# CharlesCoffey17835PSET1

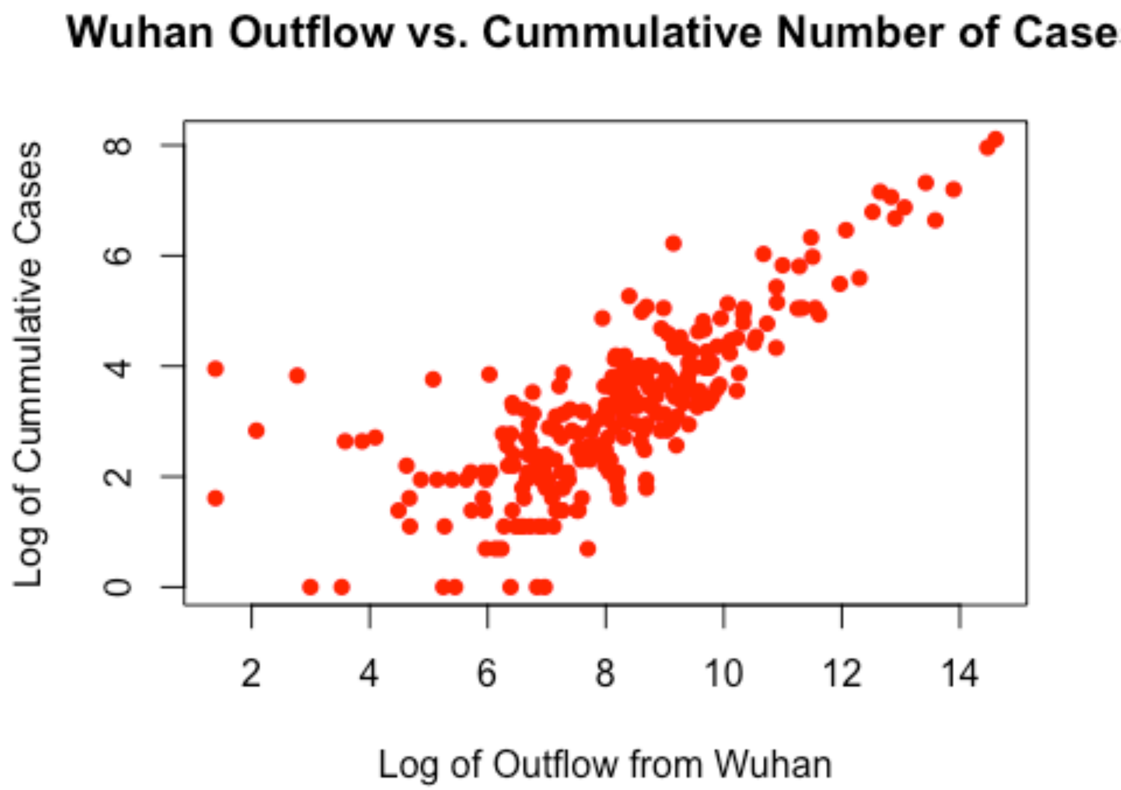
Charles Coffey

9/16/2020

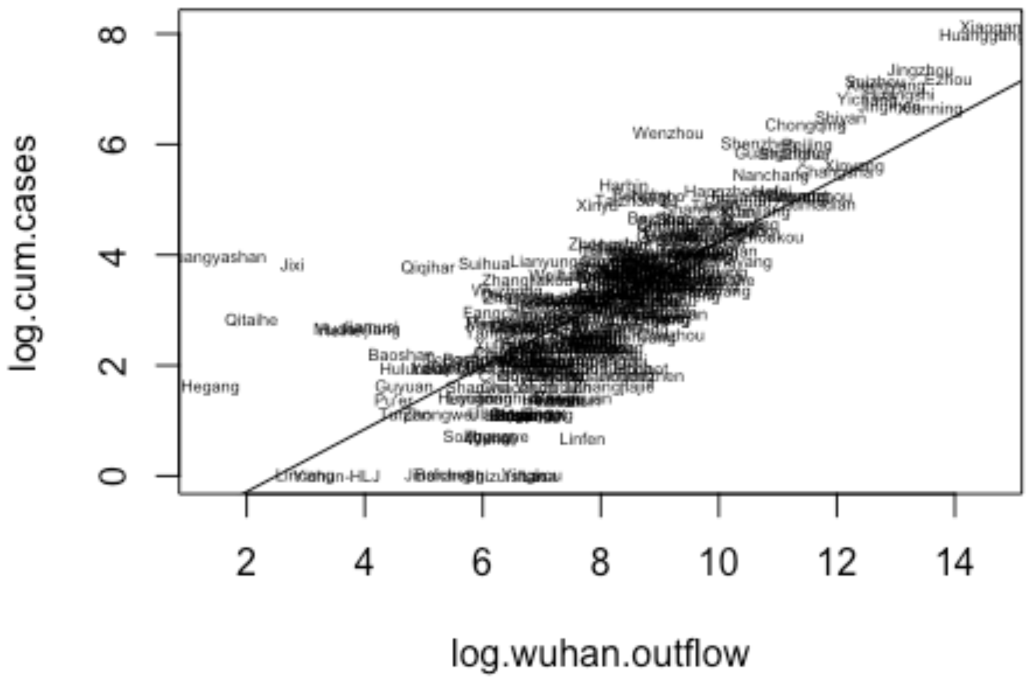
**QUESTION 1**

```
library(ISLR)
library(MASS)

#1.1 load csv files

cities_info = read.csv("cities_info.csv");
distance_to_wuhan = read.csv("distance_to_wuhan.csv");

#1.2 merged data set with all info
cities_all <- merge(cities_info, distance_to_wuhan, by = "city_name");

head(cities_all)

##   city_name cumulative_confirmed_cases.Feb19. population.10.thousand.
## 1    Ankang                                26                     305
## 2    Anqing                                83                     531
## 3    Anshan                                 4                     344
## 4    Anshun                                 4                     301
## 5    Anyang                                53                     624
## 6  Baicheng                                 1                     191
##   GDP.10.thousand.YUAN. latitude longitude distance_to_wuhan.km.
## 1               2909198    32.41    109.01               540.663
## 2               5138061    30.31    117.02               273.560
## 3               8640987    41.07    123.00              1432.243
## 4               4488895    26.14    105.55               964.994
## 5               6166974    36.06    114.21               634.934
## 6               1676260    45.38    122.50              1821.758
##   wuhan_outflow.Jan1.to.Jan24.
## 1                         4961
## 2                        36683
## 3                         1423
## 4                         1831
## 5                        17358
## 6                          232

#1.3 making the data easier to work with because it is skewed

log.wuhan.outflow <- log(cities_all$wuhan_outflow.Jan1.to.Jan24.);
log.cum.cases <- log(cities_all$cumulative_confirmed_cases.Feb19.);
```

```
#1.4 plotting outflow from wuhan vs number of cases

plot(log.wuhan.outflow, log.cum.cases, pch = 16, col = "red",
     main = "Wuhan Outflow vs. Cummulative Number of Cases",
     xlab = "Log of Outflow from Wuhan",
     ylab = "Log of Cummulative Cases");
```




```
#1.5 plot texts instead of points
plot(log.wuhan.outflow, log.cum.cases, type="n")
text(log.wuhan.outflow, log.cum.cases, cities_all$city_name, cex = 0.45)

abline(a=-1.4193, b=0.5662)
```




```
#1.6 checking for cities with potential to be seriously affected
cities.with.risk <- character();

for (row in 1:nrow(cities_all)){

  if ((cities_all[row, 3]*10^4 > 10^7) && (cities_all[row, 8] > 10^4)){

    cities.with.risk <- c(cities.with.risk, cities_all[row,1])
  }
}

print(cities.with.risk)

## [1] "Beijing"   "Chengdu"   "Chongqing" "Fuyang"    "Handan"    "Nanyang"
## [7] "Shanghai"  "Tianjin"   "Zhoukou"
```

**QUESTION 2**

```
#2.1
results_germany_2017 <- read.csv("results_germany_2017.csv", sep = ";");
print(results_germany_2017[c(3,5),])
```

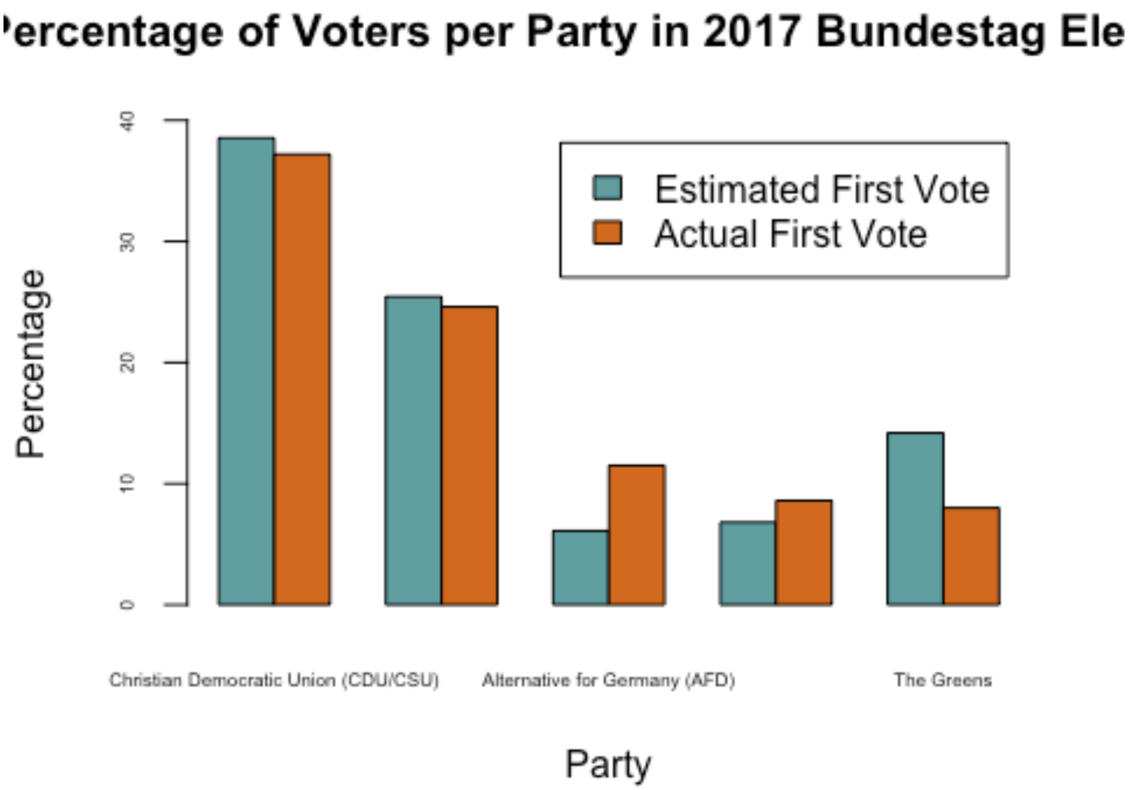
```
##                       party_name vote1_perc vote2_perc
## 3 Alternative for Germany (AFD)       11.5       12.6
## 5                     The Greens        8.0        8.9
```

Percentage of Votes of the AfD and The Greens in the First and Second Votes listed above.

```
#2.2
ESS_Germany_2018 <- read.csv("ESS_Germany_2018.csv");

  # initialize zero vectors for count number of votes for each party
num_votes_1 = integer(length(results_germany_2017$party_name))
num_votes_2 = integer(length(results_germany_2017$party_name))

  #create holding place for keeping track of votes for each party
party_vote_estimates = data.frame(num_votes_1, num_votes_2, row.names = resul
ts_germany_2017$party_name)

  #check to see which voters voted and which parties they voted for
for (respondent in 1:nrow(ESS_Germany_2018)){
  if ((ESS_Germany_2018[respondent, "vote"] == "Yes") && (!is.na(ESS_Germany_
2018[respondent, "prtvede1"]))){
    party_voted_1 = ESS_Germany_2018[respondent, "prtvede1"]
    party_vote_estimates[party_voted_1, "num_votes_1"] = party_vote_estimates
[party_voted_1, "num_votes_1"] + 1
  }
  if ((ESS_Germany_2018[respondent, "vote"] == "Yes") && (!is.na(ESS_Germany_
2018[respondent, "prtvede2"]))){
    party_voted_2 = ESS_Germany_2018[respondent, "prtvede2"]
    party_vote_estimates[party_voted_2, "num_votes_2"] = party_vote_estimates
[party_voted_2, "num_votes_2"] + 1
  }
}

  #find percentage of voters each party received
percentage_votes_1 = party_vote_estimates$num_votes_1/sum(party_vote_estimate
s$num_votes_1)*100
percentage_votes_2 = party_vote_estimates$num_votes_2/sum(party_vote_estimate
s$num_votes_2)*100
names(percentage_votes_1) = c("Percentage Voters 1")
names(percentage_votes_2) = c("Percentage Voters 2")

  #add percentage information to storage data frame
party_vote_estimates <- cbind(party_vote_estimates, percentage_votes_1, perce
ntage_votes_2)
```

```
  # calculated difference between expected outcome vs actual outcome
estimated_minus_result_1 = party_vote_estimates$percentage_votes_1 - results_
germany_2017$vote1_perc
estimated_minus_result_2 = party_vote_estimates$percentage_votes_2 - results_
germany_2017$vote2_perc

  # add differences
party_vote_estimates <- cbind(party_vote_estimates, estimated_minus_result_1,
estimated_minus_result_2)

  # create 7 column data table with all necessary information
data_table = cbind(results_germany_2017$party_name,
                   party_vote_estimates$percentage_votes_1,
                   results_germany_2017$vote1_perc,
                   party_vote_estimates$estimated_minus_result_1,
                   party_vote_estimates$percentage_votes_2,
                   results_germany_2017$vote2_perc,
                   party_vote_estimates$estimated_minus_result_2);

  # converting data table to data frame to be able to work with it
data_table = as.data.frame(data_table);
names(data_table) = c("Party Name", "Estimated First Vote", "Actual First Vot
e","Estimated1-Actual1",
                      "Estimated Second Vote", "Actual Second Vote", "Estimat
ed2-Actual2")
  #converting all numbers to numerics instead of characters
data_table$`Estimated First Vote` = as.numeric(data_table$`Estimated First Vo
te`)
data_table$`Actual First Vote` = as.numeric(data_table$`Actual First Vote`)
data_table$`Estimated1-Actual1` = as.numeric(data_table$`Estimated1-Actual1`)
data_table$`Estimated Second Vote` = as.numeric(data_table$`Estimated Second
Vote`)
data_table$`Actual Second Vote` = as.numeric(data_table$`Actual Second Vote`)
data_table$`Estimated2-Actual2` = as.numeric(data_table$`Estimated2-Actual2`)

  #order data table by most actual first votes
data_table[order(data_table$`Actual First Vote`),];

##                             Party Name Estimated First Vote Actual First V
ote
## 8      National Democratic Party (NPD)           0.06361323
0.1
## 7         Pirate Party (Piratenpartei)           0.06361323
0.2
## 9                                Other           2.60814249
2.9
## 6          Free Democratic Party (FDP)           6.17048346
7.0
## 5                           The Greens          14.18575064
```

```
8.0
## 4                 The Left (Die Linke)           6.80661578
8.6
## 3         Alternative for Germany (AFD)           6.10687023                1
1.5
## 2         Social Democratic Party (SPD)          25.44529262                2
4.6
## 1 Christian Democratic Union (CDU/CSU)          38.54961832                3
7.2
##   Estimated1-Actual1 Estimated Second Vote Actual Second Vote
## 8        -0.03638677             0.1229256                0.4
## 7        -0.13638677             0.2458513                0.4
## 9        -0.29185751             2.2126613                4.9
## 6        -0.82951654             9.0350338               10.7
## 5         6.18575064            17.7627535                8.9
## 4        -1.79338422             7.6828519                9.2
## 3        -5.39312977             6.8223725               12.6
## 2         0.84529262            21.8192993               20.5
## 1         1.34961832            34.2962508               33.0
##   Estimated2-Actual2
## 8         -0.2770744
## 7         -0.1541487
## 9         -2.6873387
## 6         -1.6649662
## 5          8.8627535
## 4         -1.5171481
## 3         -5.7776275
## 2          1.3192993
## 1          1.2962508

  #prepare data to be plotted and extract top 5 parties' information
plot_data = t(data_table[c(1:5), c(2:3)])
colnames(plot_data) <- c(data_table$`Party Name`[c(1:5)])

  #create bar plot for data
colors.names = c("cadetblue","chocolate")
par(cex.axis = 0.5)
barplot(as.matrix(plot_data),
        col = colors.names,
        beside = TRUE,
        legend=rownames(plot_data),
        xlab = "Party",
        ylab = "Percentage",
        ylim = c(0,40),
        main = "Percentage of Voters per Party in 2017 Bundestag Election")
```




```
print(data_table[c(1:5),])
```

```
##                               Party Name Estimated First Vote Actual First V
ote
## 1 Christian Democratic Union (CDU/CSU)            38.549618              3
7.2
## 2        Social Democratic Party (SPD)            25.445293              2
4.6
## 3        Alternative for Germany (AFD)             6.106870              1
1.5
## 4                  The Left (Die Linke)             6.806616
8.6
## 5                            The Greens            14.185751
8.0
##   Estimated1-Actual1 Estimated Second Vote Actual Second Vote
## 1          1.3496183             34.296251               33.0
## 2          0.8452926             21.819299               20.5
## 3         -5.3931298              6.822372               12.6
## 4         -1.7933842              7.682852                9.2
## 5          6.1857506             17.762754                8.9
##   Estimated2-Actual2
## 1           1.296251
## 2           1.319299
```

```
## 3          -5.777628
## 4          -1.517148
## 5           8.862754
```

CONCLUSION: The Alternative for Germany party and The Left Party gain from people lying because it is evident that they receive more voters than their survey results suggest that they would. People seem to feel ashamed that they vote for these two parties. They get more votes than they actually expect. AFD gains the most from this phenomenon. I do not think the data really suggests that German progressive voters are lazy. Parties that had less actual voters than estimated voters have a very small margin of error. This margin is not sufficient enough data to conclude that the progressive voters are lazy, except for the The Greens who had a significant margin of error. The differences seen with the parties that have higher actual voters than estimated voters suggest that these voters may perhaps be shy.

By the problem set's definition of "benefit", the Christian Democratic Union, Social Democratic Party, and the Greens benefit from people lying because they seem more popular in the polls than they truly are once votes are casted.

## QUESTION 3

```
#3.1
library(foreign)
LupPon_data <- read.dta("LupPon_APSR.dta");

country_names <- unique(LupPon_data$country);
years <- unique(LupPon_data$year);

redist <- na.omit(LupPon_data$redist)
ratio9050 <- na.omit(LupPon_data$ratio9050)
ratio5010 <- na.omit(LupPon_data$ratio5010)
country_3_obs_count <- 0; #count of countries with all 3 observations

for (row in 1:nrow(LupPon_data)){

  if (all(!is.na(LupPon_data[row, c("redist","ratio9050","ratio5010")]))){
    country_3_obs_count = country_3_obs_count + 1;
  }
}
print(paste("There are",toString(length(country_names)),"countries in the dat
aset.", sep = " "))

## [1] "There are 19 countries in the dataset."

print(paste("There are",toString(length(country_names)),"years in the dataset
.", sep = " "))

## [1] "There are 19 years in the dataset."

print("Years:")

## [1] "Years:"

print(years)

##  [1] 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973
1974
## [16] 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988
1989
## [31] 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003
2004
## [46] 1958 1959 2005

print(paste("There are",toString(country_3_obs_count),"country-year with all
three variables in the dataset.", sep = " "))

## [1] "There are 67 country-year with all three variables in the dataset."

#3.2

top_ordered9050 <- LupPon_data[order(LupPon_data$ratio9050, decreasing = TRUE
```

```r
)[c(1:5)], c("country", "ratio9050")]

bottom_ordered9050 <- LupPon_data[order(LupPon_data$ratio9050, decreasing = F
ALSE)[c(1:5)], c("country", "ratio9050")]

print(top_ordered9050)
```

```
##     country ratio9050
## 857     USA      2.29
## 858     USA      2.29
## 855     USA      2.28
## 856     USA      2.26
## 852     USA      2.21
```

```r
print(bottom_ordered9050)
```

```
##     country ratio9050
## 618  Norway      1.42
## 619  Norway      1.42
## 620  Norway      1.42
## 621  Norway      1.42
## 622  Norway      1.42
```

```r
#3.3

top_ordered5010 <- LupPon_data[order(LupPon_data$ratio5010, decreasing = TRUE
)[c(1:5)], c("country", "ratio5010")]

bottom_ordered5010 <- LupPon_data[order(LupPon_data$ratio5010, decreasing = F
ALSE)[c(1:5)], c("country", "ratio5010")]
print(top_ordered5010)
```

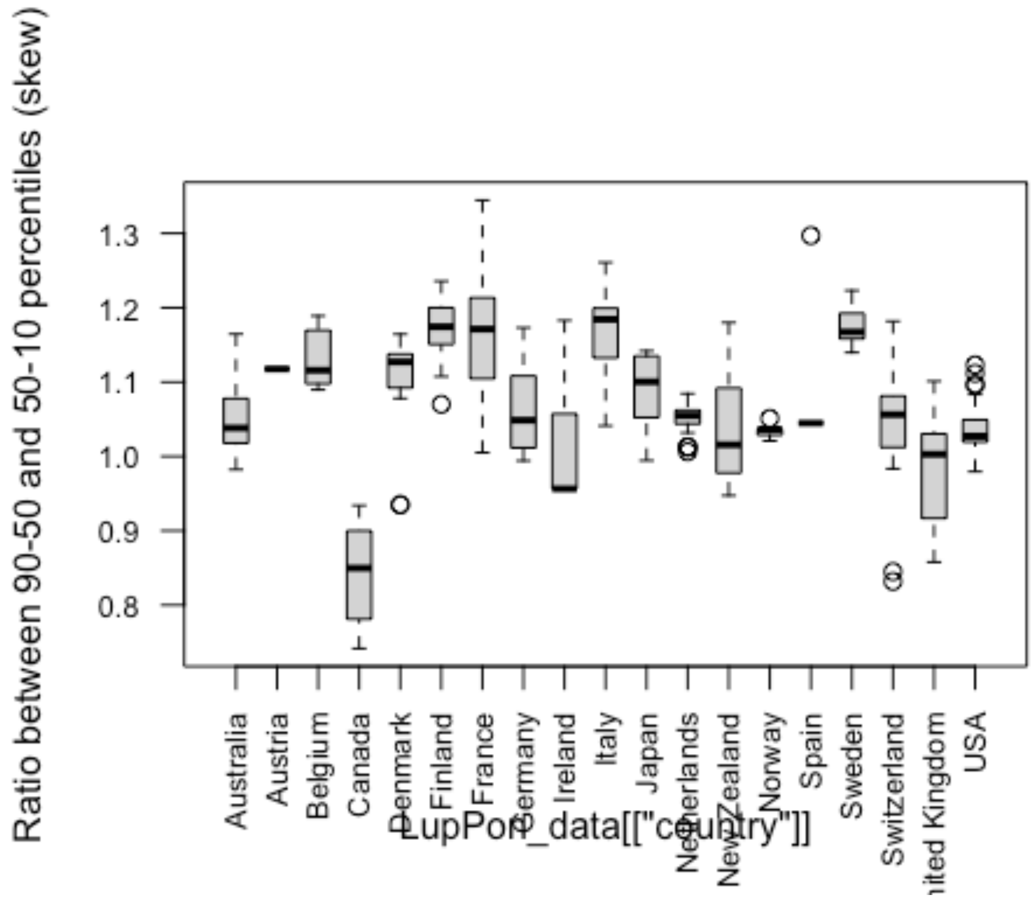
```
##     country ratio5010
## 162  Canada      2.43
## 143  Canada      2.40
## 164  Canada      2.39
## 166  Canada      2.38
## 168  Canada      2.33
```

```r
print(bottom_ordered5010)
```

```
##     country ratio5010
## 132 Belgium      1.27
## 133 Belgium      1.30
## 698  Sweden      1.30
## 701  Sweden      1.30
## 703  Sweden      1.30
```

Looking at this measure of inequality changes our results. This measure of inequality focuses on the divide between the middle and lower classes.

```r
#3.4

count1 = 0
count2 = 0

  # get rid of country-year pairs that have incomplete data
LupPon_data = LupPon_data[complete.cases(LupPon_data[c('ratio9050','ratio5010
')]),]

skew = c()
  # calculate ratios for each country-year pair
for (row in 1:nrow(LupPon_data)){
  # variable for 9050/5010
  r9overr5 = LupPon_data[row, "ratio9050"]/LupPon_data[row, "ratio5010"]

  skew = append(skew, r9overr5)

  if(r9overr5 > 1){
    count1 = count1 + 1
  }
  else{
    count2 = count2 + 1
  }
}

  #add skew to original data frame
LupPon_data = cbind(LupPon_data, skew)

  #box plot data
par(las = 2)
par(cex.axis=0.8)
boxplot(LupPon_data$skew ~ LupPon_data[["country"]], boxwex=0.6, ylab="Ratio
between 90-50 and 50-10 percentiles (skew)")
```




```
print(paste("There are", toString(count1),"country-year observations with ske
w>1.", sep = " "))

## [1] "There are 333 country-year observations with skew>1."

print(paste("There are", toString(count2),"country-year observations with ske
w<1.", sep= " "))

## [1] "There are 72 country-year observations with skew<1."
```

France has the highest skew and Canada has the lowest skew. Skews below one indicate a large amount of income disparity while those closer to one indicate less income disparity between the upper and lower classes.