Grafikausgabe mit "R" - zwei Beispiele aus der Praxis der Ökologischen Station*

Richard Müller**

Ökol. Station in der JH Sorpesee, Am Sorpesee 7, 59846 Sundern

8. September 2012

Grafische Darstellungen, die höhere Anforderungen stellen, als eine Bürosoftware normalerweise leisten kann, lassen sich oft mit dem Mathematik- und Statistikprogramm "R" lösen. Zwei Beispiele aus der Praxis der Ökologischen Station werden dargestellt.

1 Schlechte Lösungen mit Tabellenkalkulationen

Tabellenkalkulationen haben ihre Grenzen. Dabei wird hier nicht an die maximale Größe der Tabelle gedacht (die bei den Anwendungen der Ökologischen Station bei weitem nicht erreicht wird) oder an die möglichen Rechenoperationen, sondern an die grafische Ausgabe. Die Ergebnisse limnologischer Untersuchungen stehender Gewässer werden oft, abweichend von der sonst üblichen Konvention, in einem Koordinatensystem dargestellt, bei dem der Nullpunkt links oben, die x-Achse von oben nach unten und die y-Achse von links nach rechts dargestellt wird, so dass man gedanklich den See in die Grafik hineinprojizieren kann.

Durch Vertauschen der x- und y-Achsen und durch die Umkehrung der senkrechten Achse würde sich das auch mit Tabellenkalkulationen wie Excel oder OOo-Calc erreichen lassen. Sobald aber mehrere Parameter in einem System dargestellt werden, wird das schwierig. Deshalb findet man oft Grafen, die mit solchen Programmen erstellt werden, in der Form wie in Abb. 1. Da diese Art der Darstellung jedoch in der Limnologie unüblich ist, sollte sie vermieden werden. Notfalls könnte man das Bild in einem Grafikformat abspeichern, mit einer Bildbearbeitung um 90 °drehen und so dann in den Text einsetzen. Es ist offensichtlich, dass diese Vorgehensweise ein ziemlicher Behelf ist.

Das zweite Beispiel bezieht sich auf sogenannte Heatmaps oder Carpetplots. Damit Darstellungen gemeint, bei denen keine Wertepaare, sondern Wertetripel vorliegen. Der

*herunterladbar unter www.phytoplankton.info/download/r-grafik.pdf.

**r.mueller@oeko-sorpe.de

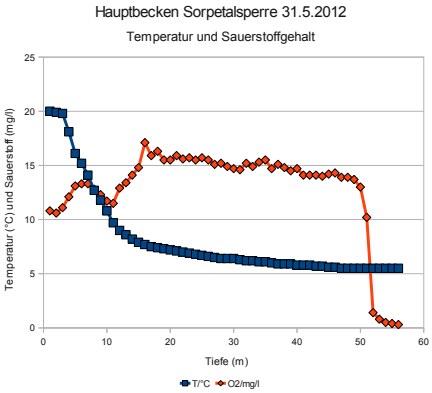

Abbildung 1: *Darstellung von zwei Parametern in einem Koordinatensystem, wie sie häufig mit Tabellenkalkulationen erstellt werden. Die Tiefenachse (x-Achse) sollte jedoch von oben nach unten verlaufen*

dritte Wert wird dabei farbcodiert in einem x-y-Diagramm dargestellt. Auch hierfür gibt es Methoden, die solche Darstellungen mit Excel oder OpenOffice Calc erlauben. So wurde besipielsweise die Strömungsgeschwindigkeit in einem Flussquerschnitt an mehreren Stellen gemessen. Damit liegen für jeden Messpunkt drei Werte vor: Meter vom Ufer (Breite, x), Meter von der Oberfläche (Tiefe, y) und die gemessene Geschwindigkeit (z). Bei einfachen Office-Lösungen, für die Makros im Internet angeboten werden [3], sähe das Ergebnis wie in Abb. 2 aus. Eines der Probleme hierbei ist, dass nicht immer über

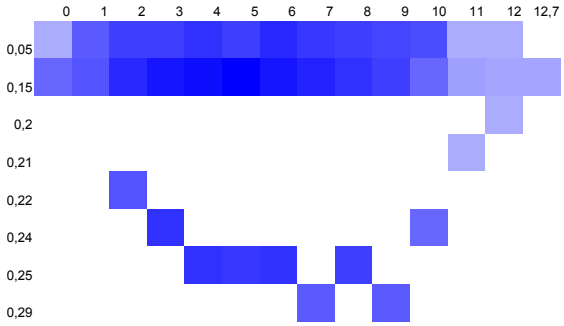

Abbildung 2: *Heatmap von Strömungsgeschwindigkeiten in einem Flussquerschnitt, erzeugt mit OpenOffice-Calc. Je dunkler die Farbe, desto höher die Geschwindigkeit*

die gesamte Flussbreite in der gleichen Tiefe gemessen wurde. Manche Werte wurde in beispielsweise in 22 cm Tiefe, andere in 25 cm Tiefe gemessen. Soll man jetzt diese beiden Werte einem Tiefen-Mittelwert zuordnen oder jeden Wert separat aufführen? Wie man es auch macht, nie werden solche Grafiken qualitativ hochwertig sein. Darüber hinaus ist die Farbgebung mangelhaft, außerdem gibt es keine Farbverläufe. Fehlende Messwerte bleiben weiß. Neben dieser Lösung finden sich im Internet auch hochwertigere, die aber

immer spezielle Mathematikprogramme eingebunden haben (Matlab oder ähnliche), wie z.B. die Lösung von Johannes Hopf, die Gnuplot verwendet [4].

2 Lösungen mit "R"

2.1 Was ist "R"?

R ist die OpenSource-Implementation der kommerziellen Mathematik- und Statistiksoftware "S" [8]. Für den normalen Computerbenutzer ist sie etwas gewöhnungsbedürftig, da sie keine mit der Maus bedienbare Oberfläche besitzt, sondern mit Eingabe von Befehlen in einem Text-Terminal arbeitet[1]. Dafür ist sie auch sehr mächtig. Dank des freien Quellcodes existieren mittlerweile über 5000 Zusatzpakete, mit denen sich unzählige Fragestellungen bearbeiten lassen. Besonders Anwendungen für biologische Probleme, vor allem für die Auswertung von DNA-Sequenzierungen und ähnlichem, sind in letzter Zeit veröffentlicht worden. R wurde auf Unix-Systemen entwickelt, mittlerweile gibt es aber Portierungen für jedes Betriebssystem. Windows- und Mac-User sind also nicht außen vor. R lässt sich von der Seite `www.r-project.org` herunterladen. Dort finden sich auch Installationsanleitungen und ein Großteil der Dokumentation.

2.2 Generierung von limnologischen x-y-Diagramme mit OpenOffice und R

Für die Generierung von einfachen x-y-Diagrammen reichen die Basisfähigkeiten von R. In der Ökologischen Station werden die Messwerte in ein standardisiertes Tabellenkalkulationsblatt eingetragen, das Buttons für die Grafikausgabe enthält (Abb. 3). Beim Drücken auf einen derartigen Knopf wird ein Makro aufgerufen, das die momentane Tabelle in eine temporäre Datei speichert und R aufruft:

```
 REM  *****  BASIC  *****
' 29.12.2007 RMü    ***** "REM" und "'" kennzeichnen einen Kommentar, d.h.
' Text hinter diesen Zeichen wird nicht als Befehl interpretiert.
Sub TO2
'--------------------Temperatur, Sauerstoff---------------------------
Dim oDocument As Object
Dim mFileProperties(1) As New com.sun.star.beans.PropertyValue
Dim sFileName As String, sFilterName As String, sFilterOptions As String,
      sHeimat as String, sProgramm as String, sR_local_ort As String
Dim Dummy()
 'Heimatverzeichnis erfragen
 sHeimat = Environ ("HOME")
 'Pfad für R-local anpassen
 sR_local_ort = sHeimat
 'Dateinamen anpassen
```

[1] Es existieren inzwischen auch grafische Benutzeroberflächen, mit denen zumindest ein Teil der Arbeit mit der Maus erledigt wird

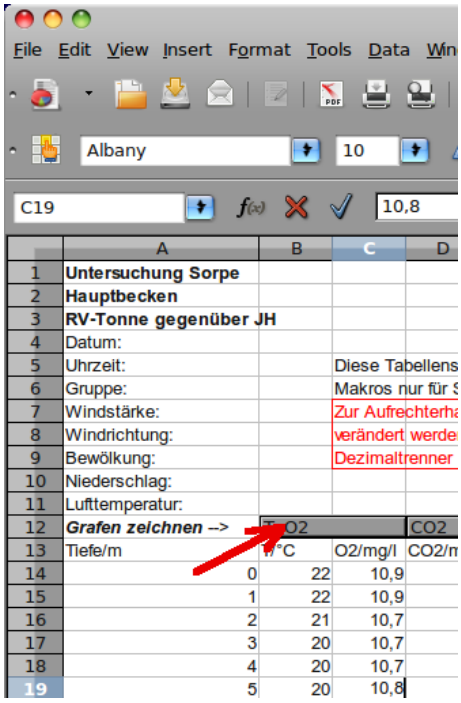

Abbildung 3: *Ausschnitt aus der Tabellenseite zur Erfassung von Messwerten mit Button zum Aufruf des Grafik-Makros*

```
GetGUIType()    ' hier wird abgefragt, ob ein Windows- oder Linux-System vorliegt.
If GetGUIType = 1 then
  ' 1 = Windows
  sFileName="file:///C:Temp/TempFile.csv"

elseif GetGUIType = 4 then sFileName="file://" + sHeimat + "Temp/TempFile.csv"
  ' 4 = Linux/UNIX
end if
 sFilterName = "Text - txt - csv (StarCalc)"
 sFilterOptions = "59,34,SYSTEM,1,1/1/1/1/1/1/1/1/1"

'Dateieigenschaften
mFileProperties(0).Name = "FilterName"
mFileProperties(0).Value = sFilterName
mFileProperties(1).Name = "FilterOptions"
mFileProperties(1).Value = sFilterOptions

oDocument = StarDesktop.ActiveFrame.Controller.Model
```

```
 'Speichert eine Kopie nach 'sFileName'
 oDocument.storeToURL(sFileName, mFileProperties())
 If GetGUIType = 1  then
 sProgramm = "C:\Programme\R\r2061\bin\Rcmd.exe BATCH C:\R-local\Haupt_TO2.R"
 ' Hier wird der Befehl zum Aufruf von R definiert ("Rcmd.exe") und ihm
 ' das R-Skript "Haupt_TO2.R" übergeben,
 ' das die grafische Darstellung von Temperatur und Sauerstoff im Hauptbecken übernimmt.
 ' "R-local" ist das Verzeichnis, das die R-Skripte enthält.
 ' Hier wird eine ältere Version von R eingesetzt (2.06.1). Mittlerweile (Aug. 2012)
 ' ist 2.15.1 aktuell.
 elseif GetGUIType = 4 then
   sProgramm = "R CMD BATCH " + sR_local_ort + "/R-local/Haupt_TO2.R"
 end if
 Shell(sProgramm)   ' und hier der Aufruf von R

End Sub

' ähnlich geht es mit den anderen Parametern weiter.
```

Listing 1: *OpenOffice-Basic-Makro zum Aufruf von R*

Sobald man auf den Button klickt, kann man kurzzeitig die Meldung von R in einem Terminalfenster sehen, das nach kurzer Zeit durch den gewünschten Grafen ersetzt wird. Falls allerdings in der Tabelle Werte falsch eingetragen wurden (z. B. mit Punkt statt mit Komma, oder bei der Verwendung von nicht alpha-numerischen Zeichen wie „<“), bricht R ab. Dann sollte man auf Fehlersuche in der Tabelle gehen.

Ein solches R-Skript soll beispielhaft erläutert werden:

```
# Hier stellt das "#" das Kommentarzeichen dar
# Skript zur Verarbeitung der Sorpe-Daten (Hauptbecken)
# Form der zugrunde liegenden Datei:
# 13 führende Zeilen (nicht ausgewertet)
# 58 Datenzeilen mit 14 Spalten
# weitere Zeilen ignorieren
# Datenfelder entweder leer ('NA') oder vom Typ 'real', d.h. keine < oder > Zeichen
# gespeichert als csv mit ';' als Feldtrenner und '"' als Texttrenner
# (C:\temp\TempFile.csv bzw. ~/Temp/TempFile.csv)
# Reihenfolge der Spalten: Tiefe, Temp., O2, CO2, pH, LF, NH4, NO2, NO3, PO4, Fe,
# Si, Chl, Phä
#
# RMue, 29.12.2007
require(tcltk)                          # Das R-Zusatzpaket "tcltk" wird nur für das
                                        # Fester des Abschlussdialoges gebraucht.
```

```
file="file://temp/TempFile.csv"          # Dateiname
windows()                                # Ausgabe auf einem Windows-System
# X11()                                  # Bei Linux oder OSX (Mac) wäre der Befehl
                                         # X11() der richtige gewesen.

#
inp <- scan(file, sep=";", dec=",", list(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),  skip = 13,
nlines = 58)
#-------------------------------------------------------------------------------
### Temperatur  ###
x <- inp[[2]]; y <- inp[[1]]             # Die zweite Spalte gilt als x-Wert, die
                                         # erste als y-Wert.
                                         # Durch die Vertauschung dieser Werte kann
                                         # das in der Limnologie übliche Koordinaten-
                                         # system generiert werden.
#
#
x1 <- x[!is.na(x)]                       # x-Werte, für die f(x) = NA ist, werden
                                         # herausgefiltert. (Das Ausrufezeichen
                                         # bedeutet eine Verneinung).
y1 <- y[!is.na(x)                        # entsprechend. NA bedeutet "kein Wert erfasst"
plot(ylim=c(60,0),                       # 0 (Tiefe) oben, Bereich senkrechte Achse
                                         # (also von 60 bis 0)
xlim=c(0,25),                            # Bereich waagerechte Achse
pch=1,                                   # Symbole: Kreise
type="b",                                # Linien und Punkte
col=2,                                   # Linien und Punkte in rot
main = "Temperatur, Sauerstoff",         # Beschriftung Graf
ylab="Tiefe (m)",                        # Beschriftung senkrechte Achse
xlab="",                                 # keine Beschriftung waagerechte Achse
x1,y1,                                   # und endlich: die x,y-Werte werden gezeichnet
axes=FALSE)                              # erstmal keine Achsen
axis(side=2)                             # linke Achse
axis(side=3)                             # obere Achse
legend(15,50,"Temperatur (°C)", pch=1, col=2, bty="n")
box()                                    # Kasten um's ganze
#
#-------------------------------------------------------------------------------
#### Sauerstoff  ###
par(new=TRUE)                            # neuer Plot ins gleiche System
x <- inp[[3]]                            # Werte der 3. Spalte (O2)
x1 <- x[!is.na(x)]
y1 <- y[!is.na(x)]
plot(ylim=c(60,0),                       # 0 (Tiefe) oben, Bereich senkrechte Achse
```

```
xlim=c(0,25),                                # Bereich waagerechte Achse
pch=2,                                       # Symbole: Dreieck
type="b",                                    # Linien und Punkte
col=4,                                       # Linien und Punkte in blau
ylab="",                                     # keine neue Beschriftung der senkrechten
xlab="",                                     # und der waagerechten Achse
x1,y1,                                       # und endlich: PLOT!
axes=FALSE)                                  # erstmal keine Achsen
axis(side=1)                                 # untere Achse
legend(15,53,"Sauerstoff (mg/l)", pch=2, col=4, bty="n")
#

res <-    tkmessageBox(title="Beenden?", # Hier gibt es noch einen Abschlussdialog,
                                         # mit dem ggf. der Graf als PDF-Datei
                                         # gespeichert werden kann.
      message="Vor dem Beenden als PDF speichern?",
      icon="question", type="okcancel")
if (tclvalue(res) == "ok")

    Datei <- tkgetSaveFile(initialdir="temp/",defaultextension=".pdf",
    initialfile="Haupt_TO2.pdf")         # Default-Dateiname
# tkmessageBox(message=tclvalue(Datei))
   dev.copy(pdf, tclvalue(Datei))
#else graphics.off()
graphics.off()
```

Listing 2: *R-Skript zur Darstellung eines limnologischen Diagramms*

2.3 Heatmaps mit R erzeugen

Auch Heatmaps lassen sich schön mit R erzeugen [6, 7]. Allerdings ist das ein bisschen aufwendiger als das eben gezeigte Beispiel. Zuerst brauchen wir eine Tabelle, in der die Messwerte gespeichert sind. Wir benutzen dazu das CSV-Format ("Comma separated value"), das universell von allen Systemen und Programmen gelesen werden kann. Es lässt sich mit Excel oder OOo-Calc erzeugen, aber auch mit jedem einfachen Texteditor wie z.B. Notepad unter Windows. Wir erzeugen eine dreispaltige Tabelle, in der die Koordinaten jeder Messung und der Messwert dargestellt ist. Dabei sollte der Punkt als Dezimaltrennzeichen verwendet werden (Excel oder OOo-Calc ggf. auf „Englisch" einstellen), und die einzelnen Felder werden durch ein Komma getrennt. Der Anfang einer solchen Tabelle könnte z.B. so aussehen:

```
x,y,z
0,-0.05,12
```

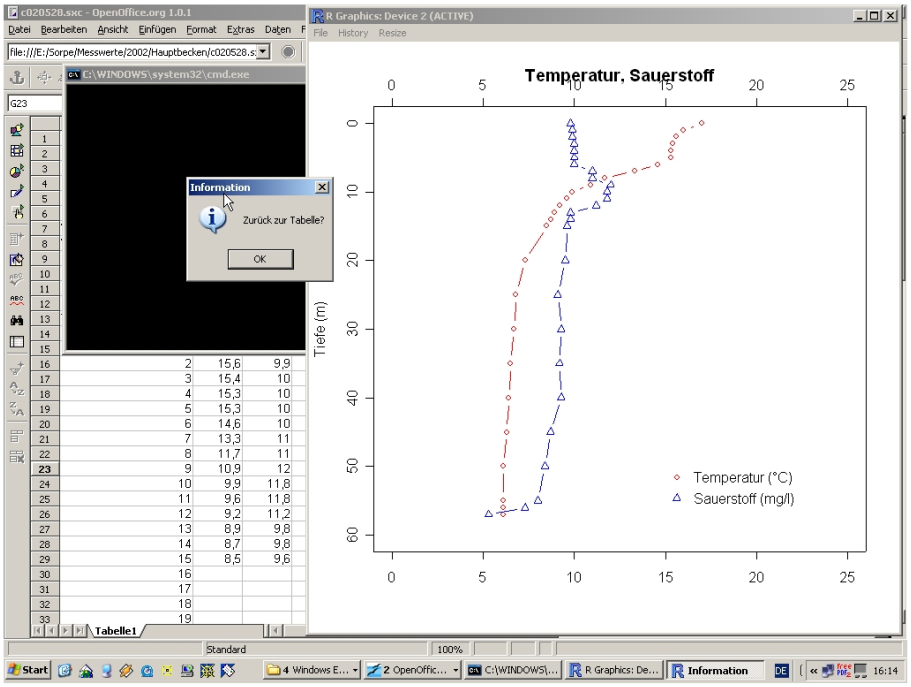

Abbildung 4: *Ausgabefenster von R mit Graf und Dialogfenster. Hier sind senkrechte und waagerechte Achse wie in der Limnologie üblich definiert. Das Dialogfenster wurde inzwischen so geändert, dass der Inhalt des Ausgabefensters als PDF-Datei gespeichert werden kann.*

```
0,-0.15,22.1
1,-0.05,24
1,-0.15,25
2,-0.05,28
2,-0.15,31
2,-0.22,25
```

Listing 3: *Ergebnisse von Strömungsgeschwindigkeitsmessungen in einem Flussquerschnitt, im CSV-Format gespeichert. Jede Zeile enthält einen x-Wert (Entfernung vom linken Ufer), einen y-Wert (Tiefe) und einen z-Wert (Strömungsgeschwindigkeit).*

Da zwei Stellen des Flusses verglichen werden sollen, sind auch zwei Messwerttabellen nötig, deren Dateinamen in unserem Fall `hoenne1.csv` bzw. `hoenne2.csv` lauten. Die beiden zu erzeugenden Heatmaps sollen mit einer gemeinsamen Legende erstellt werden. Das Ergebnis (Abb. 5) ist wesentlich besser und informativer als der erste Ansatz mit gefärbten Tabellenfeldern (Abb. 2). Diese Aufgabe ist mit R in seiner Grundform nicht zu lösen. Es müssen zwei Zusatzpakete installiert werden, was aber bei R fast automatisiert verläuft. Es handelt sich um das Paket `akima`, mit dem Messwerte interpoliert werden, die nicht gemessen wurden [1]. Ansonsten würden leere Stellen auf dem Diagramm erscheinen und es gäbe keine Farbübergänge. So macht es dem Programm auch nichts aus, dass z.B.

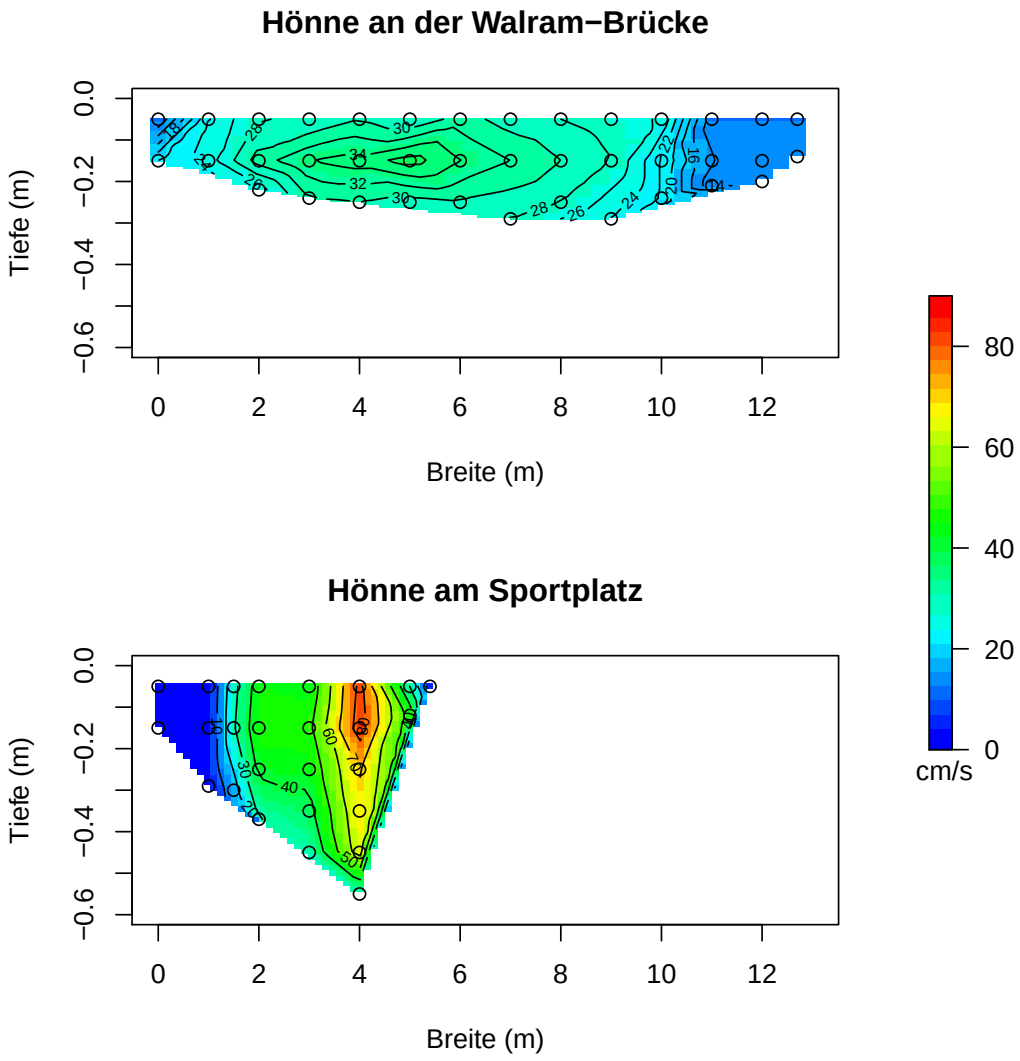

Abbildung 5: *Mit R erzeugte Heatmaps zum Vergleich der Strömungsgeschwindigkeiten an zwei Flussprofilen, aus [5]*

manche Messungen in einer Tiefe von 22 cm, andere bei 25 cm durchgeführt wurden. Um dem Betrachter zu zeigen, wo tatsächlich die Messsonde hingehalten wurde, werden auf dem fertigen Diagramm die Messstellen mit einem kleinen Kreis markiert.

Zur Erzeugung der Legende ist ein weiteres Paket `fields` nötig, denn nur dieses stellt den dazu nötigen Befehl `image.plot` zur Verfügung [2].

Nach Eingabe von "`R`" bzw. "`Rgui.exe`" öffnet sich ein Terminal, in dem alle folgenden Operationen ablaufen (Abb. 6). Die einzelnen Befehle können per Hand eingetippt und jeweils mit Return bestätigt werden, oder alle Befehle werden in einer Textdatei ge-

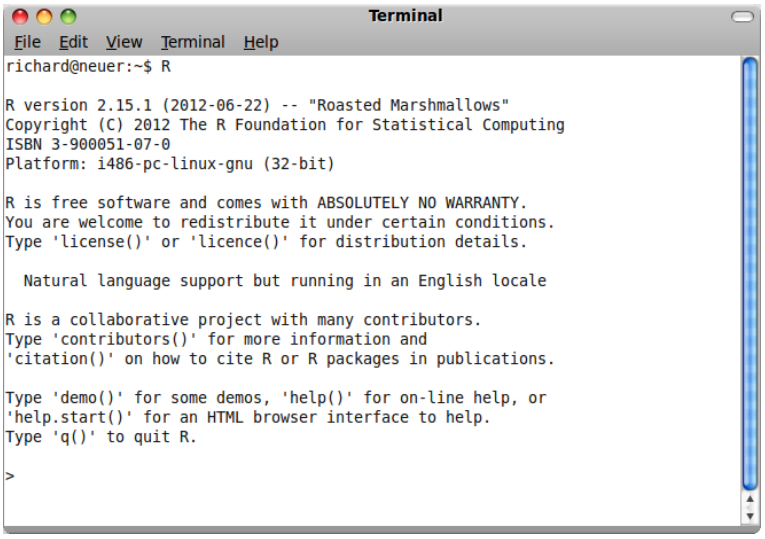

Abbildung 6: *Das R-Terminal, mit das Programm bedient wird*

speichert. Wenn diese Datei dann mit dem Befehl `source("Dateiname")` im R-Terminal aufgerufen wird, werden alle Befehle abgearbeitet und die Grafik in einem eigenen Fenster ausgegeben. Um die Abbildung zu erzeugen, wird das folgende Listing verwendet. Hier wird davon ausgegangen, dass die Dateien `hoenne1.csv` und `hoenne2.csv` im gleichen Verzeichnis liegen. Es sollte auch in das Verzeichnis gewechselt werden, bevor R aufgerufen wird. Andernfalls müsste man alle Pfade anpassen.

```
### Zwei Messungen von Strömungsgeschwindigkeitsprofilen sollen mit R dargestellt werden
### RMü 21.8.2012, r.mueller@oeko-sorpe.de

require(akima)                          # Das Paket akima wird benötigt, um
                                        # nicht vorhandene Punkte zu interpolieren
                                        # und einen schönen Farbgradienten hinzukriegen
require(fields)                         # Das Paket fields wird benötigt, um die
                                        # Farblegende zu erstellen mit image.plot

hoe1 <- read.csv("hoenne1.csv")         # Die Messwerte werden aus einer CSV-Tabelle
hoe2 <- read.csv("hoenne2.csv")         # eingelesen.

pdf("hoenne.pdf")                       # Es wird nicht auf den Bildschirm gezeichnet,
                                        # sondern ein PDF-Dokument erstellt.
                                        # Soll auf den Bildschirm gezeichnet werden,
                                        # so ist die der Befehl durch ein voran-
                                        # gestelltes '#' zu inaktivieren.

split.screen(                           # Mit split.screen wird das Ausgabefeld
```

```
                                        # aufgeteilt. Jeder "screen" kann individuell
                                        # beschrieben werden.
rbind(c(0, .8, 0, 1), c(.8, 1, 0, 1)))  # Mit rbind werden hier zwei Reihen definiert.
                                        # Die Werte geben die Positionierung an.

split.screen(c(2,1), screen = 1) -> ind  # Wir wollen ja hinterher 3 Bereiche haben:
                                        # links zwei Felder für die Abbildungen
                                        # übereinander, rechts einen schmalen Bereich
                                        # für die Legende.

zr <- range(0,90)                       # Hier wird der Bereich für die z-Werte
                                        # festgelegt:
                                        # Die Legende reicht dann von 0 bis 90 cm/s


# Berechnen und Zeichnen des ersten Bildes

screen(ind[1])                          # Der erste Screen (links oben) ist aktiv
plot(ylim = c(-0.6, 0),                 # Der Bereich der Achsen wird definiert
      xlim = c(0, 13),
      xlab = "Breite (m)",              # Beschriftung der Achsen
      ylab = "Tiefe (m)",
      x ~ y,                            # Es soll ein x-y-Diagramm werden
      hoe1,                             # Wertetripel der 1. Messstelle
      main = "Hönne an der Walram-Brücke",  # Überschrift der Grafik
      pch = NA)                         # Die Punkte für die x-y-Wertepaare sollen
                                        # unsichtbar sein ('NA')
hoe1.li <- interp(hoe1$x, hoe1$y, hoe1$z)  # Hier werden die fehlenden Punkte inter-
                                        # poliert
image(hoe1.li,                          # In das Diagramm wird ein Rasterbild aus den
                                        # interpolierten Werten gezeichnet
 zlim = zr,                             # Der vorher definierte Wertebereich wird
                                        # aufgerufen
 col = rev(rainbow(                     # Ein in R vordefiniertes Farbmodell wird
                                        # verwendet
 30, start = 0, end = 8/12)),           # mit 30 Farbabstufungen, die den z-Werten
                                        # zugeordnet werden.
 add = TRUE)                            # Das Bild wird über das bereits Vorhandene
                                        # gezeichnet
contour(hoe1.li,                        # Isotachen werden eingezeichnet
 add = TRUE)
points(hoe1,                            # Die ursprünglichen Messpunkte werden
 pch = 1)                               #  eingezeichnet, und zwar als Kreise
```

```
# Berechnen und Zeichnen des zweiten Bildes erfolgt analog zum ersten

screen(ind[2])
plot(ylim = c(-0.6, 0),
      xlim = c(0, 13),
      xlab = "Breite (m)",
      ylab = "Tiefe (m)",
      x ~ y,
      hoe2,                              # Wertetripel der 2. Messstelle
      main = "Hönne am Sportplatz",
      pch = NA)
hoe2.li <- interp(hoe2$x, hoe2$y, hoe2$z)
image(hoe2.li,
 zlim = zr,
 col = rev(rainbow(
 30, start = 0, end = 8/12)),
 add = TRUE)
contour(hoe2.li,
 add = TRUE)
points(hoe2,
 pch = 1)

# Jetzt noch die Legende

screen(2)                               # Jetzt sind wir in dem schmalen Streifen rechts

image.plot(zlim = zr,
 legend.only = TRUE,                    # Wie der Name sagt: Nur die Legende zeichnen
 smallplot = c(.1, .2, .3, .7),         # Größendefinitionen für die Legende
 col = rev(rainbow(30, start = 0, end = 8/12)))
text(2.3,0.16,"cm/s")                   # In die Legende wird noch die Einheit
                                        # hineingefummelt (mit den Platzierungskoordi-
                                        # naten muss man ein bisschen rumprobieren)
dev.off()                               # Die PDF-Ausgabe wird beendet. Falls
                                        # Bildschirmbetrachtung erwünscht,
                                        # sollte auch dieser Befehl durch '#' inaktiviert
                                        # werden

close.screen(all = TRUE)                # Die virtuellen "Screens" werden deaktiviert
```

Listing 4: *Befehle für die Darstellung der zwei Heatmaps aus Abb. 5.*

3 Do it yourself?

Für den, der Lust hat, sich in R zu vertiefen und die Beispiele einmal "nachzukochen", sind die nötigen Dateien in einem Zip-File zusammengefasst. Die Adresse lautet `www.phytoplankton.info/download/r-grafik.zip`. Die Makros für die Generierung der einfachen x-y-Diagramme sind in den dort zu findenden OOo-Vorlagen enthalten[2]. Man muss darauf achten, dass in OpenOffice die Makro-Ausführung aktiviert ist (Extras → Optionen → Sicherheit → Makro-Sicherheit). Sicherlich lässt sich das auch mit Excel machen, aber da fehlt mir die Erfahrung.

Literatur

[1] Akima, H., Gebhardt, A., Petzold, Th., Maechler, M.) (2012): akima: Interpolation of irregularly spaced data (`CRAN.R-project.org/package=akima`)

[2] Furrer, R., Nychka, D., Sain, S. (2012): fields: Tools for spatial data (`CRAN.R-project.org/package=fields`)

[3] F3K Total (2011): Farbverläufe über mehrere Zellen (Diagramm) (`http://de.openoffice.info/viewtopic.php?f=2&t=49873`)

[4] Hopf, J. (2009): Carpet-Plot 1.3 (`www.johannes-hopf.de/2009/12/carpet-plot-version-1-3/9/`)

[5] Müller, R. (2012): Strömungsgeschwindigkeiten preiswert messen. wasser in schule und bildung **1**, 17–23 `www.phytoplankton.info/wisb.html`

[6] Murrell P. (2011): Raster Images in R Graphics. The R Journal 3/1 (`http://journal.r-project.org/archive/2011-1/RJournal_2011-1_Murrell.pdf`)

[7] Ogasawara, O. (2011): R Graphical Manual (`rgm2.lab.nig.ac.jp/RGM2/index.php`)

[8] The R Core Team (2012): R: A Language and Environment for Statistical Computing (`www.R-project.org`)

[2] Lizenz: GPL