Prediction Competition - 2019 Governor Races

Alec MacMillen

10/23/2019

Overview

The goal of this project is to predict the two-party vote shares in the 2019 gubernatorial elections in Kentucky, Louisiana, and Mississippi. These are off-year elections that take place largely divorced from national elections in presidential and midterm years. In Kentucky, incumbent Republican Matt Bevin is running for reelection against Democratic challenger Andy Beshear (whose father, Steve Beshear, was governor prior to Bevin). In Louisiana, incumbent Democrat John Bel Edwards is running for reelection against Republican challenger Eddie Rispone (both of whom were the top-two vote getters in the October 12 jungle primary, advancing to the general election on November 16). In Mississippi, incumbent Republican Governor Phil Bryant is term-limited, so Democrat Jim Hood and Republican Tate Reeves are running for the open seat. The Kentucky and Mississippi elections will take place on November 5, while the Louisiana election will take place on November 16.

Broadly speaking, the approach I'm taking blends two separate models: a polls-only model, and a demographic model. The polls-only model looks at the predictive power of polls in previous elections and applies those conclusions to polls conducted this cycle. The demographic model uses data from the Cooperative Congressional Election Study (CCES) to look at the composition of the electorate and Democratic support therein in the last two major off-cycle elections in these states: first, the 2015 election (the last gubernatorial) and second, the 2018 election (the last midterm). The 2015 election offers a direct comparison with the *circumstances* of this year's elections, while the 2018 election offers more of an insight into the current *mood* of the electorate.

Model 1: Polls-only

First we'll build the polls-only model. The data for this model comes from FiveThirtyEight's database of historical polls from local, state, and federal elections dating back to 1998 and is available from FiveThirtyEight's data repository on their GitHub page. The broad strategy is to aggregate all polls by specific race to create a single weighted average prediction for each race. This weighted average is constructed by weighting a poll's results more heavily for polls that have larger sample sizes and dates closer to the election, and less heavily for smaller sample-size polls that are farther from the election. Then, these weighted predictions will be merged with actual election results and a linear model will be trained to assess the predictive power of these weighted predictions. These predictions will then be applied to the weighted average of current-year polls to come up with a polls-only prediction of two-party vote shares.

Part 1: Cleaning prior-year polling data

Start by loading the raw FiveThirtyEight polling data and performing some basic transformations, including:

- Converting the poll date and election date to date variables,
- Calculating the number of days between the median date in field and election date,
- Limiting to polls for gubernatorial and senatorial races (i.e. statewide elections),
- Dropping races where the two primary candidates were not one Democrat and one Republican,
- Calculating two-party vote shares in the poll and final election,
- Selecting only relevant variables.

It's also worth noting that I manually added gubernatorial and senatorial polls from the 2018 cycle, which were omitted.

Now we'll create a `results` table with one row for each election (because in the raw data, the results data is replicated on every row for a poll regarding the corresponding election).

```
results <- all_polls_int %>%
  group_by(year, race, location, type_simple) %>%
  summarize(DemShareAct = mean(DemShareAct),
            RepShareAct = mean(RepShareAct))
```

Next, to come up with a weighted average of polls on a race-by-race basis, we'll calculate a "total sample size" variable \texttt{ss_total} that is merely the sum of the sample sizes for all polls associated with a given election so that we can weight polls with largely sample sizes more heavily.

```
weights <- all_polls_int %>%
  group_by(race) %>%
  summarize(ss_total = sum(samplesize))
```

Now we'll calculated a weighted mean of all poll predictions (weighting directly on sample size and inversely on time to election) to come up with one single prediction for each election according to all the polls that were taken for it. We'll also output this summary table of all poll predictions for prior races into an intermediate output file \texttt{prior_year_polls.csv} for ease of access moving forward.

```
all_polls_final <- all_polls_int %>%
  left_join(weights, by = "race") %>%
  # Create weights for sample size (greater ss = greater weight),
  # days to election (more days to election = lower weight)
  mutate(sswt = samplesize / ss_total,
         dayswt = 1 / daystoelection,

         # Apply weights to polls
         DemSharePollWt = DemSharePoll*sswt*dayswt,
         RepSharePollWt = RepSharePoll*sswt*dayswt) %>%
  group_by(year, race, location, type_simple) %>%

  # Average raw predictions
  summarize(DemPredRaw = mean(DemSharePollWt),
            RepPredRaw = mean(RepSharePollWt)) %>%
  ungroup() %>%

  # Convertaverage  raw predictions into two-party vote shares
  mutate(DemPredFinal = DemPredRaw / (DemPredRaw + RepPredRaw),
         RepPredFinal = RepPredRaw / (DemPredRaw + RepPredRaw)) %>%
  select(year, race, location, type_simple, DemPredFinal, RepPredFinal) %>%

  # Merge onto actual results
  left_join(results, by = c("year", "race", "location", "type_simple")) %>%
  mutate(idx = 1:n()) %>%
  select(idx, everything()) %>%
  filter(!is.na(DemPredFinal) & !is.na(DemShareAct))

write_csv(all_polls_final, "../output/prior_year_polls.csv")
```

Part 2: Cleaning current-year polling data

FiveThirtyEight also has current polling data for 2019 races. Unfortunately, this data is in slightly different form to the historical polling database, so the cleaning process is slightly different and we will have to take a few extra steps to standardize. (Also worth noting that I manually added rows for the most recent few polls because the FiveThirtyEight data was not completely up to date.)

Now we will take several data cleaning steps, including:

- Filter to include only KY, LA, and MS races (the only ones we care about),
- Drop non-major party candidates,
- Collapse party percentage support in a poll to a single row (currently, a single poll takes up multiple rows in the dataset),
- Calculate the two-way vote share, which implicitly omits undecideds (we will address this by weighting polls closer to election day more heavily)
- Convert date strings into proper date variables, date each poll by its median date in the field and use that variable to calculate time to election,
- Select final relevant variables.

```
gov_polls_int <- gov_polls_raw %>%
  # Keep only relevant variables
  select(poll_id, question_id, pollster_id, cycle, state, sample_size, population,
         office_type, start_date, end_date, stage, answer, candidate_name,
         candidate_party, pct) %>%

  # Filter to only LA, KY, MS races in 2019
  filter(state %in% c("Kentucky", "Louisiana", "Mississippi"),
         cycle == 2019) %>%

  # Drop non-major party candidates
  filter(candidate_party %in% c("DEM", "REP")) %>%

  # Get party percentage support in a single row
  spread(key = candidate_party, value = pct) %>%
  select(-c("stage", "answer", "candidate_name")) %>%
  group_by(poll_id, question_id, pollster_id, cycle, state, sample_size, population,
           office_type, start_date, end_date) %>%
  summarize(DemSharePoll = max(DEM, na.rm = TRUE),
            RepSharePoll = max(REP, na.rm = TRUE)) %>%
  ungroup() %>%

  # Calculate two-way vote share (this omits undecideds, account for this by
  # weighting polls closer to election day more heavily)
  mutate(Dem2WayShare = DemSharePoll / (DemSharePoll + RepSharePoll),
         Rep2WayShare = RepSharePoll / (DemSharePoll + RepSharePoll),
         samplesize = sample_size) %>%

  # Convert date strings into datetime vars, create election date vars for each election
  mutate(start_date = mdy(start_date),
         end_date = mdy(end_date),

         # Date the poll to its median date in the field assign election date to each
         # election and find # of days until election
         polldate = start_date - days(ceiling(as.numeric(end_date - start_date)/2)),
```

```
        electiondate = as.Date(ifelse(state == "Louisiana", "11/16/2019", "11/5/2019"), "%m/%d/%Y"),
        daystoelection = as.numeric(electiondate - polldate)) %>%

  # Final select for relevant variables
  select(state, samplesize, population, daystoelection, Dem2WayShare, Rep2WayShare)
```

Now we can follow the same weighted mean process that we performed for the prior-year polls to come up with an aggregated prediction for each of the three races we're predicting:

```
# Create total sample size of all polls by race to create samplesize weight
weights <- gov_polls_int %>%
  group_by(state) %>%
  summarize(ss_total = sum(samplesize))

# By race, create a single weighted prediction of vote share using sample size and
# days to election to combine all polls
gov_polls_final <- gov_polls_int %>%
  left_join(weights, by = "state") %>%

  # Create "weights" for sample size (greater ss = greater weight),
  # days to election (more days to election = lower weight)
  mutate(sswt = samplesize / ss_total,
         dayswt = 1 / daystoelection,

         # Apply weights
         DemSharePollWt = Dem2WayShare*sswt*dayswt,
         RepSharePollWt = Rep2WayShare*sswt*dayswt) %>%
  group_by(state) %>%
  summarize(DemPredRaw = sum(DemSharePollWt),
            RepPredRaw = sum(RepSharePollWt)) %>%

  # Convert raw predictions into two-party vote shares
  mutate(DemPredFinal = DemPredRaw / (DemPredRaw + RepPredRaw),
         RepPredFinal = RepPredRaw / (DemPredRaw + RepPredRaw))

# Merge with common_cols file to keep columns aligned across each dataset
gov_polls_output <- common_cols %>%
  left_join(gov_polls_final, by = "state") %>%
  select(-c("state", "DemPredRaw", "RepPredRaw")) %>%
  mutate(DemShareAct = NA, RepShareAct = NA)
```

Part 3: Regression models

Now we'll use OLS regression and a leave-one-out cross validation strategy, along with several different combinations of possible filters on the data, to estimate how predictive prior-year polls are of final results. The \texttt{filter_strings} vector contains filters on the data that limit the overall poll data to subsets that might be more representative, and therefore more predictive, of the races we're predicting. These filters include:

- Keeping only polls included in states that fall into the Census Bureau's definition of "the South" (which includes all three of KY, LA and MS),
- Keeping only polls conducted during the 2010 midterms forward,

- Keeping only polls conducted in gubernatorial elections,
- And every possible combination of these three filters, including a null filter (keeping all polls).

For each of the 8 filters, we'll apply a LOOCV (leave-one-out cross validation) regression strategy that progressively drops each individual observation from the dataset and finds the line of best fit that best predicts this holdout set. The very simple regression takes the following form:

$$actual\ voteshare_r = \beta_0 + \beta_1 predicted\ voteshare_r + \varepsilon$$

It might make sense to incorporate more control variables (like the state of the economy, statewide mood towards the two major parties, etc.) and that would be an area of exploration for future models. Finally, once all of the models are trained and assessed, we use them to predict the Democratic share of the two-party vote in the current cycle.

This is the final prediction of the polls-only model:

```
gov_polls_pred
```

```
## # A tibble: 3 x 3
##    year location meanPredDemVoteSh
##   <dbl> <chr>                <dbl>
## 1  2019 KY                   0.510
## 2  2019 LA                   0.593
## 3  2019 MS                   0.478
```

Model 2: Demographic and historical

Background

In this model, I will take data from the 2015 and 2018 CCES (the last two representative off-year election cycles) and look at the share of Democratic support by demograhic group, as well as how the electorate was composed, to predict how the electorate will look in 2019.

The 2015 and 2018 CCES data are in different formats so they will require different cleaning approaches. The broad steps are the same:

- Filter the data to the state in question, dropping observations with null weights
- Limit to observations where the individual supported one of the two major party candidates (for the gubernatorial election in 2015 and US House elections in 2018),
- Recode categorical demographic variables as factors,
- Apply survey weights to estimate true population values.

A few notes: the weights in 2015 and 2018 are slightly different - the 2015 weights tether the sample to the population of all adults, while the 2018 weights tether the sample to validated voters (i.e. those that actual voted). Because there were no Senate races in 2018 in any of the three states at issue, we approximate electorate support using US House elections - this may not be perfectly representative as local and statewide elections may have different dynamics in play. In both cases, candidate support refers to expressed *actual* support in the election.

```r
design15 <- function(fips) {
  state <- cces2015 %>% filter(inputstate == fips & !is.na(weight))

  small <- state %>%
    select(weight, birthyr:race, inputstate, starts_with("CC15_316")) %>%
    # Filter to two-party preference in governor election
    filter(CC15_316a %in% c(1, 2) | CC15_316b %in% c(1, 2) | CC15_316c %in% c(1, 2)) %>%
    # Recode demographic categorical vars as factors
    mutate(gender = recode(as.factor(gender), `1` = 0L, `2` = 1L),
           race = recode(as.factor(race), `1` = 1L, `2` = 2L, `3` = 3L, `4` = 4L, `5` = 5L,
                         `6` = 5L, `7` = 5L, `8` = 5L, .default = 5L),
           educ = recode(as.factor(educ), `1` = 1L, `2` = 2L, `3` = 3L, `4` = 4L, `5` = 5L, .default =
           age = as.factor(cut(2015 - as.numeric(birthyr), breaks = c(17, 35, 50, 65, 110))))

  # Recode categorical vars for party support
  if (fips == 21) {
    small <- small %>%
      mutate(votedem = recode(as.factor(
        ifelse(CC15_316a == 2, 1, 0)), `0` = 0L, `1` = 1L))
  } else if (fips == 22) {
    small <- small %>%
      mutate(votedem = recode(as.factor(
        ifelse(CC15_316b == 1, 1, 0)), `0` = 0L, `1` = 1L))
  } else if (fips == 28) {
    small <- small %>%
      mutate(votedem = recode(as.factor(
        ifelse(CC15_316c == 2, 1, 0)), `0` = 0L, `1` = 1L))
  }

  # Select only relevant vars
  small <- small %>% select(weight, gender, race, educ, age, votedem)

  design <- svydesign(id = ~0, weights = ~weight, data = small)
}
```

```r
design18 <- function(fips) {
  state <- cces2018 %>% filter(inputstate == fips & !is.na(vvweight_post))

  small <- state %>%
    select(caseid:race, CC18_412, HouseCand1Party_post, HouseCand2Party_post) %>%
    # Filter to two-party verified vote
    filter(CC18_412 %in% c(1, 2)) %>%
    # Recode categorical vars as factors
    mutate(gender = recode(gender, `1` = 0L, `2` = 1L),
           race = recode(race, `1` = 1L, `2` = 2L, `3` = 3L, `4` = 4L, `5` = 5L,
                         `6` = 5L, `7` = 5L, `8` = 5L, .default = 5L),
           educ = recode(educ, `1` = 1L, `2` = 2L, `3` = 3L, `4` = 4L, `5` = 5L, .default = 1L),
           age = as.factor(cut(2018 - birthyr, breaks = c(17, 35, 50, 65, 110))),
           votedem = recode(ifelse(CC18_412 == 1, 1, 0), `0` = 0L, `1` = 1L)) %>%
    select(vvweight_post, gender, race, educ, age, votedem)

  design <- svydesign(id = ~0, weights = ~vvweight_post, data = small)
}
```

Now with the functions designed, we can create our modeled electorates for each state in the two cycles, and check our estimation of the created `votedem` variable against actual election votes.

```
# 2015
ky15 <- design15(21)
la15 <- design15(22)
ms15 <- design15(28)

# 2018
ky18 <- design18(21)
la18 <- design18(22)
ms18 <- design18(28)

# Quick check to see how closely these vote predictions from CCES match actual totals (2015)
svymean(~votedem, ky15, na.rm=TRUE) # Predicted 47.03% Dem 2party Governor vote share, actual 45.48%
```

```
##            mean     SE
## votedem 0.47029 0.0523
```

```
svymean(~votedem, la15, na.rm=TRUE) # Predicted 56.87% Dem 2party Governor vote share, actual 56.2%
```

```
##           mean     SE
## votedem 0.5687 0.0657
```

```
svymean(~votedem, ms15, na.rm=TRUE) # Predicted 33.89% Dem 2party Governor vote share, actual 32.66%
```

```
##            mean     SE
## votedem 0.33885 0.0751
```

```
# Quick check to see how closely these vote predictions from CCES match actual totals (2018)
svymean(~votedem, ky18, na.rm=TRUE) # Predicted 46.33% Dem 2party US House vote share, actual 39.59%
```

```
##            mean    SE
## votedem 0.46328 0.035
```

```
svymean(~votedem, la18, na.rm=TRUE) # Predicted 38.39% Dem 2party US House vote share, actual 39.82%
```

```
##           mean    SE
## votedem 0.3839 0.041
```

```
svymean(~votedem, ms18, na.rm=TRUE) # Predicted 40.14% Dem 2party US House vote share, actual 45.83%
```

```
##            mean     SE
## votedem 0.40144 0.0574
```

The next step is to build the electorate for each of the two years at hand (2015 and 2018). In each case, we'll bin the electorate by gender, race, education level, and age, and summarize what proportion of the overall electorate voted Democrat and fell into each of these bins. Then we'll combine each of the two electorates into a single state model.

```
build_electorate <- function(survey, year) {
  #
  demsupport <- as_tibble(prop.table(svytable(~gender + race + educ + age + votedem, design = survey)))
    filter(votedem == 1) %>%
    select(-votedem) %>%
    rename(percentdem = n)

  cols <- c("percentdem")
  to_append <- as.character(year)
  demsupport <- demsupport %>% rename_at(cols, list(~paste0(., to_append)))
}



state_model <- function(fips) {
  #
  d15 <- design15(fips)
  d18 <- design18(fips)

  e15 <- build_electorate(d15, 15)
  e18 <- build_electorate(d18, 18)

  output <- e15 %>%
    full_join(e18, by = c("gender", "race", "educ", "age")) %>%
    replace_na(list(percentdem15 = 0, percentdem18 = 0))
}


ky_model <- state_model(21)
la_model <- state_model(22)
ms_model <- state_model(28)
```

Finally, we will use the two electorates to create a weighted prediction of how each demographic bin will turn out and support the Democratic candidate. Because there are compelling reasons to think that each of the two elections are likely to resemble 2019, I'll weight both of them equally. The 2015 election is a direct corollary to 2019: the same types of voters that turned out to vote for governor in an off-year are likely to do so again in 2019. At the same time, the national (and perhaps even local) political mood is very different in 2019 than it was in 2018, so it might be the case that voter preferences in 2019 more closely resemble those expressed in 2018.

```
predict_vote_share <- function(model) {
  #
  prediction <- model %>%
    mutate(percentdem19 = .5*percentdem15 + .5*percentdem18)

  prediction %>% summarize(sum(percentdem19)) %>% .[[1]]
}

ky_cces <- predict_vote_share(ky_model)
la_cces <- predict_vote_share(la_model)
ms_cces <- predict_vote_share(ms_model)

cces_pred <- tibble(year = 2019, location = c("KY", "LA", "MS"), cces_pred_pct = c(ky_cces, la_cces, ms
```

```
cces_pred
```

```
## # A tibble: 3 x 3
##    year location cces_pred_pct
##   <dbl> <chr>            <dbl>
## 1  2019 KY               0.467
## 2  2019 LA               0.476
## 3  2019 MS               0.370
```

This table represents a "pure" demographic prediction of how much of the two-party vote share the Democratic candidate should expect to win for governor in 2019. The demographic model is noticeably more bearish on the Democrats than the polls-only model - perhaps suggesting that the political mood in 2019 is noticeably more pro-Democratic than it was in these previous election cycles. In a longer-term project, I would be interested in running different model specifications and seeing how this demographic model predicted actual election results, but for now, this basic construction will do. In any case, I don't have a strong prior for how the two models (polls and demographics) should be weighted together. I think current-year polls have much more predictive power than generalized demographic models, so I'll simply weight the poll model at 2/3 of my overall prediction and the demographic model at 1/3.

```
all_pred <- gov_polls_pred %>%
  left_join(cces_pred, by = c("year", "location")) %>%
  mutate(FinalDemVtshPred = .67*meanPredDemVoteSh + .33*cces_pred_pct,
         FinalRepVtshPred = 1 - FinalDemVtshPred) %>%
  select(location, FinalDemVtshPred, FinalRepVtshPred)

all_pred
```

```
## # A tibble: 3 x 3
##   location FinalDemVtshPred FinalRepVtshPred
##   <chr>               <dbl>            <dbl>
## 1 KY                  0.496            0.504
## 2 LA                  0.555            0.445
## 3 MS                  0.443            0.557
```

The final predictions are as follows:

- **Kentucky**: Democrat Andy Beshear 49.57%, Republican Matt Bevin 50.43%
- **Louisiana**: Democrat John Bel Edwards 55.45%, Republican Eddie Rispone 44.55%
- **Mississippi**: Democrat Jim Hood 44.27%, Republican Tate Reeves 55.73%