Estimation of Regression Coefficients

Computational Statistics

November 13, 2023

```r
library(tidyverse)
```

Galton's family heights data

This problem set is adapted from the book Introduction to Data Science. Load the `GaltonFamilies` dataset from the `HistData` package.

```r
library(HistData)
data("GaltonFamilies")
```

Create a subset called `heights_son`, which contains pairs of son's and father's heights, where for every family a son is randomly selected.

```r
heights_son <- GaltonFamilies |>
    filter(gender == "male") |>
    group_by(family) |>
    sample_n(1) |>
    ungroup() |>
    select(father, childHeight) |>
    rename(son = childHeight)
```

1a. Suppose one of the pairs in `heights_son` is selected at random. Without knowing anything about the father, what is your prediction of the son's height?

Solution: At this point in the question, knowing or not knowing the father's height has no impact on the son's height. Therefore without knowing the height of the father, the best prediction for the son's height can be approximated by defining it's value to be near the mean value. This can be justified by selecting a random pair of heights and compare the son's height from the selected pair with the mean height of the sons.

```r
#calculating mean height of the sons for comparison
mean(heights_son$son)
```

```
## [1] 69.27709
```

```r
#Picking a random sample from father-son height pair
son_height<-sample(heights_son$son, size=1)
#son_height<-heights_son$son[fat_height]
print(son_height)
```

```
## [1] 68
```

From the results it can be said that, the predicted height of the son i.e mean height is near to the randomly selected height.

1b. Given the height of a father (e.g., `x` inches), we could make a better prediction by taking a conditional mean, that is, an average of the heights of all sons whose fathers are `x` inches tall. Suppose a father is 70 inches tall, predict the son's height with conditional mean. Round the father heights in `heights_son` to the nearest inch (e.g., 70.4 is rounded to 70) when calculating the conditional mean.

Solution:

```r
# Round father's height to the nearest inch
heights_son <- heights_son |> mutate(round_father_height = round(father, 0))

# Calculate the conditional mean
predicted_son_height <-
  heights_son |>
  filter(round_father_height == 70) |>
  summarise(mean_son_height = mean(heights_son$son))
```

Given the height of the father is 70 inches, the predicted height of the son rounded to 2 decimals is **69.28**.

1c. Predict the height of a son whose father is 70 inches tall with linear regression. Use the `lm()` and `predict()` functions to answer the question.

Solution: An alternate approach to predict the son's height given the father is 70 inches tall is to fit a linear model and make prediction based on the model data.

```r
# Specify the father height for prediction
given_father_height <- 70

# Create a simple linear model
model <- lm(son ~ round_father_height, data = heights_son)

# Create a data frame for prediction
new_data <- data.frame(round_father_height = given_father_height)

# Use the predict function to get the predicted son's height
predicted_height <- predict(model, newdata = new_data)
```

Predicted height of the son whose father is 70 inches tall using the linear model fitting approach is **69.69**.

1d. Investigate the stability of the estimates with conditional mean and linear regression. Run two sets of Monte Carlo simulations with $B = 1000$. In every iteration, randomly select $N = 50$ pairs from `heights_son` (without replacement). Predict the height of a son whose father is 70 inches tall with conditional mean in one Monte Carlo simulation, and with linear regression in the other. Save the results with conditional mean in a vector called `conditional_avg`, and the linear regression results in `lm_prediction`. Calculate the mean and standard deviation for both `conditional_avg` and `lm_prediction`, and explain the results.

Solution:

```r
B <- 1000
N <- 50

# Initialize empty vectors
conditional_avg_2 <- vector("numeric", length = B)
lm_prediction_2 <- vector("numeric", length = B)

for(i in 1:B){
  sample_data<-heights_son |> sample_n(N, replace=FALSE)
  conditional_mean_2<-mean(sample_data$son[sample_data$father==70], na.rm=TRUE)
  conditional_avg_2[i]<-conditional_mean_2
}

for(i in 1:B){
  sample_data_2<-heights_son |> sample_n(N, replace=FALSE)

  lm_model_sim<-lm(son~father, data=sample_data_2)
  lm_prediction_sim<-predict(lm_model_sim, newdata=data.frame(father=70))
  lm_prediction_2[i]<-lm_prediction_sim
}
# Calculate mean and standard deviation for both conditional_avg and lm_prediction
mean_conditional_avg_2 <- mean(conditional_avg_2)
sd_conditional_avg_2 <- sd(conditional_avg_2)

mean_lm_prediction_2 <- mean(lm_prediction_2)
sd_lm_prediction_2 <- sd(lm_prediction_2)
```

Conditional Mean:

Mean: **69.3879268** Standard Deviation: **0.9438779**

Linear Regression:

Mean: **69.6964615** Standard Deviation: **0.3057654**

Inference:

The results show that the linear regression estimates are more stable than the conditional mean estimates. This is because linear regression takes into account the relationship between the father's height and the son's height, while the conditional mean only considers the average height of sons with fathers of a specific height. Lower the standard deviation, higher is the reliability/stability of the prediction. Therefore, linear regression is highly stable prediction model. As a result, the conditional mean is more susceptible to outliers and fluctuations in the data.

Standard errors of the regression coefficient estimates

Consider the model

$$Y = -2 + 0.5X + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, 0.25)$. Given a vector of predictor values

```
set.seed(1)  # DO NOT CHANGE THE SEED
x_obs <- runif(50)
```

a vector of corresponding response values is simulated based on the model.

```
n <- length(x_obs)
y_obs <- -2 + 0.5 * x_obs + rnorm(n, mean = 0, sd = 0.5)
```

With the observed data (i.e., `x_obs` and `y_obs`), a linear model is fitted and the summary is shown below:

```
fit <- lm(y_obs ~ x_obs)
summary(fit)
```

```
##
## Call:
## lm(formula = y_obs ~ x_obs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95577 -0.28737 -0.07316  0.25517  1.14679
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.9686     0.1456 -13.520   <2e-16 ***
## x_obs         0.5339     0.2439   2.189   0.0335 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.4649 on 48 degrees of freedom
## Multiple R-squared:  0.09074,    Adjusted R-squared:  0.0718
## F-statistic:  4.79 on 1 and 48 DF,  p-value: 0.03352
```

Based on the model fit, the standard errors of the coefficient estimates are $\hat{\beta}_0 = 0.1456072$ and $\hat{\beta}_1 = 0.2439399$

2a. Under the frequentist notion of repeated sampling, run simulations to confirm the estimated standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$.

Solution:

The idea behind simulation is to generate multiple datasets under the assumed model and fit the linear regression model to each dataset. We then examine the distribution of the estimated coefficients across these datasets to compare with the standard errors obtained from the original fit.

```r
# Number of simulations
B <- 1000

# Initialize vectors to store simulated coefficients
beta0_sim <- numeric(B)
beta1_sim <- numeric(B)

# simulations
for (i in 1:B) {
  # Simulate data
  x_sim <- runif(n)
  y_sim <- -2 + 0.5 * x_sim + rnorm(n, mean = 0, sd = 0.5)

  # Fit linear regression model to simulated data
  new_fit_sim <- lm(y_sim ~ x_sim)

  # Store simulated coefficients
  beta0_sim[i] <- coef(new_fit_sim)[1]
  beta1_sim[i] <- coef(new_fit_sim)[2]
}

# Calculating the standard deviations of simulated coefficients
se_beta0_sim <- sd(beta0_sim)
se_beta1_sim <- sd(beta1_sim)

## Calculating the mean of simulated coefficients
me_beta0_sim<-mean(beta0_sim)
me_beta1_sim<-mean(beta1_sim)
```

Simulated Coefficients:

Beta0: **-1.9948413** Beta1: **0.4875202**

Simulated Standard Errors:

Beta0: **0.1442082** Beta1: **0.2527305**

Inference: The simulated coefficients are close to the true values of beta0 (-1.9686419) and beta1 (0.5338946), and the simulated standard errors are close to the estimated standard errors from the original model fit. This confirms that the estimated standard errors are reasonable/reliable.

2b. Use the bootstrap method to estimate the standard errors of $\hat{\beta}_0$ and $\hat{\beta}_1$.

Solution:

The bootstrap method involves resampling the original data with replacement to create a new dataset, which is then used to fit a new linear regression model. This process is repeated multiple times to generate a distribution of bootstrap coefficients. The standard error of the bootstrap coefficients is then estimated by taking the standard deviation of this distribution.

```r
# Define the data
bootstrap_beta0 <- vector("numeric", length = B)
bootstrap_beta1 <- vector("numeric", length = B)
bootstrap_se0 <- vector("numeric", length = B)
bootstrap_se1 <- vector("numeric", length = B)

# Run bootstrap replications
for (i in 1:B) {
  # Sample with replacement from the original data
  x_boot_sb <- sample(x_obs, replace = TRUE)
  y_boot_sb <- sample(y_obs, replace = TRUE)

  # Fit linear regression model to bootstrapped data
  bootstrap_lm<- lm(y_boot_sb ~ x_boot_sb)

  # Extract bootstrapped coefficients
  bootstrap_beta0[i] <- coef(bootstrap_lm)[1]
  bootstrap_beta1[i] <- coef(bootstrap_lm)[2]
}

# Calculate bootstrap standard errors
bootstrap_se0 <- sd(bootstrap_beta0)
bootstrap_se1 <- sd(bootstrap_beta1)
```

Bootstrap Standard Errors:

Beta0 : **0.1532811**

Beta1 : **0.2512792**