Week 5: Final

COVID Data Analysis

2/27/2022

Cleaning Data

We are going to be looking at COVID 19 cases in the US to see if there is a state that can be used as a predictor for other states. First we need to import and clean up our data. We're going to use the Johns Hopkins data sets for the US since it's a reliable source. Then We're going to change the columns for each date into a row for each date with the new column value being number of cases. We'll also remote state location information since we don't need that and reformat the date field into a date datatype. Next we'll join the information on number of cases with the information on number of deaths into a single data set.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## 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()
```

```
library(lubridate)
```

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

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

```
url_in <- "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covi
file_names <- c("time_series_covid19_confirmed_US.csv",
    "time_series_covid19_deaths_US.csv")
urls <- str_c(url_in, file_names)
US_cases <- read_csv(urls[1])
```

```
## Rows: 3342 Columns: 782
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (776): UID, code3, FIPS, Lat, Long_, 1/22/20, 1/23/20, 1/24/20, 1/25/20,...
##
## 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.
```

```
US_deaths <- read_csv(urls[2])
```

```
## Rows: 3342 Columns: 783
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (6): iso2, iso3, Admin2, Province_State, Country_Region, Combined_Key
## dbl (777): UID, code3, FIPS, Lat, Long_, Population, 1/22/20, 1/23/20, 1/24/...
##
## 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.
```

```
US_cases <- US_cases %>%
    pivot_longer(cols = -(UID:Combined_Key), names_to = "date", values_to = "cases") %>%
    select(Admin2:cases) %>%
    mutate(date = mdy(date)) %>%
    select(-c(Lat, Long_))
US_deaths <- US_deaths %>%
    pivot_longer(cols = -(UID:Population), names_to = "date", values_to = "deaths") %>%
    select(Admin2:deaths) %>%
    mutate(date = mdy(date)) %>%
    select(-c(Lat, Long_))
US <- US_cases %>%
    full_join(US_deaths)
```

```
## Joining, by = c("Admin2", "Province_State", "Country_Region", "Combined_Key",
## "date")
```

```
summary(US)
```

```
##     Admin2          Province_State     Country_Region     Combined_Key
##  Length:2576682     Length:2576682     Length:2576682     Length:2576682
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##       date                cases           Population          deaths
##  Min.   :2020-01-22   Min.   :      0   Min.   :       0   Min.   :    0.0
##  1st Qu.:2020-08-01   1st Qu.:     84   1st Qu.:    9917   1st Qu.:    1.0
##  Median :2021-02-10   Median :   1071   Median :   24892   Median :   18.0
##  Mean   :2021-02-10   Mean   :   7515   Mean   :   99604   Mean   :  126.4
##  3rd Qu.:2021-08-22   3rd Qu.:   4205   3rd Qu.:   64979   3rd Qu.:   75.0
##  Max.   :2022-03-02   Max.   :2799169   Max.   :10039107   Max.   :30853.0
```

Currently the state data is broken down by county, so we're going to change those into totals for the entire state and look at totals for the entire country.

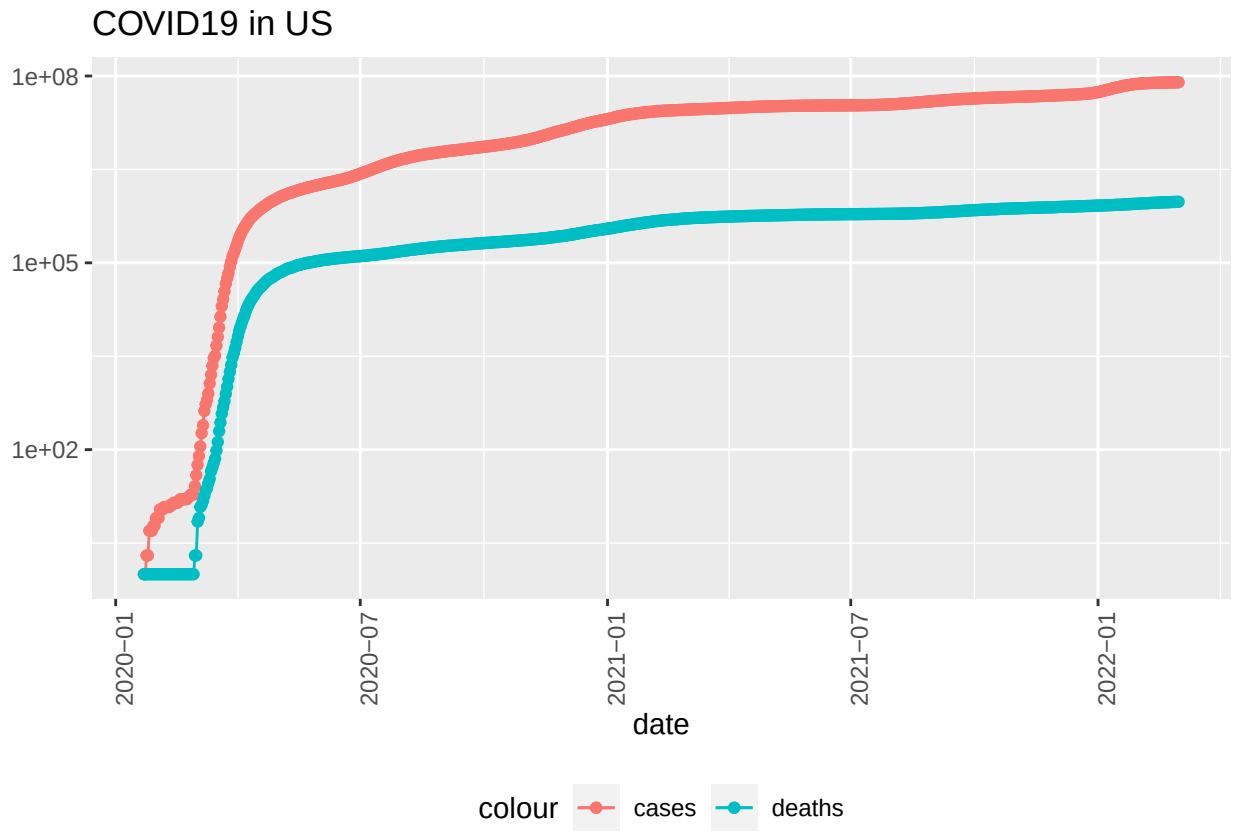
```
US_by_state <- US %>%
    group_by(Province_State, Country_Region, date) %>%
    summarize(cases = sum(cases), deaths = sum(deaths), Population = sum(Population)) %>%
    mutate(deaths_per_mill = deaths * 1e+06/Population) %>%
    select(Province_State, Country_Region, date, cases, deaths, deaths_per_mill,
        Population) %>%
    ungroup()
```

```
## `summarise()` has grouped output by 'Province_State', 'Country_Region'. You can
## override using the `.groups` argument.
```

```
US_totals <- US_by_state %>%
    group_by(Country_Region, date) %>%
    summarize(cases = sum(cases), deaths = sum(deaths), Population = sum(Population)) %>%
    mutate(deaths_per_mill = deaths * 1e+06/Population) %>%
    select(Country_Region, date, cases, deaths, deaths_per_mill, Population) %>%
    ungroup()
```

```
## `summarise()` has grouped output by 'Country_Region'. You can override using
## the `.groups` argument.
```

```
US_totals %>%
    filter(cases > 0) %>%
    ggplot(aes(x = date, y = cases)) + geom_line(aes(color = "cases")) +
    geom_point(aes(color = "cases")) + geom_line(aes(y = deaths, color = "deaths")) +
    geom_point(aes(y = deaths, color = "deaths")) + scale_y_log10() +
    theme(legend.position = "bottom", axis.text.x = element_text(angle = 90)) +
    labs(title = "COVID19 in US", y = NULL)
```



Taking a quick look at the graph of US totals (on a logarithmic scale) matches the expectations of what we've seen from other sources. Since our data looks good, let's add some extra fields that we may want for analysis. The cases and deaths are a running total, so let's add fields for the day to day change. I'd also like to add fields for cases and deaths per thousand for a finer grained analysis that is better suited to low population states.

Since we've added those fields, let's take a look at the US totals and use cases per thousand to predict deaths per thousand against all the states. To avoid visual clutter, I'm leaving all the states as blue since I just want to see an overall fit.

```
US_by_state <- US_by_state %>%
    mutate(new_cases = cases - lag(cases), new_deaths = deaths - lag(deaths),
        cases_per_thou = cases * 1000/Population, deaths_per_thou = deaths *
            1000/Population) %>%
    filter(cases > 0, Population > 0)
summary(US_by_state)
```

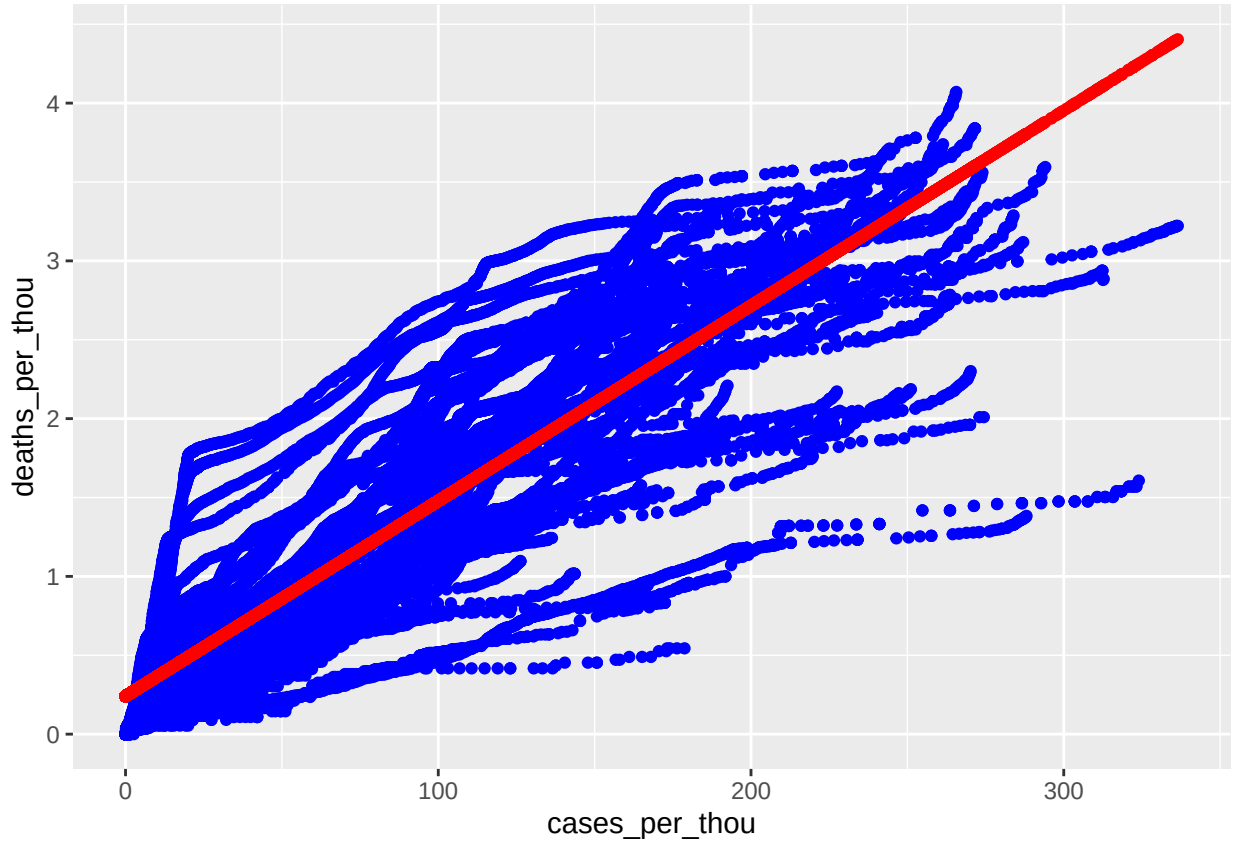
```
##  Province_State     Country_Region          date                cases
##  Length:40207       Length:40207       Min.   :2020-01-22   Min.   :      1
##  Class :character   Class :character   1st Qu.:2020-09-03   1st Qu.:  29574
##  Mode  :character   Mode  :character   Median :2021-03-05   Median : 170567
##                                        Mean   :2021-03-04   Mean   : 481614
##                                        3rd Qu.:2021-09-04   3rd Qu.: 594174
##                                        Max.   :2022-03-02   Max.   :8987156
##      deaths        deaths_per_mill    Population         new_cases
##  Min.   :    0.0   Min.   :   0.0   Min.   :   55144   Min.   :-1643049
```

```
##  1st Qu.:  586.5   1st Qu.: 284.2   1st Qu.: 1359711   1st Qu.:      63
##  Median : 2861.0   Median :1049.4   Median : 3956971   Median :     472
##  Mean   : 8101.1   Mean   :1190.1   Mean   : 6090753   Mean   :    1928
##  3rd Qu.: 9830.5   3rd Qu.:1912.6   3rd Qu.: 7278717   3rd Qu.:    1639
##  Max.   :85827.0   Max.   :4070.7   Max.   :39512223   Max.   :  206894
##    new_deaths        cases_per_thou   deaths_per_thou
##  Min.   :-18859.00   Min.   :  0.00   Min.   :0.0000
##  1st Qu.:     0.00   1st Qu.: 13.05   1st Qu.:0.2842
##  Median :     5.00   Median : 70.15   Median :1.0494
##  Mean   :    23.27   Mean   : 76.71   Mean   :1.1901
##  3rd Qu.:    23.00   3rd Qu.:116.87   3rd Qu.:1.9126
##  Max.   :  2441.00   Max.   :336.46   Max.   :4.0707
```

```
mod <- lm(deaths_per_thou ~ cases_per_thou, data = US_by_state)
summary(mod)
```

```
##
## Call:
## lm(formula = deaths_per_thou ~ cases_per_thou, data = US_by_state)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6706 -0.2402 -0.1060  0.2459  1.3149
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    2.409e-01  3.527e-03   68.31   <2e-16 ***
## cases_per_thou 1.237e-02  3.436e-05  360.11   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4699 on 40205 degrees of freedom
## Multiple R-squared:  0.7633, Adjusted R-squared:  0.7633
## F-statistic: 1.297e+05 on 1 and 40205 DF,  p-value: < 2.2e-16
```

```
US_w_pred <- US_by_state %>%
    mutate(pred = predict(mod))
US_w_pred %>%
    ggplot() + geom_point(aes(x = cases_per_thou, y = deaths_per_thou), color = "blue") +
    geom_point(aes(x = cases_per_thou, y = pred), color = "red")
```



Analysis

Now I would like to see if we can cluster states together by models. I am going to day this by creating a linear model for each state, then applying it to all the states and seeing how well the actual deaths correlate with the predicted deaths for that model. Since this data set also includes US territories, we're going to end up with 56 models applied to 56 states or territories.

```
options(warn = -1)
States <- split(US_by_state, US_by_state$Province_State)
for (i in 1:length(States)) {
    States[i][[1]] <- States[i][[1]] %>%
        mutate(cases_per_thou = as.numeric(cases_per_thou)) %>%
        mutate(deaths_per_thou = as.numeric(deaths_per_thou))
}

models = c()
states = c()
corrs = c()

for (state in States) {
    m <- lm(deaths_per_thou ~ cases_per_thou, data = state)
    name <- state$Province_State[1]
    for (sstate in States) {
        this_s <- predict(m, sstate)
        c <- cor(sstate$deaths_per_thou, this_s)
```

```
        models <- append(models, name)
        states <- append(states, sstate$Province_State[1])
        corrs <- append(corrs, c)
    }
}
results <- tibble(Model = models, State = states, Corr = corrs)
summary(results)
```

```
##     Model              State               Corr
##  Length:3136        Length:3136        Min.   :0.8293
##  Class :character   Class :character   1st Qu.:0.9567
##  Mode  :character   Mode  :character   Median :0.9739
##                                        Mean   :0.9635
##                                        3rd Qu.:0.9849
##                                        Max.   :0.9934
##                                        NA's   :111
```

We can see that our correlations range from very close matches (greater than 99%) to poor matches (84%). To cluster states into predictor sets, I'm going to take the state and model with the highest correlation out of the results. Those states will be added to a set together. Since those two states are in a set now, I'm going to remove them from the list of remaining states that need to be added to a set. From here I will loop until the set of remaining states is empty. I'm going to pull the next highest correlating model to state out of the results set. If the state used to create the model is already in a set, the new state is added to that set and removed from the results list. If the modeling state isn't in a set yet, we'll create a new set of states with those two. We will also need to add the special case for when a state has no correlating model (in this case, it's a state with no deaths) where it will be added to it's own unique set. You can think of this as a weighted graph partitioning algorithm where the states are the nodes and the edges are the models with weights corresponding to the correlation. This is a greedy algorithm (always taking the next best fit) so we may not end up with optimal set results. In other words, we could have fewer sets if we analyze the correlations differently or more sets if we do a different comparison (such as how well it compares to all states in the existing set, not just the generating model).

```
sets <- tibble(State = unique(states), Set = 0)
count <- 1
while (dim(results) != c(0, 3)) {
    next_match <- results %>%
        slice_max(order_by = Corr)
    if (dim(next_match) == c(0, 3)) {
        next_match <- results %>%
            slice(1)
        sets <- rows_update(sets, tibble(State = next_match$State, Set = count))
        count <- count + 1
        results <- results %>%
            filter(State != next_match$State)
        next
    } else {
        next_match <- next_match %>%
            slice(1)
    }
    idx <- sets %>%
        filter(State == next_match$Model)
    if (idx$Set == 0) {
        sets <- rows_update(sets, tibble(State = next_match$Model, Set = count))
```

```
        sets <- rows_update(sets, tibble(State = next_match$State, Set = count))
        count <- count + 1
        results <- results %>%
            filter(State != next_match$State) %>%
            filter(State != next_match$Model)
    } else {
        sets <- rows_update(sets, tibble(State = next_match$State, Set = idx$Set))
        results <- results %>%
            filter(State != next_match$State)
    }
}
```

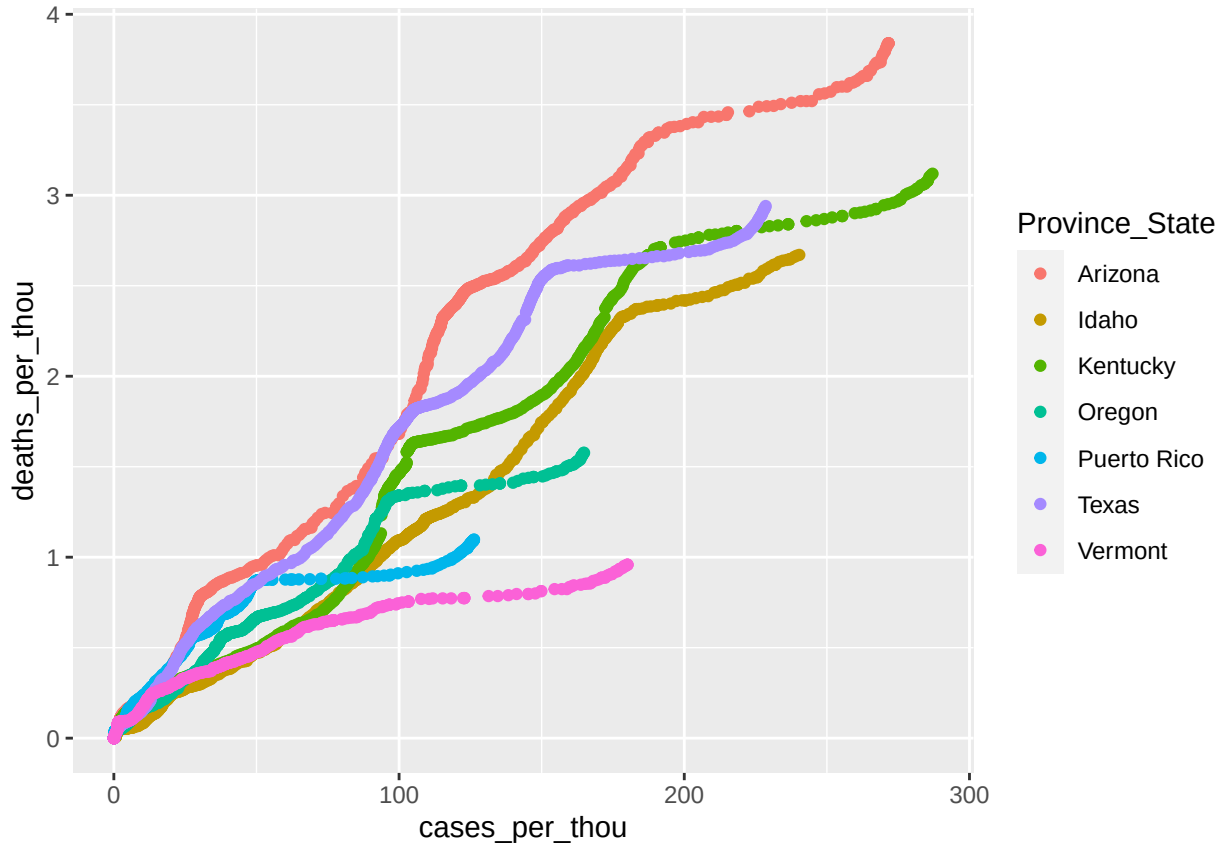
```
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
```

```
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
## Matching, by = "State"
```

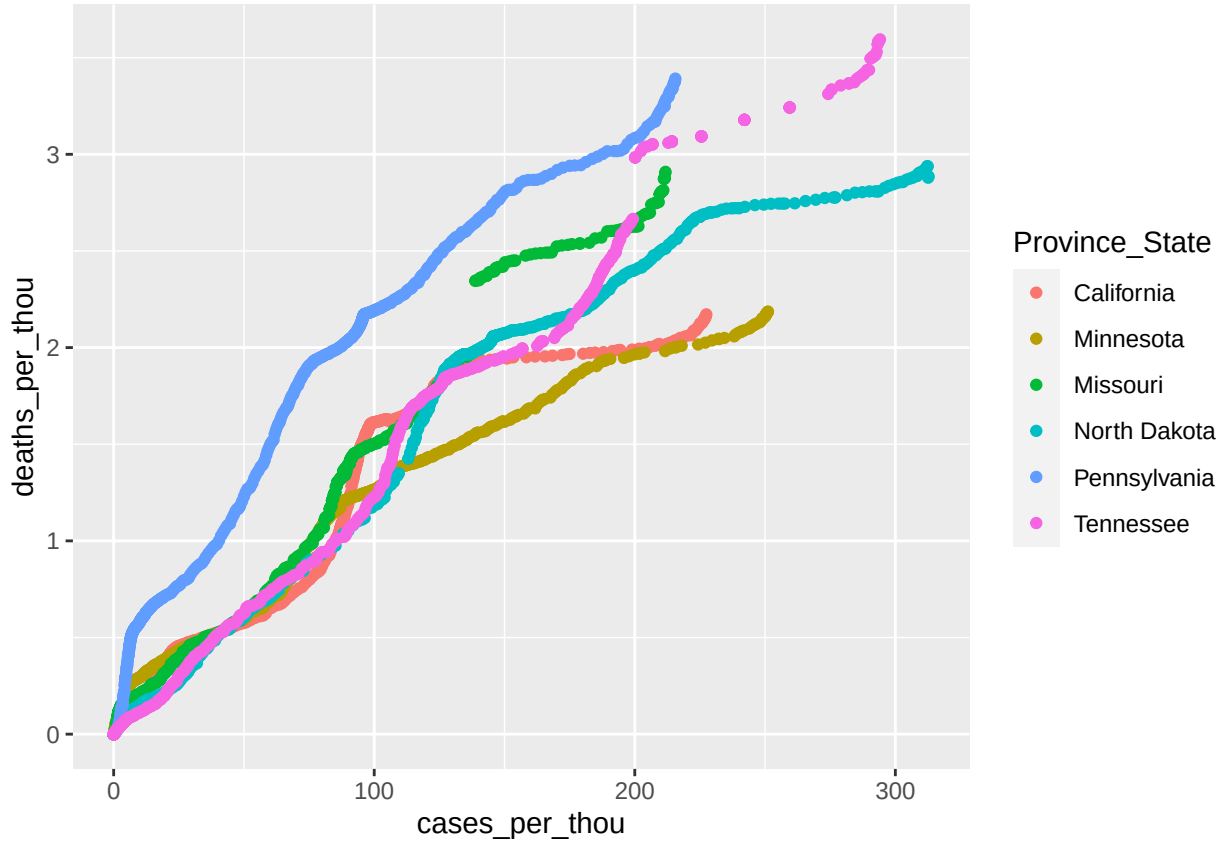
Now that we've created our sets, let's look at how the states were grouped together and the graph of their cases and deaths to see how the models look.

```
for (i in 1:(count - 1)) {
    curr_set <- sets %>%
        filter(Set == i) %>%
        select(State)
    cat(str_c("Group ", as.character(i), ":\n"))
    cat(curr_set$State, sep = "\n")
    cat("\n")
    curr_data <- curr_data <- US_by_state %>%
        filter(Province_State %in% curr_set$State)
    print(ggplot(curr_data, aes(x = cases_per_thou, y = deaths_per_thou,
        color = Province_State)) + geom_point())
}
```

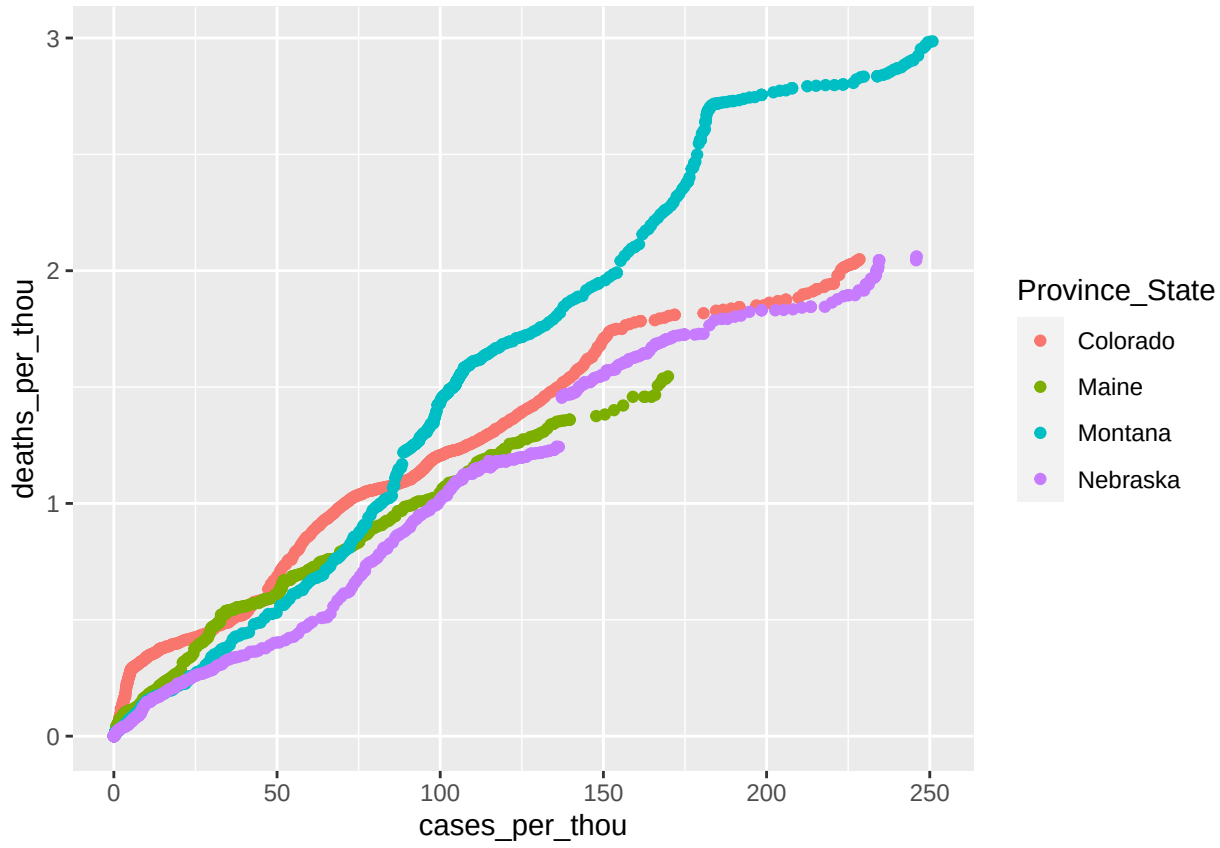
```
## Group 1:
## Arizona
## Idaho
## Kentucky
## Oregon
## Puerto Rico
## Texas
## Vermont
```



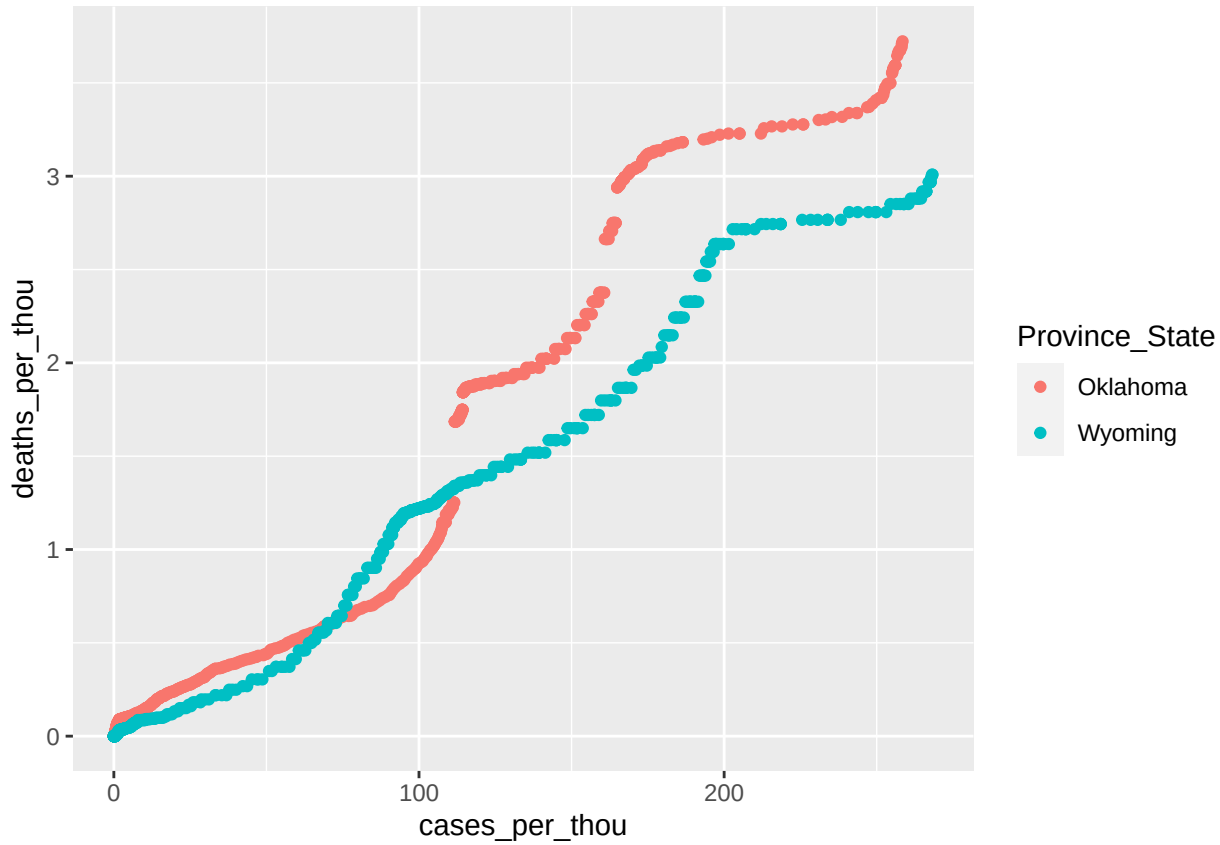

```
## Group 2:
## California
## Minnesota
## Missouri
## North Dakota
## Pennsylvania
## Tennessee
```



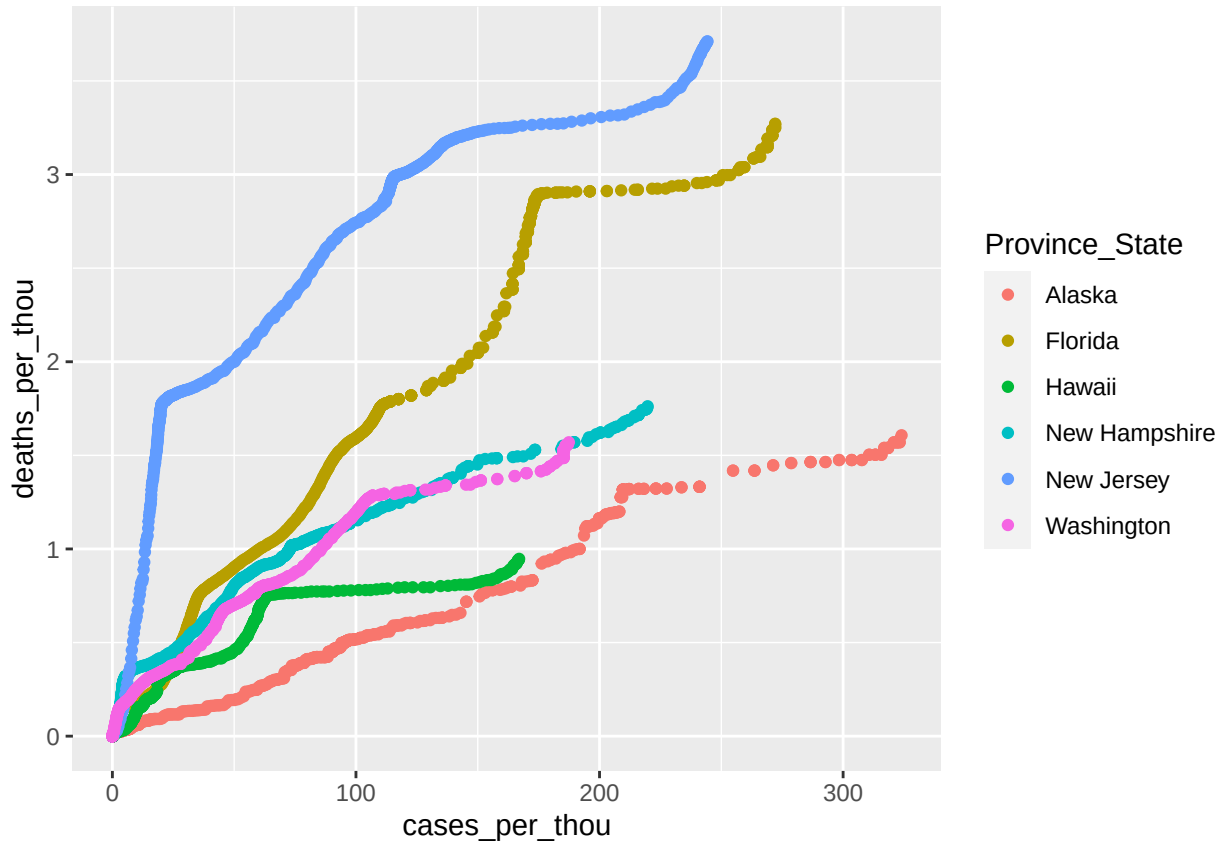

```
## Group 3:
## Colorado
## Maine
## Montana
## Nebraska
```



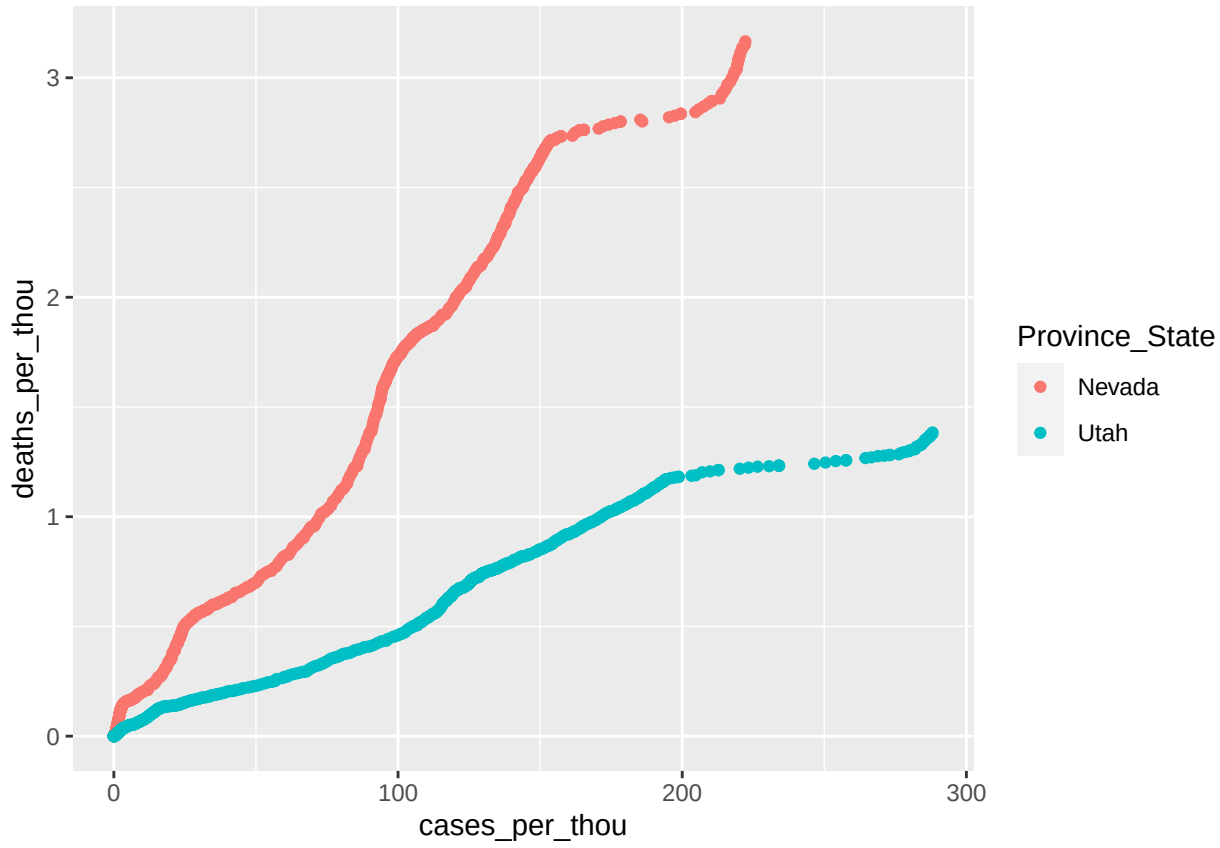

```
## Group 4:
## Oklahoma
## Wyoming
```



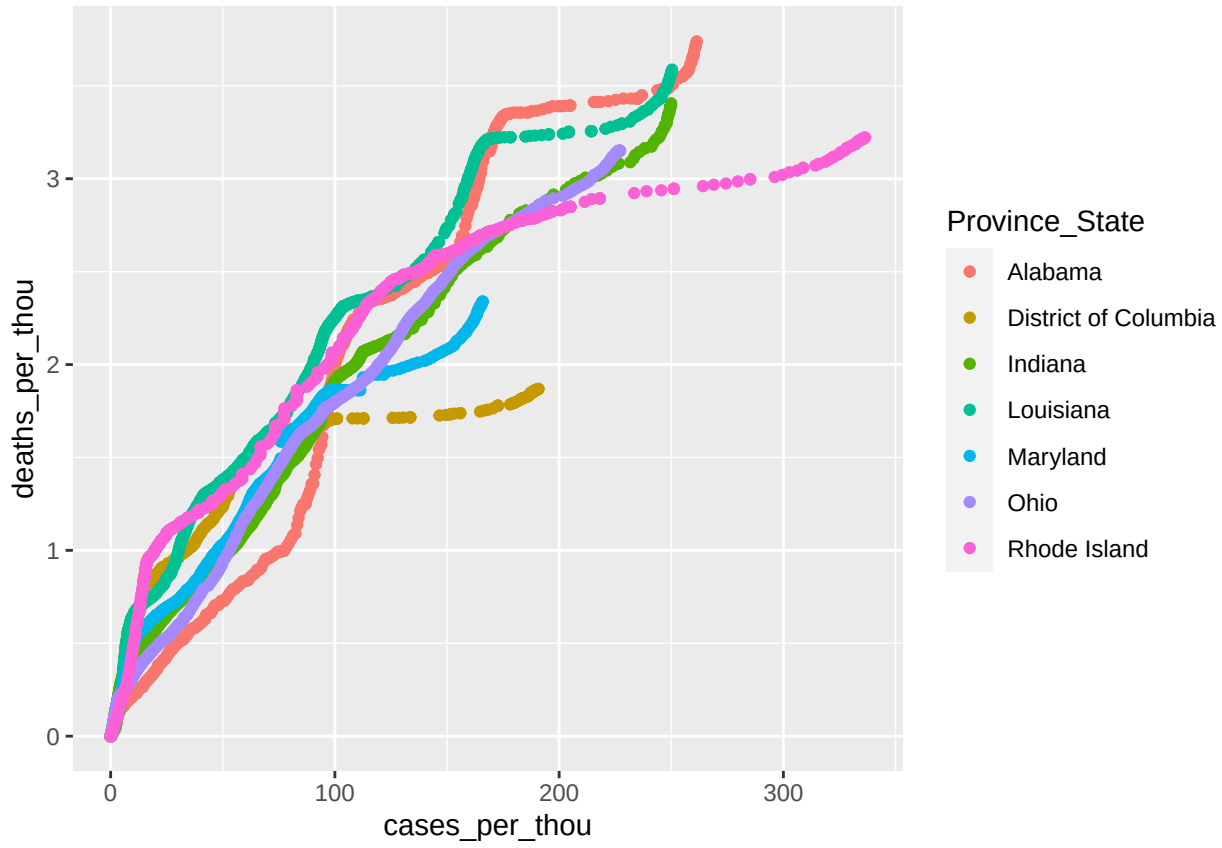

```
## Group 5:
## Alaska
## Florida
## Hawaii
## New Hampshire
## New Jersey
## Washington
```



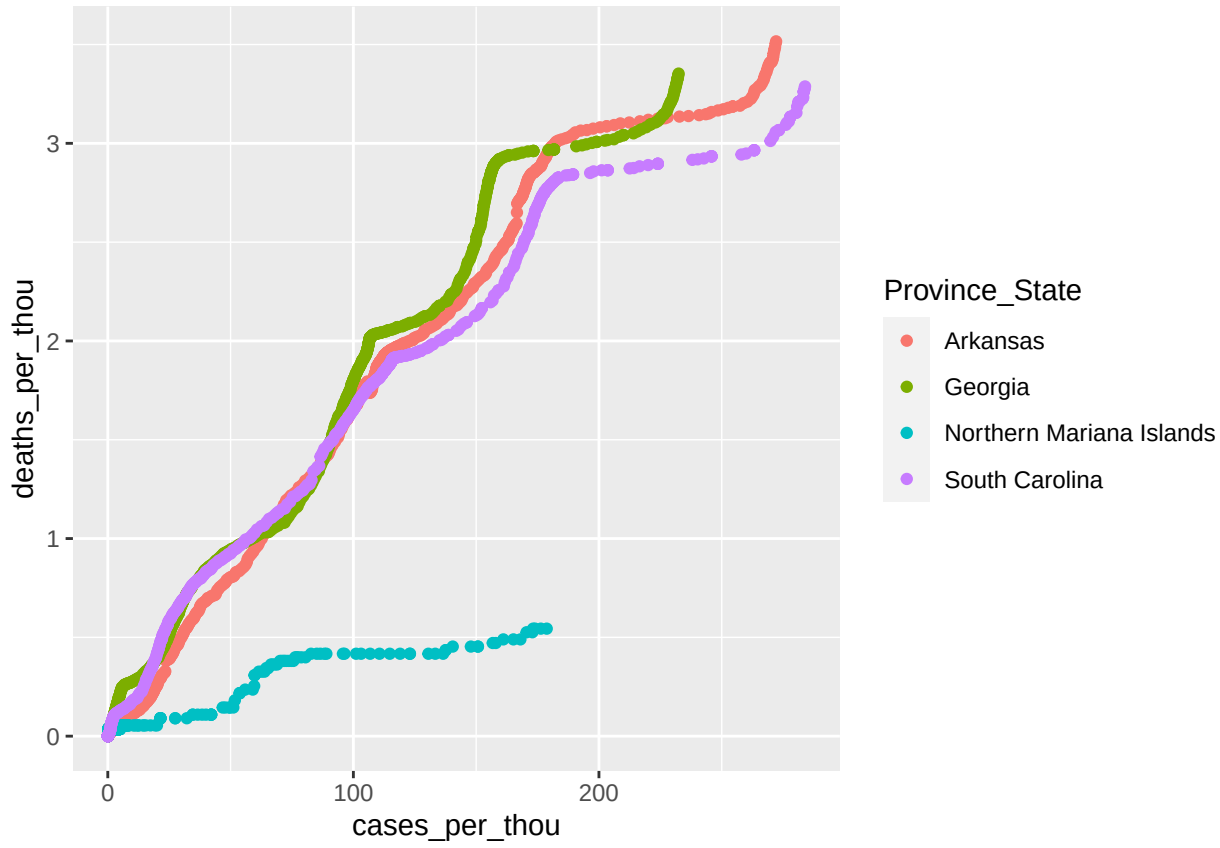

```
## Group 6:
## Nevada
## Utah
```



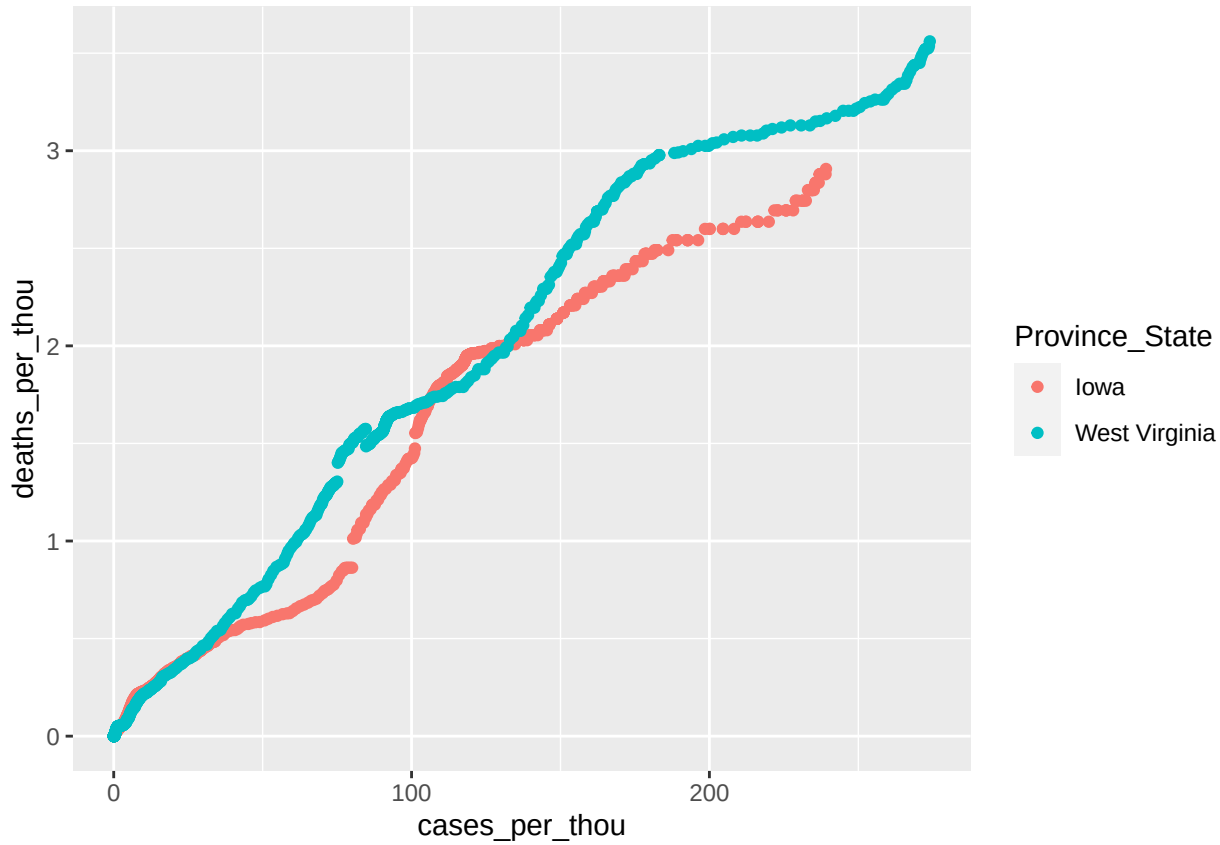

```
## Group 7:
## Alabama
## District of Columbia
## Indiana
## Louisiana
## Maryland
## Ohio
## Rhode Island
```



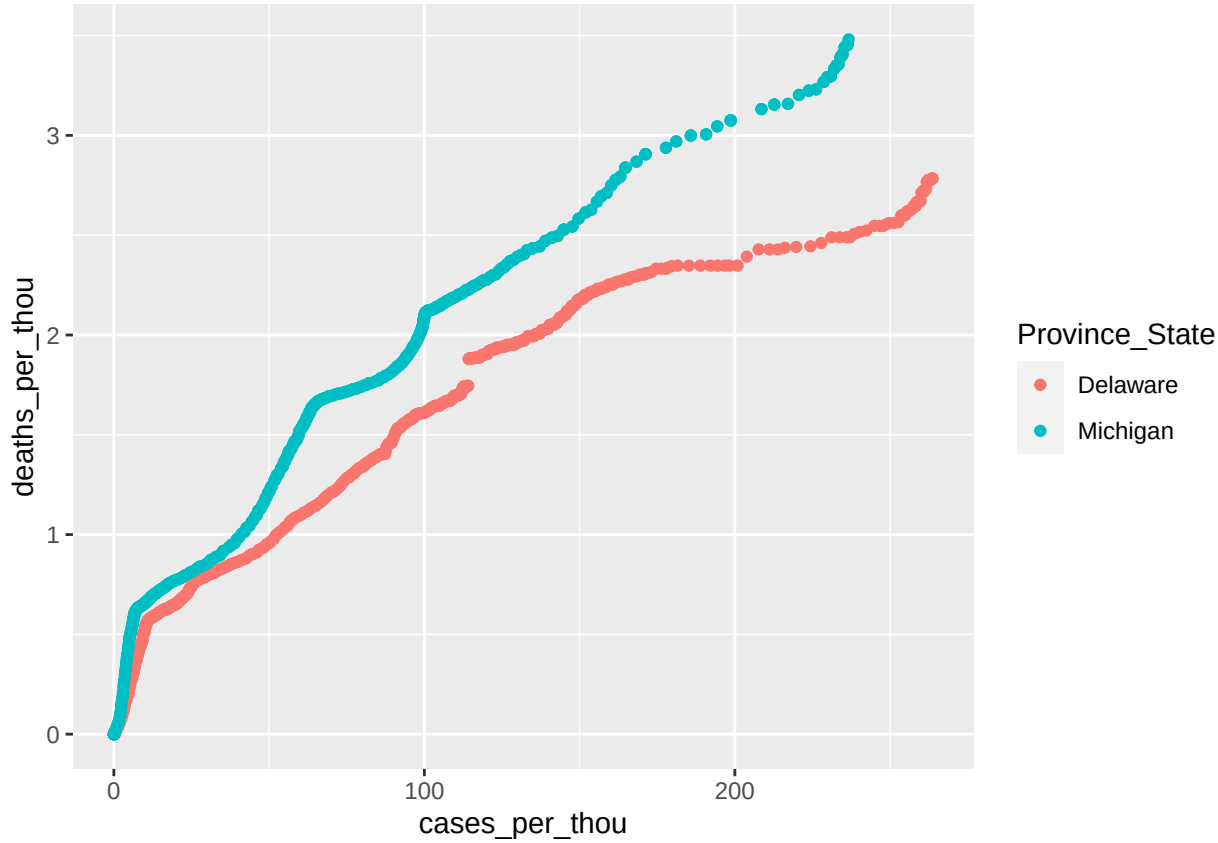

```
## Group 8:
## Arkansas
## Georgia
## Northern Mariana Islands
## South Carolina
```



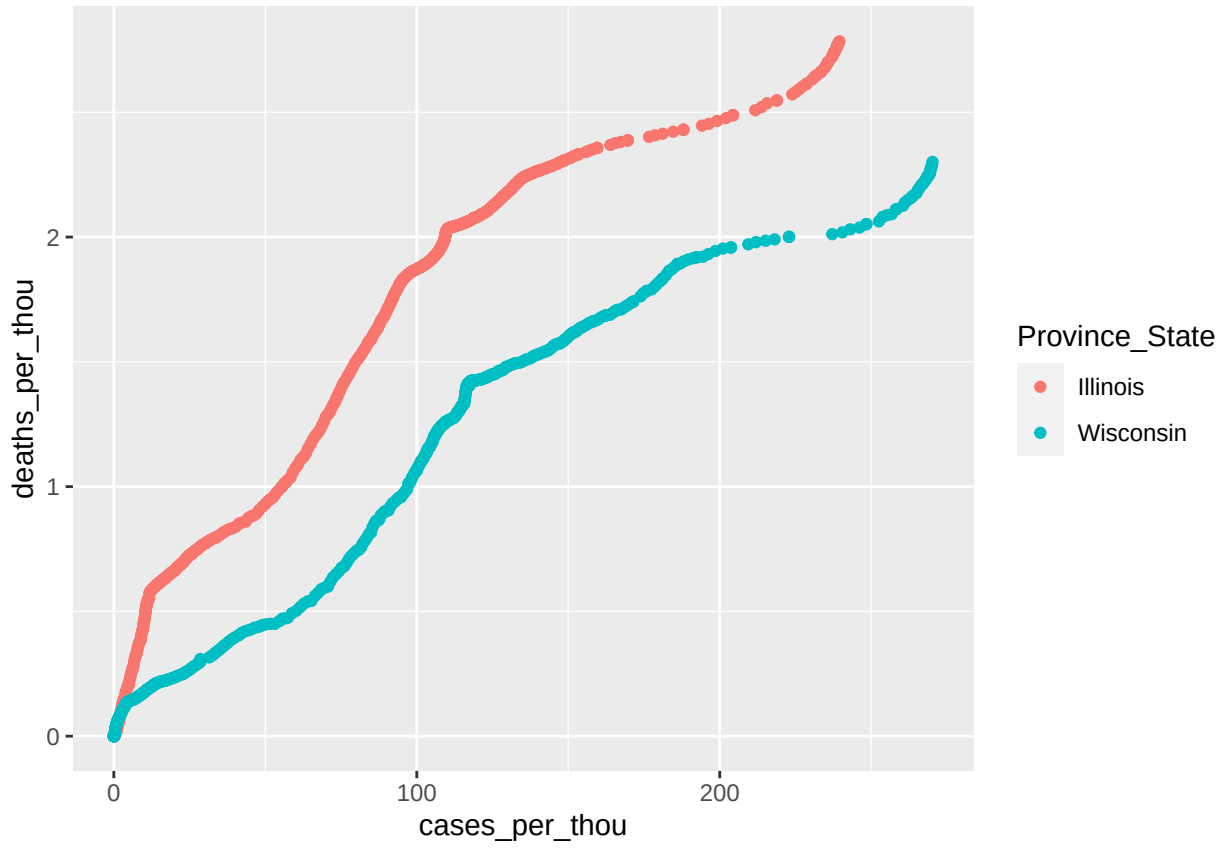

```
## Group 9:
## Iowa
## West Virginia
```



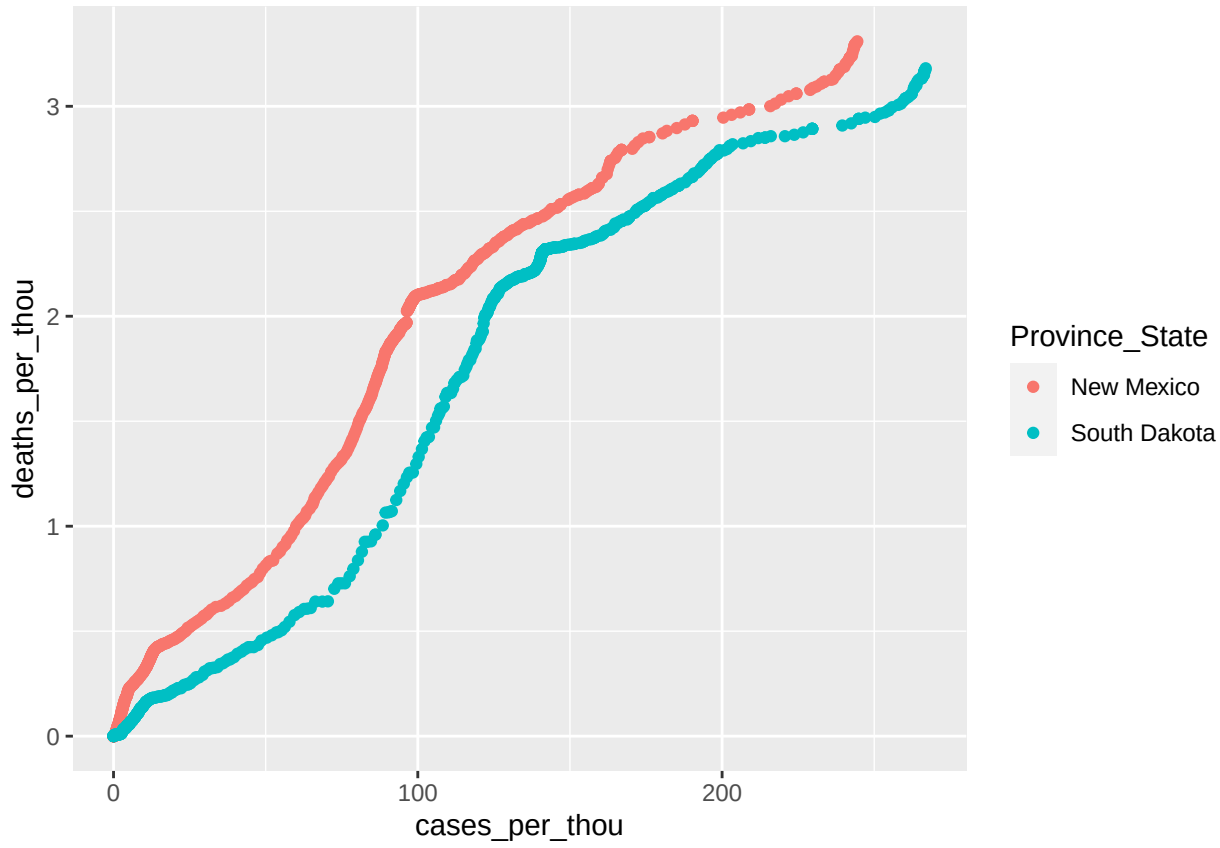

```
## Group 10:
## Delaware
## Michigan
```



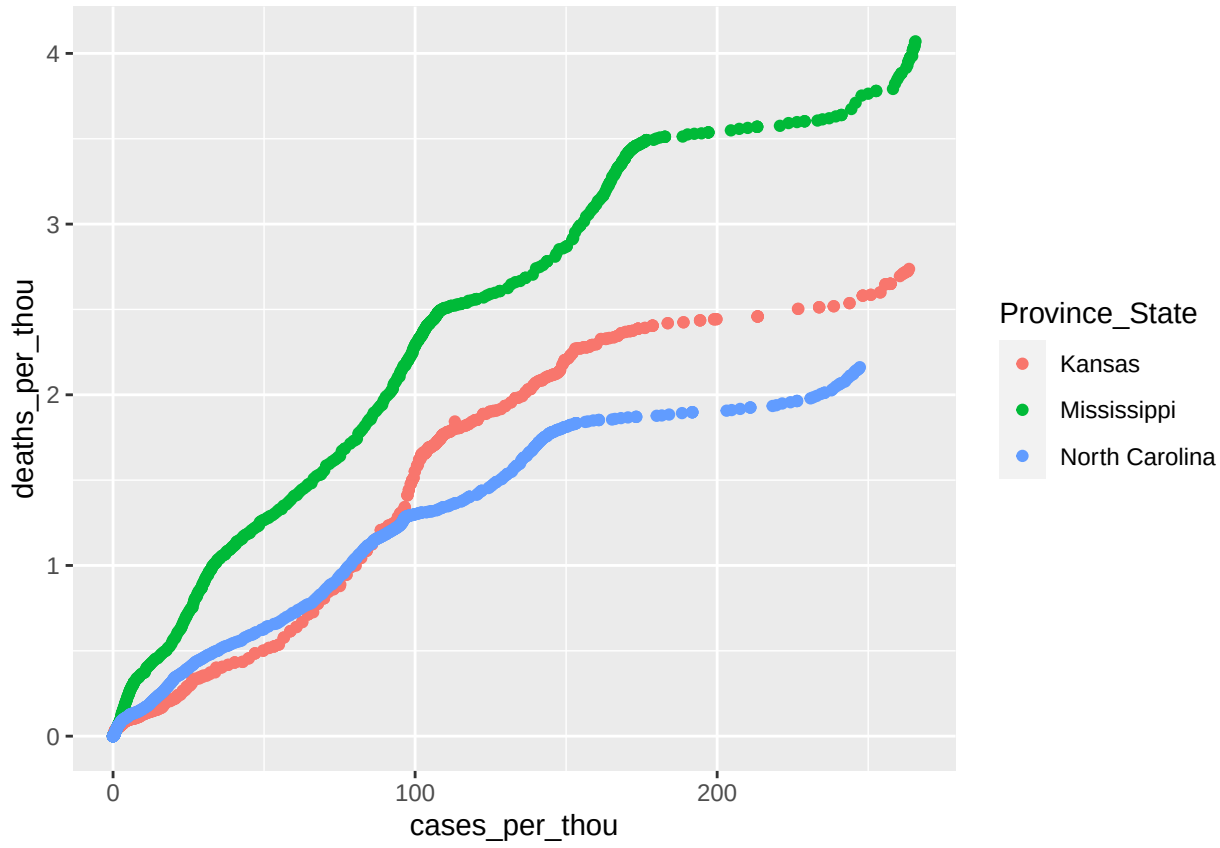

```
## Group 11:
## Illinois
## Wisconsin
```



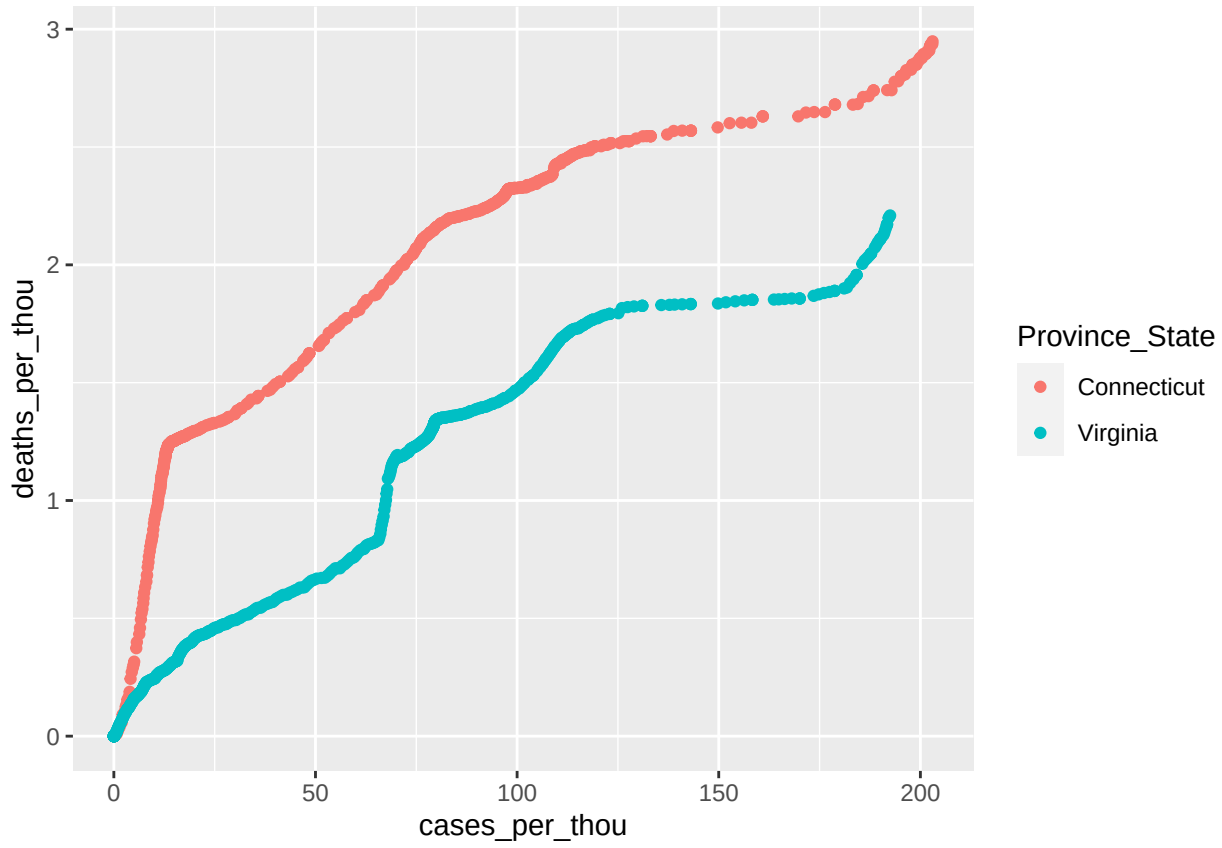

```
## Group 12:
## New Mexico
## South Dakota
```



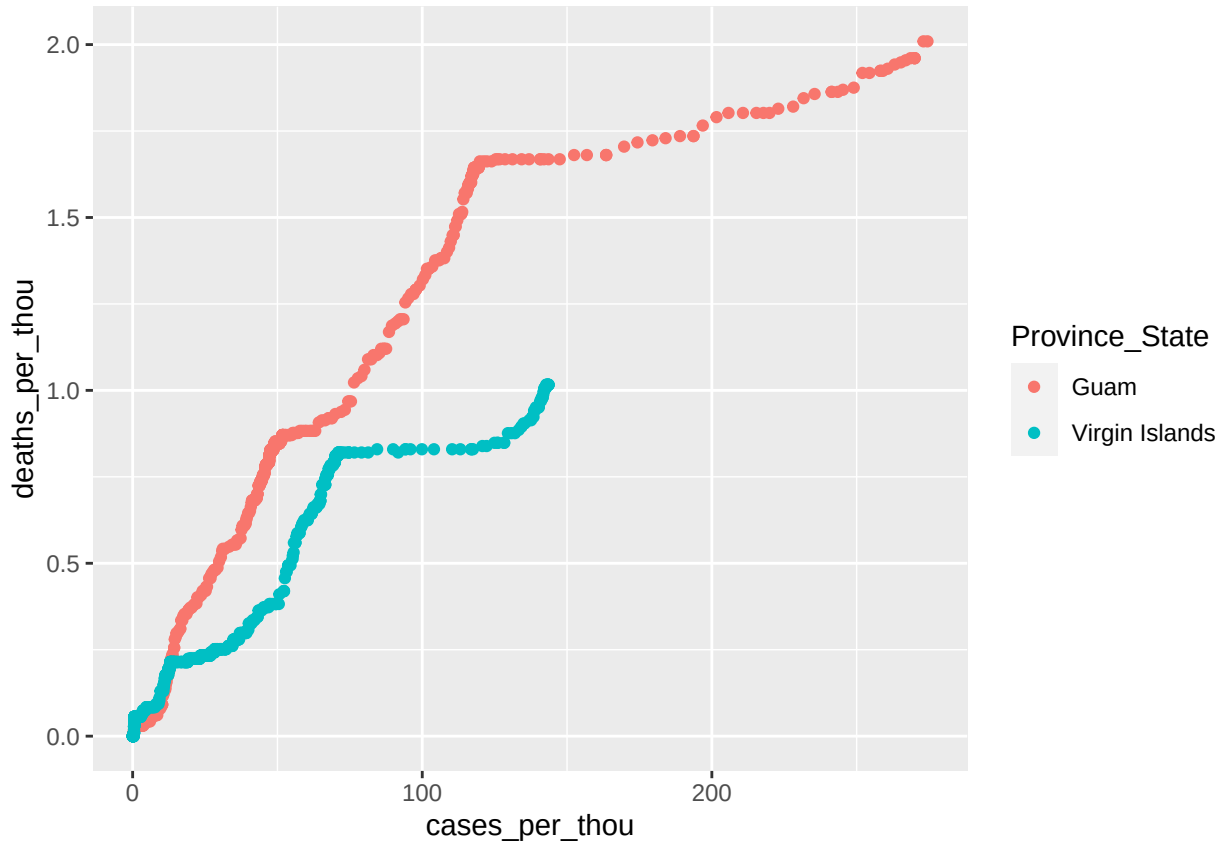

```
## Group 13:
## Kansas
## Mississippi
## North Carolina
```



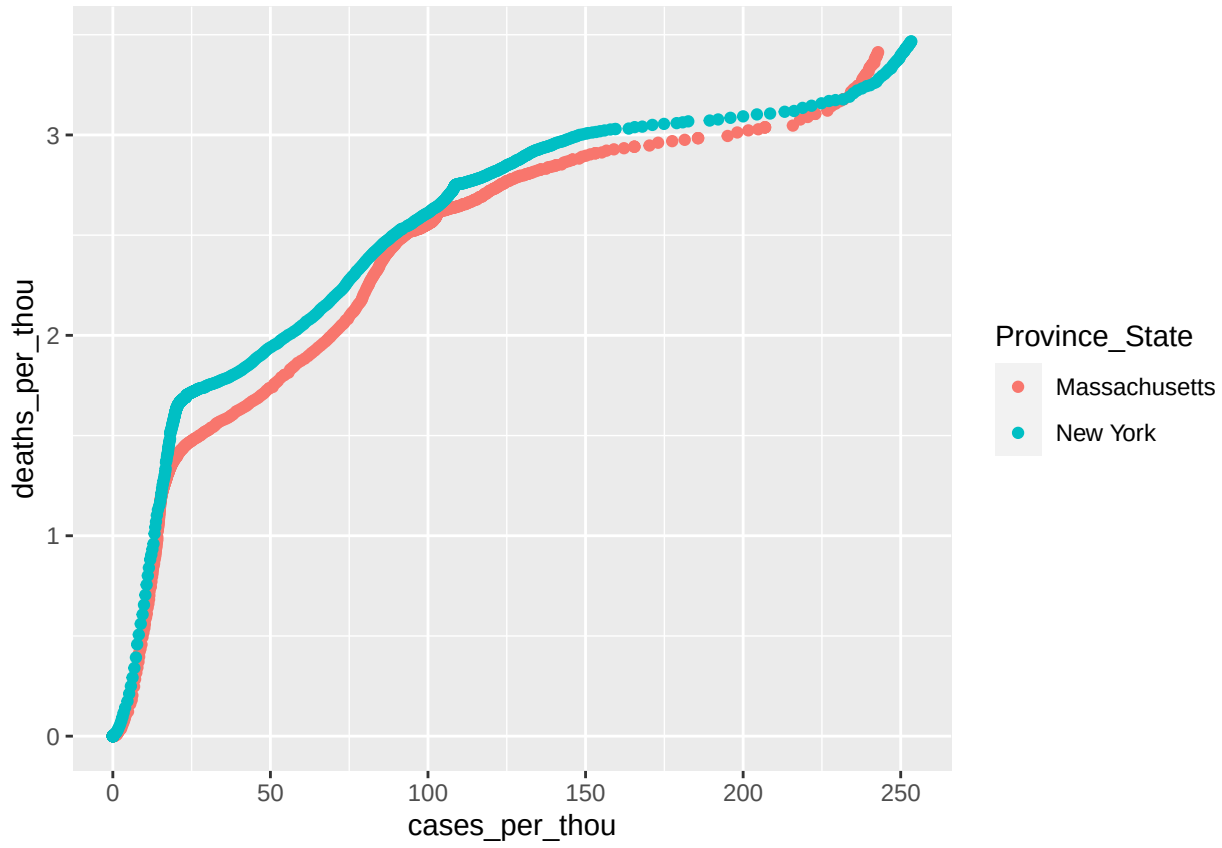

```
## Group 14:
## Connecticut
## Virginia
```



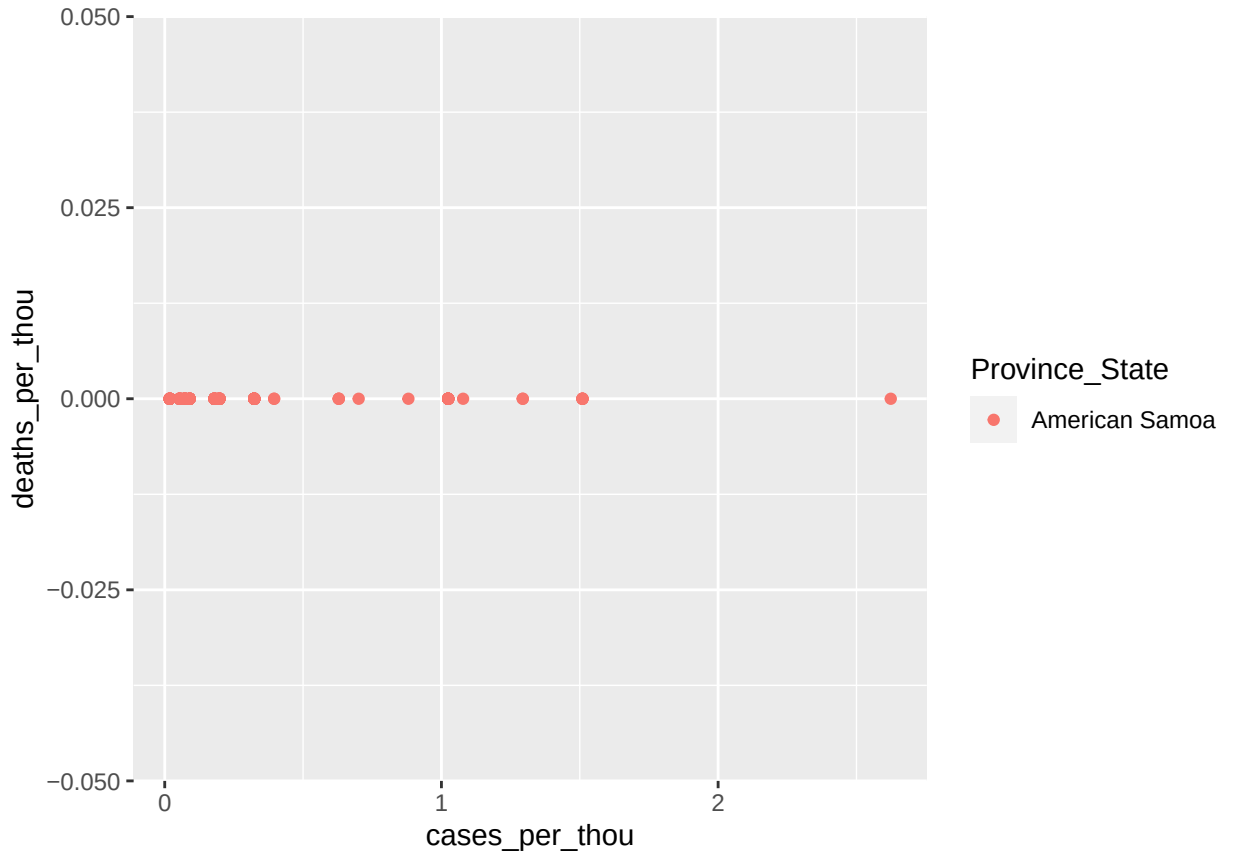

```
## Group 15:
## Guam
## Virgin Islands
```




```
## Group 16:
## Massachusetts
## New York
```




```
## Group 17:
## American Samoa
```




```
options(warn = 1)
```

For groups 1, 11, 13, and 14, while the offsets are slightly different, the curves of the actual cases looks very similar. Group 2, 3, 4, 7, 9, 10, 12, 15, and 16 are well clustered and seem like a set of states that could accurately predict the others. Group 5 is a bit more disperse, with some states clustered and others not. You'll notice Florida and New Hampshire have matching curves and are a bit divergent from the rest. This is probably a result of our greedy algorithm. If we introduced a cut off value for when we consider something a correlation match, this group probably would have been split into two, separated as I mentioned above. Along those same lines, group 6 is very divergent. Since there's only two states, either Nevada or Utah was the best predictor for the other but was a poor predictor. Introducing a cut off value on correlation matching could possibly separate these two. Group 8 seems to be in a similar situation. It's well clustered except for the North Mariana Islands. That state was probably included because it was correlated with an existing state in the set, just very poorly correlated. Again, introducing a cut off value could fix this most likely. Group 17 had no correlating models, which is why it's a single state (there were no deaths as of yet in this region).

Bias

The primary bias in generating these sets is what I've already mentioned. By not introducing a cutoff value on correlation, we're most likely including states in predictor sets that shouldn't be there. Preferably we'd want to cluster states with a very close correlation. We could handle this several ways. We could choose a cut off value for the correlation and change those values to NA. Instead of comparing a new state (the next most highest correlation), if the model generating state is already in a set, instead of just adding it to that set, see how the new state compares to the remaining models in the state. If it's a poor match to the others, we could either start a new set with just that state or NA the correlation value and continue to see

if we find a better total set fit. Either of these methods is likely to remove the outlier states from groups and produce sets that correlate highly among themselves, the difference would be in the number of potential sets generated and if we wanted to go for higher accuracy and smaller sets of states or better matching than what we have but less sets. Those would both be options I would want to compare in future analysis of this data.