Föreläsning 9

karl.sigfrid@stat.su.se

Vad har vi gjort hittills, och vad vi ska göra nu

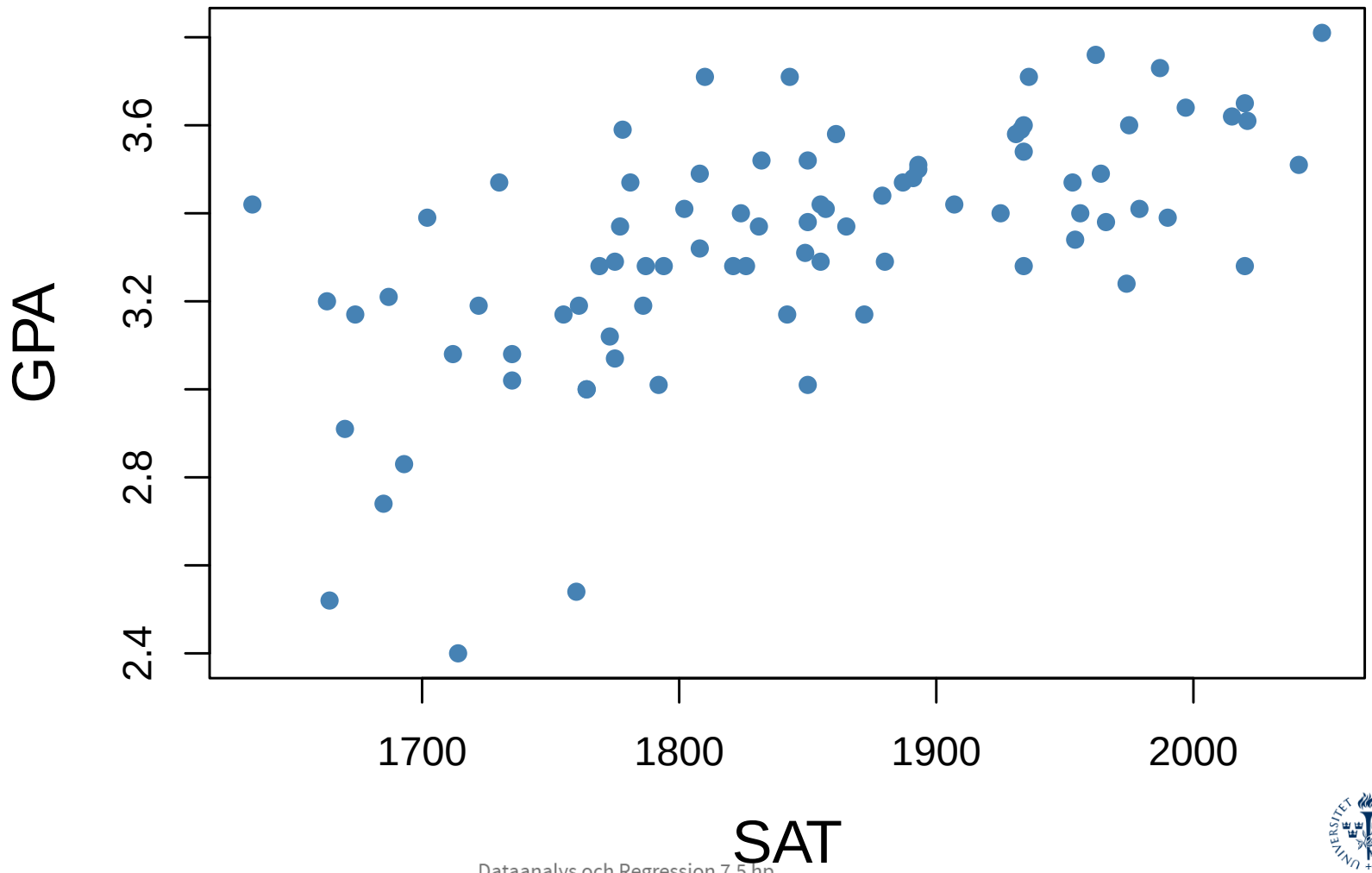
- Vi har gått igenom
  - enkel linjär regression
  - multipel linjär regression
- Nu ska vi undersöka
  - dummyvariabler i regressionsmodeller
  - modellval
  - träningsdata och testdata
  - korsvalidering

Dummyvariabler (dummy variables / indicator variables)

- Vi har hittills använt **numeriska förklaringsvariabler** i våra regressionsmodeller.
- Det är också möjligt att använda **kategoriska förklaringsvariabler**.
- Anta att vi vill prediktera amerikanska studenters snittbetyg (GPA) med *SAT score* som förklaringsvariabel. (SAT är en amerikansk variant av högskoleprovet).

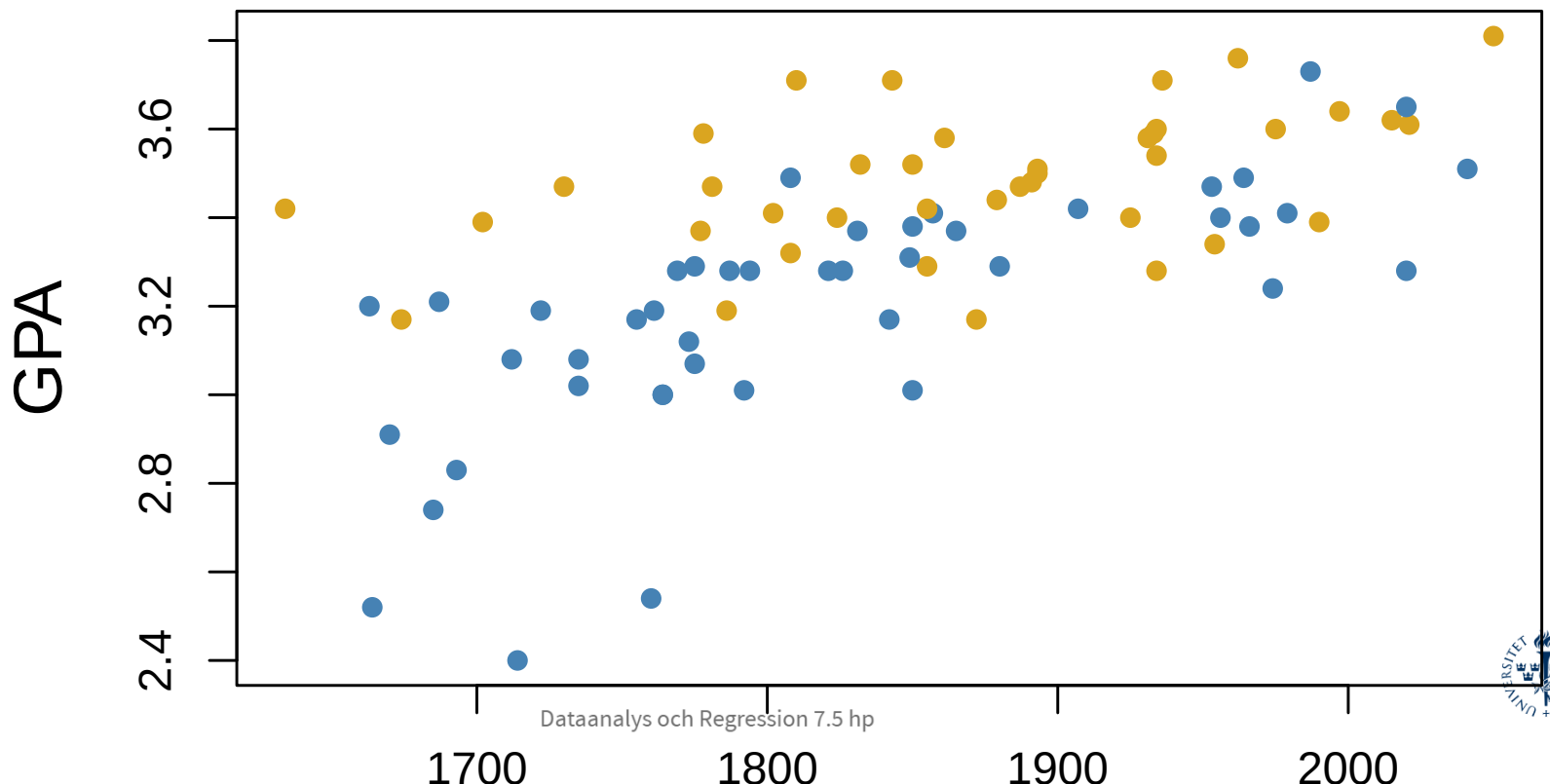
Dummyvariabler (dummy variables / indicator variables)

- Vi ritar ett spridningsdiagram mellan variablerna GPA och SAT.
- Ett spridningsdiagram mellan dessa numeriska variabler tyder på ett positivt samband.



Dummyvariabler - motivation

- Anta att vi också har en variabel **attendance**, som har värdet *yes* om studenten deltagit på minst 75% av föreläsningarna och *no* i annat fall.
- I denna figur är punkter som representerar studenter med hög närvaro **gula**.
- Studenter med hög tycks generellt ha högre snittbetyg, givet sin SAT-poäng. Det vore bra om vi kan använda det i vår regressionsmodell!



Dummyvariabler - skapa en dummyvariabel

- Vi har konstaterat att variabeln *attendance* är kategorisk.
- Vi kan använda den i en regressionsmodell om vi kodar om den så att den blir en numerisk dummyvariabel.
- En dummyvariabel har antingen värdet 1 eller värdet 0, beroende på vilken av de två kategorierna
  - Vi ger dummyvariabeln värdet **1** om attendance = "yes".
  - Vi ger dummyvariabeln har värdet **0** om attendance = "no".
- **Vi skulle lika gärna kunna göra tvärtom.** Vilken kategori som är 0 och vilken som är 1 påverkar inte våra resultat.

Dummyvariabler - skapa en dummyvariabel

- Vi skapar en dummyvariabel av vår kategoriska variabel.
- Funktionen *ifelse* går igenom varje värde av variabeln *attendance*.
  - *GPA$attendance* == *"yes"* betyder att funktionen för varje observation ska kontrollera om värdet på variabeln attendance är "yes".
  - Om det är sant att värdet är *"yes"* ska funktionen returnera *1*.
  - Om det **inte** är sant att värdet är *"yes"* ska funktionen returnera *0*.
  - Resultatet sparas i en ny variabel med namnet *high_attendance*.

```
GPA$high_attendance <- ifelse(GPA$attendance == "yes", 1, 0)
tail(GPA, 3)
```

```
    GPA  SAT attendance high_attendance
82 3.73 1987         no               0
83 3.76 1962        yes               1
84 3.81 2050        yes               1
```

Dummyvariabler - använd en dummyvariabel

- Nu när vi har vår dummyvariabel använder vi den på samma sätt 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 parametrarnas värden.

```
lm(GPA ~ SAT + high_attendance, data=GPA)$coefficients
```

```
    (Intercept)             SAT high_attendance
    0.643850459     0.001399802     0.222644088
```

Dummyvariabler - tolka en dummyvariabel

Nu när vi har våra parametervärden kan vi skriva modellen som

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

Tolkning

Modellen estimerar att en student med hög närvaro har ett snittbetyg som är 0.223 högre än en student som *inte* har hög närvaro, **givet en viss SAT-poäng**.

Exempel

Anta att vi har två studenter med samma SAT-poäng. En av dem har hög närvaro och en har det inte. Modellen estimerar att studenten med hög närvaro har ett snittbetyg som är 0.223 högre.

Dummyvariabler - tolka en dummyvariabel

- Notera att skattningen av den parameter som kopplad till dummyvariabeln säger oss hur den kategori som kodats till **1** skiljer sig jämfört med den kategori som kodats till **0**.
- När vi kodade *hög närvaro* som 1 och *ej hög närvaro* som 0 fick vi den här modellen:

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

- Om vi i stället hade kodat *ej hög närvaro* som 1 och *hög närvaro* som 0, då hade vi fått modellen

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

Dummyvariabler - tolka en dummyvariabel

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

Tolkning

- Den senare modellen estimerar 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**.
- Det predikterade medelbetyget för en enskild student blir detsamma oavsett hur vi kodar dummyvariabeln. Det beror på att **interceptet** förändras när vi kodar om dummyvariabeln. I den andra modellen blir termen $b_2 \cdot \text{NotHighAttendance}$ *0.223* lägre än i den första modellen för varje student, men interceptet är samtidigt 0.223 *högre*.

Modellval

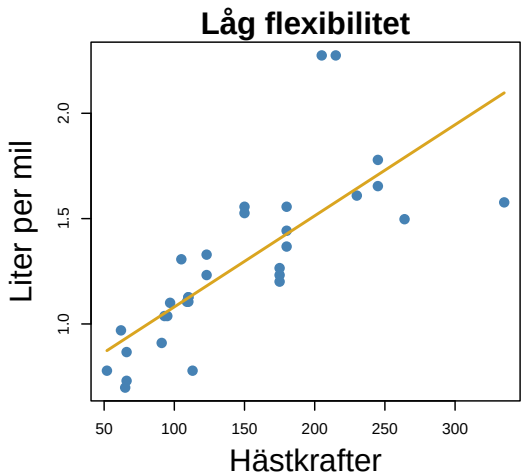
- Hittills har vi framför allt undersökt hur vi **tolkar** en regressionsmodell.
- Vi har **transformerat** variabler för hitta samband som är någorlunda linjära.
- Vi har också gått igenom måttet **R-squared** som säger oss något om hur väl modellen förklarar responsvariabelns värden.
- Nu kommer vi in på frågan om **modellval**, det vill säga frågan om vilka variabler vi ska använda i en modell och hur vissa av variablerna eventuellt ska transformeras.

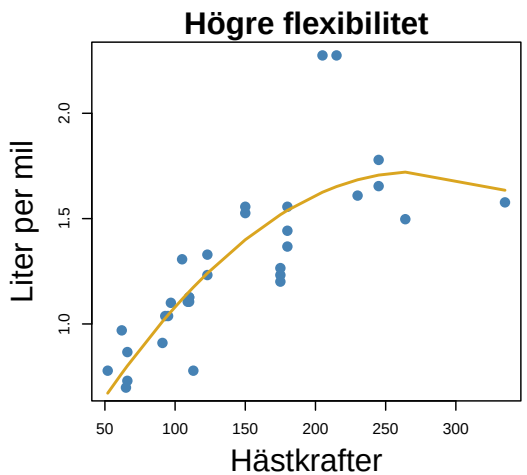
Modellval

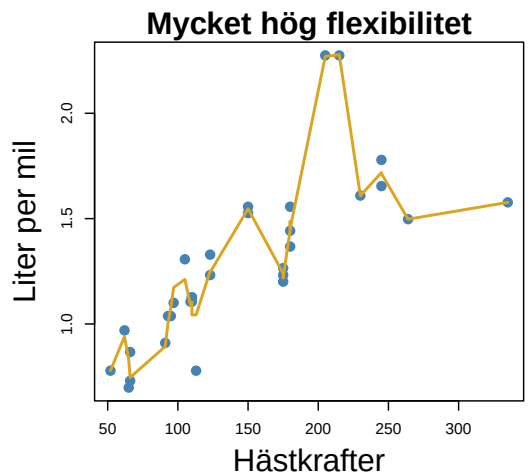
- En naturlig fråga att ställa är: Varför välja ut vissa variabler och välja bort andra? Kan vi inte bara inkludera **alla** tillgängliga variabler?
- Att använda alla variabler ger måttet **R-squared** dess största möjliga värde. Vi har tidigare lärt oss att R-squared alltid ökar när vi lägger till fler variabler.
- Problemet är att ju fler variabler vi använder desto mer **flexibel** blir modellen.
- Låt oss titta närmare på vad vi menar med att en modell är flexibel, och varför vi vill undvika att en modell *för* flexibel!

Modellval

- Här ser vi **3 regressionsmodeller** som estimerar en bils bränsleförbrukning med förklaringsvariabeln hästkrafter.
- Vi bryr oss i det här sammanhanget inte om hur modellerna är konstruerade, utan vi konstaterar bara att de har olika stor flexibilitet.
- En mer flexibel en modell följer den **responsvariabelns värden** närmare.

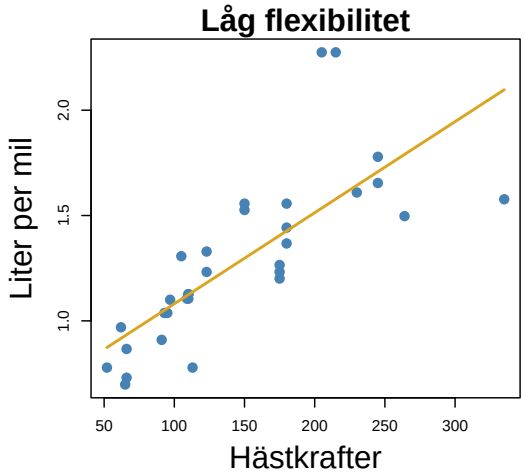


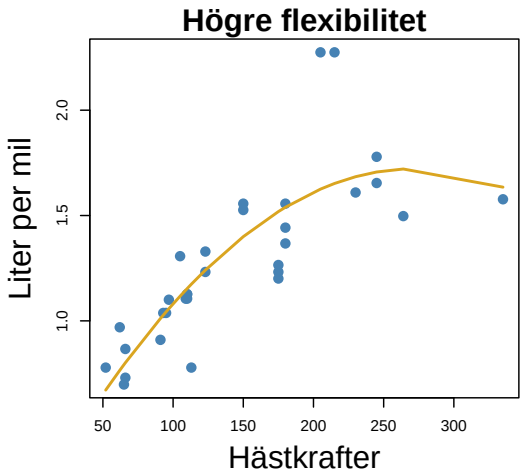


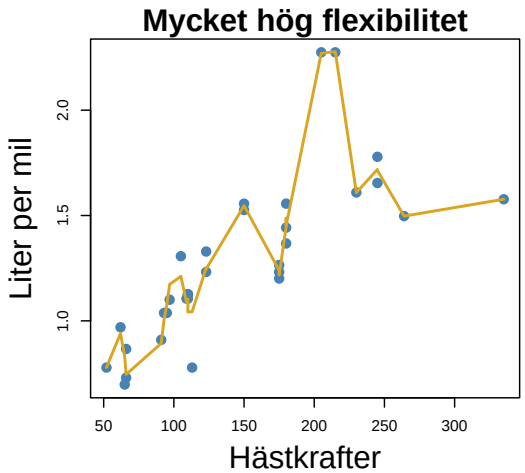


Modellval - flexibilitet

- 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**.

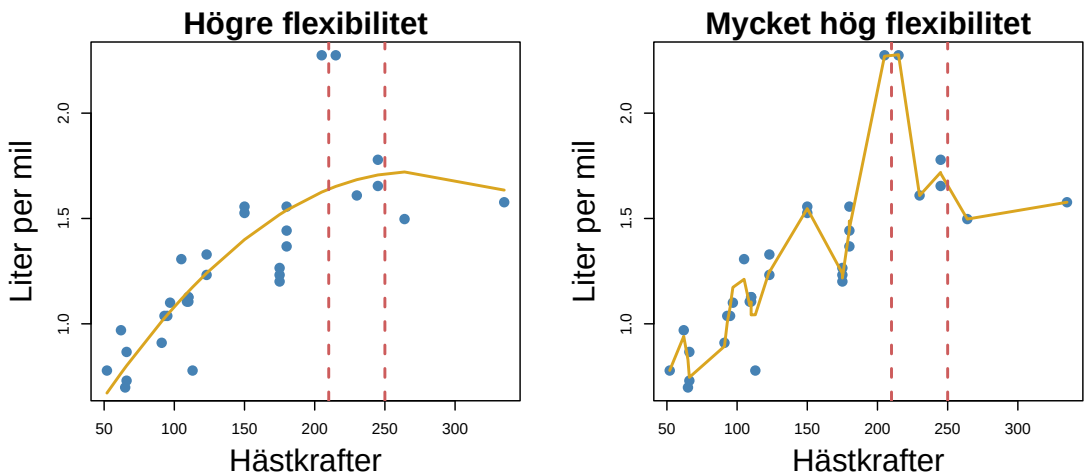






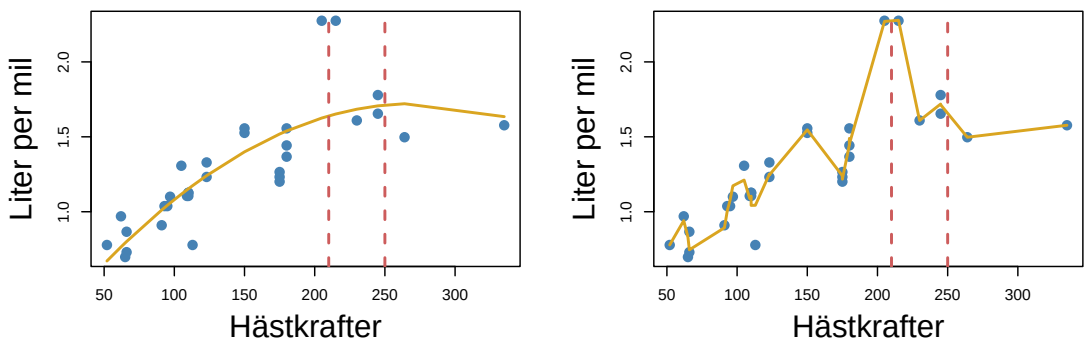
Modellval - flexibilitet

- Kom ihåg att en bra modell ska ge **rimliga etimat** för **nya observationer**.
- Om vi har en två nya bilar, en med 210 hästkrafter och en med 250 hästkrafter, vilken av modellerna ger de rimligaste prediktionerna?
- Det verkar orimligt att bilar med 210 hästkrafter **generellt** skulle ha betydligt större bränsleförbrukning än bilar med 250 hästkrafter.



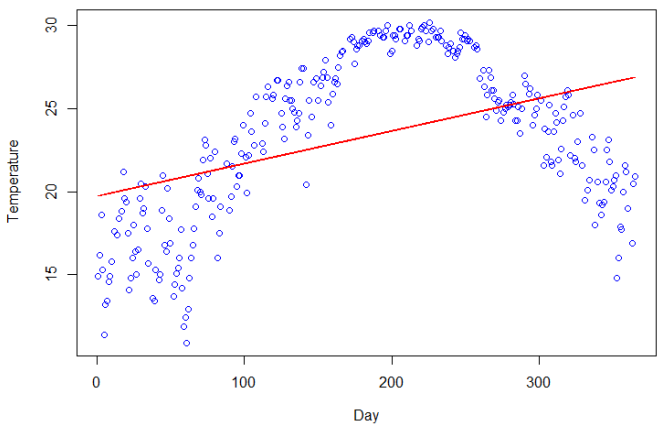
Modellval - generaliserbarhet

- Vi säger att den mindre flexibla modellen till vänster är mer **generaliserbar**. En modell som är generaliserbar ser ungefär likadan ut även om den anpassades med annan data. Modellen till vänster skulle exempelvis kunna se ungefär likadan ut om vår data innehöll 32 andra bilmodeller.
- Modellen till höger skulle förmodligen se helt annorlunda ut om vi hade andra bilmodeller i vår data.
- Vi säger att en flexibel modell med dålig generaliserbarhet är **överanpassad (overfitted)**.



Modellval - generaliserbarhet

- En oflexibel modell som inte fångar mönstret i vår data är **underanpassad (underfit)**.
- Den här grafen visar en underanpassad modell. Den räta linjen fångar inte det övergripande bågmönstret.
- Målet är att hitta en bra medelväg där modellen **fångar det övergripande mönstret** men **utan att fånga alla slumpvisa variationer**.



Modellval - generaliserbarhet

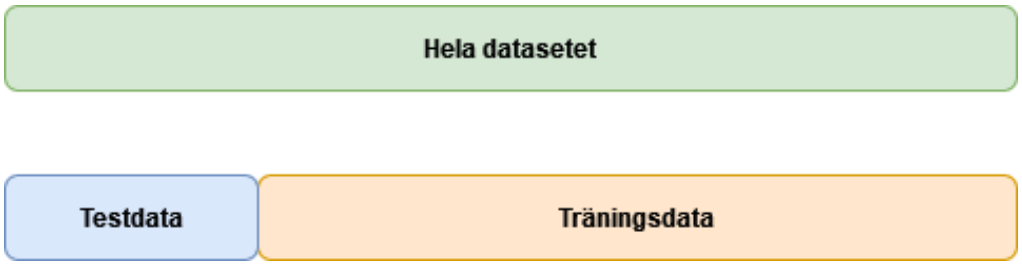
- Vi har redan konstaterat att måttet R^2 **inte tar hänsyn till generaliserbarhet**. Tvärtom blir R^2 högt för en överanpassad modell.
- Vi har tidigare talat om **adjusted R-squared**, som tar hänsyn till generaliserbarhet.

$$R^2_{\text{adjusted}} = 1 - \frac{(1 - R^2)(n - 1)}{n - k - 1}$$

- I formeln står k för antalet variabler, och vi ser att adjusted R-squared blir mindre när antalet variabler ökar.
- Att försöka maximera adjusted R-squared är en enkel strategi i sökandet efter den bästa modellen, men det finns mer sofistikerade sätt att mäta hur bra en modell är.

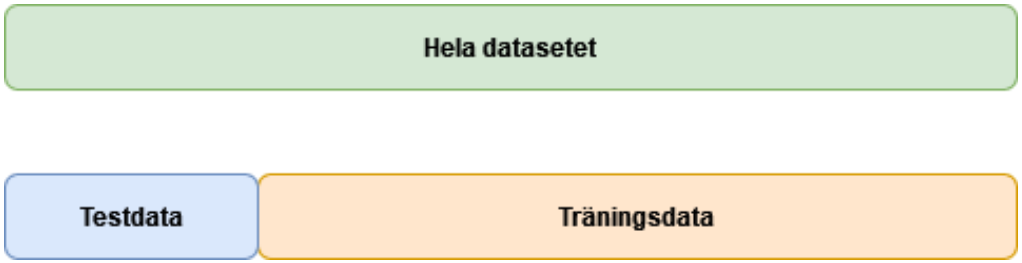
Modellval - träningsdata och testdata

- Vi har sagt att en generaliserbar modell är en modell som fungerar även för **ny data**, det vill data som vi inte använde när vi anpassade modellen.
- Ett bra sätt att bedöma om modellen är generaliserbar är att helt enkelt **testa** hur bra prediktionerna blir på ny data.
- För att göra det delar vi upp vårt dataset i två delar
  - En del är **träningsdata (training data)**. Den används för att anpassa modellen, alltså för att hitta värdet på modellens parametrar.
  - Den andra delen är **testdata**. Den används för att utvärdera modellen.



Modellval - träningsdata och testdata

- **Exempel:** Vi kan använda 70% av observationerna som träningsdata och de övriga 30% av observationerna som testdata.
- Om observationerna ligger i någon särskild ordning bör **observationerna sorteras slumpmässigt** innan vi gör en uppdelning i träningsdata och testdata. Vi vill att både träningsdata och testdata ska vara **representativ** för hela datamaterialet.



Modellval - träningsdata och testdata

Proceduren för att utvärdera en modell på testdata är

1. Dela upp observationerna i träningsdata och testdata.
2. Anpassa regressionsmodellen med hjälp av träningsdata.
3. Utvärdera modellen med hjälp av testdata.

- För att kunna genomföra steg 3 behöver vi ett mått för att utvärdera hur väl en modell passar vår testdata.
- Det finns flera möjliga mått, men här kommer vi att använda **RMSE (Rooted Mean Squared Error)**.
- Om RMSE för en modell är **lägre** betyder det att modellen är **bättre**.

Modellval - träningsdata och testdata

RMSE utvärderad på vår testdata räknas ut med formeln

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

där n_{test} är antalet observationer i vårt testset.

Uttrycket $\sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2$ är summan av de kvadrerade prediktionsfelen. Vi kallar det därför **SSE (Summed Squared Errors)**. Formeln för RMSE kan alltså också skrivas

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

Modellval - träningsdata och testdata

Uttrycket $\frac{SSE_{\text{test}}}{n_{\text{test}}}$ kallas **MSE (Mean Squared Error)**, så vi kan också skriva RMSE som

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

Modellval - träningsdata och testdata

Om vi vill kan vi räkna ut $\text{RMSE}_{\text{test}}$ i flera steg.

1. $\text{SSE}_{\text{test}} = \sum_{i=1}^{n_{\text{test}}} (y_i - \hat{y}_i)^2$
2. $\text{MSE}_{\text{test}} = \dfrac{\text{SSE}_{\text{test}}}{n_{\text{test}}}$
3. $\text{RMSE}_{\text{test}} = \sqrt{\text{MSE}_{\text{test}}}$

Eftersom vi senare ändå kommer att behöva räkna ut SSE_{test} kan det här vara ett bra sätt att räkna ut RMSE.

Modellval - träningsdata och testdata

Låt oss nu gå igenom ett **exempel** där vi

1. delar upp ett dataset i träningsdata och testdata
2. anpassar modellen med hjälp av vår träningsdata
3. utvärderar modellen genom att räkna ut RMSE för vår testdata

Modellval - träningsdata och testdata

- För vårt exempel använder vi ett dataset med variablerna y, x_1, x_2 och x_3. Vår responsvariabel är y, och övriga variabler är förklaringsvariabler.
- Vi har totalt 100 observationer.
- Vilka förklaringsvariabler vi inkluderar är upp till oss själva. Målet är att åstadkomma en modell som ger bra prediktioner i vår testdata.

Så här ser de första raderna ut i vårt dataset:

```
         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
```

Modellval - träningsdata och testdata

Steg 1: Dela upp vårt dataset

- Vi börjar med att dela in vårt dataset i ett träningsset och ett testset.
- Det gör vi genom att först lägga observationerna i slumpvis ordning.
- Därefter låter vi de första 70 observationerna utgöra träningsdata.
- De återstående 30 observationerna blir vår testdata.

```
library(dplyr)
data_randomorder <- data %>% slice_sample(prop=1)
datatrain <- data_randomorder[1:70, ]
datatest <- data_randomorder[71:100, ]
```

Modellval - träningsdata och testdata

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

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

Modellval - träningsdata och testdata

Steg 3: Utvärdera på testdata

- Nu räknar vi ut $\text{RMSE}_{\text{test}}$. Vi gör uträkningen i flera steg. Först räknar vi ut SSE_{test}, därefter MSE_{test} och slutligen $\text{RMSE}_{\text{test}}$.

```
rsquared1 <- summary(model1)$r.squared
y_hatt_test <- predict(model1, newdata=datatest)
y_test <- datatest$y
SSE <- sum((y_test - y_hatt_test)^2)
n_test <- 30
MSE <- SSE / n_test
RMSE <- sqrt(MSE)
sprintf("R-squared=%.7f, SSE=%.7f", rsquared1, SSE)
```

```
[1] "R-squared=0.8709888, SSE=146.0526235"
```

```
sprintf("MSE=%.7f, RMSE=%.7f", MSE, RMSE)
```

```
[1] "MSE=4.8684208, RMSE=2.2064498"
```

Modellval - träningsdata och testdata

Gör samma uträkningar för modellen $y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$

```
model2 <- lm(y ~ x1 + x2 + x3, data=datatrain)
rsquared2 <- summary(model2)$r.squared
y_hatt_test <- predict(model2, newdata=datatest)
y_test <- datatest$y
SSE2 <- sum((y_test - y_hatt_test)^2)
n_test <- 30
MSE2 <- SSE2 / n_test
RMSE2 <- sqrt(MSE2)
sprintf("R-squared=%.7f, SSE=%.7f", rsquared2, SSE2)
```

```
[1] "R-squared=0.8710176, SSE=146.5228725"
```

```
sprintf("MSE=%.7f, RMSE=%.7f", MSE2, RMSE2)
```

```
[1] "MSE=4.8840958, RMSE=2.2099990"
```

Modellval - träningsdata och testdata

- Vi jämför utfallet för de båda modellerna.
- Modell 1: $y = b_0 + b_1 x_1 + b_2 x_2$
- Modell 2: $y = b_0 + b_1 x_1 + b_2 x_2 + b_3 x_3$

Model	R.squared	SSE	MSE	RMSE
Model 1	0.8709888	146.0526	4.868421	2.206450
Model 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.

Modellval - träningsdata och testdata

- Att dela upp ett dataset i träningsdata och testdata ger oss en bättre bild av hur användbar en modell är.
- Vi vet dock inte säkert om det ger en rättvis bild, särskilt om det är ett dataset med få observationer.
- Vi ska visa ett **exempel** där vi använder vår bildata och modellen $\widehat{\text{litermil}} = b_0 + b_1 \cdot \text{viktton}$.
  - Först använder vi **de 12 sista** observationerna i vårt testset och de 20 första i vårt träningsset.
  - Sedan gör vi samma sak, men använder i stället **de 12 första** i vårt testset och de 20 sista i vårt träningsset.

Modellval - träningsdata och testdata

Vi kodar vårt exempel i R. Vi börjar med att inkludera de 12 sista observationerna i vår testdata. Övriga observationer inkluderar vi i vårt träningsdata.

```
# De 12 sista observationerna används i vårt testset
carstrain <- mtcars[1:20, ]
carstest <- mtcars[21:32, ]
lmod1 <- lm(litermil ~ viktton, data=carstrain)
y_hatt <- predict(lmod1, newdata=carstest)
RMSE <- sqrt(mean((carstest$litermil - y_hatt)^2))
print(RMSE)
```

```
[1] 0.1984769
```

Modellval - träningsdata och testdata

Denna gång inkluderar vi de *första* 12 observationerna i vår testdata, och övriga observationer i vår träningsdata.

```
# De 12 första observationerna används i vårt testset
carstrain <- mtcars[13:32, ]
carstest <- mtcars[1:12, ]
lmod1 <- lm(litermil ~ viktton, data=carstrain)
y_hatt <- predict(lmod1, newdata=carstest)
RMSE <- sqrt(mean((carstest$litermil - y_hatt)^2))
print(RMSE)
```

```
[1] 0.1790871
```

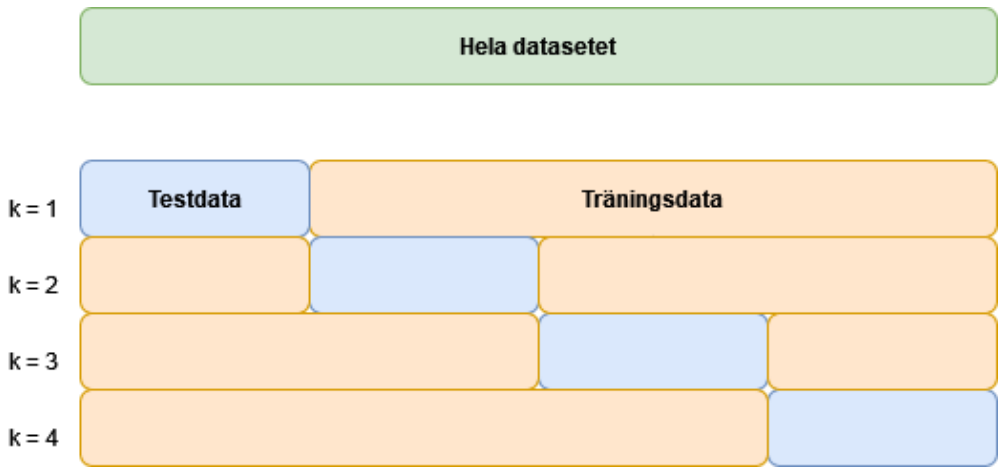
- Med de *sista* 12 observationerna som testdata fick vi RMSE $= 0.198$.
- Med de *första* 12 observationerna som testdata fick vi RMSE $= 0.179$.
- RMSE utvärderat på testdata skiljer sig alltså åt beroende på hur vi delar upp vår data.

Modellval - korsvalidering

- **Korsvalidering (cross validation)** är en metod för att utvärdera modeller som låter oss använda **alla** observationer både som träningsdata och som testdata.
- Med korsvalidering delar vi upp vårt dataset mellan träningsdata och testdata flera gånger.
- Varje gång vi gör en uppdelning är det en ny uppsättning observationer som inkluderas i vår testdata.
- Varje observation är en del av testdatan i en av uppdelningarna. I övriga uppdelningar är den en del av träningsdatan.

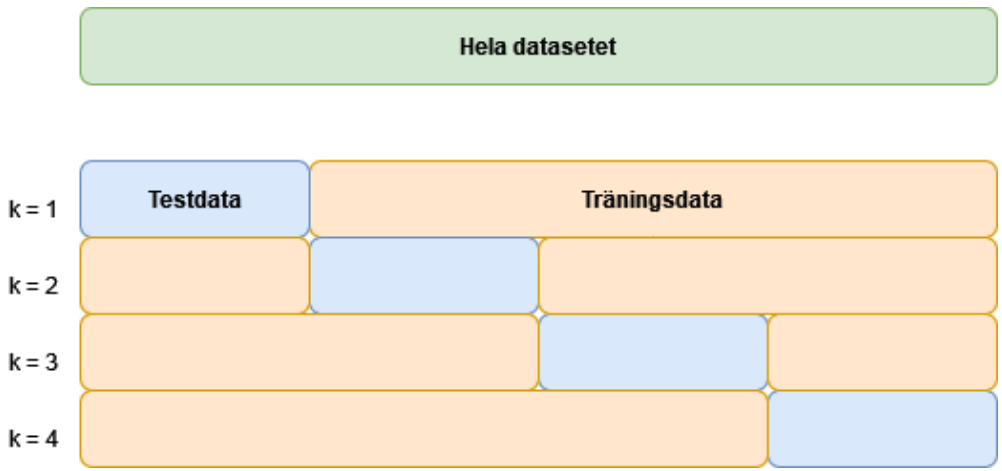
Modellval - korsvalidering

- Varje uppdelning kallar vi för en **fold**.
- Korsvalisering kallas ibland **K-fold cross validation**, där K är antalet folds.
- Korsvalidering med **K=4** betyder exempelvis att vi delar in vår data i träningsdata och testdata på 4 olika sätt.



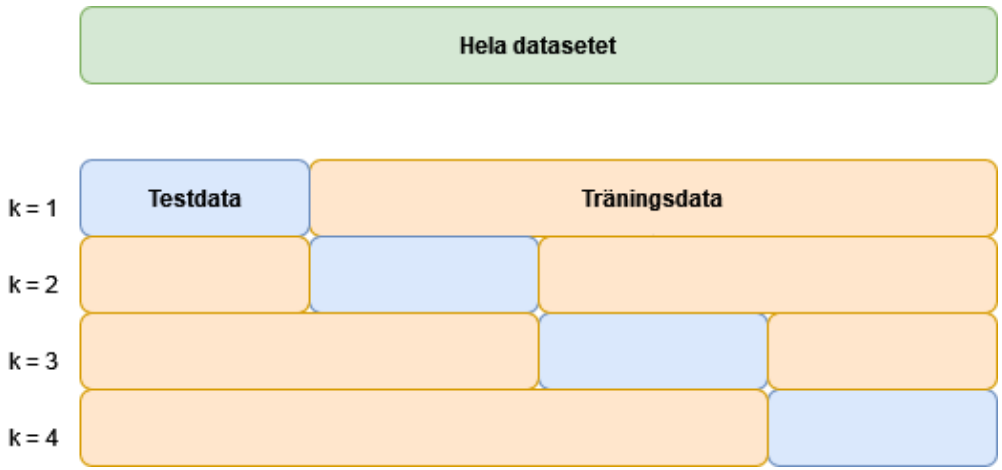
Modellval - korsvalidering

- För var och en av de 4 uppdelningarna räknar vi ut **SSE** för testdatan.
- Summan av våra 4 SSE-värden använder vi sedan till att räkna ut **RMSE**.



Modellval - korsvalidering

- Anta att vi har ett dataset med 100 observationer och $K = 4$.
  - k=1: Observation 1 till 25 är testdata.
  - k=2: Observation 26 till 50 är testdata, etc.



Modellval - korsvalidering

- Vårt mål är att räkna ut RMSE med hjälp av våra 4 folds.
- Vi börjar med att räkna ut **SSE** för var och en av våra folds.
- Vi summerar sedan våra SSE-värden

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

- När vi har räknad ut SSE_{cv} kan vi räkna ut MSE_{cv} och RMSE_{cv}.

$$\text{MSE}_{\text{cv}} = \frac{\text{SSE}_{\text{cv}}}{n}$$

$$\text{RMSE}_{\text{cv}} = \sqrt{\text{MSE}_{\text{cv}}},$$

Modellval - korsvalidering

Enkel uppdelning i träningsdata och testdata, procedur:

1. Räkna ut SSE_{test}
2. Räkna ut $\text{MSE}_{\text{test}} = \text{SSE}_{\text{test}}/n_{\text{test}}$
3. Räkna ut $\text{RMSE}_{\text{test}} = \sqrt{\text{MSE}_{\text{test}}}$

Korsvalidering, procedur:

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

Modellval - korsvalidering, exempel med K=2

- Vi ska nu gå igenom ett **exempel** på hur vi räknar ut RMSE med korsvalidering i R.
- Vi använder vår bildata, och utvärderar modellen $\widehat{\text{litermil}} = b_0 + b_1 \cdot \text{viktton}$.
- Vi sätter $K = 2$, det vill säga vi använder 2 folds.
- Vi har totalt 32 observationer.
- När vi räknar ut $\text{SSE}_{\text{test}}^{(1)}$ låter vi de 16 **första** observationerna vara testdata och övriga träningsdata.
- När vi räknar ut $\text{SSE}_{\text{test}}^{(2)}$ låter vi de 16 **sista** observationerna vara testdata och övriga träningsdata.

Modellval - korsvalidering, exempel med K=2

Den här koden använder vi för att räkna ut $\text{SSE}_{\text{test}}^{(1)}$ och $\text{SSE}_{\text{test}}^{(2)}$.

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

Modellval - korsvalidering, exempel med K=2

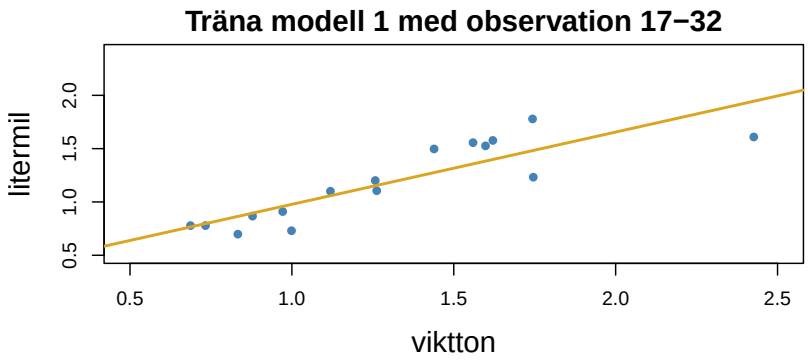
Vi skriver ut värdet på $\text{SSE}_{\text{test}}^{(1)}$ och $\text{SSE}_{\text{test}}^{(2)}$.

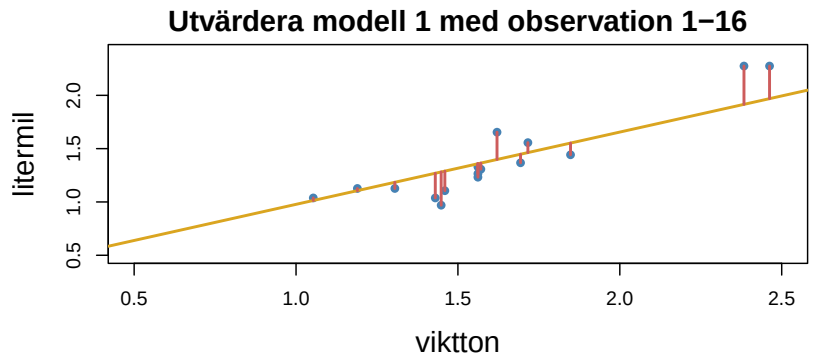
```
#Skriv ut värdet på SSE(1) och SSE(2)
sprintf("SSE1 = %.4f, SSE2 = %.4f", SSE1, SSE2)
```

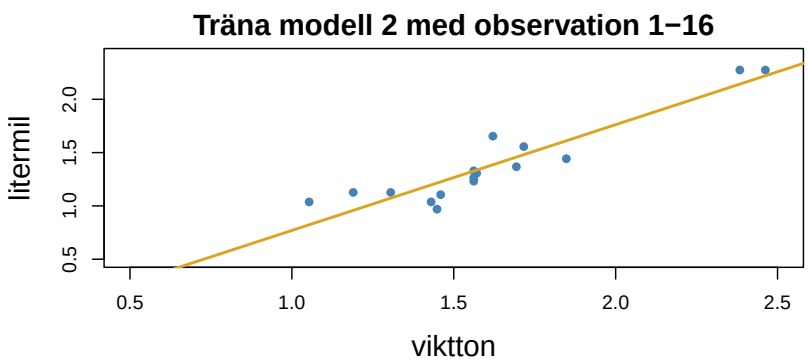
```
[1] "SSE1 = 0.5331, SSE2 = 1.0270"
```

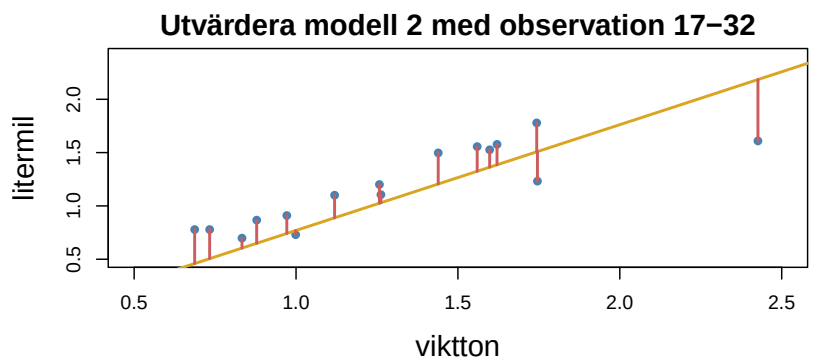
Modellval - korsvalidering, exempel med K=2

Graferna illustrerar vår data och de båda regressionsmodellerna. Om vi kvadrerar de röda linjerna i graferna till höger och summerar kvadraterna får vi $\text{SSE}_{\text{test}}^{(1)}$ (övre) och $\text{SSE}_{\text{test}}^{(2)}$ (undre).









Modellval - korsvalidering, exempel 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

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

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

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

Modellval - korsvalidering, exempel 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)
sprintf("RMSE är %.5f", RMSE_cv)
```

```
[1] "RMSE är 0.22080"
```

Modellval - korsvalidering, exempel 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.
- Vi väljer den modell om har **lägst** RMSE_{CV}.
- När vi har utvärderat våra modeller, och valt vilken modell vi ska använda, då anpassar vi vår valda modell med **all** vår data.