# Einführung in R

# 1 Was ist R?

R ist eine Open-Source-Software sowie eine flexible Programmiersprache für statistische Datenanalyse und Grafikerstellung. Zugleich ist R ein Software-System, das auf die Sprache R zurückgreift.

# 2 Geschichte

Mitte der 1970er Jahre wurde die Programmiersprache S in den USA von John Chambers und Kollegen (Bell Laboratories) entwickelt, um statistische Simulationen durchzuführen und Grafiken zu erstellen. Im Anschluss daran wurde Ende der 1980er Jahre daraus ein kommerzielles Produkt entwickelt: S-PLUS (*www.insigthful.com*). Diese Software wird seither zur explorative Datenanalyse und statistischen Modellierung von Daten vertrieben.

R ist eine weitere Entwicklung, die auf der Sprache S aufbaut. Von Robert Gentleman und Ross Ihaka Anfang der 1990er initiiert, wird R mittlerweile von zahlreichen Anwendern weiterentwickelt. Das Programm steht als GNU-Projekt (GENERAL PUBLIC LICENCE) frei zur Verfügung.

# 3 Vor- und Nachteile

Vorteile:

- Der Code, und damit die gesamte Software ist frei zugänglich.
- Neue statistische Verfahren und Prozeduren werden aktuell online bereitgestellt.
- Es existiert eine große *Fangemeinde*. Dadurch stehen zahlreiche Informationswege zur Nutzung von R sowie zur Anwendung statistischer Prozeduren und Erstellung von Grafiken zur Verfügung (u.a. Mailinglisten, FAQs, frei zugängliche Literatur).
- R ist plattformunabhängig und läuft unter den Betriebssystemen Linux, Unix, Windows, und MacOS.

Nachteile:

- R besitzt in der Standardinstallation keine umfangreiche Benutzeroberfläche. Für Nutzer, die bisher noch nicht kommandozeilenbasiert gearbeitet haben, ist das ungewohnt.
- R ist keine Interpretersprache. Der Programmiercode wird nicht kompiliert, sondern zur Laufzeit interpretiert. Bei sehr umfangreichen Rechenoperationen und sehr großen Datensätzen, kann es unter Windows zu zeitlichen Verzögerungen kommen.

# 4 Das Programm

Wenn R geöffnet wird, steht ein sehr eingeschränkte Benutzeroberfläche zur Verfügung. Außerdem ist die *R Console* geöffnet. Diese dient als Schnittstelle zwischen Nutzer und Programm. Mit der *R Console* steht ähnlich dem Arbeiten mit MS-DOS eine befehlszeilenbasierte Eingabe zur Verfügung. Es kann z.B. mit den Auf- und Abtasten zwischen älteren Eingaben gewechselt werden. Mit

Datei

erreicht man zentrale Funktionen zur Datensicherung und den Aufruf von Daten und Dateien. Unter anderem kann man über

Datei –> Neues Skript

eine neues Fenster öffnen. Dieser einfache interne Editor kann ebenfalls als Eingabeoption genutzt werden. Der Vorteil des Editors liegt in der Konservierbarkeit der Befehle. Nach dem Code in den Editor eingegeben wurde, kann mit

Bearbeiten –> Ausführung Zeile oder Auswahl

oder mit der Tastenkombination [Strg + R] die aktuelle Zeile (abhängig von der Platzierung des Cursors) oder der vorher ausgewählte Code an die R Console gesendet werden.

Verschiedenes

Unter diesem Menupunkt erhält man unter anderem Auskunft über vorhandene Objekte. über diesen Menüpunkt kann man Objekte löschen.

Unter dem Menüpunkt

Pakete

können zusätzliche Pakete aus dem Internet oder aus lokalen Verzeichnissen installiert werden. In den Paketen sind zusätzliche Daten, Informationen und Funktionen enthalten. In jeder Sitzung können dann weitere Pakete geladen werden, da R beim Start nur einige wichtige Pakete bereithält.

Die Menüpunkte unter:

Windows

beeinflussen die Anordnung der Fenster der Benutzeroberfläche. Unter:

Hilfe

werden zahlreiche Informationsmöglichkeiten angeboten. Unter anderem findet man ein internes Hilfesystem, zentrale Handbücher zu R (zu empfehlen ist die Lektüre der beiden Tutorials *An Intruduction to R* sowie *R Data Import/Export*) und den Link zum Webauftritt von R.

# 5 Eine Beispielsitzung

## 5.1 Konventionen im Skript

Um die Lesbarkeit des Skriptes zu erhöhen, werden einige Konventionen zur Darstellung bestimmter Sachverhalte getroffen.

**input** Eingabe von Code: Zeile beginnt mit ">"

**output** Ausgabe des Programms: Zeile beginnt mit "[1]"

**unvollständige inputs** Sollte ein Ausdruck am Ende einer Zeile syntaktisch nicht vollständig sein, erscheint ein "+"

**Kommentare** Bestandteile der Eingabe, die von dem Programm nicht interpretiert werden sollen, ,z.B. Hinweise zu einer Berechnung, werden mit einer Raute gekennzeichnet (#).

```
> 1 + 1

[1] 2

> 1 +

+

> 100

[1] 101

> # Das ist ein Kommentar
```

**Pakete und Funktionen** Das Paket `car` zu dem Buch "An R and S-PLUS Companion to Applied Regression" von John Fox. Die Funktion `mean()` (*Mittelwert*).

## 5.2 R als Taschenrechner

Die Grundrechenoperationen werden mit den Operatoren: `+`, `-` ,`*` ,`/` durchgeführt. Zum Potenzieren wird `^` verwendet:

```
> 12+13

[1] 25

> 121-11

[1] 110

> 4*5

[1] 20

> 5/35

[1] 0.1428571

> 2^3

[1] 8

> 10*3+1 # Punkt- vor Strichrechnung

[1] 21

> sqrt(16) # einfache Quadratwurzel

[1] 4

> exp(2) # Exponentialfunktion

[1] 7.389056

> sin(15) # Trigonometrische Funktionen

[1] 0.6502878
```

### Dateneingabe

Es existieren zahlreiche Möglichkeiten, Daten in R zu erstellen. Der einfachste Weg, besteht im Erzeugen eines **Vektors**. Ein Vektor ist eine endliche Folge von einzelnen **Elementen**. Einzelnen Elemente können beispielsweise mit der Funktion `c()` (*concatenate*)einem Vektor zugewiesen werden. Auf alle Objekte in R kann anhand ihrer Namen zugegriffen werden. In dem folgenden Beispiel werden dem R-Objekt mit dem Namen *x.1* die Werte *1*, *2* und *3* zugewiesen.

```
> x.1 <- c(1,2,3)
```

Mit dem Aufruf von *x.1* wird der Vektor ausgegeben.

```
> x.1

[1] 1 2 3
```

Daten können auch aus anderen Anwendungen importiert werden. Allerdings sind für einige Funktionen und Vorgehensweisen, wie z.B. das Importieren von Daten aus SPSS-Dateien, zusätzliche Pakete notwendig. Diese werden mit der Funktion `library()` geladen. Außerdem wird mit der Funktion `setwd()` (*set working directory*) das aktuelle Arbeitsverzeichnis definiert, in dem sich die zu importierende Datei befinden muss.[1]

```
> setwd("D:/.../Daten")  # Definieren des Pfades zum Datensatz
> library(foreign)       # Laden des Paketes foreign
> spss.allison <- read.spss("allison.sav", use.value.labels=FALSE,
+ to.data.frame=TRUE)
> spss.allison           # Aufrufen des Objektes

   INCOME SCHOOL AGE SEX
1   48000     12  54   1
2   26000     12  28   1
3   26000      7  56   1
4   48000     14  47   1
5   34000     12  60   1
...
34   6000     12  19   0
35   2000     10  25   0
```

[1] Beispiel aus: Allison, P. D. (1999). Multiple regression: A primer. Pine Forge Press: Thousand Oaks.

Mit der generischen Funktion `summary()`, die auch auf Objekte zahlreicher anderer Klassen anwendbar ist, bekommt man einen Überblick über den gesamten data frame.

```
> summary(spss.allison)
```

```
     INCOME          SCHOOL           AGE              SEX
 Min.   : 2000   Min.   : 6.00   Min.   :19.00   Min.   :0.0000
 1st Qu.: 9500   1st Qu.:11.50   1st Qu.:32.00   1st Qu.:0.0000
 Median :21000   Median :12.00   Median :38.00   Median :0.0000
 Mean   :25200   Mean   :12.69   Mean   :41.77   Mean   :0.4286
 3rd Qu.:34000   3rd Qu.:14.00   3rd Qu.:54.00   3rd Qu.:1.0000
 Max.   :81000   Max.   :20.00   Max.   :76.00   Max.   :1.0000
```

## 5.3 R als Statistikprogramm: univariate Statistik

Zuerst werden 2 Vector erstellt[2]:

```
> Gewicht <- c(60, 72, 57, 90, 95, 72)

> Groesse <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
```

Jetzt können wir z.B. unkompliziert den Body Mass Index (BMI) berechnen:

```
> BMI <- Gewicht/ Groesse^2

> BMI

[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

Der Mittelwert ($\bar{x} = \sum x_i/n$) einer Variable lässt sich unter anderem mit dem Rückgriff auf die Funktionen `sum()` (*Summe der Elemente eines numerischen Vektors*) und `length()` (*Anzahl der Elemente in einem Vektor*) berechnen.

```
> sum(Gewicht)

[1] 446

> length(Gewicht)

[1] 6

> sum(Gewicht)/length(Gewicht)

[1] 74.33333
```

Die Schätzung der Standardabweichung (für Stichprobendaten) für die Zufallsvariable *Gewicht* $\sqrt{\sum(x_i - \bar{x})^2/(n-1)}$ ergibt sich aus:

```
> mittelwert <- sum(Gewicht)/length(Gewicht)

> Gewicht-mittelwert

[1] -14.333333  -2.333333 -17.333333  15.666667  20.666667
[6] -2.333333

> sqrt(sum((Gewicht - mittelwert)^2)/ (length(Gewicht)-1))

[1] 15.42293
```

[2]Beispiel aus Delgaard, P. (2002). Introductory statistics with R. Springer: New York.

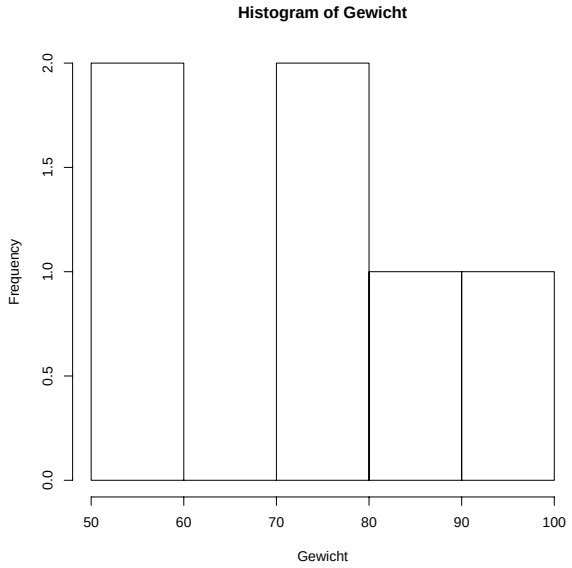


Abbildung 1: Histogramm der Variable Gewicht

Als Statistikprogramm verfügt R über feste Funktionen für Mittelwertberechnung und Ermittlung der Standardabweichung.

```
> mean(Gewicht)

[1] 74.33333

> sd(Gewicht)

[1] 15.42293
```

Für die deskriptive Datenanalyse und zum Sammeln von Informationen zu einzelnen Variablen kann man unter anderem mit der Funktion `hist()` ein Histogramm anfordern. Bei der Erzeugung von Grafiken wird innerhalb der Anwendung ein extra Fenster geöffnet.

```
> hist(Gewicht)
```

Aufgrund der ”dünnen” Datenlage ist das Histogramm nicht sonderlich aussagekräftig. Wir können mit der Funktion `rnorm()` eine normalverteilte Zufallsvariable mit 100 Fällen erzeugen und wiederum ein Histogramm anfordern.

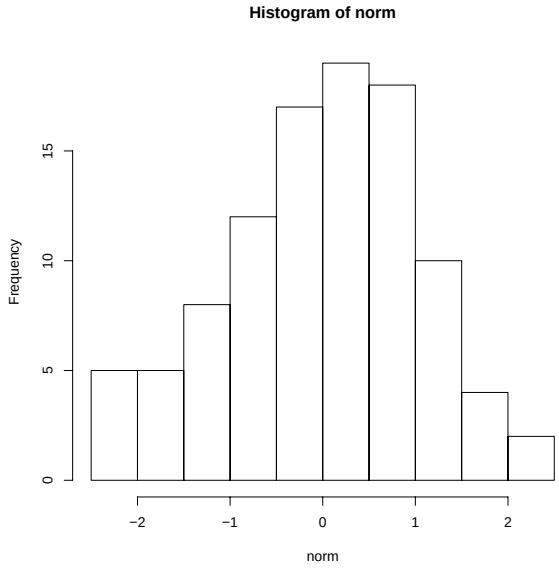


Abbildung 2: Histogramm der Zufallsvariable *norm*

```
> norm <- rnorm(100)
> hist(norm)
```

Es stehen zahlreiche andere Funktionen zur Verfügung. Unter anderem zur Erzeugung von einfachen Tabellen und der Boxplots:

```
> table(Gewicht)

57 60 72 90 95
 1  1  2  1  1

> boxplot(Gewicht)
```

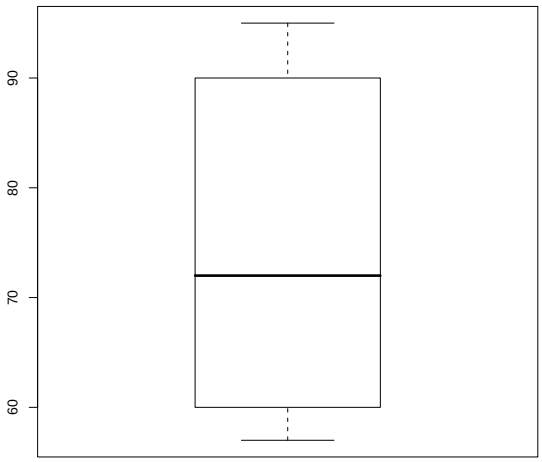


Abbildung 3: Boxplot der Zufallsvariable

# Praxis

- Legen Sie einen Vektor mit dem Namen *vector.1* bestehend aus 10 numerischen Elementen an.
- Informieren Sie sich über diesen Vektor mit Hilfe der Funktion `summary()`.
- Zeichnen Sie ein Histogramm für *vector.1*.
- Versuchen Sie mit dem Hilfesystem die Farbe der Balken des Histogramms auf blau zu setzen. Sie können zu einzelnen Funktionen zusätzliche Informationen auch über die Eingabe von *?name* erhalten. Dabei steht *name* für den Namen der Funktion (z.B. "?hist").
- Ermitteln Sie den höchsten Wert, den Niedrigsten, den Median, die Summe der enthaltenen Elemente sowie die Varianz von *vector.1*. Verwenden Sie dazu folgende Funktionen: `max()`, `min()`, `median()`, `sum()`, `var()`.
- Legen Sie einen weiteren Vektor mit dem Namen *vector.2* an.

- Informieren Sie sich darüber, wie viele Objekte Sie momentan in der Arbeitsumgebung haben(Funktion `ls()`).
- Löschen Sie das Objekt *vector.1* aus der Arbeitsumgebung. Verwenden Sie dafür die Funktion `rm()`. Informationen dazu finden Sie auch auf den Seiten 5 und 6 des Handbuchs "An Introduction to R".
- Wie lautet der Befehl, mit dem alle Objekte aus der Arbeitsumgebung gelöscht werden?

## 5.4 R als Statistikprogramm: multivariate Statistik

Gehen wir von der inhaltlich schwachen Annahme aus, dass die Größe einer Person sich positiv auf das Gewicht auswirkt. Ein Anstieg der unabhängige Variable Körpergröße bewirkt einen Anstieg der abhängigen Variable Gewicht.

Ein mögliches Verfahren zur Überprüfung der Aussage ist die einfache linerare Regression. Wir gehen weiterhin davon aus, dass alle Annahmen für die Anwendung des Verfahren erfüllt sind.

```
lm(Gewicht~Groesse)

Call:
lm(formula = Gewicht ~ Groesse)

Coefficients:
(Intercept)      Groesse
     -46.34        67.35
```

Der einfache output informiert über die beiden Parameter der Anpassungslinie in der Punktwolke auf der Basis der Kleinsten-Quadrat-Schätzmethode (KQS). Für weiterführende Informationen, kann man wieder die Funktion `summary()` verwenden.

```
summary(lm(Gewicht~Groesse))

Call:
lm(formula = Gewicht ~ Groesse)

Residuals:
      1       2       3       4       5       6
-11.527  -2.895  -7.792   8.370  24.147 -10.303

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -46.34     124.02  -0.374    0.728
Groesse        67.35      69.13   0.974    0.385


Residual standard error: 15.5 on 4 degrees of freedom Multiple
R-Squared: 0.1918,     Adjusted R-squared: -0.01027 F-statistic:
0.9492 on 1 and 4 DF,  p-value: 0.3851
```

Bei Fragen zur Modellspezifikation oder Interpretation der Ergebnisse können man z.B. auf die umfangreiche englischsprachige Hilfe zurückgreifen:

```
> help(lm)
> ?lm
```

## 5.5 R als Grafikprogramm

*R* besitzt umfangreiche Möglichkeiten zur Grafikerstellung. Neben den standardmäßig enthaltenen Optionen im Paket `base` (diese werden bei Programmstart automatisch geladen) existieren zahlreiche weitere Funktionen und Pakete.

Einfachste Plots werden mit der Funktion `plot()` erstellt:

```
> plot(Gewicht)
```

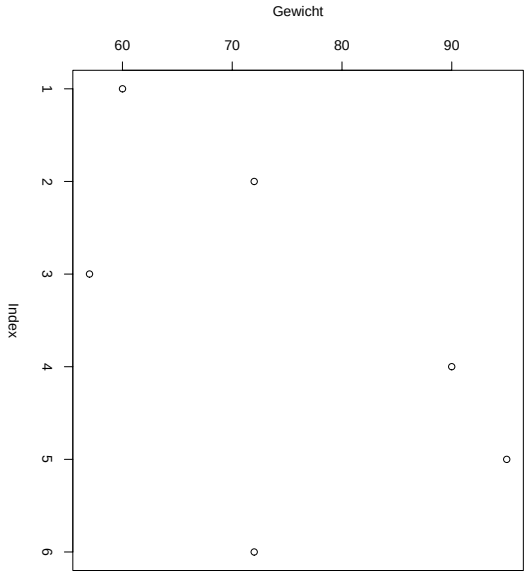


Abbildung 4: Punktwolke der Variable Gewicht

Einfache Streudiagramme können mit `plot(x, y)` erzeugt werden.

```
> plot(Groesse, Gewicht)
```

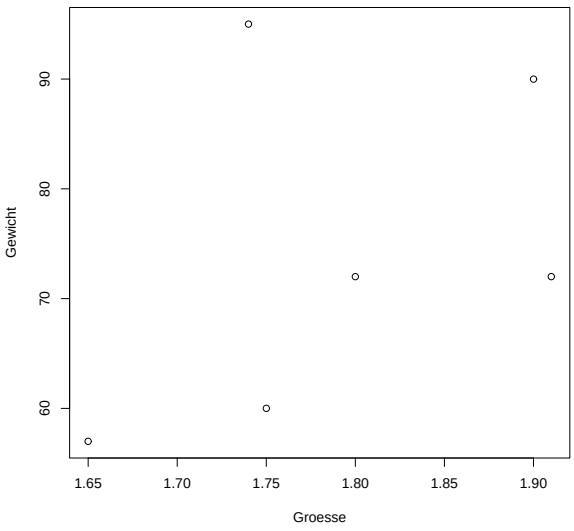


Abbildung 5: Streudiagramm der Variablen Groesse und Gewicht

Mit einfachen Modifikationen können weitere Informationen in die Grafik integriert werden:

```
> plot(Groesse, Gewicht, type = "p", col = "red", lwd=10,
+ main = "Streudiagramm Groesse gegen Gewicht", xlab="Groesse",
+ ylab="Gewicht")
```

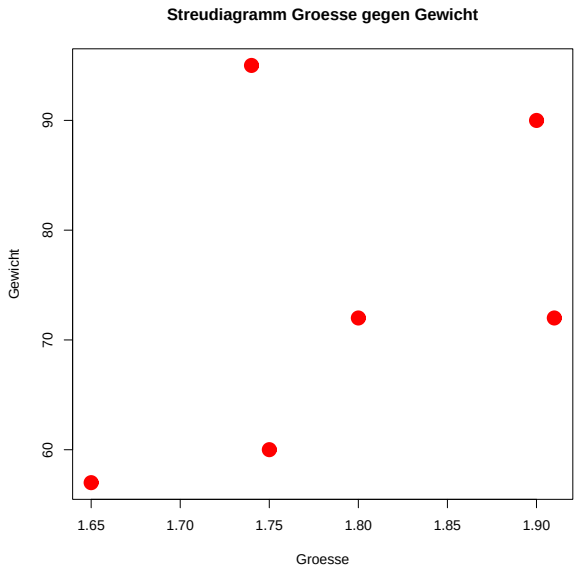


Abbildung 6: Erweitertes Streudiagramm

Ein sehr komplexes Beispiel steckt in der Webseite (www.r-project.org) von R.

## 5.6 R als Programmiersprache

Der grundsätzliche Aufbau einer Funktion sieht wie folgt aus:

```
fname<-function(Argumente)
{
Koerper der Funktion
return(Ergebnis)
}
```

Die Berechnung des Mittelwerts als Funktion könnte wie folgt aussehen:

```
> func.mw<-function(x){
+   mw<-sum(x)/length(x)
+   return(mw)
+   }
```

Aufrufen und Nutzen der Funktion erfolgt wie bereits durchgeführt:

```
func.mw(Gewicht)

74.33333
```

Der Vorteil dieser Funktionen ist unter anderem die Konservierbarkeit der Algorithmen und ständige Verfügbarkeit.

Wenn man der Frage nachgeht, wie oft die Zahl 6 gewürfelt wird, wenn man den Würfel 1000 mal wirft, können Sie z.B. auf folgende Funktion zurückgreifen. [3]

```
>  wuerfel <- function( N, Augenzahl)
+ {
+       # Generieren von N Würfen eines Würfels mit 6 Seiten:
+   x <-sample(1:6, N, replace = TRUE)
+       # Zählen, wie oft die ugenzahl "Augenzahl" vorkommt:
+       sum(x == Augenzahl)
+ }
```

Die Formel angewendet:

```
> wuerfel(1000,6)

[1] 179 # ist abhängig von den Zufallszahlen
```

[3] Formel entnommen und adaptiert bei Ligges, U. (2005). Programmieren mit R, S. 100

# Praxis

- Legen Sie 2 numerische Vektoren mit der gleichen Länge (Anzahl von Elementen) an.
- Unterstellen Sie eine Kausalbedingung und rechnen Sie eine einfache lineare Regression.
- Stellen Sie die beiden Variablen im einem Streudiagramm gegenüber.
- Setzen Sie ein Überschrift sowie Achsenbeschriftungen.

# 6 Die Website von R

### Search

Für spezifische Probleme lohnt sich die Suche in den Archiven der Mailingliste.

### Manuals

Hier findet man immer die aktuellsten Handbücher. Diese werden mit der Installation mitgeliefert.

### FAQs

Fragen können oft schon beim Lesen der FAQs beantwortet werden.

# 7 Ein effizienter Editor für R : Tinn-R

- ein speziell auf die Handhabung von R zugeschnittener Editor
- läuft nur unter Windows (9X/Me/2000/XP)
- in Delphi 5 geschrieben
- frei
- klein (2,2 MB)
- unter den Maßgaben der GPL zum Download angeboten
- Webseite zu Tinn http://tinn.solarvoid.com

## 7.1 Erweiterungen bei Tinn-R

- Erweiterungen, wie beispielsweise syntax highlighting der S sowie der R Sprache
- zusätzliches Menu und eine Symbolleiste bei Erkennen eines R -Gui
- Addons (Menu, Toolbar) interagieren mit R
- Senden von Code und ein direktes Steuern des Programms

## 7.2 Installation von Tinn-R unter Windows

Download der Software unter: SourceForge.

Installieren der ausführbaren Datei (Tinn-R_1.19.1.1_setup.exe, Version: 1.19.1.1) in ein beliebiges Verzeichnis.

## 7.3 Installieren der notwendigen Pakete

Für die Verwendung des gesamten Umfangs an Funktionen und Tools müssen noch einige Pakete installiert werden ( svIDE, svIO, svSocket, R2HTML). Das kann a) per Hand in R geschehen:

```
> library(svIDE)
> library(svIO)
> library(svSocket)
> library(R2HTML)
```

bzw. b) über ein fertiges Script:

```
> source('.../Tinn-R/custom/Rconfigure.r')
# source() ist eine Funktion zum Einlesen von Befehlen
```

oder c) über Tinn-R direkt erfolgen. Dabei kann zwischen einer temporären und einer permanenten Variante gewählt werden.

### 7.3.1 Für die aktuelle Sitzung

R –> Configure –> Temporary 

### 7.3.2 Permanent

R –> Configure –> Permanent

Beim Ausführen dieses Befehls wird eine Edition in der Datei *Rprofile.site* vorgenommen.