Dieses Tutorial soll Ihnen erlauben, sich innert kürzester Zeit ein minimales Basiswissen über *R* anzueignen, das Sie immer wieder benötigen werden.

R ist eine freie (GNU) Software und kann unentgeltlich vom Netz bezogen werden unter: `http://stat.ethz.ch/CRAN/`. Dort finden Sie auch eine ausführliche Einführung unter **Documentation**, `Manual`, "An Introduction to R" (ca. 100 Seiten als `pdf`) und etwas Kürzeres bei `Contributed`, "R for Beginners / R pour les débutants" (ca. 60 Seiten, englisch oder französisch).

R aufstarten

Die Computer in den Studentenräumen verfügen über eine Dualboot-Installation (Linux und Windows). Falls Sie sich auf einem Linux System befinden, starten Sie den Computer neu, und wählen Sie beim Aufstarten das Windows System. Melden Sie sich mit Ihrem Windows StudentInnen-Konto an und starten Sie *R* mittels *Start / All Programs / R / R 2.7.2* auf (eventuell ist bei Ihnen eine andere Version installiert).

Kreieren und Löschen von Objekten

Tippen Sie im entstandenen *R*-Fenster:
> x <- 2 (*und <ENTER> drücken*)
> x (*und <ENTER> drücken*)
Resultat: **[1] 2**
Sie haben mit dem Zuweisungsoperator **<-** ein Objekt **x** generiert. Da *R* vektorbasiert ist, ist **x** ein Vektor mit einem Element, das den Wert 2 hat. Mit der Eingabe von **x** kann man sich den Inhalt von **x** ausgeben lassen.

Weiterer Versuch (da am Ende jedes *R*-Befehls *<ENTER>* gedrückt werden muss, wird dies von nun an nicht mehr speziell angegeben):
> y <- c(3, 5) (c steht für concatenate (=verbinden))
> y
Resultat: **[1] 3 5**, ein Vektor mit zwei Elementen.

Achtung: Sie sollten für Vektoren und andere R-Objekte keine Namen wählen, die *R*-Befehle sind. Beispielsweise sind die Namen **c**, **t**, **T**, **F** und **max** "verboten".

Um anzuschauen, welche Objekte sie bereits erstellt haben, tippen Sie **ls()**. Um das Objekt **x** zu löschen, tippen Sie **rm(x)**. Um alle Objekte im Arbeitsspeicher zu löschen, verwenden Sie **rm(list=ls())**.

R-Demos

Eine Liste aller zur Verfügung stehenden Demos erhalten Sie mit dem Befehl **demo()**. Schauen Sie sich nun die Demo zu Grafiken in *R* an: **demo(graphics)**. Mit *<ENTER>* springen sie jeweils zum nächsten Bild.

Arbeiten mit einem .R (Script-)File

Es hat sich als sehr praktisch erwiesen, die Befehle nicht direkt ins *R*, sondern in einen Editor einzutippen. Von dort können sie danach ins *R* "geschickt" werden. Dieses Vorgehen ermöglicht einfaches Beheben von Tippfehlern und ein sinnvolles Abspeichern der Arbeit mit R. Die folgenden Anweisungen beziehen sich auf den Editor Tinn-R. Tinn-R ist eine freie GNU Software. Falls Sie Tinn-R auf Ihrem privaten Computer installieren möchten, können Sie das Programm unentgeltlich vom Netz beziehen unter: `http://sourceforge.net/projects/tinn-r/`. Auf den Rechnern in den Studentenräumen ist Tinn-R schon installiert.

Vor dem ersten Verwenden von Tinn-R müssen in *R* - falls nicht schon so eingestellt - folgende Änderungen vorgenommen werden: Menü *Bearbeiten –> GUI Einstellungen*. Dort muss *SDI* gewählt werden (anstatt *MDI*). Die Einstellung speichern (Klick auf *Save*). Im neuen Fenster nichts ändern, und die Einstellungen so wie vorgeschlagen (*My Documents\Rconsole*) abspeichern (Klick

auf *Save*). Danach den Rgui Konfigurationseditor durch Klick auf *OK* oder *Cancel* schliessen (allfällige Meldungen ignorieren) und *R* schliessen (Workspace nicht speichern).

Um nun mit *R* und Tinn-R zu arbeiten, muss zuerst *R* gestartet werden (*Start / All Programs / R / R 2.7.2*) . Danach wird Tinn-R gestartet (*Start / All Programs / Tinn-R / Tinn-R*). Das Starten von Tinn-R braucht etwas Zeit. Falls eine Fehlermeldung erscheint, einfach auf *OK* klicken. Die Tinn-R und *R* Fenster liegen übereinander. Das obere Fenster ist der Editor, das untere ist die *R*-Konsole.

Erzeugen Sie nun einen Ordner *RFiles* in ihrem Home directory (*wblstatXY* bzw. *My Computer\T:*), am besten im Windows Explorer mittels *File / New / Folder*.

Öffnen Sie eine neue Datei im Tinn-R (*File / New*). Tippen Sie auf die erste Zeile **z <- c(8, 13, 21)** und auf die nächste **2 * z**. Speichern Sie das File unter dem Namen *tutorial.R* im Ordner *RFiles*. Mit den Icons oben links (z. B. *Send all*, *Send selection* und *Send current line*) können Befehle direkt vom Editor in die *R*-Konsole geschickt werden. Markieren Sie die 2 Zeilen, die Sie geschrieben haben mit der linken Maustaste. Klicken Sie auf das Icon *Send selection*. Die Befehle werden in der *R*-Konsole ausgeführt.

Sie können natürlich auch direkt Befehle in der *R*-Konsole eintippen. Sie sollten aber möglichst alle *R*-Befehle ins *.R-File des Editors schreiben und von dort aus ausführen, da Sie diese Datei am Ende abspeichern können. Wenn Sie das nächste Mal mit ihrer Arbeit fortfahren wollen, öffnen Sie die Datei, markieren alle Befehle und führen diese aus (Icon *send selection* oder Icon *Send all* ohne Markierung). So sind sämtliche R-Objekte vom letzten Mal wieder vorhanden.

Tastenkombinationen

Als Alternative zu den Icons können Sie auch "Hotkeys" verwenden. Das sind Tastenkombinationen, die den selben Effekt haben wie ein Klick auf ein bestimmtes Icon. Solche Hotkeys müssen zuerst im Menü *R / Hotkeys of R* definiert werden.

Als Beispiel wird hier Alt-l definiert, um eine Zeile vom Editor ins *R* zu schicken. Im Fenster *Tinn-R hotkeys* (*R / Hotkeys of R*) wird in der Liste unter *Tinn-R functional hotKeys* "Send: line" angewählt. Dann wird bei *Get hotkey method* nur vor *Alt* ein Häkchen gesetzt (andere Häkchen entfernen). Im Feld *Key:* (bei *Get hotkey method*) L auswählen und auf *Add* klicken. Die gewählte Tastenkombination erscheint im Feld *Assigned hotkeys*. Nun muss man unter *Option* noch *Active* auswählen und dann das *Tinn-R hotkeys* Fenster mit einem Klick auf *OK* verlassen.

Setzen Sie den Cursor im Skript *tutorial.R* auf die Zeile, die Sie ausführen möchten. Verwenden Sie die Tastenkombination **Alt-l**. Der Computer wechselt zum *R*-Fenster, und der Befehl wird dort ausgeführt (genau wie wenn Sie auf das Icon *Send current line* geklickt hätten). Nun können Sie selbst entscheiden, ob Sie weitere Hotkeys definieren möchten, oder ob Sie lieber mit den Icons arbeiten.

In diesem Abschnitt lernen Sie noch zwei wichtige Tastenkombinationen kennen, die schon definiert sind. Wenn Sie gleichzeitig die **Ctrl**-Taste und das **Minus**zeichen **im Nummernblock** drücken, wird der **Zuweisungspfeil " <-"** erstellt.

Wenn Sie eine Funktion verwenden wollen, und nicht mehr genau wissen, wie sie funktioniert, können Sie mit der **Taste F1** Hilfe anfordern: Setzen Sie den Cursor auf den Funktionsnamen und drücken Sie F1. Es erscheint ein Fenster mit der **Hilfe zu dieser Funktion**.

Rechnen mit Vektoren

Wechseln Sie wieder zum Editor und geben Sie in der dritten Zeile von *tutorial.R* folgendes ein (ergibt die ersten 8 Fibonacci-Zahlen):
fib <- c(1, 1, 2, 3, 5, z)
Führen Sie den Befehl aus (Icon *Send current line* oder **Alt-l**), und schauen Sie sich **fib** im *R*-Fenster an. Tippen Sie dann **2 * fib + 1**, **fib * fib** und **log(fib)** auf die folgenden drei Zeilen in *tutorial.R*. Markieren Sie alle 3 Zeilen mit der linken Maustaste und führen Sie die Befehle aus (Icon *Send selection*). Schauen Sie sich die Resultate an und überlegen Sie sich, wie sie zustande gekommen sind.

Nun erzeugen Sie die Sequenz 2, 4, 6 als Objekt s: **s <- 2 * (1:3)**. Schauen Sie, was **fib[3]**, **fib[4:7]**, **fib[s]**, **fib[c(3, 5)]** und **fib[-c(3, 5)]** ergeben.

Bilden Sie einen Vektor **x** mit 8 Elementen, die teils positiv, teils negativ sind. Schauen Sie, was **x > 0** und **fib[x > 0]** ergeben.

Bilden von und Rechnen mit Matrizen

Bilden Sie zwei Vektoren **x <- 1:4** und **y <- 5:8** und die Matrizen **mat1 <- cbind(x, y)** und **mat2 <- rbind(x, y, x+y)**. (cbind steht für column-bind, rbind für row-bind.) Schauen Sie sich zuerst die ganzen Matrizen und dann auch folgendes an: **mat2[3,2]**, **mat2[2,]** und **mat2[,1]**.

Rechnen mit Matrizen folgt denselben Regeln wie das Rechnen mit Vektoren: Es wird elementweise gerechnet. Wenn Sie das übliche Matrixprodukt und nicht das elementweise Produkt benötigen, verwenden Sie %*%, z.B. **mat2 %*% mat1**.

Data Frames

Ein Data Frame ist eine verallgemeinerte Matrix. Der Hauptunterschied zwischen Data Frames und Matrizen ist, dass in Matrizen alle Elemente vom selben Typ (z.B. numeric, character) sein müssen, während in Data Frames jede Kolonne einen anderen Typ haben kann. Im allgemeinen liegen unsere Datensätze als Data Frames vor.

Einlesen und Anschauen von Datensätzen

ASCII-Dateien werden am einfachsten mit dem Befehl **read.table** eingelesen. **read.table** funktioniert auch für Datensätze vom Netz.

Zum Beispiel versuchen Sie:
no2 <- read.table("http://stat.ethz.ch/Teaching/Datasets/no2Basel.dat", header=TRUE)
Sie können das erzeugte Objekt direkt mittels **no2** anschauen. Auf einzelne Variablen lässt sich z.B. mittels **no2$NO2** zugreifen. **no2** ist noch einigermassen übersichtlich, aber es lohnt sich, ein Objekt zuerst mittels des Befehls **str** anzuschauen, der nicht alle Elemente des Objekts anzeigt, dafür aber seine Struktur und seinen Typ: **str(no2)**.
Nützliche Informationen über einzelne Spalten von **no2** erhalten Sie mit dem Befehl **summary(no2)**. Der Befehl **summary** liefert auch bei komplexen *R*-Objekten, wie z.B. dem Resultat eines statistischen Tests oder einer Regression, die nützlichen Informationen.

Wie die Datei im Original aussieht, können Sie sich anschauen, indem Sie die obige URL in einem Browser (Mozilla Firefox, Internet Explorer, etc.) eingeben.

Grafiken

Zeichnen Sie ein Histogramm der NO2-Werte im no2-Datensatz.
par(mfrow = c(1,2)) *# Anzahl Bilder unter- (1) bzw. nebeneinander (2); Papier sparen!*
hist(no2$NO2) *# Histogramm zeichnen*
Zeichnen Sie ein Streudiagramm, um den Zusammenhang zwischen NO2-Gehalt und Temperatur zu sehen.
plot(no2$Temp, no2$NO2)

Einen Titel fügen Sie zu Ihrer Grafik mittels **title("Titel xy")** hinzu. Klicken auf den Print-Button im *R*-Fenster druckt die Grafik aus.

Funktionen in *R*

R hat viele eingebaute Funktionen, welche einen Wert zurückgeben oder eine Figur zeichnen. Um den zurückgegebenen Wert weiter zu verwenden, muss er einem Objekt zugeordnet werden: Wenn wir z.B. $P[X = 10]$ für $X \sim Binom(20, 0.5)$ berechnen und im Objekt **wkeit** speichern wollen, so verwenden wir: **wkeit <- dbinom(10, 20, 0.5)**.

Die Argumente können entweder über die Position (wie oben) oder über ihren Namen identifiziert werden: **dbinom(size = 20, prob = 0.5, x = 10)** ergibt dasselbe wie **dbinom(10, 20, 0.5)**.

Die Namen der Argumente kann man mit **args** abfragen: **args(dbinom)**, bzw. alternativ in der Hilfe nachschlagen.

Oft haben Funktionen freiwillige Argumente, d.h. solche bei denen Defaultwerte vorgegeben sind, die zum Zuge kommen, wenn kein anderer Wert gesetzt wird. Was wird mit dem Aufruf von **dbinom(10, 20, 0.5, log = TRUE)** berechnet?

Wir betrachten nun die Binomialverteilung noch etwas genauer:
x <- 30:70
plot(x, dbinom(x, 100, 0.5))
plot(x, dbinom(x, 100, 0.5), type = "h")
plot(x, dbinom(x, 100, 0.5), type = "l")
("l" steht für **l**ine, der default für type ist "p" für **p**oints, "h" steht für **h**istogram-like)

n <- 1:100
plot(n, dbinom(n, 2*n, 0.5))
Was wird hier dargestellt? Kommentieren Sie, was Sie sehen.

Hilfe in *R* (Wichtig für später!)

Wenn Sie Details zu einem Befehl wissen wollen, z.B. zum oben verwendeten **plot**-Befehl, verwenden Sie die online-Hilfe von *R* : **help(plot)** oder **?plot**. Das Beispiel am Ende der Hilfeseite können Sie mittels **example(plot)** ausführen. (Klicken Sie auf die Grafik um die nächste zu sehen.)

Alternative zum **help**-Befehl: **help.start()** startet im Browser die html-Hilfe von *R*.

Wenn Sie zu einem Begriff Befehle suchen, deren Namen Sie noch nicht kennen, z.B. zu Histogramm, dann liefert **help.search("histogram")** eine Liste von Befehlen, die mit dem Begriff assoziiert sind. (In Klammern ist dabei jeweils das Package aufgeführt, in welchem der entsprechende Befehl vorkommt.) Wir verwenden vorerst vorallem das Package "base", das automatisch geladen wird. Andere Packages müssen zuerst mittels **library(*PackageName*)** geladen werden, bevor ihre Befehle verwendet werden können. Mit **library(help=*PackageName*)** erhält man eine Übersicht der im Package vorhandenen Funktionen.

R beenden

Speichern Sie das File *tutorial.R*. Wenn Sie das nächste Mal mit dem Tutorial fortfahren wollen, öffnen Sie im Editor *tutorial.R*, markieren den ganzen Inhalt des Files mit der linken Maustaste und schicken ihn ins *R*. Damit sind Sie wieder gleich weit wie zum Zeitpunkt, als Sie *R* beendeten. Dies wird vor allem nützlich sein, wenn Sie Übungen lösen und nicht in einem Mal fertig werden. Daher sollten Sie die Befehle von Übung 1 in ein File *uebung1.R* schreiben, die von Übung 2 in *uebung2.R* etc.

Mit dem Befehl **q()** beenden Sie die *R*-Session. Antworten Sie **No** auf die Frage **Workspace sichern?**