Datorövning 1 i Regressions- och tidsserieanalys

Maria Anna Di Lucca (SAS version) och Mattias Villani (översättning till R notebook)

Denna datorövning tränar dig i hur man:
1. beräknar en skattning av korrelationen, ρ, mellan två variabler.
2. utför ett test om noll korrelation mellan två variabler, $H_0 : \rho = 0$.
3. plottar en beroende variabel mot en oberoende variabel.
4. skattar en enkel linjär regressionsmodell.
5. beräknar och plottar konfidens- och prediktionsband för regressionslinjen.
6. beräknar konfidensintervall för parameterarna.

Uppstart - ange arbetsfolder och installera och ladda in R paket

Vi börjar med att tala om för R var våra datafiler finns (working directory). Ändra sökvägen i `setwd` kommandot nedan till den mapp som du lagt datormaterialet i på din dator. På Windowsdatorer är sökvägen av typen `C:\Users\DinNamn` eller något liknande. På Mac eller Linux (som används nedan) är det av typen `/home/DittNamn`.

```
setwd("/home/mv/Dropbox/Teaching/Regression/DatorOvningar/")
```

Vi kommer behöva paketet `readxl` för att läsa in datamaterialet från en Excelfil. Paketet installeras genom kommandot `install.packages("readxl")`. Efter det laddar vi paketet med:

```
library("readxl")
```

Läsa in datamaterial i R

För att illustrera hur man gör enkel linjär regression i R använder vi *Table 5.1* i boken "Applied Regression Analysis and Other Multivariable Methods" av Kleinbaum et.al. Datamaterialet innehåller $n = 30$ observationer på variablerna **SBP**, systolic blood pressure och **Age**. Vi kan läsa in datamaterialet i R från Excel-filen *bloodpressure.xlsx* med följande funktion från paketet `readxl`:

```
bloodpressure <- read_excel("bloodpressure.xlsx")
```

`bloodpressure` är nu en s k *data frame* (R's version av en tabell) med två kolumner `SBP` och `Age`. Genom att använda `$`-tecknet kan vi komma åt respektive variabel: `bloodpressure$SBP` och `bloodpressure$Age`.

Uppgift 1 - Beräkna korrelationen mellan två variabler samt utföra test om $\rho = 0$

För att beräkna korrelationen mellan två variabler kan man använda funktionen `cor`. Om vi också vill testa nollhypotesen $H_0 : \rho = 0$ använder vi funktionen `cor.test`:

```
cor.test(bloodpressure$SBP, bloodpressure$Age)
```

```
##
##  Pearson's product-moment correlation
##
## data:  bloodpressure$SBP and bloodpressure$Age
```

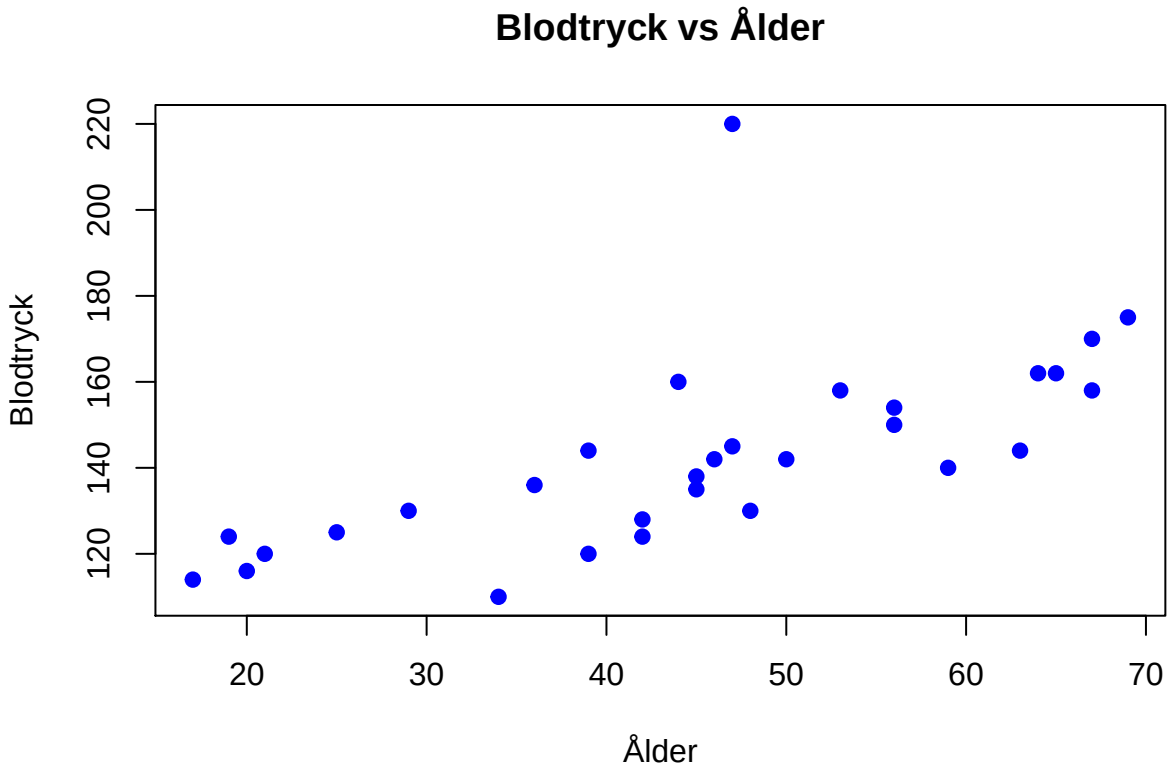
```
## t = 4.6184, df = 28, p-value = 7.867e-05
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.3895932 0.8228923
## sample estimates:
##       cor
## 0.6575673
```

Vi ser att korrelationen är 0.6575673 och att testet $H_0 : \rho = 0$ har ett p-värde som är mycket litet ($7.867e-05 = 0.00007867$) när vi testar mot alternativhypotesen $H_A : \rho \neq 0$. Vi förkastar alltså nollhypotesen på alla rimliga signifikansnivåer (t ex 0.01).

Uppgift 2 - Plotta en beroende variabel mot en oberoende variabel

I exemplet från boken är variabeln blodtryck den beroende variabeln och variabeln ålder är förklarande variabel. Vi vill nu plotta dessa två för att se om det kan föreligga ett linjärt samband mellan dem.

```
plot(bloodpressure$Age, bloodpressure$SBP, xlab = "Ålder", ylab = "Blodtryck",
     main = "Blodtryck vs Ålder", col = "blue", pch = 19)
```

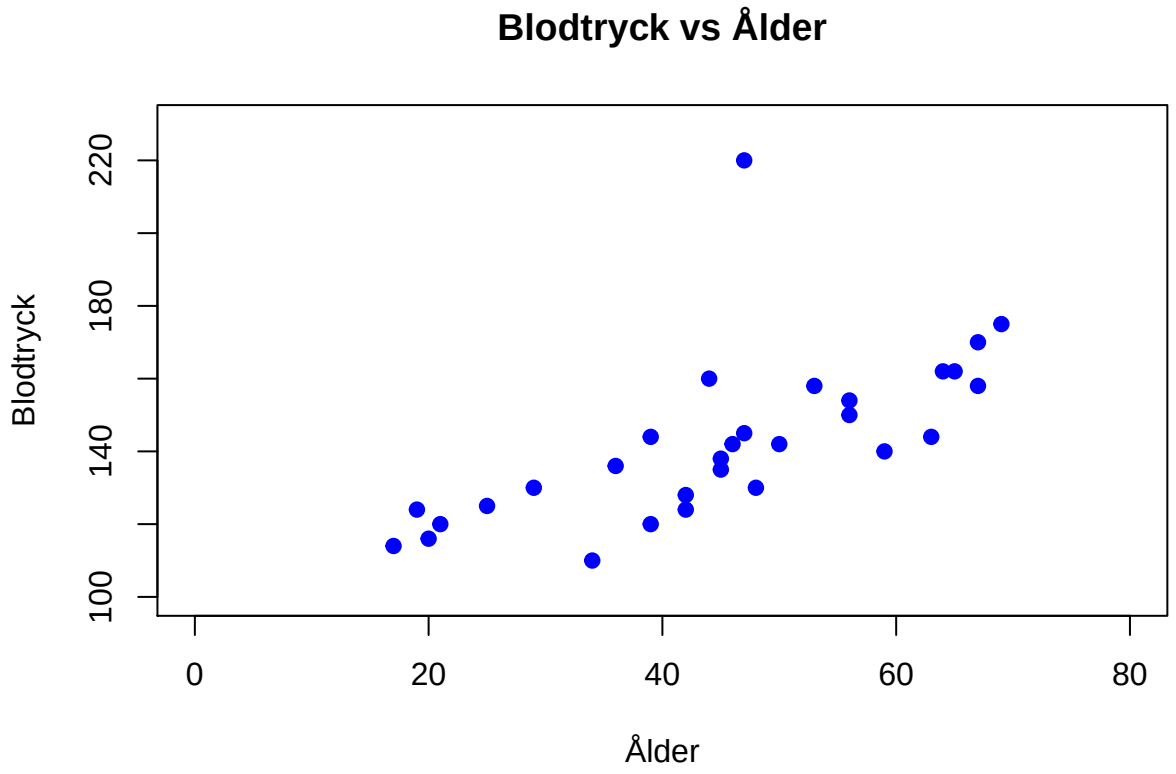


I koden ovan anger vi namnet på x-axeln med argumentet `xlab`, `ylab` för namnet på y-axeln, `main` för grafens titel, `col` för färgen på punkterna i plotten och slutligen sätter vi `pch` (står för plotting character) till 16 för att få runda ifyllda punkter. Här kan du läsa lite mer om olika s k parametrar som kan anges när man gör plottar i R, inklusive en lista med alla plotsymboler.

Vi kan också styra hur axlar i grafen ska se ut med parametrarna `xlim` och `ylim`:

```
plot(bloodpressure$Age, bloodpressure$SBP, xlab = "Ålder", ylab = "Blodtryck",
     main = "Blodtryck vs Ålder", col = "blue", pch = 19,
```

```
     xlim = c(0, 80), ylim = c(100,230))
```



Uppgift 3 - Skatta en enkel linjär regressionsmodell

Vi kan se i plotten ovan att datamaterialet verkar kunna beskrivas av en enkel linjär regressionsmodell. Funktionen `lm` i R skattar en sådan modell:

```
lmFit <- lm(SBP ~ Age, data = bloodpressure)
```

`lm`-funktionen behöver veta två saker:

- regressionsmodellen i form av en s k **R-formel** vilket skrivs´ som `SBP ~ Age` och
- vilket *datamaterial* (data frame) `data` som dessa variabler finns i.

Notera hur den s k tilde-symbolen `~` används för att tala om för R att `SBP` är den beroende variabeln (till vänster om `~`) och `Age` är den förklarande variabeln (till höger om `~`). Vi kommer längre fram se hur vi kan utöka denna R-formel till fler än en förklarande variabel. Resultatet av modellanpassningen skickar vi in i ett nytt objekt som jag har valt att döpa till `lmFit`, men du kan döpa det till vad du vill. Objektet `lmFit` är en speciell typ av variabel (av klassen `lm`, som det heter på programmeringsspråk) som innehåller en massa resultat från modellanpassningen. Vi kan anropa olika funktioner på `lmFit`-objektet för att få olika information. Vi kan t ex få en utskrift av modellanpassningen genom att skriva

```
summary(lmFit)
```

```
##
## Call:
## lm(formula = SBP ~ Age, data = bloodpressure)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.724  -6.994  -0.520   2.931  75.654
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  98.7147    10.0005   9.871 1.28e-10 ***
## Age           0.9709     0.2102   4.618 7.87e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.31 on 28 degrees of freedom
## Multiple R-squared:  0.4324, Adjusted R-squared:  0.4121
## F-statistic: 21.33 on 1 and 28 DF,  p-value: 7.867e-05
```

där vi t ex kan se att modellens intercept skattas till 98.7147 och modellens regressionskoefficient för variabeln `Age` är 0.9709; blodtrycket ökar alltså i genomsnitt med en enhet per levnadsår.

Vi kan också se att både interceptet och regressionskoefficienten för variabeln `Age` är signifikant skilda från noll eftersom p-värdena för dessa två test är mycket små ($1.28e-10$ respektive $7.87e-05$); stjärnorna till höger om p-värdet ger en snabb indikation om signifikansen (*** betyder ett p-värde mindre än 0.001).

Det test vi utför för interceptet är ett t-test med en testvariabel som är kvoten av det skattade interceptet ($\hat{\beta}_0$) minus värdet under nollhypotesen, som ju är noll, och standardfelet för denna skattning ($s_{\hat{\beta}_0}$):

$$t = \frac{\hat{\beta}_0 - 0}{s_{\hat{\beta}_0}}$$

Vi kan utläsa från utskriften (Std. Error är standardfelet) att

$$t = \frac{98.7147}{10.0005} = 9.866537,$$

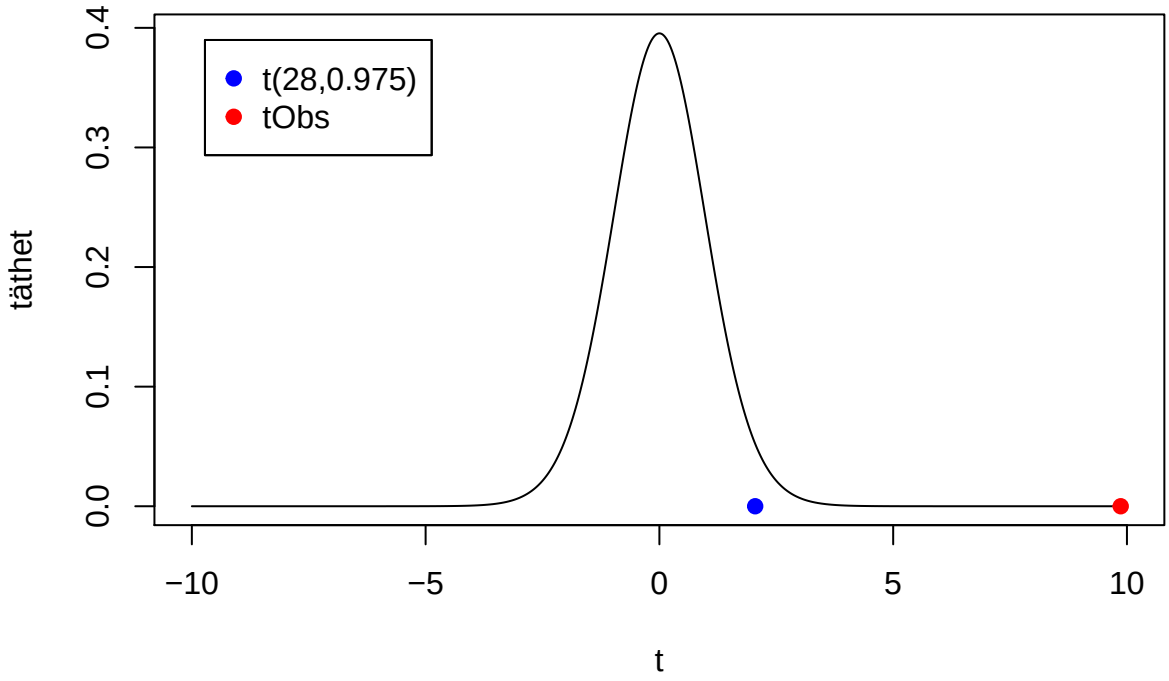
vilket bortsett från avrundningsfel är just det t-värde för interceptet som anges i utskriften. Vi vet också att denna teststatistiska följer en t-fördelning med $n - 2 = 30 - 2 = 28$ frihetsgrader, vilket också hittas i utskriften. Eftersom testet är två-sidigt ska vi utläsa det kritiska värdet $t_{28,0.975}$ ur en tabell, dvs det värde i t-fördelning med 28 frihetsgrader som har 97.5% sannolikhetsmassa till vänster i fördelningen (eller $\alpha/2 = 0.025$ till höger om den kritiska gränsen, är $\alpha = 0.05$ för vårt test på 5% nivån). Vi behöver dock ingen tabell utan kan använda R för denna beräkning med hjälp av funktionen `qt` (bokstaven q står för quantile och t för t-fördelningen):

```
qt(0.975, df = 28)
```

```
## [1] 2.048407
```

Eftersom $t_{obs} = 9.8666537$ är större än det kritiska värdet $t_{28,0.975} = 2.048407$ så förkastar vi nollhypotesen att interceptet är noll. Vi kan för tydlighets skull plotta t-fördelningen under nollhypotesen tillsammans med det kritiska värdet och det observerade värdet på teststatistikan:

```
tGrid = seq(-10,10,by = 0.01)
tCrit = qt(0.975, df = 28)
tObs = 9.866537
plot(tGrid, dt(tGrid, df = 28), type = "l", xlab = "t", ylab = "täthet")
points(tCrit,0, col = "blue", pch = 19)
points(9.866537,0, col = "red", pch = 19)
legend(x = "topleft", inset=.05, legend = c("t(28,0.975)", "tObs"), pch = c(19,19),
       cex = c(1,1), pt.lwd = c(NA,NA), col = c("blue","red") )
```



Vi ser från figuren ovan att de observerade t-värdet (röda punkten) är väldigt extremt och inte alls sannolikt om det skulle vara så att nollhypotesen faktiskt var sann. Det visste vi ju redan från det låga p-värdet, men det är alltid bra med en bild för att visualisera resultatet.

Vi kan göra liknande beräkningar för signifikanstestet av lutningskoefficienten β_1.

Slutligen kan vi se från utskriften ovan att modellens förklaringgrad är $R^2 = 0.4324$, dvs att ca 43 % av variationen i blodtryck över olika individer förklaras av deras ålder. Övriga 57 % förklaras antagligen till stor del av variabler som inte ingår i modellen t ex träningsvanor, vikt och genetiska faktorer.

Funktionen `summary` ger inte variansanalys (ANOVA) tabellen för regressionen, men det kan lätt fås genom att använda `anova` funktionen på `lmFit`-objektet:

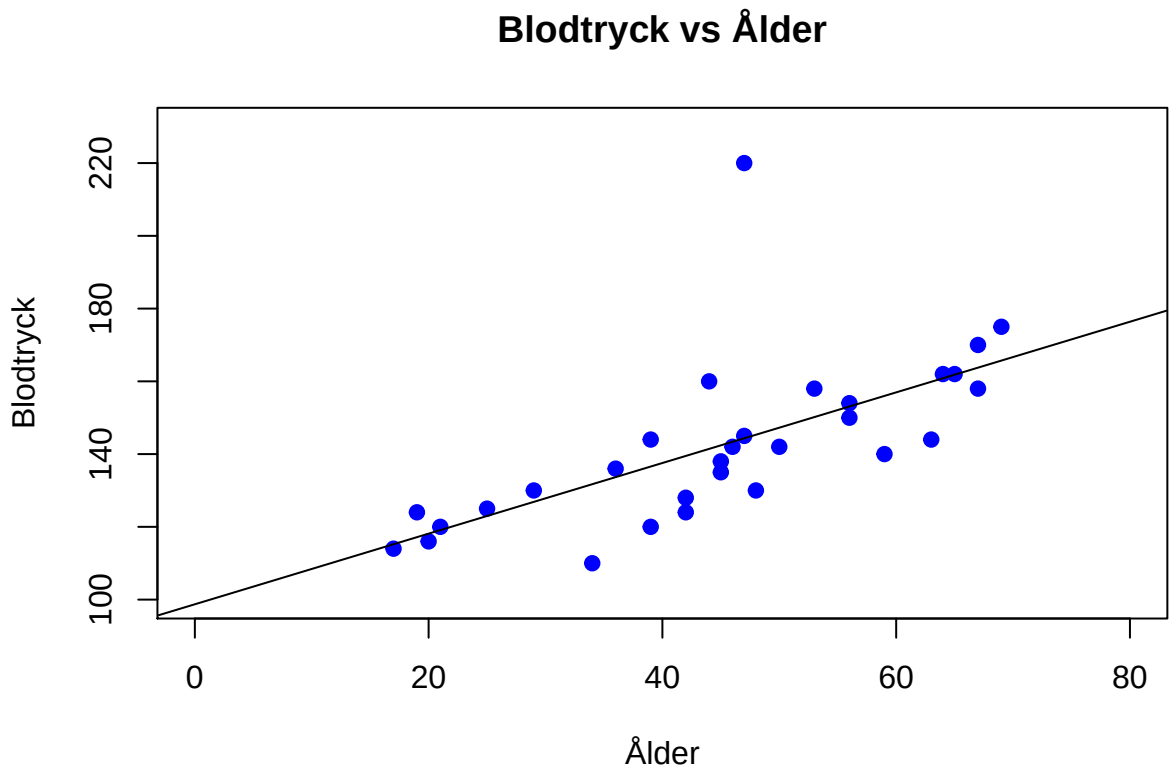
```
anova(lmFit)
```

```
## Analysis of Variance Table
##
## Response: SBP
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Age        1 6394.0  6394.0   21.33 7.867e-05 ***
## Residuals 28 8393.4   299.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Ovan använde vi `summary`-funktionen på modellanpassningsobjektet `lmFit`. Vi kan också använda funktionen `plot` på `lmFit`. Funktionen `plot` fungerar nämligen olika beroende på vilket objekt man använder den för. Prova!

För att rita in den skattade regressionlinjen i grafen ovan använder vi bara funktionen `abline` på vårt kära `lmFit` objekt:

```r
plot(bloodpressure$Age, bloodpressure$SBP, xlab = "Ålder", ylab = "Blodtryck",
     main = "Blodtryck vs Ålder", col = "blue", pch = 19,
     xlim = c(0, 80), ylim = c(100,230))
abline(lmFit)
```



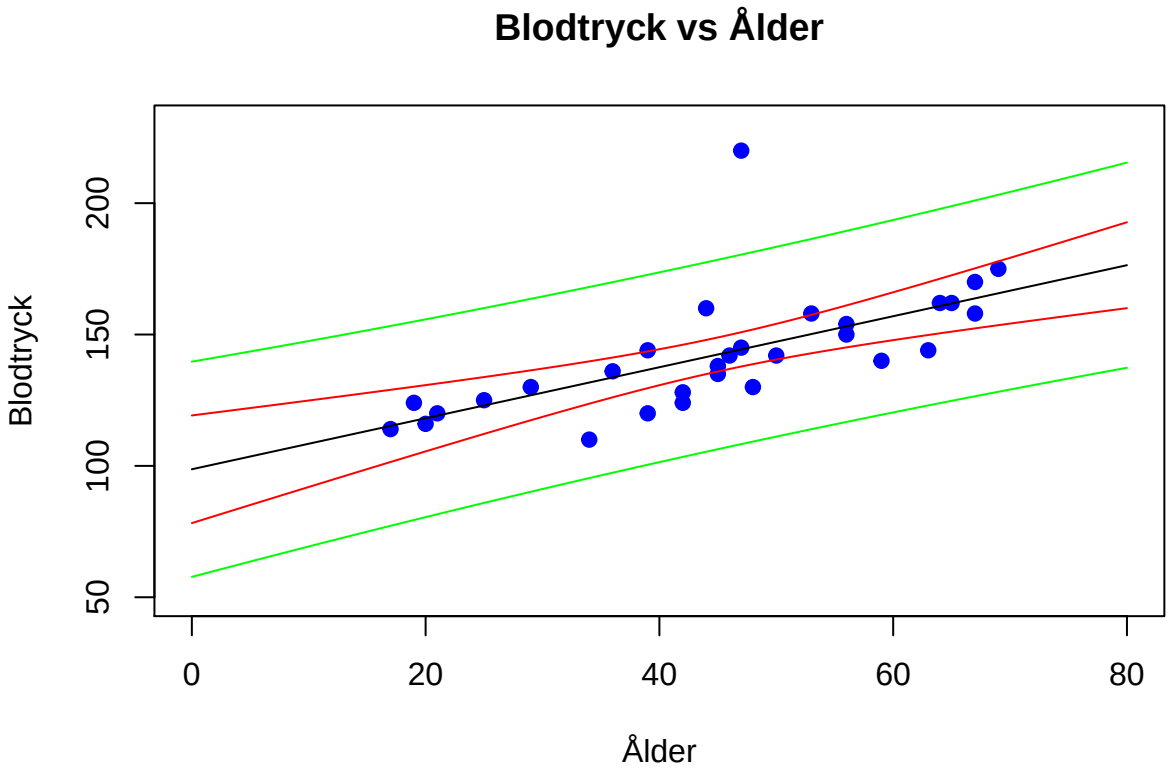
Uppgift 4 - Konfidensband för regressionslinjen och prediktionsband

Konfidensband och prediktionsband beräknas och plottas med koden:

```r
AgeGrid = 0:80 #  0:80 är värdena 0,1,2,...,80 för Age som vi predikterar för.
regLineFit = predict(lmFit, newdata = data.frame(Age = AgeGrid), interval = 'confidence')

plot(bloodpressure$Age, bloodpressure$SBP, xlab = "Ålder", ylab = "Blodtryck",
     main = "Blodtryck vs Ålder", col = "blue", pch = 19,
     xlim = c(0, 80), ylim = c(50,230))
lines(AgeGrid, regLineFit[,"fit"])
lines(AgeGrid, regLineFit[,"lwr"], col = "red")
lines(AgeGrid, regLineFit[,"upr"], col = "red")

predLineFit = predict(lmFit, newdata = data.frame(Age = AgeGrid), interval = 'prediction')
lines(AgeGrid, predLineFit[,"lwr"], col = "green")
lines(AgeGrid, predLineFit[,"upr"], col = "green")
```



Funktionen `predict` går prediktioner med modellanpassningsobjektet `lmFit` (som ju innehåller de skattade regressionskoefficienterna). Den funktionen måste veta vilka värden på den förklarande variabeln som vi vill ha prediktioner för. Det anges med argumentet `newdata` som måste vara en s k data frame (R-tabell) där kolumnerna i tabellen har precis samma namn som de förklarande variablerna i datamaterialet. Koden `data.frame(Age = AgeGrid)` gör just detta, tar våra värden på Age mellan 0 och 80 som vi vill ha prognoser för och gör om dessa värden till en data frame där kolumnen heter `Age`.

Notera också argumentet `interval` i `predict`-funktionen. Det bestämmer om vi ska beräkna konfidensintervall för regressionslinjen (`interval = 'confidence'`) eller prediktionsintervall (`interval = 'prediction'`).

Uppgift 5 - Beräkna konfidensintervall för parametrarna

För att beräkna ett 95%-igt konfidensintervall för parametrarna, β_0 och β_1 används `confint` funktionen på `lmFit` objektet:

```
confint(lmFit, level = 0.95)
```

```
##                 2.5 %     97.5 %
## (Intercept) 78.2296890 119.199747
## Age          0.5402629   1.401478
```