Problem Set 2: Ranks, Permutations, and Simulations

Statistics 139 Teaching Staff

Due: September 30, 2023

This assignment is **due Saturday, September 30 at 11:59pm**, handed into Gradescope. Remember, there are two submissions, one for your pdf, and another for your rmd file. Show your work and provide clear, convincing, and succinct explanations when asked. **Incorporate the relevant R output in this R markdown file**; choose the included R wisely. Only the key output should be displayed for each problem and the relevant parts should be **highlighted** in some way. Make sure that you write-up any interpretation of R-code in your own words (don't just provide the output).

When performing a hypothesis test, be sure to explicitly state (1) hypotheses, (2) the calculated test statistic (and degrees of fredom if appropriate), (3) the calculated p-value or critical value, and (4) the conclusion in context of the problem along with the scope of inference. Use Type I error rates of $\alpha = 0.05$ and confidence levels of 95% unless explicitly stated otherwise. You can assume all tests are two-sided unless otherwise specified.

Collaboration policy (for this and all future homeworks): You are encouraged to discuss the problems with other students, but you must write up your solutions yourself and in your own words. Copying someone else's solution, or just making trivial changes is not acceptable.

```
library(perm)
```

Problem 1.

a) Given a single sample of data with n unique values, show that the number of distinct bootstrap samples is

$$\binom{2n-1}{n}$$

Using Example 1.4.22 (stars and bars) from the Stat 110 textbook, we can apply the formula for $k = n$ and get the number of distinct bootstrap samples as $\binom{2n-1}{n}$

What does this equate to if $n = 10$?

92,378 unique bootstrap samples for $n = 10$.

```
n <- 10
choose(2*n-1, n)
```

```
## [1] 92378
```

b) Let $n = 10$. If you take just $B = 100$ bootstrap resamples, what is the probability that at least 2 of your resamples are identical (meaning, they sample all of the n individual observations the exact same number of times)? What if you take $B = 1000$ bootsrap resamples?

P(at least 2 identical resamples) = 1 - P(no identical resamples)

$$= 1 - \frac{92378 \cdot 92377 \cdot ... \cdot 92279}{92378^{100}} = 0.052$$

P(at least 2 identical resamples) = 1 - P(no identical resamples) =

$$1 - \frac{92378 \cdot 92377 \cdot ... \cdot 91379}{92378^{1000}} = 0.996$$

```r
pbirthday(100, classes = 92378, coincident = 2)
```

```
## [1] 0.0521921
```

```r
pbirthday(1000, classes = 92378, coincident = 2)
```

```
## [1] 0.9956026
```

Problem 2.

In lecture we defined the Rank Sum Test Statistic to be:

$$W = \sum_{i=1}^{n_1} Z_{1,i}$$

In class we showed that $E(W) = \frac{n_1(n_1+n_2+1)}{2}$. For this problem, show that $\text{Var}(W) = \frac{n_1 n_2(n_1+n_2+1)}{12}$.

Hint: use the fact that if all the observations come from one group (aka, W is the sum of **all** the $n_1 + n_2$ ranks), then W is a constant.

The ranks are distributed Discrete uniform (uniform because each observation is equally likely to have a certain rank, and discrete because the rank can only take on numeric values). We have $n_1 + n_2$ observations in total. Using the variance of Discrete uniform distribution, we know that

$$\text{Var}(Z_i) = \frac{(n_1+n_2)^2 - 1}{12}$$

We also know that W (sum of all the ranks) is a constant, so $\text{Var}(W) = 0$.

$$\text{Var}(W) = \text{Var}(W_1 + W_2) = \sum_{i=1}^{i=n_1+n_2} \text{Var} Z_i + \sum_{i,j diff} \text{Cov}(Z_i, Z_j)$$

$$= (n_1+n_2) \cdot \text{Var}(Z_1) + 2\binom{n_1+n_2}{2} \cdot \text{Cov}(Z_1, Z_2)$$

$$= (n_1+n_2)\frac{(n_1+n_2)^2-1}{12} + (n_1+n_2)(n_1+n_2-1)\text{Cov}(Z_1, Z_2)$$

Since $\text{Var}(W) = 0$, we get $\text{Cov}(Z_1, Z_2) = -\frac{(n_1+n_2)^2-1}{12(n_1+n_2-1)}$

$$\text{Var}(W_1) = \sum_{i=1}^{i=n_1} \text{Var} Z_i + \sum_{i,j diff} \text{Cov}(Z_i, Z_j)$$

$$= n_1 \cdot \text{Var}(Z_1) + 2\binom{n_1}{2} \cdot \text{Cov}(Z_1, Z_2)$$

$$= n_1(\frac{(n_1+n_2)^2-1}{12}) + n_1(n_1-1)(-\frac{(n_1+n_2)^2-1}{12(n_1+n_2-1)})$$

$$= n_1\frac{(n_1+n_2)^2-1}{12}(1 + \frac{1-n_1}{n_1+n_2-1})$$

$$= n_1\frac{(n_1+n_2-1)(n_1+n_2+1)}{12}\frac{n_2}{n_1+n_2-1}$$

$$= \frac{n_1 n_2(n_1+n_2+1)}{12}$$

Problem 3: Election Data Cleaning

The data set `housedata.csv` contains all the House of Representative election results from 1976-2020 (source: Harvard Data Verse). It is not in a good *rectangular* data set format (where each row represents a single election for a voting district) since each voting district may have anywhere from 1 to 10+ rows to represent it. Note: some preprocessing was already done for you. For example, candidates with less than 0.5% of the vote were removed.

The variables measured are:

- `year`: the year the election was held
- `state`: the state the election district is located
- `electiondistrict`: a unique ID for each voting district
- `runoff`: a boolean indicated if these were the results of a run-off
- `special`: a boolean indicating if these were the results of a special election (typically out of cycle)
- `candidate`: the candidates' name,
- `party`: The political party of the candidate (republican, democrat, and many others),
- `writein`: an indicator as to whether the candidate was a write-in candidate (and not on the ballot).
- `candidatevotes`: the total number of votes in the election that that candidate received.
- `totalvotes`: the total number of votes in that election.

For this (and the next) problem, we'd like to [mostly] analyze the 2020 election. Start by doing some data cleaning:

a) Use the `summary` on the entire data set, provide the output, and comment on anything that looks surprising or strange. If any of the variables appear to be the wrong type (for example, aren't numeric when they should be), make sure you fix those before going forward (this may or may not be an issue, depending on what version of R and arguments to `read.csv` you use).

The minimum `candidatevotes` value is `-1`; the number of votes for any candidate should be a non-negative value. There are quite a lot of `NA` values for `runoff`, and most elections are not run-off elections (which makes sense). And very few candidates are write-in (which also makes sense). All the variables are of the correct type (`party` can be changed into the type `factor`).

```r
# load libraries
library(ggplot2)

# load data
house_data <- read.csv("data/housedata.csv", as.is = T)

# summary
summary(house_data)
```

```
##       year          state           electiondistrict     runoff
##  Min.   :1976   Length:27476       Length:27476       Mode :logical
##  1st Qu.:1988   Class :character   Class :character   FALSE:19819
##  Median :2000   Mode  :character   Mode  :character   TRUE :8
##  Mean   :1999                                         NA's :7649
##  3rd Qu.:2010
##  Max.   :2020
##   candidate            party             writein        candidatevotes
##  Length:27476       Length:27476       Mode :logical   Min.   :    -1
```

```
##  Class :character   Class :character   FALSE:27324     1st Qu.:  9657
##  Mode  :character   Mode  :character   TRUE :152       Median : 69564
##                                                        Mean   : 74857
##                                                        3rd Qu.:117647
##                                                        Max.   :387109
##    totalvotes
##  Min.   :    -1
##  1st Qu.:159631
##  Median :203530
##  Mean   :211716
##  3rd Qu.:258293
##  Max.   :716149
```

b) Convert `party` into a categorical variable with 3 categories: 'republican', democrat', and 'other'. Provide the table for this variable.

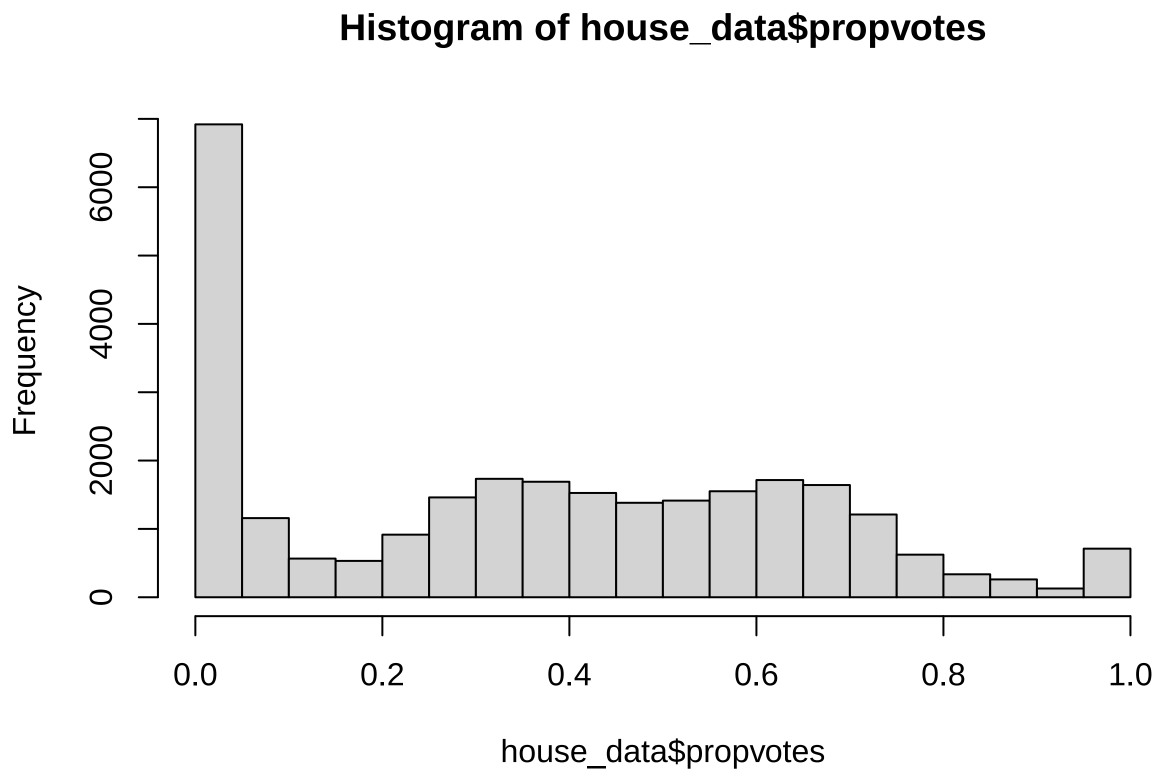
```r
# change to lowercase
house_data$party <- tolower(house_data$party)
x <- house_data$party

# create "other" as a category
x[(x != "republican") & (x != "democrat")] <- "other"

# recode to factor
house_data$party <- factor(x, levels = c("democrat", "republican", "other"),
                             labels = c("democrat", "republican", "other"))

# table
table(house_data$party)
```

```
##
##   democrat republican      other
##       9604       9264       8475
```

c) Create a new variable called `propvotes` in the data frame that converts each candidate's votes into the proportion of votes they received within their district. Plot the histogram and comment on what you notice.

A lot of candidates got nearly zero percent of the votes/very few votes. This makes sense because many candidates are in the "other" category and are less likely to win significant number of votes. A few candidates won with a landslide and would get nearly 100% proportion of the votes. The vast majority of candidates got somewhere in the middle, which is a reasonable outcome.

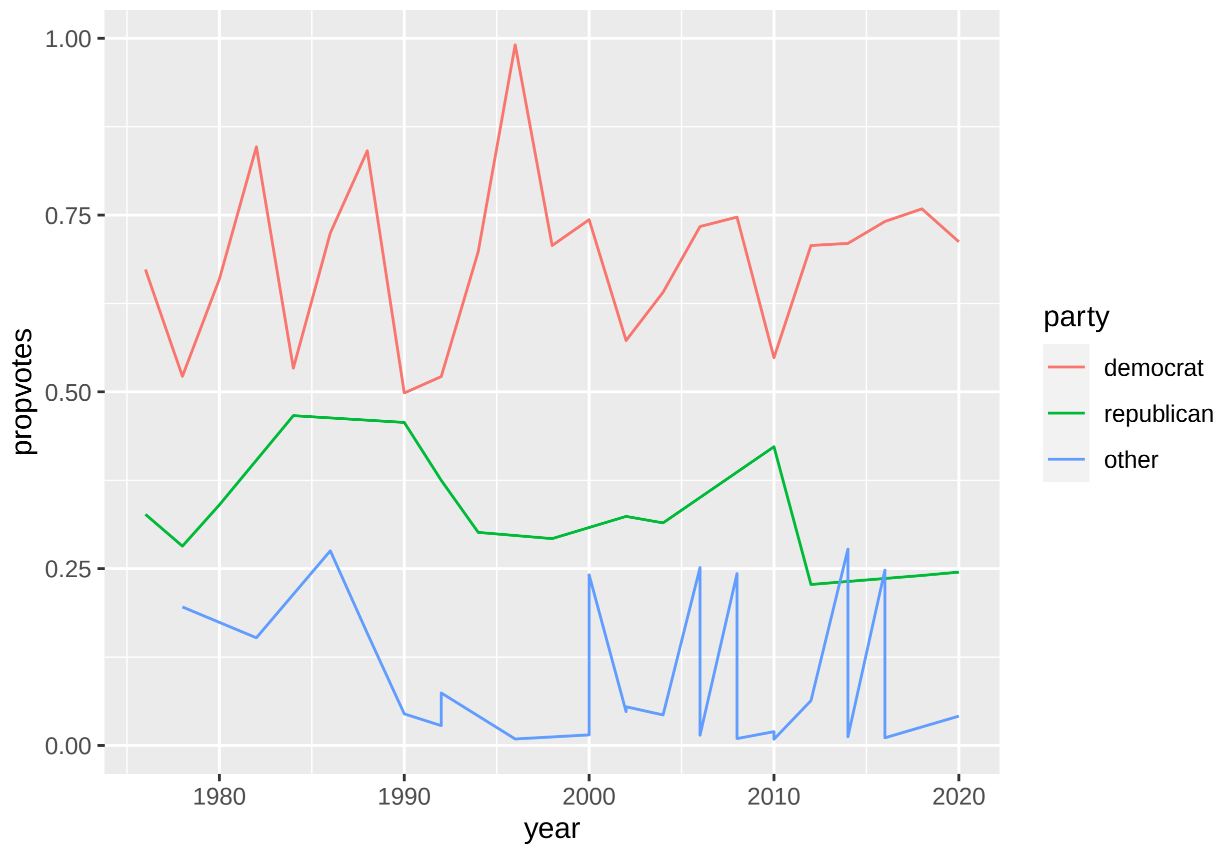
```r
house_data$propvotes <- house_data$candidatevotes / house_data$totalvotes
hist(house_data$propvotes)
```



d) Determine your election district here (or use 1 Oxford St, Cambridge, MA, as your address). Provide a visual to examine the proportion of votes for each of the 3 'parties' over time in your district (do the best you can if yours was redistricted in the past). Summarize what you find in 2-3 sentences.

Since the 1970s, Democrats have always won in district 5 of MA (where Harvard is); there was one close election in 1990. Generally the Republican candidate got more votes than candidates in "Other" parties, except for in 2014 and 2016.

```
# subset data
mydistrict <- house_data[house_data$electiondistrict=="MA5",]

# visual
ggplot(mydistrict, aes(year, propvotes, col = party)) +
  geom_line()
```



(e) Create two smaller data frames: one which only contains the results from 2018 and one from just 2020 (call them `house18`, `house20`). Report the dimensions of the two data frames. Use these (or consider just 2018 and 2020 from the original data frame) for all remaining questions.

The `house18` dataset is 1178x10, and the `house20` dataset is 1209x10.

```
# subset 2018 data
house18 <- house_data[house_data$year == 2018, ]
dim(house18)
```

```
## [1] 1178   10
```

```
# subset 2020 data
house20 <- house_data[house_data$year == 2020, ]
dim(house20)
```

```
## [1] 1209   10
```

(f) The following code splits `house18` into a list where each entry contains a separate data frame for each election district, and then `num.candidates18` is created to store the number of candidates in each election. Create a variable `winners18` that stores the winning candidate of each election. Similarly, create a variable `winner2020` and print out the first 10 winners for each of 2018 and 2020.

```r
# 2018 data
house18.list = split(house18, as.character(house18$electiondistrict))
num.candidates18 = sapply(house18.list,nrow,simplify=T)

winners18 = rep(NA,length(house18.list))
for(i in 1:length(house18.list)){
  temp = house18.list[[i]]
  winners18[i] <- temp[temp$propvotes == max(temp$propvotes), ]$candidate
}

head(winners18, 10)
```

```
##  [1] "DON YOUNG"        "BRADLEY BYRNE"    "MARTHA ROBY"      "MIKE ROGERS"
##  [5] "ROBERT ADERHOLT" "MO BROOKS"        "GARY PALMER"      "TERRI SEWELL"
##  [9] "RICK CRAWFORD"   "FRENCH HILL"
```

```r
# 2020 data
house20.list = split(house20, as.character(house20$electiondistrict))
num.candidates20 = sapply(house20.list,nrow,simplify=T)

winners20 = rep(NA,length(house20.list))
for(i in 1:length(house20.list)){
  temp = house20.list[[i]]
  winners20[i] <- temp[temp$propvotes == max(temp$propvotes), ]$candidate
}

head(winners20, 10)
```

```
##  [1] "DON YOUNG"                          "JERRY CARL"
##  [3] "BARRY MOORE"                        "MIKE ROGERS"
##  [5] "ROBERT B. ADERHOLT"                 "MO BROOKS"
##  [7] "GARY J. PALMER"                     "TERRI A. SEWELL"
##  [9] "ERIC A. \u0093RICK\u0094 CRAWFORD" "J. FRENCH HILL"
```

Problem 4: Election Data Analysis

The remaining problems refer to only the 2020 elections:

a) Answer a few exploratory questions:

i. What proportion of elections went unopposed (had a single candidate)?
ii. What proportion of elections did Democrats, Republicans, and Other parties win?
iii. When incumbents were involved (someone who won the election in 2018), what proportion of elections did they win? Note: the command `agrep` will help to deal with candidates' names changing from one election to the next based on fuzzy matching (there are other options as well).

(i) 1.15% of elections went unopposed.

(ii) Democrats won 51% of elections, Republicans won 49% of elections. Candidates in other parties never won any election.

(iii) When incumbents were involved, they won 96% of the elections.

```
# part i
mean(num.candidates20 == 1)*100
```

```
## [1] 1.149425
```

```
# part ii
# create vector of party of winners
winnersparty20 = rep(NA,length(house20.list))

# for each election
for(i in 1:length(house20.list)){

  # get the data related to that election
  temp = house20.list[[i]]

  # get the party of the winner in that election
  winnersparty20[i] <- temp[temp$propvotes == max(temp$propvotes), ]$party
}

# create table
prop.table(table(winnersparty20))
```

```
## winnersparty20
##         1         2
## 0.5103448 0.4896552
```

```
# part iii
# create vector of whether an incumbent is reelected
# 1: reelected; 0: ran again but didn't win; NA: did not run again
reelected = rep(NA, length(winners18))

# for each winner in 2018
for(i in 1:length(winners18)){
```

```
  # get name of 2018 winner
  incumbent <- winners18[i]

  # check if incumbent runs again
  if(any(grepl(incumbent, house20.list[[i]]$candidate))){

    # get name of elected person
    elected <- winners20[i]

    # if incumbent wins reelection
    if(grepl(incumbent, elected)){

      # return 1, else return 0
      reelected[i] <- 1
    } else reelected[i] <- 0
  }
}

# prop: remove NA values, calculate proportion of 1 value
reelected <- na.omit(reelected)
sum(reelected)/length(reelected)
```

```
## [1] 0.9603175
```

b) First remove all candidates with fewer than 2% of the vote for the following question: provide point estimates and 95% confidence intervals for the "3rd party effect" for Republicans and for Democrats, separately, in elections that were not unopposed. That is, how much higher (or, lower) of the proportion of the vote did a Democrat receive when in an election with 3 or more candidates vs. an election with exactly 2 candidates (after removing the low vote getters)? Do the same calculation for Republicans. The elections you should consider are those with a single Democrat, a single Republican, and one or more Other parties (vs. just a single Democrat and single Republican). You should also throw away all write-in votes. You should perform two confidence intervals: one parametric and one bootstrapped.

```
# subset
house20_sub <- house20[house20$propvotes >= 0.02 & house20$writein == FALSE,]

# list of all district
district <- unique(house20_sub$electiondistrict)

# create new vars: count of dem, rep, and other candidate in each election
house20_sub$dem.count <- NA
house20_sub$rep.count <- NA
house20_sub$other.count <- NA

# add count of candidates by party and election district
# for each election district
for(i in 1:length(district)){

  # subset for data of that district
  dist_data <- house20_sub[house20_sub$electiondistrict == district[i], ]

  # add dem count
  house20_sub[house20_sub$electiondistrict == district[i], ]$dem.count <-
```

```r
  sum(1*(dist_data$party == "democrat"))

  # add rep count
  house20_sub[house20_sub$electiondistrict == district[i], ]$rep.count <-
  sum(1*(dist_data$party == "republican"))

  # add other count
  house20_sub[house20_sub$electiondistrict == district[i], ]$other.count <-
  sum(1*(dist_data$party == "other"))

}

# subset data
# elections where there are 1 dem candidate and 1 rep candidate
two_candidates <- house20_sub[house20_sub$dem.count == 1 &
                            house20_sub$rep.count == 1 &
                            house20_sub$other.count == 0, c("party", "propvotes")]
two_candidates$num <- "two"
two_candidates_dems <- two_candidates[two_candidates$party == "democrat", "propvotes"]
two_candidates_reps <- two_candidates[two_candidates$party == "republican", "propvotes"]

# elections where there are 1 dem candidate, 1 rep candidate, and >0 other candidate
more_candidates <- house20_sub[house20_sub$dem.count == 1 &
                            house20_sub$rep.count == 1 &
                            house20_sub$other.count > 0, c("party", "propvotes")]
more_candidates$num <- "more"
more_candidates_dems <- more_candidates[more_candidates$party == "democrat", "propvotes"]
more_candidates_reps <- more_candidates[more_candidates$party == "republican", "propvotes"]

candidate_data <- rbind(two_candidates, more_candidates)
dem_data <- candidate_data[candidate_data$party == "democrat", ]
rep_data <- candidate_data[candidate_data$party == "republican", ]

#### POINT ESTIMATES
estimate_dem <- mean(more_candidates_dems) - mean(two_candidates_dems)
estimate_dem
```

```
## [1] -0.03530893
```

```r
estimate_rep <- mean(more_candidates_reps) - mean(two_candidates_reps)
estimate_rep
```

```
## [1] -0.01963929
```

```r
#### PARAMETRIC
# for dems
parametric_dem <- t.test(propvotes ~ num, dem_data)$conf.int
parametric_dem
```

```
## [1] -0.068912398 -0.001705456
## attr(,"conf.level")
## [1] 0.95
```

```r
# for reps
parametric_rep <- t.test(propvotes ~ num, rep_data)$conf.int
parametric_rep
```

```
## [1] -0.05541464  0.01613605
## attr(,"conf.level")
## [1] 0.95
```

```r
#### BOOTSTRAP
nsims <- 1000

boot_4b <- function(data_two, data_more){

  # get size of two-candidate data set
  size_two <- length(data_two)

  # generate bootstrap sample for two-candidate data set
  boot_two <- sample(data_two, size_two, replace=T)

  # get size of more-candidate data set
  size_more <- length(data_more)

  # generate bootstrap sample for more-candidate data set
  boot_more <- sample(data_more, size_more, replace=T)

  # return difference in means
  return(mean(boot_more) - mean(boot_two))
}

# for dems
boot_dems_4b <- replicate(nsims, boot_4b(two_candidates_dems, more_candidates_dems))
bootstrap_dem <- quantile(boot_dems_4b, c(0.025, 0.975))
bootstrap_dem
```

```
##         2.5%        97.5%
## -0.065953561 -0.002665627
```

```r
# for reps
boot_reps_4b <- replicate(nsims, boot_4b(two_candidates_reps, more_candidates_reps))
bootstrap_rep <- quantile(boot_reps_4b, c(0.025, 0.975))
bootstrap_rep
```

```
##        2.5%       97.5%
## -0.05857294  0.01303895
```

I calculated the 3rd party effect to be the difference between propvotes when more than 2 candidates are present vs when only 2 candidates are present.

For the Democrats, the point estimate of the 3rd party effect is -0.035. Since this is negative but still close to 0, we can expect that Democrat candidates get slightly less votes proportionally (around 3.5% less) when more than 2 candidates are present vs when there are only 2. Using the parametric method, the 95% CI for the Democrats 3rd party effect is (-0.069, -0.002), and using the bootstrap method, the 95% CI for the

Democrats 3rd party effect is (-0.066, -0.003). Both of these CIs do not contain 0 (although the upperbound is very close to 0), so we can generally expect the true 3rd party effect for Democrats to be negative.

For the Republicans, the point estimate of the 3rd party effect is -0.02. Since this is negative but still very close to 0, we can expect that Democrat candidates get slightly less votes proportionally (around 2% less) when more than 2 candidates are present vs when there are only 2. Using the parametric method, the 95% CI for the Democrats 3rd party effect is (-0.055, 0.016), and using the bootstrap method, the 95% CI for the Democrats 3rd party effect is (-0.059, 0.013). Both of these CIs do contain 0, so we can't be too sure about whether the 3rd party effect for the Republicans is negative or not (more of the interval is less than 0, and the point estimate is negative, so this is more likely).

c) Perform three tests to determine whether the "3rd party effect" was different for Democrats than for Republican: (i) a reasonable *t*-based test, (ii) a rank-based test, and (iii) a test based on resampling.

```r
# pull data
two_candidates_effect <- two_candidates_reps - two_candidates_dems
more_candidates_effect <- more_candidates_reps - more_candidates_dems

two_candidates_effect <- as.data.frame(two_candidates_effect)
two_candidates_effect$num <- "two"
colnames(two_candidates_effect) <- c("effect", "num")

more_candidates_effect <- as.data.frame(more_candidates_effect)
more_candidates_effect$num <- "more"
colnames(more_candidates_effect) <- c("effect", "num")

all_effect <- rbind(two_candidates_effect, more_candidates_effect)

# t test:
t.test(effect ~ num, all_effect)
```

```
##
##  Welch Two Sample t-test
##
## data:  effect by num
## t = 0.49585, df = 254.32, p-value = 0.6204
## alternative hypothesis: true difference in means between group more and group two is not equal to 0
## 95 percent confidence interval:
##  -0.04656474  0.07790401
## sample estimates:
## mean in group more  mean in group two
##        -0.01107831        -0.02674795
```

```r
# rank-based test
wilcox.test(effect ~ num, all_effect)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  effect by num
## W = 18755, p-value = 0.5897
## alternative hypothesis: true location shift is not equal to 0
```

```r
# resampling
permTS(effect ~ num, all_effect)
```

```
##
##  Permutation Test using Asymptotic Approximation
##
## data:  effect by num
## Z = 0.50652, p-value = 0.6125
## alternative hypothesis: true mean num=more - mean num=two is not equal to 0
## sample estimates:
## mean num=more - mean num=two
##                   0.01566963
```

```r
# manually
# nsims <- 1000
# diff.perm <- rep(NA, nsims)
# set.seed(139)
#
# for(i in 1:nsims){
#   effect.perm = sample(all_effect$effect)
#
#   diff.perm[i] = mean(effect.perm[all_effect$num=="two"]) -
#                  mean(effect.perm[all_effect$num=="more"])
# }
#
# diff.obs = mean(all_effect[all_effect$num=="two", "effect"]) - mean(all_effect[all_effect$num=="more"
#
# mean(abs(diff.perm) > abs(diff.obs)) # got 0.606
```

We are interested in the difference between the 3rd party effect for Democrats and Republicans. Let $p_{D,2}$ be `propvotes` of Democrats when there are only 2 candidates present, $p_{D,m}$ be `propvotes` of Democrats when there are more than 2 candidates present. Let $p_{R,2}$ and $p_{R,m}$ be the respective notations for Republicans. Let $p_2 = p_{R,2} - p_{D,2}$ and $p_m = p_{R,m} - p_{D,m}$. Mathematically, we are interested in testing whether $p_{D,m} - p_{D,2} = p_{R,m} - p_{R,2}$ holds. This is equivalent to testing whether $p_{R,2} - p_{D,2} = p_{R,m} - p_{D,m}$ holds or whether $p_2 = p_m$ holds. We rearrange the (in)equality into this form because the $p_{D,m}, p_{R,m}$ are paired and $p_{D,2}, p_{R,2}$ are paired, and we can compare the differences that way.

For the t-based test, I chose a 2-sample `t-test` because we are comparing two means. (1) In terms of hypotheses, H_0 states that the difference in mean difference (between D and R when there are 2 candidates vs between D and R when there are 3+ candidates) is 0. H_a states that the difference in mean difference is not 0. (2) The t-statistic is -0.4959 with df $= 254.32$. (3) The p-value is 0.6204. (4) Conclusion: since the p-value is greater than α, we fail to reject the null hypothesis and conclude that there is not enough evidence to say the difference in mean difference is not 0 (i.e. not enough evidence for the claim 3rd party effect is different between the two parties).

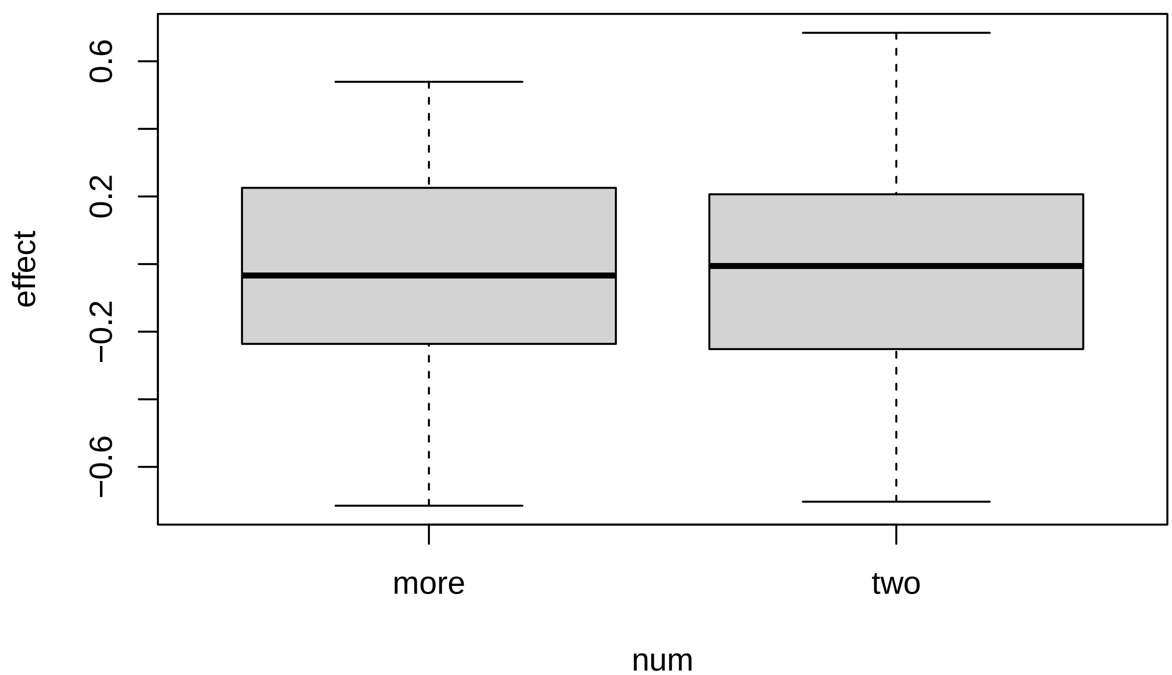
For the rank-based test, I did the `wilcox` rank sum test. (1) In terms of hypotheses, H_0 states that assuming the same underlying distribution of the differences p_2 and p_m, $Median(p_2) = Median(p_m)$. H_a states that $Median(p_2) \neq Median(p_m)$. (2) The W test statistic is 18755. (3) The p-value is 0.5897. (4) Conclusion: since the p-value is greater than α, we fail to reject the null hypothesis and conclude that there is not enough evidence to say that the 3rd party effect is different between the two parties.

For the resampling test, I did a `permutation` test. (1) In terms of hypotheses, H_0 states that the distribution of the difference in `propvotes` among Republicans and Democrats is the same for when there are 2 candidates and for when there are 3+ candidates. H_a states that the distribution of the difference is associated with whether there are 2 candidates vs 3+ candidates. (2) The Z test statistic is 0.507. (3) The p-value is 0.613. (4) Conclusion: since the p-value is greater than α, we fail to reject the null hypothesis and conclude that there is not enough evidence to say that the 3rd party effect is different between the two parties.

d) Compare the results of the three tests in the previous part. Provide visuals to support your comparisons and summarize your conclusions.

The three tests in part (c) give us the same conclusions. There is not enough evidence to support the claim that the 3rd party effect is different for the Democrats and Republicans. The boxplot below also supports this: the difference between Republicans and Democrats propvotes when 2 candidates are present vs when there are 3+ candidates present is similar.

```
boxplot(effect ~ num, all_effect)
```



Problem 5.

Which approach to estimate μ with 95% confidence is most reliable? Specifically we will compare the 4 following approaches to calculate a 95% confidence interval (CI):

1. Classic one-sample t-based CI
2. Percentile boostrap CI (our method #1 on slide 12 of lecture 5: pulling off the quantiles)
3. Studentized bootstrap CI (our method #2 on slide 12 of lecture 5: based on the t)

*Note: we ask you to code up the bootstrap intervals yourself and to not rely on the `boot` package.

Perform a simulation study (with `nsims = 500`) to estimate the coverage probability for each of these approaches under the conditions defined below. Perform `nboots = 139` bootstrap resamples each time (so it does not bog down your CPU).

a) Let's start with a single iteration. Let i.i.d. $X_i \sim N(5, 3^2)$ with $n = 10$. Calculate and report the 3 separate confidence intervals for your one sample of 10 observations. How do these compare?

```r
# set params
nsims <- 500
nboots <- 139
n_1 <- 10
n_2 <- 50
mean <- 5
sd <- 3
lambda <- 1/3
```

```r
# set seed
set.seed(139)

# generate data
x_5a <- rnorm(n_1, mean, sd)

# t-based

  # function
  t_based_ci <- function(data){

    t.test(data)$conf.int
  }

  # call fn
  t_based_5a <- t_based_ci(x_5a)
  t_based_5a
```

```
## [1] 2.615763 6.225340
## attr(,"conf.level")
## [1] 0.95
```

```r
# percentile bootstrap

  # function
  percentile_boot_ci <- function(data){
```

```r
    # sampling dist vector
    mean_boot <- rep(NA, nboots)

    # construct nboots bootstrap samples and calculate mean for each
    for(i in 1:nboots){
      boot_data <- sample(data, size = length(data), replace=T)
      mean_boot[i] <- mean(boot_data)
    }

    # return ci
    return(quantile(mean_boot, c(0.025, 0.975)))
  }

  # call fn
  percentile_boot_ci_5a <- percentile_boot_ci(x_5a)
  percentile_boot_ci_5a
```

```
##     2.5%    97.5%
## 3.186279 5.852636
```

```r
# studentized bootstrap

  # function
  studentized_boot_ci <- function(data){

    # sampling dist vector
    mean_boot <- rep(NA, nboots)

    # construct nboots bootstrap samples and calculate mean for each
    for(i in 1:nboots){
      boot_data <- sample(data, length(data), replace=T)
      mean_boot[i] <- mean(boot_data)
    }

    xbar <- mean(mean_boot)
    s_star <- sqrt(n/(n-1))*sd(mean_boot)
    t_star <- qt(0.025, n-1)

    # return ci
    return(c(xbar + t_star*s_star, xbar - t_star*s_star))
  }

  # call fn
  studentized_boot_ci_5a <- studentized_boot_ci(x_5a)
  studentized_boot_ci_5a
```

```
## [1] 2.792968 5.993266
```

The CI using one-sample t-based method is (2.616, 6.225). The CI using percentile bootstrap is (3.186, 5.853). The CI using studentized bootstrap CI is (2.793, 5.993). All 3 CIs include the true mean of 5, and the percentile bootstrap method gives you the smallest CI.

b) Let i.i.d. $X_i \sim N(5, 3^2)$ with $n = 10$ and separately when $n = 50$. Print out a 3-by-2 table (will look nice if you define it as a `data.frame` in R) of the empirically estimated coverage probabilities based on the 500 iterations (3 rows for the 3 methods, 2 columns for the 2 sample sizes).

```
# set seed
set.seed(139)

# t-based

  # function
  t_based_coverage <- function(n){

    # generate data
    data <- rnorm(n, mean, sd)

    # check if ci covers true mean
    if(t_based_ci(data)[1] < mean & t_based_ci(data)[2] > mean){
      return(1)
    }
    else return(0)
  }

  # call fn
  t_based_5b_1 <- mean(replicate(nsims, t_based_coverage(n_1)))
  t_based_5b_2 <- mean(replicate(nsims, t_based_coverage(n_2)))

# percentile bootstrap

  # function
  percentile_boot_coverage <- function(n){

    # generate data
    data <- rnorm(n, mean, sd)

    # check if ci covers true mean
    if(percentile_boot_ci(data)[1] < mean & percentile_boot_ci(data)[2] > mean){
      return(1)
    }
    else return(0)
  }

  # call fn
  percentile_boot_5b_1 <- mean(replicate(nsims, percentile_boot_coverage(n_1)))
  percentile_boot_5b_2 <- mean(replicate(nsims, percentile_boot_coverage(n_2)))

# studentized bootstrap
  # function
  studentized_boot_coverage <- function(n){

    # generate data
    data <- rnorm(n, mean, sd)

    # check if ci covers true mean
    if(studentized_boot_ci(data)[1] < mean & studentized_boot_ci(data)[2] > mean){
```

```r
      return(1)
    }
    else return(0)
  }

  # call fn
  studentized_boot_5b_1 <- mean(replicate(nsims, studentized_boot_coverage(n_1)))
  studentized_boot_5b_2 <- mean(replicate(nsims, studentized_boot_coverage(n_2)))

# display table
table_5b <- data.frame("n=10" = c(t_based_5b_1, percentile_boot_5b_1, studentized_boot_5b_1),
                       "n=50" = c(t_based_5b_2, percentile_boot_5b_2, studentized_boot_5b_2))
row.names(table_5b) <- c("t.based", "percentile.bootstrap", "studentized.bootstrap")
table_5b
```

```
##                        n.10  n.50
## t.based               0.934 0.964
## percentile.bootstrap  0.894 0.938
## studentized.bootstrap 0.944 0.980
```

c) Let i.i.d. $(Y_i - 2) \sim \text{Exponential}(\lambda = 1/3)$ with $n = 10$ and separately when $n = 50$. Print out a 3-by-2 table of the empirically estimated coverage probabilities based on the 500 iterations (3 rows for the 3 methods, 2 columns for the 2 sample sizes).

```r
# set seed
set.seed(139)
mean_exp <- 1/lambda

# t-based

  # function
  t_based_coverage <- function(n){

    # generate data
    data <- rexp(n, lambda)

    # check if ci covers true mean
    if(t_based_ci(data)[1] < mean_exp & t_based_ci(data)[2] > mean_exp){
      return(1)
    }
    else return(0)
  }

  # call fn
  t_based_5c_1 <- mean(replicate(nsims, t_based_coverage(n_1)))
  t_based_5c_2 <- mean(replicate(nsims, t_based_coverage(n_2)))

# percentile bootstrap

  # function
  percentile_boot_coverage <- function(n){

    # generate data
```

```r
    data <- data <- rexp(n, lambda)

    # check if ci covers true mean
    if(percentile_boot_ci(data)[1] < mean_exp & percentile_boot_ci(data)[2] > mean_exp){
      return(1)
    }
    else return(0)
  }

  # call fn
  percentile_boot_5c_1 <- mean(replicate(nsims, percentile_boot_coverage(n_1)))
  percentile_boot_5c_2 <- mean(replicate(nsims, percentile_boot_coverage(n_2)))

# studentized bootstrap
  # function
  studentized_boot_coverage <- function(n){

    # generate data
    data <- rexp(n, lambda)

    # check if ci covers true mean
    if(studentized_boot_ci(data)[1] < mean_exp & studentized_boot_ci(data)[2] > mean_exp){
      return(1)
    }
    else return(0)
  }

  # call fn
  studentized_boot_5c_1 <- mean(replicate(nsims, studentized_boot_coverage(n_1)))
  studentized_boot_5c_2 <- mean(replicate(nsims, studentized_boot_coverage(n_2)))

# display table
table_5c <- data.frame("n=10" = c(t_based_5c_1, percentile_boot_5c_1, studentized_boot_5c_1),
                    "n=50" = c(t_based_5c_2, percentile_boot_5c_2, studentized_boot_5c_2))
row.names(table_5c) <- c("t.based", "percentile.bootstrap", "studentized.bootstrap")
table_5c
```

```
##                        n.10  n.50
## t.based               0.928 0.940
## percentile.bootstrap  0.862 0.930
## studentized.bootstrap 0.902 0.964
```

d) Interpret the results from (b) and (c). How does sample size affect the coverage probabilities? How does the distribution of the underlying data affect the coverage probabilities? Which method(s) perform best (live up to their 95% nominal level) under the settings considered?

As the sample size increases, coverage probability also increases; this makes sense because when we get more data points, there is less randomness and we become more certain of our estimates. When the underlying distribution is nor normal, the probability coverage decreases slightly for the case n=10 (largely unaffected for the case n=50). The `t.based` method performs best under the settings considered: it is robust to the non-normal underlying distribution.

e) Use these results to justify the rampant use of the classical *t*-based methods in practice. Explain in 1-3 sentences.

When the underlying distribution is Normal, the three methods perform at comparable levels (perhaps the percentile bootstrapping method less so). But when the underlying distribution is not Normal, even with a small sample size, the t.based method still gives us really good coverage probability (close to 0.95). This means that the t-based method is very robust to assumption violations and we can reliably use it widely.