UNIVERSITÄT LEIPZIG

Einführung in die Ökonometrie - Übung

Einführung in die angewandte Regressionsanalyse mit „R“

- **Allgemeine Einführung in R**
- **Deskriptive und explorative Datenanalyse mit R**
- **Regressionsanalyse mit R**
- **Programmieren mit R**

Allgemeine Einführung in R

Ziele

- Erwerben von Grundkenntnissen über das Statistikpaket R
- Anwenden statistischer Methoden mit R

Literatur

- Rainer Schlittgen: Statistische Auswertungen mit R, Oldenbourg Verlag 2004
  (zwei Kapitel liegen eingescannt vor)
- W. N. Venables & D. M. Smith: An Introduction to R
  (PDF-Dokument aus der R-Hilfe)

Was sind Statistikpakete? (I)

- Software zur Datenanalyse

- Sie beinhalten folgende Funktionalitäten:
  - Schnittstelle zum Einlesen der Daten
    - ◊ manuelle Eingabe, Datei, Datenbank, Schnittstellen zu anderen Programmen (z.B. Excel)
  - Datenvorbearbeitung
    - ◊ Fehlwertebearbeitung
    - ◊ Datentransformation (Skalentransformationen, Zentrieren, Standardisieren, usw.)
  - Analysemethoden
  - Skriptsprache / Programmiersprache

Was sind Statistikpakete? (II)

- Ergebnispräsentation
  - ◊ Grafik, Diagramme, Tabellen
- Exportfunktion
  - ◊ Datendateien (verschiedene Formate), Dokumente, Datenbank
- Dokumentation
  - ◊ Dokumentation der implementierten Methoden
  - ◊ Dokumentation der Skriptsprache

Welche Statistikpakete gibt es? (i)

- Kommerzielle Pakete
  - Generelle Pakete
    - ◊ SPSS
    - ◊ SAS
    - ◊ S-Plus
    - ◊ Stata
    - ◊ MATLAB
    - ◊ STATISTICA und viele andere
  - Spezialisierte Pakete
    - ◊ EViews (ökonometrische Analyse)
    - ◊ AutoBox (Zeitreihenanalyse)
    - ◊ BayesX (Bayes-Statistik)
    - ◊ Mentor (Marktforschung) und viele andere

Welche Statistikpakete gibt es? (ii)

- *Open Source* Pakete
  - Generelle Pakete
    - ◊ **R** (Vetter von S-Plus) und andere
  - Spezialisierte Pakete
    - ◊ JMulTi (Zeitreihenanalyse)
    - ◊ gretl (ökonometrische Analyse)
    - ◊ Weka (Maschinenlernen) und viele andere

- *Public Domain* und *Freeware* Pakete

Proprietäre vs. Open Source Software

Proprietäre Software	*Open Source* Softweare
implementierte Methoden i.d.R. gut kommentiert (+)	i.d.R. kostenlos beziehbar (+)
hoher Bedienungskomfort durch ausgereifte GUI (+)	leicht erweiterbar, weil quelloffen (+)
Support und Schulungen erhältlich (+)	von anderen Nutzern erstellte Erweiterungen sind im Internet verfügbar (+)
Lizenzgebühren (-)	schwankende Qualität der Erweiterungen (-)
begrenzt erweiterbar (-)	Einarbeitungsaufwand (-)

Was ist R?

- R ist eine Sprache und Arbeitsumgebung für die statistische Datenbearbeitung bestehend aus
  - ◊ der Programmiersprache R selbst,
  - ◊ Laufzeitumgebung mit Interpreter und Graphic-Engine,
  - ◊ Debugger,
  - ◊ Methoden-Bank (Statistikfunktionen).
- R gibt es für die Betriebssysteme Unix, Linux, Windows, Mac.
- Der Kern von R ist die computerinterpretierbare Sprache, welche die folgenden Konstrukte / Konzepte unterstützt
  - ◊ Verzweigungen,
  - ◊ Schleifen,
  - ◊ Modularisierung (= Funktionen).

Warum R?

- R ist *Open Source* und kann daher kostenfrei genutzt werden.
- Proprietäre Lösungen wie SPSS, Splus, etc. kosten Lizenzgebühren.
- Aufgrund der permanente Weiterentwicklung durch die R-Community ist R nahe am „aktuellen Rand“ der Forschung.
- Erweiterbarkeit: In den Basispaketen nicht verfügbare Methoden können selber programmiert werden.

Download / Installation

- Aus dem Internet herunterladen von der Website http://www.r-project.org/.
- Weiter geht’s live ... :)

Grundlegende Konzepte (I)

- R starten:
  - `rgui.exe`
- R beenden:
  - `q()`
- Befehle aufrufen:
  - `befehlsname(parameter)`
  - z.B. `help()` oder `demo()`
- Zeilen, die mit `#` beginnen, sind Kommentare.

Grundlegende Konzepte (II)

- Variablen
  - Variablen sind Platzhalter für Daten.
  - Zuweisung: `Variable <- Ausdruck`
    - Bsp.: `sieben <- 7` (sieben wird der Wert 7 zugewiesen)
  - Variablen werden mit ihrer ersten Benutzung initialisiert.
  - Durch Eingabe des Variablennamens kann der Wert der Variablen auf der Konsole angezeigt werden.
    - Bsp.:
      ```
      sieben
      [1] 7
      ```
  - Mit `ls()` werden alle belegten Variablen angezeigt.

Grundlegende Konzepte (III)

- Variablen nehmen Daten auf.
- Die folgende Datentypen werden unterschieden:
  - einfache Datentypen
    - Zahlen
    - Wahrheitswerte
    - Zeichenketten
  - komplexe Datentypen
    - Vektoren
    - Matrizen
    - Datensätze

Grundlegende Konzepte (IV)

- Zeichenkette:
  - Eine Zeichenkette ist eine beliebige Folge von Ziffern, Buchstaben und andere Zeichen.
  - Zeichenketten bildet man mit Hilfe doppelter Ausführungszeichen.
    - Bsp.:

```
> Satz <- "Dies ist eine Zeichenkette!"
> Satz
[1] "Dies ist eine Zeichenkette!"
```

Grundlegende Konzepte (V)

- Vektor:
  - Ein eindimensionales Datenfeld (Array) gleichartiger Datenobjekte.
  - Ein Vektor wird mit der Funktion `c()` oder dem Sequenzoperator `:` erzeugt.
    - Bsp.:

```
> v <-c(1,2,4,8,16)

> v
[1] 1 2 4 8 16

> s <-(1:10)
> s
[1] 1 2 3 4 5 6 7 8 9 10
```

- Matrix:
  - Ein zweidimensionales Datenfeld (Array) gleichartiger Datenobjekte.
  - Eine Matrix wird durch zweidimensionale Anordnung eines Vektors erzeugt.
    - ◊ Bsp.:

```
> m<-matrix(c(8,9,2,3,10,0),2,3)

> m
     [,1] [,2] [,3]
[1,]    8    2   10
[2,]    9    3    0
```

- Datensatz:
  - Verschiedene Datentypen in verschiedenen Spalten möglich.
  - Die Spalten einzeln über Variablennamen ansprechbar.

Grundlegende Konzepte (VI)

- Grundrechenoperationen
  - `a + b` Addition
  - `a - b` Subtraktion
  - `a * b` Multiplikation
  - `a / b` Division
  - `a^b` oder `a**b` Potenzieren
  - `a %% b` Divisionsrest
  - `a %/% b` ganzzahlige Division
  - `sqrt(a)` Quadratwurzel
  - `exp(a)` Exponentialfunktion
  - `log(a)` natürlicher Logarithmus

  u.s.w.

◊ Bsp.:

```
> sieben/2
[1] 3.5

> 2*v
[1]  2  4  8 16 36

> 8+m
     [,1] [,2] [,3]
[1,]   16   10   18
[2,]   17   11    8

> exp(sieben)
[1] 1096.633

> sqrt(v)
[1] 1.000000 1.414214 2.000000 2.828427 4.242641
```

Grundlegende Konzepte (VII)

- Vektoren- und Matrizenoperationen
  - eventuell später …

- Vergleichsoperationen
  - `a < b` kleiner
  - `a <= b` kleiner gleich
  - `a == b` gleich
  - `a != b` ungleich
  - `a >= b` größer gleich
  - `a > b` größer

◊ Bsp.:

```
> sieben==6
[1] FALSE

> 10>v
[1]  TRUE  TRUE  TRUE  TRUE FALSE

> 8!=m
      [,1] [,2] [,3]
[1,] FALSE TRUE TRUE
[2,]  TRUE TRUE TRUE

> zk<-"Zeichenkette"
> zk=="Zeichenkette"
[1] TRUE
> zk=="ZEichenkette"
[1] FALSE
```

Grundlegende Konzepte (VIII)

- Zugriff auf Array-Teile:
  - ◊ Bsp.:

```
> v[1]                    Erstes Element des Vektors.
[1] 1

> v[2:4]                  Elemente 2 – 4 des Vektors.
[1] 2 4 8

> m[1,2]                  Element 1,2 der Matrix.
[1] 2

> m[1,]                   Erste Zeile der Matrix.
[1]  8  2 10

> m[,2]                   Zweite Spalte der Matrix.
[1] 2 3
```

Grundlegende Konzepte (IX)

- Daten aus Dateien laden:
  - Funktion `scan` liest Daten aus externer ASCII-Datei in einen Vektor.
    - Bsp.: `scan("benzinverbrauch.r")`
  - Funktion `read.table` liest Daten aus externer ASCII-Datei mit bereits spaltenförmig vorformatierten Daten in einen *Data Frame*.
    - Bsp.: `read.table("pkw.r",header=TRUE)`
  - Funktion `read.csv2` liest Daten aus externer EXCEL-CSV-Datei in *Data Frame*. *Data Frame*. (CSV = comma separated values)
    - Bsp.:

```
read.csv2("mietspiegel99.csv",header=TRUE)
```

Weiter geht es mit

Rainer Schlittgen, Statistische Auswertungen mit R