Overview

- Day 1:
  - Classical Statistics, Models Selection, Over- vs Underfitting
  - Linear Regression, Logistic Regression, LASSO
- Day 2:
  - Algorithmic modelling culture, Decision Trees, Pipelines
- Day 3:
  - Mini Symposium, Social Event
- Day 4:
  - Boosting, Bagging, Neural Networks and more

Day 1

Intro to R and RStudio

Classical Statistics and Over- vs Underfitting

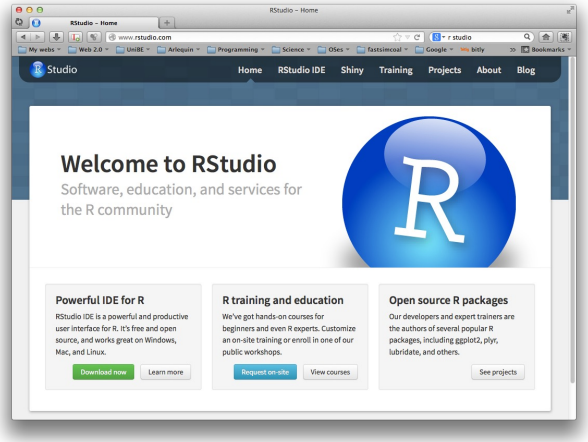
Bayesian Statistics

RStudio...

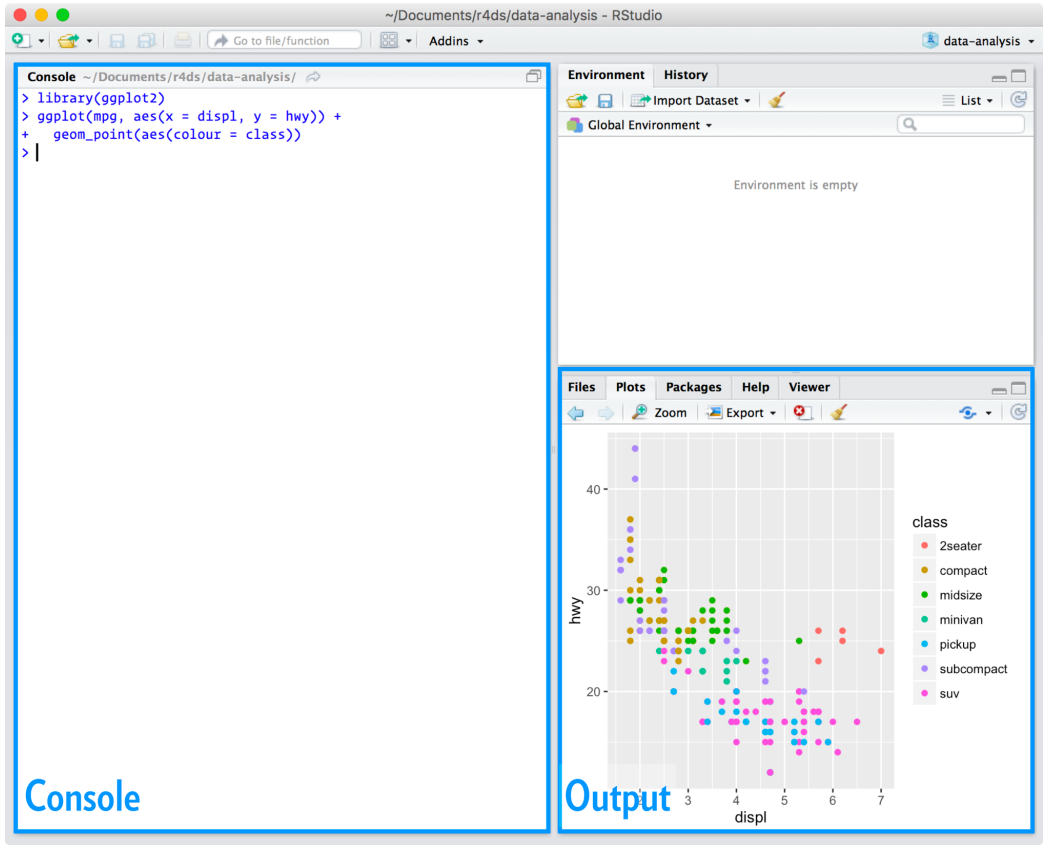
... and how to load data and visualize it

R-Studio

RStudio is an integrated development environment, or IDE, for R programming. Download and install it from http://www.rstudio.com/download.



When you start RStudio, you'll see two key regions in the interface:



GGplot 2

- Install Tidyverse:

> **install.packages**("tidyverse")

> library(tidyverse)

```
library(tidyverse)
#> ── Attaching packages ─────────────────────────────────── tidyverse 1.2.1 ──
#> ✔ ggplot2 3.1.0.9000     ✔ purrr   0.2.5
#> ✔ tibble  2.0.0          ✔ dplyr   0.7.8
#> ✔ tidyr   0.8.2          ✔ stringr 1.3.1
#> ✔ readr   1.3.1          ✔ forcats 0.3.0
#> ── Conflicts ────────────────────────────────────── tidyverse_conflicts() ──
#> ✖ dplyr::filter() masks stats::filter()
#> ✖ dplyr::lag()    masks stats::lag()
```

- Tidyverse is a collection of R packages (= additional features)
- Packages can be installed and loaded in R Studio, or directly in the console
- We will sometimes use ggplot2 (package for creating figures) but also other features

Data frames in R

- A "data frame" in R is the format in which data sets are organized (more modern version: tibble, for us the difference is not relevant)

- A matrix with variables (in the columns) and individuals (in the columns).

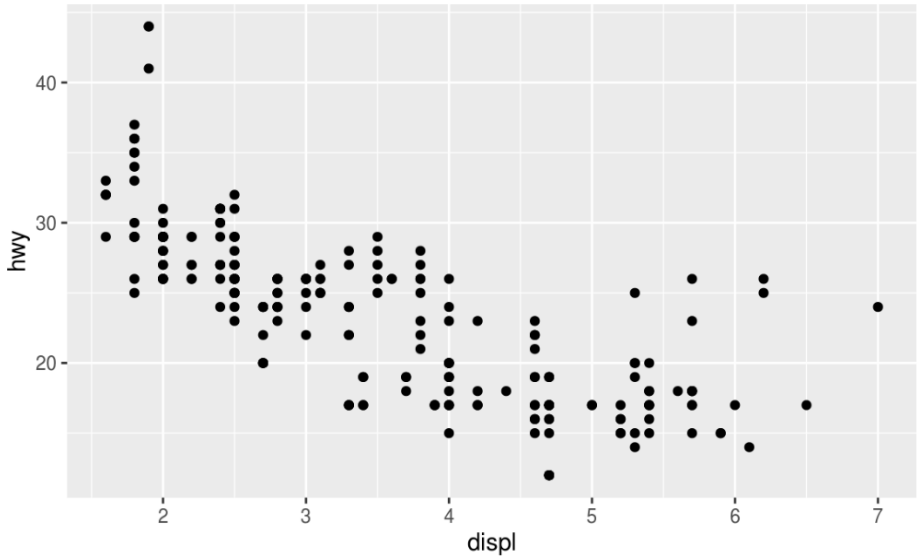
- Example:
  The data set mpg contains information on the fuel consumption of 38 car models, collected by the US Environmental Protection Agency

```
mpg
#> # A tibble: 234 x 11
#>   manufacturer model displ  year   cyl trans  drv     cty   hwy fl    class
#>   <chr>        <chr> <dbl> <int> <int> <chr>  <chr> <int> <int> <chr> <chr>
#> 1 audi         a4      1.8  1999     4 auto(… f        18    29 p     comp…
#> 2 audi         a4      1.8  1999     4 manua… f        21    29 p     comp…
#> 3 audi         a4      2    2008     4 manua… f        20    31 p     comp…
#> 4 audi         a4      2    2008     4 auto(… f        21    30 p     comp…
#> 5 audi         a4      2.8  1999     6 auto(… f        16    26 p     comp…
#> 6 audi         a4      2.8  1999     6 manua… f        18    26 p     comp…
#> # … with 228 more rows
```

GGplot

- Features that allow us to create high-quality graphics quickly and efficiently
- Designed to work well with data frames
- Very flexible and very easy (beware: sometimes "too" easy!)

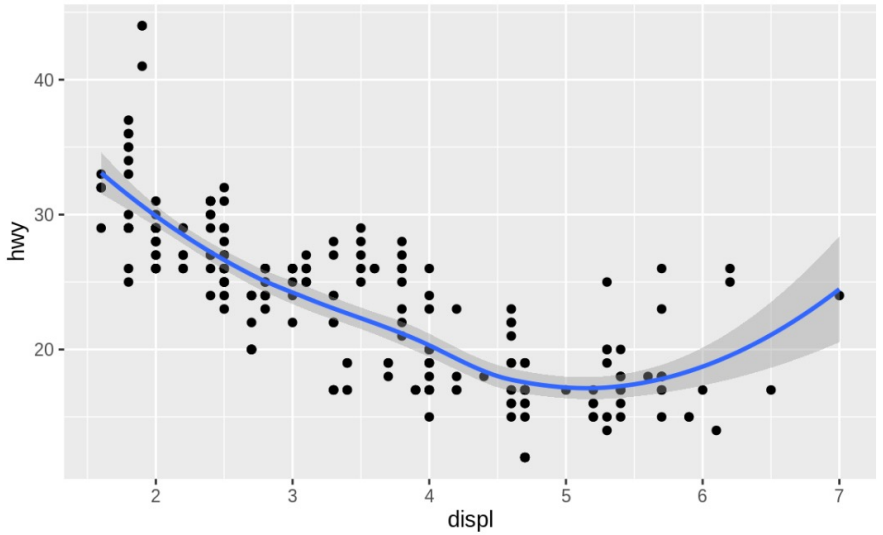
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



GGplot

- Features that allow us to create high-quality graphics quickly and efficiently
- Designed to work well with data frames
- Very flexible and very easy (beware: sometimes "too" easy!)

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```



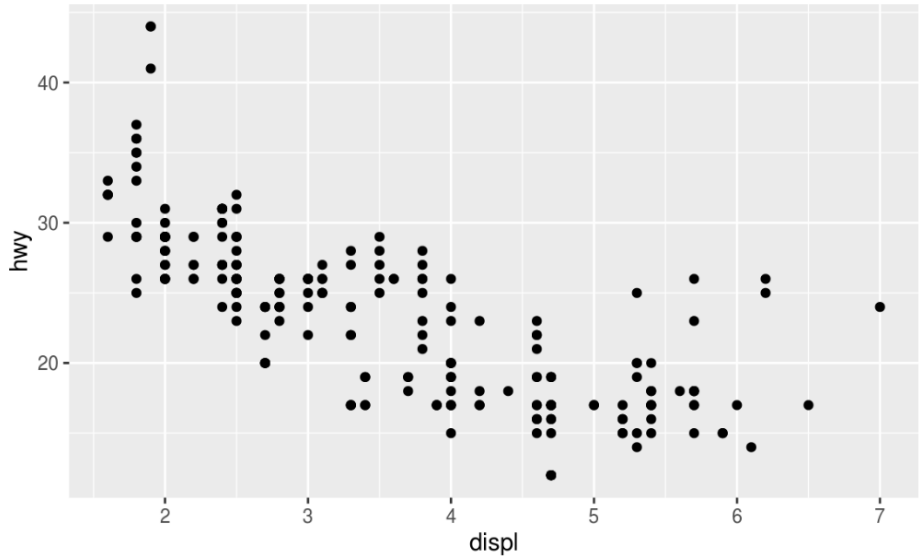
The ggplot grammar

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

- ggplot() produces an empty image
- <DATA> is the name of the dataframe to be used
- <GEOM_FUNCTION> adds the illustration
- mapping specify which variables to use

The ggplot grammar

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```



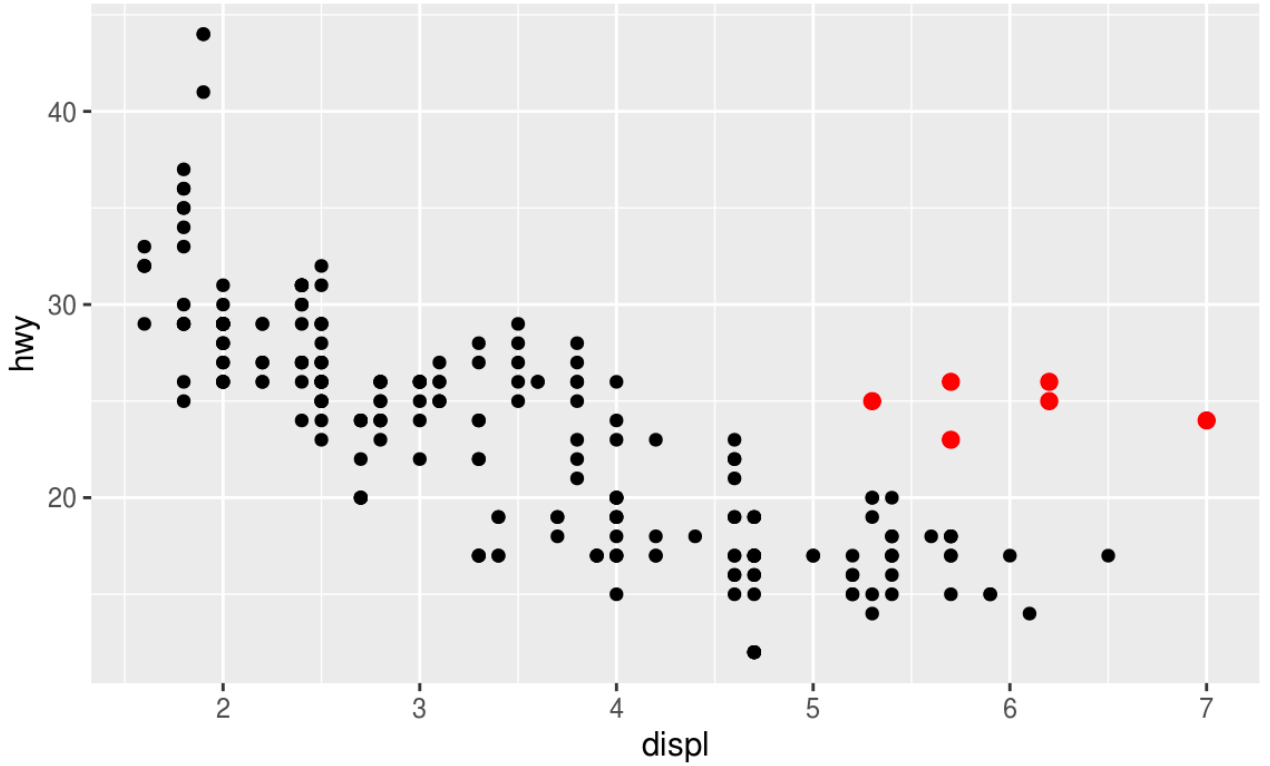
- We use the MPG dataset
- geom_point creates a scatterplot
- The x-axis is supposed to display the variable displ
- The y-axis is supposed to be the variable hwy

Aesthetic mappings

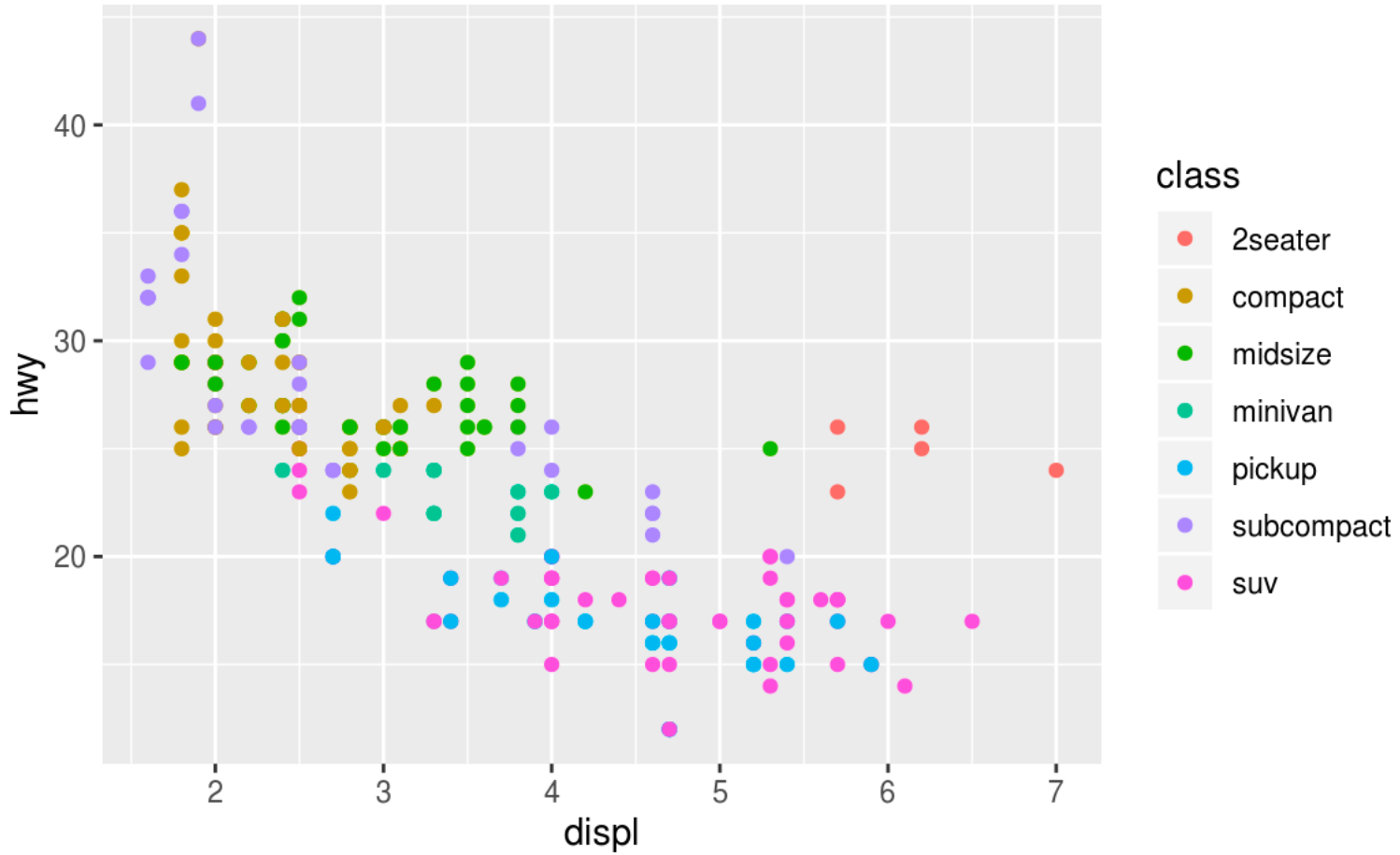
Aesthetic mappings can also include other variables

Question:

The red dots look like outliers, how can we take a closer look at them?



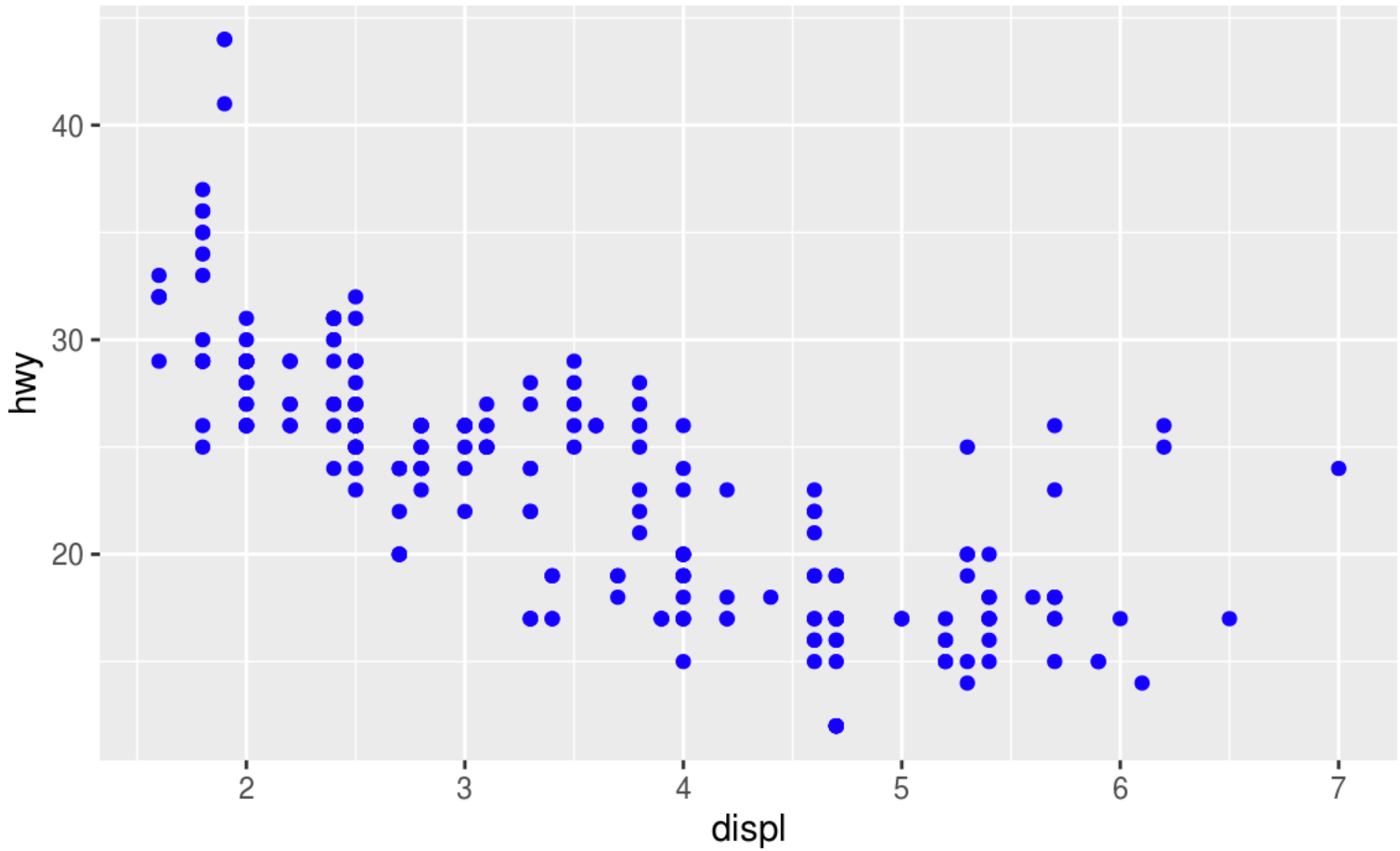
```r
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



We use the variable class to color the points accordingly.

We see that the red dots from earlier are mainly two-seaters.

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy), color = "blue")
```



Load Data

R ist flexible und kann (fast) jedes gängige Format einlesen:

```
>read.table("data.txt",header = FALSE,sep = " ", na.strings = NA)
```

Parameter	Values
file	String with filename, e.g., "dataset.txt"
header	TRUE/FALSE, indicates if first row in data file are names of columns
sep	String that indicates how the elements are separated (" ", "\t", ";", etc.)
na.strings	String that is used for missing data, e.g., "NA", "?", etc.

Load Data

There are also features for special formats:

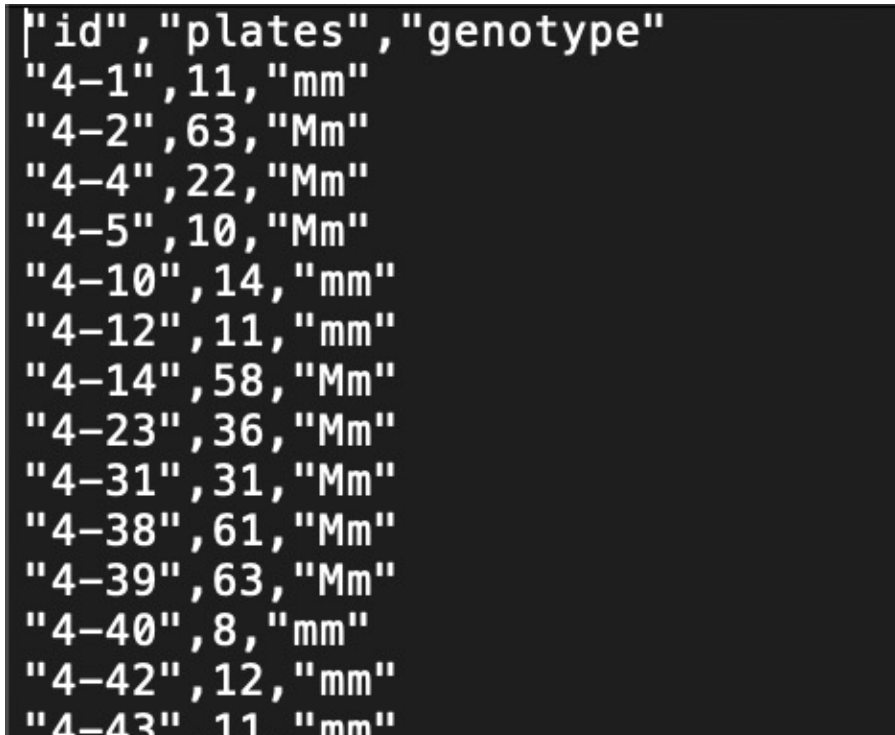
```
>stickleback =
read.csv("chap03e3SticklebackPlates.csv")
```

CSV = comma separated values

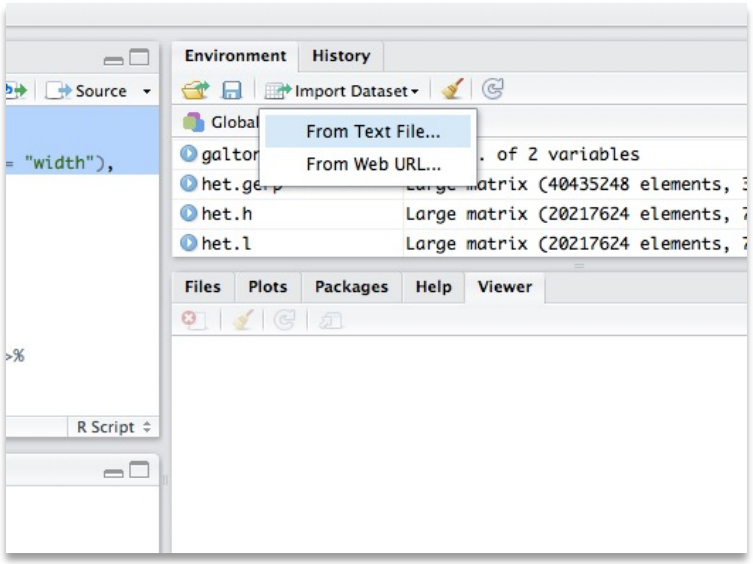
First Line: the names of the variables

After each variable comes a " , " (comma)

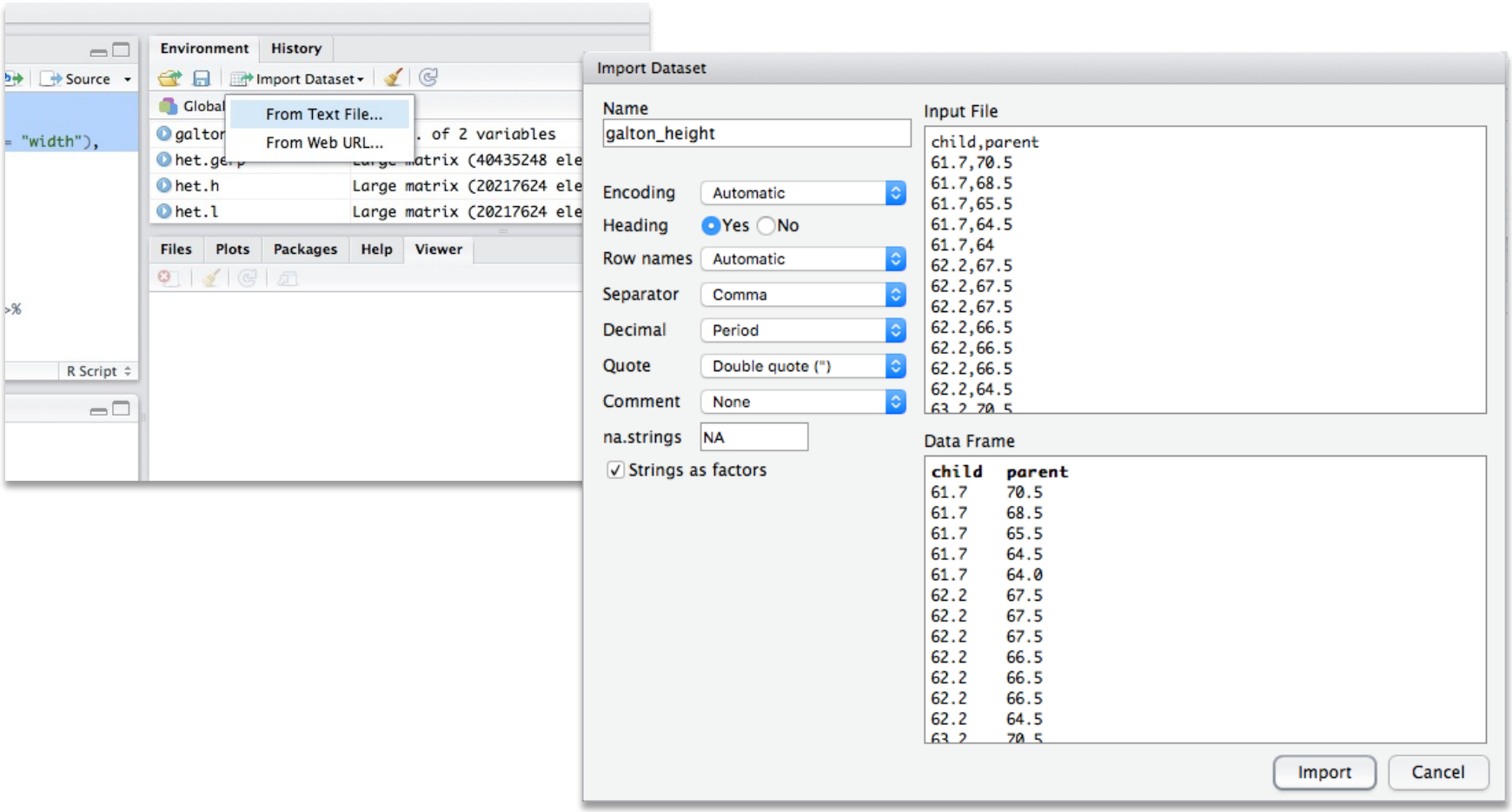
Each other line is an observation (or an individual)



Read Dataset with R Studio



Read Dataset with R Studio



References

Online Resourcen:

- Butler, M.A. (2009) Getting started in R for Biologists http://www2.hawaii.edu/~mbutler/Rquickstart/simpleR.pdf
- Paradis, E. (2005) R for Beginners. http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf
- R Development Core Team: The R Manuals. http://www.r-project.org/
- Venables, WN et al (2013) An Introduction to R http://cran.r-project.org/doc/manuals/R-intro.pdf
- Vernazzi, J. (2002) simpleR – Using R for introductory statistics http://cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf
- Yakir, B (2011) Introduction to Statistical Thinking (With R, Without Calculus) http://pluto.huji.ac.il/~msby/StatThink/

Further Reading:

- Burns, P. (2011) R inferno http://www.burns-stat.com/pages/Tutor/R_inferno.pdf
- Kabacoff, R. I. (2012) Quick R – Accessing the power of R http://www.statmethods.net/
- Schluter, D and Veen, T (2013) R workshops https://www.zoology.ubc.ca/~bio501/R/workshops/

Classical Statistics …

… and the problem of over- vs underfitting

Classical statistics

- Key Idea: (simple) model shape is assumed a priori
- Advantage: simple models, good interpretability, no black box
- Fit is evaluated based on how well the data fits the model assumptions
- Disadvantages: not very flexible, assumptions often not justified, often limited to few explanatory variables (# variables << sample size)
- Two examples:
  - Linear regression
  - Logistic regression

Linear Regression

Slope

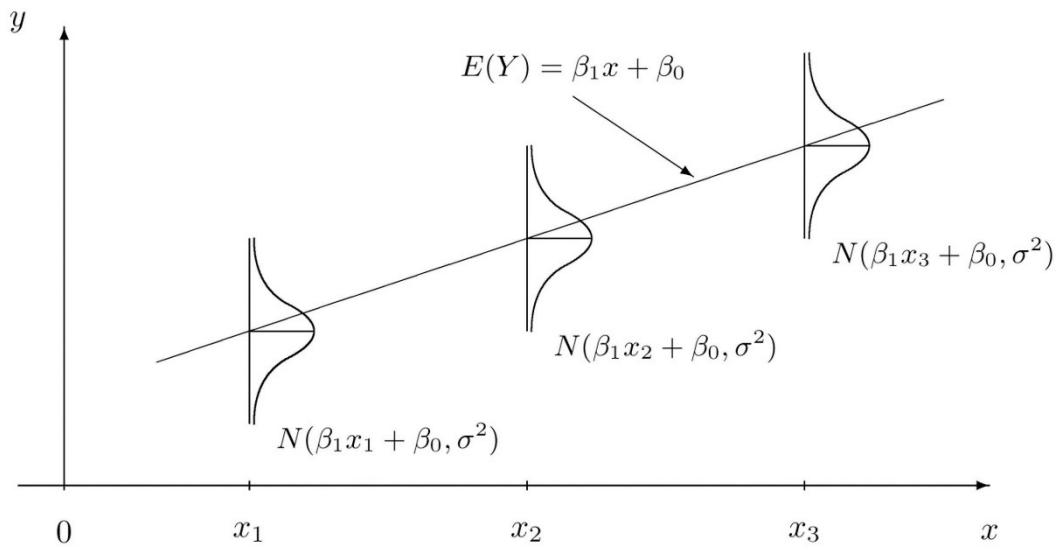
$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Response Variable

Intercept

Explanatory variable

Error term, N(0, s2) distributed



Linear Regression

Slope

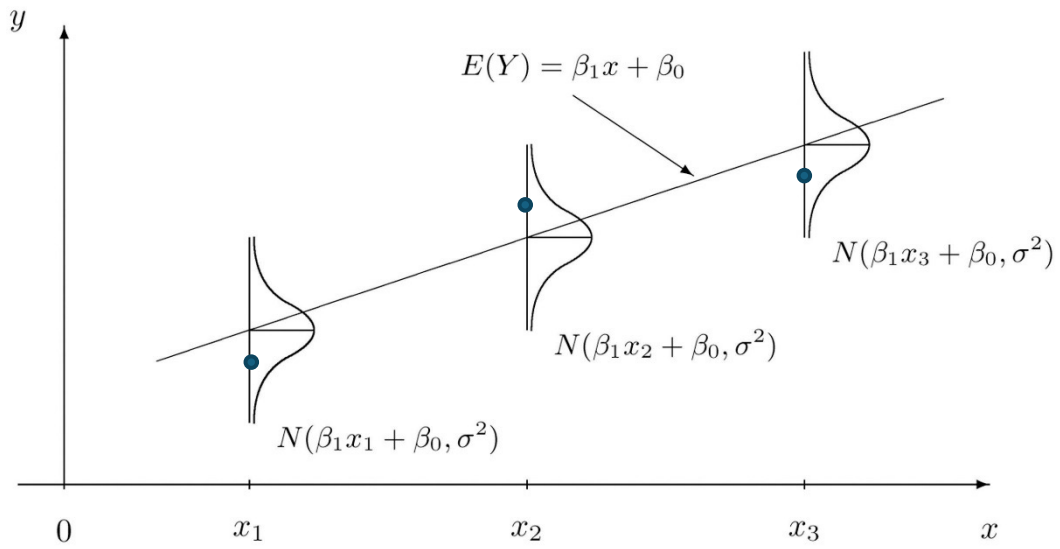
$$Y_i = \beta_0 + \beta_1 X_i + E_i$$

Intercept

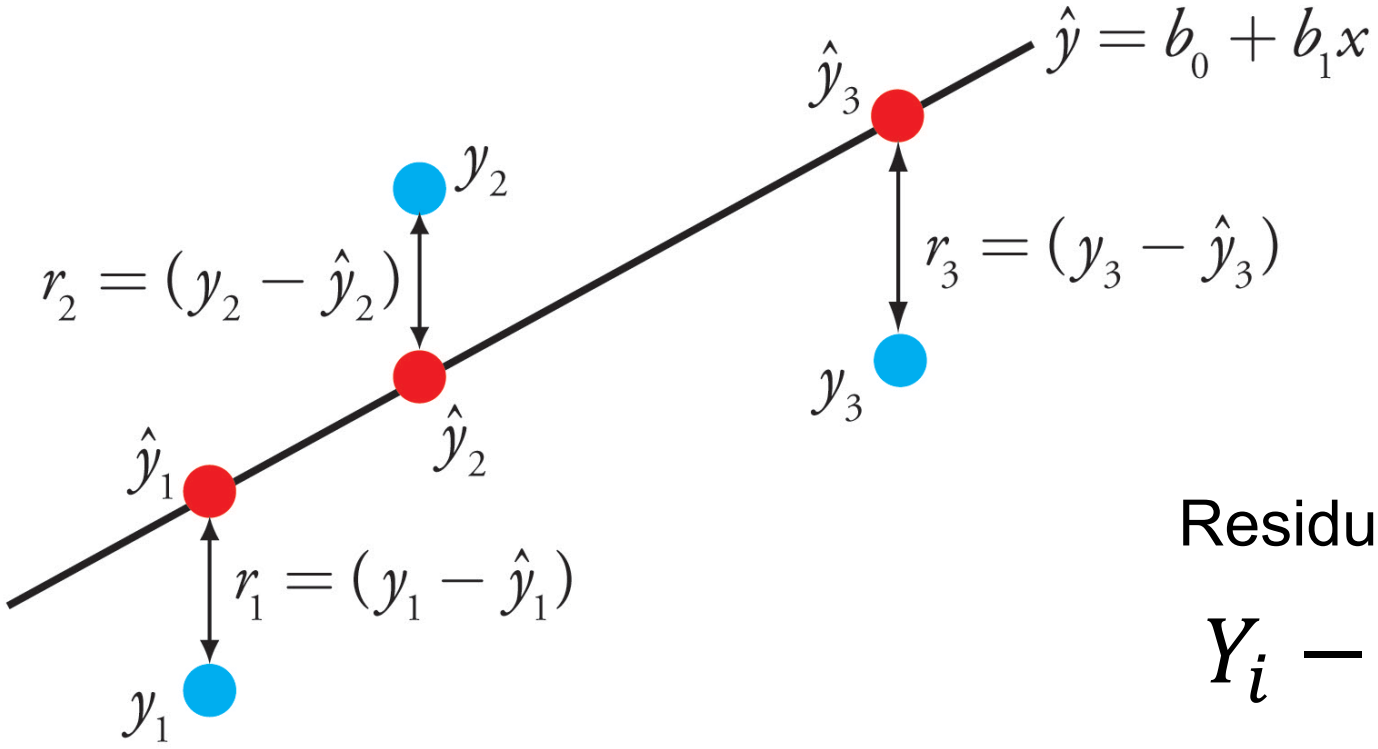
Explanatory variable

Error term, N(0, s2) distributed

Response Variable



Residuals



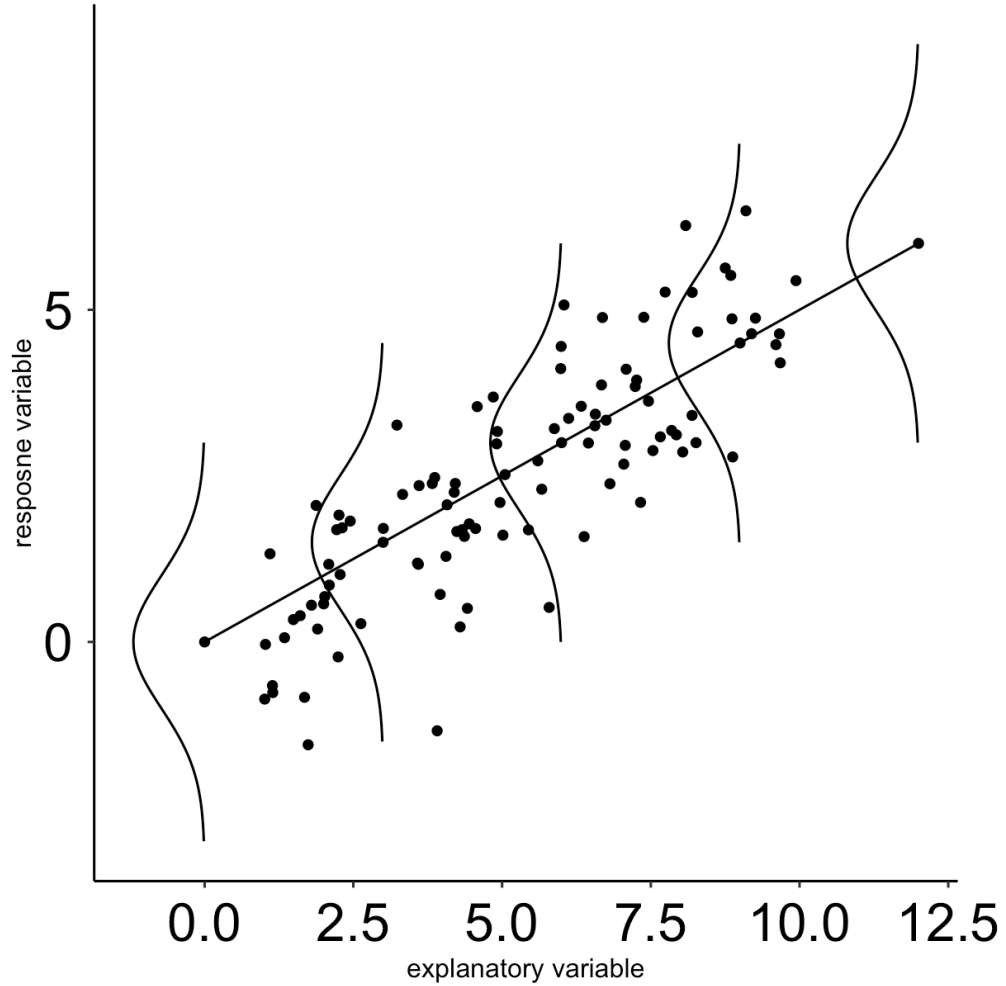
Residuals:

$$Y_i - \widehat{Y}_i$$

$\widehat{Y}_i$ is the predicted value of Y_i

$$\widehat{Y}_i = \beta_0 + \beta_1 X_i$$

Linear Regression

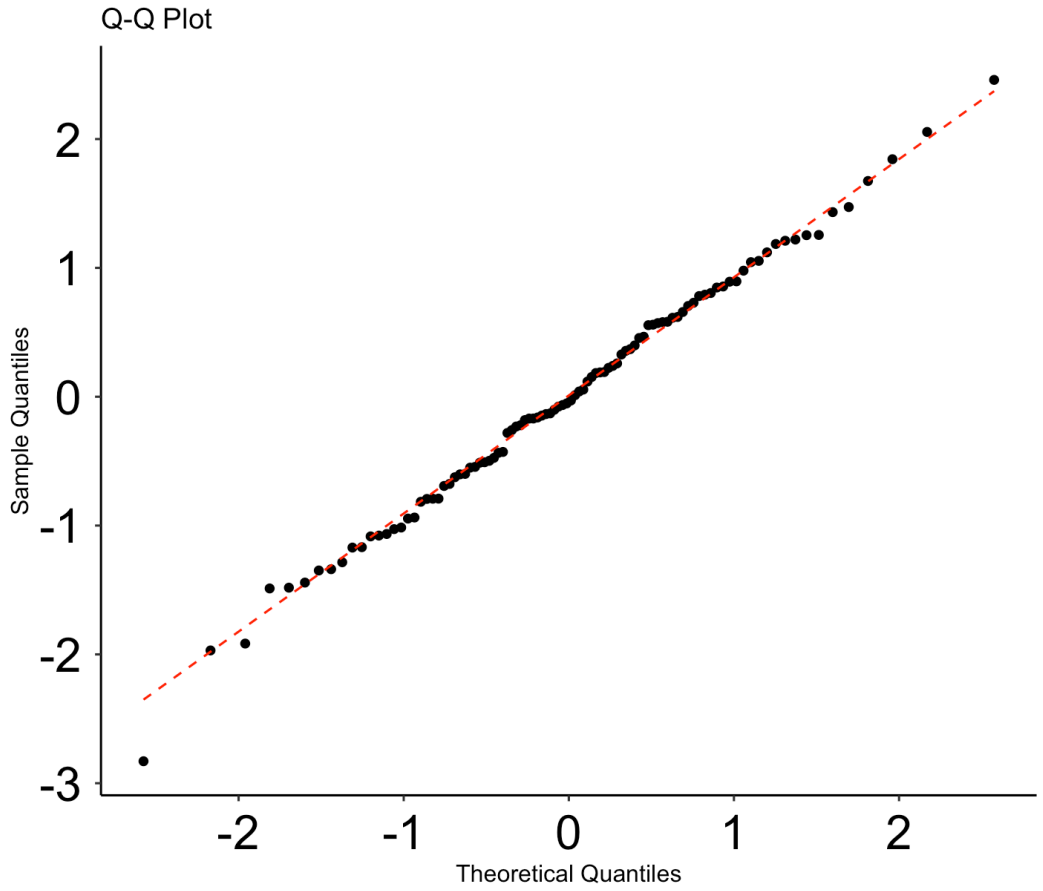


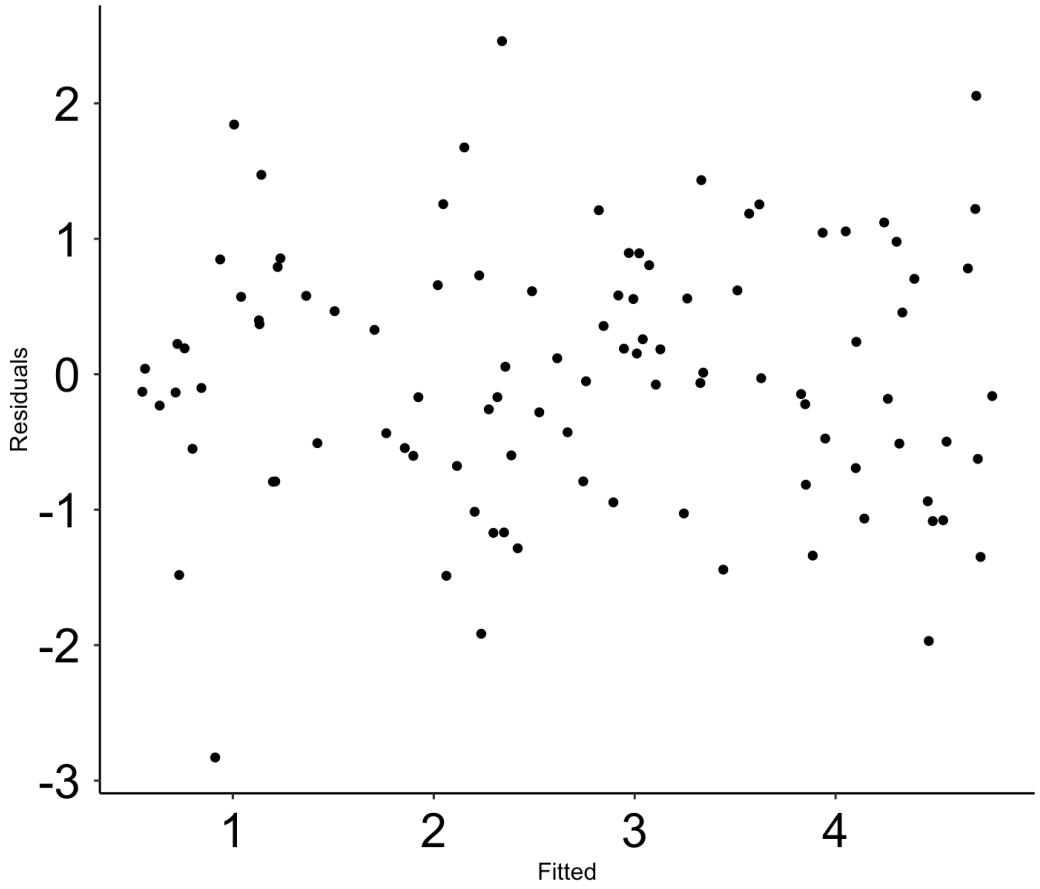
- Linear relationship between x (explanatory variable) and y (response variable)
- Normally distributed deviations from straight line
- $Y \sim \mathcal{N}(\beta_0 + \beta_1\, x, \sigma)$
- $Y = \beta_0 + \beta_1\, x + \mathrm{E}$, where $\mathrm{E} \sim \mathcal{N}(0, \sigma)$
- Strong assumptions, easy to fit

Least Squares / Likelihood

- Parameters are estimated by minimizing the sum of squared residuals
- We find the line that is “closest” to the points
- This is equivalent to maximizing the likelihood!
- Frequentist statistics:
  we infer parameters of the underlying distribution / model

Check Assumptions

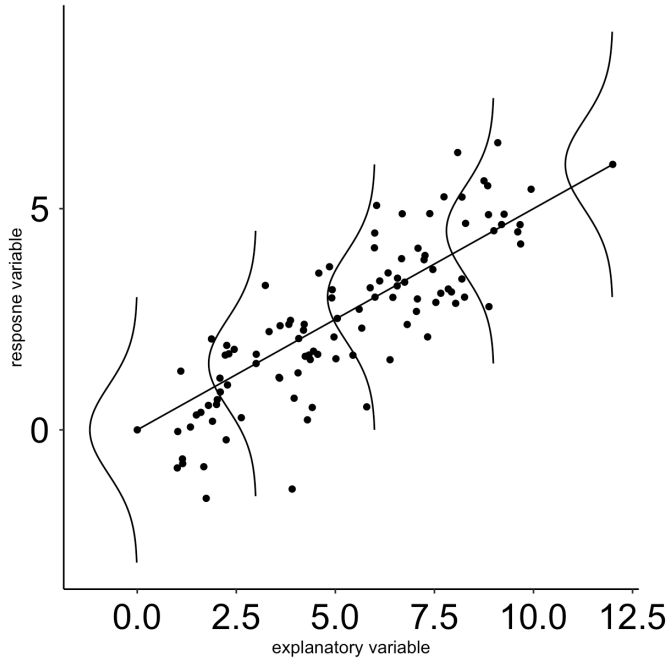




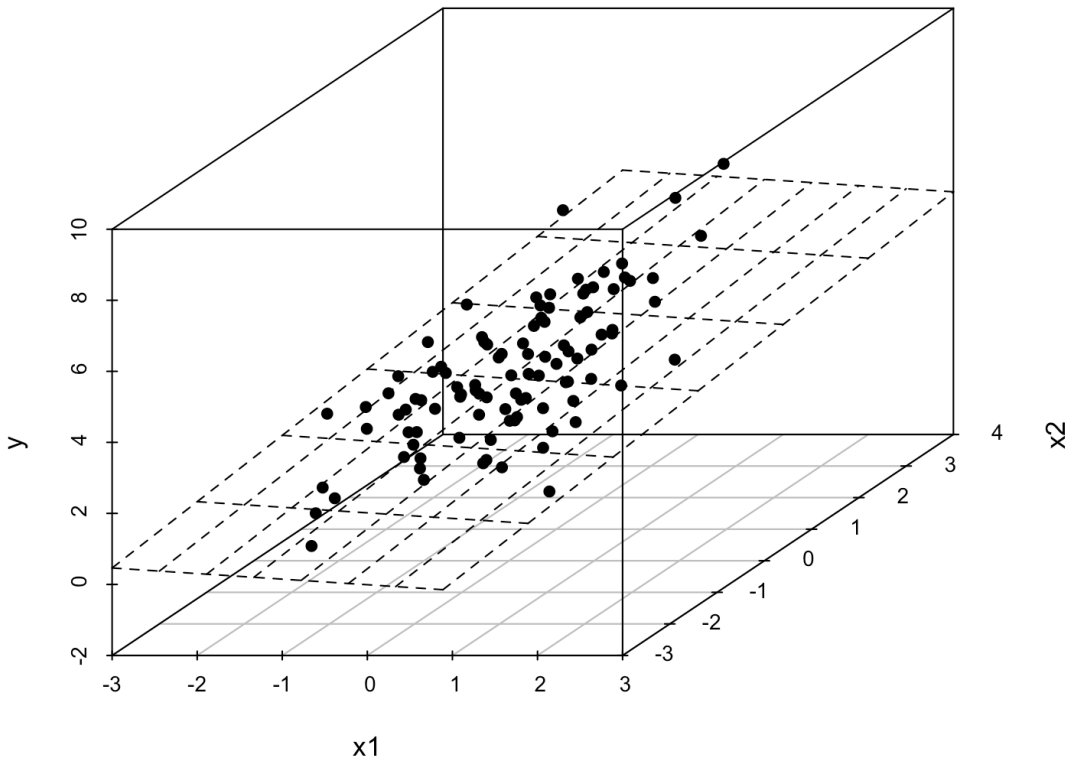
Pros and Cons

- Easy to fit
- Simple relationship and interpretation
- Can easily be generalized

- Strong assumptions
  - Linear realtionship
  - Normal residuals
  - Heteroskedasticity

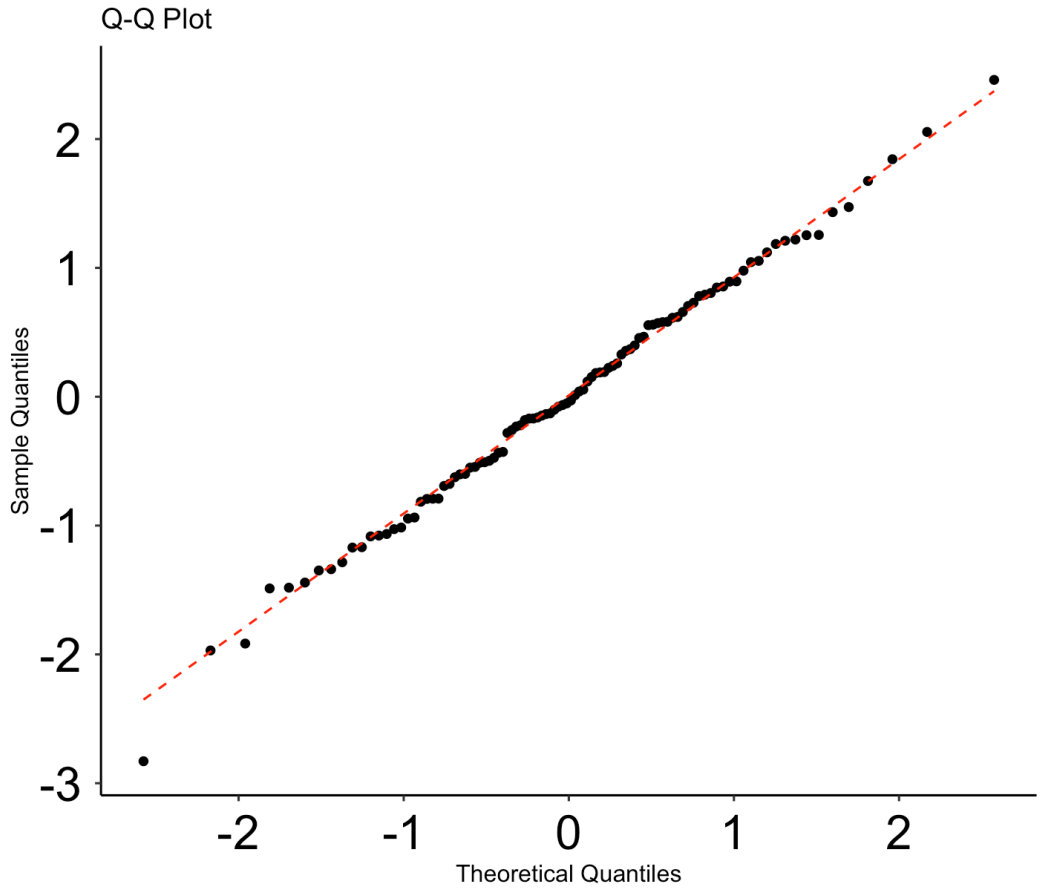


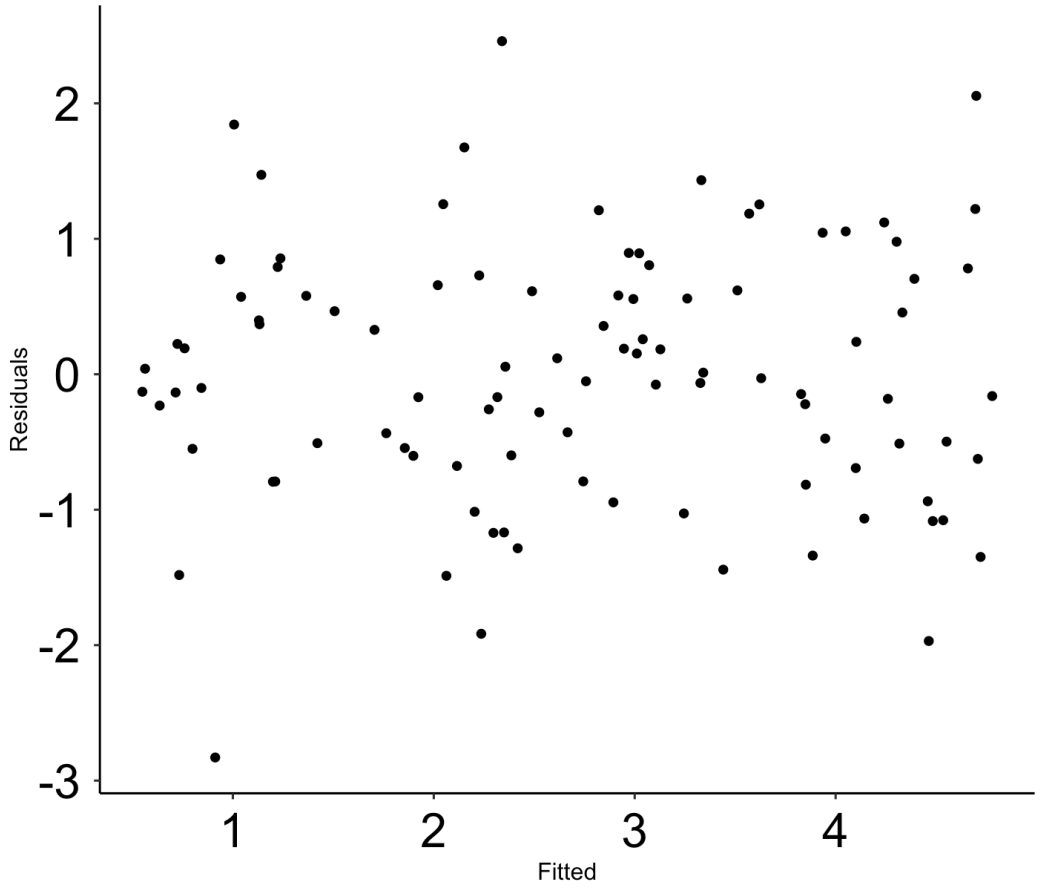
Multiple Linear Regression



- Linear relationship between x_i (explanatory variable i) and y (response variable)
- Normally distributed deviations from straight line
- $Y \sim \mathcal{N}(\beta_0 + \beta_1 x_1 + \ldots \beta_p x_p, \sigma)$
- $Y = \beta_0 + \beta_1 x_1 + \ldots \beta_p x_p + \mathrm{E}$, where $\mathrm{E} \sim \mathcal{N}(0, \sigma)$

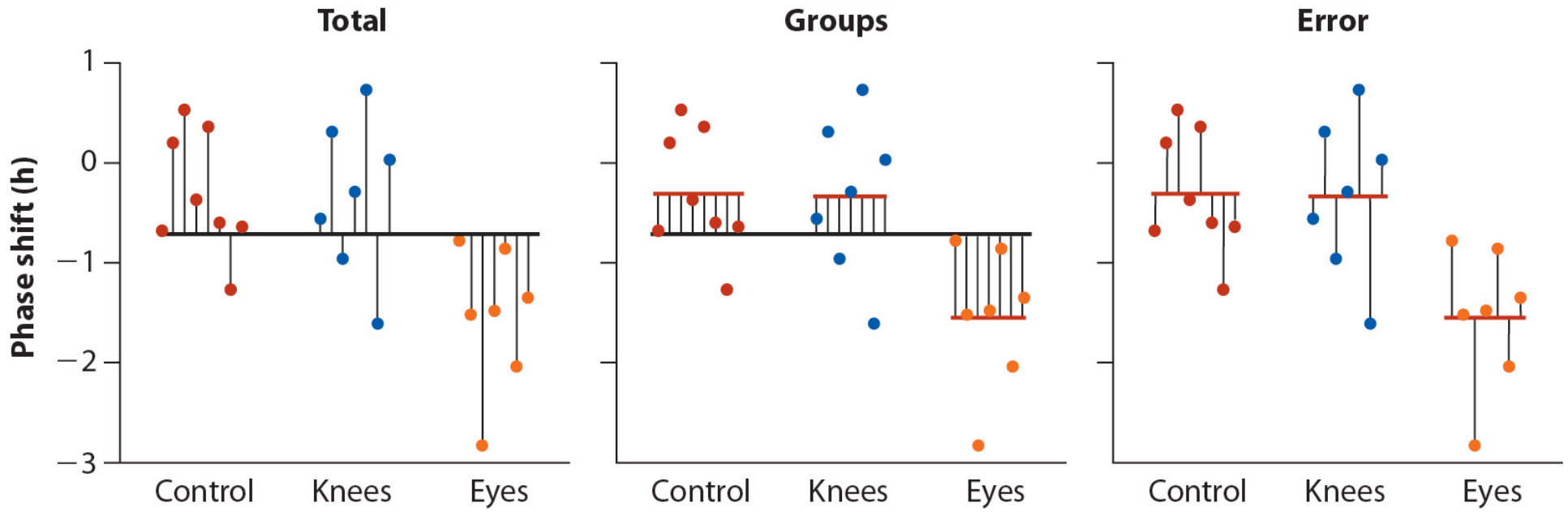
Check Assumptions





Remark:

- ANOVA is a special case of multiple linear regression
- We can therefore mix continuous and categorical variables within the same framework!



Example 1: multiple linear regression

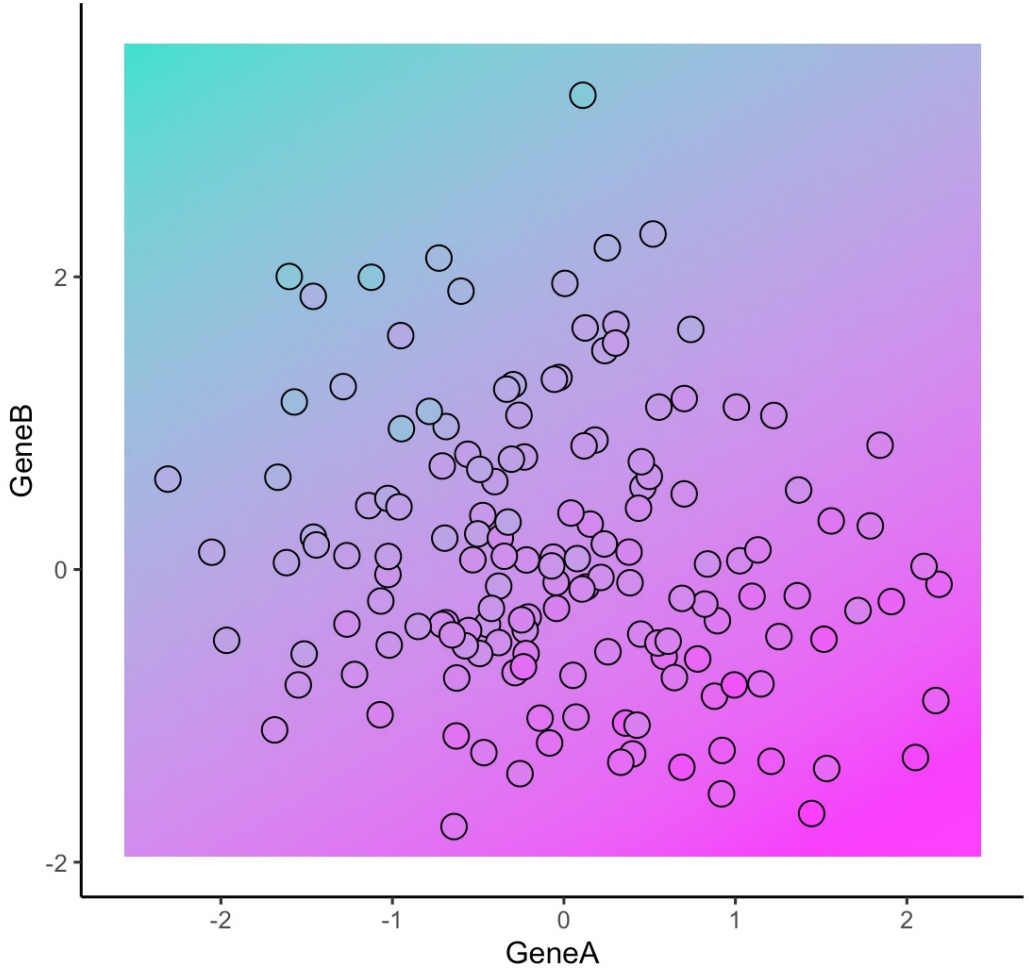
- We use a synthetic dataset for illustrative purposes. This has the advantage that we know exactly what's going on.
- In this simulated dataset, GeneA and GeneB are genes that directly influence a specific biological trait.
- GeneC and GeneD are noise genes that are correlated with the trait but do not contribute causally.
- GeneJ is in strongly correlated to GeneA but not causally linked to the trait.
- Other genes act as irrelevant noise variables or have correlations among themselves.

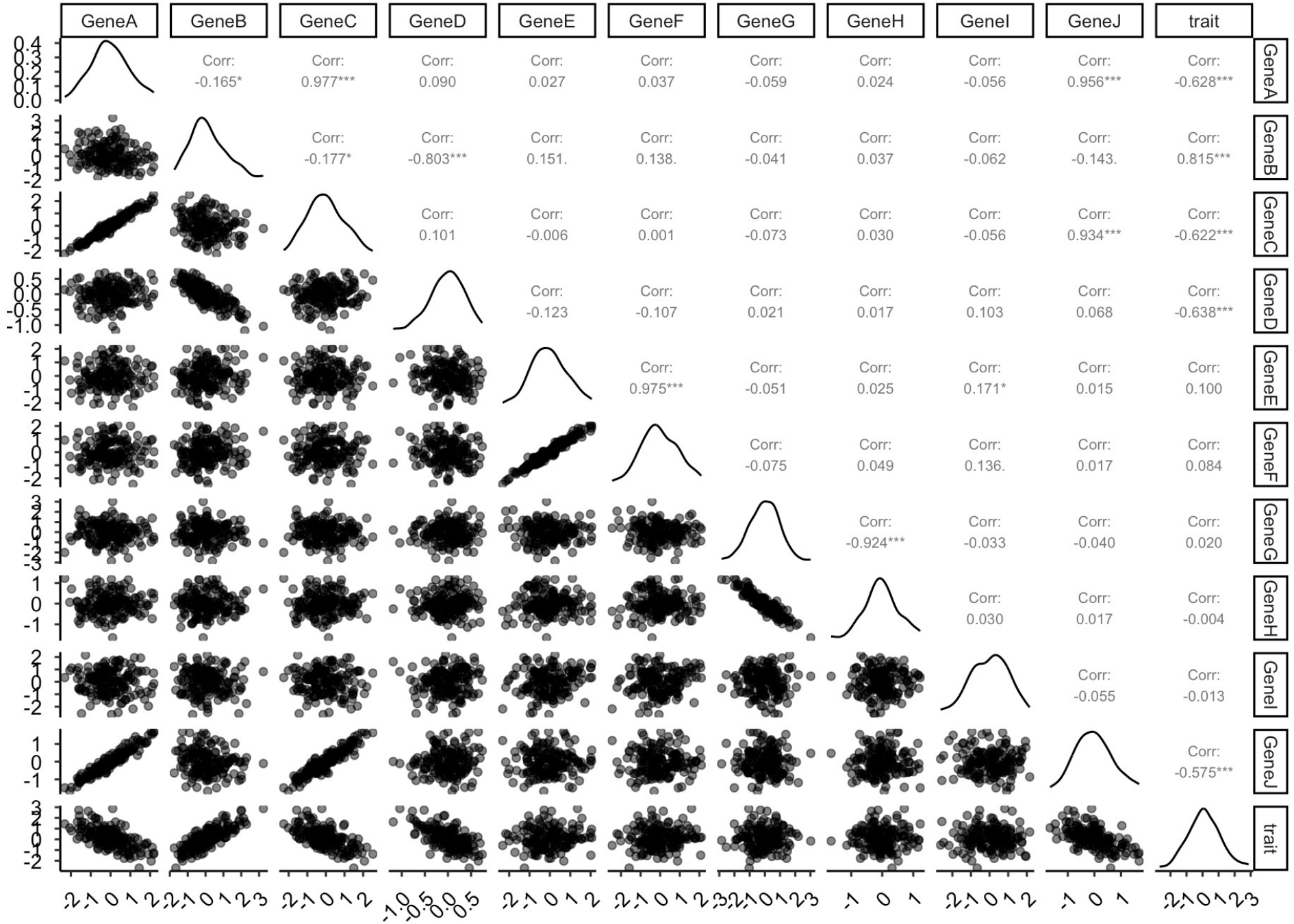
Example 1: multiple linear regression

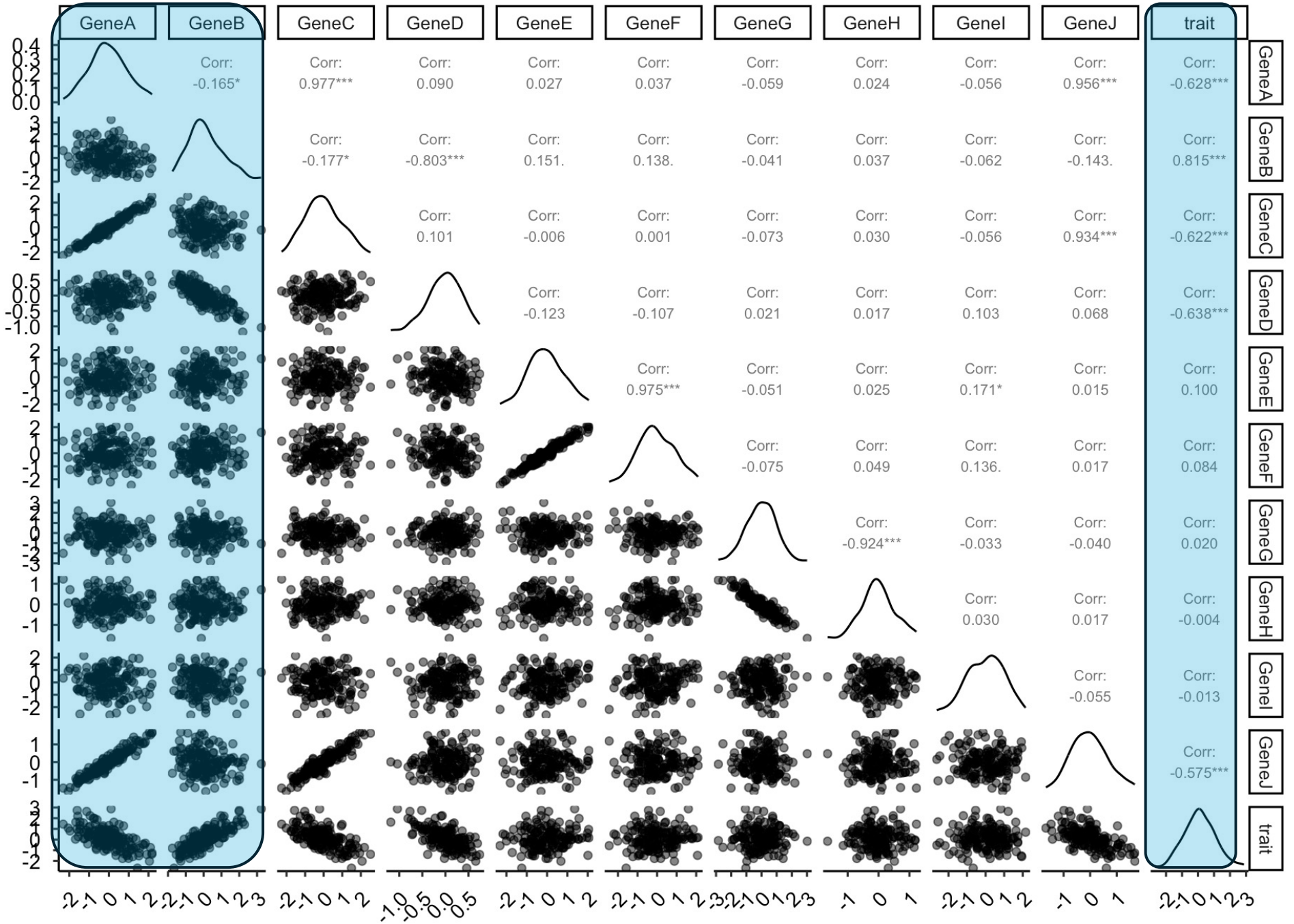
- GeneA and GeneB directly influence a biological trait:

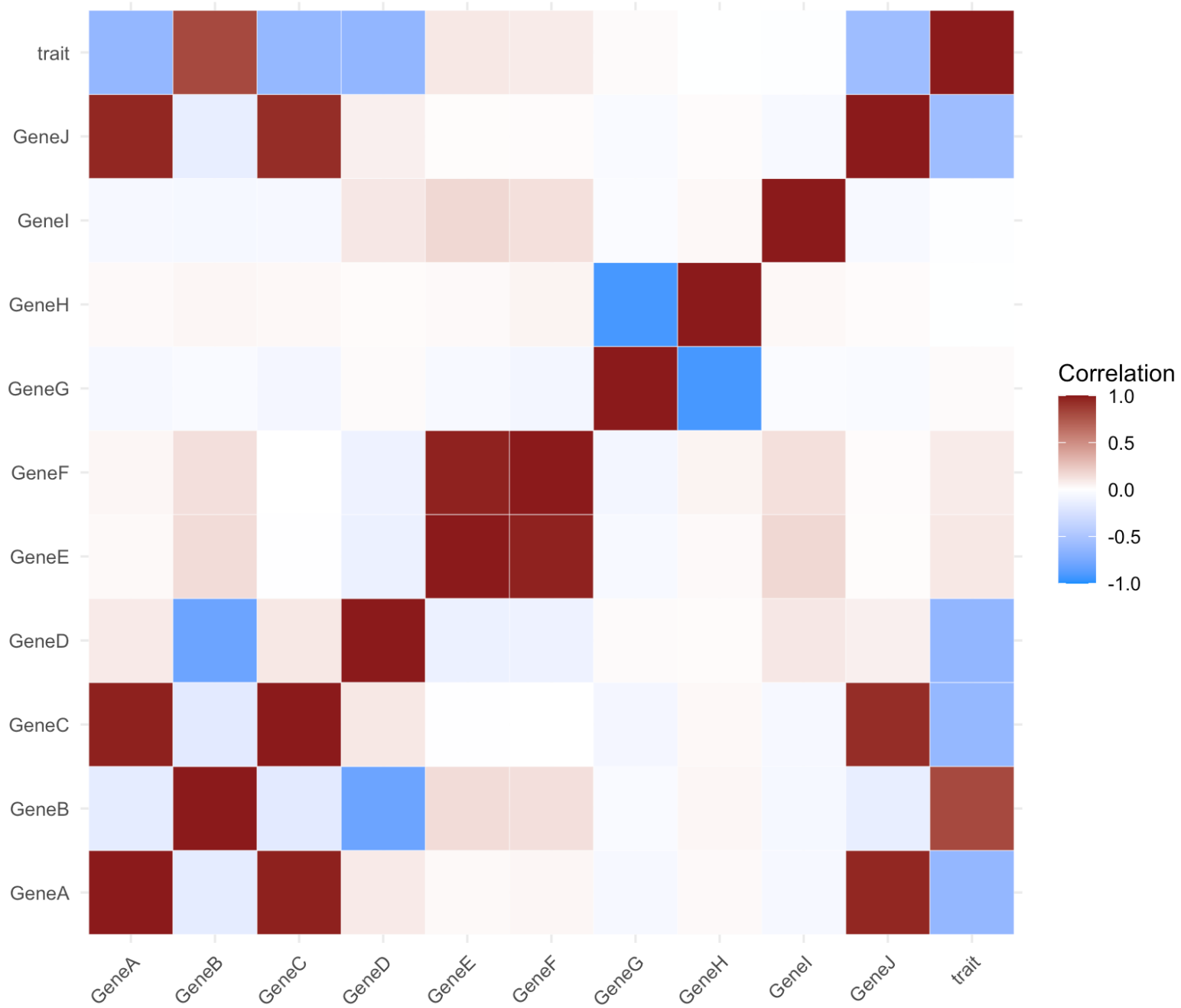
$$trait = -0.5 * GeneA + 0.8 * GeneB + N(0,0.3)$$

- All other genes are noise




GeneA
GeneB
GeneC
GeneD
GeneE
GeneF
GeneG
GeneH
GeneI
GeneJ
trait
Corr: -0.165*
Corr: 0.977***
Corr: 0.090
Corr: 0.027
Corr: 0.037
Corr: -0.059
Corr: 0.024
Corr: -0.056
Corr: 0.956***
Corr: -0.628***
Corr: -0.177*
Corr: -0.803***
Corr: 0.151.
Corr: 0.138.
Corr: -0.041
Corr: 0.037
Corr: -0.062
Corr: -0.143.
Corr: 0.815***
Corr: 0.101
Corr: -0.006
Corr: 0.001
Corr: -0.073
Corr: 0.030
Corr: -0.056
Corr: 0.934***
Corr: -0.622***
Corr: -0.123
Corr: -0.107
Corr: 0.021
Corr: 0.017
Corr: 0.103
Corr: 0.068
Corr: -0.638***
Corr: 0.975***
Corr: -0.051
Corr: 0.025
Corr: 0.171*
Corr: 0.015
Corr: 0.100
Corr: -0.075
Corr: 0.049
Corr: 0.136.
Corr: 0.017
Corr: 0.084
Corr: -0.924***
Corr: -0.033
Corr: -0.040
Corr: 0.020
Corr: 0.030
Corr: 0.017
Corr: -0.004
Corr: -0.055
Corr: -0.013
Corr: -0.575***


GeneA
GeneB
GeneC
GeneD
GeneE
GeneF
GeneG
GeneH
GeneI
GeneJ
trait
Corr: -0.165*
Corr: 0.977***
Corr: 0.090
Corr: 0.027
Corr: 0.037
Corr: -0.059
Corr: 0.024
Corr: -0.056
Corr: 0.956***
Corr: -0.628***
Corr: -0.177*
Corr: -0.803***
Corr: 0.151.
Corr: 0.138.
Corr: -0.041
Corr: 0.037
Corr: -0.062
Corr: -0.143.
Corr: 0.815***
Corr: 0.101
Corr: -0.006
Corr: 0.001
Corr: -0.073
Corr: 0.030
Corr: -0.056
Corr: 0.934***
Corr: -0.622***
Corr: -0.123
Corr: -0.107
Corr: 0.021
Corr: 0.017
Corr: 0.103
Corr: 0.068
Corr: -0.638***
Corr: 0.975***
Corr: -0.051
Corr: 0.025
Corr: 0.171*
Corr: 0.015
Corr: 0.100
Corr: -0.075
Corr: 0.049
Corr: 0.136.
Corr: 0.017
Corr: 0.084
Corr: -0.924***
Corr: -0.033
Corr: -0.040
Corr: 0.020
Corr: 0.030
Corr: 0.017
Corr: -0.004
Corr: -0.055
Corr: -0.013
Corr: -0.575***


trait
GeneJ
GeneI
GeneH
GeneG
GeneF
GeneE
GeneD
GeneC
GeneB
GeneA
GeneA
GeneB
GeneC
GeneD
GeneE
GeneF
GeneG
GeneH
GeneI
GeneJ
trait
Correlation
1.0
0.5
0.0
-0.5
-1.0

Regression Output

term	estimate	std.error	statistic	p.value
(Intercept)	-0.0005	0.0282	-0.0194	0.9845
GeneA	-0.8183	0.1648	-4.9660	0.0000
GeneB	0.8201	0.0489	16.7719	0.0000
GeneC	0.3387	0.1393	2.4315	0.0163
GeneD	0.0564	0.1340	0.4208	0.6746
GeneE	0.1489	0.1376	1.0823	0.2810
GeneF	-0.1445	0.1369	-1.0556	0.2930
GeneG	-0.1290	0.0740	-1.7432	0.0835
GeneH	-0.1400	0.1302	-1.0752	0.2841
GeneI	0.0166	0.0255	0.6500	0.5167
GeneJ	-0.0280	0.1497	-0.1873	0.8517

- While both Gene A and Gene B are correctly identified as significant variables, we also find Gene C which does not have a direct influence on the trait

Practical Session 1.1

Problems with correlated data

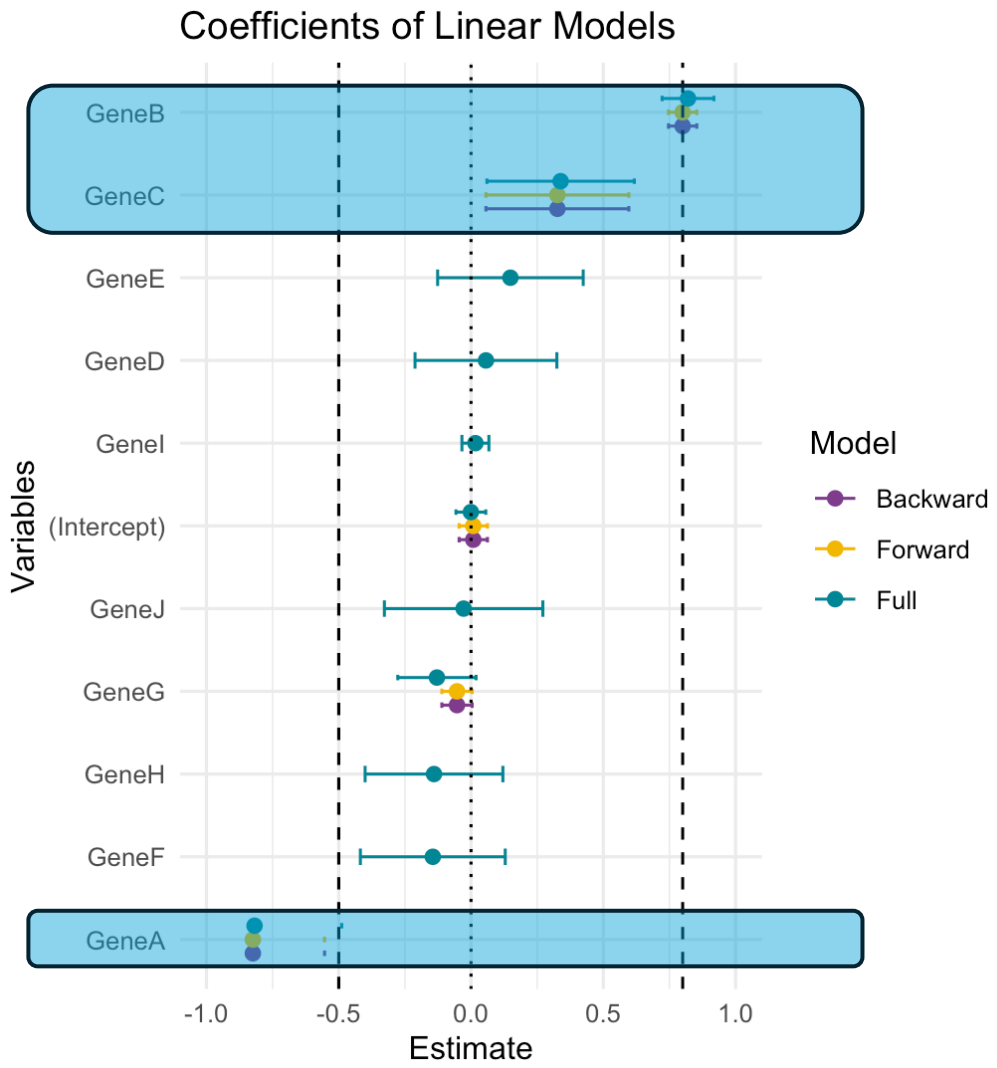
Model Selection …

.. and over- vs. underfitting

Over- and underfitting

- Model should be:
  - As simple as possible
  - As complicated as necessary
- Problem with classical statistics: a more complicated model always fits the data better than a simpler one
- When do we start to fit noise rather than signal?
- Solution1: Use likelihood ratio test (LRT) or Akaike Information Criterion (AIC) for model selection
- Solution 2: automatic variable selection
- (Check performance: test how the model generalizes to unseen data)

Solution 1: forward and backward selection



- AIC: $2\,k - \ 2\log likelihood$
- k is number of parameters in model
- We want the model with smallest AIC
- Forward: start with empty model, gradually add a variable
- Backward: start with full model, gradually remove variables

Code example:

```
# Backward selection using AIC
full_model <- lm(trait ~ ., data = X_train)

backward_model <-
stepAIC(full_model, direction = "backward", trace = FALSE)
```

Code example:

```
# Forward selection using AIC

start_model <- lm(trait ~ 1, data = X_train)


forward_model <-
stepAIC(start_model, direction = "forward", scope = list(lower =
start_model, upper = full_model), trace = FALSE)
```

Solution 2: LASSO

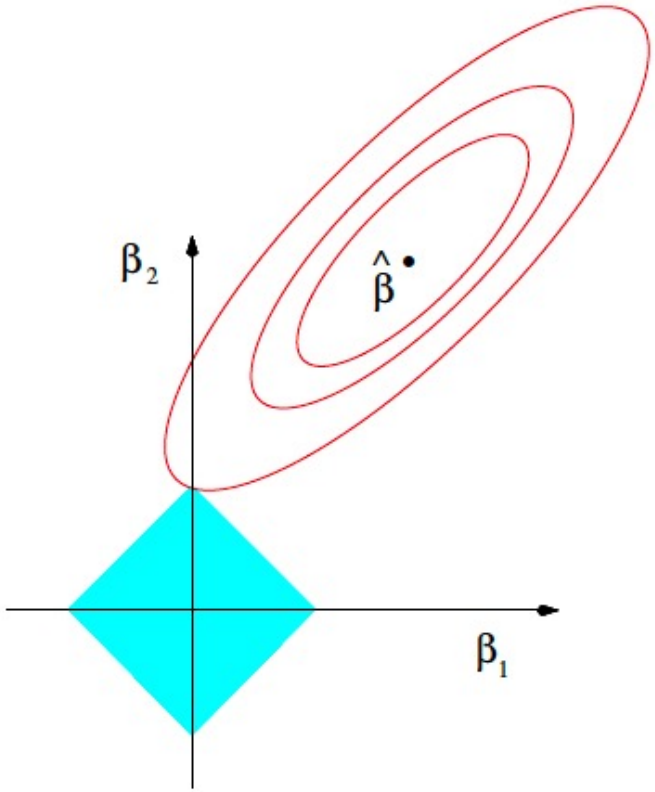
(least absolute shrinkage and selection operator)

LASSO regression: find vector of coefficients β that minimizes the penalized residual sum of squares

$$\text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j| = \|Y - \mathbf{X}\beta\|^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

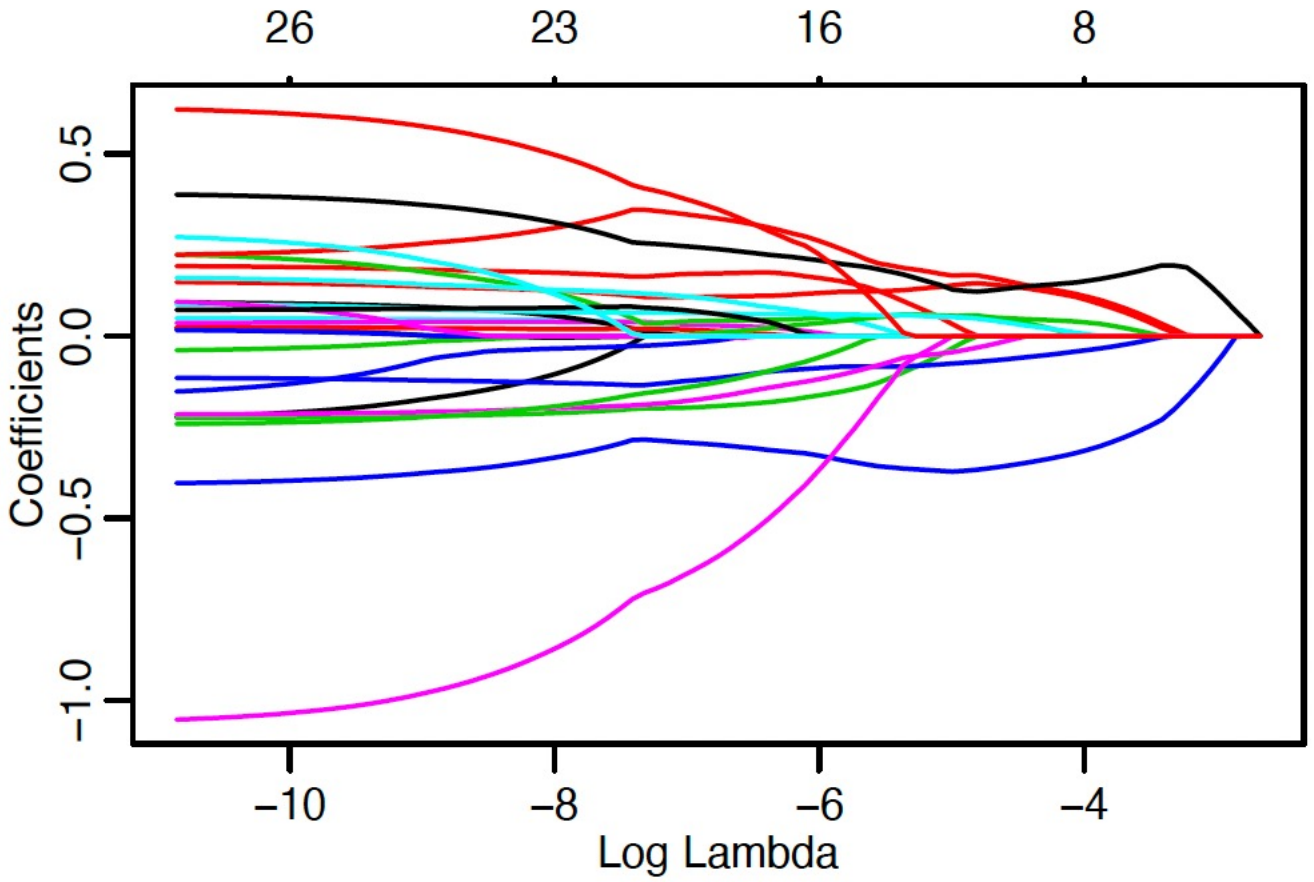
- A regression method for both variable selection and regularization to enhance prediction accuracy and interpretability
- Minimizes the sum of squared residuals subject to a constraint on the sum of the absolute values of the coefficients
- (in ML terms: Adds an L1 penalty to the loss function, which encourages sparsity in the coefficients)

Lasso



- suited for very high-dimensional data (# features >> # samples, e.g. genome-wide measurements)
- variable selection: good interpretability of fitted models
- We introduced a regularization parameter λ
- Each choice of λ leads to a different model
- How can we choose the parameter?

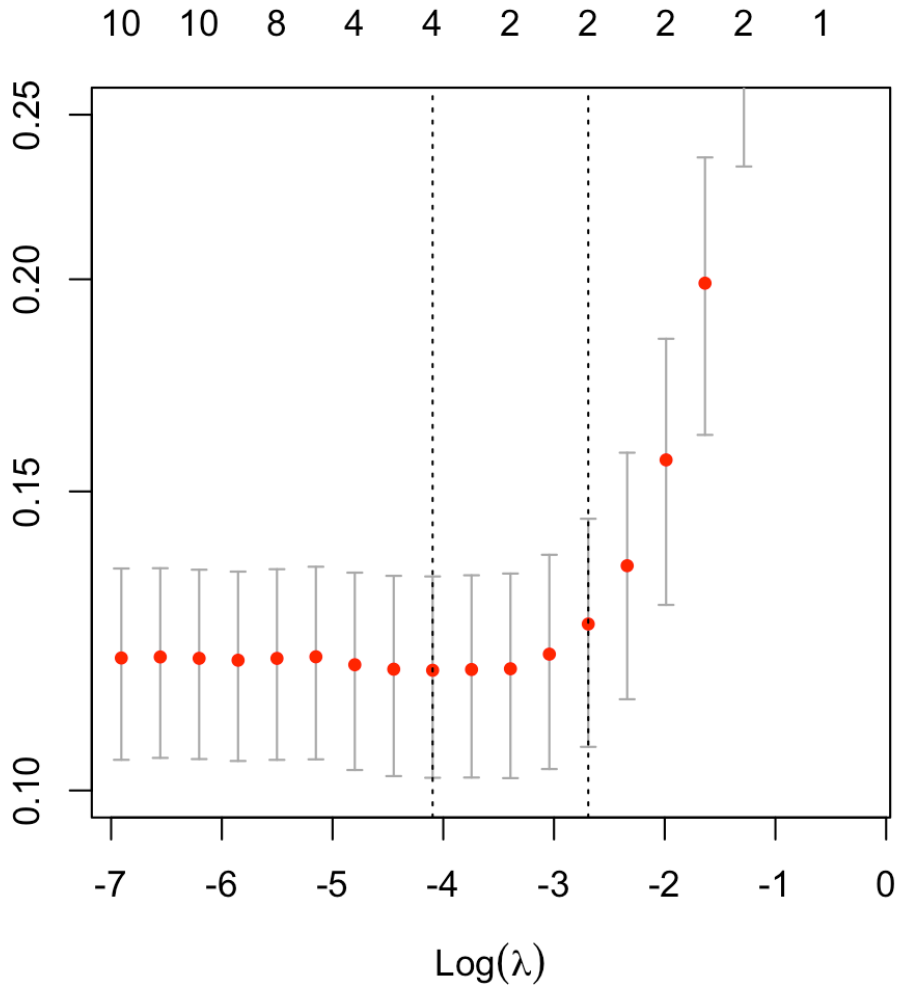
Regularization / Shrinkage



- **Coefficient Shrinkage:**
  - The L1 penalty can shrink some coefficients exactly to zero, effectively performing variable selection.

- **Regularization Parameter (λ):**
  - The strength of the penalty is controlled by the tuning parameter. Higher values of **λ** result in more coefficients being shrunk to zero.

Tuning regularization parameter



- Fit model for a range of **λ** values
- Calculate the expected prediction error for each model
- Choose the largest **λ** with an expected error that is indistinguishable from the minimal error

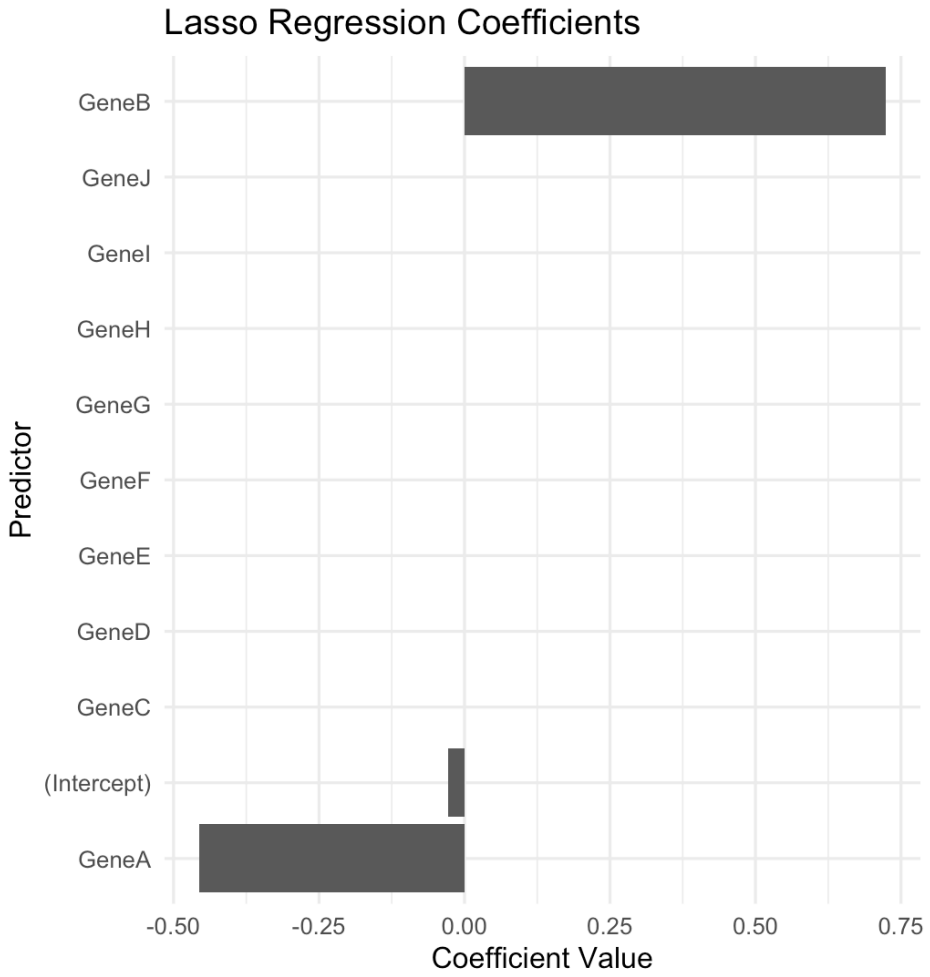
Fitting a LASSO model

```
# Fit LASSO model

lasso_model <- glmnet(X, y, alpha = 1)

# Cross-validation to find the best
lambda

cv_lasso <- cv.glmnet(X, y, alpha = 1)
best_lambda <- cv_lasso$lambda.min

# Coefficients of the best model

coef(cv_lasso, s = best_lambda)
```

- **Model Fitting:**
  Fits a LASSO model using glmnet with alpha = 1 (which specifies LASSO)
- **Cross-Validation:**
  Uses cv.glmnet to perform cross-validation and find the optimal penalty parameter
- **Results:**
  Extracts the coefficients of the best model according to lambda.min.

Coefficients of Lasso Model in our example

- Lasso sets all coefficients to 0 except for GeneA and GeneB (and intercept)
- LASSO successfully accounted for correlations among features, unlike forward or backward selection



New concept: Cross validation

- Problem encountered in LASSO regression:
  choosing a “good” tuning parameter (here: regularization parameter)

- General problem in statistics / ML: there are lots of estimators with“arbitrary” tuning parameters that are not easy to choose

- We have encountered the same problem in the context of the classical model selection: the number of variables taken into a model can be regarded as a “tuning parameter”

Cross Validation

Aim: estimate prediction performance on new data

Problem: we do not have “new” data, we only have our data set!

Solution: split the data set. Use the majority of the data points as training data on which you fit the model, the remaining data points as test data on which you evaluate the model.

Very important: do not evaluate predictions on data points already used for fitting!! This will lead to overfitting.

Leave-one-out Cross Validation

For each data point $i = 1,\ldots, n$, do:

1. Remove data point i
2. Fit a linear model on the remaining $n - 1$ data points using regularized regression
3. Predict the value of the i^{th} response variable based on the fitted model

This is very similar to bootstrapping!

K-fold Cross Validation

k-fold cross validation is a computationally simpler alternative to

leave-one-out cross validation:

Randomly partition data set into k (almost) equally spaces subsets

For each subset j = 1, …, k, do:

1. Remove the j-th subset from the data set
2. Fit a linear model on the remaining n - 1 data points using regularized regression
3. Predict the values of all data points from the removed subset

Practical Session 1.2

Linear Regression, Lasso, Model Selection, Cross Validation

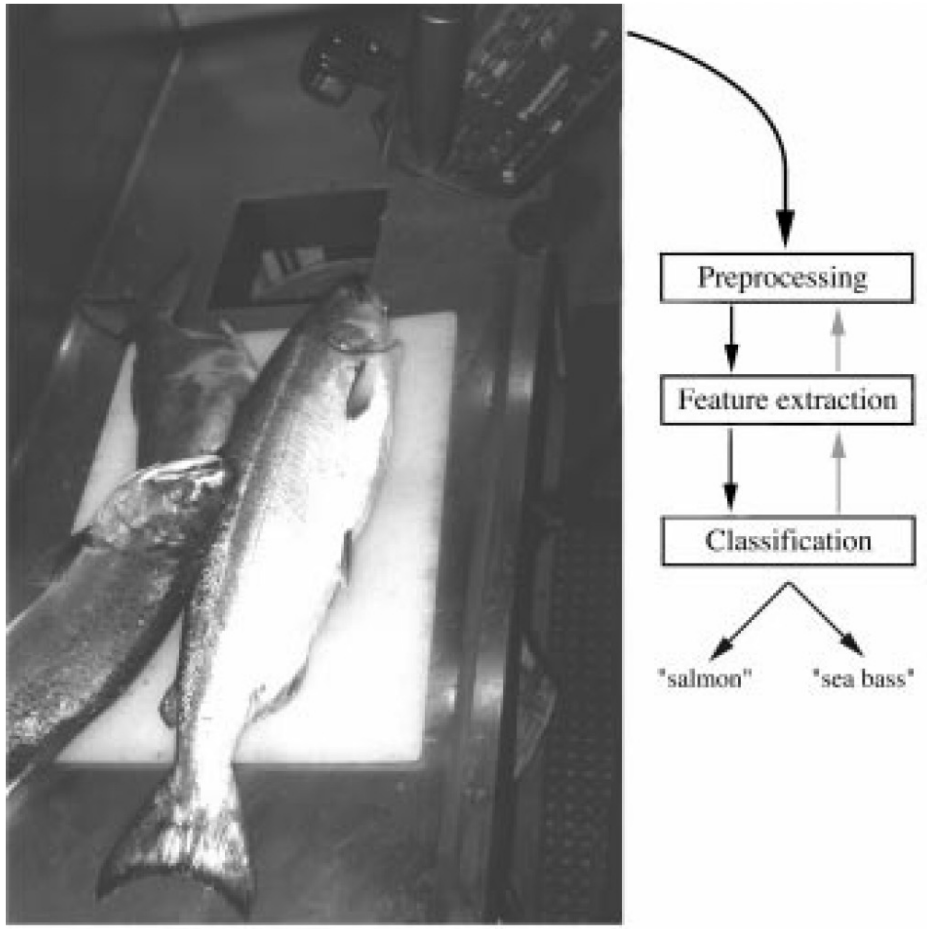
Bayes ...

... and how to classfiy things

Classification

Imaginary, introductory example from Duda et al. (2001): automatic recognition of fish

- Fish packing plant wants to automize packing of salmon and sea bass.
- Camera over a conveyor films fish, image recognition software extracts **features** (explanatory variables) *length* and *lightness*
- Aim: automatic distinction between salmon and sea bass



Setting

General setting

- Variables: p explanatory variables $X_1, \ldots, X_p$, one class label Y
- Data: n data points indicating explanatory variables and class label for n different objects
- Explanatory variables can be continuous or categorical
- Class label is always categorical; each object can belong to one of k classes: $Y \in \{0, \ldots, k-1\}$

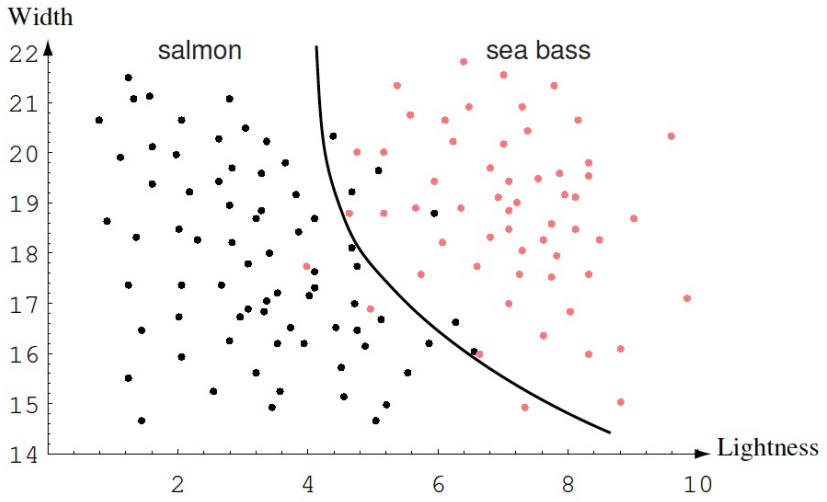
Fish example

- Explanatory variables: X_1: length, X_2: lightness
- Class label: $Y \in \{0, 1\}$, $Y = 0$ for sea bass, $Y = 1$ for salmon

Decision Boundary

- As in regression, there is a trade-off between **performance on the training data** and **simplicity of the classifier**
- Complex classifiers are more flexible, but tend to overfit data
- An optimal classifier minimizes the classification error *on new, unmeasured data*: the **generalization error**

Possible compromise between performance on training data and simplicity:



Bayesian setting: Posterior probabilities

Definition

The **posterior class probability** for class $j \in \{0, 1\}$ for an object with features $x_1, \ldots, x_p$ is given by

$$\pi_j(x_1, \ldots, x_p) = P[Y = j \mid X_1 = x_1, \ldots, X_p = x_p]$$

The posterior class probability can be calculated using **Bayes' theorem**:

$$\pi_j(x_1, \ldots, x_p) = \frac{f_{X_1,\ldots,X_p|Y=j}(x_1, \ldots, x_p) \cdot P[Y = j]}{f_{X_1,\ldots,X_p}(x_1, \ldots, x_p)}$$

Fish example: $\pi_0(x_1, x_2)$ is the probability that a fish with length x_1 and lightness x_2 on the conveyor is a sea brass.

Class-conditional probability density

Let X be an explanatory variable. In the context of classification, the likelihood $f_{X|Y=j}(x)$ $(j = 0, 1)$ is called **class-conditional probability density** of X.

Bayes Clasifier

Definition

The **Bayes classifier** is given by the following decision rule: for an object with features $x_1, \ldots, x_p$, predict its class $c(x_1, \ldots, x_p)$ as...

- ...0, if $\pi_0(x_1, \ldots, x_p) > \pi_1(x_1, \ldots, x_p)$,
- ...1, if $\pi_0(x_1, \ldots, x_p) < \pi_1(x_1, \ldots, x_p)$.

In words: the Bayes classifier predicts the class which has the highest *posterior* class probability.

Optimality of Bayes classifier

The Bayes classifier is the classifier with the *lowest* expected misclassification rate.

Hence, no classifier can beat the Bayes classifier in the long run.

Logisitic Regression

- Aim: fitting the posterior class probabilities $\pi_j(x_1, \ldots, x_p)$
- Note: since $\pi_0(x_1, \ldots, x_p) = 1 - \pi_1(x_1, \ldots, x_p)$, it is sufficient to fit one of the functions; by convention, we fit π_1.
- Idea: apply a monotone transformation to $\pi_1(\cdot)$ that maps the interval $(0, 1)$ to the real line $\mathbb{R}$.
- The **logistic transformation** accomplishes this: instead of fitting $\pi_1(\cdot)$, we fit

$$\log\left(\frac{\pi_1(x_1, \ldots, x_p)}{1 - \pi_1(x_1, \ldots, x_p)}\right) .$$

- The simplest model for fitting this transformed posterior class probability is a linear function:

$$\log\left(\frac{\pi_1(x_1, \ldots, x_p)}{1 - \pi_1(x_1, \ldots, x_p)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p .$$

Example: gene expression pt. 2

We simulate a biological dataset where gene expression levels influence the presence or absence of a disease trait (= categorical variable).

We generate synthetic gene expression data, create a response variable (`disease_presence`), and fit a logistic regression model.

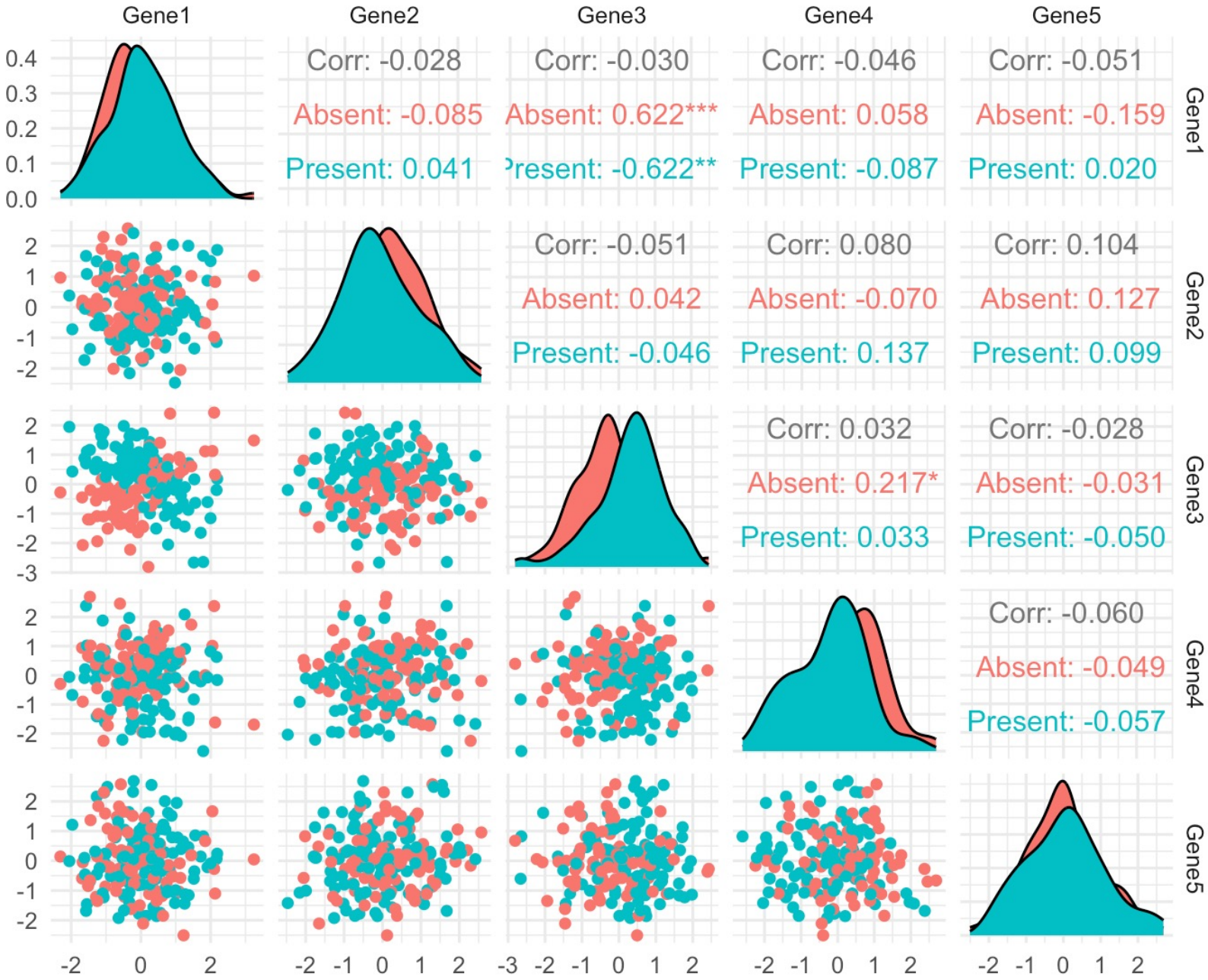
Dataset

- Number of Samples (n): 200
- Number of Features (p): 10 genes (named Gene1 to Gene10)
- Gene Expression Data: Simulated as normally distributed random values.
- Causal Genes:
  - Gene1: Positive effect with a coefficient of 0.5, plus interaction with Gene3.
  - Gene3: Positive effect with a coefficient of 0.8, plus interaction with Gene1.
  - Gene4: Negative effect with a coefficient of -0.3.
- Response Variable (disease_presence):
  - Binary (indicating the presence or absence of a disease trait) generated based on a linear combination of Gene1, Gene3, and Gene4 with some added noise.

Dataset

- Causal Genes:
  - Gene1: Positive effect with a coefficient of 0.5, plus interaction with Gene3.
  - Gene3: Positive effect with a coefficient of 0.8, plus interaction with Gene1.
  - Gene4: Negative effect with a coefficient of -0.3

```
disease_presence <- ifelse(
0.5 * Gene1 +  0.8 * Gene3 - 0.3 * Gene4 –
 2  * Gene1*Gene3 * + rnorm(n, sd=0.5) > 0,
1, 0)
```


Gene1
Gene2
Gene3
Gene4
Gene5
Corr: -0.028
Absent: -0.085
Present: 0.041
Corr: -0.030
Absent: 0.622***
Present: -0.622**
Corr: -0.046
Absent: 0.058
Present: -0.087
Corr: -0.051
Absent: -0.159
Present: 0.020
Corr: -0.051
Absent: 0.042
Present: -0.046
Corr: 0.080
Absent: -0.070
Present: 0.137
Corr: 0.104
Absent: 0.127
Present: 0.099
Corr: 0.032
Absent: 0.217*
Present: 0.033
Corr: -0.028
Absent: -0.031
Present: -0.050
Corr: -0.060
Absent: -0.049
Present: -0.057

Perform logistic regression on example data

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.31891    0.16361   1.949 0.051277 .
Gene1         0.26941    0.17722   1.520 0.128464
Gene2        -0.24082    0.16551  -1.455 0.145672
Gene3         0.82736    0.18946   4.367 1.26e-05 ***
Gene4        -0.57850    0.17151  -3.373 0.000744 ***
Gene5         0.10475    0.16203   0.646 0.517959
Gene6         0.10762    0.16665   0.646 0.518408
Gene7         0.12048    0.17324   0.695 0.486788
Gene8         0.39470    0.18067   2.185 0.028912 *
Gene9        -0.12785    0.15425  -0.829 0.407174
Gene10       -0.07877    0.15804  -0.498 0.618189
```

- Cannot handle the interaction between Gene 1 and Gene 3 (misses Gene 1)
- Finds 2 causal genes (Gene 3 and 4), but also a “noise Gene“ (Gene 8)

Perform logistic regression on example data

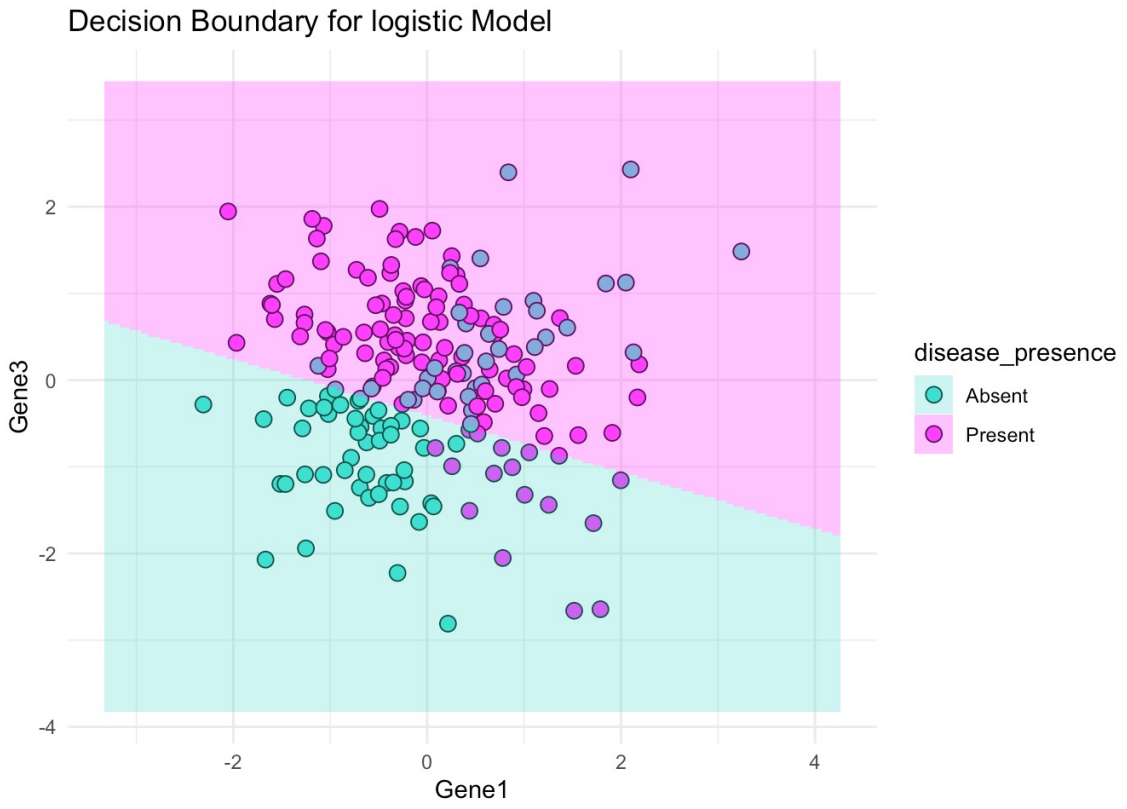
```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.31891    0.16361   1.949 0.051277 .
Gene1        0.26941    0.17722   1.520 0.128464
Gene2       -0.24082    0.16551  -1.455 0.145672
Gene3        0.82736    0.18946   4.367 1.26e-05 ***
Gene4       -0.57850    0.17151  -3.373 0.000744 ***
Gene5        0.10475    0.16203   0.646 0.517959
Gene6        0.10762    0.16665   0.646 0.518408
Gene7        0.12048    0.17324   0.695 0.486788
Gene8        0.39470    0.18067   2.185 0.028912 *
Gene9       -0.12785    0.15425  -0.829 0.407174
Gene10      -0.07877    0.15804  -0.498 0.618189
```

- Cannot handle the interaction between Gene 1 and Gene 3 (misses Gene 1)

- Finds 2 causal genes (Gene 3 and 4), but also a “noise Gene“ (Gene 8)

Logistic Regression in R

```
logistic_model <-
glm(disease_presence ~ .,
data = bio_data,
family = "binomial")
```



Practical session 1.3

Logistic Regression