County Level Poverty

1 Data Preparation

1.1 SAIPE data

```
library(tidyverse)
library(fpp3)


vt <- read.csv('SAIPE_04-22-2025.csv')

vt <- vt |>
  filter(!Name %in% c("United States", "Vermont")) |>
  mutate(Poverty.Universe = as.numeric(gsub(",", "", Poverty.Universe)),
         Number.in.Poverty = as.numeric(gsub(",", "", Number.in.Poverty)),
         county_fips = ID %% 1000,
         state_fips = ID %/% 1000) |>
  rename(county_population = Poverty.Universe, Number_in_Poverty = Number.in.Poverty, county_name = Name
  select(Year, ID, state_fips, county_fips, county_name, county_population, Number_in_Poverty)


glimpse(vt)
```

```
## Rows: 434
## Columns: 7
## $ Year              <int> 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023, 2023~
## $ ID                <int> 50001, 50003, 50005, 50007, 50009, 50011, 50013, 500~
## $ state_fips        <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, ~
## $ county_fips       <dbl> 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 1~
## $ county_name       <chr> "Addison County", "Bennington County", "Caledonia Co~
## $ county_population <dbl> 34764, 35852, 29895, 160080, 5995, 50263, 7450, 2551~
## $ Number_in_Poverty <dbl> 3010, 4272, 4192, 12459, 864, 4955, 650, 2461, 2851,~
```

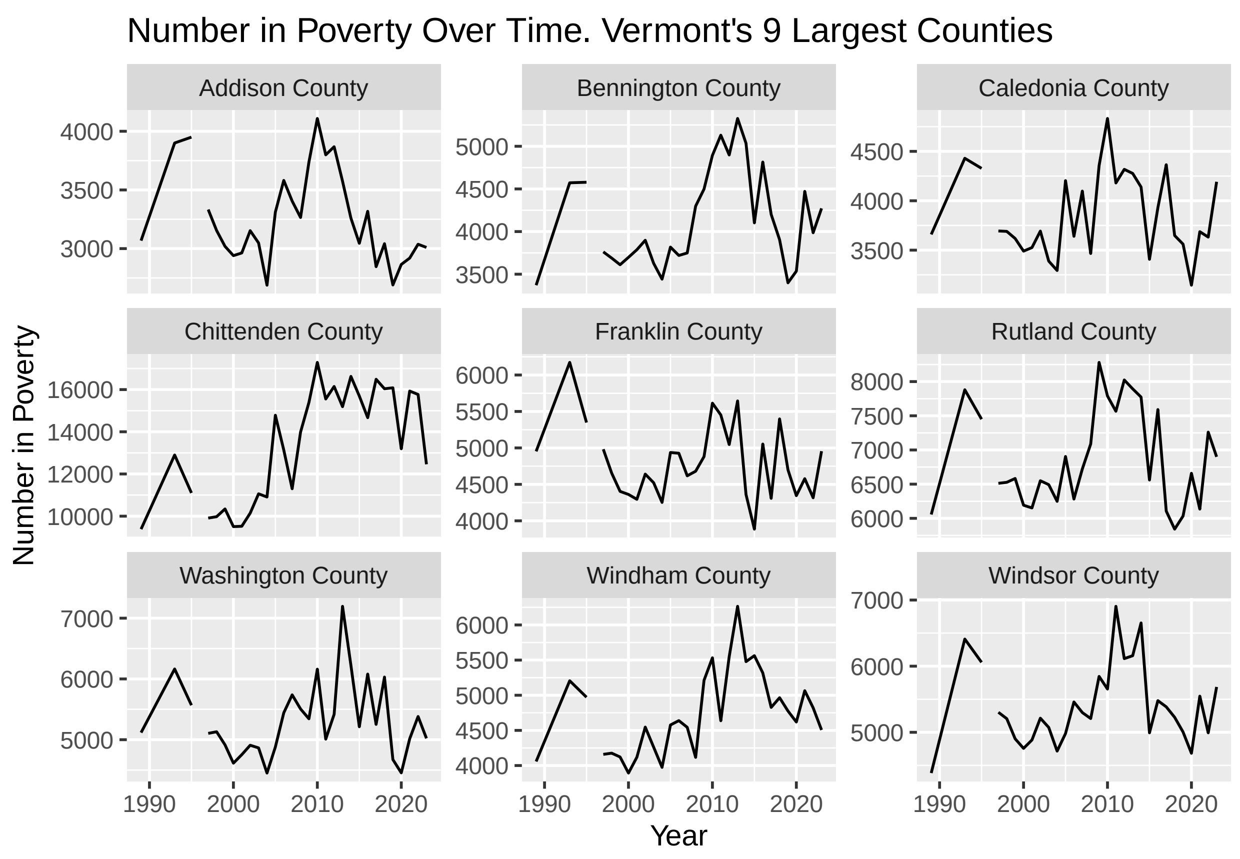
```
# largest county with highest population
vt |>
  filter(Year == 2023) |>
  slice_max(county_population, n = 1) |>
  pull(county_name)
```

```
## [1] "Chittenden County"
```

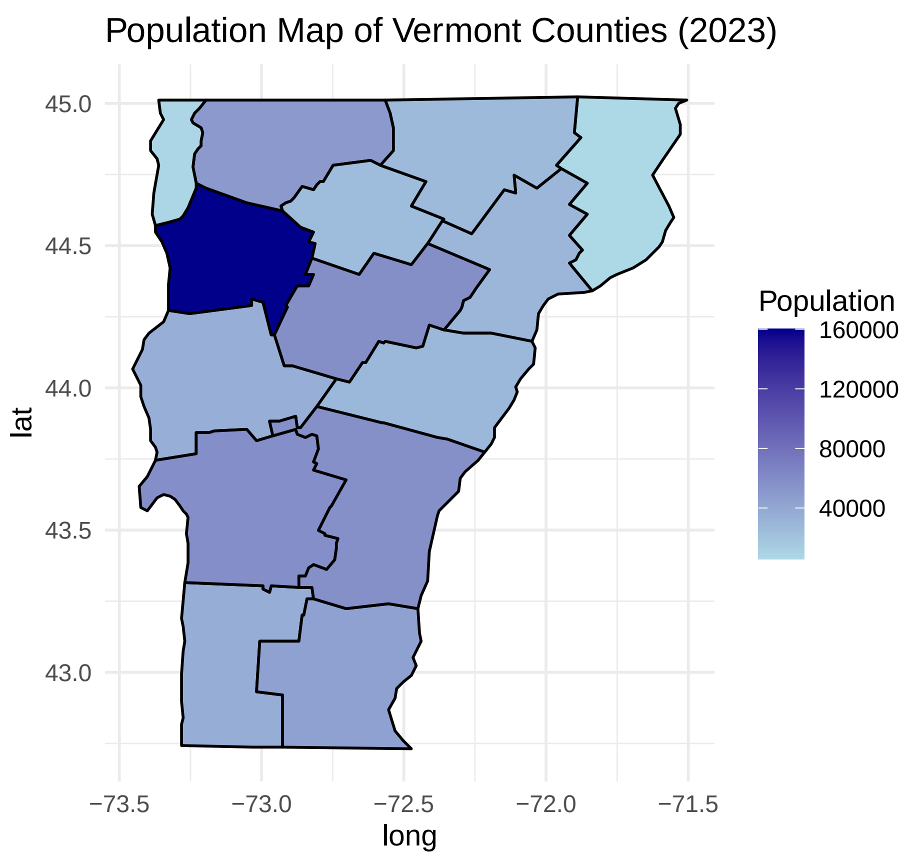
```r
# top 9 counties with highest population over the years
top9 <- vt |>
  filter(Year == 2023) |>
  slice_max(county_population, n = 9) |>
  select(county_name, county_population)

top9
```

```
##          county_name county_population
## 1 Chittenden County            160080
## 2    Rutland County             58556
## 3 Washington County             58089
## 4    Windsor County             57317
## 5   Franklin County             50263
## 6    Windham County             44444
## 7 Bennington County             35852
## 8    Addison County             34764
## 9  Caledonia County             29895
```

```r
# visualizing poverty trend in 9 largest counties
vt |>
  filter(county_name %in% top9$county_name) |>
  ggplot(aes(x = Year, y = Number_in_Poverty)) +
  geom_line() +
  facet_wrap(~ county_name, scales = "free_y") +
  labs(title = "Number in Poverty Over Time. Vermont's 9 Largest Counties", y = "Number in Poverty")
```



```r
## 1.2 Population Map

# Prepare data for mapping
vt_map_data <- vt |>
  filter(Year == 2023) |>
  mutate(county_name = tolower(gsub(" County", "", county_name)))

# Get Vermont county map data
county_map <- map_data("county", region = "vermont")

# Merge map data with population data
map_data_merged <- county_map |>
  mutate(subregion = tolower(subregion)) |>
  left_join(vt_map_data, by = c("subregion" = "county_name"))

# Plot the population map
ggplot(data = map_data_merged, aes(x = long, y = lat, group = group, fill = county_population)) +
  geom_polygon(color = "black") +
  scale_fill_gradient(low = "lightblue", high = "darkblue", name = "Population") +
  labs(title = "Population Map of Vermont Counties (2023)") +
  theme_minimal() +
  coord_fixed()
```

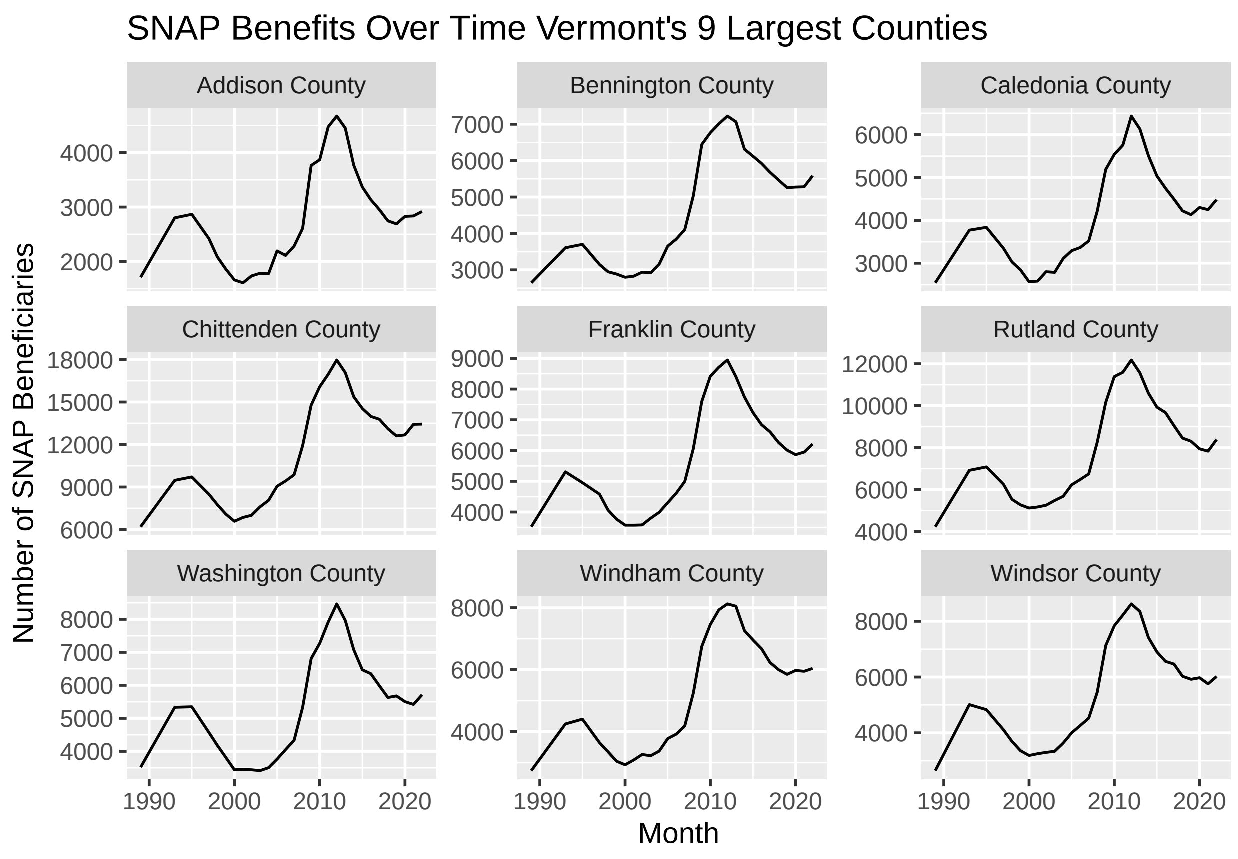


## 1.2 County SNAP Benefits

```r
snap <- read_csv('cntysnap.csv', skip = 3)

snap <- snap |>
  mutate(across(`Jul-2006`:`Jul-1997`, ~ as.numeric(gsub(",", "", .)))) |>
  rename(state_fips = `State FIPS code`, county_fips = `County FIPS code`) |>
  filter(state_fips == 50 & county_fips != 0)



snap <- snap |> pivot_longer(cols = starts_with("Jul-"), names_to = "Year",
                                                    values_to = "snap_benefits") |>
  mutate(Year = as.integer(sub("Jul-", "", Year))) |>
  separate(Name, into = c("county_name", "state_abbr"), sep = ", ")


glimpse(snap)
```

```
## Rows: 406
## Columns: 6
## $ state_fips    <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, ~
## $ county_fips   <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,~
## $ county_name   <chr> "Addison County", "Addison County", "Addison County", "A~
## $ state_abbr    <chr> "VT", "VT", "VT", "VT", "VT", "VT", "VT", "VT", "VT", "V~
## $ Year          <int> 2022, 2021, 2020, 2019, 2018, 2017, 2016, 2015, 2014, 20~
## $ snap_benefits <dbl> 2918, 2837, 2829, 2691, 2746, 2953, 3135, 3369, 3767, 44~
```

```
snap |>
filter(county_name %in% top9$county_name) |>
  ggplot(aes(x = Year, y = snap_benefits, group = county_name)) +
  geom_line() +
  facet_wrap(~ county_name, scales = "free_y") +
  labs(title = "SNAP Benefits Over Time Vermont's 9 Largest Counties",
       x = "Month", y = "Number of SNAP Beneficiaries")
```

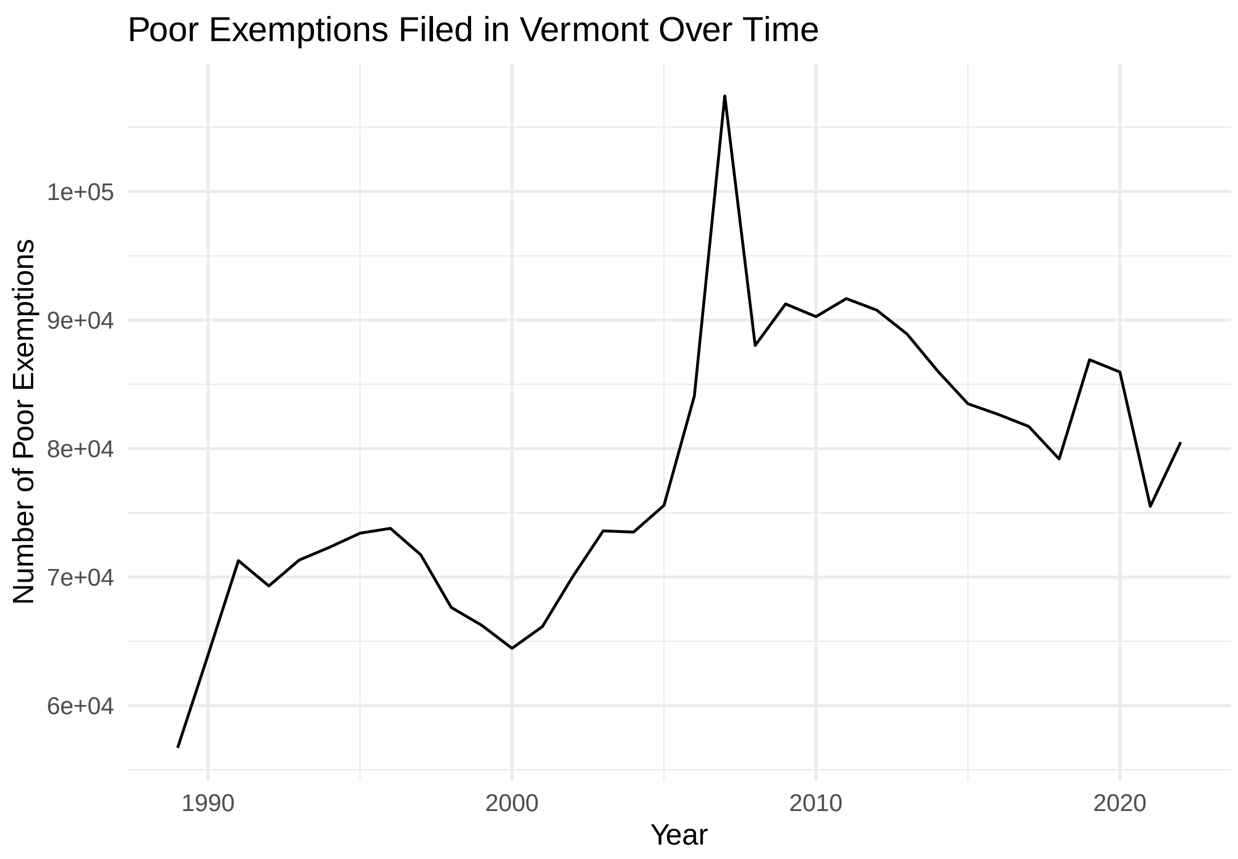


1.3 State IRS Data

```
irs <- read_csv('irs.csv', skip = 3)

irs <- irs |>
  filter(`State FIPS code` == 50) |>
  rename(state_fips = `State FIPS code`, poor_exemptions = `Poor exemptions`) |>
  select(-Name)

glimpse(irs)
```

```
## Rows: 34
## Columns: 12
## $ state_fips                      <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, ~
## $ Year                            <dbl> 1989, 1990, 1991, 1992, 1993, 1994, ~
## $ ‘Total exemptions‘                  <dbl> 491723, 491394, 493102, 492567, 4962~
## $ poor_exemptions                     <dbl> 56701, 63929, 71278, 69309, 71323, 7~
## $ ‘Age 65 and over exemptions‘        <dbl> 38098, 39422, 40033, 38955, 38836, 3~
## $ ‘Age 65 and over poor exemptions‘   <dbl> 5355, 6194, 7179, 6619, 7025, 6957, ~
## $ ‘Child exemptions‘                  <dbl> 147893, 147158, 148153, 149085, 1506~
## $ ‘Poor child exemptions‘             <dbl> 17344, 19672, 22082, 21999, 22821, 2~
## $ ‘Total exemptions under age 65‘     <dbl> 453625, 451972, 453069, 453612, 4574~
## $ ‘Poor exemptions under age 65‘      <dbl> 51346, 57735, 64099, 62690, 64298, 6~
## $ ‘Median AGI‘                        <dbl> 21492, 21792, 21796, 22614, 22856, 2~
## $ ‘Mean AGI‘                          <dbl> 29217, 29353, 29383, 30810, 31392, 3~
```

```r
irs |>
ggplot(aes(x = Year, y = poor_exemptions)) +
  geom_line() +
  labs(title = "Poor Exemptions Filed in Vermont Over Time",
    x = "Year", y = "Number of Poor Exemptions") +
  theme_minimal()
```

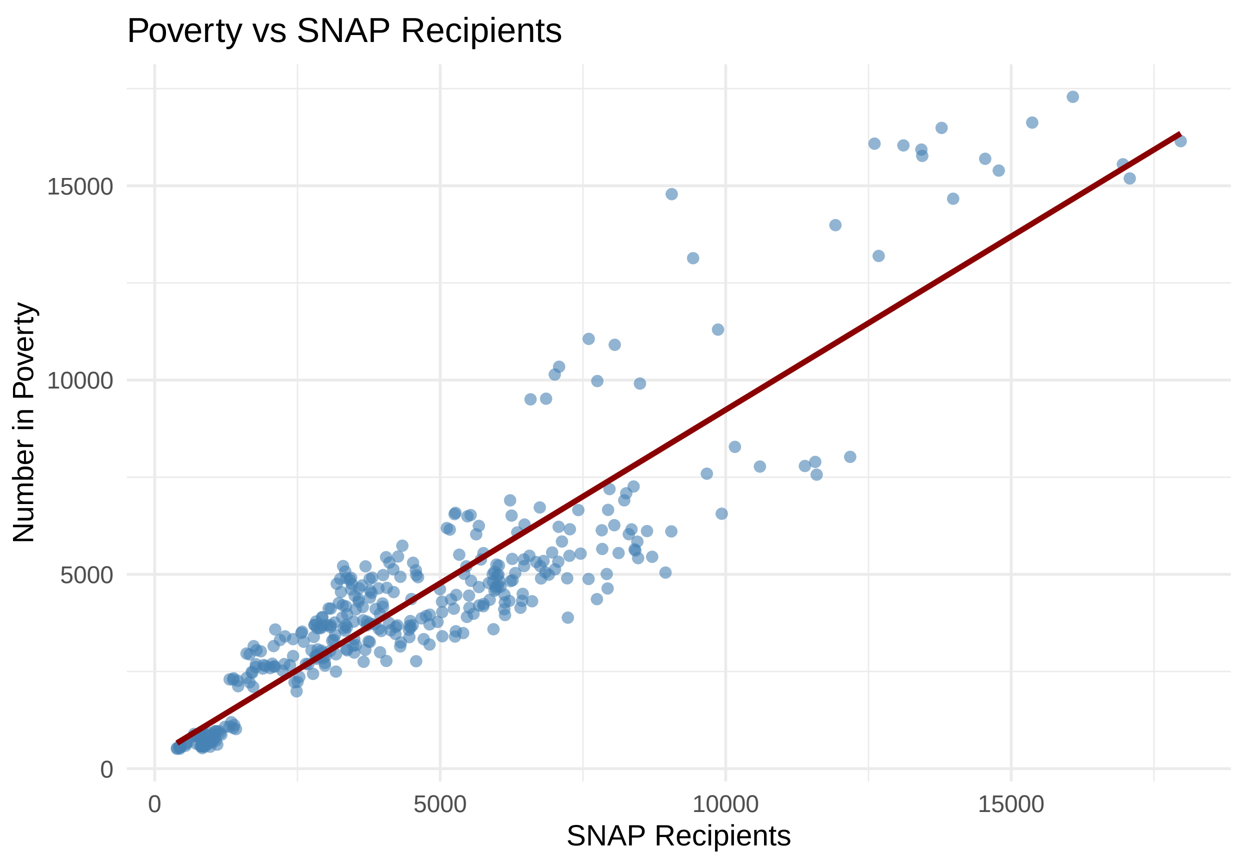


1.4 Merging the data

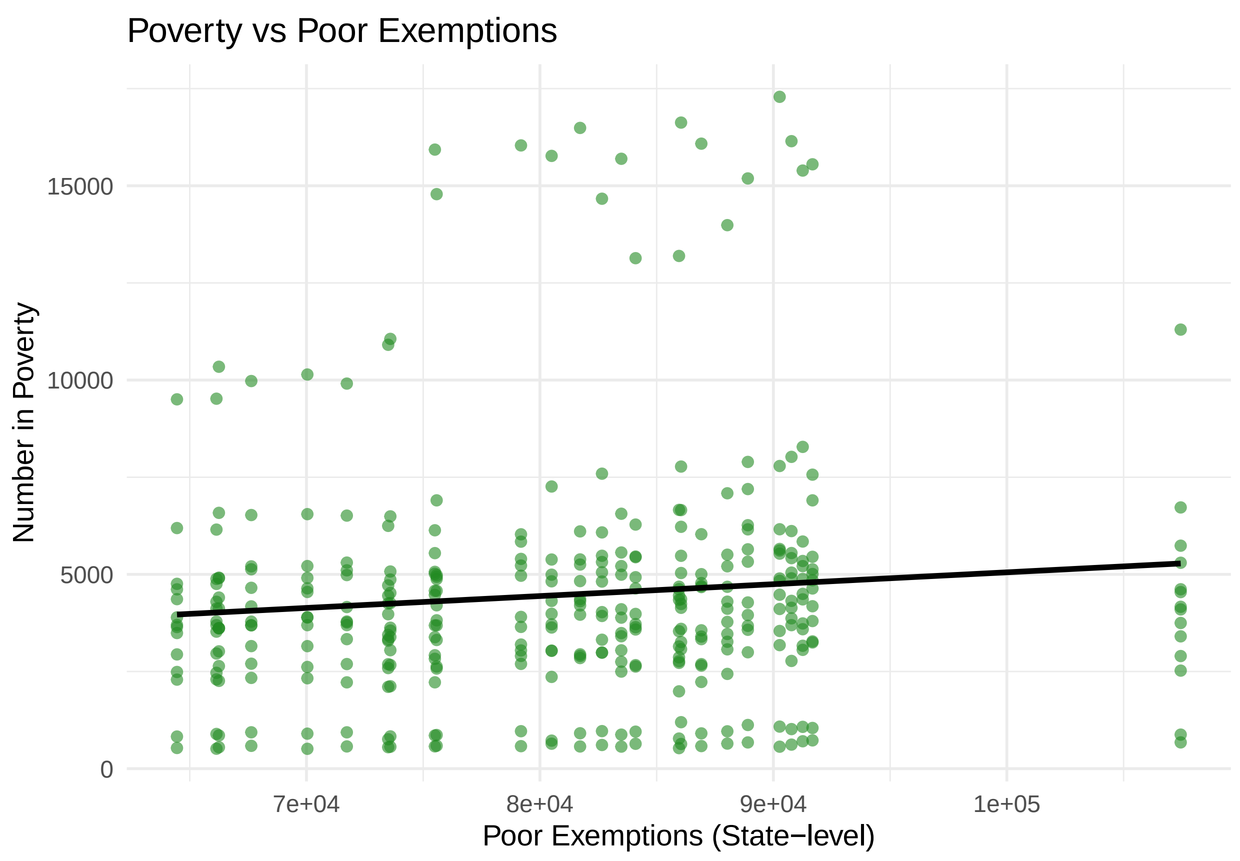
```r
merged_df <- vt |>
  left_join(snap, by = c("state_fips", "county_fips", "Year")) |>
  select(-county_name.y) |>
  rename(county_name = county_name.x)


merged_df <- merged_df |>
  left_join(irs, by = c('state_fips', 'Year')) |>
  select(Year, state_fips, county_fips, county_name, Number_in_Poverty, county_population,
    snap_benefits, poor_exemptions) |>
  filter(Year >= 1997)


clean_df <- merged_df |>
  filter(!is.na(snap_benefits), !is.na(poor_exemptions))


final_tsibble <- clean_df |>
  as_tsibble(key = c(county_fips, county_name), index = Year)

glimpse(final_tsibble)
```

```
## Rows: 364
## Columns: 8
## Key: county_fips, county_name [14]
## $ Year              <dbl> 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005~
## $ state_fips        <dbl> 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50, ~
## $ county_fips       <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ county_name       <chr> "Addison County", "Addison County", "Addison County"~
## $ Number_in_Poverty <dbl> 3333, 3154, 3019, 2940, 2963, 3153, 3048, 2688, 3311~
## $ county_population <dbl> NA, 34081, 34054, 34410, 34800, 35089, 35129, 35292,~
## $ snap_benefits     <dbl> 2423, 2083, 1858, 1661, 1608, 1734, 1783, 1773, 2194~
## $ poor_exemptions   <dbl> 71729, 67636, 66249, 64453, 66144, 70038, 73593, 735~
```

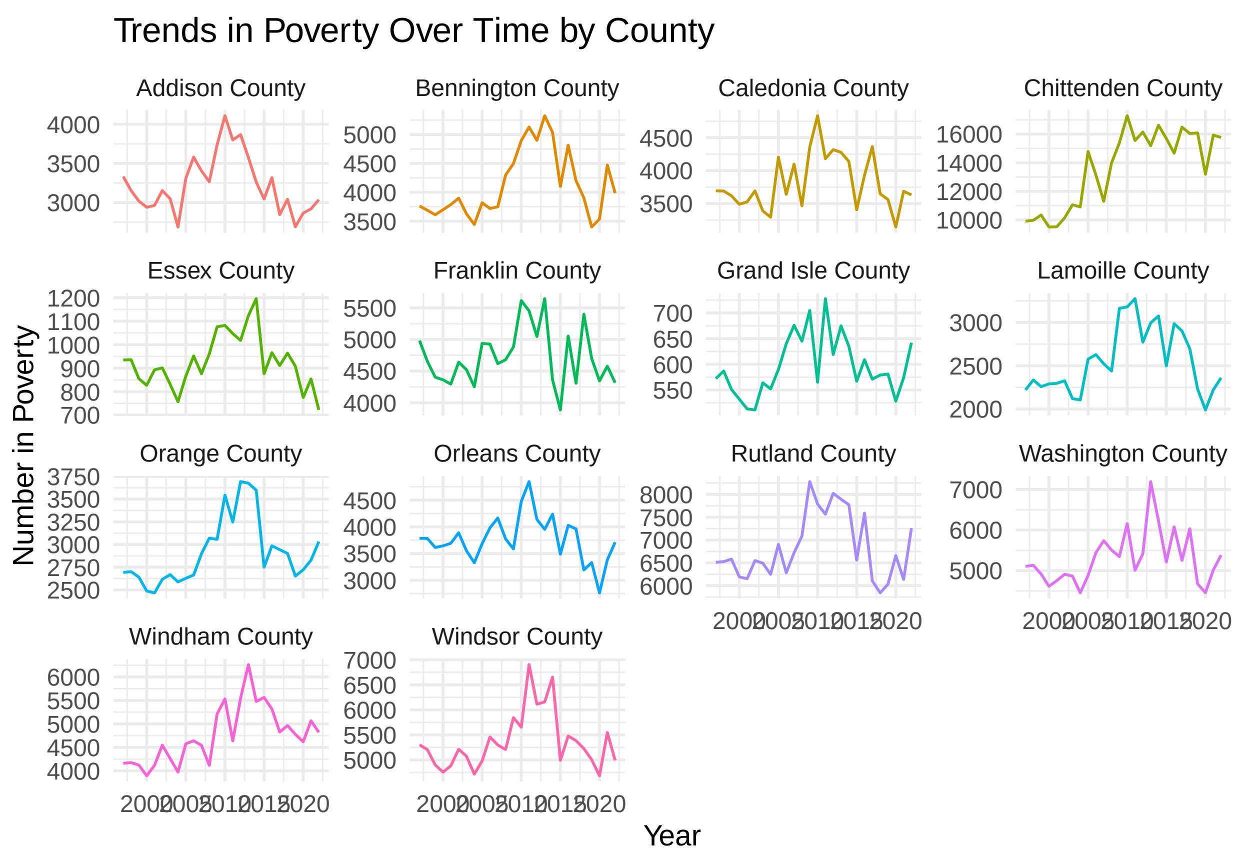
```
clean_df |>
ggplot(aes(x = snap_benefits, y = Number_in_Poverty)) +
  geom_point(alpha = 0.6, color = "steelblue") +
  geom_smooth(method = "lm", se = FALSE, color = "darkred") +
  labs(title = "Poverty vs SNAP Recipients", x = "SNAP Recipients", y = "Number in Poverty") +
  theme_minimal()
```



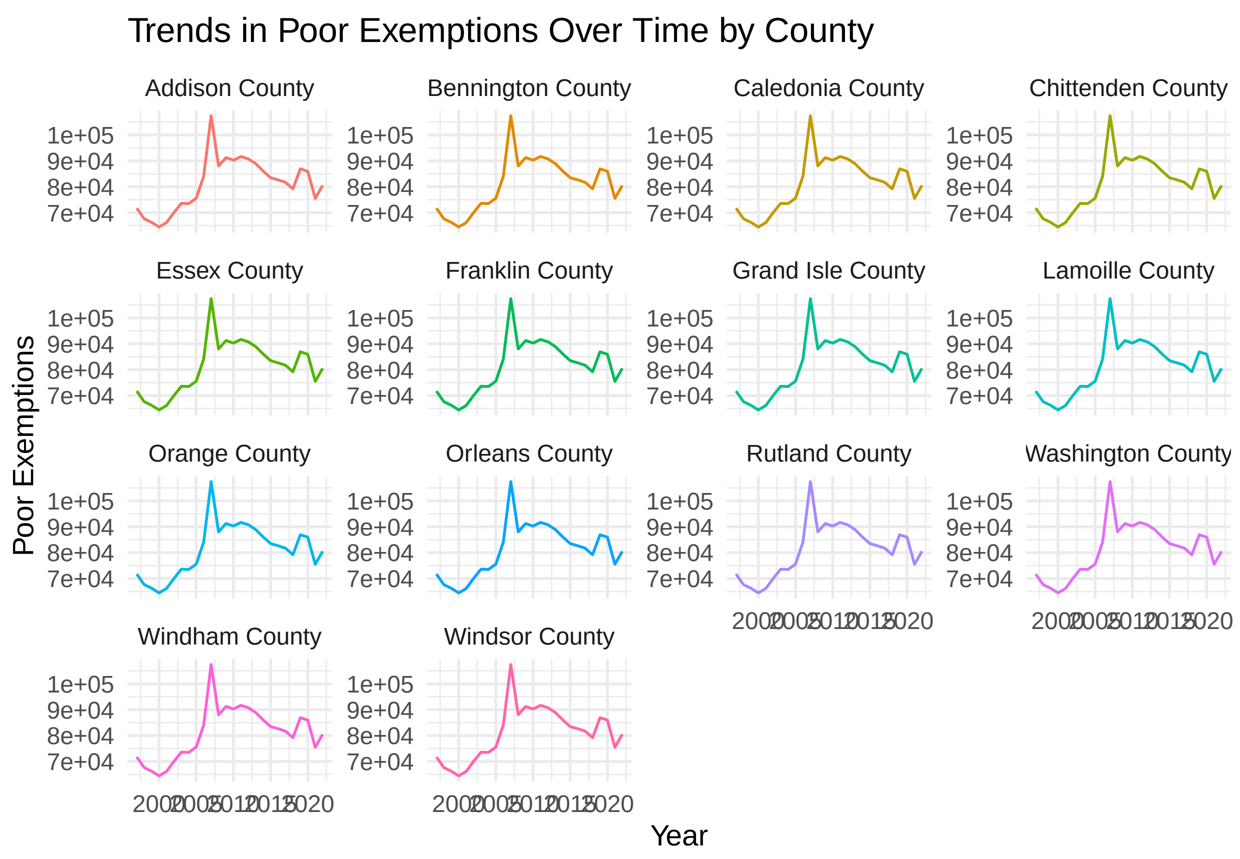
```
clean_df |>
ggplot(aes(x = poor_exemptions, y = Number_in_Poverty)) +
  geom_point(alpha = 0.6, color = "forestgreen") +
  geom_smooth(method = "lm", se = FALSE, color = "black") +
  labs(title = "Poverty vs Poor Exemptions", x = "Poor Exemptions (State-level)",
    y = "Number in Poverty") +
  theme_minimal()
```



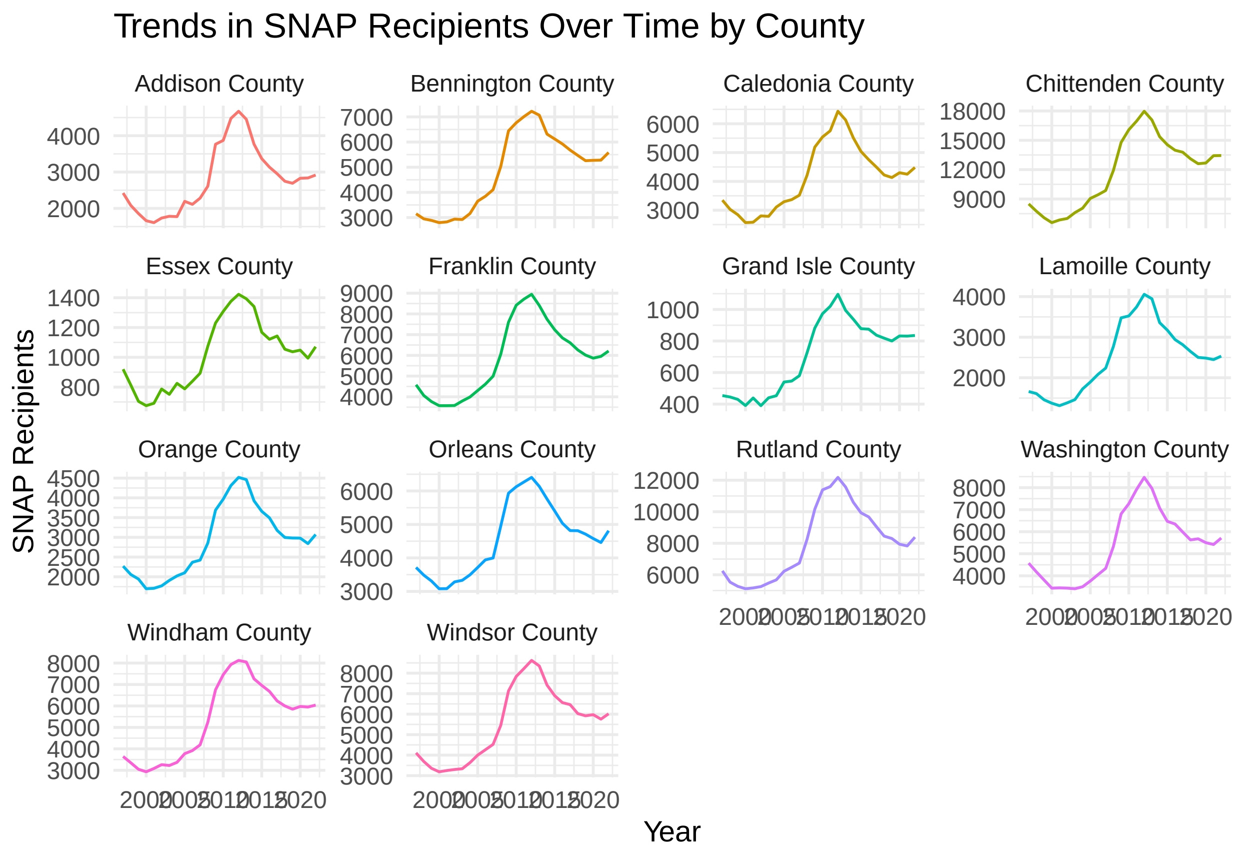
```r
clean_df |>
ggplot(aes(x = Year, y = Number_in_Poverty)) +
  geom_line(aes(color = county_name), show.legend = FALSE) +
  facet_wrap(~ county_name, scales = "free_y") +
  labs(title = "Trends in Poverty Over Time by County", x = "Year", y = "Number in Poverty") +
  theme_minimal()
```



```r
clean_df |>
  ggplot(aes(x = Year, y = poor_exemptions)) +
  geom_line(aes(color = county_name), show.legend = FALSE) +
  facet_wrap(~ county_name, scales = "free_y") +
  labs(title = "Trends in Poor Exemptions Over Time by County", x = "Year", y = "Poor Exemptions") +
  theme_minimal()
```



```r
clean_df |>
  ggplot(aes(x = Year, y = snap_benefits)) +
  geom_line(aes(color = county_name), show.legend = FALSE) +
  facet_wrap(~ county_name, scales = "free_y") +
  labs(title = "Trends in SNAP Recipients Over Time by County", x = "Year", y = "SNAP Recipients") +
  theme_minimal()
```



2 Linear models

2.1 Variable selection

```
# all 7 linear models
models <- final_tsibble |>
  model(
    model1 = TSLM(log(Number_in_Poverty) ~ log(county_population)),
    model2 = TSLM(log(Number_in_Poverty) ~ log(snap_benefits)),
    model3 = TSLM(log(Number_in_Poverty) ~ log(poor_exemptions)),
    model4 = TSLM(log(Number_in_Poverty) ~ log(county_population) + log(snap_benefits)),
    model5 = TSLM(log(Number_in_Poverty) ~ log(county_population) + log(log(poor_exemptions))),
    model6 = TSLM(log(Number_in_Poverty) ~ log(snap_benefits) + log(poor_exemptions)),
    model7 = TSLM(log(Number_in_Poverty) ~ log(county_population) + log(poor_exemptions) +
                                            log(poor_exemptions)))

glance(models) |>
  select(.model, adj_r_squared, CV, AIC, AICc, BIC) |>
  arrange(desc(adj_r_squared))
```

```
## # A tibble: 98 x 6
##    .model adj_r_squared      CV   AIC  AICc   BIC
##    <chr>          <dbl>   <dbl> <dbl> <dbl> <dbl>
##  1 model4         0.839 0.00729 -121. -119. -116.
##  2 model2         0.829 0.00782 -124. -123. -120.
##  3 model6         0.825 0.00878 -122. -120. -117.
##  4 model2         0.738 0.00405 -141. -140. -138.
##  5 model6         0.726 0.00462 -140. -138. -134.
##  6 model4         0.725 0.00599 -127. -125. -122.
##  7 model4         0.721 0.00433 -133. -131. -128.
##  8 model2         0.710 0.00484 -137. -136. -133.
##  9 model6         0.700 0.00531 -135. -133. -130.
## 10 model4         0.697 0.00513 -130. -128. -125.
## # i 88 more rows
```

```r
glance(models) |>
  select(.model, adj_r_squared, CV, AIC, AICc, BIC) |>
  arrange(CV)
```

```
## # A tibble: 98 x 6
##    .model adj_r_squared      CV   AIC  AICc   BIC
##    <chr>          <dbl>   <dbl> <dbl> <dbl> <dbl>
##  1 model4         0.693 0.00383 -138. -136. -134.
##  2 model2         0.738 0.00405 -141. -140. -138.
##  3 model4         0.721 0.00433 -133. -131. -128.
##  4 model6         0.726 0.00462 -140. -138. -134.
##  5 model2         0.710 0.00484 -137. -136. -133.
##  6 model3         0.486 0.00490 -136. -135. -132.
##  7 model4         0.697 0.00513 -130. -128. -125.
##  8 model6         0.464 0.00514 -134. -132. -129.
##  9 model6         0.700 0.00531 -135. -133. -130.
## 10 model7         0.461 0.00557 -128. -126. -123.
## # i 88 more rows
```

- Model 4 with predictors county population and snap benefits showed the highest adjusted R^2 and lowest cross-validation errors. So based on these findings, the predictors include in our best model are county population and snap benefits.

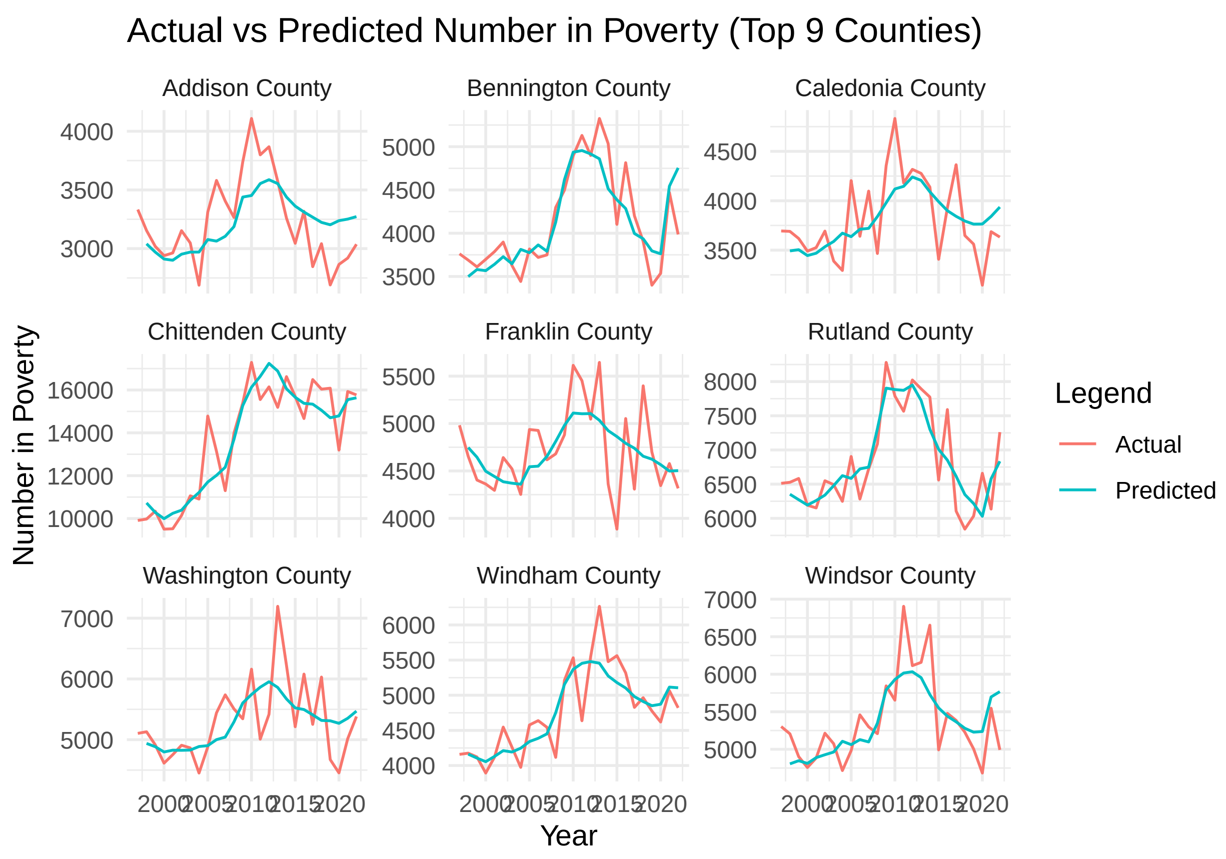
```r
glance(models) |>
  select(.model, adj_r_squared, CV, AIC, AICc, BIC) |>
  arrange(desc(adj_r_squared))
```

```
## # A tibble: 98 x 6
##    .model adj_r_squared      CV   AIC  AICc   BIC
##    <chr>          <dbl>   <dbl> <dbl> <dbl> <dbl>
##  1 model4         0.839 0.00729 -121. -119. -116.
##  2 model2         0.829 0.00782 -124. -123. -120.
##  3 model6         0.825 0.00878 -122. -120. -117.
##  4 model2         0.738 0.00405 -141. -140. -138.
##  5 model6         0.726 0.00462 -140. -138. -134.
##  6 model4         0.725 0.00599 -127. -125. -122.
##  7 model4         0.721 0.00433 -133. -131. -128.
##  8 model2         0.710 0.00484 -137. -136. -133.
##  9 model6         0.700 0.00531 -135. -133. -130.
## 10 model4         0.697 0.00513 -130. -128. -125.
## # i 88 more rows
```

```r
glance(models) |>
  select(.model, adj_r_squared, CV, AIC, AICc, BIC) |>
  arrange(CV)
```

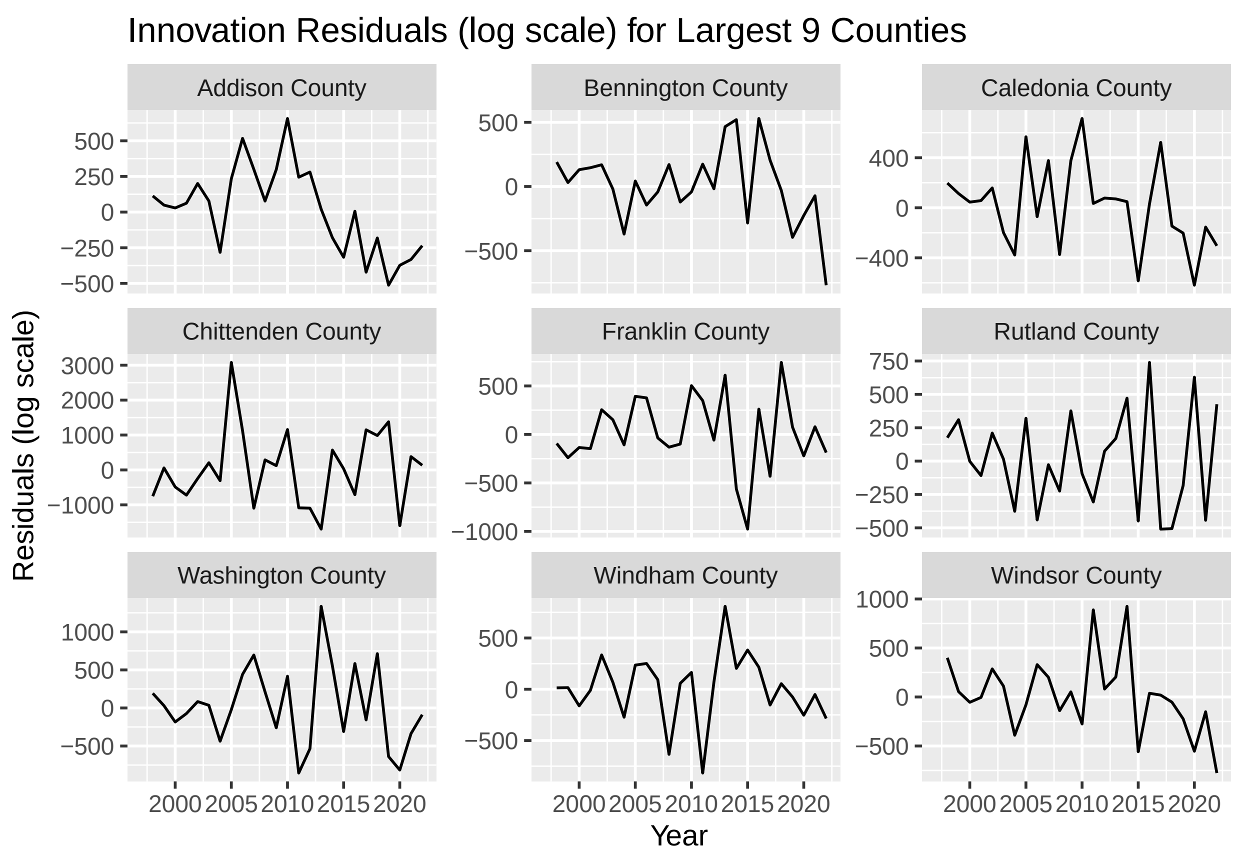
```
## # A tibble: 98 x 6
##    .model adj_r_squared      CV   AIC  AICc   BIC
##    <chr>          <dbl>   <dbl> <dbl> <dbl> <dbl>
##  1 model4         0.693 0.00383 -138. -136. -134.
##  2 model2         0.738 0.00405 -141. -140. -138.
##  3 model4         0.721 0.00433 -133. -131. -128.
##  4 model6         0.726 0.00462 -140. -138. -134.
##  5 model2         0.710 0.00484 -137. -136. -133.
##  6 model3         0.486 0.00490 -136. -135. -132.
##  7 model4         0.697 0.00513 -130. -128. -125.
##  8 model6         0.464 0.00514 -134. -132. -129.
##  9 model6         0.700 0.00531 -135. -133. -130.
## 10 model7         0.461 0.00557 -128. -126. -123.
## # i 88 more rows
```

```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name %in% top9$county_name) |>
  ggplot(aes(x = Year)) +
  geom_line(aes(y = Number_in_Poverty, color = "Actual")) +
  geom_line(aes(y = .fitted, color = "Predicted")) +
  facet_wrap(~ county_name, scales = "free_y") +
  labs(title = "Actual vs Predicted Number in Poverty (Top 9 Counties)",
       y = "Number in Poverty", color = "Legend") +
  theme_minimal()
```



2.2 Residual analysis

```
models |>
  select(model4) |>
  augment() |>
  filter(county_name %in% top9$county_name) |>
  ggplot(aes(x = Year, y = .resid)) +
  geom_line() +
  facet_wrap(~county_name, scales = "free_y") +
  labs(title = "Innovation Residuals (log scale) for Largest 9 Counties",
       y = "Residuals (log scale)", x = "Year")
```



```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Addison County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name    .model lb_stat lb_pvalue
##         <dbl> <chr>          <chr>    <dbl>     <dbl>
## 1           1 Addison County model4    35.5  0.000104
```

```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Bennington County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name       .model lb_stat lb_pvalue
##         <dbl> <chr>             <chr>    <dbl>     <dbl>
## 1           3 Bennington County model4    11.1     0.352
```

```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Caledonia County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name      .model lb_stat lb_pvalue
##         <dbl> <chr>            <chr>    <dbl>     <dbl>
## 1           5 Caledonia County model4    8.17     0.613
```

```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Chittenden County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name       .model lb_stat lb_pvalue
##         <dbl> <chr>             <chr>    <dbl>     <dbl>
## 1           7 Chittenden County model4    8.60     0.570
```

```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Essex County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name  .model lb_stat lb_pvalue
##         <dbl> <chr>        <chr>    <dbl>     <dbl>
## 1           9 Essex County model4    7.74     0.654
```

```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Franklin County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name     .model lb_stat lb_pvalue
##         <dbl> <chr>           <chr>    <dbl>     <dbl>
## 1          11 Franklin County model4    11.8     0.302
```

```r
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Grand Isle County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name       .model lb_stat lb_pvalue
##         <dbl> <chr>             <chr>    <dbl>     <dbl>
## 1          13 Grand Isle County model4    7.91     0.637
```

```
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Lamoille County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name     .model lb_stat lb_pvalue
##         <dbl> <chr>           <chr>    <dbl>     <dbl>
## 1          15 Lamoille County model4    8.15     0.615
```

```
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Orange County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name   .model lb_stat lb_pvalue
##         <dbl> <chr>         <chr>    <dbl>     <dbl>
## 1          17 Orange County model4    8.90     0.542
```

```
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Rutland County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name    .model lb_stat lb_pvalue
##         <dbl> <chr>          <chr>    <dbl>     <dbl>
## 1          21 Rutland County model4    12.8     0.234
```

```
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Washington County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name       .model lb_stat lb_pvalue
##         <dbl> <chr>             <chr>    <dbl>     <dbl>
## 1          23 Washington County model4    5.28     0.872
```

```
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Windham County') |>
  features(.innov, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 5
##   county_fips county_name    .model lb_stat lb_pvalue
##         <dbl> <chr>          <chr>    <dbl>     <dbl>
## 1          25 Windham County model4    12.7     0.241
```

```
models |>
  select(model4) |>
  augment() |>
  filter(county_name == 'Windsor County') |>
  features(.innov, ljung_box, lag = 10)
```

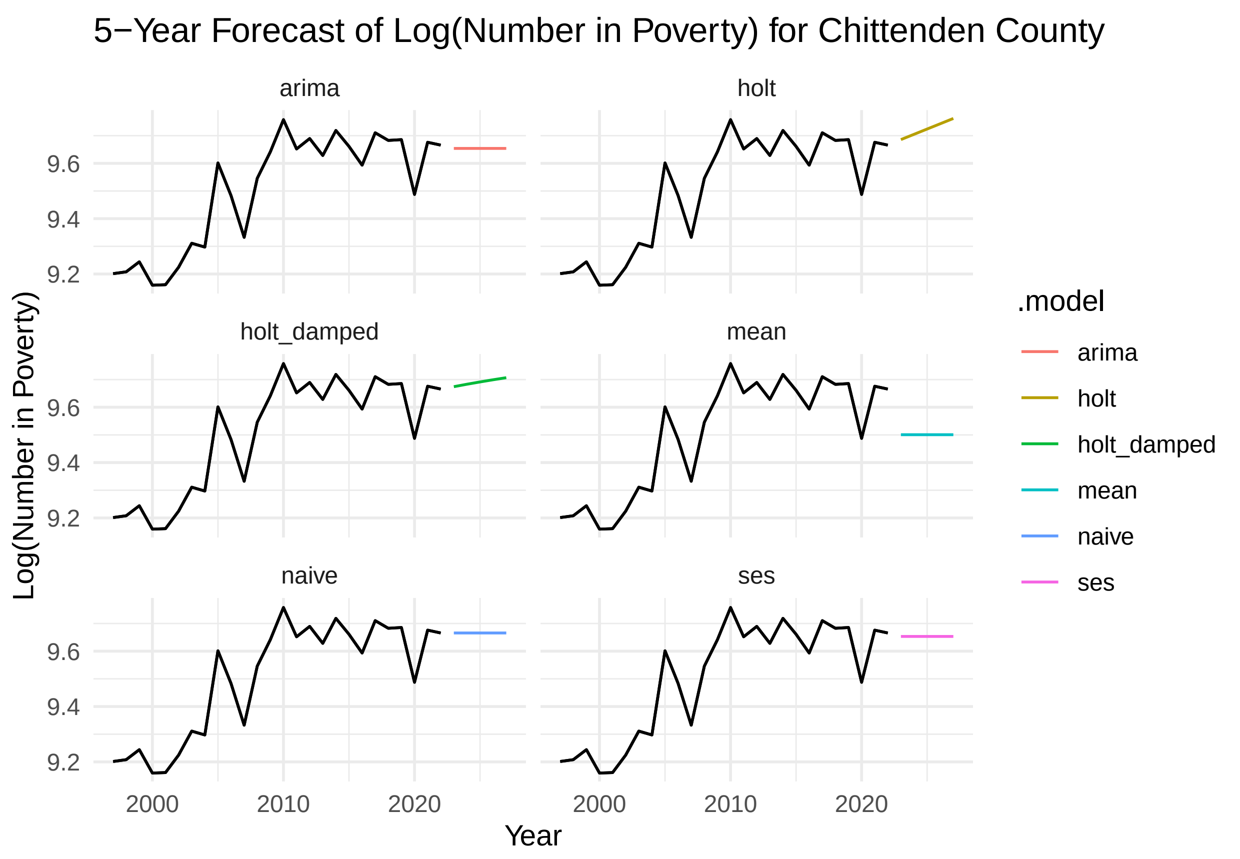
```
## # A tibble: 1 x 5
##   county_fips county_name    .model lb_stat lb_pvalue
##         <dbl> <chr>          <chr>    <dbl>     <dbl>
## 1          27 Windsor County model4    6.95     0.730
```

- Only the residuals of Addison County are significantly different than white noise.
- These findings suggests that our simple linear regression model is doing pretty good to predict number in poverty.

Part 3: Stochastic models

3.1 Single Country Forecasts

```
# Filter for Chittenden County and prepare the tsibble
chittenden_ts <- final_tsibble |>
  filter(county_name == "Chittenden County") |>
  mutate(log_poverty = log(Number_in_Poverty))

# Fit the models
chittenden_models <- chittenden_ts |>
  model(
    naive = NAIVE(log_poverty),
    mean = MEAN(log_poverty),
    ses = ETS(log_poverty ~ error("A") + trend("N") + season("N")), # Simple exponential smoothing
    holt = ETS(log_poverty ~ error("A") + trend("A") + season("N")), # Holt's method
    holt_damped = ETS(log_poverty ~ error("A") + trend("Ad") + season("N")), # Holt's method
    arima = ARIMA(log_poverty)                           # Auto ARIMA
  )

# Generate 5-year forecasts (2024-2028)
chittenden_forecasts <- chittenden_models |>
  forecast(h = 5)

# Plot the actual data and forecasts
chittenden_forecasts |>
  autoplot(chittenden_ts, level = NULL) +
  autolayer(chittenden_ts, log_poverty, color = "black") +
  facet_wrap(~ .model, ncol = 2) +
  labs(title = "5-Year Forecast of Log(Number in Poverty) for Chittenden County",
       y = "Log(Number in Poverty)", x = "Year") +
  theme_minimal()
```



```r
#  model accuracy measures
accuracy(chittenden_models) |>
  select(.model, RMSE, MAE, MAPE) |>
  arrange(RMSE)
```

```
## # A tibble: 6 x 4
##   .model       RMSE    MAE  MAPE
##   <chr>       <dbl>  <dbl> <dbl>
## 1 holt_damped 0.100 0.0711 0.749
## 2 holt        0.103 0.0721 0.757
## 3 ses         0.107 0.0776 0.814
## 4 arima       0.107 0.0780 0.818
## 5 naive       0.117 0.0902 0.945
## 6 mean        0.204 0.185  1.95
```

- Holt_damped model emerged as the best model for forecasting log(Number in Poverty) in Chittenden County with the lowest value of RMSE (0.1002210), MAE (0.07114368), and MAPE (0. The undamped Holt model closely trailed with an RMSE of 0.1025144, a MAE of 0.07211704 and a MAPE of 0.7569945, yet this aggressive linear trend projection led The SES model (RMSE=0.1073952, MAE=0.07760329, MAPE=0.8135818) was perplexed by its failure to capture trendiness resulting in a flattened forecast Similarly, the ARIMA model, with RMSE 0.1074227, MAE 0.07804024 and MAPE 0.8182591, had captured some dynamics whilst over fitting to short term fluctuations hence The Naive model, with an RMSE value of 0.1165119, MAE of 0.0902359, and MAPE of 0.9453219, was poor as it predicted a flat line at the last value Finally, the Mean model was the worst performing, with an RMSE of 0.2037611, MAE of 0.18451007, and MAPE of 1.9531604, as its flat average forecast did

3.2 Exponential smoothing models

```r
# Fit the three exponential smoothing models
exp_smooth_models <- final_tsibble |>
  mutate(log_poverty = log(Number_in_Poverty)) |> # Transform to log scale
  model(
    ses = ETS(log_poverty ~ error("A") + trend("N") + season("N")), # Simple exponential smoothing
    holt = ETS(log_poverty ~ error("A") + trend("A") + season("N")), # Holt's method
    holt_damped = ETS(log_poverty ~ error("A") + trend("Ad") + season("N")) # Holt's method
  )

# Evaluate model quality across all counties
exp_smooth_accuracy <- accuracy(exp_smooth_models) |>
  select(.model, county_name, RMSE, MAE, MAPE) |>
  group_by(.model) |>
  summarise(
    avg_RMSE = mean(RMSE, na.rm = TRUE),
    avg_MAE = mean(MAE, na.rm = TRUE),
    avg_MAPE = mean(MAPE, na.rm = TRUE)
  ) |>
  arrange(avg_RMSE)

# accuracy metrics
exp_smooth_accuracy
```

```
## # A tibble: 3 x 4
##   .model      avg_RMSE avg_MAE avg_MAPE
##   <chr>          <dbl>   <dbl>    <dbl>
## 1 holt_damped   0.0927  0.0704    0.870
## 2 holt          0.0936  0.0710    0.878
## 3 ses           0.0936  0.0711    0.877
```

The Holt_damped model does the best among all counties with the lowest average RMSE (0.09271526), MAE (0.07038092), and MAPE (0.8698863), finding a good The Holt model does not lag far behind (RMSE 0.09355634, MAE 0.07104669, MAPE 0.8775688). Although its undamped trend is simply not aligned with forecasting, The SES model, with RMSE 0.09359747 and, MAE 0.07105345 and MAPE 0.8772510, has an approach similar to Holt but fails without a trend component. When comparing Holt_damped with the rest, a clear winner is found in Holt_damped due to its accuracy and realism of predictions.

3.3 ARIMA Models

```r
# Fit auto-ARIMA models
arima_models <- final_tsibble |>
  mutate(log_poverty = log(Number_in_Poverty)) |>
  model(
    arima = ARIMA(log_poverty) # Auto ARIMA
  )

# Extract the ARIMA model specifications for each county
arima_specs <- arima_models |>
  tidy() |>
  select(county_name, .model, term) |>
  filter(term != "sigma2")
```

```
# Summarize the frequency of each ARIMA model
arima_summary <- arima_specs |>
  group_by(term) |>
  summarise(count = n()) |>
  arrange(desc(count))

arima_summary
```

```
## # A tibble: 4 x 2
##   term     count
##   <chr>    <int>
## 1 constant    12
## 2 ar1         11
## 3 ma1          2
## 4 ar2          1
```

```
# Fit ARIMA(1,1,0) and ARIMA(0,1,1) to all counties
arima_comparison <- final_tsibble |>
  mutate(log_poverty = log(Number_in_Poverty)) |>
  model(
    arima_110 = ARIMA(log_poverty ~ pdq(1,1,0)),
    arima_011 = ARIMA(log_poverty ~ pdq(0,1,1))
  )

# Evaluate model quality
arima_comparison_accuracy <- accuracy(arima_comparison) |>
  select(.model, county_name, RMSE, MAE, MAPE) |>
  group_by(.model) |>
  summarise(
    avg_RMSE = mean(RMSE, na.rm = TRUE),
    avg_MAE = mean(MAE, na.rm = TRUE),
    avg_MAPE = mean(MAPE, na.rm = TRUE)
  ) |>
  arrange(avg_RMSE)

arima_comparison_accuracy
```

```
## # A tibble: 2 x 4
##   .model    avg_RMSE avg_MAE avg_MAPE
##   <chr>        <dbl>   <dbl>    <dbl>
## 1 arima_011   0.0939  0.0712    0.879
## 2 arima_110   0.0962  0.0741    0.912
```

ARIMA (0, 1, 1) was selected over ARIMA (1, 1, 0) for forecasting log_poverty across counties because it was better in major accuracy measures and has a simpler structure. The ARIMA(0,1,1) performs no better than the ARIMA(1,1,0) with an average RMSE of 0.09392328, MAE of 0.07118617, and MAPE of 0.8787733. These metrics show that model ARIMA(0,1,1) performs best in obtaining forecasts with smaller errors and improved relative accuracy. Further, the moving average model of ARIMA (0,1,1) is less complex than the autoregressive model of ARIMA(1,1,0), increasing accuracy without overfitting and capturing the stochastic patterns of the differenced log_poverty data. The ARIMA(0,1,1) emerges as the method of choice due to greater accuracy and model parsimony for reliable and interpretable forecasts.

3.4 Cross Validation

```r
library(fable)
library(fabletools)
library(dplyr)

# rolling origin cross-validation function
rolling_cv <- function(data, model_fn, h = 1, window = NULL) {
  data |>
    stretch_tsibble(.init = window) |>
    model(model_fn) |>
    forecast(h = h) |>
    accuracy(data)
}

# Chittenden County
chittenden_data <- final_tsibble |>
  filter(county_name == "Chittenden County") |>
  mutate(log_poverty = log(Number_in_Poverty))

# Define models to compare
models_to_compare <- list(
  naive = NAIVE(log_poverty),
  mean = MEAN(log_poverty),
  ses = ETS(log_poverty ~ error("A") + trend("N") + season("N")),
  holt = ETS(log_poverty ~ error("A") + trend("A") + season("N")),
  holt_damped = ETS(log_poverty ~ error("A") + trend("Ad") + season("N")),
  arima = ARIMA(log_poverty)
)

# cross-validation
cv_results <- lapply(names(models_to_compare), function(model_name) {
  rolling_cv(chittenden_data, models_to_compare[[model_name]], h = 1, window = 5) |>
    mutate(Model = model_name)
}) |>
  bind_rows() |>
  select(Model, RMSE, MAE, MAPE) |>
  arrange(RMSE)

print(cv_results)
```

```
## # A tibble: 6 x 4
##   Model        RMSE    MAE  MAPE
##   <chr>       <dbl>  <dbl> <dbl>
## 1 ses         0.123 0.0997 1.04
## 2 holt_damped 0.125 0.0939 0.979
## 3 naive       0.126 0.101  1.06
## 4 holt        0.128 0.0999 1.04
## 5 arima       0.145 0.115  1.21
## 6 mean        0.235 0.208  2.16
```

The SES model was best on cross-validation as it has the least RMSE (0.1232406). RMSE is an important measure of forecast accuracy because it is more punitive for larger errors and is a good measure of overall performance. Although the Holt-damped model generates the least MAE (0.09385953) and MAPE (0.9789967), which indicates that average relative and absolute forecasting errors are small, the superior

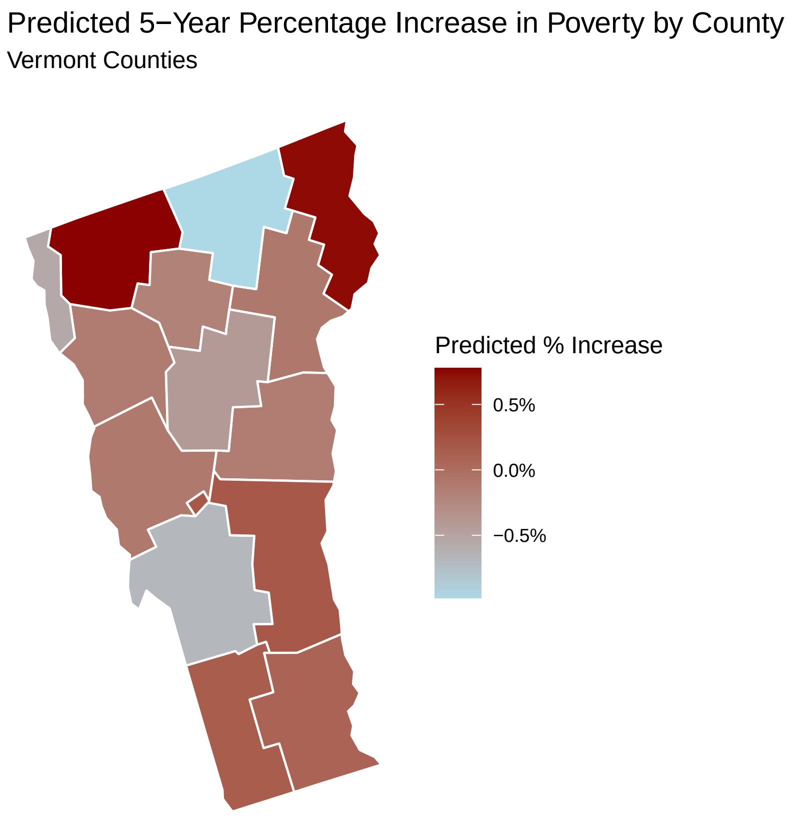
RMSE of the SES model shows that the SES can better avoid large forecasting errors. As important in time series forecasting for measuring overall error magnitude, the SES model emerges as the best model for this cross-validation analysis.

Part 4 forecasts

```r
library(usmap)

## Forecasting data
final_tsibble <- final_tsibble |>
  mutate(log_poverty = log(Number_in_Poverty))

## most recent actual values (2023)
latest_values <- final_tsibble |>
  as_tibble() |>
  filter(Year == max(Year)) |>
  select(county_name, county_fips,
         current_poverty = Number_in_Poverty,
         current_population = county_population)

## Fit the winning model (SES from cross-validation)
forecast_models <- final_tsibble |>
  model(
    ses = ETS(log_poverty ~ error("A") + trend("N") + season("N"))
  )

## Generate 5-year forecasts (2024-2028)
county_forecasts <- forecast_models |>
  forecast(h = 5)

## Calculate the predicted increase in poverty and percentage change
forecast_results <- county_forecasts |>
  as_tibble() |>
  group_by(county_name, county_fips) |>
  summarize(
    forecast_poverty = mean(exp(.mean)),
    .groups = "drop"
  ) |>
  left_join(latest_values, by = c("county_name", "county_fips")) |>
  mutate(
    poverty_increase = forecast_poverty - current_poverty,
    percent_increase = (poverty_increase / current_population) * 100
  )

## top 5 counties with highest predicted percentage increase
top5_counties <- forecast_results |>
  arrange(desc(percent_increase)) |>
  slice_head(n = 5) |>
  select(county_name, percent_increase);top5_counties
```

```
## # A tibble: 5 x 2
##   county_name       percent_increase
##   <chr>                        <dbl>
## 1 Franklin County              0.777
## 2 Essex County                 0.747
## 3 Windsor County               0.195
## 4 Bennington County            0.156
## 5 Windham County               0.0979
```

```
## Create a choropleth map of poverty increase predictions
plot_usmap(
  regions = "counties",
  include = "VT",
  data = forecast_results |>
    mutate(fips = 50000 + county_fips) |> # Vermont FIPS code is 50
    select(fips, percent_increase),
  values = "percent_increase",
  color = "white"
) +
  scale_fill_continuous(
    low = "lightblue",
    high = "darkred",
    name = "Predicted % Increase",
    label = scales::percent_format(scale = 1)
  ) +
  labs(title = "Predicted 5-Year Percentage Increase in Poverty by County",
       subtitle = "Vermont Counties") +
  theme(legend.position = "right")
```


Predicted 5−Year Percentage Increase in Poverty by County
Vermont Counties
Predicted % Increase
0.5%
0.0%
−0.5%