**Intro**

The aim of this series of blog is do predict monthly admissions to Singapore public acute adult hospitals. The dataset starts from Jan 2016 and ends in Feb 2021.

There are several approaches to forecast the admissions including

1. Summing up all the individual hospital admissions and forecasting the admissions at a national level
2. Forecasting at each hierarchical level. Every country has an organisation order to its public hospitals. In Singapore, there are 3 levels:

National level  
  |– Cluster level (Clusters are a network of hospitals based on geographical regions. There are 3 health clusters in Singapore.)  
       |– Hospital level (There are 8 public acute adult hospitals.)

Hierarchical modelling can provide figures to decision makers at each level of the healthcare system. This field of forecasting has gained much attention thanks to the M5 competition .

library(tidyverse)

# convert to tsibble

library(fpp3)

df\_tsib<- raw %>%

mutate(Date= yearmonth(as.character(Date))) %>%

# if there are two index, need to use group\_by and summarise to isolate a specific index

as\_tsibble(key= c(Hospital, Cluster), index= Date)

# A hierarchical established with `aggregate\_key`.

# LHS: parent/RHS: child

(df\_hts<- df\_tsib %>% aggregate\_key(Cluster/Hospital, Admission= sum(Admission, na.rm = T))%>%

# cant use between

mutate(Covid= case\_when(

Date==yearmonth("2020-01-01")~ "Yes",

Date==yearmonth("2020-02-01")~ "Yes",

Date==yearmonth("2020-03-01")~ "Yes",

Date==yearmonth("2020-04-01")~ "Yes",

Date==yearmonth("2020-05-01")~ "Yes",

Date==yearmonth("2020-06-01")~ "Yes",

Date==yearmonth("2020-07-01")~ "Yes",

T ~ "No")))

## # A tsibble: 744 x 5 [1M]

## # Key: Cluster, Hospital [12]

## Date Cluster Hospital Admission Covid

## <mth> <chr\*> <chr\*> <dbl> <chr>

## 1 2016 Jan <aggregated> <aggregated> 26555 No

## 2 2016 Feb <aggregated> <aggregated> 24898 No

## 3 2016 Mar <aggregated> <aggregated> 28002 No

## 4 2016 Apr <aggregated> <aggregated> 27488 No

## 5 2016 May <aggregated> <aggregated> 27280 No

## 6 2016 Jun <aggregated> <aggregated> 27724 No

## 7 2016 Jul <aggregated> <aggregated> 28349 No

## 8 2016 Aug <aggregated> <aggregated> 28640 No

## 9 2016 Sep <aggregated> <aggregated> 27309 No

## 10 2016 Oct <aggregated> <aggregated> 27790 No

## # ... with 734 more rows

Two broad approaches were used for this project:

1. Classical approach which uses e.g. ETS and ARMIA.
2. Machine learning approach.

**Reconciliation**

* For bottoms-up approach: The bottom-level can be noisy and more difficult to predict.
* For top-down approach: Traits of individual time series e.g. special events and different seasonal patterns are lost due to information aggregation and it produces less accurate forecasts at lower levels.

**Dataset**

The training set was from Jan 16 to Apr 20 (3 years, 4months) and the test set was from May 20 to Feb 21 (10 months) and the forecast future period would be Mar 21 to Dec 21 (10 months). The forecast horizon will be 10 months; in other words, to predict the admissions for the remaining of 2021.

**Feature Engineering**

Two external regressors were considered for some ARIMA models:

1. Though the peak Covid periods were over, individuals could have reframed from being unnecessarily admitted as they were afraid of the infectious nature associated with hospitals.

**Models**

Base models included:

1. ETS
2. ARIMA
3. ARIMA with Covid *(peak period)* as regressor
4. ARIMA with Covid regressor with 1 month lag
5. ARIMA with Covid regressor with 2 month lag
6. ARIMA with Covid regressor with 3 month lag

Three hierarchical forecasting techniques were used:

1. bottoms up bu
2. reconciliation using ordinary least square ols
3. reconciliation

fun\_reconcile<- function(R, M, B, BU="bu", OLS="ols", MINT="mint"){

LHS= "Admission"

RHS= R

model\_spec= as.formula(paste0(LHS, RHS, sep=""))

df\_hts %>%

filter( Date < yearmonth("2020 May")) %>%

model(base = model\_spec %>% M) %>%

reconcile(

bu = bottom\_up(base), ols = min\_trace(base, method = "ols"), mint = min\_trace(base, method = "mint\_cov")) %>%

rename({{B}} :=base) %>%

rename("{{B}}\_{BU}":= bu) %>% rename("{{B}}\_{OLS}" := ols) %>% rename("{{B}}\_{MINT}" := mint)

}

m\_ets<- fun\_reconcile("~ error() + trend() + season()", ETS, ets)

m\_arima<- fun\_reconcile("~ pdq() + PDQ()", ARIMA, arima)

m\_arima\_covid<- fun\_reconcile("~ Covid", ARIMA, arima\_covid)

m\_arima\_covidL1<- fun\_reconcile("~ Covid +lag(Covid)", ARIMA, arima\_covidL1)

m\_arima\_covidL2<- fun\_reconcile("~ Covid +lag(Covid,1)", ARIMA, arima\_covidL2)

m\_arima\_covidL3<- fun\_reconcile("~ Covid +lag(Covid,2)", ARIMA, arima\_covidL3)

# save models

save(m\_ets, m\_arima, m\_arima\_covid, m\_arima\_covidL1, m\_arima\_covidL2, m\_arima\_covidL3, file = "3bClassic")

**Evaluation**

**ARIMA**

The best ARIMA model class was selected using AICc. The best ARIMA model and ETS model were then evaluated against the test set using rmse and mae.

The best ARIMA model was one with an external regressor for Covid peak period, m\_armia\_covid, i.e. ARIMA(Admission ~ Covid).

fun\_reconcile\_glance<- function(mab){

glance(mab) %>%

group\_by(.model) %>% summarise(avg\_aicc=mean(AICc), sdv\_aicc=sd(AICc))

}

bind\_rows(fun\_reconcile\_glance(m\_arima),

fun\_reconcile\_glance(m\_arima\_covid),

fun\_reconcile\_glance(m\_arima\_covidL1),

fun\_reconcile\_glance(m\_arima\_covidL2),

fun\_reconcile\_glance(m\_arima\_covidL3)) %>%

arrange(avg\_aicc, sort=T)

## # A tibble: 20 x 3

## .model avg\_aicc sdv\_aicc

## <chr> <dbl> <dbl>

## 1 arima\_covid 685. 81.0

## 2 arima\_covid\_bu 685. 81.0

## 3 arima\_covid\_mint 685. 81.0

## 4 arima\_covid\_ols 685. 81.0

## 5 arima\_covidL3 705. 66.3

## 6 arima\_covidL3\_bu 705. 66.3

## 7 arima\_covidL3\_mint 705. 66.3

## 8 arima\_covidL3\_ols 705. 66.3

## 9 arima\_covidL1 715. 67.9

## 10 arima\_covidL1\_bu 715. 67.9

## 11 arima\_covidL1\_mint 715. 67.9

## 12 arima\_covidL1\_ols 715. 67.9

## 13 arima\_covidL2 715. 67.9

## 14 arima\_covidL2\_bu 715. 67.9

## 15 arima\_covidL2\_mint 715. 67.9

## 16 arima\_covidL2\_ols 715. 67.9

## 17 arima 720. 90.4

## 18 arima\_bu 720. 90.4

## 19 arima\_mint 720. 90.4

## 20 arima\_ols 720. 90.4

**ARIMA vs ETS**

* ARIMA was better than ETS.
* Reconciliation approaches (i.e. arima\_covid\_mint, arima\_covid\_ols) performed better as they likely captured information from all levels of the hierarchy.
* MINT technique performed better than OLS .

### test period ###

# need to create manually, horizon in forecast doesnt work w external regressors

test\_set<- df\_hts %>% filter( Date < yearmonth("2020 May")) %>% new\_data(10)%>%

mutate(Covid= case\_when(

Date==yearmonth("2020-05-01")~ "Yes",

Date==yearmonth("2020-06-01")~ "Yes",

Date==yearmonth("2020-07-01")~ "Yes",

T ~ "No"))

### function for accuracy ###

fun\_acc<- function(Forecasted){

Forecasted %>% accuracy(df\_hts, measures = list(rmse=RMSE, mae=MAE))}

### forecast on testing set ###

m\_arima\_covid\_test<- m\_arima\_covid %>% forecast(test\_set) %>% fun\_acc()

## Warning in mapply(FUN = .f, ..., MoreArgs = MoreArgs, SIMPLIFY = SIMPLIFY):

## longer argument not a multiple of length of shorter

## Warning in fc[btm] <- NextMethod(): number of items to replace is not a multiple

## of replacement length

m\_ets\_test<- m\_ets %>% forecast(test\_set) %>% fun\_acc()

## Warning in mapply(FUN = .f, ..., MoreArgs = MoreArgs, SIMPLIFY = SIMPLIFY):

## longer argument not a multiple of length of shorter

## Warning in mapply(FUN = .f, ..., MoreArgs = MoreArgs, SIMPLIFY = SIMPLIFY):

## number of items to replace is not a multiple of replacement length

### function for average accuracy ###

fun\_acc\_avg<- function(ACC){

ACC %>% group\_by(.model) %>% summarise(avg\_rmse=mean(rmse), avg\_mae=mean(mae))

}

### compare models against test ###

bind\_rows(fun\_acc\_avg(m\_arima\_covid\_test),

fun\_acc\_avg(m\_ets\_test)) %>%

arrange(avg\_rmse, sort=T)

## # A tibble: 8 x 3

## .model avg\_rmse avg\_mae

## <chr> <dbl> <dbl>

## 1 arima\_covid\_mint 847. 745.

## 2 arima\_covid\_ols 1085. 937.

## 3 arima\_covid 1117. 991.

## 4 arima\_covid\_bu 1142. 1043.

## 5 ets 1951. 1788.

## 6 ets\_ols 2193. 2028.

## 7 ets\_bu 2343. 2146.

## 8 ets\_mint 3045. 2814.

**Performance for each level**

The above accuracy was for across all the time series. The performance for individual levels were reviewed and visualized. Specific hierarchical level information is filtered out using is\_aggregated as an argument for filter.

* filter(!is\_aggregated(level)) does not aggregate members at that specific level thus analysis for that level can be conducted.
* filter(is\_aggregated(level)) aggregates values from that level and provides the aggregated information for the next level.

L\_hospital<- m\_arima\_covid %>% select(arima\_covid\_mint) %>% forecast(test\_set) %>%

filter(!is\_aggregated(Hospital))

L\_cluster<-m\_arima\_covid %>% select(arima\_covid\_mint) %>% forecast(test\_set) %>%

filter(is\_aggregated(Hospital), !is\_aggregated(Cluster))

L\_national<-m\_arima\_covid %>% select(arima\_covid\_mint) %>% forecast(test\_set) %>%

filter(is\_aggregated(Cluster), is\_aggregated(Hospital))

### Function accuracy for each level

fun\_acc\_lvl<- function(DF, L){

DF %>%

# get the accuracy for each member of the level

fun\_acc() %>%

# take the average

fun\_acc\_avg() %>%

# add level id

mutate(Level=paste(L), .before=1)

}

bind\_rows(fun\_acc\_lvl(L\_hospital, "Hospital"),

fun\_acc\_lvl(L\_cluster, "Cluster"),

fun\_acc\_lvl(L\_national, "National"))

## # A tibble: 3 x 4

## Level .model avg\_rmse avg\_mae

## <chr> <chr> <dbl> <dbl>

## 1 Hospital arima\_covid\_mint 700. 637.

## 2 Cluster arima\_covid\_mint 1123. 959.

## 3 National arima\_covid\_mint 1190. 971.