-Kurs für Forscher

Herzlich Willkommen!

Antony Unwin, Anatol Sargin, Alexander Pilhöfer
Lehrstuhl für Rechner-Orientierte Statistik und Datenanalyse
Universität Augsburg

Kurs Ziele

- Die statistische Sprache R (und ihre Literatur) vorzustellen.
- Die Eigenschaften von R zu erklären.
- Teilnehmern beizubringen, wie man mit R umgeht.
- Das Umfeld von R zu beschreiben:
  die Pakete,
  die Hilfe,
  die Webseiten,
  die Mailing Lists,
  und
  die Community.

Programm

- 9:00 - 9:15 Willkommen
- 9:15 - 10:00 -Kurs (Teil 1):
  Einführung, Datensatz laden, einfache Statistiken, Grafiken
- 10:00 - 10:30 Hands-on und Diskussion
- 10:30 - 11:00 *** Pause ***
- 11:00 - 11:45 -Kurs (Teil 2):
  Verteilungen, Tests und Modelle, Datenmanipulation
- 11:45 - 12:30 Hands-on und Diskussion
- 12:30 - 13:00 Wrap-up

Einleitende Worte

Was ist eigentlich R?

- R ist eine frei zur Verfügung stehende Umgebung zur Umsetzung statistischer Methoden und ist auch in besonderem Maße für graphische Darstellungen geeignet.

Welche Vorteile bietet R?

- Enormer Umfang an Möglichkeiten mit ständiger Erweiterung durch sog. Packages.
- Schnelle und professionelle Umsetzung von Grafiken.
- R als Sprache: leichter Zugang zu Datenmanipulationen und Methoden
- Und nicht zuletzt: R steht kostenlos zur Verfügung

Information

Woher bekomme ich ...

... die Software?

Die Software R selbst finden sie auf der R-Homepage:
(http://www.r-project.org/)
Unter Windows genügt es die *R-2.6.2-win32.exe* herunterzuladen und auszuführen.
Sie sollten über eine aktuelle Version von Java verfügen.

... die Erweiterungspakete?

Die packages können von der CRAN-R Webseite
(http://cran.r-project.org/) geladen werden oder über den Package Installer im Menü von JGR.

... Hilfe und Erklärungen?

Zur Orientierung bei der Suche nach bestimmten Inhalten können die auf der cran-R Homepage angebotenen Task Views nützlich sein.
Antworten auf Fragen zur Funktionsweise der Methoden gibt die umfangreiche R-Hilfe. Dort finden Sie auch Beispiele und Vignetten.

Und nicht zuletzt steht Ihnen der Lehrstuhl gerne bei Fragen und Problemen hilfreich zur Seite.

JGR – Java GUI for R

Grundsätzlich hat R eine eigene GUI mit der man problemlos arbeiten kann.

Am Lehrstuhl wurde darüber hinaus eine weitere Oberfläche entwickelt, die den Umgang mit R durch einige sehr nützliche Funktionen erleichtert:

 *JGR* (**J**ava **G**UI for **R**)

Diese Benutzeroberfläche steht ebenfalls kostenlos zur Verfügung und bietet unter anderem:

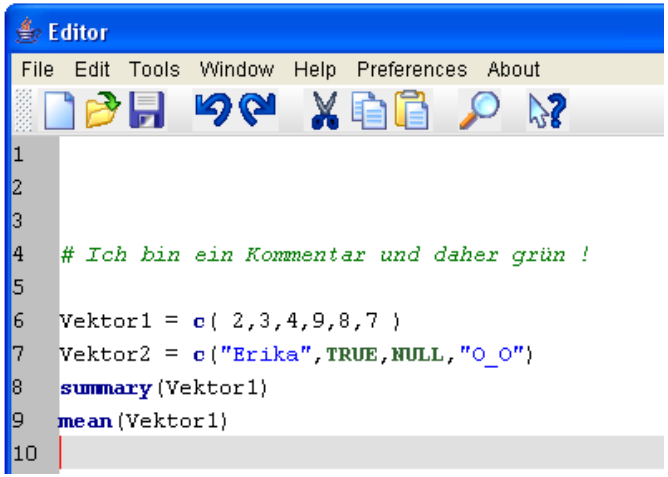
- Syntaxhervorhebung
- eingeblendete Hilfe zu Funktionen
- leichte Ausführbarkeit von Code
- einen „Objekt-Browser“ für Übersicht und schnellen Zugriff auf Objekte und Methoden
- und einiges mehr…

Sie finden JGR im Netz unter:

http://stats.math.uni-augsburg.de/software/JGR/

JGR – Java GUI for R

Der benutzte R-Code kann nicht nur in der Console eingegeben werden, sondern auch in einem Skript gespeichert und jederzeit wieder ausgeführt werden.



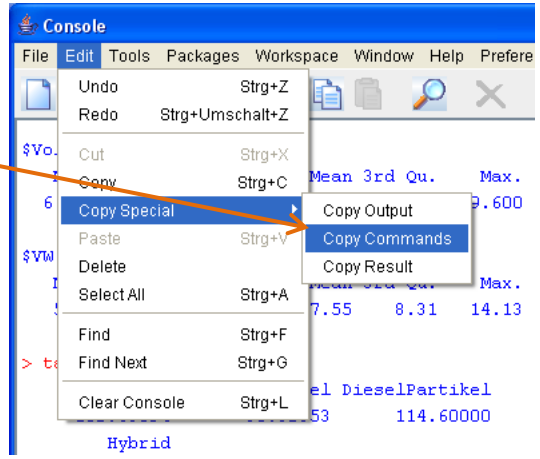
Nützlich ist hier die **Syntaxhervorhebung**:

grün	Kommentare
grün & fett	Wahrheitswerte, NULL
blau	Zeichenketten
blau & fett	R bekannte Namen von Funktionen od. Objekten

Der Code kann sehr einfach durch ***Markieren*** und ***Strg+R*** ausgeführt werden.

Sowohl der komplette Output als auch Commands und Results für sich können aus der Console heraus kopiert werden.

Es steht außerdem noch ein sehr nützlicher **Objekt-Browser** zur Verfügung.

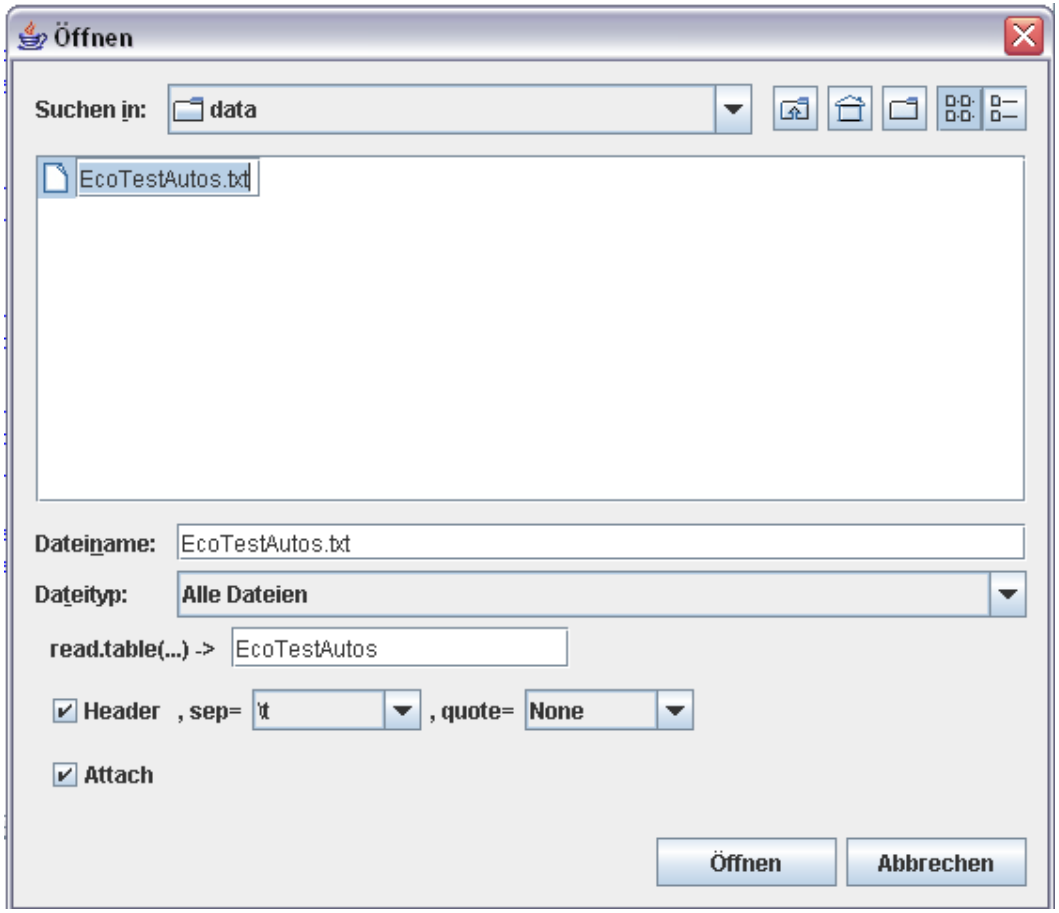


Laden eines Datensatzes

Wie lade ich einen Datensatz in R?

Der Befehl zum Laden eines Datensatzes lautet:

dataset <- read.table(„D:/…/filename.txt", header=T, sep="\t", quote="");



Alternativ kann auch über das Menü
File -> Load Data File
über einen Explorer geladen werden.

Einige Einstellungsmöglichkeiten:

header = T	Variablenbeschriftung
sep = ‚\t'	Trennungszeichen
quote = F	Texterkennungszeichen

…

Mit dem Befehl *attach(dataset)* können die einzelnen Variablen über ihren Namen direkt erreicht werden.

Struktur eines Datensatzes

Ein Datensatz ist in Zeilen (Fälle) und Spalten (Variablen) angeordnet.

In R können Objekte mit einem solchen Aufbau mit folgender Indizierung abgefragt werden:

Objektname[x,] für Zeile x

Objektname[,y] für Spalte y

Objektname[x,y] für ein einzelnes Element

Weiterhin kann auf die Variablen auch mittels ihrer Bezeichnung zugegriffen werden:
Datensatz$Variable1 oder nach vorangegangenem *attach(Datensatz)* auch nur *Variable1*

Beispiel: *Umfragedaten$Geschlecht*
Umfragedaten$Frage07

Auf die Anzahl von Zeilen/Spalten eines Objekts kann mit
ncol(Object) bzw. *nrow(Object)* zugegriffen werden.

Objekte

Objekte in R enthalten oft eine Vielzahl von strukturierten Informationen.
Der Zugriff erfolgt über die Kopplung des Objektnamens und des Namens der Unterstruktur mittels des *$*-Zeichens.
Beispiel:
Ist *lm1* ein Objekt eines linearen Modells, so kann auf die Residuen zugegriffen werden:
lm1$residuals für den gesamten Vektor und
lm1$residuals[i] für einen einzelnen Wert

```
> modell = lm(Y ~ X1*X2)
> modell$coefficients
 (Intercept)           X1           X2        X1:X2
6.804344864  1.066841389 -0.201406372  0.004736306
```

Welche Informationen ein Objekt birgt kann man mit Hilfe der Funktionen
attributes(Objekt) oder
str(Objekt) ausgeben lassen.

```
> attributes(modell)
$names
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

$class
[1] "lm"
```

Einige wichtige Statistiken

Eine kleine Übersicht über einige der wichtigsten Statistiken:

mean()
median()
min(), max()
var(),cov(),cor()
sd()
summary()
quantile()
length()
range()
Size()

```
> summary(X)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    659    1587    1910    1983    2148    5666
> var(X)
[1] 470063.4
> sd(X)
[1] 685.6117
> quantile(X,probs=c(0.2,0.5,0.8))
 20%  50%  80%
1493 1910 2204
```

(gerundete) Korrelationsmatrix:

```
> round(  cor(cbind(kW,CCM,FuelCons))  ,digits=2)
           kW  CCM FuelCons
kW       1.00 0.91     0.69
CCM      0.91 1.00     0.68
FuelCons 0.69 0.68     1.00
```

```
> range(X)
[1]  659 5666
```

Ähnlich nützlich wie die Funktion *summary()* ist auch die Funktion
tapply(Variable1, Variable2, function)
Sie gibt *function(Variable1)* für die einzelnen Niveaus von *Variable2* wieder.

```
> tapply(kW, Typ, median)
      Benzin       Diesel DieselPartikel          Gas       Hybrid
        92.0         84.0        105.0         80.5         61.0
```

Hilfe und Support

R verfügt über eine umfangreiche Hilfefunktion.
Der Aufruf erfolgt über das Menü *Help* oder über die Konsoleneingabe *?help.*
Es kann auch *?topic* für ein spezielles Topic eingegeben werden (z.B. *?summary*).

Die Hilfeseiten sind zumeist in folgendem **Aufbau** gehalten:

`Description`	Kurzbeschreibung der Methode
`Usage`	Syntax
`Arguments`	Beschreibung der Argumente
`Details`	diverse Zusatzinformationen
`Value`	Info zur Ausgabe/zum erzeugten Objekt
…	…
`See Also`	verwandte Themen
`References`	Autoren
`Examples`	Anwendungsbeispiele

Hilfe und Support

Beispiel: *?Normal*

Schlüsselwort des Hilfethemas mit Angabe des Pakets

Kurzbeschreibung

Syntax:
ohne Wert, vorne: muss übergeben werden
mit Wert: hat einen default-Wert
ohne Wert, hinten: nicht zwingend erforderlich

Arguments:
Erklärung der einzelnen Funktionsparameter

Normal {stats} R Documentation

The Normal Distribution

Description

Density, distribution function, quantile function and random generation for the normal distribution with mean equal to `mean` and standard deviation equal to `sd`.

Usage

```
dnorm(x, mean=0, sd=1, log = FALSE)
pnorm(q, mean=0, sd=1, lower.tail = TRUE, log.p = FALSE)
qnorm(p, mean=0, sd=1, lower.tail = TRUE, log.p = FALSE)
rnorm(n, mean=0, sd=1)
```

Arguments

`x,q`	vector of quantiles.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.
`mean`	vector of means.
`sd`	vector of standard deviations.
`log, log.p`	logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	logical; if TRUE (default), probabilities are $P[X <= x]$, otherwise, $P[X > x]$.

Grafiken

R bietet ebenfalls die Möglichkeit zur Erstellung professioneller Grafiken mit vielen individuellen Einstellungsmöglichkeiten.

Der **Standardbefehl *plot(Object)*** ist dabei für viele Objekttypen definiert und wird daher oft verwendet.

Weiterhin stehen natürlich eine Vielzahl anderer Befehle zur Verfügung.
Einige Beispiele:

hist()	Histogramm
boxplot()	Boxplot
barchart()	Balkendiagramm
pairs()	Matrix aus Streudiagrammen
parcoord()	Parallele Koordinaten Plot (Package: MASS)
densityplot() / histogram()	Dichteschätzer (Package: lattice)
ibar()/ihist()/ipcp/...	interaktive Grafiken (Package: iplots)
mosaicplot()	Mosaikplots

Grafiken

Grafiken bieten eine Reihe von Einstellungsmöglichkeiten.

Diese findet man entweder direkt in der Hilfe zum jeweiligen Grafikbefehl oder sie gehören einer zugrundeliegenden Familie an.

Für die meisten Befehle können die Einstellungen aus dem Grafikparameter-Set ***par*** eingesetzt werden. (sh. *?par*)

Einige der wichtigsten seien an dieser Stelle kurz erwähnt:

xlim/ylim	*=c(min,max)*	Achsenabschnitte
xaxp/yaxp	*=c(min,max,steps)*	Skalenbeschriftung
pch	*=Integer/Zeichen*	Punktsymbol
lwd	*=Double*	Linienstärke
main	*=Text*	Überschrift
xlab/ylab	*=Text*	Achsenbeschriftung

Grafiken

Es können natürlich auch diverse Grafiken kombiniert und ergänzt werden.

Beispielsweise können **zusätzliche Punkte und Linien** eingezeichnet werden:

points(newdata)	zeichnet weitere Punkte in den aktiven Plot
lines(data)	verbindet die Datenpunkte mit Linien
abline(a,b,v,h)	zeichnet Linien a + b*x ODER vertikale/horiz. Linien

-> Es können auch jeweils auch Grafikparameter gesetzt werden!

Ein **neues Grafikfenster** kann folgendermaßen geöffnet werden:
JavaGD(width,height)
JavaGD()
ohne JGR: *windows()*

Um **mehrere Grafiken** in ein Fenster zu setzen, ist der par-Befehl *mfrow() / mfcol()* nützlich:
par(mfrow=c(rows,cols))
plot(...)

Eine **Legende** lässt sich im Nachhinein hinzufügen:
legend(...) -> Die genaue Verwendung sehen wir anhand eines Beispiels klarer...

Verteilungen und Stichproben

Die Handhabung von Verteilungen, Dichten, Quantilsfunktionen und Zufallstichproben ist in R erfreulich einfacher Struktur:

Der Befehl besteht aus:
-> einem der **Buchstaben p,d,q,r** (Verteilung, Dichte, Quantilsfunktion und Stichprobe)
-> dem **Schlüsselwort der Verteilung** (zB. norm, binom, chisq,...)
Beispiele:

pnorm(q, mean=0, sd=1)	Werte der (Standard-) Normalverteilung für die Quantile q
rbinom(n=4,size=1,prob=0.5)	4 “Münzwürfe”

Schlüsselwort	Verteilung
norm	Normal
lnorm	log-Normal
exp	Exponatial
gamma	Gamma
binom	Binomial
nbinom	Neg. Binomial
pois	Poisson
chisq	Chi²

Tests, Modelle und mehr...

Als wichtiges Mittel der statistischen Datenanalyse seien an dieser Stelle einige Tests sowie Modell- und Analyseformen erwähnt:

t-Test	t.test()
Chi²-Test	chisq.test()
Verteilungstests	ks.test()
Rangsummentests	wilcox.test(), kruskal.test(), friedman.test()
Signed-Rank Tests	
Lineare Regression	lm(), glm()
Logistische Regression	
Log-Lineare Modelle	
Neuronale Netze	nnet()
ANOVA/MANOVA	aov(),anova(),manova()
Faktorenanalyse	factanal()
Hauptkomponentenanalyse	princomp(),prcomp()
Clusterverfahren	Mclust(), kmeans(), fanny(), hclust(), ...
Baummodelle	rpart(),

Vektoren und Matrizen

Zwar bietet R auch die Möglichkeit der Benutzung von Schleifen, jedoch liegt eine weitere Stärke von R in seiner vektororientierten Rechenweise.

Vektoren werden mittels der Konkatenierungsfunktion $c(x_1,....,x_n)$ zusammengefügt.

Die Benutzung wird anhand einiger Beispiele am schnellsten klar:

```
> vektor1 = c(1,2,3,4,5)
> vektor2 = c(2,4,6,8,10)
>
> vektor12 = c(vektor1,vektor2)
> differenz = vektor2-vektor1
> verhaeltnis = vektor2/vektor1
> ################################
> vektor1
[1] 1 2 3 4 5
> vektor2
[1]  2  4  6  8 10
> vektor12
 [1]  1  2  3  4  5  2  4  6  8 10
> differenz
[1] 1 2 3 4 5
> verhaeltnis
[1] 2 2 2 2 2
> differenz^2
[1]  1  4  9 16 25
>
```

Neben einzelnen Werten können auch Strings oder Vektoren* verknüpft werden.

Wichtig: die Standard-Rechenoperationen werden elementweise berechnet (auch für Matrizen)!

Zugriff auf ein einzelnes Element hat man über *vektor[x], z.B.*

```
> vektor2[4]
[1] 8
```

(*Für andere Objekte können Listen verwendet werden!)

Vektoren und Matrizen

```
> # eine Beispielmatrix:
> M
     [,1] [,2] [,3]
[1,]    4    1    1
[2,]    7    4    1
[3,]    7    7    4
> # transponierte Matrix:
> t(M)
     [,1] [,2] [,3]
[1,]    4    7    7
[2,]    1    4    7
[3,]    1    1    4
> # obere rechte Matrix:
> M[1:2,2:3]
     [,1] [,2]
[1,]    1    1
[2,]    4    1
> # die Inverse Matrix:
> round(  ginv(M) ,digits=1)
     [,1] [,2] [,3]
[1,]  0.3  0.1 -0.1
[2,] -0.6  0.2  0.1
[3,]  0.6 -0.6  0.3
> # Operation nur elementweise!
> round(  ginv(M)*M     ,digits=1)
     [,1] [,2] [,3]
[1,]  1.0  0.1 -0.1
[2,] -4.1  1.0  0.1
[3,]  4.1 -4.1  1.0
> # echte Matrix-Multiplikation:
> round(  ginv(M)%*%M   ,digits=1)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Der **Zugriff** auf Matrizen ist in der Form *Matrix[x,y]* möglich.

Wie bereits erwähnt werden Operation i.d.R. **elementweise** ausgeführt.
Für Matrixmultiplikation kann der *%*%*-Operator verwendet werden.

Das **Inverse** einer Matrix: *ginv(Matrix)*
Die **transponierte** Matrix: *t(Matrix)*

Ein weiterer nützlicher Operator ist *%in%* zur Prüfung ob ein Element in einem Objekt (Vektor) enthalten ist:

```
> daten = c("Anna","Kaspar","Theobald","unknown")
> "Anna" %in% daten
[1] TRUE
```

Vektoren und Matrizen

Zwei **spezielle Funktionen** zu Erzeugung von Vektoren:

rep(x,times) erzeugt einen Vektor mit *times* Wiederholungen von *x*
seq(from,to,by) erzeugt einen Vektor mit Werten von *from* bis *to* in Schritten der Größe *by*

```
> rep(4.3,5)
[1] 4.3 4.3 4.3 4.3 4.3
> rep( c("links","rechts"), c(3,3))
[1] "links"  "links"  "links"  "rechts" "rechts" "rechts"
> seq(1,7,0.5)
 [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0
```

So wie Vektoren aus Werten mit *c(..)* zusammengefügt werden können, gibt es auch die Möglichkeit Vektoren und Matrizen aneinanderzuhängen.

cbind(x1,...,xn) fügt *x1,...,xn* als Spalten zusammen

rbind(x1,...,xn) fügt *x1,...,xn* als Zeilen zusammen

```
> a1 = c( 1,3,5 )
> a2 = c( 2,4,6 )
> cbind(a1,a2)
     a1 a2
[1,]  1  2
[2,]  3  4
[3,]  5  6
> rbind(a1,a2)
   [,1] [,2] [,3]
a1    1    3    5
a2    2    4    6
```

Klassen

Alle Objekte in R gehören unterschiedlichen Klassen an.
Variablen eines Datensatzes (Klasse: *data.frame*) können folgenden **Klassen** angehören:

factor	Faktoren / Kategorien
integer	ganze Zahlen
double	stetige Zahlenwerte
logical	Wahrheitswerte

Variablen des Typs *factor* haben unterschiedlich viele *levels*.

levels(Variable) gibt die Ausprägungen selbst und
nlevels(Variable) deren Anzahl wieder.

```
> levels(Y)
[1] "Mann"          "Frau"          "unbekannt"     "Hermaphrodit"
> nlevels(Y)
[1] 4
```

Mit *as.<class>(Objekt)* kann ein Objekt in den entsprechenden Typ umgewandelt werden.

```
> X = as.integer(Y)
> table(X,Y)
   Y
X   Mann Frau unbekannt Hermaphrodit
  1   35    0         0            0
  2    0   39         0            0
  3    0    0        36            0
  4    0    0         0           20
> Z = as.factor(X)
> levels(Z)
[1] "1" "2" "3" "4"
```

Mit *is.<class>(Objekt)* wird abgefragt, ob das Objekt dem Typ angehört

```
> is.integer(Z)
[1] FALSE
```

class(Object) gibt die Klasse des Objekts aus.

```
> class(Z)
[1] "factor"
```

Datenimport

Ihr Datensatz liegt in einem anderen Dateiformat vor?
Zwei Möglichkeiten:

- Es ist in R möglich, nahezu alle anderen Dateiformate zu importieren.
  Beispiele:

read.spss()	liest Dateien aus SPSS ein
read.ssd()	liest Dateien aus SAS mittels read.xport() ein
odbcConnectExcel2007()	ermöglicht den Zugriff auf Excel-Spredsheets (package: RODBC)
read.xls()	liest Excel-Dateien (package: xlsReadWrite)

- Die zweite Möglichkeit: fast jede Software bietet die Möglichkeit des Datenexports in Textformat.
  Textdateien lassen sich mit *read.table()* vergleichsweise unkompliziert einlesen!

Trotzdem gibt es natürlich immer wieder Probleme (z.B. mit der Formatierung), bei denen wir Ihnen ggf. gerne zur Seite stehen!

Résumé

Hier seien noch einmal einige Vorteile und Eigenschaften von R zusammengefasst:

Der Benutzer kann vielerlei Arten von **Datensatzformaten** problemlos einlesen und für seine Analysen auf ein sehr umfangreiches „Arsenal“ an **Methoden und Paketen** einer großen internationalen Community zugreifen.

Die **Erstellung von Grafiken** gelingt durch eine **Vielzahl von Befehlen** und vor allem **Einstellungsparametern**.

Eine große Stärke von R ist die **Benutzung von Objekten**, seine **vektororientierte Rechenweise** und die benutzerfreundliche, **„sprachenähnliche“ Syntax**:

Sie ermöglichen lesbaren und **zugänglichen Code**, **Datenmanipulationen** und **Transformationen** und wahren auch bei komplexen Analysen die **Übersichtlichkeit**.

Die Bearbeitung und Wiederverwendung von Syntax gelingt i.d.R. sehr gut

Dem Benutzer wird mit der **umfangreichen Hilfefunktion** incl. Beispielen , den **Community-Foren** und **Webseiten** gute Unterstützung in seiner Arbeit geboten.