# Analyse hochdimensionaler Daten

## Vorhersage und Feature Assessment - Teil 1

# Mein Werdegang & hochdimensionale Statistik

**Wattwil im Toggenburg**

Nicolas Städler, Hobbies:
Sport und Reisen

**Studium ETH Zürich**

Doktorat in Mathematik/Statistik
Hochdimensionale
Mischungsmodelle und Missing
Data Probleme

**Post-Doc Amsterdam**


Molecular disease heterogeneity
Netzwerke & Clustering in
hochdimensionalen Daten

# Mein Werdegang & hochdimensionale Statistik

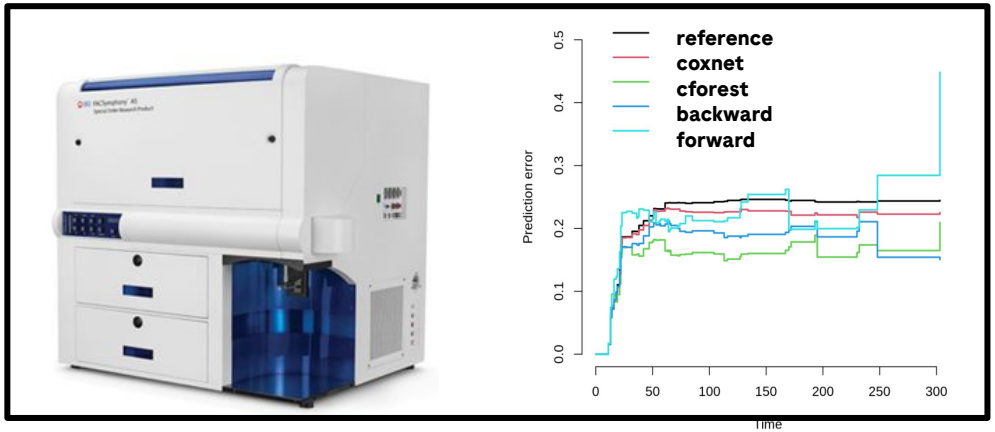
**Biostatistiker in Roche Basel**

Health Technology Assessment

Predictive Modelling and Data Analytics

Bioinformatik & Biomarker

**Klinische Biomarker und personalisierte Medizin**



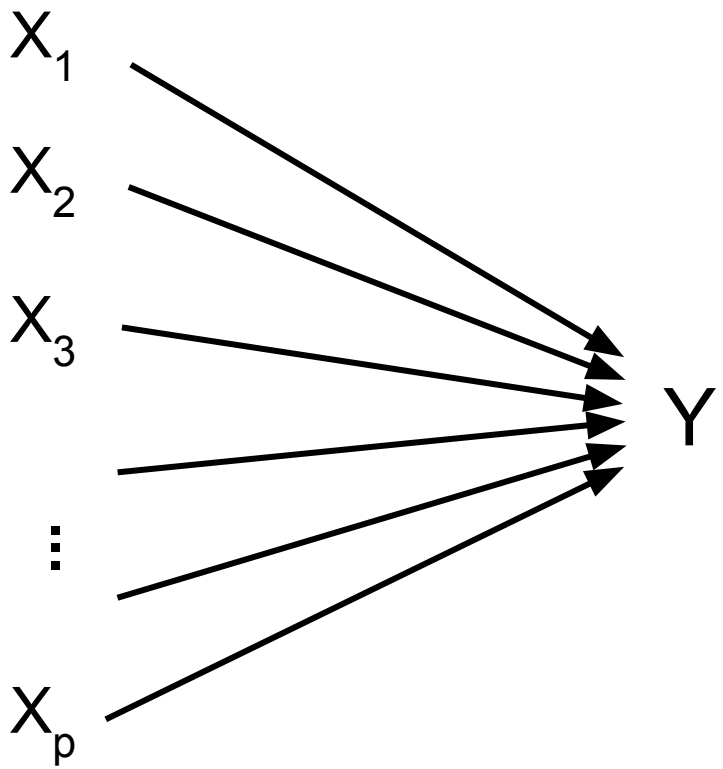

Erhebung klinischer Biomarker anhand “Omics” Technologien

Analyse hochdimensionaler Biomarker Daten, zB prädiktive Modellierung

# Hochdimensionale Statistik
# Unversity of California, Berkely

*High-dimensional statistics focuses on data sets in which the number of features is of comparable size, or larger than the number of observations. Data sets of this type present a variety of new challenges, since classical theory and methodology can break down in surprising and unexpected ways*

# Vorhersage und Feature Selektion



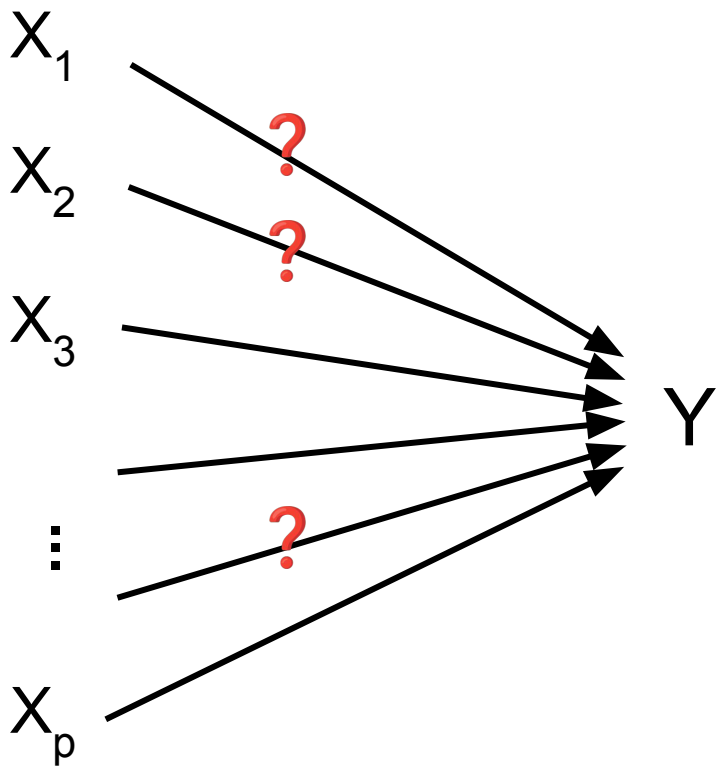

Ausgangslage

- Zielgrösse Y
- Features $X_1,...,X_p$
- Hochdimensionale Daten

**p gross, n klein**

# Feature, Kovariable, Erklärende Variable
# Zielgrösse, Response, Abhängige Variable

# Vorhersage und Feature Selektion

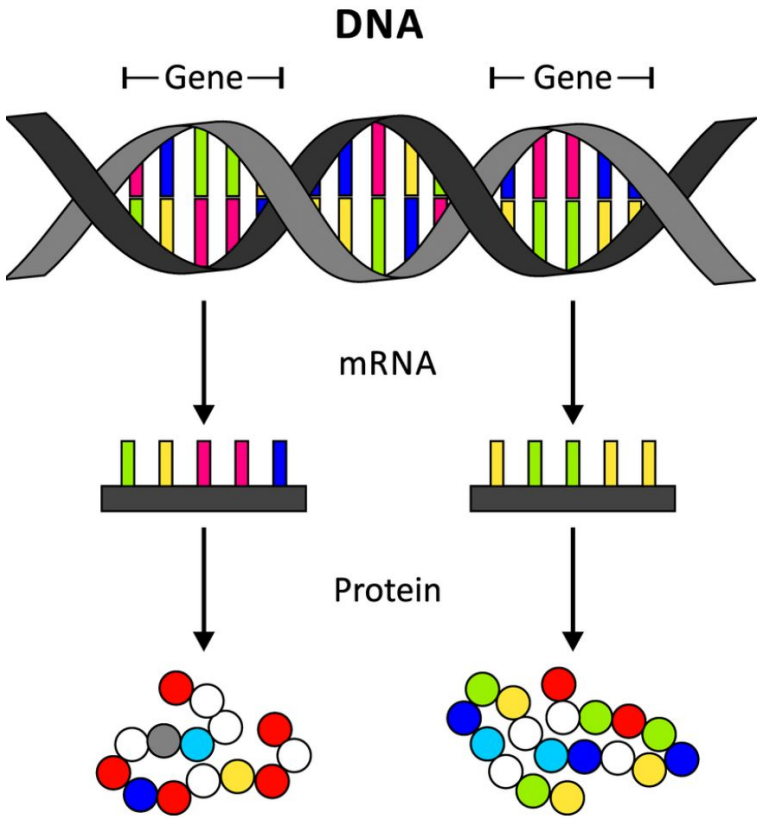



Fragestellungen

- Können wir die Zielgrösse (Y) mittels Features ($X_1$,...,$X_p$) vorhersagen? (**Vorhersage**)
- Welche “Features” erklären die Zielgrösse am besten? (**Feature Selektion**)

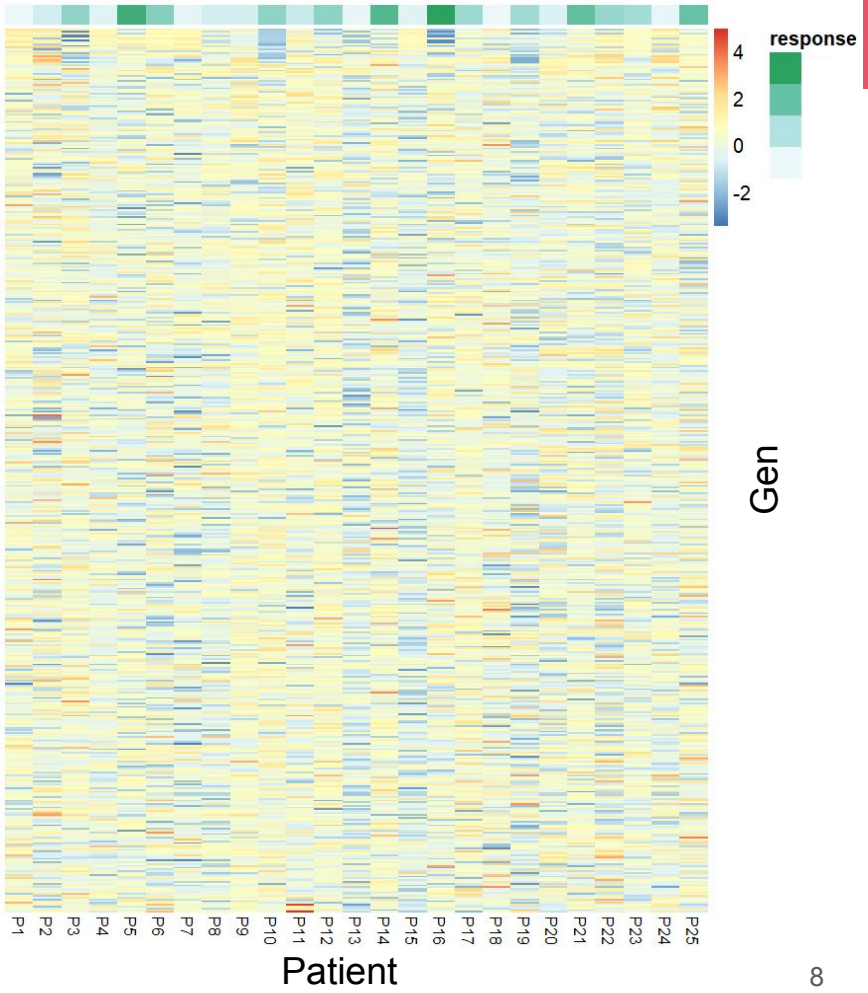
# Molekularbiologie und genomische Daten

- Die DNA ist der Bauplan des Lebens aller Organismen
- Die DNA besteht aus vielen Segmenten, sog. Genen (Mensch hat ~22’500 Gene)
- Aus Genen werden mRNA Moleküle hergestellt (Transkription)
- Aus mRNA Molekülen werden Proteine gebaut (Translation)



# Genexpression

- RNA Sequencing Technologie ermöglicht Expressions-Messung tausender Genen simultan (p~10’000).
- Die Stichprobengrösse (z.B. Patienten) ist relative klein (n~10-100).
- Können wir den Response anhand der Genexpressionsmuster vorhersagen?



# Finanzreihen

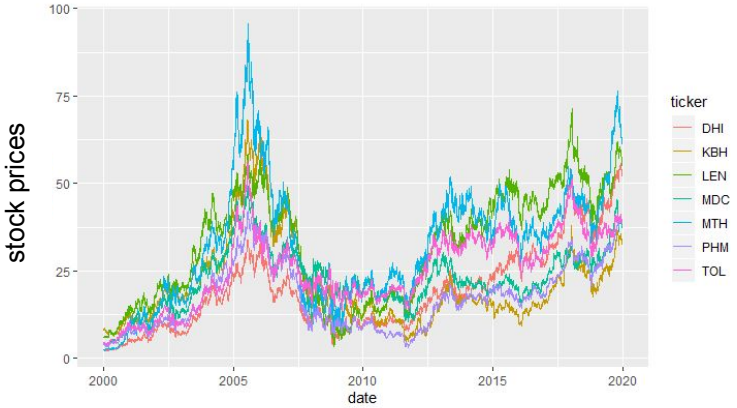
- Modellierung von Aktienkursen
- VAR(d) Modell für p Aktien

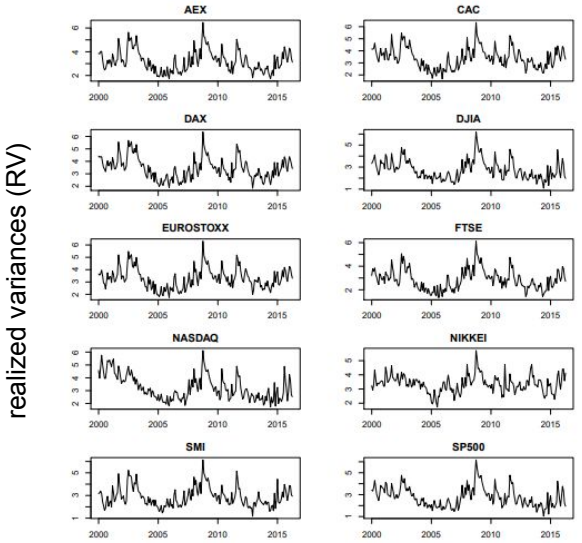
$$X_1^t = \alpha_1 + \beta_{11} X_1^{t-1} + \ldots + \beta_{1d} X_1^{t-d} + \epsilon_{t1}$$

$$\vdots \qquad\qquad \vdots$$

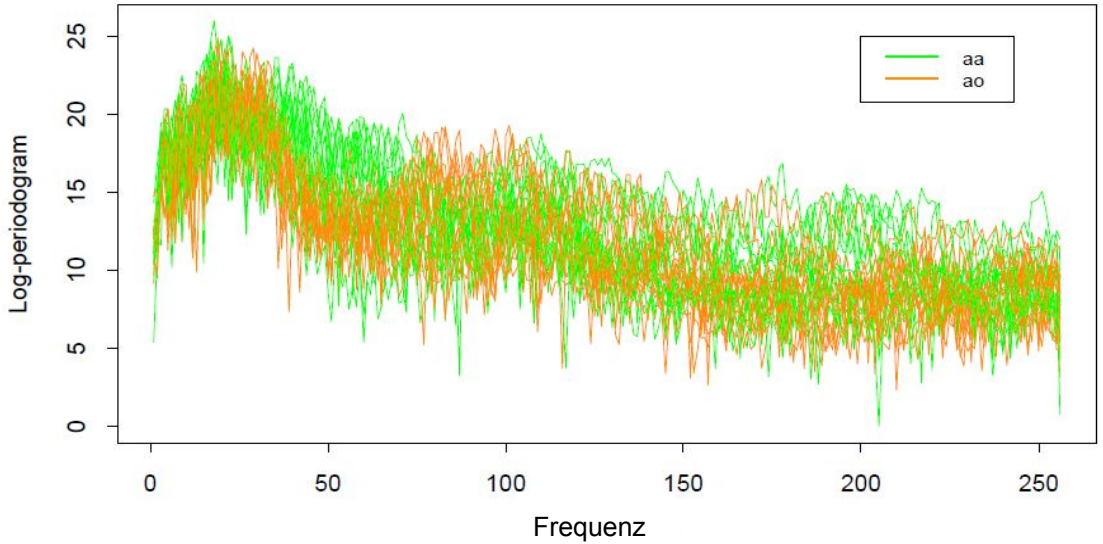
$$X_p^t = \alpha_p + \beta_{p1} X_1^{t-1} + \ldots + \beta_{pd} X_p^{t-d} + \epsilon_{pt}$$

- Grosse Anzahl Modellparameter $d \times p^2$
- Vorhersage der Aktienkurse?
- Was sagen uns die geschätzten Modellparameter?



# Spracherkennung (Speech recognition)




- Erkennung von Sprachmustern anhand digitaler Aufnahmen
- Phoneme “aa (balm)” and “ao (bought)” (n=30=15+15) gemessen an p=256 verschieden Frequenzen
- Können wir Phoneme zuverlässig vorhersagen/klassifizieren?
- Welche Frequenzen sind wichtig?

# In diesem Kurs geht es um…

- Hochdimensionale Daten
- Werkzeuge zur Vorhersage und zur Feature Selektion
- Viele praktische Übungen mit R
- Kurs Homepage: https://staedlern.github.io/highdim_stats/
- Online Skript: https://bookdown.org/staedler_n/highdimstats/
- Daten und R Code: github

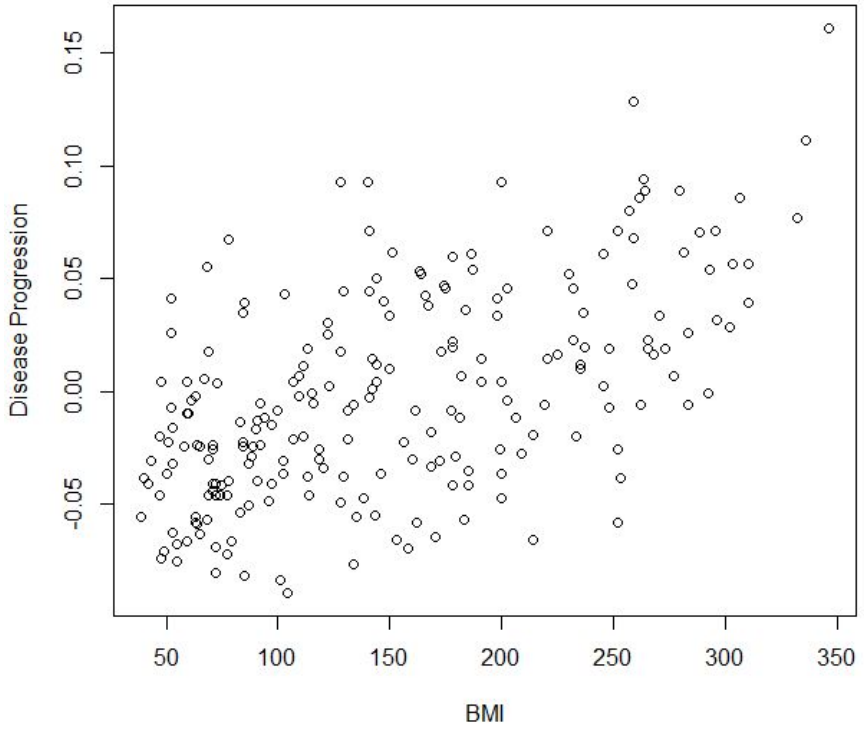
# Kursinhalt

- Lineare Regression und Methode der Kleinsten Quadrate
- Überanpassung, Generalisierungsfehler und Bias-Varianz Dilemma
- Subset Regression und Modellselektion
- Ridge -, Lasso - und Elasticnet Regression
- Klassifikation, Logistische Regression & Elasticnet
- Maschinelles Lernen: Entscheidungsbäume, Random Forest und AdaBoost
- Vorhersage und Feature Selektion für Ereigniszeitanalyse
- Multiples Testen, Bonferroni und FDR Korrektur, Schrumpfung der Varianz

# Multiple Linear Regression

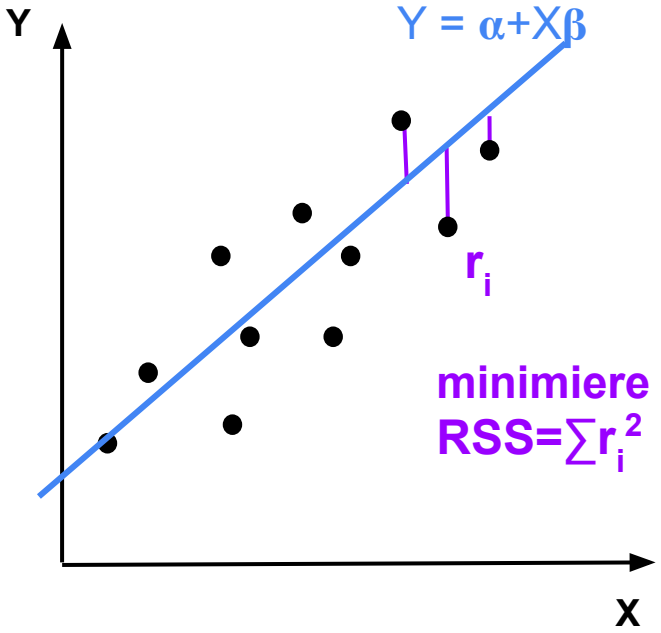
# Diabetes Beispiel

- n=442 Patienten mit Diabetes
- Y: Fortschreiten der Krankheit 1 Jahr nach Studienbeginn
- X: Alter, Geschlecht, BMI, Blutdruck, Blutserum Werte
- Aufgabe für den Statistiker:
  - Finde ein Modell, das den Krankheitsfortschritt (Y) vorhersagt.
  - Welche Variablen sind wichtige Faktoren?

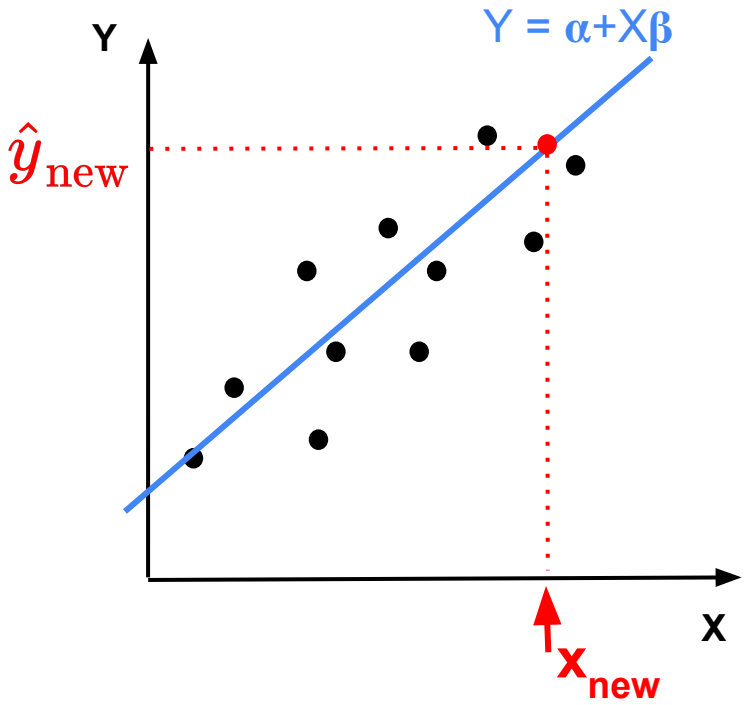



**Lineare Regression wurde erfunden für solche Aufgaben!**

# Lineare Regression




- Response Y ist kontinuierlich
- Anpassung einer Geraden




- Vorhersage neuer Datenpunkte

# Multiple Lineare Regression

- Gegeben: Zielgrösse und Kovariablen

$$Y = \beta_0 + X_1\beta_1 + \cdots + X_p\beta_p$$

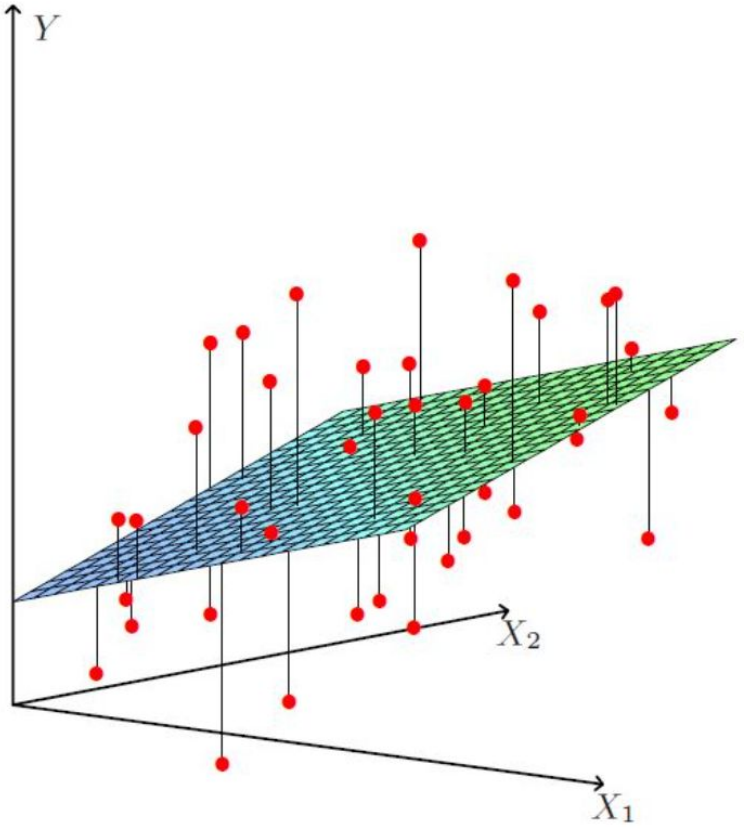
- $\beta_j$: Einfluss auf Y bei Änderung in $X_j$ und Fixierung aller anderen Variablen

# Multiple Lineare Regression

- Anpassung einer p-dimensional Hyperebene an die Datenpunkte:

$$(y_i, x_{1i}, \ldots, x_{pi})$$

$$i = 1, \ldots, n$$



# Multiple Lineare Regression

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \approx \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}$$

# Methode Kleinster Quadrate

- Residual Sum of Squares (RSS)

$$\text{RSS}(\beta) = \sum_{i=1}^{n} \overbrace{(y_i - x_i^T \beta)^2}^{= r_i}$$
$$= (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)$$
$$= \|\mathbf{y} - \mathbf{X}\beta\|_2^2.$$

# Methode Kleinster Quadrate

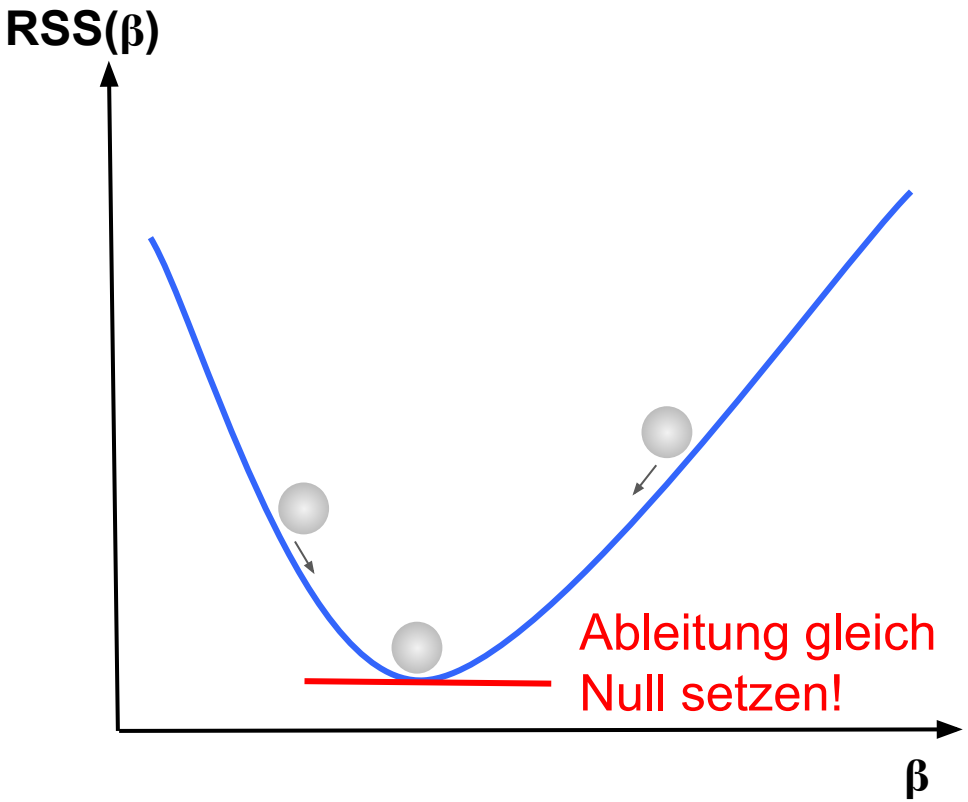
- Schätzung der Regressions Koeffizienten mittels der “Methode Kleinster Quadrate” (Ordinary Least Squares, OLS)

$$\hat{\beta} = \arg\min_{\beta} \mathrm{RSS}(\beta)$$

Optimierungsproblem:

“Finde das Argument des Minimums”

# Wie finden wir das Minimum einer Funktion?




$$\frac{\partial}{\partial \beta} \mathrm{RSS}(\beta) = 0$$

$$\Leftrightarrow \quad \mathbf{X^T y} = \mathbf{X^T X} \beta$$

$$\hat{\beta} = (\mathbf{X^T X})^{-1} \mathbf{X^T y}$$

“Geschlossene” Lösung

# Notiz: viele statistischen Verfahren haben keine geschlossene Lösung und müssen “numerisch” approximiert werden

# Multiple Lineare Regression in R - Diabetes Beispiel

```
fit <- lm(y~age+sex+bmi+map,data=data_train)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ age + sex + bmi + map, data = data_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -147.636  -42.617   -5.229   42.301  154.569
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   153.20       3.97  38.589  < 2e-16 ***
## age           -18.01      91.02  -0.198    0.843
## sex            58.44      85.47   0.684    0.495
## bmi           748.95      84.96   8.816 4.01e-16 ***
## map           422.63      95.47   4.427 1.52e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 58.86 on 216 degrees of freedom
## Multiple R-squared:  0.4072, Adjusted R-squared:  0.3962
## F-statistic:  37.1 on 4 and 216 DF,  p-value: < 2.2e-16
```

- Geschätzten Koeffizienten (mittels “kleinster Quadrate”)
- Weitere statistische Kenngrößen, z.B. p-Werte
- R-squared: wie gut passt das Modell zu den Daten?

# Überanpassung, Generalisierungsfehler und Bias-Varianz Dilemma

# Hochdimensionale Daten

- Die Anzahl der Kovariablen ist gross im Vergleich zur Beobachtungen (p>>n)
- Regression mit allen p Kovariablen, d.h. $Y=\beta_0+X_1\beta_1+...+X_p\beta_p$?
- Wieso ist dies keine gute Idee?

Probleme:

- Singularität der Designmatrix
- Überanpassung an die Daten (sogn “Overfitting”)
- Grosser Generalisierungsfehler

# Problem 1: Singularität der Designmatrix

$$\frac{\partial}{\partial \beta} \mathrm{RSS}(\beta) = 0$$

$$\Leftrightarrow \mathbf{X^T y} = \mathbf{X^T X} \beta$$

$$\hat{\beta} = (\mathbf{X^T X})^{-1} \mathbf{X^T y}$$

- p Gleichungen mit p Unbekannten
- Lösbar wenn $\mathbf{X^T X}$ invertierbar
- Wichtig: p>n impliziert $\mathbf{X^T X}$ singulär (d.h. nicht invertierbar)

# Problem 2: Überanpassung an die Daten

- Zu viele erklärende Variablen führen zu einer Überanpassung an die Daten, s.g. “Overfitting”
- Überanpassung bedeutet: das Modell beschreibt nicht nur das echte Signal, sondern auch den zufälligen Fehler
- Illustrieren dies mit Hilfe künstlich generierten Daten

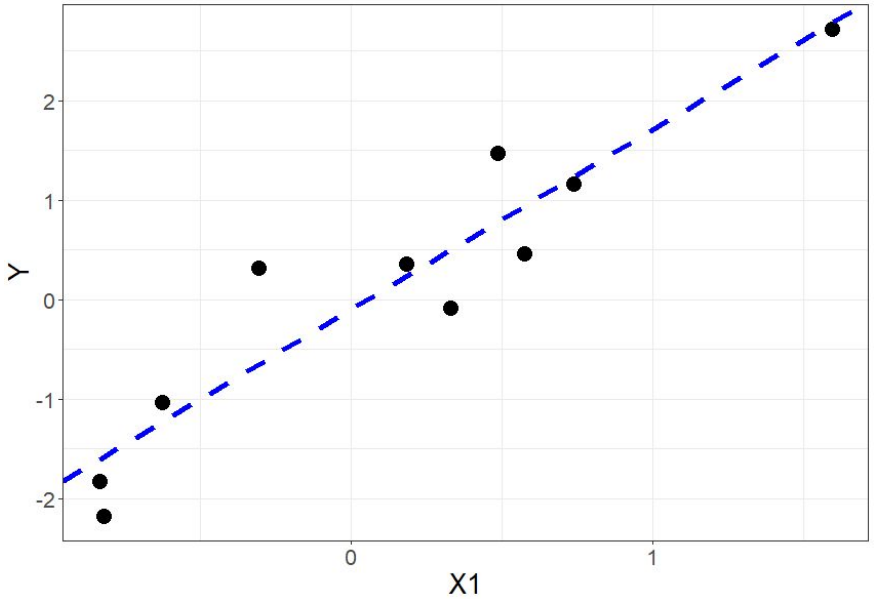
# Simuliere künstliche Daten

- n=10: $(Y_i, X_{i1}, \ldots, X_{ip})$ i=1..n
- p=15: $X_{i1}, \ldots, X_{ip}$ i.i.d N(0,1)
- Zielgrösse hängt nur von der ersten Kovariaten ab:

$$Y_i = \beta_1 X_{i1} + \epsilon_i,$$
$$\beta_1 = 2,\ \epsilon_i \sim N(0, 0.5^2)$$

```r
set.seed(1)

n <- 10

p <- 15

beta <- c(2,rep(0,p-1))


# simulate covariates

xtrain <- matrix(rnorm(n*p),n,p)

ytrain <- xtrain%*%beta+rnorm(n,sd=0.5)

dtrain <- data.frame(xtrain)

dtrain$y <- ytrain
```

# Univariates Modell “Orakel”

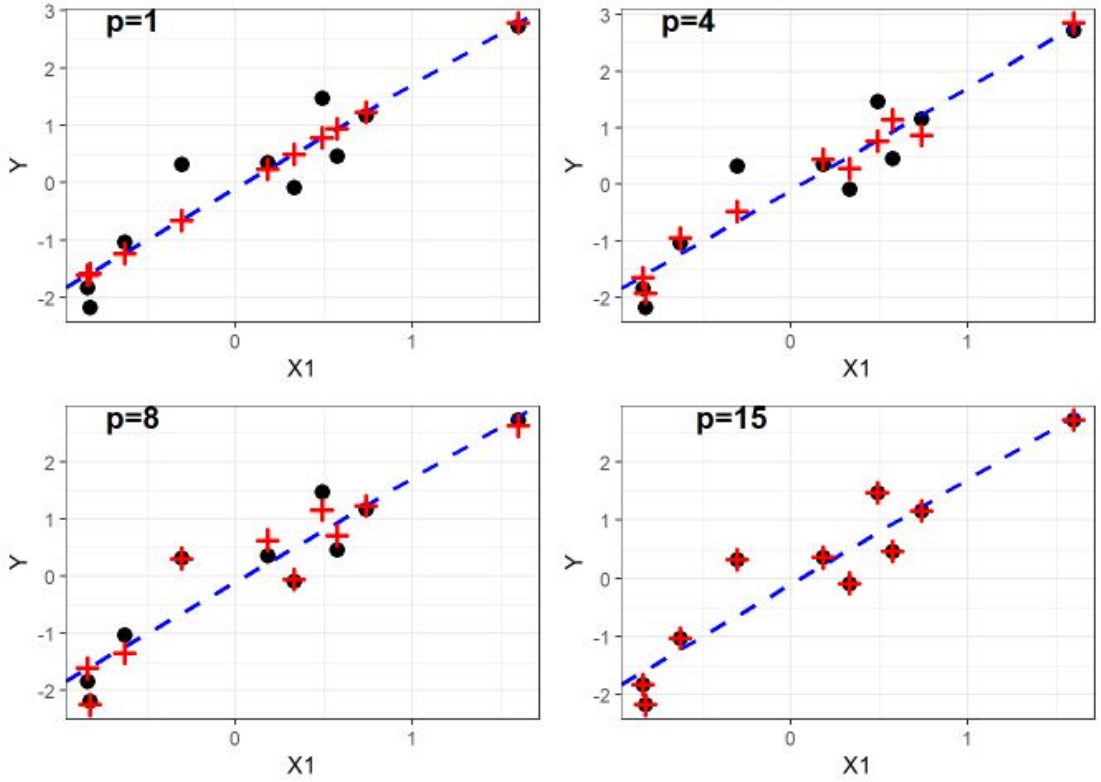



```
fit1 <- lm(y~X1,data=dtrain)
summary(fit1)
```

```
##
## Call:
## lm(formula = y ~ X1, data = dtrain)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.59574 -0.41567 -0.06222  0.18490  0.97592
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.1002     0.1785  -0.561     0.59
## X1            1.8070     0.2373   7.614 6.22e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5558 on 8 degrees of freedom
## Multiple R-squared:  0.8787, Adjusted R-squared:  0.8636
## F-statistic: 57.97 on 1 and 8 DF,  p-value: 6.223e-05
```

- Das Modell passt gut: $R^2$=0.88
- Regressionskoeffizient ist nahe beim wahren Wert
- Was passiert wenn wir “noise” Kovariablen hinzufügen, d.h. p=4, 8, 15?

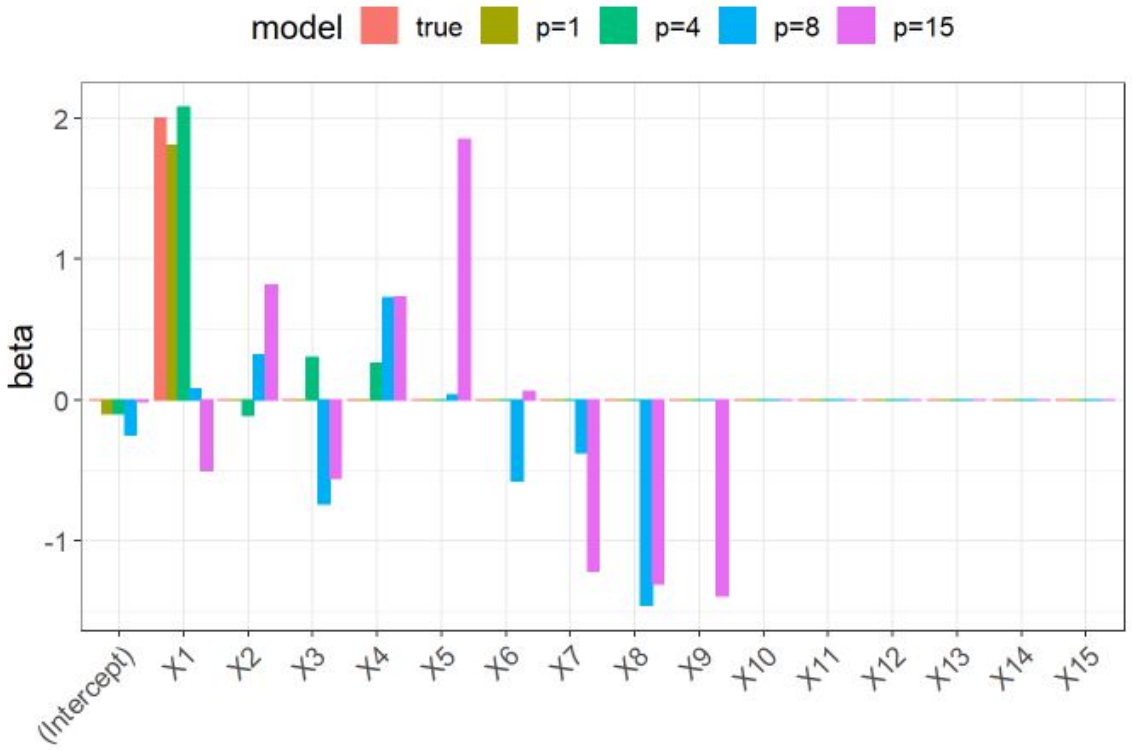
# Überanpassung an die Daten




| model | R2 |
|---|---|
| p=1 | 0.88 |
| p=4 | 0.90 |
| p=8 | 0.98 |
| p=15 | 1.00 |

- Gefittete Werte (rotes Kreuz) bewegen sich weg vom wahren Signal (blaue Linie)
- Gefittete Werte nähern sich den Datenpunkten an
- Modellierung von “Noise”
- Model “overfits” die Daten

# Überschätzung der Koeffizienten




- p gross: Überschätzung der Koeffizienten
- Modelle bewegen sich weg von der Wahrheit (trotz gutem “Fit”)

# Modell mit p=15

```
summary(fit15)
```

```
##
## Call:
## lm(formula = y ~ ., data = dtrain)
##
## Residuals:
## ALL 10 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (6 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.01592        NaN     NaN      NaN
## X1          -0.50138        NaN     NaN      NaN
## X2           0.81492        NaN     NaN      NaN
## X3          -0.56052        NaN     NaN      NaN
## X4           0.72667        NaN     NaN      NaN
## X5           1.84831        NaN     NaN      NaN
## X6           0.05759        NaN     NaN      NaN
## X7          -1.21460        NaN     NaN      NaN
## X8          -1.30908        NaN     NaN      NaN
## X9          -1.39005        NaN     NaN      NaN
## X10               NA         NA      NA       NA
## X11               NA         NA      NA       NA
## X12               NA         NA      NA       NA
## X13               NA         NA      NA       NA
## X14               NA         NA      NA       NA
## X15               NA         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 9 and 0 DF,  p-value: NA
```

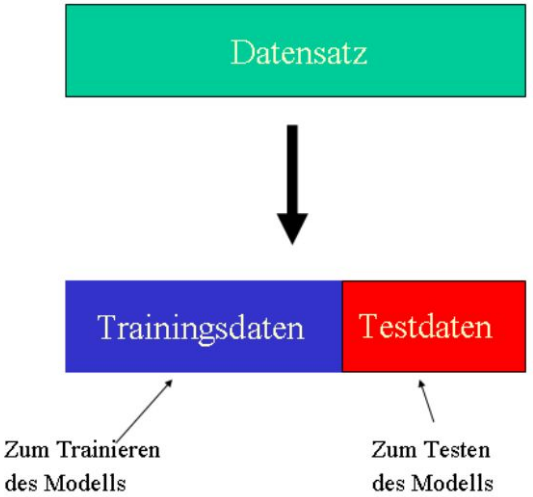
- Perfekter Fit: “no residual degrees of freedom”
- Koeffizienten nicht definiert wegen “singularities”
- p>N impliziert $X^TX$ **singulär**

```
x <- model.matrix(fit15)
det(t(x)%*%x)
```

```
## [1] -2.8449e-81
```

# Was ist ein gutes Modell? Der Generalisierungsfehler

- Ziel ist ein Modell mit guter Vorhersage
- Neuer Input $x_{new}$; Vorhersage $\hat{y} = x_{\text{new}}^T \hat{\beta}$
- Generalisierungsfehler: $y_{\text{new}} - \hat{y}$




- In der Praxis: Training- & Testdaten
- Root-mean-squared-error misst die zu erwartende Abweichung der Vorhersage

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n_{\text{test}}} (y_{\text{test,i}} - \hat{y}_i)^2}{n_{\text{test}}}}$$

# Generalisierungsfehler

$$= \sigma_{\varepsilon}^{2} + \text{Varianz} + \text{Bias}^2$$

$\sigma_{\varepsilon}^{2}$: inhärente Fehler "Noise"

Varianz: = wie "variable" (zerstreut) ist die Vorhersage?

Bias$^2$: = systematische Verzerrung (zum wahren Wert)?

Varianz + Bias$^2$: reduzierbarer Fehler

- Ein gutes Modell versucht den Reduzierbaren Fehler zu minimieren
- Unter gewissen Annahmen: Varianz $\cong \sigma_{\epsilon}^{2} \frac{p}{N}$
- Bias-Varianz Dilemma: komplexe Modelle (viele Kovariablen) haben einen kleinen Bias, aber eine grosse Varianz

# In der Praxis…

- Training und Test Daten
- Simulieren “Dummy” Testdaten
- Wie gut sind die Modelle p=1, 4, 8 und 15?
- Beachte: der inhärente Fehler ist σ=0.5
- Für p=8 und 15: RMSE 6-8 mal grösser!

```r
# simulate test data
xtest <- matrix(rnorm(n*p),n,p)
ytest <- xtest%*%beta+rnorm(n,sd=0.5)
dtest <- data.frame(xtest)
dtest$y <- ytest

# prediction
pred1 <- predict(fit1,newdata = dtest)
pred4 <- predict(fit4,newdata = dtest)
pred8 <- predict(fit8,newdata = dtest)
pred15 <- predict(fit15,newdata = dtest)

# rmse
rmse <- data.frame(
  RMSE(pred1,ytest),RMSE(pred4,ytest),
  RMSE(pred8,ytest),RMSE(pred15,ytest)
)
```

| | p=1 | p=4 | p=8 | p=15 |
|---|---|---|---|---|
| RMSE | 0.57 | 0.72 | 3.21 | 3.9 |

# Take Home Punkte

Schwierigkeiten

- Kleinste-Quadrate-Schätzer ist Singulär für p>n
- Überanpassung an die Daten “Overfitting”; $R^2$ ist keine nützliche Größe
- Überschätzung der Regressionskoeffizienten

Konzepte

➔ 10:1 Faustregel: zur Vermeidung von Overfitting sollte p < n/10
➔ Generalisierungsfehler: Training, Testdaten, RMSE
➔ Gutes Modell = kleiner Generalisierungsfehler!
➔ Bias-Varianz Dilemma

Jetzt: Methoden die gut funktionieren mit hochdimensionalen Daten!

# Subset Regression und Modellselektion

# Regularisierte Lineare Regression
# Kleinste Quadrate unter Nebenbedingungen

- Kleinste Quadrate Verfahren als Optimierungsproblem

$$\hat{\beta} = \arg\min_{\beta} \ \mathrm{RSS}(\beta)$$

- Overfitting führt zu Überschätzung der Koeffizienten
- Methoden mit gewisse Nebenbedingungen (NB) an $\boldsymbol{\beta}$

$$\hat{\beta} = \arg\min_{\text{NB für } \beta} \ \mathrm{RSS}(\beta)$$

# Subset-Regression

- Minimiere Kleinste-Quadrate unter NB: nur Koeffizienten in S sind relevant

$$\hat{\beta}_S = \underset{\beta_j = 0 \ \forall j \notin S}{\operatorname{argmin}} \operatorname{RSS}(\beta)$$

- OLS Schätzer basierend auf Subgruppe “S” von Kovariablen

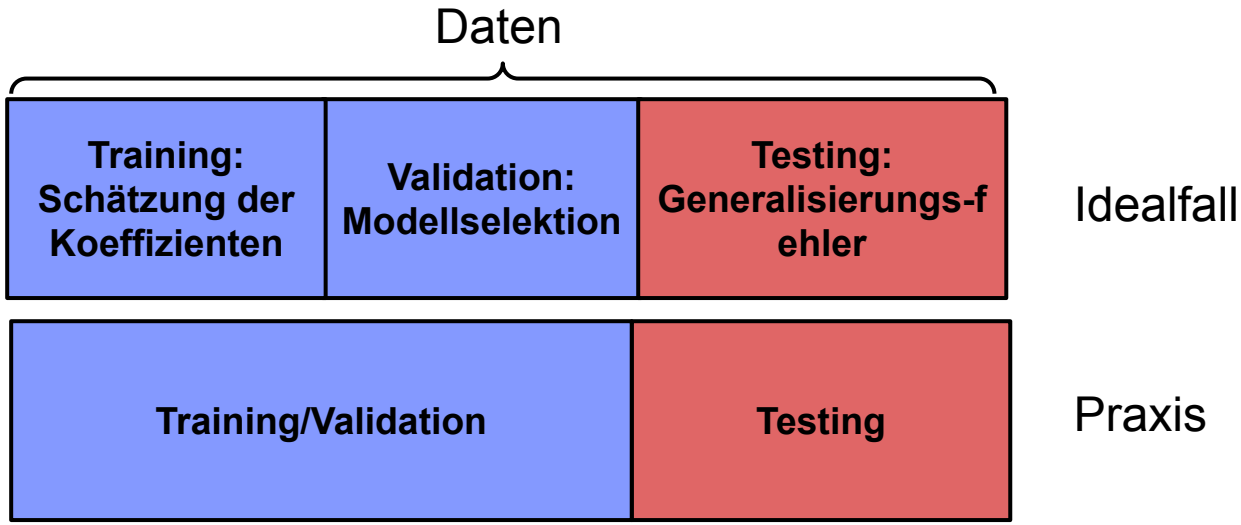
$$\hat{\beta}_S = (\mathbf{X}_S^T \mathbf{X}_S)^{-1} \mathbf{X}_S^T \mathbf{y}$$

- Betrachte Sequenz $S_1$, $S_2$,…, $S_M$

# Best Subset- und Schrittweise Regression

- Es gibt verschieden Verfahren zur Generierung von Subgruppen
  - Schrittweise Vorwärtsselektion: starte mit “leerem Modell”, füge wichtige Variablen schrittweise hinzu (F-Statistik)
  - Schrittweise Rückwärtsselektion: starte mit “vollem Modell”, eliminiere schrittweise weniger wichtige Variablen (F-statistik)
  - Alle Subset-Selektion
- Sequenz von Subgruppen $S_1, S_2, \ldots, S_M$
- Wähle die “beste” Subgruppe (bestes Modell)

# Modellselektion: Wähle das “Beste” Modell?




- Schätzung der Koeffizienten für verschiedene $S_1,...,S_M$
- Modellselektion: wähle das beste Modell $S_{opt}$

} Trainieren des Modells

- Berechne den Generalisierungsfehler von $S_{opt}$

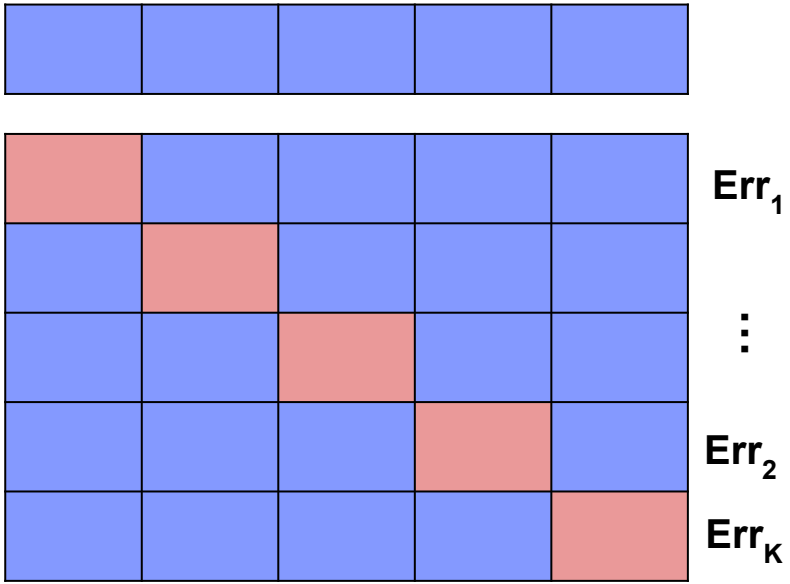
} Testen des Modells

# Modellselektion

- Bestes Modell, kleinster Generalisierungsfehler
- Modellselektion ist Teil des “Trainings”
- LÖSUNG: Approximiere den Generalisierungsfehler mittels Trainingsdaten
  - Kreuzvalidierung
  - Informationskriterium, z.B. AIC, BIC, Cp-Wert

# Kreuzvalidierung

- Teile die Trainingsdaten in K Stücke
- Wiederhole “Train/Test” K-mal
- Kreuzvalidierungsfehler (CV) = Mittelwert aller $Err_1,\ldots,Err_K$



# Informationskriterium

≈ Goodness-of-Fit + Modell Komplexität

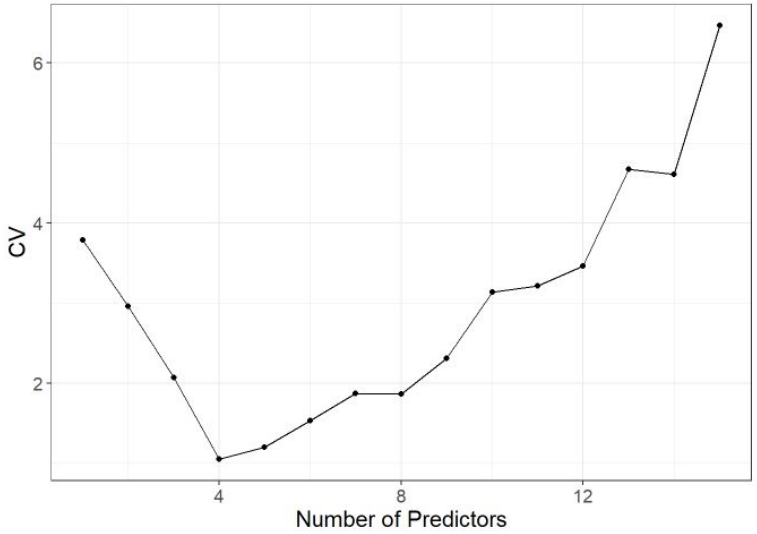
Beispiele:

$$\mathrm{Cp} = \frac{1}{\mathrm{n}}\mathrm{RSS}(\hat{\beta}) + 2\frac{\mathrm{p}}{\mathrm{n}}\hat{\sigma}_{\epsilon}$$

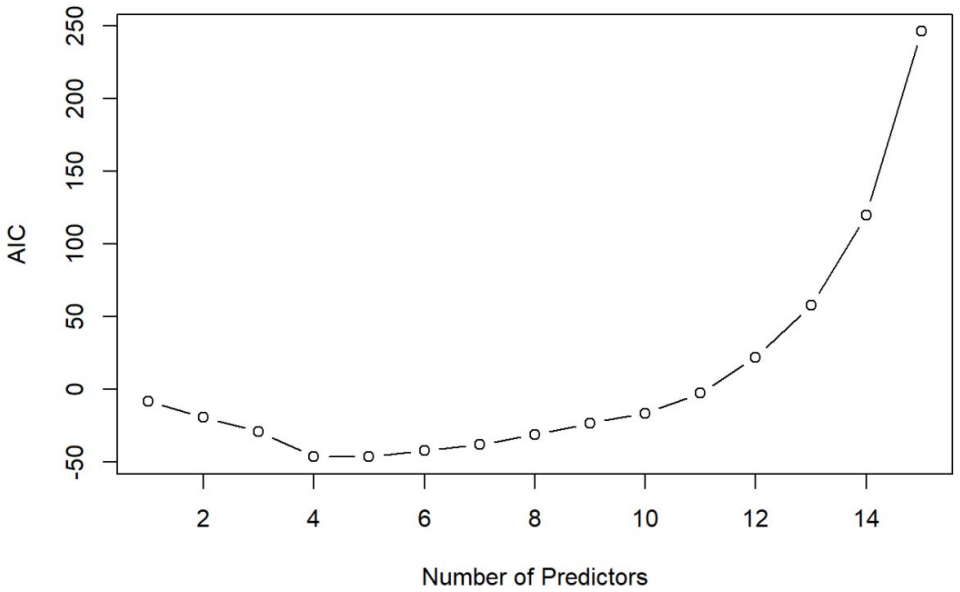
$$\mathrm{AIC} = -2\,\mathrm{LogLik} + 2\mathrm{p}$$

# Beispiel Dummy Daten

## Kreuzvalidierung




## Informationskriterium



# Beispiel Dummy Daten

- Modellselektion in R: regsubsets (leaps) und stepAIC (MASS)
- Beispiel Vorwärtsselektion mittels AIC

```
# Forward regression
fit0 <- lm(y~1,data=dtrain)
fit.fw <- stepAIC(fit0,
                  direction="forward",
                  scope=
                    list(lower=fit0,
                         upper=paste("~", paste(colnames(dtrain[,-10]),
                                                collapse=" + "))
                    ),
                  trace = FALSE
)
```

```
kable(as.data.frame(fit.fw$anova),digits=3)
```

| Step | Df | Deviance | Resid. Df | Resid. Dev | AIC |
|---|---|---|---|---|---|
| | NA | NA | 9 | 22.468 | 10.095 |
| + X1 | 1 | 20.017 | 8 | 2.450 | -10.064 |
| + X4 | 1 | 0.883 | 7 | 1.567 | -12.535 |
| + X9 | 1 | 0.376 | 6 | 1.191 | -13.277 |

```
kable(broom::tidy(fit.fw),digits=3)
```

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | 0.210 | 0.157 | 1.334 | 0.231 |
| X1 | 1.611 | 0.243 | 6.624 | 0.001 |
| X4 | -0.508 | 0.205 | -2.475 | 0.048 |
| X9 | -0.322 | 0.234 | -1.376 | 0.218 |

# Kursinhalt

- *Lineare Regression und Methode der Kleinsten Quadrate*
- *Überanpassung, Generalisierungsfehler und Bias-Varianz Dilemma*
- *Subset Regression und Modellselektion*
- Modellselektion, Regularisierung und Ridge Regression
- Schrumpfung und Hauptachsen, Effektive Freiheitsgrade, Bayessche Inferenz, Smoothing Splines
- *Lasso, Elasticnet und hochdimensionale P-Werte*
- *Klassifikation, Logistische Regression & Elasticnet*
- *Maschinelles Lernen: Entscheidungsbäume, Random Forest und AdaBoost*
- *Vorhersage und Feature Selektion für Ereigniszeitanalyse*
- *Multiples Testen, Bonferroni und FDR Korrektur, Schrumpfung der Varianz*

# Regularisierte Lineare Regression
# Kleinste Quadrate unter Nebenbedingungen

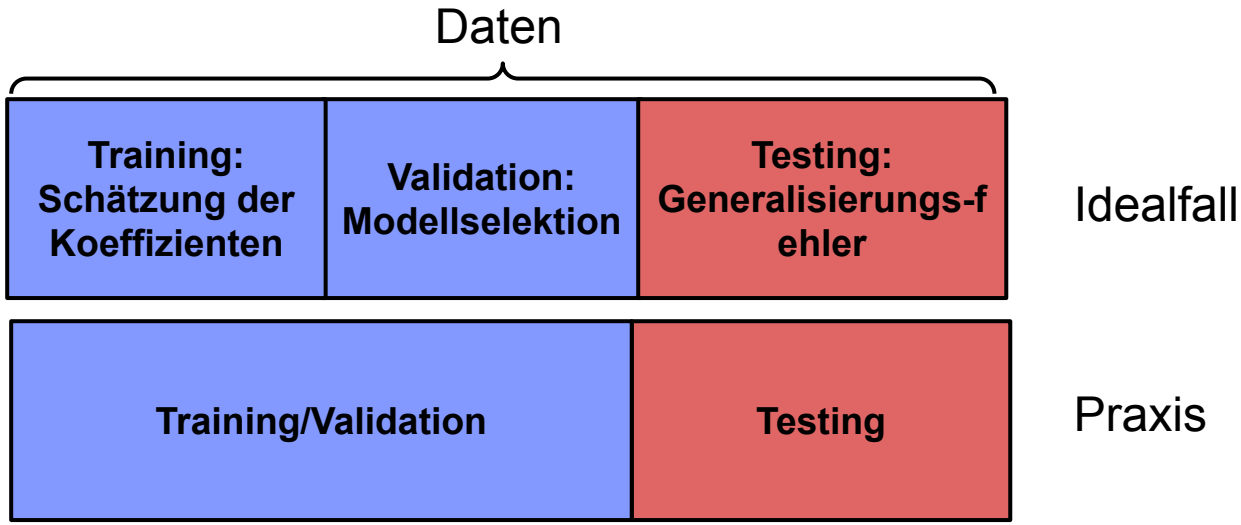
- Kleinste Quadrate Verfahren als Optimierungsproblem

$$\hat{\beta} = \arg\min_{\beta} \ \mathrm{RSS}(\beta)$$

- Overfitting führt zu Überschätzung der Koeffizienten
- Methoden mit gewisse Nebenbedingungen (NB) an $\boldsymbol{\beta}$

$$\hat{\beta} = \arg\min_{\text{NB für } \beta} \ \mathrm{RSS}(\beta)$$

# Modellselektion: Wähle das “Beste” Modell?




- Schätzung der Koeffizienten für verschiedene $S_1,...,S_M$
- Modellselektion: wähle das beste Modell $S_{opt}$

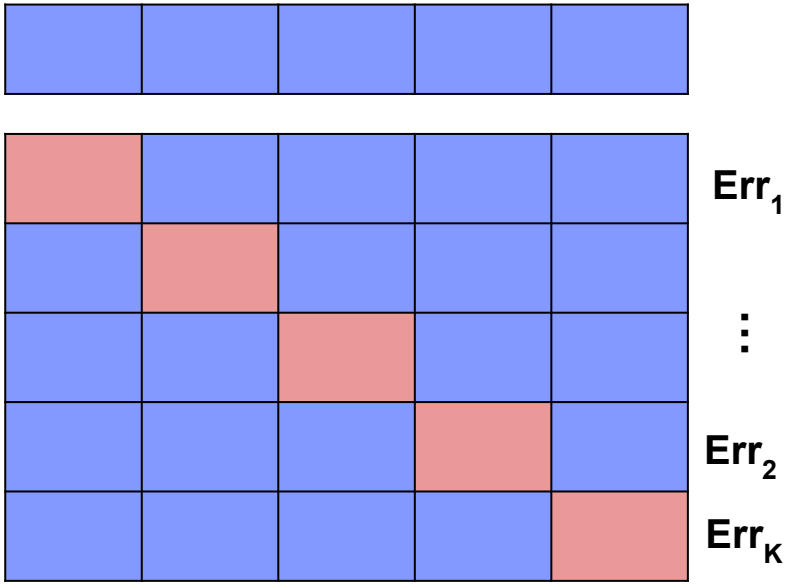
} Trainieren/Validieren des Modells

- Berechne den Generalisierungsfehler von $S_{opt}$

} Testen des Modells

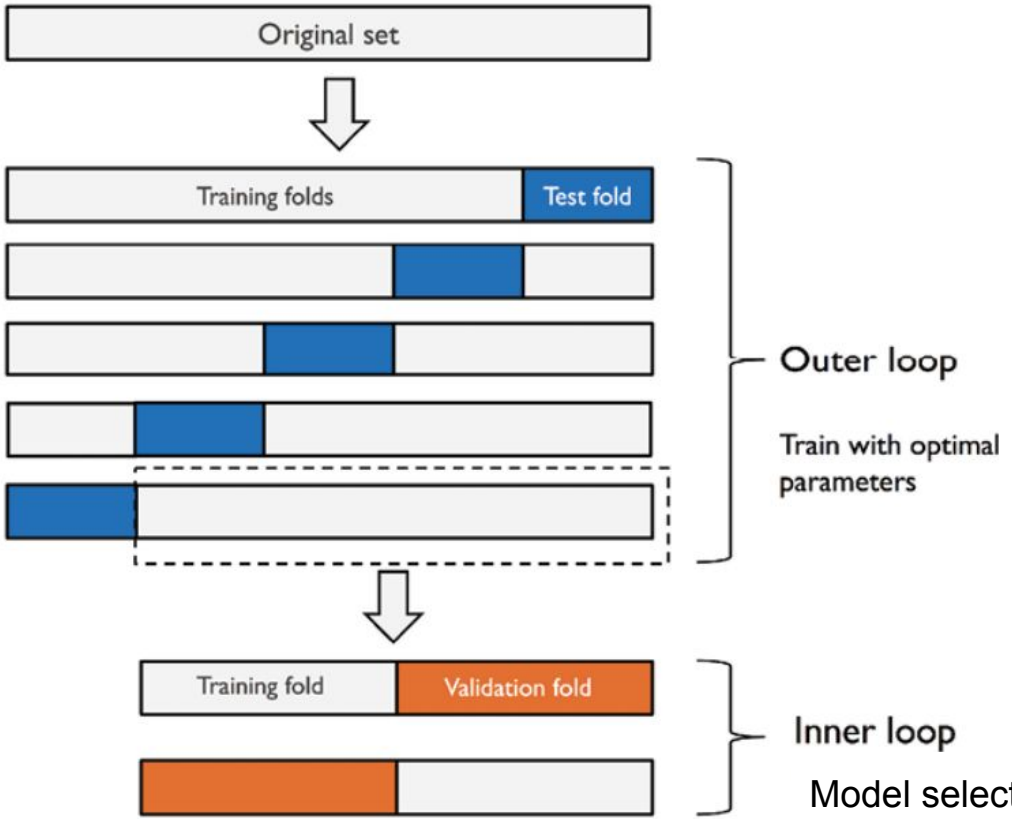
# Kreuzvalidierung

- Teile die Trainingsdaten in K Stücke
- Wiederhole “Train/Test” K-mal
- Kreuzvalidierungsfehler (CV) = Mittelwert aller $Err_1,\ldots,Err_K$




Wichtig:
Kreuzvalidierungsfehler entspricht nicht dem Generalisierungsfehler

# Nested Cross-Validation



# Ridge Regression

# Ridge Regression

- Nebenbedingung: beschränke die Grösse der Koeffizienten

$$\|\beta\|_2^2 = \sum_{j=1}^{p} \beta_j^2 \leq c$$

“L2-Norm”

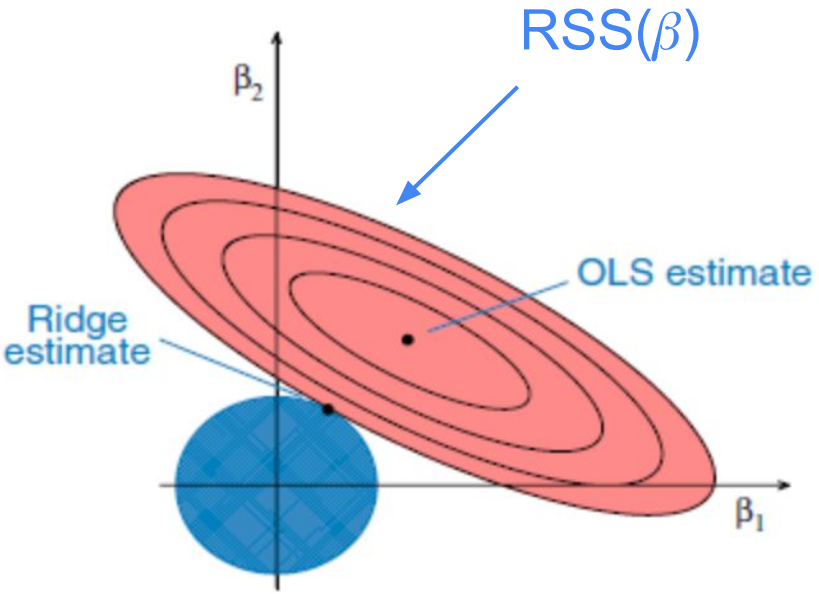
- Optimierungsproblem:

$$\hat{\beta}_c^{\text{Ridge}} = \arg \min_{\|\beta\|_2^2 \leq c} \text{RSS}(\beta)$$

# Ridge Regression

Geometrische Anschauung:

$$\hat{\beta}_c^{\text{Ridge}} = \arg \min_{\|\beta\|_2^2 \leq c} \text{RSS}(\beta)$$




- RSS(β) wird minimiert unter der Nebenbedingung dass β im “blauen” Kreis zu liegen kommt
- Der OLS Schätzer wird Richtung Nullpunkt geschrumpft

# Ridge Regression - L2 Penalization

- Nebenbedingung als Penalty (“Lagrange Multiplikator”)
- Alternative (äquivalente) Formulierung

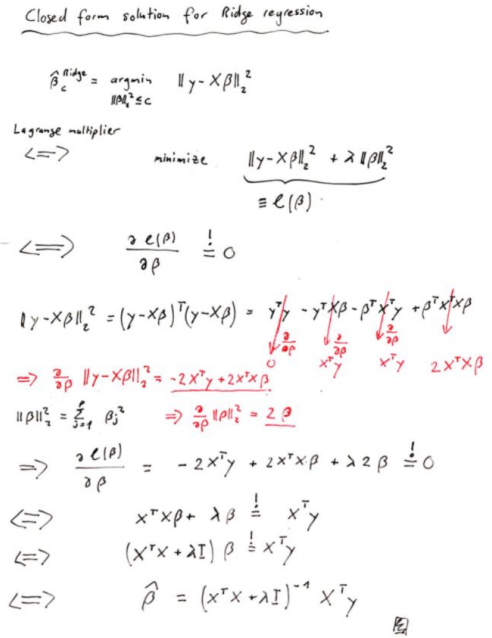
$$\hat{\beta}_{\lambda}^{\text{Ridge}} = \arg\min_{\beta} \underbrace{\text{RSS}(\beta)}_{\text{Goodness-of-Fit}} + \underbrace{\lambda \|\beta\|_2^2}_{\text{Modellkomplexität}}$$

Regularisierungs-, Strafterm
Lambda: Tuning-Parameter

# Ridge Regression - Analytischen Lösung

$$\hat{\beta}_{\lambda}^{\text{Ridge}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}.$$

- Invertierbar auch für p>n
- Lambda=0: Ridge=OLS estimate



# Wahl des Tuning-Parameters (Modellselektion)

- Berechne den Ridge-Schätzer für eine Sequenz von Lambda's ("äquivalent zu der Sequenz von Subgruppen")

  $0<\lambda_1 < \lambda_2 <\ldots < \lambda_M < \infty$

- Kreuzvalidierung zur Ermittlung eines optimalen Lambda (oder Informationskriteriums)
- Glmnet package für Ridge Regression in R
  - Funktion glmnet: berechnet Ridge-Schätzer für eine Sequenz von Lambdas
  - Funktion cv.glmnet: Optimales lambda mittels Kreuzvalidierung

# Simuliere künstliche Daten

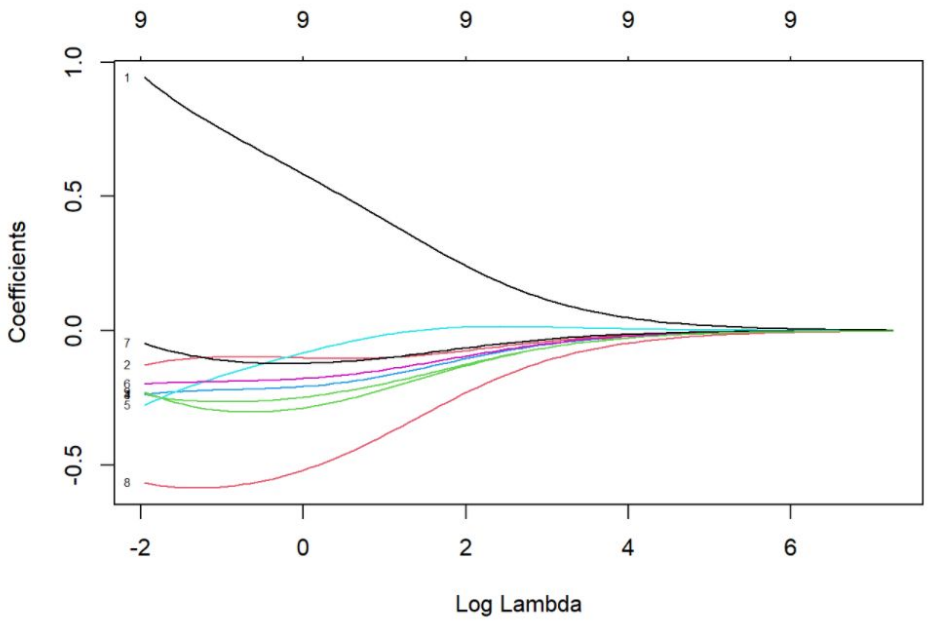
- n=10: $(Y_i, X_{i1}, \ldots, X_{ip})$ i=1..n
- p=15: $X_{i1}, \ldots, X_{ip}$ i.i.d N(0,1)
- Zielgrösse hängt nur von der ersten Kovariaten ab:

$$Y_i = \beta_1 X_{i1} + \epsilon_i,$$
$$\beta_1 = 2,\ \epsilon_i \sim N(0, 0.5^2)$$

```
set.seed(1)

n <- 10

p <- 15

beta <- c(2,rep(0,p-1))


# simulate covariates

xtrain <- matrix(rnorm(n*p),n,p)

ytrain <- xtrain%*%beta+rnorm(n,sd=0.5)

dtrain <- data.frame(xtrain)

dtrain$y <- ytrain
```

# Ridge Regression - Dummy Daten

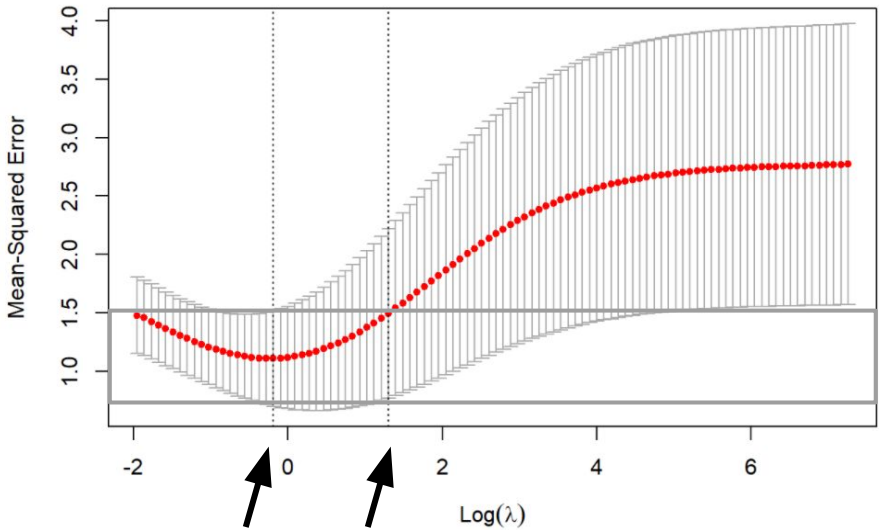
```
fit.ridge.glmnet <-glmnet(x=xtrain,y=ytrain,alpha=0)
plot(fit.ridge.glmnet,xvar="lambda",label=TRUE)
```




- Wichtig: wähle alpha = 0 für Ridge Regression
- Sogenannter “Trace plot”
- Für steigendes Lambda werden die Koeffizienten zum Nullpunkt geschrumpft
- Erster Koeffizient ist der wichtigste (wie erwartet!)

# Ridge Regression - Dummy Daten

```
cv.ridge.glmnet <-cv.glmnet(x=xtrain,y=ytrain,alpha=0)
plot(cv.ridge.glmnet)
```




- Kreuzvalidierungs Plot
- Lambda.min: kleinster CV Fehler
- Lambda.1se: grösstes lambda innerhalb 1 SE des kleinsten CVs

```
cv.ridge.glmnet$lambda.min
```

```
## [1] 0.8286695
```

```
cv.ridge.glmnet$lambda.1se
```

```
## [1] 3.671521
```

# Diabetes Beispiel

Live Demo
mit RStudio

- n=442 Patienten mit Diabetes
- Y: Fortschreiten der Krankheit 1 Jahr nach Studienbeginn
- X: Alter, Geschlecht, BMI, Blutdruck, Blutserum Werte und quadratische Term (p=64)
- Aufgabe für den Statistiker:
  - Finde ein Modell, das den Krankheitsfortschritt (Y) vorhersagt.
  - Welche Variablen sind wichtige Faktoren?

# Diabetes Beispiel

## Training und Test Daten

```
library(lars) # lars package contains the diabetes data
data("diabetes")
data <- as.data.frame(cbind(y=diabetes$y,diabetes$x2))
colnames(data) <- gsub(":",".",colnames(data))
train_ind <- sample(seq(nrow(data)),size=nrow(data)/2)
data_train <- data[train_ind,]
xtrain <- as.matrix(data_train[,-1])
ytrain <- data_train[,1]
data_test <- data[-train_ind,]
xtest <- as.matrix(data_test[,-1])
ytest <- data_test[,1]
```

## Vorwärts Regression mittels AIC

```
# Forward regression
fit0 <- lm(y~1,data=data_train)
fit.fw <- stepAIC(fit0,direction="forward",
                  scope=list(lower=fit0,
                             upper=paste("~",
                                         paste(
                                           colnames(data_train[,-1]),
                                           collapse=" + ")
                             )
                  ),
                  trace = FALSE
)
```

# Diabetes Beispiel

## Vorwärtsselektion mittels AIC

| Step | Df | Deviance | Resid. Df | Resid. Dev | AIC |
|---|---|---|---|---|---|
| | NA | NA | 220 | 1262297.5 | 1913.71 |
| + bmi | 1 | 434735.33 | 219 | 827562.1 | 1822.40 |
| + ltg | 1 | 155835.95 | 218 | 671726.2 | 1778.30 |
| + age.sex | 1 | 47106.62 | 217 | 624619.6 | 1764.23 |
| + map | 1 | 29740.28 | 216 | 594879.3 | 1755.45 |
| + bmi.glu | 1 | 22952.37 | 215 | 571926.9 | 1748.75 |
| + hdl | 1 | 19077.03 | 214 | 552849.9 | 1743.25 |
| + sex | 1 | 15702.72 | 213 | 537147.2 | 1738.89 |
| + hdl.tch | 1 | 9543.83 | 212 | 527603.3 | 1736.92 |
| + sex.ldl | 1 | 5735.62 | 211 | 521867.7 | 1736.51 |
| + tch.ltg | 1 | 6279.00 | 210 | 515588.7 | 1735.83 |
| + age.map | 1 | 5342.10 | 209 | 510246.6 | 1735.53 |

## Selektiertes Modell

| term | estimate | std.error | statistic | p.value |
|---|---|---|---|---|
| (Intercept) | 155.72 | 3.36 | 46.29 | 0.00 |
| bmi | 466.07 | 81.82 | 5.70 | 0.00 |
| ltg | 497.33 | 94.05 | 5.29 | 0.00 |
| age.sex | 274.22 | 76.35 | 3.59 | 0.00 |
| map | 315.78 | 80.98 | 3.90 | 0.00 |
| bmi.glu | 206.59 | 74.57 | 2.77 | 0.01 |
| hdl | -392.14 | 94.40 | -4.15 | 0.00 |
| sex | -201.94 | 80.87 | -2.50 | 0.01 |
| hdl.tch | -210.17 | 87.81 | -2.39 | 0.02 |
| sex.ldl | 118.77 | 74.81 | 1.59 | 0.11 |
| tch.ltg | -146.12 | 89.83 | -1.63 | 0.11 |
| age.map | 119.49 | 80.78 | 1.48 | 0.14 |

**Test Daten:** RMSE=59.9 (Modell mit allen p=64 Variablen: RMSE=84.5)

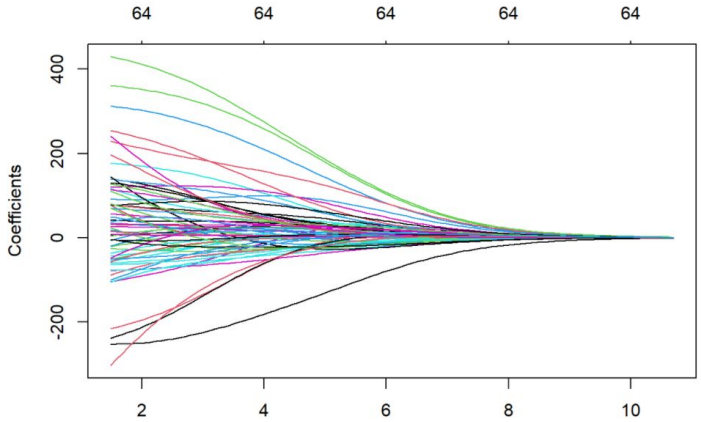


# p-Werte des selektierten Modells sind verzerrt

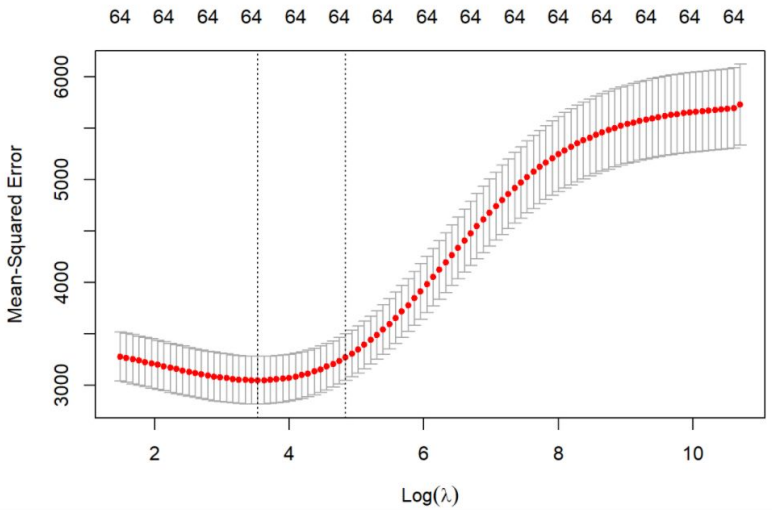
# Diabetes Beispiel

## Ridge Regression - Trace Plot

```
# Ridge
set.seed(1515)
fit.ridge <- glmnet(xtrain,ytrain,alpha=0)
fit.ridge.cv <- cv.glmnet(xtrain,ytrain,alpha=0)
plot(fit.ridge,xvar="lambda")
```




## Kreuzvalidierung




**Test Daten:** RMSE=62.63 (Modell Vorwärtsregression: RMSE=59.89)

# The Caret Package

- Caret package: Classification And REgression Training
- Funktionen zur Generierung prädiktiver Modelle:
  - Daten Splitting (Train, Test)
  - Datenvorverarbeitung (“pre-processing”)
  - Feature Selektion
  - Modell Tuning mittels Kreuzvalidierung
  - Schätzung der “Variable Importance”
- Für mehr information https://topepo.github.io/caret/
- Übungsaufgabe zum Caret package

# Eigenschaften von Ridge Regression

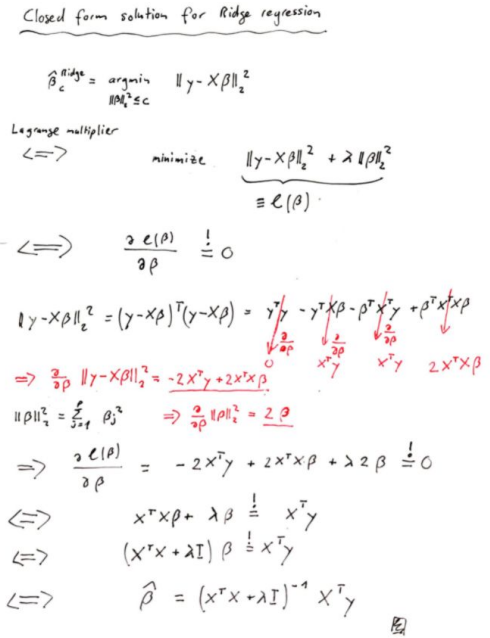
# Weitere Eigenschaften von Ridge Regression

- Analytische Lösung des Optimierungsproblems
- Schrumpfung in Richtung der Hauptkomponenten
- Die Effektiven Freiheitsgrade
- Schrumpfung und Bayesianischen Statistik

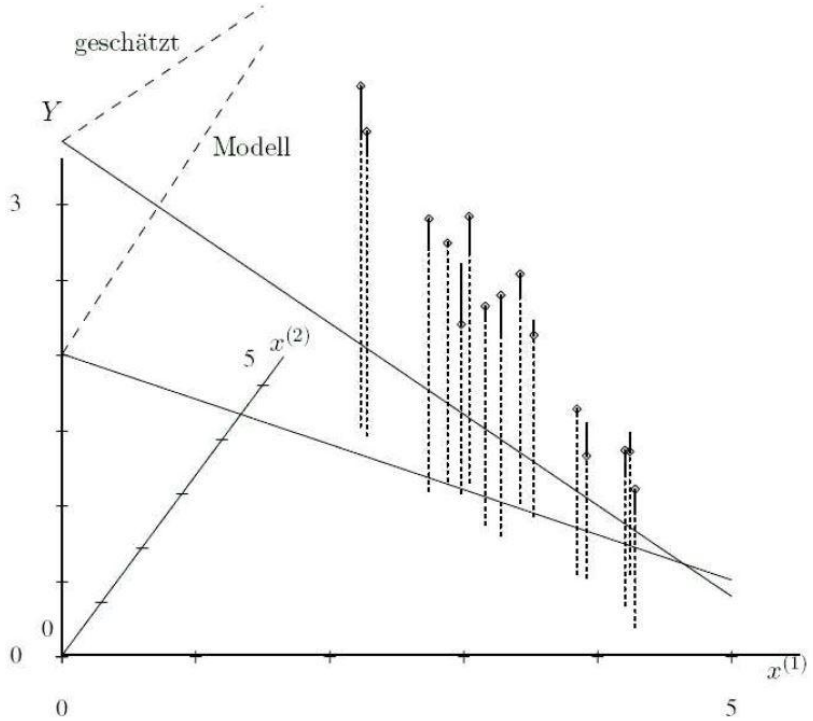
# Ridge Regression - Analytischen Lösung (sh Übung)

$$\hat{\beta}_{\lambda}^{\text{Ridge}} = (\mathbf{X}^{\mathbf{T}}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y}.$$

- Invertierbar auch für p>n
- Lambda=0: Ridge=OLS estimate

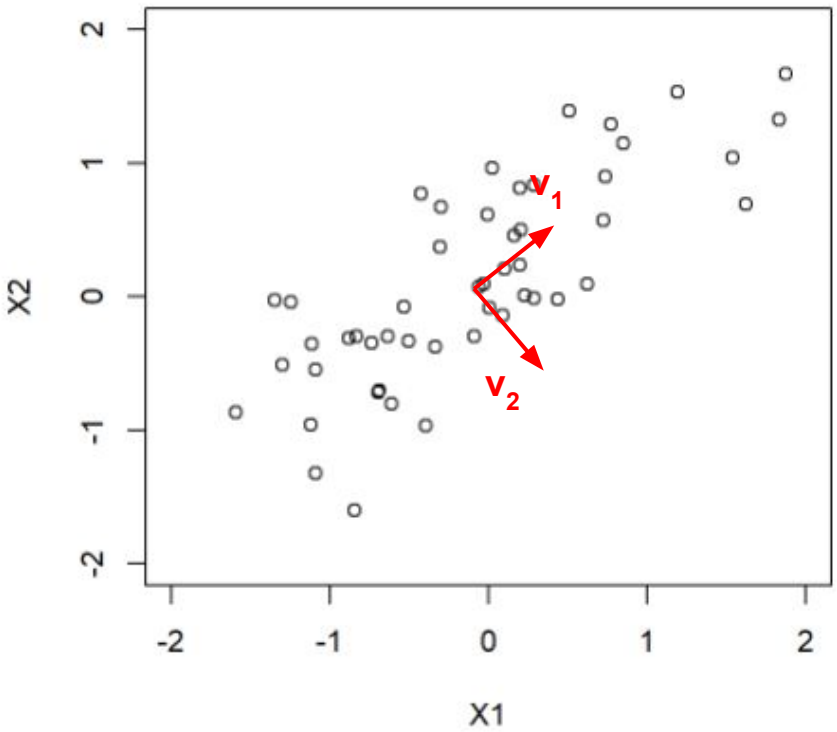


# Kollinearität




- Korrelation zwischen $X_1$ und $X_2$
- Instabilität bei der Anpassung einer Hyperebene

# Kollinearität

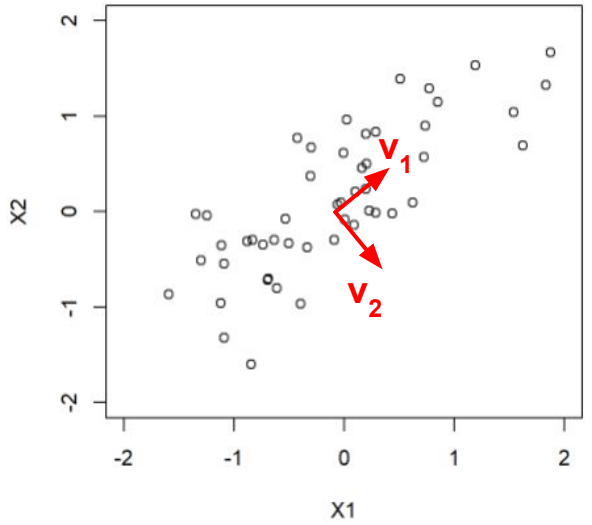


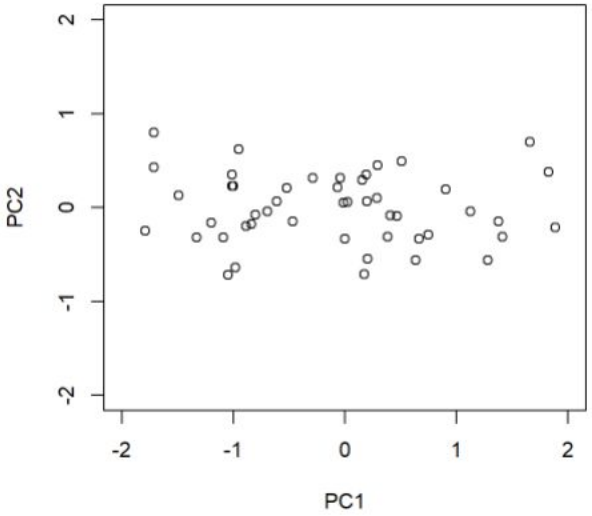

- Kollinearität der Kovariablen: Instabilität bei der Anpassung einer Hyperebene
- Hyperebene stabil in Richtung $v_1$ grosse Varianz, viel Information
- Hyperebene instabil in Richtung $v_2$ kleine Varianz, wenig Information

# Hauptachsentransformation

- Singulärwertzerlegung $\mathbf{X} = \mathbf{UDV^T}$
- Spalten von **V**: Hauptachsenrichtungen
- Spalten von **U**: Spaltenraum von **X**
- Spalten von **UD**: Hauptachsen von **X**
- Singulärwerte $d_1 \geq \ldots \geq d_p \geq 0$
- $$\hat{\mathbf{y}}^{\mathrm{OLS}} = \sum_{j=1}^{p} \mathbf{u}_j \underbrace{\mathbf{u}_j^T \mathbf{y}}$$

Koordinate bzgl orthogonaler Basis $u_j$

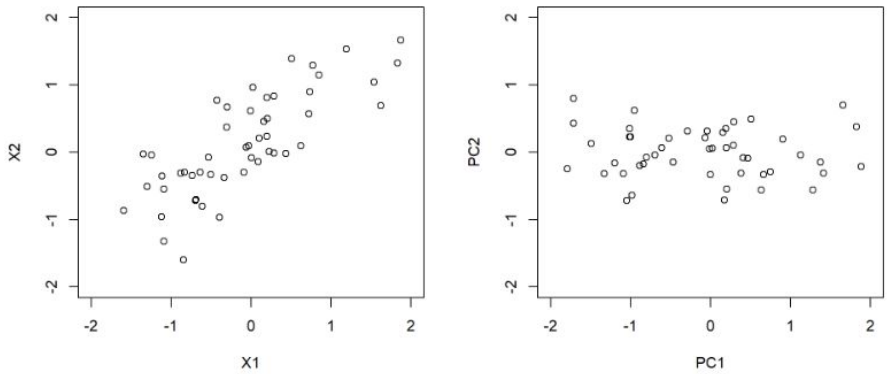


# Hauptachsentransformation in R

```r
# simulated correlated bivariate data
set.seed(1315)
n <- 50
x <- mvrnorm(n=50,mu=c(0,0),Sigma=cbind(c(1,0.8),c(0.8,1)))
colnames(x) <- c("X1","X2")

# run a principle component analysis
pc <- prcomp(x)
pc$sdev # standard dev

# same analysis using svd
cx <- sweep(x, 2, colMeans(x), "-")
sv <- svd(cx)
sqrt(sv$d^2/(nrow(x)-1))
pc$sdev
head(pc$x)
head(sv$u%*%diag(sv$d))
```

```r
par(mfrow=c(1,2))
plot(x,xlim=c(-2,2),ylim=c(-2,2))
plot(pc$x,xlim=c(-2,2),ylim=c(-2,2))
```

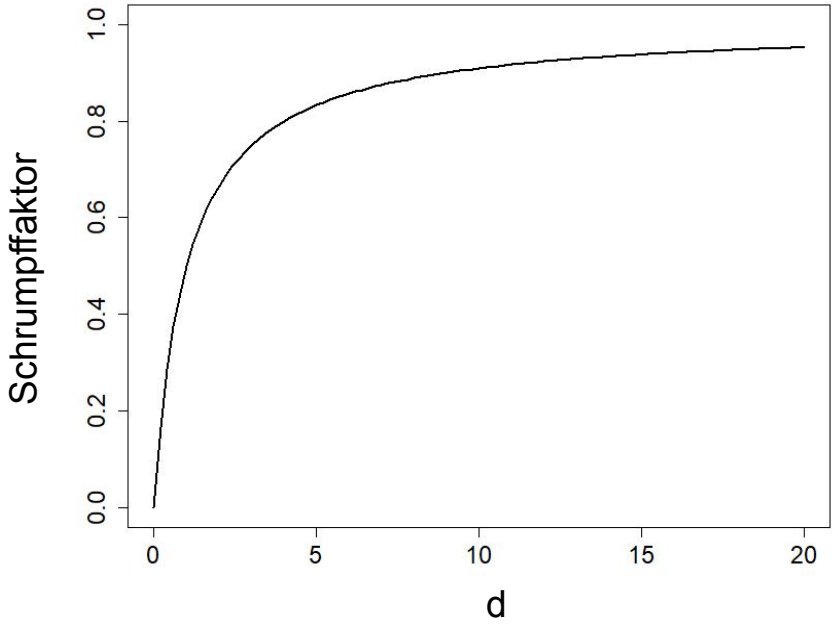


# Schrumpfung in Richtung der Hauptachsen

- Ridge Regression

$$\hat{\mathbf{y}}^{\text{Ridge}} = \sum_{j=1}^{p} \mathbf{u}_j \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u}_j^T \mathbf{y}.$$

➔ Stärkste Schrumpfung in Richtung letzter Hauptachse (kleinste Varianz, geringste Information)



# Die Effektiven Freiheitsgrade

- Multiple Lineare Regression: p Freiheitsgrade (DF)
- Was sind die Freiheitsgrade für Ridge Regression? $DF_{Ridge}$=p ???
- Aufgrund der Nebenbedingungen sollten die “effektiven” Freiheitsgrade von Ridge kleiner als p sein

- Allgemeine Definition der effektiven Freiheitsgrade
  - für sog “Lineare Fitting Methoden”, dh $\hat{\mathbf{y}} = \mathbf{S}\mathbf{y}$
  - Effektive Freiheitsgrade: $\nu_{\mathbf{S}} = trace(\mathbf{S})$
  - Beispiele: Lineare Regression und Ridge Regression

# Die Effektiven Freiheitsgrade

- Für Ridge Regression kann zeigen:

$$\nu_\lambda^{\text{ridge}} = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2+\lambda}$$

- Dummy Daten:

```
# get singular values
fit.svd <- svd(xtrain) #fit.svd$d


# ridge degree of freedom for lambdaopt
df_lambdaopt <- sum(fit.svd$d^2/(fit.svd$d^2+cv.ridge.glmnet$lambda.1se))
df_lambdaopt
```

```
## [1] 4.390408
```

Dummy Daten: DF≈4

# Likelihood-Theorie

- Bisher: Schätzung der Koeffizienten mittels RSS($\beta$) Kriterium
- Likelihood-Theorie
  - gegeben die Daten und eine Likelihood Funktion *p(D|$\beta$)*
  - schätze die Koeffizienten durch Maximierung der Likelihood Funktion

- Lineare Regression

$$Y_i|X_i, \beta \sim N(X_i^T\beta, \sigma^2),\ i = 1, \ldots, n$$

$$p(Y_i|X_i; \beta) = \frac{1}{2\pi\sigma^2}\exp\left(\frac{-(Y_i - X_i^T\beta)^2}{2\sigma^2}\right)$$

# Likelihood Funktion - Lineare Regression

- Likelihood Funktion

$$p(D|\beta) = p(Y_1|X_1;\beta) \times \ldots \times p(Y_n|X_n;\beta)$$

$$\log p(D|\beta) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - X_i^T\beta)^2$$

$$\max_{\beta} \log p(D|\beta) \iff \min_{\beta} \mathrm{RSS}(\beta)$$

Maximum Likelihood und Kleinste Quadrate Methode identisch

# Bayessche Statistik und Regularisierung

In Bayesianischer Statistik sind die Daten ($D$) UND die Parameter ($\beta$) Zufallsvariablen

- Likelihood: $p(D|\beta)$
- A-priori-Verteilung: $p(\beta)$
- Inferenz basiert auf A-posteriori-Verteilung $p(\beta|D) = \frac{p(D|\beta)p(\beta)}{P(D)}$

$$\log p(\beta|D) \propto \log p(D|\beta) + \log p(\beta)$$

A-Priori-Verteilung führt zur Schrumpfung der Koeffizienten

# Nebenbedingungen als A-Priori-Verteilung

$$Y_i | X_i, \beta \sim N(X_i^T \beta, \sigma^2), \; i = 1, \ldots, n$$

$$\beta_j \sim N(0, \tau^2), \; j = 1, \ldots, p$$

$$\arg\max_{\beta} \; \log p(\beta | D) \Longleftrightarrow \arg\min_{\beta} \; \mathrm{RSS}(\beta) + \frac{\sigma^2}{\tau^2} \|\beta\|_2^2$$

A-Priori-Verteilung führt zur Schrumpfung der Koeffizienten

# Bayessche Inferenz und MCMC-Verfahren

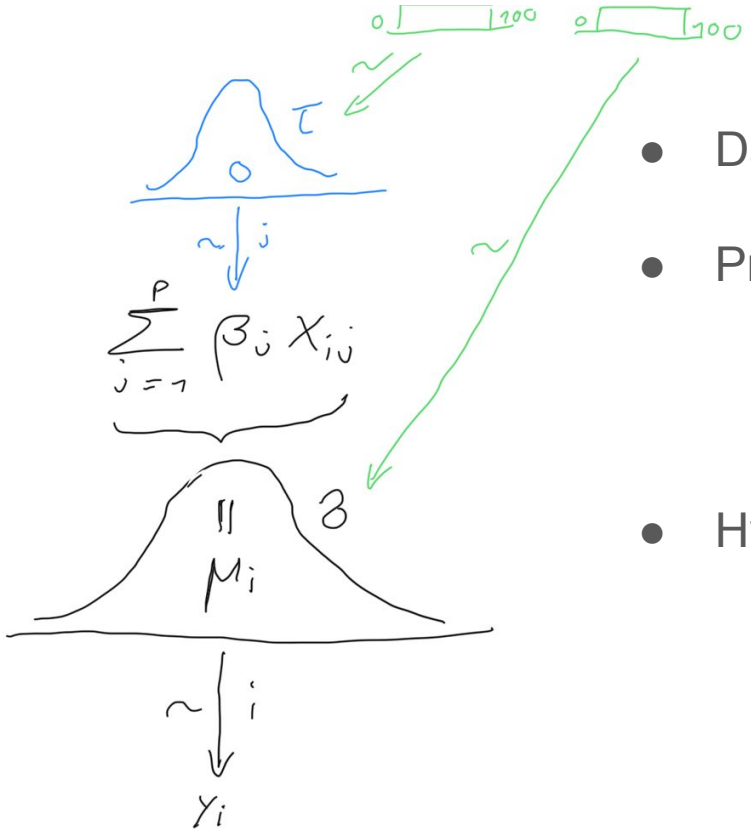
- Bayessche Inferenz: simulieren von der a-posteriori Verteilung

$$\beta^{(1)}, \ldots, \beta^{(B)} \sim p(\beta|D)$$

$$\bar{\beta} = E(\beta|D) \approx \frac{1}{B} \sum_{b=1}^{B} \beta^{(b)}$$

- A-posteriori Verteilung hat oft keine analytische Lösung → Approximation der Simulation mittels dem MCMC-Verfahren (Markov Chain Monte Carlo)
- MCMC-Verfahren mittels der Software BUGS oder JAGS

# Bayessche Ridge Regression




- Data $Y_i|X_i, \beta \sim N(X_i^T \beta, \sigma^2),\ i = 1, \ldots, n$
- Prior $\beta_j \sim N(0, \tau^2),\ j = 1, \ldots, p$

  $\sigma \sim Uni(0, 100)$
- Hyperprior $\tau \sim Uni(0, 100)$

# Bayessche Ridge Regression

## JAG Modell

```
# setup jags model
jags.m <- jags.model(textConnection(bayesian_ridge),
                     data=dat.jags,
                     inits=inits,
                     n.chains=3,
                     quiet=TRUE)
```

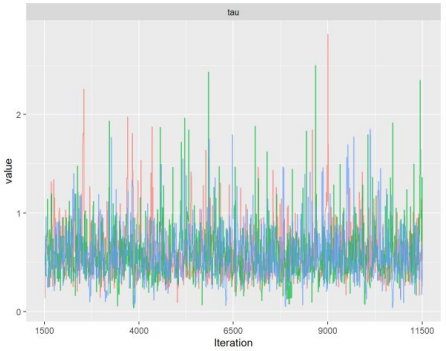
```
bayesian_ridge <-"model{
  for (i in 1:n){
    y[i] ~ dnorm (mu[i], 1/sig^2)
    mu[i] <- inprod(b,x[i,])
  }
  for (j in 1:p){
    b[j] ~ dnorm (0, 1/tau^2)
  }
  sig~dunif(0,100)
  tau~dunif(0,100)
}
"
```

## A-posteriori Samples

```
# burn-in
update(jags.m, n.iter=500)


# mcmc samples for inference
posterior.samples <- coda.samples( jags.m,
                                   variable.names = c("b","sig","tau"),
                                   n.iter=10000,thin=10) # thinning=10
```

## Traceplot

```
library(ggmcmc)
ggs.mcmc <- ggs(posterior.samples)
ggs_traceplot(ggs.mcmc,family="tau")
```



# Bayessche Ridge Regression

## Zusammenfassung MCMC

```
library(MCMCvis)
MCMCsummary(posterior.samples,
            round=2,
            params=c("sig","tau","b"))%>%
  kable
```

Gelman-Rubin Statistik:
Rhat < 1.1 “Konvergenz”

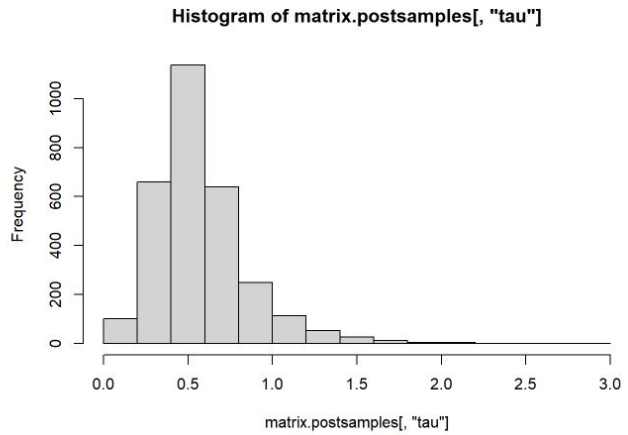
| | mean | sd | 2.5% | 50% | 97.5% | Rhat | n.eff |
|---|---|---|---|---|---|---|---|
| sig | 0.86 | 0.41 | 0.36 | 0.76 | 1.91 | 1 | 2076 |
| tau | 0.57 | 0.28 | 0.17 | 0.52 | 1.29 | 1 | 1848 |
| b[1] | 0.72 | 0.56 | -0.21 | 0.66 | 1.99 | 1 | 1865 |
| b[2] | -0.09 | 0.29 | -0.66 | -0.08 | 0.50 | 1 | 2487 |
| b[3] | -0.29 | 0.36 | -1.02 | -0.30 | 0.43 | 1 | 2900 |
| b[4] | -0.19 | 0.36 | -0.92 | -0.18 | 0.53 | 1 | 2875 |
| b[5] | -0.11 | 0.51 | -1.23 | -0.08 | 0.90 | 1 | 2366 |
| b[6] | -0.14 | 0.32 | -0.78 | -0.14 | 0.45 | 1 | 2759 |
| b[7] | -0.05 | 0.29 | -0.60 | -0.06 | 0.53 | 1 | 2278 |
| b[8] | -0.53 | 0.43 | -1.42 | -0.52 | 0.31 | 1 | 2811 |
| b[9] | -0.18 | 0.38 | -0.92 | -0.18 | 0.56 | 1 | 2321 |

## A-posteriori Verteilung

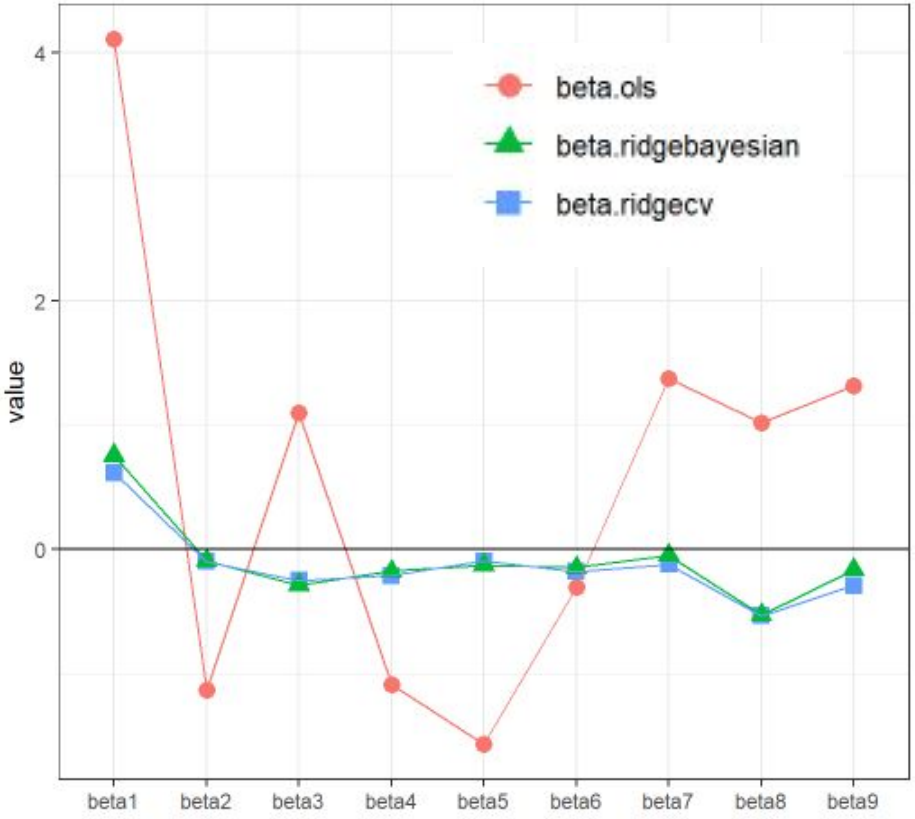
```
# posterior samples as matrix
matrix.postsamples <- as.matrix(posterior.samples)
dim(matrix.postsamples)
```

```
## [1] 3000   11
```

```
# histogram of posterior
hist(matrix.postsamples[,"tau"])
```



# Dummy Daten



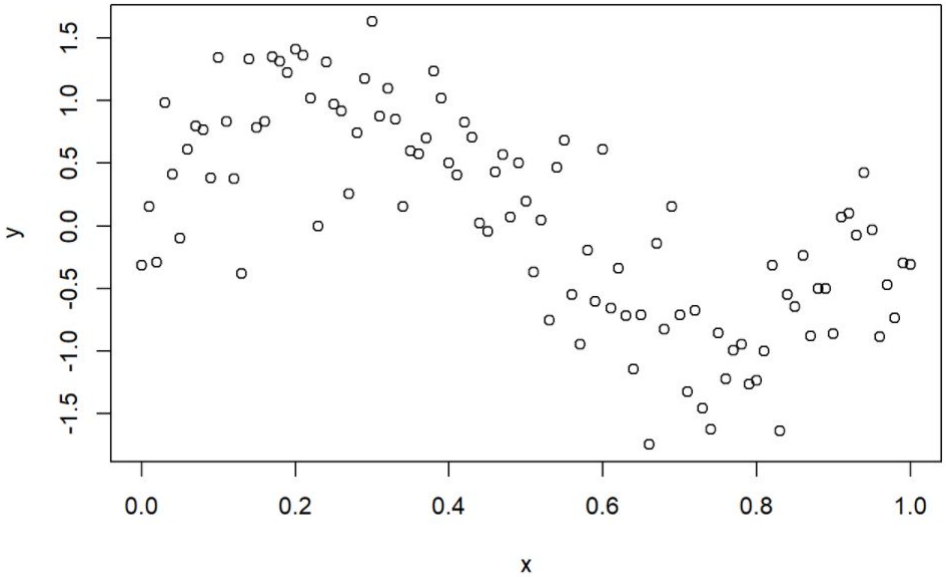

- Ridge Regression (glmnet, lambda mit CV) und Bayessche Ridge Regression fast identisch
- Schrumpfung der Koeffizienten

# Smoothing Splines und Ridge Regression

# Approximation nichtlinearer Zusammenhänge

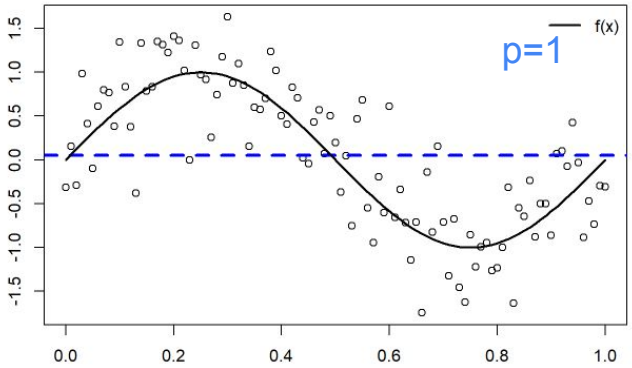
- Probleme betreffend “p gross, n klein” und deren Lösungsansätze spielen eine wichtige Rolle in vielen Disziplinen der Statistik (nicht immer offensichtlich)

- Beispiel
  - Response Y und Kovariable X
  - Nichtlinearer Zusammenhang (sinusoidal)

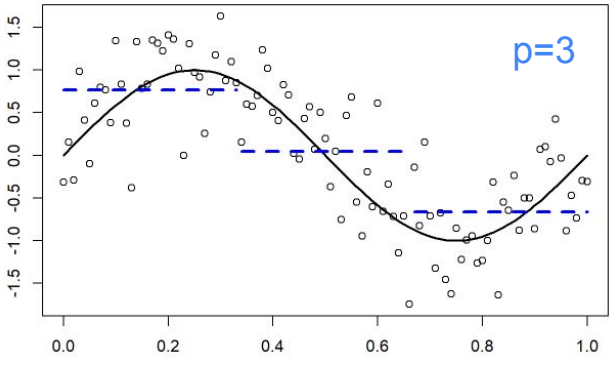


# Was ist ein Spline?

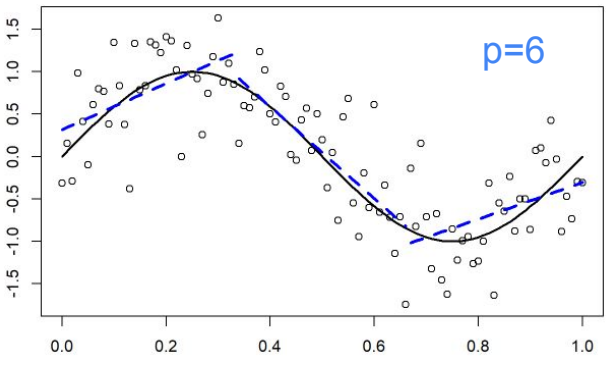
konstant




stückweise konstant




stückweise linear




- Spline: stückweise Polynome; “Glattheit” an den Knoten
- Ein Spline kann mittels sogenannter B-Spline Basisfunktionen beschrieben werden
- p=K+d+1 (K: Anz Knoten, d: Grad des Polynoms); typisch Wahl d=3 “kubisch”

$$f(X) = \sum_{m=1}^{p} \beta_m B_m(X)$$

# Spline Basis und OLS Regression

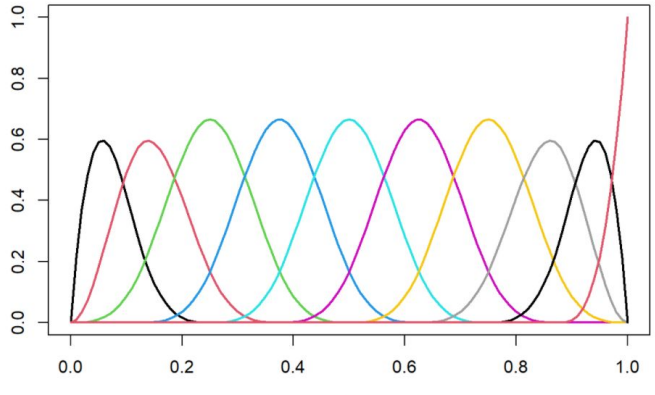
- $Y \approx \sum_{m=1}^{p} \beta_m B_m(X)$
- p=K+d+1 (K: Anz Knoten, d: Grad des Polynoms); d=3 “kubisch”

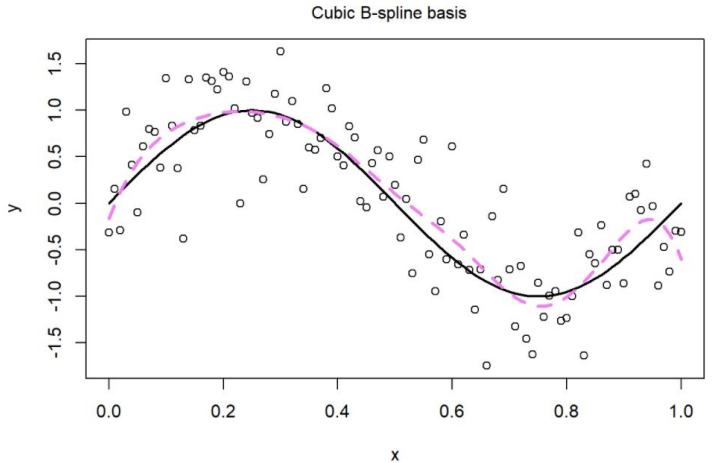
```
spl <- bs(x,df=10)
```

- Koeffizienten werden mittels OLS geschätzt

```
fit.csp <- lm(y~spl)
```

- Im Beispiel, N=100. Wähle p=10 (1:10 Regel)



# Smoothing Splines und Ridge Regression

- Nehme p=n (d.h. Maximale Anz Knotenpunkte)
- Schätze mittels Ridge Regression $\hat{\beta}_\lambda = \mathbf{argmin}\ \|\mathbf{y} - \mathbf{B}\beta\|^2 + \lambda\beta^{\mathbf{T}}\mathbf{\Omega}\beta$
- Lambda: Kreuzvalidierung oder Effektive Freiheitsgrade

```
fit.smsp.df10 <- smooth.spline(x, y, df = 10)
fit.smsp.df30 <- smooth.spline(x, y, df = 30)
fit.smsp.cv <- smooth.spline(x, y) # smoothing
```

```
fit.smsp.cv$df
```

```
## [1] 6.458247
```