Predicting MIGROS Yogurts Sales

Natalie Borter Beatrice Stierli Sinah Schüpfer Adrian Peng
Loris Lindemann Raphael Suter Jonathan Hahn

20.10.2021

```
# Setting working directory to the directory of the file
knitr::opts_knit$set(root.dir = getwd())
source("collecting_data.R")

set.seed(123)

# define the keywords for google trend analysis
keyword.vec<-c("Joghurt", "gesund essen")

data.folder <-("data_hs21")
pred.data <-FALSE
# list of all yogurts with own data frame for each yogurt
dfs<-load.data.frames(keyword.vec, data.folder, pred.data)

for (i in seq(1, length(dfs))){
  xx<-dfs[[i]][,c(1:21)]
  colnames(xx) <- xx %>%
    colnames(.) %>%
    gsub(".x", "", .)
  dfs[[i]]<-xx
  }
```

Describe the dependent variable: waiting times and some of the key predictors

```
## define two functions to assess model fit

## rmse = root mean squared error
rmse<-function(actual, predicted){
  round((sum((actual-predicted)^2)/length(actual))^.5,2)
}
## mean absolute percentage error
mape<-function(actual, predicted){
  round(mean(100*abs((actual-predicted)/actual)),2)
}
```

Prediction of unknown Regressors

- milk prices

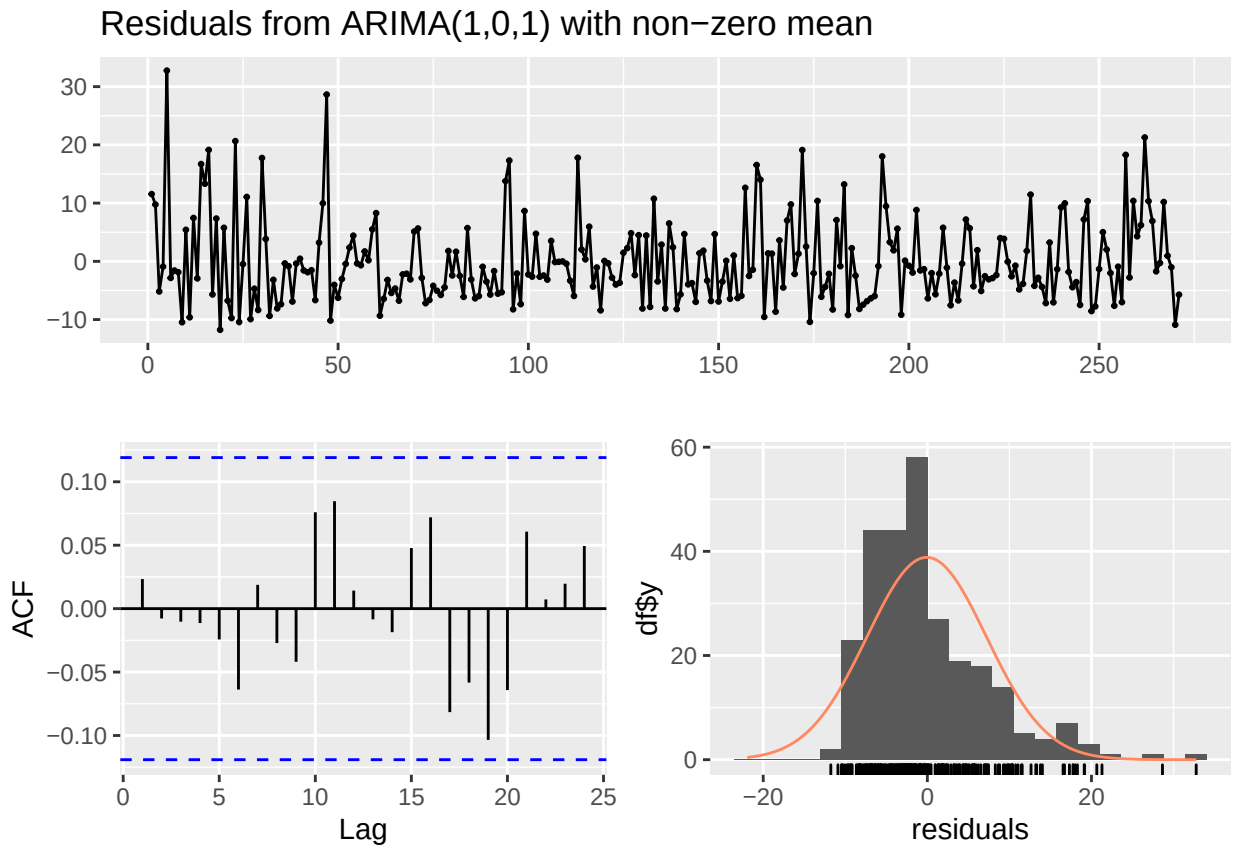
- yogurt prices
- google trends

```
library(forecast)

gesund<-dfs[[1]]$gesund.essen
Joghurt<-dfs[[1]]$Joghurt

joghurt.price<-dfs[[1]]$kon.yougurtpr.180.fru
milk.price<-dfs[[1]]$prod.milchpreis.a

ntest <- 15 # measurements months

len <- length(gesund)
```

Predict the Google trends value of the key word "gesund essen".

Predict the future google trends value for the keyword "gesund essen" in Switzerland.

```
trainingg<-gesund[c(1:(len-ntest))]
testingg<-gesund[c((len-ntest+1):len)]
arima.ges<-auto.arima(trainingg, ic = "aic")
ges.pred <- predict(arima.ges, n.ahead = ntest)
checkresiduals(arima.ges)
```

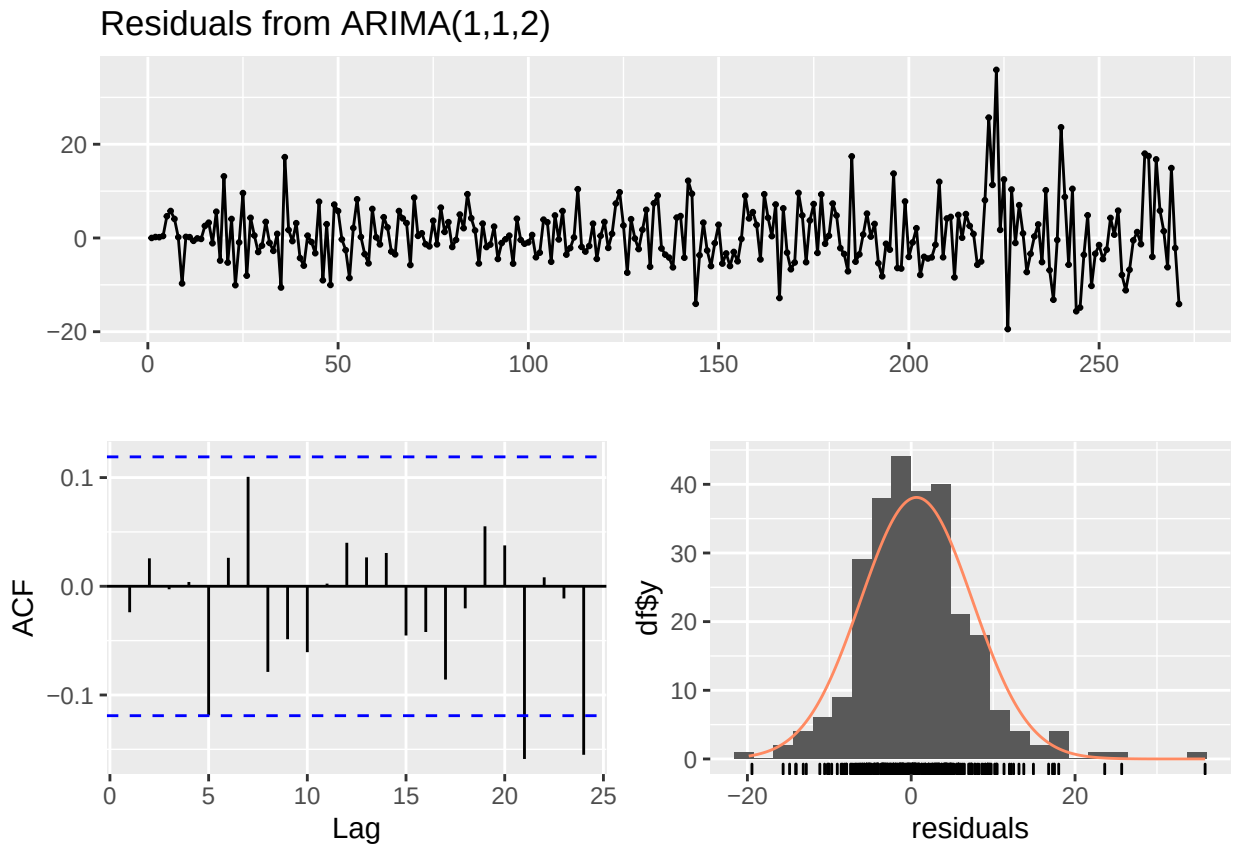


```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1) with non-zero mean
## Q* = 3.9678, df = 7, p-value = 0.7835
##
## Model df: 3.   Total lags used: 10
```

Predict the Google trends value of the key word "Joghurt".

Predict the future google trends value for the keyword "Joghurt" in Switzerland.

```
trainingJ<-Joghurt[c(1:(len-ntest))]
testingJ<-Joghurt[c((len-ntest+1):len)]
arima.Jog<-auto.arima(trainingJ, ic = "aic")
Jog.pred <- predict(arima.Jog, n.ahead = ntest)
checkresiduals(arima.Jog)
```

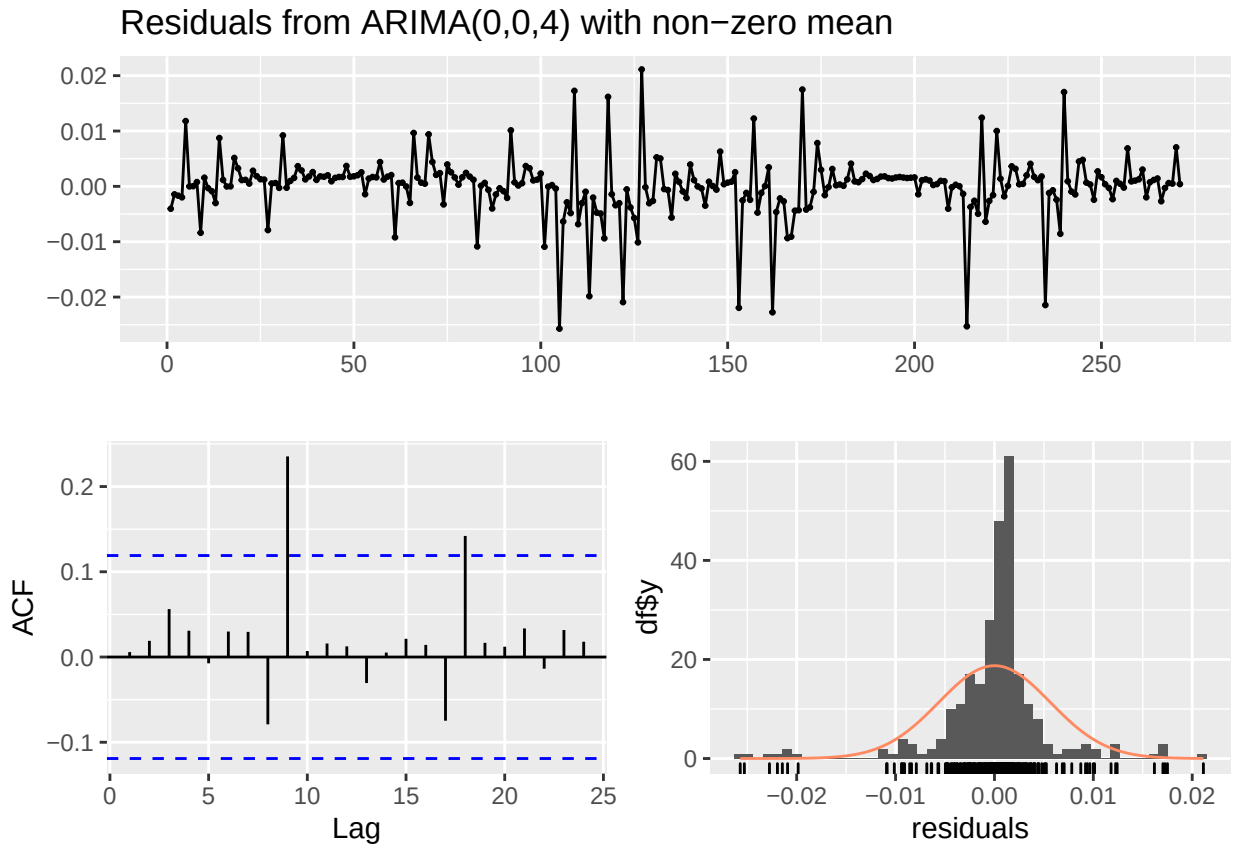



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,2)
## Q* = 10.756, df = 7, p-value = 0.1496
##
## Model df: 3.   Total lags used: 10
```

Predict the yogurt price

Predict the future yogurt price in Switzerland.

```
trainingJP<-joghurt.price[c(1:(len-ntest))]
testingJP<-joghurt.price[c((len-ntest+1):len)]
arima.JP<-auto.arima(trainingJP, ic = "aic")
JP.pred <- predict(arima.JP, n.ahead = ntest)
checkresiduals(arima.JP)
```



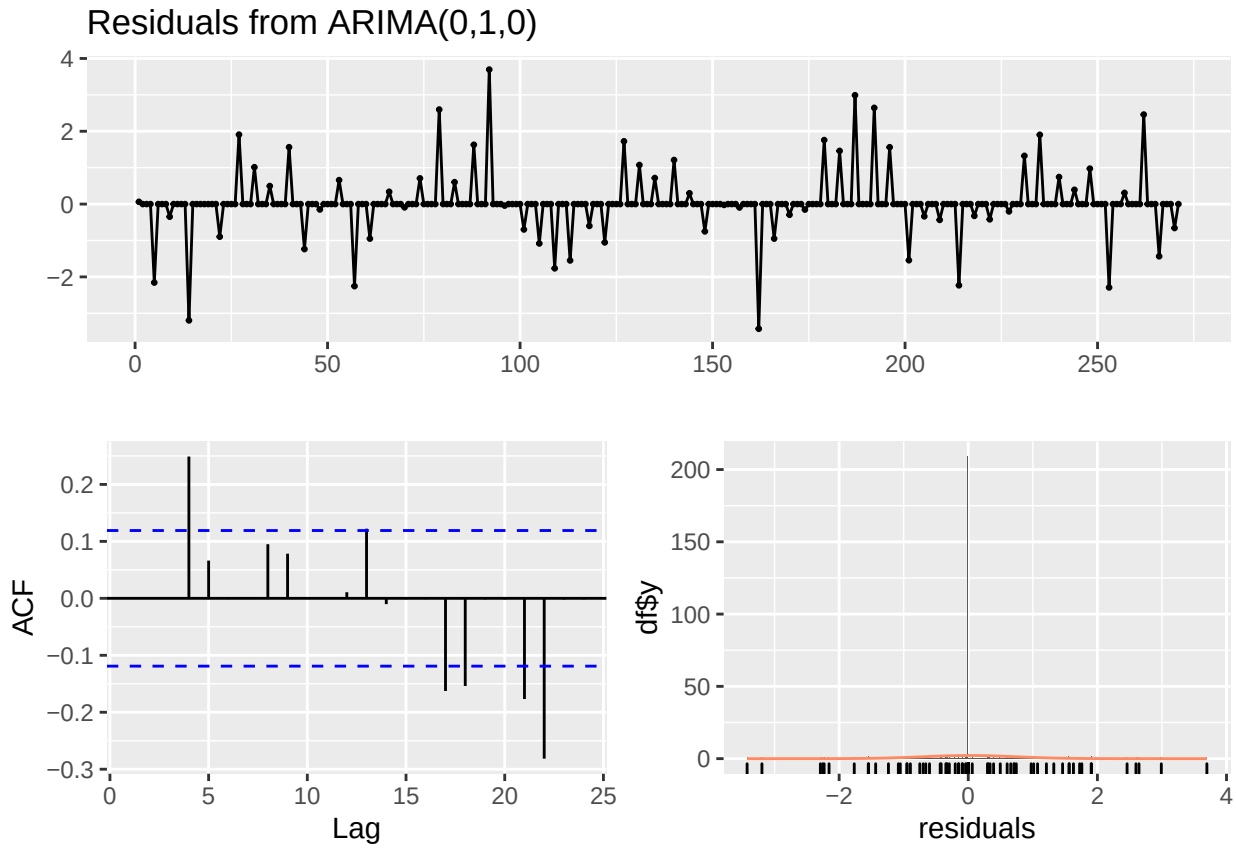

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,0,4) with non-zero mean
## Q* = 19.16, df = 5, p-value = 0.001795
##
## Model df: 5.   Total lags used: 10
```

Predict the milk price

Predict the future milk price in Switzerland.

```
trainingMP<-milk.price[c(1:(len-ntest))]
testingMP<-milk.price[c((len-ntest+1):len)]
```

```
arima.MP<-auto.arima(trainingMP, ic = "aic")
MP.pred <- predict(arima.MP, n.ahead = ntest)
checkresiduals(arima.MP)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,0)
## Q* = 22.67, df = 10, p-value = 0.01203
##
## Model df: 0.   Total lags used: 10
```

Create a series of known values and predicted ones.

```
ts.pred.gesund <- c(trainingg, as.numeric(ges.pred$pred))
ts.pred.joghurt <- c(trainingJ, as.numeric(Jog.pred$pred))
ts.pred.joghurt.price <- c(trainingJP, as.numeric(JP.pred$pred))
ts.pred.milk.price <- c(trainingMP, as.numeric(MP.pred$pred))
```

Apply each model to each yogurt

A loop that applies each model to each yogurt. The relevant data is stored for comparison.

```
library(party)
library(prophet)
library(tidyverse)
library(xgboost)

## Define data frame for each model

d_lm = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

d = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

d_glm = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

d_prop1 = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

d_prop2 = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

d_xgboost = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

d_factorM = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
```

```
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

d_factorR = data.frame(prod = rep(0,19),
                  model = rep(0,19),
                  rmse_t=rep(0,19),
                  rmse_test=rep(0,19),
                  mape_t=rep(0,19),
                  mape_test=rep(0,19))

prediction <- list()
prophet_plots <- list()
```

```
## Loop over each yogurt and apply data on each model

for(i in seq(1,length(dfs))){

  ## replace Null values
  dfs[[i]][is.na(dfs[[i]])] <- 0

  ## add prediction of Gtrends
  dfs[[i]]["pred.joghurt"]<-ts.pred.joghurt
  dfs[[i]]["pred.gesund"]<-ts.pred.gesund

  ## add prediction of milk and yogurt prices
  dfs[[i]]["pred.joghurt.price"]<-ts.pred.joghurt.price
  dfs[[i]]["pred.milk.price"]<-ts.pred.milk.price

  df_final<-dfs[[i]]

  y.name<-df_final$article_name[1]
  print(paste("Train Models for", y.name, sep = " "))

  ## create lockdown
  ld_start = as.Date("08.04.2020", format="%d.%m.%Y")
  ld_end = as.Date("26.04.2020", format="%d.%m.%Y")
  ld<-data.frame("Date" = seq(ld_start,ld_end, by = 'days'), row.names = )
  ld[,"Bezeichnung"]<-"Lockdown"

  #events<-rbind(ld, hG_factoreneva)
  events <- ld

  h<-data_frame(
    holiday = events[,"Bezeichnung"],
    ds = as.Date(events[,"Date"]),
    lower_window = 0,
    upper_window = 1
  )

  ## refactor columns for prophet package
  A<-df_final[,c("yrwk_start", "sales")]
  names(A)<-c("ds","y")
```

```r
AL<-df_final[,c("yrwk_start",
                "sales",
                "year",
                "month",
                "week",
                "promo_01",
                "promo_02",
                "promo_03",
                "promo_04",
                "promo_05",
                "pred.joghurt",
                "pred.gesund",
                "pred.joghurt.price",
                "pred.milk.price")]

names(AL)<-c("ds",
             "y",
             "year",
             "month",
             "week",
             "promo_01",
             "promo_02",
             "promo_03",
             "promo_04",
             "promo_05",
             "pred.joghurt",
             "pred.gesund",
             "pred.joghurt.price",
             "pred.milk.price")

## the test set is the last 4 weeks measured,
## the training set is everything else

len <- nrow(AL)
training<-AL[1:(len-ntest),]
testing<-AL[(len-ntest+1):len,]

## ------------------------------------------------------------------------

## Linear Model

mod<-lm(y~ds+
          week+
          month+
          year+
          as.numeric(promo_01)+
          as.numeric(promo_02)
        +as.numeric(promo_03)
        +as.numeric(promo_04)
        +as.numeric(promo_05)
        +pred.joghurt
        +pred.gesund
        +pred.joghurt.price
```

```
        +pred.milk.price
        +pred.milk.price, data=training)

testingA<-mod$coefficients[1]+mod$coefficients[2]*as.numeric(testing$week)+mod$coefficients[3]*as.num

testing$pred_LM <- predict(mod, newdata = testing)
training$pred_LM <-mod$fitted.values

prod <- df_final$article_name[1]
model <- "LM"
a <- c(rmse(training$y, training$pred_LM),rmse(testing$y, testing$pred_LM))
b <- c(mape(training$y, training$pred_LM),mape(testing$y, testing$pred_LM))
c <- c(prod,model,a,b)
d_lm[i, ] = c

## ------------------------------------------------------------------------

## GLM

fit_glm<-glm(y~ds+
             week +
             month+
             year+
             as.numeric(promo_01)+
             as.numeric(promo_02)+
             as.numeric(promo_03)+
             as.numeric(promo_04)+
             as.numeric(promo_05)
             #+pred.joghurt
             #+pred.gesund
             #+pred.joghurt.price
             #+pred.milk.price
             ,
             data=training,
             family="quasipoisson")
summary(fit_glm)
training$pred_GLM <-  exp(predict(fit_glm))
testing$pred_GLM <-  exp(predict(fit_glm, newdata = testing))

model <- "GLM"
a_glm <- c(rmse(training$y, training$pred_GLM),rmse(testing$y, testing$pred_GLM))
b_glm <- c(mape(training$y, training$pred_GLM),mape(testing$y, testing$pred_GLM))
c_glm <- c(prod,model,a_glm,b_glm)
d_glm[i, ] = c_glm

## ------------------------------------------------------------------------

## Regression Tree

## tree with party

tree <- ctree(y~#ds+ #ds taken out -> date is not supported
                week +
```

```
                month +
                year +
                promo_01+
                promo_02+
                promo_03+
                promo_04+
                promo_05
                +pred.joghurt
                +pred.gesund
                +pred.joghurt.price
                +pred.milk.price,
                data=training)

## show tree
#plot(tree)

training$pred_RegressionTree <-predict(tree)
testing$pred_RegressionTree <- predict(tree, newdata = testing)

model <- "RegressionTree"
a <- c(rmse(training$y, training$pred_RegressionTree),rmse(testing$y, testing$pred_RegressionTree))
b <- c(mape(training$y, training$pred_RegressionTree),mape(testing$y, testing$pred_RegressionTree))
c <- c(prod,model,a,b)
d[i, ] = c

## ------------------------------------------------------------------------

## Basic Prophet Model

## specify holidays on weekly basis;
## if holiday in a week the whole week is marked as holiday
## (because data is provided on a weekly basis)

years <- c(2016, 2017, 2018, 2019, 2020, 2021)
country.name <- 'CH'
df <- prophet:::make_holidays_df(years, country.name)

## make sure that each holiday hots over one week
df$lower_window<--4
df$upper_window<-4

##combine holidays with the data frame
df2<-select(df,"holiday","ds","lower_window","upper_window")

s<-rbind(h,df2)

m <- prophet(holidays=s, mcmc_samples=300,
             holidays_prior_scale=0.5,
              changepoint_prior_scale=0.01,
             yearly.seasonality=TRUE)
#m <- add_country_holidays(m, country_name = 'CH')
#m <- add_country_holidays(m, df)
m <- fit.prophet(m, training)
```

```
future <- make_future_dataframe(m, periods = ntest, freq = "week", include_history = TRUE)
fcst <- predict(m, future)
n<-nrow(training)
training$pred_Prophet1<-fcst$yhat[1:n]
testing$pred_Prophet1<-fcst$yhat[(n+1):(n+ntest)]

model <- "Prophet1"
a <- c(rmse(training$y, training$pred_Prophet1),rmse(testing$y, testing$pred_Prophet1))
b <- c(mape(training$y, training$pred_Prophet1),mape(testing$y, testing$pred_Prophet1))
c <- c(prod,model,a,b)
d_prop1[i, ] = c

## ------------------------------------------------------------------------

## Complex Prophet Model (with regressors)

m_h_r <- prophet(holidays=s, mcmc_samples=300,
           holidays_prior_scale=0.5,
           changepoint_prior_scale=0.01,
           #seasonality_mode='multiplicative',
           yearly.seasonality=TRUE,
           #weekly.seasonality=TRUE,
           #daily.seasonality=FALSE
           )
## add regressors to the model
m_h_r <- add_regressor(m_h_r, 'promo_01')
m_h_r <- add_regressor(m_h_r, "promo_02")
m_h_r <- add_regressor(m_h_r, 'promo_03')
m_h_r <- add_regressor(m_h_r, "promo_04")
m_h_r <- add_regressor(m_h_r, 'promo_05')
m_h_r <- add_regressor(m_h_r, "pred.joghurt")
m_h_r <- add_regressor(m_h_r, "pred.gesund")
m_h_r <- add_regressor(m_h_r, "pred.joghurt.price")
m_h_r <- add_regressor(m_h_r, "pred.milk.price")

m_h_r <- fit.prophet(m_h_r, training)

## create the data frame for the prediction
future <- make_future_dataframe(m_h_r, periods = ntest, freq = "week", include_history = TRUE)

## ad actual values to the prediction
future$promo_01<- AL$promo_01
future$promo_02<- AL$promo_02
future$promo_03<- AL$promo_03
future$promo_04<- AL$promo_04
future$promo_05<- AL$promo_05
future$pred.joghurt<- AL$pred.joghurt
future$pred.gesund<- AL$pred.gesund
future$pred.joghurt.price<- AL$pred.joghurt.price
future$pred.milk.price<- AL$pred.milk.price

## predict
fcst_h_r <- predict(m_h_r, future)
```

```
n<-nrow(training)

training$pred_Prophet2<-fcst_h_r$yhat[1:n]
testing$pred_Prophet2<-fcst_h_r$yhat[(n+1):(n+ntest)]

model <- "Prophet2"
a <- c(rmse(training$y, training$pred_Prophet2),rmse(testing$y, testing$pred_Prophet2))
b <- c(mape(training$y, training$pred_Prophet2),mape(testing$y, testing$pred_Prophet2))
c <- c(prod,model,a,b)
d_prop2[i, ] = c

## ------------------------------------------------------------------------

## XGBOOST

## We need to provide the x and y variables
## separately and in the following format:

#training
X <- AL%>%
  select(-y,
         -ds
         )%>%
  mutate(across(where(is.factor), as.numeric))%>%
  as.matrix(.)

X_train <- X[1:(len-ntest),]

y <- AL[1:(len-ntest),]%>%
  select(y) %>%
  as.matrix(.)

## Train model
fit.gbt <- xgboost(data = X_train,
                   label = y,
                   nrounds=60,
                   objective = "count:poisson",
                   verbose = 0)

## Make predictions on training data
pred_gbt_train <- predict(fit.gbt, X[1:(len-ntest),])
## Make predictions on test data
pred_gbt_test <- predict(fit.gbt, X[(len-ntest+1):len,])

## Which variables are important?
#require(ggplot2)
#library(Ckmeans.1d.dp)
#importance=xgb.importance(colnames(X), model = fit.gbt)
#xgb.ggplot.importance(importance)

training$pred_XGBOOST<-pred_gbt_train
testing$pred_XGBOOST<-pred_gbt_test
```

```
c(rmse(training$y, training$pred_XGBOOST),rmse(testing$y, testing$pred_XGBOOST))
c(mape(training$y, training$pred_XGBOOST),mape(testing$y, testing$pred_XGBOOST))

model <- "XGBOOST"
a <- c(rmse(training$y, training$pred_XGBOOST),rmse(testing$y, testing$pred_XGBOOST))
b <- c(mape(training$y, training$pred_XGBOOST),mape(testing$y, testing$pred_XGBOOST))
c <- c(prod,model,a,b)
d_xgboost[i, ] = c

## ------------------------------------------------------------------------

## Prophet with factor scores

## WHATCH OUT! y is replaced with factor scores to apply the prophet model

AL$y.MCLASSI<-df_final$MClassi
training$y<-AL$y.MCLASSI[1:(len-ntest)]
testing$y<-AL$y.MCLASSI[(len-ntest+1):len]

m_h_r <- prophet(holidays=s, mcmc_samples=300,
                 holidays_prior_scale=0.5,
                 changepoint_prior_scale=0.01,
                 #seasonality_mode='multiplicative',
                 yearly.seasonality=TRUE,
                 #weekly.seasonality=TRUE,
                 #daily.seasonality=FALSE
)

m_h_r <- add_regressor(m_h_r, 'promo_01')
m_h_r <- add_regressor(m_h_r, "promo_02")
m_h_r <- add_regressor(m_h_r, 'promo_03')
m_h_r <- add_regressor(m_h_r, "promo_04")
m_h_r <- add_regressor(m_h_r, 'promo_05')
m_h_r <- add_regressor(m_h_r, "pred.joghurt")
m_h_r <- add_regressor(m_h_r, "pred.gesund")
m_h_r <- add_regressor(m_h_r, "pred.joghurt.price")
m_h_r <- add_regressor(m_h_r, "pred.milk.price")
m_h_r <- fit.prophet(m_h_r, training)

future <- make_future_dataframe(m_h_r, periods = ntest, freq = "week", include_history = TRUE)

future$promo_01<- AL[,c("promo_01")]
future$promo_02<- AL[,c("promo_02")]
future$promo_03<- AL[,c("promo_03")]
future$promo_04<- AL[,c("promo_04")]
future$promo_05<- AL[,c("promo_05")]
future$pred.joghurt<- AL[,c("pred.joghurt")]
future$pred.gesund<- AL[,c("pred.gesund")]
future$pred.joghurt.price<- AL[,c("pred.joghurt.price")]
future$pred.milk.price<- AL[,c("pred.milk.price")]

fcst_h_r <- predict(m_h_r, future)
```

```
n<-nrow(training)

training$pred_ProphetScoresM<-fcst_h_r$yhat[1:n]
testing$pred_ProphetScoresM<-fcst_h_r$yhat[(n+1):(n+ntest)]

sales<-df_final$sales
scores<-c(training$y,testing$pred_ProphetScoresM)

mod<-lm(sales~scores)
#diff<-mod$fitted.values-sales
training$pred_ProphetScoresM<-mod$fitted.values[1:(len-ntest)]
testing$pred_ProphetScoresM<-mod$fitted.values[(len-ntest+1):len]

c(rmse(sales[1:(len-ntest)],mod$fitted.values[1:(len-ntest)]))
c(rmse(sales[(len-ntest+1):len],mod$fitted.values[(len-ntest+1):len]))

model <- "ProphetScoresM"
a <- c(rmse(sales[1:(len-ntest)],mod$fitted.values[1:(len-ntest)]),
       rmse(sales[(len-ntest+1):len],mod$fitted.values[(len-ntest+1):len]))
b <- c(mape(sales[1:(len-ntest)],mod$fitted.values[1:(len-ntest)]),
       mape(sales[(len-ntest+1):len],mod$fitted.values[(len-ntest+1):len]))
c <- c(prod,model,a,b)
d_factorM[i, ] = c

#plot(m_h_r, fcst_h_r) + add_changepoints_to_plot(m_h_r)
#prophet_plot_components(m_h_r,fcst_h_r, render_plot = TRUE)


## ------------------------------------------------------------------------

## Prophet with factor scores Rest
AL$y.Rest<-df_final$Rest
training$y<-AL$y.Rest[1:(len-ntest)]
testing$y<-AL$y.Rest[(len-ntest+1):len]

m_h_r <- prophet(holidays=s, mcmc_samples=300,
                 holidays_prior_scale=0.5,
                 changepoint_prior_scale=0.01,
                 #seasonality_mode='multiplicative',
                 yearly.seasonality=TRUE,
                 #weekly.seasonality=TRUE,
                 #daily.seasonality=FALSE
)

m_h_r <- add_regressor(m_h_r, 'promo_01')
m_h_r <- add_regressor(m_h_r, "promo_02")
m_h_r <- add_regressor(m_h_r, 'promo_03')
m_h_r <- add_regressor(m_h_r, "promo_04")
m_h_r <- add_regressor(m_h_r, 'promo_05')
m_h_r <- add_regressor(m_h_r, "pred.joghurt")
m_h_r <- add_regressor(m_h_r, "pred.gesund")
m_h_r <- add_regressor(m_h_r, "pred.joghurt.price")
m_h_r <- add_regressor(m_h_r, "pred.milk.price")
```

```
  m_h_r <- fit.prophet(m_h_r, training)

  future <- make_future_dataframe(m_h_r, periods = ntest, freq = "week", include_history = TRUE)

  future$promo_01<- AL[,c("promo_01")]
  future$promo_02<- AL[,c("promo_02")]
  future$promo_03<- AL[,c("promo_03")]
  future$promo_04<- AL[,c("promo_04")]
  future$promo_05<- AL[,c("promo_05")]
  future$pred.joghurt<- AL[,c("pred.joghurt")]
  future$pred.gesund<- AL[,c("pred.gesund")]
  future$pred.joghurt.price<- AL[,c("pred.joghurt.price")]
  future$pred.milk.price<- AL[,c("pred.milk.price")]

  fcst_h_r <- predict(m_h_r, future)

  n<-nrow(training)

  training$pred_ProphetScoresRest<-fcst_h_r$yhat[1:n]
  testing$pred_ProphetScoresRest<-fcst_h_r$yhat[(n+1):(n+ntest)]

  sales<-df_final$sales
  scores<-c(training$y,testing$pred_ProphetScoresRest)

  mod<-lm(sales~scores)
  diff<-mod$fitted.values-sales

  training$pred_ProphetScoresRest<-mod$fitted.values[1:(len-ntest)]
  testing$pred_ProphetScoresRest<-mod$fitted.values[(len-ntest+1):len]

  model <- "ProphetScoresRest"
  a <- c(rmse(sales[1:(len-ntest)],mod$fitted.values[1:(len-ntest)]),
         rmse(sales[(len-ntest+1):len],mod$fitted.values[(len-ntest+1):len]))
  b <- c(mape(sales[1:(len-ntest)],mod$fitted.values[1:(len-ntest)]),
         mape(sales[(len-ntest+1):len],mod$fitted.values[(len-ntest+1):len]))
  c <- c(prod,model,a,b)
  d_factorR[i, ] = c

  ## overwrite y from prophet to true sales again

  training$y<-df_final$sales[1:(len-ntest)]
  testing$y<-df_final$sales[(len-ntest+1):len]

  ## store predicted data for plotting
  prediction[[i]]<-testing
  #print(c("All modells predicted for ", prod))
}
```

```
## [1] "Train Models for BIO FAIRT.JOG.MOKKA 180G"
## [1] "Train Models for MB JOGHURT NATUR 500G"
## [1] "Train Models for M-CLAS JOG. HEIDELBE 200G"
## [1] "Train Models for M-CLAS JOG. NATURE 200G"
## [1] "Train Models for M-CLAS JOG. HIMBEER 200G"
```

```
## [1] "Train Models for M-CLAS JOG. SCHOKOLA 200G"
## [1] "Train Models for M-CLAS JOGHURT MOKKA 200G"
## [1] "Train Models for M-CLAS JOG. APF/MANG 200G"
## [1] "Train Models for M-CLAS JOG. ERDBEER 200G"
## [1] "Train Models for M-CLAS JOG. VANILLE 200G"
## [1] "Train Models for M-CLAS JOG. HASELNUS 200G"
## [1] "Train Models for BIFIDUS JOGH. NATURE 500G"
## [1] "Train Models for BIO JOGHURT NATURE 180G"
## [1] "Train Models for EXC JOGHURT TRUFFES 150G"
## [1] "Train Models for YOGOS GRECQUE NATURE 180G"
## [1] "Train Models for BIO JOGHURT NATURE 500G"
## [1] "Train Models for BIFIDUS JOGH. NATURE 150G"
## [1] "Train Models for VALFLORA CREME FRAICHE NA"
## [1] "Train Models for AHA JOGHURT LAKTOSEF CLAS"
```

Best models for each Yogurt

```
#install.packages("lemon")
library(lemon)
```

```
## Warning: package 'lemon' was built under R version 4.0.5
```

```
knit_print.data.frame <- lemon_print
```

```
library(hablar)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:hablar':
##
##     na_if
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
## create mape data frame of all models
## set correct data type (char to number)
y.m.df<-rbind(d_lm, d, d_glm, d_prop1, d_prop2, d_xgboost, d_factorM, d_factorR)%>%
  retype()
```

```
## get best performing model for each yogurt
best.models <-y.m.df %>%
  group_by(prod) %>%
  filter(mape_test == min(mape_test))%>%
  select(prod, model)
best.models
```

prod	model
BIO JOGHURT NATURE 180G	RegressionTree
YOGOS GRECQUE NATURE 180G	Prophet1
BIO FAIRT.JOG.MOKKA 180G	Prophet2
EXC JOGHURT TRUFFES 150G	Prophet2
M-CLAS JOG. NATURE 200G	XGBOOST
M-CLAS JOG. VANILLE 200G	XGBOOST
BIFIDUS JOGH. NATURE 500G	XGBOOST
VALFLORA CREME FRAICHE NA	XGBOOST
MB JOGHURT NATUR 500G	ProphetScoresM
M-CLAS JOG. HEIDELBE 200G	ProphetScoresM
M-CLAS JOG. HIMBEER 200G	ProphetScoresM
M-CLAS JOG. SCHOKOLA 200G	ProphetScoresM
M-CLAS JOGHURT MOKKA 200G	ProphetScoresM
M-CLAS JOG. APF/MANG 200G	ProphetScoresM
M-CLAS JOG. ERDBEER 200G	ProphetScoresM
M-CLAS JOG. HASELNUS 200G	ProphetScoresM
BIO JOGHURT NATURE 500G	ProphetScoresM
BIFIDUS JOGH. NATURE 150G	ProphetScoresM
AHA JOGHURT LAKTOSEF CLAS	ProphetScoresRest

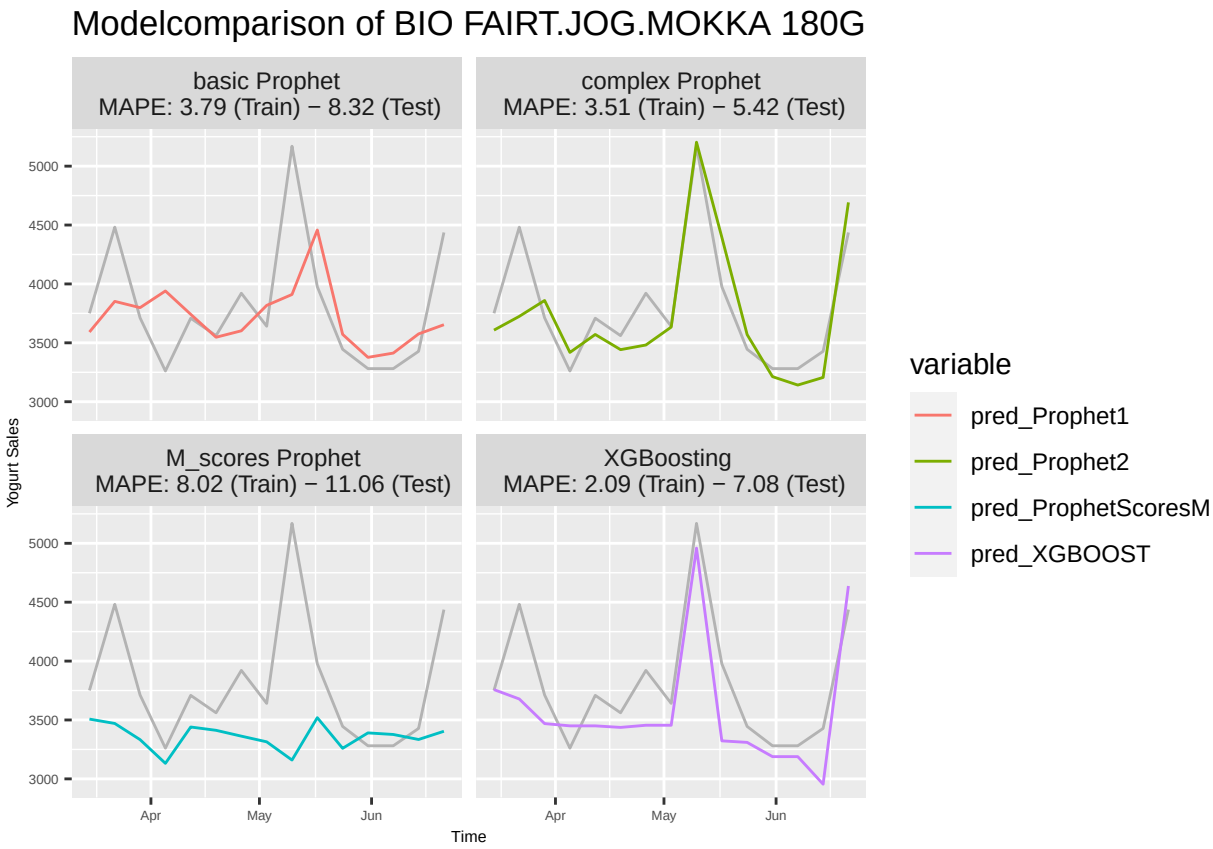
Plot predictions

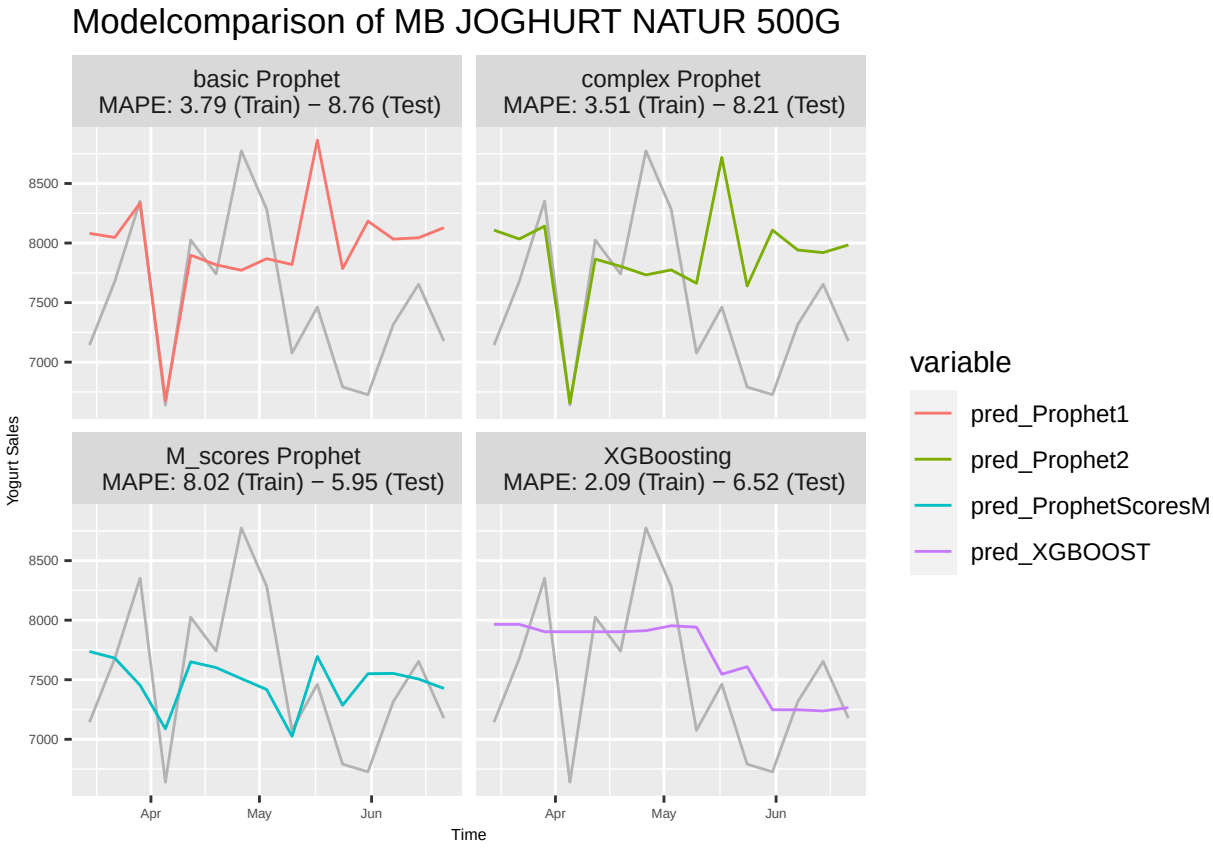
```
library(ggplot2)
```

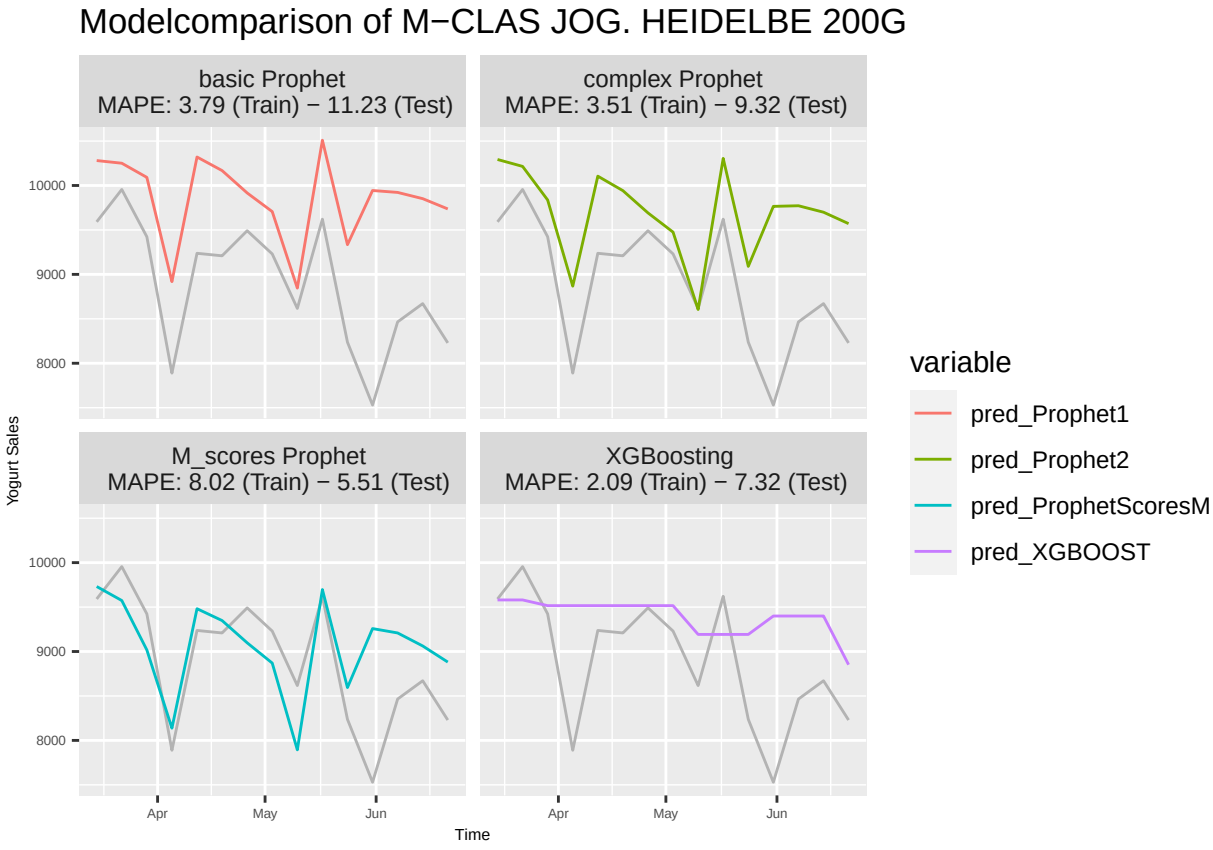
```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:lemon':
##
##     CoordCartesian, element_render
```

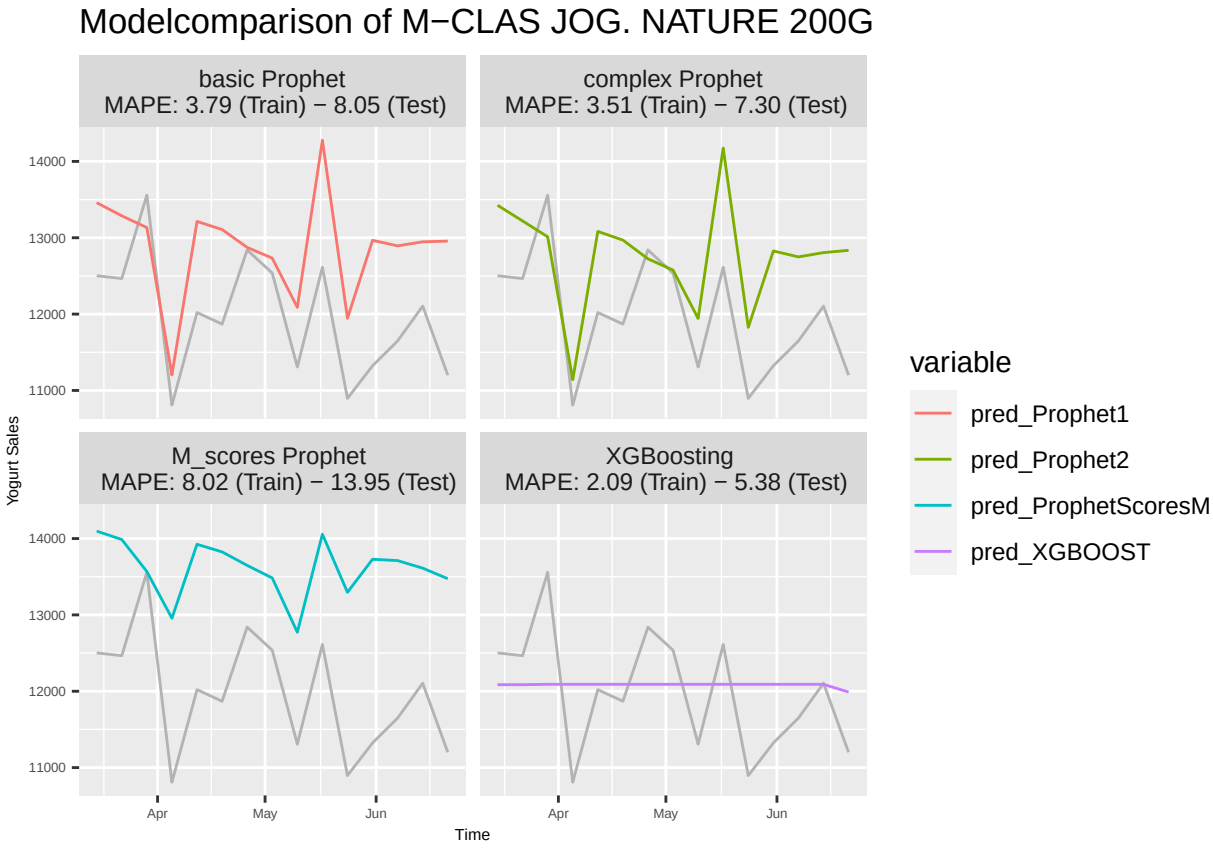
```
for(i in seq(1,length(prediction))){
  #i <-3

  yogurt <- dfs[[i]]$article_name[1]

  testing<-prediction[[i]]

  testqstat<- data.table::melt(prediction[[i]], id="ds", measure=c("pred_Prophet1",
                                                                   "pred_Prophet2",
                                                                   "pred_ProphetScoresM",
```

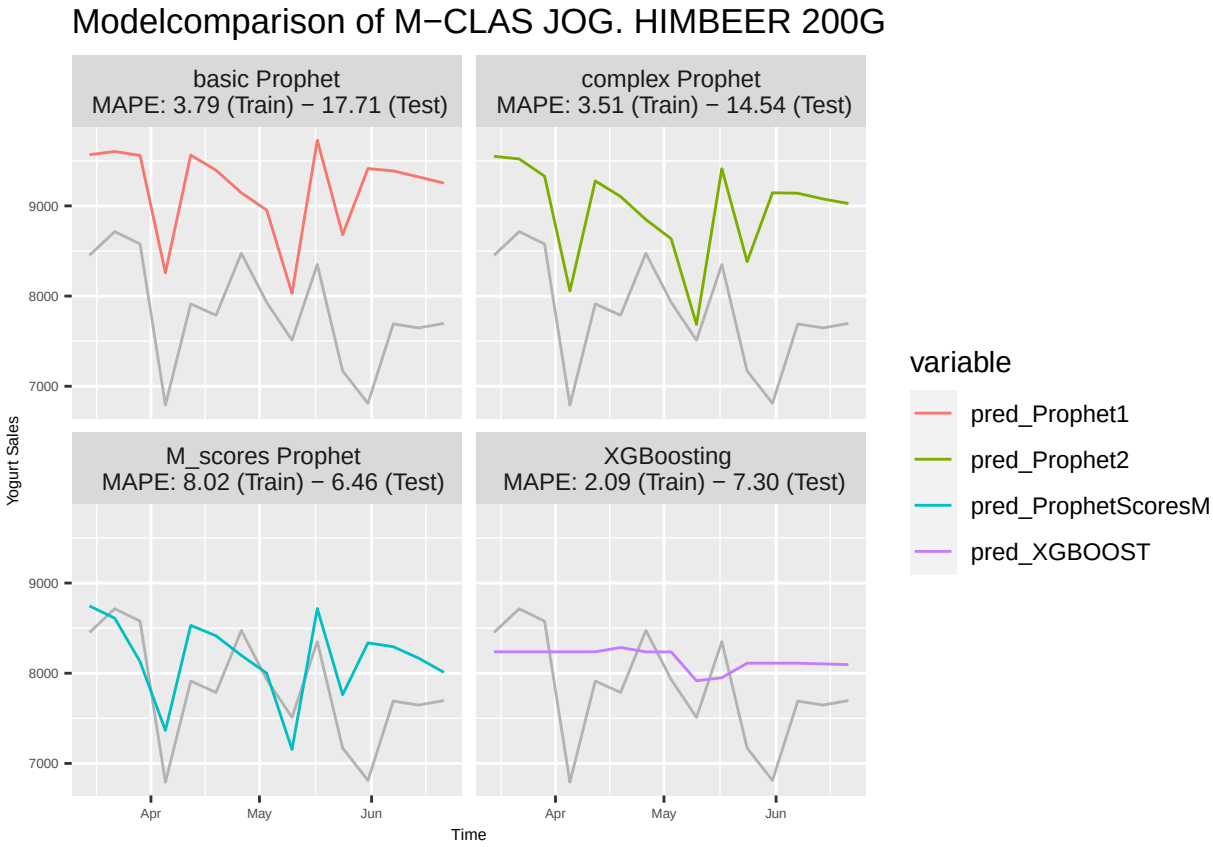
```
                                                    "pred_XGBOOST"))
  model_names<- c(
    'pred_Prophet1' = sprintf("basic Prophet\n MAPE: %.2f (Train) - %.2f (Test)",
                      mape(training$y, training$pred_Prophet1),
                      mape(testing$y, testing$pred_Prophet1)),

    'pred_Prophet2' = sprintf("complex Prophet\n MAPE: %.2f (Train) - %.2f (Test)",
                      mape(training$y, training$pred_Prophet2),
                      mape(testing$y, testing$pred_Prophet2)),

    'pred_ProphetScoresM' = sprintf("M_scores Prophet \n  MAPE: %.2f (Train) - %.2f (Test)",
                      mape(training$y, training$pred_ProphetScoresM),
                      mape(testing$y, testing$pred_ProphetScoresM)),

    'pred_XGBOOST' = sprintf("XGBoosting \n MAPE: %.2f (Train) - %.2f (Test)",
                      mape(training$y, training$pred_XGBOOST),
                      mape(testing$y, testing$pred_XGBOOST))
  )

  g1 <- ggplot(aes(y=value, x=ds, color=variable), data=testqstat) +

    xlim(min(testing$ds), max(testing$ds))+
    geom_line(data = testing, aes(y=y), color="grey70") +
    geom_line() +
    facet_wrap(~variable, ncol=2, labeller = as_labeller(model_names))+
    theme(axis.text = element_text(size=5), axis.title = element_text(size=6)) +
    labs(title=paste(c("Modelcomparison of ", yogurt), collapse="",sep=""),
         x="Time", y="Yogurt Sales")#+
    #theme(legend.position = "none")
  print(g1)
}
```

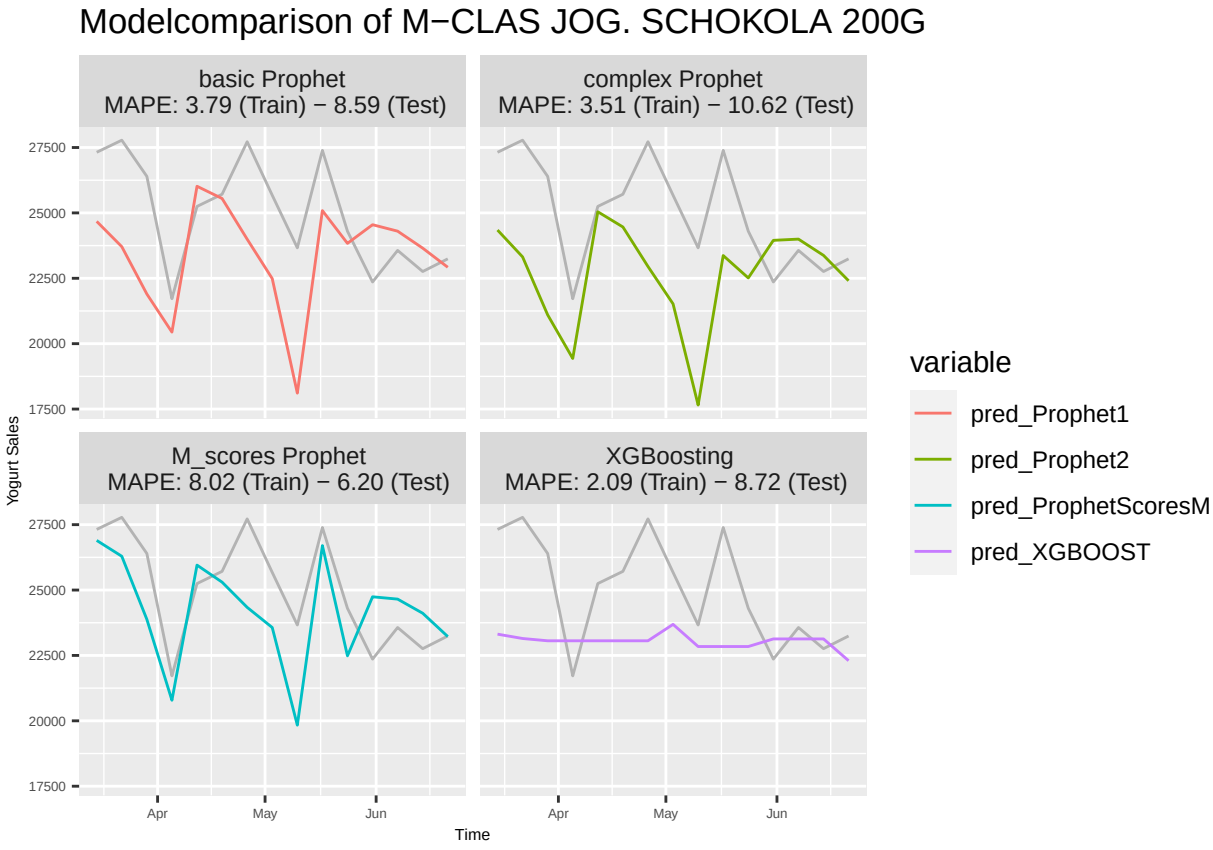

Modelcomparison of BIO FAIRT.JOG.MOKKA 180G
basic Prophet
MAPE: 3.79 (Train) − 8.32 (Test)
complex Prophet
MAPE: 3.51 (Train) − 5.42 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 11.06 (Test)
XGBoosting
MAPE: 2.09 (Train) − 7.08 (Test)
Yogurt Sales
5000
4500
4000
3500
3000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

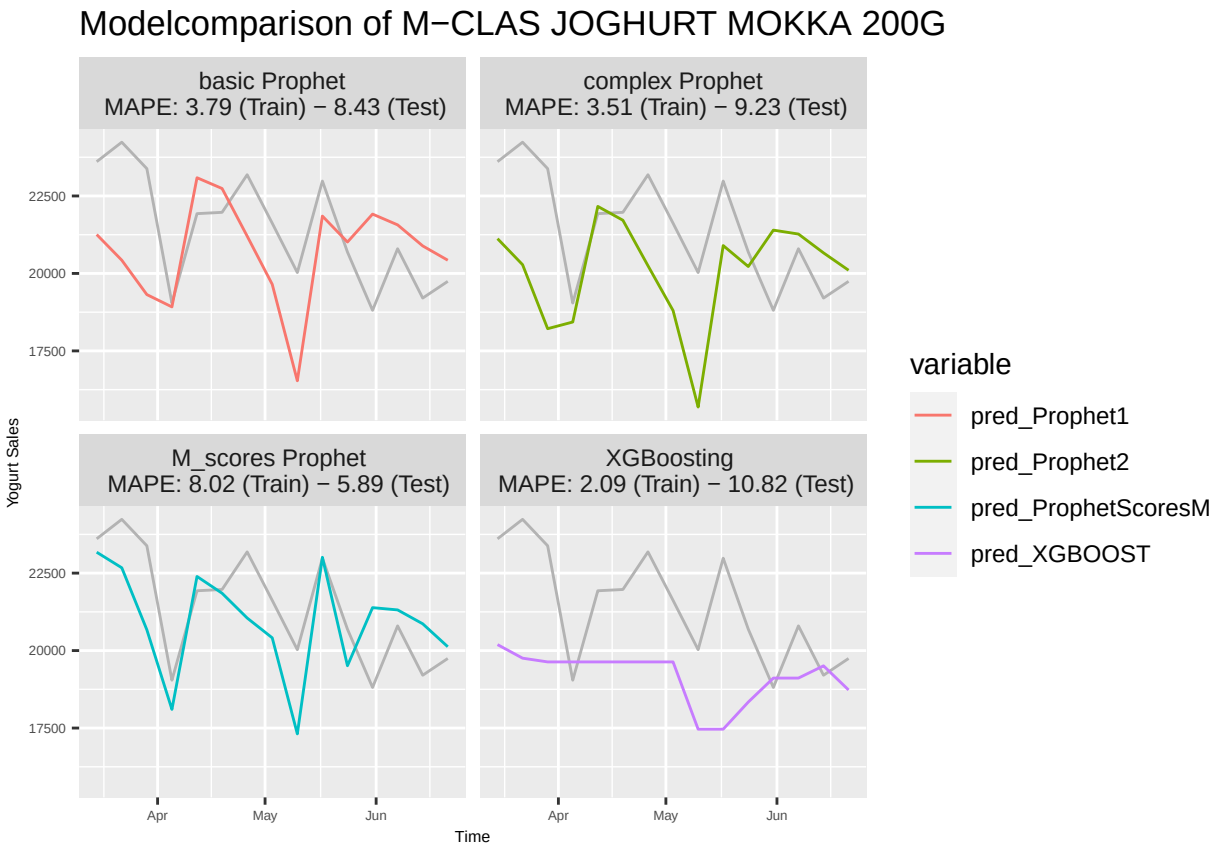

Modelcomparison of MB JOGHURT NATUR 500G
basic Prophet
MAPE: 3.79 (Train) − 8.76 (Test)
complex Prophet
MAPE: 3.51 (Train) − 8.21 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 5.95 (Test)
XGBoosting
MAPE: 2.09 (Train) − 6.52 (Test)
Yogurt Sales
8500
8000
7500
7000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

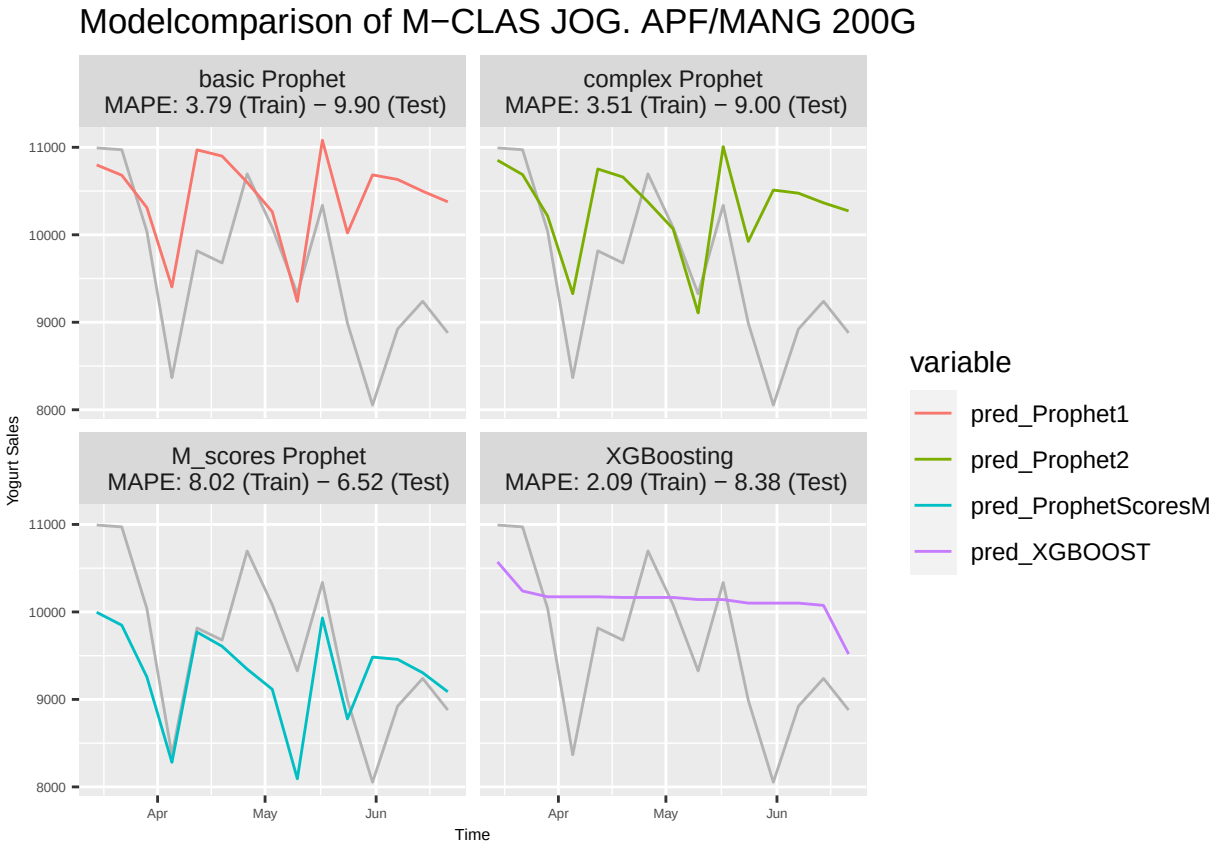

Modelcomparison of M−CLAS JOG. HEIDELBE 200G
basic Prophet
MAPE: 3.79 (Train) − 11.23 (Test)
complex Prophet
MAPE: 3.51 (Train) − 9.32 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 5.51 (Test)
XGBoosting
MAPE: 2.09 (Train) − 7.32 (Test)
Yogurt Sales
10000
9000
8000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

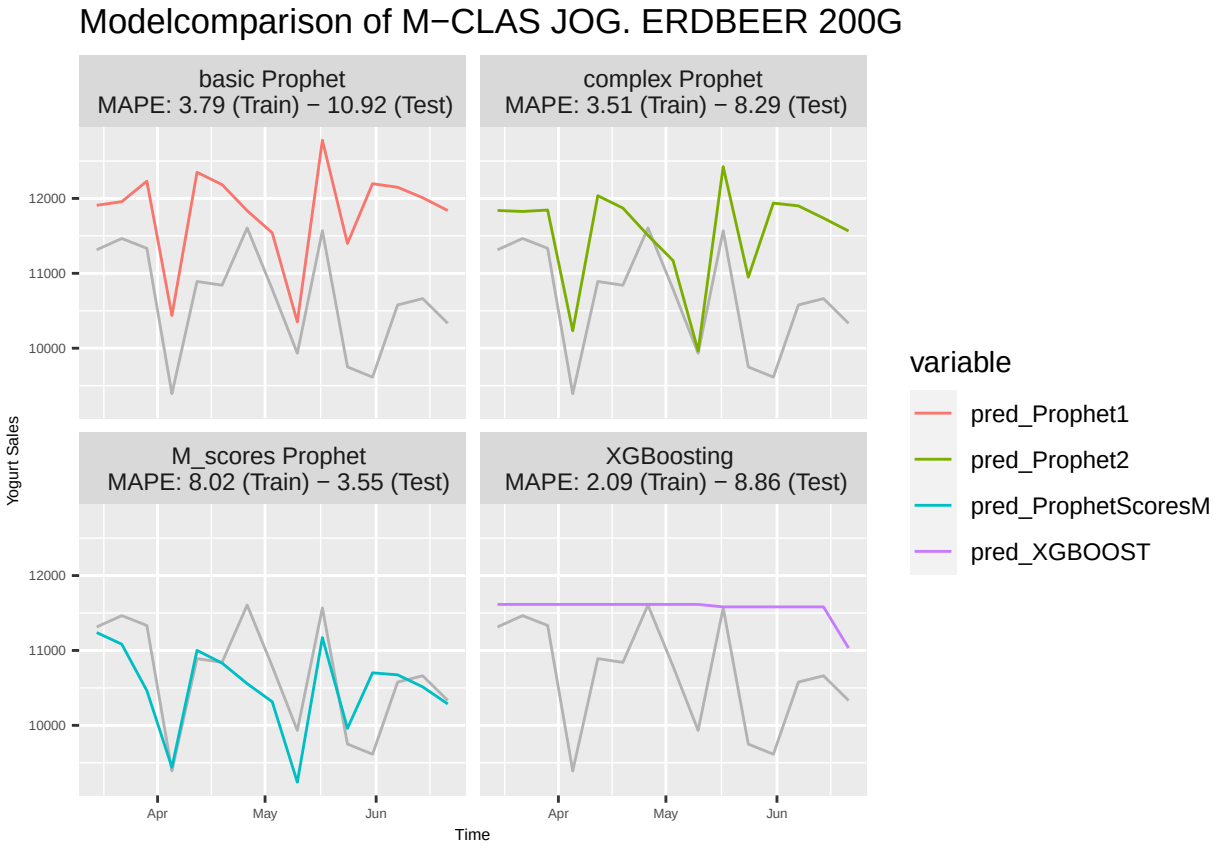

Modelcomparison of M−CLAS JOG. NATURE 200G
basic Prophet
MAPE: 3.79 (Train) − 8.05 (Test)
complex Prophet
MAPE: 3.51 (Train) − 7.30 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 13.95 (Test)
XGBoosting
MAPE: 2.09 (Train) − 5.38 (Test)
Yogurt Sales
14000
13000
12000
11000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

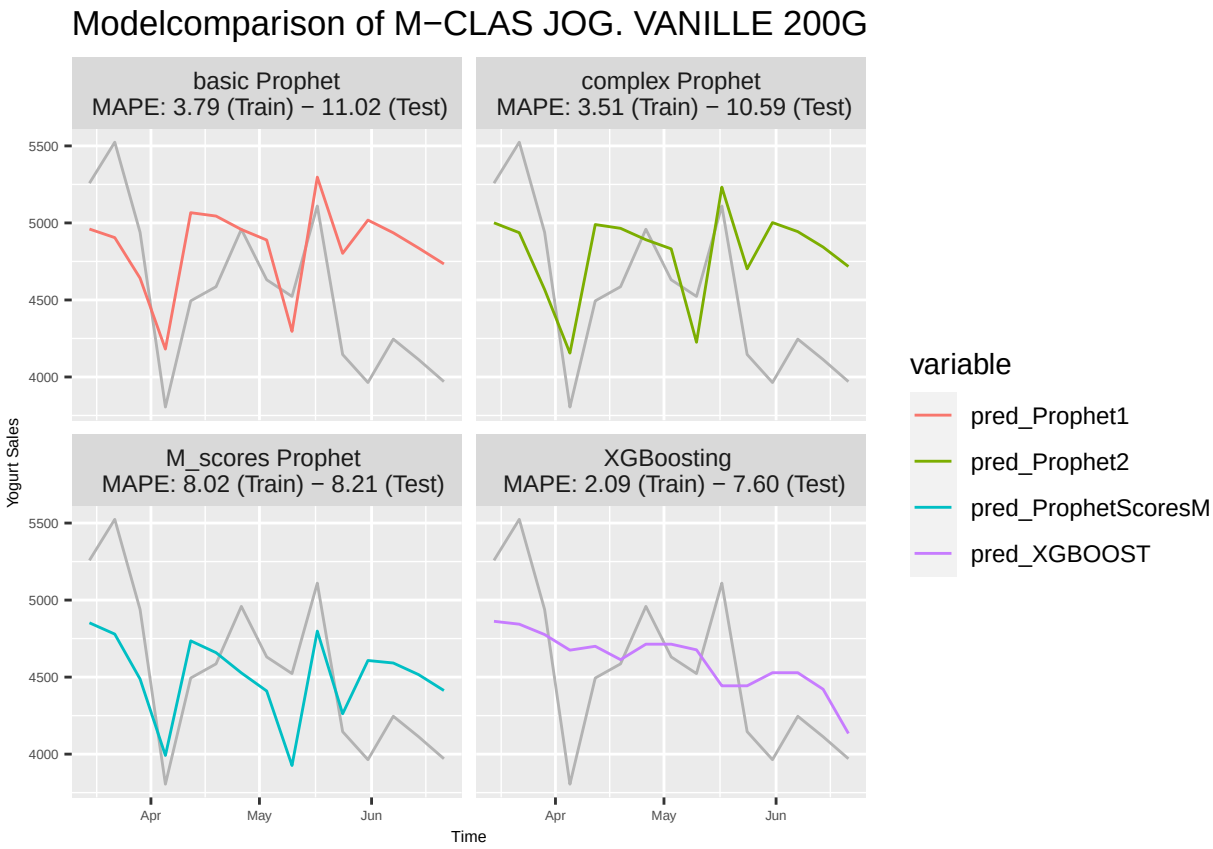

Modelcomparison of M−CLAS JOG. HIMBEER 200G
basic Prophet
MAPE: 3.79 (Train) − 17.71 (Test)
complex Prophet
MAPE: 3.51 (Train) − 14.54 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 6.46 (Test)
XGBoosting
MAPE: 2.09 (Train) − 7.30 (Test)
Yogurt Sales
9000
8000
7000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

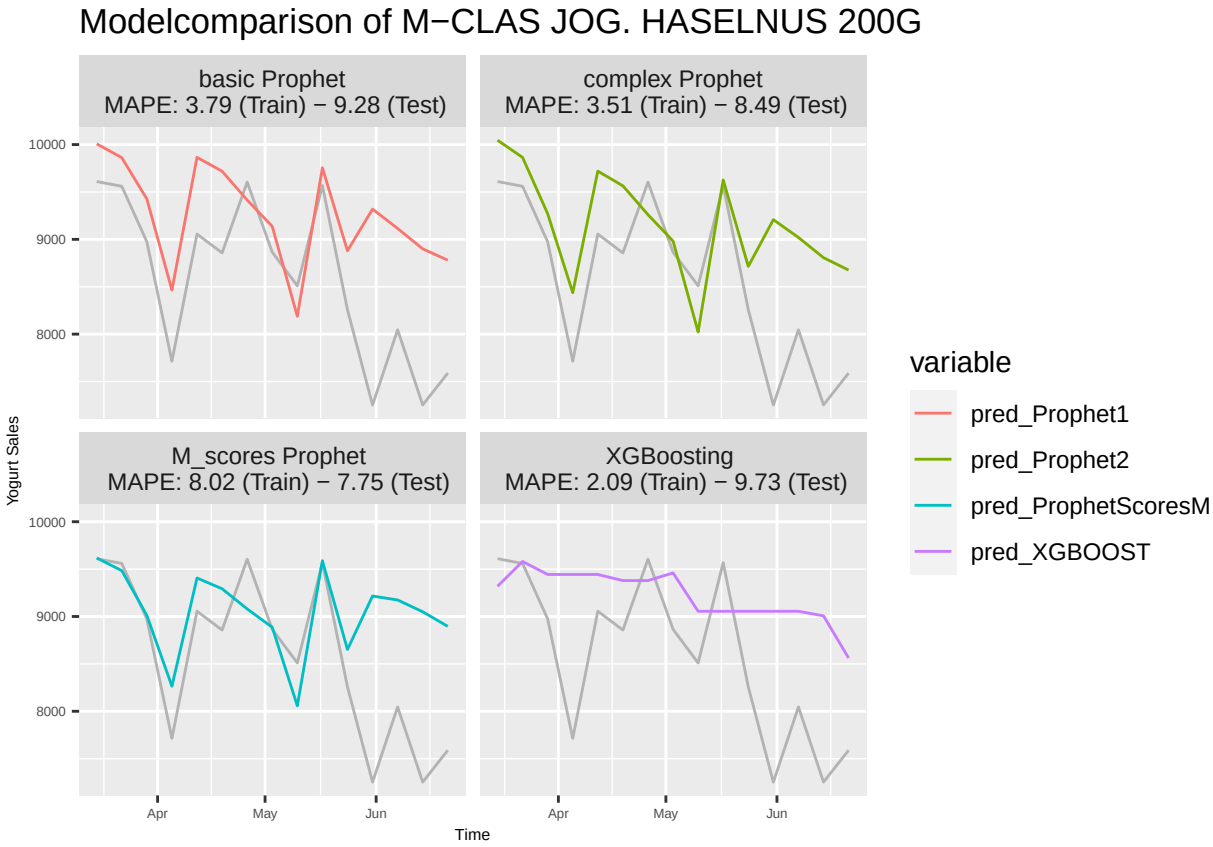

Modelcomparison of M−CLAS JOG. SCHOKOLA 200G
basic Prophet
MAPE: 3.79 (Train) − 8.59 (Test)
complex Prophet
MAPE: 3.51 (Train) − 10.62 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 6.20 (Test)
XGBoosting
MAPE: 2.09 (Train) − 8.72 (Test)
Yogurt Sales
27500
25000
22500
20000
17500
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

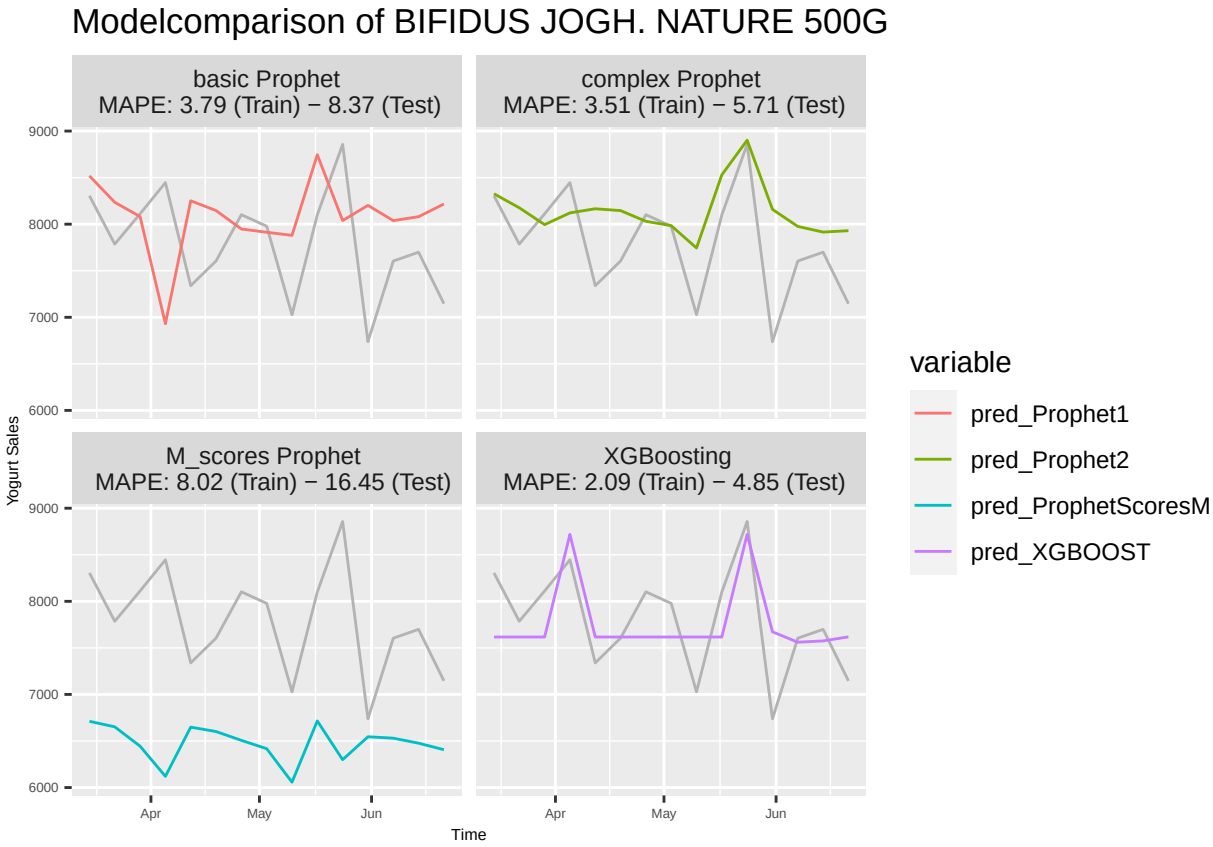

Modelcomparison of M−CLAS JOGHURT MOKKA 200G
basic Prophet
MAPE: 3.79 (Train) − 8.43 (Test)
complex Prophet
MAPE: 3.51 (Train) − 9.23 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 5.89 (Test)
XGBoosting
MAPE: 2.09 (Train) − 10.82 (Test)
Yogurt Sales
22500
20000
17500
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

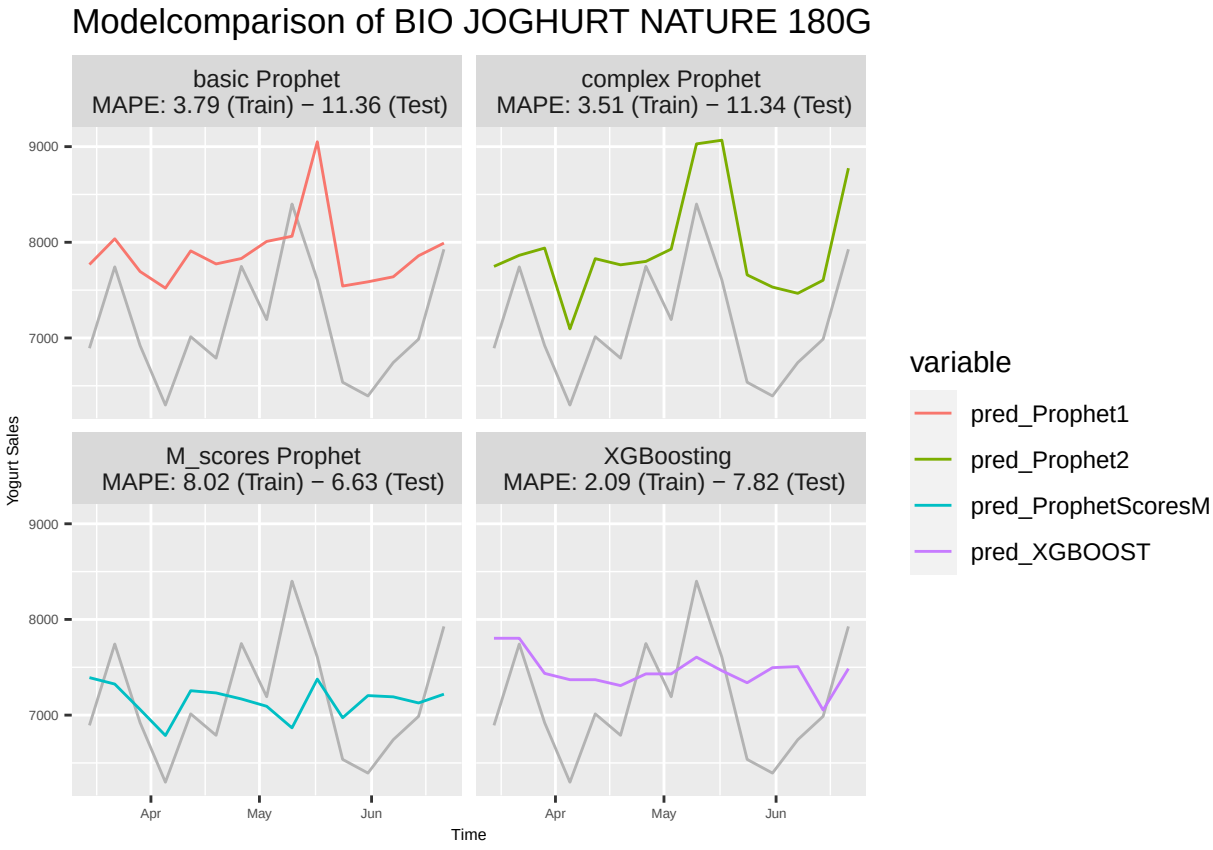

Modelcomparison of M−CLAS JOG. APF/MANG 200G
basic Prophet
MAPE: 3.79 (Train) − 9.90 (Test)
complex Prophet
MAPE: 3.51 (Train) − 9.00 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 6.52 (Test)
XGBoosting
MAPE: 2.09 (Train) − 8.38 (Test)
Yogurt Sales
11000
10000
9000
8000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

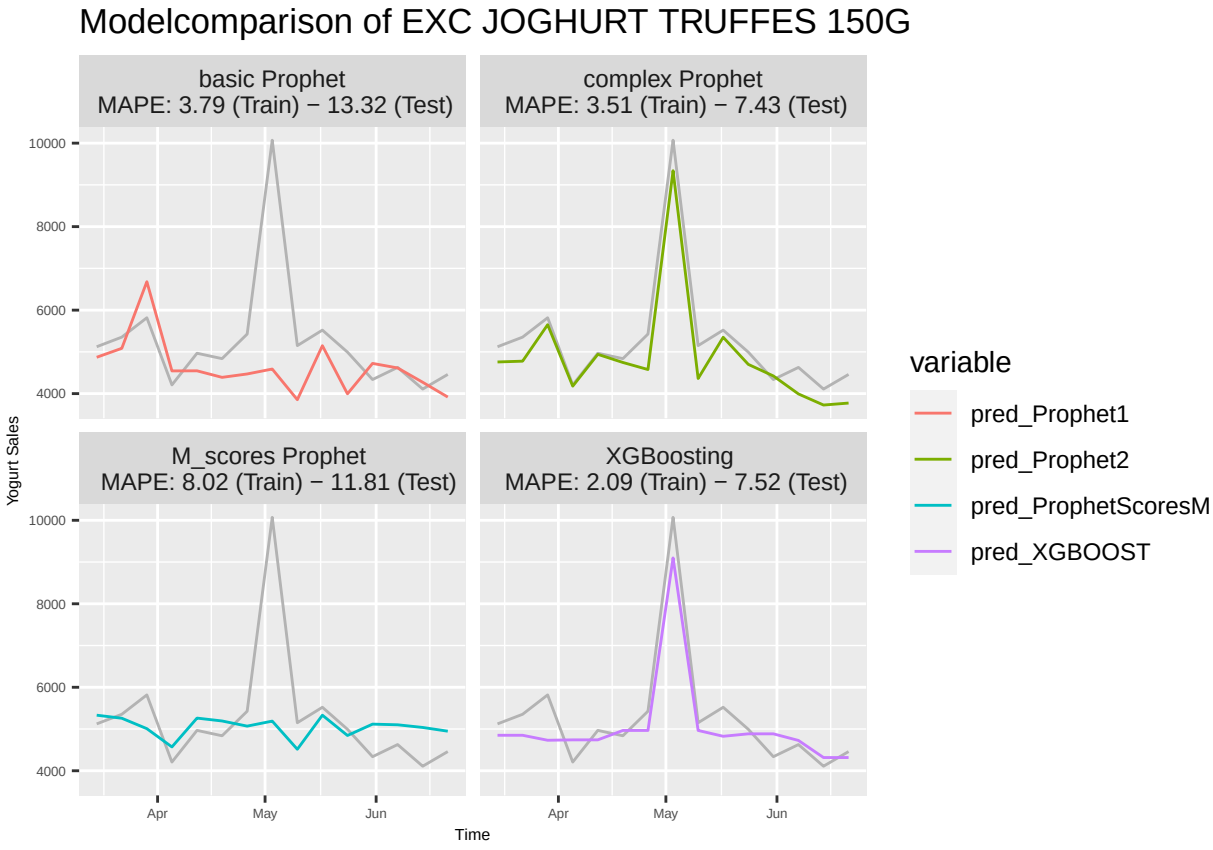

Modelcomparison of M−CLAS JOG. ERDBEER 200G
basic Prophet
MAPE: 3.79 (Train) − 10.92 (Test)
complex Prophet
MAPE: 3.51 (Train) − 8.29 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 3.55 (Test)
XGBoosting
MAPE: 2.09 (Train) − 8.86 (Test)
Yogurt Sales
12000
11000
10000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

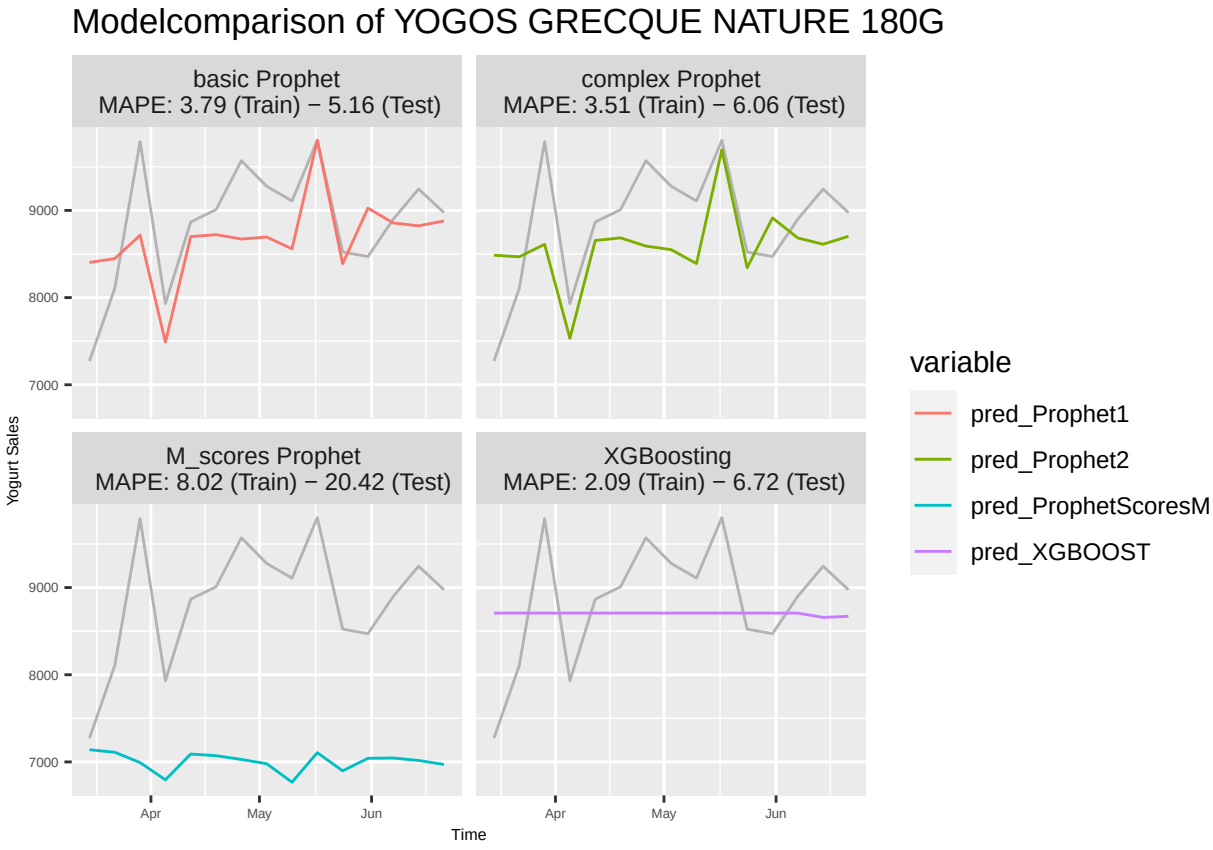

Modelcomparison of M−CLAS JOG. VANILLE 200G
basic Prophet
MAPE: 3.79 (Train) − 11.02 (Test)
complex Prophet
MAPE: 3.51 (Train) − 10.59 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 8.21 (Test)
XGBoosting
MAPE: 2.09 (Train) − 7.60 (Test)
Yogurt Sales
5500
5000
4500
4000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

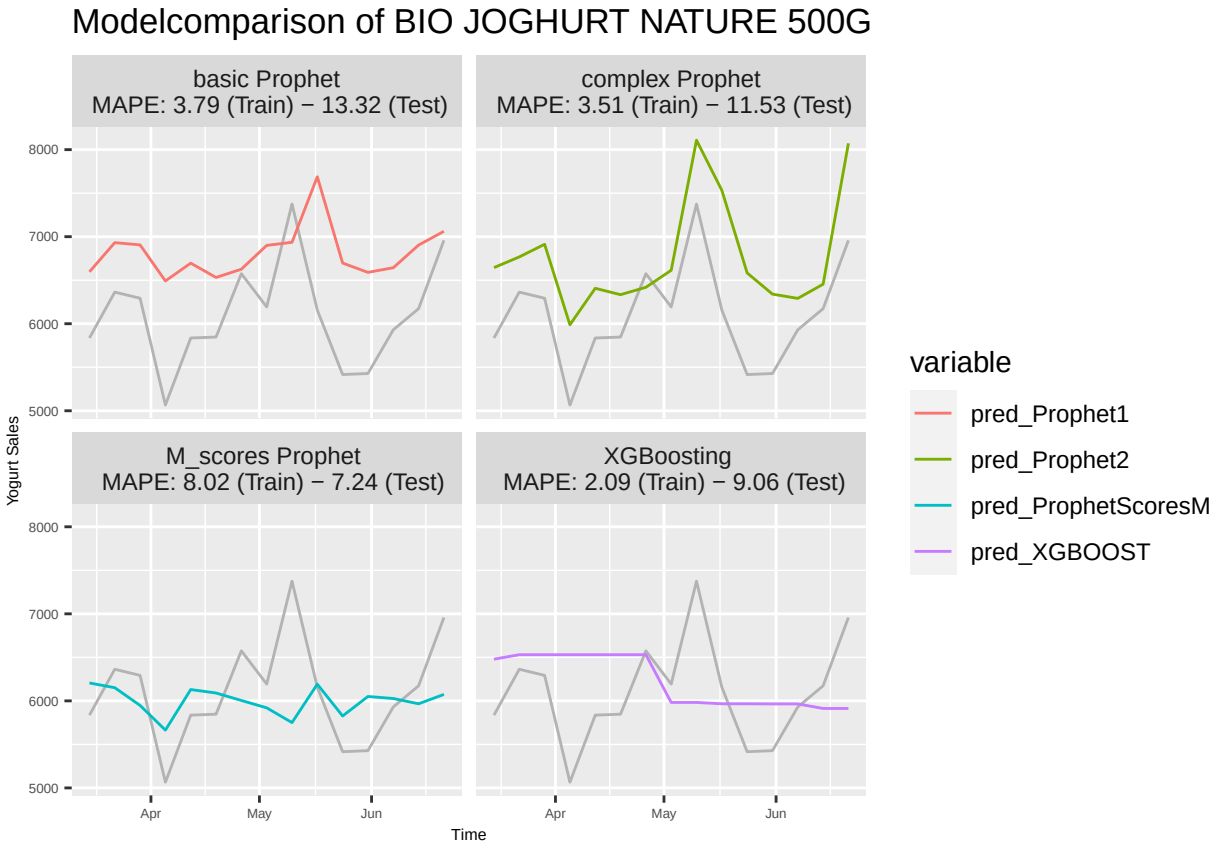

Modelcomparison of M−CLAS JOG. HASELNUS 200G
basic Prophet
MAPE: 3.79 (Train) − 9.28 (Test)
complex Prophet
MAPE: 3.51 (Train) − 8.49 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 7.75 (Test)
XGBoosting
MAPE: 2.09 (Train) − 9.73 (Test)
Yogurt Sales
10000
9000
8000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

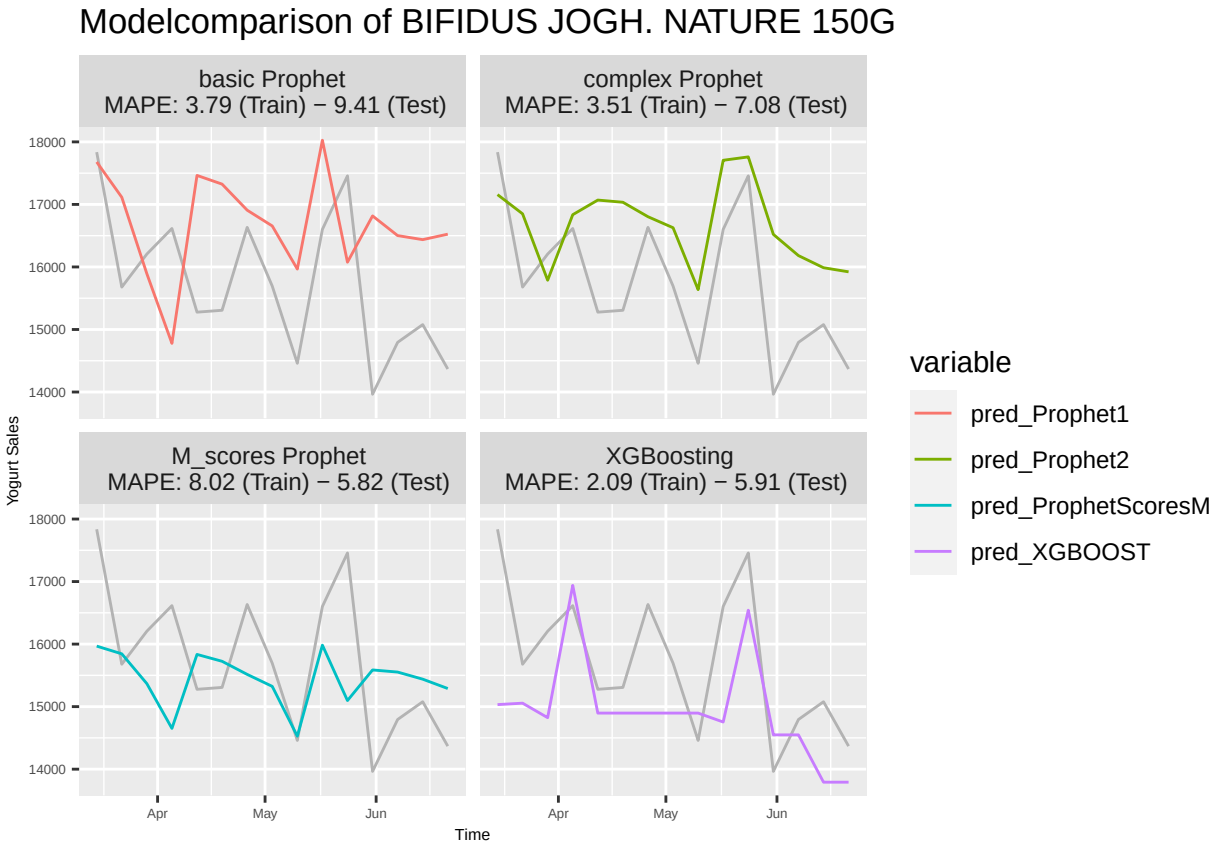

Modelcomparison of BIFIDUS JOGH. NATURE 500G
basic Prophet
MAPE: 3.79 (Train) − 8.37 (Test)
complex Prophet
MAPE: 3.51 (Train) − 5.71 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 16.45 (Test)
XGBoosting
MAPE: 2.09 (Train) − 4.85 (Test)
Yogurt Sales
9000
8000
7000
6000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

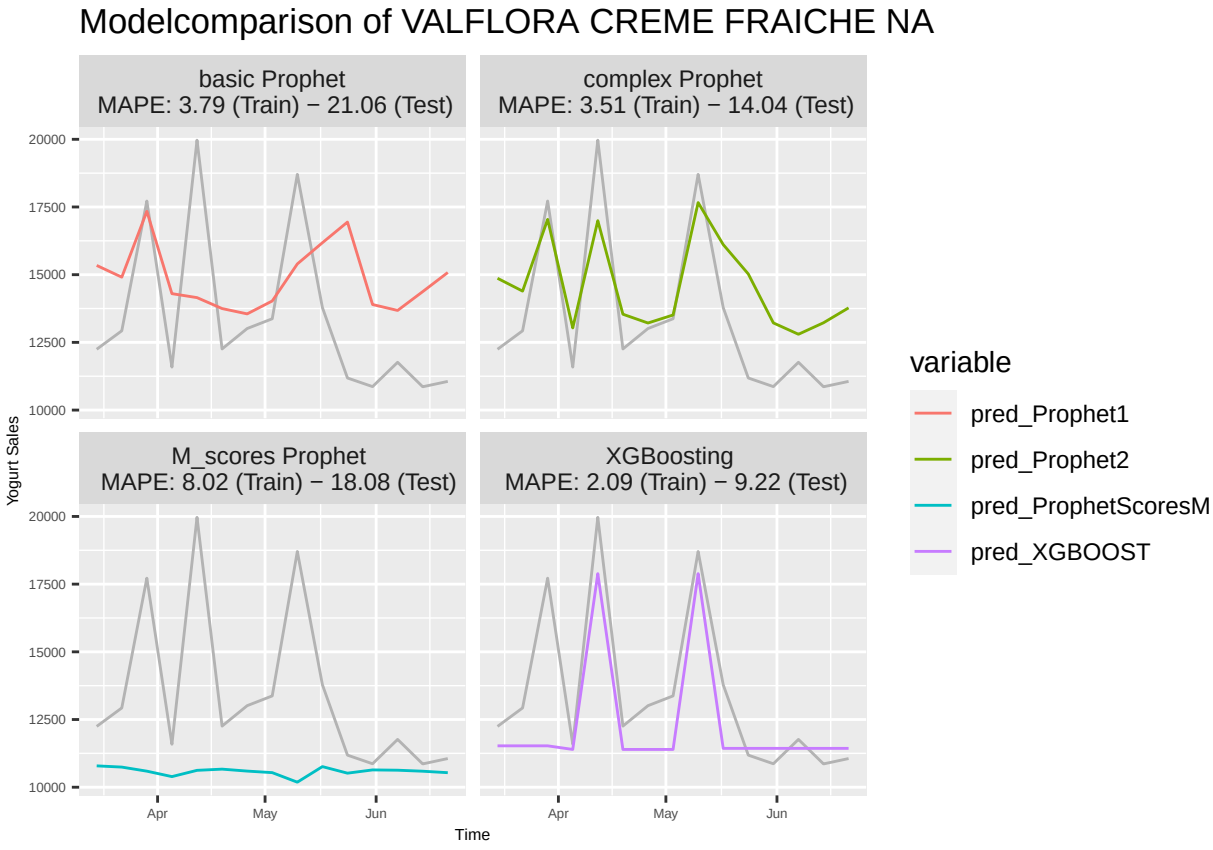

Modelcomparison of BIO JOGHURT NATURE 180G
basic Prophet
MAPE: 3.79 (Train) − 11.36 (Test)
complex Prophet
MAPE: 3.51 (Train) − 11.34 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 6.63 (Test)
XGBoosting
MAPE: 2.09 (Train) − 7.82 (Test)
Yogurt Sales
9000
8000
7000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST

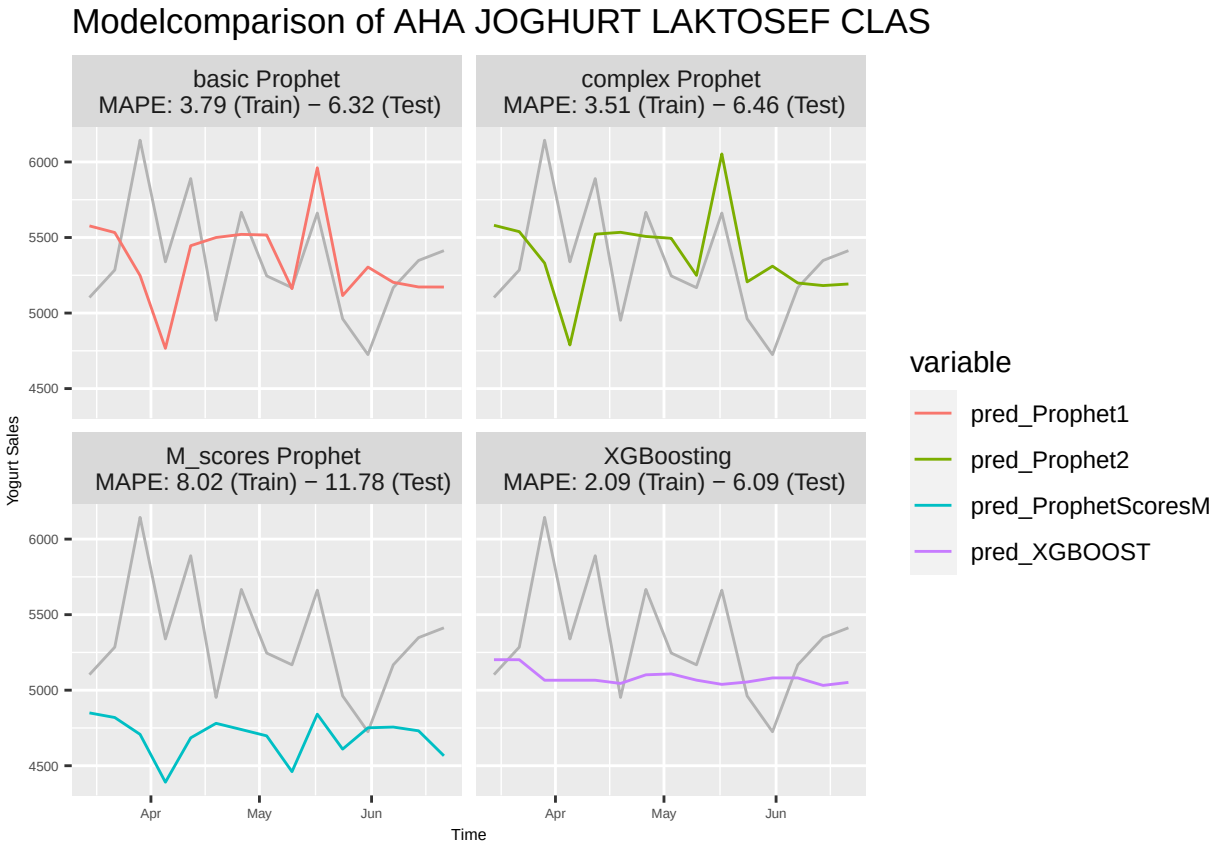

Modelcomparison of EXC JOGHURT TRUFFES 150G
basic Prophet
MAPE: 3.79 (Train) − 13.32 (Test)
complex Prophet
MAPE: 3.51 (Train) − 7.43 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 11.81 (Test)
XGBoosting
MAPE: 2.09 (Train) − 7.52 (Test)
Yogurt Sales
10000
8000
6000
4000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST


Modelcomparison of YOGOS GRECQUE NATURE 180G
basic Prophet
MAPE: 3.79 (Train) − 5.16 (Test)
complex Prophet
MAPE: 3.51 (Train) − 6.06 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 20.42 (Test)
XGBoosting
MAPE: 2.09 (Train) − 6.72 (Test)
Yogurt Sales
9000
8000
7000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST


Modelcomparison of BIO JOGHURT NATURE 500G
basic Prophet
MAPE: 3.79 (Train) − 13.32 (Test)
complex Prophet
MAPE: 3.51 (Train) − 11.53 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 7.24 (Test)
XGBoosting
MAPE: 2.09 (Train) − 9.06 (Test)
Yogurt Sales
8000
7000
6000
5000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST


Modelcomparison of BIFIDUS JOGH. NATURE 150G
basic Prophet
MAPE: 3.79 (Train) − 9.41 (Test)
complex Prophet
MAPE: 3.51 (Train) − 7.08 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 5.82 (Test)
XGBoosting
MAPE: 2.09 (Train) − 5.91 (Test)
Yogurt Sales
18000
17000
16000
15000
14000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST


Modelcomparison of VALFLORA CREME FRAICHE NA
basic Prophet
MAPE: 3.79 (Train) − 21.06 (Test)
complex Prophet
MAPE: 3.51 (Train) − 14.04 (Test)
M_scores Prophet
MAPE: 8.02 (Train) − 18.08 (Test)
XGBoosting
MAPE: 2.09 (Train) − 9.22 (Test)
Yogurt Sales
20000
17500
15000
12500
10000
Apr
May
Jun
Time
variable
pred_Prophet1
pred_Prophet2
pred_ProphetScoresM
pred_XGBOOST



Overview of best performing model

```
## create data framewith all MAPE in testset
mape_test<-cbind(as.numeric(d_lm$mape_test),as.numeric(d_glm$mape_test),as.numeric(d$mape_test),as.numer

colnames(mape_test)<-c("lm","glm","tree","prop1","prop2","xgboost","factorM","factorR")
rownames(mape_test)<-d_glm[,1]

min<-apply(mape_test, 1, FUN = min)
a<-mape_test==min
colSums(a)
```

```
##      lm     glm    tree   prop1   prop2 xgboost factorM factorR
##       0       0       1       1       2       4      10       1
```