1. Recap

1. Tune again

Modelling Retuning

* 1. Retune Random Forest
  2. Retune Prophet boost
  3. Performance (after retuning) 3 Ensemble

3.1 Peformance (ensemble) 4 Performance (individual levels)

Hospital Cluster level National level

1. The future

Hospital Cluster National

1. KIV Plans

Errors

# Recap

The aim of this series of blog is to predict monthly admissions to Singapore public acute adult hospitals. The admissions were treated as a hierarchical time series as every country has a hierarchical order to its public hospitals including Singapore. The levels are:

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.)

Admissions were forecasted at each level for Mar 21- Dec 21

library(tidyverse) library(tidymodels) library(timetk) library(modeltime) library(modeltime.ensemble)

# dataset (training data and future forecast data and pre-processing recipes were done in the past post. cv too) The output was uploaded onto my github.

url\_datasets<-url("https://github.com/notast/hierarchical-forecasting/blob/ main/5Data\_CV.RData?raw=true")

load(url\_datasets) close(url\_datasets)

to\_predictfuture %>% distinct(Date) ## # A tibble: 10 x 1

|  |  |  |
| --- | --- | --- |
| ## |  | Date |
| ## |  | <date> |
| ## | 1 | 2021-03-01 |
| ## | 2 | 2021-04-01 |
| ## | 3 | 2021-05-01 |
| ## | 4 | 2021-06-01 |
| ## | 5 | 2021-07-01 |
| ## | 6 | 2021-08-01 |
| ## | 7 | 2021-09-01 |
| ## | 8 | 2021-10-01 |
| ## | 9 | 2021-11-01 |
| ## | 10 | 2021-12-01 |

The training dataset was from Jan 16 – Apr 20.

training(splits) %>% distinct(Date) %>% tail(10)

|  |  |  |
| --- | --- | --- |
| ## | # | A tibble: 10 x 1 |
| ## |  | Date |
| ## |  | <date> |
| ## | 1 | 2019-07-01 |
| ## | 2 | 2019-08-01 |
| ## | 3 | 2019-09-01 |
| ## | 4 | 2019-10-01 |
| ## | 5 | 2019-11-01 |
| ## | 6 | 2019-12-01 |
| ## | 7 | 2020-01-01 |
| ## | 8 | 2020-02-01 |
| ## | 9 | 2020-03-01 |
| ## | 10 | 2020-04-01 |

The testing dataset was from May 20- Feb 21.

testing(splits) %>% distinct(Date) %>% tail(10)

|  |  |  |
| --- | --- | --- |
| ## | # | A tibble: 10 x 1 |
| ## |  | Date |
| ## |  | <date> |
| ## | 1 | 2020-05-01 |
| ## | 2 | 2020-06-01 |
| ## | 3 | 2020-07-01 |
| ## | 4 | 2020-08-01 |
| ## | 5 | 2020-09-01 |
| ## | 6 | 2020-10-01 |
| ## | 7 | 2020-11-01 |
| ## | 8 | 2020-12-01 |
| ## | 9 | 2021-01-01 |
| ## | 10 | 2021-02-01 |

The admissions were forecasted using:

* 1. Classical Approach:

ii. Traditional techniques e.g. bottoms up

ii. Newer techniques e.g. reconciliation were employed

* 1. Machine Learning Approach
     1. The best combination of features to be used for machine learning were screened
     2. Several machine learning techniques were trial
     3. In this post, the best models (Random Forest and Prophet Boost) were retuned and ensemble model was created to achieve better performance

# tune grid and wf

url\_TgWf<- url("https://github.com/notast/hierarchical-forecasting/blob/main/ 5Retunning\_objects.RData?raw=true")

load(url\_TgWf) close(url\_TgWf)

# finalized and fitted wf

url\_ffwf<- url("https://github.com/notast/hierarchical-forecasting/blob/main/5FFWf. RData?raw=true")

load(url\_ffwf) close(url\_ffwf)

# metric

metrics\_custom= metric\_set( rmse, mae

# Tune again

Modelling and retuning is easy with tidymodels

**Modelling**

* 1. Set up the model
  2. Add the recipe and model into a workflow
  3. Tune the workflow which in turn tunes parameters of the recipe and/or the model inside the workflow
  4. Finalize the workflow with the best tuned parameters
  5. Fit the finalized workflow with its best tuned recipe and best tuned model onto the whole training data.

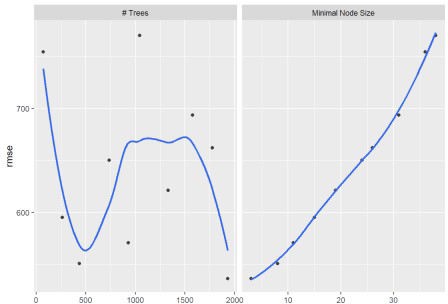
## Retuning

* 1. Plot the current parameters against its corresponding rmse
  2. Identify a range of parameters which correspond to better rmse and create an updated set of parameter information.
  3. Retune the workflow with the updated set of parameter information.
  4. Finalized the updated workflow with the best retuned parameters.
  5. Fit the new finalized workflow onto the whole training data.

## Retune Random Forest

rf\_t %>% autoplot(metric="rmse") + geom\_smooth(se=F)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



param\_rf2<- rf\_wf %>% parameters() %>% update(trees=dials::trees(range=c(1600, 2500)),

min\_n= dials::min\_n(range(2, 9)))

set.seed(69) rf2\_t<- tune\_grid(

object= rf\_wf, resamples = folds, param\_info = param\_rf2, grid=20,

metrics = metrics\_custom)

fun\_fwf<- function(wf, t\_wf){ finalize\_workflow(

x= wf,

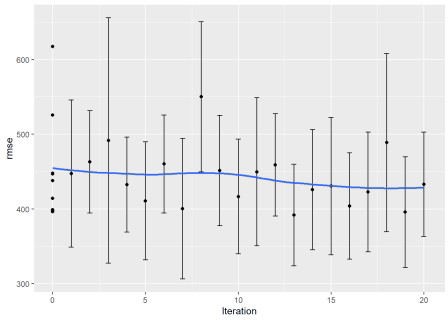
parameters= select\_best(t\_wf, "rmse")) %>% fit(splits\_train)

}

rf2\_f<-fun\_fwf(rf\_wf, rf2\_t)

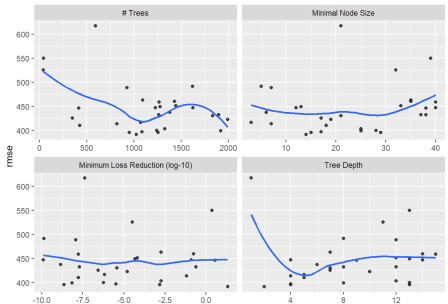
## Retune Prophet boost

pb\_t %>% autoplot(metric="rmse", "performance") + geom\_smooth(se=F) ## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



pb\_t %>% autoplot(metric="rmse") + geom\_smooth(se=F)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



param\_pb2<- pb\_wf %>% parameters() %>% update(min\_n= dials::min\_n(range=c(10,30)),

tree\_depth= dials::tree\_depth(range=c(3,10)))

all\_cores <- parallel::detectCores(logical = FALSE) library(doParallel)

cl <- makePSOCKcluster(all\_cores) registerDoParallel(cl)

set.seed(69)

pb2\_t<- tune\_bayes( object= pb\_wf, resamples = folds,

iter = 25, initial = 9,

param\_info = param\_pb2, metrics = metrics\_custom,

control = control\_bayes(no\_improve = 25)) pb2\_f<- fun\_fwf(pb\_wf, pb2\_t)

# save retuned finalized and fitted wf

save(rf2\_f, pb2\_f, file = "6Retuned\_FinalFitWf.RData")

## Performance (after retuning)

Both Prophet Boost and Random Forest benefited from retuning. Prophet Boost benefited the most as XGB performs more optimally when its parameters are tweak.

e2\_table<-modeltime\_table(rf\_f, pb\_f,rf2\_f, pb2\_f) %>% update\_model\_description(1, "RF") %>% update\_model\_description(2,

"PB") %>%

update\_model\_description(3, "RF\_retuned") %>% update\_model\_description(4, "PB\_retuned")

e2\_cal<-e2\_table %>% modeltime\_calibrate(testing(splits)) e2\_cal %>% modeltime\_accuracy(metric\_set = metrics\_custom) %>% arrange(rmse, sort=T)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ##  ## | # | A tibble:  .model\_id | 4 x 5  .model\_desc | .type | rmse | mae |
| ## |  | <int> | <chr> | <chr> | <dbl> | <dbl> |
| ## | 1 | 3 | RF\_retuned | Test | 545. | 409. |
| ## | 2 | 1 | RF | Test | 549. | 410. |
| ## | 3 | 4 | PB\_retuned | Test | 945. | 673. |
| ## | 4 | 2 | PB | Test | 1137. | 799. |

# Ensemble

Both tuned and original Random Forests were trial in the ensemble as the performance between the models were minimal. The retuned Prophet Boost was nominated to be the default Prophet Boost for the ensemble as its accuracy was markedly better than the original version. As Random Forest performed much better than Prophet Boost, the weightage given to Random Forest was at least 80%.

fun\_ensemble<- function(M1, W1, W2){ ensemble\_table<-e2\_table %>%

filter(.model\_desc %in% c(M1, "PB\_retuned")) %>% ensemble\_weighted(loadings = c(W1, W2)) %>% modeltime\_table() %>%

modeltime\_accuracy(testing(splits), metric\_set = metrics\_custom)

ensemble\_table %>%

mutate(Model1\_wt= paste0(M1,"::",W1 ), Model2\_wt=paste0("PB\_retuned::", W2)) %>%

select(Model1\_wt, Model2\_wt, rmse, mae)

}

ensemble\_rmse<-bind\_rows(fun\_ensemble("RF\_retuned", 9,1),

fun\_ensemble("RF\_retuned", 8,2),

fun\_ensemble("RF", 9,1),

fun\_ensemble("RF", 8,2))

## 3.1 Peformance (ensemble)

All the ensemble models performed better than its member models. Better performing ensemble models had a stronger bias to Random Forest.

ensemble\_rmse%>% arrange(rmse,sort=T) ## # A tibble: 4 x 4

## Model1\_wt Model2\_wt rmse mae ## <chr> <chr> <dbl> <dbl> ## 1 RF\_retuned::9 PB\_retuned::1 535. 412.

## 2 RF::9 PB\_retuned::1 538. 412.

## 3 RF\_retuned::8 PB\_retuned::2 539. 421.

## 4 RF::8 PB\_retuned::2 543. 423.

e3\_ensembled\_table <- e2\_table %>%

filter(.model\_desc %in% c("RF\_retuned", "PB\_retuned")) %>% ensemble\_weighted(loadings = c(9, 1)) %>% modeltime\_table()

# Performance (individual levels)

The performance above was for all hierarchical levels, here the performance for each specific level was reviewed and visualized.

e3\_ensembled\_cal <-modeltime\_calibrate(e3\_ensembled\_table, new\_data = testing(splits))

modeltime\_accuracy and was unable to evaluate the accuracy for individual levels, a customised function was built.

e3\_ensembled\_fcast<- e3\_ensembled\_cal %>% modeltime\_forecast(new\_data = testing(splits), actual\_data =

to\_train, keep\_data = T) %>% filter(.key!="actual") %>%

select(c(Date, Level, Name, Admission, .value, .conf\_lo, .conf\_hi))

fun\_lvlacc<- function(L){ e3\_ensembled\_fcast %>%

filter(Level==L) %>% metrics\_custom(truth=.value, estimate=Admission)

}

fun\_lvlplt<- function(DF, L, TT){ DF %>% filter(Level==L) %>%

ggplot(aes(Date))+ geom\_line(aes(y=Admission), col="blue")+ geom\_line(aes(y=.value), col="red")+

geom\_ribbon(aes(ymin=.conf\_lo, ymax=.conf\_hi), fill = "red", alpha = 0.1)+

facet\_wrap(vars(Name), scales = "free\_y") + labs(title = paste(TT))

}

## Hospital

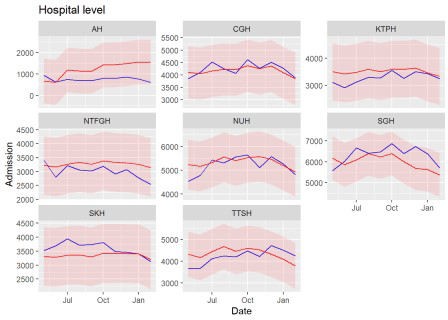
(fun\_lvlacc("Hospital\_id")) ## # A tibble: 2 x 3

## .metric .estimator .estimate ## <chr> <chr> <dbl>

## 1 rmse standard 393.

## 2 mae standard 321.

(fun\_lvlplt(e3\_ensembled\_fcast, "Hospital\_id", "Hospital level"))



## Cluster level

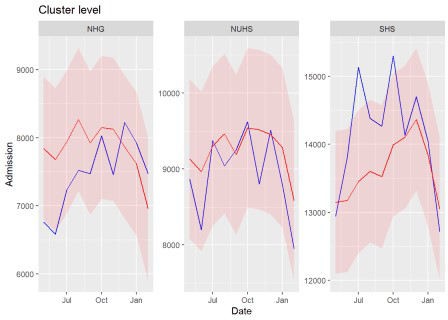
(fun\_lvlacc("Cluster\_id")) ## # A tibble: 2 x 3

## .metric .estimator .estimate ## <chr> <chr> <dbl>

## 1 rmse standard 657.

## 2 mae standard 528.

(fun\_lvlplt(e3\_ensembled\_fcast, "Cluster\_id", "Cluster level"))



## National level

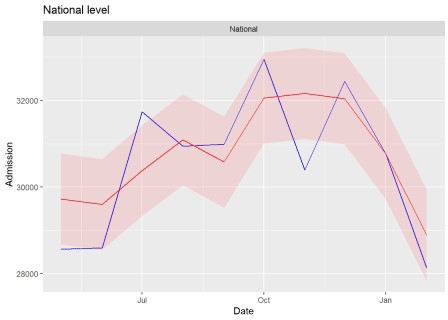
(fun\_lvlacc("National\_id")) ## # A tibble: 2 x 3

## .metric .estimator .estimate ## <chr> <chr> <dbl>

## 1 rmse standard 949.

## 2 mae standard 789.

(fun\_lvlplt(e3\_ensembled\_fcast, "National\_id", "National level"))



# The future

Below are the forecasted future admissions for Mar 21- Dec 21.

Future<-modeltime\_refit(e3\_ensembled\_cal, to\_train, control = control\_refit(allow\_par = T)) %>%

modeltime\_forecast(new\_data = to\_predictfuture, actual\_data =

to\_train, keep\_data = T)

fun\_lvlplt\_Future<- function(L){ Future%>%

filter\_by\_time(.start\_date = last(Date) %-time% "24 month", .end\_date

= "end") %>%

filter(Level==L) %>% group\_by(Name) %>% plot\_modeltime\_forecast(

.facet\_ncol = 3,

.conf\_interval\_show = TRUE,

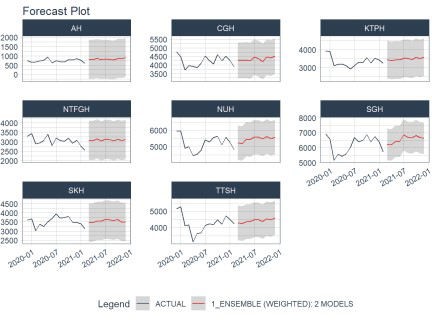
.interactive = F) + guides(x = guide\_axis(angle = 30))

}

## Hospital

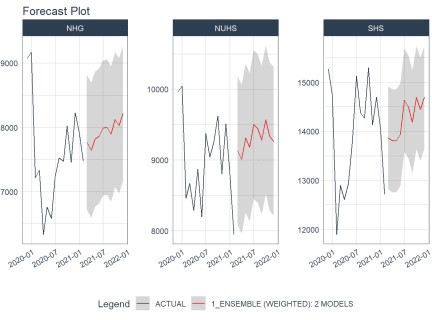
Most of the hospital level forecast were relatively flat; perhaps, due to insufficient data compared to cluster and national level which had more observations from aggregation of subordinate levels.

fun\_lvlplt\_Future("Hospital\_id")



## Cluster

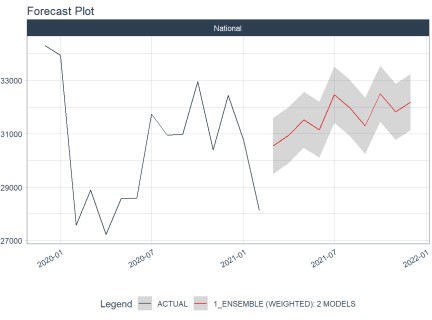
fun\_lvlplt\_Future("Cluster\_id")



## National

The forecast for cluster and national level appeared more plausible with some peaks and dips with an upward trend. Nonetheless, forecasting during this Covid period is challenging. Any forecast can be thrown off the rails as the situation is erratic and dynamic.

fun\_lvlplt\_Future("National\_id")



# KIV Plans

The modeltime package has tremendously facilitated time series forcasting the tidyverse

way. The forecasting could have been improved by

trying a greater variety of machine learning techniques offered by tidymodels

replacing XGB in Prophet Boost with other tree-based boost models like Catboost or

lightGBM

using deep learning.

**Errors**

A stacked ensembled was attempted but there was an error with resample predictions which was needed before feeding into ennsemble\_model\_spec.

r\_resamples<-splits\_train %>%

time\_series\_cv(date\_var=Date, assess = 10, cumulative = T, skip =

5)

## Overlapping Timestamps Detected. Processing overlapping time series together using sliding windows.

e\_table<-modeltime\_table(glm\_f, mars\_f, rf\_f, xgb\_f, pb\_f)

r\_fitresamples <- e\_table %>% modeltime\_fit\_resamples(r\_resamples, control = control\_resamples(allow\_par = F, save\_pred = T,

verbose = T))

## -- Fitting Resamples

## \* Model ID: 1 GLMNET

## i Slice1: preprocessor 1/1

## x Slice1: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice2: preprocessor 1/1

## x Slice2: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice3: preprocessor 1/1

## x Slice3: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice4: preprocessor 1/1

## x Slice4: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice5: preprocessor 1/1

## x Slice5: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice6: preprocessor 1/1

## x Slice6: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice7: preprocessor 1/1

## x Slice7: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice8: preprocessor 1/1

## x Slice8: preprocessor 1/1: Error: Only one factor level in Covid: no

## \* Model ID: 2 EARTH

## i Slice1: preprocessor 1/1

## x Slice1: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice2: preprocessor 1/1

## x Slice2: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice3: preprocessor 1/1

## x Slice3: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice4: preprocessor 1/1

## x Slice4: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice5: preprocessor 1/1

## x Slice5: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice6: preprocessor 1/1

## x Slice6: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice7: preprocessor 1/1

## x Slice7: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice8: preprocessor 1/1

## x Slice8: preprocessor 1/1: Error: Only one factor level in Covid: no

## \* Model ID: 3 RANGER

## i Slice1: preprocessor 1/1

## x Slice1: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice2: preprocessor 1/1

## x Slice2: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice3: preprocessor 1/1

## x Slice3: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice4: preprocessor 1/1

## x Slice4: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice5: preprocessor 1/1

## x Slice5: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice6: preprocessor 1/1

## x Slice6: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice7: preprocessor 1/1

## x Slice7: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice8: preprocessor 1/1

## x Slice8: preprocessor 1/1: Error: Only one factor level in Covid: no

## \* Model ID: 4 XGBOOST

## i Slice1: preprocessor 1/1

## x Slice1: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice2: preprocessor 1/1

## x Slice2: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice3: preprocessor 1/1

## x Slice3: preprocessor 1/1: Error: Only one factor level in Covid:

no

## i Slice4: preprocessor 1/1

## x Slice4: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice5: preprocessor 1/1

## x Slice5: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice6: preprocessor 1/1

## x Slice6: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice7: preprocessor 1/1

## x Slice7: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice8: preprocessor 1/1

## x Slice8: preprocessor 1/1: Error: Only one factor level in Covid: no

## \* Model ID: 5 PROPHET W/ XGBOOST ERRORS

## i Slice1: preprocessor 1/1

## x Slice1: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice2: preprocessor 1/1

## x Slice2: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice3: preprocessor 1/1

## x Slice3: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice4: preprocessor 1/1

## x Slice4: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice5: preprocessor 1/1

## x Slice5: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice6: preprocessor 1/1

## x Slice6: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice7: preprocessor 1/1

## x Slice7: preprocessor 1/1: Error: Only one factor level in Covid: no

## i Slice8: preprocessor 1/1

## x Slice8: preprocessor 1/1: Error: Only one factor level in Covid: no

## 6.99 sec elapsed modeltime\_resample\_accuracy( r\_fitresamples

## Error: Only strings can be converted to symbols # snap shot of resamples

#r\_resamples %>% tk\_time\_series\_cv\_plan() %>% filter(Name== "CGH") %>% filter(.id %in% c("Slice1","Slice2","Slice3","Slice8")) %>% group\_by(.id) %>% plot\_time\_series(.date\_var = Date, .value= Admission, .interactive=F, .color\_var= `.key`, .smooth=F)

The error with calculating the accuracy was likely due to missing predictions when the resample was fitted due to the Covid dummy variable. The peak period of Covid was only from Jan-Jul 20

thus there would have been plenty of samples with only no Covid peak periods. One alternative was to omit Covid as a predictor but as the pandemic still looms on, it was more appropriate to retain this variable when forecasting.

r\_fitresamples$.resample\_results[[1]]$.predictions ## [[1]]

## NULL ##

## [[2]] ## NULL ##

## [[3]] ## NULL ##

## [[4]] ## NULL ##

## [[5]] ## NULL ##

## [[6]] ## NULL ##

## [[7]] ## NULL ##

## [[8]] ## NULL

r\_fitresamples$.resample\_results[[2]]$.predictions ## [[1]]

## NULL ##

## [[2]] ## NULL ##

## [[3]] ## NULL ##

## [[4]] ## NULL ##

## [[5]] ## NULL ##

## [[6]] ## NULL ##

## [[7]] ## NULL ##

## [[8]] ## NULL