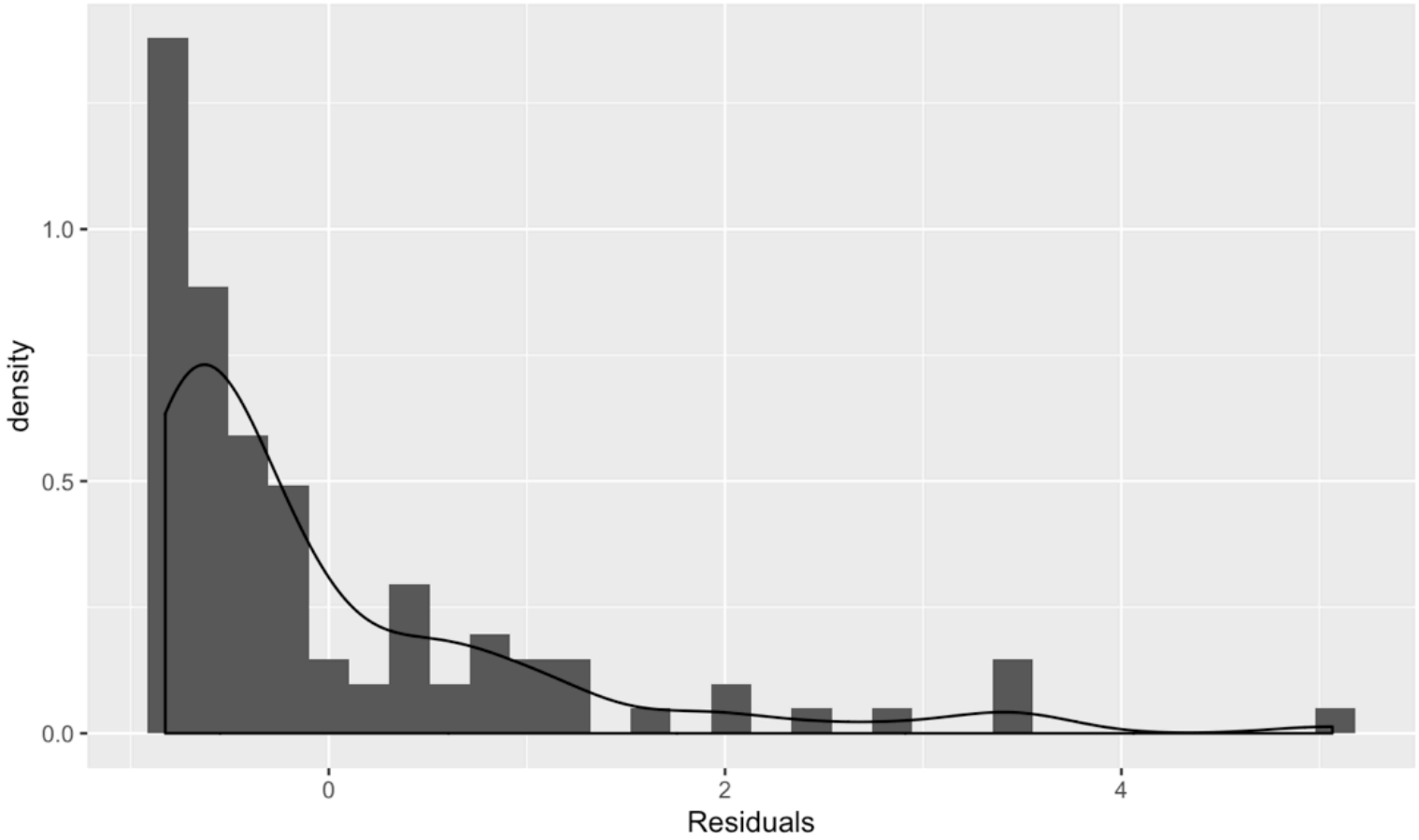


# Violated assumptions #3




Here the residual variance is not constant, the residuals are not normally distributed and the relation between $Y$ and $X$ is not linear!

# Residual variance



# Checking assumptions

# anscombe example

```
fit <- anscombe %$%
  lm(y1 ~ x1)

fit %>% summary
```

```
##
## Call:
## lm(formula = y1 ~ x1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.92127 -0.45577 -0.04136  0.70941  1.83882
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0001     1.1247   2.667  0.02573 *
## x1            0.5001     0.1179   4.241  0.00217 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.237 on 9 degrees of freedom
## Multiple R-squared:  0.6665, Adjusted R-squared:  0.6295
## F-statistic: 17.99 on 1 and 9 DF,  p-value: 0.00217
```

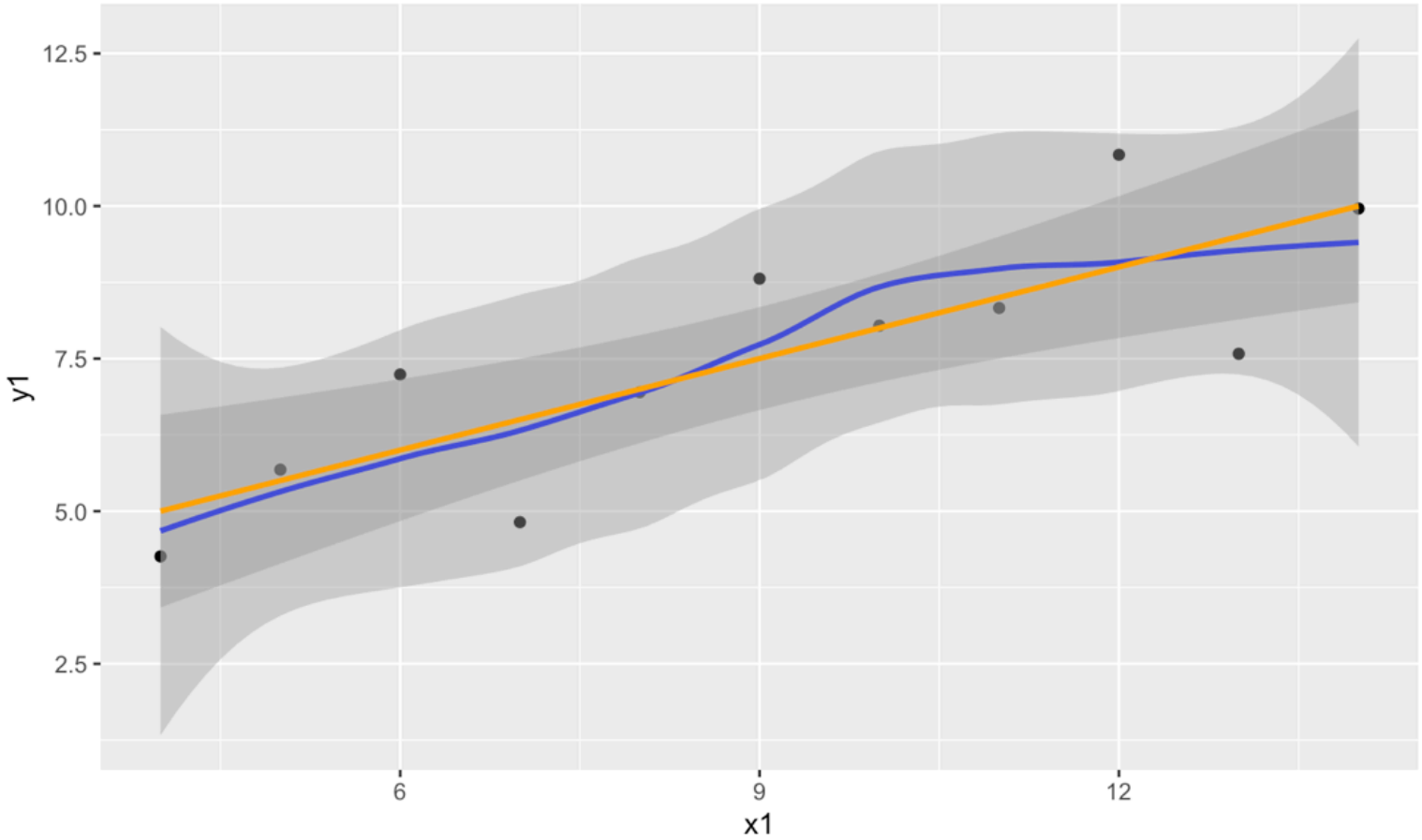
# Checking assumptions

1. linearity
   - scatterplot $y$ and $x$
   - include loess curve when in doubt
   - does a squared term improve fit?
   - plot predicted against observed
2. normality residuals
   - normal probability plots `qqnorm()`
   - if sample is small; `qqnorm` with simulated errors cf. `rnorm(n, 0, s)`
3. constant error variance
   - residual plot
   - scale-location plot

# Linearity

```
anscombe %>%
  ggplot(aes(x1, y1)) +
  geom_point() +
  geom_smooth(method = "loess", col = "blue") +
  geom_smooth(method = "lm", col = "orange")
```

# Linearity




The loess curve suggests slight non-linearity

# Linearity

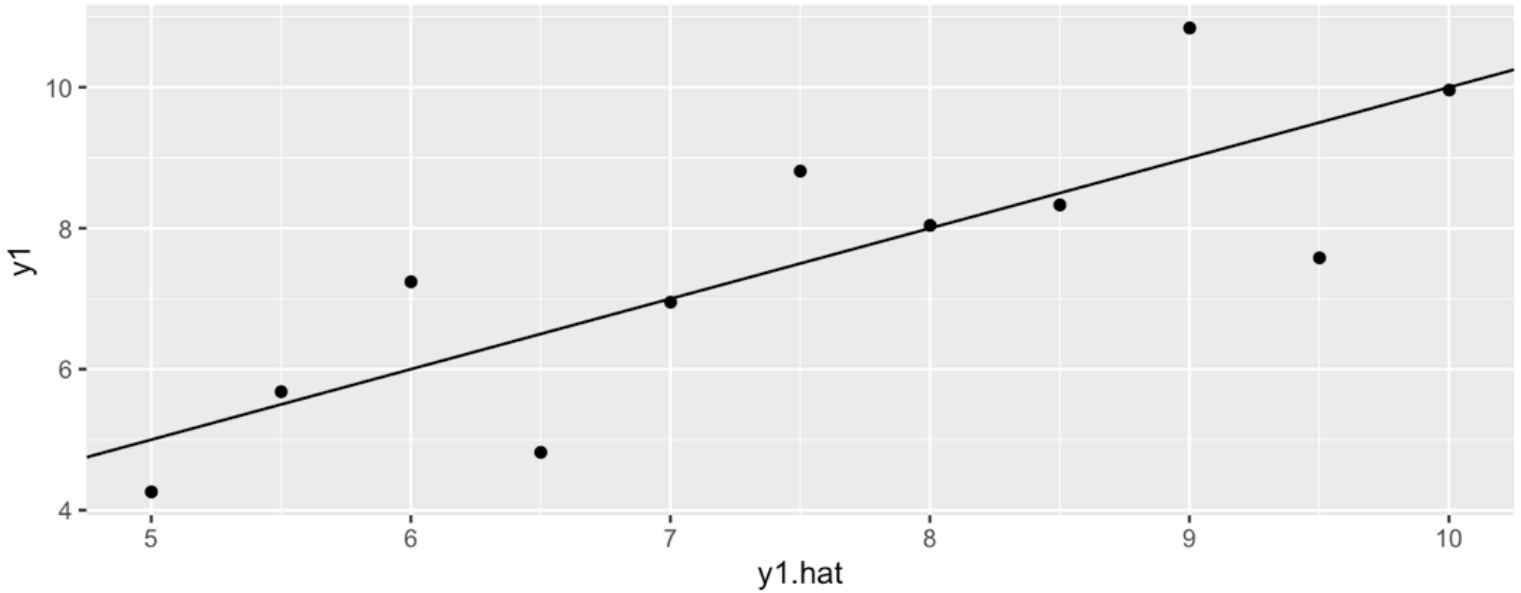
Does adding a squared term improve fit?

```
anscombe %$%
  lm(y1 ~ x1 + I(x1^2)) %>%
  summary()
```

```
##
## Call:
## lm(formula = y1 ~ x1 + I(x1^2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8704 -0.3481 -0.2456  0.7129  1.8072
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.75507    3.28815   0.230    0.824
## x1           1.06925    0.78982   1.354    0.213
## I(x1^2)     -0.03162    0.04336  -0.729    0.487
##
## Residual standard error: 1.27 on 8 degrees of freedom
## Multiple R-squared:  0.6873, Adjusted R-squared:  0.6092
## F-statistic: 8.793 on 2 and 8 DF,  p-value: 0.009558
```

# Linearity

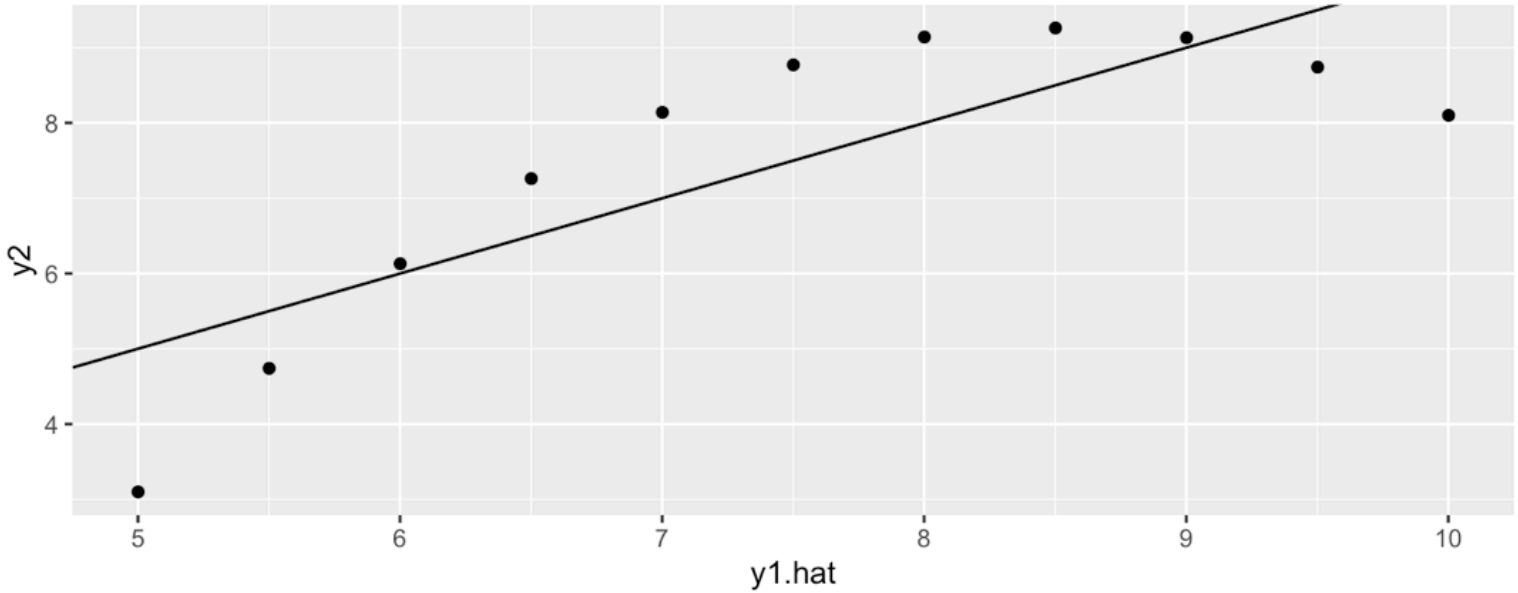
```
data.frame(y1 = anscombe$y1, y1.hat = predict(fit)) %>%
  ggplot(aes(x=y1.hat, y=y1)) + geom_point() + geom_abline(intercept = 0, slope = 1)
```




Plot predicted `y1` against observed `y1`. This is a useful technique to investigate linearity for models with multiple predictors.

# Violation of linearity

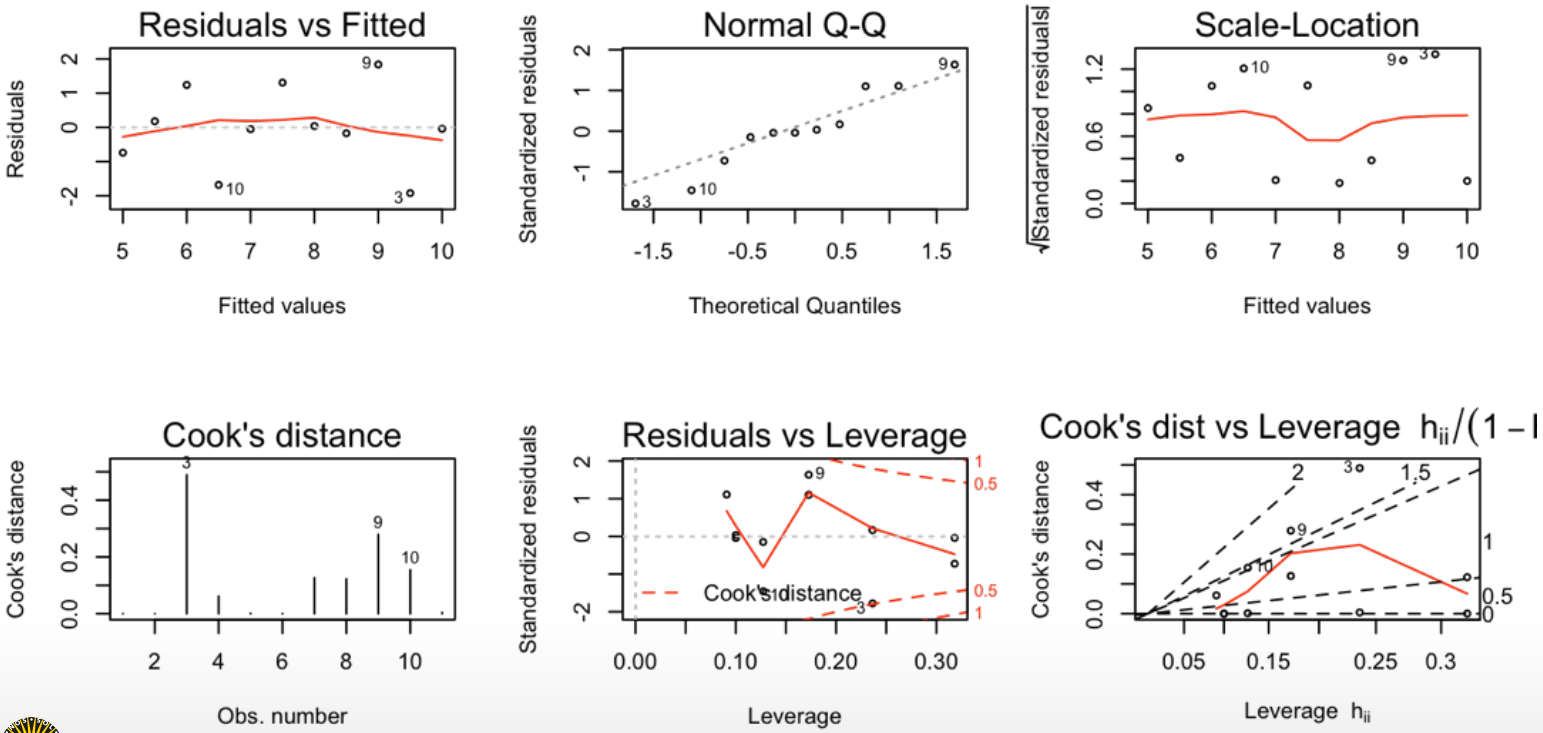
```
data.frame(y2 = anscombe$y2, y1.hat = predict(fit)) %>%
  ggplot(aes(x=y1.hat, y=y2)) + geom_point() + geom_abline(intercept = 0, slope = 1)
```




When we substitute observed `y1` with observed `y2`, it is apparent that the relation `(y2, x1)` is not linear.

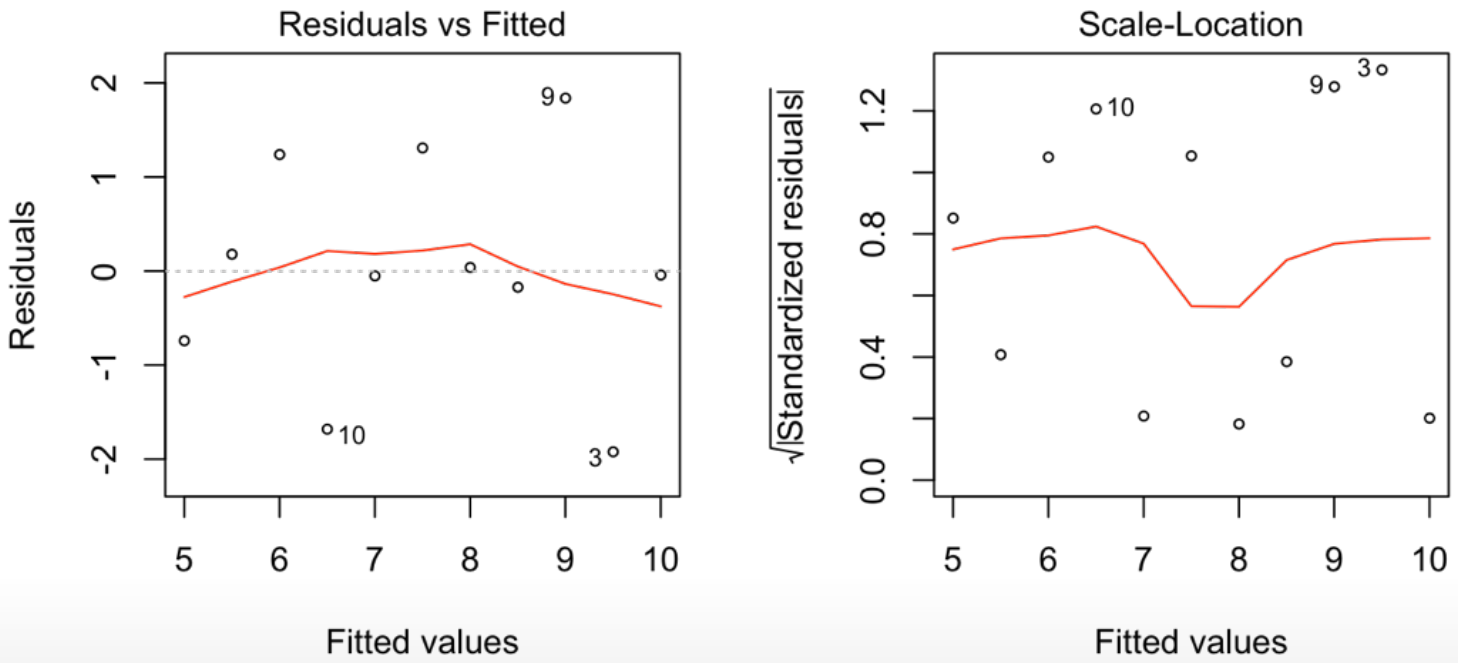
# Plots in class `lm`

```
par(mfrow = c(2, 3))
fit %>%
  plot(which = c(1:6), cex = .6)
```



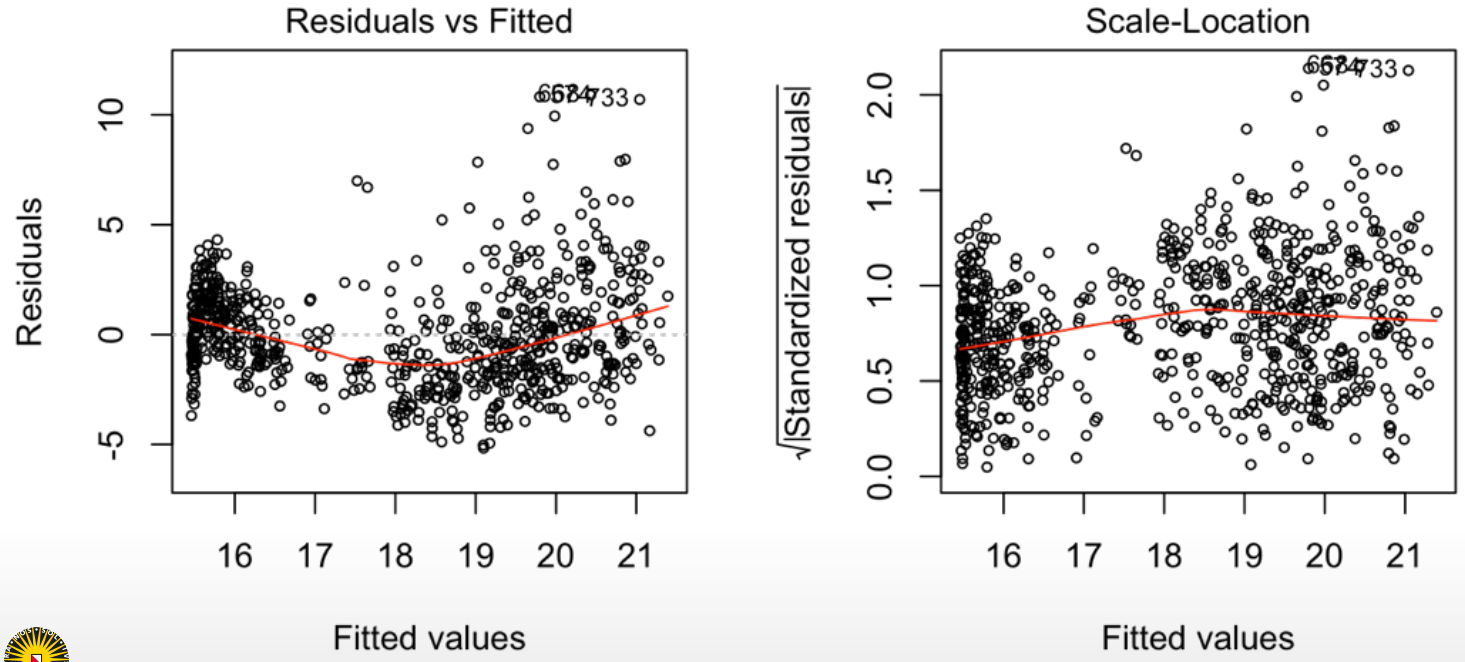
# Constant error variance?

```
par(mfrow = c(1, 2))
fit %>%
  plot(which = c(1, 3), cex = .6)
```



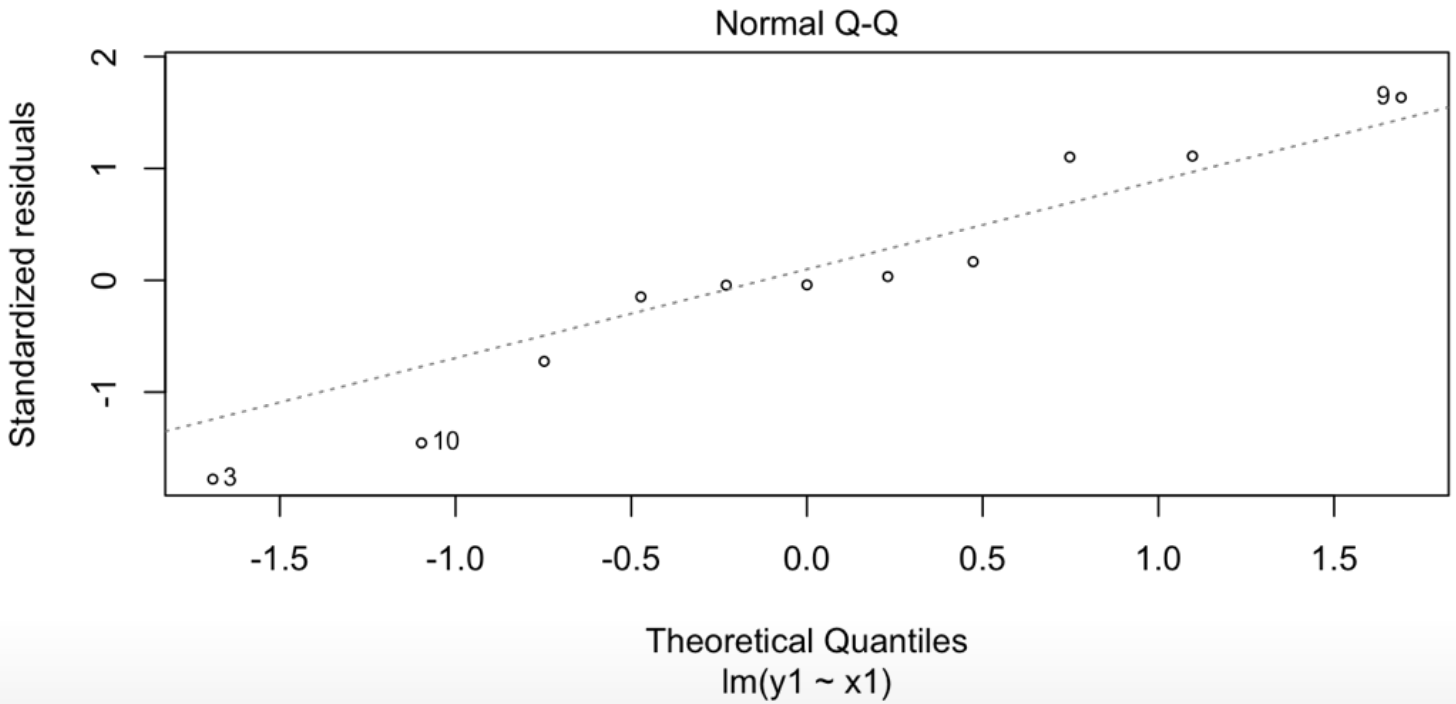

# No constant error variance!

```
par(mfrow = c(1, 2))
boys %$%
  lm(bmi ~ age) %>%
  plot(which = c(1, 3), cex = .6)
```



# Normality of errors

```
fit %>%
  plot(which = 2, cex = .6)
```




The QQplot shows some divergence from normality at the tails

# Outliers, influence and robust regression

# Outliers and influential cases

Leverage: see the fit line as a lever.

- some points pull/push harder; they have more leverage

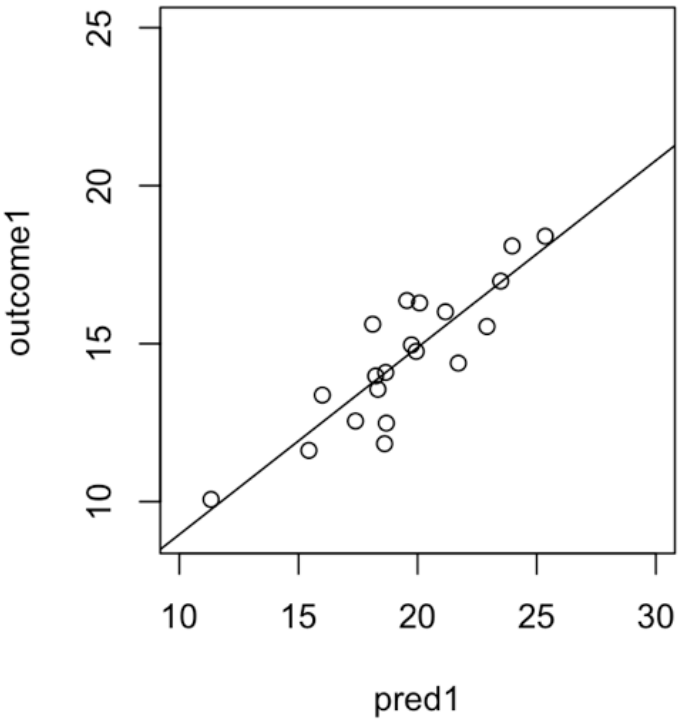
Standardized residuals:

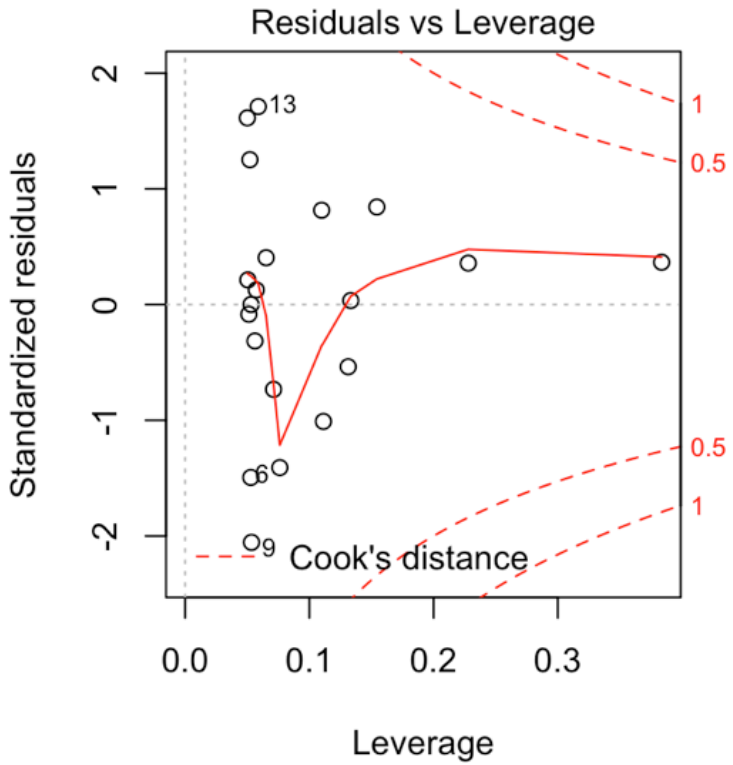
- The values that have more leverage tend to be closer to the line
- The line is fit so as to be closer to them
- The residual standard deviation can differ at different points on $X$ - even if the error standard deviation is constant.
- Therefore we standardize the residuals so that they have constant variance (assuming homoscedasticity).

Cook's distance: how far the predicted values would move if your model were fit without the data point in question.

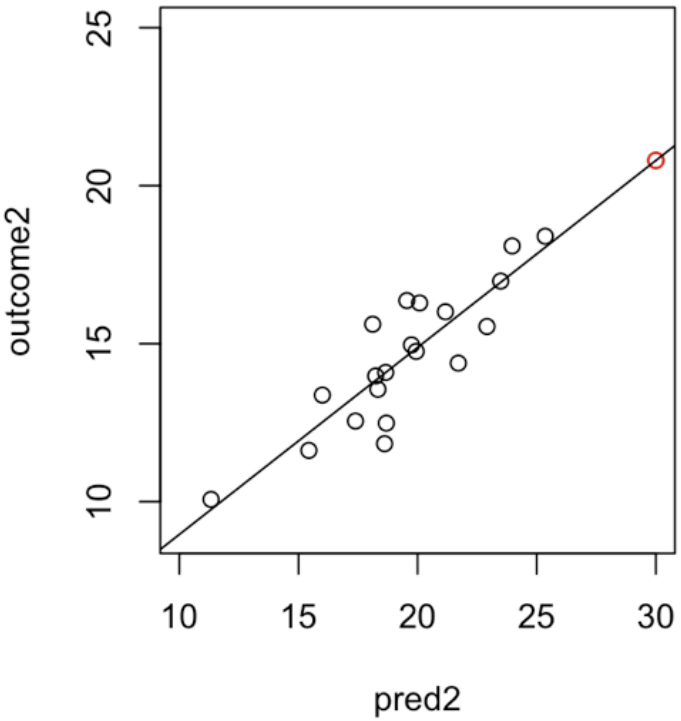
- it is a function of the leverage and standardized residual associated with each data point

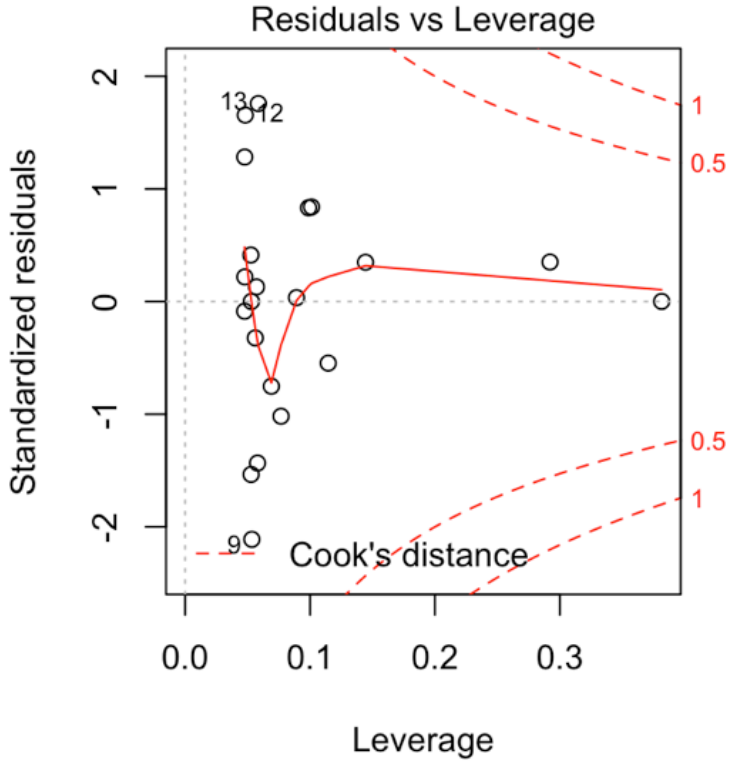
# Fine


outcome1
pred1
10
15
20
25
30

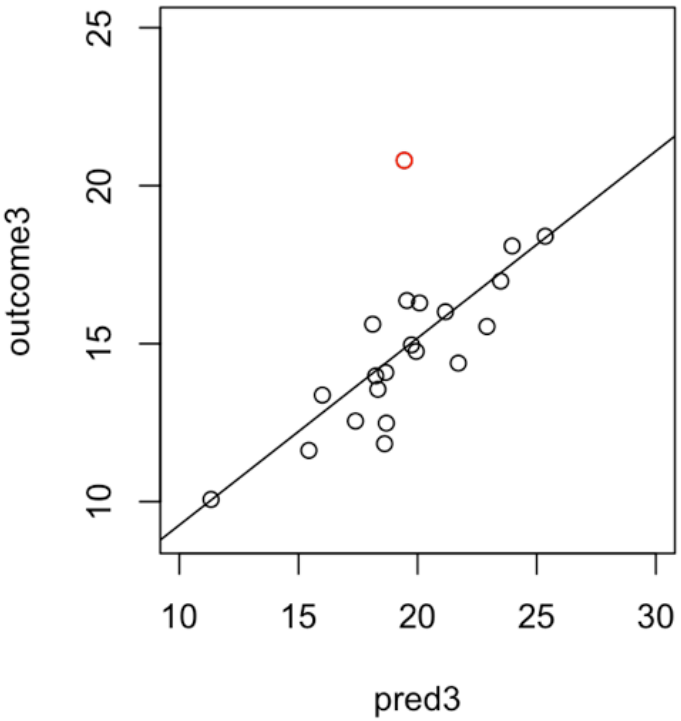

Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance
13
6
9
1
0.5
0.5
1
0.0
0.1
0.2
0.3
-2
-1
0
1
2

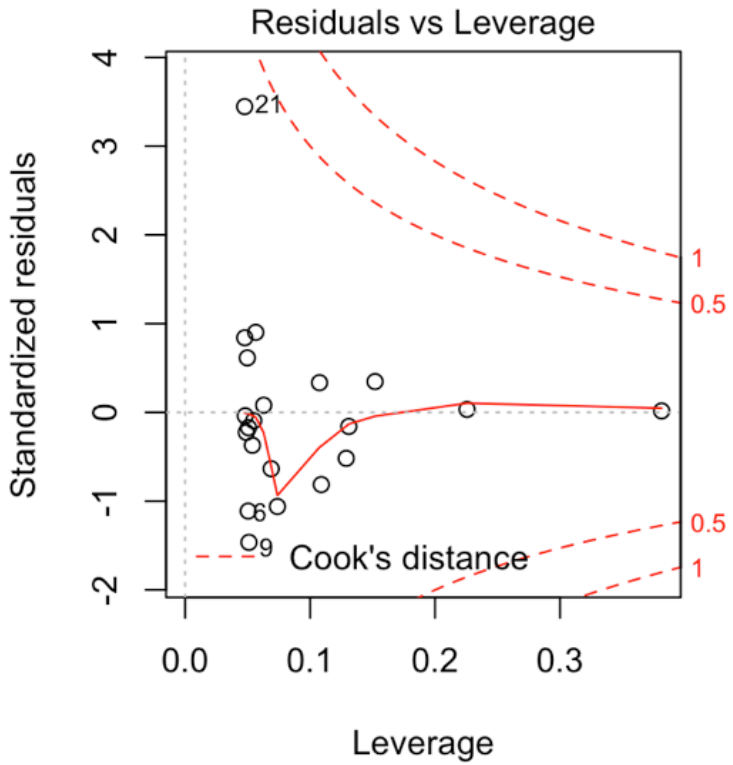
# High leverage, low residual



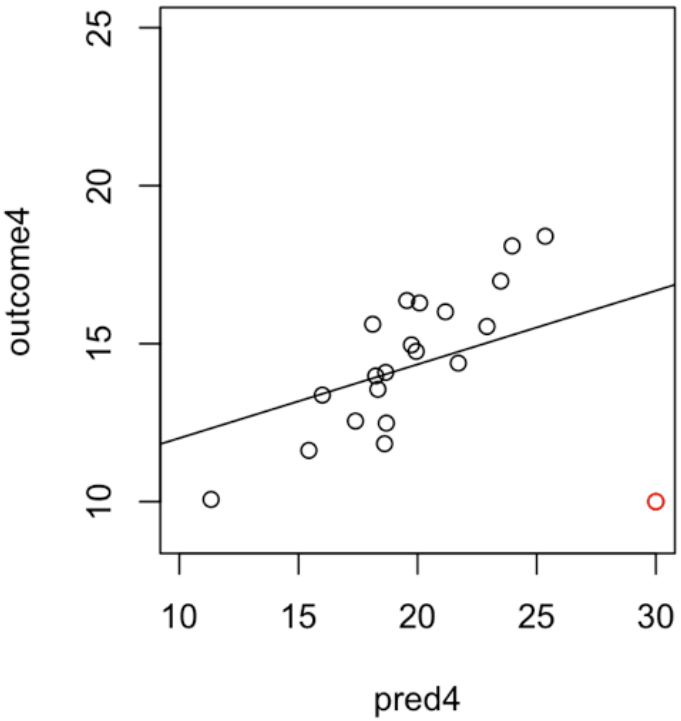



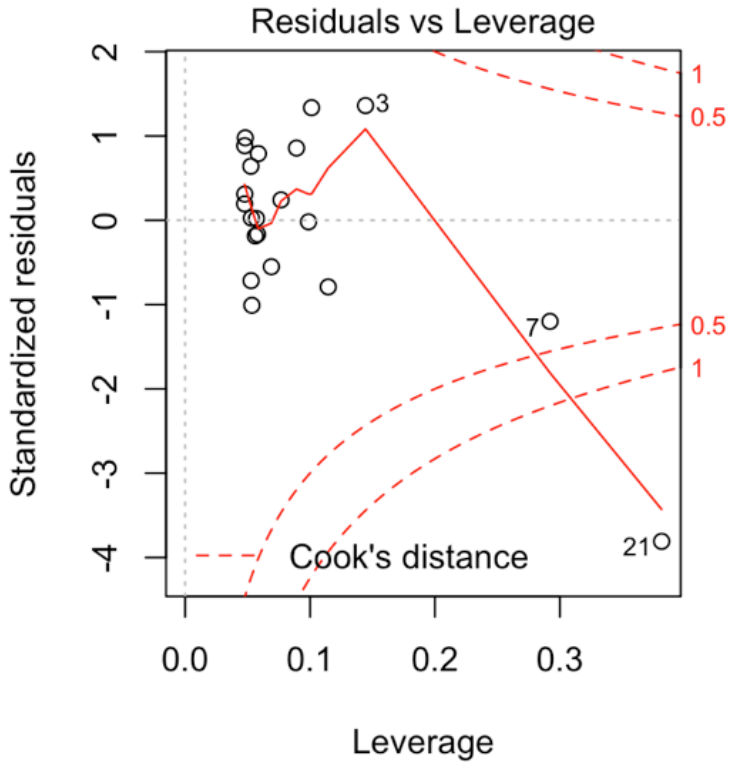

# Low leverage, high residual



# High leverage, high residual



# Outliers and influential cases

Outliers are cases with large $e_z$ (standardized residuals).

If the model is ***correct*** we expect:

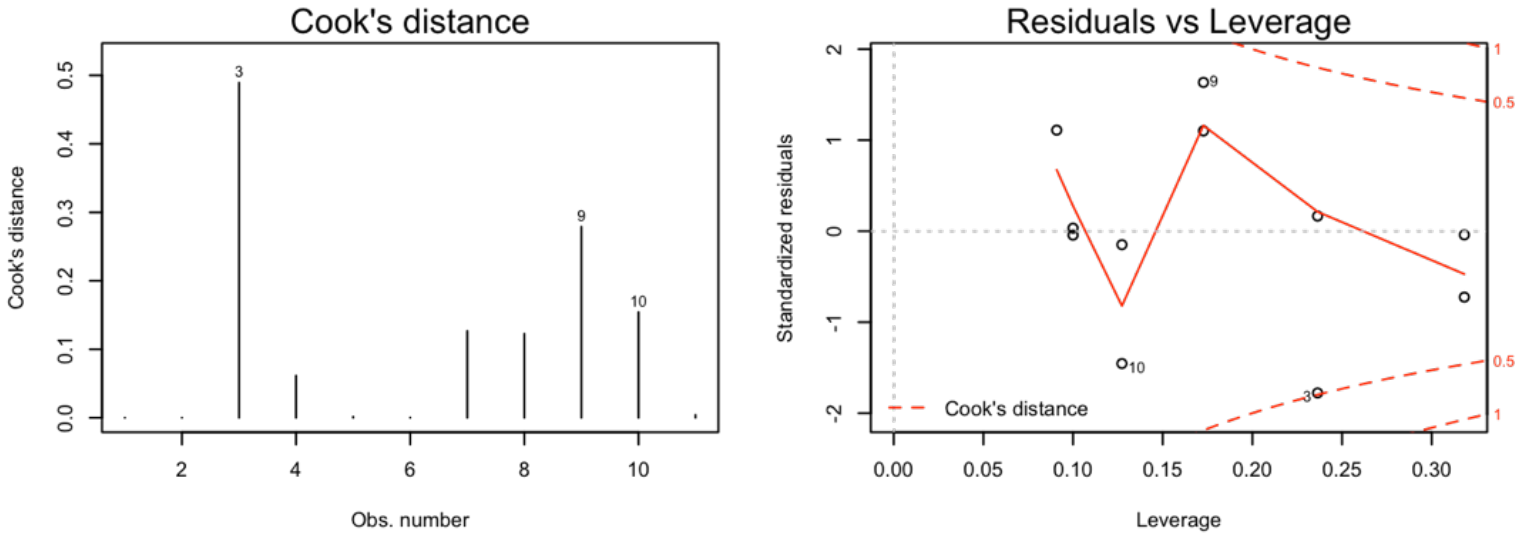
- 5% of standardized residuals $|e_z| > 1.96$
- 1% of standardized residuals $|e_z| > 2.58$
- 0% of standardized residuals $|e_z| > 3.3$

Influential cases are cases with large influence on parameter estimates

- cases with Cook's Distance $> 1$, or
- cases with Cook's Distance much larger than the rest

# Outliers and influential cases

```
par(mfrow = c(1, 2), cex = .6)
fit %>% plot(which = c(4, 5))
```




There are no cases with $|e_z| > 2$, so no outliers (right plot). There are no cases with Cook's Distance $> 1$, but case 3 stands out

# Generating data

# Creating our own data

We saw that we could create vectors

```
c(1, 2, 3, 4, 3, 2, 1)
```

```
## [1] 1 2 3 4 3 2 1
```

```
c(1:4, 3:1)
```

```
## [1] 1 2 3 4 3 2 1
```

We could create matrices

```
matrix(1:15, nrow = 3, ncol = 5, byrow = TRUE)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
```

# Creating our own data

## Vectors from matrices

```
mat <- matrix(1:18, 3, 6)
mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    4    7   10   13   16
## [2,]    2    5    8   11   14   17
## [3,]    3    6    9   12   15   18
```

```
c(mat)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
```

# But we can also draw a sample

Of the same size,

```
sample(mat)
```

```
##  [1]  1  7 17 18 12 14  3 13  9 15  8 16 11 10  6  2  4  5
```

smaller size,

```
sample(mat, size = 5)
```

```
## [1] 16 14  3  5 15
```

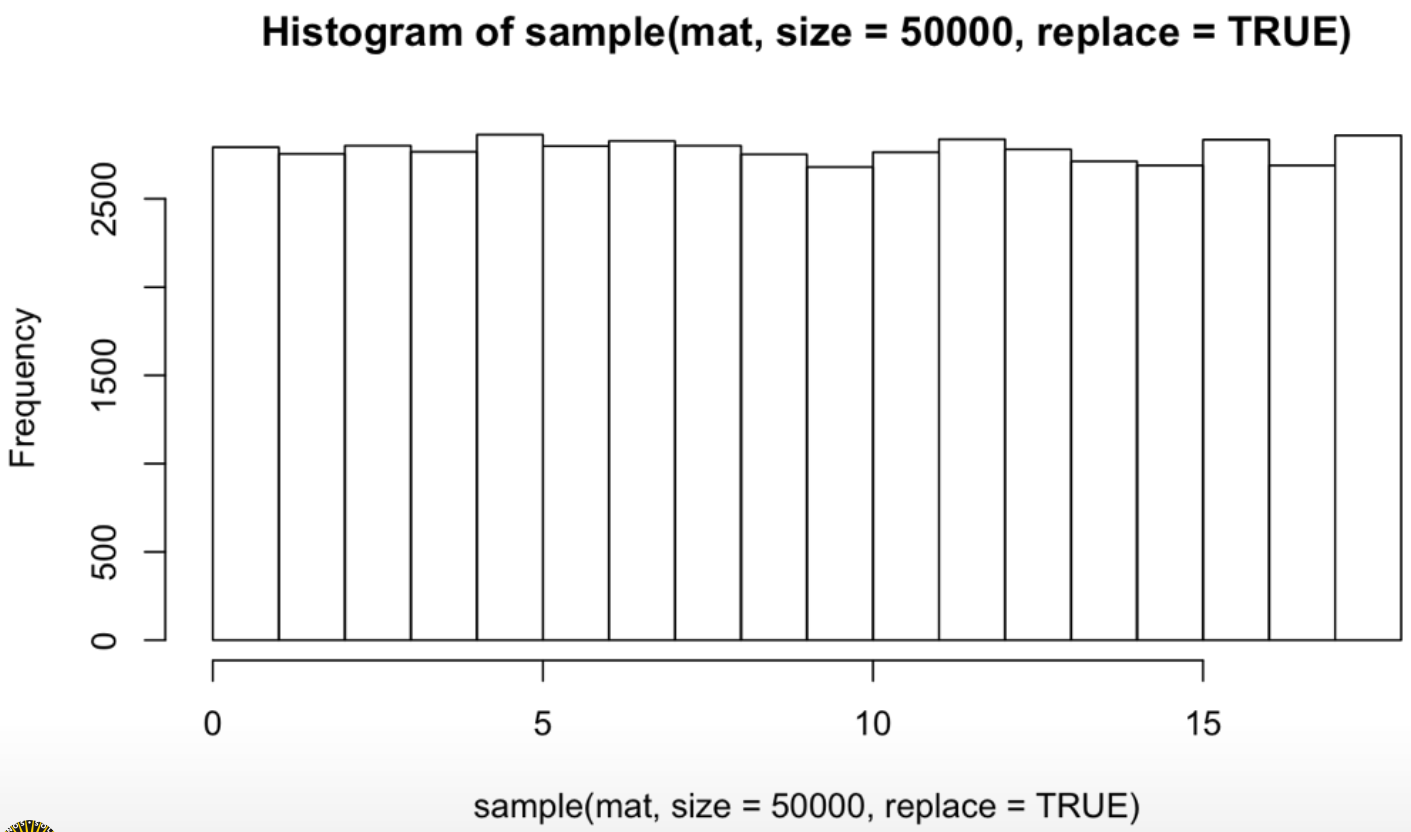
or larger size

```
sample(mat, size = 50, replace = TRUE)
```

```
##  [1]  3 18 11 18  6  8  4  1 13  7  1 17  6 17  8  3 11  7 18  7 18 10  5
## [24] 11  1  6  3 16  8  3  8  4  5  1  6  8 16  5  1  5  8 16  2 16 10  2
## [47]  4 10  9 11
```

# We can do random sampling

```
hist(sample(mat, size = 50000, replace=TRUE), breaks = 0:18)
```

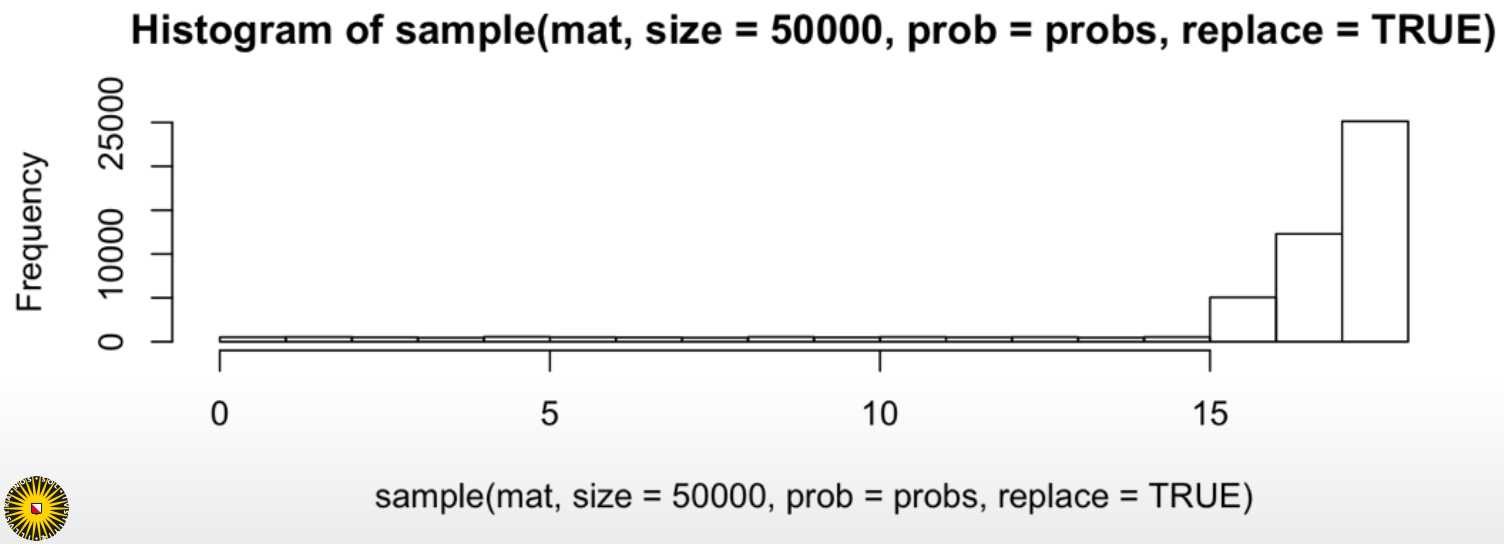



# Or nonrandom sampling

```
probs <- c(rep(.01, 15), .1, .25, .50)
probs
```

```
##  [1] 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
## [15] 0.01 0.10 0.25 0.50
```

```
hist(sample(mat, size = 50000, prob = probs, replace=TRUE), breaks = 0:18)
```



# We can replicate individual samples

```
set.seed(123)
sample(mat, size = 50, replace = TRUE)
```

```
##  [1] 15 14  3 10 18 11  5 14  5  9  3  8  7 10  9  4 14 17 11  7 12 15 10
## [24] 13  7  9  9 10  7  6  2  5  8 12 13 18  1  6 15  9 15 16  6 11  8  7
## [47] 16 17 18 17
```

```
set.seed(123)
sample(mat, size = 50, replace = TRUE)
```

```
##  [1] 15 14  3 10 18 11  5 14  5  9  3  8  7 10  9  4 14 17 11  7 12 15 10
## [24] 13  7  9  9 10  7  6  2  5  8 12 13 18  1  6 15  9 15 16  6 11  8  7
## [47] 16 17 18 17
```

# We can replicate a chain of samples

```
set.seed(123)
sample(mat, size = 5, replace = TRUE)
```

```
## [1] 15 14  3 10 18
```

```
sample(mat, size = 7, replace = TRUE)
```

```
## [1] 11  5 14  5  9  3  8
```

```
set.seed(123)
sample(mat, size = 5, replace = TRUE)
```

```
## [1] 15 14  3 10 18
```

```
sample(mat, size = 7, replace = TRUE)
```

```
## [1] 11  5 14  5  9  3  8
```

# The random seed

The random seed is a number used to initialize the pseudo-random number generator

If replication is needed, pseudorandom number generators must be used

- Pseudorandom number generators generate a sequence of numbers
- The properties of generated number sequences approximates the properties of random number sequences
- Pseudorandom number generators are not truly random, because the process is determined by an initial value.

The initial value (the seed) itself does not need to be random.

- The resulting process is random because the seed value is not used to generate randomness
- It merely forms the starting point of the algorithm for which the results are random.

# Why fix the random seed

When an `R` instance is started, there is initially no seed. In that case, `R` will create one from the current time and process ID.

- Hence, different sessions will give different results when random numbers are involved.
- When you store a workspace and reopen it, you will continue from the seed specified within the workspace.

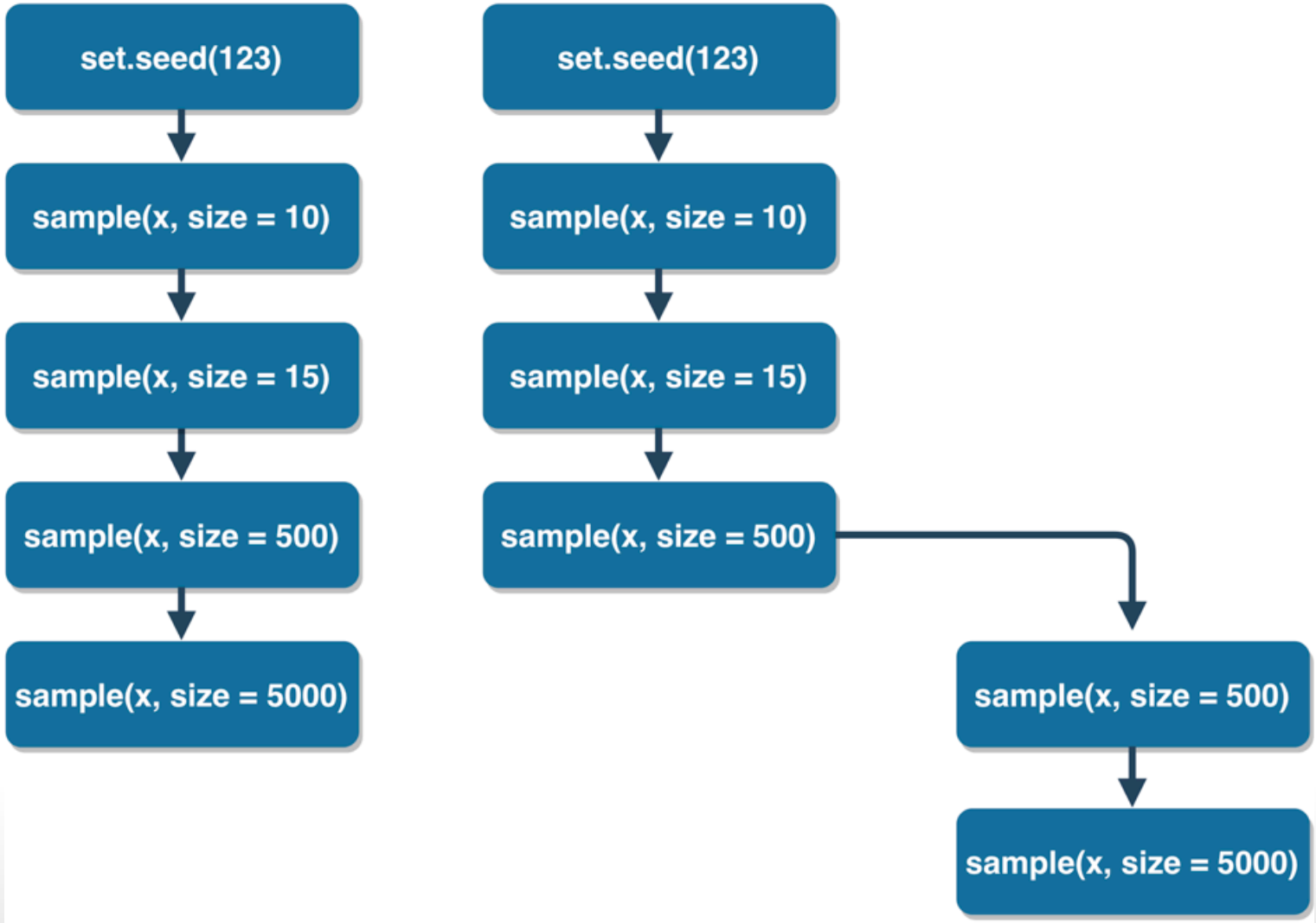
***If we fix the random seed we can exactly replicate the random process***

`If the method has not changed:` the results of the process will be identical when using the same seed.

- Replications allows for verification
- But beware: the process depends on the seed
  - The results obtained could theoretically be extremely rare and would not have occured with every other potential seed
  - Run another seed before publishing your results

# Random seeds

# Random processes




# Drawing data

We can draw data from a standard normal distribution

```
hist(rnorm(1000, mean = 5, sd = 1))
```

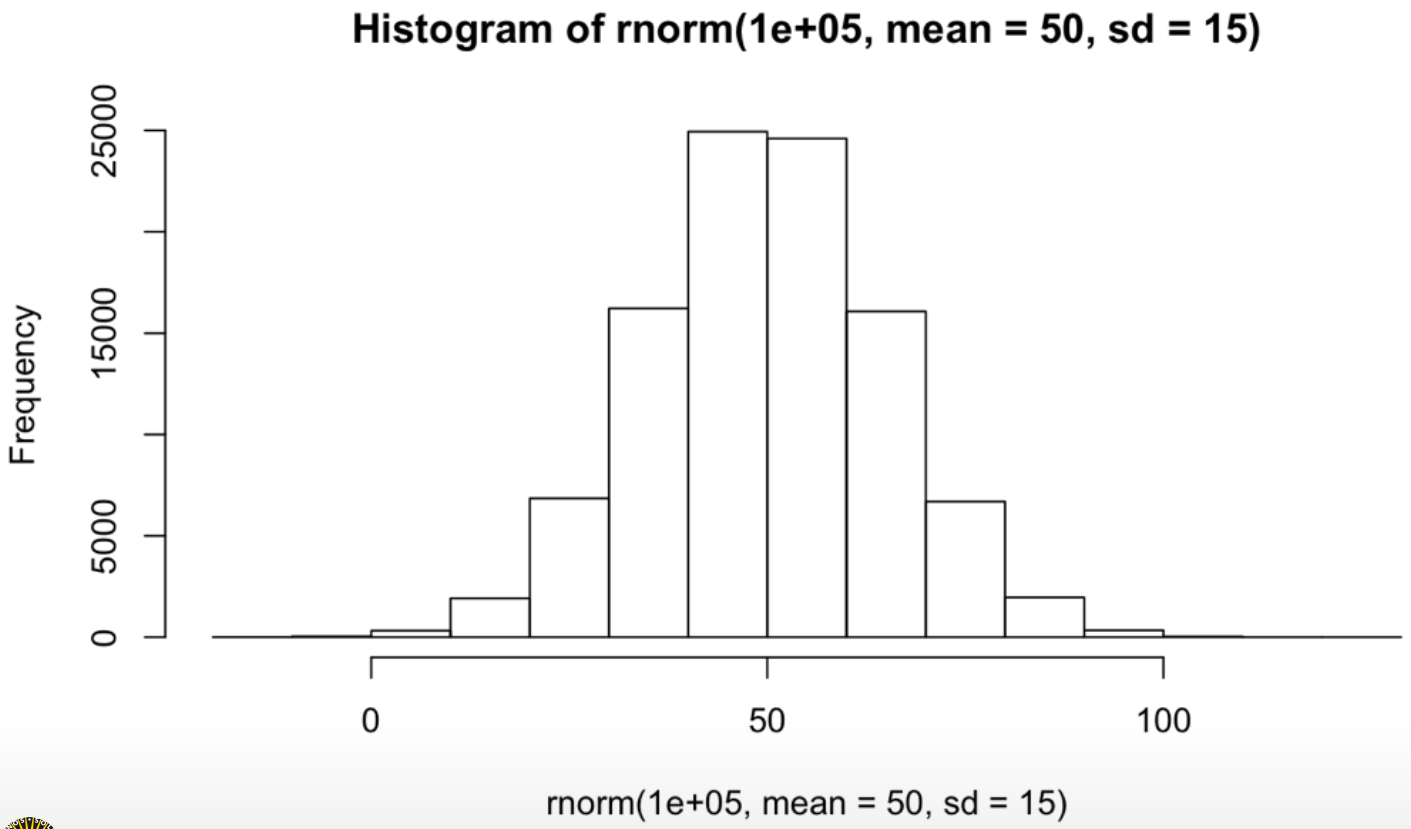
# Drawing data

We can draw data from a specific normal distribution

```
hist(rnorm(1000, mean = 50, sd = 15))
```

# Drawing data: many values

```
hist(rnorm(100000, mean = 50, sd = 15))
```




# A review of inference concepts

# Basic concepts

# Parameters and statistics

Random sample

- set of values drawn independently from a population

Population parameter

- Property of the population (e.g. $\mu$, $\sigma$, $\rho$)

Sample statistic

- Property of the sample (e.g. $\bar{x}$, $s$, $r$)

# Sampling distributions

The sampling distribution is the distribution of a sample statistic

## Central Limit Theorem

- ***Given a distribution with mean $\mu$ and standard deviation $\sigma$, the sampling distribution of the mean approaches a normal distribution with mean $\mu$ and standard deviation $\frac{\sigma}{\sqrt{n}}$ as $n \to \infty$***

Conditions for the CLT to hold

- $\mu$ and $\sigma$ known
- the sample size $n$ is sufficiently large

# Hypothesis tests and confidence intervals

# Hypothesis tests

Statistical tests work with a null hypothesis, e.g.:

$$H_0 : \mu = \mu_0$$

- If population parameters $\mu$ and $\sigma$ are unknown, we need to use sample mean $\bar{x}$ and sample standard deviation $s$ as estimators
- as a consequence, the sample mean has a $t$-distribution

$$t_{(df)} = \frac{\bar{x} - \mu_0}{SEM}$$

# $t$-distribution

The $t$-distribution has larger variance than the normal distribution (more uncertainty).

- degrees of freedom: $df = n-$ number of estimated means
- the higher the degrees of freedom (the larger the sample) the more it resembles a normal distribution

# $t$-distribution

```
curve(dt(x, 100), -3, 3, ylab = "density")
curve(dt(x,   2), -3, 3, ylab = "", add = T, col = "red")
curve(dt(x,   1), -3, 3, ylab = "", add = T, col = "blue")
legend(1.8, .4, c("t(df=100)", "t(df=2)", "t(df=1)"),
       col = c("black", "red", "blue"), lty=1)
```

# $t$ versus normal

```
curve(dnorm(x), -3, 3, ylab = "density")
curve(dt(x, 2), -3, 3, ylab = "", add = T, col = "red")
curve(dt(x, 1), -3, 3, ylab = "", add = T, col = "blue")
legend(1.8, .4, c("normal", "t(df=2)", "t(df=1)"),
       col = c("black", "red", "blue"), lty=1)
```

# $p$-values

The $p$-value in this situation is the probability that $\bar{x}$ is at least that much different from $\mu_0$:

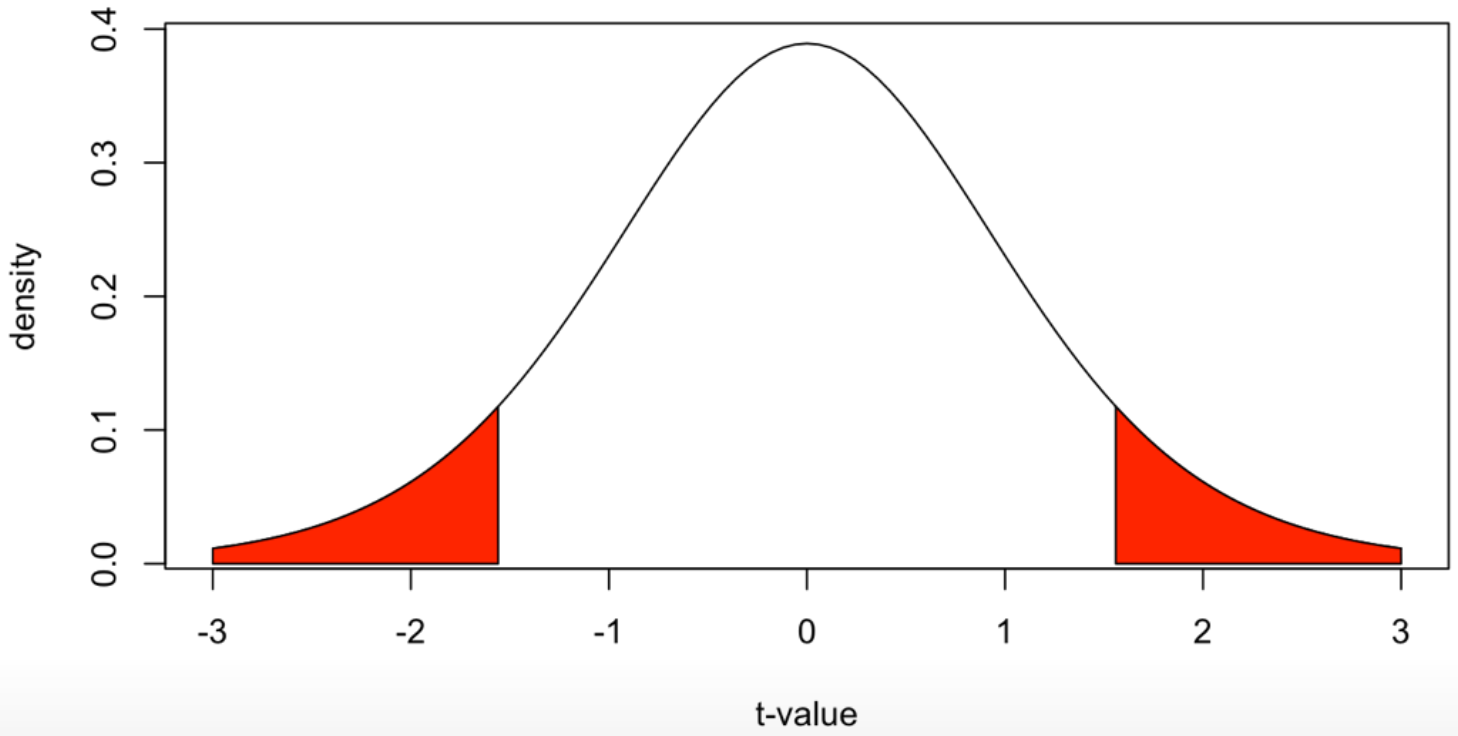
- P($\bar{X} \geq \bar{x}$ |$\mu$=0)

We would reject $H_0$ if $p$ is smaller than the experimenters' (that would be you) predetermined significance level $\alpha$:

- two-sided test if $H_A : \mu \neq \mu_0$ (upper and lower ($\alpha$/2) * 100%)
- one-sided test if $H_A : \mu > \mu_0$ or $H_A : \mu < \mu_0$ (upper or lower $\alpha$ * 100%)

# $p$-values

Example of two-sided test for $t_{(df=10)}$ given that $P(t < -1.559) = 7.5\%$ ($\alpha$ = 0.15)




# $p$-values: code for the figure

```
t0      <- qt(.075, 10)

cord.x1 <- c(-3, seq(-3, t0, 0.01), t0)
cord.y1 <- c(0, dt(seq(-3, t0, 0.01), 10), 0)
cord.x2 <- c(-t0, seq(-t0, 3, 0.01), 3)
cord.y2 <- c(0, dt(seq(-t0, 3, 0.01), 10), 0)

curve(dt(x,10),xlim=c(-3,3),ylab="density",main='',xlab="t-value")
polygon(cord.x1,cord.y1,col='red')
polygon(cord.x2,cord.y2,col='red')
```

# Misconceptions

1. The p-value is not the probability that the null hypothesis is true or the probability that the alternative hypothesis is false. It is not connected to either.
2. The p-value is not the probability that a finding is “merely a fluke.” In fact, the calculation of the p-value is based on the assumption that every finding is the product of chance alone.
3. The p-value is not the probability of falsely rejecting the null hypothesis.
4. The p-value is not the probability that replicating the experiment would yield the same conclusion.
5. The significance level, $\alpha$, is not determined by the p-value. The significance level is decided by the experimenter a-priori and compared to the p-value that is obtained a-posteriori.
6. The p-value does not indicate the size or importance of the observed effect - they are related together with sample size.

# 95% confidence interval

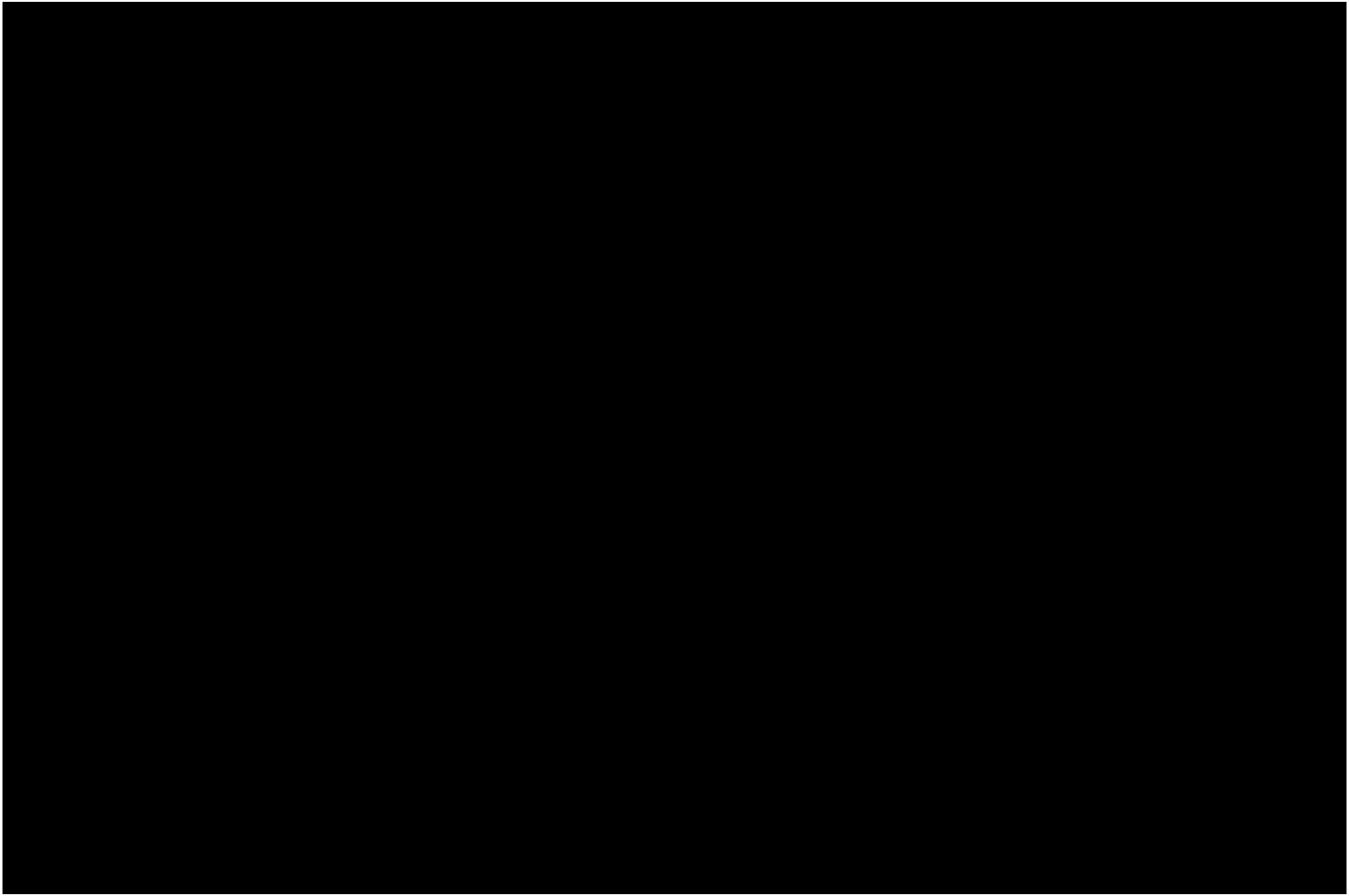
*If an infinite number of samples were drawn and CI's computed, then the true population mean $\mu$ would be in **at least** 95% of these intervals*

$$95\%\ CI = \bar{x} \pm t_{(1-\alpha/2)} \cdot SEM$$

Example

```
x.bar <- 7.6 # sample mean
SEM   <- 2.1 # standard error of the mean
n     <- 11 # sample size
df    <- n-1 # degrees of freedom
alpha <- .15 # significance level
t.crit <- qt(1 - alpha / 2, df) # t(1 - alpha / 2) for df = 10
c(x.bar - t.crit * SEM, x.bar + t.crit * SEM)
```

```
## [1]  4.325605 10.874395
```



The new form of the problem of estimation of the collective character $\theta$ may be stated as follows: given any positive number $\varepsilon < 1$, to associate with any possible value of $x$ an interval

$$\theta_1(x) < \theta_2(x) \quad . \quad . \quad . \quad . \quad . \quad . \quad (1)$$

such that if we accept the rule of stating that the unknown value of the collective character $\theta$ is contained within the limits

$$\theta_1(x') \leq \theta \leq \theta_2(x'). \quad . \quad . \quad . \quad . \quad . \quad (2)$$

every time the actual sampling provides us with the value $x = x'$, the probability of our being wrong is less than or at most equal to $1 - \varepsilon$, and this whatever the probability law *a priori*, $\varphi(\theta)$.

The value of $\varepsilon$, chosen in a quite arbitrary manner, I propose to call the "confidence coefficient." If we choose, for instance, $\varepsilon = \cdot 99$ and find for every possible $x$ the intervals $[\theta_1(x), \theta_2(x)]$ having the properties defined, we could roughly describe the position by saying that we have 99 per cent. confidence in the fact that $\theta$ is contained between $\theta_1(x)$ and $\theta_2(x)$. The numbers $\theta_1(x)$ and $\theta_2(x)$ are what R. A. Fisher calls the fiducial limits of $\theta$.

# Misconceptions

Confidence intervals are frequently misunderstood, even well-established researchers sometimes misinterpret them. .

1. A realised 95% CI does not mean:

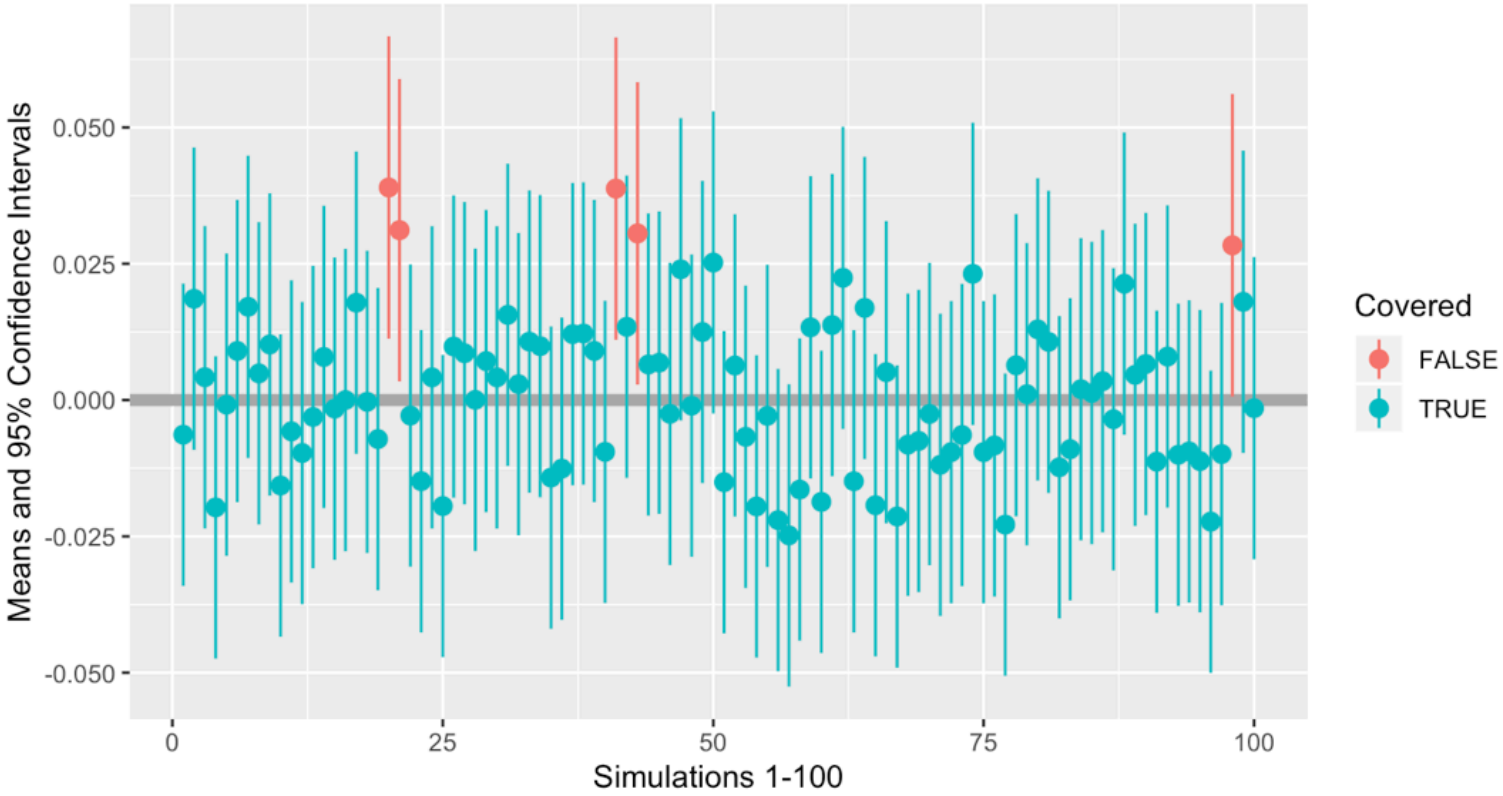
- that there is a 95% probability the population parameter lies within the interval
- that there is a 95% probability that the interval covers the population parameter

  Once an experiment is done and an interval is calculated, the interval either covers, or does not cover the parameter value. Probability is no longer involved.

  The 95% probability only has to do with the estimation procedure.

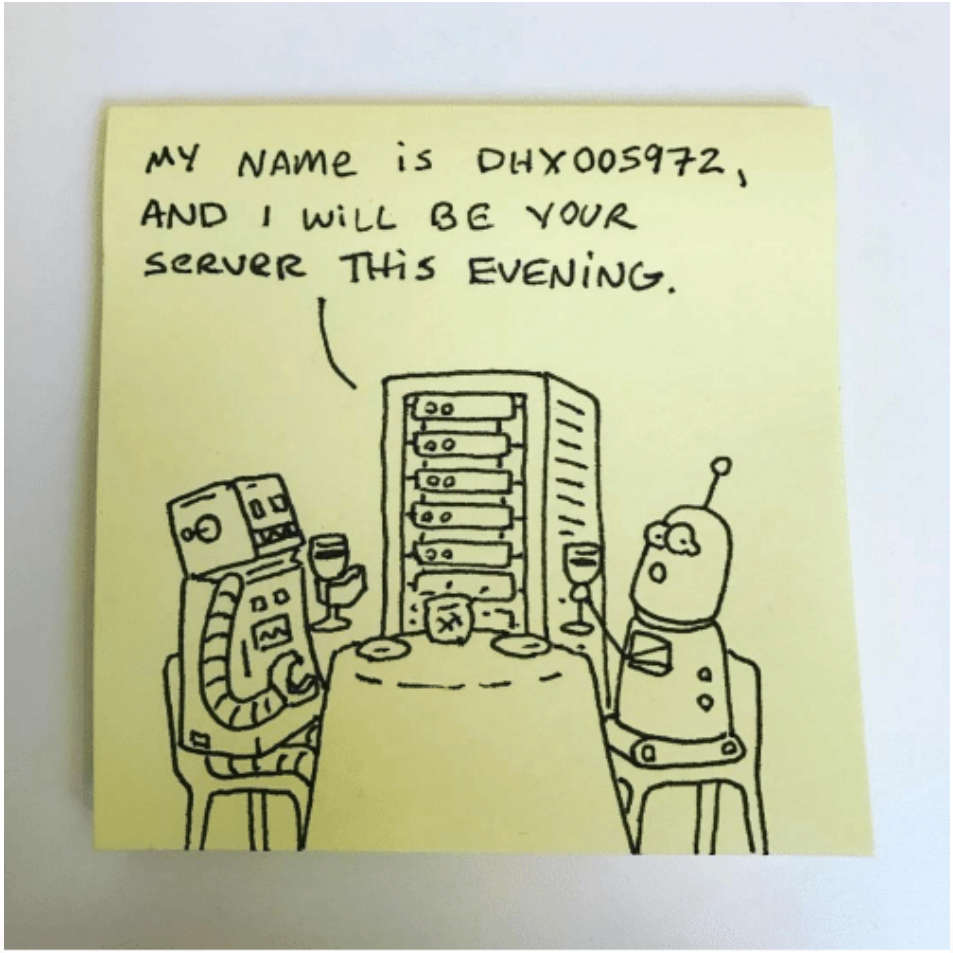
1. A 95% confidence interval does not mean that 95% of the sample data lie within the interval.
2. A confidence interval is not a range of plausible values for the sample mean, though it may be understood as an estimate of plausible values for the population parameter.
3. A particular confidence interval of 95% calculated from an experiment does not mean that there is a 95% probability of a sample mean from a repeat of the experiment falling within this interval.

# Confidence intervals




100 simulated samples from a population with $\mu = 0$ and $\sigma^2 = 1$. Out of 100 samples, only 5 samples have confidence intervals that do not cover the population mean.

# To conclude




When you go out for a byte