Final Project

Aksheytha Chelikavada

2023-05-14

1 DATA PREPARATION

Part 1.1

```
library(dplyr)
saipe_raw <- read.csv("C:\\Users\\akshe\\Downloads\\SAIPE_04-14-2023.csv")

saipe_mn <- saipe_raw %>% filter(!(Name == "Minnesota")) %>% filter(!(Name == "United States"))%
>% select(Year, FIPS = ID, Name, Pop = Poverty.Universe, Poverty = Number.in.Poverty)
```

Find the largest county, and the nine largest counties by population

```
largest_county_pop <- saipe_mn %>% group_by(FIPS, Name) %>% summarize(Pop = mean(Pop, na.rm = TR
UE)) %>%  arrange(desc(Pop)) %>% head(n = 9)
```

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

```
largest_county_pop
```

FIPS <int>	Name <chr>	Pop <dbl>
27053	Hennepin County	1152894.8
27123	Ramsey County	503860.2
27037	Dakota County	395766.3
27003	Anoka County	329222.4
27163	Washington County	232358.9
27137	St. Louis County	190716.9
27145	Stearns County	141531.2
27109	Olmsted County	141012.9
27139	Scott County	126144.7

9 rows

```
FIPSvalue <- saipe_mn %>% group_by(FIPS, Name) %>% summarize(Pop = mean(Pop, na.rm = TRUE)) %>%
arrange(desc(Pop)) %>% head(n = 9) %>% pull(FIPS)
```

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

```
biggest_county <- saipe_mn %>% group_by(FIPS, Name) %>% summarize(Pop = mean(Pop, na.rm = TRUE))
%>%  arrange(desc(Pop)) %>% head(n = 1)
```

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

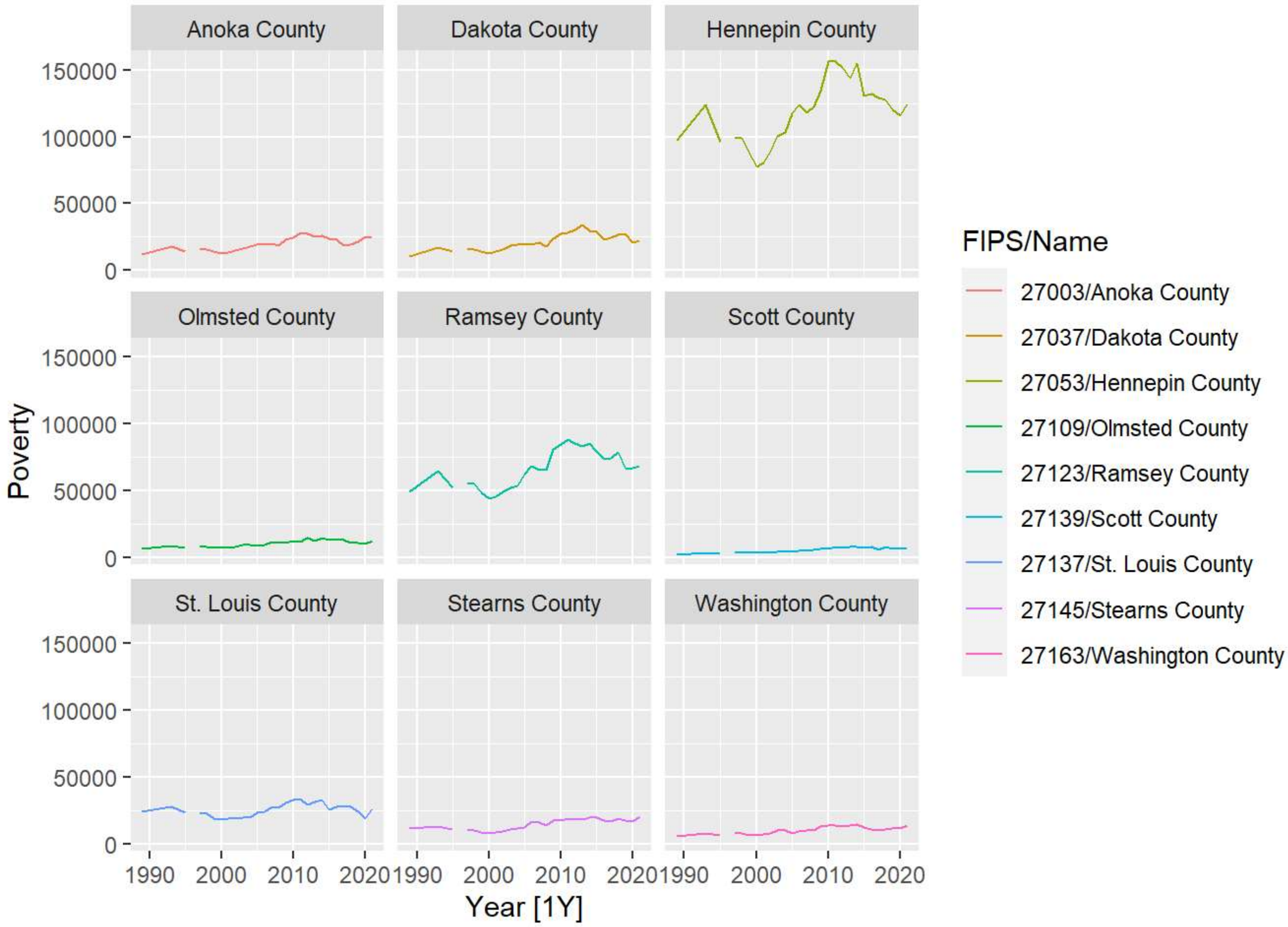
```
biggest_county
```

FIPS <int>	Name <chr>	Pop <dbl>
27053	Hennepin County	1152895

1 row

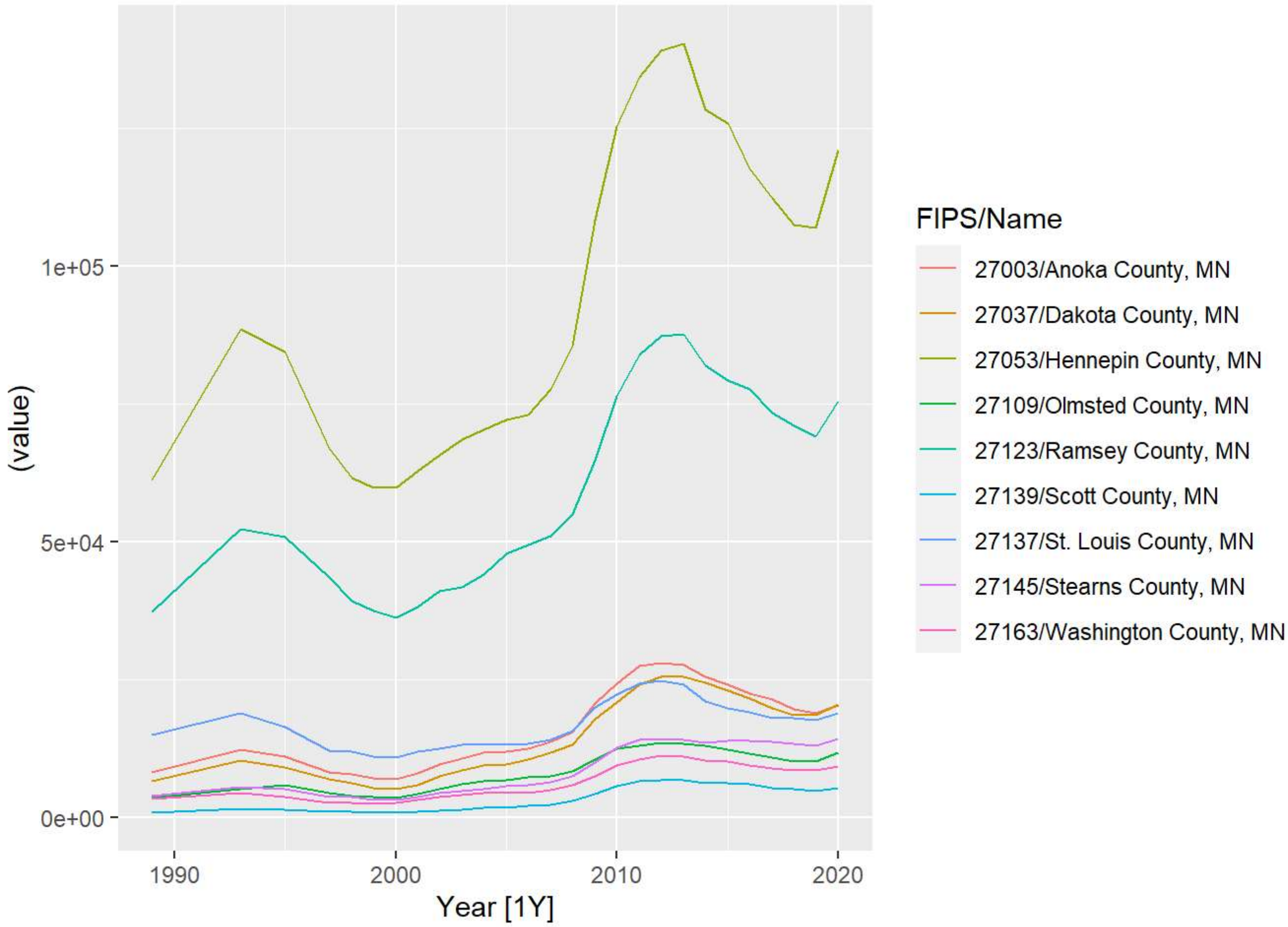
Make a time plot showing the number in poverty for each of the nine largest counties

```
library(dplyr)
library(ggplot2)
library(gtrendsR)
library(tsibble)
library(feasts)

saipe_mn_tsibble <- saipe_mn %>% as_tsibble(index = Year, key = c(FIPS, Name)) %>% filter(FIPS %
in% FIPSvalue)

saipe_mn_tsibble  %>% autoplot(Poverty) + facet_wrap(vars(Name))
```



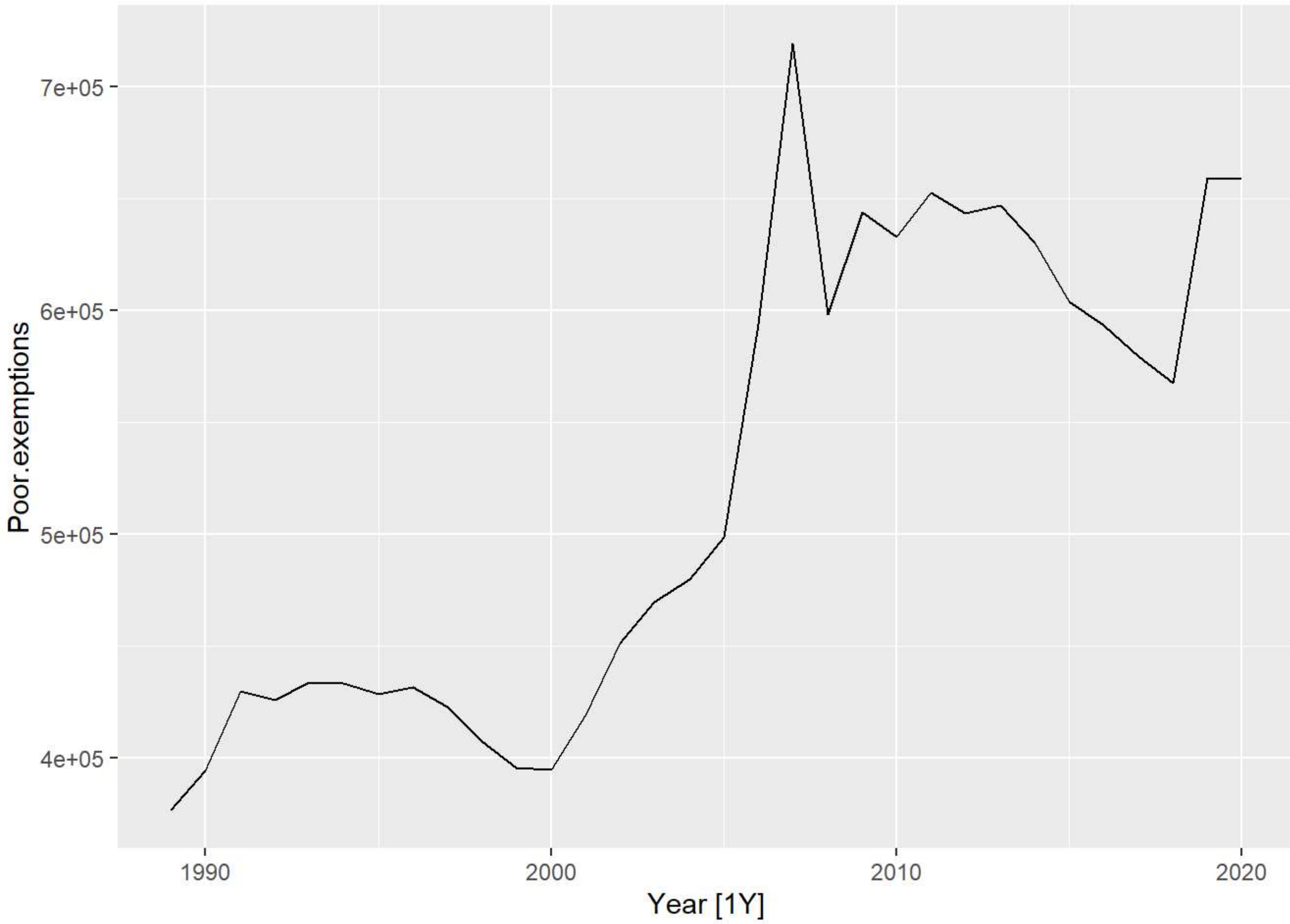
Part 1.2

```
library(stringr)
library(lubridate)
library(tidyverse)
library(readr)

cntySnap_raw <- read.csv("C:\\Users\\akshe\\Downloads\\cntysnap.csv",skip = 4, sep ="," )

mnCnty <- cntySnap_raw %>% filter(grepl("MN", Name))

code_mnCnty <- mnCnty %>% mutate(FIPS = paste("27",str_pad(County.FIPS.code, width = 3, pad =
"0"), sep = ""))

pivot_code_mnCnty <- code_mnCnty %>% pivot_longer(cols = starts_with("Jul")) %>% mutate(value =
as.integer(str_remove(value, ","))) %>% filter(FIPS %in% FIPSvalue) %>% mutate(Year = year(yearm
onth(name))) %>% as_tsibble(index = Year, key = c(FIPS, Name))

pivot_code_mnCnty  %>% ggplot2::autoplot((value))
```



Part 1.3

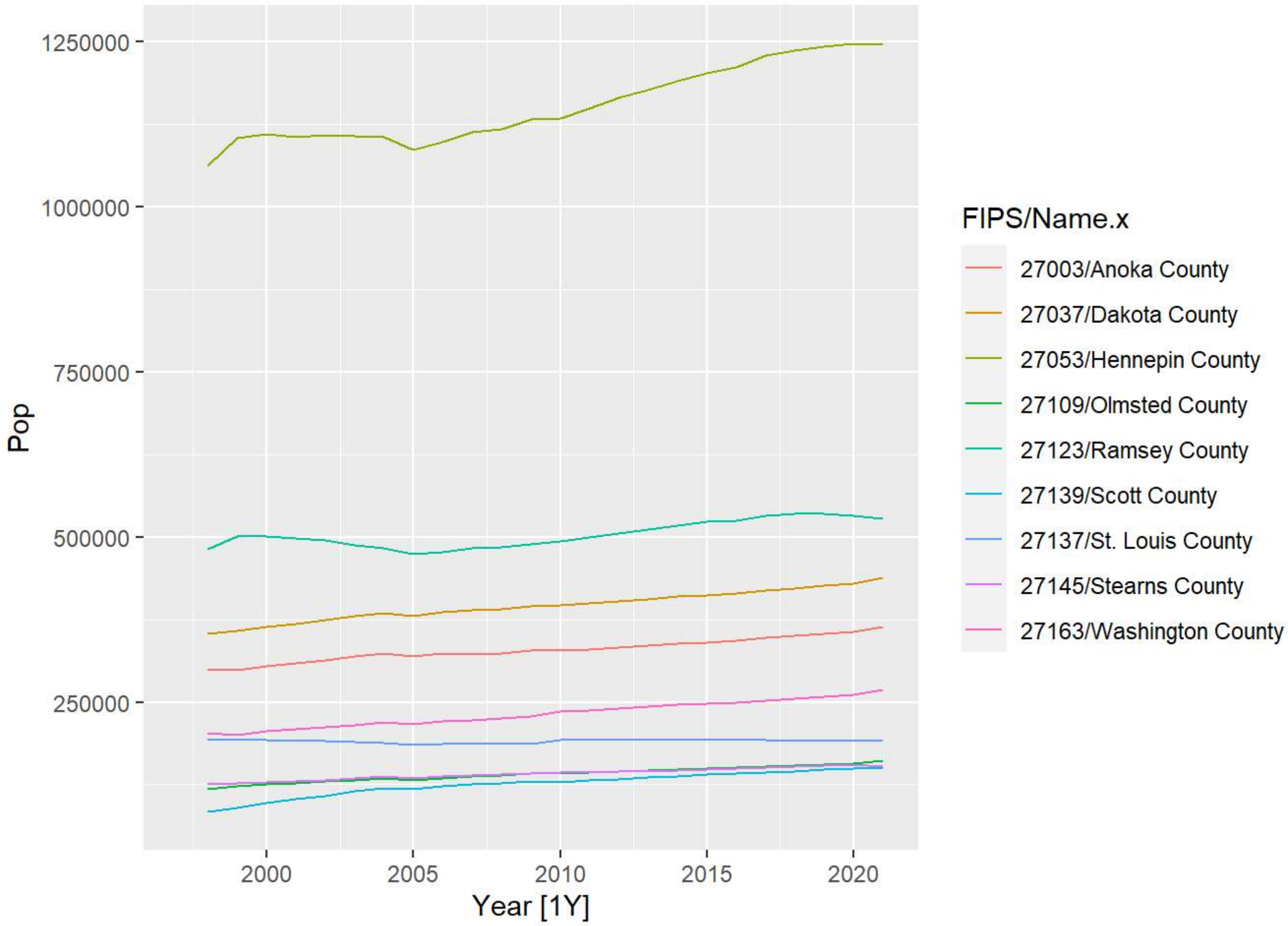
```
raw_irs <- read.csv("C:\\Users\\akshe\\Downloads\\irs.csv", skip = 4)

ts_irs <- raw_irs %>% filter(Name == "Minnesota") %>% mutate(Poor.exemptions = as.integer(str_re
move(Poor.exemptions, ","))) %>% as_tsibble(index = Year)

ts_irs %>% autoplot(Poor.exemptions)
```


7e+05
6e+05
5e+05
4e+05
Poor.exemptions
1990
2000
2010
2020
Year [1Y]

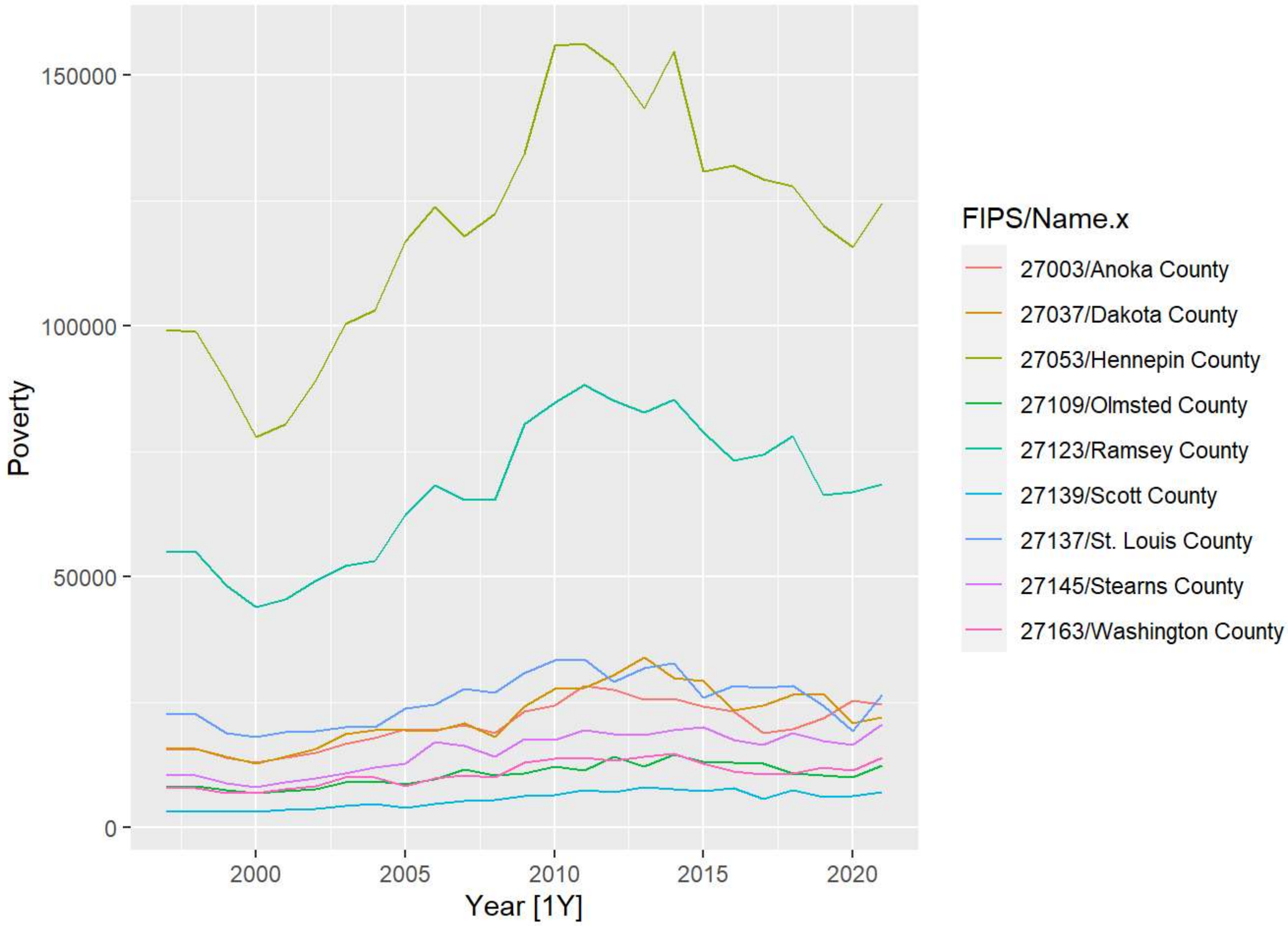
Part 1.4

```
library(lubridate)
library(tidyverse)
library(readr)

join_ts_irs <- raw_irs %>% filter(Name == "Minnesota") %>% dplyr::select(Year, Poor.exemptions)
%>% mutate(Poor.exemptions = as.integer(str_remove(Poor.exemptions, ","))) %>% as_tsibble(index
= Year)

pivot_code_mnCnty_all <- code_mnCnty %>% pivot_longer(cols = starts_with("Jul")) %>% mutate(valu
e = as.integer(str_remove(value, ","))) %>%  mutate(Year = year(yearmonth(name))) %>% as_tsibble
(index = Year, key = c(FIPS, Name))

join_mnCnty_all <- pivot_code_mnCnty_all %>% dplyr::select(FIPS, value, Year)

new_join_mnCnty_all <- join_mnCnty_all %>% mutate(FIPS = as.integer(FIPS))

saipe_mn_join1 <- left_join(saipe_mn, new_join_mnCnty_all, by=c('Year','FIPS'))

final_join_ts <- left_join(saipe_mn_join1, join_ts_irs, by = 'Year') %>% filter(Year >= 1997) %
>% as_tsibble(index = Year, key = c(FIPS, Name.x))

graph_final_ts <- final_join_ts %>% filter(FIPS %in% FIPSvalue)

graph_final_ts %>% autoplot(Pop)
```

```
## Warning: Removed 9 rows containing missing values (`geom_line()`).
```

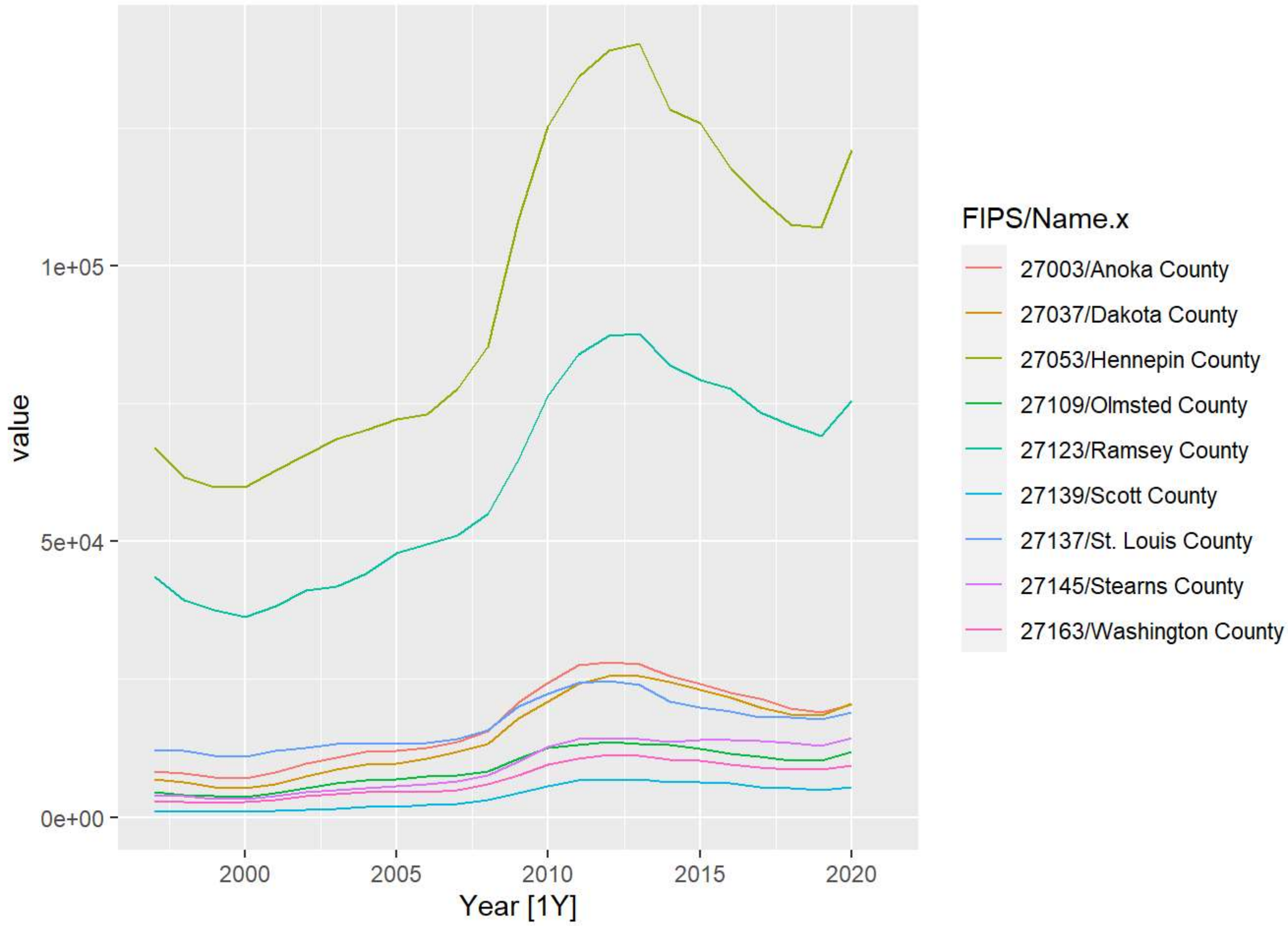



```
graph_final_ts %>% autoplot(Poverty)
```



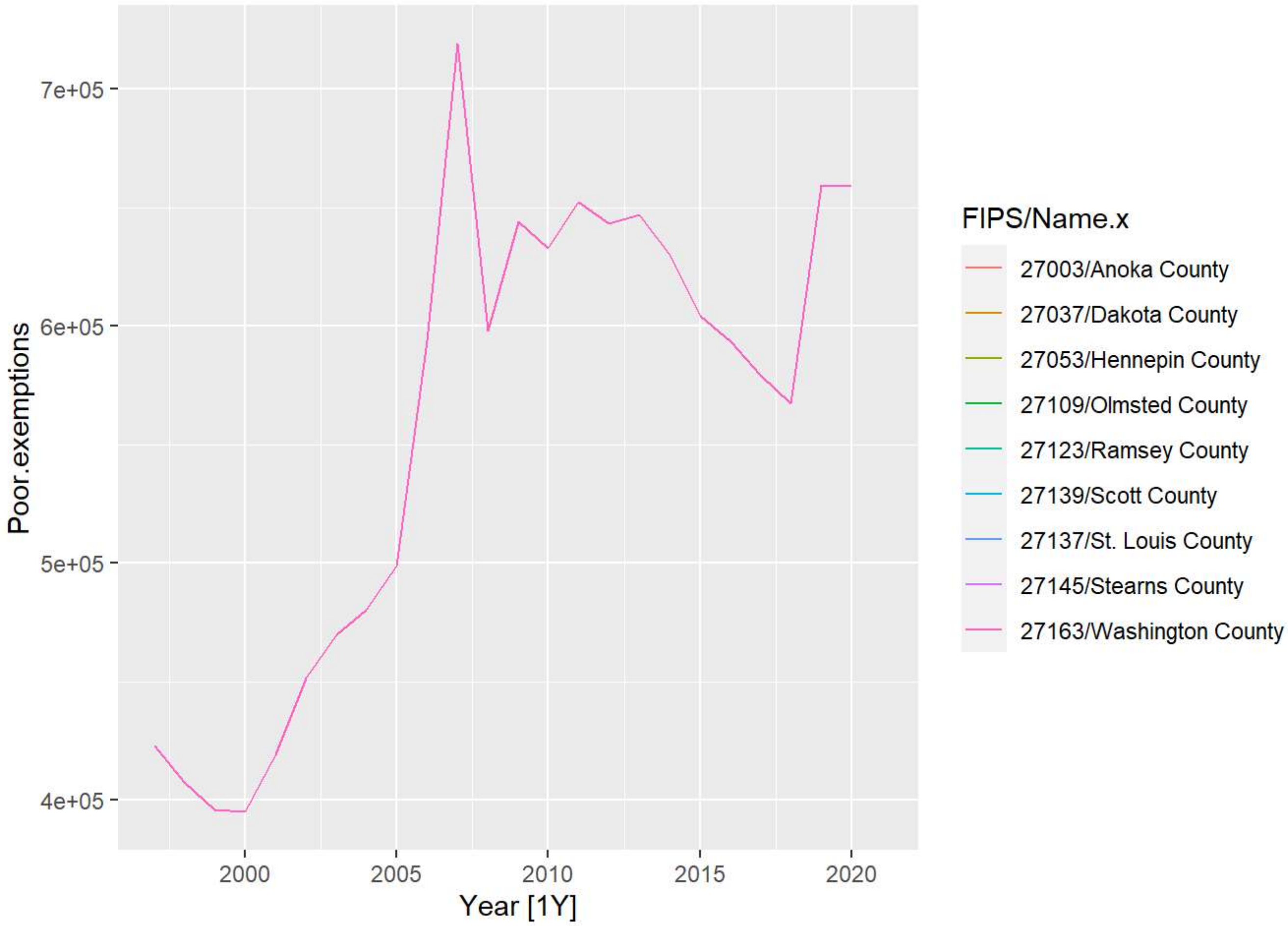

```
graph_final_ts %>% autoplot(value)
```

```
## Warning: Removed 9 rows containing missing values (`geom_line()`).
```



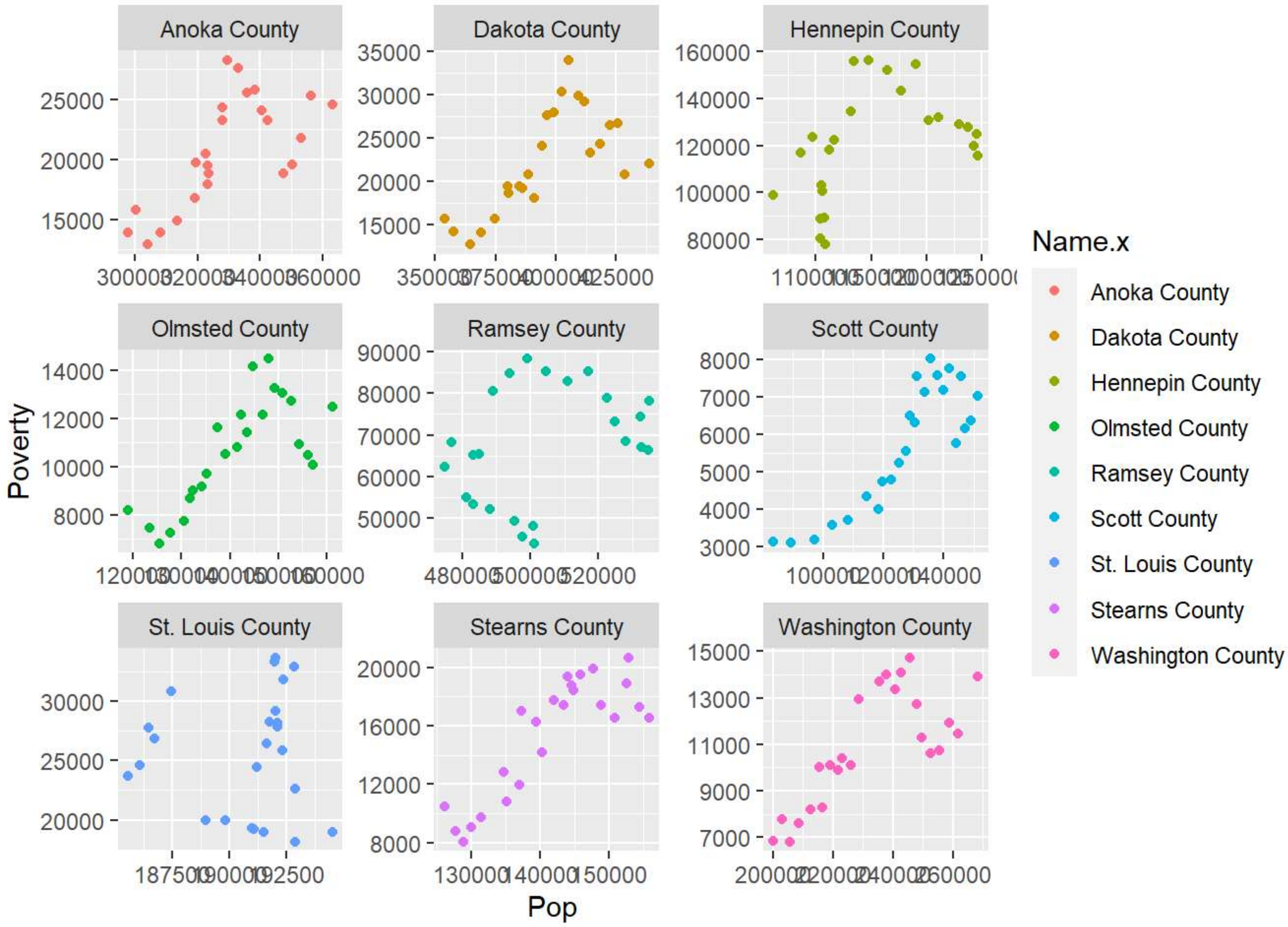

```
graph_final_ts %>% autoplot(Poor.exemptions)
```

```
## Warning: Removed 9 rows containing missing values (`geom_line()`).
```



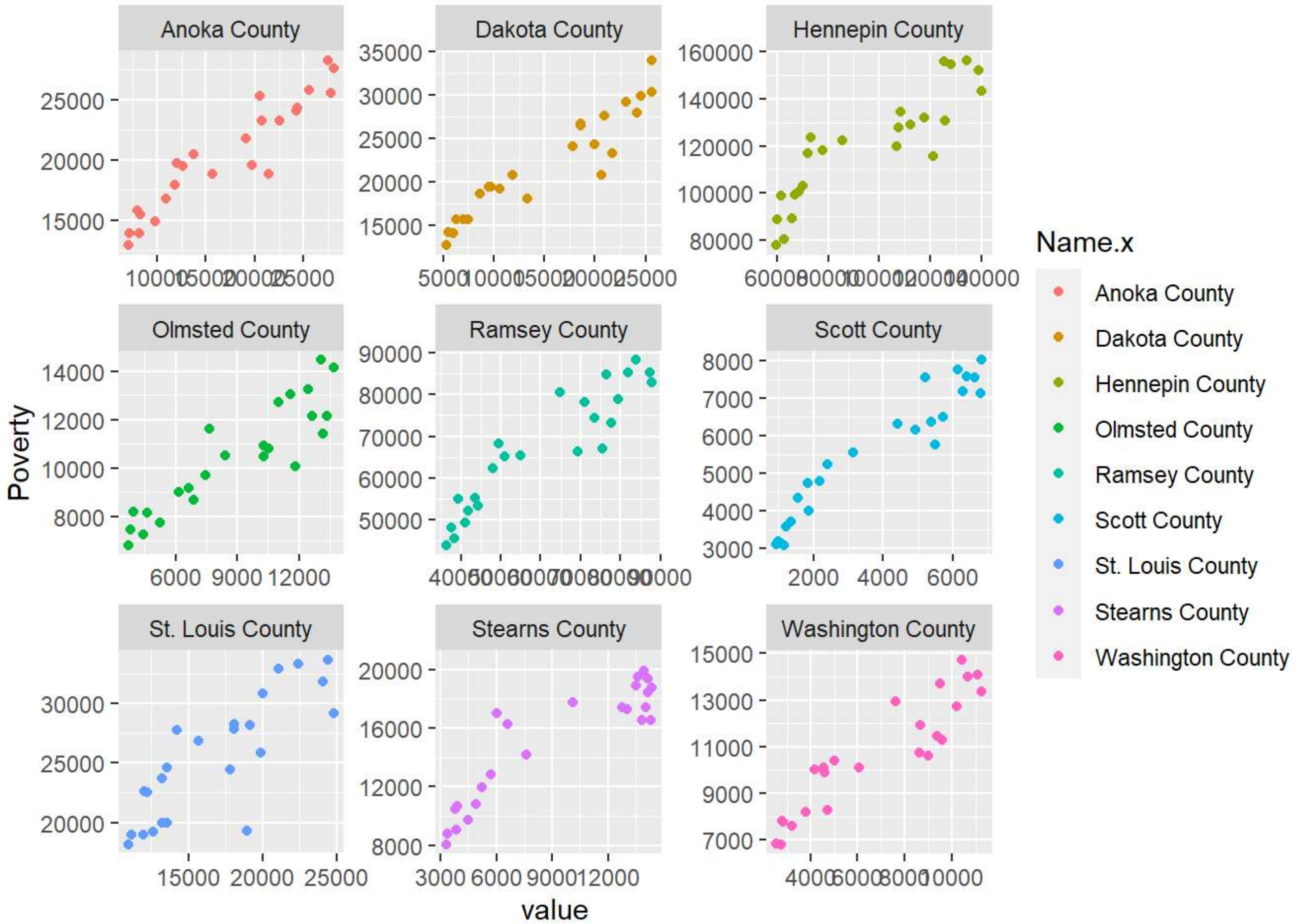

```
graph_final_ts %>% as_tibble() %>% ggplot(aes(x = Pop, y = Poverty, color = Name.x)) + geom_poin
t() + facet_wrap(vars(Name.x), scales = "free")
```

```
## Warning: Removed 9 rows containing missing values (`geom_point()`).
```



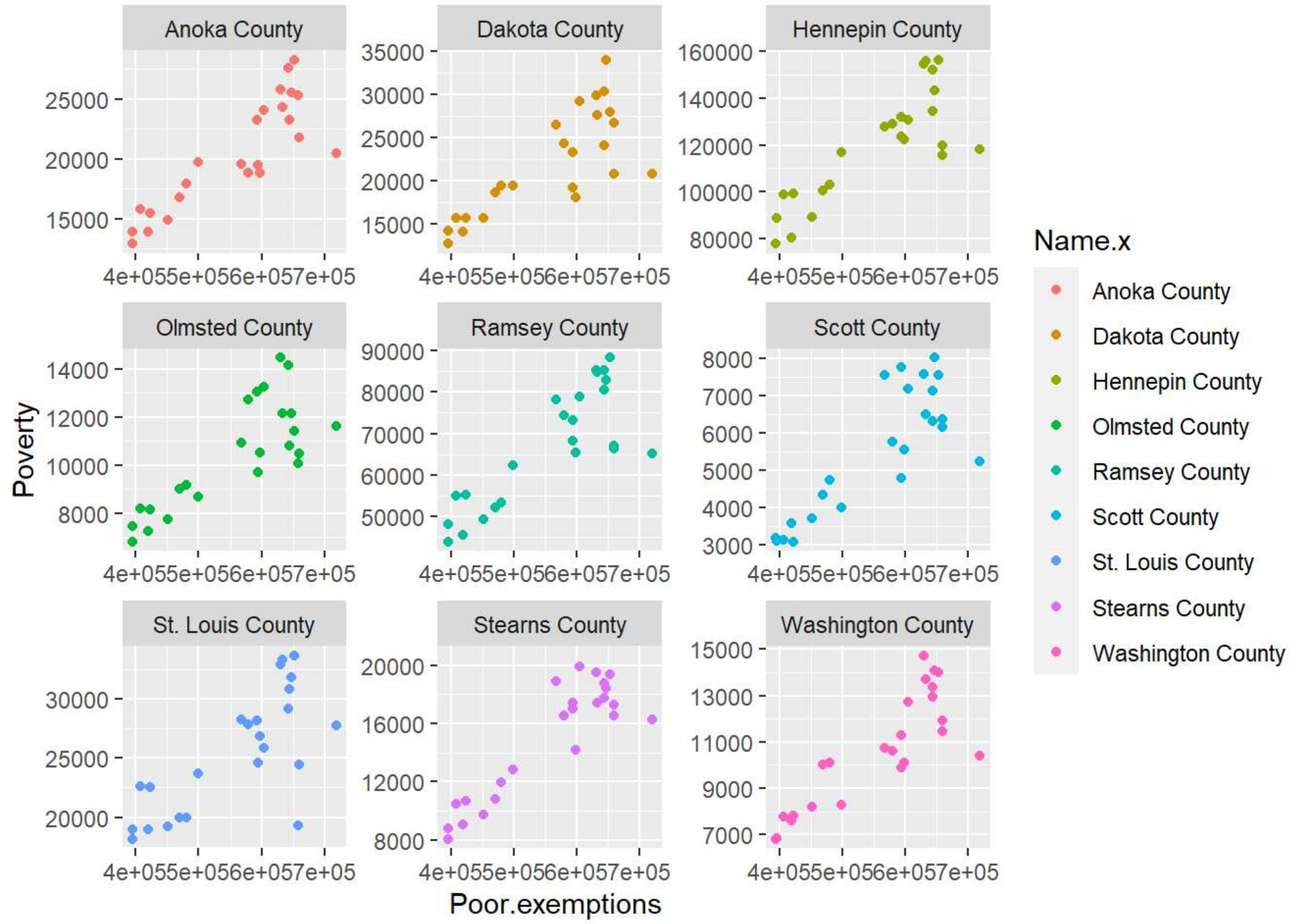

```
graph_final_ts %>% as_tibble() %>% ggplot(aes(x = value, y = Poverty, color = Name.x)) + geom_po
int() + facet_wrap(vars(Name.x), scales = "free")
```

```
## Warning: Removed 9 rows containing missing values (`geom_point()`).
```




```
graph_final_ts %>% as_tibble() %>% ggplot(aes(x = Poor.exemptions, y = Poverty, color = Name.x))
+ geom_point() + facet_wrap(vars(Name.x), scales = "free")
```

```
## Warning: Removed 9 rows containing missing values (`geom_point()`).
```



2 Linear Models

Part 2.1

```
library(forecast)
library(dplyr)
library(lubridate)
library(fpp3)
test_final_ts <-  final_join_ts %>% model(t1 = TSLM(log(Poverty) ~ log(Pop)),
                                          t2 = TSLM(log(Poverty) ~ log(value)),
                                          t3 = TSLM(log(Poverty) ~ log(Poor.exemptions)),
                                          t4 = TSLM(log(Poverty) ~ log(Poor.exemptions)+log(val
ue)),
                                          t5 = TSLM(log(Poverty) ~ log(Poor.exemptions)+log(Po
p)),
                                          t6 = TSLM(log(Poverty) ~ log(Pop)+ log(value)),
                                          t7 = TSLM(log(Poverty) ~ log(Pop) + log(value) + log
(Poor.exemptions)))


glance(test_final_ts) |> group_by(.model) %>% summarise(CV = sum(CV), AIC = sum(AIC)) %>% arrang
e(CV, AIC) |>
  dplyr::select(.model,CV, AIC)
```

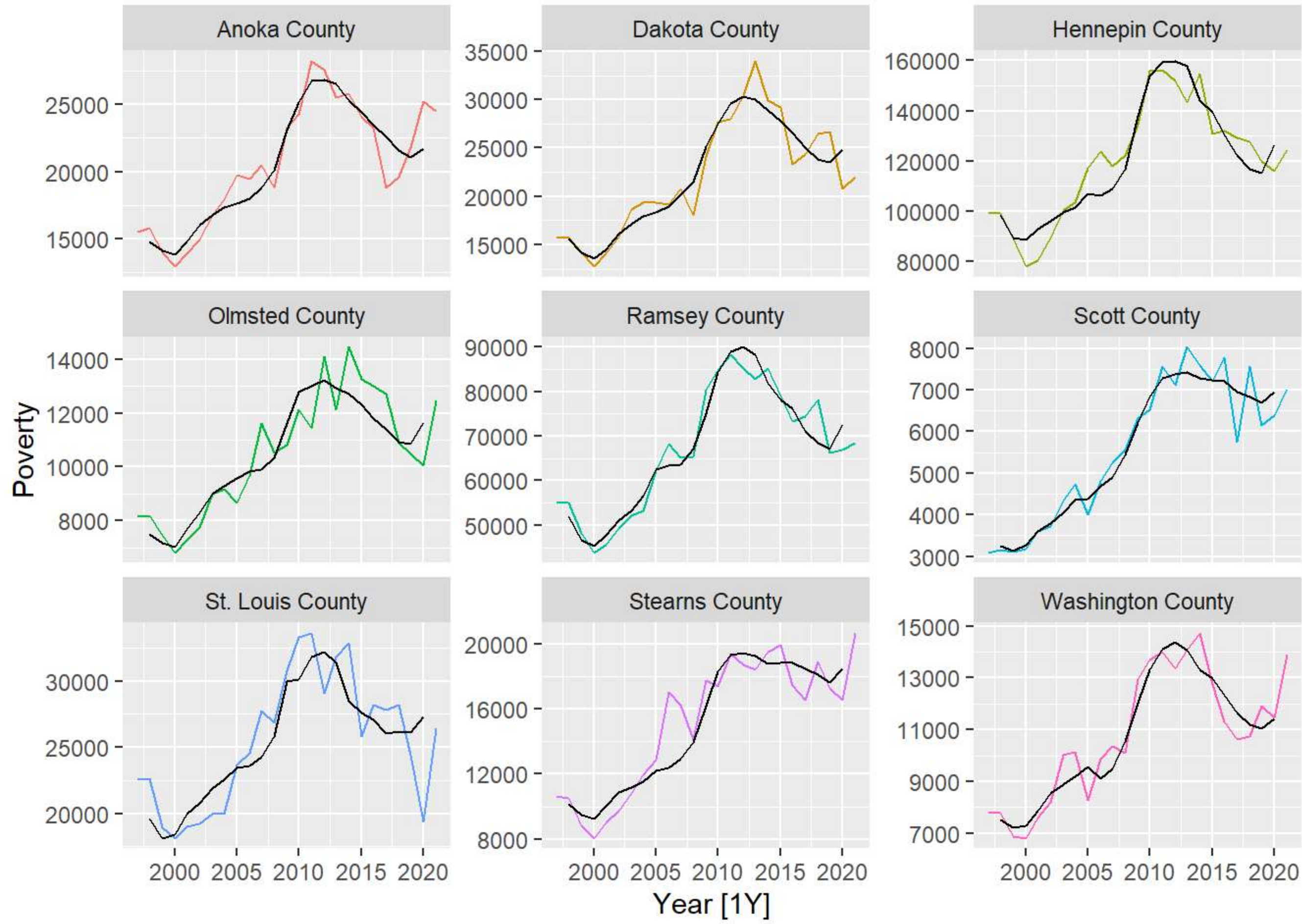
.model <chr>	CV <dbl>	AIC <dbl>
t6	1.026588	-8915.156
t7	1.058490	-8948.438
t2	1.089927	-9094.049
t4	1.160378	-9076.633
t5	1.418251	-8325.862
t3	1.421167	-8548.495
t1	1.798562	-8143.715

7 rows

The model that does the best across all counties is t6 which is TSLM(log(Poverty) ~ log(Pop)+ log(value)). This best model includes poverty, population, and value.

```
bestModel <- final_join_ts %>% model(TSLM(log(Poverty) ~ log(Pop)+ log(value)))

bestModel %>% filter(FIPS %in% FIPSvalue) %>% augment() %>% autoplot(Poverty) + geom_line(aes(y
= .fitted), color = "Black") + facet_wrap(vars(Name.x), scales = "free_y") + theme(legend.positi
on = "none")
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```



Part 2.2

```
plotRes<- bestModel %>% filter(FIPS %in% FIPSvalue) %>% augment()

autoplot(plotRes, .innov) + facet_wrap(vars(Name.x))
```

```
## Warning: Removed 18 rows containing missing values (`geom_line()`).
```


```
allCountyFIPS <- bestModel %>% pull(FIPS)

bestModel %>% augment() %>% features(.innov, ljung_box) %>% arrange(lb_pvalue)
```

FIPS <int>	Name.x <chr>	.model <chr>	l
27097	Morrison County	TSLM(log(Poverty) ~ log(Pop) + log(value))	10.9893
27145	Stearns County	TSLM(log(Poverty) ~ log(Pop) + log(value))	6.6098
27053	Hennepin County	TSLM(log(Poverty) ~ log(Pop) + log(value))	5.5879
27049	Goodhue County	TSLM(log(Poverty) ~ log(Pop) + log(value))	5.2105
27015	Brown County	TSLM(log(Poverty) ~ log(Pop) + log(value))	4.7831
27149	Stevens County	TSLM(log(Poverty) ~ log(Pop) + log(value))	4.7304
27059	Isanti County	TSLM(log(Poverty) ~ log(Pop) + log(value))	4.6021
27083	Lyon County	TSLM(log(Poverty) ~ log(Pop) + log(value))	4.5081
27139	Scott County	TSLM(log(Poverty) ~ log(Pop) + log(value))	3.6793
27107	Norman County	TSLM(log(Poverty) ~ log(Pop) + log(value))	3.3470

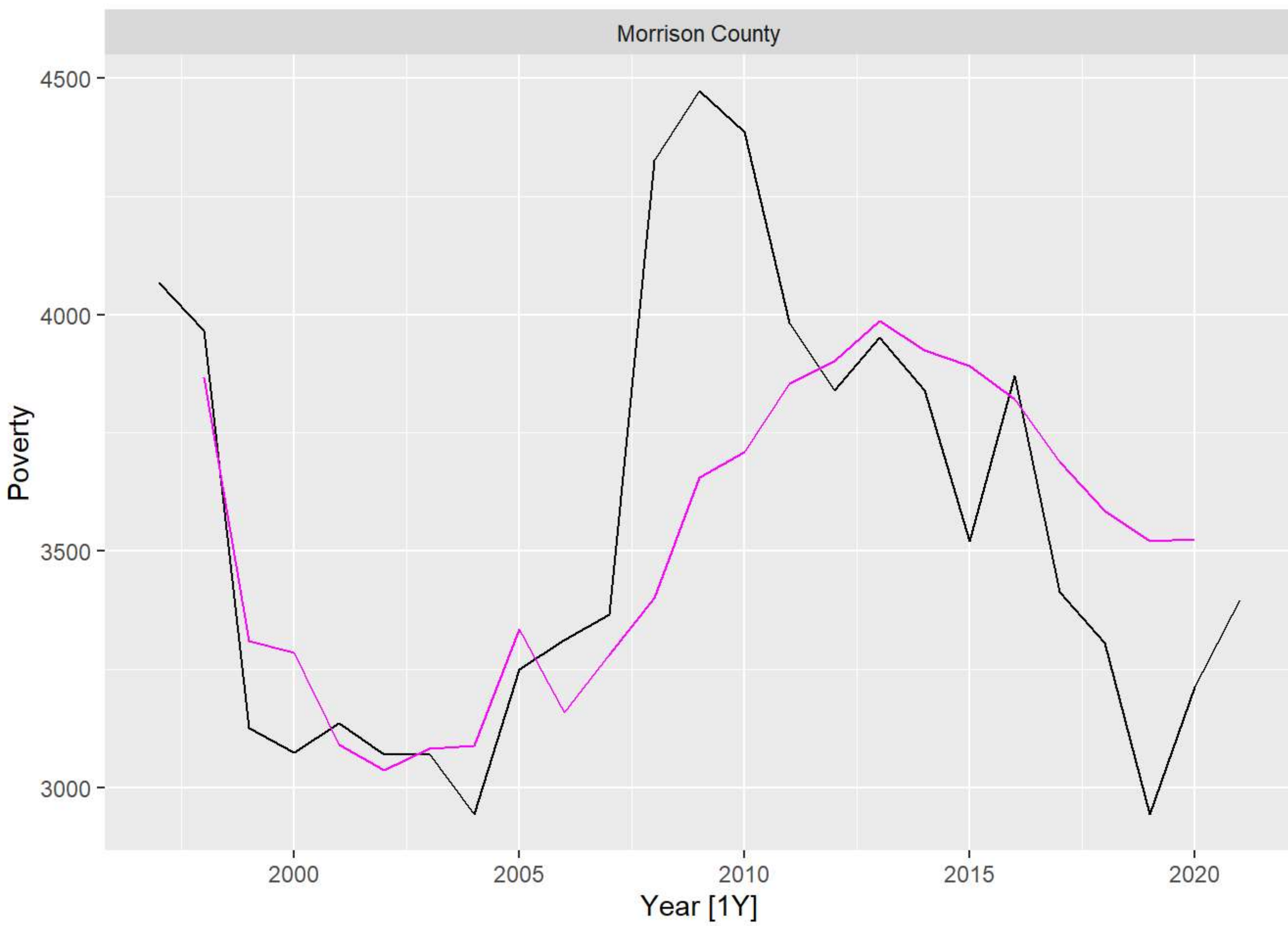
1-10 of 87 rows Previous 1 2 3 4 5 6 ... 9 Next

I found one county that was significantly different from white noise. The FIPS code for the county is 27097 and the name is Morrison County.

Because this p-value is so significantly different from white noise, I an going to make a residual plot of it

```
bestModel %>% filter(FIPS == 27097) %>% augment() %>% autoplot(Poverty) + geom_line(aes(y = .fitted), color = "Magenta") + facet_wrap(vars(Name.x), scales = "free") + theme(legend.position = "none")
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```

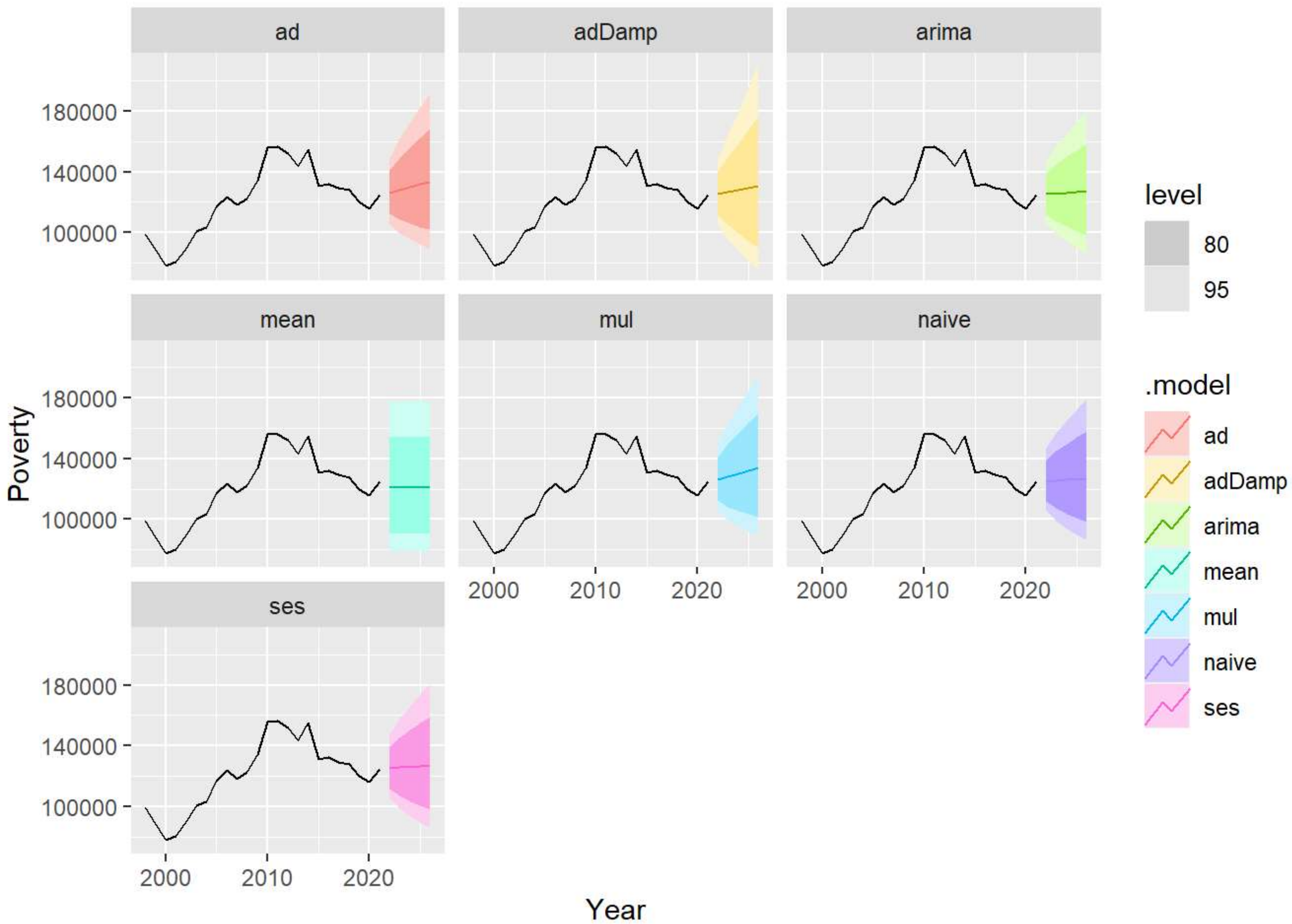


This model did a pretty good job as we have a few p values below 0.05. Except for one exception, Morrison county got a p value of 0.0009.

3 Stochastic Models

Part 3.1

```
saipe_hen <- saipe_mn %>% filter(!(Year %in% c(1996, 1989, 1993, 1995, 1997))) %>% as_tsibble(in
dex = Year, key = c(FIPS, Name)) %>% filter(FIPS == 27053)

hen_model <-  saipe_hen %>% model(naive = NAIVE(log(Poverty)),
                                  mean = MEAN(log(Poverty)),
                                  ses = ETS(log(Poverty) ~ error("A") + trend("N") + se
ason("N")),
                                  adDamp = ETS(log((Poverty)) ~ error('A') + trend('A
d')),
                                  ad = ETS(log((Poverty)) ~ error('A') + trend('A')),
                                  mul = ETS(log((Poverty)) ~ error('M') + trend('A')),
                                  arima = ARIMA(log(Poverty)))

hen_model  %>% forecast(h = '5 year') %>%  autoplot(saipe_hen) + facet_wrap(.~.model)
```




```
glance(hen_model)
```

FIPS <int>	Name <chr>	.mo… <chr>	sigma2 <dbl>	log_lik <dbl>	AIC <dbl>	AICc <dbl>	BIC <dbl>
27053	Hennepin County	naive	7.118151e-03	NA	NA	NA	NA

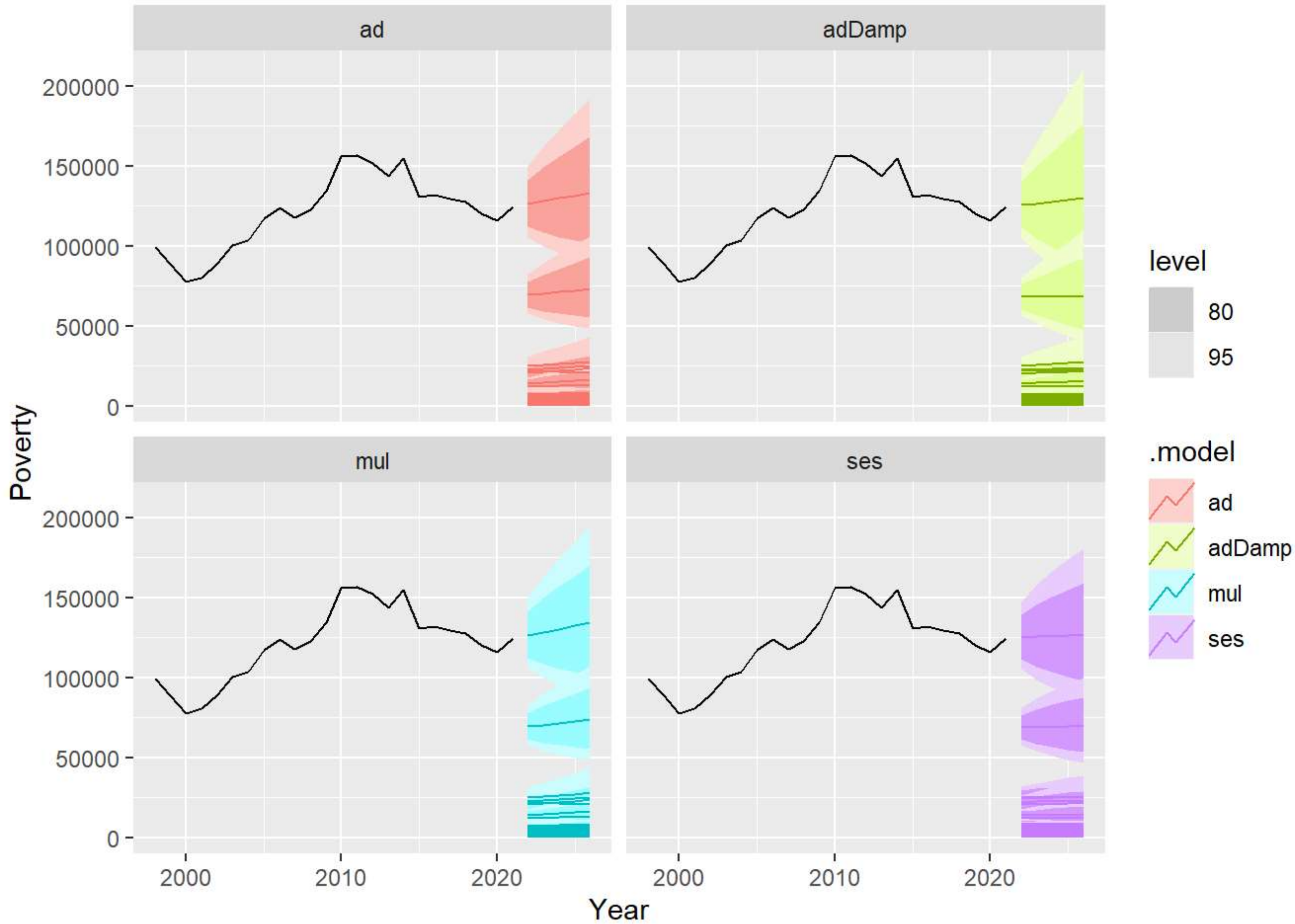
FIPS <int>	Name <chr>	.mo… <chr>	sigma2 <dbl>	log_lik <dbl>	AIC <dbl>	AICc <dbl>	BIC <dbl>	
27053	Hennepin County	mean	4.123677e-02	NA	NA	NA	NA	
27053	Hennepin County	ses	7.224379e-03	22.07102	-38.14204	-36.94204	-34.60788	0.006
27053	Hennepin County	adDamp	7.939057e-03	22.69826	-33.39652	-28.45535	-26.32820	0.006
27053	Hennepin County	ad	7.912110e-03	22.12354	-34.24708	-30.91375	-28.35681	0.006
27053	Hennepin County	mul	5.755145e-05	22.21374	-34.42747	-31.09414	-28.53720	0.006
27053	Hennepin County	arima	6.915717e-03	24.57451	-47.14901	-46.95854	-46.01352	

7 rows | 1-9 of 13 columns

The ARIMA model works the best.

Part 3.2

```
saipe_all <- saipe_mn %>% filter(!(Year %in% c(1996, 1989, 1993, 1995, 1997))) %>% as_tsibble(in
dex = Year, key = c(FIPS, Name))

all_model <-  saipe_all %>% model(
                                           ses = ETS(log(Poverty) ~ error("A") + trend("N") + se
ason("N")),
                                           adDamp = ETS(log((Poverty)) ~ error('A') + trend('A
d')),
                                           ad = ETS(log((Poverty)) ~ error('A') + trend('A')),
                                           mul = ETS(log((Poverty)) ~ error('M') + trend('A')))


all_model  %>% forecast(h = '5 year') %>%  autoplot(saipe_hen) + facet_wrap(.~.model)
```




```
glance(all_model) |> group_by(.model) %>% summarise(AIC = sum(AIC)) %>% arrange(AIC) |>
  dplyr::select(.model, AIC)
```

.model <chr>	AIC <dbl>
ses	-2096.389
mul	-1808.956
ad	-1792.932
adDamp	-1691.114

4 rows

The ses model did the best compared to the rest of the models. The reason why I chose ses is because it has the lowest AIC score

Part 3.3

```
arimaFit <- saipe_all %>% model(ARIMA(log(Poverty)))
arimaFit
```

FIPS <int>	Name <chr>	ARIMA(log(Poverty)) <lst_mdl>
27001	Aitkin County	<lst_mdl>
27003	Anoka County	<lst_mdl>
27005	Becker County	<lst_mdl>
27007	Beltrami County	<lst_mdl>
27009	Benton County	<lst_mdl>
27011	Big Stone County	<lst_mdl>
27013	Blue Earth County	<lst_mdl>
27015	Brown County	<lst_mdl>
27017	Carlton County	<lst_mdl>
27019	Carver County	<lst_mdl>

1-10 of 87 rows Previous **1** 2 3 4 5 6 ... 9 Next

```
arimaFits <- saipe_all %>% model(fit100 = ARIMA(log(Poverty) ~ 1 + pdq(1,0,0)),
                                 fit001 = ARIMA(log(Poverty) ~ 1 + pdq(0,0,1)))

glance(arimaFits) |> group_by(.model) %>% summarise(AIC = sum(AIC)) %>% arrange(AIC) |>
  dplyr::select(.model, AIC)
```

.model <chr>	AIC <dbl>
fit100	-2890.767
fit001	-2430.380

2 rows

(1,0,0) with mean and (0,0,1) with mean are the most common. from the data, (1,0,0) did the best.

Part 3.4

```
saipe_all_tr <- saipe_all |>
  stretch_tsibble(.init = 15, .step = 1)

fit_mn <- saipe_all_tr |>
  model(fit100 = ARIMA(log(Poverty) ~ 1 + pdq(1,0,0)),
        ses = ETS(log(Poverty) ~ error("A") + trend("N") + season("N")))
```

```
## Warning in wrap_arima(y, order = c(p, d, q), seasonal = list(order = c(P, :
## possible convergence problem: optim gave code = 1
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```
#{r crossValidate, cache = TRUE}
```

```
acc <- fit_mn  %>% forecast(h = 5) %>% fabletools::accuracy(data = saipe_all)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as m
issing.
## 5 observations are missing between 2022 and 2026
```

```
acc %>% group_by(.model) %>% summarize(sqrt(sum(RMSE*RMSE)))
```

.model <chr>	sqrt(sum(RMSE * RMSE)) <dbl>
fit100	14767.56
ses	21093.26

2 rows

fit100 is the winning model

4 Forecasts

```
county_fit <- saipe_all %>% model(fit100 = ARIMA(log(Poverty) ~ 1 + pdq(1,0,0)))

forecast_county_fit <- county_fit %>% forecast(h = '5 year') %>% filter(Year == 2026)
forecast_county_fit
```

FIPS <int>	Name <chr>	.model <chr>	Year <dbl>	Poverty <dist>	.mean <dbl>
27001	Aitkin County	fit100	2026	<dist>	2016.5337
27003	Anoka County	fit100	2026	<dist>	22947.8938
27005	Becker County	fit100	2026	<dist>	3910.9250
27007	Beltrami County	fit100	2026	<dist>	7371.4128
27009	Benton County	fit100	2026	<dist>	3516.1425
27011	Big Stone County	fit100	2026	<dist>	623.7314
27013	Blue Earth County	fit100	2026	<dist>	8346.1491
27015	Brown County	fit100	2026	<dist>	2038.4145
27017	Carlton County	fit100	2026	<dist>	3437.9499

FIPS <int>	Name <chr>	.model <chr>	Year <dbl>	Poverty <dist>	.mean <dbl>
27019	Carver County	fit100	2026	<dist>	4388.0930

1-10 of 87 rows Previous **1** 2 3 4 5 6 ... 9 Next

```
saipe_all_2021 <- saipe_all %>% filter(Year == 2021)

predInterval <- forecast_county_fit$.mean - saipe_all_2021$Poverty

percentInc <- predInterval / saipe_all_2021$Pop
percentInc
```

```
##  [1]  0.0199995975 -0.0043755936 -0.0005209070  0.0124713765 -0.0007564218
##  [6] -0.0036471633 -0.0046760367 -0.0095197602 -0.0048081651 -0.0054919046
## [11]  0.0115031456 -0.0090543961 -0.0042551978 -0.0215181409  0.0108869987
## [16] -0.0078919837 -0.0067864864 -0.0077888530 -0.0005830512  0.0002750693
## [21] -0.0102591145  0.0040501895 -0.0180010073  0.0022709735 -0.0044450544
## [26] -0.0003449262 -0.0020378421  0.0063040356  0.0044579159 -0.0080759047
## [31] -0.0029374856 -0.0060809735 -0.0121038574 -0.0189781861  0.0108450052
## [36]  0.0077298588  0.0017628111  0.0004702091  0.0069274649  0.0020012624
## [41]  0.0187558132 -0.0004011838 -0.0047751445  0.0040135899  0.0075527731
## [46]  0.0056455305 -0.0033637636 -0.0032742624  0.0040771040 -0.0067401031
## [51]  0.0059574756 -0.0124413159 -0.0007550823  0.0058657569 -0.0063020511
## [56]  0.0068476062  0.0037474871  0.0097180598  0.0004393243  0.0007866880
## [61]  0.0047891879 -0.0017646822  0.0136849143  0.0033268525  0.0026286080
## [66] -0.0060105721 -0.0064095185 -0.0156758093 -0.0034687685 -0.0030712612
## [71] -0.0065165093 -0.0003586393 -0.0114680895 -0.0104184813 -0.0056770822
## [76] -0.0039706988 -0.0007174410  0.0089118977  0.0003044520  0.0114856323
## [81]  0.0001509818 -0.0049778942 -0.0098698974 -0.0034224433 -0.0018183674
## [86] -0.0022250308  0.0029996122
```

```
highestValues <- tail(sort(percentInc), 5)
index <- which(percentInc %in% highestValues)

fiveCounties <- saipe_all_2021[c(1,4,41,58,63), 'Name']
fiveCounties
```

Name <chr>
Aitkin County
Beltrami County
Lincoln County
Pine County

Name <chr>
Red Lake County

5 rows

```
library(usmap)
library(ggplot2)
forecast_county_fit_usmap <- forecast_county_fit %>% as_tibble()

colnames(forecast_county_fit_usmap)[1] <- "fips"

names(forecast_county_fit_usmap)
```

```
## [1] "fips"    "Name"    ".model"  "Year"    "Poverty" ".mean"
```

```
plot_usmap(data = forecast_county_fit_usmap, values = ".mean", include = c("MN"), color = "blu
e") +
  scale_fill_continuous(low = "white", high = "blue", name = "Poverty Estimates", label = scale
s::comma) +
  labs(title = "Minnesota", subtitle = "Poverty Estimates for Minnesota Counties in 2026") +
  theme(legend.position = "right")
```