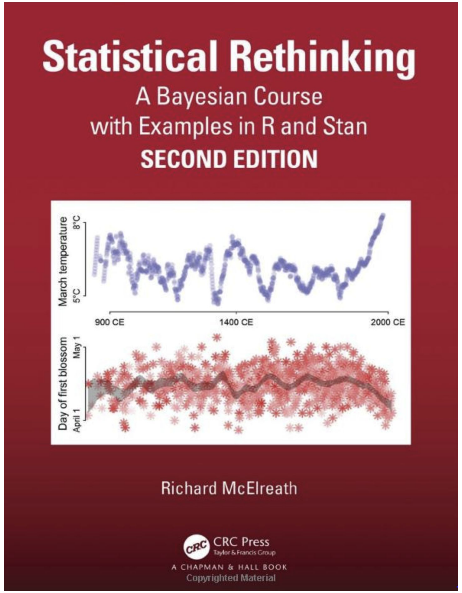
Introduction to Data Science

Model Testing

Prof. Dr. Ralf Lämmel & **M.Sc. Johannes Härtel**
(johanneshaertel@uni-koblenz.de)



[McElreath20]

The major source for this lecture.

The problem of testing models

Asking an empirical question

If asking an empirical question, we are typically interested in variables and functions (relating variables) that we **cannot directly observe.**

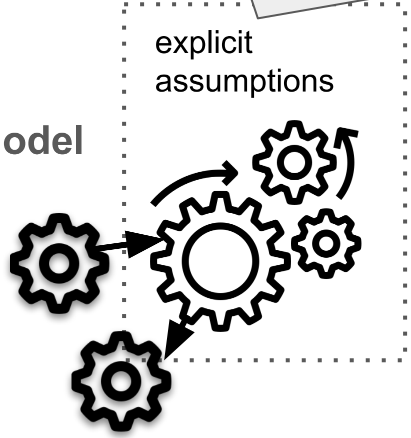
Known variables that we can observe.

Unknown variables and functions, relating variables, are hidden in a **black box**.

Answering an empirical question using a statistic model

To answer the question, **we interpret a model of the reality,** inferring missing variables as parameters. Our model formulates assumptions on the reality that do not need to be correct.

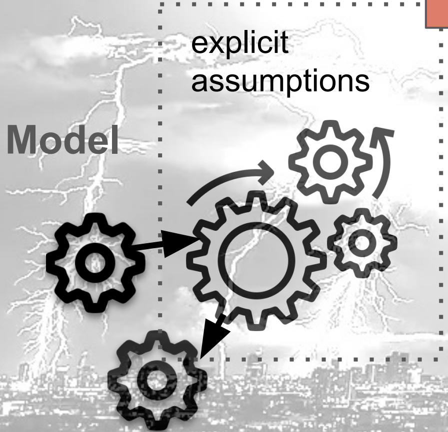
Model



We have been doing this using ULAM and STAN.

The central **problem of testing**

It is **not possible** to **directly compare** our model and the reality (the true model).

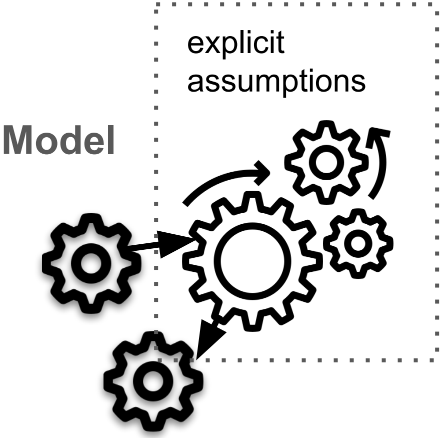


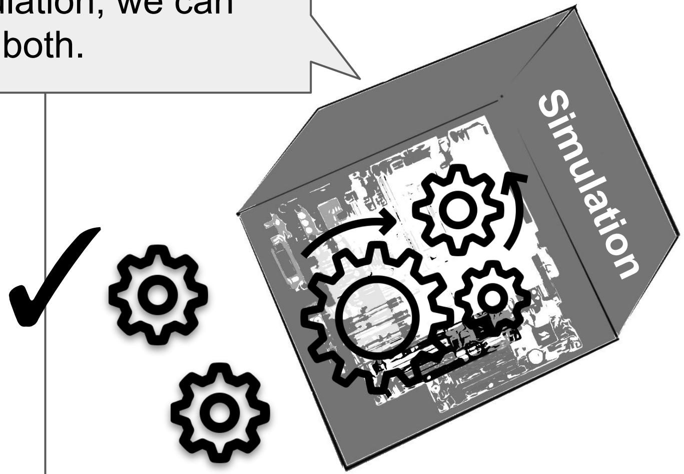
We have been doing this using ULAM and STAN.

Simulation-based testing

Simulation-based testing

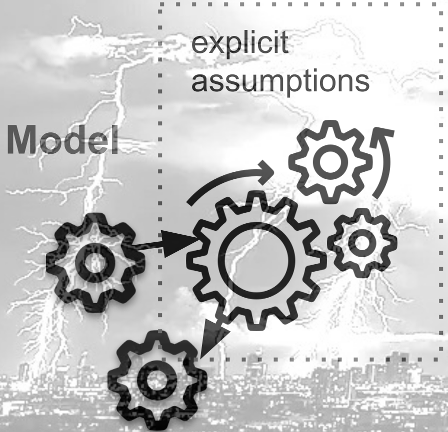
If we **replace the reality** by a simulation, we can compare both.



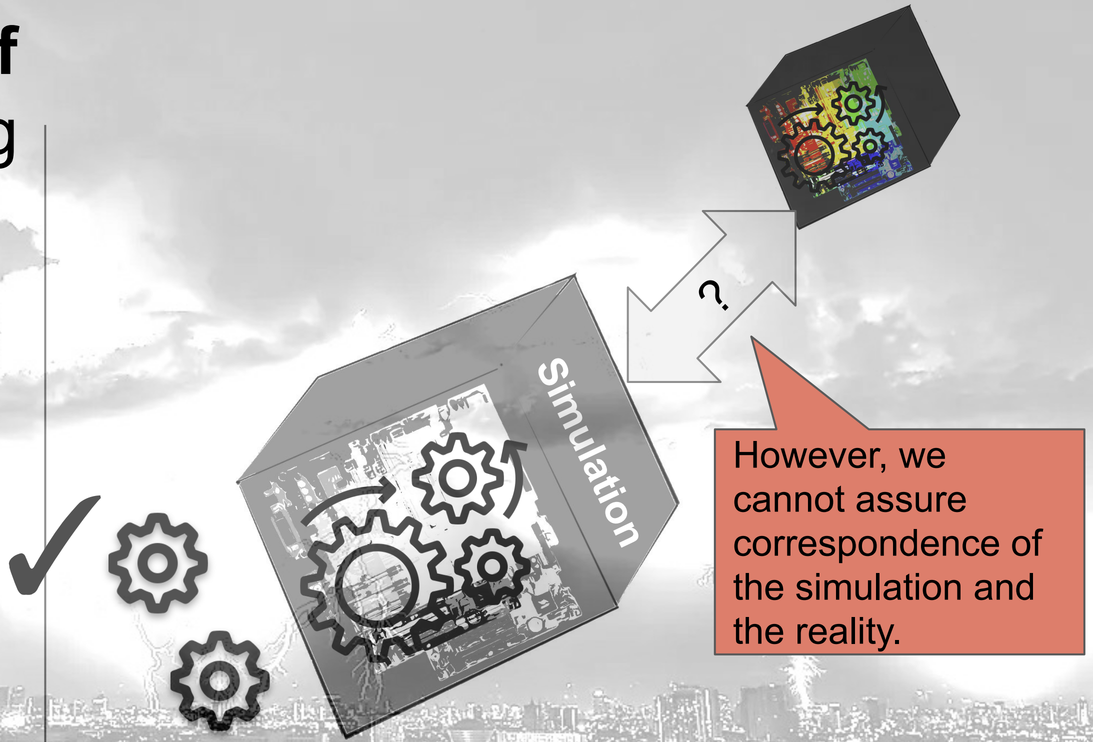


We have been doing this using ULAM and STAN.

The central problem of simulation-based testing



We have been doing this using ULAM and STAN.



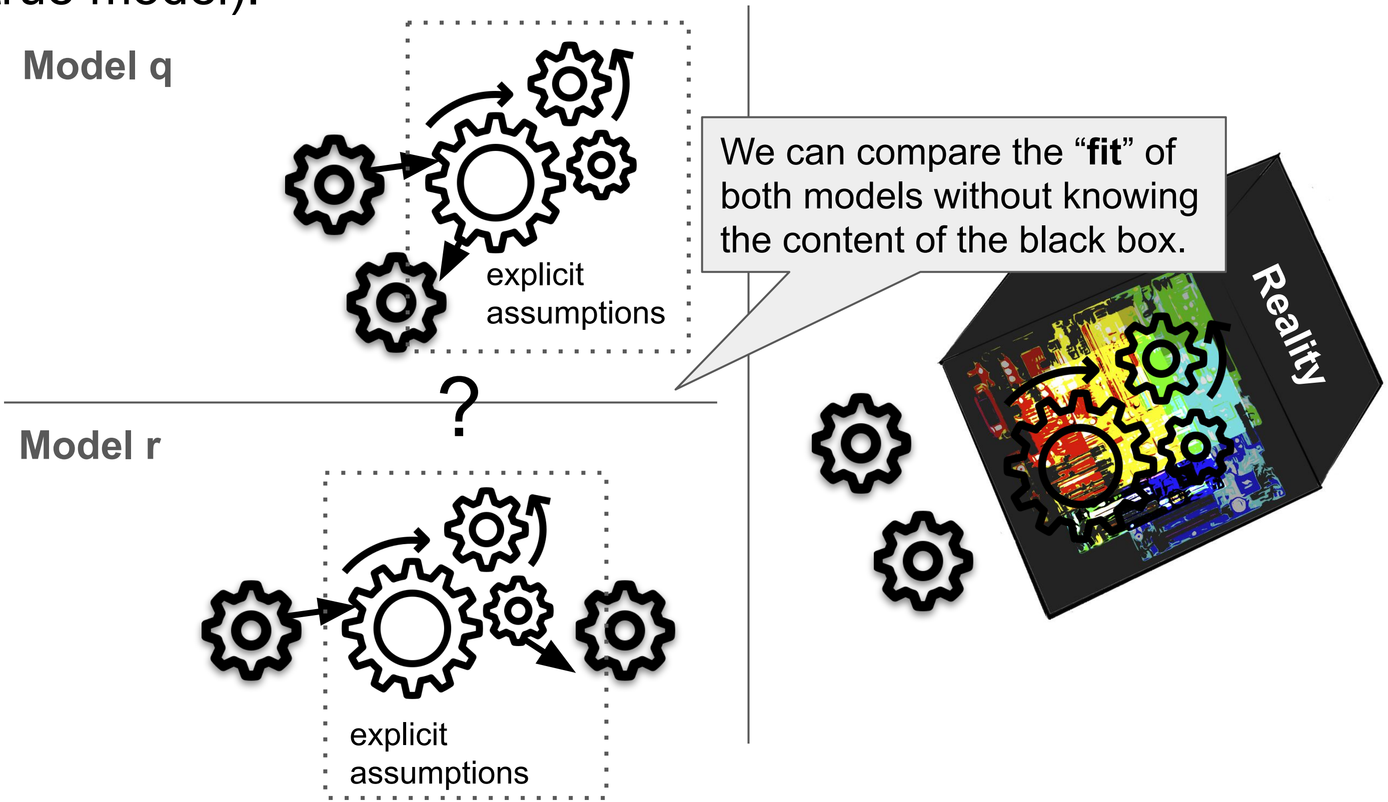
Benefits of simulation-based testing

- Simulation-based testing in a nutshell:
  - We **simulate** the (unknown and known) **variables** and the functions relating the variables.
  - We **model** the **known variables** and include unknown variables as parameters.
  - We **check** if the model comes to the **right conclusions**, e.g., infers the unknown variables as parameters.
- This testing method uncover weaknesses and limitations of the model, like **bugs, minimal amount of data (power analysis), false interpretation, or over-parameterization.**
- Simulation cannot check the correspondence between reality (true model) and our model.

Model comparison

The model comparison strategy:

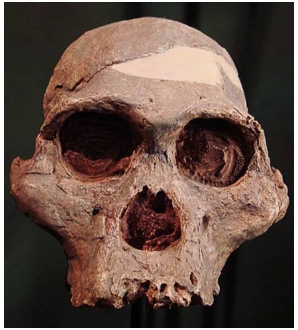
The model that fits the observed variables better, should be **closer to the reality** (true model).

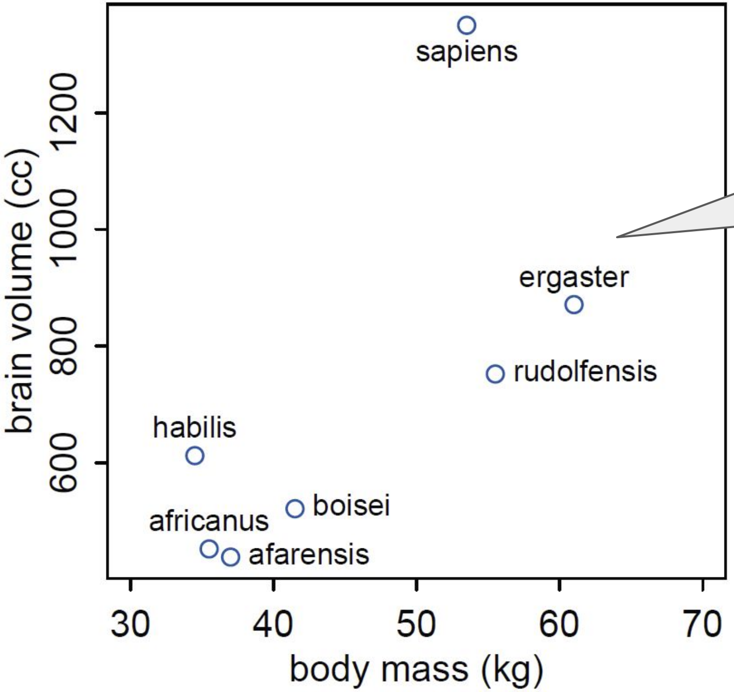


Example
(and some pitfalls)

Example: Examining the relation between body mass and brain volume (using polynomial regression)

We are **missing** the functional relation between both.





The **observed variable** *body mass* and *brain volume*.

(Copy [McElreath20])

One possible definition of the model's fit

In the context of linear (regression) models, one often uses R^2 to characterize the fit.

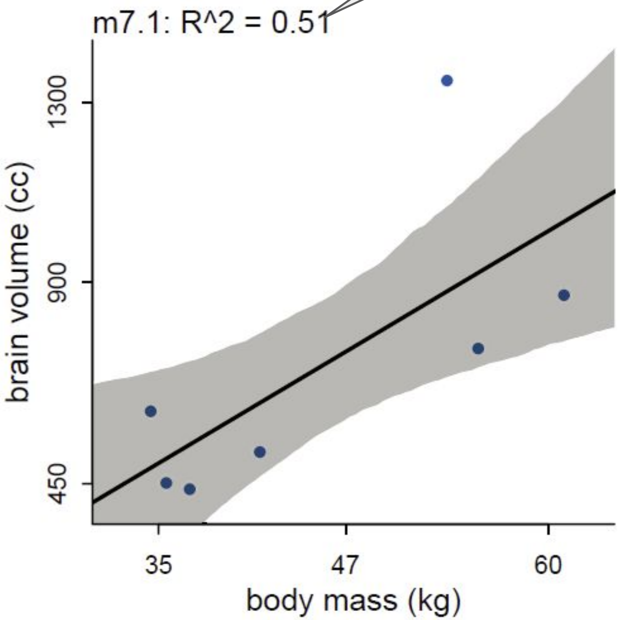
The residual is the difference between the outcome and the predicted outcome.

$$R^2 = \frac{\mathrm{var}(\mathrm{outcome}) - \mathrm{var}(\mathrm{residuals})}{\mathrm{var}(\mathrm{outcome})} = 1 - \frac{\mathrm{var}(\mathrm{residuals})}{\mathrm{var}(\mathrm{outcome})}$$

(Copy [McElreath20])

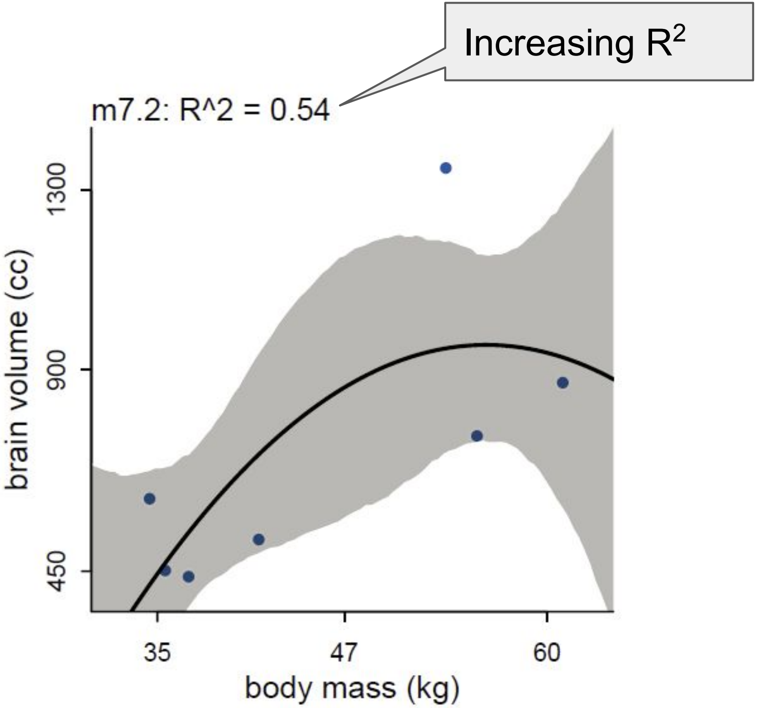
Learning too much: Polynomial regression (degree 1)

Fit R^2



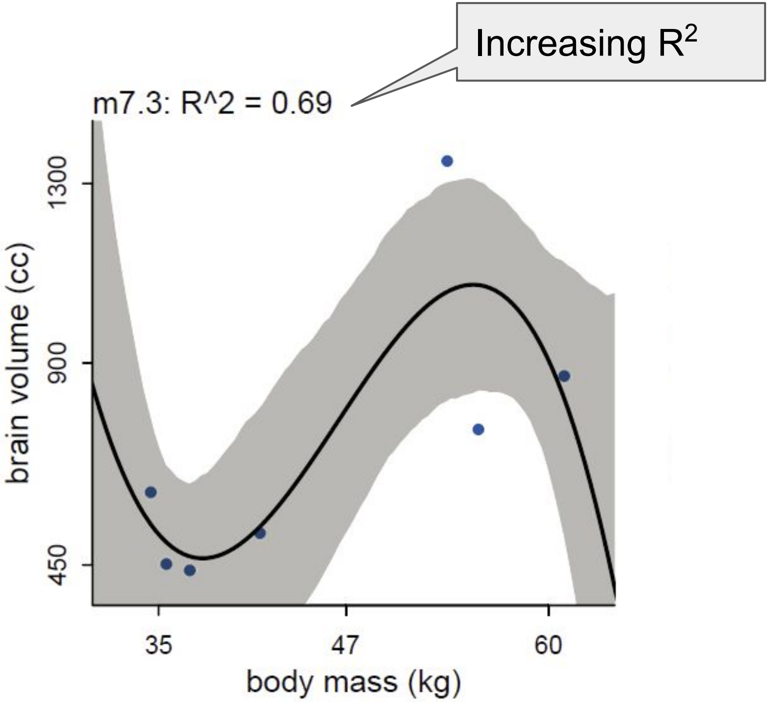
(Copy [McElreath20])

Learning too much: Polynomial regression (degree 2)



(Copy [McElreath20])

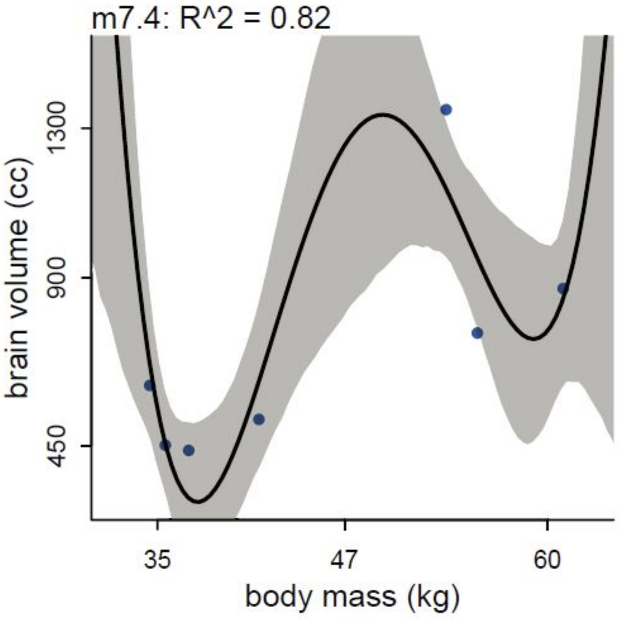
Learning too much: Polynomial regression (degree 4)



(Copy [McElreath20])

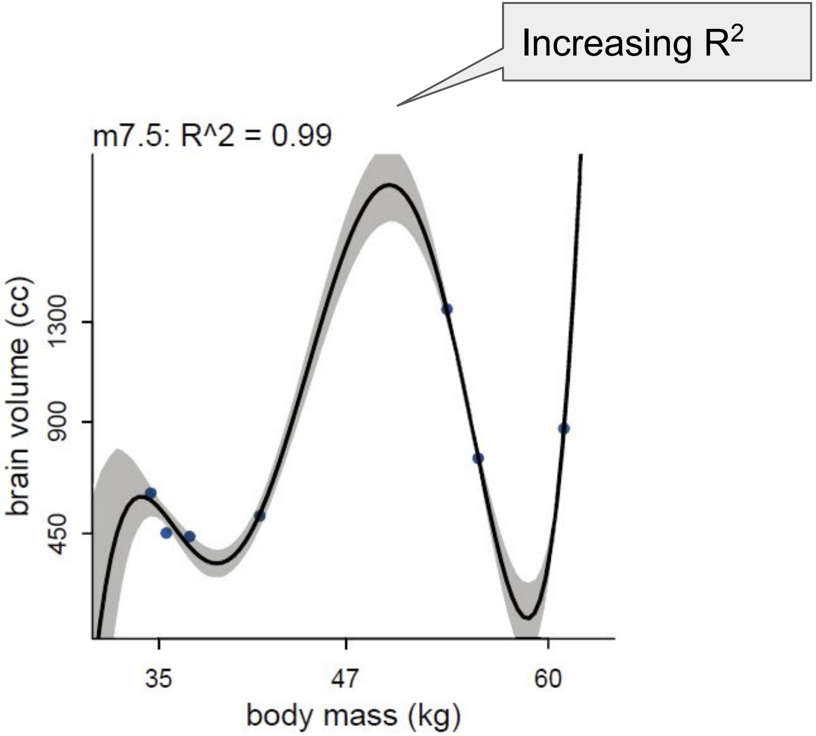
Learning too much: Polynomial regression (degree 4)

Increasing R^2



(Copy [McElreath20])

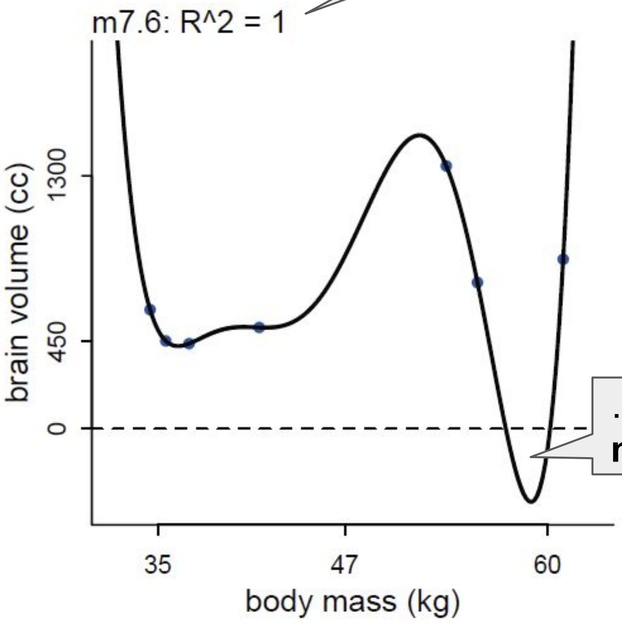
Learning too much: Polynomial regression (degree 5)



(Copy [McElreath20])

Learning too much: Polynomial regression (degree 6)

Perfect fit…



… but this model is **nonsense**.

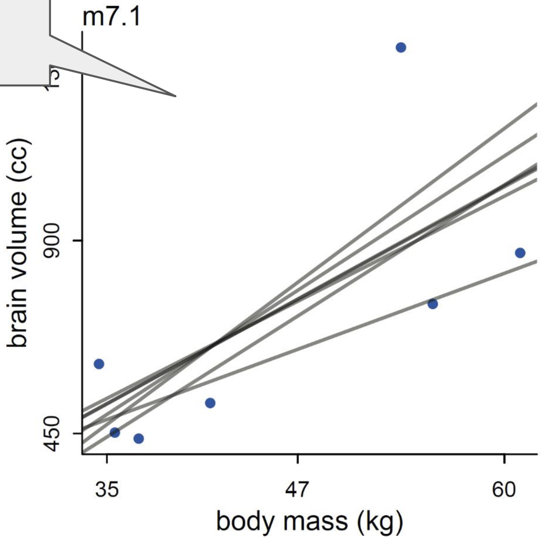
(Copy [McElreath20])

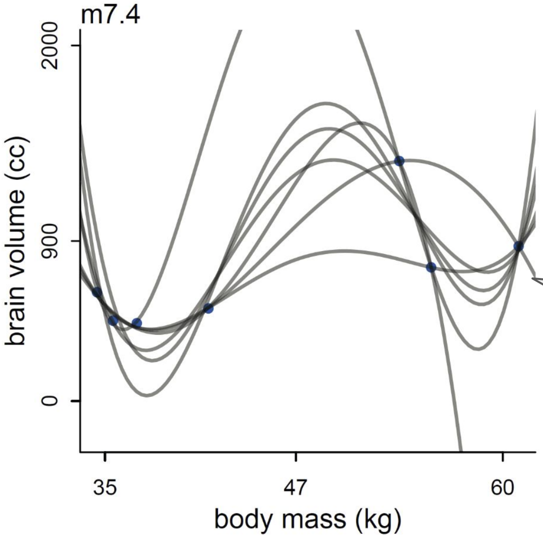
Learning too little: Polynomial regression

Remove one point and getting almost the same regression line means that we are insensitive to the sample.

Different regression lines are shown when removing single points from the sample.

Less sensitive to the sample.





Sensitive to the sample.

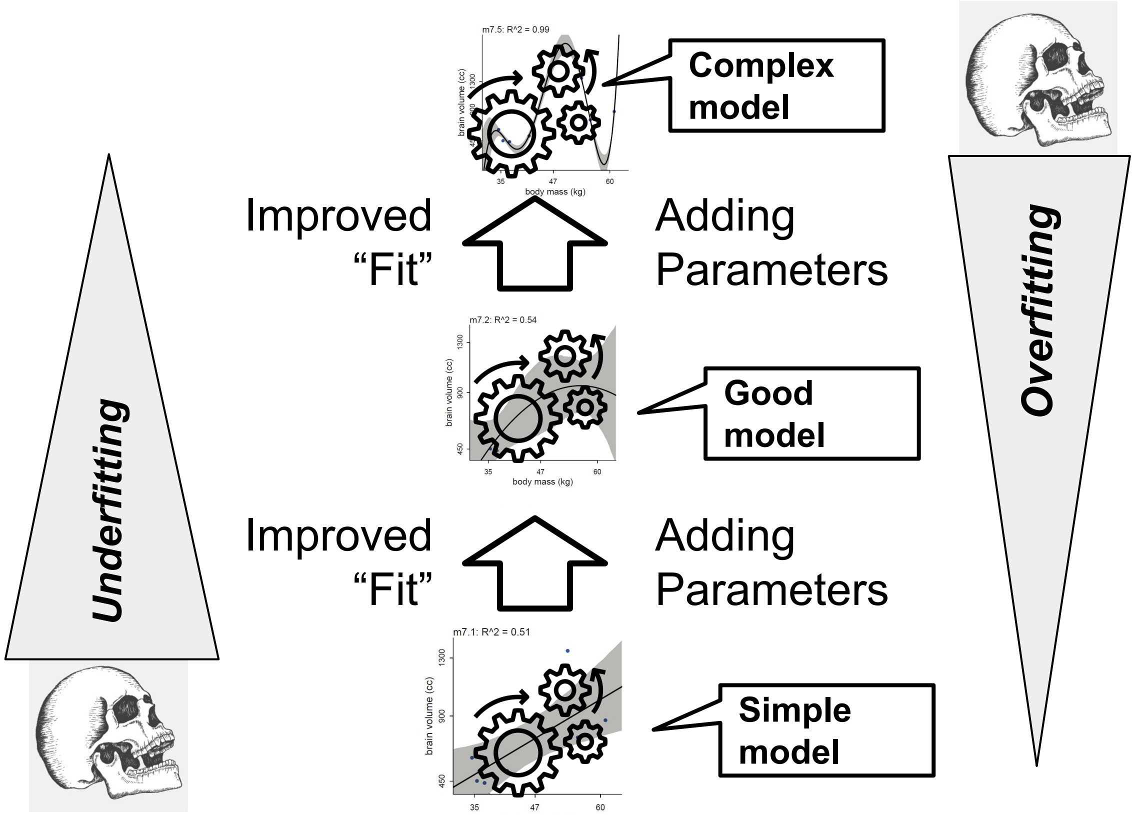
Learning too much: **overfitting**

Learning too little: **underfitting**

The balance between underfitting and overfitting is sometime called bias-variance trade-off (see [McElreath20] page 201).

A conceptual view

- **Adding parameters** to a model (almost) always improves the fit, even if parameters are meaningless (**overfitting)**.
- **Not adding parameters** (and variables) to a model may learn to little (**underfitting)**.
- **Navigating between both** is the major challenge of a model comparison.



Navigating between underfitting and overfitting

Navigating between underfitting and overfitting

1. We compare **different models** from simple to complex to prevent underfitting.

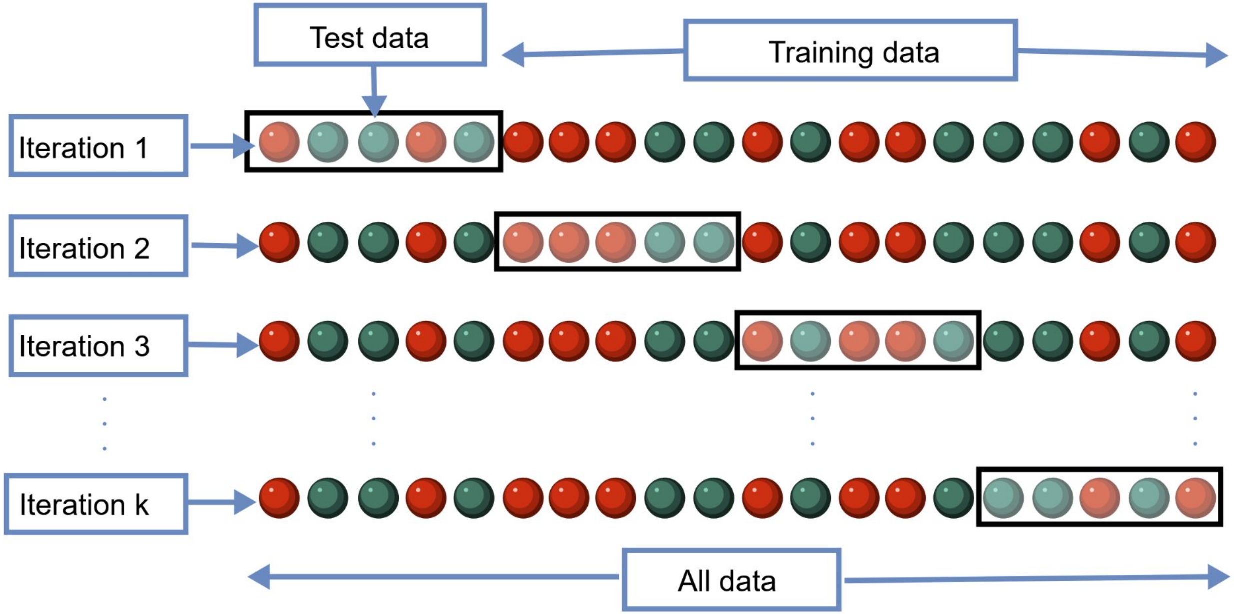
2. We evaluate the models **out-of-sample** to prevent overfitting.

    a. Cross-validation variants can be used:

        i. Training sample: used to fit the model

        ii. Test sample: used to evaluate the model using a criterion.

K-fold cross-validation variant



Source: https://en.wikipedia.org/wiki/Cross-validation_(statistics)

Demo

Simulating data where we have no effect of X2

```
N <- 100 # Number of observations.
# Observed variables.
X1 <- rnorm(N) # Normal distributed variable.
X2 <- rnorm(N)  # Normal distributed variable.

# Unobserved variables (later parameters in the model).
a <- 0.5
b1 <- 0.4
b2 <- 0.0 # No effect!
sigma <- 0.4

mu <- a + b1 * X1 + b2 * X2 # Linear model (same as mu <- a + b1 * X1 because b2 = 0)

# Observed output variables.
Y <- rnorm(N, mean = mu, sd = sigma) # Output variable.
```

Stan model 1 using X1

```
model1 <- stan(model_code = "
data{
 int<lower=0> N;
 vector[N] X1;
 vector[N] Y;
}
parameters{
 real a;
 real b1;
 real<lower=0> sigma;
}
model{
 Y ~ normal(a + b1 * X1 , sigma);
}
", data = list(X1 = X1, N = N, Y = Y), chains = 1)
```

Stan model 2 using X1 and X2

```
model2 <- stan(model_code = "
data{
 int<lower=0> N;
 vector[N] X1;
 vector[N] X2;
 vector[N] Y;
}
parameters{
 real a;
 real b1;
 real b2;
 real<lower=0> sigma;
}
model{
 Y ~ normal(a + b1 * X1 + b2 * X2, sigma);
}
", data = list(X1 = X1, X2 = X2, N = N, Y = Y), chains = 1)
```

Comparing the sum of the square errors (a criterion evaluating fit)

```r
# Extract samples from the posterior.
samples1 <- extract(model1)
samples2 <- extract(model2)

# We are not using the full posterior to predict (shame on us, don’t do this at home!!!)
Ypred1 <- mean(samples1$a) + mean(samples1$b1) * X1
Ypred2 <- mean(samples2$a) + mean(samples2$b1) * X1 + mean(samples2$b2) * X2

# Compute the sum of the square error.
error1 <- sum((Ypred1 - Y)^2) # ≈ 16.93
error2 <- sum((Ypred2 - Y)^2) # ≈ 16.89 (this error is smaller)
```

The overfitted model 2 is preferred.

Evaluating the model out-of-sample

Splitting the simulated data into test and train set

```
N <- 200 # Number of observations.

# … simulation …

# Splitting the data into test and train set.
Ytest <- Y[101:200] # Second 100 entries
Y <- Y[1:100] # First 100 entries

X1test <- X1[101:200]
X1 <- X1[1:100]

X2test <- X2[101:200]
X2 <- X2[1:100]
```

This is just a single split. You are supposed to fix this in the assignment.

Comparing the sum of the square errors on the test set

```r
# We are not using the full posterior (shame on us!!!)
Ypred1 <- mean(samples1$a) + mean(samples1$b1) * X1test
Ypred2 <- mean(samples2$a) + mean(samples2$b1) * X1test + mean(samples2$b2) * X2test

# Compute sum square error.
error1 <- sum((Ypred1 - Ytest)^2) # 15.45
error2 <- sum((Ypred2 - Ytest)^2) # 15.56 (error is bigger)
```

The overfitted model 2 is **NOT preferred**.

Additional concepts to prevent overfitting

- Regularization
  - Regularization **shrink the parameters towards zero** so that the model does not get over-excited about the data. This can be done by priors.
- Information criteria (AIC, WAIC, PSIS)
  - Cross-validation is expensive since it needs to i) split the data set several times, ii) fit a model and iii) evaluated it on the test set.
  - There are **theoretical estimates of the out-of-sample fit**, that can also be used, called information criteria.

Summary

- Simulation-based testing vs. model comparison.
- Leaning too much, learning to little (overfitting and underfitting).
- Navigating through different models using a criterion evaluated out-of-sample.
- Cross-validation