Exploratory Data Analysis for 2-18 COD data

Load settings and data

```
source("code/00_config.R")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
##
## Attaching package: 'lubridate'
##
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
##
##
## Linking to GEOS 3.10.2, GDAL 3.4.2, PROJ 8.2.1; sf_use_s2() is TRUE
##
##
## Attaching package: 'raster'
##
##
## The following object is masked from 'package:dplyr':
##
##     select
##
##
## terra 1.6.17
##
##
## Attaching package: 'terra'
##
##
## The following object is masked from 'package:tidyr':
##
##     extract
##
##
## To enable caching of data, set `options(tigris_use_cache = TRUE)`
## in your R script or .Rprofile.
##
##
## Attaching package: 'tigris'
##
##
## The following object is masked from 'package:terra':
```

```
##
##     blocks
##
##
## Be sure to set the download folder using ‘prism_set_dl_dir()‘.
##
##
## Attaching package: 'tictoc'
##
##
## The following objects are masked from 'package:terra':
##
##     shift, size
##
##
## The following object is masked from 'package:raster':
##
##     shift
##
##
##
## Attaching package: 'tidylog'
##
##
## The following object is masked from 'package:raster':
##
##     select
##
##
## The following objects are masked from 'package:dplyr':
##
##     add_count, add_tally, anti_join, count, distinct, distinct_all,
##     distinct_at, distinct_if, filter, filter_all, filter_at, filter_if,
##     full_join, group_by, group_by_all, group_by_at, group_by_if,
##     inner_join, left_join, mutate, mutate_all, mutate_at, mutate_if,
##     relocate, rename, rename_all, rename_at, rename_if, rename_with,
##     right_join, sample_frac, sample_n, select, select_all, select_at,
##     select_if, semi_join, slice, slice_head, slice_max, slice_min,
##     slice_sample, slice_tail, summarise, summarise_all, summarise_at,
##     summarise_if, summarize, summarize_all, summarize_at, summarize_if,
##     tally, top_frac, top_n, transmute, transmute_all, transmute_at,
##     transmute_if, ungroup
##
##
## The following objects are masked from 'package:tidyr':
##
##     drop_na, fill, gather, pivot_longer, pivot_wider, replace_na,
##     spread, uncount
##
##
## The following object is masked from 'package:stats':
##
##     filter
##
```

```
##
## Loading required package: Matrix
##
##
## Attaching package: 'Matrix'
##
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
##
## Loading required package: foreach
##
##
## Attaching package: 'foreach'
##
##
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
##
##
## Loading required package: parallel
##
## This is INLA_22.12.16 built 2022-12-23 13:43:44 UTC.
##  - See www.r-inla.org/contact-us for how to get help.
##  - To enable PARDISO sparse library; see inla.pardiso()
##
##
## Attaching package: 'INLA'
##
##
## The following object is masked from 'package:fixest':
##
##     f
##
##
## Your original .Renviron will be backed up and stored in your R HOME directory if needed.
##
## Your API key has been stored in your .Renviron and can be accessed by Sys.getenv("CENSUS_API_KEY").
## To use now, restart R or run `readRenviron("~/.Renviron")`
##
## mutate: converted 'FIPS' from integer to character (0 new NA)
```

```r
COD_raw <- readRDS("data_raw/age_standardized_rates/age_standardized_rates-county_month-all_sex-all_mar

COD <-
    COD_raw %>%
    filter(year >= 2006 &
               race_eth == "all" &
               age_group == "all_ages" &
               sex == "all" &
               race_eth == "all") %>%
    group_by(fipsihme, year, month) %>%
    summarise(pop = sum(pop), n_deaths = sum(n_deaths), rate = n_deaths / pop) %>%
    ungroup()
```

```
## filter: removed 15,051,816 rows (96%), 559,500 rows remaining
## group_by: 3 grouping variables (fipsihme, year, month)
## summarise: now 559,500 rows and 6 columns, 2 group variables remaining (fipsihme, year)
## ungroup: no grouping variables
```

```
fipsihme_sf <-
    counties(year = 2020) %>% # year should be 2006 once those data are working again
    filter(STATEFP %in% non_CONUS_FIPS == F) %>%
    dplyr::mutate(COUNTYFP = paste0(STATEFP, COUNTYFP)) %>%
    dplyr::select(COUNTYFP) %>%
    left_join(
        ihme_fips %>% dplyr::select(COUNTYFP = orig_fips, ihme_fips),
        by=c("COUNTYFP")
    ) %>%
    mutate(
        ihme_fips = ifelse(is.na(ihme_fips), COUNTYFP, ihme_fips)
    ) %>%
    dplyr::select(fipsihme = ihme_fips, COUNTYFP)
```

```
## filter: removed 126 rows (4%), 3,108 rows remaining
## left_join: added one column (ihme_fips)
##            > rows only in x   3,062
##            > rows only in y  (    31)
##            > matched rows        46
##            >                 =======
##            > rows total       3,108
## mutate: changed 3,062 values (99%) of 'ihme_fips' (3062 fewer NA)
```

```
COD_sf <-
    COD %>%
    left_join(
        fipsihme_sf %>% dplyr::select(fipsihme, geometry)
    ) %>%
    st_as_sf() #key step to recover sf geometry for some reason
```

```
## Joining, by = "fipsihme"
## left_join: added one column (geometry)
## > rows only in x 4,368
## > rows only in y ( 0)
## > matched rows 559,272 (includes duplicates)
## > =========
## > rows total 563,640
```

Maps:

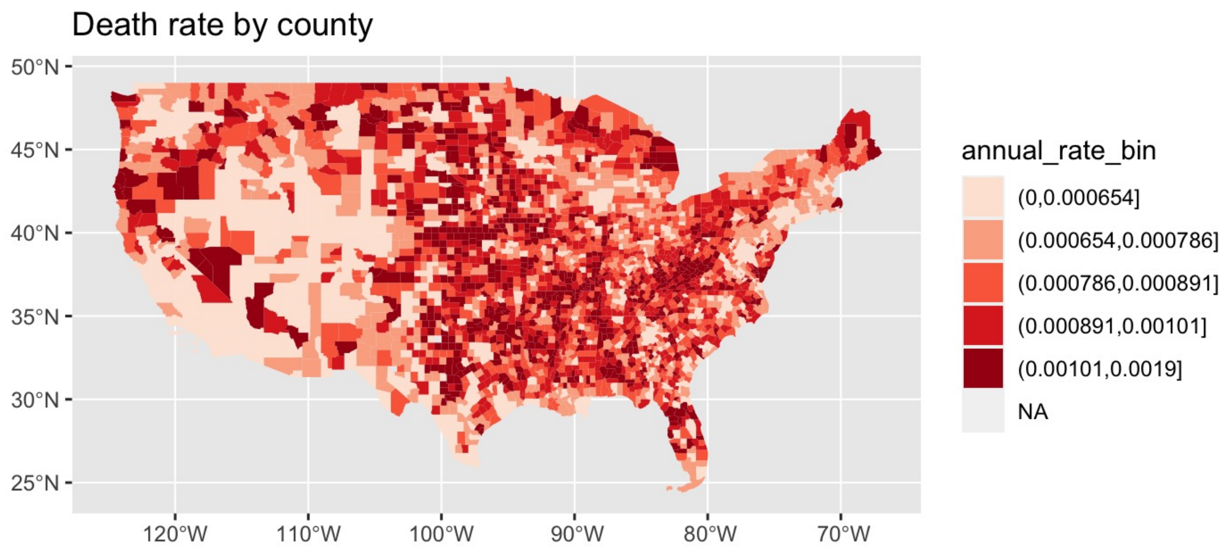


Figure 1: map of death rate by county

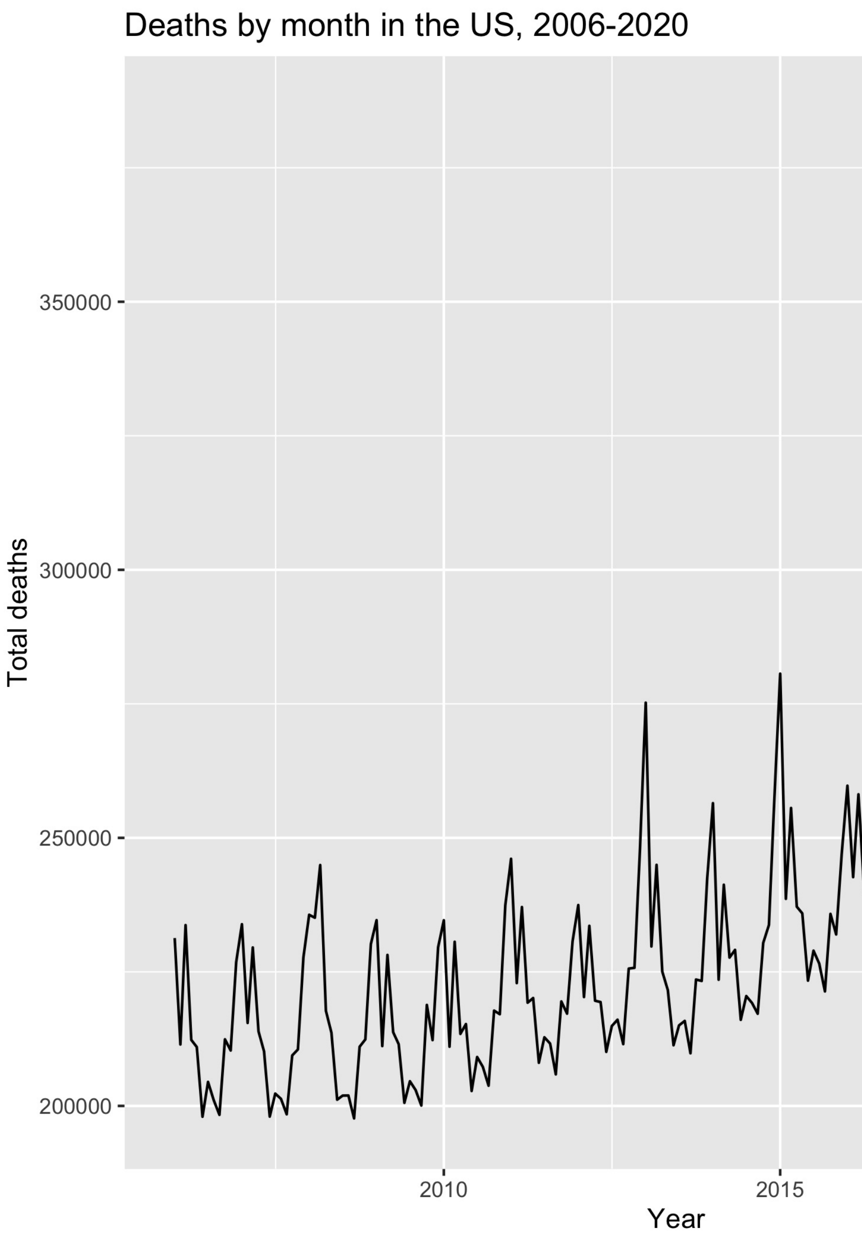
```r
acm_county_2010 <-
    COD %>%
    group_by(fipsihme, year) %>%
    summarise(annual_rate = mean(rate)) %>%
    ungroup() %>%
    filter(year == 2010) %>%
    mutate(
        annual_rate_bin = cut(annual_rate, breaks = quantile(annual_rate, seq(0, 1, .2)))
    ) %>%
    left_join(
        fipsihme_sf %>% dplyr::select(fipsihme, geometry)
    ) %>%
    st_as_sf() %>% #key step to recover sf geometry for some reason
    ggplot(aes(fill = annual_rate_bin)) +
    geom_sf(color = NA) + #color=NA to remove county borders
    scale_fill_brewer(palette = "Reds") +
    ggtitle("Death rate by county")
```

All-cause mortality rate by county for each year

```
## group_by: 2 grouping variables (fipsihme, year)

## summarise: now 46,625 rows and 3 columns, one group variable remaining (fipsihme)

## ungroup: no grouping variables

## filter: removed 43,517 rows (93%), 3,108 rows remaining

## mutate: new variable 'annual_rate_bin' (factor) with 6 unique values and <1% NA

## Joining, by = "fipsihme"
## left_join: added one column (geometry)
## > rows only in x 24
## > rows only in y ( 1)
## > matched rows 3,107 (includes duplicates)
## > =======
## > rows total 3,131
```

```r
#ggsave(filename = "plots/2-17-23_COD_EDA/2010_county_deathrates.jpg", plot = acm_county_2010, device =

# library(tmap)
# tmap_test <-
#     COD %>%
#     group_by(fipsihme, year) %>%
#     summarise(annual_rate = mean(rate)) %>%
#     ungroup() %>%
#     filter(year == 2010) %>%
#     left_join(
#         fipsihme_sf %>% dplyr::select(fipsihme, geometry)
#     ) %>%
#     st_as_sf() #key step to recover sf geometry for some reason
#
# tmap_test %>%
#     dplyr::filter(!sf::st_is_empty(geometry)) %>%
#     tm_shape() +
#     tm_fill("annual_rate")
#
# tmap_save(tmap_test, filename = "plots/tmap.jpg")
```

Time series



Whole US Should match readily available data

```r
### For some reason this code chunk is terminating R, I have no clue why
# monthly_US_deaths <-
#     COD_sf %>%
#     mutate(yearmonth = paste(year, month, sep="-")) %>%
#     mutate(yearmonth = ym(yearmonth)) %>%
#     group_by(yearmonth) %>%
#     summarise(n_deaths = sum(n_deaths), us_pop = sum(pop)) %>%
#     ggplot(aes(x = yearmonth, y = n_deaths)) +
#     geom_line() +
#     ggtitle("Deaths by month in the US, 2006-2020") +
#     xlab("Year") + ylab("Total deaths")

#ggsave(filename = "plots/2-17-23_COD_EDA/monthly_US_deaths.jpg", plot=monthly_US_deaths, device="jpg")
```

```r
largest_counties <-
    COD %>% ungroup() %>%
    filter(year == 2010) %>% #could have chosen different year
    dplyr::select(fipsihme, pop, year) %>%
    distinct() %>%
    arrange(desc(pop)) %>%
    slice(1:10) %>%
    left_join(
        counties() %>% dplyr::select(GEOID, NAME),
        by=c("fipsihme" = "GEOID")
    ) %>%
    select(-c(pop, year)) %>%
    st_drop_geometry()
```

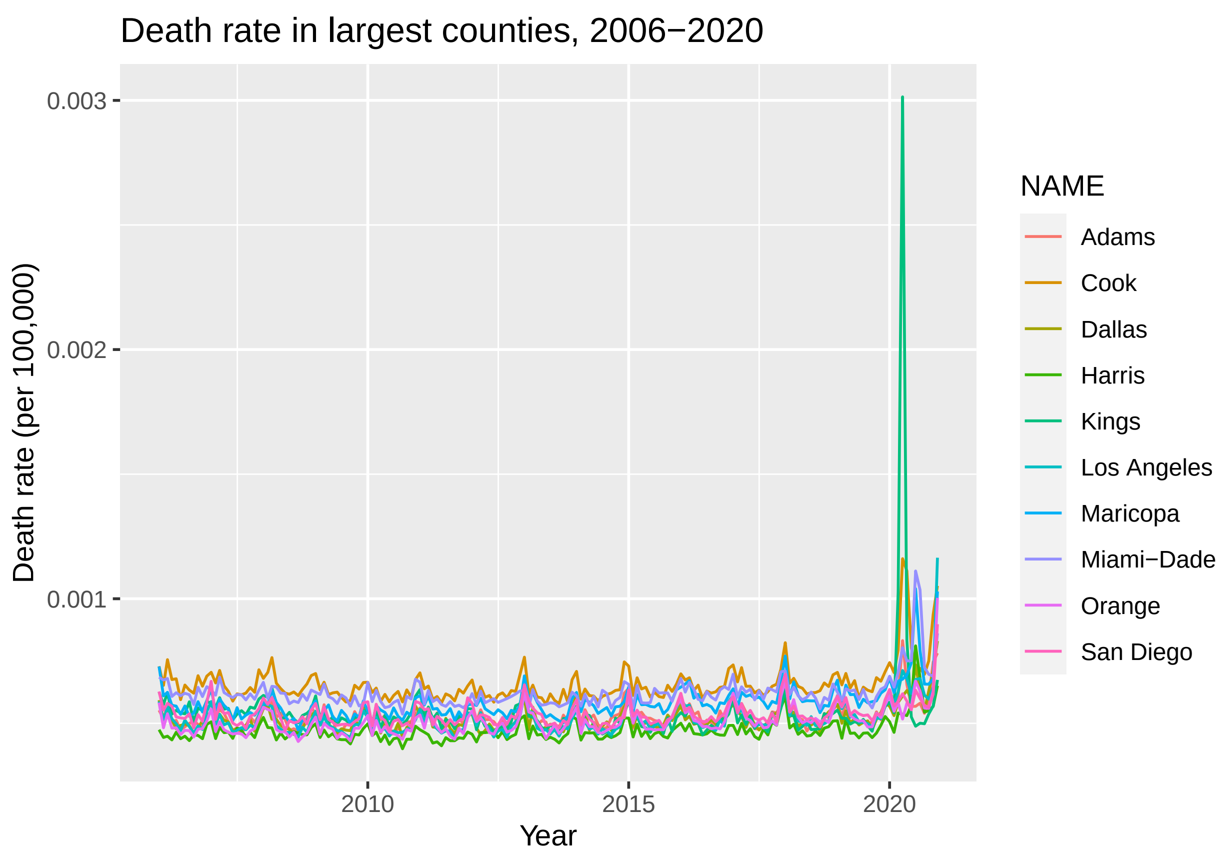
Time series of ACM rate and raw death counts for 10 largest counties

```
## ungroup: no grouping variables

## filter: removed 522,204 rows (93%), 37,296 rows remaining

## distinct: removed 34,188 rows (92%), 3,108 rows remaining

## slice: removed 3,098 rows (>99%), 10 rows remaining

## Retrieving data for the year 2020

## left_join: added 2 columns (NAME, geometry)

##            > rows only in x       0

##            > rows only in y  (3,224)

##            > matched rows        10
##            >                 =======

##            > rows total          10

## select: dropped 2 variables (pop, year)
```

```
monthly_deaths_largest_counties <-
    COD_sf %>%
    right_join(largest_counties) %>%
    mutate(yearmonth = paste(year, month, sep="-")) %>%
    mutate(yearmonth = ym(yearmonth)) %>%
    ggplot(aes(x = yearmonth, y = rate)) +
    geom_line(aes(color = NAME))
```

```
## Joining, by = c("fipsihme", "geometry")

## right_join: added one column (NAME)

##             > rows only in x  (561,840)

##             > rows only in y         0

##             > matched rows       1,800    (includes duplicates)

##             >                 =========

##             > rows total         1,800

## mutate: new variable 'yearmonth' (character) with 180 unique values and 0% NA

## mutate: converted 'yearmonth' from character to Date (0 new NA)
```

```
monthly_deaths_largest_counties +
    ggtitle("Death rate in largest counties, 2006-2020") +
    xlab("Year") + ylab("Death rate (per 100,000)")
```



```
# counties to look closely at
large_counties =
    list("Los Angeles County" = "06037",
         "Harris County" = "48201",
         "Miami-Dade County" = "12086",
         "Cook County" = "17031",
         "New York County" = "36061",
         "San Francisco County" = "06075")

medium_counties =
    list("Jefferson County" = "01073",
         "Mecklenburg County" = "37119",
         "Lucas County" = "39095",
         "Fresno County" = "06019",
         "Boulder County" = "08013",
         "East Baton Rouge Parish" = "22033")

small_counties =
    list("Stearns County" = "27145",
         "Pike County" = "01109",
         "Clark County" = "18019",
         "Steuben County" = "36101",
         "Aroostook County" = "23003",
         "Douglas County" = "41019")

# Extract data from the lists above, make label for size category
```

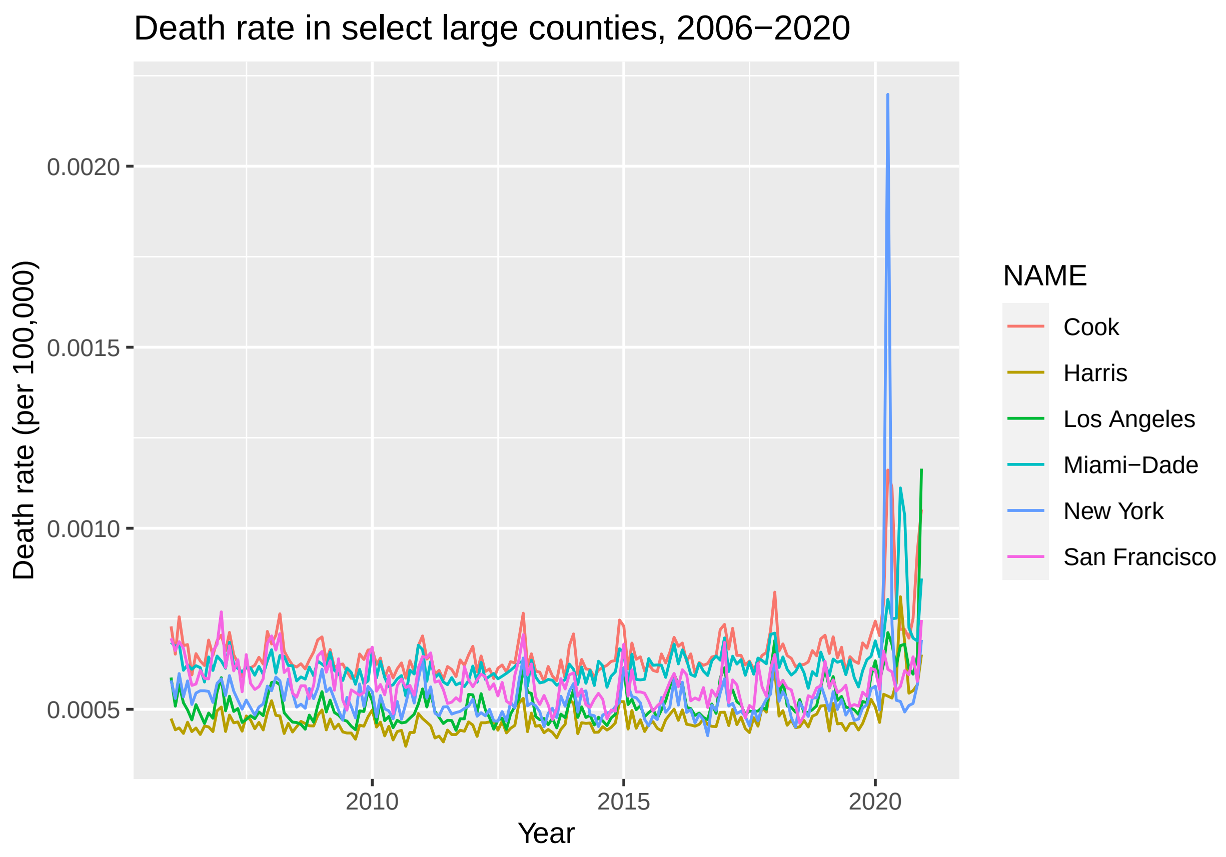
```
pull_county_data_by_size = function(size) {
    COD %>%
        filter(fipsihme %in% unname(unlist(!!sym(glue::glue("{size}_counties"))))) %>%
        mutate(type = size)
}

county_specific_df <- lapply(c("small", "medium", "large"), pull_county_data_by_size) %>% bind_rows()
```

```
## filter: removed 558,420 rows (>99%), 1,080 rows remaining

## mutate: new variable 'type' (character) with one unique value and 0% NA

## filter: removed 558,600 rows (>99%), 900 rows remaining

## mutate: new variable 'type' (character) with one unique value and 0% NA

## filter: removed 558,420 rows (>99%), 1,080 rows remaining

## mutate: new variable 'type' (character) with one unique value and 0% NA
```

```
monthly_rates_largecounty <-
    county_specific_df %>%
    filter(type == "large") %>%
    left_join(
        counties() %>% dplyr::select(GEOID, NAME),
        by=c("fipsihme" = "GEOID")
    ) %>%
    mutate(yearmonth = paste(year, month, sep="-")) %>%
    mutate(yearmonth = ym(yearmonth)) %>%
    ggplot(aes(x = yearmonth, y = rate)) +
    geom_line(aes(color = NAME))
```

```
## filter: removed 1,980 rows (65%), 1,080 rows remaining

## Retrieving data for the year 2020

## left_join: added 2 columns (NAME, geometry)

##              > rows only in x       0

##              > rows only in y  (3,228)

##              > matched rows     1,080

##              >                =======

##              > rows total       1,080

## mutate: new variable 'yearmonth' (character) with 180 unique values and 0% NA

## mutate: converted 'yearmonth' from character to Date (0 new NA)
```

```
monthly_rates_largecounty +
    ggtitle("Death rate in select large counties, 2006-2020") +
    xlab("Year") + ylab("Death rate (per 100,000)")
```



```
monthly_rates_mediumcounty <-
    county_specific_df %>%
    filter(type == "medium") %>%
    left_join(
        counties() %>% dplyr::select(GEOID, NAME),
        by=c("fipsihme" = "GEOID")
    ) %>%
    mutate(yearmonth = paste(year, month, sep="-")) %>%
    mutate(yearmonth = ym(yearmonth)) %>%
    ggplot(aes(x = yearmonth, y = rate)) +
    geom_line(aes(color = NAME))
```

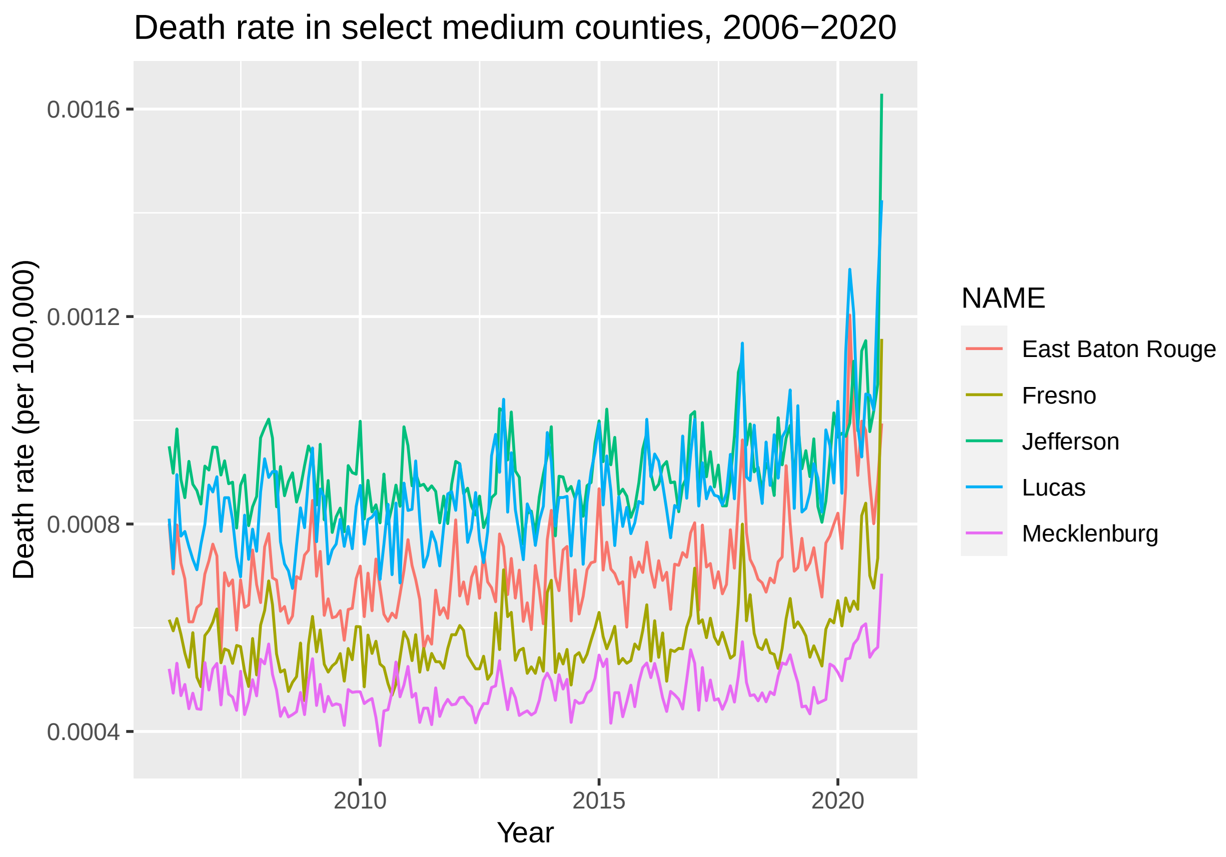
```
## filter: removed 2,160 rows (71%), 900 rows remaining
```

```
## Retrieving data for the year 2020
```

```
## left_join: added 2 columns (NAME, geometry)
```

```
##             > rows only in x       0
##              > rows only in y  (3,229)
##              > matched rows       900
##              >                 =======
##              > rows total         900
## mutate: new variable 'yearmonth' (character) with 180 unique values and 0% NA
## mutate: converted 'yearmonth' from character to Date (0 new NA)
```

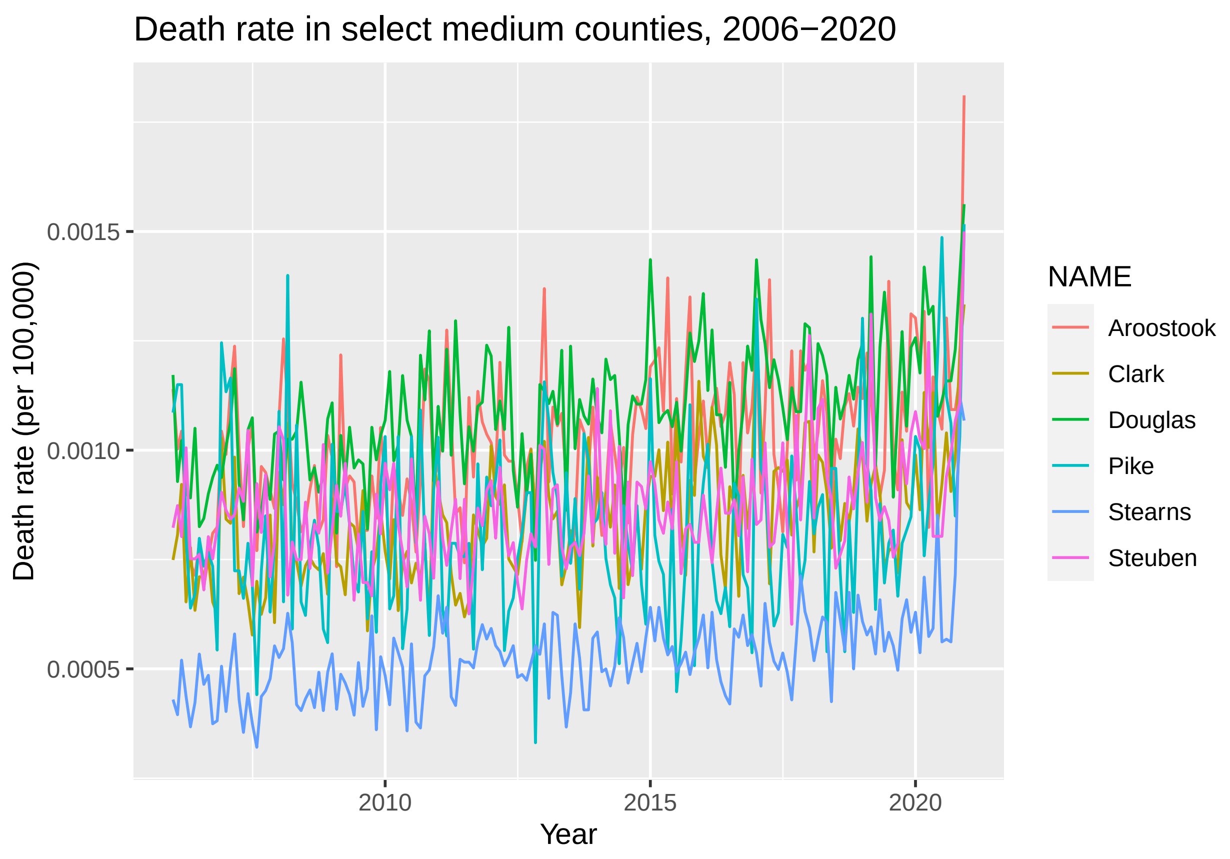
```
monthly_rates_mediumcounty +
    ggtitle("Death rate in select medium counties, 2006-2020") +
    xlab("Year") + ylab("Death rate (per 100,000)")
```



```
monthly_rates_smallcounty <-
    county_specific_df %>%
    filter(type == "small") %>%
    left_join(
        counties() %>% dplyr::select(GEOID, NAME),
        by=c("fipsihme" = "GEOID")
    ) %>%
    mutate(yearmonth = paste(year, month, sep="-")) %>%
    mutate(yearmonth = ym(yearmonth)) %>%
    ggplot(aes(x = yearmonth, y = rate)) +
    geom_line(aes(color = NAME))
```

```
## filter: removed 1,980 rows (65%), 1,080 rows remaining

## Retrieving data for the year 2020

## left_join: added 2 columns (NAME, geometry)

##            > rows only in x       0

##            > rows only in y  (3,228)

##            > matched rows     1,080

##            >                 =======

##            > rows total       1,080

## mutate: new variable 'yearmonth' (character) with 180 unique values and 0% NA

## mutate: converted 'yearmonth' from character to Date (0 new NA)
```

```
monthly_rates_smallcounty +
    ggtitle("Death rate in select medium counties, 2006-2020") +
    xlab("Year") + ylab("Death rate (per 100,000)")
```



Sanity checks

Compare population counts to ACS5 counts Vast majority of counties appear colinear but there are some exceptions

```
acs2009to2020 <- read_csv("data/acs2009to2020.csv")
```

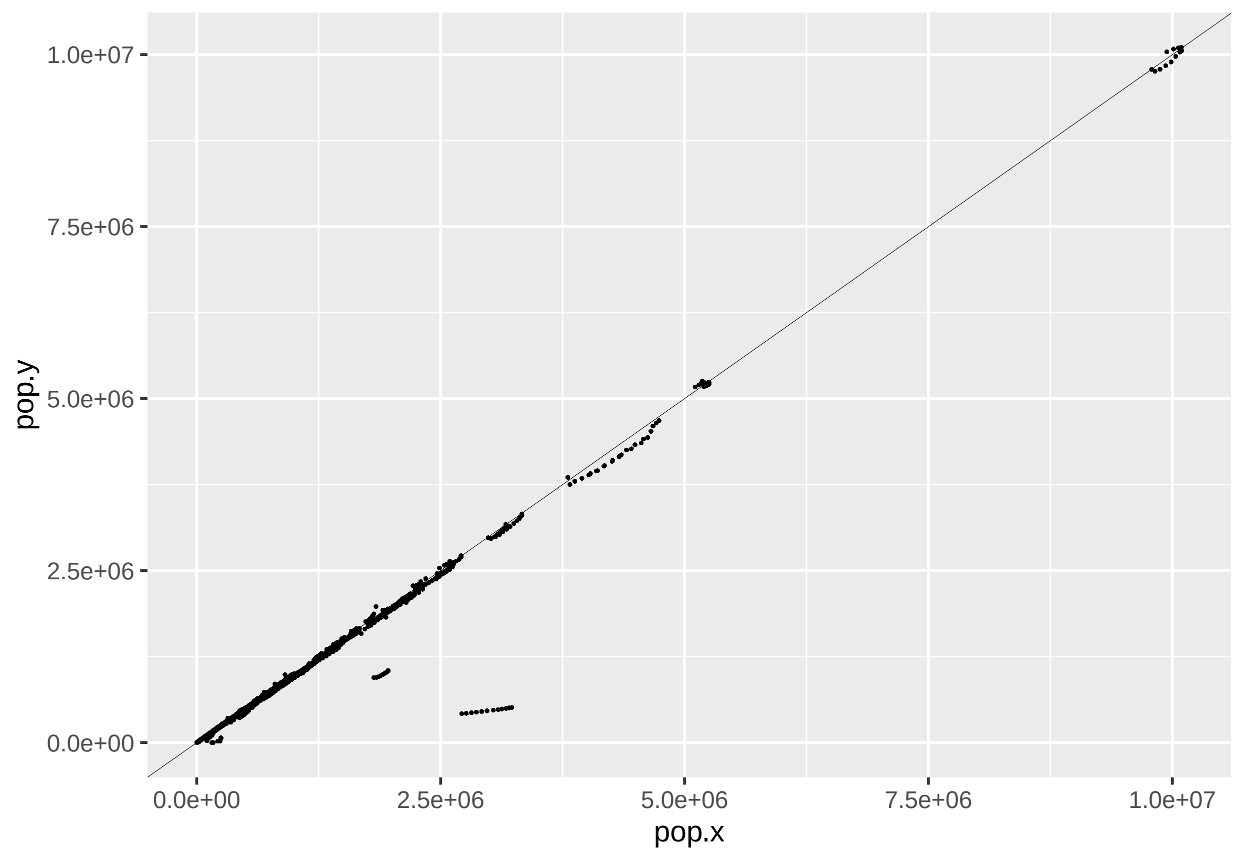
```
## Rows: 38646 Columns: 58
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): GEOID, NAME
## dbl (56): year, pop, male, male_u15, male_15to19, male_20to24, male_25to29, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
COD_ACS <-
    COD %>%
    filter(year >= 2009) %>%
    left_join(
        acs2009to2020,
        by=c("fipsihme" = "GEOID", "year")
    )
```

```
## filter: removed 111,924 rows (20%), 447,576 rows remaining
## left_join: added 57 columns (pop.x, NAME, pop.y, male, male_u15, ...)
##            > rows only in x       144
##            > rows only in y  (  1,360)
##            > matched rows     447,432
##            >                 =========
##            > rows total       447,576
```

```
COD_ACS %>%
    ggplot(aes(x = pop.x, y = pop.y)) +
    geom_point(size=.1) +
    geom_abline(slope = 1, linewidth = .1)
```

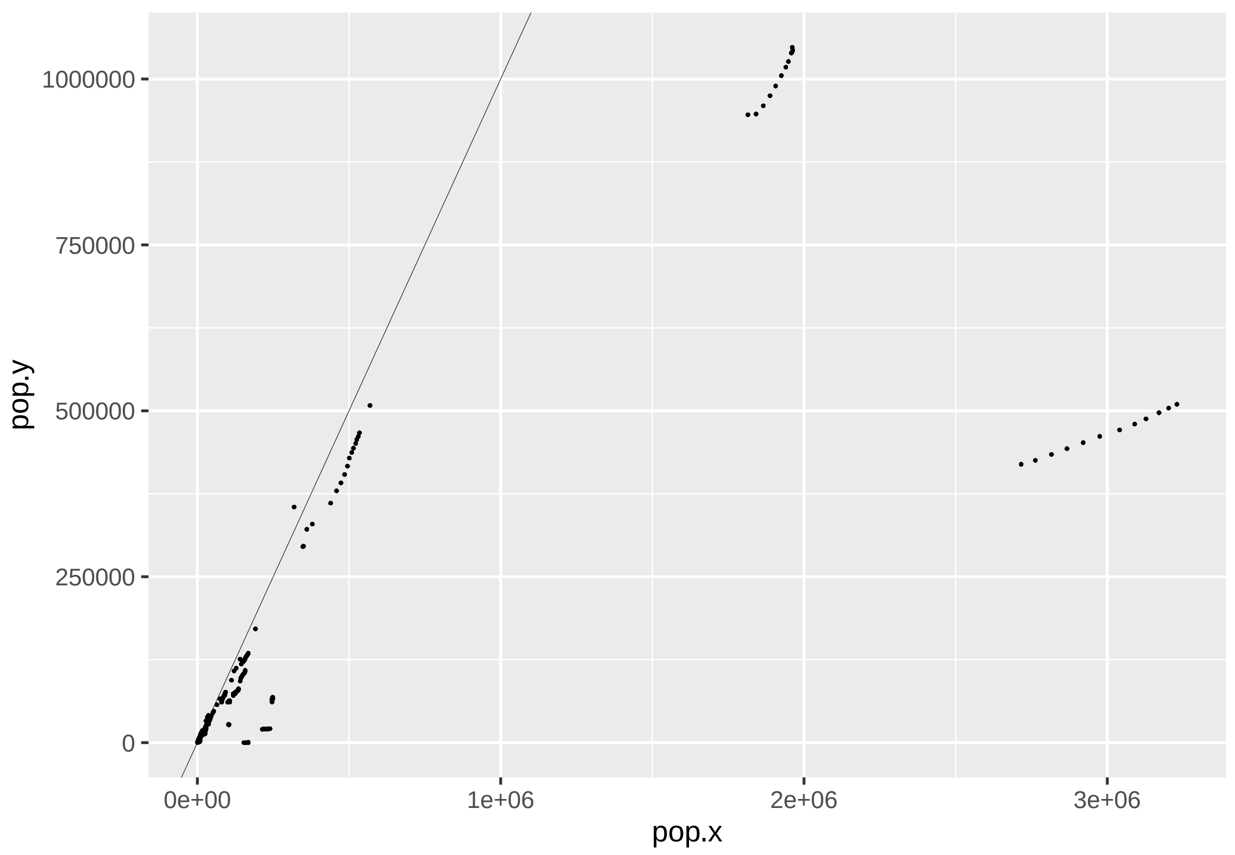
```
## Warning: Removed 144 rows containing missing values (`geom_point()`).
```



#### Investigate non-collinear counties Set a 10% margin of error

```
COD_ACS %>%
    filter(pop.y < .9 * pop.x | pop.y > 1.1 * pop.x) %>%
    ggplot(aes(x = pop.x, y = pop.y)) +
    geom_point(size=.1) +
    geom_abline(slope = 1, linewidth = .1)
```

```
## filter: removed 439,188 rows (98%), 8,388 rows remaining
```



```
pop_mismatch <-
    COD_ACS %>%
    filter(pop.y < .9 * pop.x | pop.y > 1.1 * pop.x)
```

```
## filter: removed 439,188 rows (98%), 8,388 rows remaining
```

```
#shows that 2009 (and that ACS grouping) has more of these mismatches
pop_mismatch %>%
    dplyr::select(fipsihme, year, NAME, pop.x, pop.y) %>%
    distinct() %>%
    group_by(year) %>%
    summarise(n = n())
```

```
## distinct: removed 7,689 rows (92%), 699 rows remaining

## group_by: one grouping variable (year)

## summarise: now 12 rows and 2 columns, ungrouped

## # A tibble: 12 x 2
##     year     n
##    <dbl> <int>
##  1  2009   139
##  2  2010    53
##  3  2011    50
##  4  2012    46
##  5  2013    50
##  6  2014    51
##  7  2015    49
##  8  2016    57
##  9  2017    57
## 10  2018    48
## 11  2019    50
## 12  2020    49
```

```
#AK and CO have a disproportionate amount of these mismatch counties
pop_mismatch %>%
    dplyr::select(fipsihme, year, NAME, pop.x, pop.y) %>%
    mutate(statefip = str_sub(fipsihme, 1, 2)) %>%
    left_join(STATEFP, by = c("statefip" = "FIPS")) %>%
    distinct() %>%
    group_by(state) %>%
    summarise(n = n())
```

```
## mutate: new variable 'statefip' (character) with 34 unique values and 0% NA

## left_join: added one column (state)

##              > rows only in x        0

##              > rows only in y  (    21)

##              > matched rows      8,388

##              >                 =======

##              > rows total        8,388

## distinct: removed 7,689 rows (92%), 699 rows remaining

## group_by: one grouping variable (state)

## summarise: now 34 rows and 2 columns, ungrouped

## # A tibble: 34 x 2
##    state     n
##    <chr> <int>
##  1 AK       63
##  2 AL        1
##  3 AZ       15
##  4 CA        2
##  5 CO       53
##  6 FL        4
##  7 GA       25
##  8 HI       12
##  9 IA        1
## 10 ID       18
## # ... with 24 more rows
## # i Use `print(n = ...)` to see more rows
```

Counties where deaths $>$ 50% of population Saw some crop up when doing an analysis earlier but with this cleaner process, those seem to have gone away (good)

```
COD %>%
    filter(n_deaths >= pop * .5)
```

```
## filter: removed all rows (100%)
```

```
## # A tibble: 0 x 6
## # ... with 6 variables: fipsihme <chr>, year <dbl>, month <dbl>, pop <dbl>,
## #   n_deaths <dbl>, rate <dbl>
## # i Use ‘colnames()‘ to see all variable names
```