Practice9

Sebastian Montesinos

Due by midnight, Friday, April 29

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)

-

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.

```
# 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: Many functions are r are vectorized, meaning they automatically run themselves across all values inside of a vector. Thus, using a for loop is inefficient since you are iterating over the same vector over and over again. This code below is more efficient since it takes advantage of the fact that the function to square values is already vectorized.

```
# 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.

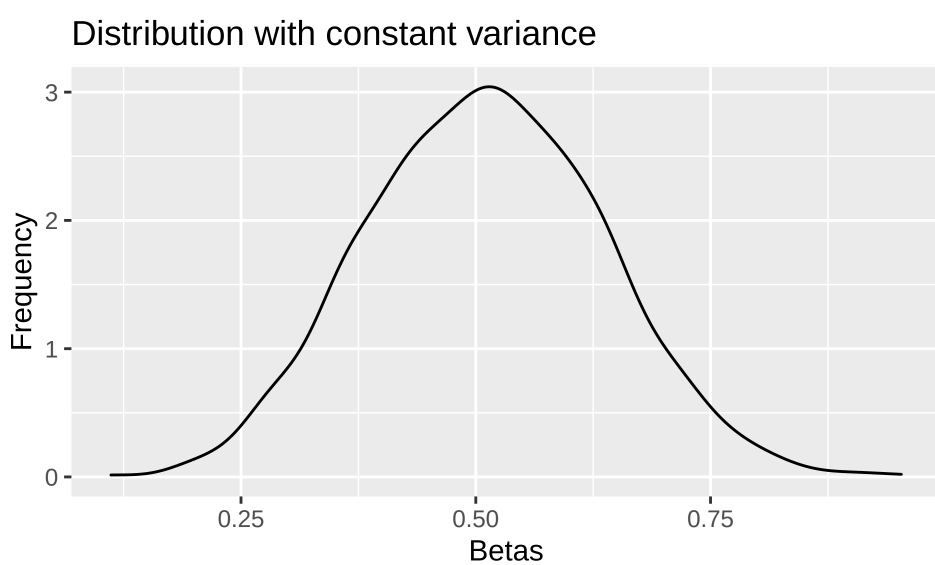
- Does the distribution of the $beta_1$ parameter estimates follow a normal distribution in both cases?
- Is it centered around β_1 in both cases?
- How does the variability in the distributions compare (variance in $\hat{\beta}_1$ when the constant variance assumption is met vs. when it is violated)?

Solution: The distribution of the beta_1 parameter does follow a normal distribution in both cases, and this normal distribution is centered around beta_1. However, the variability differs: when you violate the constant variance assumption, the variability goes way up, resulting in a greater standard deviation. As long as constant variance assumption is not violated, the variability remains regular.

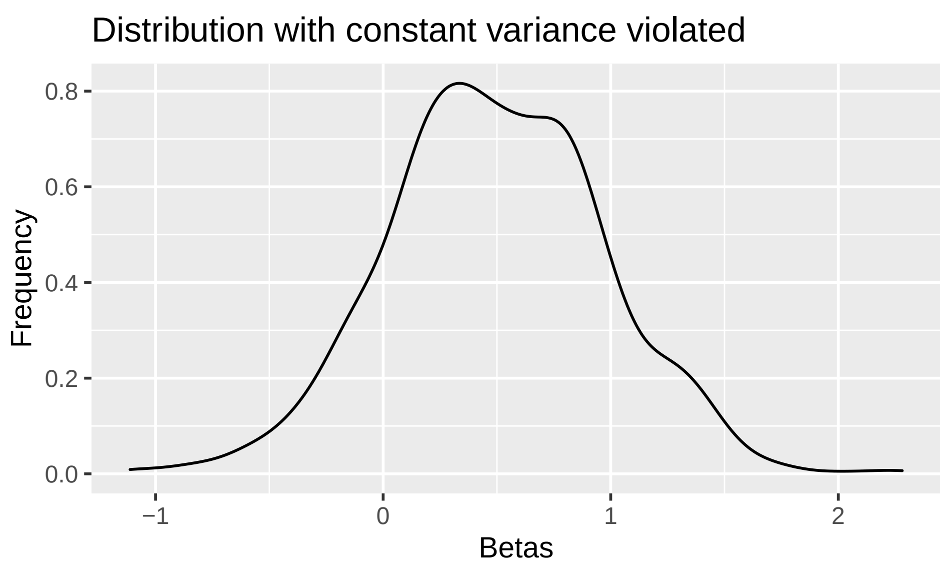
```
# Goal: repeatedly generate data, fit the model,
# and extract the beta1 coefficient (1,000 times)
# for both models (1) and (2)

# set seed for reproducibility
set.seed(231)

# number of simulations
n_sim <- 1000

# number of observations in each sample
n_obs <- 250

betas_overall <- rep(NA, n_sim)
betas_overall2 <- rep(NA, n_sim)

for(i in 1:n_sim){
# 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
```

```r
# Generate data
# 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
beta1 <- summary(mod1)$coeff["x1","Estimate"]
beta2 <- summary(mod2)$coeff["x1","Estimate"]

betas_overall[i] <- beta1
betas_overall2[i] <- beta2

}
```

```r
# target visualization: sampling distribution of \hat{beta}_1
#                  (histogram or density plot of \beta_1 estimates), by model
# target summary numbers: mean and sd/variance of beta_1 estimates, by model

#Plot for beta1
ggplot(data = data.frame(betas_overall),
       aes(x = betas_overall)) +
  geom_density() +
  labs(title = "Distribution with constant variance",
       x = "Betas",
       y = "Frequency")
```



```r
#Plot for beta2
ggplot(data = data.frame(betas_overall2),
       aes(x = betas_overall2)) +
      labs(title = "Distribution with constant variance violated",
           x = "Betas",
           y = "Frequency") +
  geom_density()
```




```r
# create target summaries
mosaic::fav_stats(betas_overall)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2

##        min        Q1    median        Q3       max      mean        sd    n
##  0.1114762 0.4220203 0.5122121 0.5981999 0.9532145 0.5106249 0.1253904 1000
##  missing
##        0
```

```r
mosaic::fav_stats(betas_overall2)
```

```
##       min        Q1    median        Q3      max      mean        sd    n
##  -1.11203 0.1689546 0.4826652 0.8134597 2.281027 0.4940002 0.4763352 1000
##  missing
##        0
```

3 - SQL with Airline Flights

```r
# 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             planes
```

```r
dbGetQuery(aircon, "EXPLAIN airports")
```

```
##     Field          Type Null Key Default Extra
## 1     faa    varchar(3)   NO PRI
## 2    name  varchar(255)  YES        <NA>
## 3     lat decimal(10,7)  YES        <NA>
## 4     lon decimal(10,7)  YES        <NA>
## 5     alt       int(11)  YES        <NA>
## 6      tz   smallint(4)  YES        <NA>
## 7     dst       char(1)  YES        <NA>
## 8    city  varchar(255)  YES        <NA>
## 9 country  varchar(255)  YES        <NA>
```

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

```
## Warning in .local(conn, statement, ...): Decimal MySQL column 2 imported as
## numeric

## Warning in .local(conn, statement, ...): Decimal MySQL column 3 imported as
## numeric

##   faa                           name      lat       lon  alt tz dst
## 1 04G             Lansdowne Airport 41.13047 -80.61958 1044 -5   A
## 2 06A Moton Field Municipal Airport 32.46057 -85.68003  264 -6   A
## 3 06C           Schaumburg Regional 41.98934 -88.10124  801 -6   A
## 4 06N               Randall Airport 41.43191 -74.39156  523 -5   A
## 5 09J         Jekyll Island Airport 31.07447 -81.42778   11 -5   A
##            city       country
## 1    Youngstown United States
## 2      Tuskegee United States
## 3    Schaumburg United States
## 4    Middletown United States
## 5 Jekyll Island United States
```

part 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.

Note: This is a different version of the data used in the prep. The years are different! Be sure you are using the correct connection.

Solution: 2010 to 2017.

```sql
SELECT count(*) as N, year FROM flights GROUP BY year
```

Table 1: 8 records

N	year
6450117	2010
6085281	2011
6096762	2012
6369482	2013
5819811	2014
5819079	2015
5617658	2016
5674621	2017

part b - How many domestic flights flew into Dallas-Fort Worth (DFW) on May 14, 2010? Use SQL to compute this number. (You can use R code to check it, if you wish.)

Solution: 754 flights.

```sql
SELECT COUNT(*) as N  FROM flights WHERE day = 14 AND year = 2010 AND month = 5 AND dest = 'DFW'
```

Table 2: 1 records

N
754

part c - Among the flights that flew into Dallas-Fort Worth (DFW) on May 14, 2010, 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 60 minutes or longer? (Again, you can use R code to check it, if you wish.)

Solution: 5 carriers had an average arrival delay of 60 minutes or longer.

```sql
SELECT carrier,
COUNT(*) as n,
AVG(arr_delay)
as avg_arr_delay
FROM flights WHERE day = 14 AND year = 2010 AND month = 5 AND dest = 'DFW'
GROUP BY carrier
HAVING avg_arr_delay > 60
```

Table 3: 5 records

carrier	n	avg_arr_delay
9E	1	82.0000
CO	8	125.8750
F9	4	62.0000
UA	9	91.2222
YV	1	67.0000

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:

Freeway overpass–

Blossoms in graffiti on

fog-wrapped June mornings

Gravity is lost

Floating out of captain's chair

Bang head on ceiling

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:

Displaying data

Is hard - pick graphics with care

Maintain clarity