

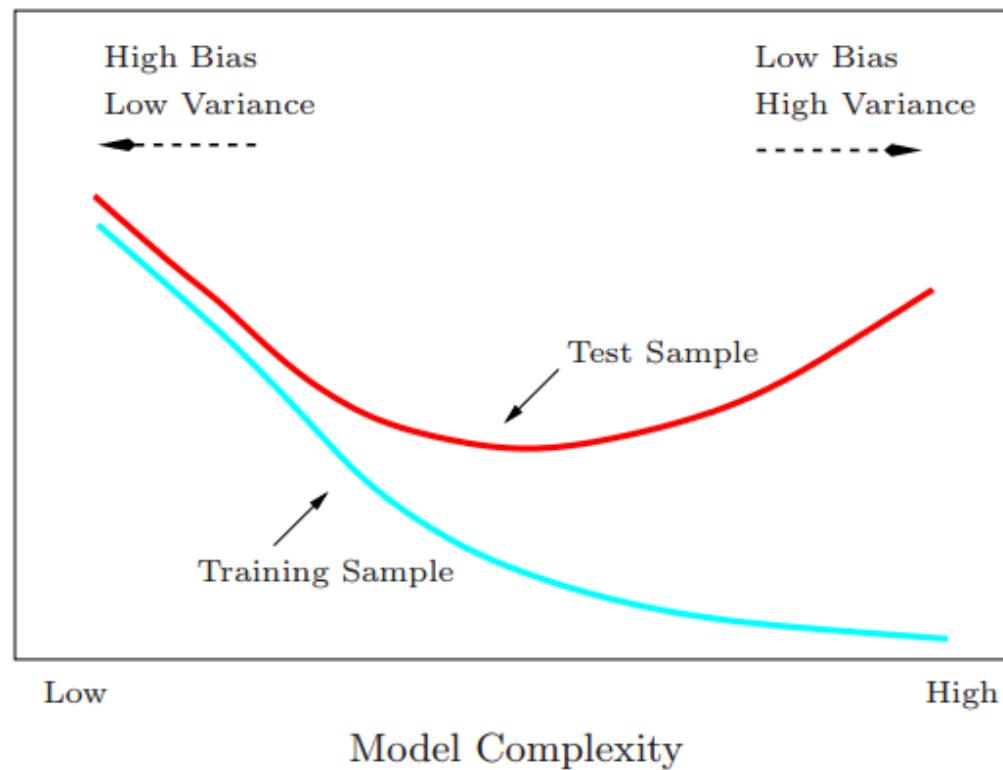
# Data Science mit R: Statistisches Lernen anwenden

Friedrich Loser

## 2. Variance-Bias-Tradeoff: Komplexitätsoptimum finden

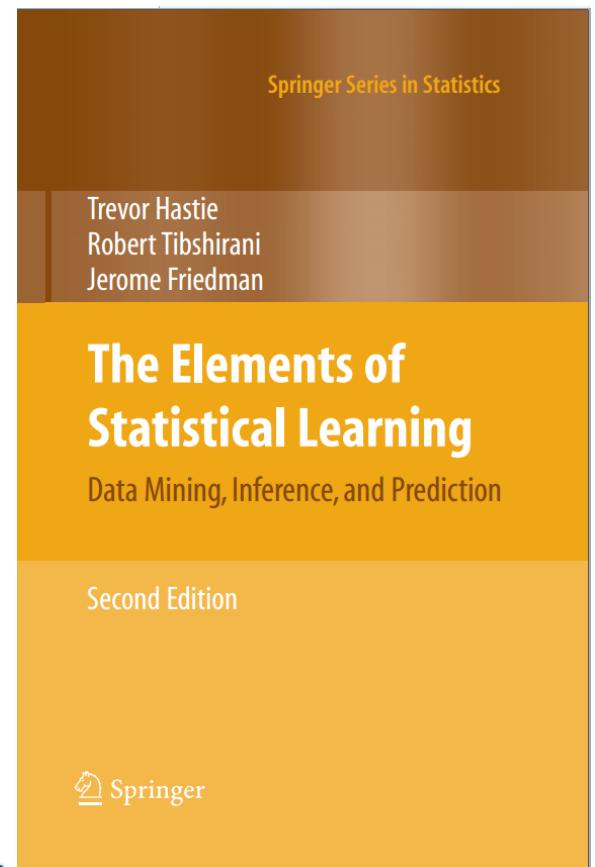
### 2. Overview of Supervised Learning

Prediction Error



Quelle:

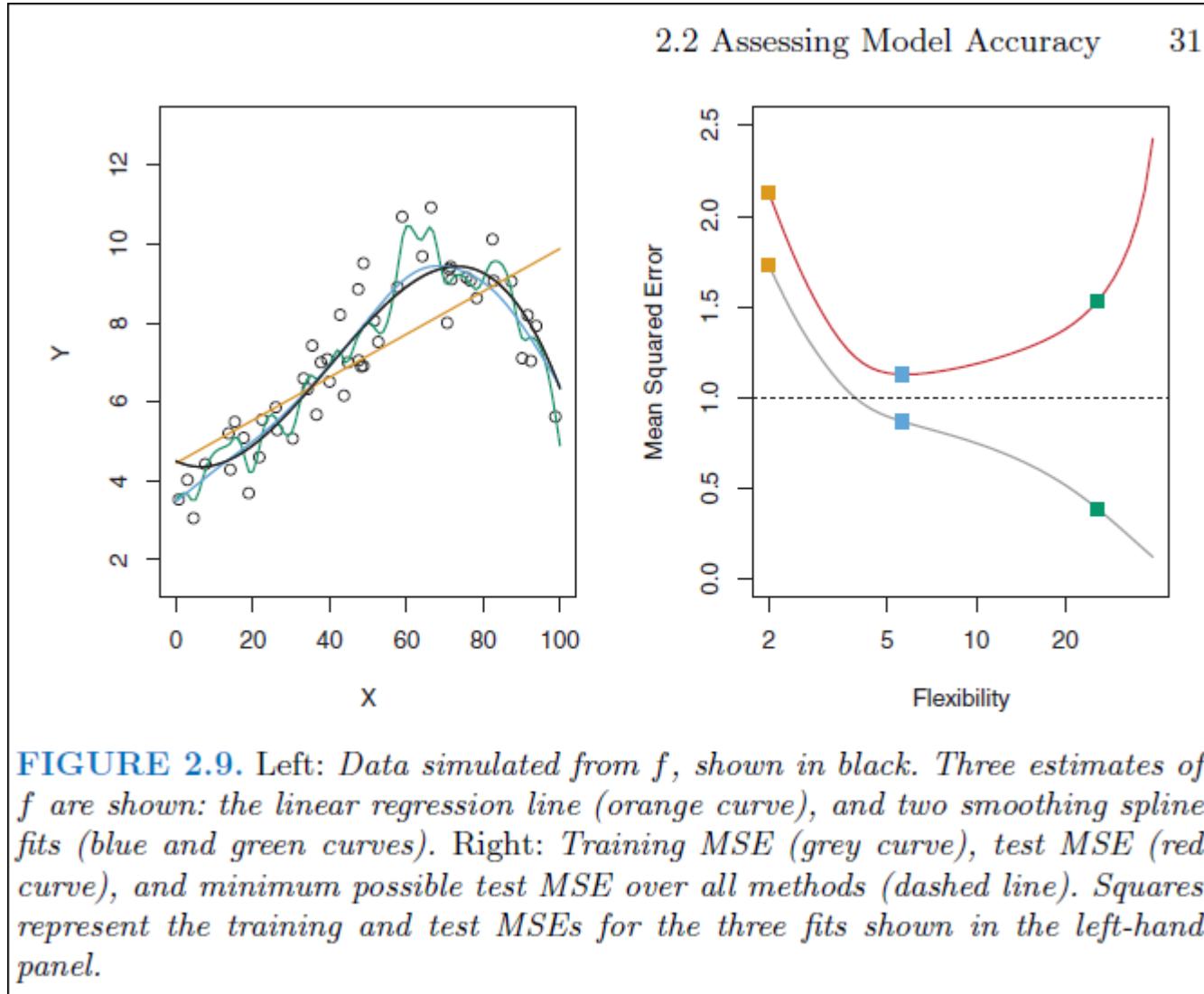
<http://statweb.stanford.edu/~tibs/ElemStatLearn/>



**FIGURE 2.11.** *Test and training error as a function of model complexity.*

2008, 764 Seiten

## 2. Variance-Bias-Tradeoff: Komplexitätsoptimum, Beispiel



Quelle:  
An Introduction to Statistical Learning  
with Applications in R

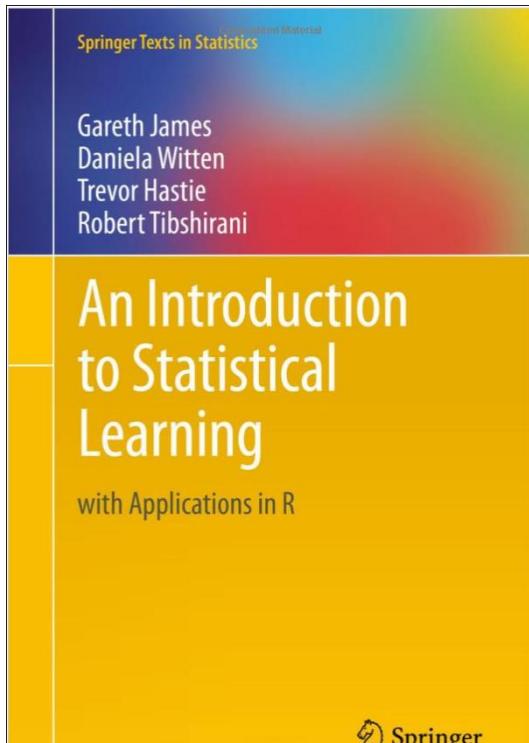
nächste  
Folie

# 3. Quellen und Links: Zum Nachschlagen

## Screenshots und R-Skripte aus:

Quelle für Kap. 7 bis 14.1:

<http://www-bcf.usc.edu/~gareth/ISL/>



2013, 440 Seiten

siehe auch:  
<https://statlearning.class.stanford.edu/>

## Data Science/-Mining/-Machine Learning mit R für Aktuare Eine kurze, praktische Einführung

Friedrich Loser, Okt. 2016

### Inhaltsverzeichnis

#### A) Überblick und erste Schritte mit R

1. Inhalt und Ziel
2. Begriffe des Statistischen Lernens
3. Quellen und weiterführende Links
4. R-Umgebung erzeugen, R kennen lernen und ausprobieren
5. Titanic-Daten einlesen, visualisieren und aufbereiten
6. R für den Aktuaralltag: Excel, SAS und Funktionen für Aktuare
7. Unüberwachtes Lernen: Clusteranalyse und Hauptkomponentenanalyse

#### B) Überwachtes Lernen: Von der Regressionsrechnung zum Machine Learning

8. Regression: Multivariate lineare Regression, Interaktionen und Polynome
9. Klassifikation: Logistische Regression und Vergleich mit k-Nearest-Neighbors
10. Sampling: Training und Test, Cross-Validation, Bootstrapping und Standardfehler
11. Regularisierung linearer Modelle: Ridge-Regression und LASSO
12. Support Vector Classifier und Support Vector Machine (SVM), ROC-Kurve
13. Entscheidungsbäume: Splits, Pruning, Bagging und Random Forests
14. Gradient Boosting Machines, Sparse Data und XGBOOST
15. Künstliche Neuronale Netze (1): Einführung, einfache Regressionsmodelle
16. Künstliche Neuronale Netze (2): Multidimensionale Klassifikation
17. Personalisierte Empfehlungen: Kollaboratives Filtern und Singulärwertzerlegung

#### C) Mit R in Richtung Big Data

18. Entziffern mit Random Forest und XGBOOST, Laufzeiten, kaggle.com
19. Handschrifterkennung mit MXNet: Deep Learning und Convolutional Neural Nets
20. GPU-Computing mit MXNet: Graphikkarten und Software
21. Big Data mit H2O: Java-Umgebung einrichten, neuronales Netz trainieren, R vs. FLOW
22. Abschluss mit und ohne R

### Anhang

- 10a. Messmethodik
- Einsparungen:
- Propensity-Score-Matching
- (Logistische Regression und Match-Stichprobe)

# 3. Quellen und Links: Websites (und Großmeister)

<https://www.kaggle.com/c/allstate-claims-severity>



Allstate®  
You're in good hands.

Dashboard

- Home
- Data
- Make a submission

Jobs • 2,663 teams  
**Allstate Claims Severity**

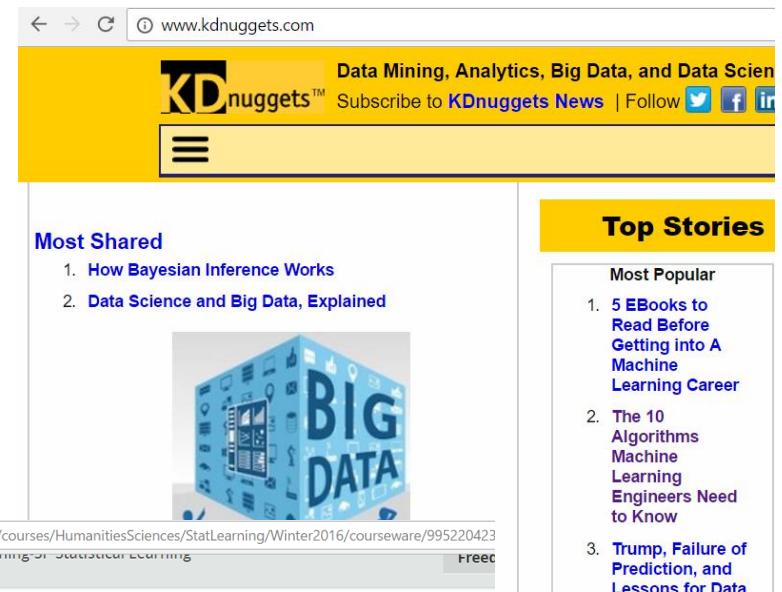
Entry Deadline

Mon 10 Oct 2016      Mon 12 Dec 2016

Competition Details » Get the Data » Make a submission

How severe is an insurance claim?

[www.kdnuggets.com](http://www.kdnuggets.com)



KDnuggets™ Data Mining, Analytics, Big Data, and Data Science  
Subscribe to [KDnuggets News](#) | Follow [Twitter](#) [Facebook](#) [LinkedIn](#)

**Most Shared**

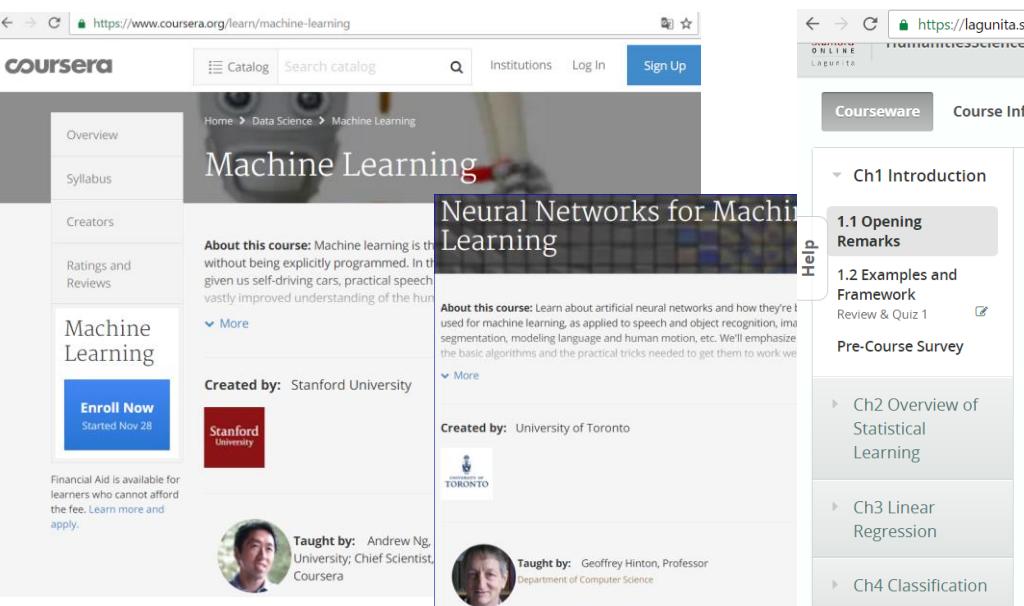
1. [How Bayesian Inference Works](#)
2. [Data Science and Big Data, Explained](#)

**Top Stories**

**Most Popular**

1. [5 EBooks to Read Before Getting Into A Machine Learning Career](#)
2. [The 10 Algorithms Machine Learning Engineers Need to Know](#)
3. [Trump, Failure of Prediction, and Lessons for Data Scientists](#)
4. [21 Must-Know Data Science Interview Questions and Answers](#)

<https://www.coursera.org/learn/machine-learning>



Coursera

Machine Learning

Neural Networks for Machine Learning

About this course: Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed. In this course, you will learn how to build neural network models, which can be applied to various tasks such as image recognition and natural language processing. You will also learn how to use these models to make predictions and decisions based on data.

Created by: Stanford University

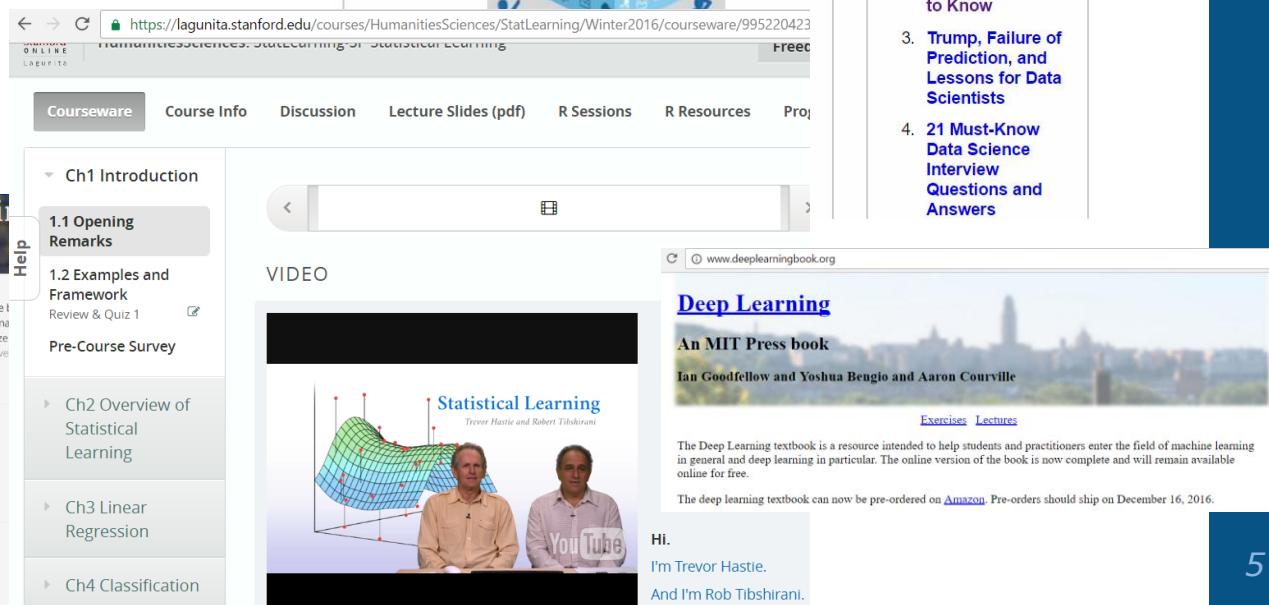
Enroll Now Started Nov 28

Financial Aid is available for learners who cannot afford the fee. Learn more and apply.

Taught by: Andrew Ng, University Chief Scientist, Coursera

Taught by: Geoffrey Hinton, Professor, Department of Computer Science

<https://lagunita.stanford.edu/courses/HumanitiesSciences/StatLearning/Winter2016/courseware/995220423>



Courseware Course Info Discussion Lecture Slides (pdf) R Sessions R Resources Projects

Ch1 Introduction

1.1 Opening Remarks

1.2 Examples and Framework

Review & Quiz 1

Pre-Course Survey

Ch2 Overview of Statistical Learning

Ch3 Linear Regression

Ch4 Classification

VIDEO

Statistical Learning Trevor Hastie and Robert Tibshirani

Deep Learning An MIT Press book Ian Goodfellow and Yoshua Bengio and Aaron Courville

Exercises Lectures

The Deep Learning textbook is a resource intended to help students and practitioners enter the field of machine learning in general and deep learning in particular. The online version of the book is now complete and will remain available online for free.

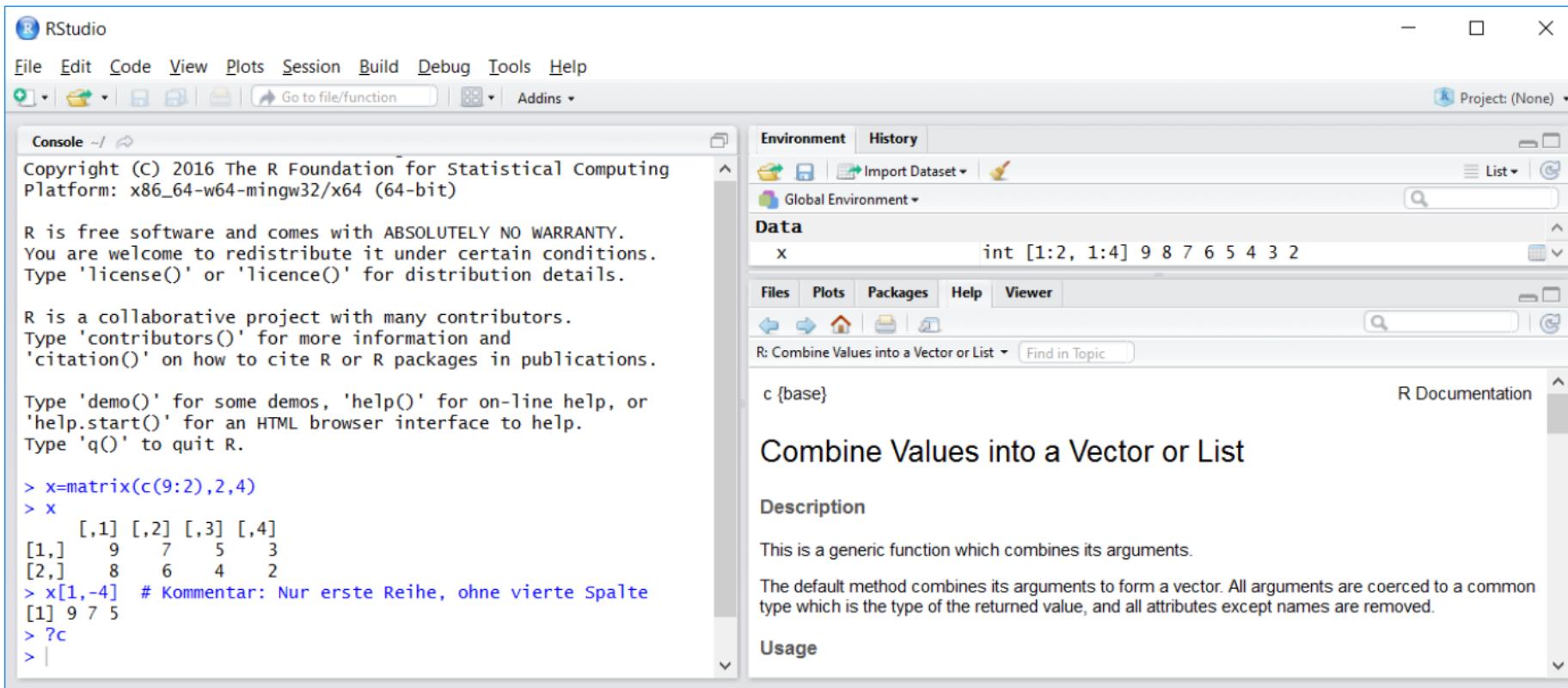
The deep learning textbook can now be pre-ordered on [Amazon](#). Pre-orders should ship on December 16, 2016.

Hi.  
I'm Trevor Hastie.  
And I'm Rob Tibshirani.

## 4. R-Umgebung erzeugen und ausprobieren

Mit folgenden Schritten kann in wenigen Minuten eine komfortable R-Umgebung aufgebaut werden:

- Die aktuelle R-Version von <https://cran.r-project.org/bin/windows/base/> herunterladen und als 64-Bit-Variante installieren (gibt es auch für Linux und OS X).
- Die ebenfalls kostenlose integrierte Entwicklungsumgebung und graphische Benutzeroberfläche RStudio von <https://www.rstudio.com> herunterladen und installieren.



The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. The left pane is the Console, displaying the R startup message, license information, and a session of R code execution. The right pane is divided into several sections: Environment (listing Global Environment), Data (showing a variable 'x' with values 9, 8, 7, 6, 5, 4, 3, 2), Files, Plots, Packages, Help, and Viewer. A search bar is at the bottom of the right pane. The bottom right corner shows the R Documentation for the 'c' function, titled 'Combine Values into a Vector or List'. The page describes 'c' as a generic function that combines arguments into a vector. The code example shown in the console is:

```

> x=matrix(c(9:2),2,4)
> x
 [,1] [,2] [,3] [,4]
[1,]    9    7    5    3
[2,]    8    6    4    2
> x[1,-4] # Kommentar: Nur erste Reihe, ohne vierte Spalte
[1] 9 7 5
> ?c
>

```

# 4. R kennen lernen: Data Frame, Funktionen, Packages

Übung: Wenden Sie auf x die Funktionen `mean`, `median`, `min`, `max`, `sd`, `var(as.vector(x))` und `sort` an.

Ein weiteres wichtiges Objekt ist der data frame. Damit können unterschiedliche Merkmalstypen, wie sie in Datensätzen regelmäßig vorkommen, in einem Objekt gespeichert werden. Die Elemente können wie bei einer Matrix angesprochen werden.

```
dat <- data.frame(sex=c("m","f","m","f"),age=c(23,45,67,89),class=c("1","3","3","2"))
dat
ran <- c(1,3,4)
dat[ran,]      # Auswahl von Datensätzen
dat[-ran,]     # Komplement dazu (z.B. Testdatei)
dat[,-2]       # Ohne zweites Merkmal (age)
```

Als nächstes wollen wir Funktionsaufrufe kennengelernt und ein paar Zufallszahlen erzeugen:

```
set.seed(3)          # Startwert setzen (für reproduzierbare Ergebnisse sorgen)
x=rnorm(100)        # 100 standardnormalverteilte Pseudozufallszahlen erzeugen
sum(x)              # Summe (Nahe Null wäre gut)
length(x)           # Länge des Vektors (100)
y=x+rnorm(100,mean=50,sd=.2)
cor(x,y)            # Korrelationkoeffizient
plot(x,y)
```

Die oben verwendeten Funktionen werden bereits mit der R-Installation mitgeliefert. Jeder R-Nutzer kann weitere, eigene Funktionen definieren und ausführen.

Eine Vielzahl von Funktionen kann gebündelt über Bibliotheken, sogenannte Packages geladen werden. Sehr einfach geht das in RStudio über das Menü „Tools“, Unterpunkt „Install Packages ...“.

# 5. Titanic-Passagierdaten einlesen, ansehen, auswerten

Die beiden heruntergeladenen Trainings- und Testdateien (siehe Validierungskonzept in Kap. 10) wollen wir nun als data frames in R einlesen und einen Blick auf die Daten werfen:

```
##R 5.1
test <- read.csv("D:/downloads/titanic_test.csv")           # Pfad/Name anpassen. Windows:
train <- read.csv("D:/downloads/titanic_train.csv")         # "\\" durch "/" ersetzen.
dim(test)      # Anzahl Datensätze und Merkmale anzeigen
dim(train)
str(train)     # Struktur des data frame anzeigen
head(train)    # Die ersten Zeilen andrucken
##t
```

Die Trainingsdatei umfasst demnach 891 Passagiere (Zeilen) und 12 Merkmale (Spalten) und beginnt mit folgenden Zeilen:

|   | PassengerId | Survived | Pclass | Name  | Sex    | Age | SibSp | Parch | Ticket           | Fare    | Cabin | Embarked |
|---|-------------|----------|--------|---|--------|-----|-------|-------|------------------|---------|-------|----------|
| 1 | 1           | 0        | 3      | Braund, Mr. Owen Harris                             | male   | 22  | 1     | 0     | A/5 21171        | 7.2500  |       | S        |
| 2 | 2           | 1        | 1      | Cumings, Mrs. John Bradley (Florence Briggs Thayer) | female | 38  | 1     | 0     | PC 17599         | 71.2833 | C85   | C        |
| 3 | 3           | 1        | 3      | Heikkinen, Miss. Laina                              | female | 26  | 0     | 0     | STON/O2. 3101282 | 7.9250  |       | S        |
| 4 | 4           | 1        | 1      | Futrelle, Mrs. Jacques Heath (Lily May Peel)        | female | 35  | 1     | 0     | 113803           | 53.1000 | C123  | S        |
| 5 | 5           | 0        | 3      | Allen, Mr. William Henry                            | male   | 35  | 0     | 0     | 373450           | 8.0500  |       | S        |
| 6 | 6           | 0        | 3      | Moran, Mr. James                                    | male   | NA  | 0     | 0     | 330877           | 8.4583  |       | Q        |

**Überlebenshäufigkeiten nach (Klasse und) Geschlecht:**

```
> prop.table(table(Sex, Survived),1)
   Survived
Sex
  0          0
  female  0.2579618 0.7420382
  male   0.8110919 0.1889081
```

**Überlebenshäufigkeiten nach (Klasse und) Geschlecht:**

```
> aggregate(Survived~Pclass + Sex, data=train, FUN=function(x) {sum(x)/length(x)})
   Pclass  Sex  Survived
1      1 female 0.9680851
2      2 female 0.9210526
3      3 female 0.5000000
4      1 male   0.3688525
5      2 male   0.1574074
6      3 male   0.1354467
```

# 6. R für den Aktuaralltag

R-Packages für Aktuare:

- [actuar](#)
- [ChainLadder](#)
- [copula](#)
- [InsuranceData](#)
- [Lifecontingencies](#)
- ...

R-Schnittstellen:

- Excel-Add-In „RExcel“
- SAS/IML und SAS Enterprise Miner: R-Skripte einbinden
- [sqldf](#): SQL-Skripte in R ausführen
- [foreign](#): Import-Funktionen für SAS-, SPSS- und STATA-Dateien

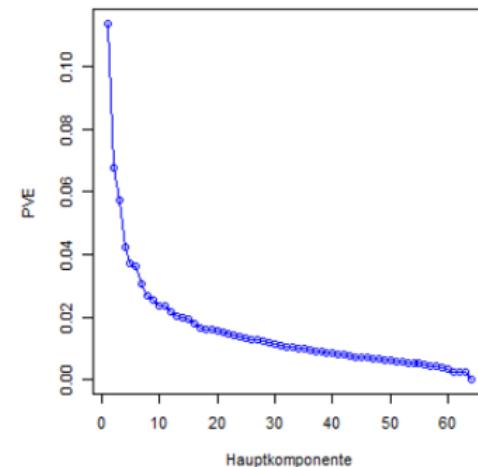
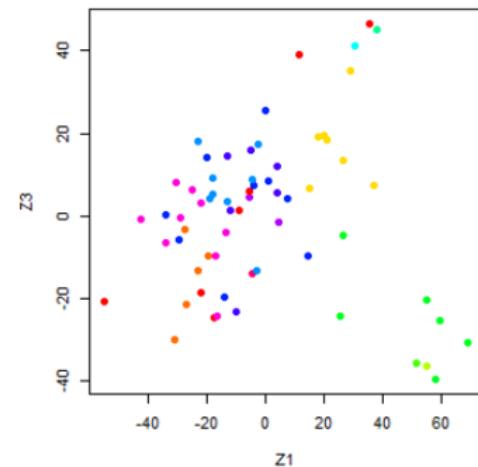
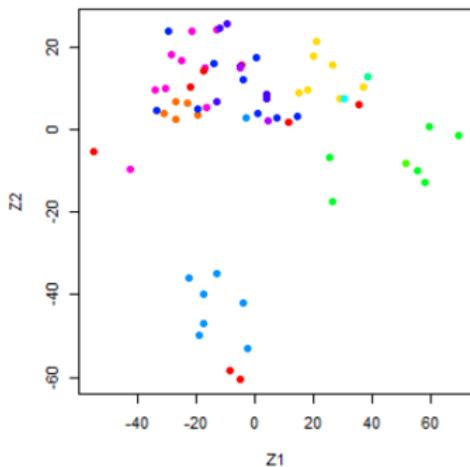
# 7. Unüberwachtes Lernen: Hauptkomponentenanalyse

- Beim unüberwachten Lernen gibt es keine vorherzusagende Zielgröße. Das Ziel ist die Beschreibung von Zusammenhängen und Mustern zwischen den beschreibenden Merkmalen.

```

library (ISLR)                                     # Enthält die NCI60-Daten.      64 Krebszelllinien mit
nci.labs=NCI60$labs                            # labs: Krebsart             je 6.830 Meßwerten
nci.data=NCI60$data                             # data: Messwerte
dim(nci.data)                                    # Anzahl Spalten und Zeilen anzeigen
table(nci.labs) |                                # Verteilung der Krebs-Typen
pr.out=prcomp(nci.data, scale=TRUE) # Hauptkomponenten berechnen
summary(pr.out)                                  # Std und Varianzanteil für 64 Komponenten
cols=function(vec) { cols=rainbow(length(unique(vec)))
  return(cols[as.numeric(as.factor(vec))]) }
par(mfrow=c(1,3))
plot(pr.out$x[,1:2], col=cols(nci.labs), pch=19, xlab="z1",ylab="z2")
plot(pr.out$x[,c(1,3)], col=cols(nci.labs), pch=19, xlab="z1",ylab="z3")
pve=pr.out$sdev^2/sum(pr.out$sdev^2)      # Anteil erklärter Varianz
plot(pve, type="o", ylab="PVE", xlab="Hauptkomponente", col="blue")
##t

```



# 8. Ausgangspunkt: Multivariate lineare Regression

Die weiteren Kapitel stellen Verfahren zum überwachten Lernen vor. Um von bekanntem Grund zu starten, beginnen wir mit einer multivariaten Regression. Dazu muss zunächst das R-Package **MASS** installiert werden. Dieses Package enthält eine große Sammlung an Datensätzen und Funktionen, darunter die Datei **Boston**, in der von 506 Bezirken um die US-amerikanische Stadt Boston die mittleren **Hauspreise medv** sowie 13 erklärende Merkmale enthalten sind.

Wir wollen sehen, wovon der Hauspreis abhängt und fitten drei lineare Modelle:

```
##R 8.1
library(MASS)
fit1=lm(medv~.,Boston)          # 1. Lineares Modell: Fit mit allen Merkmalen
summary(fit1)                   # => Adjusted R-squared = 0.7334
fit2=update(fit1,~.-age-indus)   # 2. Lineares Modell: Fit ohne die schwachen Merkmale
summary(fit2)                   # => Adjusted R-squared = 0.7344

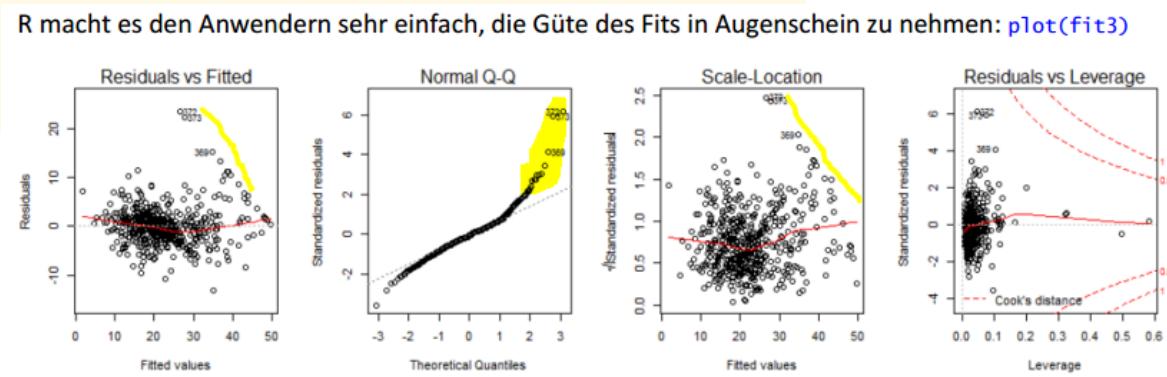
# 3. Fit mit qualitativem Merkmal, 2 Interaktionen und 2 Polynomen (in versch. Schreibweisen)
Boston$zn2=ifelse(Boston$zn>0,'c1','c0') # Qualitatives Merkmal ableiten
fit3=lm(medv~.-age-indus-lstat-zn+zn2*rm+zn2*nox+I(dis^2)+poly(lstat,5),Boston)

par(mfrow=c(1,4))
plot(fit3)
```

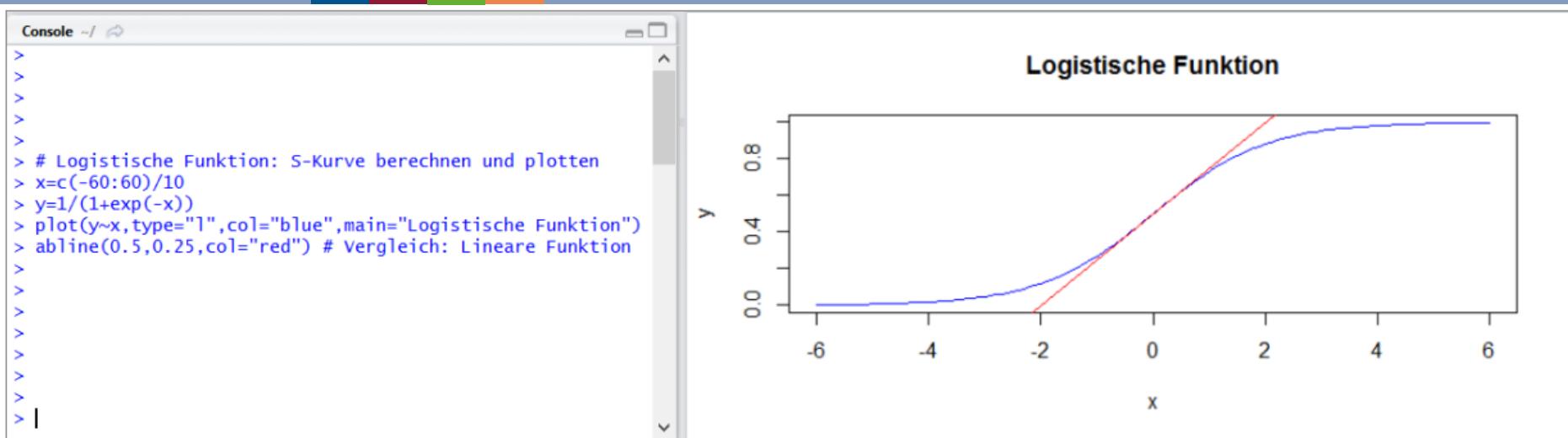
summary(fit3) # => Ergebnis:

```
poly(lstat, 5)4 2.197e+01 4.053e+00 5.421 9.34e-08 ***
poly(lstat, 5)5 -1.754e+01 3.985e+00 -4.401 1.32e-05 ***
rm:zn2c1 3.293e+00 6.534e-01 5.039 6.60e-07 ***
nox:zn2c1 1.545e+01 6.756e+00 2.286 0.022681 *
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.902 on 487 degrees of freedom
Multiple R-squared: 0.8264, Adjusted R-squared: 0.82
F-statistic: 128.8 on 18 and 487 DF, p-value: < 2.2e-16
```



# 9. Klassifikation: Logistische Regression (+Funktion)



```
library(ISLR) # Enthält den Datensatz Caravan (5.822 Personen, davon 6% Käufer, 85 Vars)
dim(Caravan)
attach(Caravan)
summary(Purchase)
test=1:1000
test.Y=Purchase[test]           # test.Y enthält tatsächliches Kaufverhalten
# Logistische Regression (Verallgemeinertes Lineares Modell)
glm.fit=glm(Purchase~.,data=Caravan,family=binomial,subset=-test)
test.probs=predict(glm.fit,Caravan[test,],type="response")
test.pred=rep("No",1000)
test.pred[test.probs >.25]="Yes" # test.pred enthält prognostiziertes Kaufverhalten
table(test.pred ,test.Y)        # 11/(11+22)=33%
```

## 9. Klassifikation: K-Nearest-Neighbors

Als Alternative zu einem Regressionsmodell, dessen umfangreiche Parameterliste wir uns hier noch nicht einmal angeschaut haben, wollen wir nun die gänzlich parameterfreie K-Nächste-Nachbarn-Methode (KNN) ausprobieren und die Ergebnisse vergleichen. Bei diesem Verfahren wird die durchschnittliche Distanz der K Beobachtungen im Trainingsdatensatz, die dem interessierenden Punkt  $y$  am nächsten liegen, berechnet. Im zweidimensionalen Fall entspricht dies einer kreisförmigen Nachbarschaft. Damit diese „rund“ ist, müssen die Input-Variablen standardisiert werden. Je höher K, umso mehr Nachbarn werden in die Durchschnittsberechnung einbezogen und umso größer sind der Glättungseffekt und die Varianz. Für die Trainingsdaten ist K=1 die optimale Lösung, denn der Schätzfehler für den Abstand zu sich selbst ist 0. Das KNN-Verfahren ist daher anfällig für das „Overfitting“ und ein Fall für die im folgenden Kapitel beschriebenen Sampling-Methoden.

(man sieht nichts, es gibt kein Modell, nur eine – hier – mäßig gute Prognose. Weiter ...)

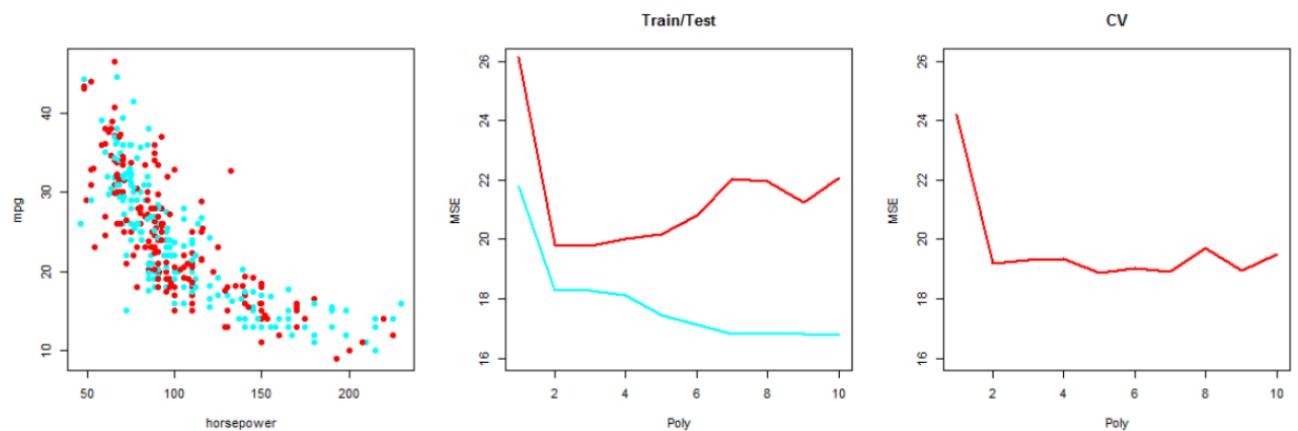
# 10. Sampling: Trainings- und Teststichprobe

Eine gute Prognose ist auf Basis der Trainingsdaten sehr einfach möglich. Es reicht, sich die Beispiele zu merken. Das fundamentale Ziel des statistischen Lernens ist jedoch, über die Beispiele hinaus zu verallgemeinern. Hier spielen die Sampling-Verfahren eine entscheidende Rolle.

In den vorangegangenen Kapiteln wurden bereits die Unterteilung der Daten in Trainings- und Teststichprobe verwendet. Das Modell wird an die Trainingsdaten „trainiert“, also angepasst. Die Prognosegüte wird dann auf Basis der Testdaten bewertet. Diesen Validierungsansatz wollen wir jetzt näher anschauen.

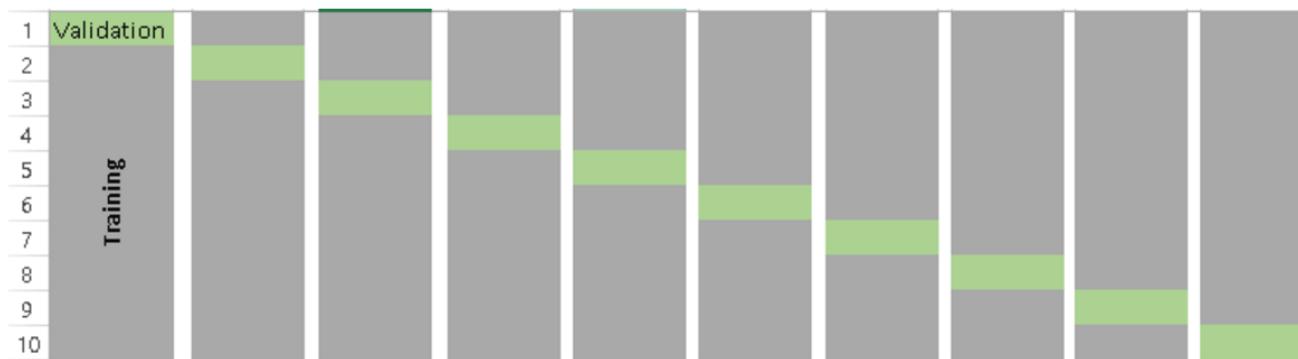
Overfitting ist nicht nur ein Problem von parameterfreien Verfahren. Auch die klassische Regression läuft bei zu vielen Parametern ins Overfitting. Das zeigen wir am Beispiel eines älteren US-amerikanischen Datensatzes mit 392 verschiedenen Autotypen, für die wir ein Modell für den Kraftstoffverbrauch in Abhängigkeit von der Motorleistung erstellen:   -> R      (das sollte dann dabei herauskommen):

Der Kraftstoffverbrauch wird dabei als Reichweite „miles per gallon“ angegeben (rot: Testdaten):



# 10. Sampling: Cross Validation (CV)

Als nächstes wollen wir den Trainings- und Testdatenansatz verallgemeinern zur K-fachen Kreuzvalidierung (Cross-Validation, CV). Hier wird jedem Datensatz eine Zufallszahl zwischen 1 und K zugeordnet. Von diesen K überschneidungsfreien Teilgruppen wird jede einmal als Testdatei verwendet, die Modellbildung also K mal auf Basis aller anderen Teildatensätze durchgeführt und das Ergebnis gemittelt. Das Ergebnis ist für unser Polynom in der vorherigen Graphik rechts abgebildet. Dazu wurde der Datensatz in 10 gleich große Stichproben unterteilt und iterativ jede Stichprobe einmal zur Testdatei. Das zu testende Modell wurde jeweils an den restlichen Daten (90%) trainiert.



Now, one of most commonly asked question is, "**How to choose right value of k?**"

Always remember, lower value of K is more biased and hence undesirable. On the other hand, higher value of K is less biased, but can suffer from large variability. It is good to know that, smaller value of k always takes us towards validation set approach, whereas higher value of k leads to LOOCV approach. Hence, it is often suggested to use k=10.

Quelle:

<https://www.analyticsvidhya.com/blog/2015/11/improve-model-performance-cross-validation-in-python-r/>

Analytics Vidhya  
Learn Everything About Analytics

SKILLTEST:  
TREE BASED ALGORITHM

HOME BLOG JOBS TRAININGS DISCUSS LEARNING PATHS DATAHACK STORIES

Home > Business Analytics > Improve Your Model Performance using Cross Validation (in Python and R)

## Improve Your Model Performance using Cross Validation (in Python and R)

BUSINESS ANALYTICS | PYTHON | R

SUNIL RAY, NOVEMBER 18, 2015 / 14

# 10. Sampling: Bootstrapping, Standardfehler

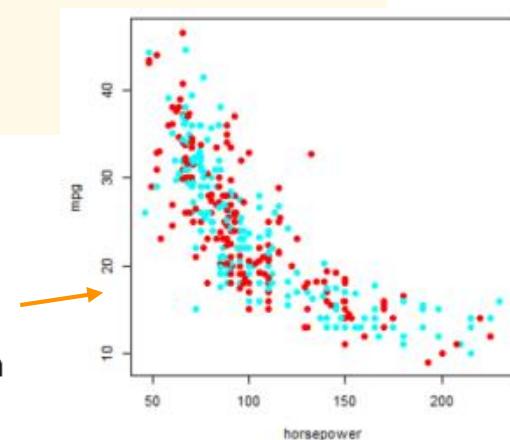
Einen anderen Weg geht das Bootstrap-Verfahren. Hier wird aus der Originaldatei eine große Anzahl von Stichproben gezogen, und zwar „mit Zurücklegen“. Datensätze aus der Originaldatei können also mehrfach, einfach oder gar nicht in einer Stichprobe vorkommen. Dieses nichtparametrische Verfahren kann dazu benutzt werden, den Standardfehler der Koeffizienten eines linearen Modells über die Streuung der gezogenen Stichproben zu ermitteln. Das Ergebnis vergleichen wir mit den klassischen Standardfehlern:

```
boot.fn=function(data,index) return (coef(lm(mpg~horsepower,data=data,subset=index)))
boot(Auto,boot.fn,1000)                                     # Ergebnis:
# Bootstrap Statistics :
#   original    bias      std. error
# t1* 39.9358610 0.0269563085 0.859851825
# t2* -0.1578447 -0.0002906457 0.007402954

# Vergleich mit Standardfehlern der Regressionskoeffizienten eines linearen Modells:
summary(lm(mpg~horsepower,data=Auto))$coef      # Ergebnis:
#                   Estimate Std. Error   t value Pr(>|t|)
# (Intercept) 39.9358610 0.717498656 55.65984 1.220362e-187
# horsepower  -0.1578447 0.006445501 -24.48914 7.031989e-81
```

Was ist nun richtig?

Platt gesagt beruht die Berechnung des klassischen Standardfehlers auf der Annahme, dass es einen rein linearen Zusammenhang gibt. Das trifft hier allerdings nicht zu. Der Bootstrap-Ansatz ist frei von diesen und weiteren Annahmen und sollte daher die zutreffendere Schätzung für den Standardfehler abgeben.



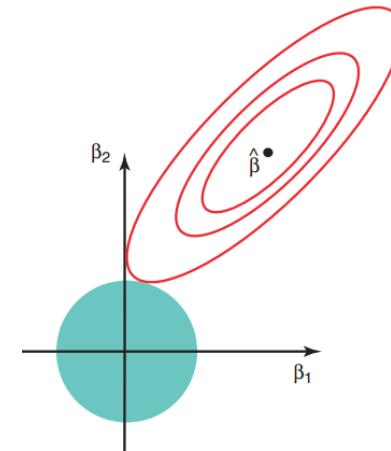
# 11. Regularisierung: Ridge-Regression und LASSO

**Ridge regression** is very similar to least squares, except that the coefficients are estimated by minimizing a slightly different quantity. In particular, the ridge regression coefficient estimates  $\hat{\beta}^R$  are the values that minimize

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^p \beta_j^2, \quad (6.5)$$

where  $\lambda \geq 0$  is a *tuning parameter*, to be determined separately. Equation 6.5 trades off two different criteria. As with least squares, ridge regression seeks coefficient estimates that fit the data well, by making the RSS small. However, the second term,  $\lambda \sum_j \beta_j^2$ , called a *shrinkage penalty*, is

Nachteil: Keine Parameterreduktion, i.d.R. alle  $\beta > 0$ , problematisch bei  $p > n$

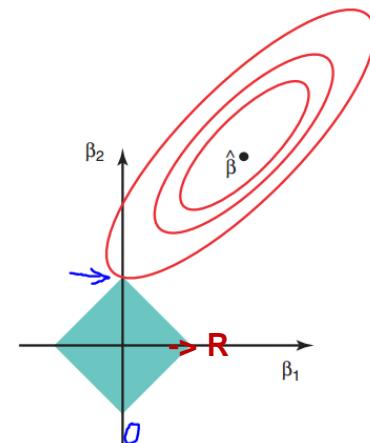


We say that the lasso yields *sparse* models—that is, models that involve only a subset of the variables. As in ridge regression, selecting a good value of  $\lambda$  for the lasso is critical; we defer this discussion to Section 6.2.3, where we use **cross-validation**.

The **lasso** is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients,  $\hat{\beta}_\lambda^L$ , minimize the quantity

$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS} + \lambda \sum_{j=1}^p |\beta_j|. \quad (6.7)$$

Comparing (6.7) to (6.5), we see that the lasso and ridge regression have similar formulations. The only difference is that the  $\beta_j^2$  term in the ridge regression penalty (6.5) has been replaced by  $|\beta_j|$  in the lasso penalty (6.7). In statistical parlance, the lasso uses an  $\ell_1$  (pronounced “ell 1”) penalty



# 11. Regularisierung: Beispiel zu Ridge-R. und LASSO

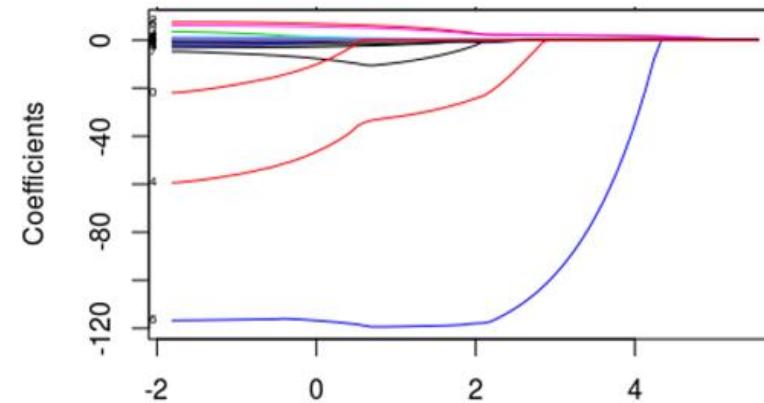
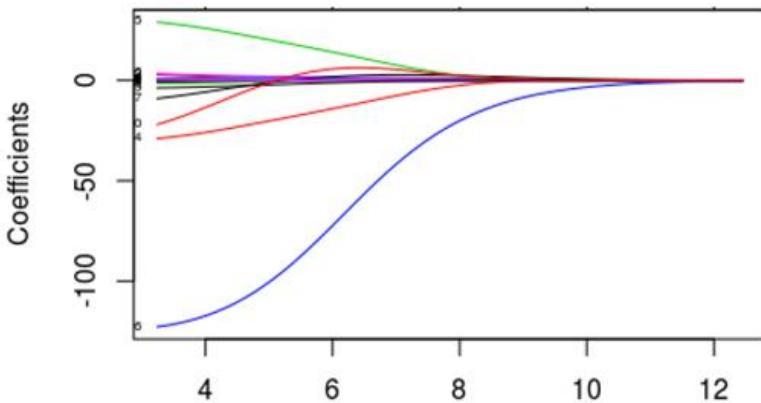
Wir wollen nun auf Basis des Baseball-Datensatzes `Hitters` das Gehalt amerikanischer Baseballspieler vorhersagen und dazu Ridge-Regression und Lasso mit der Funktion `glmnet` anwenden.

```

Hitters=na.omit(Hitters)                      # Datensätze mit fehlenden Werten entfernen
with(Hitters,sum(is.na(salary)))              # Zähle Datensätze ohne Salary. Ergebnis: 0
dim(Hitters)                                  # Ergebnis: Noch 263 Datensätze, 20 Merkmale
x=model.matrix(Salary~.-1,data=Hitters)        # Matrix, ohne Salary. Bildet num. Dummy-Merkmale
y=Hitters$Salary

# Schrumpfen der Koeffizienten:
fit.ridge=glmnet(x,y,alpha=0)                  # Ridge-Regression fitten
plot(fit.ridge,xvar="lambda",label=TRUE)         # Schrumpfen der Koeffizienten plotten
fit.lasso=glmnet(x,y)                          # Lasso fitten. Default: alpha=1
plot(fit.lasso,xvar="lambda",label=TRUE)          # Ergebnis x: log(Lambda)
  
```

Die Koeffizienten ( $\beta_j$ ) für Ridge-Regression (links) und Lasso (rechts) in Abhängigkeit von  $\log(\lambda)$ :



Nächstes:  
 Suche  
 bestes  
 Lambda

# 12. Support Vector Classifier, SV Machine

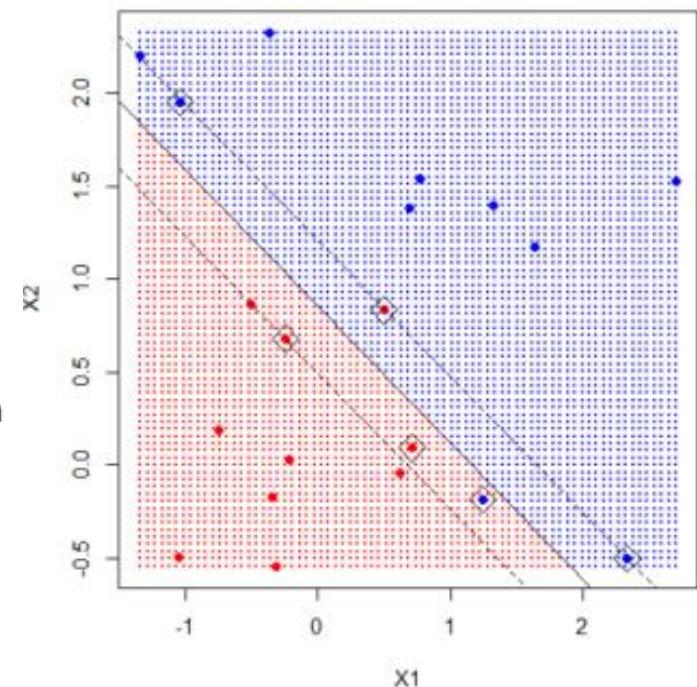
Bei Klassifikationsaufgaben wird für die abhängige Variable eine möglichst trennscharfe Hyperebene im Merkmalsraum gesucht, im zweidimensionalen Fall also eine Trennlinie, im einfachsten Fall eine Gerade. Diese Gerade wird auf Basis der am dichtesten liegenden Datenpunkte in der Grenzregion berechnet. Ein Datenpunkt, der die Berechnung der Trennlinie „unterstützt“, wird als „Support Vector“ bezeichnet. Das Besondere an diesem Verfahren ist, dass weiter von der Grenzlinie entfernte Punkte keinen Einfluss haben.

Die Lage eines einzelnen Datenpunktes kann großen Einfluss auf die Lage der Trennlinie haben. Liegt der Datenpunkt auf der „falschen“ Seite, ist genau genommen gar keine Trennlinie möglich. Damit das Verfahren auch in diesem nicht seltenen Fall angewandt werden kann, wird ein „Budget“  $C$  für Grenzverletzungen akzeptiert. Dieser Kostenparameter wird über Kreuzvalidierung getunt, wir kennen das bereits aus dem letzten Kapitel (was dem einen sein  $\lambda$ , ist dem andern sein  $C$ ).

Eine „Support Vector Machine“ ist die Erweiterung dieses Konzepts um Kernel.

„Support Vector Classifier“

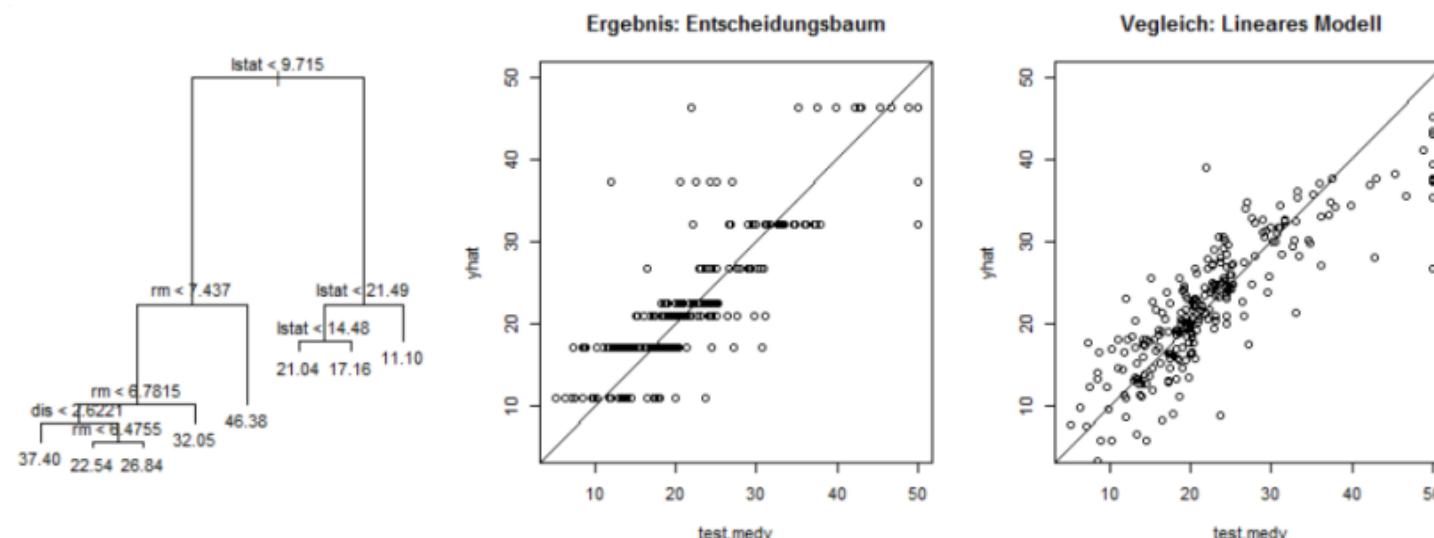
SVs (Rauten), Margins



( um SVMs ist es in den letzten Jahren etwas ruhiger geworden, siehe auch Kap. 19. Weiter ...)

# 13. Entscheidungsbäume, Pruning

Wir starten mit dem Grundprinzip des rekursiven binären Teilens. Bei einem Regressionsproblem wird für jedes Merkmal der Splitpunkt gesucht, der zum größten Rückgang der RSS führt. Der erste Split wird dann mit dem besten Merkmalsplitpunkt durchgeführt. Im Beispiel unten mit den bereits bekannten Hauspreisdaten wird für die gewählte Trainingsstichprobe ein optimaler erster Splitpunkt für  $lstat < 9,715$  ermittelt.



Obwohl es sich um ein Regressionsproblem handelt und für den Hauspreis nur acht verschiedene Werte vorhergesagt werden, ist das Prognoseergebnis nicht schlechter als das vom zweiten linearen Modell aus Kapitel 8. In beiden Fällen wird der Hauspreis auf etwa 5.000\$ genau geschätzt (RMSE).

Unser erster Entscheidungsbau ist „links unten“ etwas ins Overfitting gelaufen. Overfitting ist für Entscheidungsbäume recht typisch, sie neigen zum „verwachsen“ und sollten zurückgeschnitten werden. Dieses „Pruning“ kann durch die Einbeziehung der Testdaten (die dann nicht mehr neutral sind) oder durch Kreuzvalidierung durchgeführt werden. Der Tuningansatz gleicht dem aus Kapitel 11.

Entscheidungsbäume gelten als „weak learner“.

Verbesserungen:  
Bagging +  
Random Forest ++  
Boosting +++

# 13. Bagging, Random Forest

Durch Bootstrapping (Kap. 10) kann Varianzreduktion erreicht werden. Diesen Umstand verwendet die Bagging-Methode, bei der B verschiedene Bootstrap-Trainingsstichproben gezogen werden und dazu jeweils ein Entscheidungsbaum mit allen p Merkmalen gerechnet wird. Diese B Entscheidungsbäume führen zu B Prognoseergebnissen, aus denen bei Regressionsproblemen ein Durchschnittswert berechnet und bei Klassifikationsproblemen eine Mehrheitsentscheidung gefällt wird.

( Zwischen-  
stufe)

Die Methode „Random Forest“ geht noch einen Schritt weiter und dekorreliert die Entscheidungsbäume dadurch, dass jeder Split auf Basis einer neuen zufälligen Auswahl von m Prädiktormerkmalen durchgeführt wird. Gemäß der Faustformel  $m \approx \sqrt{p}$  werden für nicht allzu schmale Datensätze die Mehrheit der Merkmale, und damit auch häufig genug die stärksten Merkmale von der Split-Berechnung ausgeschlossen. Dadurch wachsen die Bäume viel unterschiedlicher und sind unkorrelierter.

Die Funktion `randomForest` rechnet dazu 500 (Default-Wert) teilweise sehr unterschiedliche Bäume. Deren Mittelung führt zwar zu einer guten Prognose, verhindert aber eine einfache Darstellung als Entscheidungsbaum. Das ist in diesem Vergleich der Hauptnachteil von Bagging und Random Forest. Diesem Nachteil haben aber auch andere leistungsfähige Verfahren.

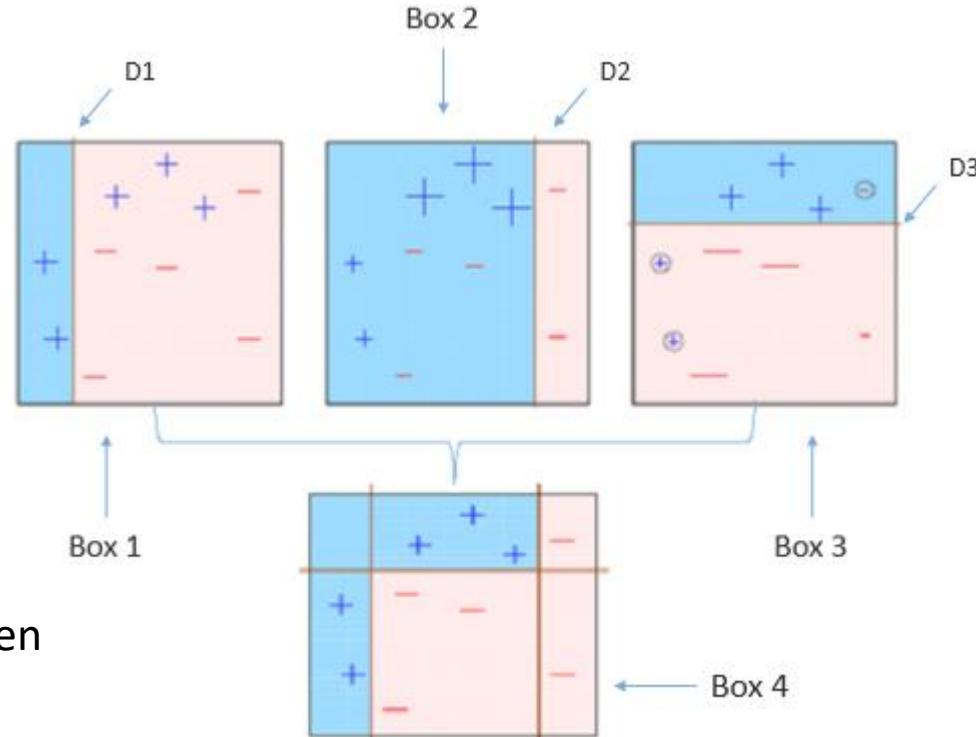
Bagging und die Weiterentwicklung Random Forest setzen voll auf den Zufall und die Unabhängigkeit einer großen Anzahl von Stichproben und erreichen damit das Ziel der Varianzreduktion.

# 14. Gradient Boosting Machines

Boosting ist ein allgemeiner Ansatz, der für viele Verfahren angewendet werden kann. Generell wird hier aus einer Vielzahl von sogenannten schwachen Lernern ein starker Lerner erzeugt. Boosting von Regressionsbäumen verwendet wie Bagging eine hohe Anzahl B an Entscheidungsbäumen. Diese sind beim Boosting jedoch voneinander abhängig und sehr klein (d Splits). Jeder Baum hängt von den zuvor erzeugten Bäumen ab und versucht die Prognosegenauigkeit zu erhöhen.

- Die Gewichte werden durch ein Gradientenverfahren ermittelt.
- Es wird eine Kostenfunktion mit Schrumpfungsparameter  $\lambda$  verwendet.
- Tree-Boosting hat drei Tuning-Parameter ( $B, d, \lambda$ ).

Gradient Boosting Machines können Klassifikation, Regression und Ranking durchführen.



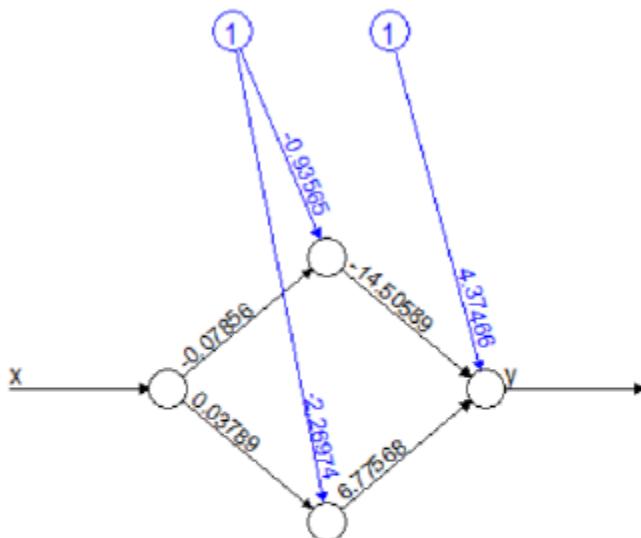
Visualisierungen und Erläuterungen siehe:  
<https://www.analyticsvidhya.com/blog/2015/11/quick-introduction-boosting-algorithms-machine-learning/>

„Champion“:  
**xgboost** (sehr schnell)

# 15. Künstliche Neuronale Netze (NN): Regression

Beispiel: Die Quadratwurzel [0 bis 100] durch eine kleines NN abbilden

Vorgehensweise: Zufallszahlen simulieren, Quadratwurzel berechnen, beides in Trainingsdatei schreiben, NN trainieren und testen. Ergebnis:



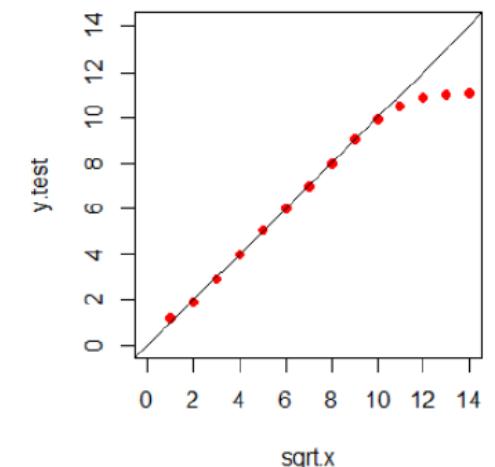
Error: 0.027624 Steps: 3675

Wir wollen nun für die Zahl  $x=100$  die Prognose (Quadratwurzel) dieses Netzes nachrechnen:

$$y = 4,375 + -14,506 / (1 + \exp(-(100^2 * -0,0786 + -0,94))) + 6,776 / (1 + \exp(-(100^2 * 0,0379 + 2,27))) = 9,93$$

Für sehr große x:

$$y = 4,375 + -14,506 / (1 + \exp(-(100^2 * -0,0786 + -0,94))) + 6,776 / (1 + \exp(-(100^2 * 0,0379 + 2,27)))$$



Neuronale Netze haben Schwierigkeiten mit Ereignissen außerhalb ihres Trainingsbereiches.  
Maßnahmen: Mehr Trainingsdaten, größeres NN

# 16. Künstliche Neuronale Netze (NN): Klassifikation

Der Datensatz `iris` ist in der Bibliothek `MASS` enthalten und umfasst für 150 verschiedene Pflanzen je vier Messwerte zu Längen und Breiten der Blätter sowie die Angabe, um welche der drei Arten setosa, virginica oder versicolor es sich handelt.

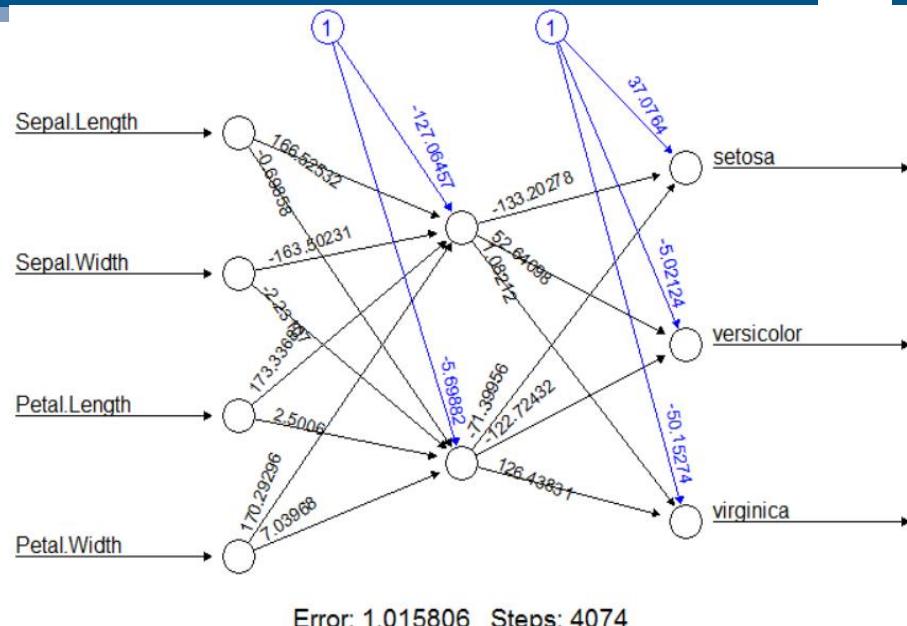
Wir verwenden für das NN die gleiche Funktion und auch die gleiche hidden layer mit nur zwei Neuronen wie beim Wurzel-Beispiel im vorangegangenen Kapitel. Allerdings ergibt sich durch die vier Input-Variablen und drei Output-Variablen ein etwas komplexeres Neuronales Netz:

In den weiteren Schritten wird das neuronale Netz auf die Testdaten übertragen, die Klassifikationsentscheidung auf Basis des höchsten Prognosewertes getroffen und der Testfehler berechnet:

```

pred = compute(net, sc_test[,1:4])
y_pred = apply(pred$net.result, 1, which.max) # Höchsten Wert auswählen,
y_true = apply(csc_test[,5:7], 1, which.max) # auf Testdaten anwenden und
confusion_matrix = table(y_true, y_pred)      # Genauigkeit berechnen:
> head(pred$net.result)
     [,1]          [,2]          [,3]
1  1 0.005741347327 0.000000000000000000000000000000001898786506
5  1 0.005795409208 0.000000000000000000000000000000001880435072
6  1 0.005495686772 0.000000000000000000000000000000001986795975
11 1 0.005881211319 0.000000000000000000000000000000001852012400
12 1 0.005541638201 0.000000000000000000000000000000001969731200
17 1 0.005650910538 0.000000000000000000000000000000001930282601
> print(confusion_matrix)
      y_pred
y_true 1 2 3
  1 18 0 0
  2 0 14 0
  3 0 2 16
> print(paste("Accuracy:", sum(diag(confusion_matrix))/sum(confusion_matrix)))
[1] "Accuracy: 0.96"

```



Für große nichtlineare Datensätze und entsprechend große, tiefe Modelle brauchen wir eine sehr schnelle, parallel arbeitende und RAM-schonende Funktion, siehe Kapitel 19.

# 17. „Big Business“: Personalisierte Empfehlungen

Wir wollen nun Filmempfehlungen auf Basis der Filmdatenbank „MovieLens“ erstellen. Das R-Package `recommenderlab` stellt mit der Datei `MovieLens` einen kleinen Auszug aus der Filmdatenbank bereit und enthält die Funktion `similarity`, mit der wir den Cosinus-Winkel zwischen Bewertungsvektoren berechnen und damit Filme mit einem ähnlichen Publikumsgeschmack erkennen:

```
##R 17.1
library(recommenderlab)          # Die Filmbewertungen der Nutzer ansehen:
data(MovieLense)    # Daten bereitstellen. Die Datei enthält 99.392 Filmbewertungen
print(MovieLense) # Matrixgröße/-art anzeigen: n=943 User (Zeilen), p=1664 Filme (Spalten)
print(table(as.vector(as(MovieLense, "matrix")))) # "Sterneverteilung" (1 bis 5) anzeigen
summary(colCounts(MovieLense))      # Filmbewertungen je Nutzer: Median 59.73
summary(rowCounts(MovieLense))      # Bewertungen je Film: Median 105.4
print(colCounts(MovieLense[,1:5])) # Anzahl der Bewertungen für die ersten 5 Filme
print(colMeans(MovieLense)[1:5])   # Durchschnittsrating (Spaltendurchschnitt) -"-"
# Kollaboratives Filtern: Cosinus-Ähnlichkeiten zu Film 1 (Toy Story) berechnen
cos.filml=similarity(MovieLense[,1],MovieLense[,-1],method="cosine",which="items")
colnames(cos.filml)[which(cos.filml>0.69)] # Filme mit der höchsten Ähnlichkeit. Ergebnis:
##t
[1] "Star wars (1977)"           "Return of the Jedi (1983)"
```



MovieLens is a web site that helps people find movies to watch. It has hundreds of thousands of registered users. We conduct online field experiments in MovieLens in the areas of automated content recommendation, recommendation interfaces, tagging-based recommenders and interfaces, member-maintained databases, and intelligent user interface design.

Als nächstes wollen wir vorhersagen, ob sich eine Person für einen bestimmten Film interessiert und den Erfolg dieser Vorhersage überprüfen. Dazu verwenden wir die Singulärwertzerlegung (SVD), die beim Netflix-Preis als erfolgreiche Methode für Empfehlungen bekannt wurde, und zerlegen unsere  $n \times p$ -Filmbewertungsmatrix A folgendermaßen:  $A = UDV^T$ . Die Komponenten der User-Matrix U werden dann in einem Random-Forest-Modell (Kap. 13) als erklärende Merkmale für das Ereignis „Film angesehen“ verwendet.

-> R

## 18. MNIST-Daten entziffern mit Random Forest

Bei den bisherigen Anwendungsbeispielen handelte es sich um recht kleine Datensätze. Die Anzahl der erforderlichen Berechnungen war entsprechend gering und daher die Laufzeit der Fits und Trainings unerheblich.

Das ändert sich nun, wir wenden uns der Bilderkennung zu. Ein viel getesteter Referenzdatensatz sind die 70.000 Bilder von handschriftlichen, sauber getrennten Ziffern der MNIST-Database (Mixed National Institute of Standards and Technology). Diese Bilddaten werden von kaggle.com als einfach einzulesende Trainings- und Testdaten mit einer Auflösung von 28x28 Bildpunkten bereitgestellt:

```
library(readr)                                # Nötig für read_csv
train = read_csv("D:/downloads/mnist_train.csv") # 42000 Images mit 784 Bildpunkten + Label
test = read_csv("D:/downloads/mnist_test.csv")   # 28000 Images mit 784 Bildpunkten
par(mfrow = c(1,20)) # in RStudio Plot-Fenster entsprechend anpassen. Nun Pixel aufbereiten:
for (i in 1:20){ y = as.matrix(train[i, 2:785]); dim(y) = c(28, 28)
  image( y[,nrow(y):1], axes = FALSE, col = gray(255:0/255) )
  text(0.2, 0, train[i,1], cex = 4, col = 2, pos = 3 ) # und Ziffernwert rot überblenden:
```



Fazit nach zwei Versuchen mit der Funktion `randomForest`:

- Lange Laufzeiten und schlechte Ressourcennutzung (keine Parallelisierung)
- Fehlklassifikationsrate (upload kaggle.com) mit 4-5% zu hoch

Warum dieses Beispiel „Handschrifterkennung“?

- Eine der ersten ML-Anwendungen
- Referenzdatensatz für Verfahrensvergleiche
- Einfaches Bilderkennungsbeispiel, hohe Erfolgsquote

# 18. Entziffern mit XGBOOST, Laufzeitmessung

Dieses Problem von klassischen R-Funktionen, die nur auf einem einzigen PC-Rechenkern ausgeführt werden und dabei auch noch reichlich Arbeitsspeicher verlangen, hat auch die R-Funktion `gbm`. XGBOOST hingegen ist hinter den R-Kulissen in C++ programmiert und nutzt die OpenMP-Schnittstelle für die Parallelverarbeitung der Prozessoren. Daher führen wir das folgende Boosting mit XGBOOST durch und lesen nun die MNIST-Daten mit der Standardfunktion `read.csv` ein.

```
library(xgboost)
train=read.csv("D:/downloads/mnist_train.csv") # 42000 Images, 784 Bildpunkte + Label
test =read.csv("D:/downloads/mnist_test.csv") # 28000 Images, 784 Bildpunkte
train.images=train[,-1]           # Bilddaten und Ziffern in verschiedene Objekte schreiben
train.digits=train[,1]

# Boosting Trees (default: booster="gbtree", eta=0.3, max_depth=6)
set.seed(1)
tic <- proc.time()               # Startzeit merken
mod=xgboost(data=xgb.DMatrix(model.matrix(~.,data=train.images),label=train.digits),
            num_class=10, nrounds=100, early.stop.round=3,
            params=list(objective="multi:softmax",eval_metric="mlogloss"))
print(proc.time() - tic)          # Training-Laufzeit xgboost berechnen
pred=predict(mod,newdata=xgb.DMatrix(model.matrix(~.,data=test)))
write.csv(data.frame(ImageId=1:nrow(test),Label=pred),"mnist_xgb.csv",quote=F,row.names=F)
```

Ergebnis: Kürzere Laufzeit und Fehlklassifikationsrate auf 2,7% gesenkt.

# 19. Handschrifterkennung mit MXNet: Deep Learning

MXNet.io bezeichnet sich als flexible und effiziente Bibliothek für Deep Learning. MXNet wurde Ende 2015 vorgestellt und hat teilweise die gleichen Autoren wie XGBOOST, siehe <https://arxiv.org/abs/1512.01274>. MXNet unterstützt aktuell sieben Programmier- und Skriptsprachen, darunter auch R. Es läuft auf Windows, Linux, OS X und auf den großen Cloud-Plattformen. Es rechnet sowohl auf CPUs als auch auf GPUs (Graphikkarten) und ist sehr schnell. Zur Wahl stehen Neuronale Netze von Typ „feed-forward“, „recurrent“ und „convolutional“.

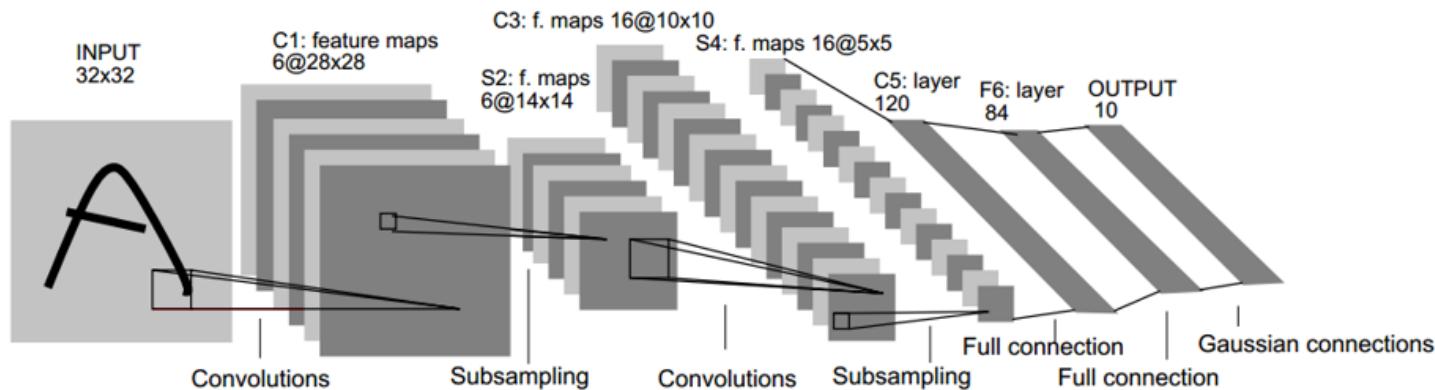
In unserer ersten Anwendung berechnen wir ein zweilagiges Multi Layer Perceptron mit 128 Neuronen in der ersten und 64 Neuronen in der zweiten hidden layer. Für beide Schichten verwenden wir die Aktivierungsfunktion ReLU, die zu schneller Konvergenz und zu einem schlanken Modell führt. Mit der Softmax-Funktion werden die 10 Outputs gleich auf 1 normiert.

-> R

Das nicht mehr ganz kleine Modell ist in Rekordzeit nach wenigen Sekunden trainiert. Ist es auch gut? Zur Beantwortung dieser Frage laden wir wieder die Score-Datei auf kaggle.com hoch. Ja, das Modell ist unser bisher bestes, wir haben nur noch 2% Fehlklassifikationen und nähern uns den Top 25%.

# 19. Convolutional Neural Net (CNN): „LeNet-5“

Unsere MNIST-Klassifikationsbemühungen schließen wir mit einem sogenannten Convolutional Neural Network (CNN) ab. CNNs sind bekannt für ihre gute Performance bei Bilderkennungsaufgaben. Sie nutzen den Umstand, dass räumlich dicht beieinander liegende Punkte in einem Zusammenhang stehen und hoch korreliert sind. Das geschieht durch lokale Verbindungsstrukturen, gemeinsame Gewichte und ein räumliches Subsampling, siehe die Animation von Yann LeCun <http://yann.lecun.com/exdb/lenet/>. Im Artikel von LeCun et al. wird das erfolgreiche LeNet-5 mit folgenden Schichten vorgestellt, siehe auch <http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf>:



Das Modell läuft trotz der theoretischen Vorteile gegenüber dem voll verknüpften Modell deutlich langsamer durch und kann auch auf einen Power-PC bei voller Auslastung mehrere Minuten Rechenzeit benötigen. Aber es lohnt sich, wir sind im public leaderboard auf kaggle nun in den Top 15%, die Fehlklassifikationsrate schrumpft auf 0,9%!

-> R

# 20. GPU-Computing mit MXNet: Graphikkarten, Software

Aktuelle „Gaming“-Graphikkarten haben inzwischen mehrere Tausend Prozessorkerne. Sie eignen sich daher ganz hervorragend für die Parallelverarbeitung unserer Berechnungen. MXNet verwendet für die GPU-Nutzung die „CUDA deep learning library“ cuDNN des Graphikartenherstellers NVidia, die allerdings aus lizenzrechtlichen Gründen nicht von MXNet ins R-Package eingebunden werden kann. Daher müssen wir nun einen weiten, aber auch lehrreichen Weg gehen. Wir benötigen:

- Microsoft Visual Studio 2013 (Community Edition, kostenlos),
- das NVidia CUDA Toolkit 8.0 (1,2 GB),
- das aktuelle MXNet package (R-package/ und nocudnn/) und
- die Bibliothek cuDNN samt Registrierung als Entwickler.

Nun folgen wir der Anleitung [http://mxnet.io/get\\_started/setup.html#installing-mxnet-on-a-gpu](http://mxnet.io/get_started/setup.html#installing-mxnet-on-a-gpu) und führen die zahlreichen erforderlichen Installationsschritte mit etwas Glück und Erfahrung korrekt durch.

Wenn das alles geklappt hat wird es ganz einfach. Im R-Skript muss nur das Device angepasst werden.

Am Prognoseergebnis ändert sich dadurch nichts. Die Modellbildung sollte nun rund 10 mal so schnell erfolgt sein. Je nach Leistungsfähigkeitsverhältnis zwischen CPU und GPU sind auch andere Größenordnungen möglich. CPU-Processing kann sogar schneller sein, wie Kutkina und Feuerriegel in <http://www.is.uni-freiburg.de/resources/r-oeffentlicher-zugriff/deep-learning-in-r/> zeigen, siehe den MNIST-Vergleich in der dortigen Tabelle 3.

# 21. Big Data mit H2O: Java-Umgebung, R vs. FLOW

H2O beschreibt sich als schnelle und skalierbare „open source machine learning platform“, siehe <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html>. Im Vergleich zu MXNet verfügt H2O über eine große Bandbreite von ML-Methoden und kann auf Rechnerclustern ausgeführt werden, nutzt jedoch keine GPUs. H2O.ai tritt selbstbewußt auf und wendet sich mit seinen Dienstleistungen an Unternehmen, die in Open Source Analytics einen strategischen Vorteil sehen. Darunter auch Versicherungen, siehe Referenzvideos unter <http://www.h2o.ai/customers/>. H2O benötigt das Java Runtime Environment und kann in Python, R und der eigenen Oberfläche FLOW verwendet werden.

Folgende Schritte sind erforderlich, um in RStudio das Package h2o nutzen zu können:

- Java Runtime Environment (64 Bit, Version 8) von <https://www.java.com/de/download/> herunterladen und installieren.
- Das aktuelle h2o-zipfile von <http://www.h2o.ai/download/> herunterladen, entzippen und im Terminal (Windows: „CMD“ aufrufen, mit „CD“ ins richtige Unterverzeichnis wechseln) H2O mit folgender Anweisung starten:

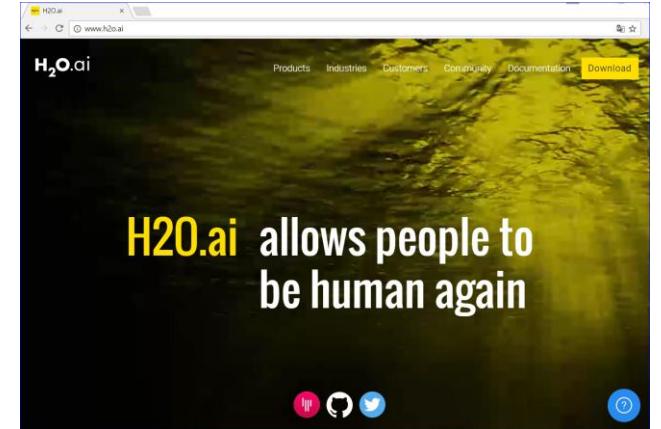
`java -jar h2o.jar`

Dieses Fenster bleibt bis zum Schluss geöffnet.

- In RStudio folgende Installationen vornehmen:

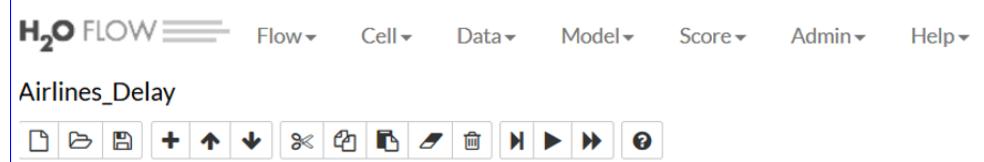
```
# Packages, die von H2O benötigt werden, installieren
if (!("methods" %in% rownames(installed.packages()))){install.packages("methods") }
if (!("statmod" %in% rownames(installed.packages()))){install.packages("statmod") }
if (!("stats" %in% rownames(installed.packages()))){install.packages("stats") }
if (!("graphics" %in% rownames(installed.packages()))){install.packages("graphics") }
if (!("RCurl" %in% rownames(installed.packages()))){install.packages("RCurl") }
if (!("jsonlite" %in% rownames(installed.packages()))){install.packages("jsonlite") }
if (!("tools" %in% rownames(installed.packages()))){install.packages("tools") }
if (!("utils" %in% rownames(installed.packages()))){install.packages("utils") }

# Das H2O-Package herunterladen, installieren und initialisieren (Stand 15.10.2016)
install.packages("h2o",type="source",repos=c("http://h2o-release.s3.amazonaws.com/h2o/r"))
library(h2o)
localH2O = h2o.init(ntreads=-1) # -1: alle Kerne nutzen
demo(h2o.kmeans) # Demo starten, H2O kennen lernen
```



Die Analyse wird über Java an R vorbei ausgeführt. Die Daten liegen nicht in R vor, es wird hier lediglich R-Syntax verwendet und damit R-Nutzern die Möglichkeit geboten, in einer vertrauten Weise die Vorteile von H2O zu nutzen. Die entstandenen Ergebnisdaten können bei Bedarf durch h2o-Funktionen ins „normale“ R überführt und dort weiter verarbeitet werden.

Im Airline-Beispiel wird R eigentlich nicht benötigt. Als Alternative zu R kann nach Schritt b) die Analyse-Oberfläche H2O FLOW über den Browser <http://localhost:54321/flow/index.html> gestartet und die Analyse über die Click-Oberfläche durchgeführt werden:



H2O kann auf Big-Data-Systemen wie Apache Spark und Hadoop sowie in Cloud-Computing-Umgebungen wie Amazon EC2 ausgeführt werden.

# 22. Abschluss mit und ohne R: Ensembles, Faces

## Ensembles

Unter <http://mlwave.com/kaggle-ensembling-guide/> gibt es eine sehr einfache und ansprechende Beschreibung der Ensembling-Ansätze für Wettbewerbe auf kaggle.com. Im dort beschriebenen kleinen Umfang können Ensembles spürbare Verbesserungen erzielen. Inzwischen findet bei den Wettbewerben im Kampf um winzige Verbesserungen häufig eine Materialschlacht mit umfangreichen Ensembles statt, deren praktische Umsetzbarkeit fraglich ist.

## Gesichtserkennung

Im Tutorial <https://www.kaggle.com/c/facial-keypoints-detection> wird gezeigt, wie man mit R in Gesichtern Schlüsselpunkte erkennen kann. Der Gesamtprozess der automatischen Gesichtserkennung und Identifizierung besteht aus den Schritten Gesichter finden, Gesichter ausrichten, Gesichtsmerkmale ermitteln und abgleichen und schließlich den Namen dazu finden, siehe den Überblick in Part 4 von <https://medium.com/@ageitgey>. Im Analytischen Kern wird dabei ein CNN mit mehreren hundert Millionen Gesichterbildern trainiert.

# 22. Abschluss mit und ohne R: GitHub und Linux

## Github

GitHub.com stellt Softwareentwicklungsprojekte im Web auf seinen Servern bereit. Die Nutzer stehen dabei mit ihren Quellcodes im Mittelpunkt und bilden ein soziales Netz. Für eine lockere Kurzbeschreibung siehe <http://t3n.de/news/eigentlich-github-472886/>. Eine Mitmachanleitung befindet sich hier: <https://guides.github.com/activities/hello-world/>. GitHub ist für öffentliche Projekte kostenlos. Zahlreiche Entwicklerinnen und Entwickler nutzen GitHub für ML-Projekte.

## Linux

Einige ML-Bibliotheken und Programme wie H2O funktionieren nur oder am besten mit dem Betriebssystem Linux. Für unsere Zwecke und übliche PCs ist die Linux-Distribution Ubuntu (64 bit) gut geeignet. Falls Sie über einen alten, nicht mehr benötigten PC oder Laptop verfügen, können Sie das gefahrlos ausprobieren und Linux parallel zu Windows oder als alleiniges System installieren, siehe

[http://www.pcwelt.de/ratgeber/Linux\\_neben\\_Windows\\_installieren\\_-\\_so\\_geht\\_s-Multiboot-8634306.html](http://www.pcwelt.de/ratgeber/Linux_neben_Windows_installieren_-_so_geht_s-Multiboot-8634306.html).

# 22. Abschluss mit und ohne R: Python und Tensorflow

## Python

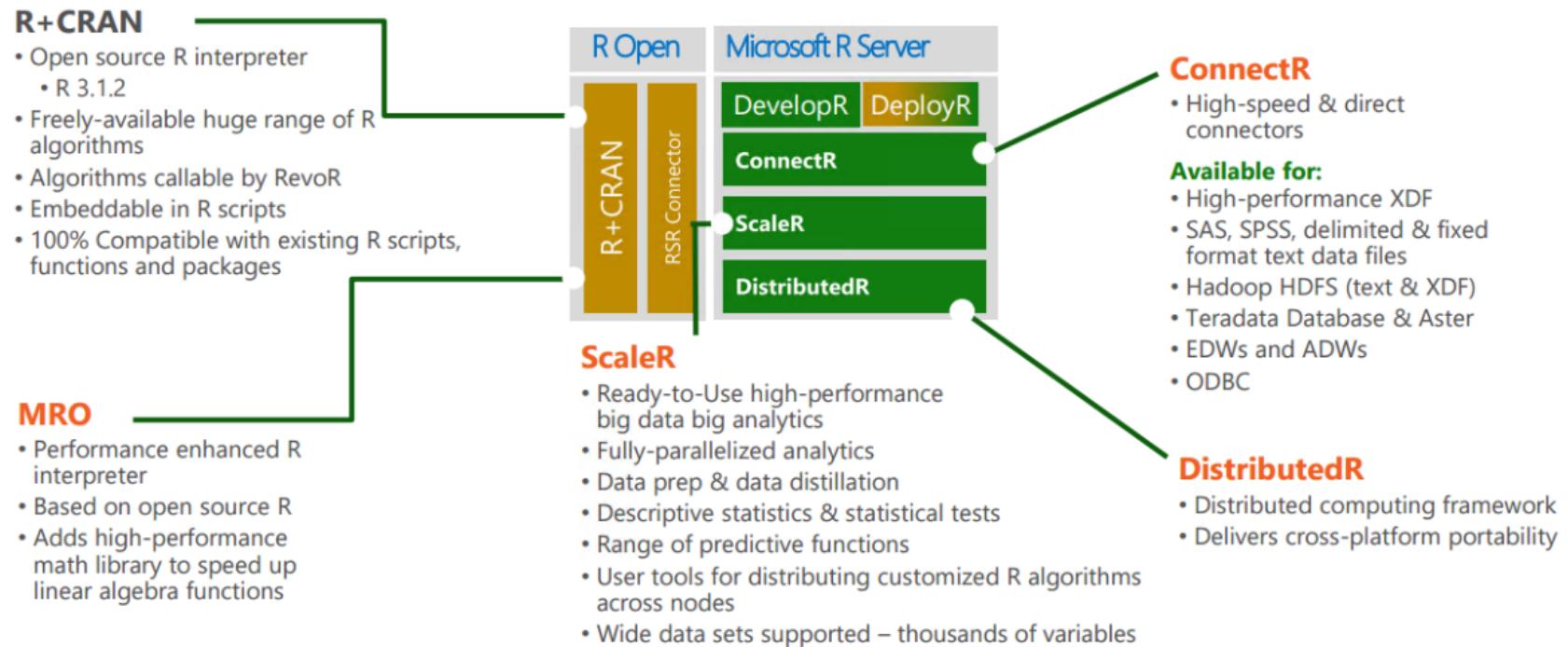
Python ist eine interpretierte höhere Programmiersprache und hat eine sehr aktive Entwicklergemeinde, die mächtige ML-Bibliotheken erstellt hat und weiter entwickelt. Wer ernsthaft Machine Learning betreiben möchte, kommt an Python als Ergänzung zu R kaum vorbei. Die Einstiegshürde für Python ist höher als für R. Die parallel existierenden Versionswelten 2.7x und 3.x können Anfängern den Start erschweren.

## Tensorflow, Übersetzungen

Die Open-Source ML-Bibliothek Tensorflow wurde in November 2015 vom Google Brain Team herausgegeben und in kurzer Zeit zum am meisten gefolgten und kopierten ML-Projekt auf GitHub. Zum Kennenlernen gibt es gleich zwei Tutorials auf Basis der MNIST-Daten und am Ende eine ziemlich coole „Tinker“-Anwendung, siehe [https://www.tensorflow.org/versions/r0.11/get\\_started/index.html](https://www.tensorflow.org/versions/r0.11/get_started/index.html). Im Zusammenhang mit Tensorflow berichtet Google von großen Fortschritten bei der maschinellen Übersetzung von Sprache, siehe <https://research.googleblog.com/2016/09/a-neural-network-for-machine.html>. Bei der Modellbildung werden rekurrente neurale Netze inzwischen mit ganzen Sätzen trainiert.

# 22. Abschluss mit und ohne R: Microsoft R Server / R Open

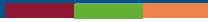
## The Microsoft R Server Platform



"MRS solves two problems associated with using R: capacity (handling the size of datasets and models) and speed."

[http://htmlpreview.github.io/?https://github.com/lixzhang/R-MRO-MRS/blob/master/Introduction\\_to\\_MRO\\_and\\_MRS.html](http://htmlpreview.github.io/?https://github.com/lixzhang/R-MRO-MRS/blob/master/Introduction_to_MRO_and_MRS.html)

# Vielen Dank für Ihr Interesse !



## Fragen ?

# 10a. Messmethodik Einsparungen (1)

Effekt der Teilnahme an einer Maßnahme auf eine Ergebnisvariable:

$$\Delta_i = Y_i^1 - Y_i^0 \quad \forall i \in I \quad (1)$$

mit

$\Delta_i$ : Veränderung des Werts der Ergebnisvariablen für den  $i$ -ten Teilnehmer der Maßnahme,

$Y_i^1$ : Wert der Ergebnisvariablen für den  $i$ -ten Teilnehmer der Maßnahme,

$Y_i^0$ : Wert der Ergebnisvariablen für den  $i$ -ten Teilnehmer, wenn dieser nicht an der Maßnahme teilgenommen hätte,

$I$ : Indexmenge der Teilnehmer an der Maßnahme.

Das Problem: Das Ergebnis der Nichtteilnahme der Teilnehmer ( $i$ ) ist nicht beobachtbar.  
 Gleiches gilt für die Teilnahme von Nicht-Teilnehmern ( $j$ ).

Quelle: Gensler et al. "Einsatzmöglichkeiten der Matching Methode zur Berücksichtigung von Selbstselektion", JfB (2005) 55: 37-62

Weitere Literatur:

- Rosenbaum P, Rubin D (1983) The central role of the propensity score in observational studies for causal effects. *Biometrika* 70:41–55
- Rosenbaum P, Rubin D (1985) Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *Am Stat* 39:33–38
- Christian Pfeifer (2007), "Homogene und heterogene Teilnahmeeffekte des Hamburger Kombilohnmodells. Ein Verfahrensvergleich von Propensity Score Matching und OLS-Regression "
- Felder S, Werblow A „Anreizwirkungen wählbarer Selbstbehalte; Das Selbstbehaltmodell der Techniker Krankenkasse“, Nomos 2005
- DAV-Fachgrundsatz: [https://aktuar.de/unsere-themen/fachgrundsaetze-oeffentlich/2013-03-01-Gutachten\\_zu\\_Wahltarifen\\_final.pdf](https://aktuar.de/unsere-themen/fachgrundsaetze-oeffentlich/2013-03-01-Gutachten_zu_Wahltarifen_final.pdf)

Für eine Verfahrenseinordnung siehe auch den Beitrag von Oliver Kuss et al. im Deutschen Ärzteblatt vom 05.09.2016

<http://www.aerzteblatt.de/archiv/181706>

# 10a. Messmethodik Einsparungen (2)

Die Umgebung:  $\Delta = E[Y_i^1] - E[Y_j^0]$  (2)

mit

$\Delta$ : durchschnittliche Veränderung des Werts der Ergebnisvariablen,

$E[Y_i^1]$ : durchschnittlicher Wert der Ergebnisvariablen für die Teilnehmer an der Maßnahme,

$E[Y_j^0]$ : durchschnittlicher Wert der Ergebnisvariablen für die Nicht-Teilnehmer an der Maßnahme.

Der dargestellte Vergleich der Gruppenmittelwerte basiert jedoch auf der Annahme, dass die Zuordnung der Probanden auf die beiden Gruppen zufällig erfolgt und

Quelle: Gensler et al.

Matching:

- Bildung von Zwillingspaaren, die sich nur beim Teilnahmestatus unterscheiden.
- Je mehr Störvariablen berücksichtigt werden, desto besser kann der Selbstselektionseffekt evaluiert werden.

Dimensionalitätsproblem:

- Je mehr Störvariablen berücksichtigt werden, desto wahrscheinlicher wird kein gleichartiger Matching-Partner gefunden.

Lösungsmöglichkeit: Propensity-Score (Teilnahmewahrscheinlichkeit)

- Der Propensity-Score wird üblicherweise mittels Probit- oder Logit-Modellen geschätzt, bei denen die abhängige Variable die getroffene Teilnahmeentscheidung darstellt.

# 10a. Messmethodik Einsparungen (3)

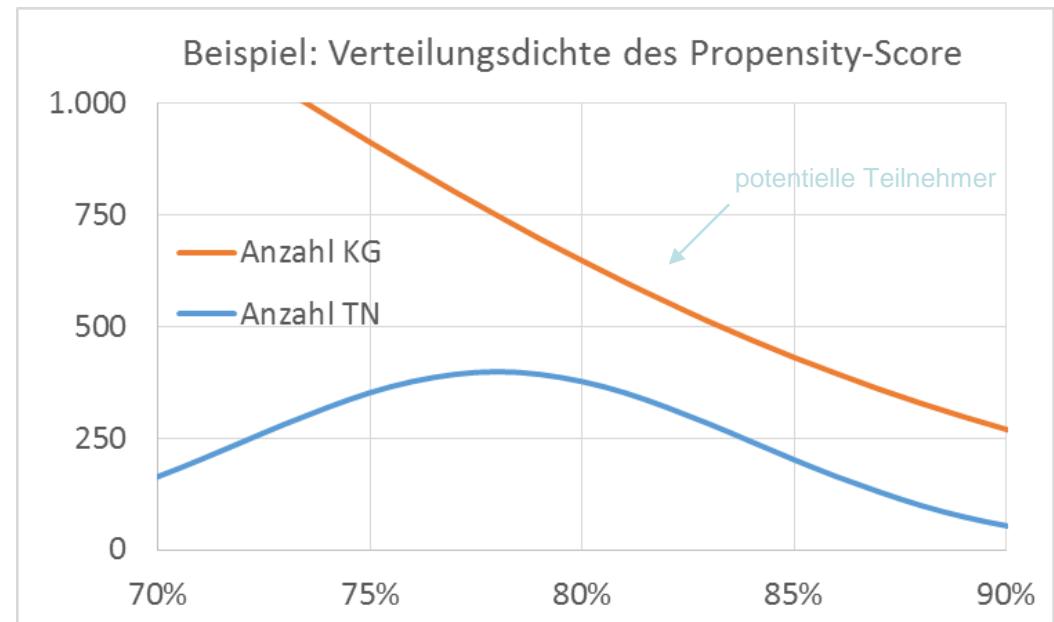
Matching-Methode „Propensity Score“:

Berechnungsbeispiel mit SAS: Schrittweise logistische Regression mit Gewichtung. Variablenatz ähnlich zur [Evaluation Felder/Werblow](#).

```
proc logistic data=Testdaten;
model Gruppe (event='T')=Alter Alter_m25h2 Anzahl_Kinder Einkommen
  Verheiratet1 Geschl_Weibl1 Freiw_Vers1 Selbstst1 Zugangsjahr
  Leistungen_VJ Leistungen_VVJ /selection=stepwise;
output=Testdaten_Scores pred=TNWahrsch;
weight=Stichprobengewicht;
```

**Alternative mit R: siehe Kap.9**

| Beispielhaft      | Regressionsergebnis: |          | Scoring: |         |
|-------------------|----------------------|----------|----------|---------|
|                   | Parameter            | Estimate | Bspl. 1  | Bspl. 2 |
| Intercept         | -2,200               |          | 1        | 1       |
| Alter             | 0,120                |          | 40       | 40      |
| Alter_m25h2       | -0,005               |          | 225      | 225     |
| Anzahl_Kinder     | -0,500               |          | 0        | 2       |
| Verheiratet1      | -0,900               |          | 0        | 1       |
| Geschl_Weibl1     | 0,400                |          | 1        | 0       |
| Freiw_Vers1       | 1,500                |          | 1        | 0       |
| Leistungen_VJ     | -0,001               |          | 830      | 830     |
| Leistungen_VVJ    | -0,001               |          | 770      | 770     |
| Propensity-Score: |                      | 85,5%    | 11,7%    |         |



# 10a. Messmethodik Einsparungen (4)

## Matching-Algorithmus: z.B. SAS-Macro %PSMatching

SAS Global Forum 2007

Statistics and Data Analysis

Paper 185-2007

### Local and Global Optimal Propensity Score Matching

Marcelo Coca-Perraillon

Health Care Policy Department, Harvard Medical School, Boston, MA

#### ABSTRACT

Propensity score-matching methods are often used to control for bias in observational studies when randomization is not possible. This paper describes how to match samples using both local and global optimal matching algorithms. The paper includes macros to perform the nearest available neighbor, caliper, and radius matching methods with or without replacement and matching treated observations to one or many controls. The similarity between observations is evaluated using both the absolute value and the Mahalanobis distance that includes the propensity score along with other covariates. This paper also explains how to find a global optimal match with a variable number of controls using network flows. SAS® 9.1, SAS/STAT®, and SAS/OR® are required.

#### APPENDIX A. %PSMatching macro

```
%macro PSMatching(datatreatment=, datacontrol=, method=, numberofcontrols=, caliper=,
replacement=);
/* Create copies of the treated units if N > 1 */;
data _Treatment0(drop= i);
  set Treatment;
  do i= 1 to &numberofcontrols;
    RandomNumber= ranuni(12345);
    output;
  end;
run;

/* Randomly sort both datasets */
proc sort data= _Treatment0 out= _Treatment(drop= RandomNumber);
  by RandomNumber;
```

*Alternative mit R: MatchIt*

Ausführung für ein matching mit Distanzobergrenze, ein Partner, ohne zurücklegen:

```
%psmatching(
  datatreatment=TNdatei,
  datacontrol=KGdatei,
  method=Caliper,
  caliper=0.01,
  nummerofcontrols=1,
  replacement=no);
```

Ergebnis:

Tabelle mit Matching-Paaren

| MatchedTo | IdSelected |
|-----------|------------|
| Treated   | Control    |
| ...       | ...        |
| 3825357   | 7412182    |
| ...       | ...        |

# 10a. Messmethodik Einsparungen (5)

Matching-Ergebnis: Beispiel zu Paarbildung

| ID      | Alter | Alter_ m25h2 | Anzahl Kinder | Verheir atet1 | Geschl Weibl1 | Freiw_ Vers1 | Leistun gen_VJ | Leistung en_VVJ | Gruppe | TNWahrsch |
|---------|-------|--------------|---------------|---------------|---------------|--------------|----------------|-----------------|--------|-----------|
|         |       |              |               |               |               |              |                |                 |        | Treated   |
| 6333173 | 40    | 225          | 0             | 0             | 1             | 1            | 850            | 710             | K      | 0,859965  |
| 1451185 | 39    | 196          | 0             | 0             | 1             | 1            | 750            | 840             | K      | 0,859362  |
| 7412182 | 41    | 256          | 0             | 0             | 1             | 1            | 840            | 720             | K      | 0,855697  |
| 3825357 | 40    | 225          | 0             | 0             | 1             | 1            | 830            | 770             | T      | 0,855078  |
| 5671754 | 38    | 169          | 0             | 0             | 1             | 1            | 800            | 850             | K      | 0,853835  |
| 2744563 | 39    | 196          | 0             | 0             | 1             | 1            | 850            | 800             | K      | 0,851953  |
| 4599068 | 37    | 144          | 0             | 0             | 1             | 1            | 800            | 900             | K      | 0,848129  |

| MatchedTo | IdSelected |
|-----------|------------|
| Treated   | Control    |
| ...       | ...        |
| 3825357   | 7412182    |
| ...       | ...        |

Absteigend  
sortiert nach  
TNWahrsch

Paar

Nach dem Matching werden nur noch die Paare betrachtet.

Matching-Güte prüfen über z.B. Signifikanztests und Standardized-Bias (SB)

**Schätzung des Effektes: Difference-In-Differences (DID) Schätzer**