We're looking to expand our proxy-state analysis to additional years, in order to make sure we're comfortable using PA as a proxy for MA crime. Here we're combining data from 2014-2018 (from what was usually known as table 69, aggregated arrest rates, save one year it was reported as table 2) and repeating our look at euclidian distance vs MA with both standardized and non-standardized variables.

So first things first, let's load our libraries and data:

```
knitr::opts_chunk$set(warning = F)
library(here) ## relative pathways
```

```
## here() starts at /home/mikemahoney218/codebase/clean-slate
```

```
library(readxl) ## reading excel
library(dplyr) ## data manipulation
```

```
##
## 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
```

```
library(tidyr) ## nesting dataframes
library(purrr) ## map reduce functions
library(ggplot2)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
fbi_data <- read_excel(
  here(
    "data",
    "cleaned",
    "fbi_aggregated_data_combined/FBI_aggregate_crime_data_2014_2018.xlsx"
    )
  )
```

Now we're repeating the same analysis done in the other FBI folder here, but nesting across state and year (so we'll compare each each state in 2018 to MA 2018).

```
nested_fbi_data <- fbi_data %>%
  filter(age_category == "Under 18") %>%
  ## these variables are the same ones done with 2014 data
  select(state,
         year,
         robbery,
         property_crime,
```

```r
         burglary,
         larceny_theft,
         motor_vehicle_theft,
         estimated_population) %>%
  mutate(year = as.character(year)) %>%
  ## get per-capita crime rate
  mutate_if(is.numeric, funs(. / estimated_population)) %>%
  select(-estimated_population) %>%
  ## this bit is wonky if you don't know R; I'm creating a column of dataframes
  ## containing data for only that state
  nest(nested = -c(state, year)) %>%
  mutate(state = regmatches(state,
                            regexpr("[[:alpha:]]*\\s?[[:alpha:]]*",
                                    state)))
```

Calculate each state's distance from MA:

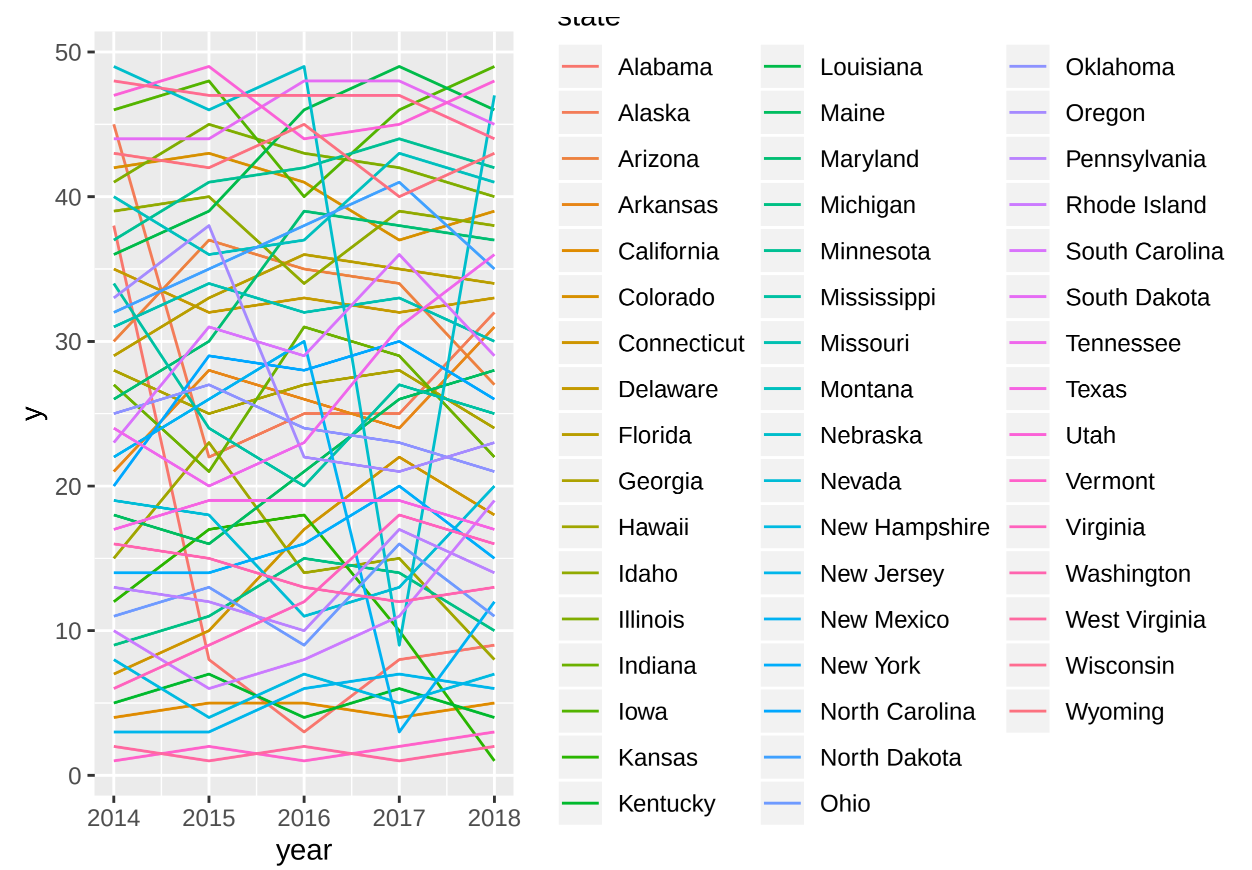
```r
ranked_distances <- nested_fbi_data %>%
  left_join(nested_fbi_data %>%
  filter(state == "Massachusetts") %>%
    select(-state, mass = nested),
  by = "year") %>%
  mutate(dist_tables = map2(nested, mass, vctrs::vec_rbind),
         dist_score = map_dbl(dist_tables, dist),
          year = as.numeric(year)) %>%
  filter(!is.infinite(dist_score) &
           state != "Massachusetts") %>%
  arrange(year, dist_score) %>%
  group_by(year) %>%
  # y is the rank of the state for that year -- a rank of 1 means its
  # the most similar to MA that year
  mutate(y = seq(1, 49)) %>%
    ungroup()

ranked_distances
```

```
## # A tibble: 245 x 7
##    state       year     nested        mass dist_tables   dist_score     y
##    <chr>      <dbl> <list<df[,5> <list<df[,> <list>             <dbl> <int>
##  1 Vermont     2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.0000442     1
##  2 West Virg~  2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.0000580     2
##  3 New Jersey  2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000237      3
##  4 California  2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000304      4
##  5 Kentucky    2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000332      5
##  6 Virginia    2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000369      6
##  7 Connectic~  2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000372      7
##  8 New Hamps~  2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000373      8
##  9 Michigan    2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000386      9
## 10 Rhode Isl~  2014      [1 x 5]     [1 x 5] <tibble [2 x~  0.000386     10
## # ... with 235 more rows
```

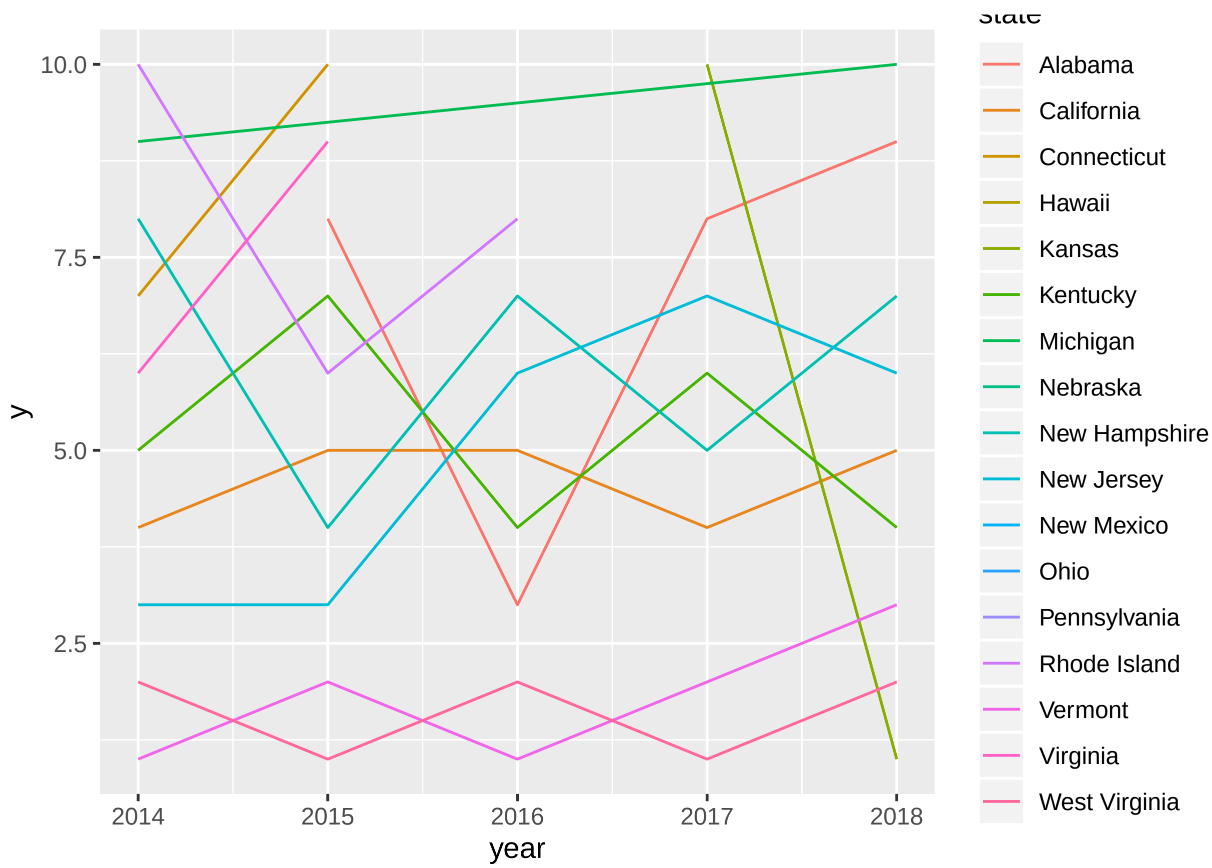
And quickly graph that:

```r
ranked_distances %>%
  ggplot(aes(year, y, color = state)) +
  geom_line()
```



Gross! Let's quickly graph that better:

```
ranked_distances %>%
  filter(y < 11) %>%
  ggplot(aes(year, y, color = state)) +
  geom_line()
```



"Better", at any rate. I'm going to set aside visualizations for a second and just pull PA rankings:

```
ranked_distances %>%
  filter(state == "Pennsylvania")
```

```
## # A tibble: 5 x 7
##   state        year       nested        mass dist_tables    dist_score     y
##   <chr>       <dbl> <list<df[,5> <list<df[,> <list>              <dbl> <int>
## 1 Pennsylva~  2014      [1 x 5]     [1 x 5] <tibble [2 x ~   0.000463    13
## 2 Pennsylva~  2015      [1 x 5]     [1 x 5] <tibble [2 x ~   0.000396    12
## 3 Pennsylva~  2016      [1 x 5]     [1 x 5] <tibble [2 x ~   0.000347    10
## 4 Pennsylva~  2017      [1 x 5]     [1 x 5] <tibble [2 x ~   0.000370    17
## 5 Pennsylva~  2018      [1 x 5]     [1 x 5] <tibble [2 x ~   0.000325    14
```

So PA almost never cracks the top 10, except for 2016. However, it only drops out of the top 15 once – and that's 2017, when a few other states have wild variances in our first graph. I wonder what the average ranking is across the board – looks like most states are pretty stable:

```
library(magrittr)
ranked_distances %>%
  group_by(state) %>%
  summarise(mean_rank = mean(y), median_rank = median(y)) %>%
  arrange(mean_rank) %T>%
  write.csv("state_ranks.csv")
```

```
## # A tibble: 49 x 3
##    state          mean_rank median_rank
##    <chr>              <dbl>       <int>
```

```
##   1 West Virginia       1.6           2
##   2 Vermont             1.8           2
##   3 California          4.6           5
##   4 New Jersey          5             6
##   5 Kentucky            5.2           5
##   6 New Hampshire       6.2           7
##   7 Rhode Island       10.8          10
##   8 Kansas             11.6          12
##   9 Michigan           11.8          11
##  10 Ohio               12            11
## # ... with 39 more rows
```

I'm frankly shocked that WV is in the top bracket; KY is also surprising, but the rest of the states make sense. It also feels to me like we've got a few tiers here – the top-tier proxies are states from WV to KY, which are typically in that top five bucket. Then the secondary tier (RI -> HI, maybe) hovers around 13.

PA is in that second tier, with a mean ranking of 13.2 (again, 1 is closest) and median ranking of... 13. It feels valuable to me to spend time looking into how accessible data for those top-tier states might be – but if not, I think we've reinforced that PA is a decent option nontheless.

For completeness, I should make the same table based on standardized crime rates:

```r
nested_scaled_data <- fbi_data %>%
  filter(age_category == "Under 18") %>%
  ## these variables are the same ones done with 2014 data
  select(state,
         year,
         robbery,
         property_crime,
         burglary,
         larceny_theft,
         motor_vehicle_theft,
         estimated_population) %>%
  mutate(year = as.character(year)) %>%
  ## get per-capita crime rate
  mutate_if(is.numeric, funs(. / estimated_population)) %>%
  mutate_if(is.numeric, scale) %>%
  select(-estimated_population) %>%
  ## this bit is wonky if you don't know R; I'm creating a column of dataframes
  ## containing data for only that state
  nest(nested = -c(state, year)) %>%
  mutate(state = regmatches(state,
                            regexpr("[[:alpha:]]*\\s?[[:alpha:]]*",
                                    state)))


ranked_scaled_distances <- nested_scaled_data %>%
  left_join(nested_scaled_data %>%
  filter(state == "Massachusetts") %>%
    select(-state, mass = nested),
  by = "year") %>%
  mutate(dist_tables = map2(nested, mass, rbind),
         dist_tables = map(dist_tables, as_tibble),
         dist_score = map_dbl(dist_tables, dist),
          year = as.numeric(year)) %>%
  filter(!is.infinite(dist_score) &
```

```r
           state != "Massachusetts") %>%
  arrange(year, dist_score) %>%
  group_by(year) %>%
  # y is the rank of the state for that year -- a rank of 1 means its
  # the most similar to MA that year
  mutate(y = seq(1, 50)) %>%
    ungroup()

ranked_scaled_distances
```

```
## # A tibble: 250 x 7
##    state        year       nested        mass dist_tables   dist_score     y
##    <chr>       <dbl> <list<df[,5> <list<df[,> <list>              <dbl> <int>
##  1 North Dak~  2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.105     1
##  2 Vermont     2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.125     2
##  3 Kansas      2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.196     3
##  4 Wyoming     2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.198     4
##  5 Virginia    2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.431     5
##  6 Montana     2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.435     6
##  7 New Hamps~  2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.568     7
##  8 Alabama     2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.808     8
##  9 Minnesota   2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.856     9
## 10 Michigan    2014       [1 x 5]     [1 x 5] <tibble [2 x~      0.969    10
## # ... with 240 more rows
```

```r
ranked_scaled_distances %>%
  group_by(state) %>%
  summarise(mean_rank = mean(y), median_rank = median(y)) %>%
  arrange(mean_rank) %T>%
  write.csv("state_scaled_ranks.csv")
```

```
## # A tibble: 50 x 3
##    state          mean_rank median_rank
##    <chr>              <dbl>       <int>
##  1 Vermont              3             2
##  2 New Hampshire        4.4           3
##  3 West Virginia        6.6           6
##  4 Kansas               7.4           5
##  5 Virginia             8             7
##  6 Hawaii               9             8
##  7 North Dakota        10.2           6
##  8 Montana             10.8           6
##  9 New Jersey          10.8          10
## 10 New Mexico          11.2          12
## # ... with 40 more rows
```

PA drops to 14th, and some weird contenders (ND? MT?) enter the upper tiers. I think I'm justified in saying unscaled data is probably a better approach, but I don't think it changes much.