Forelesning 11: Moddelering

Sok-2009 h24

Eirik Eriksen Heen & ChatGPT

06. Nov 2024

Introduksjon til Lineær Regresjon

Lineær regresjon er en statistisk metode som hjelper oss med å modellere og forstå forholdet mellom variabler. Hovedmålet er å beskrive en **avhengig variabel** (utfall) ved hjelp av én eller flere **uavhengige variabler** (forklarende variabler).

Formålet med Lineær Regresjon

Når vi bruker regresjon, ønsker vi å finne en modell som kan beskrive eller forutsi hvordan den avhengige variabelen endrer seg med hensyn til de uavhengige variablene. Generelt kan vi uttrykke modellen som en funksjon:

$$y = f(x_i) + \epsilon$$

Her:

y : Den avhengige variabelen vi ønsker å forstå eller forutsi.

x : En eller flere uavhengige variabler som kan påvirke y.

f : En funksjon som beskriver den antatte sammenhengen mellom x og y – dette er modellen vi ønsker å estimere.

ϵ : En feilterm som representerer tilfeldig variasjon eller «støy» som modellen ikke forklarer.

Ofte kalles f vår **modell**, mens x refereres til som **signal** og ϵ som **støy**. Målet med regresjonsanalyse er å estimere modellen f på en måte som gir innsikt i hvordan x relaterer seg til y og lar oss vurdere påliteligheten i prediksjonene.

For lineær regresjon antar vi en spesifikk form for $f(x_i)$:

$$y = \beta_0 + \beta_1 \cdot x + \epsilon$$

Her:

- β_0 er konstantleddet eller interceptet, som representerer verdien av y når $x = 0$.
- β_1 er koeffisienten til x, som viser hvor mye y endrer seg for hver enhetsendring i x.

To Typer Modellering

Det finnes to hovedtilnærminger når vi bruker regresjon:

1. **Forklaring**
    - Vi ønsker å forstå hvordan de uavhengige variablene påvirker den avhengige variabelen.
    - Eksempel: Bruke data for å undersøke hvordan utdanning og arbeidserfaring påvirker lønn.
2. **Prediksjon**
    - Vi ønsker å bruke en modell for å forutsi verdien til den avhengige variabelen for nye observasjoner.
    - Eksempel: Bygge en modell for å predikere lønn gitt utdanning og arbeidserfaring.

Estimert Modell og «Hatt»-Notasjon

Når vi bruker data til å beregne koeffisientene i modellen, estimerer vi dem. Dette innebærer at vi går fra en teoretisk modell med ukjente parametere (β_0 og β_1) til en **estimert modell** der disse parametrene har spesifikke verdier basert på dataene. Den estimerte modellen skrives slik:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 \cdot x$$

I denne estimerte modellen:

- **Koeffisientene** får en «hatt» ($\hat{\beta}_0$ og $\hat{\beta}_1$) for å vise at de er estimater, ikke de «sanne» verdiene. Estimatene er basert på dataene vi har og er det beste anslaget vi har for de sanne koeffisientene. Men vi vet ikke hva de sanne verdiene er.
- **Feiltermen** (ϵ) forsvinner og erstattes av **residualer**, som representerer forskjellen mellom faktiske og predikerte verdier ($y - \hat{y}$).

- **Estimat** ($\hat{y}$) er modellens prediksjon for en gitt x-verdi, basert på de estimerte koeffisientene.

Residualene – Modellens Feil

Når vi estimerer en modell, gir residualene oss informasjon om nøyaktigheten. En residual er forskjellen mellom en observasjons faktiske verdi og modellens prediksjon:

$$Residual = y - \hat{y} = \epsilon$$

Residualene viser oss hvor godt modellen passer dataene: Jo mindre residualene er, desto bedre samsvar er det mellom modellen og dataene. Vi kan bruke residualene til å evaluere modellens kvalitet og nøyaktighet.

Sammendrag

En lineær regresjonsmodell lar oss beskrive eller predikere en avhengig variabel ut fra uavhengige variabler. Når modellen er estimert, mister vi feiltermen, og koeffisientene får «hatt»-notasjon. Modellens nøyaktighet vurderes gjennom residualene, som viser hvor langt unna de predikerte verdiene er fra de faktiske verdiene.

Dette gir grunnlaget for å diskutere hvordan residualer og andre mål kan hjelpe oss med å evaluere modellens kvalitet, noe vi vil utforske videre i dette notatet.

Eksempel og oppgave

Analyse av Sammenhengen Mellom Evaluering og Skjønnhet Vi undersøker sammenhengen mellom lærerevalueringer (score) og læreres gjennomsnittlige skjønnhetsscore (bty_avg). Dataene ble analysert ved hjelp av en lineær regresjonsmodell:

$$\text{score} = \beta_0 + \beta_1 \cdot \text{btyavg} + \epsilon$$

```r
# Gjennoomfører regresjonen
modell_1 <- lm(score ~ bty_avg, data = evals)

summary(modell_1)
```

```
Call:
lm(formula = score ~ bty_avg, data = evals)

Residuals:
       Min         1Q     Median         3Q        Max
-1.9246515 -0.3690318  0.1419855  0.3976941  0.9308793

Coefficients:
             Estimate Std. Error  t value    Pr(>|t|)
(Intercept) 3.88033795 0.07614297 50.96121  < 2.22e-16 ***
bty_avg     0.06663704 0.01629115  4.09038 0.000050827 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5348351 on 461 degrees of freedom
Multiple R-squared:  0.03502226,    Adjusted R-squared:  0.03292903
F-statistic: 16.73123 on 1 and 461 DF,  p-value: 0.00005082731
```

OBS! Se at Residual standard error: 0.5348 på outputen over. Vi trenger den til senere

Fra dette kan vi sette opp den estimerte modellen:

$$\hat{y} = 3.880 + 0.0666 \cdot \widehat{\text{bty}}_{\text{avg}}$$

- **Tolkning av Koeffisientene**
  - a) Hva representerer konstantleddet (intercept) på 3.8803 i denne modellen?
  - b) Tolk koeffisienten til **bty_avg** (0.0666) og forklar hva dette sier om sammenhengen mellom skjønnhet og evaluering.
- **Statistisk Signifikans**
  - a) Er koeffisienten til **bty_avg** statistisk signifikant? Begrunn svaret ved å se på p-verdien og t-verdien.
  - b) Hva betyr det for vår modell at koeffisienten er statistisk signifikant?
- **Evaluering av Modellens Forklaringskraft**
  - a) Hva indikerer R^2-verdien på 0.0350 om modellens evne til å forklare variasjonen i **score**?
  - b) Diskuter om denne modellen er egnet til å predikere **score** basert på **bty_avg**, basert på verdiene til R^2 og Residual Standard Error.

- **Utvidelse av Modellen**
  - Gi et forslag til en eller flere variabler som kunne inkluderes i modellen for å forbedre forklaringskraften. Begrunn valget ditt kort.

Hva Gjør en Modell God?

Når vi har bygd en regresjonsmodell, ønsker vi å evaluere hvor godt modellen beskriver dataene. En viktig del av denne evalueringen er å se på **residualene**, som er forskjellen mellom de observerte verdiene og de verdiene modellen predikerer.

Residualer og Feilterm

Residualene representerer modellens "feil" ved hvert datapunkt. De viser hvor langt unna de predikerte verdiene ($\hat{y}$) er fra de observerte verdiene (y). Dette kan uttrykkes med følgende formel for residualen (e):

$$\text{Residual} = y - \hat{y} = \epsilon$$

Her:

- y er den faktiske observerte verdien av den avhengige variabelen.
- $\hat{y}$ er verdien predikert av modellen.
- e (eller residual) viser avviket mellom prediksjon og observasjon.

Et mål på hvor godt modellen passer dataene er å undersøke størrelsen på residualene. **Jo mindre residualene er, desto bedre er modellen til å fange opp signalet i dataene**. Dersom alle residualene er nær null, betyr det at modellen treffer dataene veldig godt. Men i praksis vil det alltid være noen avvik, som kan skyldes tilfeldige variasjoner eller støy som modellen ikke kan forklare.

Eksempel: Forskjellen mellom Faktiske og Predikerte Verdier

Tabellen nedenfor viser et utvalg av fem observasjoner fra datasettet, der vi sammenligner de faktiske verdiene av **score** (evalueringen) med modellens prediksjoner basert på **bty_avg** (skjønnhetsscore). Kolonnen "Avvik" viser forskjellen (residualen) mellom den faktiske og predikerte verdien.

Først finner vi de predikerte verdiene før vi sammenligner dem med de faktiske verdiene for å beregne residualene.

```
# Kode for å lage tabellen

# for å finne residualene bruker vi koden "predict(modell_1)"
predict(modell_1) %>%
  head()
```

```
          1           2           3           4           5           6
4.213523140 4.213523140 4.213523140 4.213523140 4.080249066 4.080249066
```

Vi lager et nytt datasett som inneholder de faktiske verdiene, de predikerte verdiene og residualene for de fem første observasjonene.

```
# Vi legger til en kollone med prediksjonene i datasettet
evals_1 <- evals %>%
  mutate(prediksjon = predict(modell_1))

# velger ut kun score, bty_avg og prediksjon for evals_1

evals_1 <- evals_1 %>%
  # velger ut kun score, bty_avg og prediksjon
  dplyr::select(bty_avg,score, prediksjon) %>%
  # legger til residualer
  mutate(avvik_error = score - prediksjon)

set.seed(1) # Setter en tilfeldig frø for reproduksjon
# Velger 5 tilfeldig radene for å sammenligne predikerte og faktiske verdier
evals_1 <- evals_1 %>%
  slice_sample(n = 5) # Velger 5 tilfeldige rader
evals_1
```

```
# A tibble: 5 x 4
  bty_avg score prediksjon avvik_error
    <dbl> <dbl>      <dbl>       <dbl>
1    2.33   4.5       4.04       0.464
2    4.33   3.9       4.17      -0.269
3    3      3.7       4.08      -0.380
4    6.83   3.8       4.34      -0.536
5    3.33   4.7       4.10       0.598
```

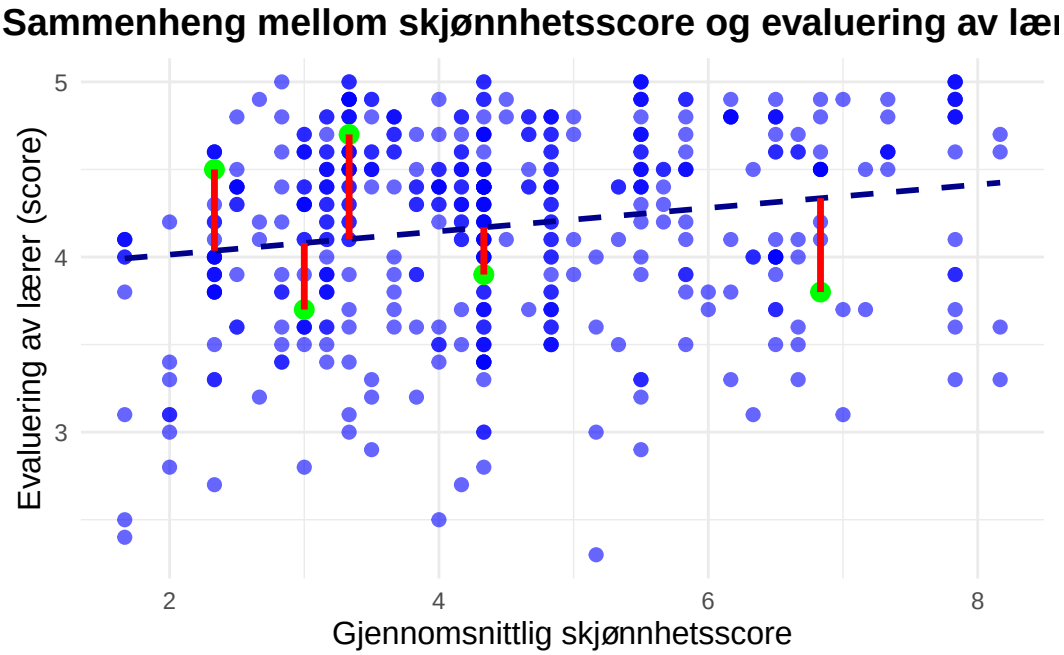
Tabellen viser hvordan den estimerte modellen predikerer verdier som kan være nær, men ikke alltid helt presise, sammenlignet med de faktiske verdiene.

Visualisering av Faktiske og Predikerte Verdier

Her har vi prediksjonen til de seks første observasjonene i datasettet. Vi kan nå sammenligne disse prediksjonene med de faktiske verdiene for å se hvor godt modellen treffer.

I plottet nedenfor visualiseres forholdet mellom skjønnhetsscore (**bty_avg**) og evaluering (**score**) ved hjelp av regresjonsmodellen. De røde linjene representerer **residualene** – avstanden mellom de faktiske og predikerte verdiene for de fem utvalgte punktene.

```
# Kode for å lage plottet

# Lager et scatterplot med regresjonslinje, og visualiserer avstanden mellom faktiske og pre
ggplot(evals, aes(x = bty_avg, y = score)) +
  geom_point(alpha = 0.6, color = "blue", size = 2) + # Plott av alle datapunkter
  geom_point(data = evals_1, aes(x = bty_avg, y = score), color = "green", size = 3) + # Fre
  geom_smooth(method = "lm", se = FALSE, color = "darkblue", linetype = "dashed") + # Legger
  geom_segment(data = evals_1,
               aes(x = bty_avg, xend = bty_avg, y = score, yend = prediksjon),
               color = "red", linetype = "solid", size=1.2) + # Viser avstand mellom faktisk
  labs(title = "Sammenheng mellom skjønnhetsscore og evaluering av lærere",
       x = "Gjennomsnittlig skjønnhetsscore",
       y = "Evaluering av lærer (score)") +
  theme_minimal() + # Bruker minimalistisk tema for bedre lesbarhet
  theme(plot.title = element_text(hjust = 0.5, face = "bold", size = 14), # Sentrerer og fre
        axis.title = element_text(size = 12)) # Øker skriftstørrelsen på aksetitler
```



Dette plottet viser hvordan residualene (røde linjer) illustrerer forskjellen mellom de faktiske verdiene og modellens prediksjoner. Jo kortere disse linjene er, desto bedre beskriver modellen variasjonen i dataene.

Vi måler alle disse avvikene, men i eksempelet over illustrerer vi kun 5 av dem.

Residual Standard Error (RSE)

Residual Standard Error (RSE) er et mål på hvor mye de observerte verdiene i datasettet avviker fra de verdiene modellen predikerer. RSE gir en gjennomsnittlig størrelse på residualene, eller «feilene», etter at vi har kvadrert dem og justert for modellens kompleksitet.

Utledning av RSE

RSE beregnes ved å finne summen av kvadrerte residualer (SSR), dele denne summen på antall frihetsgrader, og deretter ta kvadratroten av resultatet. Dette gir oss et mål som kan tolkes som en gjennomsnittlig prediksjonsfeil for modellen:

$$\text{RSE} = \sqrt{\frac{\sum(y - \hat{y})^2}{n - k - 1}}$$

Her:

- y er den observerte verdien.
- $\hat{y}$ er den predikerte verdien fra modellen.
- n er antallet observasjoner i datasettet.
- k er antallet uavhengige variabler i modellen.

Den justeringen med $n - k - 1$ tar hensyn til kompleksiteten i modellen ved å kompensere for antallet variabler som kan påvirke tilpasningen. Dette sikrer at RSE gir et mer nøyaktig bilde av modellens ytelse, selv når modellen har flere variabler.

Hvorfor Bruke RSE?

RSE gir oss en standardisert måte å evaluere modellen på, ved å uttrykke feilen i samme enhet som den avhengige variabelen. En lav RSE indikerer at modellen predikerer den avhengige variabelen med høy nøyaktighet, mens en høy RSE antyder større feil og dermed lavere nøyaktighet. RSE er derfor et nyttig mål for å sammenligne hvor godt forskjellige modeller passer dataene.

Beregning av RSE

Vi kan beregne RSE for regresjonsmodellen vår i R ved først å hente residualene, deretter beregne SSR, dele på frihetsgradene, og ta kvadratroten av resultatet.

```
# Bygger en lineær regresjonsmodell med 'bty_avg' som forklarende variabel og 'score' som avh
modell_2 <- lm(score ~ bty_avg, data = evals)

# Henter residualene fra modellen og beregner RSE
residuals <- resid(modell_2) # Residualene fra modellen
SSR <- sum(residuals^2) # Kvadrerer og summerer residualene
n <- nrow(evals) # Antall observasjoner
k <- 1 # Antall forklarende variabler

# Beregner RSE
RSE <- sqrt(SSR / (n - k - 1))
RSE
```

```
[1] 0.5348350999
```

```
rm(n, k, residuals, SSR, RSE) # Fjerner midlertidige variabler
```

Her ser vi at vi får samme verdi nå som vi fikk tidligere. Dette viser at RSE gir oss en standardisert måte å evaluere modellen på, ved å uttrykke feilen i samme enhet som den avhengige variabelen.

Hvordan Bruke RSE til Modellvurdering

1. **Lav RSE betyr bedre modell**
    - En modell med lav RSE predikerer den avhengige variabelen med høyere nøyaktighet, da den gjennomsnittlige residualen er lavere.
2. **Bruk av RSE til Modellvalg**
    - RSE kan brukes som et sammenligningsmål for å velge mellom flere modeller. En modell med lavere RSE har generelt bedre prediksjonsnøyaktighet på treningsdataene.
3. **Begrensninger – Overtilpasning**
    - Selv om en modell har lav RSE på treningsdataene, er det ikke nødvendigvis slik at den vil prestere godt på nye data. Dette fenomenet, kalt **overtilpasning** (overfitting), oppstår når modellen tilpasser seg treningsdataene for nøye og ikke generaliserer godt.

Oppsummering

Residual Standard Error (RSE) gir et mål på hvor godt modellen predikerer dataene. Ved å minimere RSE oppnår vi en modell som i størst mulig grad fanger opp sammenhengen i dataene, men uten overtilpasning. Likevel må vi være oppmerksomme på at en modell med lav RSE på treningsdataene ikke nødvendigvis vil fungere godt på nye data, og vi bør derfor bruke flere metoder for å evaluere modellens generaliseringsevne.

Out-of-Sample Testing

Når vi bygger en regresjonsmodell, er det ikke nok å bare se på hvor godt den passer treningsdataene. Det er avgjørende å vurdere om modellen også vil fungere godt på nye, ukjente data. Dette er fordi en modell som passer veldig godt til treningsdataene kan ha tilpasset seg tilfeldigheter i dataene som ikke vil gjenta seg i nye observasjoner. **Out-of-sample** testing hjelper oss med å evaluere modellens evne til å generalisere, altså hvor godt den kan predikere på data den ikke har sett før.

Hva er Out-of-Sample Testing?

Out-of-sample testing innebærer å dele datasettet i to deler:

1. **Treningssett** – Dette er dataene vi bruker til å tilpasse modellen og beregne koeffisientene.
2. **Testsett** – Dette er en egen del av dataene som brukes til å teste modellen. Testsettet fungerer som «nye» data for modellen, og lar oss evaluere hvordan modellen presterer på data den ikke har sett under treningen.

Ved å teste modellen på testsettet får vi et mål på modellens **generaliseringsevne** – hvor godt modellen kan predikere nye data. Hvis modellen presterer godt både på treningssettet og testsettet, er det et godt tegn på at den kan gi nøyaktige prediksjoner på nye data.

Hvorfor bruker vi Mean Squared Error?

For å evaluere nøyaktigheten til modellen på testsettet bruker vi ofte **Mean Squared Error (MSE)**, som måler den gjennomsnittlige kvadrerte feilen mellom de faktiske og de predikerte verdiene. MSE gir oss en oppsummering av modellens avvik på testsettet. En lav MSE betyr at modellen har lav gjennomsnittlig feil, og dermed høy nøyaktighet på testdataene. MSE gir et mer realistisk bilde av modellens ytelse fordi det viser hvordan modellen fungerer på nye data, noe som reduserer risikoen for overtilpasning (overfitting).

Hva er overtilpasning (overfitting)?

Overtilpasning oppstår når en modell tilpasser seg treningsdataene for nøye og fanger opp ikke bare den generelle trenden, men også tilfeldige variasjoner og «støy» i dataene. Dette kan føre til en modell som ser ut til å prestere svært godt på treningssettet, men som feiler når den skal predikere på nye, ukjente data.

Når en modell er overtilpasset, er den i praksis «overtrent» på treningsdataene og har mistet sin generaliseringsevne. Overtilpassede modeller kan være svært komplekse, med mange parametere som gjør at modellen «husker» detaljene i treningsdataene. Dette kan resultere i unødvendige svingninger og tilpasninger som ikke vil være relevante i et testsett eller i fremtidige data.

Hvordan Oppdager vi Overtilpasning?

Out-of-sample testing er en av de mest effektive metodene for å oppdage overtilpasning. Hvis en modell har veldig lav feil på treningssettet (lav RSE), men betydelig høyere feil på testsettet (høy MSE), er det et tegn på at modellen kan være overtilpasset. Ved å bruke MSE på testsettet får vi et mer realistisk bilde av modellens ytelse og kan unngå å velge en modell som bare presterer godt på treningsdataene.

Hvordan Kan Vi Unngå Overtilpasning?

For å unngå overtilpasning kan vi:

1. **Bruke enklere modeller** som fokuserer på hovedtrenden i dataene og ikke på små, tilfeldige variasjoner.
2. **Begrense antall forklarende variabler** i modellen for å redusere kompleksiteten.
3. **Bruke out-of-sample testing** (f.eks. et testsett) for å sørge for at modellen presterer godt også på nye data, ikke bare treningsdataene.

En balanse mellom modellens kompleksitet og generaliseringsevne gir en robust modell som både passer dataene godt og gir gode prediksjoner på nye data.

Beregning av Mean Squared Error (MSE)

For å måle hvor godt modellen fungerer på testsettet, kan vi bruke **Mean Squared Error (MSE)**. MSE beregner den gjennomsnittlige kvadrerte avstanden mellom de observerte verdiene og de predikerte verdiene. En lav MSE indikerer at modellen predikerer med høy nøyaktighet på testdataene.

Formelen for MSE er:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Her:

- y_i er den faktiske verdien.
- $\hat{y}$ er den predikerte verdien fra modellen.
- n er antall observasjoner i testsettet.

Hvis vi tar kvadratroten av MSE, får vi **Root Mean Squared Error (RMSE)**, som gir en mer tolkbar verdi i samme enhet som den avhengige variabelen. Men når vi sammenligner modeller, er MSE og RMSE like nyttige. Men for lesbarhet kan RMSE være mer intuitivt.

Beregning av Root Mean Squared Error (RMSE)

For å beregne RMSE, tar vi kvadratroten av MSE. Dette gir oss en gjennomsnittlig feil i samme enhet som den avhengige variabelen.

$$\text{RMSE} = \sqrt{\text{MSE}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Tokningen av RMSE er at det gir en gjennomsnittlig feil i samme enhet som den avhengige variabelen. Dette gjør det lettere å tolke modellens nøyaktighet og sammenligne modeller.

Eksempel på Out-of-Sample Testing

La oss bruke 80 % av datasettet som treningssett og de resterende 20 % som testsett. Vi bygger modellen på treningssettet og beregner MSE på testsettet for å vurdere modellens nøyaktighet.

```
# Del datasettet i treningssett (80%) og testsett (20%)
set.seed(1) # Setter frø for å sikre samme tilfeldige deling
train_index <- sample(seq_len(nrow(evals)), size = 0.8 * nrow(evals))
train_data <- evals[train_index, ]
test_data <- evals[-train_index, ]

# Bygger en lineær regresjonsmodell på treningssettet
modell <- lm(score ~ bty_avg, data = train_data)

# Predikerer score på testsettet og beregner RMSE
test_predictions <- predict(modell, newdata = test_data)

# Beregner RMSE på testsettet
rmse <- sqrt( mean((test_data$score - test_predictions)^2) )
rmse
```

```
[1] 0.514345086
```

```
rm(train_index, train_data, test_data, modell, test_predictions, rmse) # Fjerner midlertidig
```

I dette eksempelet:

1. **`sample`** brukes til å trekke en tilfeldig andel av dataene som treningssett (80 %).
2. Modellen trenes på treningssettet.

3. Modellens prediksjoner sammenlignes med de faktiske verdiene i testsettet, og vi beregner MSE som et mål på prediksjonsfeilen.

Her fant vi kunne en **RMSE**. Men vi kunne ha fått en annen verdi gitt at vi delte datasettet på en annen måte. Derfor er det viktig å gjøre flere delinger og gjennomsnittet av RMSE for å få en mer nøyaktig verdi.

Vi kan bruke koden ´replicate´ for å gjøre dette. Vi gjennomfører koden 20 ganger og beregner RMSE for hver repetisjon. Deretter beregner vi gjennomsnittlig RMSE over de 20 repetisjonene.

```
set.seed(1) # Setter frø for å sikre samme tilfeldige deling

# Setter antall repetisjoner
n_repeats <- 20

# Kjører koden 20 ganger og beregner RMSE for hver repetisjon
rmse <- replicate(n_repeats, {
  # Deler datasettet i treningssett (80%) og testsett (20%)
  train_index <- sample(seq_len(nrow(evals)), size = 0.8 * nrow(evals))

  # Bygger en lineær regresjonsmodell på treningssettet
  modell <- lm(score ~ bty_avg,
               # her hentes ut kun treningsdatasettet
               data = evals[train_index, ])

  # Predikerer score på testsettet og beregner RMSE
  test_predictions <- predict(modell,
                              # her hentes ut kun testdatasettet
                              newdata = evals[-train_index, ])
  # regner ut RMSE
  sqrt(mean((evals[-train_index, ]$score - test_predictions)^2))
})

# Beregner gjennomsnittlig RMSE over de 20 repetisjonene og lagrer resultatet som modell 1
resultat <- tibble(rmse, modell = 1)

# beregner gjennomsnittlig RMSE for modell 1
resultat %>%
  summarize(mean(rmse))
```

```
# A tibble: 1 x 1
  `mean(rmse)`
```

```
          <dbl>
1         0.526
```

Vi ser at vi ikke får det samme resultatet som tidligere. Dette viser at RMSE kan variere avhengig av hvordan datasettet deles inn i trenings- og testsett. Ved å gjennomføre flere delinger og beregne gjennomsnittet av RMSE, får vi et mer nøyaktig mål på modellens ytelse.

Hvordan Bruke Out-of-Sample MSE til Modellvalg

1. **Lav Out-of-Sample MSE betyr bedre modell**
    - En modell med lav MSE på testsettet gir mer presise prediksjoner og har høyere generaliseringsevne.
2. **Sammenligning av Modeller**
    - Hvis vi har flere modeller, kan MSE på testsettet brukes til å sammenligne modellene. Modellen med lavest out-of-sample MSE er vanligvis den som predikerer best på nye data.
3. **Fordelen med Out-of-Sample Testing**
    - Out-of-sample testing gir et realistisk mål på modellens ytelse fordi den testes på data den ikke har sett før, noe som reduserer risikoen for overtilpasning.

Oppsummering

Out-of-sample testing ved å dele datasettet i et treningssett og et testsett gir oss et enkelt og effektivt mål på modellens generaliseringsevne. Ved å måle MSE på testsettet kan vi vurdere hvor godt modellen vil prestere på nye data, og dermed velge en modell som både er nøyaktig og robust.

Sammenligning av Modeller

For å finne den modellen som gir best prediksjoner, kan vi sammenligne ulike modeller med forskjellige forklarende variabler. Her tester vi to modeller:

1. En enkel modell med én variabel (**bty_avg**).
2. En utvidet modell med to variabler (**bty_avg** og **age**).

Vi beregner **Residual Standard Error (RSE)** på testsettet for å evaluere hvilken modell som predikerer best.

Eksempel med Flere Modeller

Modell 1: Regresjon med Én Variabel (bty_avg)

Vi har allredet regnet ut RSE for modellen med en variabel (**bty_avg**). Denne verdien er gitt it resultat. Nå vil vi sammenligne denne modellen med en utvidet modell som inkluderer en ekstra variabel (**age**).

Modell 2: Regresjon med To Variabler (bty_avg og age)

Deretter utvider vi modellen med variabelen **age** som en ekstra forklarende variabel, og beregner RSE og MSE på samme måte som for Modell 1.

```
set.seed(1) # Setter frø for å sikre samme tilfeldige deling

# Setter antall repetisjoner
n_repeats <- 20

# Kjører koden 20 ganger og beregner RMSE for hver repetisjon
rmse <- replicate(n_repeats, {
  # Deler datasettet i treningssett (80%) og testsett (20%)
  train_index <- sample(seq_len(nrow(evals)), size = 0.8 * nrow(evals))

  # Bygger en lineær regresjonsmodell på treningssettet
  modell <- lm(score ~ bty_avg+age,
               # her hentes ut kun treningsdatasettet
               data = evals[train_index, ])

  # Predikerer score på testsettet og beregner RMSE
  test_predictions <- predict(modell,
                              # her hentes ut kun testdatasettet
                              newdata = evals[-train_index, ])
  # regner ut RMSE
  sqrt(mean((evals[-train_index, ]$score - test_predictions)^2))
})

# Omgjør RMSE til en tibble og legger til modellnummer
temp <- tibble(rmse, modell = 2)

# Legger til resultatet for modell 2 til resultatet for modell 1
resultat <- bind_rows(resultat, temp)
```

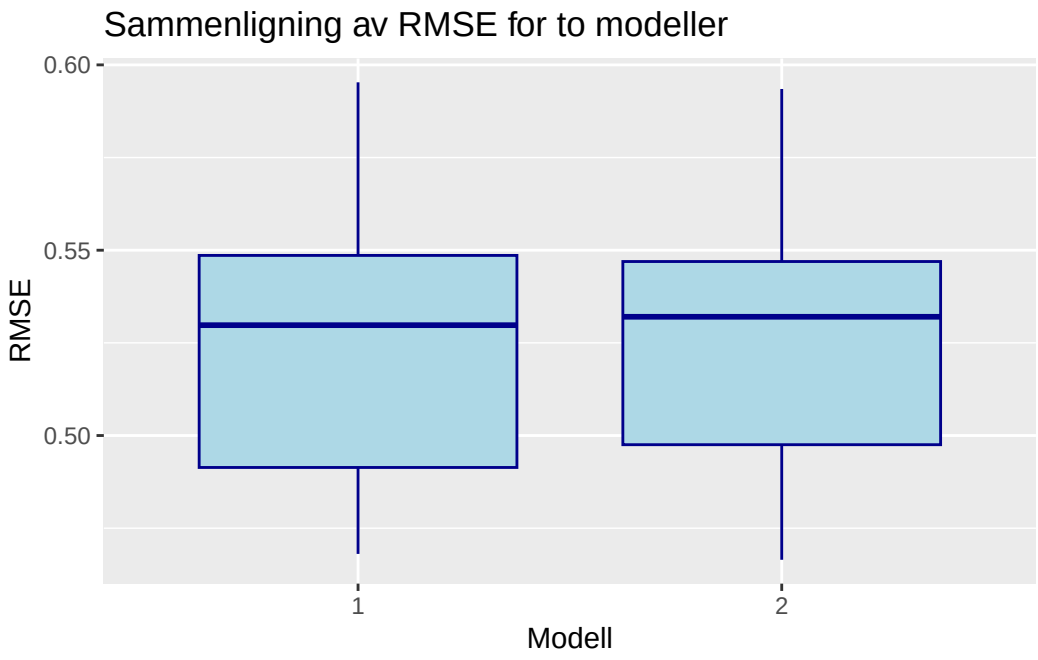
```r
# Beregner gjennomsnittlig RMSE for begge modellene
resultat %>%
  group_by(modell) %>%
  summarize(mean(rmse))
```

```
# A tibble: 2 x 2
  modell `mean(rmse)`
   <dbl>        <dbl>
1      1        0.526
2      2        0.527
```

Fra gjennomsnitlig rmse for begge modellene kan vi sammenligne hvilken modell som gir best prediksjoner. Her ser det ut som om RMSE går opp hvis vi legger til alder som en variabel. Dette kan indikere at modellen med bare **bty_avg** gir bedre prediksjoner på testsettet.

Vi kan visualisere resultatene for å se hvordan modellene presterer i forhold til hverandre.

```r
# Lager histogram for RMSE for begge modellene
ggplot(resultat, aes(y=rmse, x = factor(modell) )) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  labs(title = "Sammenligning av RMSE for to modeller",
       x = "Modell",
       y = "RMSE")
```



Det ser ikke ut til at det er noe stor forskell, for å være sikker kan vi gjennomføre en t-test mellom dem for å være sikker.

```
# Utfører en t-test for å sammenligne RMSE for de to modellene
t.test(rmse ~ modell, data = resultat)
```

```
    Welch Two Sample t-test

data:  rmse by modell
t = -0.15167879, df = 37.997911, p-value = 0.8802428
alternative hypothesis: true difference in means between group 1 and group 2 is not equal to
95 percent confidence interval:
 -0.02417774614  0.02080722954
sample estimates:
mean in group 1 mean in group 2
   0.5257881304    0.5274733887
```

Vi ser at p-verdien er høy, noe som indikerer at det ikke er en signifikant forskjell i RMSE mellom de to modellene. Dette betyr at vi ikke har nok bevis for å si at en modell er bedre enn den andre basert på RMSE alene. Hvis det er ikke er en forbedring ved å legge til en variabel, så holde seg til den enklere modellen.

Modell 3: Regresjon med Tre Variabler

Vi kan også teste en modell med tre variabler for å se om dette gir bedre prediksjoner enn de to tidligere modellene. Vi legger til variablene kjønn og etnisitet som forklarende variabler i modellen.

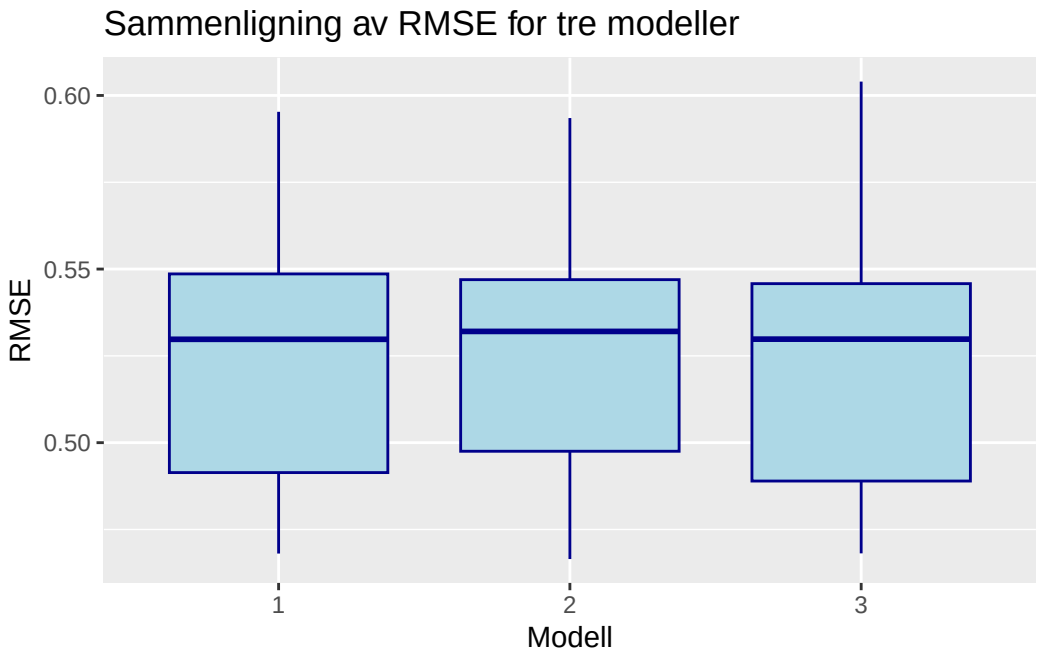
```
set.seed(1) # Setter frø for å sikre samme tilfeldige deling

# Setter antall repetisjoner
n_repeats <- 20

# Kjører koden 20 ganger og beregner RMSE for hver repetisjon
rmse <- replicate(n_repeats, {
  # Deler datasettet i treningssett (80%) og testsett (20%)
  train_index <- sample(seq_len(nrow(evals)), size = 0.8 * nrow(evals))

  # Bygger en lineær regresjonsmodell på treningssettet
  modell <- lm(score ~ bty_avg+gender,
```

```
                # her hentes ut kun treningsdatasettet
                data = evals[train_index, ])

  # Predikerer score på testsettet og beregner RMSE
  test_predictions <- predict(modell,
                              # her hentes ut kun testdatasettet
                              newdata = evals[-train_index, ])
  # regner ut RMSE
  sqrt(mean((evals[-train_index, ]$score - test_predictions)^2))
})

# Omgjør RMSE til en tibble og legger til modellnummer
temp <- tibble(rmse, modell = 3)

# Legger til resultatet for modell 3 til resultatet for modell 1 og 2
resultat <- bind_rows(resultat, temp)

# Beregner gjennomsnittlig RMSE for alle tre modellene
resultat %>%
  group_by(modell) %>%
  summarize(mean(rmse))
```

```
# A tibble: 3 x 2
  modell `mean(rmse)`
   <dbl>        <dbl>
1      1        0.526
2      2        0.527
3      3        0.523
```

Vi ser at RMSE går litt ned når vi legger til kjønn som en variabel. Først sammenlikner vi resultatene i en boksplott.

```
# Lager histogram for RMSE for alle tre modellene
ggplot(resultat, aes(y=rmse, x = factor(modell) )) +
  geom_boxplot(fill = "lightblue", color = "darkblue") +
  labs(title = "Sammenligning av RMSE for tre modeller",
       x = "Modell",
       y = "RMSE")
```



Det ser ikke ut til at det er stor forskell. Vi kan gjennomføre en t-test for å være sikker.

```
# gjennomfører en t-test for å sammenligne RMSE for de tre modellene, men kun for modell 1 og
temp <- resultat %>%
  filter(modell %in% c(1,3))

t.test(rmse ~ modell, data = temp )
```

```
    Welch Two Sample t-test

data:  rmse by modell
t = 0.21687578, df = 37.963009, p-value = 0.8294665
alternative hypothesis: true difference in means between group 1 and group 3 is not equal to
95 percent confidence interval:
 -0.02032872266  0.02520684697
sample estimates:
mean in group 1 mean in group 3
   0.5257881304    0.5233490682
```

Ikke overraskende ser vi at det ikke er noe forskell mellom modell 1 og 3. Dette betyr at det ikke er noen forbedring ved å legge til kjønn som en variabel. Derfor er det best å holde seg til den enkleste modellen.

Oppgave

Vi har nå sett på hvordan vi kan evaluere modeller ved hjelp av **Residual Standard Error (RSE)** og **Mean Squared Error (MSE)**. Vi har også sett på hvordan vi kan bruke **Out-of-Sample Testing** for å vurdere modellens generaliseringsevne og unngå overtilpasning.

Nå er det på tide å teste dine ferdigheter! Her er en oppgave for deg: For datasettet **house_prices**, bygg en regresjonsmodell som predikerer husprisene basert på de tilgjengelige variablene. Del datasettet i et treningssett og et testsett, og beregn **RMSE** for modellen på testsettet. Sammenlign modellen med en enkel modell som bare inkluderer én variabel, og diskuter hvilken modell som gir best prediksjoner.

Pairwise T-Test og Multiple Testing Korreksjon

Når vi ønsker å undersøke forskjeller mellom flere grupper, som høydeforskjeller mellom etniske grupper, trenger vi ofte å utføre flere t-tester for hver parvise kombinasjon av gruppene. I R kan vi bruke funksjonen `pairwise.t.test` for å utføre disse parvise testene på en enkel måte. Men når vi tester mange grupper samtidig, øker risikoen for tilfeldige signifikante resultater (Type I-feil), og derfor trenger vi også å bruke en metode for **multiple testing korreksjon**.

I dette delkapittelet bruker vi NHANES-datasettet for å analysere om det er signifikante høydeforskjeller mellom forskjellige etniske grupper, samtidig som vi kontrollerer for risikoen for falske positive ved å bruke korreksjonsmetoder.

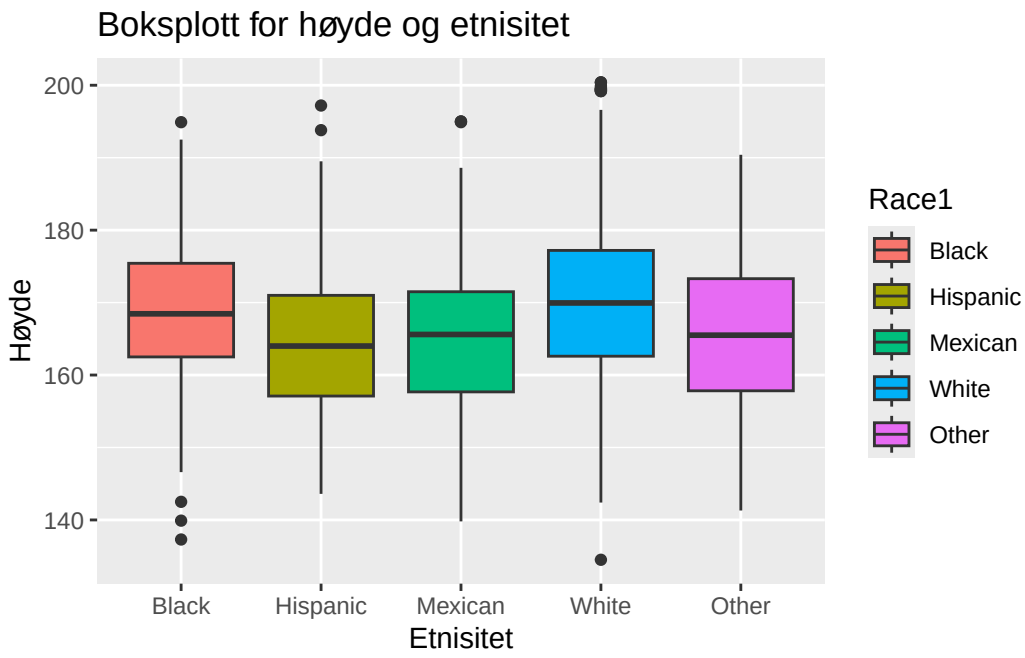
Utføre Pairwise T-Test med NHANES-data

`pairwise.t.test` i R gjør det enkelt å teste høydeforskjeller mellom grupper i NHANES-datasettet. La oss først laste inn dataene og se på de relevante variablene, `Height` (høyde) og `Race1` (etnisitet). Vi filterer også for personer som er under 20 år, for å fokusere på voksne.

```r
# Laster nødvendige pakker og data
library(NHANES)

# Filtrerer datasettet for relevante variabler
data <- NHANES %>%
  filter(Age >= 20) %>% # Fokuserer på voksne
  dplyr::select(Height, Race1) %>%
  filter(!is.na(Height), !is.na(Race1)) # Fjerner NA-verdier
```

Før vi gjennomfører en test plotter vi dataen i et boksplott for å se hvordan høyden varierer mellom de ulike etnisitetene.

```r
# Lager et boksplott for høyde og etnisitet
ggplot(data, aes(x = Race1, y = Height, fill = Race1)) +
  geom_boxplot() +
  labs(title = "Boksplott for høyde og etnisitet",
       x = "Etnisitet",
       y = "Høyde")
```



Vi lager også en tabell for å se på gjennomsnitts høyden og antall observasjoner for hver etnisitet.

```r
# Lager en tabell for gjennomsnittlig høyde og antall observasjoner per etnisitet
data %>%
  group_by(Race1) %>%
  summarize(mean_height = mean(Height), n = n())
```

```
# A tibble: 5 x 3
  Race1    mean_height     n
  <fct>          <dbl> <int>
1 Black           169.   816
2 Hispanic        164.   416
3 Mexican         165.   596
4 White           170.  4812
5 Other           165.   542
```

Nå har vi dataene klare med høyde og etnisitet som variabler. Vi kan bruke `pairwise.t.test` til å utføre parvise t-tester for å undersøke høydeforskjeller mellom gruppene.

```
# Utfører parvise t-tester med Bonferroni-korreksjon
pairwise.t.test(data$Height, data$Race1, p.adjust.method = "bonferroni")
```

```
    Pairwise comparisons using t tests with pooled SD

data:  data$Height and data$Race1

         Black      Hispanic   Mexican    White
Hispanic 1.0123e-15 -          -          -
Mexican  1.3039e-13 1.000000   -          -
White    0.081871   < 2.22e-16 < 2.22e-16 -
Other    3.2637e-10 0.409563   1.000000   < 2.22e-16

P value adjustment method: bonferroni
```

Her sammenligner vi høyden mellom alle par av etnisiteter i `Race1`, og vi har valgt Bonferroni-korreksjon som metode for å justere for multiple testing. Funksjonen returnerer p-verdiene for hver sammenligning, som gir en indikasjon på hvor sannsynlig det er at forskjellene i høyde mellom gruppene er tilfeldige.

Multiple Testing Korreksjon

Første kolloen har vi p-verdiene for Black, de er forskjellige fra alle enn White. P-verdiene for Hispanic er forskjellige fra alle enn Mexican og Other. P-verdiene for White er forskjellige fra alle.

Hvorfor Bruke Korreksjon for Multiple Testing?

Når vi utfører flere parvise t-tester, øker sannsynligheten for falske positive. Det vil si at vi kan finne tilsynelatende signifikante forskjeller som skyldes tilfeldigheter snarere enn reelle forskjeller. Dette er et vanlig problem når man tester mange grupper samtidig. Ved å bruke multiple testing korreksjon reduserer vi denne risikoen og får et mer pålitelig resultat.

I `pairwise.t.test` kan vi spesifisere flere ulike korreksjonsmetoder for å kontrollere Type I-feil (falske positive).

Vanlige Korreksjonsmetoder

1. **Bonferroni Korreksjon**: Denne metoden justerer p-verdiene ved å multiplisere dem med antall sammenligninger. Bonferroni-korreksjon er konservativ og reduserer sjansen for Type I-feil kraftig, men den kan også øke sjansen for Type II-feil (falsk negativ). Dette er nyttig når vi ønsker en streng kontroll over signifikansnivået.
2. **Holm Korreksjon**: En mindre konservativ metode enn Bonferroni. Holm-korreksjon sorterer p-verdiene i stigende rekkefølge og justerer dem stegvis. Dette gjør metoden litt mer fleksibel og mindre streng, samtidig som den kontrollerer Type I-feil effektivt.
3. **Benjamini-Hochberg (BH) Korreksjon**: I motsetning til Bonferroni og Holm, som kontrollerer sannsynligheten for falske positive for hver test, kontrollerer BH-korreksjonen **false discovery rate** (FDR). BH-metoden er spesielt nyttig når vi tester mange grupper og kan akseptere noen få falske positive.

Eksempel med Flere Korreksjonsmetoder

La oss se på hvordan p-verdiene påvirkes av forskjellige korreksjonsmetoder ved å bruke NHANES-dataene:

```
# Utfører parvise t-tester med forskjellige korreksjonsmetoder
pairwise.t.test(data$Height, data$Race1, p.adjust.method = "bonferroni")  # Bonferroni-korrel
```

```
    Pairwise comparisons using t tests with pooled SD

data:  data$Height and data$Race1

         Black      Hispanic   Mexican    White
Hispanic 1.0123e-15 -          -          -
Mexican  1.3039e-13 1.000000   -          -
White    0.081871   < 2.22e-16 < 2.22e-16 -
Other    3.2637e-10 0.409563   1.000000   < 2.22e-16

P value adjustment method: bonferroni
```

```
pairwise.t.test(data$Height, data$Race1, p.adjust.method = "holm")        # Holm-korreksjon
```

```
    Pairwise comparisons using t tests with pooled SD
```

```
data:  data$Height and data$Race1

         Black      Hispanic   Mexican    White
Hispanic 7.0863e-16 -          -          -
Mexican  7.8236e-14 0.362898   -          -
White    0.032748   < 2.22e-16 < 2.22e-16 -
Other    1.6318e-10 0.122869   0.419927   < 2.22e-16

P value adjustment method: holm
```

```r
pairwise.t.test(data$Height, data$Race1, p.adjust.method = "BH")          # Benjamini-Hochbe
```

```
    Pairwise comparisons using t tests with pooled SD

data:  data$Height and data$Race1

         Black      Hispanic   Mexican    White
Hispanic 2.5308e-16 -          -          -
Mexican  2.6079e-14 0.201610   -          -
White    0.011696   < 2.22e-16 < 2.22e-16 -
Other    5.4395e-11 0.051195   0.419927   < 2.22e-16

P value adjustment method: BH
```

Ved å sammenligne resultatene kan vi se hvordan p-verdiene endrer seg avhengig av korreksjonsmetoden. Bonferroni vil typisk gi høyere (mer konservative) p-verdier, mens Benjamini-Hochberg gir en mindre streng justering, noe som kan gjøre det lettere å identifisere signifikante forskjeller hvis vi kan akseptere noen falske positive.

Tolkning av Resultatene

Når vi tolker resultatene fra `pairwise.t.test` med multiple testing korreksjon, bør vi merke oss følgende:

- **P-verdier**: En lav p-verdi mellom to grupper indikerer en signifikant forskjell i høyde mellom de to etnisitetene etter korreksjon. Hvilke sammenligninger som viser signifikante forskjeller vil avhenge av korreksjonsmetoden.

- **Korreksjonsmetodevalg**: Valget av korreksjonsmetode påvirker strengen til testene våre. Hvis vi har mange sammenligninger og kan akseptere litt høyere risiko for falske positive, kan vi bruke Benjamini-Hochberg. Hvis vi ønsker streng kontroll, kan vi velge Bonferroni.

Oppsummering

Ved å bruke `pairwise.t.test` i R kan vi enkelt sammenligne gjennomsnitt mellom flere etniske grupper i NHANES-datasettet. Ved å bruke multiple testing korreksjoner som Bonferroni, Holm, eller Benjamini-Hochberg, reduserer vi risikoen for falske positive og sikrer at resultatene er robuste. Slik kan vi avgjøre om det er reelle høydeforskjeller mellom etnisiteter uten å overdrive betydningen av tilfeldige funn.

Oppgave

I forrige oppgave lagde du RMSE for tre modeller som predikerte huspriser basert på variablene i datasettet **house_prices**. Nå er det på tide å teste dine ferdigheter i multiple testing korreksjon! Bruk `pairwise.t.test` til å sammenligne gjennomsnittlige huspriser mellom de ulike etniske gruppene i datasettet. Bruk Bonferroni, Holm og Benjamini-Hochberg korreksjoner for å kontrollere for falske positive, og diskuter resultatene.