pdf\_test

Pedro Pablo Skorin

12/3/2021

library(knitr)

## Warning: package 'knitr' was built under R version 4.0.5

opts\_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)

##########################################  
## Point Estimate Forecasting Functions ##  
##########################################  
  
# L2-Boosting AIC function  
boosting\_reg\_aic = function(Y\_or, Y, X, v, h, ratio\_start = 0.8, Mstop = 3500) {  
   
 n\_tot <- length(Y)  
 n\_out <- ceiling(n\_tot - ratio\_start\*n\_tot)  
 ind\_out <- seq(to = n\_tot, by = 1, length = n\_out)  
 Y\_predicted = c(Y\_or[ind\_out[1]])  
 varimp\_df = data.frame(rep(0, (ncol(X)+1)))  
 selected\_var = c()  
   
 for(i in 1:n\_out){  
 ind\_in <- seq(from = 1, to = ind\_out[i] - h, by = 1)  
 y\_extra = c()  
 x\_reg <- X[head(ind\_in,-1),] # x independent t = 1, ..., T.in-h  
 x0\_reg <- matrix(X[tail(ind\_in,1),], nrow = 1)  
   
 for(j in 1:h) {  
  
 # expanding window  
 y\_dep <- append(Y[tail(ind\_in,-j)], y\_extra)  
 y\_reg <- as.matrix(y\_dep)  
   
 # finding m\*  
 model\_1 = glmboost(y\_reg ~ ., data = x\_reg,  
 family = Gaussian(),  
 control = boost\_control(mstop = Mstop, nu = v),  
 center = T)  
   
 AIC = AIC(model\_1, method = "corrected" , df = "actset")  
 x0\_reg\_df = data.frame(t(data.frame(unlist(x0\_reg))))  
 colnames(x0\_reg\_df) = colnames(x\_reg)  
 y\_predicted = unname(predict(model\_1[mstop(AIC)], newdata = x0\_reg\_df,  
 type = "response")[1,1])  
 cat("Selected M is: ", mstop(AIC), "\n")  
   
 # visualizing selected predictors varimp  
 varimp\_df\_partial = data.frame(varimp(model\_1))  
 sum\_reduction = sum(varimp\_df\_partial[,1])  
 varimp\_partial = varimp\_df\_partial[,1]/sum\_reduction  
 varimp\_df = cbind(varimp\_df, varimp\_partial)  
   
 # visualizing selected predictors frequency  
 selected\_var = append(selected\_var, list(model\_1$xselect()))  
   
 # output  
 y\_extra = append(y\_extra, y\_predicted)  
   
 }  
   
 Y\_predicted = append(Y\_predicted, Y\_or[(ind\_out[1]+i-(h))] + sum(y\_extra))  
 print(i/n\_out)  
   
 }  
 results <- list(forecast = Y\_predicted,  
 varimp = varimp\_df[,-1],  
 selected = selected\_var  
 )  
 return(results)  
}  
  
# L2-Boosting K-fold function  
boosting\_reg\_kfold = function(Y\_or, Y, X, v, h, ratio\_start = 0.8, Mstop = 3500) {  
   
 n\_tot <- length(Y)  
 n\_out <- ceiling(n\_tot - ratio\_start\*n\_tot)  
 ind\_out <- seq(to = n\_tot, by = 1, length = n\_out)  
 Y\_predicted = c(Y\_or[ind\_out[1]])  
 varimp\_df = data.frame(rep(0, (ncol(X)+1)))  
 selected\_var = c()  
   
 for(i in 1:n\_out){  
 ind\_in <- seq(from = 1, to = ind\_out[i] - h, by = 1)  
 y\_extra = c()  
 x\_reg <- X[head(ind\_in,-1),] # x independent t = 1, ..., T.in-h  
 x0\_reg <- matrix(X[tail(ind\_in,1),], nrow = 1)  
   
 for(j in 1:h) {  
   
 # expanding window  
 y\_dep <- append(Y[tail(ind\_in,-j)], y\_extra)  
 y\_reg <- as.matrix(y\_dep)  
   
 # finding m\*  
 model\_1 = glmboost(y\_reg ~ ., data = x\_reg,  
 family = Gaussian(),  
 control = boost\_control(mstop = Mstop, nu = v),  
 center = T)  
 cv10f <- cv(model.weights(model\_1), type = "kfold")  
 cvm <- cvrisk(model\_1, folds = cv10f, papply = lapply)  
   
 # AIC = AIC(model\_1, method = "corrected" , df = "actset")  
 x0\_reg\_df = data.frame(t(data.frame(unlist(x0\_reg))))  
 colnames(x0\_reg\_df) = colnames(x\_reg)  
 y\_predicted = unname(predict(model\_1[mstop(cvm)], newdata = x0\_reg\_df,  
 type = "response")[1,1])  
 cat("Selected M is: ", mstop(cvm), "\n")  
   
 # visualizing selected predictors varimp  
 varimp\_df\_partial = data.frame(varimp(model\_1))  
 sum\_reduction = sum(varimp\_df\_partial[,1])  
 varimp\_partial = varimp\_df\_partial[,1]/sum\_reduction  
 varimp\_df = cbind(varimp\_df, varimp\_partial)  
   
 # visualizing selected predictors frequency  
 selected\_var = append(selected\_var, list(model\_1$xselect()))  
   
 # output  
 y\_extra = append(y\_extra, y\_predicted)  
   
 }  
 Y\_predicted = append(Y\_predicted, Y\_or[(ind\_out[1]+i-(h))] + sum(y\_extra))  
 print(i/n\_out)  
   
 }  
 results <- list(forecast = Y\_predicted,  
 varimp = varimp\_df[,-1],  
 selected = selected\_var)  
 return(results)  
}  
  
# SARIMA Function  
SARIMA\_bench = function(Y\_or, Y, h, ratio\_start = 0.8) {  
   
 n\_tot <- length(Y)  
 n\_out <- ceiling(n\_tot - ratio\_start\*n\_tot)  
 ind\_out <- seq(to = n\_tot, by = 1, length = n\_out)  
 Y\_arima = c(Y\_or[ind\_out[1]])  
   
 for(i in 1:n\_out){  
 ind\_in <- seq(from = 1, to = ind\_out[i] - h, by = 1)  
 bench = arima(exp(Y\_or[ 1:(ind\_out[i] - h + 1) ]), c(1,1,0)  
 , seasonal = list(order = c(1,1,0), period = 12)  
 )  
   
 forecast\_bench = forecast(bench, h)  
 y\_predicted\_bench = forecast\_bench$mean[h]  
 y\_predicted\_arima = log(y\_predicted\_bench)  
 Y\_arima = append(Y\_arima, (y\_predicted\_arima))  
 print(i/n\_out)  
   
 }  
 results <- list(benchmark = Y\_arima)  
 return(results)  
}  
  
#############################################  
## Interval Estimate Forecasting Functions ##  
#############################################  
  
# L2-Boosting quantile function  
boosting\_reg\_quantile = function(Y\_or, Y, X, v, h, ratio\_start = 0.8, Mstop = 3500, tau\_in = 0.5, offset\_in = 0.5, m\_mult = 4) {  
   
 n\_tot <- length(Y)  
 n\_out <- ceiling(n\_tot - ratio\_start\*n\_tot)  
 ind\_out <- seq(to = n\_tot, by = 1, length = n\_out)  
 Y\_predicted = c(Y\_or[ind\_out[1]])  
 varimp\_df = data.frame(rep(0, (ncol(X)+1)))  
 selected\_var = c()  
   
 for(i in 1:n\_out){  
 ind\_in <- seq(from = 1, to = ind\_out[i] - h, by = 1)  
 y\_extra = c()  
 x\_reg <- X[head(ind\_in,-1),] # x independent t = 1, ..., T.in-h  
 x0\_reg <- matrix(X[tail(ind\_in,1),], nrow = 1)  
   
 for(j in 1:h) {  
   
 # expanding window  
 y\_dep <- append(Y[tail(ind\_in,-j)], y\_extra)  
 y\_reg <- as.matrix(y\_dep)  
   
 # finding m\*  
 model\_1 = glmboost(y\_reg ~ ., data = x\_reg,  
 family = QuantReg(tau=tau\_in, qoffset = offset\_in),  
 control = boost\_control(mstop = m\_mult\*Mstop, nu = v),  
 center = T)  
   
 model\_2 = glmboost(y\_reg ~ ., data = x\_reg,  
 family = Gaussian(),  
 control = boost\_control(mstop = Mstop, nu = v),  
 center = T)  
   
 AIC = AIC(model\_2, method = "corrected" , df = "actset")  
 x0\_reg\_df = data.frame(t(data.frame(unlist(x0\_reg))))  
 colnames(x0\_reg\_df) = colnames(x\_reg)  
 y\_predicted = unname(predict(model\_1[m\_mult\*mstop(AIC)], newdata = x0\_reg\_df  
 #,type = "link"  
 )[1, 1])  
 cat("Selected M is: ", m\_mult\*mstop(AIC), "\n")  
  
 # visualizing selected predictors varimp  
 varimp\_df\_partial = data.frame(varimp(model\_1))  
 sum\_reduction = sum(varimp\_df\_partial[,1])  
 varimp\_partial = varimp\_df\_partial[,1]/sum\_reduction  
 varimp\_df = cbind(varimp\_df, varimp\_partial)  
   
 # visualizing selected predictors frequency  
 selected\_var = append(selected\_var, list(model\_1$xselect()))  
   
 # output  
 y\_extra = append(y\_extra, y\_predicted)  
   
 }  
 Y\_predicted = append(Y\_predicted, Y\_or[(ind\_out[1]+i-(h))] + sum(y\_extra))  
 print(i/n\_out)  
   
 }  
 results <- list(forecast = Y\_predicted,  
 varimp = varimp\_df[,-1],  
 selected = selected\_var  
 )  
 return(results)  
}  
  
# Sarima quantile function  
quantile\_sarima = function(Y\_or, Y, h, ratio\_start = 0.79, tau\_in) {  
 Y\_lag = tail(Y,-12)  
 Y\_1 = head(tail(Y,-11),-1)  
 Y\_12 = head(Y,-12)   
 X = as.data.frame(cbind(Y\_1,Y\_12))  
   
 n\_tot <- length(Y\_lag)  
 n\_out <- ceiling(n\_tot - ratio\_start\*n\_tot)  
 Y\_or\_lag = tail(Y\_or,-12)  
 ind\_out <- seq(to = n\_tot, by = 1, length = n\_out)  
 Y\_predicted = c(Y\_or[ind\_out[1]])  
   
 for(i in 1:n\_out){  
 ind\_in <- seq(from = 1, to = ind\_out[i] - h, by = 1)  
 y\_extra = c()  
 x\_reg <- X[head(ind\_in,-1),] # x independent t = 1, ..., T.in-h  
 x0\_reg <- matrix(X[tail(ind\_in,1),], nrow = 1)  
   
 for(j in 1:h) {  
   
 # expanding window  
 y\_dep <- append(Y[tail(ind\_in,-j)], y\_extra)  
 y\_reg <- as.matrix(y\_dep)  
 q = rq(Y\_lag ~ Y\_1 + Y\_12, tau = tau\_in)  
 x0\_reg\_df = data.frame(t(data.frame(unlist(x0\_reg))))  
 colnames(x0\_reg\_df) = colnames(x\_reg)  
 y\_predicted = unname(predict(q, newdata = x0\_reg\_df,  
 type = "percentile"))  
 # output  
 y\_extra = append(y\_extra, y\_predicted)  
 }  
 Y\_predicted = append(Y\_predicted, Y\_or[(ind\_out[1]+i-(h))] + sum(y\_extra))  
 print(i/n\_out)  
   
 }  
 results <- list(forecast = Y\_predicted)  
   
 return(results)  
}  
  
#####################  
## Other Functions ##  
#####################  
  
# Performance measures  
evaluation = function(Z, W, index, texto) {  
   
 cat("Evaluation of", texto)  
   
 MAPE = mean((abs(exp(W[index+1])[]-exp(Z[-1])[])/exp(W[index+1])[]))\*100  
 cat("\n MAPE:", MAPE)  
   
 MPE = max((exp(Z[-1]) - exp(W[index+1]))/(exp(W[index+1])))\*100  
 MNE = min((exp(Z[-1]) - exp(W[index+1]))/(exp(W[index+1])))\*100  
 cat("\n MPE: ", MPE)  
 cat("\n MNE: ", MNE)  
   
 P90 = quantile(abs((exp(Z[-1]) - exp(W[index+1]))/(exp(W[index+1])))\*100, 0.9)  
 P95 = quantile(abs((exp(Z[-1]) - exp(W[index+1]))/(exp(W[index+1])))\*100, 0.95)  
 cat("\n P90: ", P90)  
 cat("\n P95: ", P95)  
   
 RMSFE = mean((exp(Z[-1]) - exp(W[index+1]))^2)  
 cat("\n RMSFE: ", RMSFE)  
 return(RMSFE)  
  
}  
  
# Adding Lags (There is definitely an easier way to do this)  
add\_lags = function(X,Y) {  
 X = cbind(Y, X)  
   
 X\_sem\_1 = head(X,-1)  
 X\_sem\_2 = head(X\_sem\_1,-1)  
 X\_sem\_3 = head(X\_sem\_2,-1)  
 X\_sem\_4 = head(X\_sem\_3,-1)  
 X\_sem\_5 = head(X\_sem\_4,-1)  
 X\_sem\_6 = head(X\_sem\_5,-1)  
 X\_sem\_7 = head(X\_sem\_6,-1)  
 X\_sem\_8 = head(X\_sem\_7,-1)  
 X\_sem\_9 = head(X\_sem\_8,-1)  
 X\_sem\_10 = head(X\_sem\_9,-1)  
 X\_sem\_11 = head(X\_sem\_10,-1)  
   
 X\_s\_0 = tail(X,nrow(X\_sem\_11))  
 X\_s\_1 = tail(X\_sem\_1,nrow(X\_sem\_11))  
 X\_s\_2 = tail(X\_sem\_2,nrow(X\_sem\_11))  
 X\_s\_3 = tail(X\_sem\_3,nrow(X\_sem\_11))  
 X\_s\_4 = tail(X\_sem\_4,nrow(X\_sem\_11))  
 X\_s\_5 = tail(X\_sem\_5,nrow(X\_sem\_11))  
 X\_s\_6 = tail(X\_sem\_6,nrow(X\_sem\_11))  
 X\_s\_7 = tail(X\_sem\_7,nrow(X\_sem\_11))  
 X\_s\_8 = tail(X\_sem\_8,nrow(X\_sem\_11))  
 X\_s\_9 = tail(X\_sem\_9,nrow(X\_sem\_11))  
 X\_s\_10 = tail(X\_sem\_10,nrow(X\_sem\_11))  
 X\_s\_11 = tail(X\_sem\_11,nrow(X\_sem\_11))  
  
 X\_total = cbind(X\_s\_0,  
 X\_s\_1,  
 X\_s\_2,  
 X\_s\_3,  
 X\_s\_4,  
 X\_s\_5,  
 X\_s\_6,  
 X\_s\_7,  
 X\_s\_8,  
 X\_s\_9,  
 X\_s\_10,  
 X\_s\_11  
 )  
   
 troca\_nome = function(x, numero = n) {return(paste("L",n," - ", x, sep = ""))}  
 nome\_all = c()  
 for (i in 1:12) {  
 n = i  
 nome\_in = unname(lapply(colnames(X), troca\_nome))  
 nome\_all = c(nome\_all, nome\_in)  
 }  
 colnames(X\_total) = unlist(nome\_all)  
 return(X\_total)  
}