HW1: forecasting election results (V1)

Summary

Learning goals in this HW are:

- more on functions and iteration: using and understanding code and functions provided to you, writing new ones. the code will form the basis for hw2 (making an R package)
- working with git (making commits) and keep your github remote repo in synch with your local one

In this HW, we introduce a new simulation exercise. This time, we simulate data that may represent weekly polling data for an election in the form of the % support for candidate A. The goal is to fit a random walk model to the data and then forecast the election outcome at the national level. We will continue with the same application and set up in HW2, when making an R package.

For this HW, you are provided code to simulate the data (including a function that is stored in subfolder R), and some code for aggregation to the national level and visualization (in the Rmd below). This set up will be the start of your R package.

It's up to you to

1. work through the current code and understand what's happening
2. fit a RW model (as explained below) to estimate its parameters
3. forecast the national level election outcome

Step 0

A. Load libraries

B. Save functions in R sub-folder and source all.

Notes:

- in HW2, we will replace this source() by making this code base into an R package; this set-up follows the basic set up for that.
- If you open the R script and make changes, you can source the script when saving by checking the box "Source on Save"

```r
# this is code to source any .R file in the subfolder R
Rfiles <- list.files(file.path(paste0(getwd(),"/R/")), ".R")
Rfiles <- Rfiles[grepl(".R", Rfiles)]
sapply(paste0(paste0(getwd(),"/R/"), Rfiles), source)
```

```
##         /Users/Colombian727/Downloads/hw1-Mansamusa007-main infinity/R/simulate_data.R
## value   ?
## visible FALSE
```

Step 1: simulate the data

Approach to obtain proportion p(s,t) for state s, week t:

- fix proportions p(s,0) at t=0
- in each state, for logit-transformed proportions, use a random walk model with drift d and standard deviation sd_{rw} for subsequent years.

The random walk model is defined as follows:

$$\text{logit}(p(s,t)) = \text{logit}(p(s,t-1)) + \varepsilon(s,t)$$

where $\varepsilon(s,t) \sim N(d, sd_{rw})$. Or in pseudo R-code:

logit(p(s, t)) = logit(p(s, t-1)) + rnorm(1, mean = d, sd = sd_rw)

```
nstates <- 52

# props at t = 0:
set.seed(123)
rw0 <- runif(nstates, 45, 60)

# RW-based simulations
rw <- simulate_rw(rw0 = rw0, sd_rw = 0.05,
                  n_steps = 10, # weeks here
                  drift = 0.01,
                  seed = 1234)
rw
```

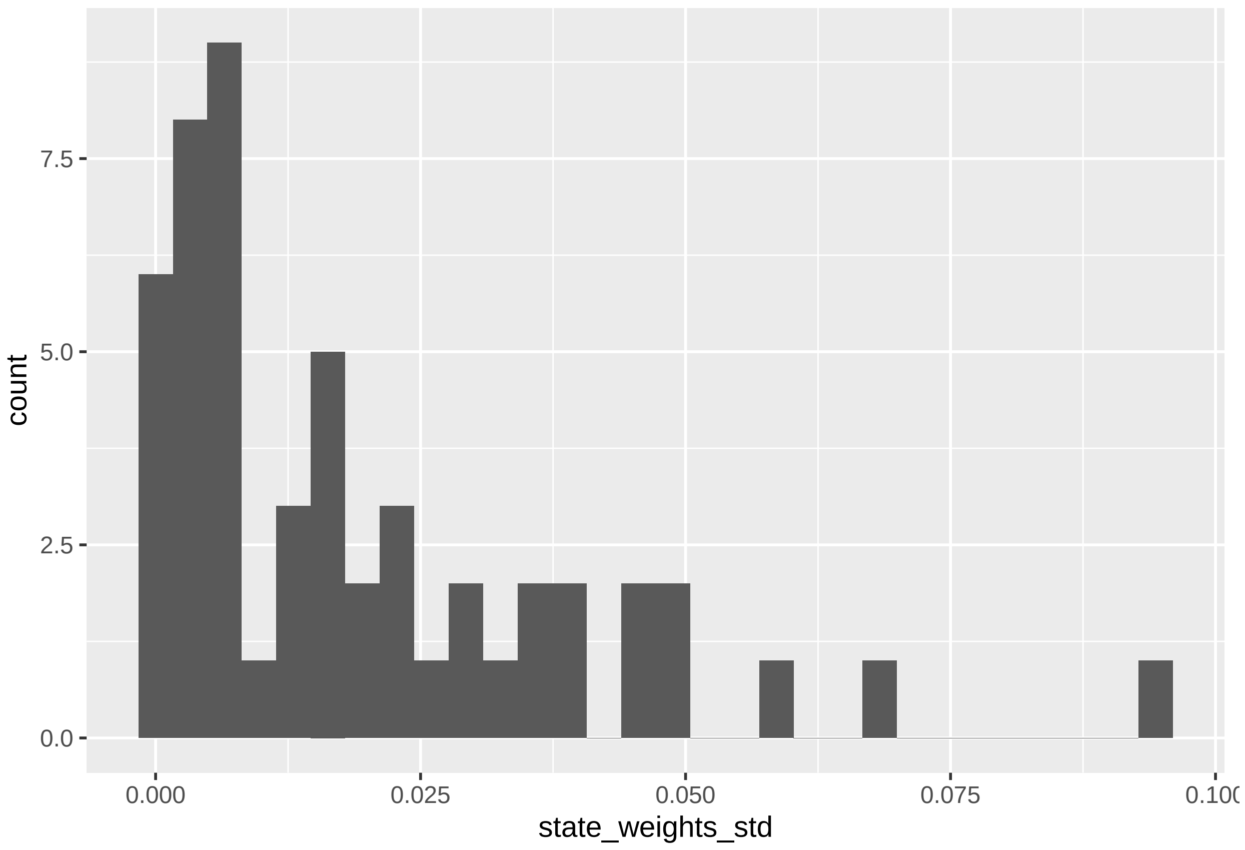
```
## # A tibble: 52 x 12
## # Groups:   state [52]
##    state  `0`   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`
##    <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1  49.3  48.1  48.7  50.3  47.6  48.4  49.2  48.8  48.3  47.9  47.0
##  2     2  56.8  56.5  55.5  54.8  55.1  56.5  56.7  56.3  55.4  54.6  57.8
##  3     3  51.1  51.6  51.2  50.9  51.7  51.1  49.5  50.5  49.5  49.7  48.8
##  4     4  58.2  59.8  59.5  58.9  58.5  56.8  55.6  53.1  51.7  51.6  51.2
##  5     5  59.1  61.1  60.0  59.3  59.2  58.2  57.3  56.1  54.8  54.4  54.1
##  6     6  45.7  43.7  43.2  42.1  41.1  41.2  42.1  44.4  43.7  45.9  44.7
##  7     7  52.9  54.0  57.4  57.6  57.0  57.2  59.6  58.5  60.4  62.2  62.8
##  8     8  58.4  58.6  58.3  58.1  59.2  61.9  61.9  60.5  59.9  60.4  60.3
##  9     9  53.3  53.3  53.3  51.9  51.9  53.2  54.3  55.3  55.0  55.0  53.8
## 10    10  51.8  52.0  52.6  55.0  56.5  56.1  56.8  55.6  56.9  58.4  61.2
## # ... with 42 more rows
```

Approach to obtain state weights… here we sample from a Gamma distribution to get some positive outcomes, and then standardize

```
set.seed(123456)
state_weights <- rgamma(nstates, 1, 1)
state_weights_dat <- tibble(
  state = seq_len(nstates),
  state_weights_std = state_weights/sum(state_weights))
```

```
ggplot(state_weights_dat) +
  geom_histogram(aes(x = state_weights_std))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Step 2: calculate the national aggregate percentage.

The national aggregated percentage is the weighted average of the state-specific outcomes, with weights given by the standardized state weights.

Easy approach here: just go for a data set in the long format, then add average for each year.

Note: we could consider re-organizing this information to avoid repeated information

```
rw_long <-
  rw %>%
    left_join(state_weights_dat) %>%
    pivot_longer(-c(state, state_weights_std),
              names_to = "t",
              values_to = "percent") %>%
    mutate(t = as.numeric(t)) %>%
    group_by(t) %>%
    mutate(agg = sum(percent*state_weights_std))
```

```
## Joining, by = "state"
```

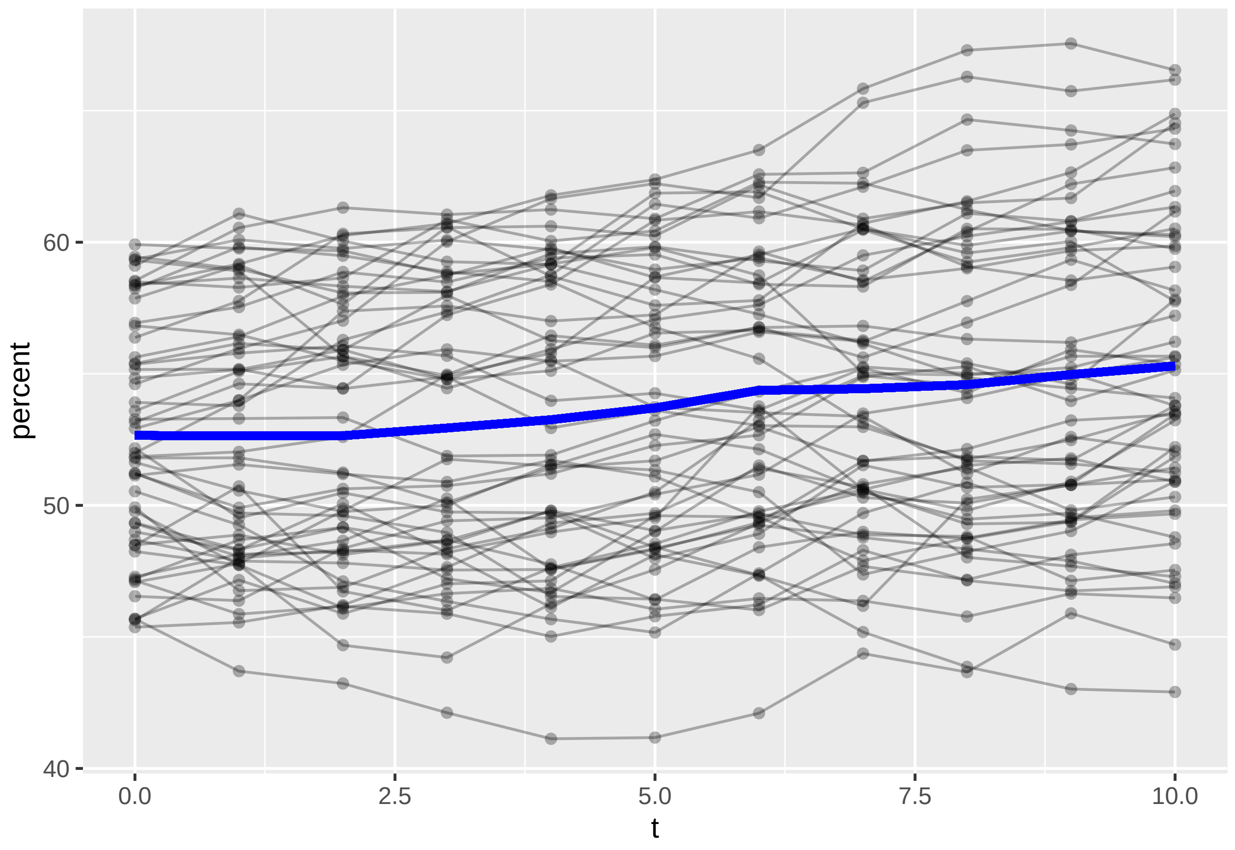
```
rw_long
```

```
## # A tibble: 572 x 5
## # Groups:   t [11]
##    state state_weights_std     t percent   agg
##    <int>             <dbl> <dbl>   <dbl> <dbl>
##  1     1            0.0225     0    49.3  52.7
##  2     1            0.0225     1    48.1  52.6
##  3     1            0.0225     2    48.7  52.6
##  4     1            0.0225     3    50.3  52.9
##  5     1            0.0225     4    47.6  53.3
##  6     1            0.0225     5    48.4  53.7
##  7     1            0.0225     6    49.2  54.4
##  8     1            0.0225     7    48.8  54.4
##  9     1            0.0225     8    48.3  54.6
## 10     1            0.0225     9    47.9  55.0
## # ... with 562 more rows
```

Step 3: data visualization

Let's look at what we got so far

```
rw_long %>%
  ggplot(aes(x = t, y = percent,
        group = state)) +
    geom_point(alpha = 0.3) +
    geom_line(alpha = 0.3) +
    geom_line(aes(y = agg), color = "blue", size = 1.5)
```



Step 4: fit the RW model to your "data"

Now pretend that you did not know what RW parameters (drift and sd_rw) were used, can you estimate them based on the available data?

The answer is yes!

Approach:

1. calculate the differences e(s,t) = logit(p(s,t)) - logit(p(s, t-1)).
2. your estimate for d is given by the mean of all e(s,t)
3. your estimate for sd_rw is given by the standard deviation of all e(s,t)

General approach: write some working code first, then put this into a function. Make sure the function returns the drift as well as the sd_rw estimate, ie in a list return(list(sd_rw = bla, drift = bla2))

```
new_rw_long <- rw_long %>%
  mutate(p = percent/100, logit_data = logit(p)) %>%
  group_by(state) %>%
  mutate(difference = logit_data - lag(logit_data)) %>%
  filter(!is.na(difference)) %>%
  ungroup() %>%
  mutate(drift = mean(difference)) %>%
  mutate(sd_data = sd(difference, na.rm = TRUE))
new_rw_long
```

```
## # A tibble: 520 x 10
##    state state_weights_s~     t percent   agg     p logit_data difference
##    <int>            <dbl> <dbl>   <dbl> <dbl> <dbl>      <dbl>      <dbl>
##  1     1           0.0225     1    48.1  52.6 0.481    -0.0778    -0.0504
##  2     1           0.0225     2    48.7  52.6 0.487    -0.0539     0.0239
##  3     1           0.0225     3    50.3  52.9 0.503     0.0103     0.0642
##  4     1           0.0225     4    47.6  53.3 0.476    -0.0970    -0.107
##  5     1           0.0225     5    48.4  53.7 0.484    -0.0655     0.0315
##  6     1           0.0225     6    49.2  54.4 0.492    -0.0302     0.0353
##  7     1           0.0225     7    48.8  54.4 0.488    -0.0490    -0.0187
##  8     1           0.0225     8    48.3  54.6 0.483    -0.0663    -0.0173
##  9     1           0.0225     9    47.9  55.0 0.479    -0.0845    -0.0182
## 10     1           0.0225    10    47.0  55.3 0.470    -0.119     -0.0345
## # ... with 510 more rows, and 2 more variables: drift <dbl>, sd_data <dbl>
```

```r
rw_long_trans <- function(data){
  new_data <- data %>%
    mutate(p = percent/100, logit_data = logit(p)) %>%
    group_by(state) %>%
    mutate(difference = logit_data - lag(logit_data)) %>%
    filter(!is.na(difference)) %>%
    ungroup() %>%
    summarize(drift = mean(difference), sd_data = sd(difference, na.rm = TRUE))
    return(list(sd_rw = new_data$sd_data, drift = new_data$drift))
}
rw_long_trans(rw_long)
```

```
## $sd_rw
## [1] 0.05177562
##
## $drift
## [1] 0.009955905
```

Step 5: On to making forecasts...

With the estimated drift and sd for the random walk, we can forecast trajectories for each state, using the random walk equation: logit(p(s, t)) = logit(p(s, t-1)) + rnorm(1, mean = d, sd = sd_rw)

We can then aggregate those to the national level to get a forecast with uncertainty...

Your task:

- Write a function to make state forecasts.
- Forecast 5 weeks out, 100 trajectories per state.
- Aggregate to get 100 national level trajectories
- Visualize the mean and 5th and 95th percentiles for the national forecast.

```r
forecast <- function(data, n_steps, n_weeks) {
  init_data <- data %>%
    filter(t == n_weeks)
  data_1 <- simulate_rw(rw0 = init_data$percent, sd_rw = rw_long_trans(data)$sd_rw,
                n_steps = n_steps, # weeks here
```

```r
                  drift = rw_long_trans(data)$drift,
                  seed = 1234)
  colnames(data_1) <- c("state", seq(n_weeks, n_weeks+n_steps))
  return(data_1)
}
rw_1 <- forecast(rw_long, 5, 10) %>% select(-c(state,"10"))
```

```
## Adding missing grouping variables: ‘state‘
```

```r
comb_rw <- rw %>%
  right_join(rw_1)
```

```
## Joining, by = "state"
```

```r
rw_1_long <- comb_rw %>%
  left_join(state_weights_dat) %>%
  pivot_longer(-c(state, state_weights_std),
    names_to = "t",
    values_to = "percent") %>%
  mutate(t = as.numeric(t)) %>%
  group_by(t) %>%
  mutate(agg = sum(percent*state_weights_std))
```

```
## Joining, by = "state"
```

```r
rw_1_long
```

```
## # A tibble: 832 x 5
## # Groups:   t [16]
##    state state_weights_std     t percent   agg
##    <int>             <dbl> <dbl>   <dbl> <dbl>
##  1     1            0.0225     0    49.3  52.7
##  2     1            0.0225     1    48.1  52.6
##  3     1            0.0225     2    48.7  52.6
##  4     1            0.0225     3    50.3  52.9
##  5     1            0.0225     4    47.6  53.3
##  6     1            0.0225     5    48.4  53.7
##  7     1            0.0225     6    49.2  54.4
##  8     1            0.0225     7    48.8  54.4
##  9     1            0.0225     8    48.3  54.6
## 10     1            0.0225     9    47.9  55.0
## # ... with 822 more rows
```

```r
comb_rw
```

```
## # A tibble: 52 x 17
## # Groups:   state [52]
##    state  ‘0‘   ‘1‘   ‘2‘   ‘3‘   ‘4‘   ‘5‘   ‘6‘   ‘7‘   ‘8‘   ‘9‘  ‘10‘  ‘11‘
##    <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1  49.3  48.1  48.7  50.3  47.6  48.4  49.2  48.8  48.3  47.9  47.0  45.7
##  2     2  56.8  56.5  55.5  54.8  55.1  56.5  56.7  56.3  55.4  54.6  57.8  58.7
##  3     3  51.1  51.6  51.2  50.9  51.7  51.1  49.5  50.5  49.5  49.7  48.8  48.4
##  4     4  58.2  59.8  59.5  58.9  58.5  56.8  55.6  53.1  51.7  51.6  51.2  51.3
##  5     5  59.1  61.1  60.0  59.3  59.2  58.2  57.3  56.1  54.8  54.4  54.1  54.5
##  6     6  45.7  43.7  43.2  42.1  41.1  41.2  42.1  44.4  43.7  45.9  44.7  43.1
##  7     7  52.9  54.0  57.4  57.6  57.0  57.2  59.6  58.5  60.4  62.2  62.8  64.4
##  8     8  58.4  58.6  58.3  58.1  59.2  61.9  61.9  60.5  59.9  60.4  60.3  59.1
##  9     9  53.3  53.3  53.3  51.9  51.9  53.2  54.3  55.3  55.0  55.0  53.8  55.9
## 10    10  51.8  52.0  52.6  55.0  56.5  56.1  56.8  55.6  56.9  58.4  61.2  60.2
## # ... with 42 more rows, and 4 more variables: ‘12‘ <dbl>, ‘13‘ <dbl>,
## #   ‘14‘ <dbl>, ‘15‘ <dbl>
```

TBD (by Th) whether more details/hints will be added/what part of this exercise is extra credit versus needed for full score.