INFLUENZA HOSPITALIZATIONS RESULTS

Victor Felix

January 30, 2025

This document summarizes the results of the 12 models that were developed: AUTO ARIMA, AUTO ADJACENT, AUTO EPIWEEK, ES27 TEMPERATURE, ES27 ARIMA, ES27 ADJACENT, ES27 EPIWEEK, ES64 TEMPERATURE, ES64 ARIMA, ES64 ADJACENT, ES64 EPIWEEK, ES64 TEMPERATURE. It has a series of histograms and maps that compare the results of these models based on their WIS. You can read more about the specific covariates and ensemble techniques of these models in their own Rmarkdown documents.

Loading libraries.

```
library("tidyr")
library("MMWRweek")
library("data.table")
library("caret")
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library("purrr")
```

```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:caret':
##
##     lift
```

```
## The following object is masked from 'package:data.table':
##
##     transpose
```

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

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

```
library("tseries")
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library("gtools")
library("forecast")
library("scoringutils")
```

```
## Note: scoringutils is currently undergoing major development changes (with an update planned for the
```

```
library("covidHubUtils")
library("parallel")
library("future")#https://cran.r-project.org/web/packages/future/vignettes/future-4-issues.html
```

```
##
## Attaching package: 'future'

## The following object is masked from 'package:tseries':
##
##     value

## The following object is masked from 'package:caret':
##
##     cluster
```

```
library("listenv")
```

```
##
## Attaching package: 'listenv'

## The following object is masked from 'package:purrr':
##
##     map
```

```
library("epitools")
library("ggplot2")
library("sf")
```

```
## Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2() is TRUE
```

```r
library("forcats")
library("ggplot2")
library("sf")
library("dplyr")
library("scales")  # For `breaks_extended`
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard
```

```r
library("ggplot2")
library("broom")
library("fields")
```

```
## Loading required package: spam

## Spam version 2.10-0 (2023-10-23) is loaded.
## Type 'help( Spam)' or 'demo( spam)' for a short introduction
## and overview of this package.
## Help for individual functions is also obtained by adding the
## suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

##
## Attaching package: 'spam'

## The following objects are masked from 'package:base':
##
##     backsolve, forwardsolve

## Loading required package: viridisLite

##
## Try help(fields) to get started.
```

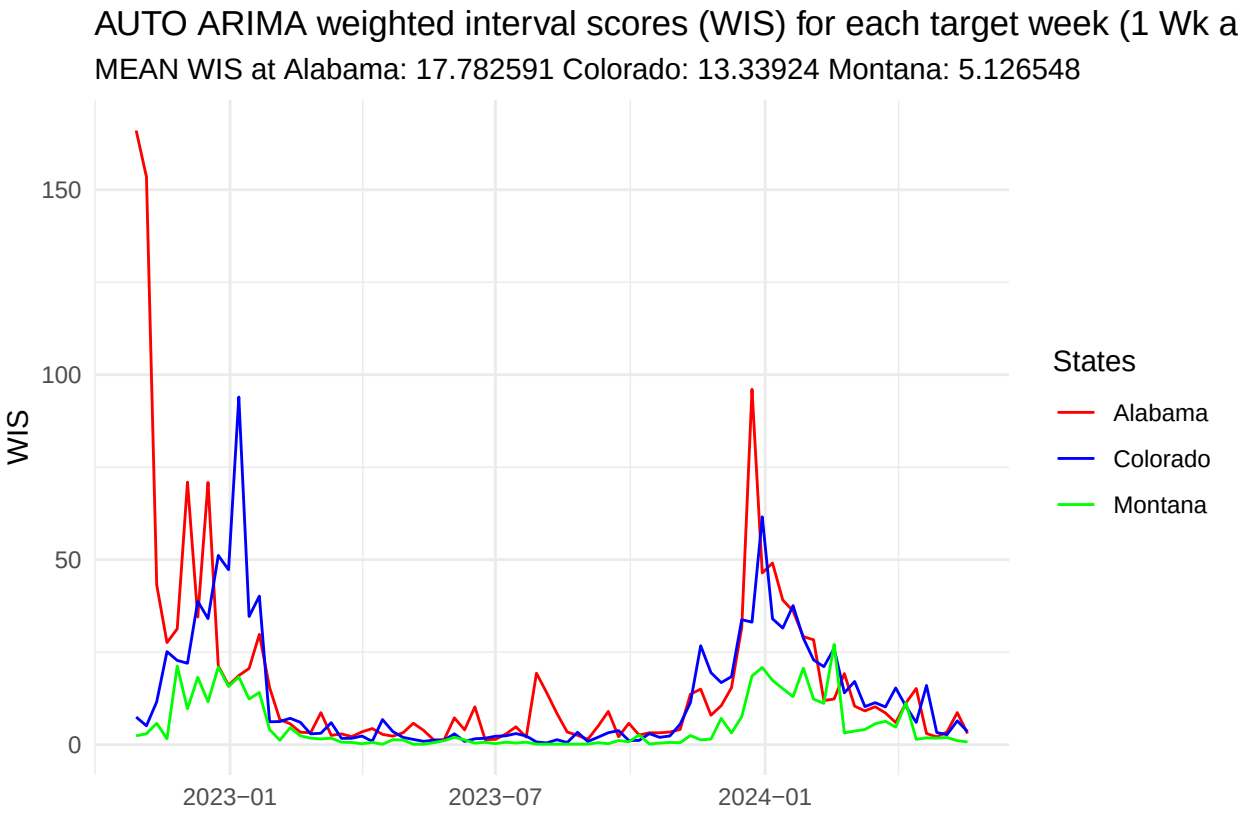
Loading the results of each model and the shapefiles of the maps.

```r
load("ES_ARIMA/ARIMA_MODELS_influenza_hospitalization.Rdata")
load("ES_ADJACENT/ADJACENT_MODELS_influenza_hospitalization.Rdata")
load("ES_EPIWEEK/EPIWEEK_MODELS_influenza_hospitalization.Rdata")
load("ES_TEMPERATURE/TEMPERATURE_MODELS_influenza_hospitalization.Rdata")

states <- read_sf("shapefiles/cb_2018_us_state_500k.shp")

states <- states %>%
  rename(STATE = NAME)
```

The results come as lists of dataframes with all states (e.g. AUTO_ARIMA_WEEK1_list) or as a single dataframe which combines all states (e.g. AUTO_ARIMA_WEEK1). Each model was run for forecasting 1-4 weeks ahead (e.g. AUTO_ARIMA_WEEK1, AUTO_ARIMA_WEEK2, AUTO_ARIMA_WEEK3, AUTO_ARIMA_WEEK4).

For easier visualization create some variables and plot some of the results for a specific model:

```
# Here we extract the results of the ES27_ARIMA model which was utilized to forecast 1 WEEK ahead for t

my_wis<- AUTO_ARIMA_WEEK1_list$Alabama[['WIS']] # get the actual cases
my_cases<- AUTO_ARIMA_WEEK1_list$Alabama[['cases']] # get the abs_error
my_forecasts<- AUTO_ARIMA_WEEK1_list$Alabama[['forecasts']] # get the number of models
my_abs_error<- AUTO_ARIMA_WEEK1_list$Alabama[['abs_error']] # get the abs_error
my_number_of_models<- AUTO_ARIMA_WEEK1_list$Alabama[['Number_of_models']] # get the number of models
my_dates<- AUTO_ARIMA_WEEK1_list$Alabama[['target_end_date']] # get the dates

# Here we will take a look at the WIS for 3 different states
Alabama<- AUTO_ARIMA_WEEK1_list$Alabama[['WIS']] # get the actual cases
Colorado<- AUTO_ARIMA_WEEK1_list$Colorado[['WIS']] # get the forecasts
Montana<- AUTO_ARIMA_WEEK1_list$Montana[['WIS']] # get the forecasts

# Assuming your data is in a data frame with columns 'my_dates', 'my_cases', and 'my_wis'
wis_data <- data.frame(
  my_dates = my_dates, # Ensure dates are in Date format
  Alabama = Alabama,
  Colorado = Colorado,
  Montana= Montana# Take the absolute value of WIS
)

# Calculate the mean WIS
mean_Alabama <- round(mean(wis_data$Alabama), 6)
mean_Colorado <- round(mean(wis_data$Colorado), 6)
mean_Montana <- round(mean(wis_data$Montana), 6)

# Let's look at WIS data for 3 different states
ggplot(wis_data) +
  geom_line(aes(x = my_dates, y = Alabama, color = "Alabama")) + # Plot cases and set legend label
  geom_line(aes(x = my_dates, y = Colorado, color = "Colorado")) + # Plot cases and set legend label
  geom_line(aes(x = my_dates, y = Montana, color = "Montana")) + # Plot WIS and set legend label
  scale_color_manual(values = c("Alabama" = "red", "Colorado" = "blue","Montana" = "green")) + # Define
  labs(
    title = "AUTO ARIMA weighted interval scores (WIS) for each target week (1 Wk ahead)",
    x = "",
    y = "WIS",
    color = "States", # Title for the legend
    subtitle = paste("MEAN WIS at Alabama:", mean_Alabama, "Colorado:", mean_Colorado, "Montana:", mean_
  ) +
  theme_minimal()
```



1 Week ahead

```
# Creating new columns with Epidemiological weeks based on target_end_week
AUTO_ARIMA_WEEK1$epiweek <- MMWRweek(AUTO_ARIMA_WEEK1$target_end_date)$MMWRweek
AUTO_ADJACENT_WEEK1$epiweek <- MMWRweek(AUTO_ADJACENT_WEEK1$target_end_date)$MMWRweek
AUTO_EPIWEEK_WEEK1$epiweek <- MMWRweek(AUTO_EPIWEEK_WEEK1$target_end_date)$MMWRweek
AUTO_TEMPERATURE_WEEK1$epiweek <- MMWRweek(AUTO_TEMPERATURE_WEEK1$target_end_date)$MMWRweek
ES27_ARIMA_WEEK1$epiweek <- MMWRweek(ES27_ARIMA_WEEK1$target_end_date)$MMWRweek
ES27_ADJACENT_WEEK1$epiweek <- MMWRweek(ES27_ADJACENT_WEEK1$target_end_date)$MMWRweek
ES27_EPIWEEK_WEEK1$epiweek <- MMWRweek(ES27_EPIWEEK_WEEK1$target_end_date)$MMWRweek
ES27_TEMPERATURE_WEEK1$epiweek <- MMWRweek(ES27_TEMPERATURE_WEEK1$target_end_date)$MMWRweek
ES64_ARIMA_WEEK1$epiweek <- MMWRweek(ES64_ARIMA_WEEK1$target_end_date)$MMWRweek
ES64_ADJACENT_WEEK1$epiweek <- MMWRweek(ES64_ADJACENT_WEEK1$target_end_date)$MMWRweek
ES64_EPIWEEK_WEEK1$epiweek <- MMWRweek(ES64_EPIWEEK_WEEK1$target_end_date)$MMWRweek
ES64_TEMPERATURE_WEEK1$epiweek <- MMWRweek(ES64_TEMPERATURE_WEEK1$target_end_date)$MMWRweek

# Dataframe that will be analysed for 1 Week Ahead
df_W1 <- data.frame(
  STATE = AUTO_ARIMA_WEEK1$State,
  Julian_date = AUTO_ARIMA_WEEK1$target_end_date,
  epiweek = AUTO_ARIMA_WEEK1$epiweek,
  AUTO_AR=AUTO_ARIMA_WEEK1$WIS,
  AUTO_ADJ=AUTO_ADJACENT_WEEK1$WIS,
  AUTO_EPI=AUTO_EPIWEEK_WEEK1$WIS,
  AUTO_TEMP=AUTO_TEMPERATURE_WEEK1$WIS,
```

```
  ES27_AR=ES27_ARIMA_WEEK1$WIS,
  ES27_ADJ=ES27_ADJACENT_WEEK1$WIS,
  ES27_EPI=ES27_EPIWEEK_WEEK1$WIS,
  ES27_TEMP=ES27_TEMPERATURE_WEEK1$WIS,
  ES64_AR=ES64_ARIMA_WEEK1$WIS,
  ES64_ADJ=ES64_ADJACENT_WEEK1$WIS,
  ES64_EPI=ES64_EPIWEEK_WEEK1$WIS,
  ES64_TEMP=ES64_TEMPERATURE_WEEK1$WIS
)

head(df_W1)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
## 1 Alabama  2022-10-29      43 165.99356 134.59990 168.27023 159.89830 139.84021
## 2 Alabama  2022-11-05      44 153.47250 137.06187 148.70080 218.67479 100.86004
## 3 Alabama  2022-11-12      45  43.18034  64.87635  42.51522  42.36641  34.93781
## 4 Alabama  2022-11-19      46  27.60494 106.98205  27.28608  28.17573  59.91360
## 5 Alabama  2022-11-26      47  31.30424  23.32520  29.88371  53.34637  34.09012
## 6 Alabama  2022-12-03      48  70.95218  70.22938 131.35736  70.00950  65.91650
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 113.79883 161.36980 184.41783 139.69660 116.46242 153.78382 181.98328
## 2  79.31657 106.21177 102.53653 105.44443  82.07607 110.52089 113.33075
## 3  58.25485  34.63321  34.79463  35.56109  64.74396  34.87466  35.44034
## 4 106.57038  58.66039  59.09638  64.66790 104.61953  64.29840  63.60785
## 5  34.30285  32.85987  41.23058  34.37792  28.77257  33.43688  43.22223
## 6  46.12597  64.01918  65.79085  65.94855  45.54739  61.20944  65.60103
```

2 Weeks ahead

```
# Creating new columns with Epidemiological weeks based on target_end_week
AUTO_ARIMA_WEEK2$epiweek <- MMWRweek(AUTO_ARIMA_WEEK2$target_end_date)$MMWRweek
AUTO_ADJACENT_WEEK2$epiweek <- MMWRweek(AUTO_ADJACENT_WEEK2$target_end_date)$MMWRweek
AUTO_EPIWEEK_WEEK2$epiweek <- MMWRweek(AUTO_EPIWEEK_WEEK2$target_end_date)$MMWRweek
AUTO_TEMPERATURE_WEEK2$epiweek <- MMWRweek(AUTO_TEMPERATURE_WEEK2$target_end_date)$MMWRweek
ES27_ARIMA_WEEK2$epiweek <- MMWRweek(ES27_ARIMA_WEEK2$target_end_date)$MMWRweek
ES27_ADJACENT_WEEK2$epiweek <- MMWRweek(ES27_ADJACENT_WEEK2$target_end_date)$MMWRweek
ES27_EPIWEEK_WEEK2$epiweek <- MMWRweek(ES27_EPIWEEK_WEEK2$target_end_date)$MMWRweek
ES27_TEMPERATURE_WEEK2$epiweek <- MMWRweek(ES27_TEMPERATURE_WEEK2$target_end_date)$MMWRweek
ES64_ARIMA_WEEK2$epiweek <- MMWRweek(ES64_ARIMA_WEEK2$target_end_date)$MMWRweek
ES64_ADJACENT_WEEK2$epiweek <- MMWRweek(ES64_ADJACENT_WEEK2$target_end_date)$MMWRweek
ES64_EPIWEEK_WEEK2$epiweek <- MMWRweek(ES64_EPIWEEK_WEEK2$target_end_date)$MMWRweek
ES64_TEMPERATURE_WEEK2$epiweek <- MMWRweek(ES64_TEMPERATURE_WEEK2$target_end_date)$MMWRweek

# Dataframe that will be analysed for 2 Weeks Ahead
df_W2 <- data.frame(
  STATE = AUTO_ARIMA_WEEK2$State,
  Julian_date = AUTO_ARIMA_WEEK2$target_end_date,
  epiweek = AUTO_ARIMA_WEEK2$epiweek,
  AUTO_AR=AUTO_ARIMA_WEEK2$WIS,
  AUTO_ADJ=AUTO_ADJACENT_WEEK2$WIS,
  AUTO_EPI=AUTO_EPIWEEK_WEEK2$WIS,
```

```
  AUTO_TEMP=AUTO_TEMPERATURE_WEEK2$WIS,
  ES27_AR=ES27_ARIMA_WEEK2$WIS,
  ES27_ADJ=ES27_ADJACENT_WEEK2$WIS,
  ES27_EPI=ES27_EPIWEEK_WEEK2$WIS,
  ES27_TEMP=ES27_TEMPERATURE_WEEK2$WIS,
  ES64_AR=ES64_ARIMA_WEEK2$WIS,
  ES64_ADJ=ES64_ADJACENT_WEEK2$WIS,
  ES64_EPI=ES64_EPIWEEK_WEEK2$WIS,
  ES64_TEMP=ES64_TEMPERATURE_WEEK2$WIS
)

head(df_W2)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
## 1 Alabama  2022-11-05      44 278.56806 241.40226 286.13069 273.93279 232.80930
## 2 Alabama  2022-11-12      45 124.59739 129.68545 120.33060 226.84652  64.06489
## 3 Alabama  2022-11-19      46  27.81477 103.21610  27.57103  28.24921  59.26661
## 4 Alabama  2022-11-26      47  29.12194  66.69652  28.30185  29.22640  64.76138
## 5 Alabama  2022-12-03      48 169.48960  98.42353 160.01289  46.88537  57.02261
## 6 Alabama  2022-12-10      49  28.24570  24.39800  80.90383  28.41791  29.61265
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 202.01850 268.58074 305.50857 239.75207 200.80489 260.04622 300.76935
## 2  47.41692  68.13353  65.86551  69.53261  51.37320  72.74852  78.71901
## 3 107.17886  55.66006  57.74035  62.50216  91.88819  57.50373  60.03204
## 4  84.36786  62.02135  64.06523  56.38537  61.15356  55.67415  56.14083
## 5  57.93426  60.83642  51.58261  56.89796  67.21419  59.64801  50.56801
## 6  32.19118  30.80162  29.69000  29.33798  36.24796  30.91103  29.51284
```

3 Weeks ahead

```
# Creating new columns with Epidemiological weeks based on target_end_week
AUTO_ARIMA_WEEK3$epiweek <- MMWRweek(AUTO_ARIMA_WEEK3$target_end_date)$MMWRweek
AUTO_ADJACENT_WEEK3$epiweek <- MMWRweek(AUTO_ADJACENT_WEEK3$target_end_date)$MMWRweek
AUTO_EPIWEEK_WEEK3$epiweek <- MMWRweek(AUTO_EPIWEEK_WEEK3$target_end_date)$MMWRweek
AUTO_TEMPERATURE_WEEK3$epiweek <- MMWRweek(AUTO_TEMPERATURE_WEEK3$target_end_date)$MMWRweek
ES27_ARIMA_WEEK3$epiweek <- MMWRweek(ES27_ARIMA_WEEK3$target_end_date)$MMWRweek
ES27_ADJACENT_WEEK3$epiweek <- MMWRweek(ES27_ADJACENT_WEEK3$target_end_date)$MMWRweek
ES27_EPIWEEK_WEEK3$epiweek <- MMWRweek(ES27_EPIWEEK_WEEK3$target_end_date)$MMWRweek
ES27_TEMPERATURE_WEEK3$epiweek <- MMWRweek(ES27_TEMPERATURE_WEEK3$target_end_date)$MMWRweek
ES64_ARIMA_WEEK3$epiweek <- MMWRweek(ES64_ARIMA_WEEK3$target_end_date)$MMWRweek
ES64_ADJACENT_WEEK3$epiweek <- MMWRweek(ES64_ADJACENT_WEEK3$target_end_date)$MMWRweek
ES64_EPIWEEK_WEEK3$epiweek <- MMWRweek(ES64_EPIWEEK_WEEK3$target_end_date)$MMWRweek
ES64_TEMPERATURE_WEEK3$epiweek <- MMWRweek(ES64_TEMPERATURE_WEEK3$target_end_date)$MMWRweek

# Dataframe that will be analysed for 3 Weeks Ahead
df_W3 <- data.frame(
  STATE = AUTO_ARIMA_WEEK3$State,
  Julian_date = AUTO_ARIMA_WEEK3$target_end_date,
  epiweek = AUTO_ARIMA_WEEK3$epiweek,
  AUTO_AR=AUTO_ARIMA_WEEK3$WIS,
  AUTO_ADJ=AUTO_ADJACENT_WEEK3$WIS,
```

```
  AUTO_EPI=AUTO_EPIWEEK_WEEK3$WIS,
  AUTO_TEMP=AUTO_TEMPERATURE_WEEK3$WIS,
  ES27_AR=ES27_ARIMA_WEEK3$WIS,
  ES27_ADJ=ES27_ADJACENT_WEEK3$WIS,
  ES27_EPI=ES27_EPIWEEK_WEEK3$WIS,
  ES27_TEMP=ES27_TEMPERATURE_WEEK3$WIS,
  ES64_AR=ES64_ARIMA_WEEK3$WIS,
  ES64_ADJ=ES64_ADJACENT_WEEK3$WIS,
  ES64_EPI=ES64_EPIWEEK_WEEK3$WIS,
  ES64_TEMP=ES64_TEMPERATURE_WEEK3$WIS
)

head(df_W3)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
## 1 Alabama  2022-11-12      45 256.69274 234.72711 260.03768 252.71569 197.02540
## 2 Alabama  2022-11-19      46  96.92266 106.29578  94.60874 184.45042  38.91557
## 3 Alabama  2022-11-26      47  40.19262  77.08429  40.62706  39.49493  65.55168
## 4 Alabama  2022-12-03      48 100.49292  39.70304 137.87230  99.10252  54.21304
## 5 Alabama  2022-12-10      49 111.19752  33.81056 103.29250  44.23160  39.58325
## 6 Alabama  2022-12-17      50  45.86770  26.22535  53.48784  43.42842  46.79111
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 167.21673 235.28081 282.40390 202.14296 175.26620 218.85167 277.59133
## 2  30.92253  39.78795  39.68842  43.14254  33.86491  43.13370  49.13318
## 3  97.03931  64.54899  64.04576  69.05186  66.16617  66.73516  66.45202
## 4  48.38340  55.86974  54.24828  54.86322  46.91295  55.59332  55.03538
## 5  31.82425  38.91870  43.95971  38.78011  30.11360  38.51782  44.07631
## 6  59.77029  48.96399  45.97107  45.44272  67.02138  48.26064  44.68795
```

4 Weeks ahead

```
# Creating new columns with Epidemiological weeks based on target_end_week
AUTO_ARIMA_WEEK4$epiweek <- MMWRweek(AUTO_ARIMA_WEEK4$target_end_date)$MMWRweek
AUTO_ADJACENT_WEEK4$epiweek <- MMWRweek(AUTO_ADJACENT_WEEK4$target_end_date)$MMWRweek
AUTO_EPIWEEK_WEEK4$epiweek <- MMWRweek(AUTO_EPIWEEK_WEEK4$target_end_date)$MMWRweek
AUTO_TEMPERATURE_WEEK4$epiweek <- MMWRweek(AUTO_TEMPERATURE_WEEK4$target_end_date)$MMWRweek
ES27_ARIMA_WEEK4$epiweek <- MMWRweek(ES27_ARIMA_WEEK4$target_end_date)$MMWRweek
ES27_ADJACENT_WEEK4$epiweek <- MMWRweek(ES27_ADJACENT_WEEK4$target_end_date)$MMWRweek
ES27_EPIWEEK_WEEK4$epiweek <- MMWRweek(ES27_EPIWEEK_WEEK4$target_end_date)$MMWRweek
ES27_TEMPERATURE_WEEK4$epiweek <- MMWRweek(ES27_TEMPERATURE_WEEK4$target_end_date)$MMWRweek
ES64_ARIMA_WEEK4$epiweek <- MMWRweek(ES64_ARIMA_WEEK4$target_end_date)$MMWRweek
ES64_ADJACENT_WEEK4$epiweek <- MMWRweek(ES64_ADJACENT_WEEK4$target_end_date)$MMWRweek
ES64_EPIWEEK_WEEK4$epiweek <- MMWRweek(ES64_EPIWEEK_WEEK4$target_end_date)$MMWRweek
ES64_TEMPERATURE_WEEK4$epiweek <- MMWRweek(ES64_TEMPERATURE_WEEK4$target_end_date)$MMWRweek

# Dataframe that will be analysed for 4 Weeks Ahead
df_W4 <- data.frame(
  STATE = AUTO_ARIMA_WEEK4$State,
  Julian_date = AUTO_ARIMA_WEEK4$target_end_date,
  epiweek = AUTO_ARIMA_WEEK4$epiweek,
  AUTO_AR=AUTO_ARIMA_WEEK4$WIS,
```

```
  AUTO_ADJ=AUTO_ADJACENT_WEEK4$WIS,
  AUTO_EPI=AUTO_EPIWEEK_WEEK4$WIS,
  AUTO_TEMP=AUTO_TEMPERATURE_WEEK4$WIS,
  ES27_AR=ES27_ARIMA_WEEK4$WIS,
  ES27_ADJ=ES27_ADJACENT_WEEK4$WIS,
  ES27_EPI=ES27_EPIWEEK_WEEK4$WIS,
  ES27_TEMP=ES27_TEMPERATURE_WEEK4$WIS,
  ES64_AR=ES64_ARIMA_WEEK4$WIS,
  ES64_ADJ=ES64_ADJACENT_WEEK4$WIS,
  ES64_EPI=ES64_EPIWEEK_WEEK4$WIS,
  ES64_TEMP=ES64_TEMPERATURE_WEEK4$WIS
)

head(df_W4)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
## 1 Alabama  2022-11-19      46 195.98931 179.94665 192.32811 185.48556 134.06397
## 2 Alabama  2022-11-26      47 120.80173 147.64149 119.05866 207.16983  44.16024
## 3 Alabama  2022-12-03      48 140.83402  42.57211 143.65877 146.12380  65.51428
## 4 Alabama  2022-12-10      49  45.18555  45.20980  89.26922  44.69752  78.25003
## 5 Alabama  2022-12-17      50  74.64113  19.44344  69.08022  94.18565  65.57697
## 6 Alabama  2022-12-24      51  53.66508  23.55378  78.29285  53.01212  51.23739
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 102.64961 169.22695 215.94315 141.35692 118.05543 154.63299 211.98099
## 2  34.89809  45.50517  45.19062  50.32785  41.52011  50.00358  57.61446
## 3  59.31988  69.92298  65.53153  68.93642  51.96873  70.83908  68.86727
## 4  81.69241  74.85403  77.60473  78.93327  75.38058  77.48293  78.46584
## 5  50.87631  59.63241  76.11599  64.70533  41.28906  60.63930  77.00026
## 6  57.53025  54.94971  51.13820  50.26288  59.73382  55.99724  50.10286
```

Filter only the flu season

```
# Filter the dataframe for epiweek >= 40 or epiweek <= 20
filtered_df_W1 <- df_W1 %>%
  filter(epiweek >= 40 | epiweek <= 20)

# Display the filtered dataset
head(filtered_df_W1)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
## 1 Alabama  2022-10-29      43 165.99356 134.59990 168.27023 159.89830 139.84021
## 2 Alabama  2022-11-05      44 153.47250 137.06187 148.70080 218.67479 100.86004
## 3 Alabama  2022-11-12      45  43.18034  64.87635  42.51522  42.36641  34.93781
## 4 Alabama  2022-11-19      46  27.60494 106.98205  27.28608  28.17573  59.91360
## 5 Alabama  2022-11-26      47  31.30424  23.32520  29.88371  53.34637  34.09012
## 6 Alabama  2022-12-03      48  70.95218  70.22938 131.35736  70.00950  65.91650
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 113.79883 161.36980 184.41783 139.69660 116.46242 153.78382 181.98328
## 2  79.31657 106.21177 102.53653 105.44443  82.07607 110.52089 113.33075
## 3  58.25485  34.63321  34.79463  35.56109  64.74396  34.87466  35.44034
## 4 106.57038  58.66039  59.09638  64.66790 104.61953  64.29840  63.60785
## 5  34.30285  32.85987  41.23058  34.37792  28.77257  33.43688  43.22223
## 6  46.12597  64.01918  65.79085  65.94855  45.54739  61.20944  65.60103
```

```r
# Filter the dataframe for epiweek >= 40 or epiweek <= 20
filtered_df_W2 <- df_W2 %>%
  filter(epiweek >= 40 | epiweek <= 20)

# Display the filtered dataset
head(filtered_df_W2)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
## 1 Alabama  2022-11-05      44 278.56806 241.40226 286.13069 273.93279 232.80930
## 2 Alabama  2022-11-12      45 124.59739 129.68545 120.33060 226.84652  64.06489
## 3 Alabama  2022-11-19      46  27.81477 103.21610  27.57103  28.24921  59.26661
## 4 Alabama  2022-11-26      47  29.12194  66.69652  28.30185  29.22640  64.76138
## 5 Alabama  2022-12-03      48 169.48960  98.42353 160.01289  46.88537  57.02261
## 6 Alabama  2022-12-10      49  28.24570  24.39800  80.90383  28.41791  29.61265
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 202.01850 268.58074 305.50857 239.75207 200.80489 260.04622 300.76935
## 2  47.41692  68.13353  65.86551  69.53261  51.37320  72.74852  78.71901
## 3 107.17886  55.66006  57.74035  62.50216  91.88819  57.50373  60.03204
## 4  84.36786  62.02135  64.06523  56.38537  61.15356  55.67415  56.14083
## 5  57.93426  60.83642  51.58261  56.89796  67.21419  59.64801  50.56801
## 6  32.19118  30.80162  29.69000  29.33798  36.24796  30.91103  29.51284
```

```r
# Filter the dataframe for epiweek >= 40 or epiweek <= 20
filtered_df_W3 <- df_W3 %>%
  filter(epiweek >= 40 | epiweek <= 20)

# Display the filtered dataset
head(filtered_df_W3)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
## 1 Alabama  2022-11-12      45 256.69274 234.72711 260.03768 252.71569 197.02540
## 2 Alabama  2022-11-19      46  96.92266 106.29578  94.60874 184.45042  38.91557
## 3 Alabama  2022-11-26      47  40.19262  77.08429  40.62706  39.49493  65.55168
## 4 Alabama  2022-12-03      48 100.49292  39.70304 137.87230  99.10252  54.21304
## 5 Alabama  2022-12-10      49 111.19752  33.81056 103.29250  44.23160  39.58325
## 6 Alabama  2022-12-17      50  45.86770  26.22535  53.48784  43.42842  46.79111
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 167.21673 235.28081 282.40390 202.14296 175.26620 218.85167 277.59133
## 2  30.92253  39.78795  39.68842  43.14254  33.86491  43.13370  49.13318
## 3  97.03931  64.54899  64.04576  69.05186  66.16617  66.73516  66.45202
## 4  48.38340  55.86974  54.24828  54.86322  46.91295  55.59332  55.03538
## 5  31.82425  38.91870  43.95971  38.78011  30.11360  38.51782  44.07631
## 6  59.77029  48.96399  45.97107  45.44272  67.02138  48.26064  44.68795
```

```r
# Filter the dataframe for epiweek >= 40 or epiweek <= 20
filtered_df_W4 <- df_W4 %>%
  filter(epiweek >= 40 | epiweek <= 20)

# Display the filtered dataset
head(filtered_df_W1)
```

```
##     STATE Julian_date epiweek   AUTO_AR  AUTO_ADJ  AUTO_EPI AUTO_TEMP   ES27_AR
```

```
## 1 Alabama  2022-10-29      43 165.99356 134.59990 168.27023 159.89830 139.84021
## 2 Alabama  2022-11-05      44 153.47250 137.06187 148.70080 218.67479 100.86004
## 3 Alabama  2022-11-12      45  43.18034  64.87635  42.51522  42.36641  34.93781
## 4 Alabama  2022-11-19      46  27.60494 106.98205  27.28608  28.17573  59.91360
## 5 Alabama  2022-11-26      47  31.30424  23.32520  29.88371  53.34637  34.09012
## 6 Alabama  2022-12-03      48  70.95218  70.22938 131.35736  70.00950  65.91650
##    ES27_ADJ  ES27_EPI ES27_TEMP   ES64_AR  ES64_ADJ  ES64_EPI ES64_TEMP
## 1 113.79883 161.36980 184.41783 139.69660 116.46242 153.78382 181.98328
## 2  79.31657 106.21177 102.53653 105.44443  82.07607 110.52089 113.33075
## 3  58.25485  34.63321  34.79463  35.56109  64.74396  34.87466  35.44034
## 4 106.57038  58.66039  59.09638  64.66790 104.61953  64.29840  63.60785
## 5  34.30285  32.85987  41.23058  34.37792  28.77257  33.43688  43.22223
## 6  46.12597  64.01918  65.79085  65.94855  45.54739  61.20944  65.60103
```

Calculate mean weighted interval score for all forecast weeks on each model.

```
# Define the function
calculate_mean_wis <- function(data) {
  data %>%
    group_by(STATE) %>%
    summarize(
      AUTO_AR = mean(AUTO_AR, na.rm = TRUE),
      AUTO_ADJ = mean(AUTO_ADJ, na.rm = TRUE),
      AUTO_EPI = mean(AUTO_EPI, na.rm = TRUE),
      AUTO_TMP = mean(AUTO_TEMP, na.rm = TRUE),
      ES27_AR = mean(ES27_AR, na.rm = TRUE),
      ES27_ADJ = mean(ES27_ADJ, na.rm = TRUE),
      ES27_EPI = mean(ES27_EPI, na.rm = TRUE),
      ES27_TMP = mean(ES27_TEMP, na.rm = TRUE),
      ES64_AR = mean(ES64_AR, na.rm = TRUE),
      ES64_ADJ = mean(ES64_ADJ, na.rm = TRUE),
      ES64_EPI = mean(ES64_EPI, na.rm = TRUE),
      ES64_TMP = mean(ES64_TEMP, na.rm = TRUE)
    )
}

# Now you can use the function with any dataframe
W1 <- calculate_mean_wis(filtered_df_W1)
W2 <- calculate_mean_wis(filtered_df_W2)
W3 <- calculate_mean_wis(filtered_df_W3)
W4 <- calculate_mean_wis(filtered_df_W4)

# Display the resulting dataframe
head(W1)
```

```
## # A tibble: 6 x 13
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     21.5     20.3     22.4     23.9    20.8     19.1     21.3     21.8
## 2 Arizona     45.6     37.0     45.3     45.6    44.6     36.3     45.2     43.5
## 3 Arkansas    20.0     18.4     19.6     21.5    21.6     18.4     21.2     22.1
## 4 Califor~    81.5     77.9     94.9     94.6    89.5     79.7     88.0     96.5
## 5 Colorado    16.8     14.4     17.2     18.1    17.3     15.1     16.8     18.2
## 6 Connect~    19.2     16.8     18.7     19.0    19.8     16.9     19.1     19.2
```

```
## # i 4 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>
```

```
head(W2)
```

```
## # A tibble: 6 x 13
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     31.1     29.3     30.8     30.7    28.6     27.4     28.8     29.1
## 2 Arizona     78.1     66.1     76.7     76.9    72.6     64.7     72.8     69.4
## 3 Arkansas    30.4     27.2     30.6     32.4    35.6     28.6     34.8     35.2
## 4 Califor~   160.     152.     190.     155.    166.     151.     167.     161.
## 5 Colorado    26.7     23.2     28.6     28.4    28.4     24.7     27.2     29.1
## 6 Connect~    31.4     26.1     29.2     30.2    31.9     26.1     30.3     30.0
## # i 4 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>
```

```
head(W3)
```

```
## # A tibble: 6 x 13
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     38.0     34.0     36.9     39.2    33.3     33.0     33.5     34.1
## 2 Arizona    106.      94.6    104.     104.    100.      91.0     95.8     96.2
## 3 Arkansas    40.2     33.8     38.9     42.0    48.4     36.5     46.5     46.6
## 4 Califor~   249.     225.     284.     219.    245.     219.     248.     229.
## 5 Colorado    35.7     32.6     40.3     38.8    39.9     35.4     37.7     40.4
## 6 Connect~    46.0     38.2     40.8     43.9    45.9     37.7     44.3     42.9
## # i 4 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>
```

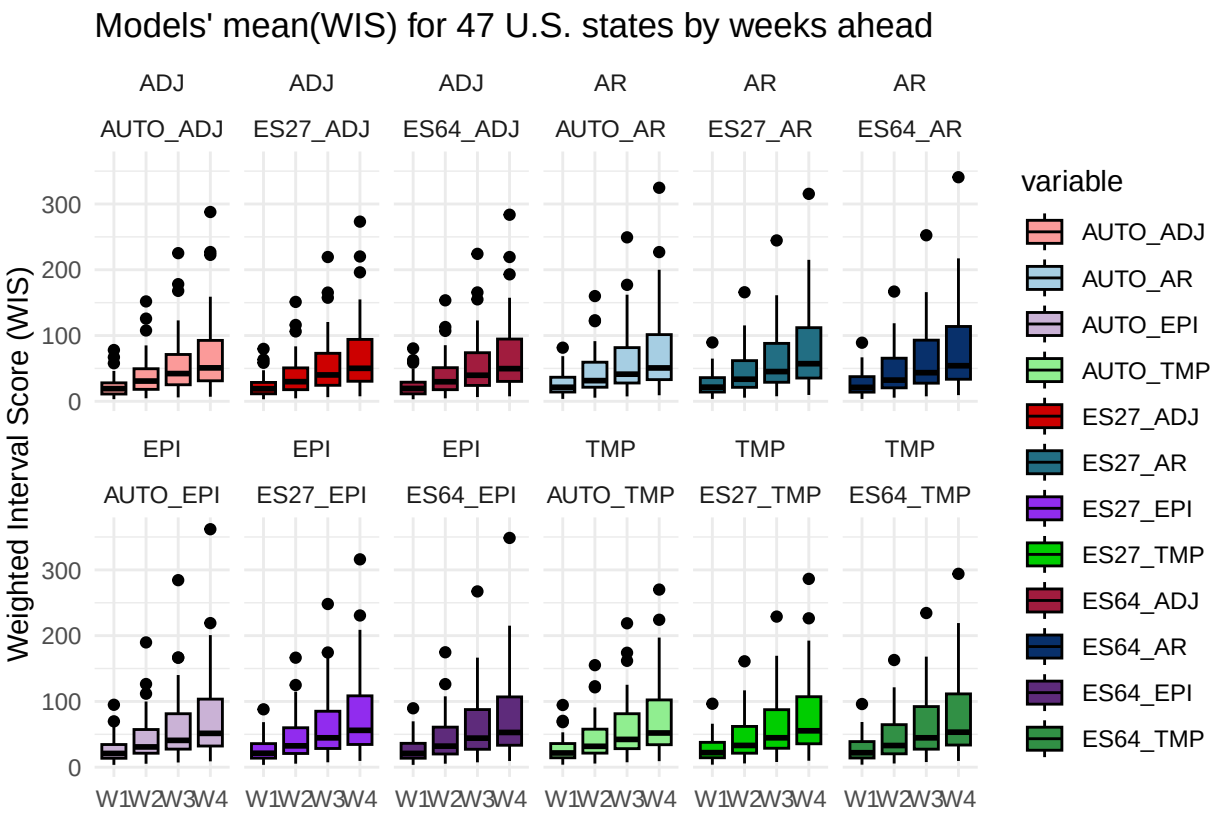
```
head(W4)
```

```
## # A tibble: 6 x 13
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     43.3     40.5     41.8     45.5    38.3     39.5     38.8     39.2
## 2 Arizona    132.     123.     130.     129.    129.     119.     121.     121.
## 3 Arkansas    50.3     40.5     48.9     52.0    64.5     44.9     61.3     60.6
## 4 Califor~   325.     288.     362.     270.    315.     273.     316.     286.
## 5 Colorado    44.1     40.2     51.5     48.2    50.6     45.5     47.8     50.7
## 6 Connect~    59.0     48.5     52.7     56.1    58.9     47.8     57.1     55.4
## # i 4 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>
```

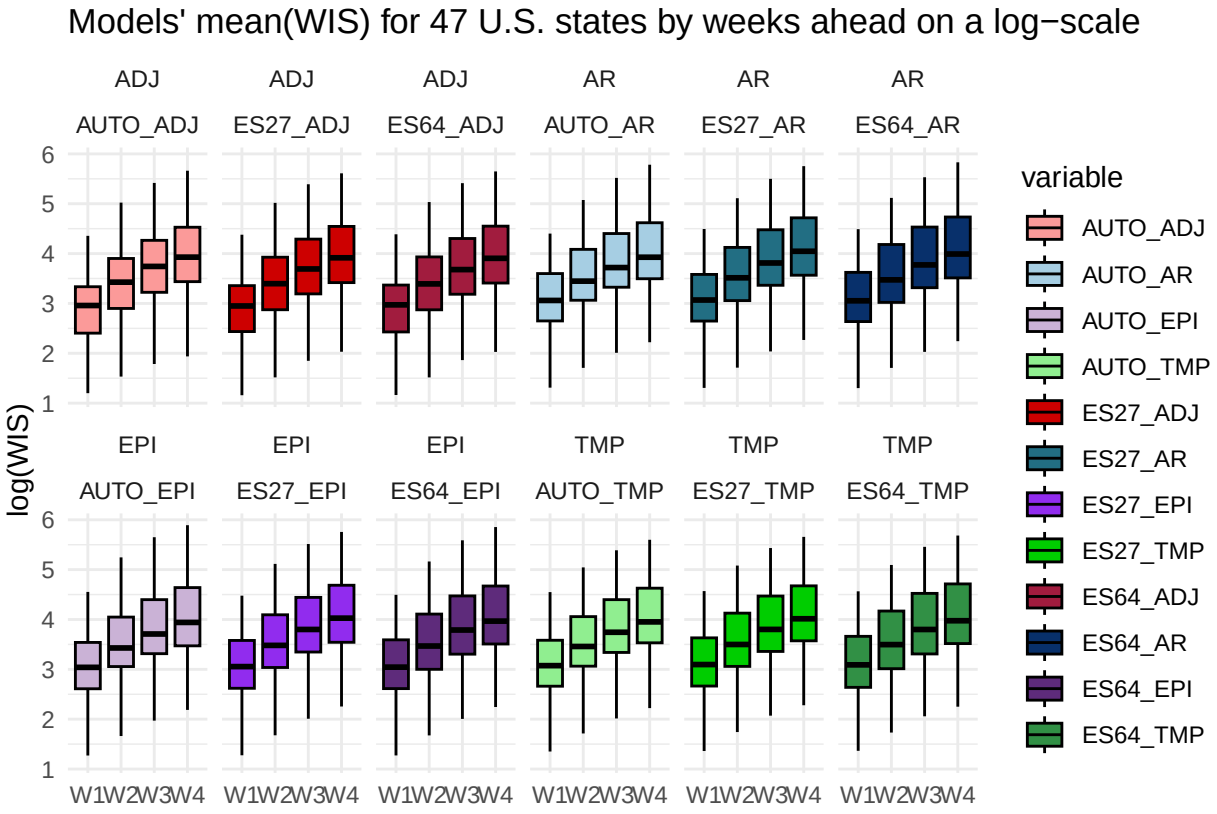
Here we have boxplots of the mean(WIS) by each state.

```
# Combine the data into one data frame
combined_data <- data.frame(
  week = rep(c("W1", "W2", "W3", "W4"), each = 47 * 12),
```

```
  variable = rep(c("AUTO_AR", "ES27_AR", "ES64_AR", "AUTO_ADJ", "ES27_ADJ", "ES64_ADJ", "AUTO_TMP", "ES

  value = c(W1$AUTO_AR, W1$ES27_AR, W1$ES64_AR, W1$AUTO_ADJ, W1$ES27_ADJ, W1$ES64_ADJ, W1$AUTO_TMP, W1$
            W2$AUTO_AR, W2$ES27_AR, W2$ES64_AR, W2$AUTO_ADJ, W2$ES27_ADJ, W2$ES64_ADJ, W2$AUTO_TMP, W2$
            W3$AUTO_AR, W3$ES27_AR, W3$ES64_AR, W3$AUTO_ADJ, W3$ES27_ADJ, W3$ES64_ADJ, W3$AUTO_TMP, W3$
            W4$AUTO_AR, W4$ES27_AR, W4$ES64_AR, W4$AUTO_ADJ, W4$ES27_ADJ, W4$ES64_ADJ, W4$AUTO_TMP, W4$
)

# Create a new column for categories
combined_data <- combined_data %>%
  mutate(category = case_when(
    grepl("AR", variable) ~ "AR",
    grepl("EPI", variable) ~ "EPI",
    grepl("TMP", variable) ~ "TMP",
    grepl("ADJ", variable) ~ "ADJ"
  ))

# Define color ramps for each category
colors_ar <- c("#a6cee3","#226e83", "#08306b")
colors_adj <- c("#fb9a99","red3", "#a11c3e")
colors_epi <- c("#cab2d6", "purple2", "#5e2b7b")
colors_tmp <- c("lightgreen","green3","#319045")

# Create a custom color mapping for each variable
custom_colors <- c(
  "AUTO_AR" = colors_ar[1], "ES27_AR" = colors_ar[2], "ES64_AR" = colors_ar[3],
  "AUTO_EPI" = colors_epi[1], "ES27_EPI" = colors_epi[2], "ES64_EPI" = colors_epi[3],
  "AUTO_TMP" = colors_tmp[1], "ES27_TMP" = colors_tmp[2], "ES64_TMP" = colors_tmp[3],
  "AUTO_ADJ" = colors_adj[1], "ES27_ADJ" = colors_adj[2], "ES64_ADJ" = colors_adj[3]
)

  # Create the box plot with additional facet for categories
  ggplot(combined_data, aes(x = week, y = value, fill = variable)) +
    geom_boxplot(color = "black") +
    scale_fill_manual(values = custom_colors) +
    facet_wrap(category ~ variable , nrow = 2) +
    labs(
      title = "Models' mean(WIS) for 47 U.S. states by weeks ahead",
      x = "",
      y = "Weighted Interval Score (WIS)"
    ) +
    theme_minimal()
```




```r
# Create the box plot with additional facet for categories
ggplot(combined_data, aes(x = week, y = log(value), fill = variable)) +
  geom_boxplot(color = "black") +
  scale_fill_manual(values = custom_colors) +
  facet_wrap(category ~ variable , nrow = 2) +
  labs(
    title = "Models' mean(WIS) for 47 U.S. states by weeks ahead on a log-scale",
    x = "",
    y = "log(WIS)"
  ) +
  theme_minimal()
```



Here I include a column with the best model result based on the lowest mean(WIS) of each state.

```
# BEST RESULT

# Extract the columns of interest
cols <- colnames(W1)[-1]
# Initialize a vector to store results
W1$Best_Result <- character(nrow(W1))

# Give me the model with lower WIS value in the best result column
for (i in 1:nrow(W1)) {
  # Find the column name with the minimum value for each row
  W1$Best_Result[i] <- cols[which.min(W1[i, cols])]
}

# REORDER BY FREQUENCY
W1$Best_Result <- fct_infreq(W1$Best_Result)

# Print merged results
head(W1)
```

```
## # A tibble: 6 x 14
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     21.5     20.3     22.4     23.9    20.8     19.1     21.3     21.8
## 2 Arizona     45.6     37.0     45.3     45.6    44.6     36.3     45.2     43.5
```

```
## 3 Arkansas    20.0     18.4     19.6     21.5    21.6     18.4     21.2     22.1
## 4 Califor~    81.5     77.9     94.9     94.6    89.5     79.7     88.0     96.5
## 5 Colorado    16.8     14.4     17.2     18.1    17.3     15.1     16.8     18.2
## 6 Connect~    19.2     16.8     18.7     19.0    19.8     16.9     19.1     19.2
## # i 5 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>, Best_Result <fct>
```

```
################################################

# Extract the columns of interest
cols <- colnames(W2)[-1]
# Initialize a vector to store results
W2$Best_Result <- character(nrow(W2))
# Give me the model with lower WIS value in the best result column
for (i in 1:nrow(W2)) {
  # Find the column name with the minimum value for each row
  W2$Best_Result[i] <- cols[which.min(W2[i, cols])]
}

# REORDER BY FREQUENCY
W2$Best_Result <- fct_infreq(W2$Best_Result)

# Print merged results
head(W2)
```

```
## # A tibble: 6 x 14
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     31.1     29.3     30.8     30.7    28.6     27.4     28.8     29.1
## 2 Arizona     78.1     66.1     76.7     76.9    72.6     64.7     72.8     69.4
## 3 Arkansas    30.4     27.2     30.6     32.4    35.6     28.6     34.8     35.2
## 4 Califor~   160.     152.     190.     155.    166.     151.     167.     161.
## 5 Colorado    26.7     23.2     28.6     28.4    28.4     24.7     27.2     29.1
## 6 Connect~    31.4     26.1     29.2     30.2    31.9     26.1     30.3     30.0
## # i 5 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>, Best_Result <fct>
```

```
#####################################
# BEST RESULT
# Extract the columns of interest
cols <- colnames(W3)[-1]
# Initialize a vector to store results
W3$Best_Result <- character(nrow(W3))

# Give me the model with lower WIS value in the best result column
for (i in 1:nrow(W3)) {
  # Find the column name with the minimum value for each row
  W3$Best_Result[i] <- cols[which.min(W3[i, cols])]
}

# REORDER BY FREQUENCY
W3$Best_Result <- fct_infreq(W3$Best_Result)
```

```r
# Print merged results
head(W3)
```

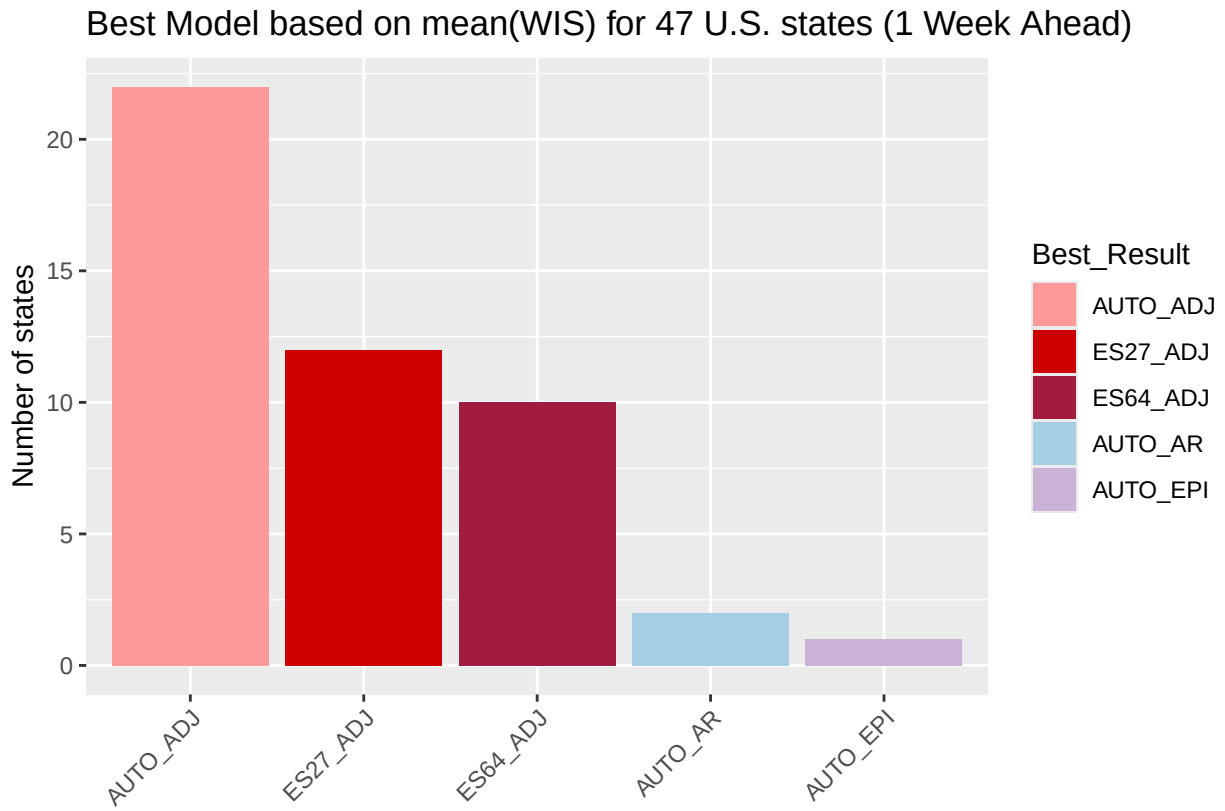
```
## # A tibble: 6 x 14
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     38.0     34.0     36.9     39.2    33.3     33.0     33.5     34.1
## 2 Arizona    106.      94.6    104.     104.    100.      91.0     95.8     96.2
## 3 Arkansas    40.2     33.8     38.9     42.0    48.4     36.5     46.5     46.6
## 4 Califor~   249.     225.     284.     219.    245.     219.     248.     229.
## 5 Colorado    35.7     32.6     40.3     38.8    39.9     35.4     37.7     40.4
## 6 Connect~    46.0     38.2     40.8     43.9    45.9     37.7     44.3     42.9
## # i 5 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>, Best_Result <fct>
```

```r
####################################
# BEST RESULT
# Extract the columns of interest
cols <- colnames(W4)[-1]
# Initialize a vector to store results
W4$Best_Result <- character(nrow(W4))
# Give me the model with lower WIS value in the best result column
for (i in 1:nrow(W4)) {
  # Find the column name with the minimum value for each row
  W4$Best_Result[i] <- cols[which.min(W4[i, cols])]
}
W4$Best_Result <- fct_infreq(W4$Best_Result)

# Print merged results
head(W4)
```

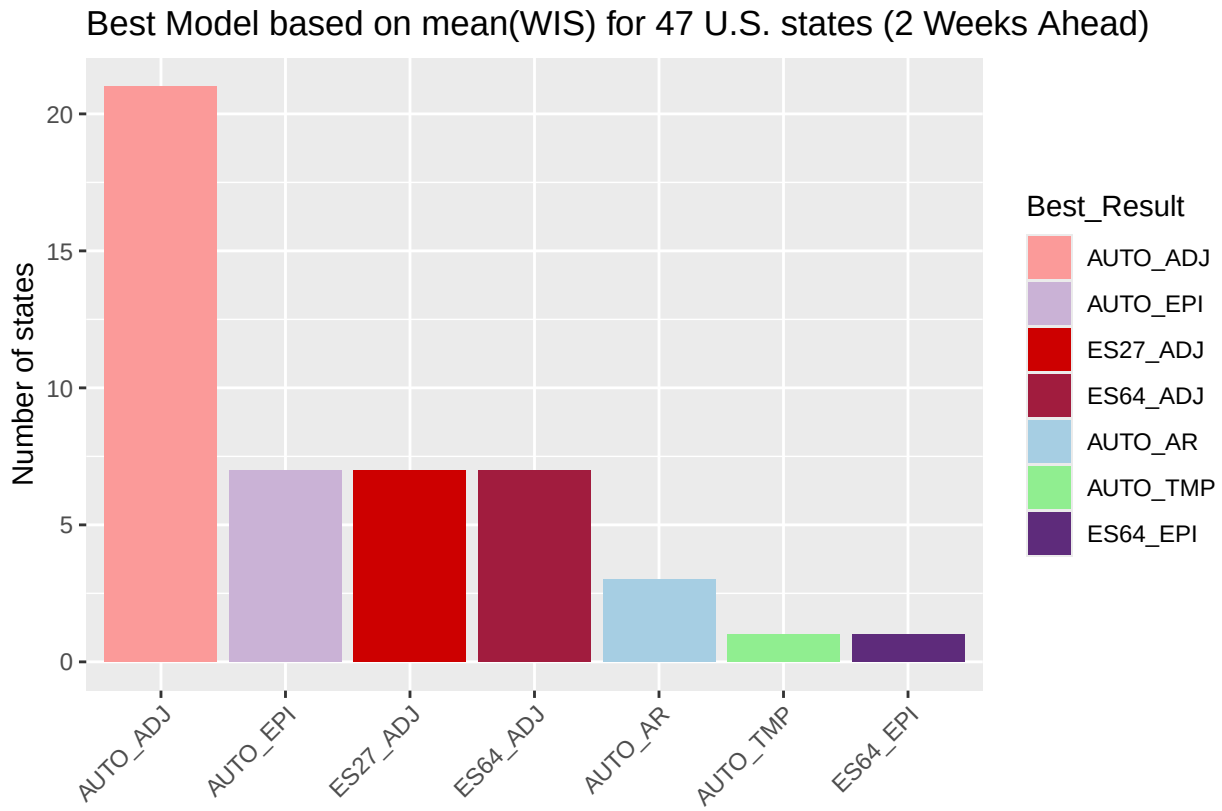
```
## # A tibble: 6 x 14
##   STATE    AUTO_AR AUTO_ADJ AUTO_EPI AUTO_TMP ES27_AR ES27_ADJ ES27_EPI ES27_TMP
##   <chr>      <dbl>    <dbl>    <dbl>    <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 Alabama     43.3     40.5     41.8     45.5    38.3     39.5     38.8     39.2
## 2 Arizona    132.     123.     130.     129.    129.     119.     121.     121.
## 3 Arkansas    50.3     40.5     48.9     52.0    64.5     44.9     61.3     60.6
## 4 Califor~   325.     288.     362.     270.    315.     273.     316.     286.
## 5 Colorado    44.1     40.2     51.5     48.2    50.6     45.5     47.8     50.7
## 6 Connect~    59.0     48.5     52.7     56.1    58.9     47.8     57.1     55.4
## # i 5 more variables: ES64_AR <dbl>, ES64_ADJ <dbl>, ES64_EPI <dbl>,
## #   ES64_TMP <dbl>, Best_Result <fct>
```

Now, let's plot the best models by each state for each forecast horizon (1-4 weeks ahead).

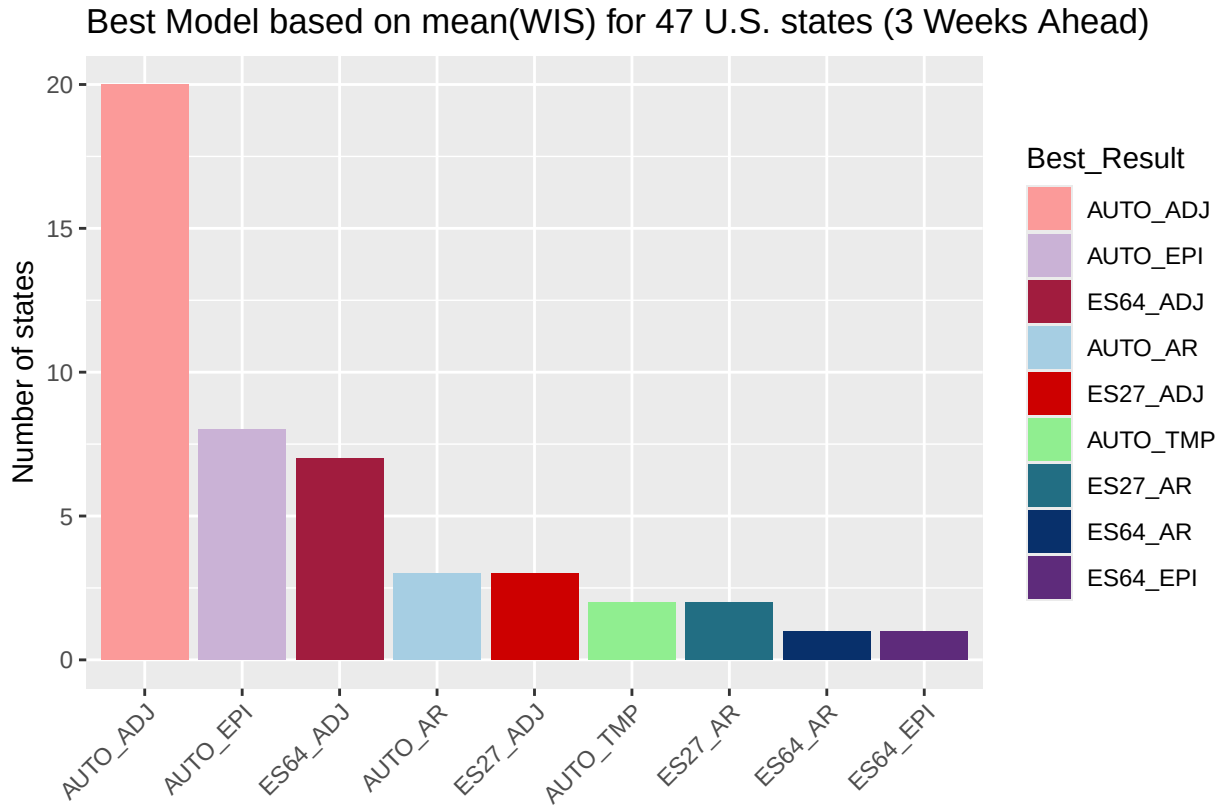
```r
# --------- WEEK1 MODELS ------------- #
ggplot(W1, aes(x = Best_Result, fill = Best_Result)) +
  geom_bar() +
  scale_fill_manual(values = custom_colors) +
  labs(title = "Best Model based on mean(WIS) for 47 U.S. states (1 Week Ahead)",
       x = "", y = "Number of states") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



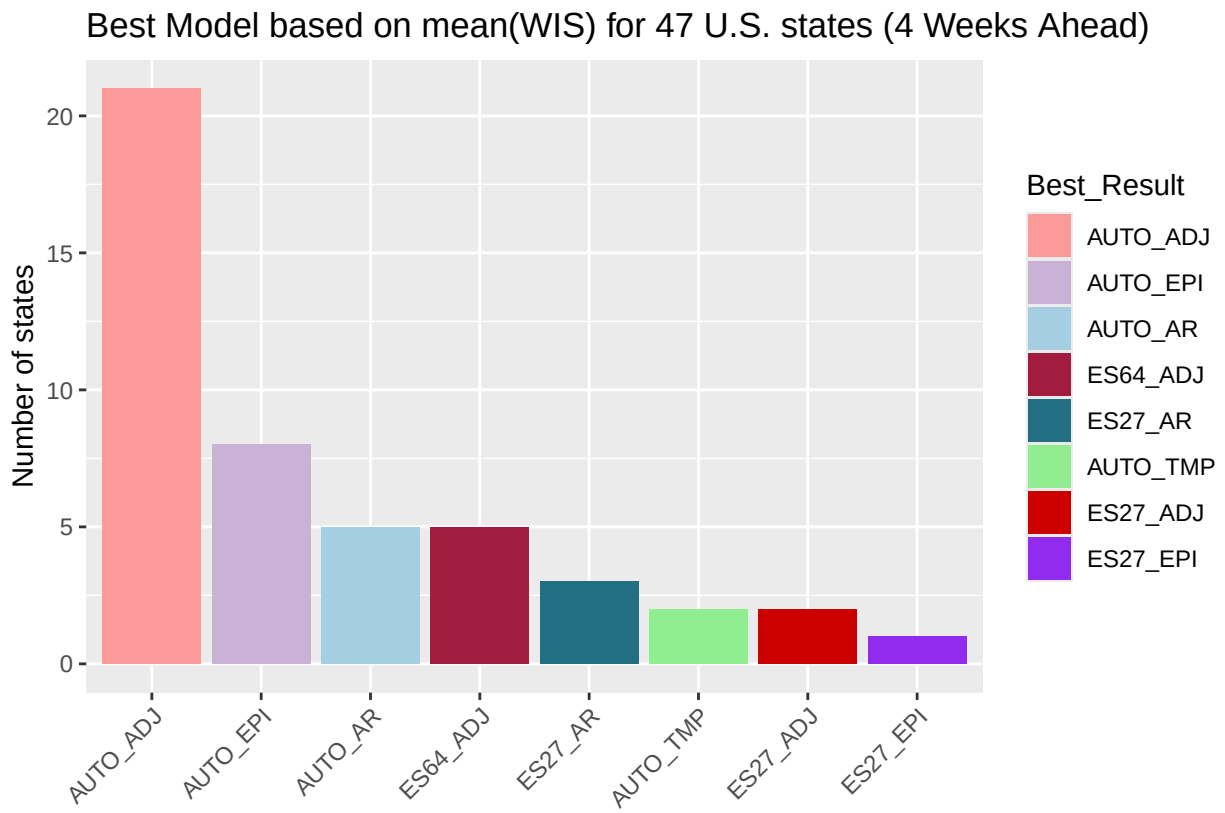

```
# --------- WEEK2 MODELS ------------- #
ggplot(W2, aes(x = Best_Result, fill = Best_Result)) +
  geom_bar() +
  scale_fill_manual(values = custom_colors) +
  labs(title = "Best Model based on mean(WIS) for 47 U.S. states (2 Weeks Ahead)",
       x = "", y = "Number of states") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# --------- WEEK3 MODELS ------------- #
ggplot(W3, aes(x = Best_Result, fill = Best_Result)) +
  geom_bar() +
  scale_fill_manual(values = custom_colors) +
  labs(title = "Best Model based on mean(WIS) for 47 U.S. states (3 Weeks Ahead)",
       x = "", y = "Number of states") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
# --------- WEEK4 MODELS ------------- #
ggplot(W4, aes(x = Best_Result, fill = Best_Result)) +
  geom_bar() +
  scale_fill_manual(values = custom_colors) +
  labs(title = "Best Model based on mean(WIS) for 47 U.S. states (4 Weeks Ahead)",
       x = "", y = "Number of states") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



This plot combines the 4 forecast horizons and the counts of best models in a single plot.

```
models <- readr::read_csv("models.csv") %>%
  mutate(ex.var = factor(ex.var,
                         levels = c("none",
                                    "mean of ILI in adjacent states in previous week",
                                    "temperature",
                                    "mean of ILI in epiweek - 52 and ILI in epiweek - 104")
                         )
         )
```

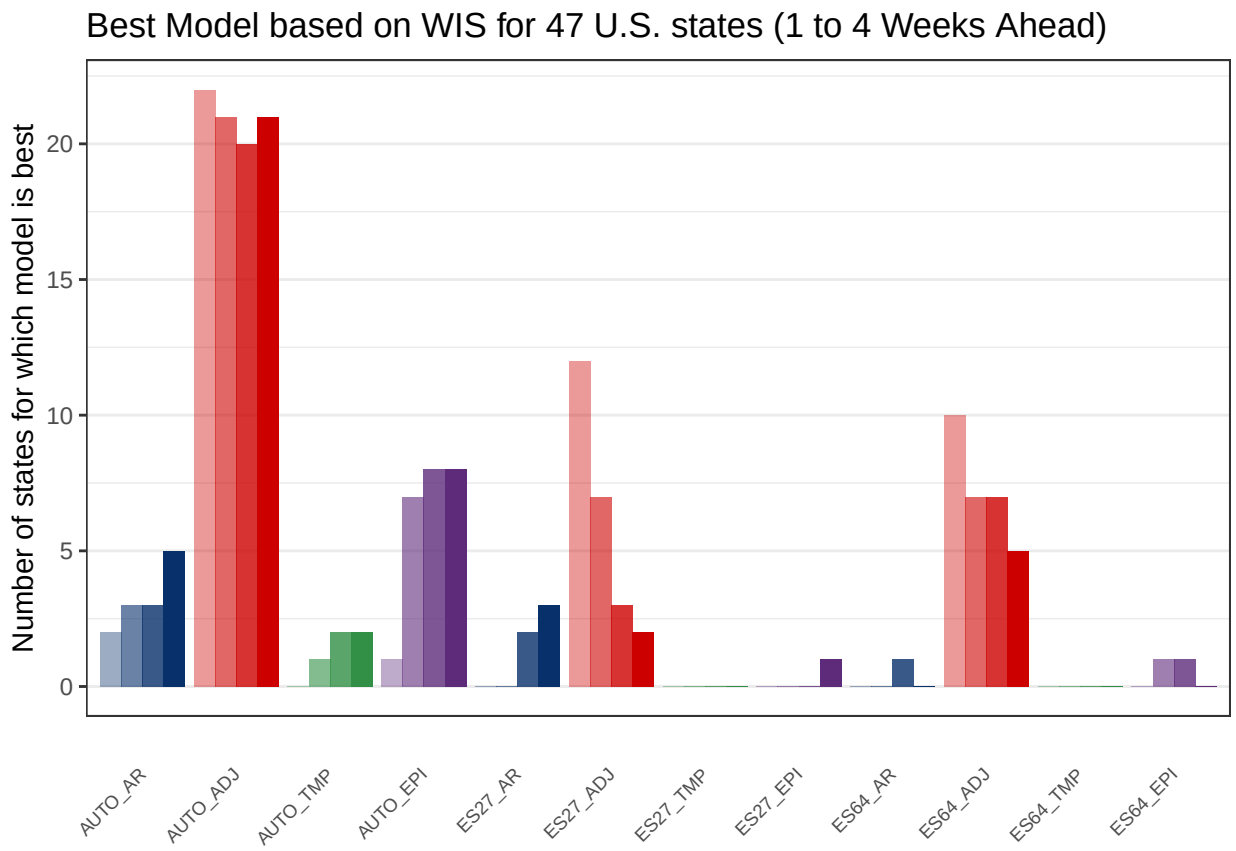
```
## Rows: 12 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): model, model.type, ex.var, ensemble.approach
## dbl (1): ensemble.size
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
thegrid <- expand_grid(model = models$model, target=paste0("W",1:4))

BestResults <- W1 %>% select(STATE, W1 = Best_Result) %>%
  full_join(W2 %>% select(STATE, W2 = Best_Result)) %>%
  full_join(W3 %>% select(STATE, W3 = Best_Result)) %>%
  full_join(W4 %>% select(STATE, W4 = Best_Result)) %>%
```

```
  pivot_longer(-STATE, names_to = "target", values_to = "Best_Result") %>%
  group_by(target, Best_Result) %>%
  summarise(n_best = n()) %>%
  rename(model = Best_Result) %>%
  ungroup() %>%
  # complete(target, model) %>%
  full_join(thegrid, by = c("target", "model")) %>%
  full_join(models) %>%
  replace_na(list(n_best = 0))
```

```
## Joining with `by = join_by(STATE)`
## Joining with `by = join_by(STATE)`
## Joining with `by = join_by(STATE)`
## `summarise()` has grouped output by 'target'. You can override using the
## `.groups` argument.
## Joining with `by = join_by(model)`
```

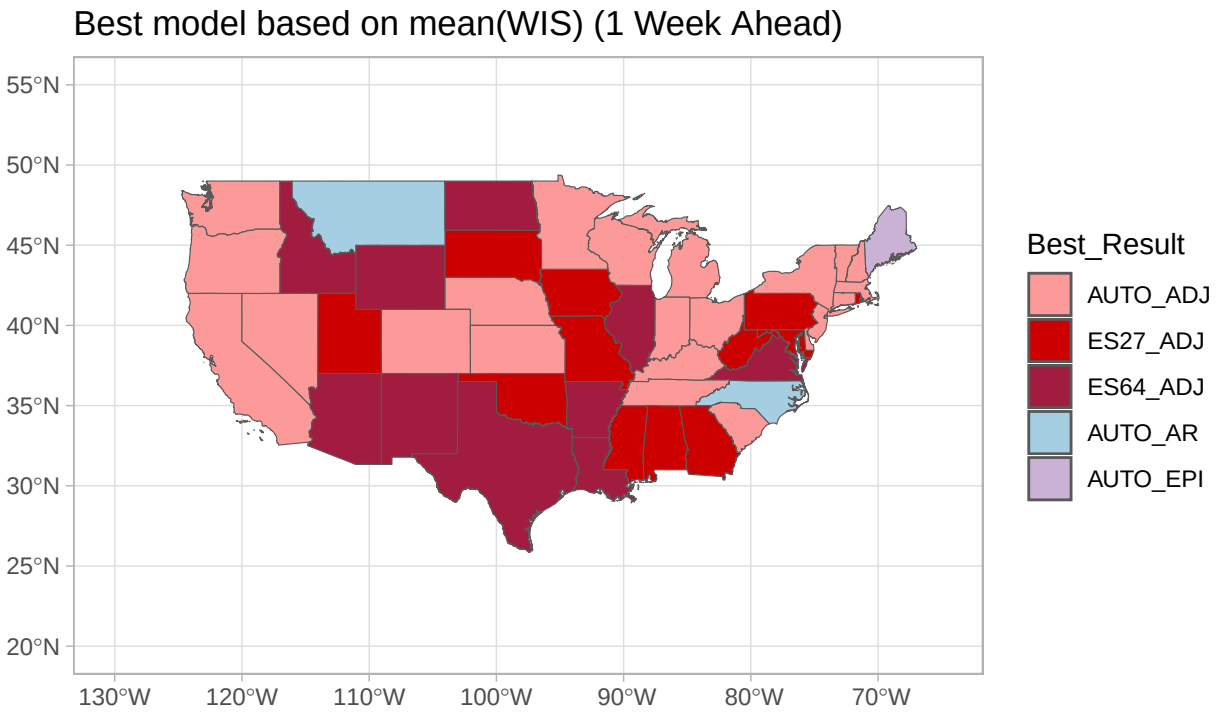
```
g <- BestResults %>%
  arrange(ex.var) %>%
  ggplot(aes(x=factor(model,
                      levels = models %>% arrange(ensemble.size, ex.var) %>% pull(model)),
             y=n_best,
             group=target,
             fill=ex.var,
             alpha=target)) +
  geom_bar(
           position="dodge",
           stat='identity') +

  labs(title = "Best Model based on WIS for 47 U.S. states (1 to 4 Weeks Ahead)",
       x = "", y="Number of states for which model is best") +
  theme_bw() +
  scale_alpha_manual(values=c(.4,.6,.8,1), na.translate = FALSE) +
  scale_fill_manual(values=c("#08306b","red3","#319045","#5e2b7b"),
                    na.translate = FALSE) +
  theme(panel.grid.major.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_text(angle = 45, size = 7, vjust = 0.5, hjust=1),
        strip.placement = "outside",
        strip.background = element_rect(fill=NA,colour="grey50"),
        panel.spacing=unit(0,"cm"),
        legend.position="none")  # This line removes the legend
g
```



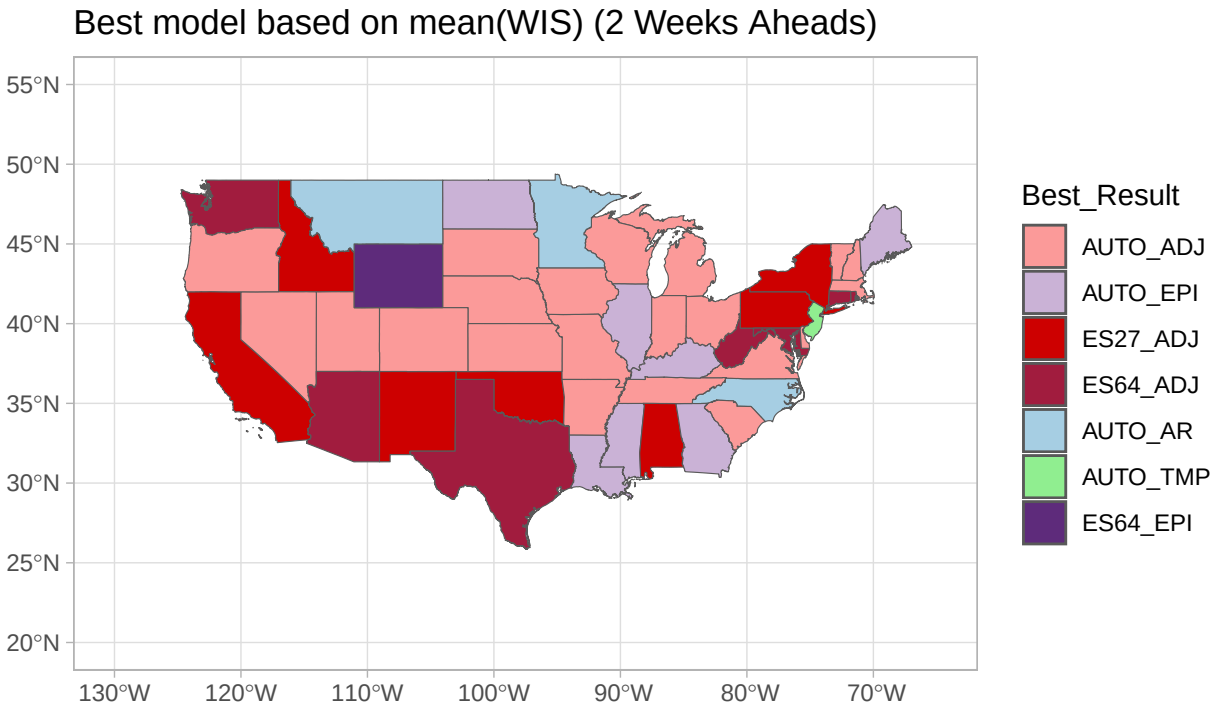

```
#g %>% plotly::ggplotly()
```

Now let's plot maps with the best models we found for each state based on the best mean(WIS).

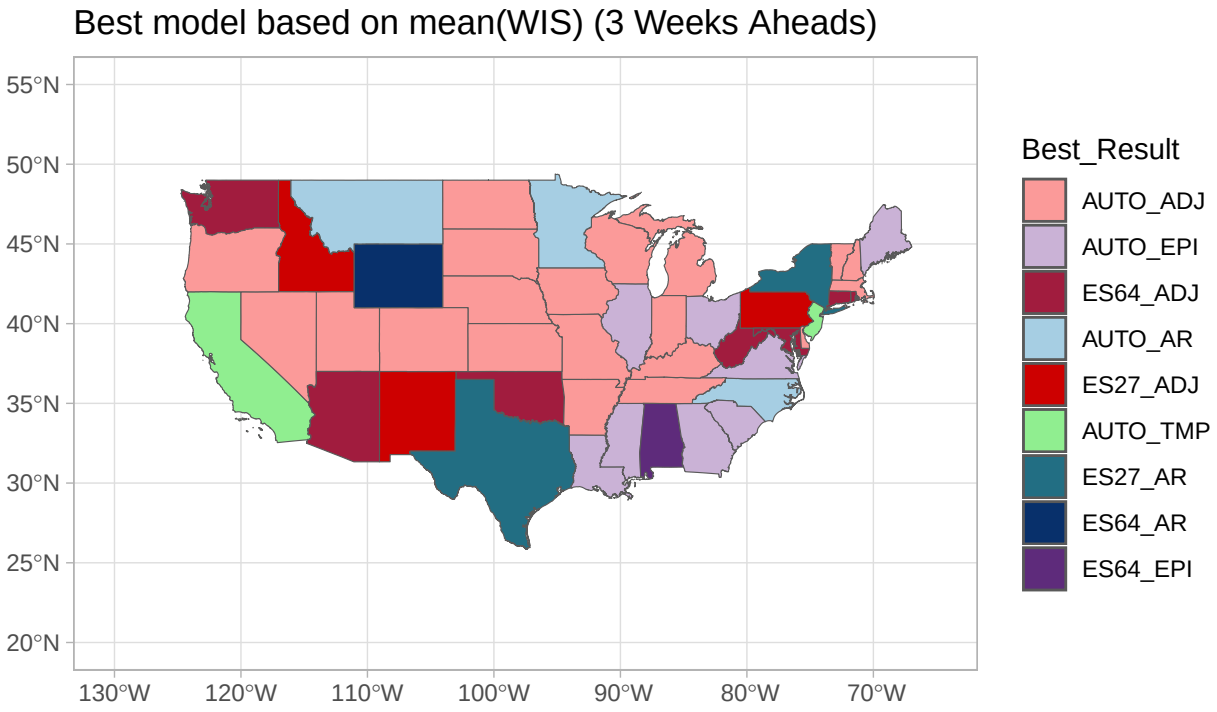
```
#################################################
# ES27 ARIMAX by ADJACENT STATES - 1 WEEK AHEAD #
#################################################

category_colors <- c(
  "AUTO_AR" = "#a6cee3", "ES27_AR" = "#226e83", "ES64_AR" = "#08306b",
  "AUTO_ADJ" = "#fb9a99", "ES27_ADJ" = "red3", "ES64_ADJ" = "#a11c3e",
  "AUTO_TMP" = "lightgreen", "ES27_TMP" = "green3", "ES64_TMP" = "#319045",
  "AUTO_EPI" = "#cab2d6", "ES27_EPI" = "purple2", "ES64_EPI" = "#5e2b7b"
)

map_week1<-left_join(states, W1, by=join_by("STATE"))%>%
  drop_na()

ES_1WEEK<- ggplot(map_week1, fill ="lightgrey") +  theme_light()  + geom_sf(aes(fill=Best_Result)) +  s

x_limits <- c(-130, -65)  # Set the desired longitude range
y_limits <- c(20, 55)     # Set the desired latitude range

ES_1WEEK + coord_sf(xlim = x_limits, ylim = y_limits)
```



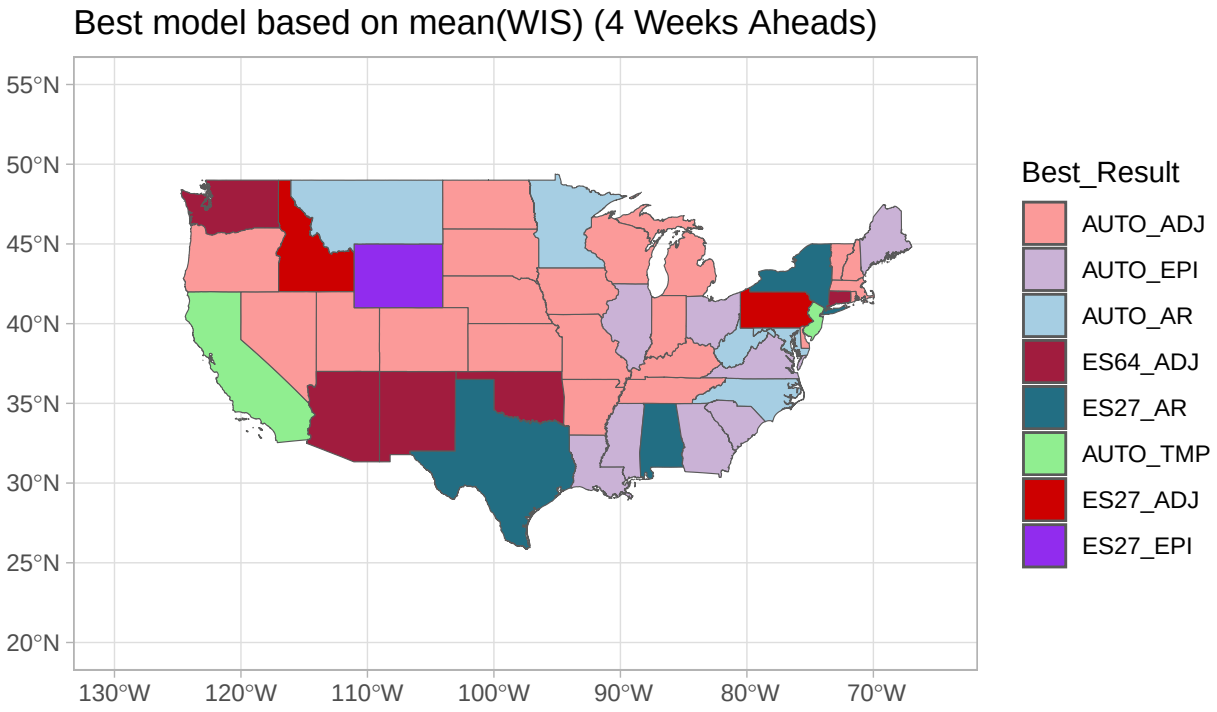

```
#####################################################
# ES27 ARIMAX by ADJACENT STATES - 2 WEEKS AHEAD #
#####################################################

map_week2<-left_join(states, W2, by=join_by("STATE"))%>%
  drop_na()

MAP_WEEK2<- ggplot(map_week2, fill ="lightgrey") +  theme_light()  + geom_sf(aes(fill=Best_Result)) +

x_limits <- c(-130, -65)  # Set the desired longitude range
y_limits <- c(20, 55)    # Set the desired latitude range

MAP_WEEK2 + coord_sf(xlim = x_limits, ylim = y_limits)
```




```
#####################################################
# ES27 ARIMAX by ADJACENT STATES - 3 WEEKS AHEAD #
#####################################################

map_week3<-left_join(states, W3, by=join_by("STATE"))%>%
  drop_na()

MAP_WEEK3<- ggplot(map_week3, fill ="lightgrey") +  theme_light()  + geom_sf(aes(fill=Best_Result)) +

x_limits <- c(-130, -65)  # Set the desired longitude range
y_limits <- c(20, 55)     # Set the desired latitude range

MAP_WEEK3 + coord_sf(xlim = x_limits, ylim = y_limits)
```



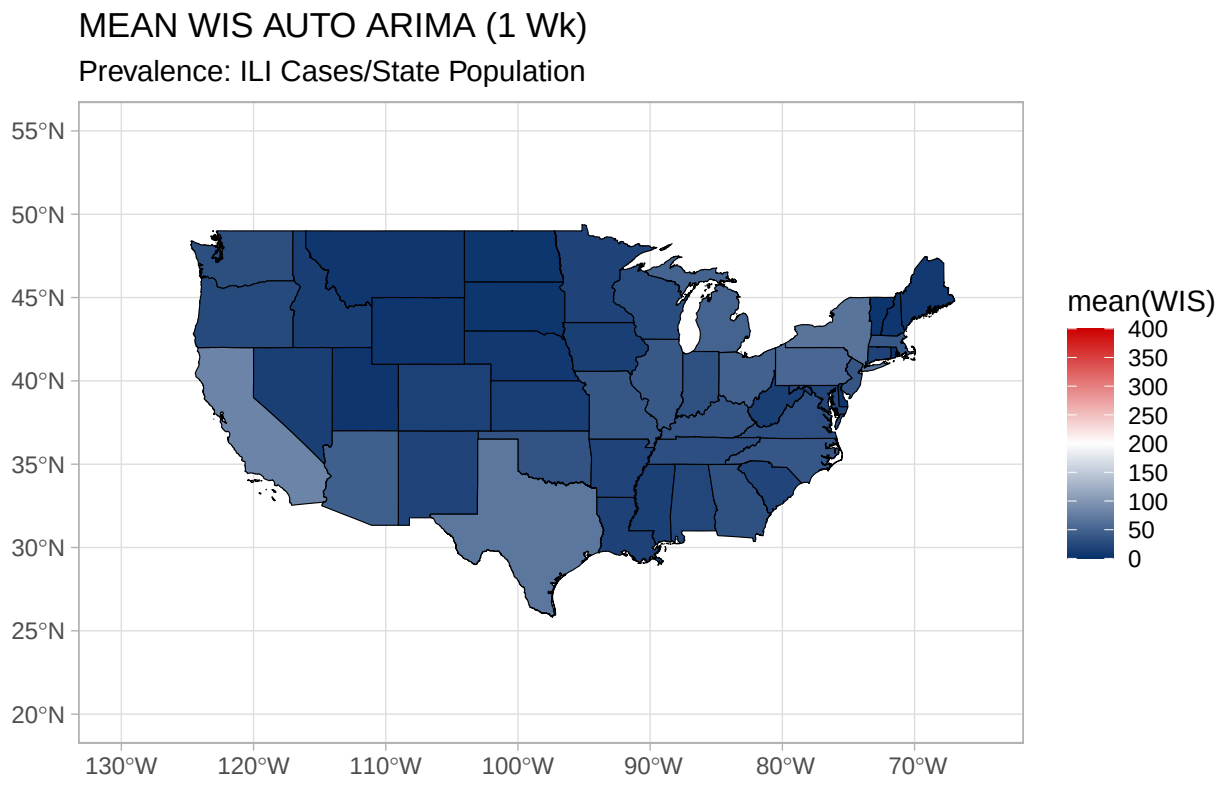

```
#####################################################
# ES27 ARIMAX by ADJACENT STATES - 4 WEEKS AHEAD #
#####################################################

map_week4<-left_join(states, W4, by=join_by("STATE"))%>%
  drop_na()

MAP_WEEK4<- ggplot(map_week4, fill ="lightgrey") +  theme_light()  + geom_sf(aes(fill=Best_Result)) +  s

x_limits <- c(-130, -65)  # Set the desired longitude range
y_limits <- c(20, 55)     # Set the desired latitude range

MAP_WEEK4 + coord_sf(xlim = x_limits, ylim = y_limits)
```



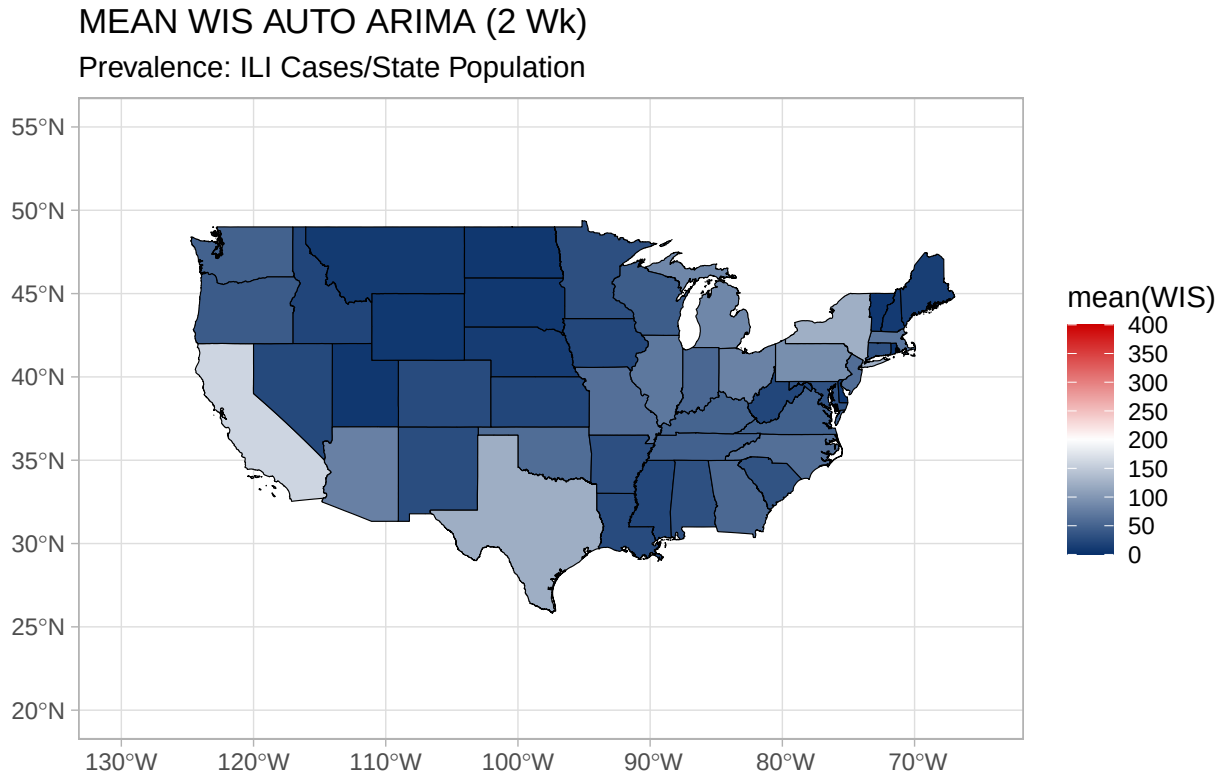
Now, let's map the mean(WIS) for the AUTO ARIMA model as an example.

```
# Calculate breaks dynamically or use a fixed range
data_range <- c(0,400)#
breaks <- pretty(data_range, n = 6)  # breaks
labels <- breaks  #

# Define color palette
colors <- colorRampPalette(c("#08306b", "white", "red3"))(100)

MAP_WEEK1 <- ggplot(map_week1) +
    theme_light() +
    geom_sf(aes(fill = (AUTO_AR)), color = "black") +
    scale_fill_gradientn(
        "mean(WIS)",
        colors = colors,
        breaks = breaks,
        labels = labels,
        limits = data_range,
        na.value = "lightgrey"
    ) +
    ggtitle("MEAN WIS AUTO ARIMA (1 Wk)") +
    labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
    coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
    theme(legend.position = "right")
```

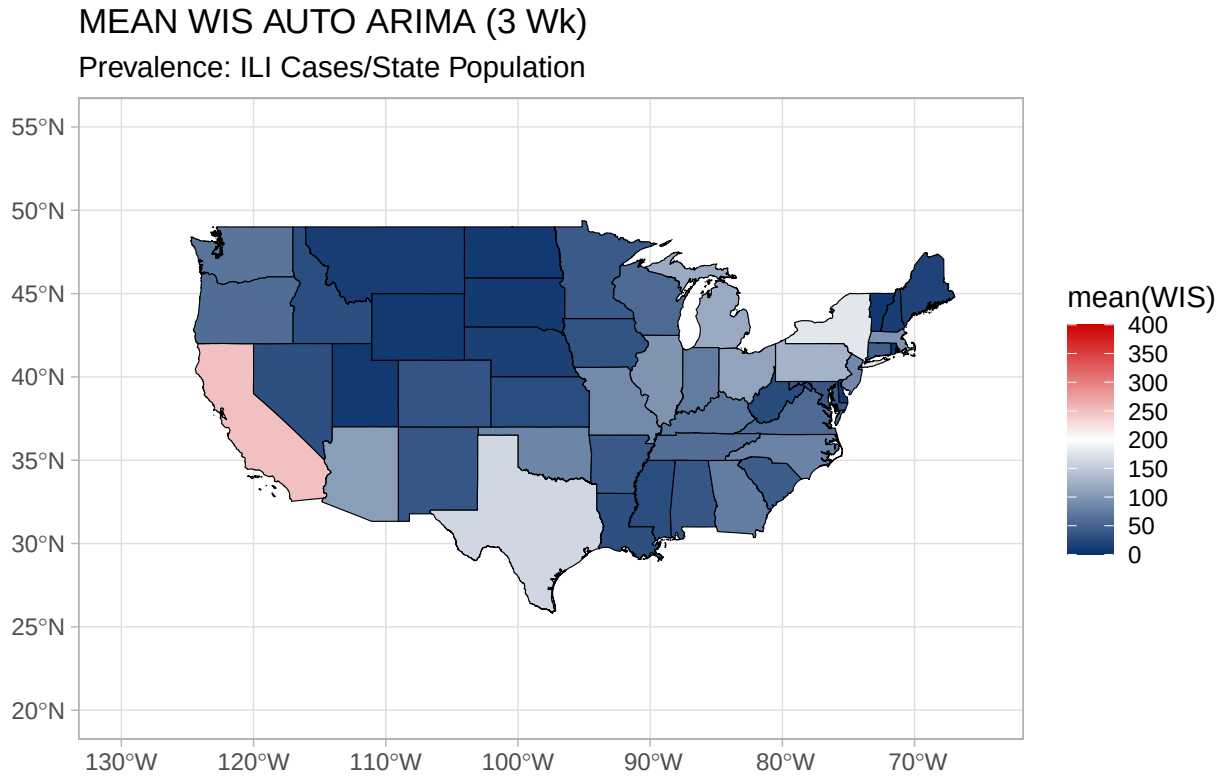
```
# Print the map
print(MAP_WEEK1)
```



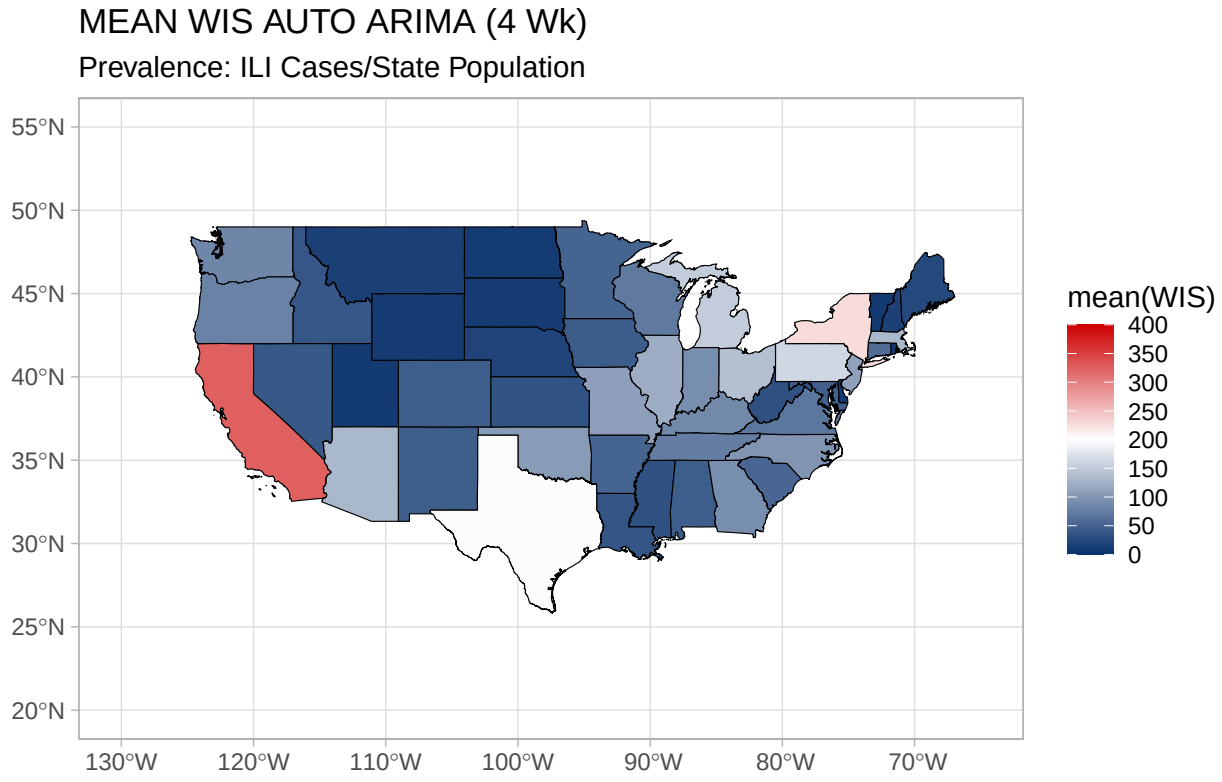

```
# Create the map
MAP_WEEK2 <- ggplot(map_week2) +
  theme_light() +
  geom_sf(aes(fill = AUTO_AR), color = "black") +
  scale_fill_gradientn(
    "mean(WIS)",
    colors = colors,
    breaks = breaks,
    labels = labels,
    limits = data_range,
    na.value = "lightgrey"
  ) +
    ggtitle("MEAN WIS AUTO ARIMA (2 Wk)") +
    labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
  coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
  theme(legend.position = "right")

# Print the map
print(MAP_WEEK2)
```




```
################################

MAP_WEEK3 <- ggplot(map_week3) +
  theme_light() +
  geom_sf(aes(fill = AUTO_AR), color = "black") +
  scale_fill_gradientn(
    "mean(WIS)",
    colors = colors,
    breaks = breaks,
    labels = labels,
    limits = data_range,
    na.value = "lightgrey"
  ) +
    ggtitle("MEAN WIS AUTO ARIMA (3 Wk)") +
  labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
  coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
  theme(legend.position = "right")

# Print the map
print(MAP_WEEK3)
```



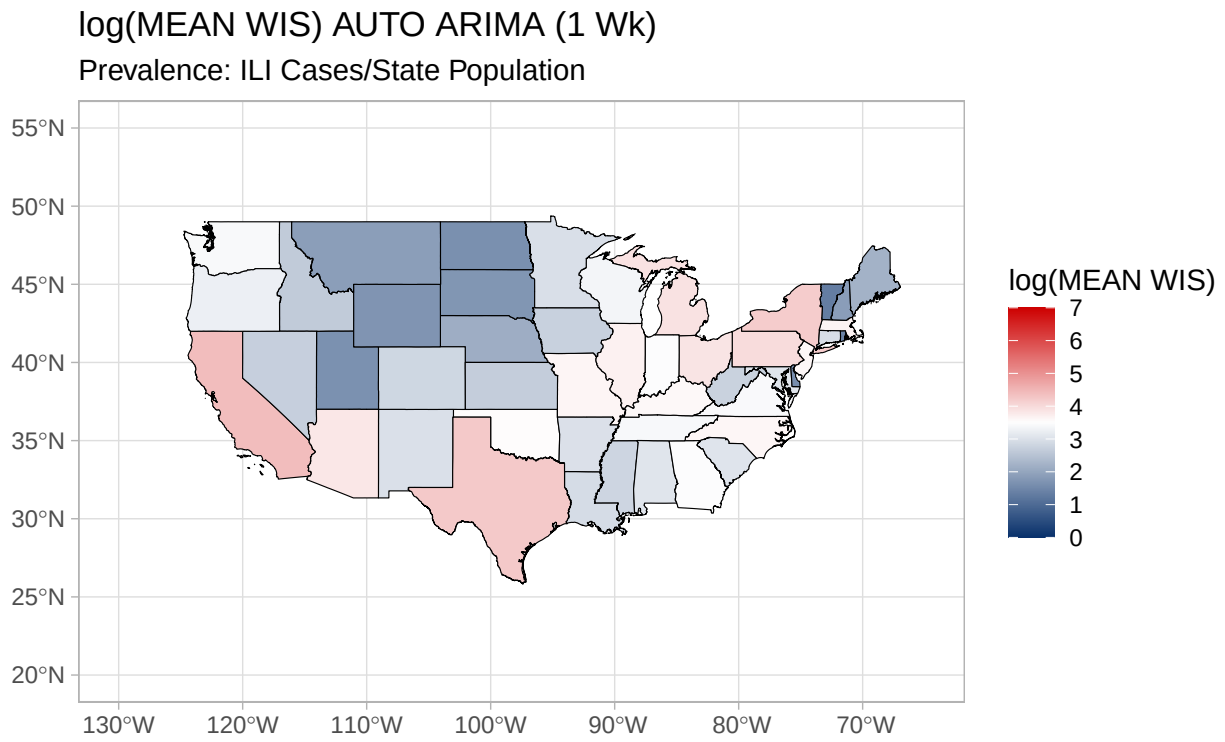

```
#######################
MAP_WEEK4 <- ggplot(map_week4) +
  theme_light() +
  geom_sf(aes(fill = AUTO_AR), color = "black") +
  scale_fill_gradientn(
    "mean(WIS)",
    colors = colors,
    breaks = breaks,
    labels = labels,
    limits = data_range,
    na.value = "lightgrey"
  ) +
  ggtitle("MEAN WIS AUTO ARIMA (4 Wk)") +
  labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
  coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
  theme(legend.position = "right")

# Print the map
print(MAP_WEEK4)
```



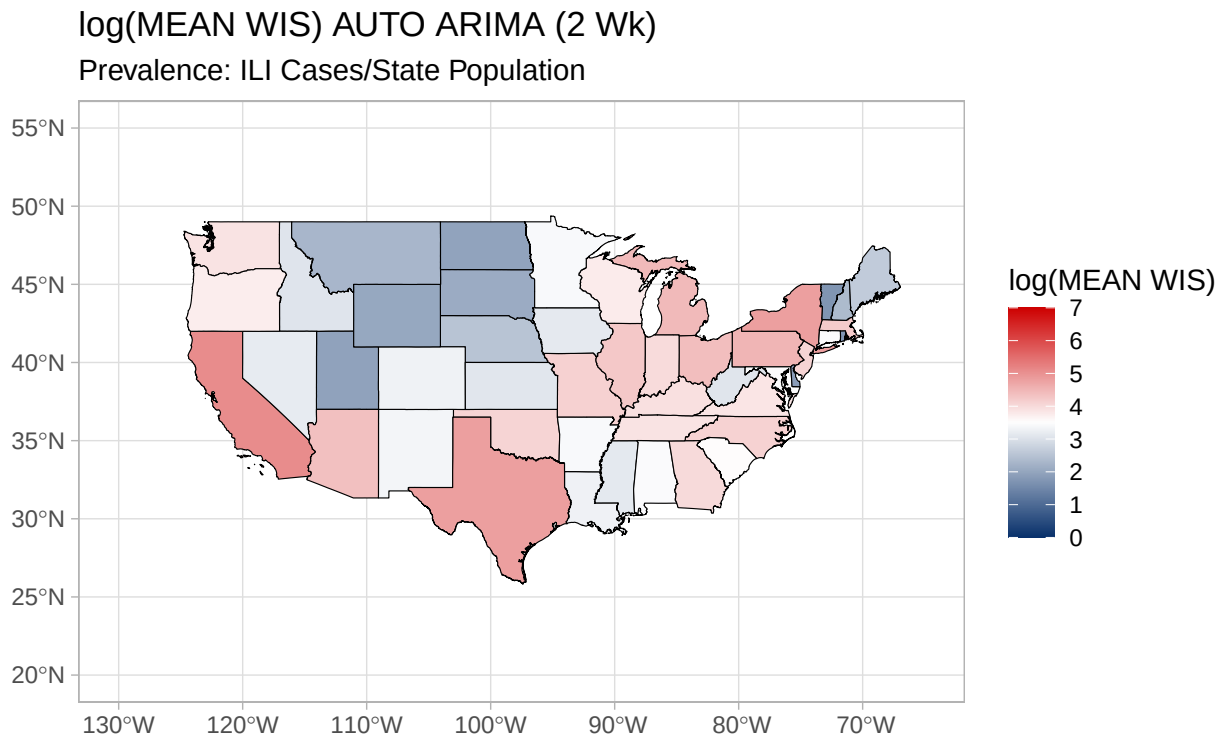
Let's plot the same map on a log scale for better visualization.

```
# Calculate breaks dynamically or use a fixed range
data_range <- c(0,7)#
breaks <- pretty(data_range, n = 6)  # breaks
labels <- breaks  #

# Define color palette
colors <- colorRampPalette(c("#08306b", "white", "red3"))(100)

MAP_WEEK1 <- ggplot(map_week1) +
    theme_light() +
    geom_sf(aes(fill =log(AUTO_AR)), color = "black") +
    scale_fill_gradientn(
        "log(MEAN WIS)",
        colors = colors,
        breaks = breaks,
        labels = labels,
        limits = data_range,
        na.value = "lightgrey"
    ) +
    ggtitle("log(MEAN WIS) AUTO ARIMA (1 Wk)") +
    labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
    coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
    theme(legend.position = "right")
```

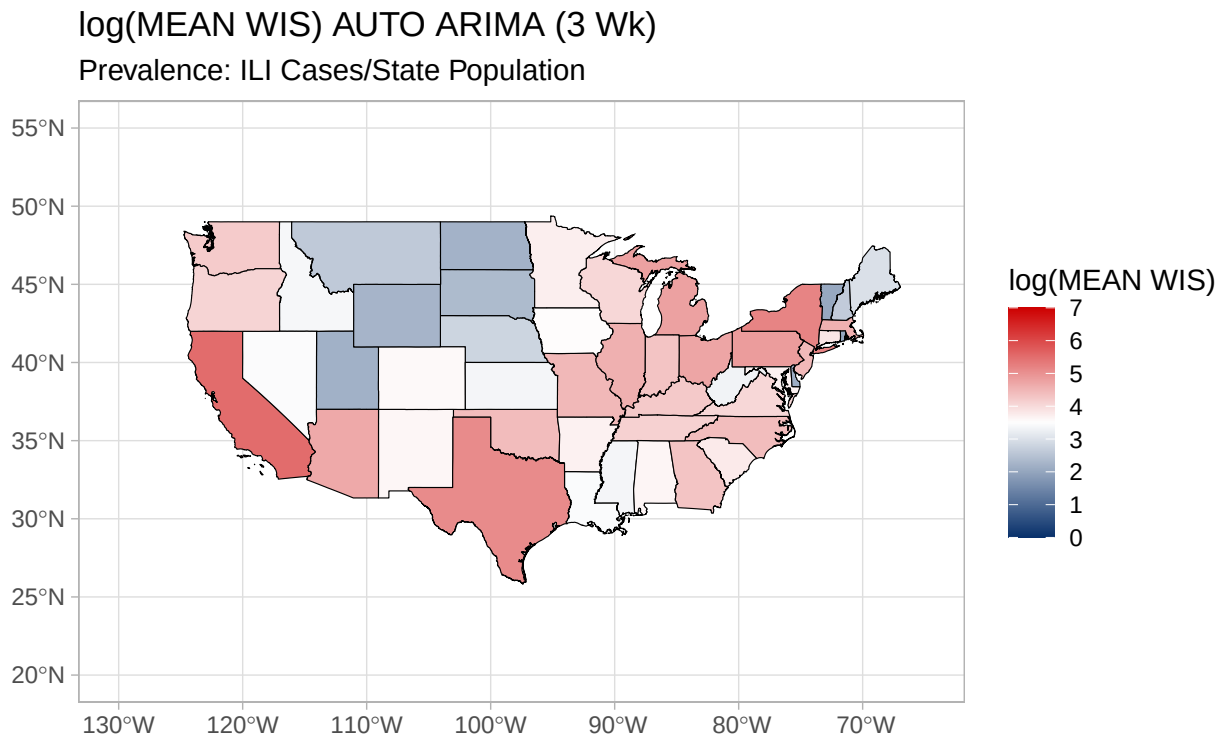
```
# Print the map
print(MAP_WEEK1)
```



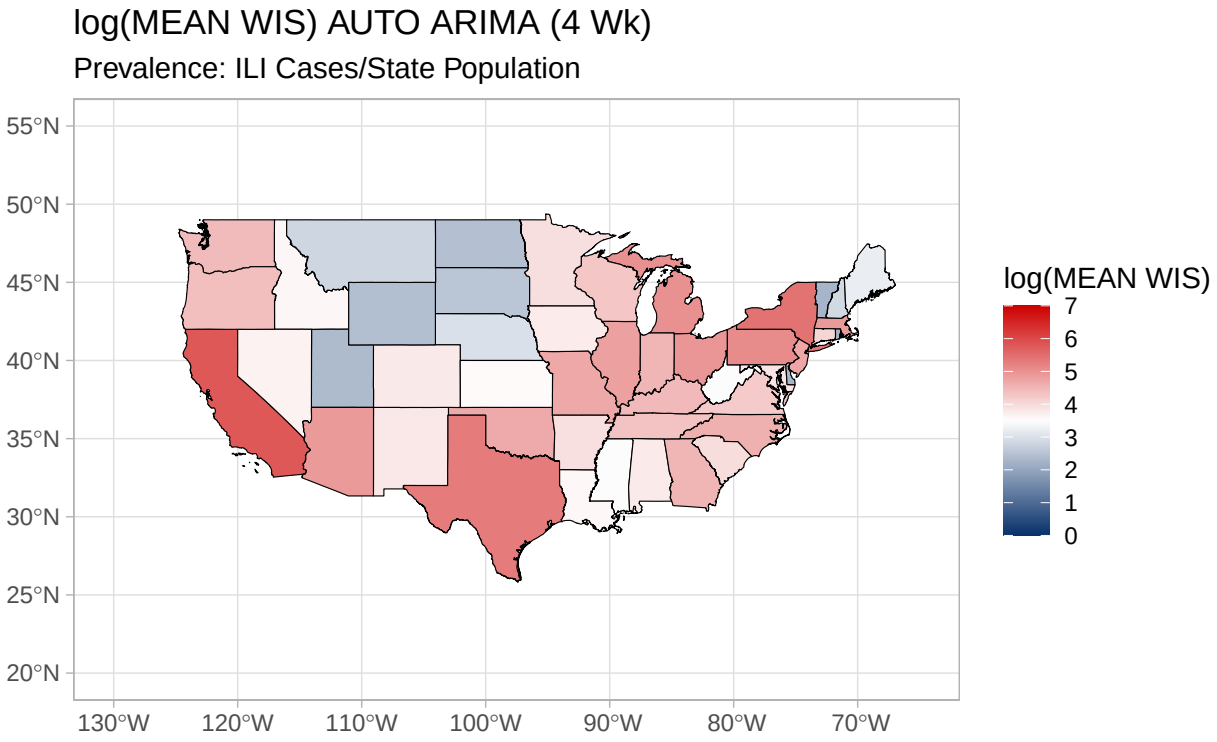

```
# Create the map
MAP_WEEK2 <- ggplot(map_week2) +
  theme_light() +
  geom_sf(aes(fill = log(AUTO_AR)), color = "black") +
  scale_fill_gradientn(
    "log(MEAN WIS)",
    colors = colors,
    breaks = breaks,
    labels = labels,
    limits = data_range,
    na.value = "lightgrey"
  ) +
    ggtitle("log(MEAN WIS) AUTO ARIMA (2 Wk)") +
    labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
  coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
  theme(legend.position = "right")

# Print the map
print(MAP_WEEK2)
```




```
################################

MAP_WEEK3 <- ggplot(map_week3) +
  theme_light() +
  geom_sf(aes(fill = log(AUTO_AR)), color = "black") +
  scale_fill_gradientn(
    "log(MEAN WIS)",
    colors = colors,
    breaks = breaks,
    labels = labels,
    limits = data_range,
    na.value = "lightgrey"
  ) +
    ggtitle("log(MEAN WIS) AUTO ARIMA (3 Wk)") +
  labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
  coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
  theme(legend.position = "right")

# Print the map
print(MAP_WEEK3)
```




```
#######################
MAP_WEEK4 <- ggplot(map_week4) +
  theme_light() +
  geom_sf(aes(fill = log(AUTO_AR)), color = "black") +
  scale_fill_gradientn(
    "log(MEAN WIS)",
    colors = colors,
    breaks = breaks,
    labels = labels,
    limits = data_range,
    na.value = "lightgrey"
  ) +
  ggtitle("log(MEAN WIS) AUTO ARIMA (4 Wk)") +
  labs(subtitle = "Prevalence: ILI Cases/State Population") +  # Add your subtitle here
  coord_sf(xlim = c(-130, -65), ylim = c(20, 55)) +
  theme(legend.position = "right")

# Print the map
print(MAP_WEEK4)
```



Let's calculate the mean(WIS) improvement relative to the AUTO ARIMA model. We will compare each model with the AUTO ARIMA model for the same states. Later we sum the results of these comparisons. Positive results indicate that there was a general improvement in the mean(WIS) for a given model type among all states. Note that larger values on specific states may override a general trend and make the results less informative. So I do not strongly advice this analysis.

```
calculate_percentage_of_improvement <- function(data) {
  return(data.frame(
    AUTO_AR = (data$AUTO_AR - data$AUTO_AR),
    ES27_AR = ((data$AUTO_AR - data$ES27_AR) / data$AUTO_AR) * 100,
    ES64_AR = ((data$AUTO_AR - data$ES64_AR) / data$AUTO_AR) * 100,
    AUTO_ADJ = ((data$AUTO_AR - data$AUTO_ADJ) / data$AUTO_AR) * 100,
    ES27_ADJ = ((data$AUTO_AR - data$ES27_ADJ) / data$AUTO_AR) * 100,
    ES64_ADJ = ((data$AUTO_AR - data$ES64_ADJ) / data$AUTO_AR) * 100,
    AUTO_TMP = ((data$AUTO_AR - data$AUTO_TMP) / data$AUTO_AR) * 100,
    ES27_TMP = ((data$AUTO_AR - data$ES27_TMP) / data$AUTO_AR) * 100,
    ES64_TMP = ((data$AUTO_AR - data$ES64_TMP) / data$AUTO_AR) * 100,
    AUTO_EPI = ((data$AUTO_AR - data$AUTO_EPI) / data$AUTO_AR) * 100,
    ES27_EPI = ((data$AUTO_AR - data$ES27_EPI) / data$AUTO_AR) * 100,
    ES64_EPI = ((data$AUTO_AR - data$ES64_EPI) / data$AUTO_AR) * 100
  ))
}

# Calculate percentage of improvemtn
W1_percentage_of_improvement <- calculate_percentage_of_improvement(W1)
W2_percentage_of_improvement <- calculate_percentage_of_improvement(W2)
```

```r
W3_percentage_of_improvement <- calculate_percentage_of_improvement(W3)
W4_percentage_of_improvement <- calculate_percentage_of_improvement(W4)
```

Now let's plot a map of percentage of WIS improvement for each state.

```r
# MAP 1 week ahead

W1_map <- data.frame(
  STATE = W1$STATE,
  best_model = W1$Best_Result,
  percentage_improvement = apply(W1_percentage_of_improvement, 1, max)
)

# Merge the best model data with the states map
W1_map <- states %>%
  left_join(W1_map, by = c("STATE")) %>%
  filter(!is.na(percentage_improvement))

################################### MAP 2 weeka ahead

W2_map <- data.frame(
  STATE = W2$STATE,
  best_model = W2$Best_Result,
  percentage_improvement = apply(W2_percentage_of_improvement, 1, max)
)

# Merge the best model data with the states map
W2_map <- states %>%
  left_join(W2_map, by = c("STATE")) %>%
  filter(!is.na(percentage_improvement))


################################### MAP 3 weeka ahead

W3_map <- data.frame(
  STATE = W3$STATE,
  best_model = W3$Best_Result,
  percentage_improvement = apply(W3_percentage_of_improvement, 1, max)
)

# Merge the best model data with the states map
W3_map <- states %>%
  left_join(W3_map, by = c("STATE")) %>%
  filter(!is.na(percentage_improvement))

################################### MAP 4 weeka ahead

W4_map <- data.frame(
  STATE = W4$STATE,
  best_model = W4$Best_Result,
  percentage_improvement = apply(W4_percentage_of_improvement, 1, max)
)
```

```r
# Merge the best model data with the states map
W4_map <- states %>%
  left_join(W4_map, by = c("STATE")) %>%
  filter(!is.na(percentage_improvement))
```

Let's plot some maps with the best models in each state and the percentage of improvement compared to the AUTO ARIMA models for the same state.

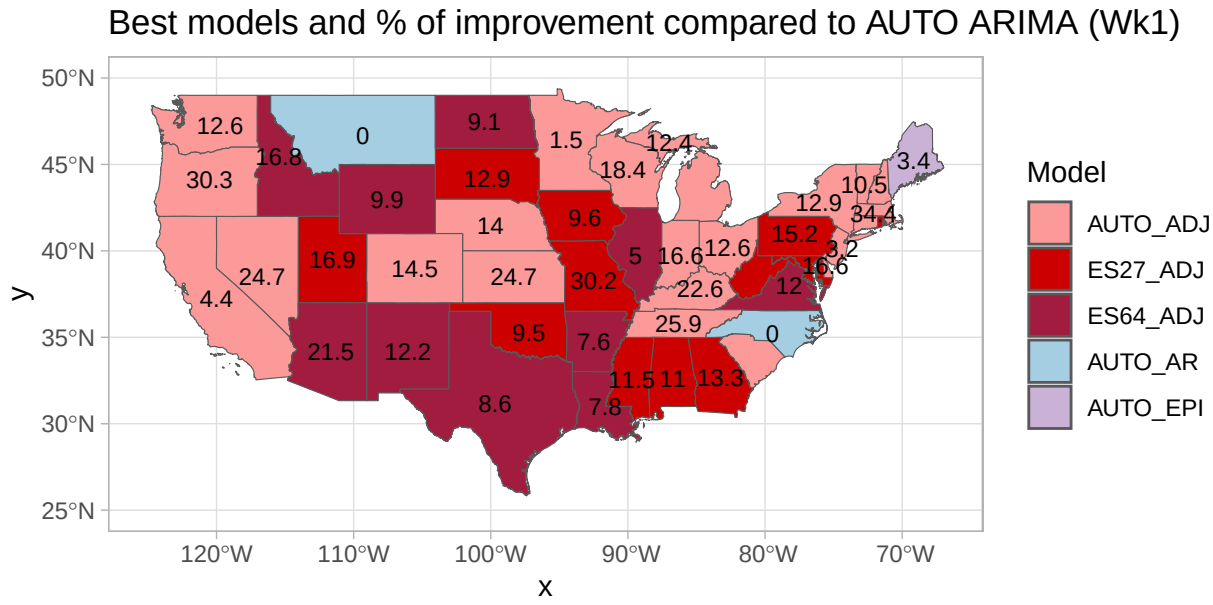
Here I define some colors categories that I will use on the next maps.

```r
# Define colors for the modelss
category_colors <- c(
  "AUTO_AR" = "#a6cee3",
  "ES27_AR" = "#226e83",
  "ES64_AR" = "#08306b",
  "AUTO_ADJ" = "#fb9a99",
  "ES27_ADJ" = "red3",
  "ES64_ADJ" = "#a11c3e",
  "AUTO_TMP" = "lightgreen",
  "ES27_TMP" = "green3",
  "ES64_TMP" = "#319045",
  "AUTO_EPI" = "#cab2d6",
  "ES27_EPI" = "purple2",
  "ES64_EPI" = "#5e2b7b"
)
```

1 week ahead percentage of improvement

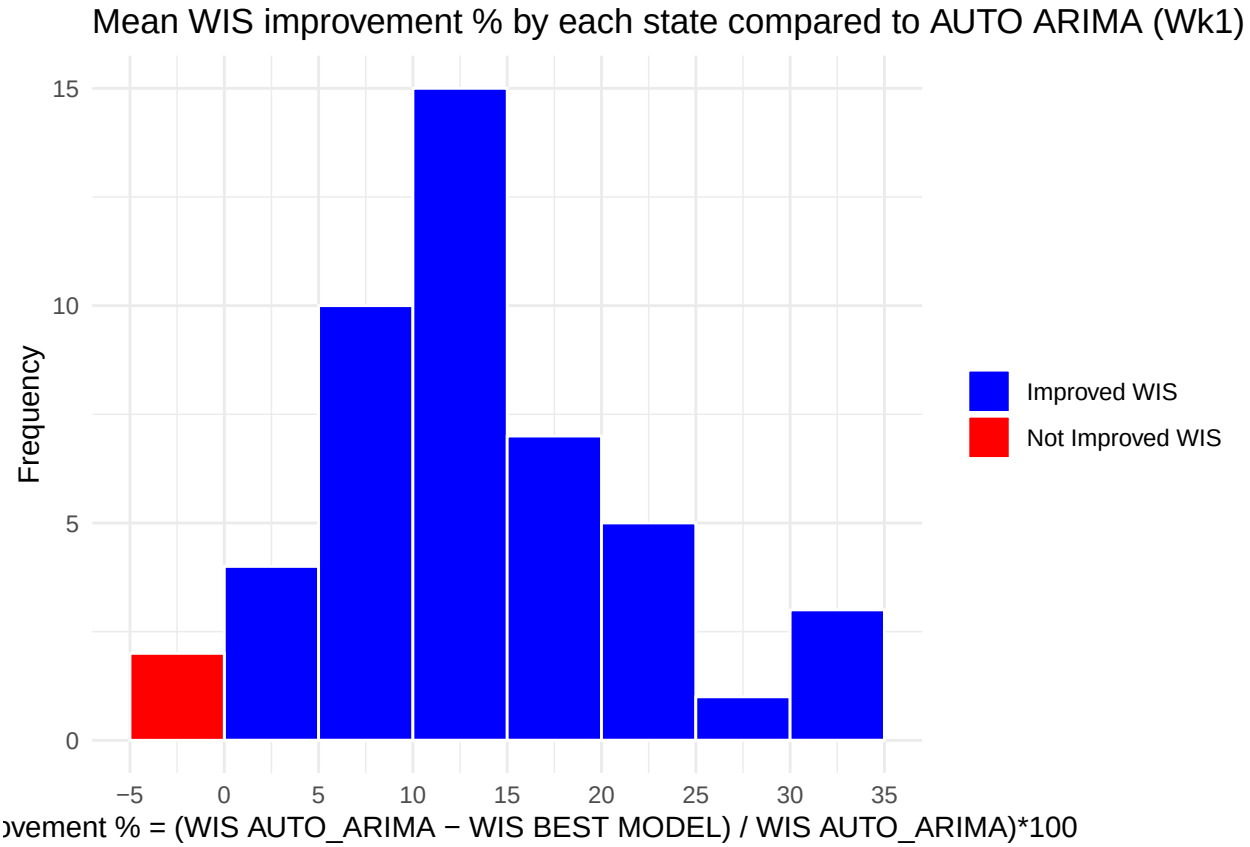
```r
# Assuming merged_data already includes the necessary columns: model and percentage_improveme
# Create the map plot for 1 Week Ahead with best models and percentage improvement
ES_1WEEK <- ggplot(W1_map) +
  geom_sf(aes(fill = best_model)) +  # Fill based on the best model
  scale_fill_manual(values = category_colors) +
  ggtitle("Best models and % of improvement compared to AUTO ARIMA (Wk1)") +
  theme_light() +
  theme(legend.position = "right") +
  geom_sf_text(data = W1_map, aes(label = round(percentage_improvement,1)),  # Round to whole numbers
               size = 3,
               color = "black",
               check_overlap = TRUE)+  # Display percentage improvement in each state
  labs(fill = "Model")  # Label for the legend

x_limits <- c(-125, -67)  # Set the desired longitude range
y_limits <- c(25, 50)    # Set the desired latitude range

ES_1WEEK + coord_sf(xlim = x_limits, ylim = y_limits)
```

```
## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
## give correct results for longitude/latitude data
```



Histogram of the percentage of improvement for all states - 1 week ahead

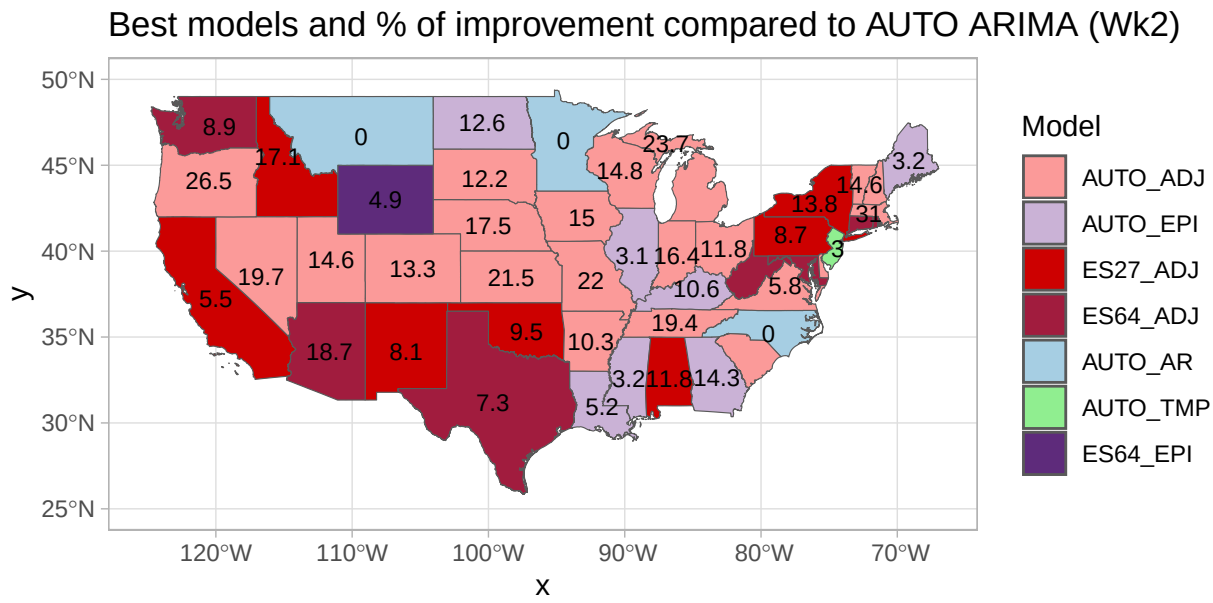
```
# Assuming your data is in W4_map$percentage_improvement
# Create a new column to classify positive and negative values
W1_map$category <- ifelse(W1_map$percentage_improvement > 0, "Improved WIS", "Not Improved WIS")

# Create the ggplot histogram
ggplot(W1_map, aes(x = percentage_improvement, fill = category)) +
  geom_histogram(binwidth = 5, color = "white", boundary = 0) +  # Set bin width to 5
  scale_fill_manual(values = c("Improved WIS" = "blue", "Not Improved WIS" = "red")) +  # Color mapping
  scale_x_continuous(breaks = seq(-5, 35, by = 5), limits = c(-5, 35)) +  # Set x-axis breaks
  labs(title = "Mean WIS improvement % by each state compared to AUTO ARIMA (Wk1)",
       x = "Improvement % = (WIS AUTO_ARIMA - WIS BEST MODEL) / WIS AUTO_ARIMA)*100",
       y = "Frequency", fill=NULL) +
  theme_minimal()
```



2 weeks ahead

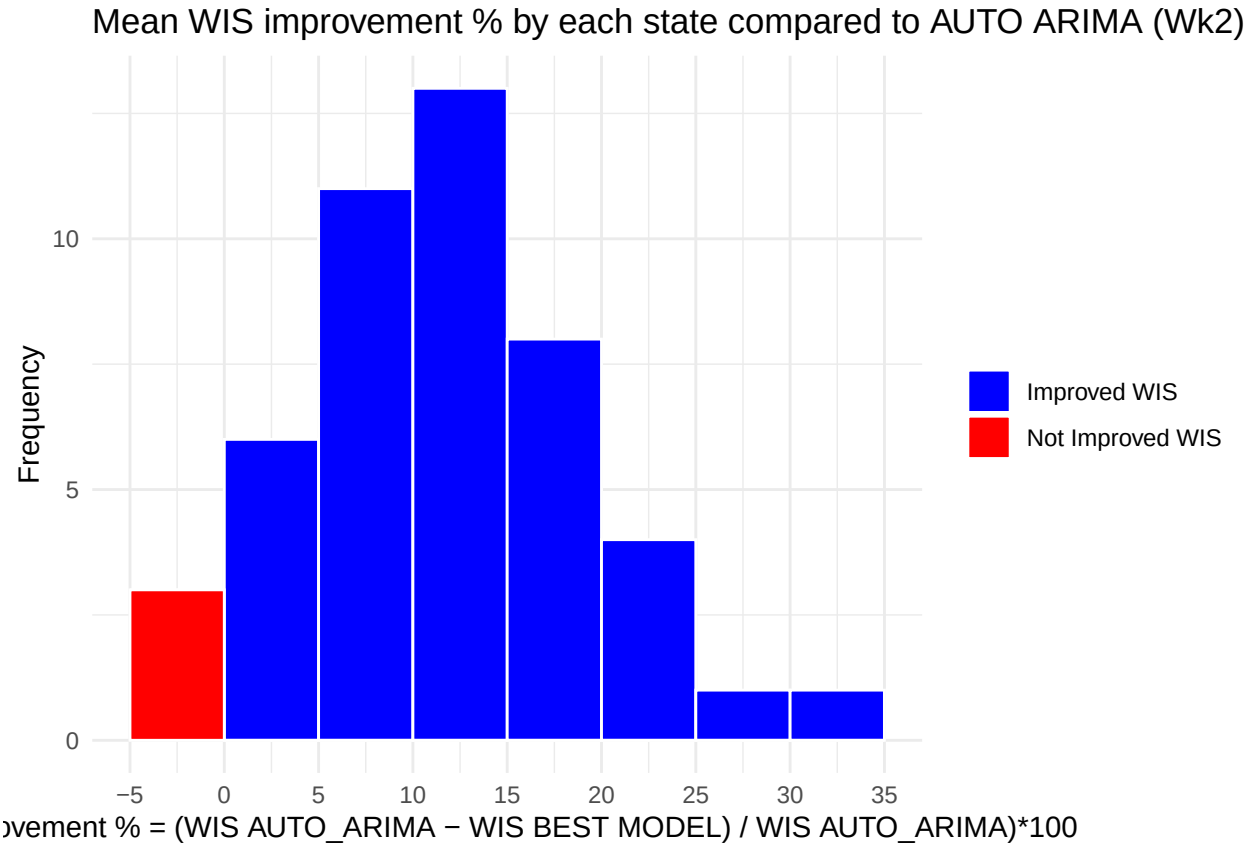
```
# Assuming merged_data already includes the necessary columns: model and percentage_improveme
# Create the map plot for 1 Week Ahead with best models and percentage improvement
ES_2WEEK <- ggplot(W2_map) +
  geom_sf(aes(fill = best_model)) +  # Fill based on the best model
  scale_fill_manual(values = category_colors) +
  ggtitle("Best models and % of improvement compared to AUTO ARIMA (Wk2)") +
  theme_light() +
  theme(legend.position = "right") +
  geom_sf_text(data = W2_map, aes(label = round(percentage_improvement,1)),  # Round to whole numbers
               size = 3,
               color = "black",
               check_overlap = TRUE)+  # Display percentage improvement in each state
  labs(fill = "Model")  # Label for the legend

x_limits <- c(-125, -67)  # Set the desired longitude range
y_limits <- c(25, 50)     # Set the desired latitude range

ES_2WEEK + coord_sf(xlim = x_limits, ylim = y_limits)
```

```
## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
## give correct results for longitude/latitude data
```



Histogram of the percentage of improvement for all states - 2 weeks ahead

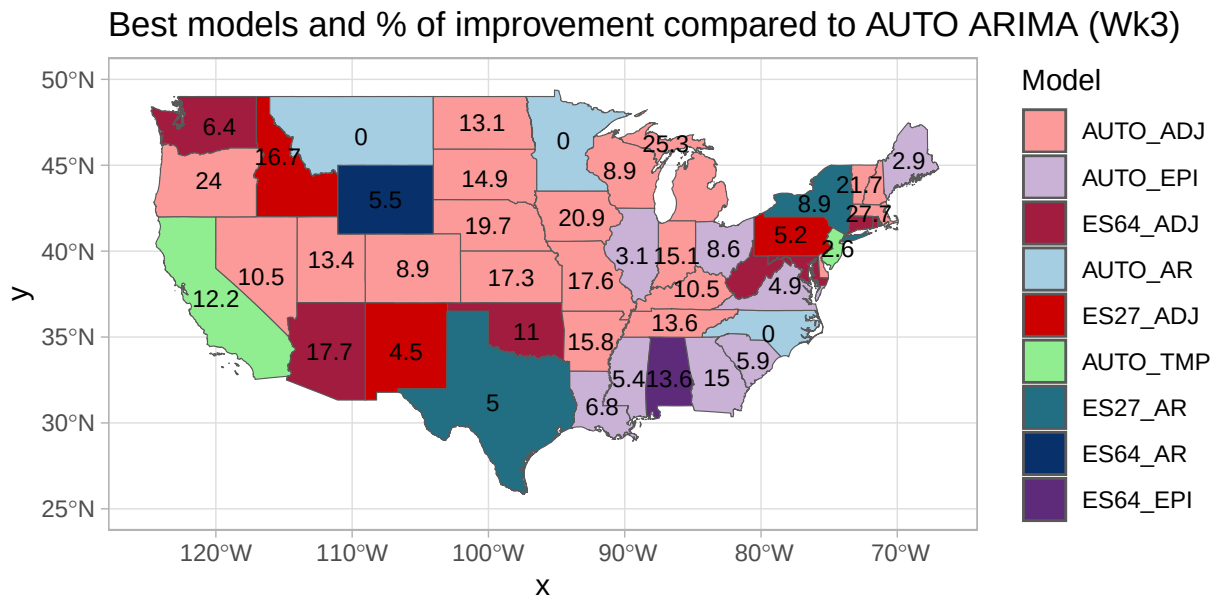
```
# Assuming your data is in W4_map$percentage_improvement
# Create a new column to classify positive and negative values
W2_map$category <- ifelse(W2_map$percentage_improvement > 0, "Improved WIS", "Not Improved WIS")

# Create the ggplot histogram
ggplot(W2_map, aes(x = percentage_improvement, fill = category)) +
  geom_histogram(binwidth = 5, color = "white", boundary = 0) +  # Set bin width to 5
  scale_fill_manual(values = c("Improved WIS" = "blue", "Not Improved WIS" = "red")) +  # Color mapping
  scale_x_continuous(breaks = seq(-5, 35, by = 5), limits = c(-5, 35)) +  # Set x-axis breaks
  labs(title = "Mean WIS improvement % by each state compared to AUTO ARIMA (Wk2)",
       x = "Improvement % = (WIS AUTO_ARIMA - WIS BEST MODEL) / WIS AUTO_ARIMA)*100",
       y = "Frequency", fill=NULL) +
  theme_minimal()
```



3 weeks ahead

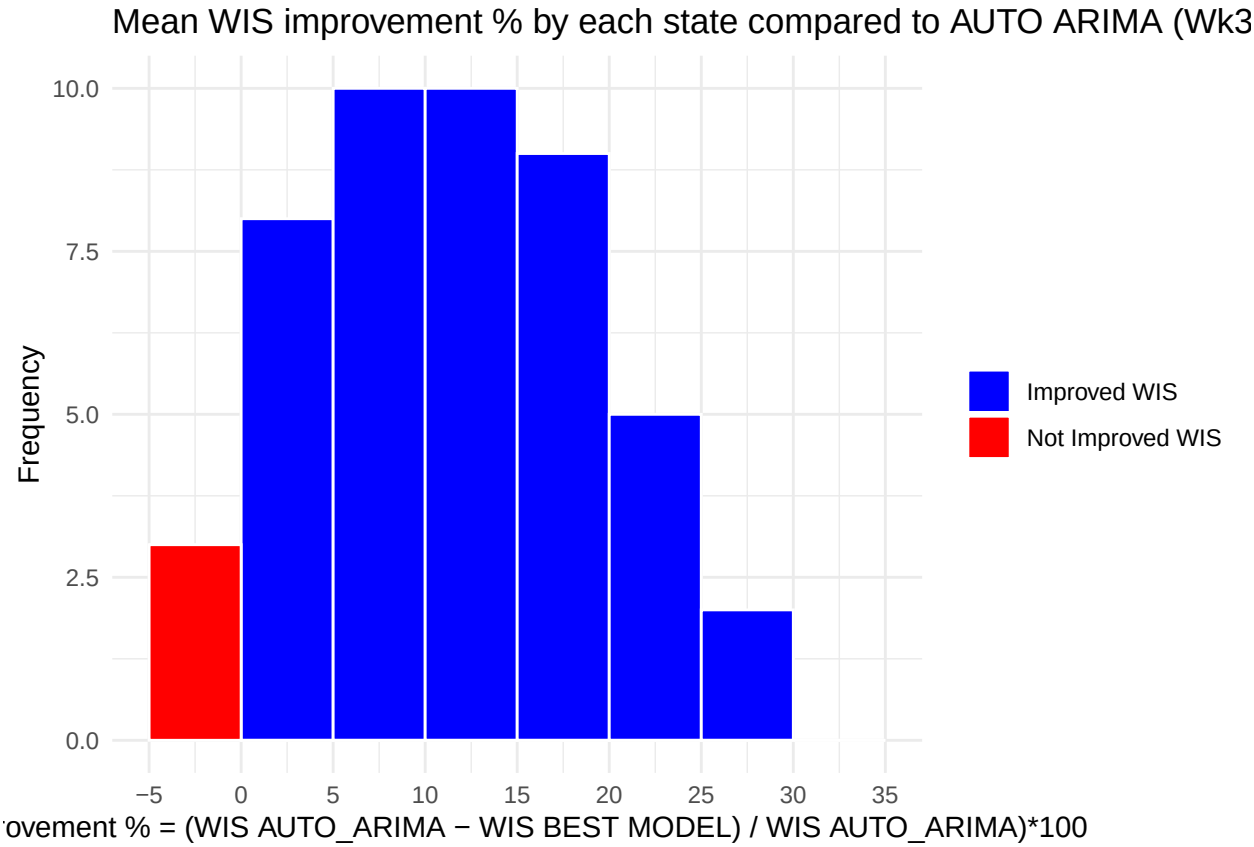
```
# Assuming merged_data already includes the necessary columns: model and percentage_improveme
# Create the map plot for 1 Week Ahead with best models and percentage improvement
ES_3WEEK <- ggplot(W3_map) +
  geom_sf(aes(fill = best_model)) +  # Fill based on the best model
  scale_fill_manual(values = category_colors) +
  ggtitle("Best models and % of improvement compared to AUTO ARIMA (Wk3)") +
  theme_light() +
  theme(legend.position = "right") +
  geom_sf_text(data = W3_map, aes(label = round(percentage_improvement,1)),  # Round to whole numbers
               size = 3,
               color = "black",
               check_overlap = TRUE)+  # Display percentage improvement in each state
  labs(fill = "Model")  # Label for the legend

x_limits <- c(-125, -67)  # Set the desired longitude range
y_limits <- c(25, 50)     # Set the desired latitude range

ES_3WEEK + coord_sf(xlim = x_limits, ylim = y_limits)
```

```
## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
## give correct results for longitude/latitude data
```



Histogram of the percentage of improvement for all states - 3 weeks ahead

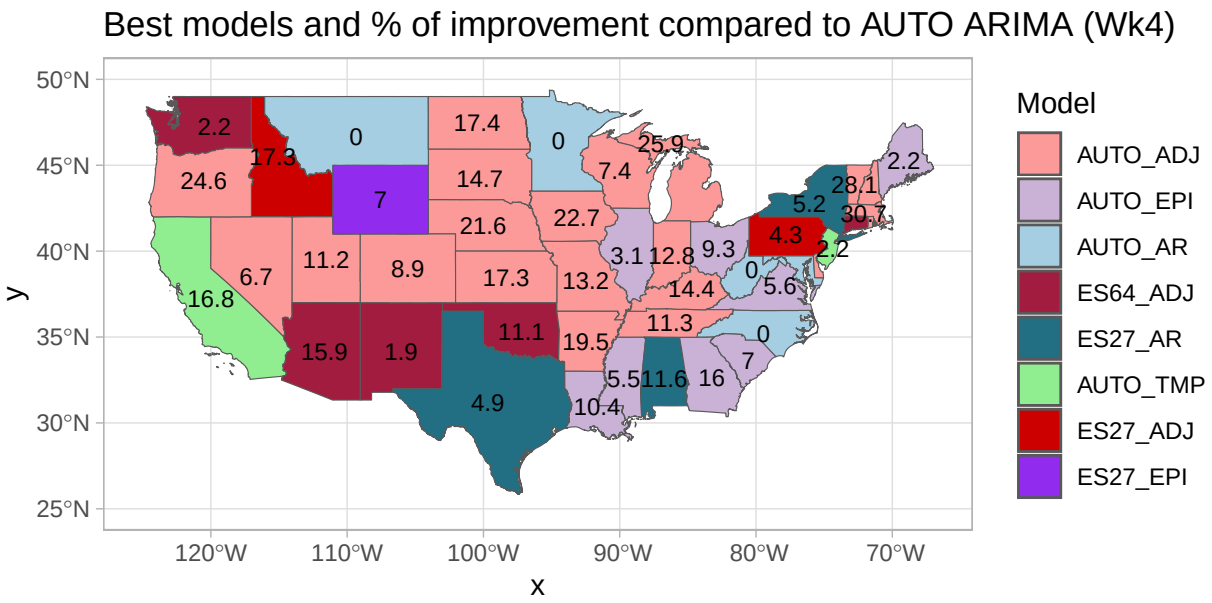
```
# Assuming your data is in W4_map$percentage_improvement
# Create a new column to classify positive and negative values
W3_map$category <- ifelse(W3_map$percentage_improvement > 0, "Improved WIS", "Not Improved WIS")

# Create the ggplot histogram
ggplot(W3_map, aes(x = percentage_improvement, fill = category)) +
  geom_histogram(binwidth = 5, color = "white", boundary = 0) +  # Set bin width to 5
  scale_fill_manual(values = c("Improved WIS" = "blue", "Not Improved WIS" = "red")) +  # Color mapping
  scale_x_continuous(breaks = seq(-5, 35, by = 5), limits = c(-5, 35)) +  # Set x-axis breaks
  labs(title = "Mean WIS improvement % by each state compared to AUTO ARIMA (Wk3)",
       x = "Improvement % = (WIS AUTO_ARIMA - WIS BEST MODEL) / WIS AUTO_ARIMA)*100",
       y = "Frequency", fill=NULL) +
  theme_minimal()
```



4 weeks ahead

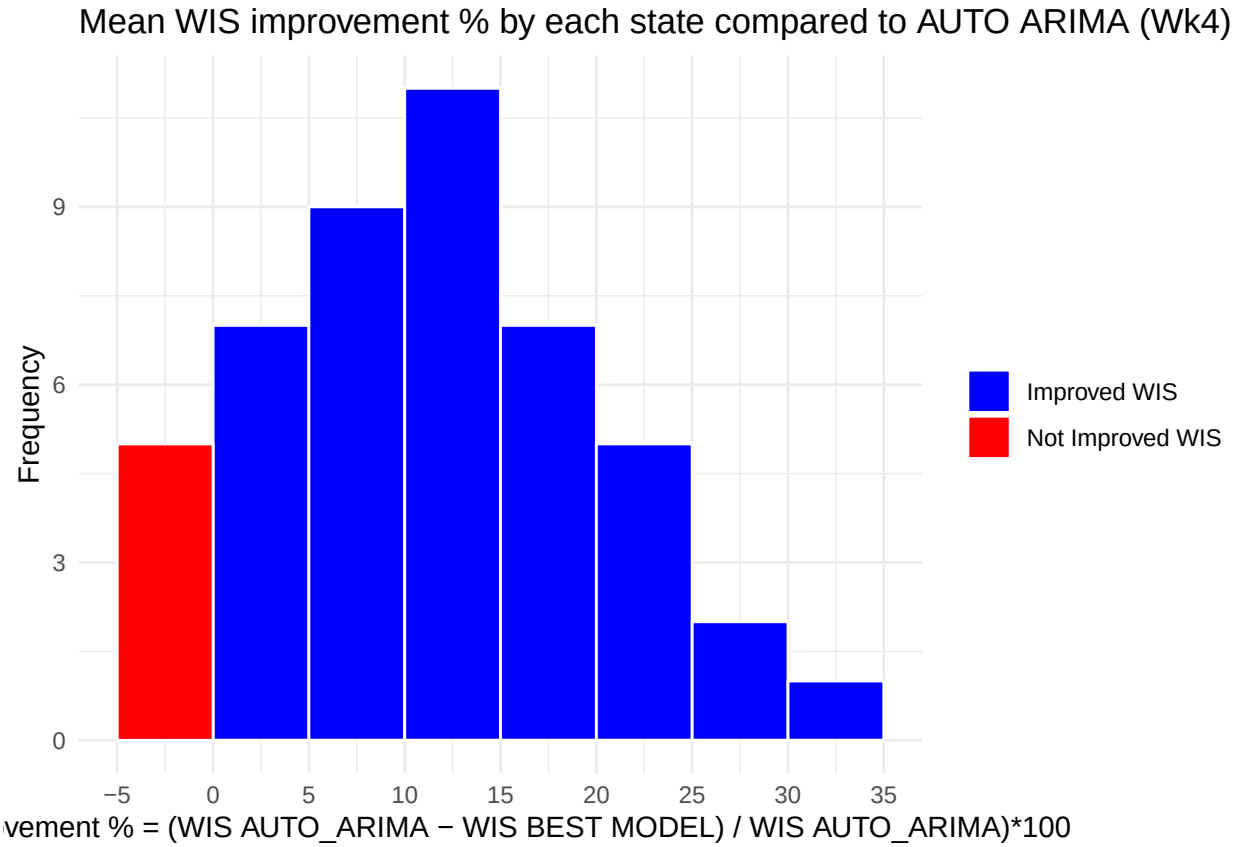
```
# Assuming merged_data already includes the necessary columns: model and percentage_improveme
# Create the map plot for 1 Week Ahead with best models and percentage improvement
ES_4WEEK <- ggplot(W4_map) +
  geom_sf(aes(fill = best_model)) +  # Fill based on the best model
  scale_fill_manual(values = category_colors) +
  ggtitle("Best models and % of improvement compared to AUTO ARIMA (Wk4)") +
  theme_light() +
  theme(legend.position = "right") +
  geom_sf_text(data = W4_map, aes(label = round(percentage_improvement,1)),  # Round to whole numbers
               size = 3,
               color = "black",
               check_overlap = TRUE)+  # Display percentage improvement in each state
  labs(fill = "Model")  # Label for the legend

x_limits <- c(-125, -67)  # Set the desired longitude range
y_limits <- c(25, 50)     # Set the desired latitude range

ES_4WEEK + coord_sf(xlim = x_limits, ylim = y_limits)
```

```
## Warning in st_point_on_surface.sfc(sf::st_zm(x)): st_point_on_surface may not
## give correct results for longitude/latitude data
```



Histogram of the percentage of improvement for all states - 4 weeks ahead

```r
# Assuming your data is in W4_map$percentage_improvement
# Create a new column to classify positive and negative values
W4_map$category <- ifelse(W4_map$percentage_improvement > 0, "Improved WIS", "Not Improved WIS")

# Create the ggplot histogram
ggplot(W4_map, aes(x = percentage_improvement, fill = category)) +
  geom_histogram(binwidth = 5, color = "white", boundary = 0) +  # Set bin width to 5
  scale_fill_manual(values = c("Improved WIS" = "blue", "Not Improved WIS" = "red")) +  # Color mapping
  scale_x_continuous(breaks = seq(-5, 35, by = 5), limits = c(-5, 35)) +  # Set x-axis breaks
  labs(title = "Mean WIS improvement % by each state compared to AUTO ARIMA (Wk4)",
       x = "Improvement % = (WIS AUTO_ARIMA - WIS BEST MODEL) / WIS AUTO_ARIMA)*100",
       y = "Frequency", fill=NULL) +
  theme_minimal()
```

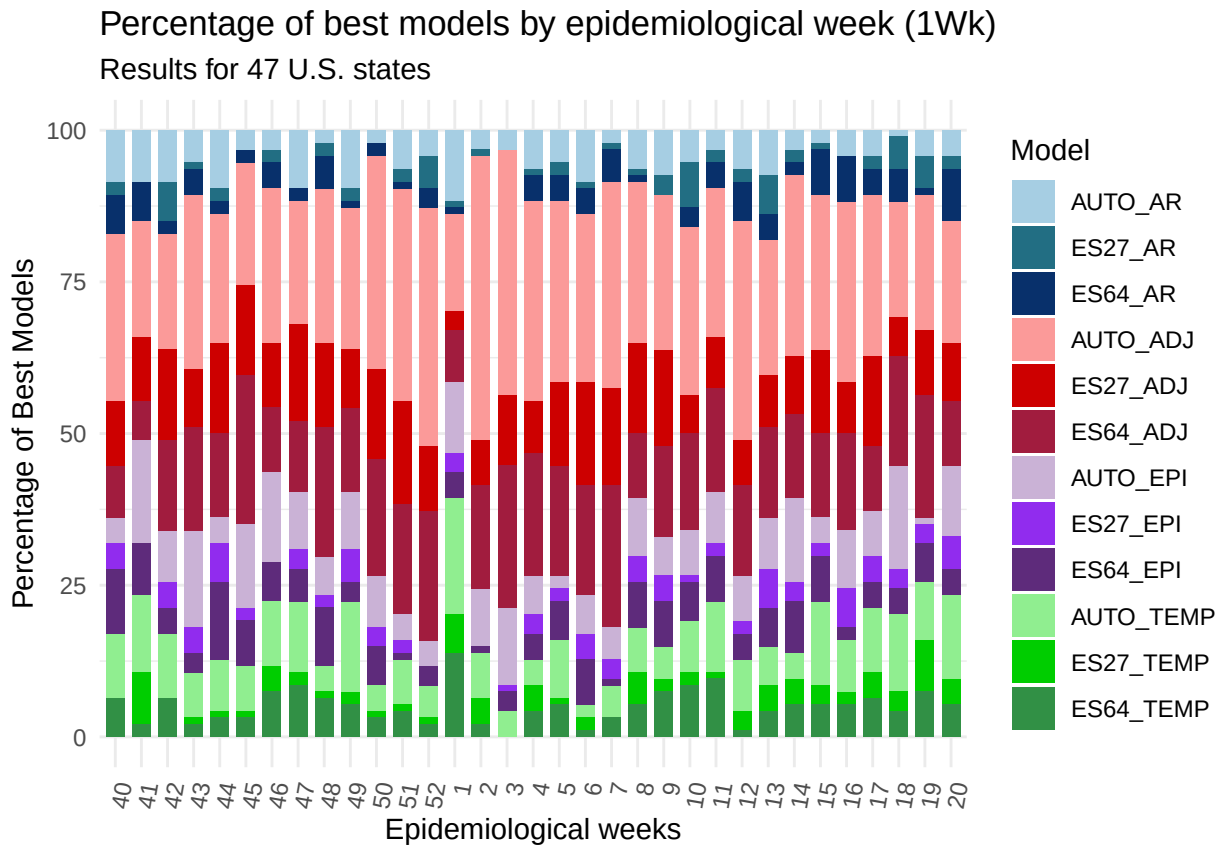


Now let's evaluate which are the best models by each epidemiological week Here I define some colors and model order for the next plots.

```
# Define a color palette for models
model_colors <- c(
  "AUTO_AR" = "#a6cee3",  # Blue
  "ES27_AR" = "#226e83",  # Light blue
  "ES64_AR" = "#08306b",  # Green
  "AUTO_ADJ" = "#fb9a99", # Red
  "ES27_ADJ" = "red3", # Light red #fb9a99
  "ES64_ADJ" = "#a11c3e", # Orange
  "AUTO_EPI" = "#cab2d6", # Yellow
  "ES27_EPI" = "purple2", # Lavender
  "ES64_EPI" = "#5e2b7b", # Purple
  "AUTO_TEMP" = "lightgreen",# Light orange
  "ES27_TEMP" = "green3", # Light green
  "ES64_TEMP" = "#319045" # Pale yellow
)

# Define the model order
model_order <- c(
  "AUTO_AR", "ES27_AR", "ES64_AR",
  "AUTO_ADJ", "ES27_ADJ", "ES64_ADJ",
  "AUTO_EPI", "ES27_EPI", "ES64_EPI",
  "AUTO_TEMP", "ES27_TEMP", "ES64_TEMP"
)
```

1 week ahead

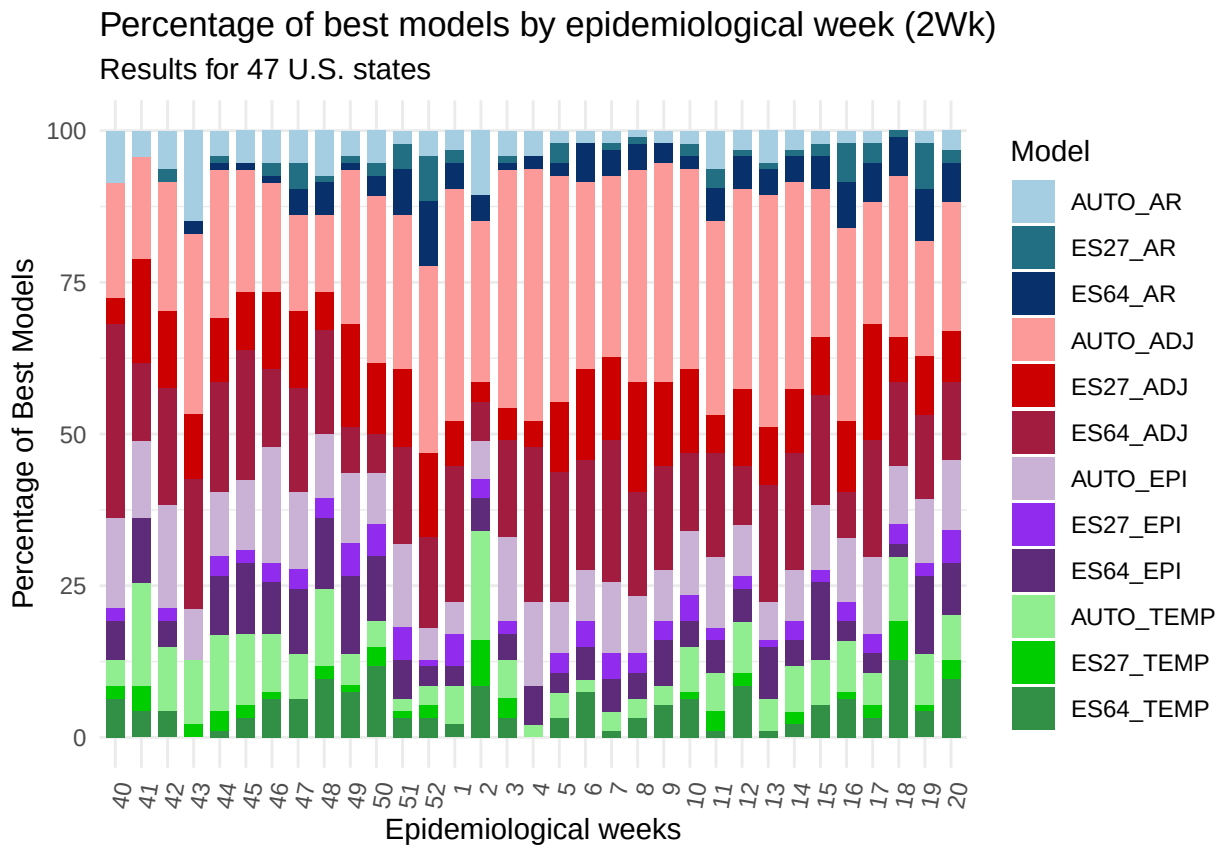
```
# Reshape the dataframe to long format for easier processing
df_long <- filtered_df_W1 %>%
  pivot_longer(cols = -c(STATE, Julian_date, epiweek),  # Keep Julian_date, epiweek, and STATE
               names_to = "model", values_to = "WIS") %>%
  mutate(model = gsub("_WIS", "", model))  # Remove _WIS suffix from model names

# Find the best model by Julian_date and STATE
best_model_per_state_date <- df_long %>%
  group_by(STATE, Julian_date) %>%
  slice(which.min(WIS)) %>%
  ungroup()

# Count occurrences of each best model by epiweeks across all states
counts_per_week <- best_model_per_state_date %>%
  group_by(epiweek, model) %>%
  summarize(count = n(), .groups = 'drop')

# Convert 'model' to a factor with custom levels
counts_per_week$model <- factor(counts_per_week$model, levels = model_order)

# Reorder epiweeks to start from week 39 to 53, then 1 to 38
epiweek_levels <- c(40:52, 1:20)

counts_per_week <- counts_per_week %>%
  mutate(epiweek = factor(epiweek, levels = epiweek_levels))

# Calculate total counts for each epiweek
total_counts_per_week <- counts_per_week %>%
  group_by(epiweek) %>%
  summarize(total_count = sum(count), .groups = 'drop')

# Join total counts back to counts_per_week and calculate percentages
counts_per_week <- counts_per_week %>%
  left_join(total_counts_per_week, by = "epiweek") %>%
  mutate(percentage = (count / total_count) * 100)
# Plot the best model counts as percentages
ggplot() +
  geom_bar(data = counts_per_week, aes(x = epiweek, y = percentage, fill = model),
           stat = "identity", position = "stack", width = 0.7) +
  scale_x_discrete(labels = as.character(epiweek_levels)) +
  labs(x = "Epidemiological weeks",
       y = "Percentage of Best Models",
       fill = "Model",
       title = "Percentage of best models by epidemiological week (1Wk)",
       subtitle = "Results for 47 U.S. states") +
  scale_fill_manual(values = model_colors) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 80, hjust = 1))
```



2 weeks ahead

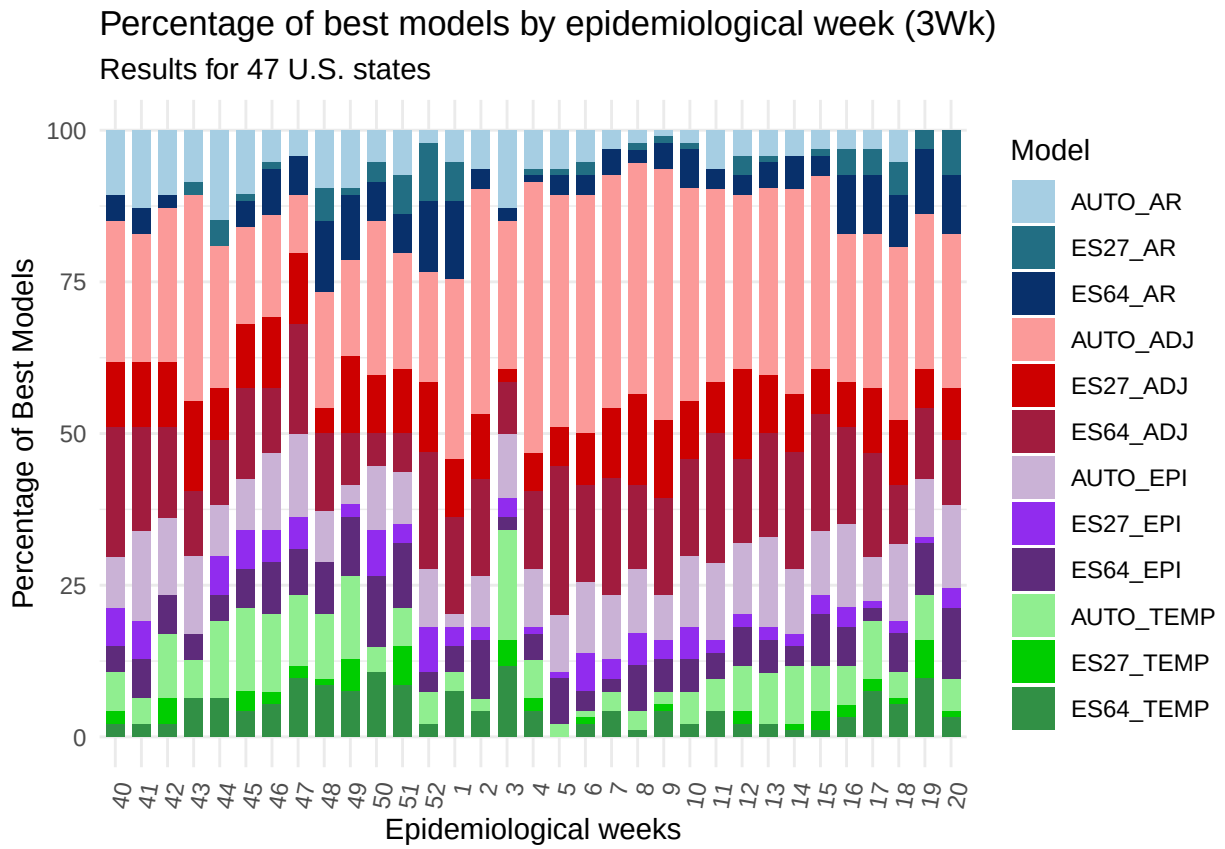
```
# Reshape the dataframe to long format for easier processing
df_long <- filtered_df_W2 %>%
  pivot_longer(cols = -c(STATE, Julian_date, epiweek),  # Keep Julian_date, epiweek, and STATE
               names_to = "model", values_to = "WIS") %>%
  mutate(model = gsub("_WIS", "", model))  # Remove _WIS suffix from model names

# Find the best model by Julian_date and STATE
best_model_per_state_date <- df_long %>%
  group_by(STATE, Julian_date) %>%
  slice(which.min(WIS)) %>%
  ungroup()

# Count occurrences of each best model by epiweeks across all states
counts_per_week <- best_model_per_state_date %>%
  group_by(epiweek, model) %>%
  summarize(count = n(), .groups = 'drop')

# Convert 'model' to a factor with custom levels
counts_per_week$model <- factor(counts_per_week$model, levels = model_order)

# Reorder epiweeks to start from week 39 to 53, then 1 to 38
epiweek_levels <- c(40:52, 1:20)

counts_per_week <- counts_per_week %>%
```

```
  mutate(epiweek = factor(epiweek, levels = epiweek_levels))

# Calculate total counts for each epiweek
total_counts_per_week <- counts_per_week %>%
  group_by(epiweek) %>%
  summarize(total_count = sum(count), .groups = 'drop')

# Join total counts back to counts_per_week and calculate percentages
counts_per_week <- counts_per_week %>%
  left_join(total_counts_per_week, by = "epiweek") %>%
  mutate(percentage = (count / total_count) * 100)
# Plot the best model counts as percentages
ggplot() +
  geom_bar(data = counts_per_week, aes(x = epiweek, y = percentage, fill = model),
           stat = "identity", position = "stack", width = 0.7) +
  scale_x_discrete(labels = as.character(epiweek_levels)) +
  labs(x = "Epidemiological weeks",
       y = "Percentage of Best Models",
       fill = "Model",
       title = "Percentage of best models by epidemiological week (2Wk)",
       subtitle = "Results for 47 U.S. states") +
  scale_fill_manual(values = model_colors) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 80, hjust = 1))
```



3 weeks ahead

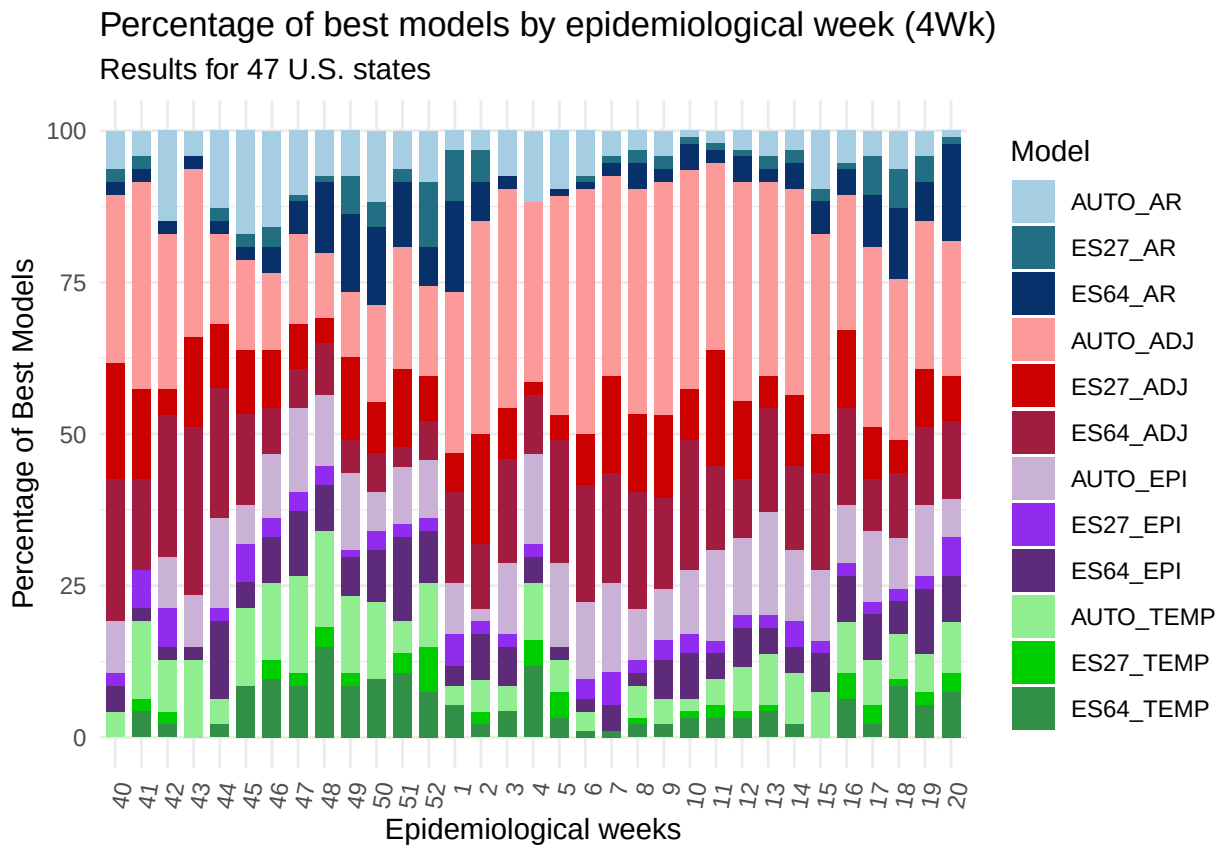
```r
# Reshape the dataframe to long format for easier processing
df_long <- filtered_df_W3 %>%
  pivot_longer(cols = -c(STATE, Julian_date, epiweek),  # Keep Julian_date, epiweek, and STATE
               names_to = "model", values_to = "WIS") %>%
  mutate(model = gsub("_WIS", "", model))  # Remove _WIS suffix from model names

# Find the best model by Julian_date and STATE
best_model_per_state_date <- df_long %>%
  group_by(STATE, Julian_date) %>%
  slice(which.min(WIS)) %>%
  ungroup()

# Count occurrences of each best model by epiweeks across all states
counts_per_week <- best_model_per_state_date %>%
  group_by(epiweek, model) %>%
  summarize(count = n(), .groups = 'drop')

# Convert 'model' to a factor with custom levels
counts_per_week$model <- factor(counts_per_week$model, levels = model_order)

# Reorder epiweeks to start from week 39 to 53, then 1 to 38
epiweek_levels <- c(40:52, 1:20)

counts_per_week <- counts_per_week %>%
  mutate(epiweek = factor(epiweek, levels = epiweek_levels))

# Calculate total counts for each epiweek
total_counts_per_week <- counts_per_week %>%
  group_by(epiweek) %>%
  summarize(total_count = sum(count), .groups = 'drop')

# Join total counts back to counts_per_week and calculate percentages
counts_per_week <- counts_per_week %>%
  left_join(total_counts_per_week, by = "epiweek") %>%
  mutate(percentage = (count / total_count) * 100)
# Plot the best model counts as percentages
ggplot() +
  geom_bar(data = counts_per_week, aes(x = epiweek, y = percentage, fill = model),
           stat = "identity", position = "stack", width = 0.7) +
  scale_x_discrete(labels = as.character(epiweek_levels)) +
  labs(x = "Epidemiological weeks",
       y = "Percentage of Best Models",
       fill = "Model",
       title = "Percentage of best models by epidemiological week (3Wk)",
       subtitle = "Results for 47 U.S. states") +
  scale_fill_manual(values = model_colors) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 80, hjust = 1))
```



4 weeks ahead

```
# Reshape the dataframe to long format for easier processing
df_long <- filtered_df_W4 %>%
  pivot_longer(cols = -c(STATE, Julian_date, epiweek),  # Keep Julian_date, epiweek, and STATE
               names_to = "model", values_to = "WIS") %>%
  mutate(model = gsub("_WIS", "", model))  # Remove _WIS suffix from model names

# Find the best model by Julian_date and STATE
best_model_per_state_date <- df_long %>%
  group_by(STATE, Julian_date) %>%
  slice(which.min(WIS)) %>%
  ungroup()

# Count occurrences of each best model by epiweeks across all states
counts_per_week <- best_model_per_state_date %>%
  group_by(epiweek, model) %>%
  summarize(count = n(), .groups = 'drop')

# Convert 'model' to a factor with custom levels
counts_per_week$model <- factor(counts_per_week$model, levels = model_order)

# Reorder epiweeks to start from week 39 to 53, then 1 to 38
epiweek_levels <- c(40:52, 1:20)

counts_per_week <- counts_per_week %>%
```

```r
  mutate(epiweek = factor(epiweek, levels = epiweek_levels))

# Calculate total counts for each epiweek
total_counts_per_week <- counts_per_week %>%
  group_by(epiweek) %>%
  summarize(total_count = sum(count), .groups = 'drop')

# Join total counts back to counts_per_week and calculate percentages
counts_per_week <- counts_per_week %>%
  left_join(total_counts_per_week, by = "epiweek") %>%
  mutate(percentage = (count / total_count) * 100)
# Plot the best model counts as percentages
ggplot() +
  geom_bar(data = counts_per_week, aes(x = epiweek, y = percentage, fill = model),
           stat = "identity", position = "stack", width = 0.7) +
  scale_x_discrete(labels = as.character(epiweek_levels)) +
  labs(x = "Epidemiological weeks",
       y = "Percentage of Best Models",
       fill = "Model",
       title = "Percentage of best models by epidemiological week (4Wk)",
       subtitle = "Results for 47 U.S. states") +
  scale_fill_manual(values = model_colors) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 80, hjust = 1))
```



Loading datasets for regression analysis

```
pop_data <- read.csv("regression_features/population_data.csv") # resident population and population de
sovi_data <- read.csv("regression_features/sovi_2010.2014.csv") # SOVI index
bric_data<-read.csv("regression_features/bric2015.csv") # BRIC index
humidity_data<-read.csv("regression_features/humidity_climatology_1990_2020.csv") # ERA5 Specific Humid
temperature_data<-read.csv("regression_features/temperature_climatology_1990_2020.csv") # ERA5 Specific
```

REGRESSION MODELS

We will run the regression for evaluating if the best model log(WIS) which represent its performance is related to given independent variables.

```
# List of data frames
WIS_dataframes <- list(W1 = W1, W2 = W2, W3 = W3, W4=W4)
regression_models<-data.frame()
```

All Regression results

AUTO_ADJ WIS x RESIDENT POPULATION 2020

```
for(i in c(1,2,3,4)){

  # Combining data for the same states
  colnames(WIS_dataframes[[i]])[1] <- "STATE"
  WIS_dataframes[[i]]$STATE <- as.character(WIS_dataframes[[i]]$STATE)
  WIS_pop_data <- inner_join(WIS_dataframes[[i]], pop_data, by = "STATE")
  # Fitting the regression model
  model <- lm(log1p(WIS_pop_data$AUTO_ADJ) ~ log(WIS_pop_data$Resident_population_2020))
  # View the model summary
  model_summary <- summary(model)
  # Getting the R and p values for the plot
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # Saving the results in a dataframe
  regression_models2 <- data.frame(
    independent_variable = "Resident Population (2020)",
    Week_Ahead = i,
    r_squared = r_squared,
    p_value = p_value
  )
  # Append to the main results dataframe
  regression_models <- rbind(regression_models, regression_models2)
}
```

AUTO_ADJ WIS x POPULATION DENSITY

```
for(i in c(1,2,3,4)){

  WIS_pop_data <- inner_join(WIS_dataframes[[i]], pop_data, by = "STATE")
  # Fit the regression model
  model <- lm(log1p(WIS_pop_data$AUTO_ADJ) ~ log(WIS_pop_data$Population_density_2020))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
```

```
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="Population Density (2020)","Week_Ahead"=i, "r_s
  # Append to the main results dataframe
  regression_models<-rbind(regression_models,regression_models2)
}
```

Here we will weight the Social Vulnerability Index (SoVI) and Baseline Resilience Indicators for Communities (BRIC) county indexes which for each states based on the population size in each county.

```
################################################################
# SOVI BY STATE
# weighted by population size in each county
sovi_by_state <- sovi_data %>%
  filter(!is.nan(sovi)) %>%  # Exclude rows where 'sovi' is NaN
  group_by(state.name) %>%
  summarize(weighted_mean = weighted.mean(sovi, w = population.2020, na.rm = TRUE))

colnames(sovi_by_state)[1] <- "STATE"

################################################################
# BRIC BY STATE
# weighted by population size in each county

bric_by_state <- bric_data %>%
  group_by(state.name) %>%
  summarize(across(16:22, ~ weighted.mean(.x, w = population.2020, na.rm = TRUE), .names = "weighted_mea
colnames(bric_by_state)[1] <- "STATE"
```

AUTO_ADJ WIS x SOVI

```
for(i in c(1,2,3,4)){

  WIS_sovi<-NULL
  WIS_sovi <- inner_join(WIS_dataframes[[i]], sovi_by_state, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_sovi$AUTO_ADJ) ~ (WIS_sovi$weighted_mean))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="Social Vulnerability Index (SoVI)","Week_Ahead
  # appending the results
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x BRIC SOCIAL

```
for(i in 1:4){
  WIS_bric <- inner_join(WIS_dataframes[[i]], bric_by_state, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_bric$AUTO_ADJ) ~ (WIS_bric$weighted_mean_z_bric.social))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="BRIC Social (2015)","Week_Ahead"=i, "r_squared
  # appending the results
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x BRIC ECONOMIC

```
for(i in 1:4){
  WIS_bric <- inner_join(WIS_dataframes[[i]], bric_by_state, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_bric$AUTO_ADJ) ~ (WIS_bric$weighted_mean_z_bric.economic))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="BRIC Economic (2015)","Week_Ahead"=i, "r_squar
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x BRIC Infrastructure

```
for (i in 1:4){
  WIS_bric <- inner_join(WIS_dataframes[[i]], bric_by_state, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_bric$AUTO_ADJ) ~ (WIS_bric$weighted_mean_z_bric.infrastructure))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="BRIC Infrastructure (2015)","Week_Ahead"=i, "r
  # appending results
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x BRIC institutional

```
for (i in 1:4){
  WIS_bric <- inner_join(WIS_dataframes[[i]], bric_by_state, by = "STATE")
```

```
  # Fit the regression model
  model <- lm(log(WIS_bric$AUTO_ADJ) ~ (WIS_bric$weighted_mean_z_bric.institutional))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="BRIC Institutional (2015)","Week_Ahead"=i, "r_s
  # appending results
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x BRIC community

```
for(i in 1:4){
  WIS_bric <- inner_join(WIS_dataframes[[1]], bric_by_state, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_bric$AUTO_ADJ) ~ (WIS_bric$weighted_mean_z_bric.community))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="BRIC Community (2015)","Week_Ahead"=i, "r_squa
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x BRIC environment

```
for(i in 1:4){
  WIS_bric <- inner_join(WIS_dataframes[[i]], bric_by_state, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_bric$AUTO_ADJ) ~ (WIS_bric$weighted_mean_z_bric.environment))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="BRIC Environment (2015)","Week_Ahead"=i, "r_sq
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x BRIC total

```
for(i in 1:4){
  # BRIC INDEX
  WIS_bric <- inner_join(WIS_dataframes[[i]], bric_by_state, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_bric$AUTO_ADJ) ~ (WIS_bric$weighted_mean_z_bric.total))
```

```
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="BRIC total (2015)","Week_Ahead"=i, "r_squared"=
  regression_models<-rbind(regression_models,regression_models2)
}
```

TEMPERATUE - ERA5

Now we will look at regression models that uses mean temperature and specific humidity.

AUTO_ADJ WIS x Temperature ERA5 Data

```
for(i in 1:4){
  # TEMPERATURE DATA from ERA5
  WIS_temp <- inner_join(WIS_dataframes[[i]], temperature_data, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_temp$AUTO_ADJ) ~ (WIS_temp$mean))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="Mean Temperature (1990-2020)","Week_Ahead"=i,
  # appending results
  regression_models<-rbind(regression_models,regression_models2)
}
```

AUTO_ADJ WIS x Specific Humidity ERA5 Data

```
# regression results
print(regression_models)
```

```
##                 independent_variable Week_Ahead r_squared  p_value
## 1         Resident Population (2020)          1     0.865 3.62e-21
## 2         Resident Population (2020)          2     0.866 3.03e-21
## 3         Resident Population (2020)          3     0.864 4.17e-21
## 4         Resident Population (2020)          4     0.859 9.75e-21
## 5          Population Density (2020)          1     0.192 2.05e-03
## 6          Population Density (2020)          2     0.193 2.00e-03
## 7          Population Density (2020)          3     0.197 1.76e-03
## 8          Population Density (2020)          4     0.196 1.85e-03
## 9  Social Vulnerability Index (SoVI)          1     0.086 4.52e-02
## 10 Social Vulnerability Index (SoVI)          2     0.088 4.24e-02
## 11 Social Vulnerability Index (SoVI)          3     0.094 3.62e-02
## 12 Social Vulnerability Index (SoVI)          4     0.099 3.12e-02
```

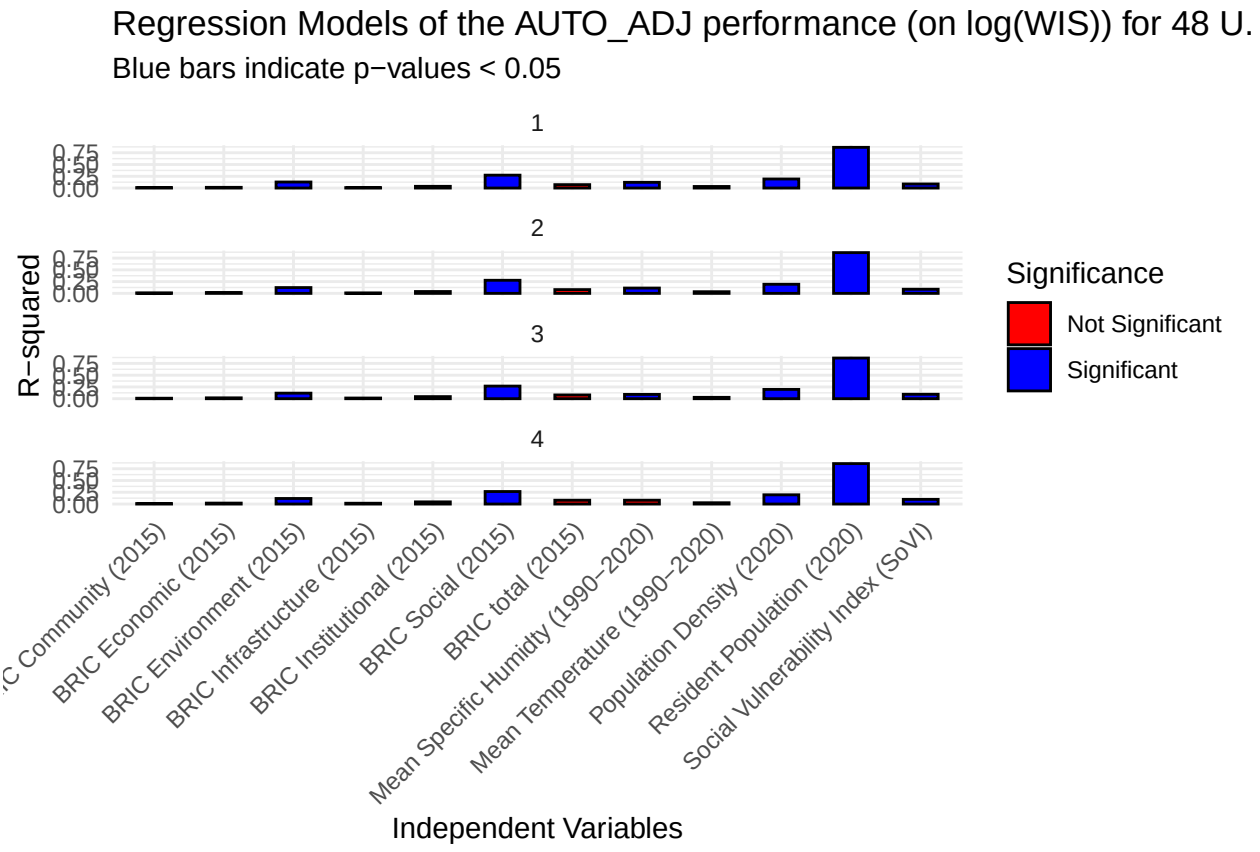
```
## 13                BRIC Social (2015)          1     0.274 1.59e-04
## 14                BRIC Social (2015)          2     0.278 1.40e-04
## 15                BRIC Social (2015)          3     0.268 1.95e-04
## 16                BRIC Social (2015)          4     0.266 2.09e-04
## 17              BRIC Economic (2015)          1     0.015 4.18e-01
## 18              BRIC Economic (2015)          2     0.018 3.68e-01
## 19              BRIC Economic (2015)          3     0.018 3.64e-01
## 20              BRIC Economic (2015)          4     0.020 3.39e-01
## 21        BRIC Infrastructure (2015)          1     0.009 5.29e-01
## 22        BRIC Infrastructure (2015)          2     0.009 5.23e-01
## 23        BRIC Infrastructure (2015)          3     0.014 4.23e-01
## 24        BRIC Infrastructure (2015)          4     0.017 3.88e-01
## 25         BRIC Institutional (2015)          1     0.036 2.04e-01
## 26         BRIC Institutional (2015)          2     0.039 1.84e-01
## 27         BRIC Institutional (2015)          3     0.044 1.57e-01
## 28         BRIC Institutional (2015)          4     0.046 1.48e-01
## 29             BRIC Community (2015)          1     0.010 5.13e-01
## 30             BRIC Community (2015)          2     0.010 5.13e-01
## 31             BRIC Community (2015)          3     0.010 5.13e-01
## 32             BRIC Community (2015)          4     0.010 5.13e-01
## 33           BRIC Environment (2015)          1     0.128 1.35e-02
## 34           BRIC Environment (2015)          2     0.123 1.57e-02
## 35           BRIC Environment (2015)          3     0.118 1.81e-02
## 36           BRIC Environment (2015)          4     0.118 1.82e-02
## 37                 BRIC total (2015)          1     0.072 6.78e-02
## 38                 BRIC total (2015)          2     0.080 5.42e-02
## 39                 BRIC total (2015)          3     0.080 5.42e-02
## 40                 BRIC total (2015)          4     0.081 5.18e-02
## 41      Mean Temperature (1990-2020)          1     0.033 2.19e-01
## 42      Mean Temperature (1990-2020)          2     0.035 2.10e-01
## 43      Mean Temperature (1990-2020)          3     0.029 2.54e-01
## 44      Mean Temperature (1990-2020)          4     0.027 2.67e-01
```

```r
for(i in 1:4){
  # HUMIDITY DATA from ERA5
  WIS_humidity <- inner_join(WIS_dataframes[[i]], humidity_data, by = "STATE")
  # Fit the regression model
  model <- lm(log(WIS_humidity$AUTO_ADJ) ~ (WIS_humidity$mean))
  # View the model summary
  model_summary <- summary(model)
  # Extract R-squared and p-value
  r_squared <- round(model_summary$r.squared, 3)
  p_value <- signif(model_summary$coefficients[2, 4], 3)
  # saving the results in a dataframe
  regression_models2<-data.frame("independent_variable"="Mean Specific Humidty (1990-2020)","Week_Ahead
  # appending results
  regression_models<-rbind(regression_models,regression_models2)
}
```

SUMMARY OF RESULTS

```r
# Add a significance indicator for coloring
data <- regression_models %>%
  mutate(Significant = ifelse(p_value < 0.05, "Significant", "Not Significant"))
```

```r
# Plot R-squared with conditional coloring and faceting by Week_Ahead
ggplot(data, aes(x = independent_variable, y = r_squared, fill = Significant)) +
  geom_col(color = "black", width = 0.5) +
  scale_fill_manual(values = c("Significant" = "blue", "Not Significant" = "red")) +
  labs(
    title = "Regression Models of the AUTO_ADJ performance (on log(WIS)) for 48 U.S. States (1 to 4 week
    subtitle = "Blue bars indicate p-values < 0.05",
    x = "Independent Variables",
    y = "R-squared",
    fill = "Significance"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  facet_wrap(~Week_Ahead, ncol = 1)
```




```r
# Save the plot with specified size
ggsave("regression_plot.jpg", width = 10, height = 14, dpi = 300)
```

Friedman variance test ##########################

1 WEEK AHEAD Post Hoc Wilcox test

```r
library(lme4)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'

## The following object is masked from 'package:spam':
##
##     det

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
library(ez)

# Reshape the data into long format
long_data <- W1 %>%
  pivot_longer(cols = c(-STATE,-Best_Result) , names_to = "Type", values_to = "Value")%>%
  mutate(Type = recode(Type, "AUTO_AR" = "AUTO_AAR"))

########################
# Friedman test since we have related samples (models by each STATE)
# and non-normally distributed data
# therefore, it could be better to use this non-parametric test

week1_wis_test<-friedman.test(Value ~ Type | STATE, data = long_data)
# results
friedman_results_w1<-data.frame("chi_squared"=week1_wis_test$statistic, "p_value"=week1_wis_test$p.valu
#friedman_results_w1

#############################################################################
wilcox_results <- pairwise.wilcox.test(long_data$Value, long_data$Type,
                                       p.adjust.method ="BY",
                                       paired = TRUE)
p_values<-wilcox_results$p.value
p_values<-data.frame(p_values)

#############################################################################
p_df1 <- data.frame(Models = rownames(p_values), PValue = p_values$AUTO_AAR, WeekAhead=1)
p_df1 <-drop_na(p_df1)
#p_df1
```

2 WEEKS AHEAD Post Hoc Wilcox test

```
# Reshape the data into long format
long_data <- W2 %>%
  pivot_longer(cols = c(-STATE,-Best_Result) , names_to = "Type", values_to = "Value")%>%
  mutate(Type = recode(Type, "AUTO_AR" = "AUTO_AAR"))

########################
# Friedman test since we have related samples (models by each STATE)
# and non-normally distributed data
# therefore, it could be better to use this non-parametric test
week2_wis_test<-friedman.test(Value ~ Type | STATE, data = long_data)
# adding results to the dataframe
```

```
friedman_results_w2<-data.frame("chi_squared"=week2_wis_test$statistic, "p_value"=week2_wis_test$p.valu

#############################################################################
wilcox_results <- pairwise.wilcox.test(long_data$Value, long_data$Type,
                                       p.adjust.method ="BY"   ,
                                       paired = TRUE)

p_values<-wilcox_results$p.value
p_values<-data.frame(p_values)

#############################################################################
p_df2 <- data.frame(Models = rownames(p_values), PValue = p_values$AUTO_AAR, WeekAhead=2)
p_df2 <-drop_na(p_df2)
#p_df2
```

3 WEEKS AHEAD Post Hoc Wilcox test

```
# Reshape the data into long format
long_data <- W3 %>%
  pivot_longer(cols = c(-STATE,-Best_Result) , names_to = "Type", values_to = "Value")%>%
  mutate(Type = recode(Type, "AUTO_AR" = "AUTO_AAR"))

######################
# Friedman test since we have related samples (models by each STATE)
# and non-normally distributed data
# therefore, it could be better to use this non-parametric test
week3_wis_test<-friedman.test(Value ~ Type | STATE, data = long_data)
# adding results to the dataframe
friedman_results_w3<-data.frame("chi_squared"=week3_wis_test$statistic, "p_value"=week3_wis_test$p.valu

#############################################################################
wilcox_results <- pairwise.wilcox.test(long_data$Value, long_data$Type,
                                       p.adjust.method ="BY"   ,
                                       paired = TRUE)

p_values<-wilcox_results$p.value
p_values<-data.frame(p_values)

#############################################################################
p_df3 <- data.frame(Models = rownames(p_values), PValue = p_values$AUTO_AAR, WeekAhead=3)
p_df3 <-drop_na(p_df3)
#p_df3
```

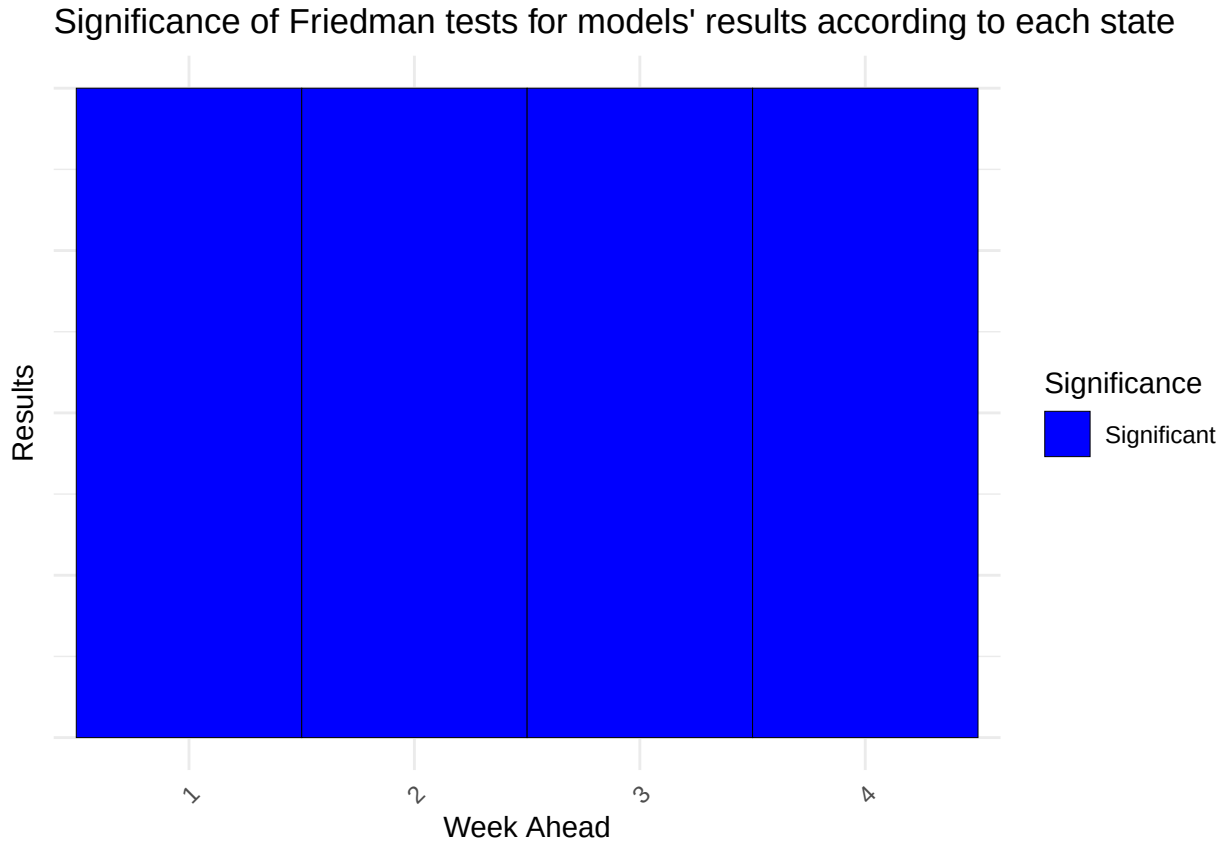
4 WEEKS AHEAD Post Hoc Wilcox test

```
# Reshape the data into long format
long_data <- W4 %>%
  pivot_longer(cols = c(-STATE,-Best_Result) , names_to = "Type", values_to = "Value")%>%
  mutate(Type = recode(Type, "AUTO_AR" = "AUTO_AAR"))

######################
# Friedman test since we have related samples (models by each STATE)
# and non-normally distributed data
```

```r
# therefore, it could be better to use this non-parametric test
week4_wis_test<-friedman.test(Value ~ Type | STATE, data = long_data)
# adding results to the dataframe
friedman_results_w4<-data.frame("chi_squared"=week4_wis_test$statistic, "p_value"=week4_wis_test$p.valu

##############################################################################
wilcox_results <- pairwise.wilcox.test(long_data$Value, long_data$Type,
                                       p.adjust.method ="BY"  ,
                                       paired = TRUE)

p_values<-wilcox_results$p.value
p_values<-data.frame(p_values)

##############################################################################
p_df4 <- data.frame(Models = rownames(p_values), PValue = p_values$AUTO_AAR, WeekAhead=4)
p_df4 <-drop_na(p_df4)
#p_df4
```

Friedman test

```r
friedman_results<-rbind("1week_ahead"=friedman_results_w1,"2week_ahead"=friedman_results_w2,"3week_ahea

weeks_<-data.frame("weeks_ahead"=c(1,2,3,4))

friedman_results<-cbind(friedman_results,weeks_)

# Define significance threshold (e.g., 0.05)
significance_level <- 0.05

# Add a new column to classify significance
friedman_results$significance <- ifelse(friedman_results$p_value < significance_level, "Significant", "

# Create the barplot
ggplot(friedman_results, aes(x = factor(weeks_ahead), y = 1, fill = significance)) +
  geom_tile(color = "black") +  # Add black borders to squares
  scale_fill_manual(values = c("Significant" = "blue", "Not Significant" = "red")) +
  labs(title = "Significance of Friedman tests for models' results according to each state",
       x = "Week Ahead",
       y = "Results",
       fill = "Significance") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        axis.text.y = element_blank(),   # Remove y-axis values
        axis.ticks.y = element_blank())
```



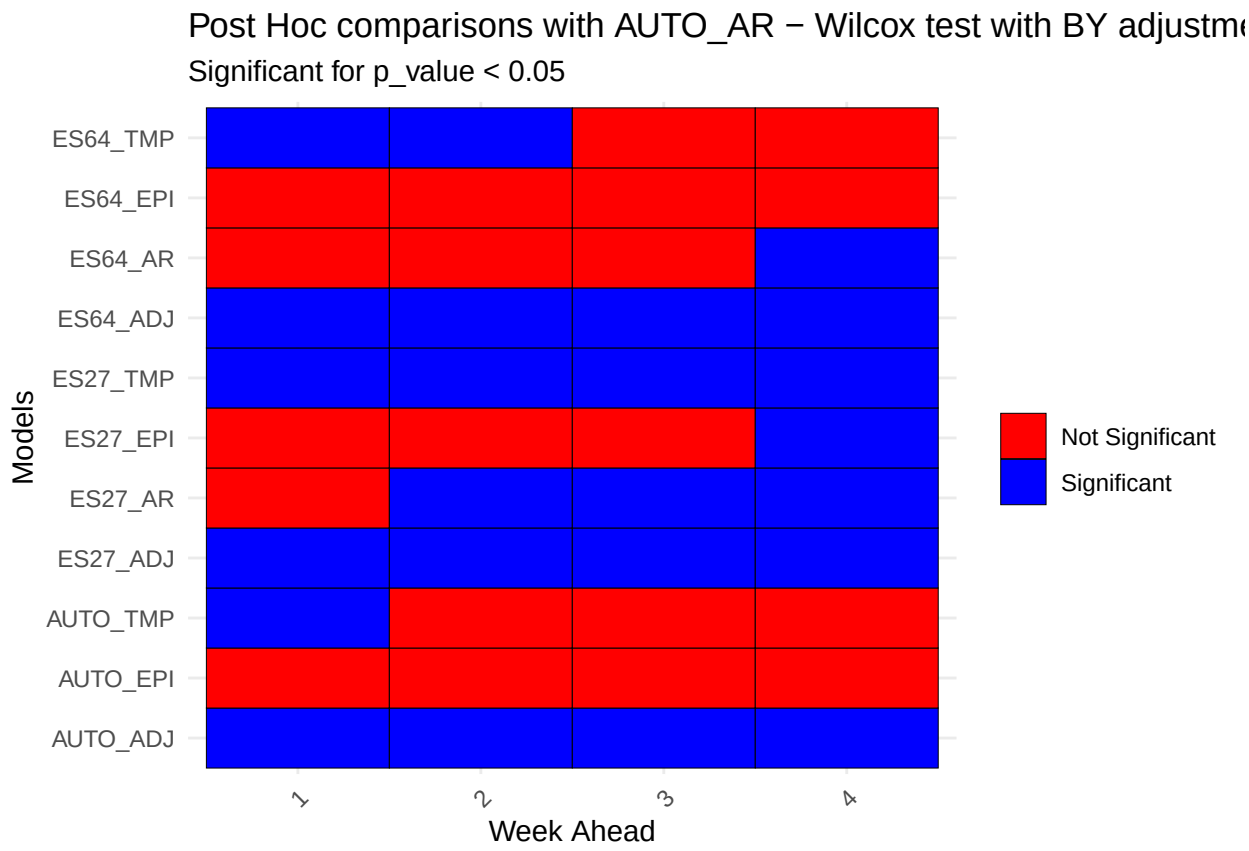
Summary of Post Hoc Wilcox test with p.adjust.method Benjamini-Yekutieli (BY)

```
all_p_values<-rbind(p_df1,p_df2,p_df3,p_df4)
all_p_values
```

```
##      Models       PValue WeekAhead
## 1  AUTO_ADJ 7.887701e-09         1
## 2  AUTO_EPI 4.542348e-01         1
## 3  AUTO_TMP 4.907479e-04         1
## 4  ES27_ADJ 1.020690e-07         1
## 5   ES27_AR 1.944409e-01         1
## 6  ES27_EPI 1.000000e+00         1
## 7  ES27_TMP 1.734716e-05         1
## 8  ES64_ADJ 8.909747e-07         1
## 9   ES64_AR 1.824974e-01         1
## 10 ES64_EPI 1.000000e+00         1
## 11 ES64_TMP 8.531395e-06         1
## 12 AUTO_ADJ 6.130204e-05         2
## 13 AUTO_EPI 2.978452e-01         2
## 14 AUTO_TMP 1.000000e+00         2
## 15 ES27_ADJ 1.015871e-05         2
## 16  ES27_AR 6.489603e-03         2
## 17 ES27_EPI 9.861797e-01         2
## 18 ES27_TMP 9.676854e-04         2
## 19 ES64_ADJ 5.138604e-06         2
## 20  ES64_AR 3.151566e-01         2
```

```
## 21 ES64_EPI 1.000000e+00         2
## 22 ES64_TMP 1.721197e-02         2
## 23 AUTO_ADJ 3.308830e-03         3
## 24 AUTO_EPI 9.702402e-01         3
## 25 AUTO_TMP 1.000000e+00         3
## 26 ES27_ADJ 5.004756e-04         3
## 27  ES27_AR 3.882637e-03         3
## 28 ES27_EPI 7.103520e-02         3
## 29 ES27_TMP 5.946847e-03         3
## 30 ES64_ADJ 3.543268e-04         3
## 31  ES64_AR 6.216691e-02         3
## 32 ES64_EPI 3.474059e-01         3
## 33 ES64_TMP 6.812808e-02         3
## 34 AUTO_ADJ 1.365303e-02         4
## 35 AUTO_EPI 1.000000e+00         4
## 36 AUTO_TMP 1.000000e+00         4
## 37 ES27_ADJ 4.350505e-02         4
## 38  ES27_AR 4.259751e-04         4
## 39 ES27_EPI 9.234780e-03         4
## 40 ES27_TMP 1.178278e-03         4
## 41 ES64_ADJ 3.144871e-02         4
## 42  ES64_AR 7.712217e-03         4
## 43 ES64_EPI 1.056350e-01         4
## 44 ES64_TMP 5.043555e-02         4
```

```r
# Define significance threshold
alpha <- 0.05

# Create significance column
p_df <- all_p_values %>%
  mutate(Significance = ifelse(PValue < alpha, "Significant", "Not Significant"))

# Plot
ggplot(p_df, aes(x = factor(WeekAhead), y = Models, fill = Significance)) +
  geom_tile(color = "black") +  # Add black borders to squares
  scale_fill_manual(values = c("Significant" = "blue", "Not Significant" = "red")) +
  labs(title = "Post Hoc comparisons with AUTO_AR - Wilcox test with BY adjustment",
       subtitle = "Significant for p_value < 0.05",
       x = "Week Ahead",
       y = "Models",
       fill = "") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



This figure is just a example of a probabilistic forecast for 4 weeks ahead.

```r
# Load necessary library
library(ggplot2)

# Example data
data <- data.frame(
  horizon = c(1, 2, 3, 4),
  mean = c(10, 20, 30, 40),
  lower = c(5, 9, 15, 20),
  upper = c(15, 28, 40, 60)
)

# Create the plot
ggplot(data, aes(x = horizon, y = mean)) +
  geom_line(color = 'blue', size = 4) +           # Line for the forecast mean
  geom_point(color = 'red', size = 8) +           # Points for the forecast mean
  geom_ribbon(aes(ymin = lower, ymax = upper), alpha = 0.2) +  # Confidence interval ribbon
  labs(
    x = 'Forecast Horizon',
    y = 'Forecast Value',
    title = 'Probabilistic Forecast with 99% Confidence Interval'
  ) +
  theme_minimal() +                    # Clean theme
  theme(
    axis.title = element_text(size = 12),
```

```
    axis.text = element_text(size = 10),
    plot.title = element_text(size = 14, face = 'bold'),
    legend.text = element_text(size = 12),  # Increase legend text size
    legend.title = element_text(size = 12)  # Increase legend title size
  )
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```