The Covid19 pandemic had, and unfortunately still have, a significant impact on most of the major industries. While for some sectors, the impact was positive (such as online retails, internet and steaming providers, etc.), it was negative for others (such as transportation, tourism, entertainment, etc.). In both cases, we can leverage time series modeling to quantify the effect of the Covid19 on the sector.

One simplistic approach for quantifying the impact of the pandemic (whether it is positive or negative) would include the following steps:

- Split the series into pre-covid and post-covid
- Train a time series model with the pre-covid data. This would enable us to simulate the value of the series if there was no pandemic
- Using the trained model to forecast the horizon of the post-covid series
- Use the difference between the forecast and actual (post-covid series) to quantify the Covid19 impact on the series

To demonstrate this approach, I will use the `sfo_passengers` dataset from the sfo package. The `sfo_passengers` dataset provides monthly statistics about the San Francisco International Airport (SFO) air traffic between July 2005 and September 2020. More details about the dataset available in the following post and vignette.

For this analysis, we will use the following packages:

- `sfo` – the passenger air traffic data
- `dplyr` – data prep
- `plotly` – data visualization
- `TSstudio` – time series analysis and forecasting

```
library(sfo)
library(dplyr)
library(plotly)
library(TSstudio)
```

Data

As mentioned above, the `sfo_passengers` dataset provides monthly statistics about the air passenger traffic at SFO airport since 2005. That includes monthly information about the number of passengers by different categories such as operating airline, region, terminal, etc. Let's load the data:

```
data("sfo_passengers")

str(sfo_passengers)
## 'data.frame':    22576 obs. of  12 variables:
##  $ activity_period           : int  202009 202009 202009 202009 202009 202009 202009
202009 202009 202009 ...
##  $ operating_airline         : chr  "United Airlines" "United Airlines" "United
Airlines" "United Airlines" ...
##  $ operating_airline_iata_code: chr  "UA" "UA" "UA" "UA" ...
##  $ published_airline         : chr  "United Airlines" "United Airlines" "United
Airlines" "United Airlines" ...
##  $ published_airline_iata_code: chr  "UA" "UA" "UA" "UA" ...
##  $ geo_summary               : chr  "International" "International" "International"
"International" ...
##  $ geo_region                : chr  "Mexico" "Mexico" "Mexico" "Mexico" ...
##  $ activity_type_code        : chr  "Enplaned" "Enplaned" "Enplaned" "Deplaned" ...
##  $ price_category_code       : chr  "Other" "Other" "Other" "Other" ...
##  $ terminal                  : chr  "Terminal 3" "Terminal 3" "International"
"International" ...
##  $ boarding_area             : chr  "F" "E" "G" "G" ...
##  $ passenger_count           : int  6712 396 376 6817 3851 3700 71 83 65 45 ...
```

Before we will convert the data into time series format, we will transform the `activity_period` into a `Date` format:

```
df <- sfo_passengers %>%
  mutate(date = as.Date(paste(substr(sfo_passengers$activity_period, 1,4),
                              substr(sfo_passengers$activity_period, 5,6),
                              "01", sep ="/")))
```

Next, we will transform the dataset into a time series format by grouping the passenger by the `date` variable:

```
df <- df %>%
  group_by(date) %>%
  summarise(y = sum(passenger_count), .groups = "drop")

head(df)
## # A tibble: 6 x 2
##   date             y
##
## 1 2005-07-01 3225769
## 2 2005-08-01 3195866
## 3 2005-09-01 2740553
## 4 2005-10-01 2770715
## 5 2005-11-01 2617333
## 6 2005-12-01 2671797
```

Now, we have a monthly time series:

```
plot_ly(data = df,
        x = ~ date,
        y = ~ y,
        type = "scatter",
        mode = "line",
        name = "Total Passengers") %>%
  add_segments(x = as.Date("2020-02-01"),
               xend = as.Date("2020-02-01"),
               y = min(df$y),
               yend = max(df$y) * 1.05,
               line = list(color = "black", dash = "dash"),
               showlegend = FALSE) %>%
  add_annotations(text = "Pre-Covid19",
                  x = as.Date("2018-09-01"),
                  y = max(df$y) * 1.05,
                  showarrow = FALSE) %>%
  add_annotations(text = "Post-Covid19",
                  x = as.Date("2021-08-01"),
                  y = max(df$y) * 1.05,
                  showarrow = FALSE) %>%
  layout(title = "Total Number of Air Passengers - SFO Airport",
         yaxis = list(title = "Number of Passengers"),
         xaxis = list(title = "Source: San Francisco Open Data Portal"))
```

As can be observed in the plot above, the Covid19 effect is well pronounced since March 2020. To quantify the effect of the pandemic on the number of passengers, we will split the series into the following two series:

- Pre-Covid19 series- all observations prior to March 2020
- Post-Covid19 series- all observations post (include) to March 2020

We will use the Pre-Covid19 series to train a time series model. That will enable us to forecast the number of passengers as if there was no pandemic.

```
pre_covid <- df %>%
  dplyr::filter(date < as.Date("2020-03-01")) %>%
  dplyr::arrange(date)

post_covid <- df %>%
  dplyr::filter(date >= as.Date("2020-03-01")) %>%
  dplyr::arrange(date)
```

We will use the `pre_covid` series to train a time series model. That will enable us to forecast the number of passengers as there was no pandemic. Once we forecast the corresponding observations of the `post_covid` series with the `pre_covid` data, we could quantify the impact of the Covid19 on the total number of passengers.

Analyzing the data

Before we forecast the series, let's run a quick exploratory analysis on the series to identify its main characteristics. We will use the TSstudio package to visualize the `pre_covid` series. Note that the package does not support, yet, the `tsibble` object. Therefore, we will convert the series into a `ts` object first:

```
ts.obj <- ts(pre_covid$y, start = c(2005, 7), frequency = 12)
```

The series before the outbreak of the pandemic:

```
ts_plot(ts.obj,
        title = "Total Number of Air Passengers - SFO Airport",
        Ytitle = "Number of Passengers",
        slider = TRUE)
```

Like most series that describes monthly air passenger traffic, the series has a strong monthly seasonal pattern and a positive trend. You can also notice that the seasonal component's oscillation has become larger since 2017 (compared to previous years). Let's use the `ts_seasonal` function to create a seasonal plot of the series:

```
ts_seasonal(ts.obj = ts.obj, type = "all")
```

As can see in the seasonal plot, the monthly seasonal effect kept overtime while the series continue to grow from year to year.

Similarly, we can review the series correlation with its past lags using the ACF and PACF functions:

```
ts_cor(ts.obj = ts.obj, lag.max = 36)
```

And as expected, we can see strong correlation between the series and the first and seasonal lags. We will leverage this information to select time-series models for seasonal data.

Forecast the Pre-Covid19 series

One of my favorite forecasting strategies is a combination of horse racing between different time series models and backtesting as a training approach. Backtesting is the time series equivalent of the machine learning cross-validation training approach. The idea here is simple - test each model with backtesting, and select the one that performed best, on average, on the different testing partition.

The `train_model` function from the **TSstudio** package enables us to apply this strategy seamlessly using models from the **forecast** and **stats** packages. For simplicity, we will use different flavors of `ETS` and `Holt-Winters` models and out-of-the-box `auto.arima` and `tslm` models. For the backtesting, we will split the series into 6 testing partitions, each 12 months spaced by 3 months from each other.

The `methods` argument defines the models to use and the `train_method` argument defines the setting of the backtesting. Can find more details about the function here.

```
methods <- list(ets1 = list(method = "ets",
                            method_arg = list(opt.crit = "lik"),
                            notes = "ETS model opt.crit=lik"),
                ets2 = list(method = "ets",
                            method_arg = list(opt.crit = "amse"),
                            notes = "ETS model opt.crit=amse"),
                ets3 = list(method = "ets",
                            method_arg = list(opt.crit = "mse"),
                            notes = "ETS model opt.crit=mse"),
                auto_arima = list(method = "auto.arima",
                              notes = "Auto ARIMA"),
                hw1 = list(method = "HoltWinters",
                           method_arg = NULL,
                           notes = "HoltWinters Model"),
                 hw2 = list(method = "HoltWinters",
                           method_arg = list(seasonal = "multiplicative"),
                           notes = "HW with multip. seasonality"),
                tslm = list(method = "tslm",
                            method_arg = list(formula = input ~ trend + season),
                            notes = "tslm with trend and seasonal"))


train_method = list(partitions = 6,
                    sample.out = 12,
                    space = 3)
```

After we defined the methods and `train_method` arguments we will use the `train_model` function to train the models. Note that the forecast horizon is set the the length of the `post_covid` series. In addition we will set the `MAPA` as the error metric to evaluate the performance of the different models on the testing partitions:

```
md <- train_model(input = ts.obj,
                  methods = methods,
                  train_method = train_method,
                  horizon = nrow(post_covid),
                  error = "MAPE")
## # A tibble: 7 x 7
##   model_id   model       notes                          avg_mape avg_rmse
`avg_coverage_80%` `avg_coverage_95%`
##
```

```
## 1 hw1        HoltWinters HoltWinters Model             0.0277  149046.
0.792             0.958
## 2 ets2       ets         ETS model opt.crit=amse       0.0284  161754.
0.792             0.972
## 3 hw2        HoltWinters HW with multip. seasonality   0.0300  171702.
0.528             0.847
## 4 ets1       ets         ETS model opt.crit=lik        0.0307  173337.
0.833             0.958
## 5 ets3       ets         ETS model opt.crit=mse        0.0311  174238.
0.861             0.972
## 6 auto_arima auto.arima  Auto ARIMA                    0.0334  184381.
0.597             0.889
## 7 tslm       tslm        tslm with trend and seasonal  0.0370  223194.
0.569             0.75
```

Based on the `leaderboard` table from the `train_model` function, the model that performed best on average on the different testing partitions is the **Holt-Winters** model (first version - `hw1`). The model achieved, on average, the lowest MAPE (2.76%) and RMSE (149046) compared to the other models which evaluated. In addition, the model achieved a close to perfect coverage of the model prediction intervals with an average coverage of 79.2% and 95.8% for the 80% and 95% prediction interval, respectively. We can review the error distribution across the different partitions for each model with the `plot_error` function:

```
plot_error(md)
```

The `plot_model` enables us to animate the forecasted values of each model on the different testing partitions of the backtesting:

```
plot_model(md)
```

We will select the Holt-Winters model (`hw1`) to calculate the Covid19 effect. We will add the selected forecast to the `post_covid` dataset:

```
post_covid$yhat <- as.numeric(md$forecast$hw1$forecast$mean)
post_covid$upper95 <- as.numeric(md$forecast$hw1$forecast$upper[,2])
post_covid$lower95 <- as.numeric(md$forecast$hw1$forecast$lower[,2])
```

Quantify the Covid19 impact

After we added forecasted values, it is straightforward to calculate the monthly impact of the Covid19 on the number of passengers at SFO airport:

```
post_covid$passengers_loss <- post_covid$y - post_covid$yhat

post_covid
## # A tibble: 7 x 6
##   date             y     yhat  upper95  lower95 passengers_loss
##
## 1 2020-03-01 1885466 4675574. 4860610. 4490537.       -2790108.
## 2 2020-04-01  138817 4756550. 4963034. 4550066.       -4617733.
## 3 2020-05-01  286570 5062504. 5288613. 4836395.       -4775934.
## 4 2020-06-01  555119 5476241. 5720592. 5231889.       -4921122.
## 5 2020-07-01  765274 5641839. 5903342. 5380336.       -4876565.
## 6 2020-08-01  852578 5650253. 5928018. 5372488.       -4797675.
## 7 2020-09-01  905992 4541560. 4834848. 4248272.       -3635568.
```

And, the estimated total number of passengers decrease between March and September 2020 as result of the pandemic:

```
sum(post_covid$passengers_loss)
## [1] -30414705
```

Similarly, we can visualize the Covid19 effect on the air passenger traffic:

```
plot_ly() %>%
  add_ribbons(x = post_covid$date,
              ymin = post_covid$y,
              ymax = post_covid$yhat,
              line = list(color = 'rgba(255, 0, 0, 0.05)'),
              fillcolor = 'rgba(255, 0, 0, 0.6)',
              name = "Estimated Loss") %>%
  add_segments(x = as.Date("2020-02-01"),
               xend = as.Date("2020-02-01"),
               y = min(df$y),
               yend = max(df$y) * 1.05,
               line = list(color = "black", dash = "dash"),
               showlegend = FALSE) %>%
  add_annotations(text = "Pre-Covid19",
                  x = as.Date("2017-09-01"),
                  y = max(df$y) * 1.05,
                  showarrow = FALSE) %>%
  add_annotations(text = "Post-Covid19",
                  x = as.Date("2020-09-01"),
                  y = max(df$y) * 1.05,
```

```
                  showarrow = FALSE) %>%
  add_annotations(text = paste("Estimated decrease in", "
",
                               "passengers volume: ~30M",
                               sep = ""),
                  x = as.Date("2020-05-01"),
                  y = 2 * 10 ^ 6,
                  arrowhead = 1,
                  ax = -130,
                  ay = -40,
                  showarrow = TRUE) %>%
  add_lines(x = df$date,
            y = df$y,
            line = list(color = "#1f77b4"),
            name = "Actual") %>%
  layout(title = "Covid19 Impact on SFO Air Passenger Traffic",
         yaxis = list(title = "Number of Passengers"),
         xaxis = list(title = "Time Series Model - Holt-Winters",
                      range = c(as.Date("2015-01-01"), as.Date("2021-01-01"))),
         legend = list(x = 0, y = 0.95))
```

Applications

Once we estimate the decrease in passengers' number, we can quantify losses caused by the Covid19. For example, if each passenger on average pays $10 airport tax, then the estimated tax loss is about 300 Million USD for the specific period.

In addition, as the underline forecast is a point estimate. Therefore you can leverage the prediction interval to provide a range for the drop in air passenger traffic.

Last but not least, you can use a top-bottom approach and distribute the forecast for some of the available categories in the data. For example, the drop of passengers by

- Airline provider
- Region
- Domestic / International flights