0 Før kurset

Dette afsnit består af en oversigt over de R-ting kursusdeltagerne forventes at kunne før kurset. Der er kun få detaljer; der henvises i stedet til Sørensen (2015) og R-afsnittene i Ditlevsen and Sørensen (2011).

At arbejde med R og RStudio

Jeg regner med at du

- har installeret R og RStudio på din computer
- ved hvordan du arbejder med R og RStudio, fx skriver og gemmer R-programmer til senere brug

Lav en særskilt mappe på din computer til R-arbejde på Statistik 1.

Husk at R som udgangspunkt læser og skriver til det aktuelle „working directory“, altså at den leder efter filer og gemmer filer i denne mappe medmindre du angiver en anden sti. Working directory kan ændres for den nuværende R-session via Session menuen (Set Working Directory) eller med funktionen `setwd`. Det kan ændres permanent via menupunktet RStudio (vælg præferencer, og skriv/vælg den ønskede mappe i det øverste felt).

Referencer: Sørensen (2015), afsnit 1.

Indlæsning af data

Jeg regner med at du

- kan indlæse `.txt`-filer med R-funktionen `read.table`
- forstår strukturen af et datasæt (betydning af rækker og søjler)
- forstår forskellen på datasæt, variable, og observationer
- ved hvordan du kan bruge variablene i et datasæt ($-syntaksen, `with`, evt. `attach`). Afsnit 1 nedenfor omtaler også funktionen `with`

Referencer: Sørensen (2015), afsnit 3.

Vektorer

Jeg regner med at du

- ved at beregninger med vektorer foregår koordinatvis
- kan lave simple manipulationer med vektorer, fx lægge vektorer sammen og beregne funktioner på alle elementer af en vektor.

Referencer: Sørensen (2015), afsnit 2.

Grafik

Jeg regner med at du kan lave simple figurer. Mere specifikt:

- Scatterplots med R-funktionen `plot`
- Histogrammer med R-funktionen `hist`
- QQ-plots til sammenligning med normalfordelingen med R-funktionen `qqnorm`

Referencer: Sørensen (2015), afsnit 5.

Summary statistics

Jeg regner med at du kan beregne simple summary statistics, for eksempel ved hjælp af funktionerne `mean` (gennemsnit), `median` (median), `var` (stikprøvevarians), `sd` (stikprøvespredning).

Referencer: Sørensen (2015), afsnit 4.

En og to stikprøver

Jeg regner med at du

- kan bruge `pt` og `qt` til at beregne sandsynligheder og fraktiler i t-fordelingen (husk at angive antal frihedsgrader)
- kan bruge funktionen `t.test` til analyse af en og to stikprøver

Referencer: Ditlevsen and Sørensen (2011), afsnit 4.7 og 5.8.

Lineær regression med `lm`

Vi skal bruge `lm` meget i kurset, så der kommer mere om den senere i notatet. Men fra SS regner jeg med at du kan

- fitte en lineær regressionsmodel med `lm` og udtrække resultaterne med `summary`
- lave scatterplot af data og indtegne den fittede regressionsline
- udføre modelkontrol ved hjælp af residualerne (brug funktionerne `fitted` og `rstandard` til fittede værdier og standardiserede residualer)

Referencer: Ditlevsen and Sørensen (2011), afsnit 6.9.

1 Kursusuge 1

Transformation og delmængder af datasæt (data frames)

Det er som hovedregel mest hensigtsmæssigt at foretage så mange datamanipulationer (transformation, udtræk af deldatasæt mm.) i selve datasættet, snarere end "udenfor". Så er det nemmere at holde styr på de forskellige variable, og at undgå at have flere versioner af de samme variable.

Betragt i det følgende legetøjsdatasættet `myData` med fire variable (fra start):

```
> myData
      x    y z w
1  4.91 4.48 1 A
2  5.79 4.61 1 B
3  4.33 4.04 1 A
4  6.36 4.98 1 B
5  6.51 5.43 1 A
6  2.39 3.17 2 B
7  5.69 4.34 2 A
8  5.32 4.38 2 B
9  7.02 5.12 2 A
10 3.58 3.54 2 B
```

Tranformation Transformation af variable i et datasæt kan foretages "direkte" eller ved hjælp af funktionen `transform`. Hvis du fx vil lave to nye variable med kvadratrods- og logaritmetransformerede værdier af `x` henholdsvis `y`, kan det gøres således

```
myData$sqrt.x <- sqrt(myData$x)
myData$logy <- log(myData$y)
```

eller således

```
myData <- transform(myData, sqrt.x=sqrt(x), logy=log(y))
```

I begge tilfælde er resultatet at datasættet `myData` nu har seks variable, nemlig de fire oprindelige samt de to nye:

```
> myData
      x    y z w   sqrt.x     logy
1  4.91 4.48 1 A 2.215852 1.499623
2  5.79 4.61 1 B 2.406242 1.528228
3  4.33 4.04 1 A 2.080865 1.396245
4  6.36 4.98 1 B 2.521904 1.605430
5  6.51 5.43 1 A 2.551470 1.691939
6  2.39 3.17 2 B 1.545962 1.153732
7  5.69 4.34 2 A 2.385372 1.467874
8  5.32 4.38 2 B 2.306513 1.477049
9  7.02 5.12 2 A 2.649528 1.633154
10 3.58 3.54 2 B 1.892089 1.264127
```

Delmængder af datasæt Delmængder af datasæt konstrueres nemt med funktionen `subset`. Her er nogle eksempler:

```
> subset(myData, x<5)               # Udvælger datalinier baseret på x
      x    y z w   sqrt.x     logy
1  4.91 4.48 1 A 2.215852 1.499623
3  4.33 4.04 1 A 2.080865 1.396245
6  2.39 3.17 2 B 1.545962 1.153732
10 3.58 3.54 2 B 1.892089 1.264127

> subset(myData, w=="A")            # Bemærk == og anførselstegn
     x    y z w   sqrt.x     logy
1 4.91 4.48 1 A 2.215852 1.499623
3 4.33 4.04 1 A 2.080865 1.396245
5 6.51 5.43 1 A 2.551470 1.691939
7 5.69 4.34 2 A 2.385372 1.467874
9 7.02 5.12 2 A 2.649528 1.633154

> subset(myData, x<5 & w=="A")      # & er logisk 'og'
     x    y z w   sqrt.x     logy
1 4.91 4.48 1 A 2.215852 1.499623
3 4.33 4.04 1 A 2.080865 1.396245

> subset(myData, x<5 | w=="A")      # | er logisk 'eller'
      x    y z w   sqrt.x     logy
1  4.91 4.48 1 A 2.215852 1.499623
3  4.33 4.04 1 A 2.080865 1.396245
5  6.51 5.43 1 A 2.551470 1.691939
6  2.39 3.17 2 B 1.545962 1.153732
7  5.69 4.34 2 A 2.385372 1.467874
9  7.02 5.12 2 A 2.649528 1.633154
10 3.58 3.54 2 B 1.892089 1.264127

> subset(myData, x<5 & w=="A", select=c(x,w))  # select udvælger variable
     x w
1 4.91 A
3 4.33 A

> subset(myData, x<5 & w=="A", select=-c(x,w))
     y z   sqrt.x     logy
1 4.48 1 2.215852 1.499623
3 4.04 1 2.080865 1.396245
```

Enkelte datalinier kan nemt udtrækkes eller fjernes vha. [ ]-syntaksen:

```
> myData[3,]
     x    y z w   sqrt.x     logy
3 4.33 4.04 1 A 2.080865 1.396245

> myData[2:4,]
     x    y z w   sqrt.x     logy
2 5.79 4.61 1 B 2.406242 1.528228
3 4.33 4.04 1 A 2.080865 1.396245
4 6.36 4.98 1 B 2.521904 1.605430

> myData[-c(1,4,7,8,9),]
```

```
      x    y z w   sqrt.x     logy
2  5.79 4.61 1 B 2.406242 1.528228
3  4.33 4.04 1 A 2.080865 1.396245
5  6.51 5.43 1 A 2.551470 1.691939
6  2.39 3.17 2 B 1.545962 1.153732
10 3.58 3.54 2 B 1.892089 1.264127
```

Brug `with` hvis du vil undgå at bruge `attach`

Det kan være lidt farlige at "attache" datasæt, især hvis man har flere datasæt med samme variabelnavne. Så er det nemlig ikke altid så nemt at gennemskue hvilken version R bruger hvornår.

Sommetider kan `with`-funktionen være nyttig. For eksempel:

```
> mean(x)                # R ved ikke at den skal kigge i myData efter x
Error in mean(x) : object 'x' not found
> with(myData, mean(x))  # Nu ved R hvor den skal lede...
[1] 5.19
```

Statistisk analyse med `lm`

Vi kommer til at bruge `lm` rigtigt meget den første halvdel af kurset, nemlig til estimation af de såkaldte lineære normalfordelingsmodeller. Lige nu er det vigtige følgende:

- Et kald til `lm` laver et modelobjekt, der indeholder alskens information om den fittede model.
- Man kan bruge diverse funktioner der udtrækker nyttig information om dette objekt, fx `summary`, `confint`, `fitted`, `residuals`. Der er mange andre, men det kan vi komme tibage til.
- Outputtet fra `summary` er særligt vigtigt, især den del der står under `Coefficients`. I denne del er der en linie for hver parameter i modellen. Alle tal i linien vedrører den samme parameter og angiver estimat, standard error (estimeret spredning på estimatet) samt til sidst t-teststørrelsen og p-værdien for hypotesen om at den pågældende parameter er 0. Hypoteseværdien for parameteren er altid 0 — uanset om dette er en interessant hypotese eller ej! Vi snakker om hypotesetest i kursusuge 3 (og senere).

  Det er altafgørende at være i stand til at fortolke parametrene i modellen (og dermed estimaterne). Det skal vi lege lidt med i opgave HS.2–HS.4 i kursusuge 1, og vi vender tilbage til det senere i kurset.

Simulation

Simulation er et ekstremt nyttigt redskab til at undersøge fordelingsegenskaber ved stokastiske variable, fx hvis man ikke kan regne matematisk på fordelingerne. Princippet er at man simulerer et stort antal uafhængige udfald af den stokastiske variabel, og undersøger disse empirisk. Det er nemt at simulere data i R. Alle standardfordelinger (og også en masse ikke-standardfordelinger) er implementeret, således at man kan simulere fra dem. For eksempel:

```
> rnorm(5)                # 5 udfald fra N(0,1)
[1] -0.25473888  0.30563785  2.23979118 -0.09413188  0.56692140

> rnorm(6, mean=4, sd=.5) # 6 udfald af N(4,0.25). Bemærk at sd er spredning
[1] 3.780646 4.522978 4.146555 3.943649 4.523744 4.352841

> runif(4)                # 4 udfald fra ligefordelingen på (0,1)
[1] 0.70379046 0.49022381 0.06773676 0.13687836

> rpois(10, lambda=5)     # 10 udfald fra Poissonford. med middelværdi 5
 [1]  3  6  3  8  2  2  9  2 10  3
```

Løkker

I forbindelse med simulationsstudier vil man typisk foretage en række beregninger et stort antal gange og gemme resultatet fra hver beregning. `for`-løkker er nyttige til dette. Løkker nemme at forstå og kode! Derfor vil vi benytte dem selvom man ofte kan lave mere effektive implementationer.

Den typiske konstruktion fremgår af eksemplet nedenfor. Bemærk specielt at den variabel som bruges til at lægge resultatet i, skal defineres før den bruges (initialiering). I hver gentagelse af løkken lægges en ny værdi ind i variablen, og efter løkken kan værdierne undersøges nærmere.

```
> gennemsnit <- rep(NA, 10)   # Initialisering, fx med manglende værdier
> gennemsnit
[1] NA NA NA NA NA NA NA NA NA NA

> for (i in 1:10)
  {
    x <- rpois(20, lambda=7)    # Simulation af 20 Po iss(7)-udfald
    gennemsnit[i] <- mean(x)    # Gennemsnit af de 20 udfald lægges i variabel
  }

> gennemsnit                  # Indeholder nu de 10 gennemsnitværdier
 [1] 7.30 6.80 6.85 7.85 7.40 8.40 5.70 6.80 7.55 7.30
```