Lab_04R

36-290 – Statistical Research Methodology

Week 4 Thursday – Fall 2021

Preliminaries

Goal

The goal of this lab is to explore the concepts of mean-squared error and the bias-variance tradeoff with the simplest model imaginable: a constant-amplitude model from which data are sampled.

Questions

Question 1

In order to perform data splitting in `R`, you need to know how to sample integers. The integers would, here, represent rows of your data frame. In a code chunk below, use the functions `sample()` and `sample.int()` to sample three numbers from the integer sequence 1 to 10, without replacement. Then use `sample()` to sample ten numbers from the integer sequence 10 to 14, with replacement.

```
set.seed(101)
sample(1:10, 3, replace=FALSE, prob=NULL)
```

```
## [1]  9 10  6
```

```
sample.int(10, size=3, replace=FALSE, prob=NULL)
```

```
## [1] 7 1 2
```

```
sample(10:14, 10, replace=TRUE, prob=NULL)
```

```
##  [1] 14 13 12 12 12 10 12 12 11 13
```

```
#sample(x:10, 3, replace = FALSE, prob = NULL)
#sample.int(10, size = 3, replace = FALSE, prob = NULL,
           #useHash = (!replace && is.null(prob) && size <= 2="" n="" &&="">  1e7))
```

Question 2

In order to help ensure that your research is reproducible, you should always set the seed for the random number generator when there is randomness involved in your analysis. Here, the randomness would be what rows of your data frame are selected to be in the training set. In a code chunk below, show that setting the seed "works" by setting it, then sampling five numbers from the range 1 to 10, then setting the seed again to the same value as before, then sampling five numbers again. You should get the same five numbers!

```
set.seed(110)
sample(1:10, 5, replace=FALSE, prob=NULL)
```

```
## [1] 4 8 6 3 1
```

```
set.seed(110)
sample(1:10, 5, replace=FALSE, prob=NULL)
```

```
## [1] 4 8 6 3 1
```

Question 3

When you sample with replacement, approximately what proportion of the population gets selected one or more times? To answer this, use a `for` loop. For instance, define a vector called `prop`, initializing it to have value `NA` repeated 1000 times. (Look at old notes if you don't remember how to do this.) Then define a `for` loop, which will look something like this:

```
for ( ii in 1:1000 ) {
  # DO STUFF
}
```

```
set.seed(101)
prop <- rep(NA, 1000)

for ( ii in 1:1000 ) {
  x <- sample(1:100, 100, replace=TRUE, prob=NULL)
  prop[ii] =length(unique(x))
}

mean(x)
```

```
## [1] 51.19
```

Where I say "DO STUFF," what I mean is: sample 100 numbers from the integer vector 1 to 100 with replacement, determine how many unique values there are, and save that number to `prop[ii]`. When the `for` loop is done, we will have 1000 values; the mean of those values is your estimate of the proportion of the population (actually, 100 times the proportion of the population) that gets selected one or more times.

You can actually compute this number directly, exactly, using methods you learned *this week* in 225. Ask me if you are interested.

By the way…the reason you are doing this is that selecting rows of a data frame with replacement underlies the so-called "bootstrapping" algorithm for estimating uncertainties. Bootstrap a dataset, run your analysis, set aside a number of interest that comes out of your analysis, bootstrap another, etc., until you build up a distribution for that number. The width of the distribution is an estimate of your uncertainty about the number.

Question 4

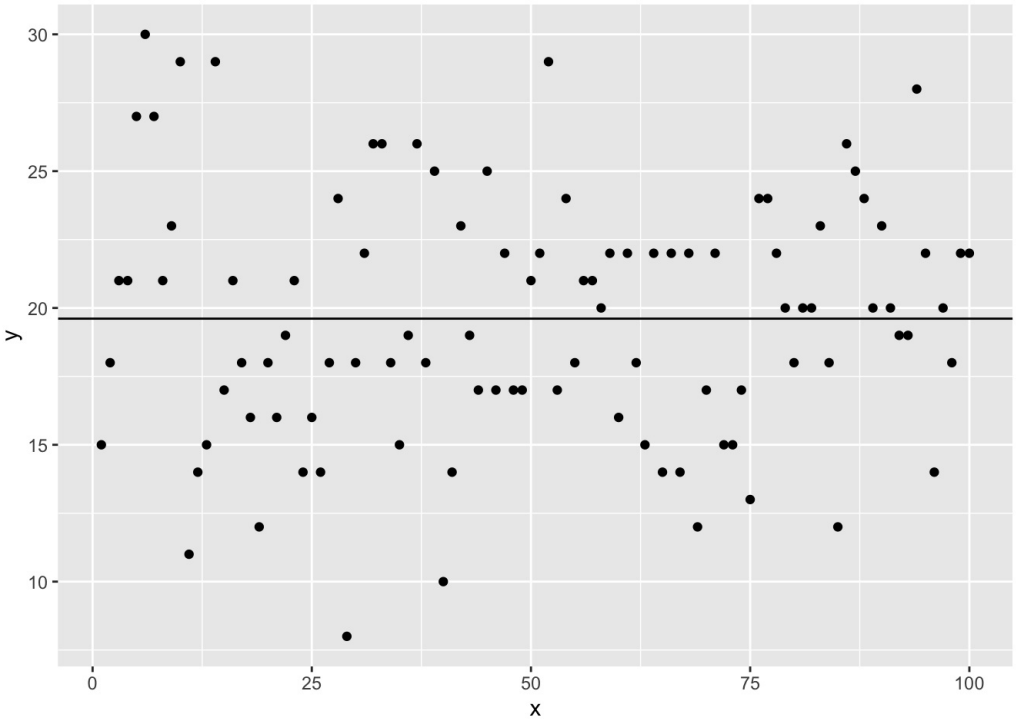
Set a random number seed, and sample 100 data from a Poisson distribution with parameter $\lambda = 20$. (See `rpois()`.) Plot the data in `ggplot` by first making a data frame with the first column simply being the row number and the second column being your data. Add a horizontal line showing the mean of the data.

```
set.seed(46)
library(ggplot2)

x <- 1:100
y <- rpois(100, lambda=20)

df_new_p <- data.frame("Row number" = x, "Data" =y)
df_new_p
```

```
##     Row.number Data
## 1            1   15
## 2            2   18
## 3            3   21
## 4            4   21
## 5            5   27
## 6            6   30
## 7            7   27
## 8            8   21
## 9            9   23
## 10          10   29
## 11          11   11
## 12          12   14
## 13          13   15
## 14          14   29
## 15          15   17
## 16          16   21
## 17          17   18
## 18          18   16
## 19          19   12
## 20          20   18
## 21          21   16
## 22          22   19
## 23          23   21
## 24          24   14
## 25          25   16
## 26          26   14
## 27          27   18
## 28          28   24
```

```
## 29          29    8
## 30          30   18
## 31          31   22
## 32          32   26
## 33          33   26
## 34          34   18
## 35          35   15
## 36          36   19
## 37          37   26
## 38          38   18
## 39          39   25
## 40          40   10
## 41          41   14
## 42          42   23
## 43          43   19
## 44          44   17
## 45          45   25
## 46          46   17
## 47          47   22
## 48          48   17
## 49          49   17
## 50          50   21
## 51          51   22
## 52          52   29
## 53          53   17
## 54          54   24
## 55          55   18
## 56          56   21
## 57          57   21
## 58          58   20
## 59          59   22
## 60          60   16
## 61          61   22
## 62          62   18
## 63          63   15
## 64          64   22
## 65          65   14
## 66          66   22
## 67          67   14
## 68          68   22
## 69          69   12
## 70          70   17
## 71          71   22
## 72          72   15
## 73          73   15
## 74          74   17
## 75          75   13
## 76          76   24
## 77          77   24
## 78          78   22
## 79          79   20
## 80          80   18
## 81          81   20
## 82          82   20
## 83          83   23
## 84          84   18
## 85          85   12
## 86          86   26
## 87          87   25
## 88          88   24
## 89          89   20
## 90          90   23
## 91          91   20
## 92          92   19
## 93          93   19
## 94          94   28
## 95          95   22
## 96          96   14
## 97          97   20
## 98          98   18
## 99          99   22
## 100        100   22
```

```
yint <- mean(y)

ggplot(data=df_new_p, aes(x,y)) + geom_point() + geom_hline(yintercept = yint)
```



MSE simulator

The following is an example of a MSE simulator (assuming 100 Poisson-distributed data with $\lambda = 20$). What the simulator is doing is the following: it generates a dataset, then randomly splits the sample in two in 100,000 different ways. For each random split, we train a model using half the data and test it using the other half. We then collect all 100,000 values of the test-set MSE; the point is to show you that the test-set MSE can vary widely in value depending on just which of the 100 data are used for training versus for testing.

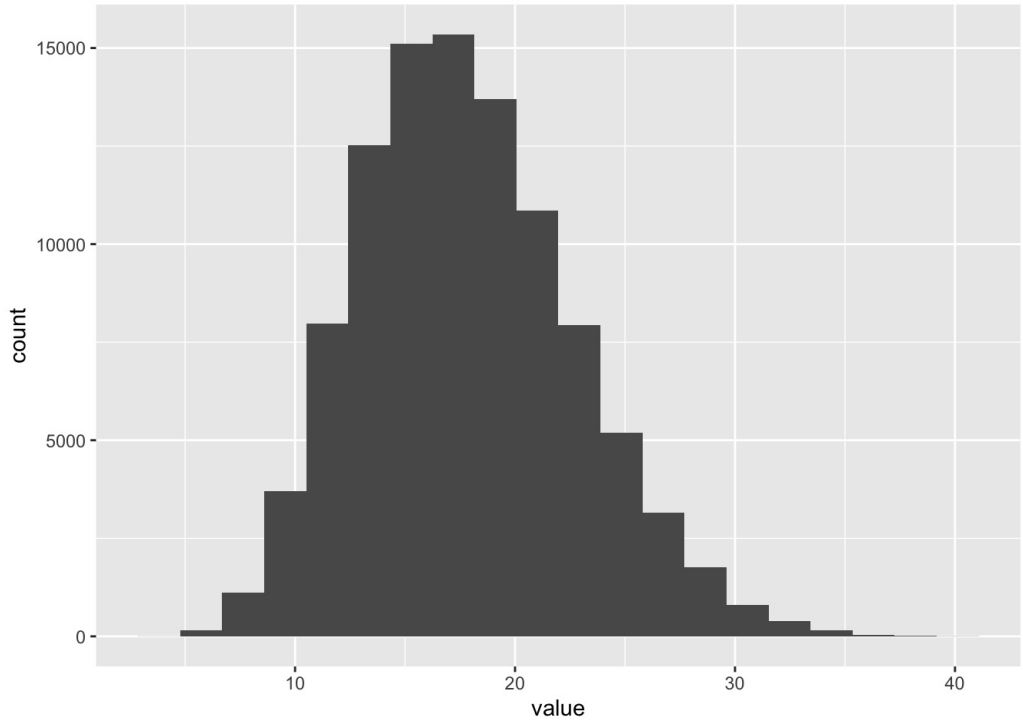
```
num.sim  = 100000  # number of simulations (ad hoc selection)
fraction = 0.5     # fraction of observations in the training set (also ad hoc selection)

set.seed(101)
data = rpois(100,lambda=20)
test.mse = rep(NA,num.sim)             # set aside array to hold the test-set MSEs
for ( ii in 1:num.sim ) {
  s = sample(100,round(fraction*100))  # assumes n = 100 observed data
  lambda.hat = mean(data[s])           # MLE for the Poisson lambda for the training data
  test.mse[ii] = mean((data[-s]-lambda.hat)^2)  # the mean of the squared error, test data
}
```

You would characterize `test.mse` using, e.g., a histogram, the `summary()` function, the `mean()`, `var()`, and `sd()` functions, etc. (The test-set MSE is a statistic, and is thus a random variable, and is thus sampled from a pdf when some mean and some variance.)

Question 5

Using the code provided above (and altering it as necessary), pick a fraction of data that is to be used for training, and characterize the distribution of test-set MSE values by using the `summary()` function and by plotting a histogram. What is, e.g., the interquartile range? Is it large compared to the mean, or small? Note that the irreducible error is, on average, 20, which is the variable of a Poisson distribution with $\lambda = 20$. So you might expect the mean of the test-set MSE distribution to be 20 or larger. However, it may not be, because the mean is a random variable that changes as you sample new datasets. (Not new splits of a given dataset, but new datasets, period. In other words, if you change the seed, you'll get a new dataset, and a new test-set MSE distribution with a new mean.)

```
num.sim  = 100000  # number of simulations (ad hoc selection)
fraction = 0.25     # fraction of observations in the training set (also ad hoc selection)

set.seed(101)
data = rpois(100,lambda=20)
test.mse = rep(NA,num.sim)             # set aside array to hold the test-set MSEs
for ( ii in 1:num.sim ) {
  s = sample(100,round(fraction*100))  # assumes n = 100 observed data
  lambda.hat = mean(data[-s])           # MLE for the Poisson lambda for the training data
  test.mse[ii] = mean((data[s]-lambda.hat)^2)  # the mean of the squared error, test data
}

data_f=data.frame(value=test.mse)
suppressMessages(library(ggplot2))
ggplot(data_f, aes(x=value)) + geom_histogram(bins = 20)
```




```
19.894-15.946
```

```
## [1] 3.948
```

```
The interquartile range is: 19.894-15.946 =3.948


It is relatively small compared to the mean
```

Question 6

Let's say you wanted to compare the test-set MSE that results from linear regression, random forest, and boosting, and let's say for each analysis you will split the data into a single training and a single test set. Given the result you observed in Q4, what is the most important step to take when splitting the data?

```
Setting the seed?
```

Cross-Validation

To carry out cross-validation, you first want to select the number of folds, which we denote k. Once you've done that, you assign all data randomly to a folds. Note that because you are doing a random assignment, it may not be the case that, e.g., exactly one-fifth of the data appears in each fold in 5-fold CV.

Here's a code that assigns data to folds:

```
set.seed(101)
data = rpois(100,lambda=20)

n = length(data)
k = 5
set.seed(102)
fold = sample(k,n,replace=TRUE)
print(fold[1:20])
```

```
##  [1] 1 4 1 2 4 3 3 5 3 5 4 4 4 1 1 2 2 3 4 3
```

Question 7

Combine the codes above, and add some new code, so as to estimate the test-set MSE for 5-fold CV. Just do this once, i.e., you are not simulating this process 100,000 times. Just once. This is straightforward to do if you are an experienced R programmer, but won't necessarily be as easy for you. Try to do it, then come to office hours if you are having issues. Here's a pseudocode:

```
1) set seed and generate dataset [in real life, replace this with input data]
2) determine the length of the dataset [in real life, the number of rows in the data frame]
3) map each datum to a fold [map each data frame row to a fold]
4) use a for-loop to loop over each fold...call the looping index ii
  a) set aside the data of fold ii as the test data
  b) combine the data of all other folds as the training data
  c) fit the model to the training data
  d) generate predictions for each test datum...save these and set them aside
5) use the saved predictions for each datum, and the actual value of each datum, to compute the MSE
Done.
```

```
set.seed(101)
data = rpois(100,lambda=20)
n = length(data)
k = 5

set.seed(103)
pred.test = rep(NA,100)
for ( ii in 1:5) {
  fold = sample(k,n,replace=TRUE)
  test = data[fold==ii]
  train = data[fold!=ii]
  model = mean(train)
  pred.test[fold==ii]
  pred.test[ii] = mean((data[fold]- model)^2)
}
```