Midterm: Forecasting Job Interest with Time Series Analysis

Emma Kruis

2020-02-01

Instructions

This script reviews *Time Series Analysis* as part of the *Midterm Review*. You will use content from the lecture and assignment materials on *Time Series Analysis* to complete this script. You will *copy and paste* relevant code from those files into this script and answer the associated questions for each task. You will respond to questions in each section after executing relevant code to answer a question. You will submit this script to its *Submissions* folder on *D2L* as part of the *Midterm Review*. For this script, 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*.

For the *Midterm Review*, create the project directory: *~/mgt_592/assignments/midterm_review*. Convert your project directory into a formal *R Project* directory by going to the *File* menu in *RStudio*, selecting *New Project...*, choosing *Existing Directory*, and going to your *~/mgt_592/assignments/midterm_review* folder to select it as the top-level directory for this **R Project**.
The project directory should contain the following folders: *scripts*, *data*, and *plots*. Store this script in the *scripts* folder and the relevant data in the *data* folder.

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.

Task 1: Load Libraries

For this task, you will load the libraries you need for this script.

Task 1.1

In this code chunk, 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 before loading the libraries.

You will use functions from these packages to complete this script.

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

```
## here() starts at /Users/emmakruis/Library/Mobile Documents/com~apple~CloudDocs/year_2/WQ21/mgt_592/as
```

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

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
## skimr to summarize data
library(skimr)

## flextable for creating tables
library(flextable)
```

```
##
## Attaching package: 'flextable'

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

```
## lubridate to work with dates
library(lubridate)
```

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

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

```
## tidymodels for modeling flow
library(tidymodels)
```

```
## -- Attaching packages --------------------------------------- tidymodels 0.1.2 --

## v broom     0.7.5      v recipes   0.1.15
## v dials     0.0.9      v rsample   0.0.9
## v infer     0.5.4      v tune      0.1.3
## v modeldata 0.1.0      v workflows 0.2.2
## v parsnip   0.1.5      v yardstick 0.0.7

## -- Conflicts ------------------------------------------ tidymodels_conflicts() --
## x flextable::compose() masks purrr::compose()
## x scales::discard()    masks purrr::discard()
## x dplyr::filter()      masks stats::filter()
## x recipes::fixed()     masks stringr::fixed()
## x dplyr::lag()         masks stats::lag()
## x yardstick::spec()    masks readr::spec()
## x recipes::step()      masks stats::step()
```

```
## 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)
```

```
## Loading required package: modeltime.resample
```

Task 2: Import Data

For this task, you will import the data file: **job_interest_search.rds**.

Task 2.1

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.

```
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"
```

Task 3: Clean and Prepare Data

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

Task 3.1

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**.

```
### clean and prepare data
interest_work <- interest_raw %>%
  ## select desired element from list
  pluck("interest_over_time") %>%
  ## convert to tibble object
  as_tibble() %>%
  ## select variables of interest
  select(date, hits, geo) %>%
  ## alter variables
  mutate(
    # change date format
    date = ymd(date),
    # convert to factor
    geo = as_factor(geo),
    # recode factor
    geo = fct_recode(
      # variable
      geo,
      # USA
      "United States of America" = "US",
      # India
      "India" = "IN",
      # Great Britain
      "Great Britain" = "GB",
      # Australia
      "Australia" = "AU",
      # Brazil
      "Brazil" = "BR"
    )
  ) %>%
  ## rename variables
  rename(
    # relative interest
    rel_interest = hits,
    # country
    country = geo
  )

## preview data
interest_work
```

```
## # A tibble: 600 x 3
##    date       rel_interest country
##    <date>            <int> <fct>
##  1 2011-01-01           13 United States of America
##  2 2011-02-01            8 United States of America
##  3 2011-03-01            7 United States of America
##  4 2011-04-01            3 United States of America
##  5 2011-05-01           13 United States of America
```

```
##  6 2011-06-01           24 United States of America
##  7 2011-07-01            5 United States of America
##  8 2011-08-01            9 United States of America
##  9 2011-09-01           24 United States of America
## 10 2011-10-01           18 United States of America
## # ... with 590 more rows
```

Task 4: Examine Data

For this task, you will examine the data.

Task 4.1

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

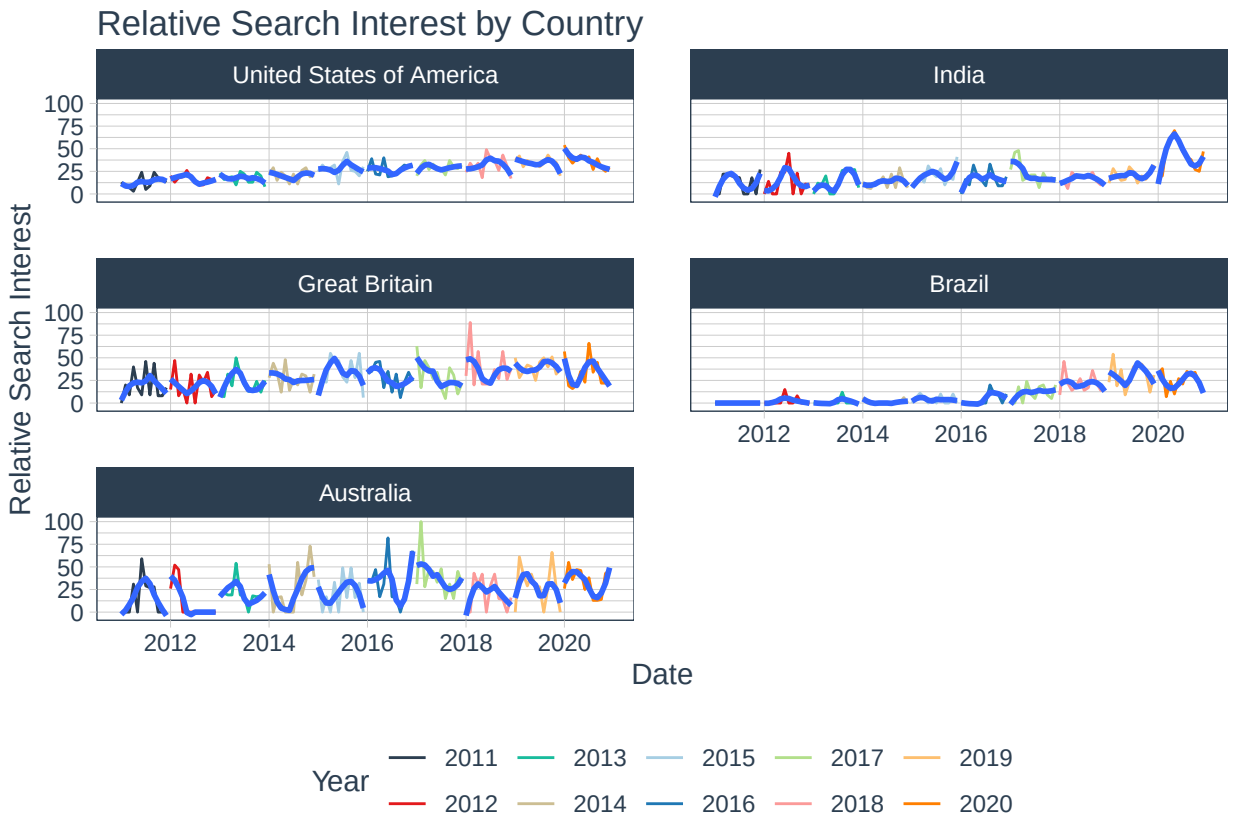
1. **date** as the *date* variable,
2. **rel_interest** as the *value* variable,
3. **country** as the *facet* variable and fixing the *scales* of the facets and creating *two* columns of facets,
4. labeling the x-axis, y-axis, and legend and providing an appropriate title, and
5. creating a static plot.

Questions 4.1: Answer these questions: (1) Describe the differences in *trends* in *relative search interest* in *India* and *Brazil*. (2) Which country tends to have the highest spikes in *relative search interest*?

Responses 4.1: *(1) India had more consistent growth in relative search interest over the year whereas Brazil did not have much growth in relative search interest until around 2018 (2) Australia.*

```
### static plot of complete time series by year
## call data
interest_work %>%
  ## plot
  plot_time_series(
    # date variable
    .date_var = date,
    # outcome variable
    .value = rel_interest,
    # facet variable
    .facet_vars = country,
    # fix scales
    .facet_scales = "fixed",
    # number of columns
    .facet_ncol = 2,
    # color by year
    .color_var = year(date),
    # x-axis label
    .x_lab = "Date",
    # y-axis label
    .y_lab = "Relative Search Interest",
    # color legend
    .color_lab = "Year",
    # title
    .title = "Relative Search Interest by Country",
```

```
  # interactive
  .interactive = FALSE
)
```



Task 5: Time Series Validation

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

Task 5.1

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

Then, create a validation split object for **brazil_ts** using **time_series_split()**. Set the *date* variable, **assess** to **12 months**, and **cumulative** to **TRUE**. Name the object **data_split**.

```
### select one time series
## save as object
brazil_ts <- interest_work %>%
  ## filter
  filter(country == "Brazil")

### create split
## save as object
data_split <- brazil_ts %>%
  ## split the data
```

```
  time_series_split(
    # date variable
    date_var = date,
    # assess
    assess = "12 months",
    # cumulative
    cumulative = TRUE
  )
```

Task 6: Prepare Model Features

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

Task 6.1

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()**.

```
### modeling recipe
## save as object
recipe_spec <-
  ## set initial formula and data
  recipe(rel_interest ~ date, training(data_split)) %>%
  ## calculate date features
  step_timeseries_signature(date) %>%
  ## remove unnecessary features
  step_rm(
    # string match
    matches("(.iso$)|(.xts$)|(week)|(day)|(hour)|(minute)|(second)|(am.pm)")
  ) %>%
  ## normalize some features
  step_normalize(date_index.num, date_year) %>%
  ## add one-hot encoding
  step_dummy(all_nominal(), one_hot = TRUE)
```

Task 7: Time Series Models

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

Task 7.1

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 7.1: 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 7.1: *(1) -0.2396 (2) 0.0111 (3)additive (4) No seasonality.*

```
### exponential smoothing
## save as object
wrkflw_fit_ets <- workflow() %>%
  ## add model to workflow
  add_model(
    # auto-generate exponential smoothing specification
    exp_smoothing() %>%
      # estimator
      set_engine(engine = "ets")
  ) %>%
  ## add recipe
  add_recipe(
    # specify recipe
    recipe_spec %>%
      # remove from recipe
      step_rm(
        # remove all predictors
        all_predictors(),
        # except for date
        -date
      )
  ) %>%
  ## fit workflow to training data
  fit(training(data_split))
```

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

```
## view estimated model
wrkflw_fit_ets
```

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

```
## * step_rm()
##
## -- Model -----------------------------------------------------------------
## ETS(A,A,N)
##
## Call:
##  forecast::ets(y = outcome, model = model_ets, damped = damping_ets)
##
##   Smoothing parameters:
##     alpha = 0.0111
##     beta  = 0.0111
##
##   Initial states:
##     l = -0.2396
##     b = 0.3085
##
##   sigma:  7.7661
##
##      AIC     AICc      BIC
## 954.3440 954.9323 967.7547
```

Task 7.2

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.

Questions 7.2: What kind of *ARIMA* was estimated?

Responses 7.2: *Simple exponential smoothing model.*

```
### autoregressive integrated moving average
## save as object
wrkflw_fit_arima <- workflow() %>%
  ## add model to workflow
  add_model(
    # auto-generate ARIMA specification
    arima_reg() %>%
      # estimator
      set_engine(engine = "auto_arima")
  ) %>%
  ## add recipe
  add_recipe(
    # specify recipe
    recipe_spec %>%
      # remove from recipe
      step_rm(
        # remove all predictors
        all_predictors(),
```

```
        # except for date
        -date
      )
  ) %>%
  ## fit workflow to training data
  fit(training(data_split))
```

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

```
## view estimated model
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(0,1,1)(0,0,1)[12] with drift
##
## Coefficients:
##           ma1    sma1   drift
##       -0.8772  0.4787  0.3148
## s.e.   0.0384  0.1077  0.1319
##
## sigma^2 estimated as 54.56:  log likelihood=-366.47
## AIC=740.94   AICc=741.33   BIC=751.63
```

Task 7.3

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.

Questions 7.3: What is the *seasonality mode* of the model?

Responses 7.3: *additive*.

```
### prophet
## save as object
wrkflw_fit_prophet <- workflow() %>%
  ## add model to workflow
  add_model(
    # auto-generate prophet specification
    prophet_reg() %>%
      # estimator
      set_engine(engine = "prophet")
  ) %>%
  ## add recipe
  add_recipe(
    # specify recipe
    recipe_spec %>%
      # remove from recipe
      step_rm(
        # remove all predictors
        all_predictors(),
        # except for date
        -date
      )
  ) %>%
  ## fit workflow to training data
  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.
```

```
## view estimated model
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
```

Task 8: Evaluate Accuracy of Models

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

Task 8.1

Create a models table named **models_tbl** consisting of the three 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 8.1: Answer these questions: (1) What is the *ensemble* prediction for *2020-10-01*? (2) Was the *ETS* or *ARIMA* model *less* wrong with the prediction for *2020-03-01*?

Responses 8.1: *(1) 38.9 (2) ETS* .

```
### place models in a single table
## save as object
models_tbl <- modeltime_table(
  # ETS
  wrkflw_fit_ets,
  # ARIMA
  wrkflw_fit_arima,
  # prophet
  wrkflw_fit_prophet
)



### select models for ensemble
## save as object
ensemble_set <- models_tbl %>%
  ## create average of models
  ensemble_average(type = "mean")



### ensemble table
## save as object
ensemble_tbl <- modeltime_table(
  # ensemble
```

```
  ensemble_set
) %>%
  ## combine tables
  combine_modeltime_tables(models_tbl)


### calculate model accuracy using testing data
## save as object
models_calibrate <- ensemble_tbl %>%
  ## evaluate on testing data
  modeltime_calibrate(
    # call testing data
    testing(data_split)
  )


### view predictions
## call data
models_calibrate %>%
  ## unnest prediction list
  unnest(.calibration_data) %>%
  ## print all rows
  print(n = Inf)
```

```
## # A tibble: 48 x 8
##    .model_id .model  .model_desc .type date       .actual .prediction .residuals
##        <int> <list>  <chr>       <chr> <date>       <dbl>       <dbl>      <dbl>
##  1         1 <ensem~ ENSEMBLE (~ Test  2020-01-01      32        32.7     -0.677
##  2         1 <ensem~ ENSEMBLE (~ Test  2020-02-01      38        37.8      0.230
##  3         1 <ensem~ ENSEMBLE (~ Test  2020-03-01       7        33.8    -26.8
##  4         1 <ensem~ ENSEMBLE (~ Test  2020-04-01      24        36.3    -12.3
##  5         1 <ensem~ ENSEMBLE (~ Test  2020-05-01      10        33.1    -23.1
##  6         1 <ensem~ ENSEMBLE (~ Test  2020-06-01      27        35.5     -8.50
##  7         1 <ensem~ ENSEMBLE (~ Test  2020-07-01      21        38.2    -17.2
##  8         1 <ensem~ ENSEMBLE (~ Test  2020-08-01      35        41.7     -6.66
##  9         1 <ensem~ ENSEMBLE (~ Test  2020-09-01      32        40.7     -8.66
## 10         1 <ensem~ ENSEMBLE (~ Test  2020-10-01      34        38.9     -4.92
## 11         1 <ensem~ ENSEMBLE (~ Test  2020-11-01      19        35.3    -16.3
## 12         1 <ensem~ ENSEMBLE (~ Test  2020-12-01      12        40.2    -28.2
## 13         2 <workf~ ETS(A,A,N)  Test  2020-01-01      32        39.0     -6.96
## 14         2 <workf~ ETS(A,A,N)  Test  2020-02-01      38        39.8     -1.79
## 15         2 <workf~ ETS(A,A,N)  Test  2020-03-01       7        40.6    -33.6
## 16         2 <workf~ ETS(A,A,N)  Test  2020-04-01      24        41.4    -17.4
## 17         2 <workf~ ETS(A,A,N)  Test  2020-05-01      10        42.3    -32.3
## 18         2 <workf~ ETS(A,A,N)  Test  2020-06-01      27        43.1    -16.1
## 19         2 <workf~ ETS(A,A,N)  Test  2020-07-01      21        43.9    -22.9
## 20         2 <workf~ ETS(A,A,N)  Test  2020-08-01      35        44.7     -9.75
## 21         2 <workf~ ETS(A,A,N)  Test  2020-09-01      32        45.6    -13.6
## 22         2 <workf~ ETS(A,A,N)  Test  2020-10-01      34        46.4    -12.4
## 23         2 <workf~ ETS(A,A,N)  Test  2020-11-01      19        47.2    -28.2
## 24         2 <workf~ ETS(A,A,N)  Test  2020-12-01      12        48.1    -36.1
## 25         3 <workf~ ARIMA(0,1,~ Test  2020-01-01      32        31.8      0.205
## 26         3 <workf~ ARIMA(0,1,~ Test  2020-02-01      38        38.6     -0.591
```

```
## 27         3 <workf~ ARIMA(0,1,~ Test  2020-03-01       7        29.6    -22.6
## 28         3 <workf~ ARIMA(0,1,~ Test  2020-04-01      24        38.2    -14.2
## 29         3 <workf~ ARIMA(0,1,~ Test  2020-05-01      10        26.6    -16.6
## 30         3 <workf~ ARIMA(0,1,~ Test  2020-06-01      27        30.0     -2.99
## 31         3 <workf~ ARIMA(0,1,~ Test  2020-07-01      21        40.7    -19.7
## 32         3 <workf~ ARIMA(0,1,~ Test  2020-08-01      35        42.3     -7.34
## 33         3 <workf~ ARIMA(0,1,~ Test  2020-09-01      32        38.6     -6.60
## 34         3 <workf~ ARIMA(0,1,~ Test  2020-10-01      34        37.0     -3.01
## 35         3 <workf~ ARIMA(0,1,~ Test  2020-11-01      19        28.3     -9.31
## 36         3 <workf~ ARIMA(0,1,~ Test  2020-12-01      12        38.3    -26.3
## 37         4 <workf~ PROPHET     Test  2020-01-01      32        27.3      4.72
## 38         4 <workf~ PROPHET     Test  2020-02-01      38        34.9      3.07
## 39         4 <workf~ PROPHET     Test  2020-03-01       7        31.2    -24.2
## 40         4 <workf~ PROPHET     Test  2020-04-01      24        29.2     -5.21
## 41         4 <workf~ PROPHET     Test  2020-05-01      10        30.4    -20.4
## 42         4 <workf~ PROPHET     Test  2020-06-01      27        33.4     -6.42
## 43         4 <workf~ PROPHET     Test  2020-07-01      21        30.1     -9.07
## 44         4 <workf~ PROPHET     Test  2020-08-01      35        37.9     -2.89
## 45         4 <workf~ PROPHET     Test  2020-09-01      32        37.8     -5.80
## 46         4 <workf~ PROPHET     Test  2020-10-01      34        33.4      0.643
## 47         4 <workf~ PROPHET     Test  2020-11-01      19        30.3    -11.3
## 48         4 <workf~ PROPHET     Test  2020-12-01      12        34.1    -22.1
```

Task 8.2

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 **brazil_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**.

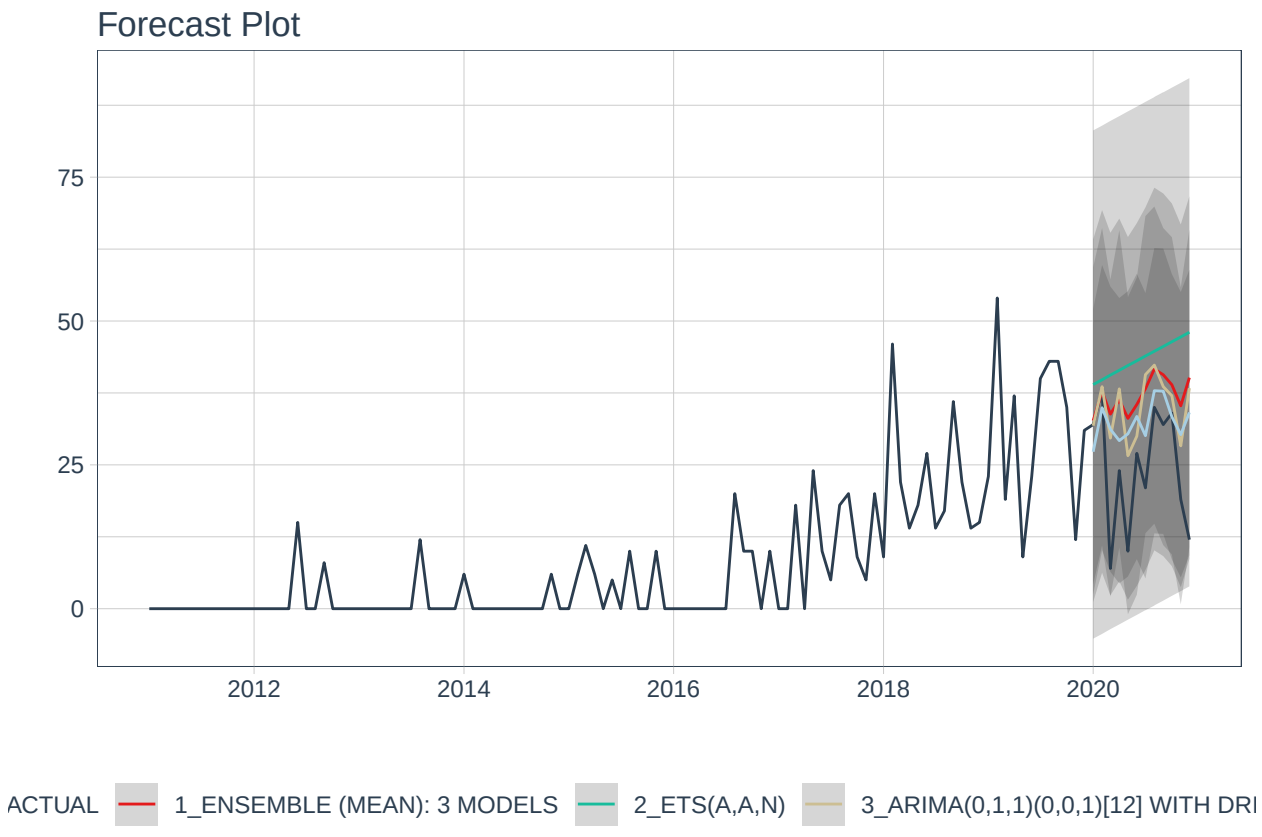
Questions 8.2: Answer these questions: (1) Describe the difference between the *ETS* and *ARIMA* predictions using the interactive plot. (2) Describe the difference between the *ensemble* and *prophet* predictions using the interactive plot. (3) What is the **mase** of the *ARIMA*? (4) Based on **rmse**, which model performs the best?

Responses 8.2: *(1) The ETS is a straight line and the ARIMA varies more (2) Ensemble and Prophet are similar but the Ensemble is overall higher than prophet (3) 0.8994191 (4) The ETS.*

```
### visualize the forecasts
## call data
models_calibrate_plot <- models_calibrate %>%
  ## forecast
  modeltime_forecast(
    # testing data
    new_data = testing(data_split),
    # complete time series
    actual_data = brazil_ts
  ) %>%
  ## plot
  plot_modeltime_forecast(
    # interactive
    .interactive = FALSE
  )
```

```
## display plot
models_calibrate_plot
```

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




```
### print model accuracy results
## call data
models_calibrate %>%
  ## extract accuracy
  modeltime_accuracy() %>%
  ## flextable
  flextable() %>%
  ## make header row bold
  bold(part = "header") %>%
  ## make header background gray
  bg(bg = "#D3D3D3", part = "header") %>%
  ## fit rows neatly based on window
  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	n
1	ENSEMBLE (MEAN): 3 MODELS	Test	12.790205	96.84991	1.065
2	ETS(A,A,N)	Test	19.255620	135.53326	1.604
3	ARIMA(0,1,1)(0,0,1)[12] WITH DRIFT	Test	10.793029	81.21251	0.899
4	PROPHET	Test	9.647038	77.72953	0.803

Task 9: Save Object

For this task, you will save a plot.

Task 9.1

Save **models_calibrate_plot** as **ts_brazil.png** in the **plots** folder of the project directory using **ggsave()**. Make sure to create the plots again by setting the *interactive* mode to **FALSE**. Use a width of *9 inches* and height of *6 inches* for all plots.

```
## save a single plot to a file
ggsave(
  ## file path
  here("plots", "ts_brazil.png"),
  ## plot object
  plot = models_calibrate_plot,
  ## dimensions
  units = "in", width = 9, height = 6)
```

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

Task 10: Conceptual Question

For your last task, you will respond to a conceptual question.

Question 10.1: Describe what it means to difference a time series.

Response 10.1: *To difference a time series means to help stablize the mean of a time series by removing changes in the level of time series. This eliminates/reduces trend and seasonality..*