Seminar 1

Martin Søyland

Seminaropplegg

Seminarrekka

Dato	Tid	Aktivitet	Sted
ma. 28. aug.	10:15–12:00	Seminar i databehandling R 2	ES PC-stue 351
ma. 11. sep.	10:15–12:00	Seminar i databehandling R 2	ES PC-stue 351
to. 14. sep.	**08:15–10:00**	**Seminar i databehandling R 2**	**ES PC-stue 351**
ma. 18. sep.	10:15–12:00	Seminar i databehandling R 2	ES PC-stue 351
ma. 25. sep.	10:15–12:00	Seminar i databehandling R 2	ES PC-stue 351
ma. 2. okt.	10:15–12:00	Seminar i databehandling R 2	ES PC-stue 351
ma. 9. okt.	**10:15–12:00**	**Obligatorisk prøve**	**ES PC-stue 351**

Semestersidene.

Egen datamaskin

I selve seminarene har dere valget mellom å bruke maskinene som står i seminarrommet via deres UiO-bruker (R er forhåndsinstallert på alle maskinene) eller dere kan bruke egen bærbar PC/Mac. Hvis dere bruker egen maskin forventer jeg at dere har installert både R og RStudio før det første seminaret.

Slack

Jeg har satt opp et Slack-team (slack.com) som dere kan bruke gjennom seminarrekka. Hensikten med dette er å tilrettelegge for en platform der dere kan sette frem problemer dere støter på med R (eller med oppgaver til teori-seminarene og spørsmål angående forelesningene) med hverandre, samarbeide om å løse disse, og at spørsmål bare jeg kan svare på blir åpne for alle. Slack er også et verktøy som blir brukt mer og mer i både arbeidsliv og forskning, så det kan være god verdi i å få kjennskap til hvordan det fungerer.

Dere får en link på UiO-mailen med

Linker

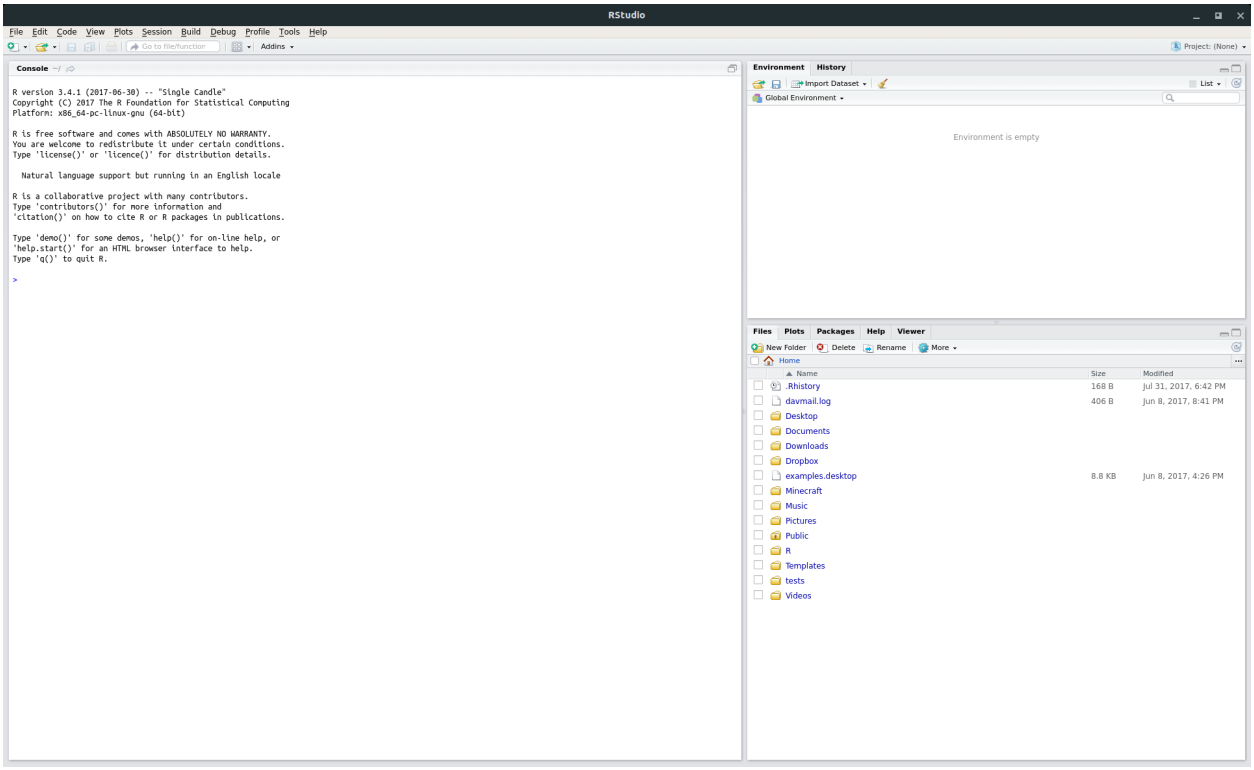
- Last ned R
- Last ned Rstudio
- R cheat sheet

- Stilguide for R
- Mappestruktureringsforslag
- Bruke prosjekter i R
- Guide til ggplot2
- Data fra Fivethirtyeight
- Facebookgruppe for R

Grunnleggende (repetisjon fra STV1020)

For de som starter helt fra scratch med R vil jeg anbefale å gå gjennom materiellet fra STV1020.

I dette dokumentet skal vi gå gjennom det mest grunnleggende med R. Det første man må gjøre er å installere R og Rstudio (den siste er valgfri, men sterkt anbefalt). Du skal da få opp et vidu som ser ut som dette:



Script og konsoll

På bildet over viser tre vindu:

1. Til venstre viser console. Dette er motoren til R; det er her vi får ut kodene vi skal kjøre.
2. Oppe til høyre har vi Environment. Her kommer *objektene* vi lagrer
3. Nede til høyre er filene i mappen vi jobber fra. Her er det også tabs for plot (figurer), pakker og hjelp

Men vi mangler ett vindu; vinduet der vi skriver inn koden har ikke kommet opp enda. For å få dette opp trykker vi på arket helt overst til venstre (under "File") og så på "R Script" fra rullegardinmenyen (eventuelt

kan man bruke hurtigtasten Ctrl/Cmd + Shift + N). Vinduet som da dukker opp er det vi skal tilbringe mest tid i; det er her vi skriver den **reproduserbare** koden som R skal gjøre om til outputen vi vil ha. For å sende en linje med kode fra **script** til **console** kan vi bruke knappen *run* øverst til høyre i script-vinduet, eller bare trykke Ctrl / Cmd + Enter når markøren står på linjen vi vil sende.

R som kalkulator

R (og de fleste andre programmeringsspråk) er enkelt sagt veldig avanserte kalkulatorer:

```
1 + 2  # Dette er linjen med kode jeg sender til console.
```

```
## [1] 3
```

```
# Linjen over er det console gir meg som svar

3 * 4
```

```
## [1] 12
```

```
10/2
```

```
## [1] 5
```

```
4 * (10/1.5)
```

```
## [1] 26.66667
```

```
# De to neste linjene er sannhetsevalueringer; er det på venstresiden de samme som det på
# høyresiden av likhetstegnene?

2 * 2 == 4
```

```
## [1] TRUE
```

```
2 * 2 == 5
```

```
## [1] FALSE
```

Indeksering, vektorer og objekter

Legg merke til at det kommer opp [1] før outputen vi forventer. Dette er R sin måte å vise til en verdis **indeks** i en rekke med tall. Alle linjene over har kun 1 verdi som svar. Men det er veldig sjelden vi opererer med bare 1 verdi om gangen; vi bruker da heller **vektorer** av verdier:

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
2 * 1:10
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
1:10/1:10
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1
```

```
100.5:0.5
```

```
##   [1] 100.5  99.5  98.5  97.5  96.5  95.5  94.5  93.5  92.5  91.5  90.5
##  [12]  89.5  88.5  87.5  86.5  85.5  84.5  83.5  82.5  81.5  80.5  79.5
##  [23]  78.5  77.5  76.5  75.5  74.5  73.5  72.5  71.5  70.5  69.5  68.5
##  [34]  67.5  66.5  65.5  64.5  63.5  62.5  61.5  60.5  59.5  58.5  57.5
##  [45]  56.5  55.5  54.5  53.5  52.5  51.5  50.5  49.5  48.5  47.5  46.5
##  [56]  45.5  44.5  43.5  42.5  41.5  40.5  39.5  38.5  37.5  36.5  35.5
##  [67]  34.5  33.5  32.5  31.5  30.5  29.5  28.5  27.5  26.5  25.5  24.5
##  [78]  23.5  22.5  21.5  20.5  19.5  18.5  17.5  16.5  15.5  14.5  13.5
##  [89]  12.5  11.5  10.5   9.5   8.5   7.5   6.5   5.5   4.5   3.5   2.5
## [100]   1.5   0.5
```

Den siste linjen viser indeksnummerene til flere av verdiene i rekken tall vi spør om: tall nummer 12 i rekken tall er 89.5. Hvor mange verdier er det totalt i denne rekken?

R er det vi kaller et objektorientert programeringsspråk: vi lager objekter og jobber med disse objektene. Objektene kan være forskjellige klasse/målenivå (heltall, desimaltall, tekst, rangert, osv)

```
Tiern <- 1:10

Tiern
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
Tiern * Tiern
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```
Tiern[4]
```

```
## [1] 4
```

```
ABC <- c("A", "B", "C")

ABC[2]
```

```
## [1] "B"
```

```
ABC[c(3, 1, 2)]
```

```
## [1] "C" "A" "B"
```

Working directory og paths

De fleste av dere kommer til å jobbe med data som ligger lokalt på datamaskinen deres. Det er derfor viktig at vi skjønner hvordan mappestrukturen på dataen er bygget opp (dette varier mellom mac/windows/linux).

Hele tiden når vi skal "snakke" med filer i R, er det viktig å vite hvor i mappestrukturen R mener at vi befinner oss. Koden under får R til å vise oss hvor R jobber fra:

```
getwd()  # get working directory
```

```
## [1] "/home/martigso/Dropbox/PhD/Undervisning/Seminarleder/STV4020/høst_2017/Gruppe 2/docs"
```

Vi kan også endre dette (her er det viktig å skille mellom mac/windows/linux):

```
# Mac/Linux:
setwd("~/R/der/du/vil/jobbe/fra")

# Windows
setwd("C:/Users/Navn/R/der/du/vil/jobbe/fra")
```

Etter dette kan du sjekke hvilke filer og mapper som ligger i mappen du jobber fra – da jobber vi med relative paths:

```
list.files()
```

```
## [1] "seminar1.nb.html" "seminar1.pdf"     "seminar1.Rmd"
```

```
# En mappe under:
list.files("../")
```

```
## [1] "data"           "docs"           "Gruppe 2.Rproj" "pics"
## [5] "README.md"      "scripts"
```

```
# En mappe under, og i mappen scripts
list.files("../scripts")
```

```
## [1] "1seminar.R"
```

Det kan være lurt å sette seg inn i jobbe med prosjekter (den blå kuben helt øverst til høyre i R-studio), men vi får dessverre ikke tid å gå gjennom det her.

Datasett

Det vi jobber mest med er to-dimensjonale dataset: variabler er kolonner, enheter er rader og hver celle er en verdi. Nedenfor laster vi inn et datasett vi kan jobbe med videre dette seminaret. Data er passasjerer fra Titanic og variabler på om de overlevde, klasse, pris, osv. Dere kan enten laste ned data ved å skrive inn nettaddressen under i nettleseren og legge denne filen i mappen dere jobber fra:

```
setwd("~/Der/du/vil/jobbe/fra")

passengers <- read.csv("titanic.csv")
```

Eller dere kan laste den direkte inn i R via linken:

```
passengers <- read.csv("https://folk.uio.no/martigso/stv4020/titanic.csv")
```

La oss se på noen helt basic funksjoner vi kan bruke på datasettet:

```
class(passengers)
head(passengers)
tail(passengers)
colnames(passengers)
```

```
View(passengers)  # Denne gir et vindu med data i.
```

Allerede nå er det kommet mange **funksjoner** vi bruker. Funksjoner er en slags "sort-boks" vi sender informasjon til, for så å få ut det funksjonen er definert til å gi oss, gitt data vi putter inn. Funksjonen *head()* for eksempel er definert til å vise de første enhetene i en vektor, dataset, matrise eller tabell. Det er veldig nyttig å se på hjelpefilene til funksjonene man bruker:

```
?head
```

Da får vi opp bildet nedenfor i vinduet nede til høyre. Her har vi først en kort beskrivelse av hva funksjonen gjør, så hvordan den brukes. Disse lærer man seg å lese etter hvert som man jobber med hjelpefilene. Den viktigste seksjonen i starten er **Arguments**; her listes argumentene man kan gi til funksjonen og beskrivelse av hva disse skal være. Så funksjonen `head()` tar argumentene x og n, hvor den første er et objekt (f.eks et dataset) og den andre er et enkelt heltall/integer (f.eks 4) som gir antall elementer man skal vise med funksjonen. Legg merke til at n har 6 som default – disse verdiene står i seksjonen **Usage**. Derfor fikk vi 6 linjer når vi skrev `head(passengers)` over, uten å spesifiser n. Det er også viktig å spesifisere argumentene riktig; hvis vi gjør det i rekkefølgen **Usage** viser, slipper vi å skrive `n = 4` for eksempel:

```
head(passengers, 1)
```

```
##   PassengerId Survived Pclass                     Name  Sex Age SibSp Parch
## 1           1        0      3 Braund, Mr. Owen Harris male  22     1     0
##      Ticket Fare Cabin Embarked
## 1 A/5 21171 7.25              S
```

```
head(n = 1, x = passengers)
```

```
##   PassengerId Survived Pclass                     Name  Sex Age SibSp Parch
## 1           1        0      3 Braund, Mr. Owen Harris male  22     1     0
##      Ticket Fare Cabin Embarked
## 1 A/5 21171 7.25              S
```

Koden under vil gi en feilmelding fordi objektet *passengers* ikke er et enkelt heltall:

```
head(1, passengers)
```

head {utils} → Funksjon og {pakke funksjonen ligger i} R Documentation

Return the First or Last Part of an Object → Bare en overskrift

Description

Returns the first or last parts of a vector, matrix, table, data frame or function. Since `head()` and `tail()` are generic functions, they may also have been extended to other classes.

→ Kort beskrivelse av funksjonen

Usage

→ Hvordan bruke både head og tail

```
head(x, ...)
## Default S3 method:
head(x, n = 6L, ...)
## S3 method for class 'data.frame'
head(x, n = 6L, ...)
## S3 method for class 'matrix'
head(x, n = 6L, ...)
## S3 method for class 'ftable'
head(x, n = 6L, ...)
## S3 method for class 'table'
head(x, n = 6L, ...)
## S3 method for class 'function'
head(x, n = 6L, ...)

tail(x, ...)
## Default S3 method:
tail(x, n = 6L, ...)
## S3 method for class 'data.frame'
tail(x, n = 6L, ...)
## S3 method for class 'matrix'
tail(x, n = 6L, addrownums = TRUE, ...)
## S3 method for class 'ftable'
tail(x, n = 6L, addrownums = FALSE, ...)
## S3 method for class 'table'
tail(x, n = 6L, addrownums = TRUE, ...)
## S3 method for class 'function'
tail(x, n = 6L, ...)
```

Arguments → Beskrivelse av argumentene funksjonen tar

x	an object
n	a single integer. If positive, size for the resulting object: number of elements for a vector (including lists), rows for a matrix or data frame or lines for a function. If negative, all but the n last/first number of elements of x.
addrownums	if there are no row names, create them from the row numbers.
...	arguments to be passed to or from other methods.

Details → Detaljert beskrivelse av funksjonen

For matrices, 2-dim tables and data frames, `head()` (`tail()`) returns the first (last) `n` rows when `n > 0` or all but the last (first) `n` rows when `n < 0`. `head.matrix()` and `tail.matrix()` are exported. For functions, the lines of the deparsed function are returned as character strings.

If a matrix has no row names, then `tail()` will add row names of the form `"[n,]"` to the result, so that it looks similar to the last lines of x when printed. Setting `addrownums = FALSE` suppresses this behaviour.

Value → Hva vi får av output fra funksjonen

An object (usually) like x but generally smaller. For `ftable` objects x, a transformed `format(x)`.

Author(s)

Patrick Burns, improved and corrected by R-Core. Negative argument added by Vincent Goulet.

Examples → Eksempler på bruk av funksjonen

```
head(letters)
head(letters, n = -6L)

head(freeny.x, n = 10L)
head(freeny.y)

tail(letters)
tail(letters, n = -6L)

tail(freeny.x)
tail(freeny.y)

tail(library)

head(stats::ftable(Titanic))
```

Jobbe med variabler i dataset

Det første å gjøre når man har et dataset man ikke kjenner godt er å få en oversikt over variablene det inneholder. Nedenfor vises summary for alle variablene i data, summary for bare alder, gjennomsnitt for om passasjeren overlevde, en tabell med antall passasjerer som tilhørte hver av de tre klassene på båten og et histogram over alder.

```
summary(passengers)
```

```
##   PassengerId       Survived          Pclass
##  Min.   :  1.0   Min.   :0.0000   Min.   :1.000
##  1st Qu.:223.5   1st Qu.:0.0000   1st Qu.:2.000
##  Median :446.0   Median :0.0000   Median :3.000
##  Mean   :446.0   Mean   :0.3838   Mean   :2.309
##  3rd Qu.:668.5   3rd Qu.:1.0000   3rd Qu.:3.000
##  Max.   :891.0   Max.   :1.0000   Max.   :3.000
##
##                                     Name         Sex           Age
##  Abbing, Mr. Anthony                   :  1   female:314   Min.   : 0.42
##  Abbott, Mr. Rossmore Edward           :  1   male  :577   1st Qu.:20.12
##  Abbott, Mrs. Stanton (Rosa Hunt)      :  1                Median :28.00
##  Abelson, Mr. Samuel                   :  1                Mean   :29.70
##  Abelson, Mrs. Samuel (Hannah Wizosky):  1                3rd Qu.:38.00
##  Adahl, Mr. Mauritz Nils Martin        :  1                Max.   :80.00
##  (Other)                               :885                NA's   :177
##      SibSp           Parch             Ticket         Fare
##  Min.   :0.000   Min.   :0.0000   1601    :  7   Min.   :  0.000
##  1st Qu.:0.000   1st Qu.:0.0000   347082  :  7   1st Qu.:  7.896
##  Median :0.000   Median :0.0000   CA. 2343:  7   Median : 14.454
##  Mean   :0.523   Mean   :0.3816   3101295 :  6   Mean   : 32.099
##  3rd Qu.:1.000   3rd Qu.:0.0000   347088  :  6   3rd Qu.: 30.848
##  Max.   :8.000   Max.   :6.0000   CA 2144 :  6   Max.   :512.329
##                                   (Other) :852   NA's   :4
##          Cabin     Embarked
##             :687    :  2
##  B96 B98    :  4   C:168
##  C23 C25 C27:  4   Q: 77
##  G6         :  4   S:644
##  C22 C26    :  3
##  D          :  3
##  (Other)    :186
```

```
summary(passengers$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
```

```
##    0.42   20.12   28.00   29.70   38.00   80.00     177
```

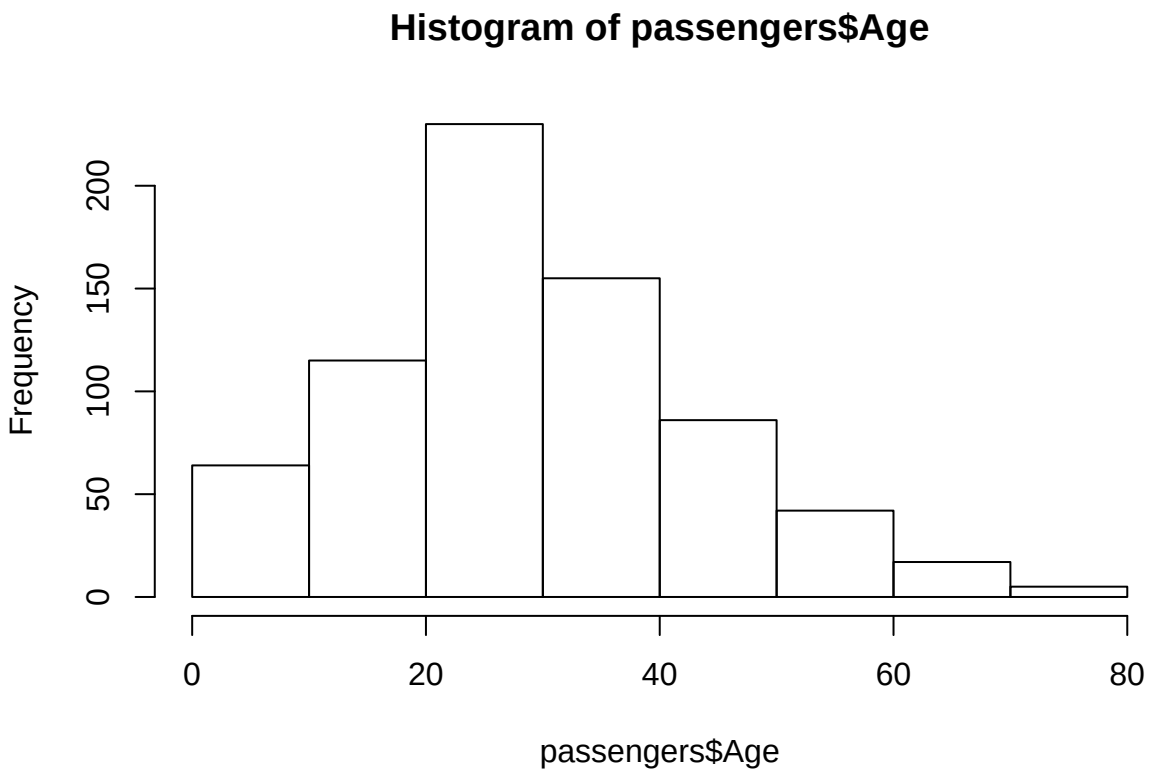
```
mean(passengers$Survived)
```

```
## [1] 0.3838384
```

```
table(passengers$Pclass)
```

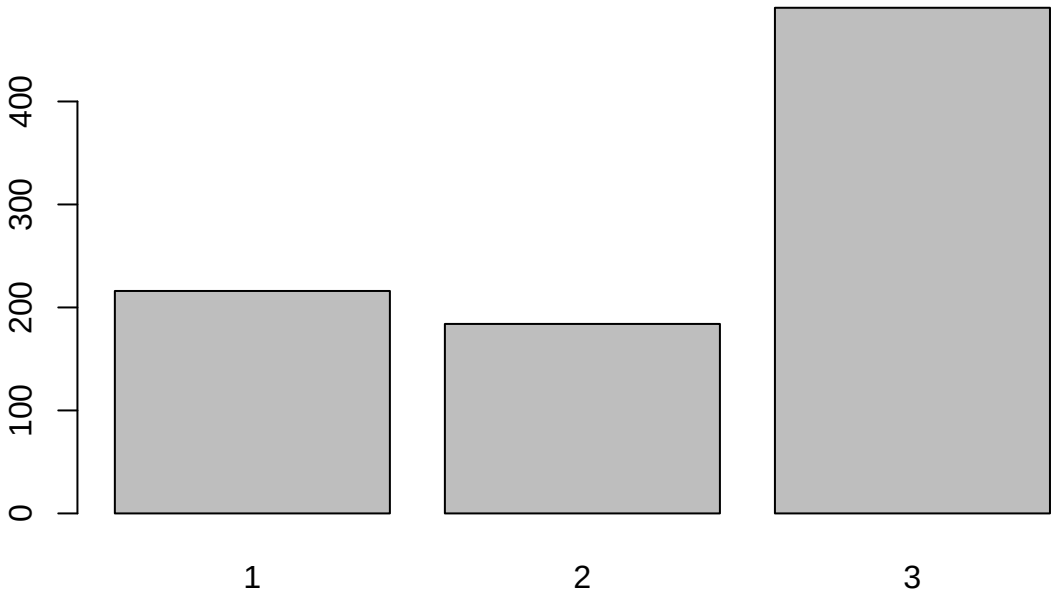
```
##
##   1   2   3
## 216 184 491
```

```
hist(passengers$Age)
```



Personlig liker jeg figurer bedre enn tabeller, så tabellen kan enkelt illustreres som dette:

```
klassetabell <- table(passengers$Pclass)
barplot(klassetabell)
```




```r
barplot(table(passengers$Pclass))  #Vi trenger ikke gå innom objektet
```

Noen ganger trenger vil å håndere data som er **missing** (variabler vi ikke har informasjon om på noen/alle enheter). Enheter med missing tar verdien `NA` ("Not Available") i R. Kodelinjen nedenfor viser at vi ikke får sjekket gjennomsnitt på alder; vi kan ikke regne gjennomsnitt på rekker med tall som har `NA` i seg. Vi kartlegger derfor hvor mange missing vi har på aldervariabelen. `is.na()` sjekker for hver rad i datasettet om enheten har missing på aldersvariabelen og gir `TRUE` hvis den har det og `FALSE` hvis den ikke har det. `table()` teller opp disse to kategoriene. På den siste linjen sier vi at vi skal regne gjennomsnitt for alder ved å fjerne alle missing. Hvor mange enheter har missing på variabelen?

```r
mean(passengers$Age)
```

```
## [1] NA
```

```r
table(is.na(passengers$Age))
```

```
##
## FALSE  TRUE
##   714   177
```

```r
mean(passengers$Age, na.rm = TRUE)
```

```
## [1] 29.69912
```

Litt omkoding

Ofte er vi heller ikke fornøyd med hvordan data er strukturert. Her er en av hovedfordelene med R; vi kan gjøre så og si hva som helst for å få dataene i det formatet vi ønsker. La oss si at vi, for eksempel, har en hypotese om at eldre personer hadde mindre sannsynlighet for å overleve enn yngre personer. Som dere husker fra forelesning :) kan det være lurt å sentrere variabler som alder fordi vi sjelden har et naturlig nullpunkt, som igjen gjør at konstantleddet i en evt regresjon ikke gir substansiell mening. La oss derfor sentrere alder:

```r
median(passengers$Age, na.rm = TRUE)
```

```
## [1] 28
```

```r
passengers$age_cent <- passengers$Age - median(passengers$Age, na.rm = TRUE)

table(passengers$age_cent[1:10], passengers$Age[1:10])
```

```
##
##       2 14 22 26 27 35 38 54
##   -26 1  0  0  0  0  0  0  0
##   -14 0  1  0  0  0  0  0  0
##   -6  0  0  1  0  0  0  0  0
##   -2  0  0  0  1  0  0  0  0
##   -1  0  0  0  0  1  0  0  0
##   7   0  0  0  0  0  2  0  0
##   10  0  0  0  0  0  0  1  0
##   26  0  0  0  0  0  0  0  1
```

Over har jeg laget en ny variabel i datasettet *passengers* som heter *age_sent*. Den skal være sentrert til median: vi trekker fra medianen til aldervariabelen fra alle verdier på aldervariabelen. Legg også merke til at jeg validerer at det ble riktig med `table()`.

Først kan vi sjekke korrelasjonen mellom de to variablene våre. Her bruker vi funksjonen `cor()` for bare korrelasjonsestimat, og `cor.test()` for å se om estimatet er signifikant forskjellig fra null:

```r
cor(passengers$age_cent, passengers$Survived)
```

```
## [1] NA
```

```r
cor(passengers$age_cent, passengers$Survived, use = "complete.obs")
```

```
## [1] -0.07722109
```

```r
cor.test(passengers$age_cent, passengers$Survived, use = "complete.obs")
```

```
##
##  Pearson's product-moment correlation
##
## data:  passengers$age_cent and passengers$Survived
## t = -2.0667, df = 712, p-value = 0.03912
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.149744955 -0.003870727
## sample estimates:
##         cor
## -0.07722109
```

Også her må vi håndtere missingverdier (ref første linje over). Men med korrelasjon er det, som dere vet,

forskjellige måter å håndere missing på: pairwise og listwise exclusion. Dette er ikke viktig med korrelasjon mellom bare to variabler, men med flere variabler er det viktig:

```
cor(passengers[, c("age_cent", "Survived", "Fare")], use = "complete.obs")
cor(passengers[, c("age_cent", "Survived", "Fare")], use = "pairwise.complete.obs")
```

```
##             age_cent    Survived       Fare
## age_cent  1.00000000 -0.07692265 0.09638814
## Survived -0.07692265  1.00000000 0.27128592
## Fare      0.09638814  0.27128592 1.00000000
##             age_cent    Survived       Fare
## age_cent  1.00000000 -0.07722109 0.09638814
## Survived -0.07722109  1.00000000 0.25965960
## Fare      0.09638814  0.25965960 1.00000000
```

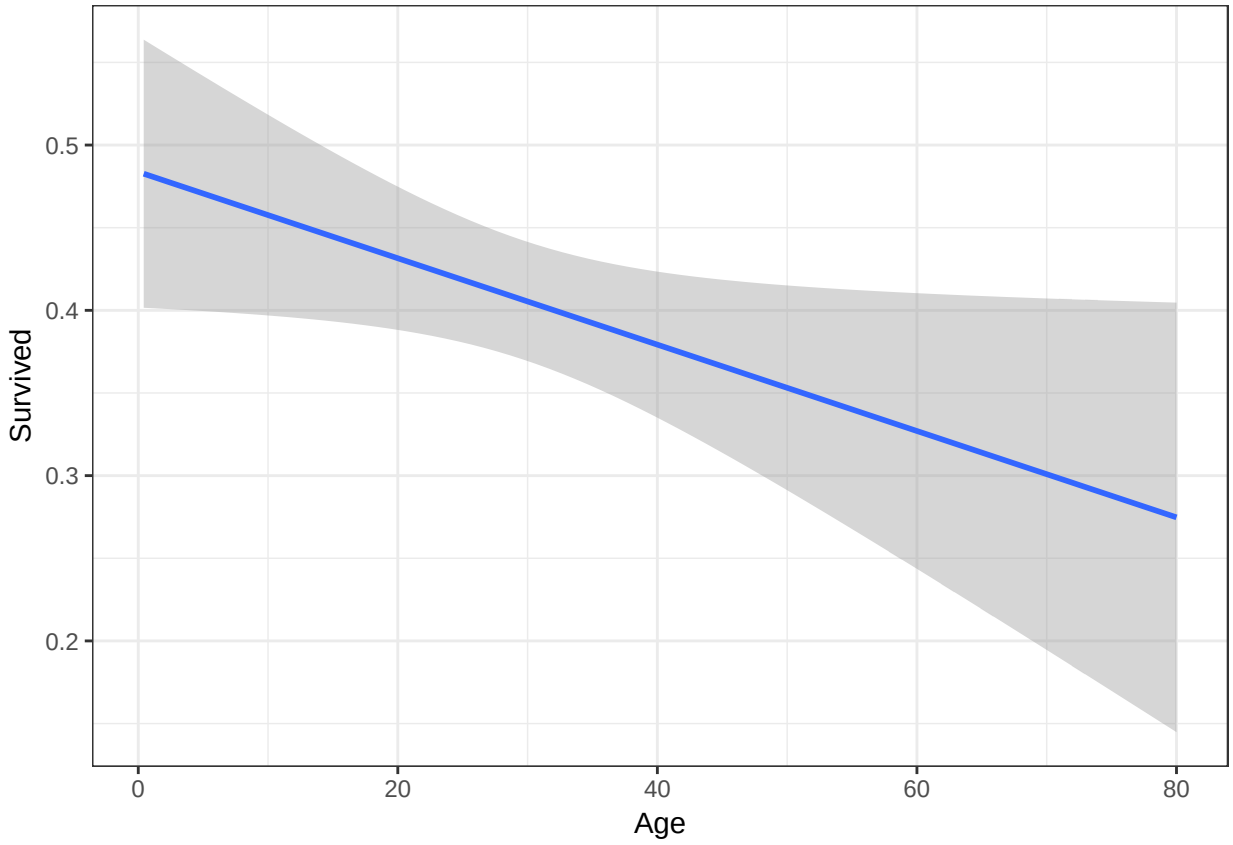
Da gjenstår det bare å kjøre en linær regresjon (OLS) for å teste hypotesen. For dette bruker vi funksjonen `lm()`. Se på hjelpefilen om koden under gir mening.

```
pass_reg <- lm(Survived ~ age_cent, data = passengers)
summary(pass_reg)
```

```
##
## Call:
## lm(formula = Survived ~ age_cent, data = passengers)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4811 -0.4158 -0.3662  0.5789  0.7252
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.410601   0.018476  22.224   <2e-16 ***
## age_cent    -0.002613   0.001264  -2.067   0.0391 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4903 on 712 degrees of freedom
##   (177 observations deleted due to missingness)
## Multiple R-squared:  0.005963,   Adjusted R-squared:  0.004567
## F-statistic: 4.271 on 1 and 712 DF,  p-value: 0.03912
```

```r
# install.packages('ggplot')
library(ggplot2)
theme_set(theme_bw())
ggplot(passengers, aes(x = Age, y = Survived)) + geom_smooth(method = "lm")
```

```
## Warning: Removed 177 rows containing non-finite values (stat_smooth).
```

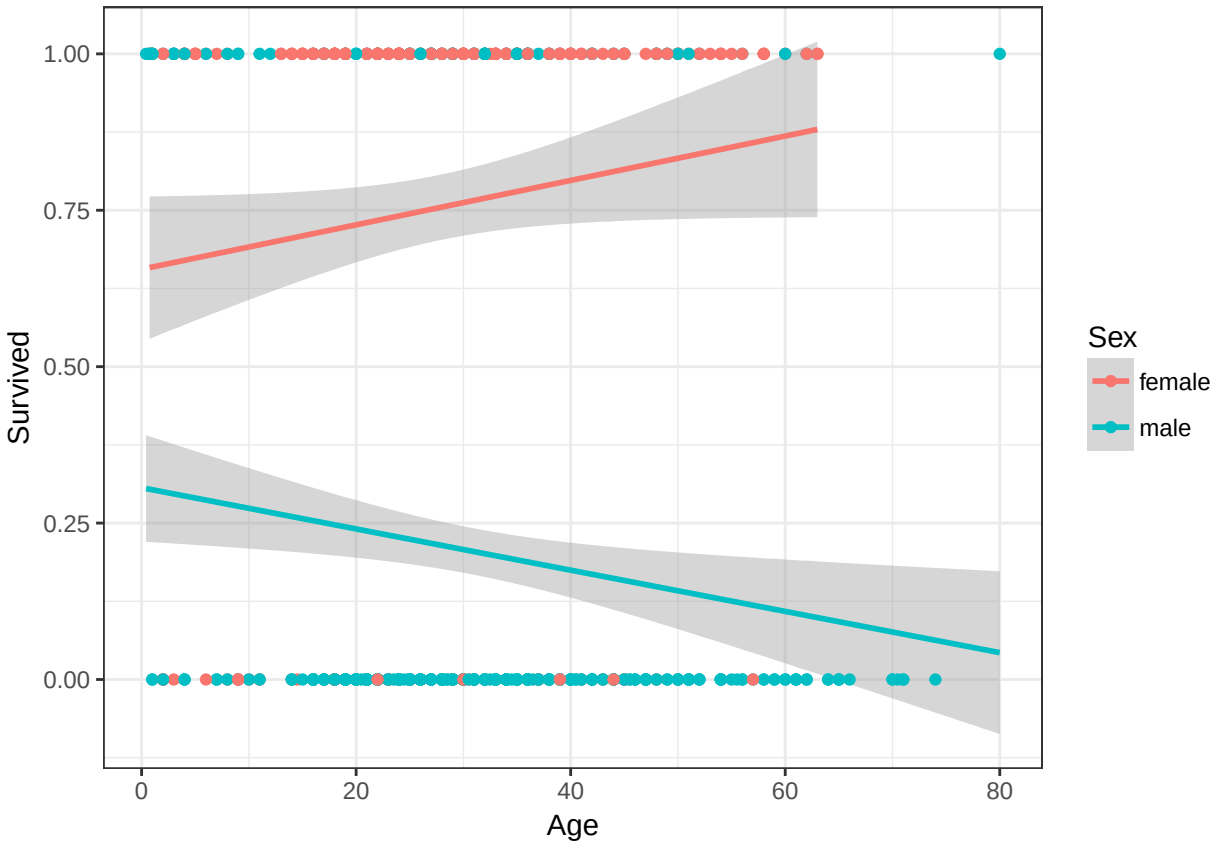


Kan dere tenke dere noen variabler som vi burde inkludere i denne regresjonen?

```
ggplot(passengers, aes(x = Age, y = Survived, color = Sex)) + geom_smooth(method = "lm") +
    geom_point()
```

```
## Warning: Removed 177 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 177 rows containing missing values (geom_point).
```



Bonus for LaTeXelskere:

```
# install.packages('stargazer')
library(stargazer)
```

```
##
## Please cite as:
##  Hlavac, Marek (2015). stargazer: Well-Formatted Regression and Summary Statistics Tables.
##  R package version 5.2. http://CRAN.R-project.org/package=stargazer
```

```
stargazer(pass_reg, star.cutoffs = c(0.05, 0.01, 0.001), ci = TRUE)
```

% Table created by stargazer v.5.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: sø., aug. 27, 2017 - kl. 15.56 +0200

Table 2:

	Dependent variable:
	Survived
age_cent	-0.003^{*}
	$(-0.005, -0.0001)$
Constant	0.411^{***}
	(0.374, 0.447)
Observations	714
R^2	0.006
Adjusted R^2	0.005
Residual Std. Error	0.490 (df = 712)
F Statistic	4.271^{*} (df = 1; 712)
Note:	*p<0.05; **p<0.01; ***p<0.001