Einführung in R

Starten von R

Im **SR A** loggen Sie sich zunächst unter CentOS ein, öffnen eine Konsole und führen den Befehl `rdesktop -f zivtserv.uni-muenster.de` aus. Loggen Sie sich anschließend unter Windows ein, und starten Sie R *(Start → Programme → R)*.

In den **CIP-Pools** loggen Sie sich unter Windows wie gewohnt ein, und starten dann die Remotedesktopverbindung *(Startmenü → Programme → Zubehör)*. Geben Sie als Computer `zivtserv.uni-muenster.de` ein, und klicken Sie auf Verbinden. Es erscheint ein neues Anmeldefenster, in dem Sie sich wiederum mit Ihrem üblichen Benutzernamen und Kennwort einloggen. Wählen Sie als Domäne *(Anmelden an)* UNI-MUENSTER. Nun können Sie R direkt vom Desktop aus starten.

Die Software R ist frei verfügbar. Zur Installation auf Ihrem **privaten Computer** können Sie diese von `http://www.r-project.org/` herunterladen.

Abgabe

Programme werden als .R Dateien in sogenannten *R Skripten* gespeichert. Ein neues Skript erstellen Sie über *Datei → Neues Skript*. Es erscheint ein Eingabefenster, in dem Sie die Befehle eingeben können. Diese werden ausgeführt, indem Sie sie markieren und `Strg + R` drücken. Mit der Tastenkombination `Strg + S` können Sie ein Skript speichern und über *Datei → Öffne Skript* ein Skript laden.

Alle erstellten Plots speichern Sie bitte als (einzelne) PDF-Dateien ab (Grafik-Fenster auswählen, *Menü → Datei → Speichern als*). Das .R Skript und die Plots mailen Sie bitte bis zum normalen Abgabetermin an `matti.schneider@uni-muenster.de`

Grundlegende Bedienung

R bietet eine interaktive Umgebung, in der man mittels gewisser Befehle Daten direkt eingeben und analysieren kann. Eine ausführliche Befehlsübersicht finden Sie auf den Seiten des R-Praktikums: `http://wwwmath.uni-muenster.de/statistik/lehre/SS12/PrakStat/`

Hier wollen wir nur die grundlegenden Befehle und Strukturen vorstellen, die für die Aufgaben notwendig sind. Zusätzlich können Sie zu jedem Befehl `command` mit `?command` die R-Hilfe aufrufen.

Eine Variable `x` mit dem Wert 4 erstellen Sie mit `x <- 4`. Einen Vektor `v` mit den Einträgen $(4, 2, 1)$ erstellen Sie mit `v <- c(4,2,1)`. Den Vektor $(1, ..., n)$ erhalten Sie durch `1:n`. Ebenso können Sie jedes Ergebnis einer Rechnung einer Variablen zuordnen.

Dies ist insbesondere dann sinnvoll, wenn mit dem Ergebnis noch weitergerechnet werden soll. Alle definierten Variablen könnnen Sie sich mit `ls()` anzeigen lassen, durch die Eingabe des Variablennamens können Sie sich ihren Wert anzeigen lassen.

Logische Vektoren

Ein logischer Vektor ist ein Vektor, der aus den boolschen Werten `TRUE` und `FALSE` besteht, etwa `b <- c(TRUE,FALSE,TRUE)`. Ist `v` ein Vektor mit n Zahleneinträgen, so erhalten Sie mit `v>=10` einen logischen Vektor der Länge n, dessen i-ter Eintrag `TRUE` ist, falls der i-te Eintrag von `v` größer oder gleich 10 ist. Den Operator `>=` nennen wir *logischen Operator*, von denen es noch weitere gibt:

`==`	gleich	`!=`	ungleich
`<`	kleiner	`>`	größer
`<=`	kleiner gleich	`>=`	größer gleich

Vektorarithmetik

Die Arithmetik zweier **gleichlanger** Vektoren `v` und `w` funktioniert in R komponentenweise, d.h. der Befehl `v+w` liefert den Vektor $(v_1 + w_1, \ldots, v_n + w_n)$. Die Addition können Sie selbstverständlich durch eine andere Operation ersetzen, etwa die Division: `c(0,1,1,2,3) / 1:5` ergibt den Vektor $(0, 0.5, \frac{1}{3}, 0.5, 0.6)$.

Funktionen für Vektoren

Elementare Funktionen `f` (etwa sin, exp, Multiplikation mit 4, ...) können Sie auch auf einen Vektor v durch `f(v)` anwenden. Auch hier erfolgt die Auswertung komponentenweise. Darüber hinaus gibt es spezielle **Vektorenfunktionen**, wie z.B. `sum` oder `cumsum`. Wenden Sie diese auf einen Vektor `v` an, so erhalten Sie die Summe bzw. die kumulierte Summe der Einträge von `v`. Letzteres entspricht dem Vektor $(v_1, v_1 + v_2, \ldots, v_1 + \ldots + v_n)$. Beide Funktionen können Sie auch auf einen logischen Vektor anwenden. Bei der Rechnung wird `TRUE` auf 1 und `FALSE` auf 0 gesetzt.

Zufallszahlen für diskrete Verteilungen

n (Pseudo-)Zufallszahlen, die unabhängig gemäß der Verteilung $p = (p_1, ..., p_k)$ auf $\{1, \ldots, k\}$ gesampelt werden, erhalten Sie durch `sample(1:k, n, replace = TRUE, prob = p)`. Den Vektor `p` können Sie im Aufruf von `sample` weglassen, wenn er die Gleichverteilung auf $\{1, \ldots, k\}$ angibt. Das Argument `replace` erwartet einen boolschen Wert, der angibt, ob eine bereits gezogene Zahl ein weiteres Mal auftauchen kann (`replace = TRUE` entspricht dem Ziehen mit Zurücklegen), oder eben nicht (`replace = FALSE` entspricht dem Ziehen ohne Zurücklegen).

Plotten in R

Mit `plot(v)` können Sie die Punkte $(1, v_1), (2, v_2), \ldots, (n, v_n)$ in einen Plot zeichnen lassen. Dieser Befehl hat viele weitere optionale Argumente, wie etwa `type`, mit dem spezifiziert werden kann, ob die Werte als einzelne Punkte geplottet (`type="p"`), oder die Punkte miteinander verbunden werden sollen (`type="l"`). Außerdem gibt es die Möglichkeit die x- und y-Achse mit Hilfe der Argumente `xlab` und `ylab` zu beschriften, sowie eine Überschrift für den ganzen Plot durch das Setzen des Arguments `main` zu erstellen. Der Aufruf von `plot` sieht dann wie folgt aus:
`plot(v, type = "l", xlab="text", ylab="text", main="text")`.

In einen bestehenden Plot kann man nun noch weitere Punkte (mit der Funktion `points`) oder Linien (mit dem Befehl `abline`) einfügen. Beispielsweise zeichnet `abline(v=1)` eine vertikale Linie durch den Punkt $(1, 0)$ und `abline(h=1)` eine horizontale Linie durch den Punkt $(0, 1)$.

Schleifen

Schleifenprozesse sind Vorgänge, die vom Programm immer wiederholt werden, bis eine gewisse Bedingung erfüllt ist. Die wichtigsten Schleifen sind die `for`- und die `while`-Schleife. Die `for`-Schleife unterliegt folgender Syntax:
`for(Name in Vektor) { Körper der Schleife }`

Dadurch wird eine Variable, die „`Name`" heißt, schrittweise gleich den Elementen des Vektors „`Vektor`" gesetzt. In jedem Schritt wird für den entsprechenden Wert des Vektors die Befehle innerhalb der geschweiften Klammern ausgeführt. Die Befehle im Körper der Schleife werden also sooft ausgeführt, wie es Elemente im Vektor „`Vektor`" gibt

Will man einen Vorgang wiederholen, bei dem man nicht weiß, nach wievielen Schritten die Schleife abbrechen soll, hilft die `while` Schleife. Sie unterliegt der Syntax:
`while(Bedingung) { Körper der Schleife }`

Hier wird vor jedem Schleifen-Schritt die Bedingung überprüft: Solange diese `TRUE` ist, werden die Befehle ausgeführt. Tritt `FALSE` ein, wird die Schleife beendet.