Datenanalyse mittels R im SQL Server

arelium GmbH

- Microsoft-Partner mit den Kompetenzen
  - Gold Data Analytics (Business Intelligence),
  - Silver Application Development
- Gründungsjahr: 2007
- Firmensitz: Langenfeld (Rheinland)
- Mitarbeiter: ca. 15
- Geschäftsführer: Markus Delhofen, Martin Kopp
- Kunden: verschiedene Branchen mit Schwerpunkt auf Medien

Lea Kaufmann

- Bachelor of Science Mathematik
- Master mit Schwerpunkt stochastische Analysis/ Statistik
- Werkstudentin bei der arelium GmbH
- Datenanalyse in R

Markus Delhofen

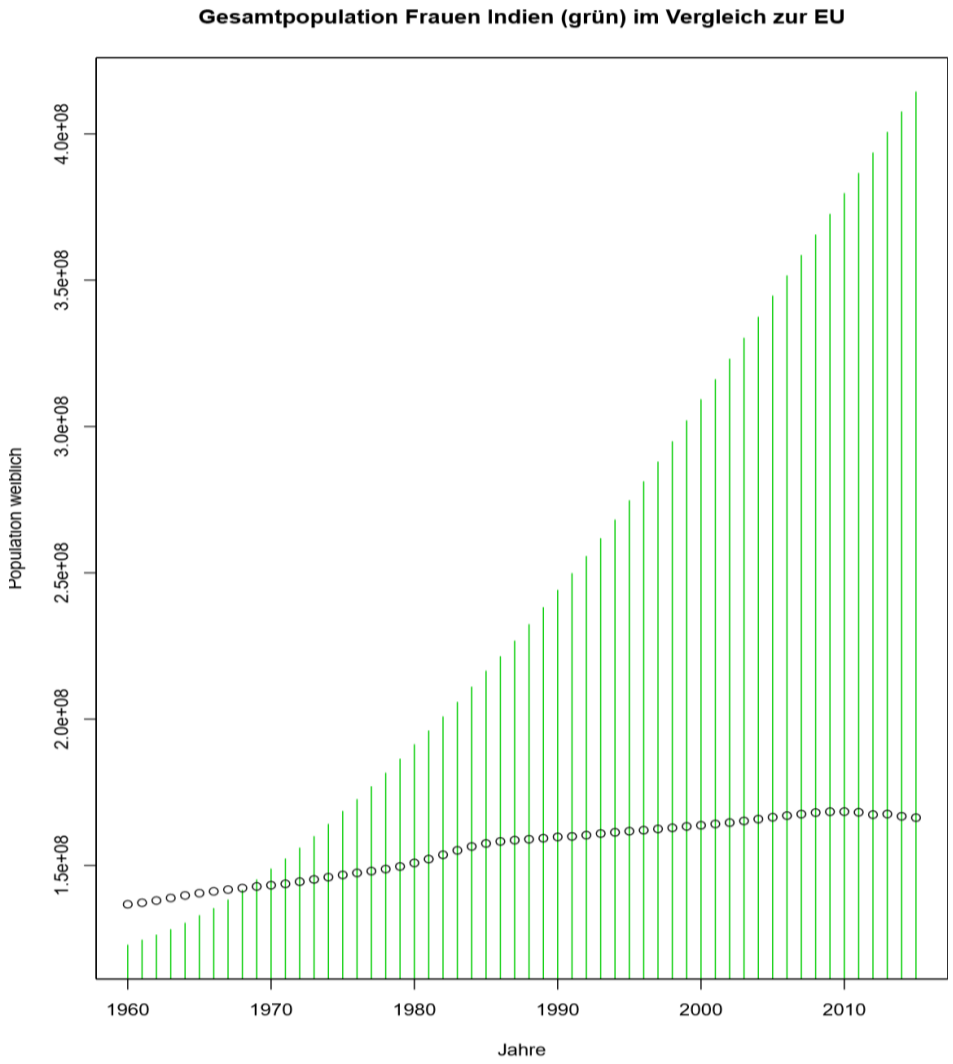
- Diplom Wirtschaftsinformatiker
- BI-Architektur
- BI-Entwicklung
- MCSE Business Intelligence
- Aktuelle Projekte
  - DWH-Architektur
  - SSIS/SSAS-Entwicklung

Was macht man mit R?

- Statistische Berechnungen
  - Vorhersagen
  - Schätzen von Parametern
  - Testen von Hypothesen
  - Modellierung von (stochastischen) Prozessen (Brownsche Bewegung)
- Grafiken

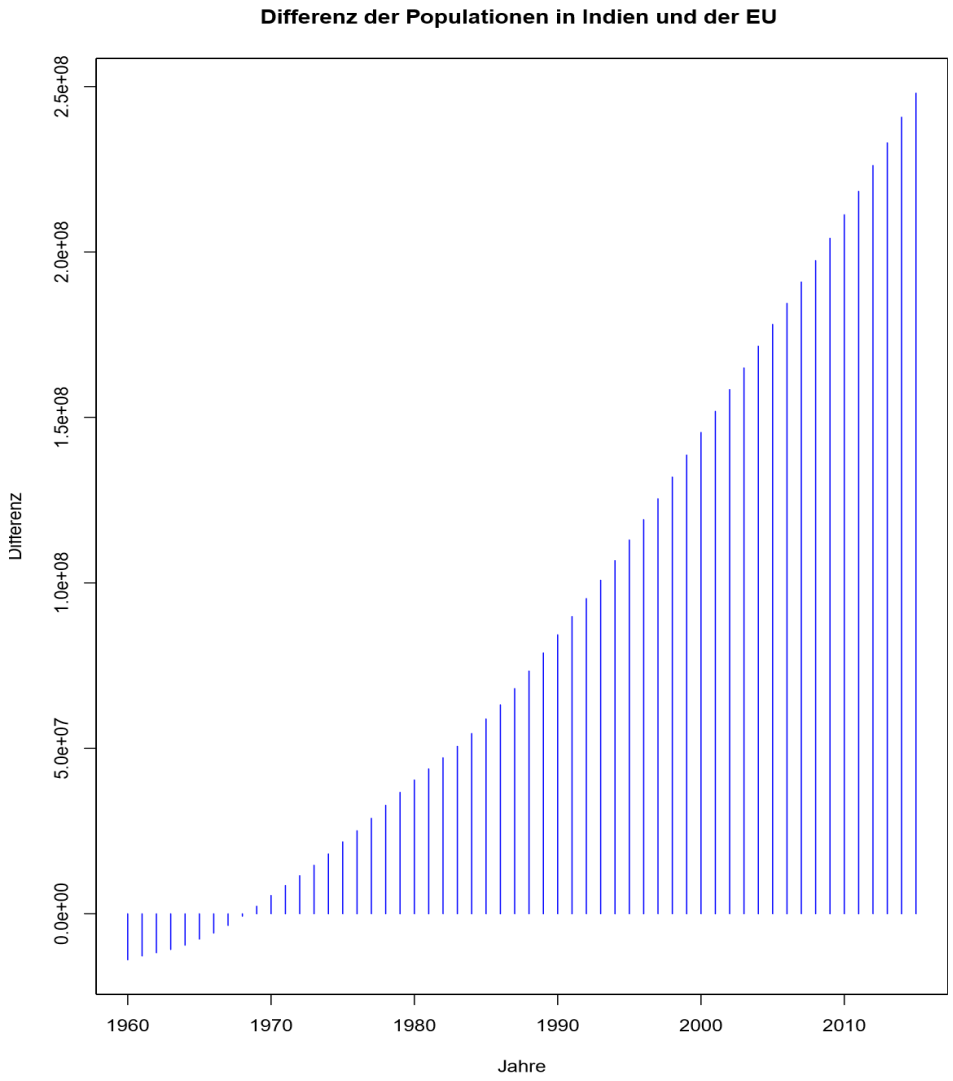
Beispiele

Beispiel A



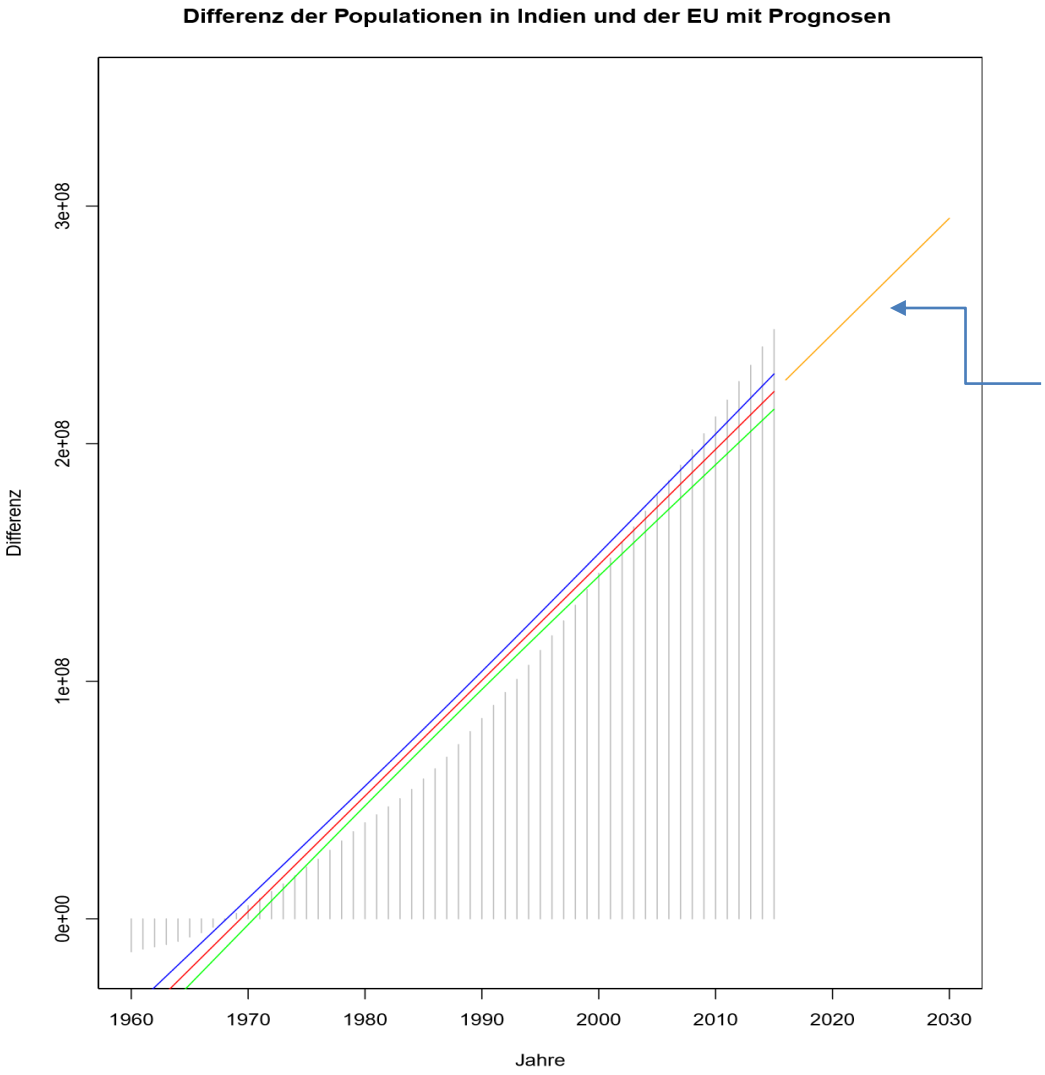
- Verlauf der Population von Frauen zwischen 15 und 64 in Indien im Vergleich zur EU
- Linearer Anstieg der Differenz zu erwarten
- Idee: betrachte die Differenz und erstelle eine Prognose für die Zukunft

Beispiel A



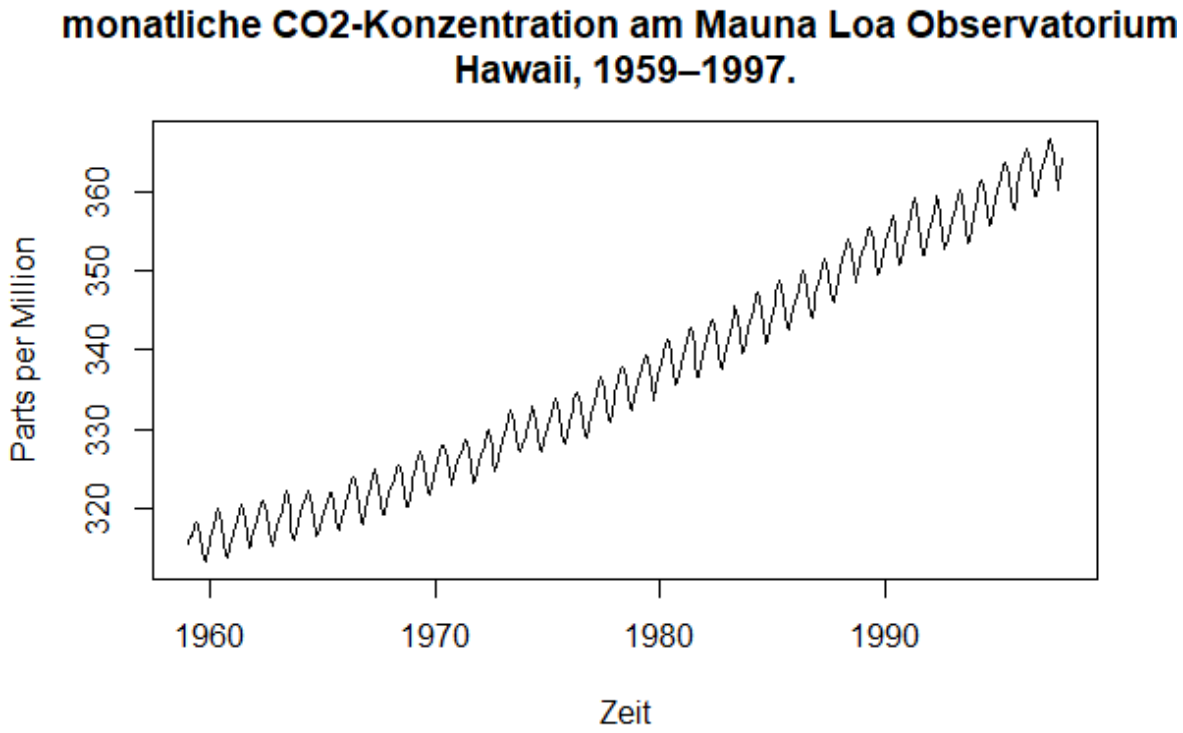
- Lineare Approximation direkt auf den ersten Blick sinnvoll, später Daten wo eine lineare Approximation nicht sinnvoll wäre
- In der Praxis sind Prognosen von großer Bedeutung
- Idee Prognose: erkenne eine Gesetzmäßigkeit in Abhängigkeit von der Jahresanzahl (oder bei anderen Modellen zB Wert des Vorjahres etc)
- Modellannahme hier: **Diff=a+Jahr*b**

Beispiel A



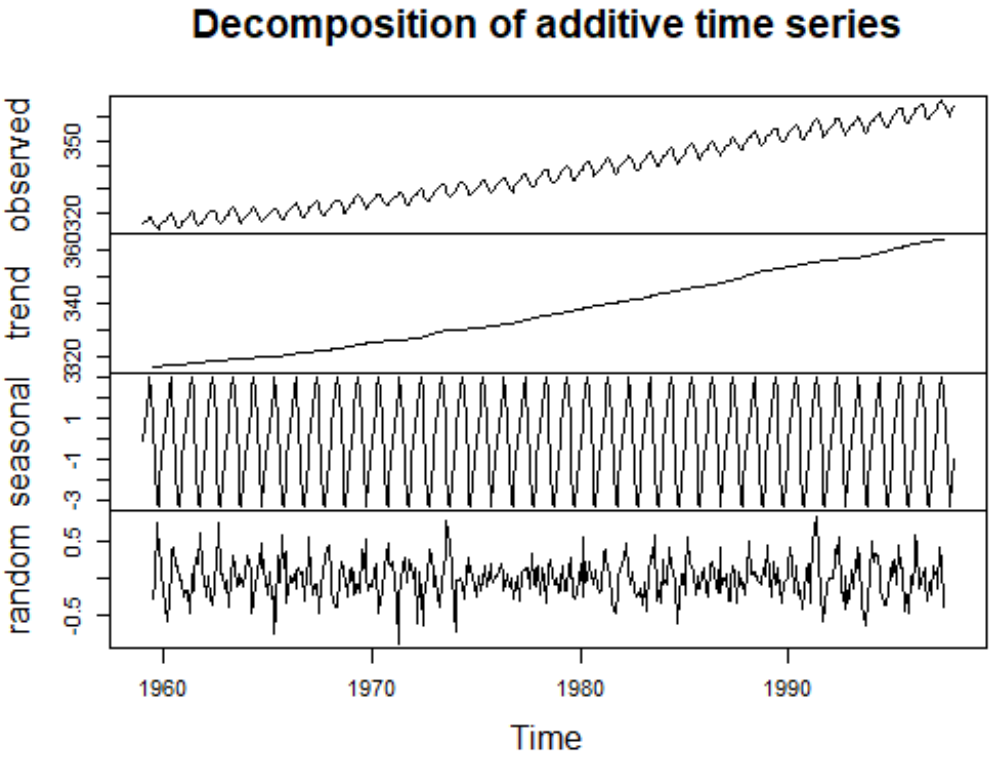
- Lineare Regression liefert R mittels **lm(Eingabedaten)**
- Anschließende Prognose erstellen (unter Angabe eines Vertrauensbereiches) mittels **predict(),** erhalten geschätzte Trendgerade
- Es ist zu sehen, dass das lineare Modell hier sinnvoll ist, R liefert dafür auch einen Indikator bei der Bestimmung des Modells (hier 0,97)

Beispiel B



- Trend und Saisonkomponente erkennbar
- Lineares Modell nicht sinnvoll
- Je nach Trend und Saison geeignetes Vorhersagemodell wählen
- Zur weiteren Analyse: Zerlegung der Zeitreihe

Beispiel B



Additives Modell:
ZR = T + S + R

- Mittels **decompose()** zerlegt R in einem Befehl die Zeitreihe in die gewünschten Komponenten

Beispiel C

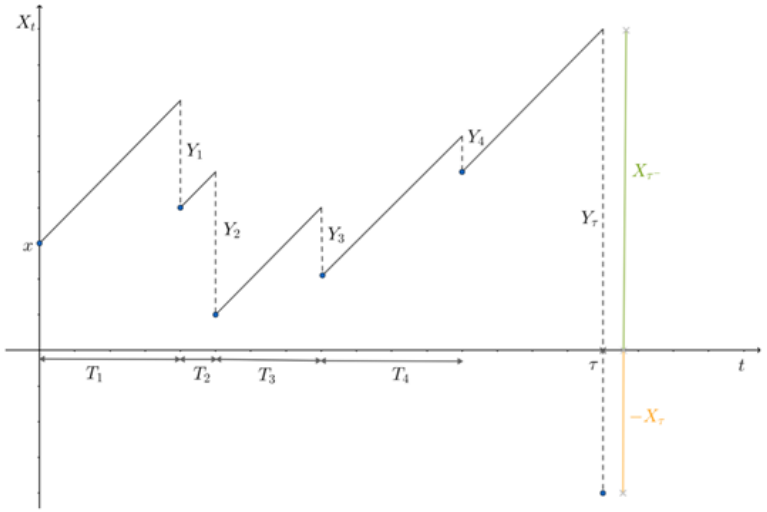


Figure 3.1: Verlauf der Reserve X_t im Cramér Lundberg Modell

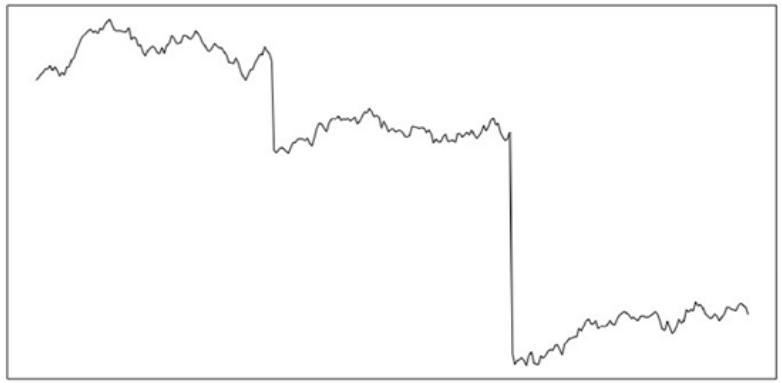
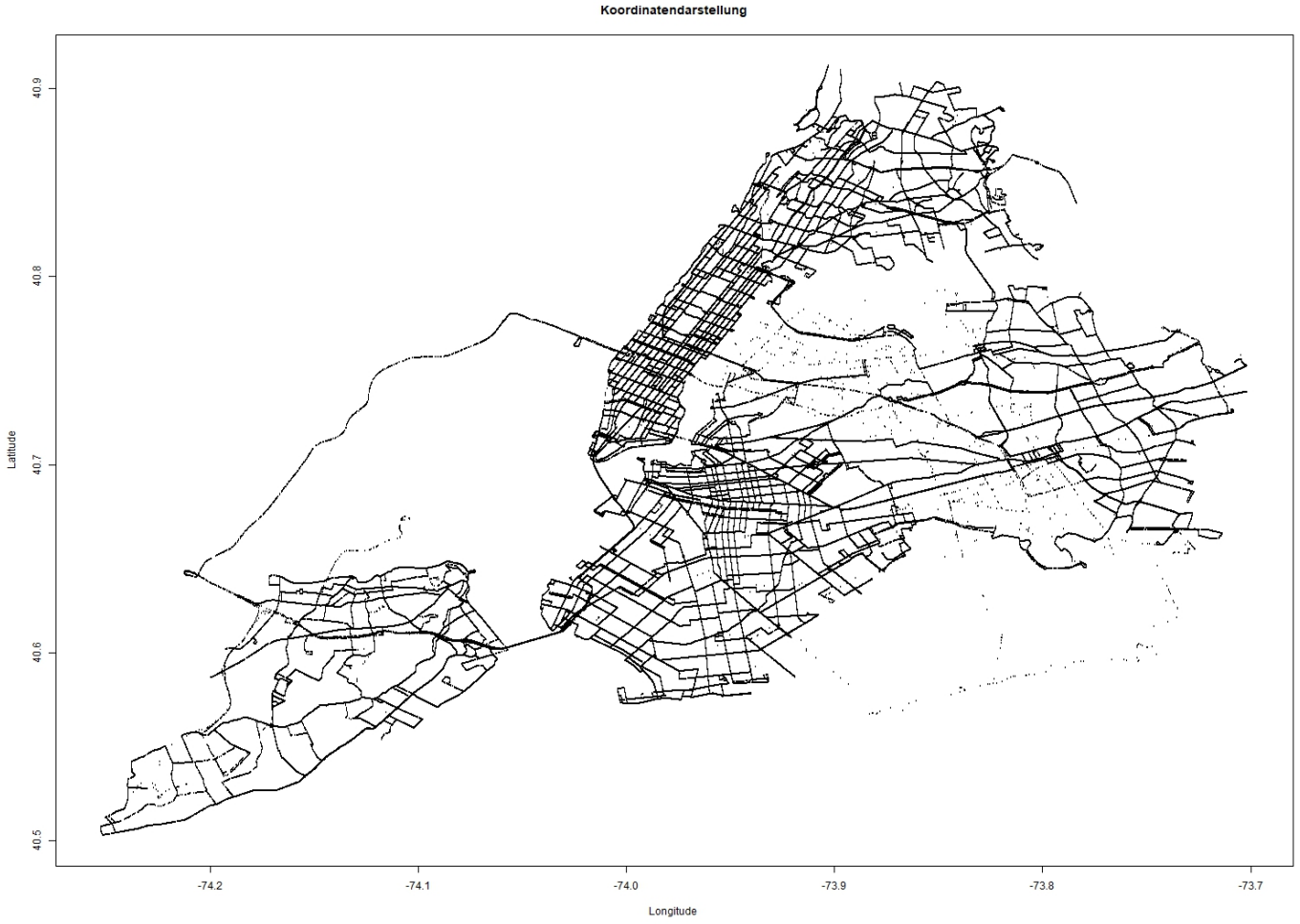


Figure 8.1: Verlauf des Versicherungsportfolios im perturbierten Risikomodell

- Simulation von Risikoprozessen mit Perturbationsanteil möglich (rechts)
  - Brownsche Bewegung (Zitterbewegung) beschreibt Unsicherheiten
  - Verwendung zum Beispiel beim Modellieren von Versicherungsportfolios

Beispiel D

- Plot durch Einbindung von R in SQL



Grundlagen in R

Rechnen in R

```
> 5+6
[1] 11
> 6*9
[1] 54
> sqrt(16)
[1] 4

> v<-5
> w<-8
> v-w
[1] -3
> w*sqrt(v)
[1] 17.88854
```

- Kennt mathematische Funktionen wie sqrt(),cos(),sin(),… sowie mathematische Größen wie Pi etc.
- Zuweisung mit <-
  - Gilt auch für Zuweisung von Vektoren, Matrizen, Data Frames etc.
- Löschen von Variablen mit **rm()**

Vektoren

- Mit **c()** („combine“) werden die angegebenen Elemente in einem Vektor aneinander gereiht
- Funktioniert auch als Kombination von Vektoren

```
> v<-c(1,2,-7)
> v
[1]  1  2 -7
> w<-c(c(1,2),c(-8,9,0,322,1))
> w
[1]   1   2  -8   9   0 322   1
```

Vektoren

- Mit **seq()** wird eine Folge erstellt vom angegebenen Start- bis Endpunkt in der gewünschten Schrittweite (darf auch negativ sein)
- Weitere Möglichkeit ist **rep()**

```
> seq(from=3,to=30,by=3)
 [1]  3  6  9 12 15 18 21 24 27 30
> seq(from=3,to=-30,by=-3)
 [1]   3   0  -3  -6  -9 -12 -15 -18 -21 -24 -27 -30
> seq(from=30,to=3,by=-3)
 [1] 30 27 24 21 18 15 12  9  6  3
```

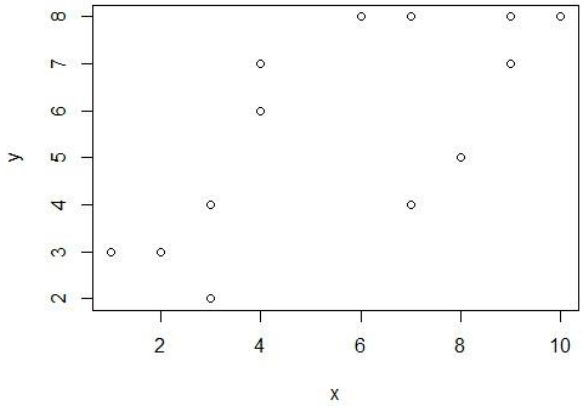
Demo

Funktionen für Vektoren

Funktionsaufruf	Funktion
sum(), prod()	Liefert Summe/Produkt der Einträge
min(),max()	Minimum/Maximum der Einträge
range()	Minimum/Maximum der Einträge in einem Vektor
length()	Länge des Vektors
sd()	Empirische Standartabweichung
mean()	Arithmetisches Mittel der Einträge
var()	Empirische Varianz der Einträge

Korrelation

```
> x <- c(2,4,3,6,7,8,1,4,9,3,10,7,4,9)
> y <- c(3,6,2,8,4,5,3,7,7,4,8,8,6,8)
```



- Testen auf **linearen** Zusammenhang zweier Merkmale mithilfe des Korrelationskoeffizienten
- Vermutung: je größer x, desto größer y
- Mit **cor.test(x,y)** liefert R einen Wert zwischen –1 und 1 (hier 0.6956255)
  - 1 steht für den perfekte positive linearen Zusammenhang
  - 0 steht für keinen *linearen* Zusammenhang

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\cdot\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

Bsp. Berechnung **ohne** R

Zugriff Einträge

```
> x<-rep(c(1,6,3),2)
> x
[1] 1 6 3 1 6 3
> x[2]
[1] 6
> x[2:5]
[1] 6 3 1 6
> x[x>3]
[1] 6 6
> x>3
[1] FALSE  TRUE FALSE
FALSE  TRUE FALSE
```

- X[n] greift auf Eintrag n zu
- X[n:m] greift auf Eintrag n bis m zu
- Funktioniert bei Matrizen analog mit M[i,j] Zeile i, Spalte j
- Mit x[x>k] werden nur die Einträge von x ausgegeben, die größer als k sind
- Funktioniert auch für andere Abfragen in Klammern, die true oder false liefern

Matrizen

```
> x<-c(1,2,3)
> y<-c(4,5,6)
> cbind(x,y)
     x y
[1,] 1 4
[2,] 2 5
[3,] 3 6
> rbind(x,y)
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6
```

- Rbind() reiht die Vektoren zeilenweise aneinander und cbind() spaltenweise
- Matrizen sinnvoll zum lösen linearer Gleichungssysteme
- Zum speichern von Daten nicht geeignet, da die Spalten nicht von verschiedenen Datentypen sein können

Lösung: Data Frames

Funktionen für Matrizen

Funktionsaufruf	Funktion
%*%	Matrixmultiplikation
chol()	Choleski Zerleung
diag()	Abfragen/Setzen der Hauptdiagonalen
rowsums(),colsums()	Zeilen/Spaltensumme
rowmeans(),colmeans()	Arithmetisches Mittel der Zeilen/Spalten
eigen()	Eigenwerte und Eigenvektoren
nrow(),ncol()	Anzahl Zeilen/Spalten
t()	Transponieren einer Matrix

Demo

Data Frames

- Speichern von verschiedenen Datentypen in einer Matrixähnlichen Struktur
- Damit lassen sich Vektoren/Matrizen beliebiger Datentypen spaltenweise zusammenfassen
- Länge der Vektoren/Anzahl Zeilen der Matrizen muss gleich sein
- **Vorteil zu Matrizen:** verschiedene Datentypen
- Typische Datenstruktur für statistische Analysen

Data Frames

- **Data.frame(Vektor1,Vektor2,..)** fügt die angegebenen Vektoren in einem Data Frame zusammen

```
> name<-c("Name1","Name2","Name3")
> groesse<-c(153,154,155)
> schuhgroesse<-c(40,41,42)
> D<-data.frame(name,groesse,schuhgroesse)
```

Data Frames

- **View(Name des Data Frames)** zeigt das Data Frame an, mit **fix(Name des Data Frames)** lässt es sich nachträglich bearbeiten

	name	groesse	schuhgroesse
1	Name1	153	40
2	Name2	154	41
3	Name3	155	42

Data Frames

- Für aussagekräftige Grafiken und Analysen müssen wir auf bestimmte Zeilen/Spalten des Data Frames zugreifen können und ggf. nach bestimmten Eigenschaften filtern
- **Head(Dataframe,n)** zeigt die obersten n Einträge des Dataframes an
- **Dataframe[i,]** zeigt alle Einträge von Zeile i an (Spalte analog)
- **Dataframe[i,j:k]** zeigt alle Einträge von Zeile i von Spalte j bis k an
- Mit **Dataframe$Spaltenname** können wir uns eine bestimmte Spalte als Vektor ausgeben lassen, dort können dann weitere Einschränkungen gewählt werden

Data Frames

- Mit **subset()**kann eine bestimmte Teilmenge des Data Frames extrahiert werden
- **Which(Bedingung)** liefert die Zeilennummer des Eintrags, der diese Bedingung erfüllt (bzw. einen Vektor mit den Einträgen aller Zeilen, die diese Bedingung erfüllen)
- **Any(), All()**prüfen Bedingung und liefern true/false

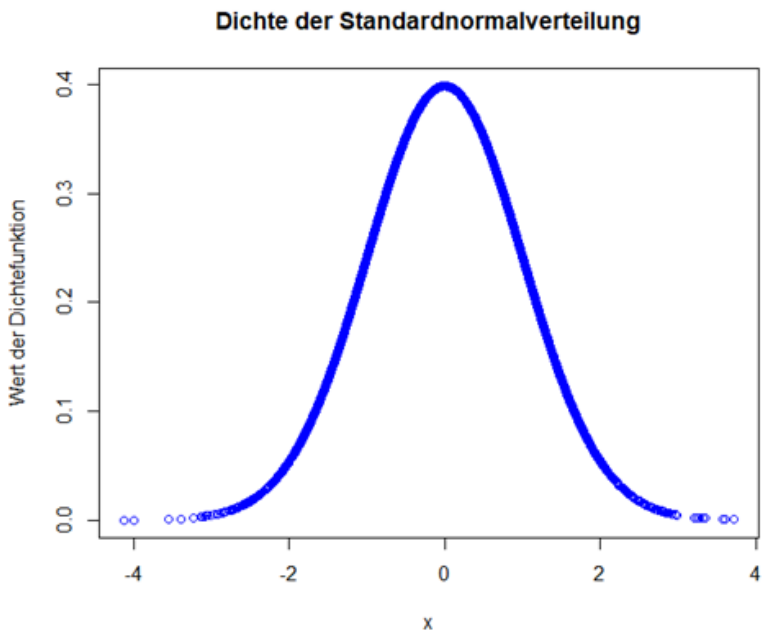
```
> subset(D,!D$name=="Name1")
  name groesse schuhgroesse
2 Name2     154          41
3 Name3     155          42
> which(D$name=="Name1")
[1] 1
> any(D$groesse>154)
[1] TRUE
```

Demo

Grafiken

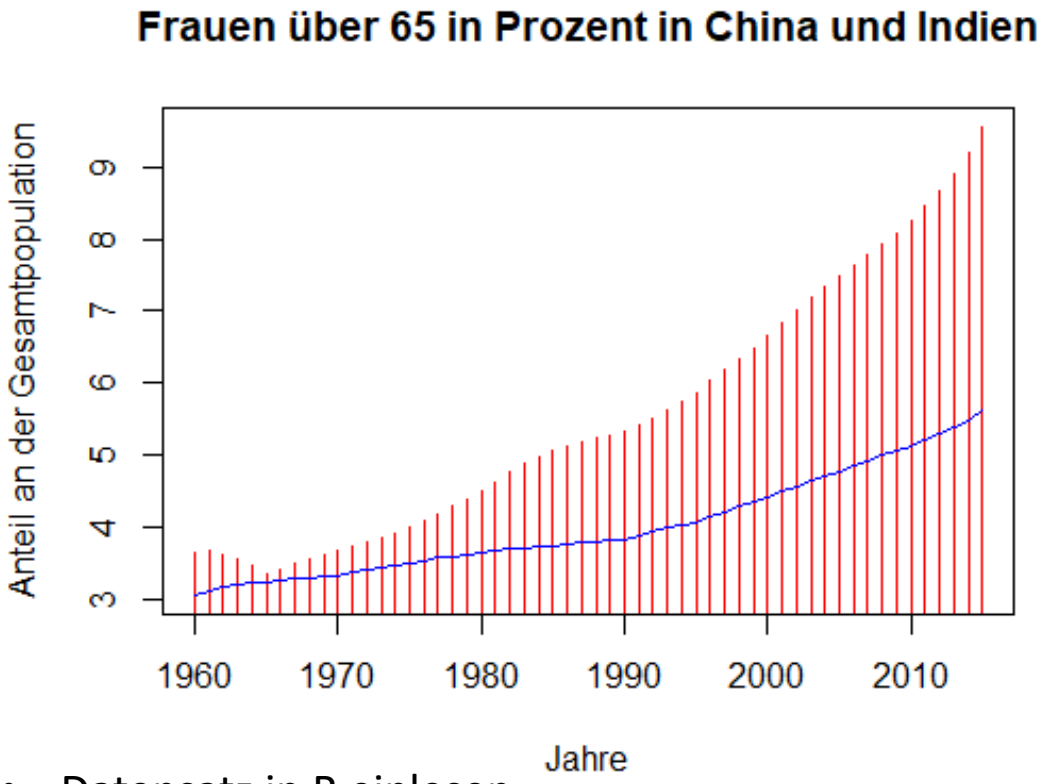
- Darstellung zweier Vektoren x und y in einem plot mittels **plot(x,y,Option1,Option2,…)**
  - Hinzufügen weiterer Linien oder Punkte von Einträgen zweier Vektoren mit **Lines(),points()**
  - Die Einträge solcher Vektoren extrahiert man dann typischerweise vorher aus dem gegebenen Data Frame
- weitere Möglichkeiten Grafiken zu erstellen: **hist(), boxplot(),pie(),curve(),…**

Grafiken



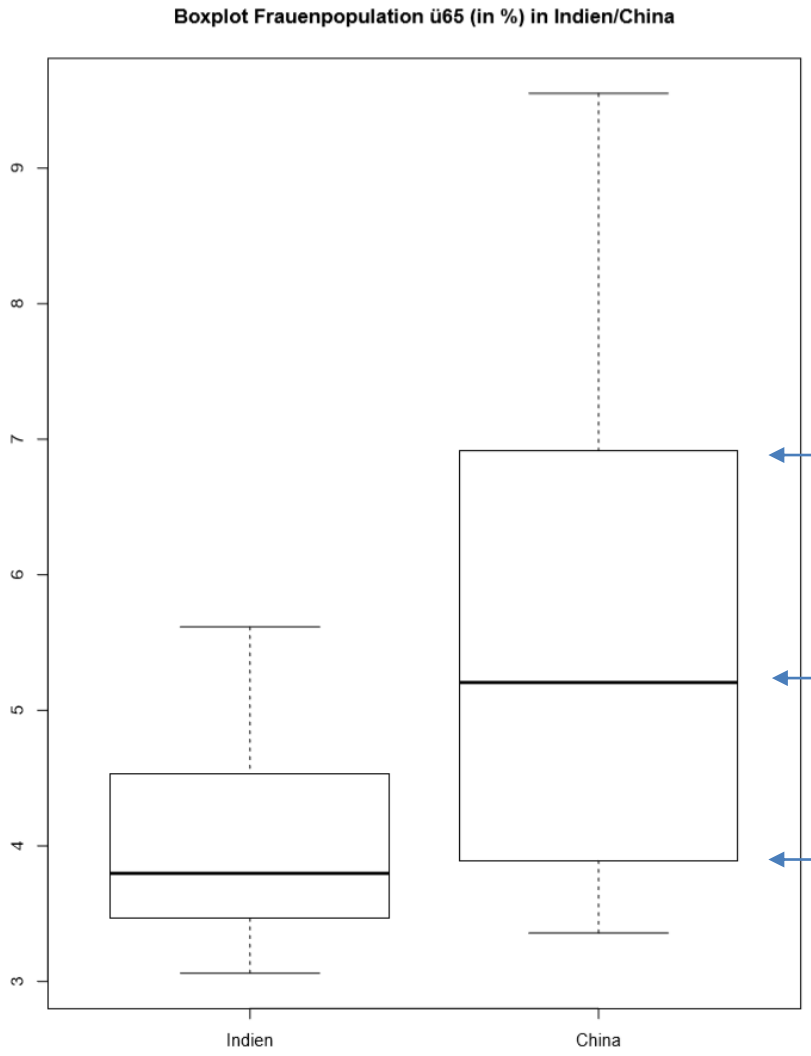

```
x<-rnorm(10000)
> y<-dnorm(x)
> plot(x,y,main="Dichte der Standardnormalverteilung",xlab="x",ylab="Wert der
Dichtefunktion",col='blue')
```

Frauen über 65 (in % der Gesamtpopulation)



- Datensatz in R einlesen
- Es wird deutlich, dass in China der prozentuale Anteil der Frauen über 65 deutlich steigt
- Große Streuung um den Median, Verdeutlichung mittels **Boxplot**

Boxplot



- Boxplot verdeutlicht Lage und Streuungsmaße
- Ausdehnung der Box ist der Wertebereich, in dem sich die mittleren 50 % der Daten befinden

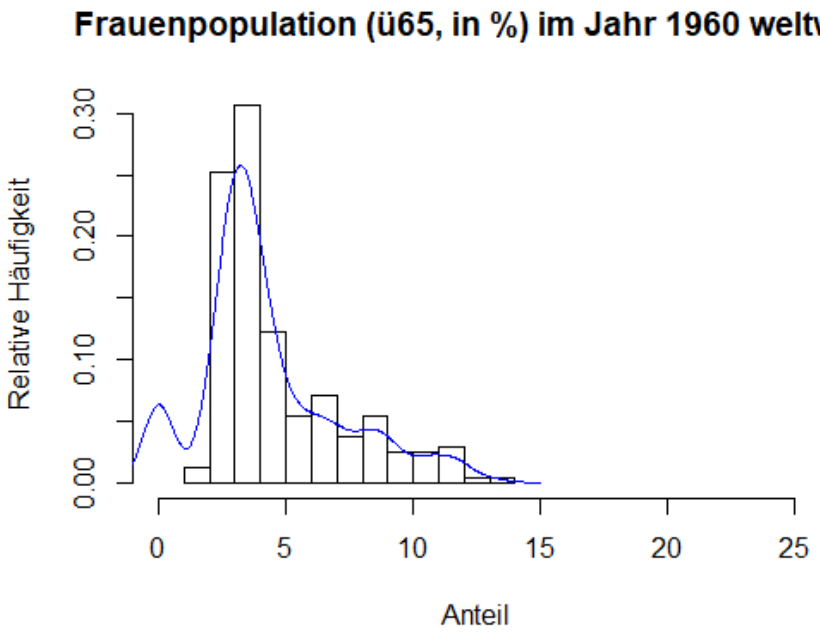
Oberes Quartil

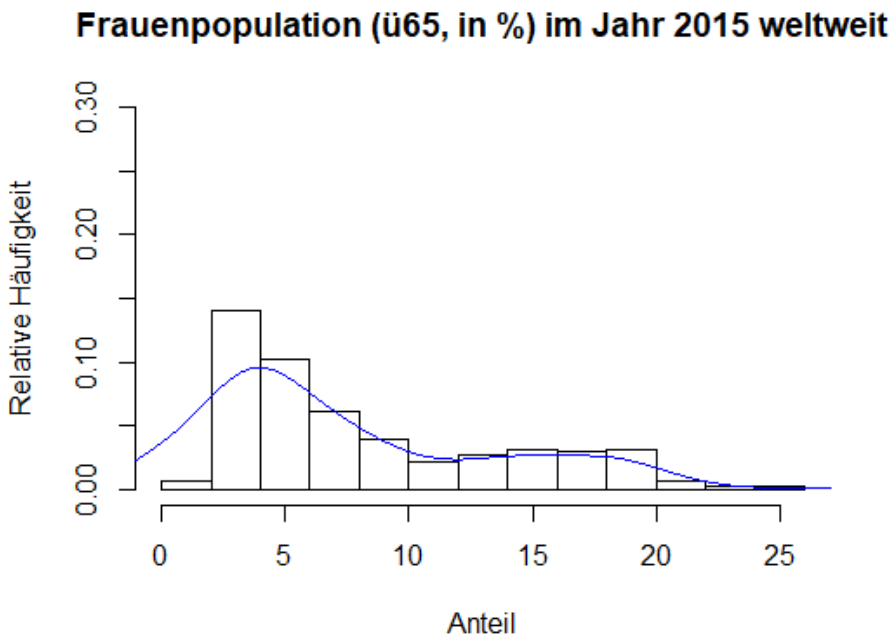
Median

Unteres Quartil

Frauen über 65 (in % der Gesamtpopulation)

Histogramm der prozentualen Anteile weltweit mit relativen Häufigkeiten und Dichteplot





Wichtige statistische Parameter mit einem Befehl: **summary()**

Machine Learning Services

- SQL Server 2016: R Services
- seit SQL 2017: Machine Learning Services

- Varianten:
  - Als Service mit Verbindung zur relationalen Engine
  - Standalone ohne Verknüpfung zur relationalen Engine

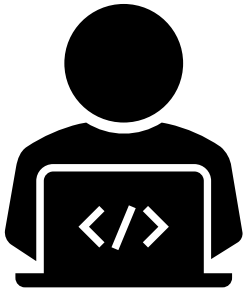
R Script ausführen

- sp_execute_external_script

```
EXEC sp_execute_external_script
     @language = N'R'
    ,@script = N''
```

- Ein- und Ausgabe ist ein Data Frame

- Script und Eingabe werden als String übergeben



Demo

Fazit

- Einbindung von R Skripten ist sehr einfach
- Ergebnisse können einfach in Tabellen zur Weiterverarbeitung übernommen werden
- In Skripten müssen Hochkammatas maskiert werden, was Copy&Paste erschwert
- Es kann nur ein Data Frame übergeben werden
- Der Umgang mit Plots ist umständlich

Vielen Dank für die Aufmerksamkeit!

lea.kaufmann@arelium.de

markus.delhofen@arelium.de