

# Problem Set 4

Political Data Science - Spring 2020

Gencer, Alper Sukru

Due May 1, 10:00 AM (Before Class)

## Instructions

1. The following questions should each be answered within an R script. Be sure to provide many comments in the script to facilitate grading. Undocumented code will not be graded.
2. Work on git. Fork the repository found at <https://github.com/domlockett/PDS-PS3> and add your code, committing and pushing frequently. Use meaningful commit messages – these may affect your grade.
3. You may work in teams, but each student should develop their own R script. To be clear, there should be no copy and paste. Each keystroke in the assignment should be your own.
4. If you have any questions regarding the Problem Set, contact the TAs or use their office hours.
5. For students new to programming, this may take a while. Get started.
6. You will need to install ggplot2 and dplyr to complete this dataset.

## Question 1 - Sample Statistics

Load the following data: <http://politicaldatascience.com/PDS/Datasets/GSS-data.csv>.

The variable `poleff11` asks participants to rate their level of agreement with the statement “People like me don’t have any say about what the government does” (see the codebook for more information on all variables in this dataset at: [http://politicaldatascience.com/PDS/Datasets/gss\\_codebook.csv](http://politicaldatascience.com/PDS/Datasets/gss_codebook.csv)).

1. Convert this variable into a numeric where higher values indicate higher levels of political efficacy (1- strongly agrees with the statement; 5- strongly disagrees with the statment) and all other values (‘Cant choose’ etc.) become NA’s.
2. What is the proportion of individuals from the entire sample who feel as though they "have a say in the government?"
3. Using a sample of 25 from this dataset. What is the average proportion who feel as though hey have a say?
4. Pull a random sample of 25 from the `poleff11` data and calculate the mean for this outcome. Now repeat this process 500 times and store these values in a variable called `trials 25`.
5. Now create a variable called `trials 100` where we do 500 trials with  $n = 100$  instead of 25.
6. Draw a histogram of the sampling distribution for the two trials ( $n = 25$  vs.  $n = 100$ ) you just conducted. Give the plots meaningful titles and axis labels. Save these plots in your repository.
7. What notable difference occur when we use a larger sample size in our trials?

## Answer 1 - Sample Statistics

Load the following data: <http://politicaldatascience.com/PDS/Datasets/GSS-data.csv>.

```
rm(list = ls())
gss.data <- read.csv("http://politicaldatascience.com/PDS/Datasets/GSS-data.csv")
gss.data <- gss.data[-c(2349, 2350), ] ## included some unrelated information
```

Convert this variable into a numeric where higher values indicate higher levels of political efficacy (1- strongly agrees with the statement; 5- strongly disagrees with the statement) and all other values ('Cant choose' etc.) become NA's:

```
levels(gss.data$poleff11)

## [1] "" "Agree"
## [3] "Cant choose" "Disagree"
## [5] "Neither agree nor disagree" "No answer"
## [7] "Not applicable" "Strongly agree"
## [9] "Strongly disagree"

gss.data$poleff11.recoded[gss.data$poleff11 == "Strongly agree"] <- 1
gss.data$poleff11.recoded[gss.data$poleff11 == "Agree"] <- 2
gss.data$poleff11.recoded[gss.data$poleff11 == "Neither agree nor disagree"] <- 3
gss.data$poleff11.recoded[gss.data$poleff11 == "Disagree"] <- 4
gss.data$poleff11.recoded[gss.data$poleff11 == "Strongly disagree"] <- 5
gss.data$poleff11.recoded[gss.data$poleff11 == "Cant choose"] <- NA
gss.data$poleff11.recoded[gss.data$poleff11 == "Not applicable"] <- NA
gss.data$poleff11.recoded[gss.data$poleff11 == "No answer"] <- NA
```

What is the proportion of individuals from the entire sample who feel as though they "have a say in the government?"

```
##### For this, I'll be looking at the proportion of people who "disagree (4) \ strongly disagree (5)

#--- including NA values:
prop_have.a.say.na <- sum(table(gss.data$poleff11.recoded)[4:5])/nrow((gss.data))
prop_have.a.say.na

## [1] 0.1937819

#--- excluding NA values:
prop_have.a.say <- sum(table(gss.data$poleff11.recoded)[4:5])/length(na.omit(gss.data$poleff11.recoded))
prop_have.a.say

## [1] 0.3953084
```

Using a sample of 25 from this dataset. What is the average proportion who feel as though they have a say?:

```
set.seed(20200425)
```

```
#--- including NA values:
```

```
sample25_na <- sample(gss.data$poleff11.recoded, size = 25)
sum(sample25_na >= 4, na.rm = T) / 25
```

```
## [1] 0.24
```

```
#--- excluding NA values:
```

```
sample25_no.na <- sample(na.omit(gss.data$poleff11.recoded), size = 25)
sum(sample25_no.na >= 4, na.rm = T) / 25
```

```
## [1] 0.36
```

\subsection{Pull a random sample of 25 from the poleff11 data and calculate the mean for this outcome. Now repeat this process 500 times and store these values in a variable called \textbf{trials\_25}:}

```
##### For convenience, I will be solving by removing NA's.
```

```
#--- Random one sample:
```

```
mean(sample(na.omit(gss.data$poleff11.recoded), size = 25, replace = T))
```

```
## [1] 2.6
```

```
#--- Random 500 samples:
```

```
trials_25 <- NULL
for(i in 1:500){
  trials_25 <- c(trials_25, mean(sample(na.omit(gss.data$poleff11.recoded), size = 25, replace = T)))
}
```

\subsection{Now create a variable called trials\_100 where we do 500 trials with n=100 instead of 25:}

```
##### For convenience, I will be solving by removing NA's.
```

```
#--- Random one sample with n = 100:
```

```
mean(sample(na.omit(gss.data$poleff11.recoded), size = 100, replace = T))
```

```
## [1] 2.92
```

```
#--- Random 500 samples with n = 100:
```

```
trials_100 <- NULL
for(i in 1:500){
  trials_100 <- c(trials_100, mean(sample(na.omit(gss.data$poleff11.recoded), size = 100, replace = T)))
}
```

Draw a histogram of the sampling distribution for the two trials (n=25 vs. n=100) you just conducted. Give the plots meaningful titles and axis labels. Save these plots in your repository.:

```
library(tidyverse)
```

```
##install.packages("ggpubr")
```

```
library(ggpubr)
```

```
df.trials <- as.data.frame(cbind(trials_25, trials_100))
```

```

p_trials_100 <- ggplot(df.trials) +
  geom_histogram(aes(trials_100), bins = 100, fill = "red") +
  xlab("Sample Means") +
  ggtitle("Sampling Dist. with Size 100") +
  ylim(0,50) +
  xlim(2,4) +
  theme_light()

p_trials_25 <- ggplot(df.trials) +
  geom_histogram(aes(trials_25), bins = 100, fill = "blue") +
  xlab("Sample Means") +
  ggtitle("Sampling Dist. with Size 25") +
  ylim(0,50) +
  xlim(2,4) +
  theme_light()

ggarrange(p_trials_100, p_trials_25,
          ncol = 2, nrow = 1)

```

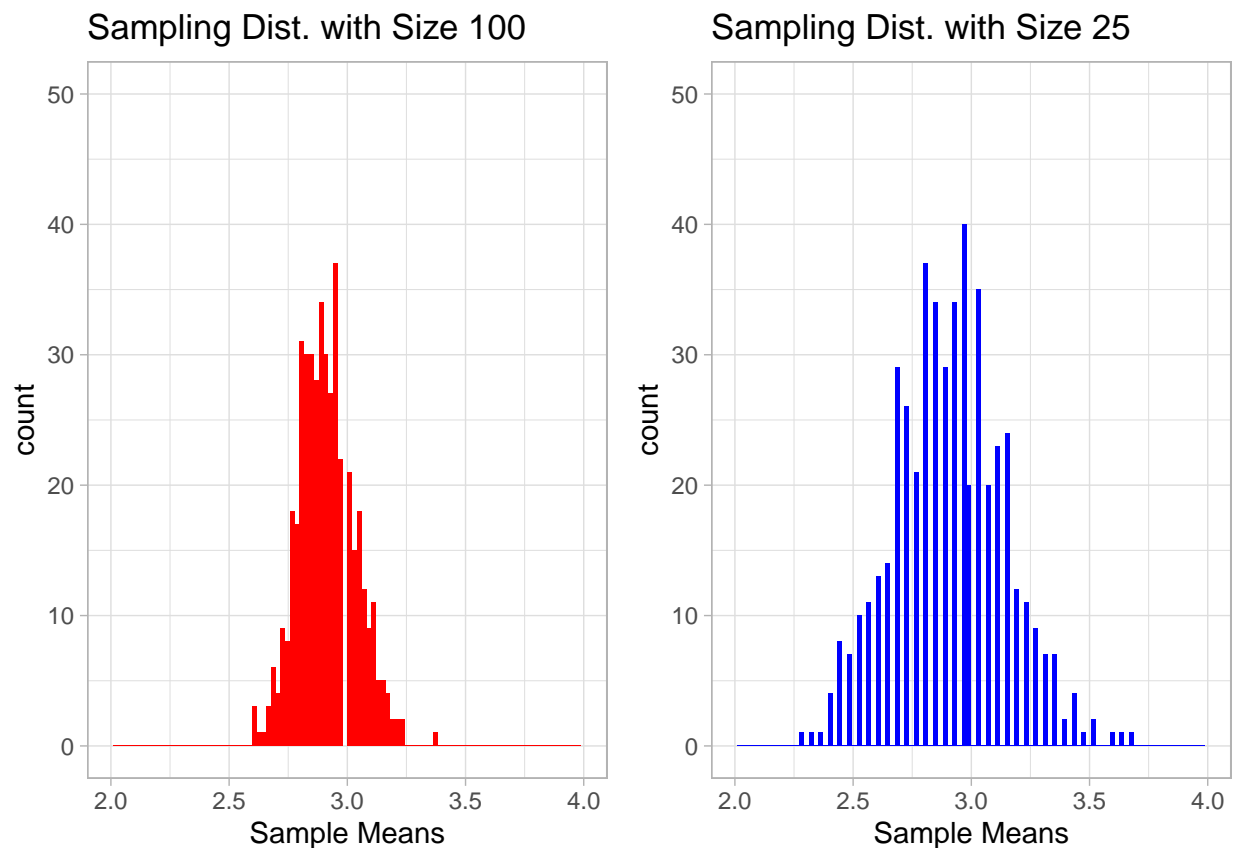


Figure 1: Sampling Distributions with Varying Sample Sizes

## What notable difference occur when we use a larger sample size in our trials?:

As it can be seen in the graph, the spread of the sampling distribution with sample size 100 is narrower than the one with sample size 25.

```
library(knitr)

table <- cbind(mean(na.omit(gss.data$po1eff11.recoded)), mean(trials_100),
               sd(trials_100), mean(trials_25), sd(trials_25))

colnames(table) <- c("True Mean", "Size-100 Mean", "Size-100 SD",
                    "Size-25 Mean", "Size-25 SD")

kable(table)
```

True Mean	Size-100 Mean	Size-100 SD	Size-25 Mean	Size-25 SD
2.908775	2.91692	0.1183373	2.90648	0.2374067

Also see from the table that the mean of size-100 trials is closer to the true mean and at the same time, the sampling distribution of sample mean has less variance (smaller standard error).

## Question 2 - Supervised Learning

Load the following data: <http://politicaldatascience.com/PDS/Datasets/SenateForecast/PollingCandidateData92-16.csv>. This is data for incumbents running for re-election to the US Senate.

```
rm(list = ls())
poll.data <- read.csv("http://politicaldatascience.com/PDS/Datasets/SenateForecast/PollingCandidateData92-16.csv")
```

- Poll Percentage.of.Vote.won.x is the percentage of the vote the candidate won.
- The other variables are mostly self-explanatory or have been used before in class.
- However, this dataset differs in that it is organized at the poll level. That is, there is one row for each poll of each senate race.
- So there are some new variables including: the polling firm, the starting date of the poll, the “days left” until Election Day, sample size, and the numberSupport (the number of respondents in that poll who indicated they supported the incumbent candidate.)
- There is also a win variable that indicates whether the incumbent candidate won the election.

Re-organize the data so it is at the election level (as opposed to the poll level):

- This means you will have to figure out how to reduce the polling data into a summary statistic.
- You might try to do this a couple of different ways based on sample size and date of the poll for use later.

```
poll.data.2 <- poll.data %>%
  group_by(CandidateIdentifier) %>%
  filter(row_number(Percentage.of.Vote.won.x) == 1) %>%
  select(-c("pollster", "poll_period", "daysLeft"))
```

Randomly select 20 percent of your data to use as a “validation sample” to assess the quality of your model. You will use this division of the data in the rest of the problems below:

```
##### 20 percent of data as validation (test) data:

library(rsample)

#--- See the following courses chunk below:
```

Using the Poll Percentage.of.Vote.won.x variable, create at least two linear regression models to predict vote share for incumbents:

- You are free to do this any way you want, but you must assess the quality of your model using cross-validation.
- Train your model on your “training” data (80

- Provide an appropriate summary statistic for your competing models using only the validation set. (Meaning: what is your out-of-sample performance?)

```
##### Linear Baseline Models:
####

rmse0 <- c()      ##### Model 0   Percentage.of.Vote.won.x ~ Democrat
rmse1 <- c()      ##### Model 1   Percentage.of.Vote.won.x ~ Democrat + incumbent
rmse2 <- c()      ##### Model 2   Percentage.of.Vote.won.x ~ Democrat + pvi
rmse3 <- c()      ##### Model 3   Percentage.of.Vote.won.x ~ Democrat + cycle
rmse4 <- c()      ##### Model 4   Percentage.of.Vote.won.x ~ Democrat + weightexperience
rmse5 <- c()      ##### Model 4   Percentage.of.Vote.won.x ~ Democrat + state

####
#### Model 0   Percentage.of.Vote.won.x ~ Democrat
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse0 <- c(rmse0, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
mean(rmse0)

## [1] 12.07364

####
#### Model 1   Percentage.of.Vote.won.x ~ Democrat + Incumbent
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + factor(Incumbent), data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse1 <- c(rmse1, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

####
#### Model 2   Percentage.of.Vote.won.x ~ Democrat + pvi
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + pvi, data = elect_train)
  pred <- predict(model, newdata = elect_test)
```



```

rmse2 <- c(rmse2, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

####
#### Model 3 Percentage.of.Vote.won.x ~ Democrat + cycle
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + factor(cycle), data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse3 <- c(rmse3, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

####
#### Model 4 Percentage.of.Vote.won.x ~ Democrat + weightexperience
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + weightexperience, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse4 <- c(rmse4, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

####
#### Model 5 Percentage.of.Vote.won.x ~ Democrat + state
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + factor(state), data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse5 <- c(rmse5, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

df.1 <- data.frame(cbind(mean(rmse0), mean(rmse1), mean(rmse2), mean(rmse3), mean(rmse4), mean(rmse5)))
colnames(df.1) <- c("Baseline (BL)", "BL + Incumbent", "BL + pvi", "BL + cycle", "BL + weightexperience", "BL + state")
kable(df.1)

```

Baseline (BL)	BL + Incumbent	BL + pvi	BL + cycle	BL + weightexperience	BL + state
12.07364	8.657701	12.08963	12.28019	8.921921	12.93576

See that the models with 1) Incumbent and 2) weightexperience are the models we are to focus on.

```
##### Linear Baseline Models:
####

rmse6 <- c()      ##### Model 6   Percentage.of.Vote.won.x ~ Democrat + poly(Incumbent, 2)
rmse7 <- c()      ##### Model 7   Percentage.of.Vote.won.x ~ Democrat * Incumbent
rmse8 <- c()      ##### Model 8   Percentage.of.Vote.won.x ~ Democrat + poly(weightexperience, 2)
rmse9 <- c()      ##### Model 9   Percentage.of.Vote.won.x ~ Democrat * weightexperience
rmse10<- c()      ##### Model 10  Percentage.of.Vote.won.x ~ Democrat + Incumbent + weightexperience
rmse11<- c()      ##### Model 10  Percentage.of.Vote.won.x ~ Democrat + Incumbent * weightexperience


####
#### Model 6   Percentage.of.Vote.won.x ~ Democrat + poly(Incumbent, 2)
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + poly(Incumbent, 2) , data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse6 <- c(rmse6, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}


####
#### Model 7   Percentage.of.Vote.won.x ~ Democrat * Incumbent
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat * Incumbent, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse7 <- c(rmse7, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}


####
#### ##### Model 8   Percentage.of.Vote.won.x ~ Democrat + poly(weightexperience, 2)
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + poly(weightexperience, 2) , data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse8 <- c(rmse8, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}
```

```

}

####
#### ##### Model 9 Percentage.of.Vote.won.x ~ Democrat * weightexperience
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat * weightexperience, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse9 <- c(rmse9, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

####
#### ##### Model 10 Percentage.of.Vote.won.x ~ Democrat + Incumbent + weightexperience
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat + Incumbent + weightexperience, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse10 <- c(rmse10, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

####
#### ##### Model 11 Percentage.of.Vote.won.x ~ Democrat + Incumbent * weightexperience
####

for(i in 1:500){
  split_electData <- initial_split(poll.data.2, prop=.8)
  elect_train <- training(split_electData)
  elect_test <- testing(split_electData)
  model <- lm(Percentage.of.Vote.won.x ~ Democrat * Incumbent * weightexperience, data = elect_train)
  pred <- predict(model, newdata = elect_test)
  rmse11 <- c(rmse11, sqrt(mean((pred-elect_test$Percentage.of.Vote.won.x)^2)))
}

df.2 <- data.frame(cbind(mean(rmse0), mean(rmse6), mean(rmse7), mean(rmse8), mean(rmse9),
                           mean(rmse10), mean(rmse11)))
colnames(df.2) <- c("BL", "Poly Inc", "Int. Inc.", "Poly weight",
                   "Int. weight", "BL + Inc + weight",
                   "Int. Inc and weight")
kable(df.2)

```

BL	Poly Inc	Int. Inc.	Poly weight	Int. weight	BL + Inc + weight	Int. Inc and weight
12.07364	8.63787	8.644567	8.833417	8.940184	8.240711	8.212627

As it can be seen above, the smallest root mean square errors belongs to the additive model of democrat, weightexperience, and incumbency.

Now, using the win variable as your outcome, create at least 3 classification models. You should again assess each model on your “validation” set using appropriate methods. You must fit at least one of each:

- linear classifier,
- random forest model,
- K-nearest neighbors.

**Now you are going to assess your classifiers using the 2018 election.:**

- Most of the data you need is here: <http://politicaldatascience.com/PDS/Datasets/SenateForecast/PollingCandidateData1>
- BUT, this dataset is missing (a) the final outcome and (b) a lot of the polling data.
- Scrape the election results and polls from [ballotpedia.org](http://ballotpedia.org).
- This does not need to be perfect, but should demonstrate the basic skills covered on webscraping.
- Assess how each of your classifiers performs for 2018 using appropriate metrics.