SML 201 Project 2

Justin Tran

November 24, 2017

Project 2 is due by 11:59pm on Wednesday November 29. Please submit **both a .Rmd and a .pdf** files on Blackboard **and** bring a hard copy of the pdf file to the first lecture after the due date. Make sure that you have all your digital signatures along with the honor pledge in each of these documents (there should be more than one signature if you work in groups).

If you are completing this project in a group, please have only **one** person in your group turning in the .Rmd and .pdf files; other people in your group should turn in a .R file that contains the names of everyone in your group.

Please type your name(s) after "Digitally signed:" below the honor pledge to serve as digital signature(s). Put the pledge and your signature(s) at the beginning of each document that you turn in.

> I pledge my honor that I have not violated the honor code when completing this assignment.

Digitally signed: Justin Tran

In order to receive full credits, please have sensible titles and axis labels for all your graphs and adjust values for all the relevant graphical parameters so that your plots are informative. Also, all answers must be written in complete sentences.

Question 1

Part a

```
load("aug_poll.RData")
oct_poll = subset(aug_poll, aug_poll$end.date >= as.Date("10/01/2016",
    format = "%m/%d/%Y"))
nrow(oct_poll)
[1] 2256
```

I created a subset for `oct_poll` that contains all polls ending on or after October 1st. It has `nrow(oct_poll)` rows.

Part b

```
poll.oct2016 = subset(oct_poll, select = -c(grade,
    end.date))  # Remove unnecessary columns
poll.oct2016$rawpoll_clinton = (poll.oct2016$rawpoll_clinton/100) *
    poll.oct2016$samplesize  # Transform to pop. from percentage
poll.oct2016$rawpoll_trump = (poll.oct2016$rawpoll_trump/100) *
    poll.oct2016$samplesize  # Transform to pop. from percentage

by.sample = aggregate(poll.oct2016$samplesize, by = list(state = poll.oct2016$state),
    FUN = sum)  # Total sampled by state
```

```
by.clinton = aggregate(poll.oct2016$rawpoll_clinton,
    by = list(state = poll.oct2016$state), FUN = sum)  # Total clinton voters by state
by.clinton = by.clinton["x"]/by.sample["x"]  # Aggregate proportion of clinton voters
by.trump = aggregate(poll.oct2016$rawpoll_trump, by = list(state = poll.oct2016$state),
    FUN = sum)  # Total trump voters by state
by.trump = by.trump["x"]/by.sample["x"]  # Aggregate proportion of clinton voters

# Combine into one dataframe
poll.oct2016 = data.frame(by.sample, by.clinton, by.trump)
colnames(poll.oct2016) = c("state", "samplesize", "rawpoll_clinton",
    "rawpoll_trump")
head(poll.oct2016)
       state samplesize rawpoll_clinton rawpoll_trump
1    Alabama      12063       0.2957986     0.5407581
2     Alaska       8874       0.3276403     0.3906257
3    Arizona      43827       0.4232979     0.4142356
4   Arkansas      10200       0.3375070     0.5016534
5 California      43545       0.5253698     0.2811303
6   Colorado      43023       0.4361154     0.3670996
```

To create this dataframe, I had to handle the aggregation of percentages. I first changed the `rawpoll` columns to show population rather than percentage. Then, I aggregated the `samplesize` by state by summing. Next, I aggregated `rawpoll` proportions for Clinton and Trump by summing up the populations by state then dividing by `samplesize` per state. Finally, I combined each of these data frames together.

Question 2

```
true.perc = read.csv(file = "2016_election_result.csv")
true.perc = merge(poll.oct2016, true.perc, by = "state")

poll.diff = true.perc$rawpoll_trump - true.perc$rawpoll_clinton
vote.diff = true.perc$trump.vote - true.perc$clinton.vote
deviation = poll.diff - vote.diff

election = data.frame(true.perc, poll.diff, vote.diff,
    deviation)
head(election)
       state samplesize rawpoll_clinton rawpoll_trump clinton.vote
1    Alabama      12063       0.2957986     0.5407581    0.3435795
2     Alaska       8874       0.3276403     0.3906257    0.3655087
3    Arizona      43827       0.4232979     0.4142356    0.4512602
4   Arkansas      10200       0.3375070     0.5016534    0.3365312
5 California      43545       0.5253698     0.2811303    0.6172640
6   Colorado      43023       0.4361154     0.3670996    0.4815698
  trump.vote    poll.diff   vote.diff   deviation
1  0.6208309  0.244959513  0.27725147 -0.03229195
2  0.5128151  0.062985429  0.14730641 -0.08432098
3  0.4867162 -0.009062222  0.03545595 -0.04451817
4  0.6057410  0.164146382  0.26920978 -0.10506340
5  0.3161711 -0.244239504 -0.30109293  0.05685343
6  0.4325140 -0.069015815 -0.04905583 -0.01995998
```

I loaded the .csv file and merged the resulting dataframe with `poll.oct2016` to create `true.perc`. I then

calculated the 3 additional column values `poll.diff`, `vote.diff`, and `deviation`. Finally, I merged these additional columns with `true.perc` to create `election`.

Question 3

part a

```
# Number of SE units each element in poll.diff is
# away from vote.diff
std.unit = election$deviation/sd(election$poll.diff)
# If poll.diff and vote.diff have the same sign
same.sign = election$poll.diff * election$vote.diff >
    0

# Standard error of the paired t test
significant.se = sqrt(var(election$rawpoll_trump)/length(election$rawpoll_trump) +
    var(election$rawpoll_clinton)/length(election$rawpoll_clinton))

# Upper value of the CI for estimating prop. Trump
# supporters - prop. Clinton supporters
significant.high = apply(election[, 3:4], 2, function(x) election$rawpoll_trump -
    election$rawpoll_clinton + 1 * (qnorm(0.975) *
    significant.se))
# Remove unnec. columns
significant.high = subset(significant.high, select = -c(rawpoll_trump))

# Lower value of the CI for estimating prop. Trump
# supporters - prop. Clinton supporters
significant.low = apply(election[, 3:4], 2, function(x) election$rawpoll_trump -
    election$rawpoll_clinton - 1 * (qnorm(0.975) *
    significant.se))
# Remove unnec. columns
significant.low = subset(significant.low, select = -c(rawpoll_clinton))

# Dataframe displaying true if the CI doesn't
# contain 0 and false otherwise
significant = data.frame(significant.low, significant.high)
head(significant)  # Temporary column names that will be overwritten
  rawpoll_trump rawpoll_clinton
1    0.20824147      0.28167756
2    0.02626739      0.09970347
3   -0.04578027      0.02765582
4    0.12742834      0.20086443
5   -0.28095755     -0.20752146
6   -0.10573386     -0.03229777
significant = !(significant[, 1] < 0 & significant[,
    2] > 0)

# Add significant dataframe to election dataframe
election = data.frame(election, std.unit, same.sign,
    significant)
head(election)
       state samplesize rawpoll_clinton rawpoll_trump clinton.vote
```

```
1    Alabama      12063       0.2957986     0.5407581    0.3435795
2     Alaska       8874       0.3276403     0.3906257    0.3655087
3    Arizona      43827       0.4232979     0.4142356    0.4512602
4   Arkansas      10200       0.3375070     0.5016534    0.3365312
5 California      43545       0.5253698     0.2811303    0.6172640
6   Colorado      43023       0.4361154     0.3670996    0.4815698
  trump.vote    poll.diff   vote.diff   deviation   std.unit same.sign
1  0.6208309  0.244959513  0.27725147 -0.03229195 -0.1766510      TRUE
2  0.5128151  0.062985429  0.14730641 -0.08432098 -0.4612724      TRUE
3  0.4867162 -0.009062222  0.03545595 -0.04451817 -0.2435338     FALSE
4  0.6057410  0.164146382  0.26920978 -0.10506340 -0.5747425      TRUE
5  0.3161711 -0.244239504 -0.30109293  0.05685343  0.3110130      TRUE
6  0.4325140 -0.069015815 -0.04905583 -0.01995998 -0.1091898      TRUE
  significant
1        TRUE
2        TRUE
3       FALSE
4        TRUE
5        TRUE
6        TRUE
```

I first created the `std.unit` vector using the deviation column of `election` divided by the standard deviation of `vote.diff` column. I used `deviation` because it is the difference between `poll.diff` and `vote.diff`.

I created `same.sign` by checking if the product of `poll.diff` and `vote.diff` was positive indicating the same signs.

For `significant`, I assume all samples are simple random samples and the observations are independent. The sample sizes are also > 30 so I are able to create a confidence interval for each state's data. I first calculated the standard error which was 0.018734. Then I calculated the lower and upper bounds of the 95% confidence interval for each state. For example, 0.2082415, 0.2816776 is the confidence interval for polls in Alabama. I then calculated if the confidence interval included 0 for each state and placed that vector in `significant`.

part b

```
sum(election$significant == T)  # Number of CIs not including 0 in interval
[1] 45

sum(election$significant == T & election$same.sign ==
    T)  # Correct prediction for those not including 0
[1] 42

sum(election$significant == F)  # Number of CIs including 0 in interval
[1] 6

sum(election$significant == F & election$same.sign ==
    T)  # Correct prediction for those including 0
[1] 3
```

The winning candidate was predicted correctly if `same.sign` had a FALSE value for the state. There were 45 states that did not include 0 in the confidence interval and 6 states that did include 0 in the confidence interval. For those not including 0, 42 states predicted the election for their state correctly. For those including 0, 3 states predicted the election for their state correctly.

part c

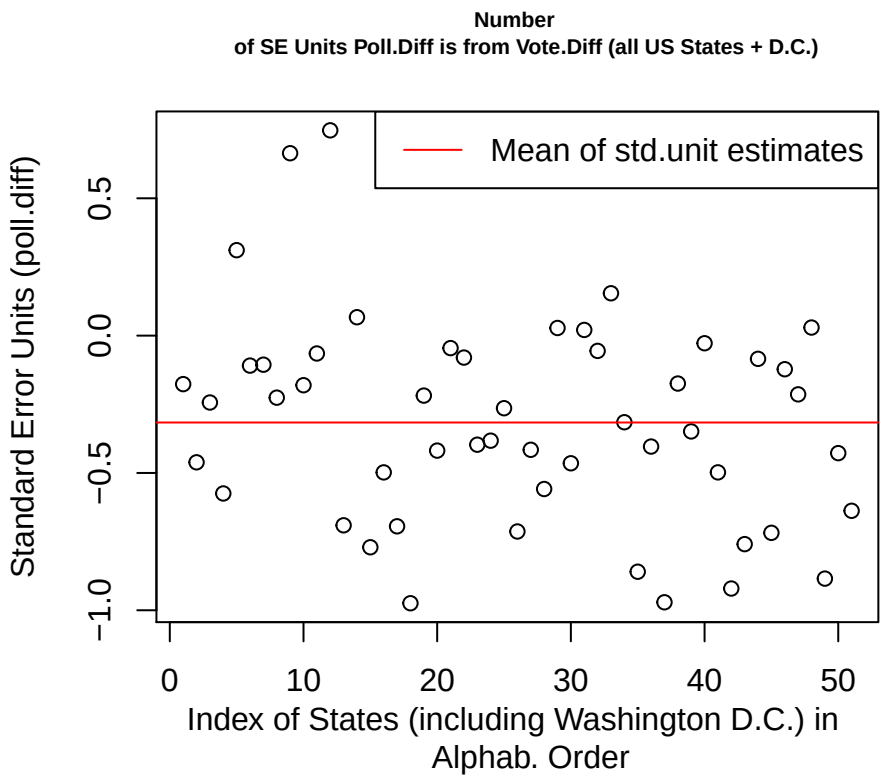
```
# Incorrect predictions by states with strictly
# pos. or neg. intervals
sum(election$significant == T & election$same.sign ==
    F)
[1] 3
# States associated with the above CIs
election$state[election$significant == T & election$same.sign ==
    F]
[1] Michigan     Pennsylvania Wisconsin
51 Levels: Alabama Alaska Arizona Arkansas California ... Wyoming
```

The number of CIs that give either strictly-positive or strictly-negative interval estimates and predict the winner of the state wrong is given by `sum(election$significant == T & election$same.sign == F)` and its value is 3. The 3 states were Michigan, Pennsylvania, Wisconsin.

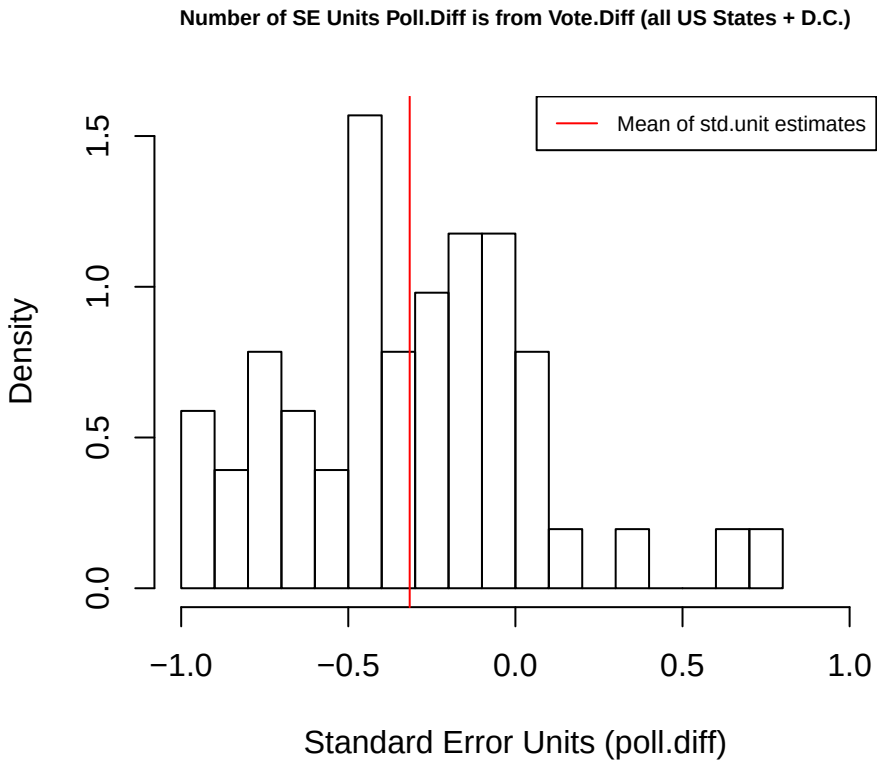
part d

Plot a scatterplot and a histogram of `std.unit`. Do the errors in the estimates look like they have mean zero?

```
plot(election$std.unit, xlab = "Index of States (including Washington D.C.) in
\t Alphab. Order",
    ylab = "Standard Error Units (poll.diff)", main = "Number
\t of SE Units Poll.Diff is from Vote.Diff (all US States + D.C.)",
    cex.main = 0.7)
abline(h = mean(election$std.unit), col = "Red")
legend("topright", "Mean of std.unit estimates", col = c("Red"),
    lty = 1)
```




```
hist(election$std.unit, breaks = 20, freq = F, xlim = c(-1,
    1), xlab = "Standard Error Units (poll.diff)",
    main = "Number of SE Units Poll.Diff is from Vote.Diff (all US States + D.C.)",
    cex.main = 0.7)
abline(v = mean(election$std.unit), col = "Red")
legend("topright", "Mean of std.unit estimates", col = c("Red"),
    lty = 1, cex = 0.7)
```



The errors in the estimates do not look like they have mean 0. I created the scatterplot and histogram for `std.unit` and placed a red line indicating the mean `std.unit` value. The line was placed well below the mean zero. In addition, many values on the scatterplot were below mean zero and the density of the histogram was not centered at mean zero. This may imply that there is bias in the polls but we will investigate that more.

part e

```
# Num of states with poll differences smaller than
# actual election differences
sum(election$poll.diff < election$vote.diff)
[1] 43

# Test for independence and for poll and vote
# differences
chisq.test(x = election$poll.diff, y = election$vote.diff)

    Pearson's Chi-squared test

data:  election$poll.diff and election$vote.diff
X-squared = 2550, df = 2500, p-value = 0.2383

# P-value
chisq.test(x = election$poll.diff, y = election$vote.diff)$p.value
[1] 0.2382758
```

For independence: The null hypothesis is that the errors for all 50 states plus D.C. is independent. The alternative hypothesis says that they are dependent. Because we have a large p-value (0.2383) that is > 0.01 we fail to reject the null hypothesis and believe the two differences are independent.

For poll and vote differences: The null hypothesis says there is an equally likely chance for the poll differences to be greater or less than the actual election difference (a normal distribution). Because we have a large p-value (0.2383) that is > 0.01 we fail to reject the null hypothesis and believe the distribution to be normal 99% of the time.

Question 4

part a

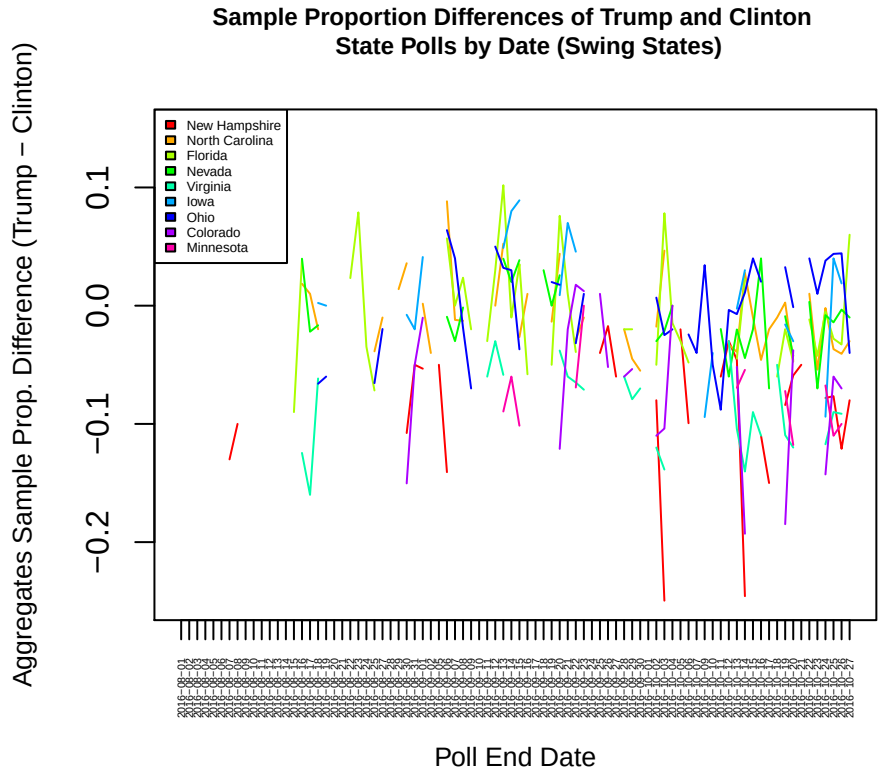
```
diff.state.date = subset(aug_poll, select = -c(grade))  # Remove unnecessary columns
diff.state.date$rawpoll_clinton = (diff.state.date$rawpoll_clinton/100) *
    diff.state.date$samplesize  # Transform to pop. from percentage
diff.state.date$rawpoll_trump = (diff.state.date$rawpoll_trump/100) *
    diff.state.date$samplesize  # Transform to pop. from percentage

diff.sample = aggregate(diff.state.date$samplesize,
    by = list(state = diff.state.date$state, end.date = diff.state.date$end.date),
    FUN = sum)  # Total sampled by state per day

diff.clinton = aggregate(diff.state.date$rawpoll_clinton,
    by = list(state = diff.state.date$state, end.date = diff.state.date$end.date),
    FUN = sum)  # Total clinton pollers by state per day

diff.trump = aggregate(diff.state.date$rawpoll_trump,
    by = list(state = diff.state.date$state, end.date = diff.state.date$end.date),
    FUN = sum)  # Total clinton pollers by state per day

diff.state.date = data.frame(diff.sample, diff.clinton,
    diff.trump)  # Combine aggregated samples by day into one dataframe

diff.state.date = subset(diff.state.date, select = c(state,
    x, x.1, x.2, end.date))  # Remove unnecessary cols

# Calculate aggregated data for sample prop. of
# trump - clinton for each day
diff.state.date$x = (diff.state.date$x.2/diff.state.date$x) -
    (diff.state.date$x.1/diff.state.date$x)

diff.state.date = subset(diff.state.date, select = -c(x.1,
    x.2))  # Remove unnecessary cols
# Reshape table to have dates as columns and states
# as rows
diff.state.date = as.matrix(reshape(diff.state.date,
    idvar = "state", timevar = "end.date", direction = "wide"))

# adjust row and column names
colnames(diff.state.date) = substring(colnames(diff.state.date),
    3)
```

```
colnames(diff.state.date)[1] = "state"
rownames(diff.state.date) = c(diff.state.date[, "state"])
diff.state.date = subset(diff.state.date, select = -c(state))

head(diff.state.date[, 1:5])
               2016-08-01 2016-08-02 2016-08-03 2016-08-04 2016-08-05
Arizona        "-0.030"   NA         NA         NA         " 0.02"
Kentucky       " 0.130"   NA         NA         " 0.16"    NA
Michigan       "-0.094"   NA         NA         "-0.11"    NA
New Hampshire  "-0.150"   NA         NA         NA         NA
Pennsylvania   "-0.130"   NA         NA         "-0.09"    NA
North Carolina NA         "0.04"     NA         NA         NA
```

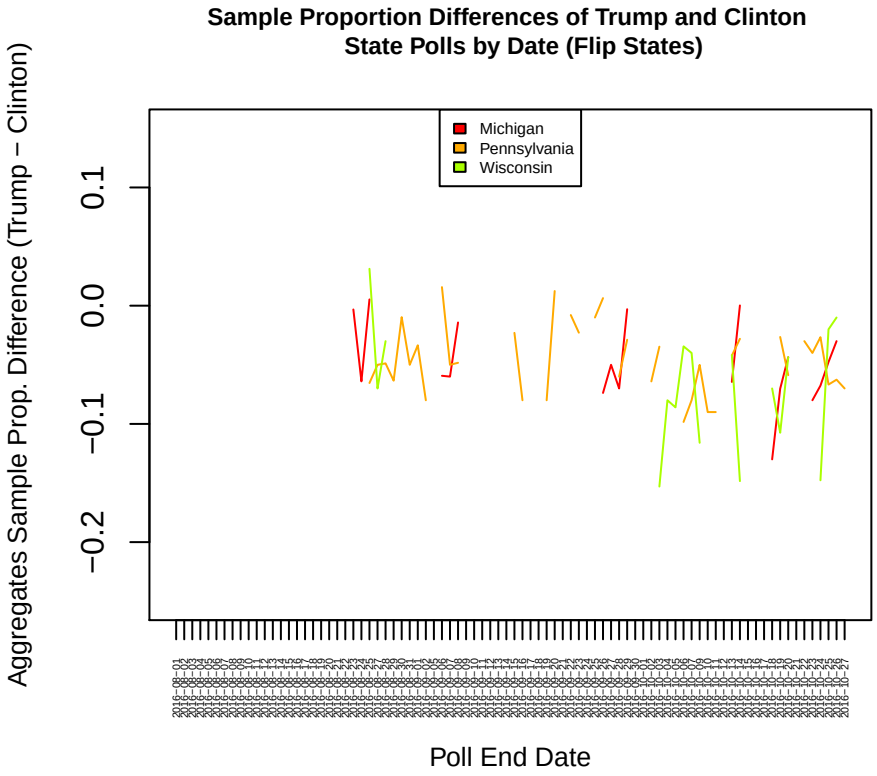
We followed a similar process to the past problems. We had to create multiple variables consisting of aggregating populations based on voting for Clinton or Trump by state and by day. We then combined this all into a single dataframe organized by columns of dates and rows of states.

part b

```
# Subset of swing states
swing = c("Colorado", "Florida", "Iowa", "Minnesota",
    "Ohio", "Nevada", "New Hampshire", "North Carolina",
    "Virginia")
swing.subset = subset(diff.state.date, rownames(diff.state.date) %in%
    swing)
col_set = rainbow(nrow(swing.subset))  # Rainbow colors

# Plot line graph with appropriate labels and
# titles
matplot(t(swing.subset), type = "l", ylim = c(-0.25,
    0.15), xaxt = "n", ylab = "Aggregates Sample Prop. Difference (Trump - Clinton)",
    xlab = "Poll End Date", main = "Sample Proportion Differences of Trump and Clinton
\t\tState Polls by Date (Swing States)",
    cex.lab = 0.8, cex.main = 0.75, lty = 1, col = col_set)
legend("topleft", cex = 0.4, "top", row.names(swing.subset),
    fill = col_set)
axis(1, at = seq(1, 84), labels = colnames(swing.subset),
    cex.axis = 0.3, las = 2)
```




```
# Flip states
flip.subset = subset(diff.state.date, rownames(diff.state.date) %in%
    c("Michigan", "Pennsylvania", "Wisconsin"))

# Plot line graph with appropriate labels and
# titles
matplot(t(flip.subset), type = "l", ylim = c(-0.25,
    0.15), xaxt = "n", ylab = "Aggregates Sample Prop. Difference (Trump - Clinton)",
    xlab = "Poll End Date", main = "Sample Proportion Differences of Trump and Clinton
\t\tState Polls by Date (Flip States)",
    cex.lab = 0.8, cex.main = 0.75, lty = 1, col = col_set)
legend("top", cex = 0.5, row.names(flip.subset), fill = col_set)
axis(1, at = seq(1, 84), labels = colnames(flip.subset),
    cex.axis = 0.3, las = 2)
```



part c

The flip states from 3c were Michigan, Pennsylvania, and Wisconsin. If we look at the second graph, the poll results of the flip states do appear to be slightly correlated and have a negative trend as the end dates become later in the year. The minimum poll results in the earlier dates never go below -0.1 but the poll results later in the dataset appear to be below 0.0 and sometimes even below -0.1.

If two states are correlated by poll results, we would expect to see their connected lines on the same date ranges to be trending upwards or downwards in the same range of poll results at the same times. That is what we observed with the graph visualizing the flip states as they trended downwards slightly. This is most evident as we view the maximum poll results for these 3 states is around +0.3 near the end of August and the max then goes no higher than 0.0 by the end of the polling season. For these reasons I would say that the flip states and their poll results are correlated in the negative direction.