Statistik och dataanalys I

F9: Regression med dummyvariabler och modellval

Valentin Zulj

Statistiska institutionen

Stockholm
University

Vad har vi gjort hittills?

- Hittills har vi pratat om
  - **Enkel** och **multipel** linjär regression
  - Hur vi bestämmer koefficienter och gör prediktioner
  - Hur vi utvärderar modeller och kontrollerar modellantaganden
  - Hur vi transformerar för att uppnå linjäritet

Vad vi ska göra nu?

- I denna föreläsning ska vi prata om
  - Gruppvisa analyser med dummyvariabler i regressionsmodeller
  - Hur vi kan välja mellan olika regressionsmodeller
  - Hur vi kan använda träningsdata och testdata i punkten ovan
  - Korsvalidering

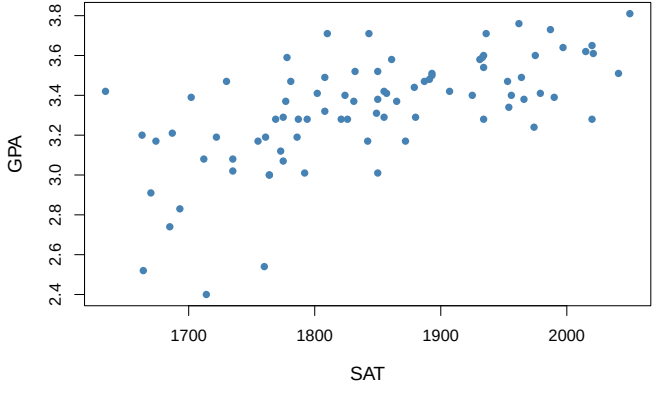
Regressionsanalys med dummyvariabler

Dummyvariabler (dummy/indicator variables)

- Vi har hittills varit noga med att bara använda **numeriska förklaringsvariabler** i våra regressionsanalyser
- Genom att vara lite fiffiga kan vi dock även använda **kategoriska förklaringsvariabler** för att göra gruppvisa regressionsanalyser
- Vi skulle t.ex. kunna dela upp analysen av bränsleförbrukning efter bilens växellåda (med kategorierna manuell och automatisk)

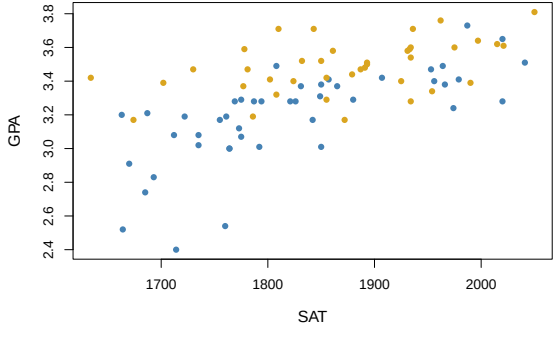
Dummyvariabler (dummy/indicator variables)

- Antag att vi vill ha en modell som gör prediktioner av amerikanska studenters snittbetyg (`GPA`), med `SAT` som förklaringsvariabel (SAT är ett "amerikanskt högskoleprov")
- De två variablerna illustreras i figuren nedan, och det verkar finnas ett positivt samband mellan dem



Motivering av regression med dummyvariabler

- Antag att vi också har variabeln `attendance`, som har värdet `yes` om studenten deltagit i minst 75% av föreläsningarna och `no` i annat fall
- Vi kan färgkoda diagrammet efter dessa två kategorier (nedan har studenter med hög närvaro fått **gula** prickar)



- Studenter med hög närvaro tycks generellt ha högre snittbetyg, givet sin SAT-poäng – detta är information som vi gärna vill använda i vår regressionsmodell!

Skapa en dummyvariabel

- Vi har konstaterat att variabeln `attendance` är kategorisk, med kategorierna `yes` och `no`
- Vi kan använda den i en regressionsmodell om vi kodar om den så att den blir en numerisk **dummyvariabel**
- En dummyvariabel kan anta två värden, 1 eller 0, beroende på vilken av de två kategorierna en observation tillhör
  - Vi ger dummyvariabeln värdet **1** om `attendance = "yes"`
  - Vi ger dummyvariabeln har värdet **0** om `attendance = "no"`
- **Vi skulle lika gärna kunna göra tvärtom**, det spelar ingen roll vilken kategori som blir 0 och vilken som blir 1 så länge vi håller koll på vilken som är vilken
- Koefficienten som är kopplad till dummyvariabeln visar hur kategorin som kodats till `1` förhåller sig till kategorin som kodats till `0`

Skapa en dummyvariabel

```
GPA$high_attendance_dummy <- ifelse(test = (GPA$high_attendance == "yes"), yes = 1, no = 0)

GPA[c(1, 16), ]
      GPA  SAT high_attendance high_attendance_dummy
  1  2.40 1714              no                     0
  16 3.17 1872             yes                     1
```

- Vi skapar en dummyvariabel av vår kategoriska variabel med funktionen `ifelse()`, som går igenom varje värde av variabeln `high_attendance`
  - `test = (GPA$high_attendance == "yes")` betyder att funktionen för varje observation testar påståendet att `high_attendance` är `"yes"`
  - Om påståendet är sant förvandlas motsvarande observation till `1`, och om påståendet är falskt blir observationen `0`
  - Resultatet sparas i en ny variabel med namnet `high_attendance_dummy`

Regression med en dummyvariabel

- När vi har skapat en dummy kan vi använda den som en vanlig numerisk variabel
- Vår modell för att prediktera snittbetyg blir

$$\widehat{\text{GPA}} = b_0 + b_1 \cdot \text{SAT} + b_2 \cdot \text{HighAttendance}$$

- Vi använder `lm()`-funktionen för att hitta koefficienternas värden

```
lm(GPA ~ SAT + high_attendance_dummy, data=GPA)$coefficients
            (Intercept)                     SAT high_attendance_dummy
            0.643850459            0.001399802            0.222644088
```

- Med siffrorna i utskriften får vi

$$\widehat{\text{GPA}} = 0.644 + 0.0014 \cdot \text{SAT} + 0.223 \cdot \text{HighAttendance}$$

- **Tolkning:** Modellen uppskattar att en student med hög närvaro har ett snittbetyg som är 0.223 högre än en student som *inte* har hög närvaro, givet samma SAT-poäng

Tolkning av regression med dummyvariabel

- När vi kodade *hög närvaro* som 1 och *ej hög närvaro* som 0 fick vi

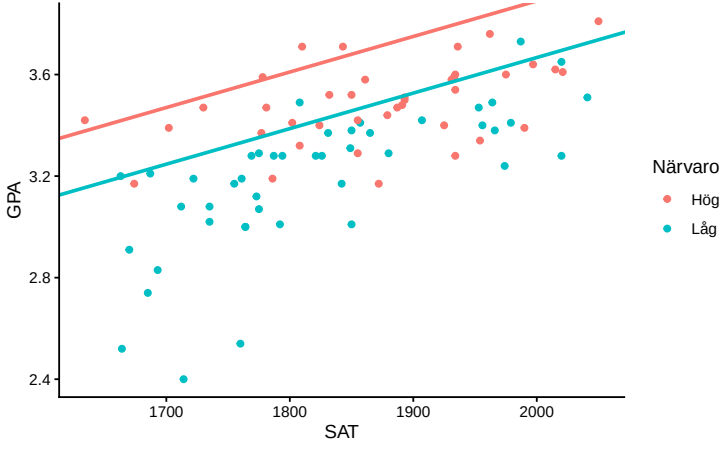
$$\widehat{\text{GPA}} = 0.644 + 0.0014 \cdot \text{SAT} + 0.223 \cdot \text{HighAttendance}$$

- Kategori `1` har **högre** snittpoäng än kategori `0`
- Om vi i stället hade kodat *ej hög närvaro* som 1 och *hög närvaro* som 0, hade vi fått motsatsen

$$\widehat{\text{GPA}} = 0.867 + 0.0014 \cdot \text{SAT} - 0.223 \cdot \text{NotHighAttendance}$$

- Denna modell uppskattar att en student **som inte har hög närvaro** har ett snittbetyg som är 0.223 **lägre** än en student som *har* hög närvaro, **givet en viss SAT-poäng**
- Båda modellerna kommer ge samma uppskattade snittbetyg för varje given student
- Detta beror på att interceptet förändras när vi kodar om dummyvariabeln
- Vi ser att interceptet i den senare modellen har justerats upp, detta på grund av att tecknet framför $b_2 \cdot \text{NotHighAttendance}$ har blivit negativt

- Dummy-variabeln gör så att analysen delas upp, och varje grupp får en enskild regressionslinje (se bilden nedan)
- **Extrauppgift:** Använd det du vet om dummyvariabeln för att ta fram intercept och lutning för de båda linjerna! Svar kommer på SDA II

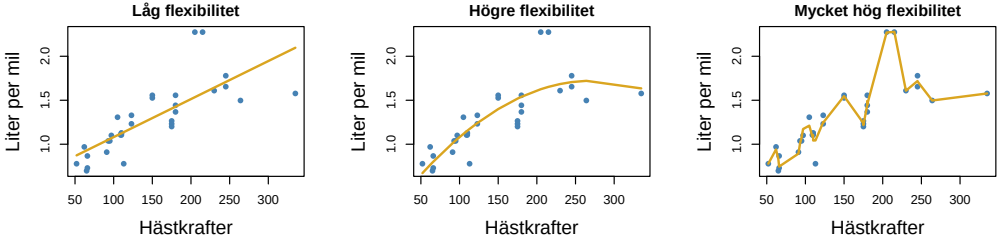


Val av regressionsmodell

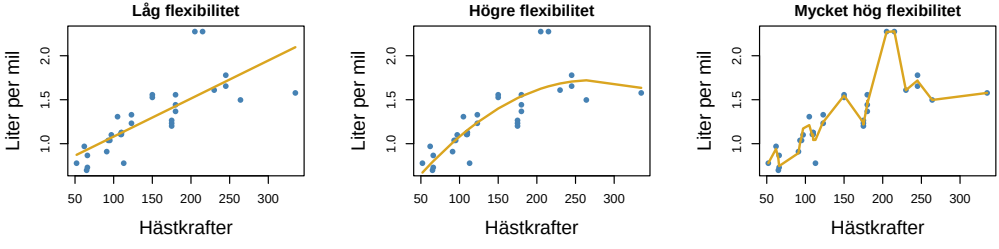
- Hittills har vi framför allt undersökt
  - Hur vi *tolkar* en regressionsmodell
  - Hur vi *transformerar* variabler för hitta samband som är någorlunda linjära
  - Hur vi använder måttet R^2 som säger oss något om hur väl modellen förklarar responsvariabelns värden
- Nu kommer vi in på frågan om *modellval*, som inbegriper frågor om t.ex.
  - Vilka variabler som är relevanta att använda i vår modell
  - Om någon/några variabler bör transformeras innan de tas med i modellen

Val av variabler

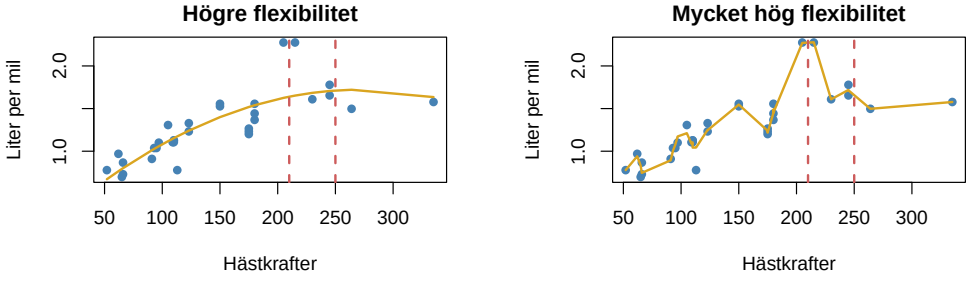
- En naturlig fråga att ställa är: Varför välja ut vissa variabler och välja bort andra – kan vi inte bara inkludera **alla** tillgängliga variabler?
- När vi tar med alla variabler gör vi R^2 så stort som möjligt, men vi vet att R^2 alltid ökar när vi lägger till fler variabler (och inte alltid är att lita på)
- I allmänhet gäller att modellen blir mer **flexibel** ju fler variabler vi lägger till, men detta kan vara både en styrka och en svaghet
- Låt oss titta närmare på vad vi menar med att en modell är flexibel, och varför vi vill undvika att en modell *för* flexibel!



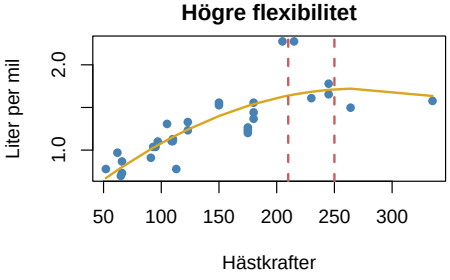
- Nedan ser vi tre regressionsmodeller som använder data om hästkrafter för att förklara bränsleförbrukning
- Vi bryr oss i det här sammanhanget inte om hur modellerna är konstruerade, utan vi konstaterar bara att de har olika stor flexibilitet
- Vi säger att en modell är mer **flexibel**, ju närmare den kan följa responsvariabelns värden

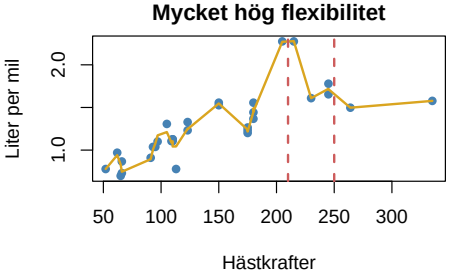


- Den **minst flexibla** modellen är en **rät linje**
- Den något mer flexibla modellen i mitten följer det övergripande mönstret
- Den **mest flexibla** modellen **anpassar sig tydligt efter de enskilda datapunkterna**

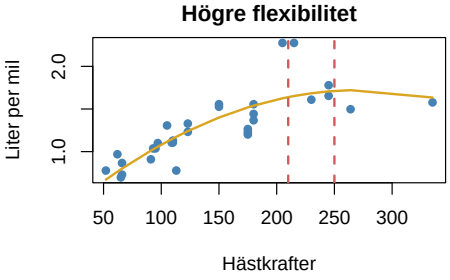


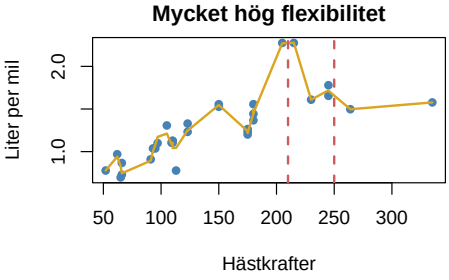
- Vi vill ofta använda regresision för att göra prediktioner, och vill så att våra modeller ska ge rimliga skattningar för nya observationer
- Om vi har två nya bilar, en med 210 hästkrafter och en med 250 hästkrafter, vilken av modellerna ger mest rimliga prediktioner?
- Det verkar orimligt att bilar med 210 hästkrafter **generellt** förbrukar mycket mer bränsle än bilar med 250 hästkrafter, som i den högra modellen



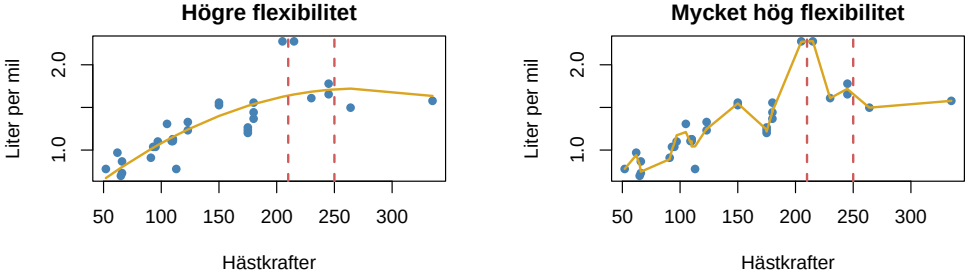


- Vi säger att en modell är **generaliserbar** om den hade sett ungefär likadan ut om vi anpassade den till andra (men liknande) data
- Vi kan tänka att vi helt plötsligt får data på 32 andra bilmodeller, och skattar de båda modellerna igen – vad skulle hända då?

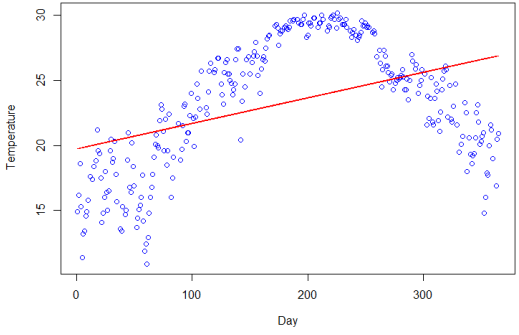




- Den vänstra modellen skulle förmodligen bli ungefär likadan, då den fångar upp ett övergripande mönster (mer generaliserbar)
- Den högra modellen skulle troligtvis se helt annorlunda ut, då den är väldigt exakt anpassad till de enskilda punkterna (mindre generaliserbar)



- En flexibel modell med dålig generaliserbarhet sägs vara **överanpassad** (overfitted)
- Den högra modellen är alltså en överanpassad modell, den är för anpassad till de enskilda punkterna, och ger en dålig bild av det övergripande mönstret



- En modell sägs vara **underanpassad** (underfitted) om den
  - **Inte** är flexibel, och
  - **Inte** fångar det övergripande mönstret i data
- Den röda linjen i bilden beskriver en underpassad modell, som inte fångar det uppenbara bågformade mönstret i datamaterialet

Trade-off mellan över- och underanpassning

- Vi har sett att både över- och underanpassade modeller är problematiska, och vi vill därför hitta en bra medelväg när vi väljer modell
- Vi vill alltså ta fram en modell som är
  - Tillräckligt flexibel för att **fånga det övergripande mönstret**
  - Tillräckligt restriktiv för att **inte fånga *precis alla* slumpvisa variationer**
- Kort sagt vill vi modellera "signalen" i data, och samtidigt ignorera "bruset"
- Kan vi se till så att en modell uppfyller båda dessa krav?
- Det är svårt att avgöra vad *tillräckligt* betyder i punkterna ovan – det är t.ex. inte självklart var gränsen mellan *tillräckligt flexibel* och *överanpassad* går
- **Men:** det finns ett gäng metoder som vi kan använda för att underlätta vårt val av modell

Modellval med hjälp av R^2_{adj}

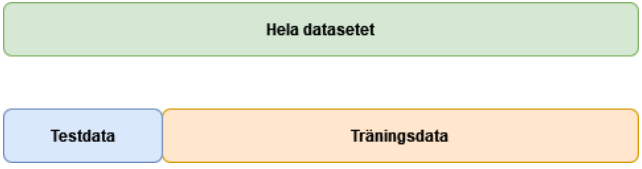
- Vi har pratat om att R^2 kan användas för att utvärdera modeller, men vi har även nämnt att R^2 alltid växer när vi lägger till variabler
- $R^2 = 1$ när modellen träffar *alla punkter*, Vilket betyder att modellen är överanpassad
- Om vi har maximering av R^2 som mål när vi väljer modell, kommer vi nästan garanterat att välja en överanpassad modell
- Vi har också talat om **justerat** R^2

$$R^2_{\text{adj}} = 1 - \frac{(1-R^2)(n-1)}{n-k-1} = 1 - \frac{SSE/(n-k-1)}{SST/(n-1)}$$

- Vi vet att R^2_{adj} blir mindre när vi lägger till irrelevanta variabler, vilket betyder att R^2_{adj} tar så hänsyn till generaliserbarhet i viss mån
- Det då är rimligt att försöka maximera R^2_{adj} när vi söker den bästa modellen
- Det finns mer dock sofistikerade sätt att mäta hur bra en modell är, som ofta ger bättre resultat i praktiken

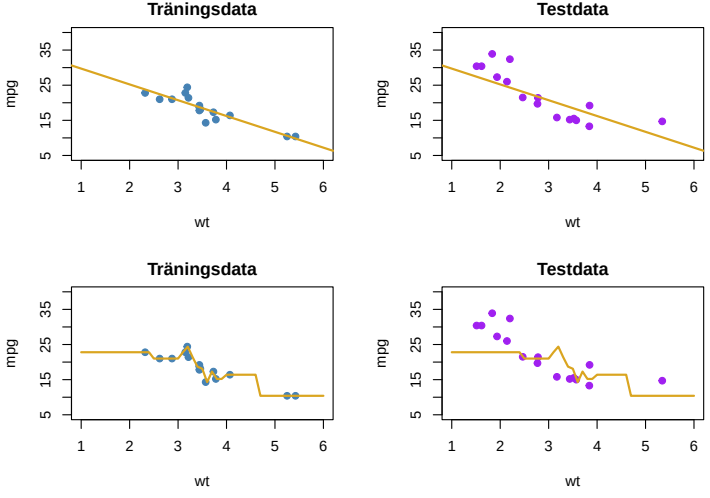
Träningsdata och testdata

- Vi har sagt att en generaliserbar modell är en modell som fungerar även för *nya data* (data som vi inte använde när vi anpassade modellen)
- Ett bra sätt att bedöma om modellen är generaliserbar är helt enkelt att **testa** hur bra prediktionerna blir på nya data
- För att göra detta kan vi dela våra data i två delar
  - Den ena delen kallas **träningsdata** (training data), och används för att anpassa modellen (hitta b_0, b_1, osv)
  - Den andra delen kallas **testdata**, och används för att utvärdera modellen



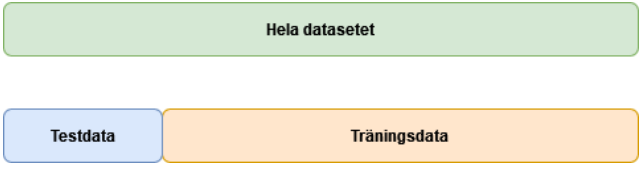
Träningsdata och testdata

- Nedan ser vi en model som inte är flexibel, och en modell som är det
- Den flexibla modellen passar träningsdata bra, men nya data väldigt dåligt



Träningsdata och testdata

- Vi kan t.ex. använda 70% av observationerna som träningsdata, och de övriga 30% av observationerna som testdata
- Om observationerna ligger i någon särskild ordning bör vi **sortera dem slumpmässigt** innan vi gör uppdelningen
- Vi vill att både tränings- och testdata ska vara **representativa** för hela datamaterialet (t.ex. inte bara ha hushåll från en stadsdel i testdata)



Träningsdata och testdata

- För att utvärdera en modell på testdata följer vi stegen nedan
  1. Dela upp observationerna i träningsdata och testdata
  2. Anpassa regressionsmodellen med hjälp av träningsdata
  3. Utvärdera modellen med hjälp av testdata
- Till steg 3 behöver vi ett mått som utvärderar hur väl en modell passar våra testdata
- Det finns många olika mått att välja, men det kanske mest vanliga är **RMSE (Root Mean Squared Error)**
- I allmänhet gäller att ju **mindre RMSE**, desto **bättre modell**

- För att räkna ut RMSE använder vi formeln nedan

$$\text{RMSE}_{\text{test}} = \sqrt{\frac{\sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2}{n_{\text{test}}}},$$

- **Notera:** Summan i formeln går över **alla observationer i testdata**, och n_{test} betecknar antalet observationer i just testdata
- Vi ser att uttrycket innanför kvadratroten ser ut som medelvärdet av modellens prediktionsfel på våra testobservationer
- Om RMSE är litet betyder det att modellen är bra på att göra prediktioner på nya data!

- Vi kommer ihåg att **SSE (Sum of Squared Errors)** betecknar summan av kvadrerade prediktionsfel,

$$\text{SSE}_{\text{test}} = \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2$$

- Vi kan alltså skriva RMSE som

$$\text{RMSE}_{\text{test}} = \sqrt{\frac{SSE_{\text{test}}}{n_{\text{test}}}}$$

- Kvoten $SSE_{\text{test}}/n_{\text{test}}$ kallas för **MSE (Mean Squared Error)**
- Vi kan därför även skriva RMSE som

$$\text{RMSE}_{\text{test}} = \sqrt{\text{MSE}_{\text{test}}}$$

Modellval - träningsdata och testdata

- Det är ofta lättast (rent räknemässigt) att beräkna $\text{RMSE}_{\text{test}}$ i flera steg
  1. Beräkna $\text{SSE}_{\text{test}} = \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2$
  2. Beräkna $\text{MSE}_{\text{test}} = \dfrac{\text{SSE}_{\text{test}}}{n_{\text{test}}}$
  3. Beräkna $\text{RMSE}_{\text{test}} = \sqrt{\text{MSE}_{\text{test}}}$
- Vi använder ofta SSE_{test} till flera olika saker, så det är bra att beräkna den separat för att slippa repetera

Träningsdata och testdata

- Vi tittar nu på ett exempel, där vi
  1. Delar upp ett datamaterial i träningsdata och testdata
  2. Anpassar modellen med hjälp av våra träningsdata
  3. Utvärderar modellen genom att räkna ut RMSE för våra testdata

Exempel

- Vi använder ett datamaterial med variablerna y, x_1, x_2 och x_3 – där y är responsvariabel och övriga variabler är förklaringsvariabler
- Vilka förklaringsvariabler vi inkluderar är upp till oss själva, och målet är att hitta en modell som ger bra prediktioner i vår testdata
- Vi har totalt 100 observationer, och de första ser ut såhär:

```
          y       x1       x2         x3
1 31.41966 16.88882 5.599934 -0.7104066
2 32.39954 25.40119 3.996941  0.2568837
3 26.59989 18.95261 4.931678 -0.2466919
4 39.44798 27.01130 7.726843 -0.3475426
```

- Som verksam statistiker är det inte alls ovanligt att på detta sätt bli tilldelad ett okänt datamaterial, och få som uppgift att identifiera en bra prediktionsmodell

Steg 1: Dela upp data

- Vi börjar med att dela upp våra data i en träningsdel och ett testdel
- Vi börjar med att tilldela observationerna en slumpmässig ordning
- Därefter låter vi de första 70 observationerna utgöra träningsdata
- Till sist får de återstående 30 observationerna bli testdata

```
library(dplyr)                                    # Paket behövs för slice

data_randomorder <- data %>% slice_sample(prop=1) # Sorterar slumpmässigt

datatrain <- data_randomorder %>% slice(1:70)     # Första 70 obs

datatest <- data_randomorder %>% slice(71:100)    # Sista 30 obs
```

Steg 2: Träna modellen $y = b_0 + b_1 x_1 + b_2 x_2$

```r
model1 <- lm(y ~ x1 + x2, data=datatrain) # Träna/skatta modellen

summary(model1)                           # Se sammanfattande resultat
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.01430    1.21551   3.303  0.00154 **
x1           0.76317    0.04482  17.028  < 2e-16 ***
x2           1.82995    0.12893  14.194  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.722 on 67 degrees of freedom
Multiple R-squared:  0.871, Adjusted R-squared:  0.8671
F-statistic: 226.2 on 2 and 67 DF,  p-value: < 2.2e-16
```

Steg 3: Utvärdera på testdata

- För att få fram $\text{RMSE}_{\text{test}}$ beräknar vi SSE_{test}, MSE_{test} och $\text{RMSE}_{\text{test}}$ i tur och ordning

```
rsquared1 <- summary(model1)$r.squared # För jämförelse

y_hatt_test <- predict(model1, newdata=datatest) # Prediktioner av testdata
y_test <- datatest$y                             # Sanna värden

SSE <- sum((y_test - y_hatt_test)^2)             # SSE

n_test <- 30                                     # Obs i testdata
MSE <- SSE / n_test                              # MSE

RMSE <- sqrt(MSE)                                # RMSE

print( round( c(rsquared1, SSE, MSE, RMSE), digits = 3 ) )
  [1]   0.871 146.053   4.868   2.206
```

- Vi har $\text{SSE}_{\text{test}} \approx 156$, $\text{MSE}_{\text{test}} \approx 4.87$ och $\text{RMSE}_{\text{test}} \approx 2.2$

Upprepa för modellen $y = b_0 + b_1x_1 + b_2x_2 + b_3x_3$

Stockholm University

- För jämförelsens skull skattar vi en till modell, med lite fler variabler

```
model2 <- lm(y ~ x1 + x2 + x3, data=datatrain)     # Skattar modell
rsquared2 <- summary(model2)$r.squared             # För jämförelse

y_hatt_test <- predict(model2, newdata=datatest)   # Prediktioner av testdata
y_test <- datatest$y                               # Sanna värden

SSE2 <- sum((y_test - y_hatt_test)^2)              # SSE

n_test <- 30                                       # Obs i testdata
MSE2 <- SSE2 / n_test                              # MSE

RMSE2 <- sqrt(MSE2)                                # RMSE

print( round( c(rsquared2, SSE2, MSE2, RMSE2), digits = 3 ) )
  [1]   0.871 146.523   4.884   2.210
```

Jämförelse av modellerna

- Vi jämför modellerna i tabellen nedan, där vi har
  - Modell 1: $y = b_0 + b_1 x_1 + b_2 x_2$
  - Modell 2: $y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$

Modell	R.squared	SSE	MSE	RMSE
Modell 1	0.8709888	146.0526	4.868421	2.206450
Modell 2	0.8710176	146.5229	4.884096	2.209999

- Skillnaderna mellan modellerna är små, men det finns saker att notera
- Baserat på R-squared kunde man tro att modell 2 är bättre, då den förklarar en aning mer av variationen i vår träningsdata
- RMSE är samtidigt något **mindre** för modell 1, vilket betyder att den modellen är lite **bättre** på att göra prediktioner på vår testdata

Träningsdata och testdata

- Att dela upp data i tränings- respektive testdelar ger oss en bättre bild av hur användbar en modell är
- Vi vet dock inte säkert om bilden blir helt rättvis, särskilt om det är ett datamaterial med få observationer
- Som exempel på detta använder vi våra bildata, och modellen $\widehat{\text{litermil}} = b_0 + b_1 \cdot \text{viktton}$
  - Först använder vi **de 12 sista** observationerna som testdata, och de 20 första som träningsdata
  - Sedan gör vi samma sak, men använder i stället **de 12 första** som testdata, och de 20 sista som träningsdata

Träningsdata och testdata

- Vi kodar upp exemplet i R, och börjar med de 12 sista observationerna som testdata

```
# De 12 sista observationerna används i vårt testset
carstrain <- mtcars %>% slice(1:20) # Träningsdata
carstest <- mtcars %>% slice(21:32) # Testdata

lmod1 <- lm(litermil ~ viktton, data=carstrain) # Skattar modell
y_hatt <- predict(lmod1, newdata=carstest)      # Prediktioner av testdata

RMSE <- sqrt(mean((carstest$litermil - y_hatt)^2)) # RMSE i ett steg
print(RMSE)
  [1] 0.1984769
```

- Vi gör nu om analysen, fast med de 12 **första** observationerna som testdata

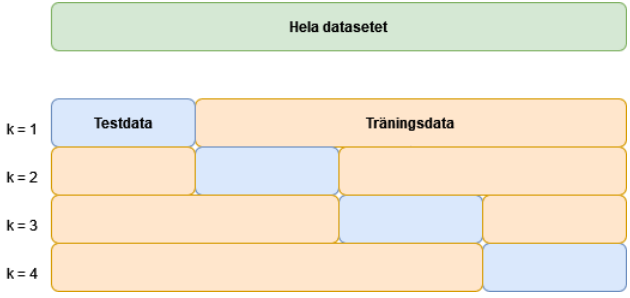
```
# De 12 första observationerna används i vårt testset
carstrain <- mtcars %>% slice(13:32) # Träningsdata
carstest <- mtcars %>% slice(1:12)   # Testdata

lmod1 <- lm(litermil ~ viktton, data=carstrain) # Skattar modell
y_hatt <- predict(lmod1, newdata=carstest)      # Prediktioner av testdata

RMSE <- sqrt(mean((carstest$litermil - y_hatt)^2)) # RMSE i ett steg
print(RMSE)
  [1] 0.1790871
```

Träningsdata och testdata

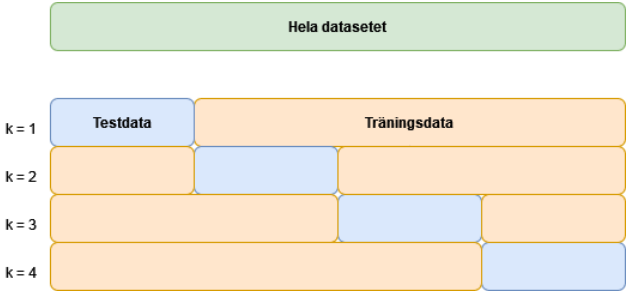
- Med de *sista* 12 observationerna som testdata fick vi RMSE = 0.198
- Med de *första* 12 observationerna som testdata fick vi RMSE = 0.179
- Vi ser så att RMSE skiljer sig relativt mycket, beroende på hur vi delar upp våra data
- Skillnaderna kan vara både större och mindre, men det är svårt att veta när
- Ett sätt att komma runt detta problem är att använda något som kallas för **korsvalidering** (cross validation)

Korsvalidering

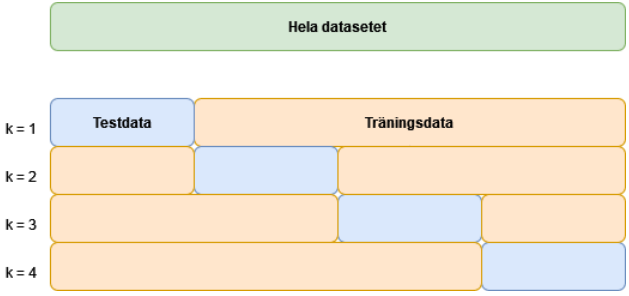
- Korsvalidering är en metod som låter oss använda **alla** observationer **både** som träningsdata **och** som testdata
- När vi använder korsvalidering delar vi upp våra data i tränings- respektive testdelar flera gånger
- Efter varje uppdelning får vi en ny uppsättning punkter som testdata
- Varje observation hamnar i testdata **en** gång, och i träningsdata i övriga uppdelningar



- Varje uppdelning kallas för en **fold**
- Korsvalisering kallas ibland K-faldig korsvalidering (*K-fold cross validation*), där K är antalet folds
- Fyrfaldig ($K = 4$) korsvalidering betyder t.ex. att vi delar in vår data i tränings- och testdata på 4 olika sätt



- För var och en av de 4 uppdelningarna räknar vi ut SSE på testdata
- Vi använder sedan summan av de 4 SSE-värdena för att beräkna RMSE



- Antag att vi har ett dataset med 100 observationer och $K = 4$
  - För $k = 1$ gäller då att observation 1 till 25 utgör testdata
  - För $k = 2$ gäller att observation 26 till 50 utgör testdata,
  - Och så vidare

- Vårt mål är att räkna ut RMSE med hjälp av våra 4 folds
- Vi börjar med att räkna ut SSE för varje enskild fold, och summerar sedan våra fyra värden – superscript (1), (2), etc visar vilken fold SSE är beräknad med

$$\text{SSE}_{\text{cv}} = \text{SSE}_{\text{test}}^{(1)} + \text{SSE}_{\text{test}}^{(2)} + \text{SSE}_{\text{test}}^{(3)} + \text{SSE}_{\text{test}}^{(4)}$$

- När vi har tagit fram SSE_{cv} kan vi beräkna

$$\text{MSE}_{\text{cv}} = \frac{\text{SSE}_{\text{cv}}}{n} \quad \text{och} \quad \text{RMSE}_{\text{cv}} = \sqrt{\text{MSE}_{\text{cv}}}$$

- Notera: cv i subscript står för cross validation

- **Procedur för enkel uppdelning i tränings- och testdata:**
  1. Räkna ut SSE_{test}
  2. Räkna ut $\text{MSE}_{\text{test}} = \text{SSE}_{\text{test}}/n_{\text{test}}$
  3. Räkna ut
$$\text{RMSE}_{\text{test}} = \sqrt{\text{MSE}_{\text{test}}}$$
- **Procedur för korsvalidering:**
  1. Räkna ut $\text{SSE}_{\text{test}}^{(1)}, \text{SSE}_{\text{test}}^{(2)}, ..., \text{SSE}_{\text{test}}^{(K)}$, och därefter
$$\text{SSE}_{\text{CV}} = \text{SSE}_{\text{test}}^{(1)} + \text{SSE}_{\text{test}}^{(2)} + ... + \text{SSE}_{\text{test}}^{(K)}$$
  2. Räkna ut $\text{MSE}_{\text{CV}} = \text{SSE}_{\text{CV}}/n$
  3. Räkna ut
$$\text{RMSE}_{\text{CV}} = \sqrt{\text{MSE}_{\text{CV}}}$$

Exempel på korsvalidering med $K = 2$

- Vi arbetar nu igenom ett exempel på korsvalidering, med hjälp av R
- Vi använder våra bildata med 32 observationer, och utvärderar den enkla modellen med vikt som förklaringsvariabel
- Vi sätter $K = 2$, det vill säga vi använder 2 folds
- När vi räknar ut $\text{SSE}_{\text{test}}^{(1)}$ låter vi de 16 **första** observationerna vara testdata, och använder övriga observationer träningsdata
- När vi räknar ut $\text{SSE}_{\text{test}}^{(2)}$ låter vi de 16 **sista** observationerna vara testdata, och använder de övriga observationerna som träningsdata

Exempel på korsvalidering med $K = 2$

- Med koden nedan kan vi beräkna $\text{SSE}_{\text{test}}^{(1)}$ och $\text{SSE}_{\text{test}}^{(2)}$

```
# Räkna ut SSE(1)
carstest <- mtcars %>% slice(1:16)
carstrain <- mtcars %>% slice(17:32)
lmod1 <- lm(litermil ~ viktton, data=carstrain)
y_hatt1 <- predict(lmod1, newdata=carstest)
y1 <- carstest$litermil
SSE1 <- sum((y1 - y_hatt1)^2)

# Räkna ut SSE(2)
carstest <- mtcars %>% slice(17:32)
carstrain <- mtcars %>% slice(1:16)
lmod2 <- lm(litermil ~ viktton, data=carstrain)
y_hatt2 <- predict(lmod2, newdata=carstest)
y2 <- carstest$litermil
SSE2 <- sum((y2 - y_hatt2)^2)
```

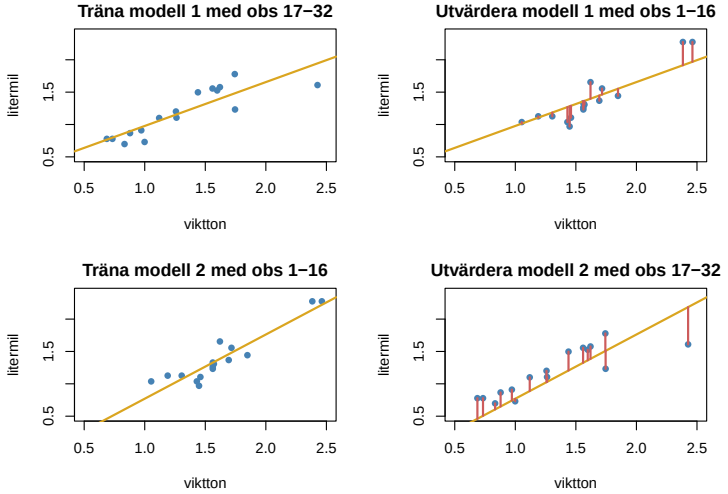
Exempel på korsvalidering med $K = 2$

Vi skriver ut värdena på $\text{SSE}_{\text{test}}^{(1)}$ och $\text{SSE}_{\text{test}}^{(2)}$

```
SSE1
  [1] 0.5330895

SSE2
  [1] 1.027022
```

Exempel på korsvalidering med $K = 2$

- Bilderna visar våra data och de två regressionsmodellerna (en för varje fold)
- Om vi kvadrerar de röda linjerna i bilderna till höger och summerar kvadraterna får vi $\text{SSE}_{\text{test}}^{(1)}$ (övre) och $\text{SSE}_{\text{test}}^{(2)}$ (undre)



Exempel på korsvalidering med $K = 2$

- Vi har räknat ut
  - $\text{SSE}_{\text{test}}^{(1)} = 0.5331$
  - $\text{SSE}_{\text{test}}^{(2)} = 1.0270$
- Nu kan vi räkna ut

$$\text{SSE}_{\text{CV}} = \text{SSE}_{\text{test}}^{(1)} + \text{SSE}_{\text{test}}^{(2)} = 0.5331 + 1.0270 = 1.5601$$

$$\text{MSE}_{\text{CV}} = \frac{\text{SSE}_{\text{CV}}}{n} = \frac{1.5601}{32} = 0.04875312$$

$$\text{RMSE}_{\text{CV}} = \sqrt{\text{MSE}_{\text{CV}}} = \sqrt{0.04875312} = 0.2208011$$

Exempel på korsvalidering med $K = 2$

- Normalt gör vi våra uträkningar i programkoden

```
n <- 32
SSE_cv <- SSE1 + SSE2
MSE_cv <- SSE_cv / n
RMSE_cv <- sqrt(MSE_cv)

RMSE_cv
  [1] 0.2208019
```

Exempel på korsvalidering med $K = 2$

- Nu har vi räknat ut RMSE_{CV} för **en** regressionsmodell
- För att jämföra **flera** modeller måste vi räkna ut RMSE_{CV} för **varje** modell
- När vi har gjort detta väljer vi den modell som har **minst** RMSE_{CV}
- När vi, till slut, har valt en modell att använda, så anpassar vi den till **alla data**
- Eftersom den valda modellen är avsedd att användas, vill vi ta vara på så mycket information som möjligt när vi anpassar den

Tack för idag!!