Multinomial CI Prediction for 2020 Election

Shanghao Zhong

2020-11-06

Load package

```
library(jsonlite)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(MultinomialCI)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

You also need to load the data. You should have the `all.state.dataset`.

```
load("MultinomialCIProject2020.RData")
all.state.dataset <- get('all.state.dataset')
```

Predifine function

The script concerns about the probability of a vote cast for a specific candidate. Assuming the votes follow a multinomial distribution, it uses the `MultinomialCI` package to calculate the confidence interval of the probability for each candidate.

```
#' Calculate county level multinomial c.i.
#'
#' @param df a data frame with vote counts.
#'           row.name is county name,
#'           and each column is the number of votes for each candidate.
#' @param alpha The significance level for the confidence intervals.
#'              Must be a real number in the interval [0, 1]
#' @return a data frame of confidence intervals for each candidate
#' @export
```

```r
CI.by.county <- function(df, alpha) {
  df <- filter(df, df[[1]] > 0)
  candidate.names <- names(df)
  county <- row.names(df)
  mat <- t(apply(df, 1, as.numeric))

  CI.per.county <- function(vec) {
    CI.per.county <- c(multinomialCI(pmax(vec, 0), alpha))
  }
  cols <- apply(expand.grid(candidate.names, c('low', 'high')), 1, paste, collapse='.')
  all.ci <- t(apply(mat, 1, CI.per.county))

  output <- data.frame(all.ci, row.names = county)
  colnames(output) <- cols

  return(output)
}
```

With the probability of a vote cast for a candidate, the final projection is done by $np \pm z \cdot \sqrt{n \cdot p \cdot (1-p)}$, where p is the probability of a candidate winning a vote, n is the expected remaining vote according to NYT.

```r
#' Calculate the lower and the upper bound of the remaining votes given the
#' probability.
#'
#' @param remaining A scalar or vector containing the remaining votes in the
#'                  county.
#' @param prob A scalar or vector containing the probability of votes going to
#'             the candidate in the county.
#' @param alpha confidence level
#'
#' @return a length 2 vector, first is the lower bound and the second is the
#'         upper.
#' @export
range_est <- function(remaining, prob, alpha=0.95) {
  lo <- remaining*prob + qnorm((1-alpha)/2)*sqrt(remaining*prob*(1-prob))
  hi <- remaining*prob - qnorm((1-alpha)/2)*sqrt(remaining*prob*(1-prob))
  return(c(sum(lo, na.rm = TRUE), sum(hi, na.rm = TRUE)))
}
```

```r
#' Helper function to convert UTC time string to EST
#'
#' @param timestring string representation of time, NYT's time in ISO format
#'
#' @return a POSIXct object in EST
#' @export
#'
#' @examples to_EST('2020-11-07T01:46:10Z')
to.EST <- function(timestring) {
  with_tz(parse_date_time(timestring, 'ymd HMS'), 'EST')
}
```

Pull in data from NYT's api

Need to change the `current.state.name` — state name used to pull data from NYT's api.

It should be spelled out and all lower case and whitespace should be replace by `-`.

e.g.

- New York: `new-york`
- Pennsylvania: `pennsylvania`
- Distinct of Columbia: `district-of-columbia`

It will store all the pulled data in `all.state.dataset`. Only new data will be stored.

```
# This script will load data for the following state.
current.state.name <- 'arizona'

nyt.api <- paste(
  'https://static01.nyt.com/elections-assets/2020/data/api/2020-11-03/race-page/',
  current.state.name,
  '/president.json',
  sep = '')

results <- fromJSON(nyt.api)
current.update.time <- max(results$data$races$counties[[1]]$last_updated)

# if the state haven't been track yet, create a slot for that.
if (!current.state.name %in% names(all.state.dataset)) {
  update.type <- "New data"
  all.state.dataset[[current.state.name]] <- list()
  all.state.dataset[[current.state.name]][[current.update.time]] <- results$data$races$counties[[1]]
  previous.update.time <- 'N/A'
} else { # if the state has been track, see if this one is new update
  previous.update.time <- max(last(all.state.dataset[[current.state.name]])
                              $last_updated)
  if (current.update.time == previous.update.time) {
    update.type <- "No update"
  } else {
    update.type <- "New update"
    all.state.dataset[[current.state.name]][[current.update.time]] <- results$data$races$counties[[1]]
  }
}

rm(results)

by.county <- last(all.state.dataset[[current.state.name]])
old.by.county <- nth(all.state.dataset[[current.state.name]],
                     max(1, length(all.state.dataset[[current.state.name]])-1))


all.votes <- data.frame(by.county$results,
                        row.names = by.county$name)
old.all.votes <- data.frame(old.by.county$results,
                            row.names = old.by.county$name)

mail.votes <- data.frame(by.county$results_absentee,
                         row.names = by.county$name)
old.mail.votes <- data.frame(old.by.county$results_absentee,
                             row.names = old.by.county$name)
```

```
cat(paste(update.type, 'in', current.state.name),
    paste("Previous Update:",
          to.EST(max(old.by.county$last_updated))),
    paste("Current Update:",
          to.EST(max(by.county$last_updated))),
    paste("Previous margin:",
          sum(old.all.votes$bidenj) - sum(old.all.votes$trumpd)),
    paste("Current margin:",
          sum(all.votes$bidenj) - sum(all.votes$trumpd)),
    sep='\n')
```

```
## No update in arizona
## Previous Update: 2020-11-06 21:01:47
## Current Update: 2020-11-06 21:04:35
## Previous margin: 29861
## Current margin: 29861
```

See the number of snapshots saved

```
sapply(all.state.dataset, length)
```

```
##      arizona pennsylvania       nevada      georgia       alaska
##            5           11            4            6            1
```

See the time at which snapshots were taken (in EST)

```
to.EST(names(all.state.dataset[[current.state.name]]))
```

```
## [1] "2020-11-06 11:09:33 EST" "2020-11-06 15:27:34 EST"
## [3] "2020-11-06 20:25:37 EST" "2020-11-06 21:01:47 EST"
## [5] "2020-11-06 21:04:35 EST"
```

Estimate the probability for each candidate

You can change `using` to update how you want to estimate the probability. Unhide one of them and hide to other to use.

first option:

- Use the difference between old data and new data
- Best for predicting the most recent trend
- Doesn't work if the difference between old data and new data is small or non-representative

second option:

- Use the mail.votes to predict old data and new data
- Work the best if the mail votes is homogeneous throughout different time
- Doesn't work But the demographics within mail data can change over time

When `using` is incomplete, the probability of each candidate will based on `all.votes`

```
# first option
using <- data.frame(
  data.matrix(last(all.state.dataset[[current.state.name]])$results)
  - data.matrix(nth(all.state.dataset[[current.state.name]], 1)$results),
  row.names = last(all.state.dataset[[current.state.name]])$name)

# second option
```

```r
# using <- mail.votes

remaining <- data.frame(exp.remaining = pmax(0, by.county$tot_exp_vote - rowSums(data.matrix(all.votes)),
                        row.names = by.county$name)
ci.mail <- merge(remaining, CI.by.county(using, 0.95), by=0)
ci.other <- merge(remaining, CI.by.county(all.votes, 0.95), by=0)
ci.other <- ci.other[ci.other$Row.names %in%
                       setdiff(ci.other$Row.names, ci.mail$Row.names), ]
est <- rbind(ci.mail, ci.other)
rm(remaining)

est <- est[order(est$Row.names), ]
rownames(est) <- est$Row.names
est$Row.names <- NULL
est[est$exp.remaining > 0, ]
```

```
##            exp.remaining bidenj.low trumpd.low jorgensenj.low write.ins.low
## Apache              9667  0.6807217  0.3074426     0.01060288  0.0000000000
## Cochise             6619  0.3402394  0.6233886     0.03222836  0.0000000000
## Coconino             890  0.4689542  0.5000000     0.02614379  0.0000000000
## La Paz               323  0.3062753  0.6794968     0.01108282  0.0001497679
## Maricopa           91437  0.3953649  0.4829649     0.02604698  0.0934567668
## Mohave              2164  0.1743745  0.8003538     0.02223907  0.0000000000
## Navajo              5536  0.3179027  0.6565895     0.02125650  0.0000000000
## Pima               25936  0.4366584  0.4743104     0.02810825  0.0546476664
## Pinal              33500  0.4015902  0.5793579     0.01320132  0.0022502250
## Santa Cruz          1454  0.6707766  0.3156144     0.01089737  0.0006650977
## Yuma                3855  0.4859678  0.4810153     0.02930252  0.0000000000
##            bidenj.high trumpd.high jorgensenj.high write.ins.high
## Apache       0.6816159   0.3083368      0.01149715   0.0004832989
## Cochise      0.3438750   0.6270242      0.03586394   0.0022543675
## Coconino     0.4747368   0.5057826      0.03192639   0.0041486084
## La Paz       0.3080035   0.6812250      0.01281100   0.0018779504
## Maricopa     0.3964692   0.4840692      0.02715123   0.0945610212
## Mohave       0.1766075   0.8025868      0.02447208   0.0012221448
## Navajo       0.3211672   0.6598541      0.02452105   0.0018474520
## Pima         0.4399260   0.4775780      0.03137591   0.0579153239
## Pinal        0.4036230   0.5813908      0.01523417   0.0042830783
## Santa Cruz   0.6718347   0.3166725      0.01195543   0.0017231596
## Yuma         0.4885116   0.4835590      0.03184627   0.0013056215
```

Estimate the final range

With the lower and upper bound of Biden's and Trump's probability in a county, we use both probability to calculate the CI. The lower end of the CI from the low probability is a candidate's lower bound, while the upper end of the CI from the high probility is a candidate's upper bound. We then see the margin using Biden's lower bound − Trump's upper bound, and using Biden's upper bound − Trump's lower bound, to calculate the final projection interval.

```r
future.lo <- range_est(est$exp.remaining, est$bidenj.low)[1] - range_est(est$exp.remaining, est$trumpd.h
future.hi <- range_est(est$exp.remaining, est$bidenj.high)[2] - range_est(est$exp.remaining, est$trumpd
current.diff <- sum(all.votes$bidenj) - sum(all.votes$trumpd)
current.diff + c(future.lo, future.hi)
```

```
## [1] 11409.97 16214.20
```