Datorlaboration 7

Josef Wilzén

7 mars 2014

Instruktioner

- Denna laboration ska göras i **grupper om två personer**. Det är viktigt att att följa gruppindelningen och inte ändra grupper. Om det är problem med grupperna så ska ni prata med Josef eller Måns.
- Det är tillåtet att diskutera med andra grupper, men att plagiera eller skriva kod åt varandra är **inte tillåtet**.
- Utgå från laborationsfilen som går att ladda ned **här**
- Laborationen består av två delar:
  - Datorlaborationen
  - Inlämningsuppgifter
- Innan du lämnar in laborationen:
  1. Starta om R-Studio eller rensa den globala miljön (Global environment) med `rm(list = ls())`.
  2. Ladda in funktionerna i R med `source`.
  3. Kontrollera att inget annat än funktionerna laddas in.
  4. Testa att funktionerna fungerar en sista gång.
- Deadline för labben framgår på **kurshemsidan**

Innehåll

Parprogrammering

Tanken är att ni ska öva på parprogrammering under labb4 till labb8.

- Detta innebär att två personer samarbetar och tillsammans löser programmeringsproblem vid en dator.
- Personerna turars om att ha rollerna:
  - **Föraren:** har kontroll över tangentbordet och skriver koden
  - **Navigatören:** Är delaktig i problemt genom att kommentera, diskutera och analysera koden som skrivs. Den här personen ska **inte** sitta och titta passivit.
- Det är viktigt att byta roller **ofta** och att båda personerna är involverade i lösningen av problemet. Vi rekomenderar att ni byter roller minst varje 30 min.
- Det är viktigt att kommentera sin kod så att ni båda kan förstå koden i efterhand. Tänk på skriva ner syftet med koden och inte exakt vad koden gör.
- Syftet med parprogrammering är:
  - Lära av varandra
  - Lära sig olika sätt att skriva kod (stilar) och olika sätt att lösa problem
  - Skriva mer effektiv kod med mindre buggar
  - Lära sig skriva kod som är lätt att förstå för andra

Del I
Datorlaboration

1 Texthantering och regular expression

Thomas Tranströmmer är en av de mest kända svenska poeterna och fick dessutom Nobelpriset i litteratur 2011. I denna uppgift ska vi pröva att arbeta lite med ord och text i R och använder då två av Thomas Tranströmer dikter som exempel. Paketet `stringr` kommer användas, så se till att ha installerat detta paket. Samtliga funktioner i `stringr`-paketet börjar med `str_` vilket gör det enkelt att arbeta med detta paket i R-studio. Vill du ha en översikt över alla funktioner som finns i `stringr`-paketet, se refrerensmanualen som finns **[här]** (eller sök efter paketet på internet).

1. Börja med att läsa in paketet `stringr` i R.

2. Läs in Thomas Tranströmers dikt The Couple (engelsk översättning) med funktionen `readLines()`, filen heter "transtrom.txt". Varje rad i dikten ska nu vara inlästa som en separat textsträng. Spara den inlästa texten som `nicePoem`.

   (a) Först vill vi kunna läsa dikten i sin helhet på ett trevligt sätt. Mellan respektive rad ska du sätta in textsträngen för ny rad `\n` som avskiljare mellan de ihopsatta textelementen. Spara den nya textsträngen som ett nytt objekt, `nicePoemOut`. [**Tips!** `str_c()`]

   (b) Använd funktionen `cat()` för att skriva ut dikten - dikten ska nu "se bra ut" i R.

   (c) Räkna ut hur många tecken som finns i respektive rad i dikten [**Tips!** Kolla på funktionerna som finns i `stringr`]

3. Vi ska nu läsa in en ytterliggare dikt av Thomas Tranströmer. Läs in dikten `transtrom2.txt` med `readlines()` och spara i `poem`. Eftersom det är en dikt på svenska behöver vi använda en svensk formatering av bokstäverna som kallas latin1 (eller ISO-8859-1). Ange därför argumentet `encoding="latin1"` i funktionen `readlines()`. Använd regular expressions för att göra följande:

   (a) Undersök vilka elemment som innehåller "V" eller "v". Välj sedan ut de elementen och skriv ut dem till konsolen. Tips: `str_dectect()`

   (b) Hitta alla ord som är exakt tre bokstäver långt. Tips: `str_extract()`, `str_extract_all()` Vad är skillnaden mellan dessa?

   (c) Byt ut alla "a" mot "???" Tips: `str_replace()`, `str_replace_all()` Vad är skillnaden mellan dessa?

(d) Utgå från första element i `poem`. Använd `str_sub()` för att välja ut det tredje, femte och sista ordet. Sätt sedan samman dessa till en ny text-sträng.

4. Skriv en funktion som givet en textvektor skriver ut det element som innehåller flest tecken. Har flera element lika många tecken så ska alla dessa element skrivas ut.

5. Skriv en funktion som givet en textvektor, ersätter alla år på formen "1974", "2031" med strängen "år". Testa med vektorn nedan.

```
c("hej 2012 sommar!!","1983 !!++2001" "Ska vi ta examen 2016")
```

2 Linjär algebra

I denna sektion kommer en del av uppgifterna att komma från kursboken "A First Course in Statistical Programming with R".

1. Utgå från exempel 6.1 i kapitel 6 i kursboken. Skriv en funktion som som kan skapa en Hiblert matris av godtycklig storlek. Kalla funktionen `HilbertMat(n=)`, där `n` är storleken på den kvadratiska matrisen.Tips: utgå från en tom matris som ni sedan fyller i en nästlad for-loop.

2. Gör uppgift 1-3 under avsnittet 6.1.1 i kursboken.

3. Vad händer i följande kod? Tips: `?class() ?"["`

```
x <- matrix(1:20, 4, 5)
x[, 1]

[1] 1 2 3 4

x[, 1, drop = FALSE]

     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
```

4. Gör uppgift 2-3 under avsnittet 6.1.2 i kursboken.

5. Använd nu funktionen diag() för att:

   (a) Skapa en enhetsmatris av storlek 12.

   (b) Skapa en diagonalmatris som har värdena $2, 3, 5, 7, 1, 2$ på diagonalen.

6. Skriv en funktionen som beräknar determinaten för en $2 * 2$ matris. Kolla i kursboken under 6.1.3 hur du kan göra det.

7. Skapa matrisen `A` nedan. Testa sedan att använda funktionerna `lower.tri()` och `upp.tri()`, vad händer?

```
A <- matrix(1:25, 5, 5)
```

8. Nu ska du skapa en funktion som kan skapa symmeteriska matriser. Det innebär att matrisen uppfyller följande $X = X^T$, eller anorlunda utryckt $x_{ij} = x_{ji}$, där i är rad index och j är kolumn index. Funktioen ska heta `createSymMat(diagVect=,otherElements=)` och ha argumenten `diagVect` vilket är en vektor med numeriska värden, `otherElements` är en kvadratisk matris med lika många kolumner som `diagVect` har element. Funktionen ska göra följande: Sätta upp en matris med bara 0 av samma storlek som `otherElements`, kalla den `result`. Sen ska diagonalelementen få värdena som finns i `diagVect`. Ta sedan de element i matrisen `otherElements` som ligger **över** diagonalen och spara dessa värden på samma ställen fast i matrisen `result`. Upprepa sen samma saka fast ta värdena **över** diagonalen på `otherElements` och spara dem **under** diagonalen i `result`, så att du erhåller en symmetrisk maris. Returnera sen `result`. Tips: `lower.tri(), upp.tri(), diag(), t(), dim()`

9. Gör uppgift 1-2 under avsnittet 6.1.3 i kursboken.

10. Funktionen `generateMatrix()` skapar slumpmässiga kvadratiska matriser med hjälp av `sample()`. Kör `generateMatrix()` så att den finns tillgänglig i din workspace. Kör sedan koden under funktionen och försök svara på frågorna. Vad innebär resultatet från funktionen `kappa()`? Hur påverkar det beräkningarna av matrisinverser? Tips: Avsnitt 6.4.4 i kursboken eller `?kappa()`.

```
generateMatrix <- function(matDim = 5, numbers = 10, seed = 12345) {
    set.seed(seed)
    mySize <- matDim^2
    x <- sample(x = numbers, size = mySize, replace = TRUE)
    y <- matrix(x, matDim, matDim)
    return(y)
}
```

```
# skapa matriser:
A <- generateMatrix(matDim = 10, numbers = -10:10, seed = 398)
B <- generateMatrix(matDim = 100, numbers = -10:10, seed = 872)
C <- generateMatrix(matDim = 1000, numbers = -10:10, seed = 812)
# undesöka matriser:
dim(A)
dim(B)
dim(C)
kappa(A)
kappa(B)
kappa(C)
# beräkna inverser:
Ainv <- solve(A)
Binv <- solve(B)
Cinv <- solve(C)
```

3 Inför labb 8

Tanken är här att ni ska göra en del förberedelser inför labb 8 som är ett miniprojekt. Till den labben behöver ni några datamaterial. Dessa ska komma från en svensk myndighet, förslagsvis från SCB.

1. Ladda först in paktet sweSCB i er session. Om du använder en egen dator behöver installera sweSCB, kolla på föreläsning 7 eller på Github hur du kan göra det.

2. Testa att använda funktionen findData() för att ladda ner ett testdata-matlerial från SCB:s hemsida. Kör koden nedan

```
minData <- findData()
```

   (a) Nu startas en interaktiv session i R-konsolen. Trycker du på "esc"-tangenten så avbryts sessionen, trycker du på "b"-tangenten så backer du ett steg i menyn, olika siffror öppnar olika menyval.

```
Enter the data (number) you want to explore:
('esc' = Quit, 'b' = Back)
```

   (b) Testa att välja: 12 "[BE] Befolkning" –> 1 "[BE0101] Befolkningsstatistik" –> 3 "[BE0101B] Medelålder" –> 1 "[BefolkningMedelAlder] Befolkningens medelålder efter region och kön. År 1998 - 2012". Nu får du frågan om nu vill ladda ner datamaterialet, svara "y". Svara även

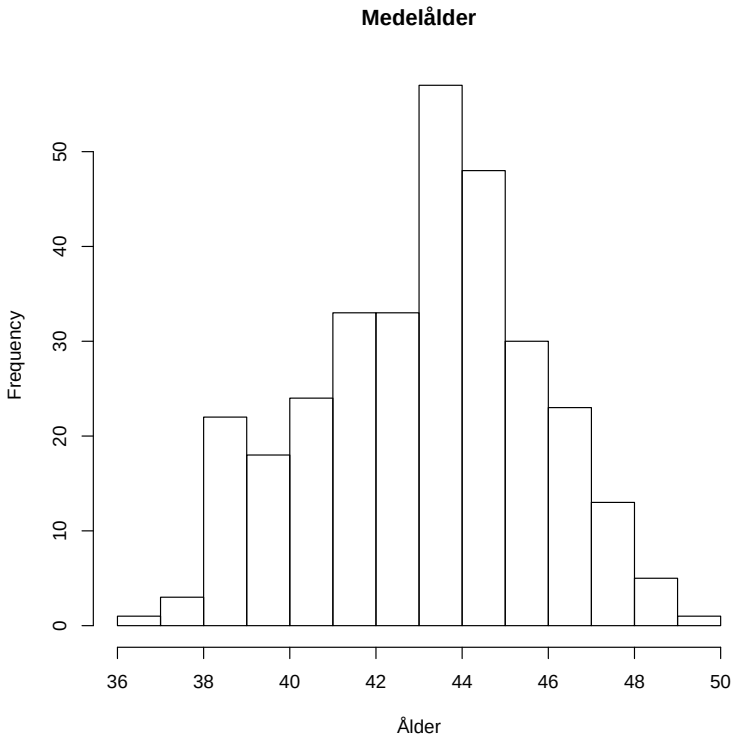
"y" på frågorna om du vill städa data och om du vill skriva ut koden som krävs för nedladdning.

(c) Nu får du frågan vilka variabler vi vill ha till vår data, välj region 2-312 genom att skriva kommandot 2:312. Nästa variabel är kön, hör vill vi välja alla kategorier, så ange kommandot * och tryck enter. Välj sedan 1 "[BE0101G9] Befolkningens medelålder" och därefter året 2012.

(d) Data ska nu laddas ner till din R-session och sparas i variabeln `minData`. Observera att nu skrivs koden som krävdes för denna specifika nedladdning ut till konsolen. Denna kod kan sparas för att ladda ner exakt samma data utan att behöva stega i menyerna igen.

(e) Testa nu att göra följande analys på datamaterialet:

```
minData2 <- minData[minData[, 2] == "totalt", ]
hist(minData2[, 5], main = "Medelålder", xlab = "Ålder")
```



(f) Gör om historgrammet ovan, men ett för bara kvinnor och ett bara för män. Ge dem lämpliga rubriker.

3. Till labb 8 ska ni använda 3 dataset. Som ska komma från någon svensk myndighet. Antingen laddar ni ner data med `sweSCB`, eller så går ni in på hemsidan för SCB eller en annan myndighet och laddar ner data manuellt. Er uppgift är att hitta data som uppfyller följande krav:

   (a) Hitta två olika dataset som har data på **kommunnivå**. Det innebär att det data ska innehålla värden för alla kommuner (eller i alla fall en majoritet av kommunerna). Ett exempel skulle kunna vara antal invånare i varje kommun. Dataseten ska ha minst **2 variabler** utöver kommun. Ni väljer själv vilka variabler som ska ingå och vilken område data ska komma ifrån. Dock rekomenderas att ni väljer data och variabler som kan tänkas ha ett samband mellan variablerna i de olika dataseten. Anledningen är den att datasetten ska användas för att göra några enkla sambandanalyser mellan variablerna i labb 8.

   (b) Hitta ett dataset som innehåller en **tidserie**, det innebär att det finns en variabel som har observerats över tiden. Kravet är att data ska innehålla data på **månadsnivå** och innehålla data från minst 3 år (36 månder). Här ska ni alltså hitta en variabel som observerats under minst 36 tidpunkter (månader). Data ska alltså innehålla två kolumner, en med variabeln som vi är intresserade av och en med tidpunkterna.

4. Läs in dataseten i 3 på valfritt sätt i R. Titta så att data ser bra ut, tex är numeriska variabler numeric? Är variabelnamnen korrekta? etc. Gör nödvändiga bearbetningar av data att det går att analysera vidare med data.

5. Slå samman de två dataseten med kommundata så det blir ett dataset som innehåller variablerna från båda dataseten. Om ni gör rätt här så ska ni få ett dataset med en variabel över kommun och minst 4 andra variabler. Detta kan göras på olika sätt, ett är att använda funktionen `merge()`. Se nedan för exempel. Här finns en video för hur ni kan använda `merge()`. Ett annant är att sortera båda dataseten likadant (efter kommun) och sen sätta samman dem `cbind()` eller något likande. Sortera en hel data.frame kan göras med `order()`.

```
x <- 1:5
grupp = letters[1:5]
y <- round(rnorm(5), 3)
grupp2 = letters[c(1, 3, 5, 4, 2)]
a <- data.frame(x, grupp)
b <- data.frame(y, grupp = grupp2)
a

  x grupp
```

```
1 1     a
2 2     b
3 3     c
4 4     d
5 5     e

b


       y grupp
1 -0.466     a
2  0.186     c
3  0.319     e
4  1.192     d
5 -0.975     b

# Olika sätt att anropa merge():
merge(x = a, y = b, by.x = "grupp", by.y = "grupp")

  grupp x      y
1     a 1 -0.466
2     b 2 -0.975
3     c 3  0.186
4     d 4  1.192
5     e 5  0.319

merge(x = a, y = b, by = "grupp")

  grupp x      y
1     a 1 -0.466
2     b 2 -0.975
3     c 3  0.186
4     d 4  1.192
5     e 5  0.319
```

6. Spara dataseten och koden som används i 3 - 5 till nästa vecka, då ni ska göra mer analyser av dessa datamaterial i labb 8. Detta är ett viktigt steg, så att ni inte måste göra om allt från början.

Del II
Inlämningsuppgifter

Tips!

Inlämningsuppgifterna innebär att konstruera funktioner. Ofta är det bra att bryta ned programmeringsuppgifter i färre små steg och testa att det fungerar i varje steg.

Ett förslag på hur du kan angripa problemet är att:

1. Lös uppgiften med vanlig kod direkt i R-Studio (precis som i datorlaborationen ovan) utan att skapa en funktion.
2. Testa att du får samma resultat som testexemplen.
3. Implementera koden du skrivit i 1. ovan som en funktion.
4. Testa att du får samma resultat som i testexemplen, nu med funktionen.

Varje uppgift kan ge flera poäng så även om du inte lyckas med alla delar i en uppgift kan du få poäng.

4 Linjär regression

I denna uppgift ska ni skriva en funktionen som kan beräkna en linjär regressionsmodel. Vi kommer att prata mer om linjär regression på föreläsning 8 och gå igenom de nybyggda funktionerna för linjär regression i R, nu ska ni "skatta modellen för hand" med hjälp av matrisalgebra. Linjär regression handlar om att undersöka om en beroende variabel y beror av en eller flera obereonde/förklarande variabler $x_1, x_2, ...x_p$. Dessa kolumnvektorer brukar sättas ihop till en matris X.

En linjär regression model med en oberoende variabel ser ut så här:

$$y = \beta_0 + \beta_1 * x_1$$

Vilket kan beskrivas som räta linjens ekvation i ett x-y-planet. Om vi har fler, tex tre stycken oberoende variabler ser modellen ut så här:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_2 * x_2$$

Om vi ska skatta en linjär regression så innebär det att vi ska hitta de bästa värdena för β i någon mening, och det är uppgiften i labben.

Ni ska nu skriva funktionen `MyOLS(X=, y=)` som har argumenten:

- `X`: en matris där kolumnerna är de oberoende variablerna i vår model
- `y`: en vektor med den variablen som är den beroende varaiblen i modellen.

Nedan följer lösningsordningen, men som exempel används datasetet "AppleLabb5.csv". Här är variablen `Close` vår y-variabel och variablerna `Open`, `High`, `Low` är våra x-variabler. Dessa utgör nu kolumnerna i `X`. "AppleLabb5.csv" används som exempel, er funktion ska fungera för ett godtyckligt dataset.

1. Förbered data:

   (a) Gör om ditt X till en matris om det är en data.frame och döp denna matris till `X`. Tips! `as.matrix()`

   (b) Gör om din vektor y till en $n \times 1$ matris. Tips: se uppgift (3)i sektionen Linjär algebra.

2. Följande linjära regressionsmodell

$$\texttt{Close} = \beta_0 + \beta_1 \cdot \texttt{Open} + \beta_2 \cdot \texttt{High} + \beta_3 \cdot \texttt{Low}$$

   kan skattas genom direkt matrisalgebra. Genomför följande steg för att göra denna beräkning.

   (a) Transponera (X^T) din matris `X` ge den namnet `XT`.

   (b) Beräkna matrismultiplikationen $X^T X$ och ge den namnet `XTX`.

   (c) Undersök om matrisen $X^T X$ är inverterbar. Om den inte är inverterbar så ska funktionen returnera strängen "Cannot compute linear regression", och funktionen ska stoppas. **Tips!** Du kan undersöka om alla egenvärden för matrisen $X^T X$ är större än 0 vilket är ett krav för att matrisen ska vara inverterbar, kolla på `?eigen() ?all()`

   (d) Beräkna inversen av $X^T X$ och namnge resultatet `XTXInv`

   (e) Beräkna din skattning av $\beta_0, \beta_1, \beta_2, \beta_3$ på följande sätt: $\hat{\beta} = (X^T X)^{-1} X^T y$ och spara resultatet som en 4×1 matris och döp den till `betaHat`. Se till att raderna i betaHat samma namn som variablerna i `X`. Se testfallen för exempel på hur naman ska se ut.

   (f) Beräkna det förväntade värdet $\hat{y}$ för varje observation, i.e, $\hat{y} = X\hat{\beta}$ och döp vektorn till `yHat`.

   (g) Beräkna residualerna $\hat{\epsilon} = y - \hat{y}$ och döp dem till `eHat`. [**Tips!** Kontrollera att `eHat` är en $n \times 1$ matris och inte en vektor, annars kan funktionen `as.matrix()` användas]

   (h) Räkna ut hur många observationer som användes i analysen, spara detta värde som `n`.

   (i) Räkna hur många parametrar som skattas i modellen, dvs hur många kolumner är det i X, spara detta värde som `p`.

   (j) Beräkna

$$s^2 = \frac{\hat{\epsilon}^T \hat{\epsilon}}{n - p}$$

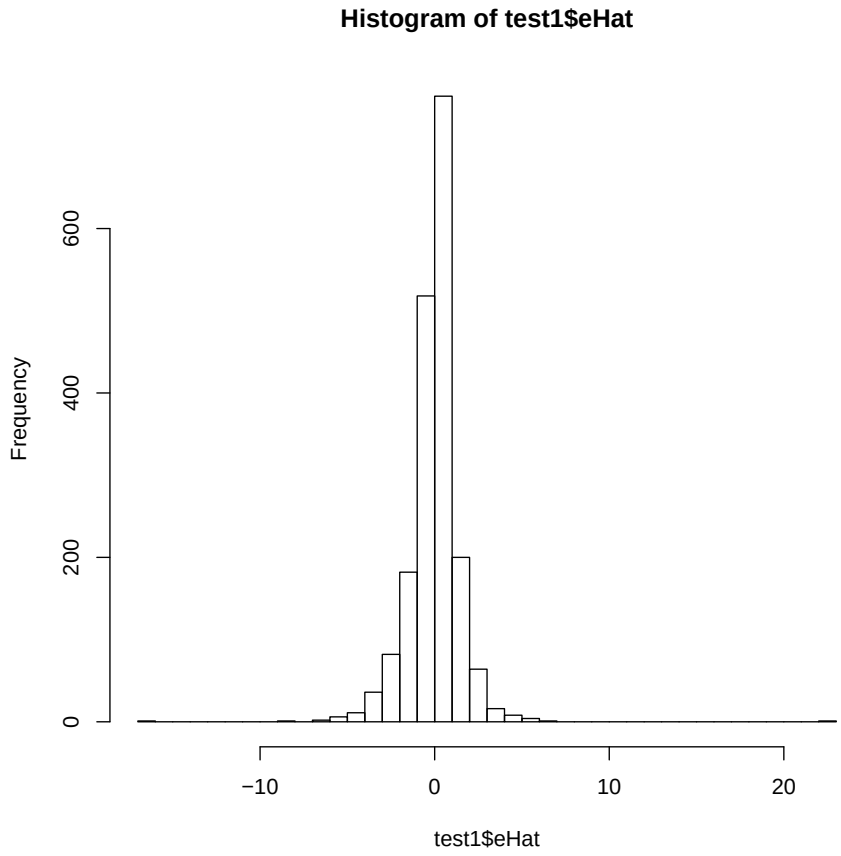
   och spara s som `sigmaHat`.

(k) Spara `betaHat`, `sigmaHat` och `eHat` i en lista där listelementen ska ha samma namn som variablen som ligger i elementet. Ordningen ska vara den som givs i texten. Döp listan till `OLSresult`. Tex första elementet ska heta `betaHat`.

(l) Returnera `OLSresult`.

Kolla nu om din funktion klarar följande testfall:

```
setwd("/home/joswi05/Dropbox/Rkurs/KursRprgm/Labs/DataFiles/")
# Test 1: Apple-data
Apple <- read.table("AppleLabb5.csv", sep = ";", header = TRUE)
X <- cbind(1, Apple[, 1:3])
names(X)[1] <- "Constant"
y <- Apple[, 4]
test1 <- MyOLS(X = X, y = y)
# undersöka:
names(test1)

[1] "betaHat"  "sigmaHat" "eHat"

test1[1:2]

$betaHat
            [,1]
Constant -0.2248
Open     -0.5486
High      0.9332
Low       0.6138

$sigmaHat
      [,1]
[1,] 1.492

summary(as.vector(test1$eHat))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-16.100  -0.552   0.127   0.000   0.660  22.400

hist(test1$eHat, 30)
```




```
# Test 2: tree-data 1
data(trees)
X <- cbind(1, trees[, 1:2])
names(X)[1] <- "Constant"
y <- trees[, 3]
test2 <- MyOLS(X = X, y = y)
test2[1:2]

$betaHat
             [,1]
Constant -57.9877
Girth      4.7082
Height     0.3393

$sigmaHat
      [,1]
```

```
[1,] 3.882

summary(as.vector(test2$eHat))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -6.410  -2.650  -0.288   0.000   2.200   8.480

# Test 3: tree-data 2
X <- cbind(1, trees[, 2, drop = FALSE])
names(X)[1] <- "Constant"
y <- trees[, 3]
test3 <- MyOLS(X = X, y = y)
test3[1:2]

$betaHat
            [,1]
Constant -87.124
Height     1.543

$sigmaHat
     [,1]
[1,] 13.4

summary(as.vector(test3$eHat))

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -21.30   -9.89   -2.89    0.00   12.10   29.90
```

5 Mer TransströmeR

5.1 Räkna ord

Nu är uppgiften att skapa en funktion som ska kunna räkna hur många gånger olika ord förekommer i texten. Funktionen ska heta `WordCount(text=)`, argumentet `text` ska vara en tecken-vektor. Funktionen ska ta en text (i form av en text vektor) och returnera en frekvensen för alla ord om ingår i texten.

1. Läs in paktet i stringr i den aktuella R-sessionen.
2. Börja med att likt uppgift 2a i sektionen "Texthantering..." sätta ihop de olika textelementen till en textsträng, men denna gång använd mellanslag som avskiljare istället för `\n`.

3. Ta bort punkter och kommatecken i textsträngen.
   [**Tips!** Ersätt punkter och komma med ett mellanslag eller ett "tomt" tecken (med `str_replace_all()`) men glöm inte att `.` är ett metatecken inom regular expressions.]

4. Gör om alla ord till endast gemener [**Tips!** `tolower()`]

5. Dela upp teckensträngen med `str_split()` för att få ut respektive ord. [**Tips!** Tänk på att du får ut en lista med denna funktion, inte en vektor. `unlist()` kan vara till hjälp.]

6. Räkna respektive ord och skapa en data.frame med respektive ord i kolumn 1 och antalet förekomster av detta ord i kolumn 2. Döp kolumn 1 till "Word", och kolumn 2 till "Freq". [**Tips!** `str_split()` och `table()`, `as.data.frame()`, `colnames()`]

7. Använd `str_c()` och `print()` för att baserat på datsetet ovan skriva ut följande mening "Det vanligaste ordet är [ordet] som förekom [frekvens] gånger i texten." [**Tips!** `which.max()`]

8. Returnera data.frame från steg 6.

Sammanfattningsvis: Funktionen `WordCount()` tar en godtycklig text (textvektor) och returnerar ett dataset med samtliga ord i textvektorn och ordens frekvenser. Funktionen ska också skriva ut meningen "Det vanligaste ordet är [ordet] som förekom [frekvens] gånger i texten.".

Kolla om testfallen nedan fungerar:

```
# test 1:
setwd("/home/joswi05/Dropbox/Rkurs/KursRprgm/Labs/DataFiles/")
text <- readLines("transtrom.txt", encoding = "latin1")
wordData1 <- WordCount(text = text)

[1] "Det vanligaste ordet ?r the som f?rekom 8 g?nger i texten."

head(wordData1[order(wordData1[, 2], decreasing = TRUE), ])

   Word Freq
59  the    8
1     a    6
2   and    4
24 have    3
40   of    3
62 they    3

# test 2:
setwd("/home/joswi05/Dropbox/Rkurs/KursRprgm/Labs/DataFiles/")
text2 <- readLines("transtrom2.txt", encoding = "latin1")
wordData2 <- WordCount(text = text2)
```

```
[1] "Det vanligaste ordet ?r han som f?rekom 3 g?nger i texten."
```

```
head(wordData2[order(wordData2[, 2], decreasing = TRUE), ])
```

```
      Word Freq
17     han    3
1     alla    2
2       av    1
3    bakum    1
4  började    1
5       de    1
```

5.2 Ändra ord

Med hjälp av funktionerna i `stringr` ska ni skriva en funktion som ska byta ut "han" och "hon" i en text mot "hen". Detta kan göras med hjälp av regular expressions. Det ska gå att göra denna operation med endast ett regular expression (om vi ignorerar versaler i hen). Det är dock inte ett måste att implementera detta med bara en rad kod.

Funktionen ska heta `henify(FileName=)`

- argumentet `FileName` ska antingen ange sökvägen till den textfil som ska användas som då slutar på ".txt" eller vara en teckenvektor innehållande en text.

Arbetsordning:

1. Läs in paktet i stringr i den aktuella R-sessionen.
2. Iakttag följade fall:
   (a) Om `FileName` är en sökväg till en fil så ska filen med texten läsas in som och sparas i en teckenvektor. Detta kan du kolla om längden av FileName=1 och om de fyra sista teckenen är ".txt". Tips: `str_sub()`, `readLines()`, tänk på att svenska texter ska kunna användas, så kolla på 2a i sektionen "Texthantering ...".
   (b) Om första elementet i FileName inte slutar på ".txt" så ska FileName användas som en textvektor i resten av funktionen.
3. Använd nu `str_replace_all()` för att ersätta Han/han och Hon/hon med Hen/hen. Här behöver ni använda regular expressions. Iakttag följande:
   - Tänk på att han/hon ska kunna börja med stor eller liten bokstav.
   - Han/hon kan komma först i strängen, någonstans i mitten, eller sist.

- Vi ska inte välja ord där han/hon är en delmängd, tex “storhandla” ska inte bli “storhendla” eller “hona” ska inte bli “hena”.

  Tips: Kolla på demo_13 för att se exempel på hur regular expression kan användas och hur `str_replace_all()` kan användas tillsammans med “backreferences”.

4. Returnera den ändrade texten som en textvektor.

Kolla om testfallen nedan fungerar:

```
setwd("/home/joswi05/Dropbox/Rkurs/KursRprgm/Labs/DataFiles/")
# test 1:
henify(FileName = "transtrom2.txt")

 [1] "När hen kom ner på gatan efter kärleksmötet"
 [2] "virvlade snö i luften."
 [3] "Vintern hade kommit"
 [4] "medan de låg hos varann."
 [5] "Natten lyste vit."
 [6] "Hen gick fort av glädje..."
 [7] "Hela staden sluttade."
 [8] "Förbipasserande leenden -"
 [9] "alla log bakum uppgällda kragar."
[10] "Det var fritt!"
[11] "Och alla frågetecken började sjunga om Guds tillvaro."
[12] "Så tyckte hen!"

# test 2:
txt1 <- c("Hon, händer hör, han", "handlar du?", "honartad hona han.")
txt1

[1] "Hon, händer hör, han" "handlar du?"          "honartad hona han."

henify(txt1)

[1] "Hen, händer hör, hen" "handlar du?"          "honartad hona hen."
```

Nu är du klar!