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

Melden Sie sich mit Ihrem Windows StudentInnen-Konto an und starten Sie *R* mittels *Start / All Programs / R / R 2.11.1* auf.

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 **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 für concatenate (=verbinden))
y
Resultat: **[1] 3 5**, ein Vektor mit zwei Elementen.

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

Um anzuschauen, welche Objekte sie bereits kreiert haben, tippen Sie **ls()**. Um das Objekt **x** zu löschen, tippen Sie **rm(x)**.

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 in *R*, sondern in einen Editor einzutippen. Von dort können sie danach zu *R* "geschickt" werden. Dieses Vorgehen ermöglicht das einfachere Beheben von Tippfehlern und ein sinnvolles Abspeichern der getanen Arbeit. Der Editor Tinn-R ist sehr geeignet für die Zusammenarbeit mit *R*.

Kreieren Sie zuerst einen Ordner *RFiles* im Folder "My Documents", am Besten im Windows Explorer mittels *File / New / Folder*.

Schliessen Sie *R* und starten Sie Tinn-R: *Start / All Programs / Tinn-R / Tinn-R* und klicken Sie auf das Symbol *R control:gui(start/close)*.

Öffnen Sie ein neues File in Tinn-R mit *File / New* und 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*. Sie können nun eine ganze Befehlssequenz aus dem Tinn-R direkt zu *R* schicken, indem Sie diese mit der linken Maustaste markieren und danach auf den Button *R send: selection (echo=TRUE)* klicken.

Rechnen mit Vektoren
Wechseln Sie zurück zu Tinn-R 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, und schauen Sie sich **fib** 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. Schauen Sie sich die Resultate an und überlegen Sie sich, wie sie zustande gekommen sind.

Von nun an sollten Sie (fast) alle *R*-Befehle ins *.R-File des Tinn-R schreiben und von dort aus ausführen. Dieses File können Sie am Ende speichern (mit dem Button **Save**). Wenn Sie das nächste mal mit ihrer Arbeit fortfahren wollen, öffnen Sie das File, markieren alle Befehle mit der linken Maustaste und führen Sie sie aus. Damit sind Sie wieder gleich weit wie zuvor.

Nun kreieren 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.

Kreieren 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 jenes mit Vektoren; es wird also elementenweise gerechnet. Wenn Sie das Matrizenprodukt und nicht das elementenweise Produkt wollen, 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 letzteren alle Elemente vom selben Typ (z.B. numeric, character) sein müssen, während in ersteren 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 sich das kreierte Objekt direkt mittels **no2** anschauen. Auf einzelne Variablen lässt sich z.B. mittels **no2$NO2** zugreifen. **no2** ist noch einigermassen übersichtlich, aber i. allg. lohnt es 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)**.
Interessante Informationen über die einzelnen 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] resp. nebeneindander [2]
                                  # wichtig zum Papier sparen!
hist(no2$NO2)                     # Histogramm zeichnen
```

Berechnen Sie nun die Regressionsgerade des NO2-Gehalts gegen die Temperatur und stellen Sie sie neben dem Histogramm grafisch dar:

```
lm.T <- lm(NO2 ~ Temp, data = no2)          # berechnet Regression
plot(no2$Temp, no2$NO2)
abline(lm.T, col = 4, lty = 2)              # col: Farbe; lty=2: gestrichelt
summary(lm.T)                               # Regressionsoutput (Details später)
```

Einen Titel fügen Sie zu Ihrer Grafik mittels **title("Titel xy")** hinzu, und das Klicken des Print-Buttons im *R*-Fenster druckt die Grafik aus.

Hilfe im *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)**. 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 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 die Library aufgeführt, in welcher der entsprechende Befehl vorkommt.) Wir verwenden vorerst vorallem die Library "base". Andere Libraries müssen zuerst mittels **library(Libname)** geladen werden, bevor ihre Befehle verwendet werden können.

***R* beenden**
Speichern Sie das File *tutorial.R*. Wenn Sie das nächste mal mit dem Tutorial fortfahren wollen, öffnen Sie in Tinn-R *tutorial.R*, markieren den ganzen Inhalt des Files mit der linken Maustaste und schicken ihn mit dem Button *Send Selection* ins *R*. Damit sind Sie wieder gleich weit wie zum Zeitpunkt, als Sie das *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 **Save workspace image?**

Weiteres Arbeiten mit *R*

R kann verwendet werden um komplexe Programme und Funktionen zu schreiben. Schauen Sie sich dazu zum Beispiel **help(for)** und **help(function)** an.