INTRODUKTION TILL R

Niklas Edvall & Andreas Gerhardsson

R är en miniräknare

I grunden är R en miniräknare. Nedan demonstreras hur de grundläggande fyra räknesätten ger förväntat resultat. Notera att text som föregås av '#' är en kommentar och behandlas inte som kod av R. Det är god praxis att **alltid** kommentera sin kod.

```
#Summera
4 + 7
```

```
[1] 11
```

```
#Subtrahera
5 - 3
```

```
[1] 2
```

```
#Multiplicera
4 * 2
```

```
[1] 8
```

```
#Division
12 / 3
```

```
[1] 4
```

`[1]` innan vår output visar bara vilken rad vår output returneras på.

Variabler

Vad som gör R kraftfullt är att vi kan ange datapunkter i variabler och använda funktioner för att göra beräkningar på dem.

För att specificera en variabel används: <-

Så vill vi att en variabel kallad `a` ska representera värdet 7 skriver vi:

```
a <- 7
b <- 3

a + b
```

```
[1] 10
```

... och passar på att representera värdet 3 med variabeln `b` och räkna ut 7+3, som nu är representerade av våra två variabler `a` och `b` - som förväntat returneras 10 som output.

Funktioner

Nedan listas några grundläggande och ofta återkommande funktioner tillsammans med en kort beskrivande kommentar. Det är sällan vi behandlar enskilda datapunkter, därför är funktionen `c()` av särskilt intresse till en början. Den låter oss kombinera flera datapunkter i en variabel.

```
c() #Combine - kombinera flera datapunkter till en variabel

sum()  #Returnerar summan
mean() #Returnerar medelvärde
sd()   #Standard Deviation - returnerar standardavvikelse
min()  #Returnerar det minsta värdet
max()  #Returnrerar det maximala värdet
```

Notera att en funktion alltid följs av en parentes (...) inom vilken man anger input till funktionen.

Vi börjar med att använda `c()` för att skapa en variabel vi döper till `length.cm` innehållande flera observationer. Här är det längden i centimeter för 8 personer. Se kommentarer i koden nedan för hur vi använder andra funktioner.

Vi använder engelska fortsättningsvis för att å, ä, ö ibland kan ställa till det.

```
#Specify observations of length (cm)
length.cm <- c(183, 201, 165, 178, 169, 191, 174, 187)

#How tall is the shortest person?
min(length.cm)
```

```
[1] 165
```

```
#How tall is the tallest person?
max(length.cm)
```

```
[1] 201
```

```
#Calculate mean length
mean(length.cm)
```

```
[1] 181
```

```
#Calculate standard deviation of length
sd(length.cm)
```

```
[1] 11.94033
```

Vi kan använda `round()` om vi bara är intresserade av 2 decimaler för standardavvikelse.

Genom att skriva `?` före en funktion får man lätt tillgång till hjälpavsnittet för en funktion och `? round()` visar att vi anger `round(data, antal decimaler)` för att avrunda ett värde, alltså:

```
round(11.94033, 2)
```

```
[1] 11.94
```

Men man kan även använda funktioner **inuti** andra funktioner, så ännu bättre är att baka in `sd()` i `round()` och direkt ange:

```
round(sd(length.cm), 2)
```

```
[1] 11.94
```

Index

I vår variabel `length.cm` har varje datapunkt också ett index, eller en 'plats'. Det första värdet (183) har index 1, det andra värdet (201) har index 2 osv..

Med hjälp av hakparentes [...], eller *square brackets* på engelska, kan man komma åt ett specifikt värde i en variabel. Nedan kollar vi vad det fjärde värdet är i vår variabel `length.cm`

```
#The fourth value in variable "length.cm"
length.cm[4]
```

```
[1] 178
```

Boolean operators

För att undersöka om ett värde är större, mindre eller lika med ett annat används så kallade *boolean operators* som returnerar `FALSE` eller `TRUE` beroende på om uttrycket är sant eller falskt.

```
a < b  # Är a mindre än b?
a > b  # Är a större än b?
a == b # Är a lika med b?
```

Kombinera index och booleans

Att använda index för sin data är ofta smidigt. Tidigare använde vi funktionen `max()` för att se hur lång den längsta personen i `length.cm` är, men det kanske också är intressant att veta *vem* som är längst. För det kan vi använda en *boolean operator* tillsammans med funktionen `which()` för att ta reda på index för observationen som är lika med vår längsta observation/försöksperson i tre steg:

```
#How tall is the tallest person?
max(length.cm)
```

```
[1] 201
```

```r
#Who in "length.cm" is 201 cm tall?
length.cm == 201
```

```
[1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```r
#What is the index of the subjects that is 201 cm tall?
which(length.cm == 201)
```

```
[1] 2
```

Men det är ännu bättre att kombinera funktionerna direkt:

```r
which(length.cm == max(length.cm))
```

```
[1] 2
```

.. och vi ser att den andra observation/försöksperson är längst.

Data frames

Oftast så har vi mer än en variabel per försöksperson i våra studier. Nedan skapar vi en ny variabel `age` för ålder och kombinerar med vår tidigare variabel för längd till en så kallad *data frame* som vi kallar `df`

En data frame är det i absolut vanligaste data-formatet i R eftersom det kan innehålla flera olika typer av variabler samtidigt.

```r
#Create new variable "age"
age <- c(27,36,29,42,51,32,22,35)

#Combine variables to data frame "df"
df <- data.frame(age, length.cm)

#Show first columns and rows of "df"
head(df)
```

```
  age length.cm
1  27       183
2  36       201
3  29       165
4  42       178
5  51       169
6  32       191
```

Som innan kan man komma åt datapunkter i en data frame genom att ange ett index inom […], men eftersom en data frame har rader och kolumner (är 2-demensionell) behöver man ange både rad och kolumn enligt: `data[rad,kolumn]`

Till exempel:

```
#What is in row 3 in the 2nd column in data frame "df"?
df[3,2]
```

```
[1] 165
```

Om man inte specificerar en rad får man alla rader tillbaka:

```
#What is in the 2nd column in data frame "df"?
df[,2]
```

```
[1] 183 201 165 178 169 191 174 187
```

Men ännu bättre är att referera till kolumnens namn, snarare än index. Kolumnen får det namn variabeln hade när vi skapade vår data frame (se nedan).

Observera att columnens namn anges i text och specificeras därför inom citationstecken.

```
#What is in column "length.cm" in data frame "df" ?
df[,"length.cm"]
```

```
[1] 183 201 165 178 169 191 174 187
```

Eftersom man ofta vill komma åt olika kolumner i en data frame finns det en genväg för just det fallet. Med dollar-symbolen $ kommer man direkt till en kolumn i en data frame:

```
df$length.cm
```

```
[1] 183 201 165 178 169 191 174 187
```

Nu blir det väldigt enkelt att återkomma till att använda olika funktioner för att analysera data, t.ex om vi nu undrar om medellängd och medelålder:

```
#Mean of length
mean(df$length.cm)
```

```
[1] 181
```

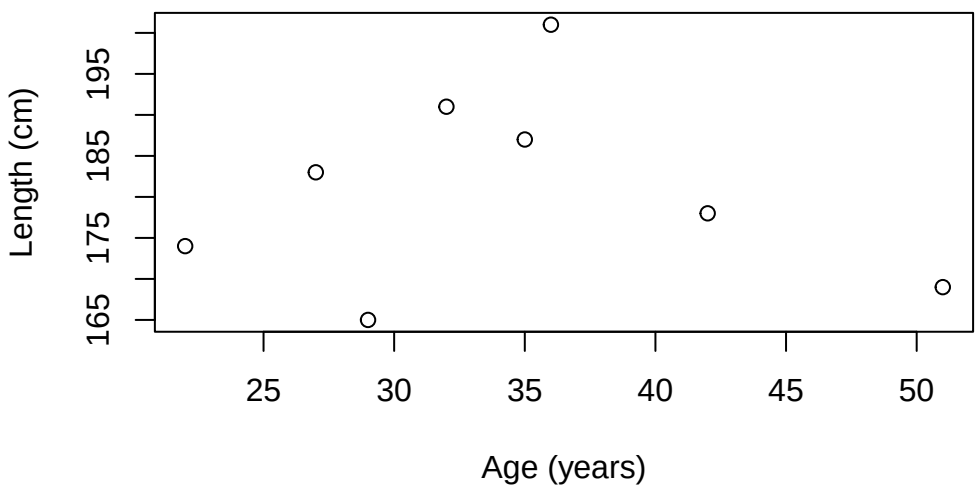
```
#Mean of age
mean(df$age)
```

```
[1] 34.25
```

Plot

Avslutningsvis använder vi funktionen `plot()` för att se om det finns något uppenbart samband mellan ålder och längd i vår data.

```r
#Scatter plot of length as function of age
plot(df$age, df$length.cm,
     xlab = "Age (years)",
     ylab = "Length (cm)")
```



Vilket det knappast ser ut som, vilket kanske är förväntat i och med att alla försökpersoner är vuxna.

Statistiska test

För att var på den säkra sidan kan vi göra ett statistiskt test för korrelation, t.ex ger funktionen `cor.tes()` oss resultatet från *Pearson's product-moment correlation,* vilken är enkel att tillämpa nu när vi vet hur man hanterar data i R.

```r
#Pearsons correlation test of age and length
cor.test(df$age, df$length.cm)
```

```
    Pearson's product-moment correlation

data:  df$age and df$length.cm
t = -0.16788, df = 6, p-value = 0.8722
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.7375141  0.6685068
sample estimates:
        cor
-0.06837721
```

I output ser vi att korrelationskofficienten är -0.068 med ett p-värde = 0.87. Ingen signifikant korrelation med andra ord.