# STAT 390 Model Report

**Auto-Arima Log**

Group 2: Cindy Ha, Willie Xie, Erica Zhang

## Table of contents

#load pkgs

```r
library(tidyverse)
library(tidymodels)
library(modeltime)
library(doParallel)
library(RcppRoll)
```

# 1 Source

# 2 https://www.r-bloggers.com/2020/06/introducing-modeltime-tidy-time-series-forecasting-using-tidymodels/

# 3 1. Read in data

```r
train_lm <- readRDS("Data/finalized_data/final_train_lm.rds")
test_lm <- readRDS("Data/finalized_data/final_test_lm.rds")
```

# 4 weekly rolling average of log of new cases

```r
complete_lm <- train_lm %>% rbind(test_lm) %>%
  group_by(location) %>%
  arrange(date, .by_group = TRUE) %>%
  mutate(
    cases_log = ifelse(is.finite(log(new_cases)), log(new_cases), 0),
    value = roll_mean(cases_log, 7, align = "right", fill = NA)) %>%
  mutate(value = ifelse(is.na(value), cases_log, value)) %>%
  arrange(date, .by_group = TRUE) %>%
  slice(which(row_number() %% 7 == 0)) %>%
  mutate(
    time_group = row_number(),
    seasonality_group = row_number() %% 53) %>%
  ungroup() %>%
  mutate(seasonality_group = as.factor(seasonality_group))
train_lm <- complete_lm %>% filter(date < as.Date("2023-01-01")) %>%
  group_by(location) %>%
  arrange(date, .by_group = TRUE) %>%
```

```
  ungroup()
test_lm <- complete_lm %>% filter(date >= as.Date("2023-01-01")) %>%
  group_by(location) %>%
  arrange(date, .by_group = TRUE) %>%
  ungroup()
```

# 5 2. Find model trend by country

```
train_lm_fix <- NULL
test_lm_fix <- NULL
country_names <- unique(train_lm$location)
for (i in country_names) {
  data = train_lm %>% filter(location == i)
  complete_data = complete_lm %>% filter(location == i)
  # find linear model by country
  lm_model = lm(value ~ 0 + time_group + seasonality_group,
              data %>% filter(between(time_group, 13, nrow(data) - 12)))
  x = complete_data %>%
    mutate(
      trend = predict(lm_model, newdata = complete_data),
      slope = as.numeric(coef(lm_model)["time_group"]),
      seasonality_add = trend - slope * time_group,
      err = value - trend) %>%
    mutate_if(is.numeric, round, 5)
  train_lm_fix <<- rbind(train_lm_fix, x %>% filter(date < as.Date("2023-01-01")))
  test_lm_fix <<- rbind(test_lm_fix, x %>% filter(date >= as.Date("2023-01-01")))
}
```

# 6 first extract countries

```
for (loc in country_names) {
  location_data <- train_lm_fix %>% filter(location == loc)
  location_name <- paste0("train_lm_fix_", make.names(loc))  # Create the name with "train
  assign(location_name, location_data, envir = .GlobalEnv)
}

for (loc in country_names) {
```

```
  location_data <- test_lm_fix %>% filter(location == loc)
  location_name <- paste0("test_lm_fix_", make.names(loc))  # Create the name with "train_
  assign(location_name, location_data, envir = .GlobalEnv)
}
```

# 7 ARIMA Model tuning for errors

# 8 3. Create validation sets for every year train + 2 month test with 4-month increments

```
data_folds <- rolling_origin(
  train_lm_fix,
  initial = 53,
  assess = 4*2,
  skip = 4*4,
  cumulative = FALSE
)
```

# 9 4. Define model, recipe, and workflow

```
autoarima_model <- arima_reg(
  seasonal_period = 53,
  non_seasonal_ar = tune(), non_seasonal_differences = tune(), non_seasonal_ma = tune(),
  seasonal_ar = tune(), seasonal_differences = tune(), seasonal_ma = tune()) %>%
  set_engine('auto_arima')

autoarima_recipe <- recipe(err ~ date, data = train_lm_fix_United.States)
# View(autoarima_recipe %>% prep() %>% bake(new_data = NULL))

autoarima_wflow <- workflow() %>%
  add_model(autoarima_model) %>%
  add_recipe(autoarima_recipe)
```

# 10 5. Setup tuning grid

```
autoarima_params <- autoarima_wflow %>%
  extract_parameter_set_dials() %>%
  update(
    non_seasonal_ar = non_seasonal_ar(c(0,5)),
    non_seasonal_differences = non_seasonal_differences(c(0,2)),
    non_seasonal_ma = non_seasonal_ma(c(0,5)),
    seasonal_ar = seasonal_ar(c(0, 2)),
    seasonal_ma = seasonal_ma(c(0, 2)),
    seasonal_differences = seasonal_differences(c(0,1))
  )
autoarima_grid <- grid_regular(autoarima_params, levels = 3)
```

# 11 6. Model Tuning

# 12 Setup parallel processing

```
# # detectCores(logical = FALSE)
# cores.cluster = makePSOCKcluster(4)
# registerDoParallel(cores.cluster)
#
# autoarima_tuned <- tune_grid(
#   autoarima_wflow,
#   resamples = data_folds,
#   grid = autoarima_grid,
#   control = control_grid(save_pred = TRUE,
#                          save_workflow = FALSE,
#                          parallel_over = "everything"),
#   metrics = metric_set(yardstick::rmse)
# )
#
# stopCluster(cores.cluster)
#
# autoarima_tuned %>% collect_metrics() %>%
#   relocate(mean) %>%
#   group_by(.metric) %>%
#   arrange(mean)
```

## 13 7. Results

```
# autoplot(autoarima_tuned, metric = "rmse")
# show_best(autoarima_tuned, metric = "rmse")
```

**8. Fit Best Model**

**argentina (3,1,3,1,0,1)**

**australia (3,0,3,1,0,1)**

**canada (3,0.5,1,0,1)**

**colombia (3,1,3,1,0,1)**

**ecuador (3,0,3,1,0,1)**

**ethiopia (3,0,3,1,0,1)**

**france (3,0,3,1,0,1)**

**germany (4,0,3,1,0,1)**

**india (3,0,3,1,0,1)**

**italy (3,0,3,1,0,1)**

**japan (3,0,3,1,0,1)**

**mexico (3,0,3,1,0,1)**

**morocco (3,0,3,1,0,1)**

**pakistan (3,0,3,1,0,1)**

**philippines (3,0,3,1,0,1)**

**russia (3,0,3,1,0,1)**

**saudi arabia (3,0,3,1,0,1)**



**south africa (3,0,3,1,0,1)**

**south korea (4,0,3,1,0,1)**

**sri lanka (3,1,3,1,0,1)**

```
autoarima_model <- arima_reg(
  seasonal_period = 53,
  non_seasonal_ar = 3, non_seasonal_differences = 1, non_seasonal_ma = 3,
  seasonal_ar = 1, seasonal_differences = 0, seasonal_ma = 1) %>%
  set_engine('auto_arima')
autoarima_recipe <- recipe(err ~ date, data = train_lm_fix_United.States)
autoarima_wflow <- workflow() %>%
  add_model(autoarima_model) %>%
  add_recipe(autoarima_recipe)
```

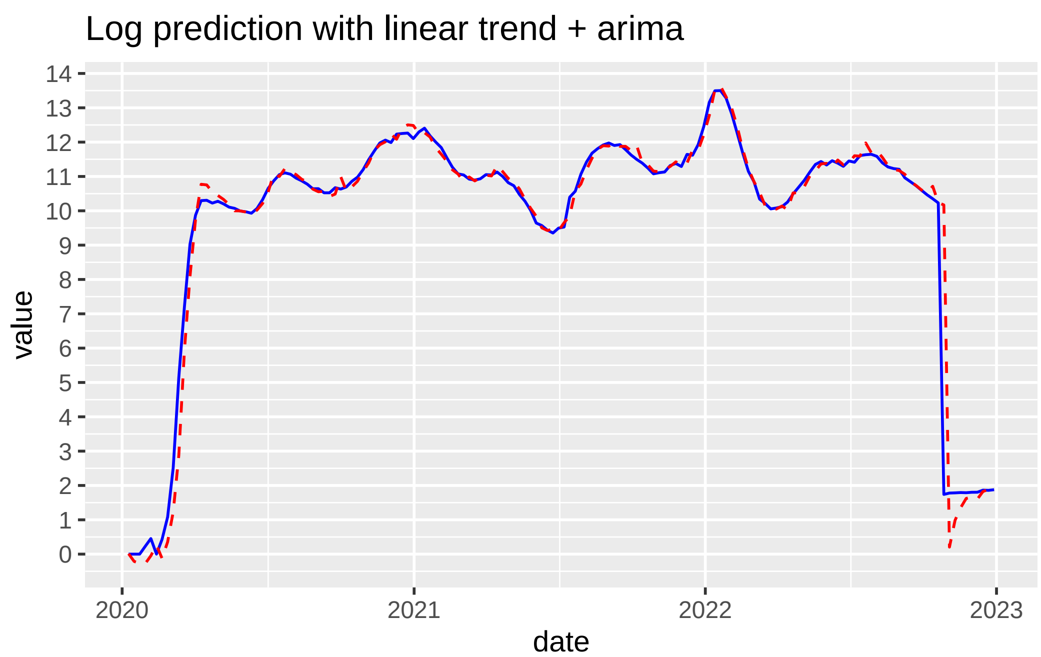
Fit training data:

```
autoarima_fit <- fit(autoarima_wflow, data = train_lm_fix_United.States)
final_train <- train_lm_fix_United.States %>%
  bind_cols(pred_err = autoarima_fit$fit$fit$fit$data$.fitted) %>%
  mutate(pred = trend + pred_err) %>%
  mutate_if(is.numeric, round, 5)
```

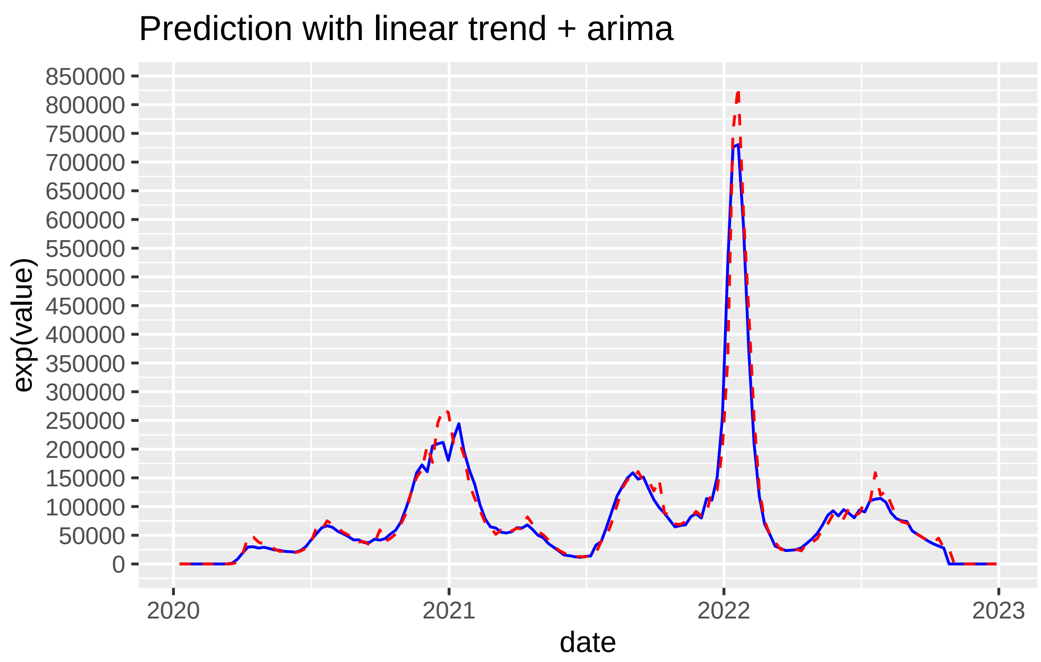
training visualization:

```
# final prediction with linear trend + arima error modelling

#log
ggplot(final_train) +
  geom_line(aes(date, value), color = 'blue') +
  geom_line(aes(date, pred), color = 'red', linetype = "dashed") +
  scale_y_continuous(n.breaks = 15) +
  labs(title = "Log prediction with linear trend + arima")
```




```
#unlog
ggplot(final_train) +
  geom_line(aes(date, exp(value)), color = 'blue') +
  geom_line(aes(date, exp(pred)), color = 'red', linetype = "dashed") +
  scale_y_continuous(n.breaks = 15) +
  labs(title = "Prediction with linear trend + arima")
```




```
library(ModelMetrics)
# rmse of error prediction
rmse(final_train$err, final_train$pred_err)
```

```
[1] 0.753966
```

```
# rmse of just linear trend
rmse(final_train$value, final_train$trend)
```

```
[1] 3.712557
```

```
rmse(exp(final_train$value), exp(final_train$trend))
```

```
[1] 74629.99
```

```
# rmse of linear trend + arima
rmse(final_train$value, final_train$pred)
```

```
[1] 0.7539661
```

```r
rmse(exp(final_train$value), exp(final_train$pred))
```

```
[1] 22461.69
```

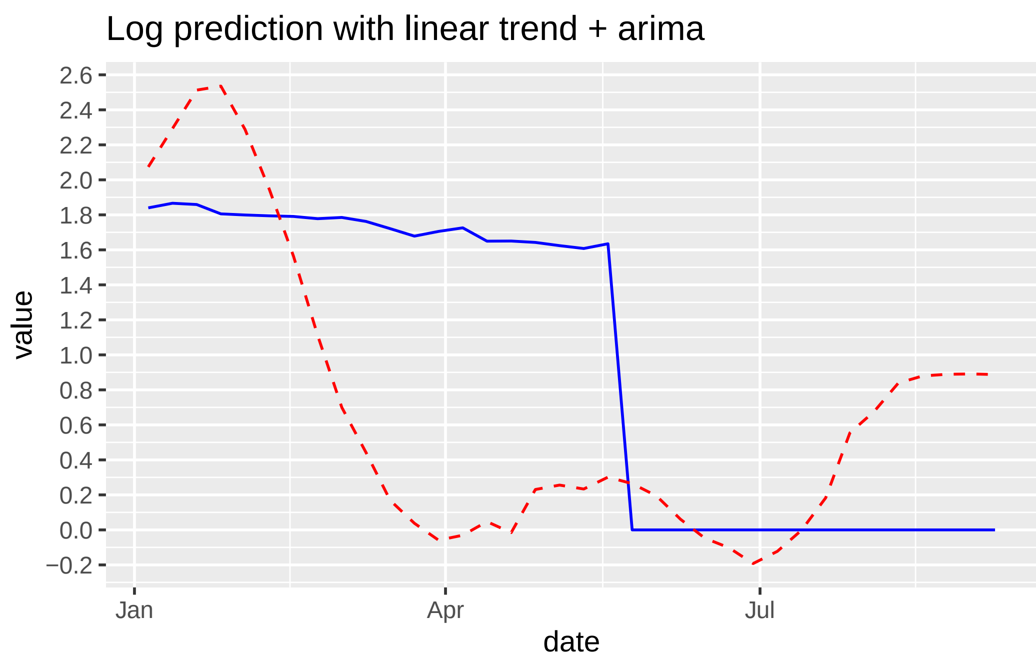
Predict test:

# 38 Testing set

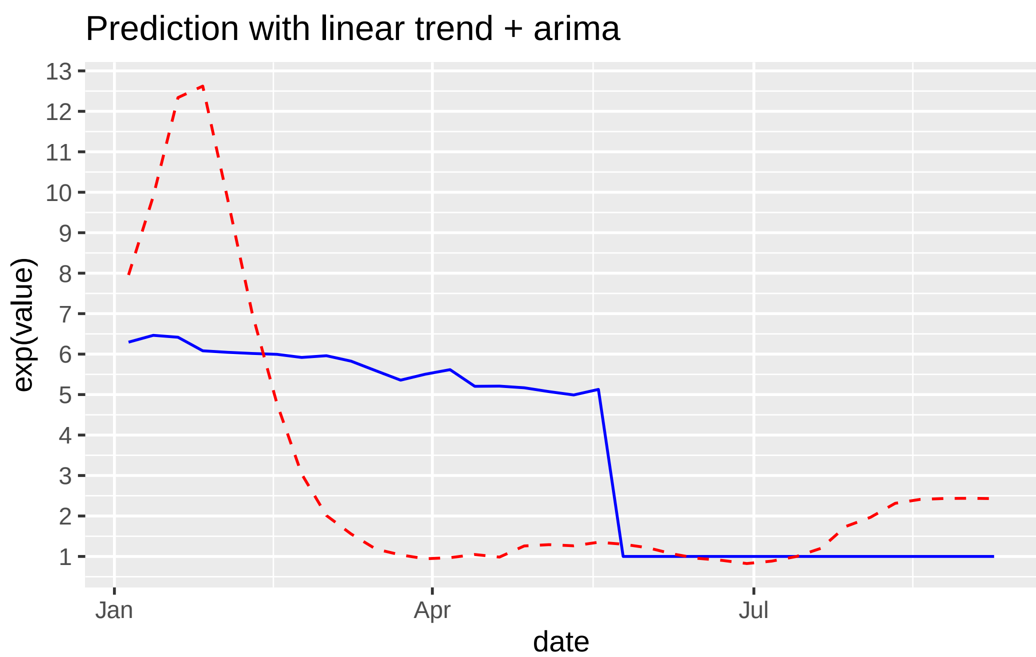
```r
final_test <- test_lm_fix_United.States %>%
  bind_cols(predict(autoarima_fit, new_data = test_lm_fix_United.States)) %>%
  rename(pred_err = .pred) %>%
  mutate(pred = trend + pred_err) %>%
  mutate_if(is.numeric, round, 5)
```

Test visualization:

```r
# final prediction with linear trend + arima error modelling
ggplot(final_test) +
  geom_line(aes(date, value), color = 'blue') +
  geom_line(aes(date, pred), color = 'red', linetype = "dashed") +
  scale_y_continuous(n.breaks = 15) +
  labs(title = "Log prediction with linear trend + arima")
```




```
ggplot(final_test) +
  geom_line(aes(date, exp(value)), color = 'blue') +
  geom_line(aes(date, exp(pred)), color = 'red', linetype = "dashed") +
  scale_y_continuous(n.breaks = 15) +
  labs(title = "Prediction with linear trend + arima")
```




RMSE result

```
# rmse of linear trend + arima
rmse(final_test$value, final_test$pred)
```

```
[1] 0.9700073
```

```
rmse(exp(final_test$value), exp(final_test$pred))
```

```
[1] 3.056902
```