Øvelse 1

Import af data, databehandling og grafer

Underviser: Troels Ankjær
Hjælpelærer: Nicklas Larsen

Introduktion til programmering i R
4 timers kursus i R
Oktober 2019

VIDEN DER STYRKER

Start (læs grundigt)

Du skal starte med at hente R og Rstudio hvis du ikke har gjort det, det gør du via disse links:
`http://www.r-project.org/`
`http://www.rstudio.com/`

I skal først installere R og derefter RStudio. RStudio er kun en brugerflade til R. Der findes flere, men dette er den mest anvendte. Opgaverne kan løses uden brugerflade men de er lavet til Rstudio. Hvis du bruger en anden brugerflade kan alle opgaver laves, men det er ikke sikkert du kan følge vejledningen 100%.

Når R og RStudio er hentet skal du hentet dataarket "husdata.csv"som du kan find her: `https://bit.ly/2oSdwwh`. Gem denne i en mappe som du kalder "Arbejdsmappe".

Åben RStudio og lav et ny fil som du gemmer i den nye "Arbejdsmappe". Opgaverne er lavet så det kan laves i et enkelt script (dvs. på i en fil). Hver gang der skal laves en ny beregning skal der skriftes linje. Når der står at en linje skal køres så menes der at du skal have markøren på linjen og trykke crtl+enter ved Windows eller cmd+enter ved Mac.
Formateringen i øvelserne er som følger:

Formatering:	Eksempel:
R kode	funktioner **dnorm** eller grafer **plot**
Indhold i datasæt og parametre	*huspriser* eller *x*

Vi starter med at definere hvor vores arbejdsmappe (working dictionary) er placeret. Hvis du skriver **getwd**() (get working dictionary) og tryk ctrl+enter kan du se hvor R som standard har defineret placeringen i output vinduet (consolen).

Vi vil gerne have ændret dette så den arbejder ud fra din "Arbejdsmappe". Gå op under **Session**, og så gå ind under **Set working Directory**. Vælg så **Choose Directory**. Nu kan i selv finde mappen på jeres computer og tryk Open. Den viser jer **setwd** i output vinduet (consolen). Det er en fordel og kopier dette og gemme i jeresi toppen af jeres script

Tjek med **getwd**() i output vinduet om den er blevet ændret.

NB: Hvis du vil skrive noter i arket så du kan huske hvad tingene gør så skal du skrive # i starten af din note. Dette sørger for at den ikke bliver læst som kode.

1. Opgave - Simple beregninger

Start med at skrive følgende beregninger i et R Script:

3 + 4

240 * 124

2123 / 124 * 234

3 / (5 + 6)

Herefter tryk ctrl+enter (cmd+enter) ved hver enkelt beregning for at køre den. Nu kommer resultatet frem en efter en i output vinduet (console) nederst. Fremadrette når der står kør det menes der ctrl+enter (cmd+enter). Ved at trykke ctrl+alt+R(alt+cmd+R) køres hele scriptet. Prøv det.

For at kunne gemme tallene skal det gøres igennem en parameter. Start med at skrive

$x < -25.$

Nu er 25 gemt i x. Det kommer dog ikke frem i output vinduet (console) som før. Prøv at skrive x i output vinduet (console) og tryk enter. Nu skulle den gerne vise indholdet af x.

Lav et parameter som hedder y som er lig 30. Gang derefter y og x med hinanden og gem det i z. Vis hvad z giver i console vinduet.

2. Opgave - Arrays

Der findes mange måder og lave Arrays på, den mest simple er selv at skrive helt præcis hvilke værdier og i hvilken rækkefølge. Dette bygger videre på den metode vi brugte til at lave et parameter fra foregående opgave. Start med at skrive:

$a < -\mathbf{c}(3,7,9,11).$

Kør a og se resultat i output vinduet. Hvis dette skal gøres med strings skal det skrives:

$q < -\mathbf{c}(\text{"}a\text{"},\text{"}b\text{"},\text{"}c\text{"},\text{"}d\text{"})$

Der findes nogle hurtigere metoder til at lave arrays på. Hvis man gerne vil have et array som går fra 1 til 10. Skrives $b < -1:10$. Hvis den samme værdi skal gentages flere gange kan funktionen rep bruges. Skriv $c < -\mathbf{rep}(1, \mathbf{times} = 5)$.

Det er også muligt at samle flere arrays eller parameter i et nyt array. Det gøres ved at skrive
$d <- \mathbf{c}(a,x)$.

Nu er der lavet et nyt array d med værdierne fra a og parameteret x. Prøv at lav et array som begynder med 1 til 10 og derefter tre 7 tal, og til sidst 10 til 1. Gør det vha. ovennævnte funktioner og ikke ved bare at taste tallene ind.

3. Opgave - Indbygget funktioner

Der er flere indbygget funktioner i R, og vi kommer til at bruge flere af dem senere i øvelse 2 om Statistiske fordelinger. Vi starter med at finde maksimum i et array. Vi bruger a array fra sidste opgave. Skriv $\mathbf{max}(a)$.

Andre indbygget
Maksimum: $\mathbf{max}(a)$
Minimum: $\mathbf{min}(a)$
Gennemsnit: $\mathbf{mean}(a)$
Median: $\mathbf{median}(a)$
Summen: $\mathbf{sum}(a)$
Standardafvigelsen: $\mathbf{sd}(a)$

Hvad er summen af a? Hvad er medianen af a?

4. Opgave - Matrice

For at lave en matrice, skal man bruge funktionen **matrix**. Det første matricen skal bruge er et array med dataelementer. Derefter skal det fortælles hvordan de data elementer skal deles op. Det gøres ved at give den antallet af rækker og kolonner. Skriv

$A = \mathbf{matrix}(\mathbf{c}(2,4,3,1,5,7), \mathbf{nrow} = 2, \mathbf{ncol} = 3)$.

Kør A og vis den i output vinduet. Læg mærke til at den fylder op ved at starte med første kolonne, så anden kolonne og så videre. Prøv at bytte om sætte 3 rækker og 2 kolonner i stedet og se hvordan den så laver den.

Produktet af rækker og kolonner (dvs. $2 \times 3 = 6$ i ovenstående tilfæle) skal altid være lig antallet af elementer i det array du skriver i matricen. Prøv at fjerne 7 tallet i matricen og kør A igen, og se at den melder fejl. Ret fejlen tilbage igen.

For at finde et særligt element i en matrice, kan man søge i efter række og kolonne. Hvis man gerne vil finde det element som er i 2 række, 3 kolonne skal man skrive $A[2,3]$. Vil man derimod have en hel række skrives $A[2,]$. Hvad er hele anden kolonne?

Gem anden kolonne i *f*. Lav en matrice *P* med 6 rækker og 3 kolonner, hvor første kolonne går fra 1 til 6, anden kolonne går fra 6 til 1 og den sidste er med rene 2-taller. Hvad er $P[2,3]$?

5. Opgave - Importere data

I starten fortalte vi R hvor vores "Arbejdsmappe" er placeret og det er her vi har placeret det data vi vil analysere. Det gør at det er nemt at importere det. Start med at skriv:

data $<-$ **read.table**("husdata.csv").

Prøv nu at skriv **View**(*data*). Separationen af værdierne ser ikke helt rigtig ud. Vi skal derfor have defineret en separator til at være ';'. Dette gøres i den linje du allerede har skrevet, så den kommer til at se således ud

data $<-$ **read.table**("husdata.csv",sep=";").

Kør nu **View**(data) igen. Hvis der stadig er problemer med separationen så kan det være at dit Excel bruger et andet tegn end ';', prøv ','

Denne gang har den importeret det hele. Der er dog stadig et problem for værdikolonnerne hedder V1, V2, osv. Derfor skal vi fortælle at første linje af data er overskrifter (header). Vi skal derfor tilføje mere til samme linje.

data $<-$ **read.table**("husdata.csv", header=TRUE,sep=";")

Kør nu endnu en gang **View**(*data*). Nu er alt som det skal være og du har et datasæt som er brugervenligt for R og overskueligt for dig selv.

Prøv og import data som kan hentes her og kald det dioxin: `http://https://bit.ly/2W2QwGZ/`

6. Opgave - Data oversigt

For at se værdierne *data* er der flere muligheder for at få et hurtigt overblik. Prøv og skriv *data* og kør linjen. Nu udskriver den alle række der er i *data*. Det er dog ikke særligt godt til et give et overblik. Denne gang skriv **summary**(*data*) og kør linjen. Nu viser den bl.a. gennemsnits-, maksimums- og minimumsværdierne for hver kolonne. Hvad er gennemsnittet for kolonnen med *huspriser*?

Andre gode funktioner:
str(*data*) - Giver et overblik over stukturen
head(*data*) - Returnere de 6 første rækker for alle kolonner
tail(*data*) - Returnere de 6 sidste rækker for alle kolonner

7. Opgave - Data variabler

Nu vil vi gerne bare se på en enkelt værdikolonne. Start med bare at skrive *huspriser* og kør linjen. Nu kommer der en fejl. Skriv nu i stedet *data*$*huspriser* og kør linjen. Nu kommer alle tal i kolonnen *huspriser*. Hver gang man skal bruge en værdikolonne fra *data* skal man skrive hvor det skal hentes fra. Det kan man slippe for hvis man skriver **attach**(*data*). Prøv nu at skriv *huspriser*. Nu kommer alle række frem igen.

I opgaven Simple beregninger lærte du at gemme du et tal i en parameter. Nu vil skal du gemme *huspriserne* i parameteren *p*. Det gør du ved at skrive $p <- huspriser$. Nu er hele kolonnen *huspriser* kopieret til *p*. Den eksistere stadig i huspriser. Prøv at vise *p*.

Gem nu kolonnen med år i parameteren *a* (dette overskriver din gamle definition af *a*).

7.1 Subset

Subset er en nem måde hvor på man kan udtrække data fra et større datasæt. Det kan være at vælge et interval eller for en faktor. Subset skal først bruge hvilke data den skal tage fra og efterfølgende hvilket betingelse. Brug det data i kaldte dioxin. Her laver i betingelse at den skal hente alt ud fra LAB 2.

datalab2 <− **subset**(dioxin,*LAB=="2"*)

Hvis det gav fejl kan det være fordi det ikke er en string LAB, men en numerisk værdi. Her skal i ændre input i funktionen eller lave det om til en string i data. Hvis det kun er nogen af kolonnerne i vil se på kan i bruge select bagefter. Her kan i give den det array med kolonner i gerne vil se.

datalab2 <− **subset**(dioxin,*LAB=="2",select=c(LAB, OXYGEB,QRAT, H2O)*)

Hvis man gerne vil have alle kolonner imellem to så kan man istedet skrive.

datalab2 <− **subset**(dioxin,*LAB=="2",LAB:H2O)*)

For at sætte nogen begrænsninger på nogen af de numeriske værdier, skal man sætte logiske sætninger op. Det kunne være sige at H2O skal være under 600.

datah2o <− **subset**(dioxin,$H2O < 600$,*LAB:H2O)*)

Det kan også sættes sammen af flere ved at sætte & imellem. Det gør at alle betingelser skal være opfyldt for at de kommer med i udtrækker. Det kunne være det skal være LAB 1 og CO2 skal være større end -10.

datah2o <− **subset**(dioxin,*H2O < 600 & LAB=="1"& CO2 > -10*, LAB:H2O))

8. Opgave - Grafer

Vi starter med at plotte huspriserne over år. Dette gøres ved at skrive **plot**(*p*). Vi får nu et plot over alle datapunkterne i forhold til et index (1,2,3,...). Tilføj nu året ved at skrive **plot**(*a,p*).

Du kan ændre teksterne på akserne ved at tilføje kommandoerne i **plot** vist nedenfor.

plot(*a*,*p*,**xlab**="Årstal", **ylab**="Huspriser (mio. kr.)")

For at ændre punkterne på grafen, så skal der tilføjes kommandoen i **type** som vist nedenfor.

plot(*a*,*p*,**xlab**="Årstal", **ylab**="Huspriser (mio. kr.)",**type**="h")

For at se flere muligheder for **type** så skriv **?plot**. For at ændre farven skal endnu en kommando tilføjes. Denne gang er **col**, som skal sættes til en farve.

plot(*a*,*p*,**xlab**="Årstal", **ylab**="Huspriser (mio. kr.)",**type**="h", **col**="red")

Det sidste der mangler for at gøre grafen færdig, vil være en passende overskrift. Det gøres ved hjælp af kommandoen **main**.

plot(*a*,*p*,**xlab**="Årstal", **ylab**="Huspriser (mio. kr.)",**type**="h", **col**="red",**main**="Boligpriser over tid")

Lav et plot over *befolk* i forhold til år og giv dem sigende tekster på akserne, grafen skal vise med linjer og farven skal være grøn.

For at få flere grafer vist samtidige skriv **par**(mfrow = c(1, 2)) over linjerne med **plot** kommandoerne. Prøv at vise de to plot du har lavet i samme vindue. Fra nu af vil den vise to grafer, hvis det skal laves om til en igen skriv **par**(mfrow = c(1, 1)) efterfølgende.

Du kan gemme grafen ved at trykke på 'Export' ovenover grafen. Her kan du gemme den som .pdf eller kopier den så du kan sætte den ind i i eksempelvis Word.

Pairs

For at få et overblik over hvordan alle variable er i forhold til hinanden kan **pairs** brugers. Prøv at skriv:

pairs(data)

Zoom ind på billedet for at kunne få et bedre overblik. Selvom der bliver zoom ind kan det stadig godt være uoverskueligt. For at gøre det bedre kan man vælge hvilke variable man gerne sammenligne.

pairs(*aar* $\sim$ *huspriser* + *cpi*+*indk*+*ci6*,**data**=data)

For at få hjælp til at se tendensen kan der laves en hjælpe linje ved hjælp af **panel=panel.smooth**

pairs(*aar* $\sim$ *huspriser* + *cpi*+*indk*+*ci6*,**data**=data, **panel**=panel.smooth)

Funktion

I de to forgående eksempler har i lavet grafer ud fra den givet data. For at lave det ud fra en funktion, skal selve funktionen skrives op først. Det gøres ved:

$y <-$**function**(x){5***cos**(2*x+3)}

Nu skal skal den plottes, og her skal plot funktionen bruge hvilken funktion vores y, og i hvilket interval. Da det er en cos funktion bruger vi $2 * \pi$

plot(*y*,*0*,*2*pi*)

Lav en funktion for $g < -2x^3 + 3x^2 + 20$ og plot den med et passende interval.