Dokumentation von R-Ergebnissen

File: ErgebnisSpeicherung.rev
in: /home/wiwi/pwolf/lehre/rkurs/documentation

13. März 2008

Inhalt

1 Einleitung

Nach einer Arbeit möchte jeder das Ergebnis mit nach Hause tragen. Das gilt insbesondere auch für Datenanalysen. Im Folgenden werden verschiedene Möglichkeiten aufgelistet, wie R-Ergebnisse auf das Papier bzw. in eine Datei gelangen können. Damit lassen sie sich auch archivieren und können zu einem späteren Zeitpunkt wieder verwendet werden. Es werden diverse Situationen unterschieden, die sich in der Vorgehensweise wie auch im Ergebnis unterscheiden.

2 Export aus der Console

Viele arbeiten lange in einem R-Console-Fenster und wollen nach der Zeit des Schaffens die wohl verdienten Früchte ernten und Ergebnisse dauerhaft sichern. Für solche Exporte fällt uns sofort die bewährte Cut-and-Paste-Technik ein, die manche unter dem Namen *Kopieren und Einfügen* kennen. Diese erfordert jedoch entsprechende Fähigkeiten des Fensters, in dem der R-Prozess läuft. Unter Windows arbeiten die meisten Anwender mit der *Anwendung* R-GUI, die Cut-and-Paste-Operationen erlaubt. Übrigens steht *Gui* hier für *graphical user interface*; dieses deutet darauf hin, dass der Anwender die R-Maschine aus einem graphischen Fenster mit Knöpfen und Menüs bedienen kann. Die vielen unbekannte rustikalere Alternative `Rterm` lässt dagegen Cut-and-Paste nicht zu. Im Folgenden befassen wir uns etwas mit Cut-and-Paste. Danach werfen wir einen kurzen Blick auf einige Gui-Angebote.

2.1 Cut-and-Paste zum Ergebnis-Export

Cut-and-Paste ist die erste Antwort des kleinen Mannes um Daten zwischen Anwendungen auszutauschen. Vorteil: bestens bekannt, sie ist schnell umgesetzt und verschiedene Ziele sind wählbar. Nachteil: Es muss eine Ziel-Anwendung geöffnet sein, bei wiederholtem Hin und Her oder bei der Modifikation von Ergebnissen ist dieses Vorgehen unbefriedigend.

Übung 1: Lassen Sie sich die Werte des Datensatzes `quakes` ausgeben und übertragen Sie die Ausgabe in ein Text-Dokument.

Übung 2: Definieren Sie

```
mat<-cbind(state.name,x=state.center$x,y=state.center$y).
```

Geben Sie die ersten fünf Zeilen von `mat` aus und übertragen Sie die Ausgabe in ein Text-Dokument. Verändern Sie die übertragenen Zeichen in dem gewählten Anwendungsprogramm so, dass das Ergebnis schön aussieht und die Dezimalpunkte untereinander stehen.

1 ⟨* 1⟩ ≡

```
mat<-cbind(state.name,x=state.center$x,y=state.center$y)
h<-mat[1:5,]
```

Output:

```
Wed Mar 12 14:18:40 2008
     state.name   x          y
[1,] "Alabama"    "-86.7509" "32.5901"
[2,] "Alaska"     "-127.25"  "49.25"
[3,] "Arizona"    "-111.625" "34.2192"
[4,] "Arkansas"   "-92.2992" "34.7336"
[5,] "California" "-119.773" "36.5341"
```

Tipp: Manchmal ist es leichter, den R-Output zu beeinflussen als später die kopierten Zeichen zu verbessern.

Wie entfernt man die Anführungsstriche? Ordne das Objekt der Klasse `noquote` zu.

2 ⟨* 1⟩+ ≡
```
h<-noquote(h)
```
Output:

```
Wed Mar 12 14:18:49 2008
     state.name x        y
[1,] Alabama    -86.7509 32.5901
[2,] Alaska     -127.25  49.25
[3,] Arizona    -111.625 34.2192
[4,] Arkansas   -92.2992 34.7336
[5,] California -119.773 36.5341
```

Wie entfernt man die Indexklammern?

3 ⟨* 1⟩+ ≡
```
h<-noquote(h)
rownames(h)<-rep("",nrow(h)); h
```

Output:

```
Wed Mar 12 14:19:17 2008
 state.name x        y
 Alabama    -86.7509 32.5901
 Alaska     -127.25  49.25
 Arizona    -111.625 34.2192
 Arkansas   -92.2992 34.7336
 California -119.773 36.5341
```

Wie können wir die Zahlen ausrichten?

4 ⟨* 1⟩+ ≡
```
h<-cbind(state.name,
         x=format(state.center$x),
         y=format(state.center$y))[1:5,]
rownames(h)<-rep("",nrow(h))
noquote(h)
```

Output:

```
Wed Mar 12 14:20:41 2008
 state.name x         y
 Alabama     -86.7509 32.5901
 Alaska     -127.2500 49.2500
 Arizona    -111.6250 34.2192
 Arkansas    -92.2992 34.7336
 California -119.7730 36.5341
```

Übung 3: Inwiefern unterscheidet sich die letzte Ausgabe von der von `data.frame(h)`?

5 ⟨* 1⟩+ ≡
```
data.frame(h)
```

2.2 R-Gui unter Windows

Das R-Gui-Fenster unter Windows erweckt mit seinen Menüs und Knöpfen den Eindruck einer modernen Oberfläche. Grundsätzlich werden in diesem Console-Fenster Anweisungen unten angefügt. Die Ergebnisse erscheinen sukzessive darunter. Markierte Einträge der Console können per Cut-and-Paste in eine andere Anwendung übertragen werden. Für den Export enthält das Datei-Menü aber auch einen Punkt, mit dem sich markierte Texte drucken oder als Datei speichern lassen.

Gegenüber `Rterm` bietet das R-Gui eine Erleichterung beim Umgang von Befehlen. Denn es lassen sich in einem einfachen Skript-Editor Anweisungen bearbeiten, durch Markieren oder Cursor-Position auswählen und ausführen. Kommandos und Ergebnisse werden zusammen in der Console wie gewohnt eingeblendet. Erarbeitete Skripte lassen sich speichern und wieder laden. Zusätzlich kann die Historie der R-Anweisungen gesichert (und wieder in ein Skript-Fenster geladen) werden.

2.3 R-Commander

Gerade für den Anfänger steht mit dem R-Commander[1] ein nettes Werkzeug bereit, das viel mehr Komfort als das normale Console-Fenster bietet. In einem eigenen Fenster lassen sich die gewünschten R-Anweisungen zusammenbasteln. Für diese Tätigkeiten erleichtern viele Menüs mit hinterlegten Operationen die Arbeit. Per Klick werden die markierten Anweisungen ausgeführt. Die Berechnungsaufträge und die zugehörigen Ergebnisse erscheinen im einem zweiten Fenster. Dieses Ausgabefenster zeigt Anweisungen und Ergebnisse, wie wir es von der Console her kennen. Jedoch können wir ohne Problem dort weitere textliche Bemerkungen einfügen.

Sowohl der Inhalt des Anweisungsfensters (das aktuelle Skript) als auch der des Ausgabefensters lassen sich als Datei speichern und können per Editor später verändert und weiter verarbeitet werden. Wir können damit unsere Analysen mit Hilfe des R-Commanders durchführen, das Protokoll abspeichern und Ausgaben mit ergänzten Kommentaren dauerhaft festhalten.

Es lässt sich zusammenfassen, dass Gui-Angebote bei der Konstruktion von Anweisungen helfen, die Verwaltung und schöne Dokumentationen von Ergebnissen stehen jedoch bei ihnen nicht im Vordergrund.

Übung 4: Berechnen Sie zu dem Datensatz `quakes` zusammenfassende Statistiken und ermitteln Sie die fünf stärksten Beben heraus. Verwenden Sie dazu zunächst das Windows-R-Gui und dann den R-Commander. Übertragen Sie jeweils die Ergebnisse in einen Text-File.

[1] Der R-Commander residiert im Paket `Rcmdr`. Er (`Commander()`) wird beim Laden des Pakets automatisch gestartet.

3 RELAX-Techniken für die interaktive Arbeit

Für die interaktive Durchführung von Analysen wird an dieser Stelle das Werkzeug `relax` empfohlen. Denn es ist genau für den Zweck der Ergebnisprotokollierung entworfen worden. Hierbei wird unterstellt, dass sich folgende Schritte wiederholen:

1. Idee formulieren und in R-Anweisungen übersetzen
2. Anweisungen ausführen lassen
3. Ergebnisse würdigen und Ergebnisse ggf. in das Dokument übernehmen

Wie kann man sich das praktisch vorstellen? Ist das Paket `relax` geladen, öffnet der Aufruf `relax()` eine neue Oberfläche mit Menüs, Knöpfen sowie oben einem Arbeits- und unten einem Ausgabe-Fenster. Im Arbeitsfenster entsteht im Laufe der Zeit das Dokument mit den Gedanken, Anweisungen und Ergebnissen. In der Regel bereitet man einen neuen Arbeitsschritt durch Betätigung des Knopfes PlanRCode vor. Dann sind im Arbeitsfenster R-Anweisungen einzutippen, ein Klick auf den Knopf EvalRCode löst die Berechnungen aus und ein Insert-Klick überträgt die Resultate ins Arbeitsfenster bzw. ins Dokument.

Für die Abgrenzung von Texten und Anweisungen gibt es eine einfache Regel: Klartext wird immer mit @ eingeleitet, wogegen Anweisungen stets auf Zeilen mit dem Musters `<<...>>=` folgen.

Dieses wollen wir an dem Beispiel der Berechnung von Statistiken zum Datensatz `quakes` vorführen. Was ist zu tun um den folgenden Output zu bekommen?

```
Wed Mar 12 17:54:10 2008
      lat              long           depth           mag
 Min.   :-38.59   Min.   :165.7   Min.   : 40.0   Min.   :4.00
 1st Qu.:-23.47   1st Qu.:179.6   1st Qu.: 99.0   1st Qu.:4.30
 Median :-20.30   Median :181.4   Median :247.0   Median :4.60
 Mean   :-20.64   Mean   :179.5   Mean   :311.4   Mean   :4.62
 3rd Qu.:-17.64   3rd Qu.:183.2   3rd Qu.:543.0   3rd Qu.:4.90
 Max.   :-10.72   Max.   :188.1   Max.   :680.0   Max.   :6.40
    stations
 Min.   : 10.00
 1st Qu.: 18.00
 Median : 27.00
 Mean   : 33.42
 3rd Qu.: 42.00
 Max.   :132.00
```

3.1 PlanRCode und Eingabe per Hand:

Drücken wir zuerst einmal den Knopf PlanRCode. Hierdurch erhalten wir einen leeren Text- und einen leeren Code-Bereich. Gedanken und Anweisungen tragen wir per Hand in den Text-Bereich, zum Beispiel: `Berechnung einiger Statistiken zum Datensatz ...` Als R-Anweisung interessiert uns der Einsatz von `summary` auf den Datensatz: `summary(quakes)`. Dann wird im Arbeitsfenster so etwas stehen wie:

```
% New Report:  Wed Mar 12 17:40:19 2008
@
Berechnung einiger Statistiken zum Datensatz quakes
...
<<*>>=
summary(quakes)
```

3.2 Aktivierung per EvalRCode

Nun wird EvalRCode betätigt und es erscheint im Outputfenster:

```
Wed Mar 12 17:27:11 2008
lat              long             depth            mag
Min.   :-38.59   Min.   :165.7   Min.   : 40.0   Min.   :4.00
1st Qu.:-23.47   1st Qu.:179.6   1st Qu.: 99.0   1st Qu.:4.30
Median :-20.30   Median :181.4   Median :247.0   Median :4.60
Mean   :-20.64   Mean   :179.5   Mean   :311.4   Mean   :4.62
3rd Qu.:-17.64   3rd Qu.:183.2   3rd Qu.:543.0   3rd Qu.:4.90
Max.   :-10.72   Max.   :188.1   Max.   :680.0   Max.   :6.40
stations
Min.   : 10.00
1st Qu.: 18.00
Median : 27.00
Mean   : 33.42
3rd Qu.: 42.00
Max.   :132.00
```

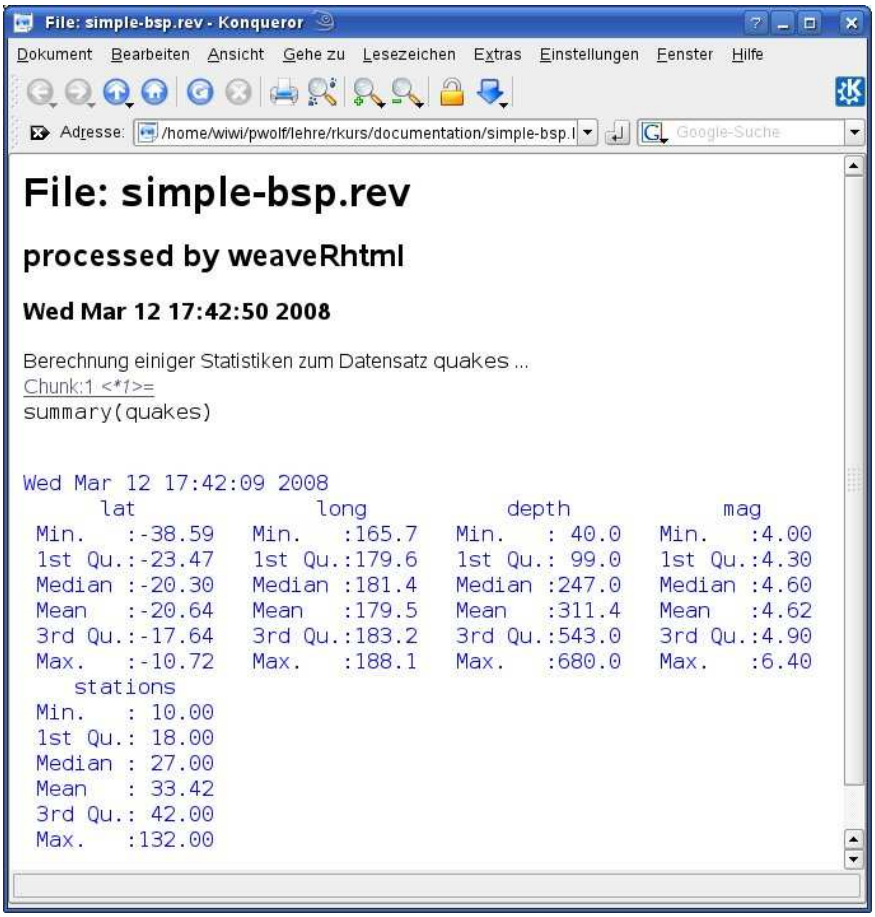
3.3 Ergebnisintegration per Insert

Insert verschiebt die Ausgaben ins Arbeitsfenster. Nun wird es den folgenden Inhalt haben:

```
% New Report:  Wed Mar 12 17:40:19 2008 @
Berechnung einiger Statistiken zum Datensatz quakes ...
«*»=
summary(quakes)
@
output-start
Wed Mar 12 17:27:11 2008
lat              long             depth            mag
Min.   :-38.59   Min.   :165.7   Min.   : 40.0   Min.   :4.00
1st Qu.:-23.47   1st Qu.:179.6   1st Qu.: 99.0   1st Qu.:4.30
Median :-20.30   Median :181.4   Median :247.0   Median :4.60
Mean   :-20.64   Mean   :179.5   Mean   :311.4   Mean   :4.62
3rd Qu.:-17.64   3rd Qu.:183.2   3rd Qu.:543.0   3rd Qu.:4.90
Max.   :-10.72   Max.   :188.1   Max.   :680.0   Max.   :6.40
stations
Min.   : 10.00
1st Qu.: 18.00
Median : 27.00
Mean   : 33.42
3rd Qu.: 42.00
Max.   :132.00
output-end
```

3.4 Speicherung als Html-File mit `SaveHtml`

Mit der Wahl `SaveHtml` aus dem `File`-Menü wird der Report gespeichert, und wir erhalten als Html-File:

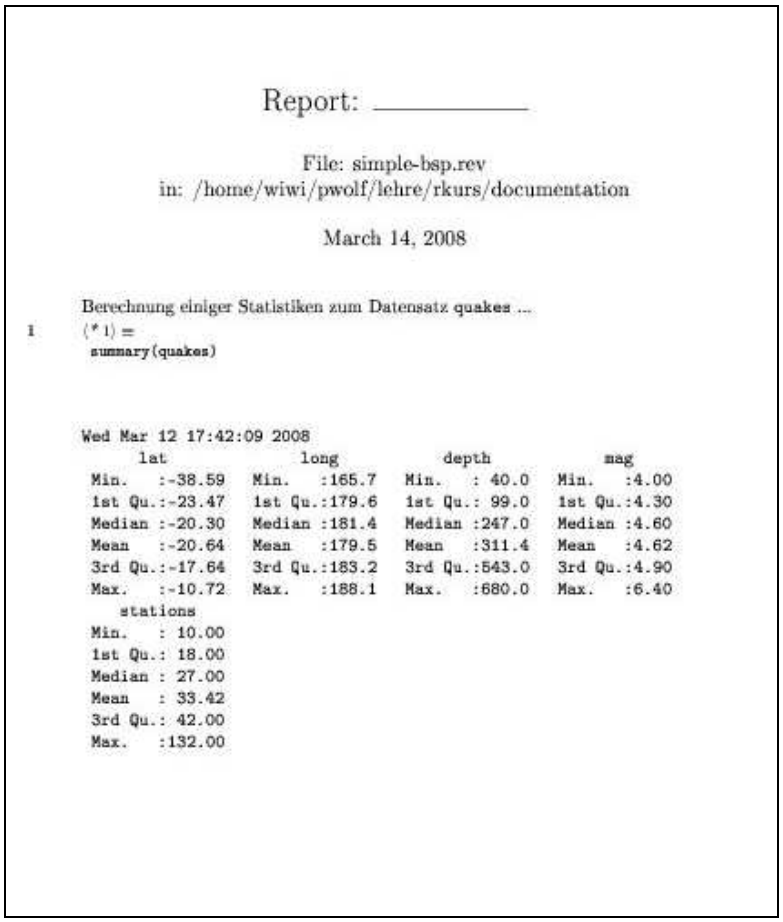


3.5 toLatex – für LaTeX-Freunde

Wer LaTeX beherrscht und installiert hat, kann statt des Html-Weges auch den LaTeX-Ausgang wählen. Hierzu ist natürlich eine Präambel erforderlich, wie sie mittels LaTeX.head aus dem Menü `Wizardry` vorgeschlagen wird.

Die Formatierung wird unkompliziert mit `ProcessReport` aus dem `Wizardry`-Menü angestoßen.

Mit `ViewReport` aus selbigem Menü können wir uns das Ergebnis anschauen:



Report: _____

File: simple-bsp.rev
in: /home/wiwi/pwolf/lehre/rkurs/documentation

March 14, 2008

Berechnung einiger Statistiken zum Datensatz **quakes** ...

```
1   (* 1) =
    summary(quakes)


Wed Mar 12 17:42:09 2008
      lat              long           depth           mag
 Min.   :-38.59   Min.   :165.7   Min.   : 40.0   Min.   :4.00
 1st Qu.:-23.47   1st Qu.:179.6   1st Qu.: 99.0   1st Qu.:4.30
 Median :-20.30   Median :181.4   Median :247.0   Median :4.60
 Mean   :-20.64   Mean   :179.5   Mean   :311.4   Mean   :4.62
 3rd Qu.:-17.64   3rd Qu.:183.2   3rd Qu.:543.0   3rd Qu.:4.90
 Max.   :-10.72   Max.   :188.1   Max.   :680.0   Max.   :6.40
    stations
 Min.   : 10.00
 1st Qu.: 18.00
 Median : 27.00
 Mean   : 33.42
 3rd Qu.: 42.00
 Max.   :132.00
```

3.6 LaTeX und `xtable`

Vielen TeXnikern wird die Form der Ausgabe noch nicht schmecken. Deshalb ist eine Verschönerung unter Verwendung von `xtable` angeraten. Die Funktion `xtable` befindet sich im Paket `xtable`, das hierfür installiert sein muss.

6 ⟨* 1⟩+ ≡

```
library(xtable,lib.loc="/home/wiwi/pwolf/lib")
xtable(summary(quakes))
```

Mit diesen wenigen Zeilen erhalten wir den LaTeX-Code der gewünschten Tabelle, falls `xtable` installiert ist:

```
\begin{table}[ht]
\begin{center}
\begin{tabular}{rllllll}
  \hline
 &      lat &      long &     depth &      mag &    stations \\
  \hline
  1 & Min.   :-38.59   & Min.   :165.7   & Min.   : 40.0   & Min.   :4.00   & Min.   : 10.00   \\
  2 & 1st Qu.:-23.47   & 1st Qu.:179.6   & 1st Qu.: 99.0   & 1st Qu.:4.30   & 1st Qu.: 18.00   \\
  3 & Median :-20.30   & Median :181.4   & Median :247.0   & Median :4.60   & Median : 27.00   \\
  4 & Mean   :-20.64   & Mean   :179.5   & Mean   :311.4   & Mean   :4.62   & Mean   : 33.42   \\
  5 & 3rd Qu.:-17.64   & 3rd Qu.:183.2   & 3rd Qu.:543.0   & 3rd Qu.:4.90   & 3rd Qu.: 42.00   \\
  6 & Max.   :-10.72   & Max.   :188.1   & Max.   :680.0   & Max.   :6.40   & Max.   :132.00   \\
   \hline
\end{tabular}
\end{center}
\end{table}
```

Diesen Output bewegen wir mit dem Insert-Knopf in das Dokument. Die Formatierung mit LaTeX führt zu folgender gesetzten Tabelle:

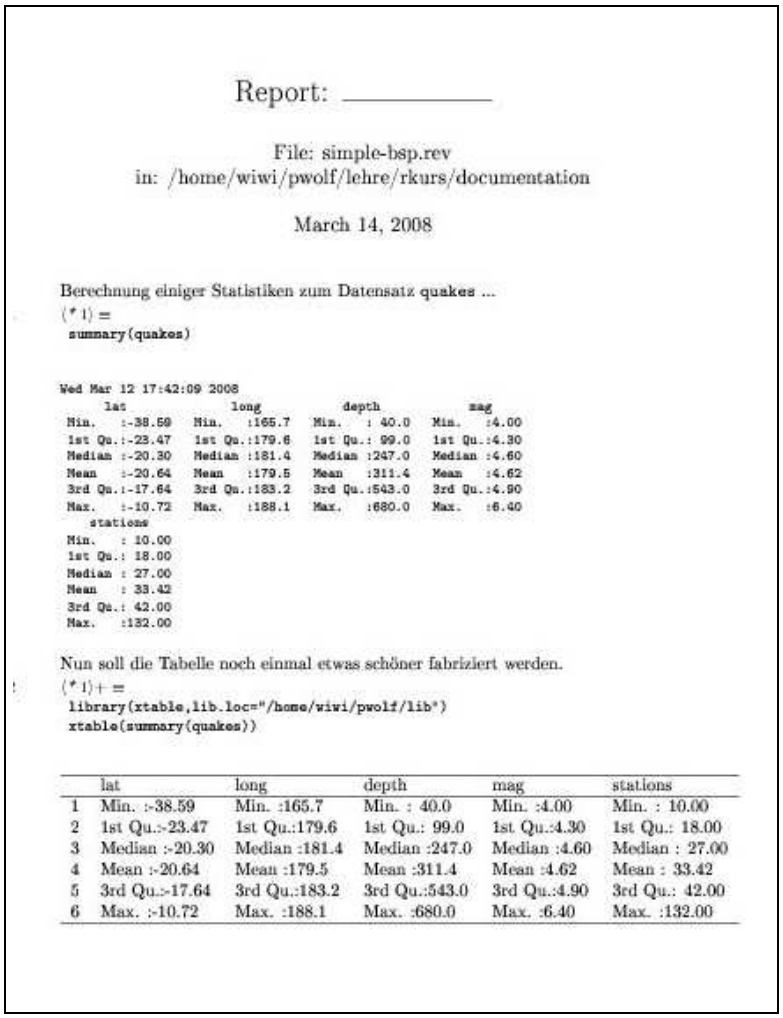
```
Wed Mar 12 18:02:50 2008
```

	lat	long	depth	mag	stations
1	Min. :-38.59	Min. :165.7	Min. : 40.0	Min. :4.00	Min. : 10.00
2	1st Qu.:-23.47	1st Qu.:179.6	1st Qu.: 99.0	1st Qu.:4.30	1st Qu.: 18.00
3	Median :-20.30	Median :181.4	Median :247.0	Median :4.60	Median : 27.00
4	Mean :-20.64	Mean :179.5	Mean :311.4	Mean :4.62	Mean : 33.42
5	3rd Qu.:-17.64	3rd Qu.:183.2	3rd Qu.:543.0	3rd Qu.:4.90	3rd Qu.: 42.00
6	Max. :-10.72	Max. :188.1	Max. :680.0	Max. :6.40	Max. :132.00

Das zusammenhängende Ergebnis ist in der nächsten Abbildung zu sehen. Das Resultat werden TeX-Freaks wahrscheinlich noch weiter bearbeiten, aber für uns soll es bis hierhin reichen.

Übung 5: Wiederholen Sie die durchgeführten Schritte mit `relax`.

Übung 6: Erstellen Sie eine Tabelle, in der die Zahlen besser ausgerichtet sind. Auch hier sollte man eher mit R das Ergebnis modifizieren als hinterher auf dem Wege der Textverarbeitung. Diese Übung ist nur für LaTeX-Experten gedacht.



Report: _____

File: simple-bsp.rev
in: /home/wiwi/pwolf/lehre/rkurs/documentation

March 14, 2008

Berechnung einiger Statistiken zum Datensatz quakes ...

⟨* 1⟩ =

```
summary(quakes)
```

```
Wed Mar 12 17:42:09 2008
      lat              long           depth           mag
 Min.   :-38.59   Min.   :165.7   Min.   : 40.0   Min.   :4.00
 1st Qu.:-23.47   1st Qu.:179.6   1st Qu.: 99.0   1st Qu.:4.30
 Median :-20.30   Median :181.4   Median :247.0   Median :4.60
 Mean   :-20.64   Mean   :179.5   Mean   :311.4   Mean   :4.62
 3rd Qu.:-17.64   3rd Qu.:183.2   3rd Qu.:543.0   3rd Qu.:4.90
 Max.   :-10.72   Max.   :188.1   Max.   :680.0   Max.   :6.40
    stations
 Min.   : 10.00
 1st Qu.: 18.00
 Median : 27.00
 Mean   : 33.42
 3rd Qu.: 42.00
 Max.   :132.00
```

Nun soll die Tabelle noch einmal etwas schöner fabriziert werden.

⟨* 1⟩+ ≡

```
library(xtable,lib.loc="/home/wiwi/pwolf/lib")
xtable(summary(quakes))
```

	lat	long	depth	mag	stations
1	Min. :-38.59	Min. :165.7	Min. : 40.0	Min. :4.00	Min. : 10.00
2	1st Qu.:-23.47	1st Qu.:179.6	1st Qu.: 99.0	1st Qu.:4.30	1st Qu.: 18.00
3	Median :-20.30	Median :181.4	Median :247.0	Median :4.60	Median : 27.00
4	Mean :-20.64	Mean :179.5	Mean :311.4	Mean :4.62	Mean : 33.42
5	3rd Qu.:-17.64	3rd Qu.:183.2	3rd Qu.:543.0	3rd Qu.:4.90	3rd Qu.: 42.00
6	Max. :-10.72	Max. :188.1	Max. :680.0	Max. :6.40	Max. :132.00

Lösung von der letzten Übung, die doch gar nicht so kompliziert ist.

7 ⟨* 1⟩+ ≡

```
library(xtable,lib.loc="/home/wiwi/pwolf/lib")
xtable(format(apply(quakes,2,summary)),align="rrrrrr")
```

Es folgt das getexte Ergebnis:

	lat	long	depth	mag	stations
Min.	-38.59	165.70	40.00	4.00	10.00
1st Qu.	-23.47	179.60	99.00	4.30	18.00
Median	-20.30	181.40	247.00	4.60	27.00
Mean	-20.64	179.50	311.40	4.62	33.42
3rd Qu.	-17.64	183.20	543.00	4.90	42.00
Max.	-10.72	188.10	680.00	6.40	132.00

4 Ergebnisse per Befehl speichern

Im interaktiven Zeitalter genießt der direkte Transport zwischen Anwendungen ein hohes Ansehen. Jedoch gibt es viele Situationen, in denen man während einer Funktionsabarbeitung Daten sichern möchte. Die hierzu vorgestellten Werkzeuge lassen sich natürlich auch per Hand einsetzen.

4.1 Ergebnisspeicherung mit `cat()`

Mit einigen Funktionen können wir Ergebnisse dauerhaft in Dateien speichern. Von dort können sie (später) in Texte integriert oder an Drucker geschickt werden.

`cat()` ist die allgemeine (Datei-) Schreibfunktion von R. Mit ihr lassen sich viele einfache Objekte in eine Datei schreiben.

Übung 7: Schreiben Sie per `cat()` den Vektor `euro` in die Datei `output.txt` und stellen Sie fest, was verloren gegangen ist. Wie sollte der Output in der Datei aussehen?

8 ⟨* 1⟩+ ≡
```
cat(file="output.txt",euro,"\n")
readLines("output.txt")
```

Output:

```
Wed Mar 12 15:02:43 2008
[1] "13.7603 40.3399 1.95583 166.386 5.94573 6.55957 0.787564 1936.27 ..."
```

Übung 8: Drucken Sie `euro` mit `cat()` mit gleicher Nachkommastellenanzahl aus und geben Sie als Trennung ein Newline-Zeichen an.

9 ⟨* 1⟩+ ≡
```
cat(format(euro),file="output.txt",sep="\n")
readLines("output.txt")
```

Output:

```
Wed Mar 12 15:06:29 2008
 [1] "  13.760300" "  40.339900" "   1.955830" " 166.386000" "   5.945730"
 [6] "   6.559570" "   0.787564" "1936.270000" "  40.339900" "   2.203710"
[11] " 200.482000"
```

Übung 9: Schreiben Sie vor die formatierten Zahlenwerte von `euro` die zugehörigen Namen. Damit die Zahlen passend untereinander stehen, werden die Namen sicherheitshalber ebenfalls formatiert.

10 ⟨* 1⟩+ ≡
```
h<-paste(format(names(euro)),format(euro))
cat(h,file="output.txt",sep="\n")
readLines("output.txt")
```

Output:

```
Wed Mar 12 15:11:26 2008
 [1] "ATS   13.760300" "BEF   40.339900" "DEM    1.955830" "ESP  166.386000"
 [5] "FIM    5.945730" "FRF    6.559570" "IEP    0.787564" "ITL 1936.270000"
 [9] "LUF   40.339900" "NLG    2.203710" "PTE  200.482000"
```

Übung 10: Schreiben Sie jeweils zwei Einträge von `euro` neben- und sechs bzw. fünf untereinander, jedoch so, dass über den Werten die zugehörigen Namen stehen. Dazu können Sie folgende Anweisungen verwenden. Untersuchen Sie Schritt für Schritt, wie das Objekt verändert wird.

11 ⟨* 1⟩+ ≡

```
h<-euro
h<-rbind(format(names(h)),format(h))
max.breite<-apply(h,2,function(x)max(nchar(x)))
h[1,]<-substring(paste(h[1,],"          ",sep=""),1,max.breite)
h<-cbind(h,"")
h<-matrix(h,2,12,TRUE)
h<-rbind(h,"\n")
h<-paste("",paste(h,collapse=" "))
cat(h,file="output.txt",sep="\n")
# zur Anzeige der Datei
h<-noquote(cbind(readLines("output.txt")))
dimnames(h)<-list(rep(" ",nrow(h))," ")
h
```

Output:

```
   ATS         IEP
     13.760300    0.787564
   BEF         ITL
     40.339900 1936.270000
   DEM         LUF
      1.955830   40.339900
   ESP         NLG
    166.386000    2.203710
   FIM         PTE
      5.945730  200.482000
   FRF
      6.559570
```

4.2 Ergebnisspeicherung mit `write.table()` & Co.

Wir sehen, dass `cat()` extrem vielfältig ist, jedoch für kompliziertere Objekte / Vorstellungen Vorarbeiten notwendig sind. Denn leider gehen beispielsweise Matrix-Strukturen verloren und Liste lassen sich schon gar nicht direkt mit `cat()` wegschreiben. Glücklicherweise helfen uns mächtigere Funktionen weiter, wie:

```
write write.table writeLines write.csv
```

Der Einzeiler `write()` greift auf `cat()` zurück und wird selten verwendet. `write.csv()` benötigt man, wenn später mit einer Tabellenkalkulation weitere Schritte unternommen werden sollen. Mit `writeLines` lassen sich die Elemente eines Objektes zeilenweise ablegen.

12 ⟨* 1⟩+ ≡

```
mat<-summary(quakes[1:10,])
writeLines(mat,con="output.txt")
noquote(readLines("output.txt"))
```

Output:

```
Wed Mar 12 15:56:47 2008
```

```
 [1] Min.   :-28.74   1st Qu.:-24.66   Median :-20.42   Mean   :-21.11
 [5] 3rd Qu.:-18.40   Max.   :-11.70   Min.   :166.1    1st Qu.:181.2
 [9] Median :181.7    Mean   :180.4    3rd Qu.:182.0    Max.   :184.3
[13] Min.   : 42.0    1st Qu.:194.2    Median :386.5    Mean   :383.3
[17] 3rd Qu.:625.0    Max.   :650.0    Min.   :4.000    1st Qu.:4.125
[21] Median :4.350    Mean   :4.470    3rd Qu.:4.775    Max.   :5.400
[25] Min.   :11.0     1st Qu.:15.0     Median :19.0     Mean   :25.3
[29] 3rd Qu.:39.5     Max.   :43.0
```

Meistens wird dieses Ergebnis uns so nicht gefallen. Darum heben wir lieber `write.table()` als zentrale Funktion zur Speicherung von Matrizen hervor.

13 ⟨* 1⟩+ ≡
```
mat<-summary(quakes[1:10,])
write.table(mat,file="output.txt",quote=FALSE,col.names=TRUE)
noquote(readLines("output.txt"))
```
Output:

```
Wed Mar 12 15:57:25 2008
[1]      lat      long     depth      mag    stations
[2]  Min.   :-28.74   Min.   :166.1   Min.   : 42.0   Min.   :4.000   Min.   :11.0
[3]  1st Qu.:-24.66   1st Qu.:181.2   1st Qu.:194.2   1st Qu.:4.125   1st Qu.:15.0
[4]  Median :-20.42   Median :181.7   Median :386.5   Median :4.350   Median :19.0
[5]  Mean   :-21.11   Mean   :180.4   Mean   :383.3   Mean   :4.470   Mean   :25.3
[6]  3rd Qu.:-18.40   3rd Qu.:182.0   3rd Qu.:625.0   3rd Qu.:4.775   3rd Qu.:39.5
[7]  Max.   :-11.70   Max.   :184.3   Max.   :650.0   Max.   :5.400   Max.   :43.0
```

Die Überschriften sind nicht perfekt angebracht. Etwas Handsteuerung hilft weiter. Zunächst stellen wir fest, wie breit der längste Eintrag jeder Spalte ist. Dann hängen wir an jeden Namen soviele Leerzeichen an, bis die Breite passend ist. Wegschreiben geschieht wie oben.

14 ⟨* 1⟩+ ≡
```
h<-summary(quakes[1:10,])
max.breite<-apply(h,2,function(x)max(nchar(x)))
colnames(h)<-substring(paste(colnames(h),"         ",sep=""),1,max.breite)
write.table(h,file="output.txt",quote=FALSE,col.names=TRUE)
noquote(readLines("output.txt"))
```

Output:

```
Wed Mar 12 16:08:57 2008
[1]      lat              long           depth            mag           stations
[2]  Min.   :-28.74   Min.   :166.1   Min.   : 42.0   Min.   :4.000   Min.   :11.0
[3]  1st Qu.:-24.66   1st Qu.:181.2   1st Qu.:194.2   1st Qu.:4.125   1st Qu.:15.0
[4]  Median :-20.42   Median :181.7   Median :386.5   Median :4.350   Median :19.0
[5]  Mean   :-21.11   Mean   :180.4   Mean   :383.3   Mean   :4.470   Mean   :25.3
[6]  3rd Qu.:-18.40   3rd Qu.:182.0   3rd Qu.:625.0   3rd Qu.:4.775   3rd Qu.:39.5
[7]  Max.   :-11.70   Max.   :184.3   Max.   :650.0   Max.   :5.400   Max.   :43.0
```

Übung 11: Was ergibt sich, wenn wir Listen mit `write.table()` speichern?

15 ⟨* 1⟩+ ≡
```
write.table(quakes[1:10,],file="output.txt",quote=FALSE,col.names=TRUE)
noquote(readLines("output.txt"))
```

Output:

```
Wed Mar 12 16:11:25 2008
```

```
 [1] lat long depth mag stations 1 -20.42 181.62 562 4.8 41
 [3] 2 -20.62 181.03 650 4.2 15  3 -26 184.1 42 5.4 43
 [5] 4 -17.97 181.66 626 4.1 19  5 -20.42 181.96 649 4 11
 [7] 6 -19.68 184.31 195 4 12    7 -11.7 166.1 82 4.8 43
 [9] 8 -28.11 181.93 194 4.4 15  9 -28.74 181.74 211 4.7 35
[11] 10 -17.47 179.59 622 4.3 19
```

Das muss noch schöner werden! Deshalb formatiere das Objekt und schreibe es erneut weg.

16 ⟨* 1⟩+ ≡

```
write.table(format(quakes[1:10,]),file="output.txt",quote=FALSE,col.names=TRUE)
noquote(readLines("output.txt"))
```

Output:

```
Wed Mar 12 16:18:48 2008
 [1] lat long depth mag stations 1 -20.42 181.62 562 4.8 41
 [3] 2 -20.62 181.03 650 4.2 15  3 -26.00 184.10  42 5.4 43
 [5] 4 -17.97 181.66 626 4.1 19  5 -20.42 181.96 649 4.0 11
 [7] 6 -19.68 184.31 195 4.0 12  7 -11.70 166.10  82 4.8 43
 [9] 8 -28.11 181.93 194 4.4 15  9 -28.74 181.74 211 4.7 35
[11] 10 -17.47 179.59 622 4.3 19
```

Beschreibe, was gefällt und was nicht!

Es könnten die Spaltennamen noch angepasst werden und die Zeilennamen sind nicht gut verarbeitet. Wir formen einmal das Objekt in eine Matrix um, entfernen die Zeilen- und Spaltennamen und fügen die Spaltennamen an die Matrix an.

17 ⟨* 1⟩+ ≡

```
h<-format(quakes[1:10,])
cnames.h<-colnames(h)
h<-as.matrix(h)
rownames(h)<-NULL
h<-rbind(colnames(h),h)
colnames(h)<-NULL
write.table(format(h),file="output.txt",quote=FALSE,
            col.names=FALSE,row.names=FALSE)
readLines("output.txt")
```

Output:

```
Wed Mar 12 16:31:48 2008
 [1] "lat      long     depth    mag      stations"
 [2] "-20.42   181.62   562      4.8      41      "
 [3] "-20.62   181.03   650      4.2      15      "
 [4] "-26.00   184.10    42      5.4      43      "
 [5] "-17.97   181.66   626      4.1      19      "
 [6] "-20.42   181.96   649      4.0      11      "
 [7] "-19.68   184.31   195      4.0      12      "
 [8] "-11.70   166.10    82      4.8      43      "
 [9] "-28.11   181.93   194      4.4      15      "
[10] "-28.74   181.74   211      4.7      35      "
[11] "-17.47   179.59   622      4.3      19      "
```

4.3 `sink()` und `capture.output()` zur Outputumleitung

Es gibt etliche `print`-Funktionen. Deshalb denkt jeder, das ist genau das, was ich suche. Jedoch besitzen diese kein `file`-Argument, so dass wir sie nicht direkt zur Speicherung von Objekten verwenden können, sondern wieder auf Cut-and-Paste zurückgreifen müssen.

In Kombination mit `sink()` sieht die Sache anders aus. Denn wir können mit `sink()` den Namen einer Datein festlegen, in der dann alle – auch mit `print()` erstellten – Ausgaben landen, bis der Kanal mit einem argumentlosen Aufruf von `sink()` wieder geschlossen wird.[2] Diese Technik können wir auch innerhalb von Funktionen anwenden, in denen Versuche mit der Cut-and-Paste-Technik unbrauchbar sind.

18 ⟨* 1⟩+ ≡
```
sink("output.txt")
base::print(quakes[1:8,])
sink()
noquote(cbind(readLines("output.txt")))
```
Diese Anweisungen fülllen die Datei genau, wie wir erwarten, mit dem Inhalt:

```
     [,1]
[1,]      lat   long depth mag stations
[2,] 1 -20.42 181.62   562 4.8       41
[3,] 2 -20.62 181.03   650 4.2       15
[4,] 3 -26.00 184.10    42 5.4       43
[5,] 4 -17.97 181.66   626 4.1       19
[6,] 5 -20.42 181.96   649 4.0       11
[7,] 6 -19.68 184.31   195 4.0       12
[8,] 7 -11.70 166.10    82 4.8       43
[9,] 8 -28.11 181.93   194 4.4       15
```

Dieses kann alternativ auch durch folgende Anweisung erzielt werden:

19 ⟨* 1⟩+ ≡
```
capture.output({base::print(quakes[1:8,])},file="output.txt")
readLines("output.txt")
```
Mit diesem Instrument lassen sich schnell einzelne Objekte in einer Ausgabe-Qualität, wie sie von der Console her bekannt ist, realisieren.

5 Externe Editoren

Die bisher beschriebenen Konzepte gehen von R als zentralem Prozess aus. Es kann aber auch vorteilhaft sein, sein Standbein außerhalb von R zu positionieren, weil sich dort die Hauptarbeit – wie etwa Texterstellung oder Tabellenkalkulation – zuträgt. Bei lokalen Fragen kann es dann interessant sein, aus der jeweiligen Software Berechnungsaufträge an R zu senden und dann die Ergebnisse in die weiteren Arbeitsprozesse zu integrieren. Unter diese Idee fallen zum Vorschläge, mit R Berechnungen für Excel-Tabellen-Felder durchzuführen.

Ein externes Standbein kann auch dadurch begründet sein, dass man kompliziertere Anweisungsgefüge oder Skript-Dateien extern erstellen und verwalten will. Damit lautet das Vorgehen: Funktionsdefinitionen außerhalb von R schreiben, dann diese ins R einladen und dort starten. Editoren wie Emacs oder der WinEdt unter Windows bieten dazu die Möglichkeit an, bestimmte Sequenzen an einen R-Prozess zu übertragen und die Ergebnisse wiederum im Editor darzustellen. Vorteilhaft ist hierbei, dass die überlegten und erarbeiteten Bausteine selbst dann überdauern, wenn der R-Prozess abstürzten sollte.

[2] Innerhalb von `relax` muss die Original-Print-Funktion `base::print` zum Einsatz kommen. Andernfalls kann der `sink`-Mechanismus nicht greifen.

6 Batch-Verarbeitung

Neben der interaktiven Arbeit gibt es Situationen, in denen extern Bearbeitungsaufträge entworfen werden, die dann mit R automatisch umgesetzt werden müssen. Dieses führt uns zur Batch-Verarbeitung.

6.1 R im Batch-Betrieb

Stehen alle R-Anweisungen in der Datei R-Input, dann kann R unter Linux mit dem Aufruf:

```
R CMD BATCH R-Input R-Output
```

oder – falls man den Vorspann nicht möchte – mit:

```
R CMD BATCH -q R-Input R-Output
```

dazu veranlasst werden die Anweisungen aus der Input-Datei auszuwerten und die Ergebnisse in der Datei `R-Output` abzulegen. Steht in `R-Input` beispielsweise:[3]

```
summary(Nile)
stem(Nile)
```

wird in der Output-Datei etwa Folgendes stehen:

```
> summary(Nile);stem(Nile)
Min.  1st Qu.  Median Mean 3rd Qu.  Max.
456.0 798.5 893.5 919.4 1032.0 1370.0

The decimal point is 2 digit(s) to the right of the |

4 | 6
5 |
6 | 5899
7 | 000123444455667778
8 | 00001122223334455555667779
9 | 0011222244466678899
10 | 0122234455
11 | 00012244566678
12 | 112356
13 | 7

>
> proc.time()
user system elapsed
1.020 0.024 1.086
```

Auf einem Windows-Rechner geht das im Prinzip auch, jedoch muss der Rechner passend hergerichtet sein. Man kann dazu beispielsweise in einer DOS-Box oder mit der *Eingabeaufforderung* Folgendes aktivieren:

```
C:\Programme\R\R-2.6.1\bin\Rcmd BATCH -q R-Input R-Output
```

Natürlich muss der Pfad zu der Anwendung `Rcmd` passend angegeben werden.

[3] Leider schreibt die Funktion `stem` ihr Ergebnis unsauber direkt in den Ausgabekanal. Dieses wird von `relax` z.Z. nicht aufgefangen. Deshalb wird empfohlen, als Alternative die Funktion `stem.leaf()` zu verwenden, die in den Paketen `aplpack` oder `Rcmdr` verhanden ist.

6.2 Sweave

Mit der einfachen Batch-Variante bekommt man optisch nur sehr magere, textlose Outputs. Unter der Verwendung von `Sweave` und LaTeX lassen sich dagegen überaus schöne kommentierte Ausdrucke erstellen. Dazu ist es erforderlich, wie für `relax` nach den Regeln des `noweb`-Systems für literate Programmierung alle notwendigen Anweisungen in Code-Chunks zu schreiben. Code-Chunk-Header beginnen wieder mit der Sequenz `<<` und enden mit `>>=`. Ein Code-Chunk wird durch eine Zeile mit einem `@` links beendet. Der Klammeraffe leitet einen Text-Chunk ein, der Texte angereichert mit LaTeX-Kommandos enthalten kann.

Ist `rohdoc.Rnw` eine Datei, die der beschriebenen Form genügt, so kann in R diese Rohdatei durch den Aufruf

```
> Sweave("rohdoc.Rnw")
```

zu einem LaTeX-Dokument `rohdoc.tex` aufbereitet werden. Die Formatierung erfordert übrigens die Einbindung des TeX-Pakets `Sweave`.

Auf einige Besonderheiten soll hier noch hingewiesen werden.

- Ausdrücke der Form `\Sexpr{` *Anweisung* `}` werden bei der Bearbeitung durch das Auswertungsergebnis der *Anweisung* ersetzt.
- In den Code-Chunk-Headern können Verarbeitungshinweise gegeben werden. Mehrere werden durch Kommata getrennt. Hier ein paar Beispiele:

`echo = FALSE`	R-Anweisungen nicht protokollieren
`fig = TRUE`	Plot realisieren
`eval = FALSE`	Code wird nicht evaluiert
`split = TRUE`	Text-Output landet in verschiedenen Dateien
`strip.white = all`	Leerzeilen werden entfernt
`results = tex`	Ergebnisse werden als TeX-Anweisungen angesehen
`results = verbatim`	Ergebnisse werden in eine Verbatim-Umgebung gepackt

- Durch Verwendung von `xtable` können mit `Sweave` also automatisch schön formatierte Rechenergebnisse in schönen Texten generiert werden.

Für weitergehende Fragen sei empfohlen, einen Blick in die Hilfe `?Sweave` und ins Manual zu werfen:

`http://www.ci.tuwien.ac.at/~leisch/Sweave/Sweave-manual-20060104.pdf`

7 Graphiken speichern

Nach den vielen Abhandlungen über textliche Speicherung werden in diesem Abschnitt noch ein paar Bemerkungen über die Speicherung von Graphiken angefügt.

Graphik-Devices. Durch Wahl eines passenden Graphik-Device lassen sich Graphiken direkt in verschiedenen Formaten erstellen. Es ist dabei besonders darauf zu achten, dass nach Fertigstellung einer Graphik eine Schließ-Operation mit `dev.off()` stattfindet. Andernfalls wird die Zieldatei unbrauchbar sein. Mit `?Devices` lassen sich die erreichbaren Devices auflisten. Zum Beispiel findet man:

```
*  'postscript' Writes PostScript graphics commands to a file
*  'pdf' Write PDF graphics commands to a file
*  'pictex' Writes LaTeX/PicTeX graphics commands to a file
*  'xfig' Device for XFIG graphics file format
*  'bitmap' bitmap pseudo-device via 'GhostScript' (if available).
```

Die Devices werden durch Funktionen aktiviert, deren Namen den Devices entsprechen. Als Hauptargument ist diesen der Dateinamen mitzuteilen. Weiterhin lassen sich Größe und andere Device spezifischen Eigenschaften per Argumente setzen. Näheres schaue man in den Hilfeseiten nach.

Graphik-Kopien per `dev.copy()`. Mit der Funktion `dev.copy()` lassen sich Kopien eines graphischen Device realisieren. Das erste Argument benennt das Zieldevice. Beispielsweise lässt sich eine Postscript-Kopie durch

```
dev.copy(postscript,"graphtest1.ps",horizontal=FALSE,
         width=4.0,height=3.0); dev.off()
```

erstellen. Die Setzung `horizontal=FALSE` ist für eine LaTeX-Einbindung wichtig, damit das Bild hinterher richtig steht. Eine solche Einbindung hat bei Einsatz des Pakets `graphicx` folgende Gestalt:

```
\begin{center}
\includegraphics[width=12cm]{graphtest1.ps}
\end{center}
```

Graphik-Kopien per Knopf-Druck. Hinter dem Kopf `SavePlot` von `relax` steckt – wie vielleicht manche schon vermutet haben – ein `dev.copy`-Aufruf. Nach dem gleichen Muster werden auch die Operationen umgesetzt, die beispielsweise unter Windows über das Datei-Menü des graphischen Device zur Graphik-Speicherung zu finden sind.