Practice9S24

BilalTariq

2024-04-21

Reminder: Practice assignments may be completed working with other individuals.

Reading

The associated reading for the material on the Practice is Chapter 7 on Iteration, Chapter 13 on Simulation, and Chapter 15 on SQL.

This is our final practice assignment!

Practicing Academic Integrity

If you worked with others or used resources outside of provided course material (anything besides our textbook, course materials in the repo, labs, R help menu) to complete this assignment, please acknowledge them below using a bulleted list.

I acknowledge the following individuals with whom I worked on this assignment:

Name(s) and corresponding problem(s)

- Andrew Palena (Problem 2 and 3)

I used the following sources to help complete this assignment:

Source(s) and corresponding problem(s)

-

1 - Iteration

The code below performs an operation that can be run with much more efficient code. Provide the more efficient code, and explain what makes it more efficient. Do not overthink this - the problem is designed to just emphasize one nice feature of R.

```
# Original Code
x <- 1:10

y <- rep(0, 10)
for(i in 1:10){
  y[i]= x[i]^2
}
y
```

```
 [1]   1   4   9  16  25  36  49  64  81 100
```

Solution:

```
# More efficient code
x <- 1:10 # you'll still want this part

y <- x^2

y
```

```
 [1]   1   4   9  16  25  36  49  64  81 100
```

2 - Simulation - Based on MDSR Exercise 13.8

What is the impact of the violation of the constant variance assumption for linear regression models? To investigate, we will repeatedly generate data from two "true" models:

(1) where the constant variance assumption is met: $y_i \sim N(\mu_i, \sigma)$, and
(2) where the constant variance assumption is violated: $y_i \sim N(\mu_i, \sigma_i)$,

where $\mu_i = -1 + 0.5 * X_{1i} + 1.5 * X_{2i}$, σ=1 in (1), $\sigma_i = 1 + X_{2i}$ in (2), and where X_1 is a binary predictor (meaning it takes the values of 0 and 1) and X_2 is Uniform(0,5).

Code to get you started with the simulation, including fitting the models, is given below. It contains NO iterations yet, but tries to help define useful values and show you how to generate the data. (Note that in (2) the standard deviation is dependent upon X_2's value, which is random; i.e., thus the constant variance assumption is violated. This means that the Y's are *not* generated from a distribution with the same variance in (2).)

For each simulation/underlying model, fit the linear regression model and display the distribution of 1,000 estimates of the β_1 parameter, the slope of X_1. Then, write a paragraph addressing the following questions.

- Does the distribution of the β_1 parameter estimates follow a normal distribution in both cases?
- Is it centered around β_1 in both cases?
- How does the variability in the distributions compare (variance in $\hat{\beta}_1$ when the constant variance assumption is met vs. when it is violated)?

Solution:

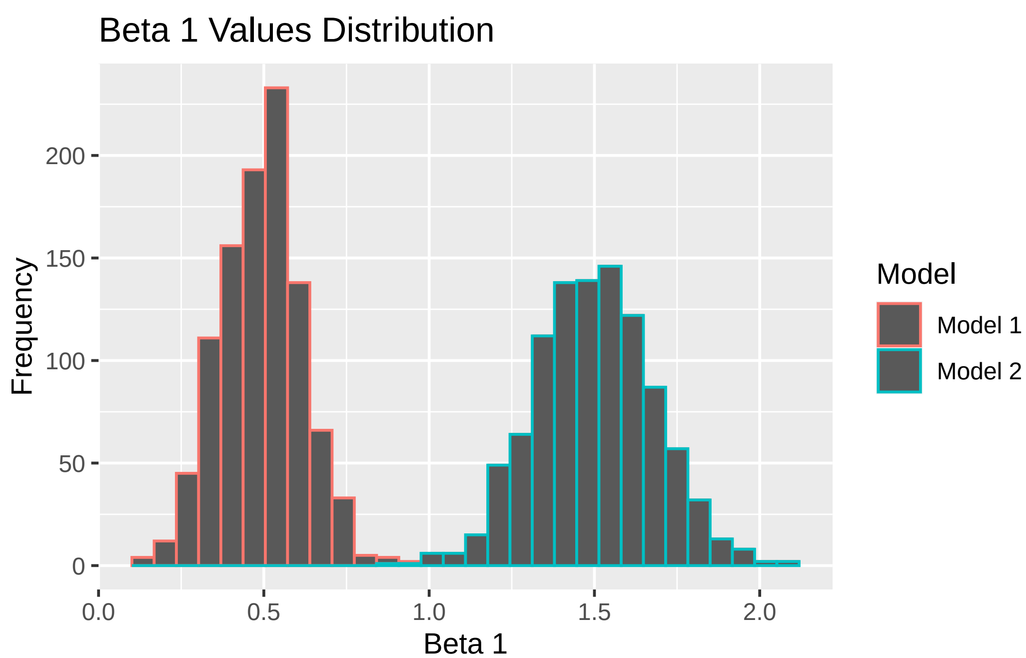
```
# Goal: repeatedly generate data, fit the model,
# and extract the beta1 coefficient (1,000 times)
# for both models (1) and (2)


# set seed for reproducibility
set.seed(231)

# number of simulations
n_sim <- 1000

#for iteration
x <- 1:n_sim

#
```

```r
beta_model1 <- 1:n_sim
beta_model2 <- 1:n_sim

# number of observations in each sample
n_obs <- 250

# set needed values for data generation
rmse <- 1
x1 <- rep(c(0,1), each=n_obs/2)
x2 <- runif(n_obs, min=0, max=5)
beta0 <- -1
beta1 <- 0.5
beta2 <- 1.5

# Generate data

for(i in 1:1000){
  # for model 1, where constant var assumption is met (sd is constant value, rmse)
  y1 <- beta0 + beta1*x1 + beta2*x2 + rnorm(n=n_obs, mean=0, sd=rmse)
  # for model 2, where constant var assumption is violated (sd depends on x2)
  y2 <- beta0 + beta1*x1 + beta2*x2 + rnorm(n=n_obs, mean=0, sd=rmse + x2)

  # Fit the linear regression model
  # for model 1
  mod1 <- lm(y1 ~ x1 + x2)
  # for model 2
  mod2 <- lm(y2 ~ x1 + x2)

  # Example to get beta_1 estimate from one model
  beta_model1[i] <- summary(mod1)$coeff["x1","Estimate"]
  beta_model2[i] <- summary(mod2)$coeff["x2", "Estimate"]
}
```

```r
# target visualization: sampling distribution of \hat{beta}_1
#                  (histogram or density plot of \beta_1 estimates), by model

#Creating dataframes so we can use them in ggplot

model1_dataframe <- data.frame(Beta1 = beta_model1)
model2_dataframe <- data.frame(Beta1 = beta_model2)
```

```r
# target summary numbers: mean and sd/variance of beta_1 estimates, by model\

model1_summaryCalculation <- c(mean(beta_model1), sd(beta_model1))
model2_summaryCalculation <- c(mean(beta_model2), sd(beta_model2))


# create target visualization

model1_df <- data.frame(Beta1 = beta_model1, Model = "Model 1")
model2_df <- data.frame(Beta1 = beta_model2, Model = "Model 2")
df_modelsVIS <- rbind(model1_df, model2_df)

ggplot(df_modelsVIS, aes(x=Beta1, color=Model)) +
  geom_histogram() +
  labs(title="Beta 1 Values Distribution", x="Beta 1", y="Frequency")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
# create target summaries
```

```r
model1_summarydf <- data.frame(
  Model = "Model 1",
  Mean = model1_summaryCalculation[1],
  SD = model1_summaryCalculation[2]
)

model2_summarydf <- data.frame(
  Model = "Model 2",
  Mean = model2_summaryCalculation[1],
  SD = model2_summaryCalculation[2]
)

summary_df <- rbind(model1_summarydf, model2_summarydf)

print(summary_df)
```

```
    Model      Mean        SD
1 Model 1 0.4910872 0.1228440
2 Model 2 1.4984557 0.1798217
```

3 - SQL with Airline Flights

```
# dbConnect_scidb is accessible from the mdsr package
aircon <- dbConnect_scidb("airlines")

# remember can use SHOW and EXPLAIN commands to explore what tables are available
# through this connection, and what variables/fields are in each table
dbGetQuery(aircon, "SHOW TABLES")
```

```
  Tables_in_airlines
1           airports
2           carriers
3            flights
4    flights_summary
5             planes
```

```
#dbGetQuery(aircon, "EXPLAIN airports")
# can view first few obs of a table to see what the fields look like
dbGetQuery(aircon, "SELECT *
                    FROM flights
                    LIMIT 0,5")
```

```
  year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
1 2013    10   1        2             10        -8      453            505
2 2013    10   1        4           2359         5      730            729
3 2013    10   1       11             15        -4      528            530
4 2013    10   1       14           2355        19      544            540
5 2013    10   1       16             17        -1      515            525
  arr_delay carrier tailnum flight origin dest air_time distance cancelled
1       -12      AA  N201AA   2400    LAX  DFW      149     1235         0
2         1      FL  N344AT    710    SFO  ATL      247     2139         0
3        -2      AA  N3KMAA   1052    SFO  DFW      182     1464         0
4         4      AA  N3ENAA   2392    SEA  ORD      191     1721         0
5       -10      UA  N38473   1614    LAX  IAH      157     1379         0
  diverted hour minute           time_hour
1        0    0     10 2013-10-01 00:10:00
2        0   23     59 2013-10-01 23:59:00
3        0    0     15 2013-10-01 00:15:00
4        0   23     55 2013-10-01 23:55:00
5        0    0     17 2013-10-01 00:17:00
```

spart a - Identify what years of data are available in the `flights` table of the airlines database using SQL code. (You can use R code to check it, if you wish).

Optional: you can also count the number of flights per year, as this will show the years available, and perhaps give you a different way to think about getting the desired information.

Solution: The years available are 2013, 2014 and 2015

```
SELECT year
FROM flights
GROUP BY year
```

Table 1: 3 records

year
2013
2014
2015

part b - How many domestic flights flew into the John F. Kennedy International Airport (JFK) on January 21, 2014? Use SQL to compute this number. (You can use R code to check it, if you wish.)

Solution: There were 241 domestic flights that flew into JFK that day.

```
SELECT COUNT(*) as N
FROM flights
WHERE year = 2014 AND day = 21 AND month = 1 AND dest = 'JFK'
```

Table 2: 1 records

N
241

part c - Among the flights that flew into the John F. Kennedy International Airport (JFK) on January 21, 2014, compute (using SQL) the number of flights and the average arrival delay time for each airline carrier. Among these flights, how many carriers had an average arrival delay of 15 minutes or longer? (Again, you can use R code to check it, if you wish.)

Solution: 5 carriers had an average arrival delay of 15 minutes or longer

```
SELECT carrier, COUNT(*) as flights, AVG(arr_delay) as avg_delay
FROM flights
WHERE year = 2014 AND month = 1 AND day = 21 AND dest = 'JFK'
GROUP BY carrier
HAVING AVG(arr_delay) >= 15;
```

Table 3: 5 records

carrier	flights	avg_delay
AA	37	15.1892
DL	47	77.7660
HA	1	106.0000
UA	14	61.0714
VX	10	15.7000

(If you are curious, you could investigate why this was a problematic day to fly into JFK (or anywhere in the Northeast USA, really.))

4 - A data science inspired haiku

Question and examples borrowed from Prof. Horton.

Haiku is one of the most important forms of traditional Japanese poetry. Haiku is today, a 17-syllable verse form consisting of three metrical units of 5, 7 and 5 syllables, respectively. Some examples:

Haiku Example 1

Freeway overpass–

Blossoms in graffiti on

fog-wrapped June mornings

Haiku Example 2

Gravity is lost

Floating out of captain's chair

Bang head on ceiling

Your turn

The applications of haiku to data science have, as yet, not been fully exploited. Your task is to write a haiku poem inspired by the material in the course.

SOLUTION:

Illumination

A flock of birds together

United, a tale