SCRIPT OCH IMPORTERA DATA

Niklas Edvall

Script

För att slippa skriva saker för hand hela tiden är steg ett när man jobbar med R att skriva ett script för sin analys. I Rstudio finns ikonen för att skapa nya filer uppe till vänster och där väljer man *R script* för att skapa ett nytt sådant.

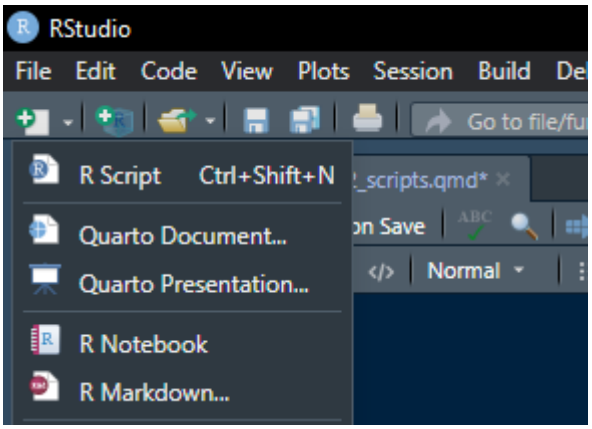


Figure 1: New File - R Script

I sitt script kan man nu skriva hela sin analys-pipeline från början till slut. Vill vi t.ex undersöka om det finns en statistisk signifikant skillnad mellan män och kvinnor att välja chokladglass före vanilj när de får frågan: Skulle du hellre äta choklad- än vaniljglass? Kan vi skriva följande analys-script.

Vi kodar kön som `M` eller `F` i variabeln `sex`, och om man svarade ja som `Y` eller nej som `N` i variabeln `ic.choco`

```
#Create variable for sex
sex <- c("M", "F", "M", "M", "F", "M", "F", "F", "M", "F", "F")

#Create variable for ice cream preference
ic.choco <- c("Y", "Y", "Y", "Y", "N", "N", "N", "N", "Y", "N", "N")

#Fishers exact test
fisher.test(sex, ic.choco)
```

```
    Fisher's Exact Test for Count Data

data:  sex and ic.choco
p-value = 0.08009
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
    0.6013099 1160.3870855
sample estimates:
odds ratio
  13.45266
```

Vi ser att testet resulterar i ett p-värde = 0.08, vilket betyder att vi inte kan påvisa statistiskt signifikant skillnad mellan grupperna på den klassiska 5%-nivån för statistisk signifikans.

Importera data och packages

Självklart är det galenskap att skriva in sin data manuellt med hjälp av funktionen `c()`

Istället vill vi importera data från en separat datafil som vi sparat. Det finns flera olika sätt att göra detta på, här använder vi ett paket som heter `readr` för att importera vår data som finns sparad i en csv-fil.

Paket? R har en massor av inbyggda funktioner men det finns oändligt många fler skräddarsydda funktioner att installera om man vill ha särskild funktionalitet. Dessa kommer i form av olika *packages* eller paket, och man kan ladda de man specifikt behöver för sitt aktuella script.

Första gången man vill använda ett paket måste man installera det med funktionen `install.packages()`, för att installera paketet `readr` anger vi `install.packages("readr")`.

När man vill använda ett paket i sitt script laddar man det med `library()`. Så, för att ladda vår data-fil med `readr` till en data frame vi kallar `dat` börjar vårat script med:

Fördelen med `readr` är att man också kan gå till *import dataset* och välja att importera data från en text-fil med `readr`. Då kan man även välja att t.ex exkludera vissa variabler eller ange olika format för variabler. Dialogrutan för att importera data skapar även en kod-snutt man klistra in i sitt script för att spara exakt parametrarna man använt för att importera data.

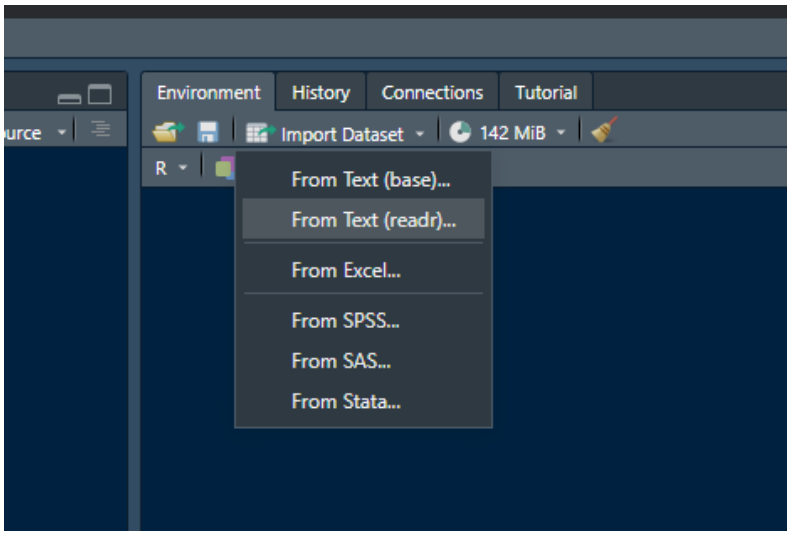


Figure 2: Import dataset

Vår exempel-data

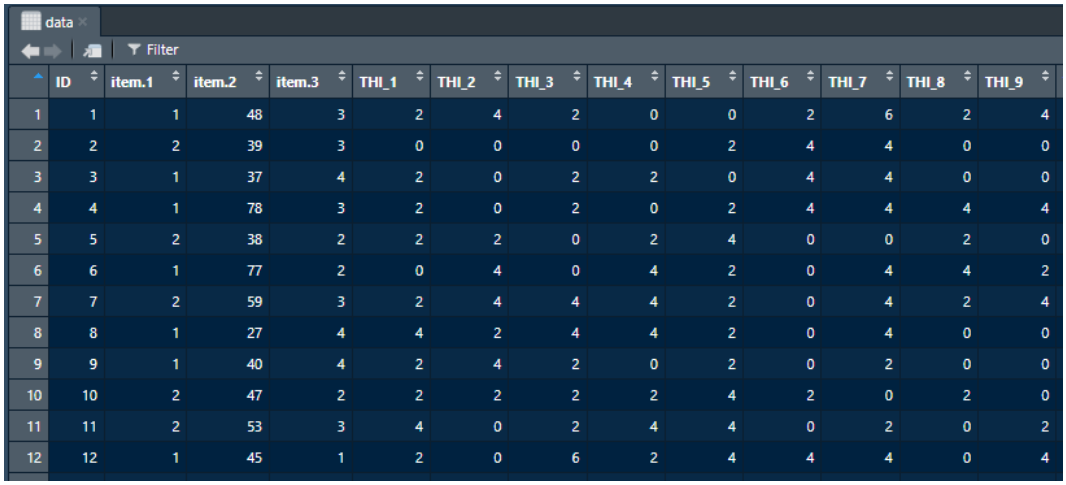
Vår exempel-data innehåller ett unikt ID-nummer per deltagare, info om kön, ålder och självrapporterad hörselstatus. Hörtrösklar vid fyra frekvenser (0.5, 1, 2 och 4 kHz) per öra och de 25 frågor som återfinns på frågeformuläret Tinnitus Handicap Inventory (THI). Vi kommer använda datan för att se om vi kan besvara:

1. Överensstämmer självrapporterad hörsel med tonmedelvärde?

2. Är det skillnad mellan hur män och kvinnor besvarar THI?
3. Förändras hörseln med åldern?

Inspektera data

För att få en första överblick kan vi inspektera vår data. Enklast är att klicka på variabeln *dat* för vår data frame i det övre fönstret till höger i Rstudio.



data

Filter

	ID	item.1	item.2	item.3	THI_1	THI_2	THI_3	THI_4	THI_5	THI_6	THI_7	THI_8	THI_9
1	1	1	48	3	2	4	2	0	0	2	6	2	4
2	2	2	39	3	0	0	0	0	2	4	4	0	0
3	3	1	37	4	2	0	2	2	0	4	4	0	0
4	4	1	78	3	2	0	2	0	2	4	4	4	4
5	5	2	38	2	2	2	0	2	4	0	0	2	0
6	6	1	77	2	0	4	0	4	2	0	4	4	2
7	7	2	59	3	2	4	4	4	2	0	4	2	4
8	8	1	27	4	4	2	4	4	2	0	4	0	0
9	9	1	40	4	2	4	2	0	2	0	2	0	0
10	10	2	47	2	2	2	2	2	4	2	0	2	0
11	11	2	53	3	4	0	2	4	4	0	2	0	2
12	12	1	45	1	2	0	6	2	4	4	4	0	4

Figure 3: Vår data frame "dat"

Nedan ser vi med funktionen `dim()` att vår data har dimension 200x34, dvs 200 observationer (rader) för 37 olika variabler (kolumner). Funktionen `names()` returnerar namnen för alla kolumner i vår data. I funktionen `head()` specificerar vi att få tillbaka de första 3 raderna och 8 kolumnerna.

```
#Dimensions of dat frame
dim(dat)
```

```
[1] 200  37
```

```
#Column names in dat frame
names(dat)
```

```
 [1] "ID"     "item.1" "item.2" "item.3" "THI_1"  "THI_2"  "THI_3"  "THI_4"
 [9] "THI_5"  "THI_6"  "THI_7"  "THI_8"  "THI_9"  "THI_10" "THI_11" "THI_12"
[17] "THI_13" "THI_14" "THI_15" "THI_16" "THI_17" "THI_18" "THI_19" "THI_20"
[25] "THI_21" "THI_22" "THI_23" "THI_24" "THI_25" "R500"   "R1000"  "R2000"
[33] "R4000"  "L500"   "L1000"  "L2000"  "L4000"
```

```
#Look at first 3 rows and 8 columns of dat
head(dat, c(3,8))
```

```
# A tibble: 3 x 8
     ID item.1 item.2 item.3 THI_1 THI_2 THI_3 THI_4
  <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
```

```
1      1      1      48      3      2      4      2      0
2      2      2      39      3      0      0      0      0
3      3      1      37      4      2      0      2      2
```

Notera att vi även ser vilken **typ** av variabel vi har i returen för `head()`, under kolumnens namn anges `<dbl>`, en förkortning för *double precision floating point* vilket betyder numerisk data. Vi behöver städa lite i vår data för att specificera vad som är numerisk data och vad som är kategorisk data.

Städa data

Kategoriska variabler

Kategoriska variabler i R kallas för faktor-variabler. Vi specificerar en faktor-variabel med funktionen `factor()`. Vill vi ange att variabeln *item.1* i vår data är en faktor är det dock ingen idé att bara ange:

```
#Make factor of item.1
factor(dat$item.1)
```

```
  [1] 1 2 1 1 2 1 2 1 1 2 2 1 2 1 2 1 2 1 1 2 1 2 1 1 2 2 1 2 1 2 1 2 2 1 2 1 2
 [38] 1 1 2 2 1 2 1 2 2 1 2 1 1 2 2 1 1 1 1 2 1 2 1 2 1 2 1 1 1 2 1 1 1 1 1 1 1
 [75] 1 2 2 1 2 2 1 2 2 2 1 2 2 1 2 1 1 2 2 1 2 2 1 2 1 1 1 1 1 2 1 2 2 1 2 1 1
[112] 1 1 2 1 1 2 1 1 1 2 1 1 1 1 2 1 1 2 2 1 1 2 1 2 2 2 1 1 2 2 2 2 2 1 1 2 1
[149] 2 1 1 2 1 1 1 2 2 1 2 2 1 2 2 1 1 1 2 1 2 1 1 1 2 1 2 2 2 1 2 1 2 1 1 1 2
[186] 2 2 1 2 1 2 1 1 2 1 1 1 1 1 1
Levels: 1 2
```

Visserligen returneras *item.1* som en faktor, men vi måste spara den outputen till vår data frame genom att ange:

```
#Make factor and write to data frame
dat$item.1 <- factor(dat$item.1)
```

Vi råkar veta att *item.1* kodar för kön och kan göra detta tydligt genom att specificera ytterligare parametrar i funktionen `factor()` enligt nedan. Parametrarna `levels` (vilka kategorier variabeln innehåller) och `labels` (vad vi vill namnge dessa) åtskiljs med kommatecken.

Kom ihåg att `? factor` alltid visar hjälpavsnittet för funktionen och ger exempel på hur den kan användas.

```
#Make factor, specify levels & labels, and write to data frame
dat$item.1 <- factor(dat$item.1,
                     levels = c(1,2),
                     labels = c("M", "F"))
```

Ändra namn på variabler i en data frame

Det vore även smidigt att ge kolumnen ett mer beskrivande namn. Minns tillbaka hur funktionen `names()` gav oss alla kolumn-namn ovan, vi kollar igen:

```
#Column names in data frame
names(dat)
```

```
 [1] "ID"     "item.1" "item.2" "item.3" "THI_1"  "THI_2"  "THI_3"  "THI_4"
 [9] "THI_5"  "THI_6"  "THI_7"  "THI_8"  "THI_9"  "THI_10" "THI_11" "THI_12"
[17] "THI_13" "THI_14" "THI_15" "THI_16" "THI_17" "THI_18" "THI_19" "THI_20"
[25] "THI_21" "THI_22" "THI_23" "THI_24" "THI_25" "R500"   "R1000"  "R2000"
[33] "R4000"  "L500"   "L1000"  "L2000"  "L4000"
```

Vi kan använda `names()` för döpa om variabler i en data frame. Vi ser att *item.1* är den andra variabeln i *data*, dvs har index 2. Men, det är strikt förbjudet att använda detta för att t.ex skriva: `names(dat)[2] <- "sex"`

Om kolumnen skulle få ett annat index, vilket ofta händer, kommer det sluta med att vi döper om någon annan kolumn och har förstört vår data totalt.

Istället hänvisar vi till ett dynamiskt index, `names(dat) == "item.1"`, som döper om alla kolumner med namnet *item.1* till *sex*

```
#Rename variable for sex
names(dat)[names(dat) == "item.1"] <- "sex"
```

På samma sätt definerar vi sen vairabeln *item.3* som faktor-variabel och ändrar namn på den och variabel *item.2*

Skapa nya variabler

Vårat dataset innehåller hörtrösklar vid fyra frekvenser (0.5, 1, 2 och 4 kHz) per öra som vi kan använda för att räkna ut tonmedelvärde (*Pure Tone Average*; PTA4). Det är lätt att skapa/ange en ny variabel i vår data frame med $ från medelvärdet av de fyra andra variablerna. Radbyte spelar ingen roll i scriptet, så länge alla symboler är på plats. Här skrivs varje referens till en variabel på egen rad för att det ska vara lättläsligt.

```
#Create variable for PTA4 Right
dat$PTA4.R <- (dat$R500 +
               dat$R1000 +
               dat$R2000 +
               dat$R4000) / 4

#Create variable for PTA4 Left
dat$PTA4.L <- (dat$L500 +
               dat$L1000 +
               dat$L2000 +
               dat$L4000) / 4
```

Vi har 25 variabler som heter THI_1, THI_2, THI_3.. osv. till THI_25. Dessa representerar svar på ett frågeformulär om tinnitus kallat *Tinnitus Handicap Inventory*. Svaren är redan kodade som antingen 0, 2 eller 4 vilket motsvarar de poäng man får för svarsalternativen. Maxpoäng är alltså 25 * 4 = 100, och ju högre poäng desto mer besvärad är man av tinnitus.

Det vore intressant att skapa en ny variabel med varje försökspersons totala poäng på THI. Vi namnger denna variabel som *THIscore*. Det finns flera sätt att göra detta på. Enklast vore att helt enkelt summera de 25 variablerna:

```
#Sum all THI-variables to new variable THIscore
dat$THIscore <- dat$THI_1 + dat$THI_2 + dat$THI_3 ...
```

Det är funktionsdugligt, men det blir mycket att skriva och blir både svårläst, oflexibelt och ostabilt om t.ex en variabel får ett nytt namn eller kodas på något annat sätt. Det finns alltid flera sätt att åstadkomma samma sak på med R, och vissa är smidigare än andra.

I det här fallet kan vi skapa en ny variabel som vi kallar *THI.names* med hjälp av funktionen `paste()` som klistrar ihop (eller 'konkatenerar') olika saker med en avskiljare (eller 'separator') som vi specificerar. Här konkatenerar vi bokstäverna "THI" med talen 1 till 25 avskiljda med ett understreck "_"

På så sätt innehåller då vår nya variabel *THI_names* namnen på alla de kolumner vi är intresserade av att summera.

```
#Create variable of column names relevant to THI
THI.names <- paste("THI", 1:25, sep="_")
```

Vi kan sen använda funktionen `apply()` för att applicera funktionen `sum()` på alla rader (rader specificerar vi med `MARGIN = 1`) i vår data frame som har ett namn som finns i variablen *THI.names* och skriva resultatet till en ny variabel som vi igen kallar *THI.score*

```
#Calculate total THI score per subject
dat$THIscore <- apply(dat[,THI.names], MARGIN = 1, FUN = sum)
```

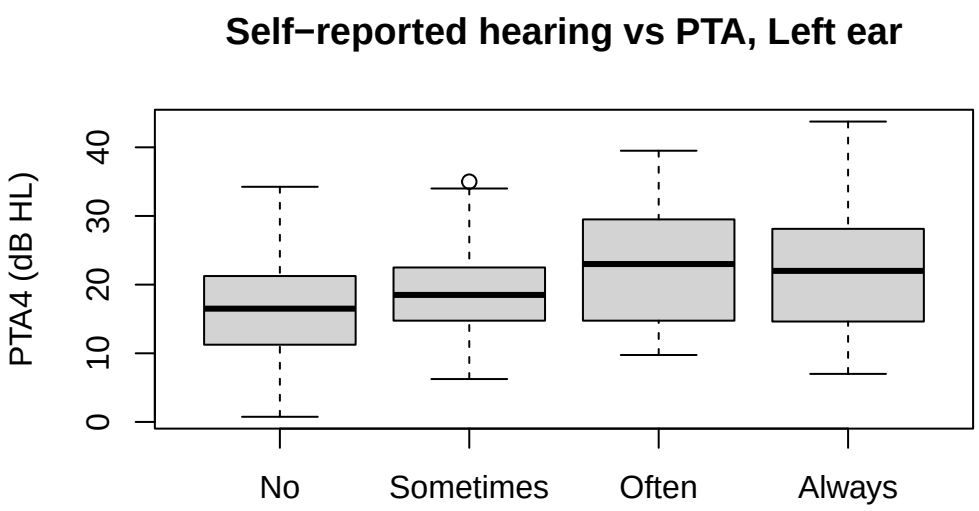
Analys

Vi kan nu besvara våra frågeställningar från början av dokumentet.

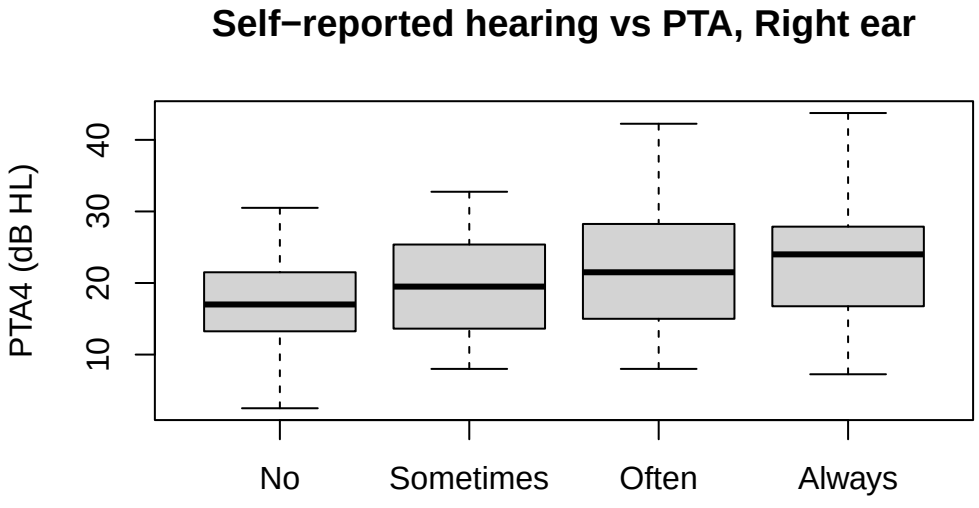
1. Överensstämmer självrapporterad hörsel med tonmedelvärde?

Vi börjar med att göra en enkel plot. När vi ger funktionen en faktor-variabel och en numerisk variabel skapas automatiskt en boxplot. Vi passar på att specificera en titel för ploten, och en rubrik för X- respektive Y-axel.

```
plot(dat$hearing, dat$PTA4.L,
     main = "Self-reported hearing vs PTA, Left ear",
     xlab = "Do you experience difficulties hearing?",
     ylab = "PTA4 (dB HL)")
```




```
plot(dat$hearing, dat$PTA4.R,
     main = "Self-reported hearing vs PTA, Right ear",
     xlab = "Do you experience difficulties hearing?",
     ylab = "PTA4 (dB HL)")
```



Vi börjar enkelt, men är du estetiskt lagd finns det massor av verktyg för att göra vackra figurer med R, se till exempel galleriet för paketet ggplot2.

För statistisk analys av om det är skillnad mellan de fyra grupperna gör vi en ANOVA med funktionen `aov()` för vänster respektive höger öras tonmedelvärde, spar resultatet från beräkningen i en ny variabel för vänster och höger öra separat och tittar på en sammanfattning av resultatet med `summary()`

```r
#Calculate anova for PTA ~ hearing
anova.L <- aov(data = dat, PTA4.L ~ hearing)
anova.R <- aov(data = dat, PTA4.R ~ hearing)

#Show ANOVA result summary
summary(anova.L)
```

```
             Df Sum Sq Mean Sq F value   Pr(>F)
hearing       3   1263   421.0   6.502 0.000324 ***
Residuals   196  12689    64.7
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(anova.R)
```

```
             Df Sum Sq Mean Sq F value   Pr(>F)
hearing       3   1174   391.3   6.533 0.000311 ***
Residuals   196  11739    59.9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

I båda fallen verkar det finnas ett starkt samband mellan tonmedelvärde och självrapporterade hörselsvårigheter ($p < 0.001$).

Vi gör ett *pairwise t-test* som post-hoc analys som också korrigerar p-värdet för flera jämförelser och returnerar en tabell över p-värdet som resulterar när alla fyra grupper jämförs sinsemellan.

```r
#Pairwise t-test Left ear PTA and self-reported hearing
pairwise.t.test(dat$PTA4.L, dat$hearing,
                p.adjust.method = "bonferroni")
```

```
    Pairwise comparisons using t tests with pooled SD

data:  dat$PTA4.L and dat$hearing

          No      Sometimes Often
Sometimes 0.62311 -         -
Often     0.00083 0.25261   -
Always    0.00293 0.53583   1.00000

P value adjustment method: bonferroni
```

```r
#Pairwise t-test Right ear PTA and self-reported hearing
pairwise.t.test(dat$PTA4.R, dat$hearing,
                p.adjust.method = "bonferroni")
```

```
    Pairwise comparisons using t tests with pooled SD

data:  dat$PTA4.R and dat$hearing

          No      Sometimes Often
Sometimes 0.29714 -         -
Often     0.00299 0.99592   -
Always    0.00058 0.46241   1.00000

P value adjustment method: bonferroni
```

2. Är det skillnad mellan hur män och kvinnor besvarar THI?

För att besvara denna fråga gör vi ett enkel chi-square-test och en box-plot på samma sätt som innan.

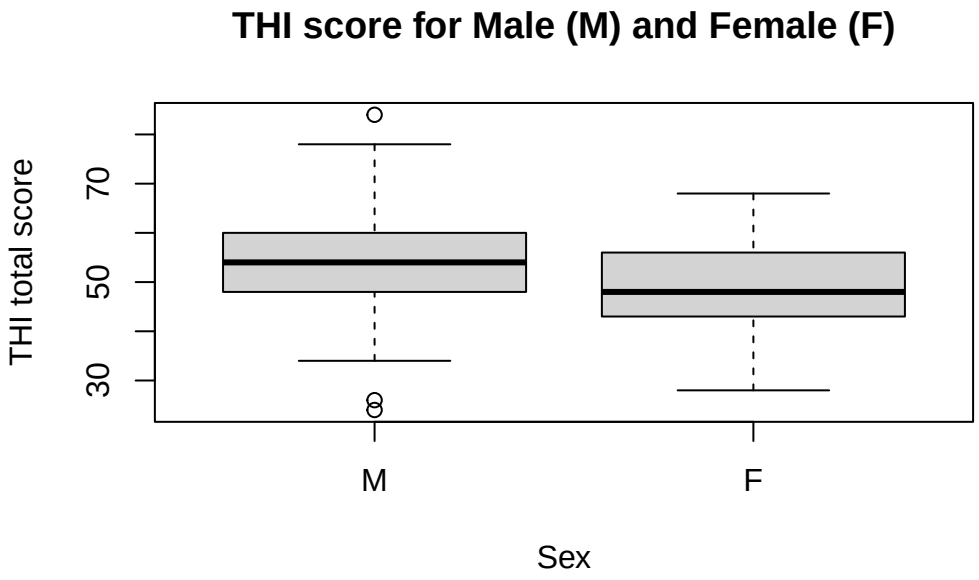
```
#Chi square test of THI score for sex
chisq.test(dat$sex, dat$THIscore)
```

```
Warning in chisq.test(dat$sex, dat$THIscore): Chi-squared approximation may be
incorrect

    Pearson's Chi-squared test

data:  dat$sex and dat$THIscore
X-squared = 35.923, df = 27, p-value = 0.117
```

```
#Box plot of THI score for sex
plot(dat$sex, dat$THIscore,
     main = "THI score for Male (M) and Female (F)",
     xlab = "Sex",
     ylab = "THI total score")
```



Vår plot verkar indikera att män fått lite högre poäng på THI men vårat test resultat 35.9, p = 0.117 påvisar ingen statistiskt signifikant skillnad mellan grupperna.

3. Förändras hörseln med åldern?

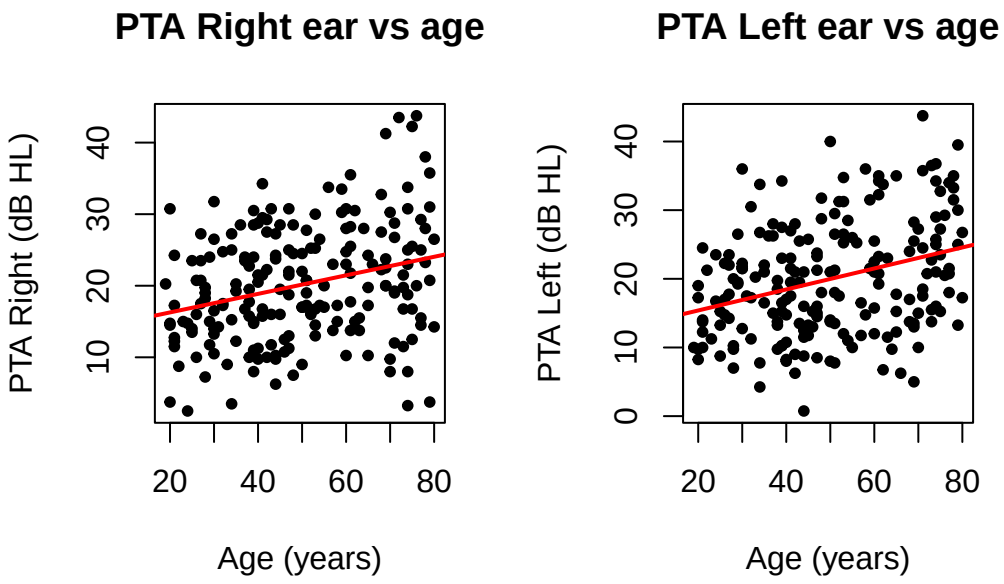
```
#Set plot space to two columns
par(mfrow=c(1,2))

#Scatter plot of PTA RIGHT as function of age
plot(dat$age, dat$PTA4.R,
     pch = 20,
     main = "PTA Right ear vs age",
     xlab = "Age (years)",
     ylab = "PTA Right (dB HL)")

#Best fit linear regression RIGHT
abline(lm(data = dat, PTA4.R ~ age), lwd = 2, col = "red")

#Scatter plot of PTA LEFT as function of age
plot(dat$age, dat$PTA4.L,
     pch = 20,
     main = "PTA Left ear vs age",
     xlab = "Age (years)",
     ylab = "PTA Left (dB HL)")

#Best fit linear regression LEFT
abline(lm(data = dat, PTA4.L ~ age), lwd = 2, col = "red")
```




```
#Save linear regression model to variable
linear.reg.L <- lm(data = dat, PTA4.L ~ age)
linear.reg.R <- lm(data = dat, PTA4.R ~ age)
```

```r
#Print summary of linear regression model
summary(linear.reg.L)
```

```
Call:
lm(formula = PTA4.L ~ age, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-18.3107  -6.1398  -0.1393   5.3527  20.5579

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.32806    1.69318   7.281 7.64e-12 ***
age          0.15301    0.03216   4.758 3.76e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.952 on 198 degrees of freedom
Multiple R-squared:  0.1026,    Adjusted R-squared:  0.09808
F-statistic: 22.64 on 1 and 198 DF,  p-value: 3.758e-06
```

```r
summary(linear.reg.R)
```

```
Call:
lm(formula = PTA4.R ~ age, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max
-20.1685  -4.8667  -0.4338   5.3292  20.4885

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.68233    1.64973   8.294 1.66e-14 ***
age          0.12957    0.03133   4.135 5.23e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.748 on 198 degrees of freedom
Multiple R-squared:  0.07951,   Adjusted R-squared:  0.07486
F-statistic:  17.1 on 1 and 198 DF,  p-value: 5.232e-05
```