Assignment: Forecasting Job Interest with Time Series Analysis

Emma Kruis

2020-01-11

Instructions

This assignment reviews the *Time Series Analysis* content. You will use the *time_series.Rmd* file I reviewed as part of the lectures for this week to complete this assignment. You will *copy and paste* relevant code from that file and update it to answer the questions in this assignment. You will respond to questions in each section after executing relevant code to answer a question. You will submit this assignment to its *Submissions* folder on *D2L*. You will submit *two* files:

1. this completed *R Markdown* script, and
2. as a first preference, a *PDF* (if you already installed `TinyTeX` properly), as a second preference, a *Microsfot Word* (if your computer has *Microsoft Word*) document, or, as a third preference, an *HTML* (if you did *not* install `TinyTeX` properly and your computer does *not* have *Microsoft Word*) file to *D2L*.

To start:

First, create a folder on your computer to save all relevant files for this course. If you did not do so already, you will want to create a folder named *mgt_592* that contains all of the materials for this course.

Second, inside of *mgt_592*, you will create a folder to host assignments. You can name that folder *assignments*.

Third, inside of *assignments*, you will create folders for each assignment. You can name the folder for this first assignment: *time_series*.

Fourth, create three additional folders in *time_series* named *scripts*, *data*, and *plots*. Store this script in the *scripts* folder and the data for this assignment in the *data* folder.

Fifth, go to the *File* menu in *RStudio*, select *New Project...*, choose *Existing Directory*, go to your *~/mgt_592/assignments/time_series* folder to select it as the top-level directory for this **R Project**.

Global Settings

The first code chunk sets the global settings for the remaining code chunks in the document. Do *not* change anything in this code chunk.

Load Packages

In this code chunk, we load the following packages:

1. **here**,
2. **tidyverse**,
3. **skimr**,

4. **flextable**,
5. **lubridate**,
6. **tidymodels**,
7. **timetk**,
8. **modeltime**, and
9. **modeltime.ensemble**.

Make sure you installed these packages when you reviewed the analytical lecture.

We will use functions from these packages to examine the data. Do *not* change anything in this code chunk.

```
### load libraries for use in current working session
## here for project work flow
library(here)

## tidyverse for data manipulation and plotting
## loads eight different libraries simultaneously
library(tidyverse)

## skimr to summarize data
library(skimr)

## flextable for creating tables
library(flextable)

## lubridate to work with dates
library(lubridate)

## tidymodels for modeling flow
library(tidymodels)

## timetk for time series data manipulation
library(timetk)

## modeltime for time series models
library(modeltime)

## modeltime.ensemble to combine time series models
library(modeltime.ensemble)
```

Task 1: Import Data

We will use the same data as in the analytical lecture: **job_interest_search.rds**. After you load the data, then you will execute other commands on the data.

Task 1.1

Use the **readRDS()** and **here()** functions to load the data file for this working session from the project **data** folder. Save the data as the object **interest_raw**. Apply **str()** to the list object.

Use **pluck()** to extract the **interest_over_time** element from the **interest_raw** list. Use **slice_tail()** to view the last **7** rows of the data table.

Questions 1.1: Answer these questions: (1) How many top-level list elements are there in **interest_raw**? (2) What *geography* (**geo**) are the last *seven* rows? (3) What is the *relative interest* (**hits**) value for the *November* date?

Responses 1.1: *(1) There are 7 top-level list elements in interest_raw; (2) AU; (3) 43* .

```
interest_raw <- readRDS(
  here("data", "job_interest_search.rds"))

str(interest_raw)
```

```
## List of 7
##  $ interest_over_time :'data.frame': 600 obs. of  7 variables:
##   ..$ date    : POSIXct[1:600], format: "2011-01-01" "2011-02-01" ...
##   ..$ hits    : int [1:600] 13 8 7 3 13 24 5 9 24 18 ...
##   ..$ keyword : chr [1:600] "people analytics" "people analytics" "people analytics" "people analytic
##   ..$ geo     : chr [1:600] "US" "US" "US" "US" ...
##   ..$ time    : chr [1:600] "2011-01-01 2020-12-01" "2011-01-01 2020-12-01" "2011-01-01 2020-12-01" "
##   ..$ gprop   : chr [1:600] "web" "web" "web" "web" ...
##   ..$ category: int [1:600] 0 0 0 0 0 0 0 0 0 0 ...
##  $ interest_by_country: NULL
##  $ interest_by_region :'data.frame': 126 obs. of  5 variables:
##   ..$ location: chr [1:126] "District of Columbia" "Massachusetts" "Virginia" "California" ...
##   ..$ hits    : int [1:126] 100 89 70 69 69 66 57 55 55 53 ...
##   ..$ keyword : chr [1:126] "people analytics" "people analytics" "people analytics" "people analytic
##   ..$ geo     : chr [1:126] "US" "US" "US" "US" ...
##   ..$ gprop   : chr [1:126] "web" "web" "web" "web" ...
##  $ interest_by_dma    :'data.frame': 210 obs. of  5 variables:
##   ..$ location: chr [1:210] "San Francisco-Oakland-San Jose CA" "Boston MA-Manchester NH" "Washingtor
##   ..$ hits    : int [1:210] 100 77 70 64 63 57 55 54 54 48 ...
##   ..$ keyword : chr [1:210] "people analytics" "people analytics" "people analytics" "people analytic
##   ..$ geo     : chr [1:210] "US" "US" "US" "US" ...
##   ..$ gprop   : chr [1:210] "web" "web" "web" "web" ...
##  $ interest_by_city   :'data.frame': 9 obs. of  5 variables:
##   ..$ location: chr [1:9] "San Francisco" "New York" "Chicago" "Mumbai" ...
##   ..$ hits    : int [1:9] 100 74 39 100 96 50 100 100 100
##   ..$ keyword : chr [1:9] "people analytics" "people analytics" "people analytics" "people analytics
##   ..$ geo     : chr [1:9] "US" "US" "US" "IN" ...
##   ..$ gprop   : chr [1:9] "web" "web" "web" "web" ...
##  $ related_topics     : NULL
##  $ related_queries    :'data.frame': 20 obs. of  6 variables:
##   ..$ subject        : chr [1:20] "100" "23" "12" "12" ...
##   ..$ related_queries: chr [1:20] "top" "top" "top" "top" ...
##   ..$ value          : chr [1:20] "google analytics" "what is people analytics" "people analytics jo
##   ..$ geo            : chr [1:20] "US" "US" "US" "US" ...
##   ..$ keyword        : chr [1:20] "people analytics" "people analytics" "people analytics" "people a
##   ..$ category       : int [1:20] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "reshapeLong")=List of 4
##   .. ..$ varying:List of 1
##   .. .. ..$ value: chr "top"
##   .. .. ..- attr(*, "v.names")= chr "value"
##   .. .. ..- attr(*, "times")= chr "top"
##   .. ..$ v.names: chr "value"
##   .. ..$ idvar  : chr "id"
```

```
##   .. ..$ timevar: chr "related_queries"
##  - attr(*, "class")= chr [1:2] "gtrends" "list"
```

```
interest_raw %>%
  pluck("interest_over_time") %>%
  slice_tail(n=7)
```

```
##         date hits          keyword geo                  time gprop category
## 1 2020-06-01   25 people analytics  AU 2011-01-01 2020-12-01   web        0
## 2 2020-07-01   38 people analytics  AU 2011-01-01 2020-12-01   web        0
## 3 2020-08-01   13 people analytics  AU 2011-01-01 2020-12-01   web        0
## 4 2020-09-01   13 people analytics  AU 2011-01-01 2020-12-01   web        0
## 5 2020-10-01   14 people analytics  AU 2011-01-01 2020-12-01   web        0
## 6 2020-11-01   43 people analytics  AU 2011-01-01 2020-12-01   web        0
## 7 2020-12-01   45 people analytics  AU 2011-01-01 2020-12-01   web        0
```

Task 2: Clean and Prepare Data

For this task, you will clean and prepare the data.

Task 2.1

Create a new **tibble** named **interest_work** from **interest_raw** in a single chained command with the following steps:

1. *pluck* the **interest_over_time** element from **interest_raw**,
2. convert to a *tibble*,
3. *select* **date**, **hits**, and **geo** variables,
4. *mutate* **date** with **ymd()**, change **geo** to a *factor* variable and recode its levels to full country names, and
5. *rename* **hits** to **rel_interest** and **geo** to **country**.

After creating **interest_work**, *arrange* the rows to view top **rel_interest** values.

Questions 2.1: Answer these questions: (1) What *country* has the *high relative interest* value for people analytics? (2) What is the *highest relative interest* value for *India*?

Responses 2.1: *(1) Australia ; (2) 70.*

```
interest_work <- interest_raw %>%
  pluck("interest_over_time") %>%
  as_tibble() %>%
  select(date, hits, geo) %>%
  mutate(
    date = ymd(date),
    geo = as_factor(geo),
    geo = fct_recode(
      geo,
      # USA
      "United States of America" = "US",
      # India
      "India" = "IN",
```

```
      # Great Britain
      "Great Britain" = "GB",
      # Australia
      "Australia" = "AU",
      # Brazil
      "Brazil" = "BR"
      )
    ) %>%
  rename(
    rel_interest = hits,
    country = geo
    )

##preview
interest_work %>%
  arrange(desc(rel_interest))
```

```
## # A tibble: 600 x 3
##    date       rel_interest country
##    <date>            <int> <fct>
##  1 2017-02-01          100 Australia
##  2 2018-02-01           89 Great Britain
##  3 2016-06-01           82 Australia
##  4 2014-11-01           73 Australia
##  5 2020-05-01           70 India
##  6 2016-12-01           68 Australia
##  7 2020-07-01           66 Great Britain
##  8 2019-10-01           66 Australia
##  9 2017-01-01           63 Great Britain
## 10 2020-04-01           62 India
## # ... with 590 more rows
```

Task 3: Examine Data

For this task, you will examine the data.

Task 3.1

Summarize **interest_work** by:

1. grouping by *country*,
2. selecting **rel_interest**, and
3. applying **skim_without_charts()**.

Questions 3.1: Answer these questions: (1) Which *country* has the *highest median relative interest* value? (2) Which *country* has the *smallest range* from smallest to highest *relative interset* value?

Responses 3.1: *(1) The country with the highest median relative interest value is Great Britain . (2) The country with the smallest range from smallest to highest relative interest is India..*

```
interest_work %>%
  group_by(country) %>%
  select(rel_interest) %>%
  skim_without_charts()
```

```
## Adding missing grouping variables: `country`
```

Table 1: Data summary

Name	Piped data
Number of rows	600
Number of columns	2
Column type frequency:	
numeric	1
Group variables	country

Variable type: numeric

skim_variable	country	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100
rel_interest	United States of America	0	1	25.49	10.24	3	18	25.0	32.25	54
rel_interest	India	0	1	18.89	13.55	0	10	18.0	24.00	70
rel_interest	Great Britain	0	1	28.62	15.87	0	17	27.0	39.00	89
rel_interest	Brazil	0	1	9.90	13.10	0	0	2.5	18.00	54
rel_interest	Australia	0	1	24.54	20.98	0	0	21.0	38.25	100

Task 3.2

Plot **interest_work** with **plot_time_series()** by specifying:

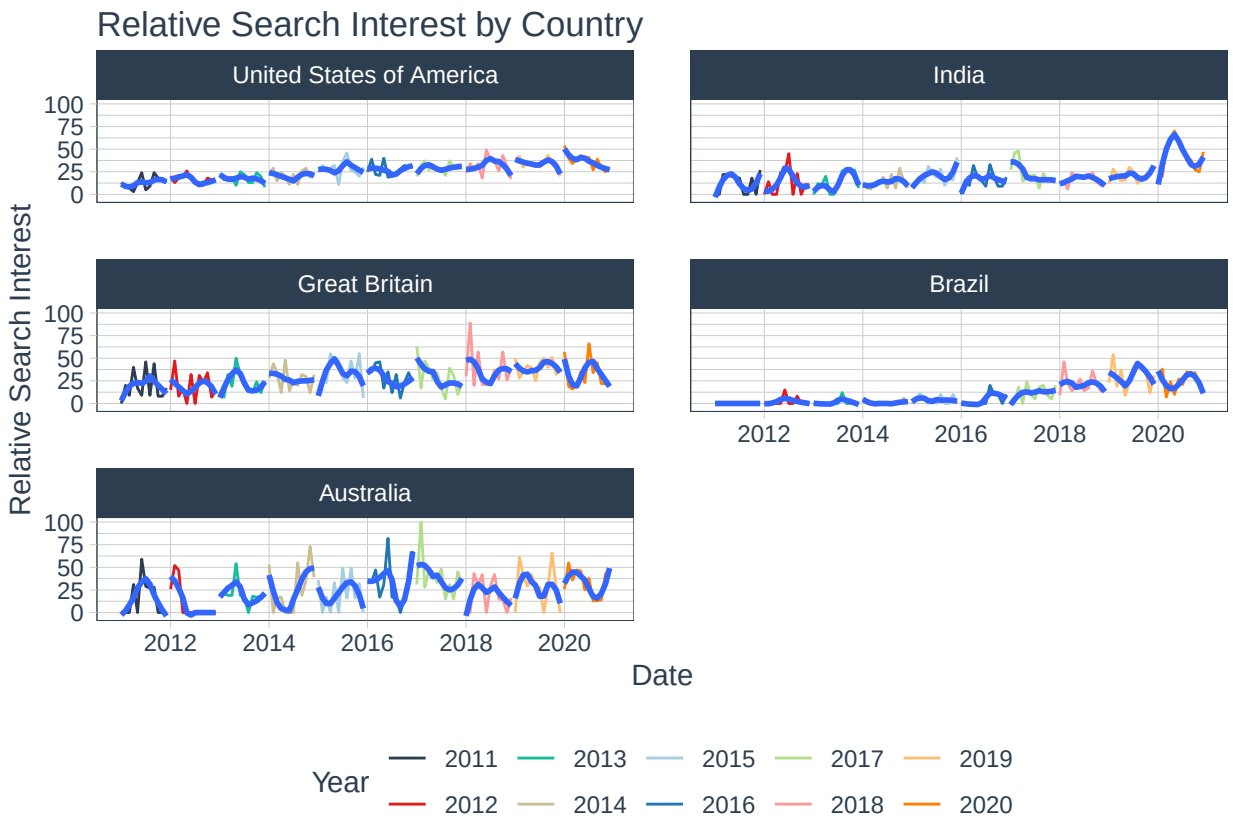
1. **date** as the *date* variable,
2. **rel_interest** as the *value* variable,
3. **country** as the *facet* variable and fixing the *scales* of the facets and creating *two* columns of facets,
4. choosing the *year* of the *date* variable as the *color* variable,
5. labeling the x-axis, y-axis, and legend and providing an appropriate title, and
6. creating a static plot.

Questions 3.2: Answer these questions: (1) Approximately, during which *year* did *Brazil* increase its search interest in people analytics? (2) During which *year* did *India* have the biggest spike in search interest for people analytics? (3) Has each country generally *increased* its search interest for people analytics over the years?

Responses 3.2: *(1) 2018 (2) 2020 (3) yes.*

```
interest_work %>%
  plot_time_series(
    .date_var = date,
```

```
    .value = rel_interest,
    .facet_vars = country,
    .facet_scales = "fixed",
    .facet_ncol = 2,
    .color_var = year(date),
    .x_lab = "Date",
    .y_lab = "Relative Search Interest",
    .color_lab = "Year",
    .title = "Relative Search Interest by Country",
    .interactive = FALSE)
```



Task 3.3

Plot **interest_work** with **plot_anomaly_diagnostics()** by specifying:

1. **date** as the *date* variable,
2. **rel_interest** as the *value* variable,
3. **country** as the *facet* variable and creating *two* columns of facets, and
4. creating a static plot.

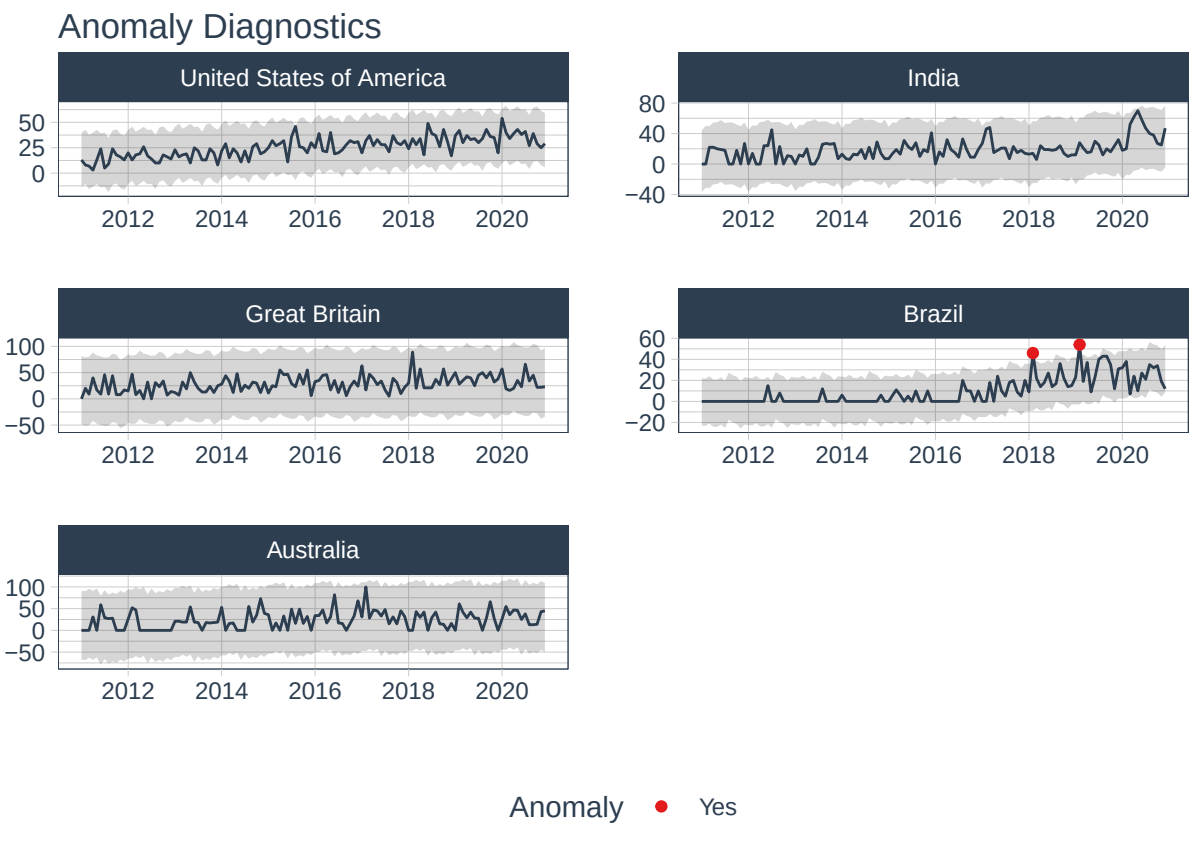
Question 3.3: Which *country* has anomalies?

Response 3.3: *Brazil has anomalies.*

```
interest_work %>%
  plot_anomaly_diagnostics(
```

```
    .date_var = date,
    .value = rel_interest,
    .facet_vars = country,
    .facet_ncol = 2,
    .interactive = FALSE
)
```

```
## frequency = 12 observations per 1 year

## trend = 60 observations per 5 years

## frequency = 12 observations per 1 year

## trend = 60 observations per 5 years

## frequency = 12 observations per 1 year

## trend = 60 observations per 5 years

## frequency = 12 observations per 1 year

## trend = 60 observations per 5 years

## frequency = 12 observations per 1 year

## trend = 60 observations per 5 years
```



Task 3.4

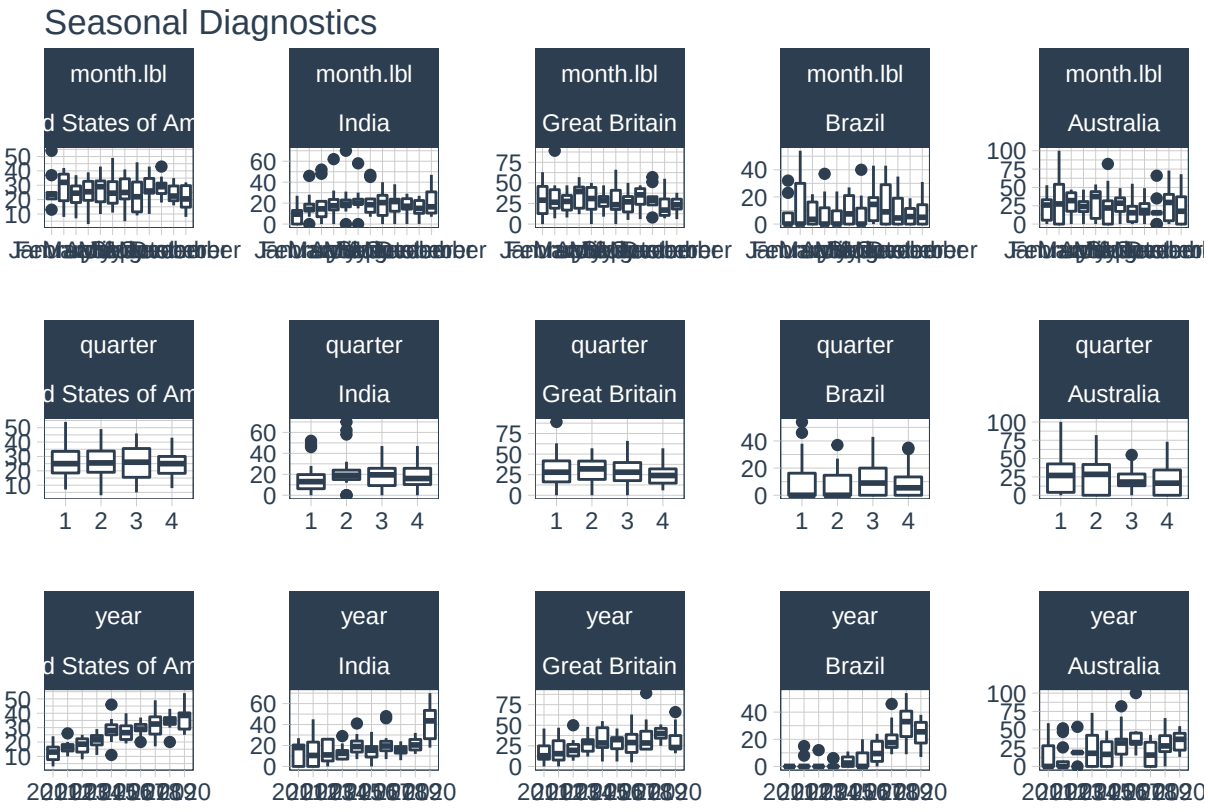
Plot **interest_work** with **plot_seasonal_diagnostics()** by specifying:

1. **date** as the *date* variable,
2. **rel_interest** as the *value* variable,
3. **country** as the *facet* variable, and
4. creating a static plot.

Questions 3.4: Answer these questions: (1) Does *Great Britain* have outliers for *December*? (2) Which *quarter* does *Australia* have an outlier? (3) Is the *median* search interest for people analytics for *Brazil* greater in *2019* or *2020*?

Responses 3.4: *(1) No (2) Quarter 3 (3) 2019.*

```
interest_work %>%
  plot_seasonal_diagnostics(
    .date_var = date,
    .value = rel_interest,
    .facet_vars = country,
    .interactive = FALSE)
```



Task 3.5

Group **interest_work** by **country** with **plot_acf_diagnostics()** by specifying:

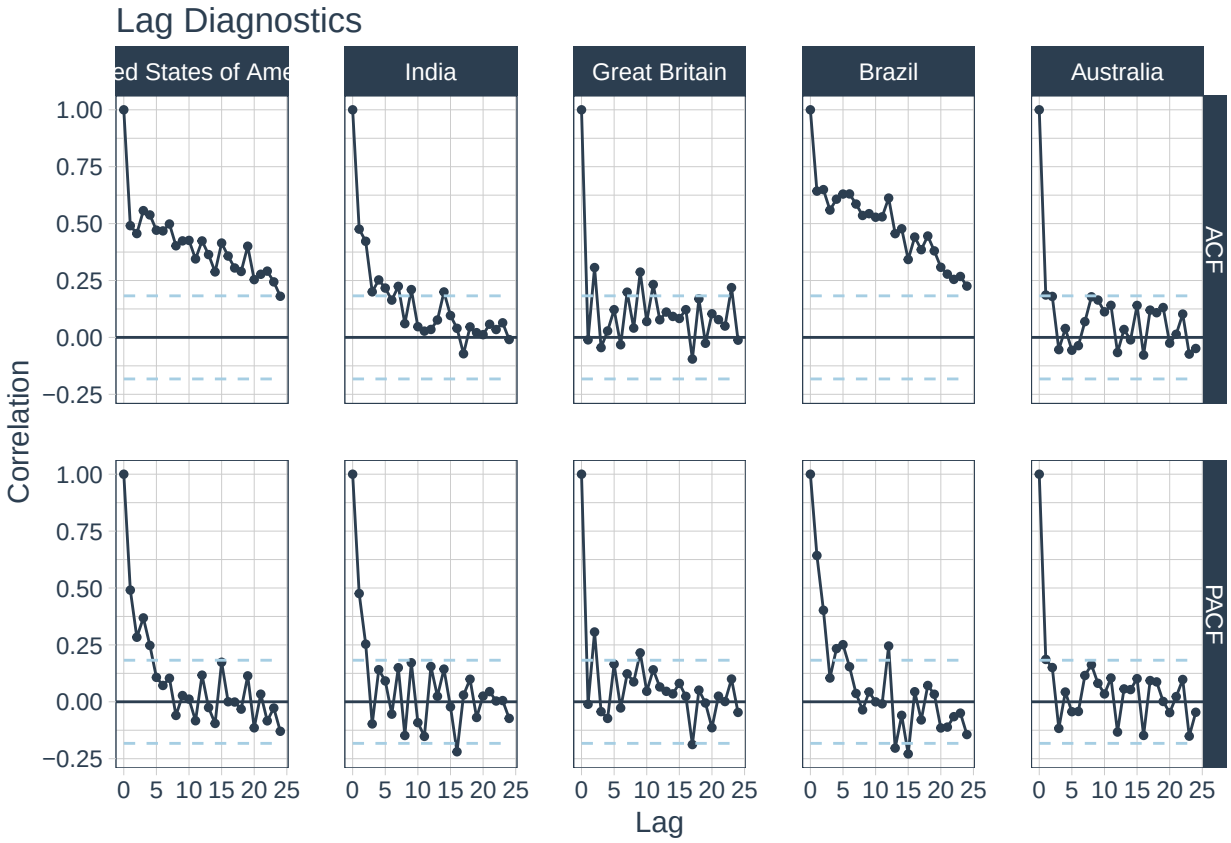
1. **date** as the *date* variable,

2. **rel_interest** as the *value* variable,
3. set the *lag* to *2 years*,
4. show the *white noise* bars, and
5. creating a static plot.

Questions 3.5: Answer these questions: (1) Which *two countries* show *high autocorrelations* values across lags? (2) Which *country* has the *smallest lag-one autocorrelation*? (3) Which *country* has the *highest lag-two partial autocorrelation*?

Responses 3.5: *(1)United States and Brazil (2) Great Britain (3) Brazil.*

```
interest_work %>%
  group_by(country) %>%
  plot_acf_diagnostics(
    .date_var = date,
    .value = rel_interest,
    .lags = "2 years",
    .show_white_noise_bars = TRUE,
    .interactive = FALSE)
```



Task 4: Time Series Validation

For this task, you will create a validation plan for one time series.

Task 4.1

Create a *data table* from **interest_work** consisting of only the time series for *Great Britain* using **filter()**. Name the data table **gb_ts**.

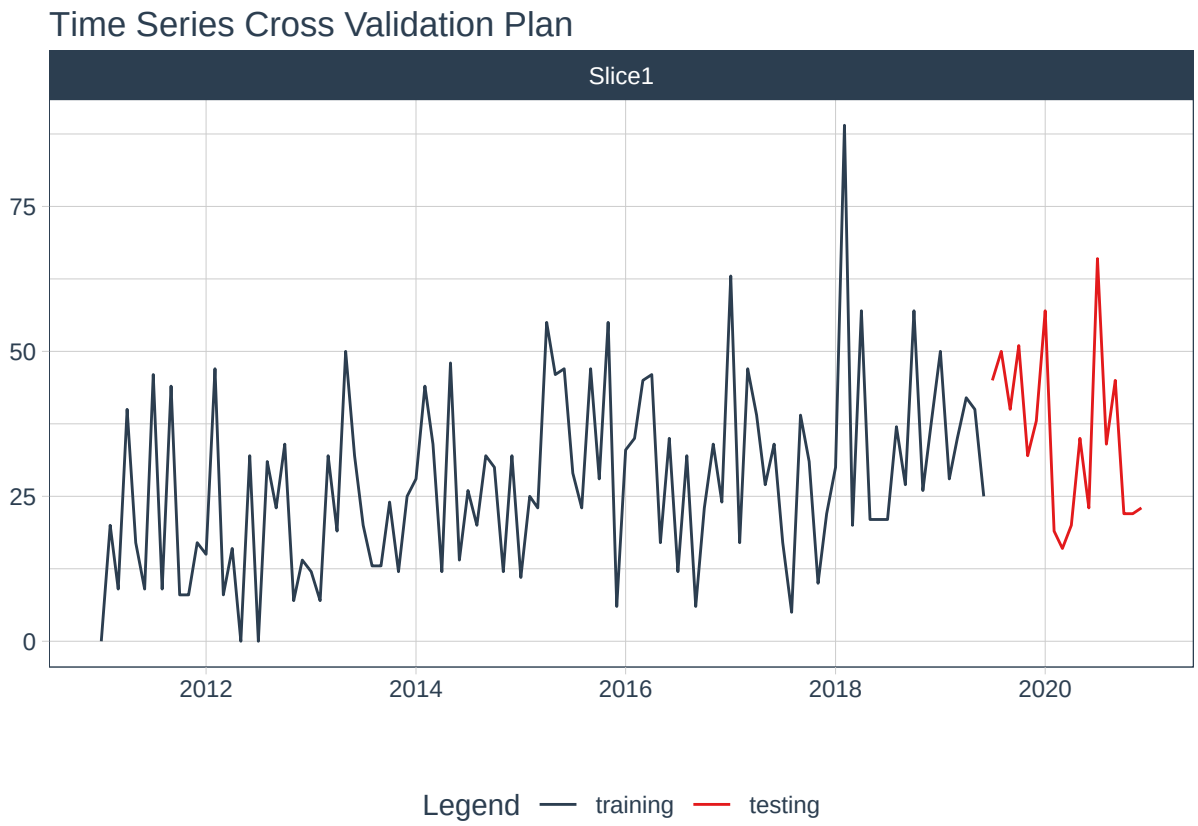
Then, create a validation split object for **gb_ts** using **time_series_split()**. Set the *date* variable, **assess** to **18 months**, and **cumulative** to **TRUE**. Name the object **data_split**.

Visualize the validation split by applying **tk_time_series_cv_plan()** and **plot_time_series_cv_plan()** to **data_split**. Set the plot to *interactive* mode.

Question 4.1: On what exact *month* and *year* is the data split?

Response 4.1: *June 2019*.

```
gb_ts <- interest_work %>%
  filter(country == "Great Britain")

data_split <- gb_ts %>%
  time_series_split(
    date_var = date,
    assess = "18 months",
    cumulative = TRUE)

data_split %>%
  tk_time_series_cv_plan() %>%
  plot_time_series_cv_plan(
    .date_var = date,
    .value = rel_interest,
    .interactive = FALSE)
```



Task 5: Prepare Model Features

For this task, you will compute featues based on the *date* variable.

Task 5.1

Create a *recipe* named **recipe_spec** by:

1. calling **recipe()** and setting the *formula* input to **rel_interest ~ date** and the *data* input to **training(data_split)**,
2. adding **date** features with **step_timeseries_signature()**,
3. removing unnecessary features using **step_rm()** and an appropriate *regular expression* inside of **matches()**,
4. normalizing the **date_index.num** and **date_year** features with **step_normalize()**, and
5. one-hot encoding all factor variables with **step_dummy()**.

Create a features data table named **model_features** by applying **prep()** and **bake(new_data = NULL)** to **recipe_spec**. Preview **model_features** with **glimpse()**

Questions 5.1: Answer these questions: (1) How many *columns* are there in **model_features**? (2) Explain what the values in **date_month.lbl_05** indicate?

Responses 5.1: *(1) 19 (2) Indicate what month it is. Therefore 05 is May which is why it has a 1 in the 5th column..*

```
recipe_spec <-
  recipe(rel_interest ~ date, training(data_split)) %>%
  step_timeseries_signature(date) %>%
  step_rm(
    matches("(.iso$)|(.xts$)|(week)|(day)|(hour)|(minute)|(second)|(am.pm)")
  ) %>%
  step_normalize(date_index.num, date_year) %>%
  step_dummy(all_nominal(), one_hot = TRUE)

model_features <- recipe_spec %>%
  prep() %>%
  bake(new_data = NULL)

glimpse(model_features)
```

```
## Rows: 102
## Columns: 19
## $ date              <date> 2011-01-01, 2011-02-01, 2011-03-01, 2011-04-01, 201~
## $ rel_interest      <int> 0, 20, 9, 40, 17, 9, 46, 9, 44, 8, 8, 17, 15, 47, 8,~
## $ date_index.num    <dbl> -1.7060722, -1.6716510, -1.6405608, -1.6061396, -1.5~
## $ date_year         <dbl> -1.5215055, -1.5215055, -1.5215055, -1.5215055, -1.5~
## $ date_half         <int> 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1~
## $ date_quarter      <int> 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 1, 1, 1, 2, 2, 2~
## $ date_month        <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 1, 2, 3, 4, 5~
## $ date_month.lbl_01 <dbl> 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0~
## $ date_month.lbl_02 <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0~
## $ date_month.lbl_03 <dbl> 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0~
## $ date_month.lbl_04 <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0~
```

```
## $ date_month.lbl_05 <dbl> 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0~
## $ date_month.lbl_06 <dbl> 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1~
## $ date_month.lbl_07 <dbl> 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ date_month.lbl_08 <dbl> 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0~
## $ date_month.lbl_09 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0~
## $ date_month.lbl_10 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0~
## $ date_month.lbl_11 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0~
## $ date_month.lbl_12 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0~
```

Task 6: Time Series Models

For this task, you will estimate a set of time series models.

Task 6.1

Estimate an *exponential smoothing* model named **wrkflw_fit_ets** by:

1. calling **workflow()**,
2. using **add_model()** to call for a *exponential smoothing* specification and estimator,
3. using **add_recipe()**, **recipe_spec**, and **step_rm()** to select only the **date** variable as a feature,
4. using **fit()** to estimate on **training(data_split)**.

View the estimated model.

Questions 6.1: Answer these questions: (1) What is the *initial state* of the *level* (**l**)? (2) What is the *smoothing parameter* for the *trend* (**beta**)? (3) Is the *trend additive* or *multiplicative*? (4) Is there a *seasonality* component?

Responses 6.1: *(1) 13.4164 (2) 1e-04 (3) Additive (4) No seasonality.*

```
wrkflw_fit_ets <- workflow() %>%
  add_model(
    exp_smoothing() %>%
      set_engine(engine = "ets")
  ) %>%
  add_recipe(
    recipe_spec %>%
      step_rm(
        all_predictors(),
        -date) ) %>%
  fit(training(data_split))
```

```
## frequency = 12 observations per 1 year
```

Task 6.2

Estimate an *ARIMA* model named **wrkflw_fit_arima** by:

1. calling **workflow()**,
2. using **add_model()** to call for a *ARIMA* specification and estimator,
3. using **add_recipe()**, **recipe_spec**, and **step_rm()** to select only the **date** variable as a feature,

4. using **fit()** to estimate on **training(data_split)**.

View the estimated model.

Estimate a *boosted ARIMA* model named **wrkflw_fit_arima_boost** by:

1. calling **workflow()**,
2. using **add_model()** to call for a *boosted ARIMA* specification and estimator,
3. using **add_recipe()** and **recipe_spec** to select all features, and
4. using **fit()** to estimate on **training(data_split)**.

View the estimated model.

Questions 6.2: Answer these questions: (1) Is there any difference in the estimated *ARIMA* of the two models? (2) What is the estimate of the *second autoregressive* parameter? (3) What is the estimate of the *first moving average* parameter?

Responses 6.2: *(1) No there is no difference (2) 0.2104 (3) -0.9353.*

```
wrkflw_fit_arima <- workflow() %>%
  add_model(
    arima_reg() %>%
      set_engine(engine = "auto_arima")
  ) %>%
  add_recipe(
    recipe_spec %>%
      step_rm(
        all_predictors(),
        -date
        ) ) %>%
  fit(training(data_split))
```

```
## frequency = 12 observations per 1 year
```

```
wrkflw_fit_arima
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: arima_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 5 Recipe Steps
##
## * step_timeseries_signature()
## * step_rm()
## * step_normalize()
## * step_dummy()
## * step_rm()
##
## -- Model -----------------------------------------------------------------------
## Series: outcome
## ARIMA(2,1,1)
##
```

```
## Coefficients:
##           ar1     ar2      ma1
##       -0.1389  0.2104  -0.9353
## s.e.   0.1042  0.1032   0.0327
##
## sigma^2 estimated as 227.1:  log likelihood=-416.82
## AIC=841.64   AICc=842.06   BIC=852.1
```

```
wrkflw_fit_arima_boost <- workflow() %>%
  add_model(
    arima_boost() %>%
      set_engine(engine = "auto_arima_xgboost")
  ) %>%
  add_recipe(
    recipe_spec
  ) %>%
  fit(training(data_split))
```

```
## frequency = 12 observations per 1 year
```

```
wrkflw_fit_arima_boost
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: arima_boost()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_timeseries_signature()
## * step_rm()
## * step_normalize()
## * step_dummy()
##
## -- Model -----------------------------------------------------------------------
## ARIMA(2,1,1) w/ XGBoost Errors
## ---
## Model 1: Auto ARIMA
## Series: outcome
## ARIMA(2,1,1)
##
## Coefficients:
##           ar1     ar2      ma1
##       -0.1389  0.2104  -0.9353
## s.e.   0.1042  0.1032   0.0327
##
## sigma^2 estimated as 227.1:  log likelihood=-416.82
## AIC=841.64   AICc=842.06   BIC=852.1
##
## ---
## Model 2: XGBoost Errors
##
## xgboost::xgb.train(params = list(eta = 0.3, max_depth = 6, gamma = 0,
```

```
##     colsample_bytree = 1, min_child_weight = 1, subsample = 1,
##     objective = "reg:squarederror"), data = x$data, nrounds = 15,
##     watchlist = x$watchlist, verbose = 0, nthread = 1)
```

Task 6.3

Estimate an *prophet* model named **wrkflw_fit_prophet** by:

1. calling **workflow()**,
2. using **add_model()** to call for a *prophet* specification and estimator,
3. using **add_recipe()**, **recipe_spec**, and **step_rm()** to select only the **date** variable as a feature,
4. using **fit()** to estimate on **training(data_split)**.

View the estimated model.

Estimate a *boosted prophet* model named **wrkflw_fit_prophet_boost** by:

1. calling **workflow()**,
2. using **add_model()** to call for a *boosted ARIMA* specification and estimator,
3. using **add_recipe()** and **recipe_spec** to select all features, and
4. using **fit()** to estimate on **training(data_split)**.

View the estimated model.

Questions 6.3: Answer these questions: (1) What is the *growth* specification of the *prophet* model? (2) What is the *number of change points* (**n.changepoints**) specification of the *prophet* model?

Responses 6.3: *(1) linear (2) 25.*

```
##prophet
wrkflw_fit_prophet <- workflow() %>%
  add_model(
    prophet_reg() %>%
      set_engine(engine = "prophet")
    ) %>%
  add_recipe(
    recipe_spec %>%
      step_rm(
        all_predictors(),
        -date
        ) ) %>%
  fit(training(data_split))
```

```
## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
wrkflw_fit_prophet
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: prophet_reg()
```

```
##
## -- Preprocessor ----------------------------------------------------------
## 5 Recipe Steps
##
## * step_timeseries_signature()
## * step_rm()
## * step_normalize()
## * step_dummy()
## * step_rm()
##
## -- Model -----------------------------------------------------------------
## PROPHET Model
## - growth: 'linear'
## - n.changepoints: 25
## - changepoint.range: 0.8
## - yearly.seasonality: 'auto'
## - weekly.seasonality: 'auto'
## - daily.seasonality: 'auto'
## - seasonality.mode: 'additive'
## - changepoint.prior.scale: 0.05
## - seasonality.prior.scale: 10
## - holidays.prior.scale: 10
## - logistic_cap: NULL
## - logistic_floor: NULL
## - extra_regressors: 0
```

```
##prophet boost
wrkflw_fit_prophet_boost <- workflow() %>%
  add_model(
    prophet_boost() %>%
      set_engine(engine = "prophet_xgboost")
  ) %>%
  add_recipe(
    recipe_spec
    ) %>%
  fit(training(data_split))
```

```
## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
wrkflw_fit_prophet_boost
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: prophet_boost()
##
## -- Preprocessor ----------------------------------------------------------
## 4 Recipe Steps
##
## * step_timeseries_signature()
## * step_rm()
## * step_normalize()
## * step_dummy()
```

```
##
## -- Model ---------------------------------------------------------------------
## PROPHET w/ XGBoost Errors
## ---
## Model 1: PROPHET
##  - growth: 'linear'
##  - n.changepoints: 25
##  - changepoint.range: 0.8
##  - yearly.seasonality: 'auto'
##  - weekly.seasonality: 'auto'
##  - daily.seasonality: 'auto'
##  - seasonality.mode: 'additive'
##  - changepoint.prior.scale: 0.05
##  - seasonality.prior.scale: 10
##  - holidays.prior.scale: 10
##  - logistic_cap: NULL
##  - logistic_floor: NULL
##
## ---
## Model 2: XGBoost Errors
##
## xgboost::xgb.train(params = list(eta = 0.3, max_depth = 6, gamma = 0,
##     colsample_bytree = 1, min_child_weight = 1, subsample = 1,
##     objective = "reg:squarederror"), data = x$data, nrounds = 15,
##     watchlist = x$watchlist, verbose = 0, nthread = 1)
```

Task 7: Evaluate Accuracy of Models

For this task, you will evaluate the accuracy of estimated models.

Task 7.1

Create a models table named **models_tbl** consisting of the five estimated models using **modeltime_table()**. Then, create an *equally-weighted ensemble* named **ensemble_set** from the models in **models_tbl** with **ensemble_average()**. Create a new models table named **ensemble_tbl** that incorporates the ensemble model applying **modeltime_table()** on **ensemble_set** and **combine_modeltime_tables()** on **models_tbl**. Then, calibrate all six models with the testing data using **modeltime_calibrate()** and name the result **models_calibrate**. Use **unnest()** on the **.calibration_data** column in **models_calibrate** and print all rows.

Questions 7.1: Answer these questions: (1) What is the *ensemble* prediction for *2020-02-01*? (2) How *wrong* is the *prophet* prediction for *2020-06-01*?

Responses 7.1: *(1) 37.7 (2) it was wrong by -11.5.*

```
models_tbl <- modeltime_table(
  wrkflw_fit_ets,
  wrkflw_fit_arima,
  wrkflw_fit_arima_boost,
  wrkflw_fit_prophet,
  wrkflw_fit_prophet_boost)
```

```
ensemble_set <- models_tbl %>%
  ensemble_average(type = "mean")

ensemble_tbl <- modeltime_table(
  ensemble_set
) %>%
  combine_modeltime_tables(models_tbl)


models_calibrate <- ensemble_tbl %>%
  modeltime_calibrate(
    testing(data_split)
    )

models_calibrate %>%
  unnest(.calibration_data) %>%
  print(n = Inf)
```

```
## # A tibble: 108 x 8
##     .model_id .model .model_desc .type date       .actual .prediction .residuals
##         <int> <list> <chr>       <chr> <date>       <dbl>       <dbl>      <dbl>
##   1         1 <ense~ ENSEMBLE (~ Test  2019-07-01      45        31.9     13.1
##   2         1 <ense~ ENSEMBLE (~ Test  2019-08-01      50        30.9     19.1
##   3         1 <ense~ ENSEMBLE (~ Test  2019-09-01      40        35.2      4.81
##   4         1 <ense~ ENSEMBLE (~ Test  2019-10-01      51        39.6     11.4
##   5         1 <ense~ ENSEMBLE (~ Test  2019-11-01      32        31.3      0.723
##   6         1 <ense~ ENSEMBLE (~ Test  2019-12-01      38        31.9      6.14
##   7         1 <ense~ ENSEMBLE (~ Test  2020-01-01      57        37.4     19.6
##   8         1 <ense~ ENSEMBLE (~ Test  2020-02-01      19        37.7    -18.7
##   9         1 <ense~ ENSEMBLE (~ Test  2020-03-01      16        37.1    -21.1
##  10         1 <ense~ ENSEMBLE (~ Test  2020-04-01      20        41.8    -21.8
##  11         1 <ense~ ENSEMBLE (~ Test  2020-05-01      35        38.7     -3.66
##  12         1 <ense~ ENSEMBLE (~ Test  2020-06-01      23        34.5    -11.5
##  13         1 <ense~ ENSEMBLE (~ Test  2020-07-01      66        32.2     33.8
##  14         1 <ense~ ENSEMBLE (~ Test  2020-08-01      34        33.4      0.592
##  15         1 <ense~ ENSEMBLE (~ Test  2020-09-01      45        36.1      8.93
##  16         1 <ense~ ENSEMBLE (~ Test  2020-10-01      22        41.5    -19.5
##  17         1 <ense~ ENSEMBLE (~ Test  2020-11-01      22        32.7    -10.7
##  18         1 <ense~ ENSEMBLE (~ Test  2020-12-01      23        33.4    -10.4
##  19         2 <work~ ETS(A,A,N)  Test  2019-07-01      45        40.1      4.91
##  20         2 <work~ ETS(A,A,N)  Test  2019-08-01      50        40.4      9.65
##  21         2 <work~ ETS(A,A,N)  Test  2019-09-01      40        40.6     -0.607
##  22         2 <work~ ETS(A,A,N)  Test  2019-10-01      51        40.9     10.1
##  23         2 <work~ ETS(A,A,N)  Test  2019-11-01      32        41.1     -9.12
##  24         2 <work~ ETS(A,A,N)  Test  2019-12-01      38        41.4     -3.38
##  25         2 <work~ ETS(A,A,N)  Test  2020-01-01      57        41.6     15.4
##  26         2 <work~ ETS(A,A,N)  Test  2020-02-01      19        41.9    -22.9
##  27         2 <work~ ETS(A,A,N)  Test  2020-03-01      16        42.1    -26.1
##  28         2 <work~ ETS(A,A,N)  Test  2020-04-01      20        42.4    -22.4
##  29         2 <work~ ETS(A,A,N)  Test  2020-05-01      35        42.7     -7.66
##  30         2 <work~ ETS(A,A,N)  Test  2020-06-01      23        42.9    -19.9
##  31         2 <work~ ETS(A,A,N)  Test  2020-07-01      66        43.2     22.8
##  32         2 <work~ ETS(A,A,N)  Test  2020-08-01      34        43.4     -9.43
```

```
##  33        2 <work~ ETS(A,A,N)  Test  2020-09-01      45        43.7      1.31
##  34        2 <work~ ETS(A,A,N)  Test  2020-10-01      22        43.9    -21.9
##  35        2 <work~ ETS(A,A,N)  Test  2020-11-01      22        44.2    -22.2
##  36        2 <work~ ETS(A,A,N)  Test  2020-12-01      23        44.5    -21.5
##  37        3 <work~ ARIMA(2,1,~ Test  2019-07-01      45        36.2      8.75
##  38        3 <work~ ARIMA(2,1,~ Test  2019-08-01      50        31.5     18.5
##  39        3 <work~ ARIMA(2,1,~ Test  2019-09-01      40        34.5      5.45
##  40        3 <work~ ARIMA(2,1,~ Test  2019-10-01      51        33.1     17.9
##  41        3 <work~ ARIMA(2,1,~ Test  2019-11-01      32        34.0     -1.97
##  42        3 <work~ ARIMA(2,1,~ Test  2019-12-01      38        33.6      4.44
##  43        3 <work~ ARIMA(2,1,~ Test  2020-01-01      57        33.8     23.2
##  44        3 <work~ ARIMA(2,1,~ Test  2020-02-01      19        33.7    -14.7
##  45        3 <work~ ARIMA(2,1,~ Test  2020-03-01      16        33.7    -17.7
##  46        3 <work~ ARIMA(2,1,~ Test  2020-04-01      20        33.7    -13.7
##  47        3 <work~ ARIMA(2,1,~ Test  2020-05-01      35        33.7      1.28
##  48        3 <work~ ARIMA(2,1,~ Test  2020-06-01      23        33.7    -10.7
##  49        3 <work~ ARIMA(2,1,~ Test  2020-07-01      66        33.7     32.3
##  50        3 <work~ ARIMA(2,1,~ Test  2020-08-01      34        33.7      0.288
##  51        3 <work~ ARIMA(2,1,~ Test  2020-09-01      45        33.7     11.3
##  52        3 <work~ ARIMA(2,1,~ Test  2020-10-01      22        33.7    -11.7
##  53        3 <work~ ARIMA(2,1,~ Test  2020-11-01      22        33.7    -11.7
##  54        3 <work~ ARIMA(2,1,~ Test  2020-12-01      23        33.7    -10.7
##  55        4 <work~ ARIMA(2,1,~ Test  2019-07-01      45        29.4     15.6
##  56        4 <work~ ARIMA(2,1,~ Test  2019-08-01      50        28.7     21.3
##  57        4 <work~ ARIMA(2,1,~ Test  2019-09-01      40        31.2      8.84
##  58        4 <work~ ARIMA(2,1,~ Test  2019-10-01      51        53.1     -2.12
##  59        4 <work~ ARIMA(2,1,~ Test  2019-11-01      32        28.6      3.37
##  60        4 <work~ ARIMA(2,1,~ Test  2019-12-01      38        28.2      9.78
##  61        4 <work~ ARIMA(2,1,~ Test  2020-01-01      57        47.8      9.25
##  62        4 <work~ ARIMA(2,1,~ Test  2020-02-01      19        33.8    -14.8
##  63        4 <work~ ARIMA(2,1,~ Test  2020-03-01      16        34.7    -18.7
##  64        4 <work~ ARIMA(2,1,~ Test  2020-04-01      20        43.1    -23.1
##  65        4 <work~ ARIMA(2,1,~ Test  2020-05-01      35        36.3     -1.35
##  66        4 <work~ ARIMA(2,1,~ Test  2020-06-01      23        25.3     -2.30
##  67        4 <work~ ARIMA(2,1,~ Test  2020-07-01      66        26.8     39.2
##  68        4 <work~ ARIMA(2,1,~ Test  2020-08-01      34        30.9      3.09
##  69        4 <work~ ARIMA(2,1,~ Test  2020-09-01      45        30.3     14.7
##  70        4 <work~ ARIMA(2,1,~ Test  2020-10-01      22        53.7    -31.7
##  71        4 <work~ ARIMA(2,1,~ Test  2020-11-01      22        28.4     -6.38
##  72        4 <work~ ARIMA(2,1,~ Test  2020-12-01      23        28.4     -5.38
##  73        5 <work~ PROPHET     Test  2019-07-01      45        31.7     13.3
##  74        5 <work~ PROPHET     Test  2019-08-01      50        31.2     18.8
##  75        5 <work~ PROPHET     Test  2019-09-01      40        39.6      0.409
##  76        5 <work~ PROPHET     Test  2019-10-01      51        39.5     11.5
##  77        5 <work~ PROPHET     Test  2019-11-01      32        30.5      1.50
##  78        5 <work~ PROPHET     Test  2019-12-01      38        32.2      5.77
##  79        5 <work~ PROPHET     Test  2020-01-01      57        37.9     19.1
##  80        5 <work~ PROPHET     Test  2020-02-01      19        46.2    -27.2
##  81        5 <work~ PROPHET     Test  2020-03-01      16        39.4    -23.4
##  82        5 <work~ PROPHET     Test  2020-04-01      20        47.4    -27.4
##  83        5 <work~ PROPHET     Test  2020-05-01      35        40.7     -5.68
##  84        5 <work~ PROPHET     Test  2020-06-01      23        39.4    -16.4
##  85        5 <work~ PROPHET     Test  2020-07-01      66        33.3     32.7
##  86        5 <work~ PROPHET     Test  2020-08-01      34        33.7      0.342
```

```
##  87         5 <work~ PROPHET     Test  2020-09-01      45        41.1      3.92
##  88         5 <work~ PROPHET     Test  2020-10-01      22        42.3    -20.3
##  89         5 <work~ PROPHET     Test  2020-11-01      22        32.8    -10.8
##  90         5 <work~ PROPHET     Test  2020-12-01      23        34.5    -11.5
##  91         6 <work~ PROPHET W/~ Test  2019-07-01      45        22.3     22.7
##  92         6 <work~ PROPHET W/~ Test  2019-08-01      50        22.9     27.1
##  93         6 <work~ PROPHET W/~ Test  2019-09-01      40        30.0      9.96
##  94         6 <work~ PROPHET W/~ Test  2019-10-01      51        31.2     19.8
##  95         6 <work~ PROPHET W/~ Test  2019-11-01      32        22.2      9.83
##  96         6 <work~ PROPHET W/~ Test  2019-12-01      38        23.9     14.1
##  97         6 <work~ PROPHET W/~ Test  2020-01-01      57        26.0     31.0
##  98         6 <work~ PROPHET W/~ Test  2020-02-01      19        32.9    -13.9
##  99         6 <work~ PROPHET W/~ Test  2020-03-01      16        35.4    -19.4
## 100         6 <work~ PROPHET W/~ Test  2020-04-01      20        42.5    -22.5
## 101         6 <work~ PROPHET W/~ Test  2020-05-01      35        39.9     -4.90
## 102         6 <work~ PROPHET W/~ Test  2020-06-01      23        31.0     -8.02
## 103         6 <work~ PROPHET W/~ Test  2020-07-01      66        23.9     42.1
## 104         6 <work~ PROPHET W/~ Test  2020-08-01      34        25.3      8.67
## 105         6 <work~ PROPHET W/~ Test  2020-09-01      45        31.5     13.5
## 106         6 <work~ PROPHET W/~ Test  2020-10-01      22        33.9    -11.9
## 107         6 <work~ PROPHET W/~ Test  2020-11-01      22        24.4     -2.43
## 108         6 <work~ PROPHET W/~ Test  2020-12-01      23        26.2     -3.18
```

Task 7.2

Create a plot named **models_calibrate_plot** to visualize the predictions in **models_calibrate**. Apply **modeltime_forecast()** and set **new_data** to **testing(data_split)** and **actual_data** to **gb_ts**. Then, apply **plot_modeltime_forecast()** with *interactive* mode set to **TRUE**. Display the plot.

Then, apply **modeltime_accuracy()** to **models_calibrate**. Apply **flextable()** with additional specifications to display the table in the **Viewer**.

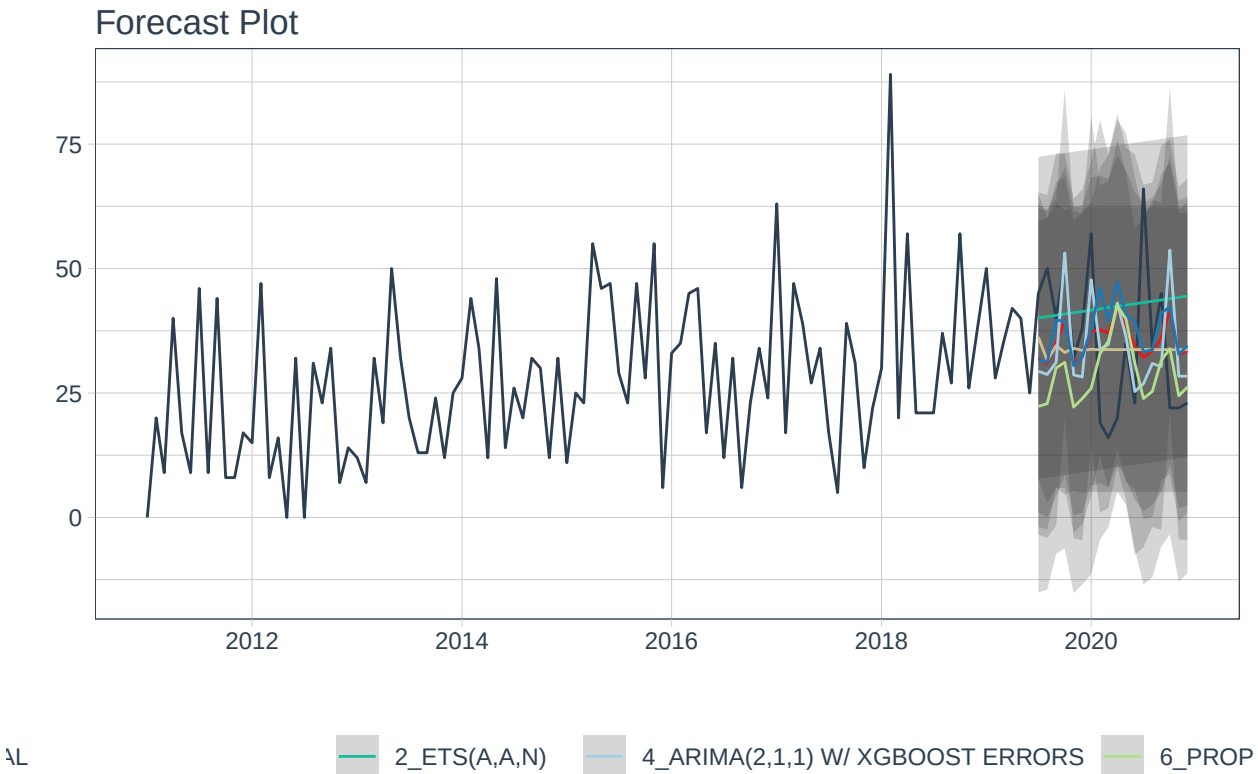
Questions 7.2: Answer these questions: (1) Describe the difference between the *prophet* and *boosted prophet* predictions using the interactive plot. (2) Describe the difference between the *ensemble*, *exponential smoothing*, and *ARIMA* predictions using the interactive plot. (3) What is the **mase** of the *boosted ARIMA*? (4) Based on **rmse**, which model performs the best?

Responses 7.2: *(1) The prophet predictions are more conservative than the boosted prophet predictions which have a wider range (2) the ARIMA predictions are a straight line. Similarly, the exponential smoothing predictions are a straight line. The ensemble predictions have a slightly larger range (3) 0.8652633 (4) The boosted ARIMA.*

```
models_calibrate_plot <- models_calibrate %>%
  modeltime_forecast(
    new_data = testing(data_split),
    actual_data = gb_ts
  ) %>%
  plot_modeltime_forecast(
    .interactive = FALSE
  )


models_calibrate_plot
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -
```

```
## Inf
```




```
models_calibrate %>%
  modeltime_accuracy() %>%
  flextable() %>%
  bold(part = "header") %>%
  bg(bg = "#D3D3D3", part = "header") %>%
  autofit()
```

```
## Warning: Warning: fonts used in ‘flextable‘ are ignored because the ‘pdflatex‘
## engine is used and not ‘xelatex‘ or ‘lualatex‘. You can avoid this warning
## by using the ‘set_flextable_defaults(fonts_ignore=TRUE)‘ command or use a
## compatible engine by defining ‘latex_engine: xelatex‘ in the YAML header of the
## R Markdown document.
```

.model_id	.model_desc	.type	mae	mape	m
1	ENSEMBLE (MEAN): 5 MODELS	Test	13.08603	44.97200	0.8827
2	ETS(A,A,N)	Test	13.96378	54.41689	0.9420
3	ARIMA(2,1,1)	Test	12.01427	38.79484	0.8104
4	ARIMA(2,1,1) W/ XGBOOST ERRORS	Test	12.82626	43.20285	0.8652
5	PROPHET	Test	13.88064	50.43162	0.9363
6	PROPHET W/ XGBOOST ERRORS	Test	15.84591	46.97096	1.0689

Task 8: Refit Models and Forecast

For this task, you will refit the models and forecast the future.

Task 8.1

Refit the models in **models_calibrate** with **modeltime_refit()** and the **data** input set to **gb_ts**. Save the refit models as **models_refit**.

Apply **modeltime_forecast()** to **models_refit** using **gb_ts** as the data to project *2 years* ahead with *85%* confidence intervals. Save the forecasts as **models_forecast**.

Create a plot named **models_forecast_plot** to visualize the predictions in **models_forecast**. Apply **plot_modeltime_forecast()** with *interactive* mode set to **TRUE**. Display the plot.

Questions 8.1: Answer these questions: (1) Describe the difference between the *prophet* and *boosted prophet* forecasts using the interactive plot. (2) Describe the difference between the *ensemble*, *exponential smoothing*, *ARIMA*, and *boosted ARIMA* forecasts using the interactive plot.

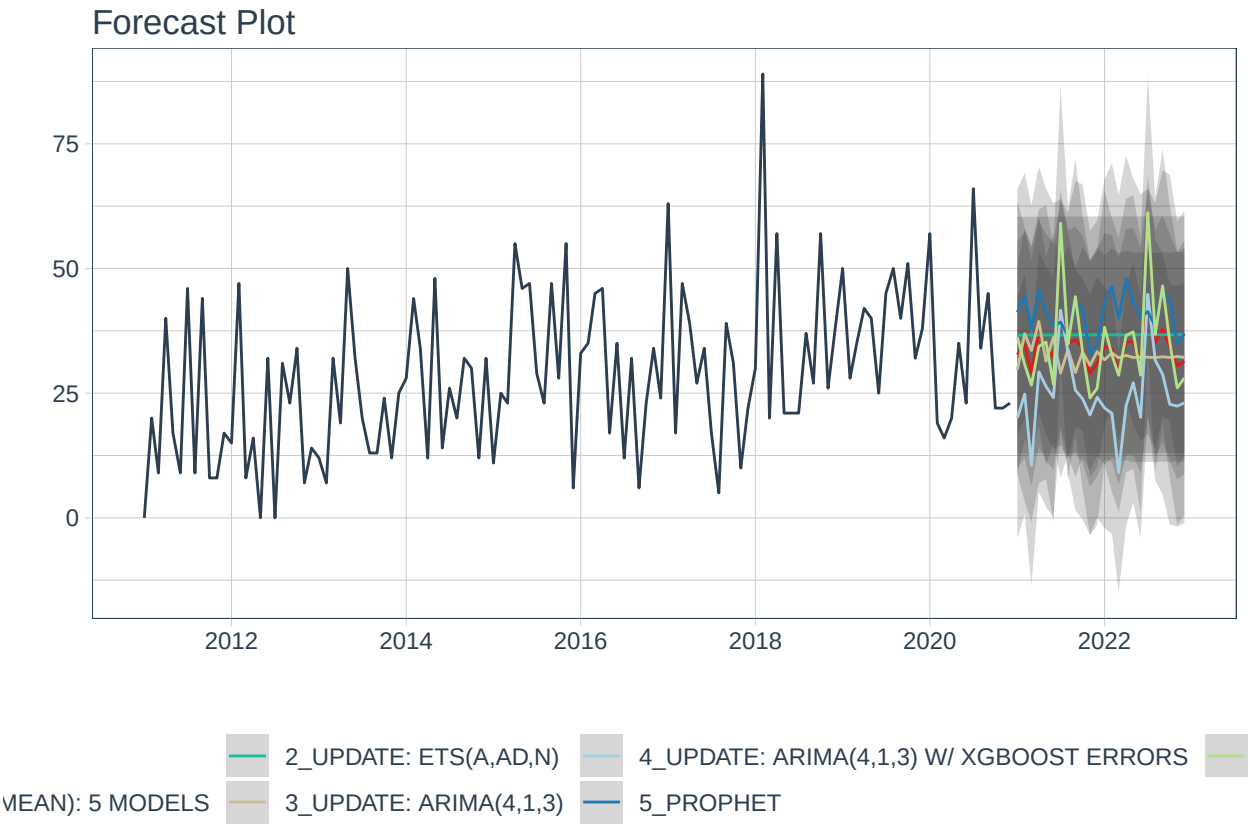
Responses 8.1: *(1)The prophet was higher that the boosted prohpet. The boosted prohphet also had a wider range of predictions (2)The exponential smoothing predictions were straight. The ARIMA predictions had a wide range of predictions but the smoothed out. The boosted ARIMA had the greatest range of predicitons. The ensemble had a wide range of predicitons but not as signifcant range as the boosted ARIMA predicitons..*

```
models_refit <- models_calibrate %>%
  modeltime_refit(
    data = gb_ts
    )
```

```
## frequency = 12 observations per 1 year
## frequency = 12 observations per 1 year
## frequency = 12 observations per 1 year

## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

## frequency = 12 observations per 1 year
## frequency = 12 observations per 1 year
## frequency = 12 observations per 1 year

## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.

## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override this.

## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override this.
```

```
models_forecast <- models_refit %>%
  modeltime_forecast(
    h = "2 years",
    actual_data = gb_ts,
    conf_interval = 0.85
  )

models_forecast_plot <- models_forecast %>%
  plot_modeltime_forecast(
    .interactive = FALSE
  )

models_forecast_plot
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -
## Inf
```



Task 9: Save Plots and Data

For this task, you will save the plots and the working data.

Task 9.1

Save the working data, **interest_work** as the data file: **interest_work.rds** in the **data** folder of the project directory using **saveRDS()**.

Save the two plot objects as **png** files in the **plots** folder of the project directory. Make sure to create the plots again by setting the *interactive* mode to **FALSE**. Save **models_calibrate_plot** as **models_calibrate.png** and **models_forecast_plot** as **models_calibrate.png**. Use a width of *9 inches* and height of *6 inches* for all plots.

```
saveRDS(
  interest_work,
  file = here("data", "interest_work.rds"))

ggsave(
  here("plots", "models_calibrate.png"),
  plot = models_calibrate_plot,
  units = "in", width = 9, height = 6)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -
## Inf
```

```
ggsave(
  here("plots", "models_forecast.png"),
  plot = models_forecast_plot,
  units = "in", width = 9, height = 6)
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning -
## Inf
```

Task 10: Conceptual Questions

For your last task, you will respond to conceptual questions based on the conceptual lectures for this week.

Question 10.1: For an *ARIMA(3, 1, 2)* model, answer these questions: (1) Was the the time series was differenced? (2) What model parameters were estimated for the time series?

Response 10.1: *(1) yes the time series was differenced (2) ARIMA(3, 1, 2) Autoregressive order is 3, Degree of Integration is 1, and Moving Average is 2..*

Question 10.2: What is the difference between an *autocorrelation* and a *partial autocorrelation*?

Response 10.2: *An autocorrelation is the relationship of an observation taking into account both the direct and indirect observations. in other words it is the linear relationship between lagged values of a time series. The partial auto correlation controls for the indirect observations. It is the linear relationship between lagged values of a time series after accounting for any intermediary lags.*

Question 10.3: Describe the process of an *additive* classical decomposition of a time series.

Response 10.3: *model time series (y) = seasonal component (s) + the trend cycle component (t) + the remainder component (rT); first compute the trend-cylce component with amoving average, next compute the detrendeding time series (y-t), then compute the seasonal component from detrended time series by averaging detrended values for that season, compute the regular component (y-t-s). .*