Rcode:

library(ggplot2)

library(forecast)

library(zoo)

library(tseries)

library(urca)

retail.data <- read.csv("retailsales.csv")

retail.ts <- ts(retail.data$retailsales, start=c(1992,1), freq=12)

# basicplots

plot(retail.ts,xlab="Year",ylab="retailsales",bty="l")

p <- autoplot(retail.ts)+theme\_light(base\_size=16)

print(p)

**# fit linear trend**

retail.lm <- tslm(retail.ts ~ trend )

retail.lm <- tslm(retail.ts ~ trend + I(trend^2) + I(trend^3))

#plot with fitted line

plot(retail.ts,xlab="Time",ylab="Retailsales")

lines(retail.lm$fitted,lwd = 2)

grid()

# ggplot2 seasonal plotting

p <- ggseasonplot(retail.ts)

print(p)

# ggplot2 seasonal plotting

p <- ggsubseriesplot(retail.ts)

print(p)

# ggplot2 seasonal plotting

p <- gglagplot(retail.ts)

print(p)

# Compute and plot autocorrelations

p <- ggAcf(retail.ts)

print(p)

**#training&validation of cubic Trend model**

retail.ts <- ts(retail.data$retailsales,start=c(1992,1), freq=12)

# forecast months into the future

stepsAhead <- 12

# define training period

nTrain <- length(retail.ts) - stepsAhead

# split data

train.ts <- window(retail.ts, start = c(1992, 1), end = c(1992, nTrain))

valid.ts <- window(retail.ts, start = c(1992, nTrain+1), end = c(1992, nTrain + stepsAhead))

# fit quadratic trend model and perform forecasts

# fit quadratic trend model and perform forecasts

retail.lm <- tslm(train.ts ~ trend + I(trend^2) + I(trend^3))

retail.lm.pred <- forecast(retail.lm, h = stepsAhead, level = 95)

plot(retail.lm.pred)

# plot fitted values in the training period

lines(retail.lm$fitted, lwd = 2)

# plot data in the validation period

lines(valid.ts)

grid()

**#Differencing + Log**

retaillog.ts <- log(retail.ts)

diff1 <- diff(retaillog.ts,lag=1)

diff12 <- diff(retaillog.ts,lag=12)

# now difference again

diff12and1 <- diff( diff(retaillog.ts,lag=12),lag=1)

plot(diff1,lwd=1)

plot(diff12,lwd=1)

plot(diff12and1, lwd=1)

difftrain.ts <- window(diff12and1, start = c(1993, 2), end = c(2019, 3))

diffvalid.ts <- window(diff12and1, start = c(2019, 4), end = c(2020, 3))

**#training&validation of cubic Trend model of differenced data**

# fit quadratic trend model and perform forecasts

# fit quadratic trend model and perform forecasts

diffretail.lm <- tslm(difftrain.ts ~ trend + I(trend^2) + I(trend^3))

diffretail.lm.pred <- forecast(diffretail.lm, h = stepsAhead, level = 95)

plot(diffretail.lm.pred)

# plot fitted values in the training period

lines(diffretail.lm$fitted, lwd = 2)

# plot data in the validation period

lines(diffvalid.ts)

grid()

**# Errors of Cubic Trend Model Forecast in Training and Validation**

valid.err <- valid.ts - retail.lm.pred$mean

valid.err

# train forecast errors

train.err <- train.ts - retail.lm$fitted.values

retail.lm$fitted.values

head (train.err)

plot(train.err)

lines(valid.err,col="red")

grid()

# find mse in training and validation periods

mseTrain <- mean( train.err^2 )

mseTrain

mseValid <- mean(valid.err^2, na.rm=TRUE)

print(mseTrain)

print(mseValid)

**# Errors of differenced Cubic Trend Model Forecast in Training and Validation**

diffvalid.err <- diffvalid.ts - diffretail.lm.pred$mean

# train forecast errors

difftrain.err <- difftrain.ts - diffretail.lm$fitted.values

diffretail.lm$fitted.values

head (difftrain.err)

plot(difftrain.err)

lines(diffvalid.err,col="red")

grid()

# find mse in training and validation periods

mseTrain <- mean( difftrain.err^2 )

mseTrain

mseValid <- mean(diffvalid.err^2, na.rm=TRUE)

print(mseTrain)

print(mseValid)

**# Errors of Cubic Trend fitted Model**

p<-ggAcf(retail.lm$residuals, lag.max=48)

print(p)

**# Errors of differenced Cubic Trend fitted Model**

p<-ggAcf(diffretail.lm$residuals, lag.max=48)

print(p)

**# Naïve Model**

# Find naive forecast

retail.lm.naive <- naive(train.ts, h = stepsAhead, level = 0)

retail.lm.naive

# quadratic trend forecast

retail.lm.pred <- forecast(retail.lm, h = stepsAhead, level = 0)

retail.lm.pred

# now generate rolling random walk forecast

rollfcast <- rep(0, stepsAhead)

for (t in (1:stepsAhead)) {

rollfcast[t] <- retail.ts[nTrain+t-1]

}

rollfcast

# this builds a time series from the rolling naive forecast

retail.roll.ts <- ts(rollfcast, start=c(1992, nTrain+1),frequency=12)

retail.roll.ts

plot(retail.lm.pred)

# plot data in the validation period

lines(valid.ts)

lines(retail.lm$fitted,lwd=2)

lines(retail.lm.naive$mean,lty=2,col="red",lwd=2)

lines(retail.roll.ts,lwd=2,col="green")

grid()

# we can now compare all forecasts

# Let's brute force find rmse

mse.trend <- mean( (valid.ts - retail.lm.pred$mean)^2)

rmse.trend <- sqrt(mse.trend)

mse.naive <- mean( (valid.ts - retail.lm.naive$mean)^2)

rmse.naive <- sqrt(mse.naive)

mse.roll <- mean( (valid.ts - retail.roll.ts)^2)

rmse.roll <- sqrt(mse.roll)

# print out various forecast summaries

# note comparisons with our own rmse calculations

# note that when used with forecast objects,

# accuracy will give results for both training/validation periods

print(rmse.trend)

print(accuracy(retail.lm.pred,valid.ts))

print(rmse.naive)

print(accuracy(retail.lm.naive,valid.ts))

print(rmse.roll)

print(accuracy(retail.roll.ts,valid.ts))

**Different Naïve Forecasts**

retail.lm.meanf <- meanf(train.ts, h=stepsAhead, level=0)

retail.lm.naive <- naive(train.ts, h = stepsAhead, level = 0)

retail.lm.snaive <- snaive(train.ts, h = stepsAhead, level=0)

retail.lm.drift <- rwf(train.ts, h = stepsAhead, drift = TRUE, level = 0)

# plot the forecast in the validation period

plot(retail.lm.naive, xlim = c(2010, 2020))

# plot data in the validation period

lines(retail.lm.meanf$mean,col="black",lwd=2)

lines(retail.lm.snaive$mean,col="green",lwd=2)

lines(retail.lm.drift$mean,col="red",lwd=2)

lines(valid.ts,lty="dashed")

legend("topleft",col=c("blue","red","green","black"),legend=c("naive","drift","seasonal","mean"),lty=1)

grid()

**# Naïve Model on differenced data**

# Find naive forecast

diffretail.lm.naive <- naive(difftrain.ts, h = stepsAhead, level = 0)

diffretail.lm.naive

# quadratic trend forecast

diffretail.lm.pred <- forecast(diffretail.lm, h = stepsAhead, level = 0)

diffretail.lm.pred

# now generate rolling random walk forecast

diffrollfcast <- rep(0, stepsAhead)

for (t in (1:stepsAhead)) {

diffrollfcast[t] <- diff12and1[nTrain+t-1]

}

diffrollfcast

# this builds a time series from the rolling naive forecast

diffretail.roll.ts <- ts(diffrollfcast, start=c(2019, 4),frequency=12)

diffretail.roll.ts

plot(diffretail.lm.pred)

# plot data in the validation period

lines(diffvalid.ts)

lines(diffretail.lm$fitted,lwd=2)

lines(diffretail.lm.naive$mean,lty=2,col="red",lwd=2)

lines(diffretail.roll.ts,lwd=2,col="green")

grid()

# we can now compare all forecasts

# Let's brute force find rmse

mse.trend <- mean( (diffvalid.ts - diffretail.lm.pred$mean)^2)

rmse.trend <- sqrt(mse.trend)

mse.naive <- mean( (diffvalid.ts - diffretail.lm.naive$mean)^2)

rmse.naive <- sqrt(mse.naive)

mse.roll <- mean( (diffvalid.ts - diffretail.roll.ts)^2)

rmse.roll <- sqrt(mse.roll)

# print out various forecast summaries

# note comparisons with our own rmse calculations

# note that when used with forecast objects,

# accuracy will give results for both training/validation periods

print(rmse.trend)

print(accuracy(diffretail.lm.pred,diffvalid.ts))

print(rmse.naive)

print(accuracy(diffretail.lm.naive, diffvalid.ts))

print(rmse.roll)

print(accuracy(diffretail.roll.ts, diffvalid.ts))

**Different Naïve Forecasts on Differenced Data**

diffretail.lm.meanf <- meanf(difftrain.ts, h=stepsAhead, level=0)

diffretail.lm.naive <- naive(difftrain.ts, h = stepsAhead, level = 0)

retail.lm.snaive <- snaive(train.ts, h = stepsAhead, level=0)

diffretail.lm.drift <- rwf(difftrain.ts, h = stepsAhead, drift = TRUE, level = 0)

# plot the forecast in the validation period

plot(diffretail.lm.naive, xlim = c(2010, 2020))

# plot data in the validation period

lines(diffretail.lm.meanf$mean,col="black",lwd=2)

lines(retail.lm.snaive$mean,col="green",lwd=2)

lines(diffretail.lm.drift$mean,col="red",lwd=2)

lines(diffvalid.ts,lty="dashed")

legend("topleft",col=c("blue","red","green","black"),legend=c("naive","drift","seasonal","mean"),lty=1)

grid()

**#Moving Average**

# trailing MA

ma.trailing <- rollmean(retail.ts, k=12, align="right")

# centered MA (two ways)

ma.centered <- ma(retail.ts,order=12)

plot(retail.ts,xlab="Time",ylab="retailsales",xlim = c(2010,2020),bty="l")

lines(ma.centered,lwd=2)

lines(ma.trailing,lwd=2, lty=2)

legend(x="topleft",c("retailsales","Centered moving average","Trailing moving average"),

lty=c(1,1,2),lwd=c(1,2,2),bty="n")

grid()

last.ma <- tail(ma.trailing, 1)

# build forecasting time series from just that last value

# a kind of naive forecast

ma.trailing.pred <- ts(rep(last.ma, stepsAhead), start=c(1992, nTrain+1),

end=c(1992, nTrain+stepsAhead), freq=12)

ma.trailing.pred

plot(train.ts,xlab="Time",ylab="retailsales",bty="l",xlim = c(2010,2020))

lines(ma.trailing,lwd=2, col= "blue")

lines(ma.trailing.pred, lwd = 2, col = "red", lty=2)

lines(valid.ts)

grid()

**#Exponential Filter on differenced data**

retaillog.ts <- log(retail.ts)

diff.ts <- diff(diff(retaillog.ts, lag=12), lag=1)

difftrain1.ts <- window(diff.ts, start = c(1993, 2), end = c(2019, 3))

diffvalid1.ts <- window(diff.ts, start = c(2019, 4), end = c(2020, 3))

# fit exponential filter

ses <- ets(difftrain1.ts, model="ANN", alpha=0.2)

ses

# find forecast

ses.pred <- forecast(ses, h=stepsAhead, level=95)

ses.pred

plot(ses.pred, ylab="Passengers",xlab="Time",bty="l", xlim=c(2010,2020),main="", flty=2)

lines(ses.pred$fitted, lwd=2,col="blue")

lines(valid.ts)

grid()

# Note, no parameter given ETS will find it

ses.opt <- ets(difftrain1.ts,model="ANN", opt.crit = 'mse')

ses.opt.pred <- forecast(ses.opt, h=stepsAhead, level=0)

ses.opt.pred

print(accuracy(ses.pred, diffvalid1.ts))

print(accuracy(ses.opt.pred, diffvalid1.ts))

#RMSE

print(sqrt(mean((diffvalid1.ts - ses.pred$mean)^2,na.rm = TRUE) ))

print(sqrt(mean((diffvalid1.ts - ses.opt.pred$mean)^2,na.rm = TRUE) ))

**Trend and Seasonality using ETS Function**

# fit additive model, with additive trend

sest <- ets(train.ts, model="AAN")

sest

sest.pred <- forecast(sest, h=stepsAhead, level=0)

# fit additive model, with additive seasonal

sess <- ets(train.ts, model="AAA")

sess

sess.pred <- forecast(sess, h=stepsAhead, level=0)

sess.pred

plot(sess.pred, ylab="retailsales",xlab="Year",bty="l",xlim=c(2010,2020),main="", flty=2)

lines(valid.ts)

grid()

print(accuracy(sest.pred,valid.ts))

print(accuracy(sess.pred,valid.ts))

**Dickey Fuller Test**

df.test <- ur.df(train.ts,type="trend",selectlags="BIC")

print(summary(df.test))

**ARMA Model**

# First, estimate additive trend, no seasonal

train.ar1 <- Arima(train.ts,order=c(2,0,0),xreg=(1:nTrain))

train.ar2 <- Arima(train.ts,order=c(2,1,0),xreg=(1:nTrain))

train.ar3 <- Arima(train.ts,order=c(2,1,2),xreg=(1:nTrain))

# now try seasonal AR model (manual)

train.ar4 <- Arima(train.ts,order=c(2,0,0),xreg=(1:nTrain),seasonal=list(order=c(1,0,0),period=12))

train.ar5 <- Arima(train.ts,order=c(2,1,0),xreg=(1:nTrain),seasonal=list(order=c(1,0,0),period=12))

# auto seasonal

train.ar6 <- auto.arima(train.ts,d=0,ic="bic",seasonal=TRUE)

train.ar7 <- auto.arima(train.ts,d=1,ic="bic",seasonal=FALSE)

# Now, build forecasts for validation periods (uses no data there)

train.pred1 <- forecast(train.ar1,xreg = (nTrain+1):(nTrain+stepsAhead), h=stepsAhead, level=0)

train.pred2 <- forecast(train.ar2,xreg = (nTrain+1):(nTrain+stepsAhead), h=stepsAhead, level=0)

train.pred3 <- forecast(train.ar3,xreg = (nTrain+1):(nTrain+stepsAhead), h=stepsAhead, level=0)

train.pred4 <- forecast(train.ar4,xreg = (nTrain+1):(nTrain+stepsAhead), h=stepsAhead, level=0)

train.pred5 <- forecast(train.ar5,xreg = (nTrain+1):(nTrain+stepsAhead), h=stepsAhead, level=0)

train.pred6 <- forecast(train.ar6, h = stepsAhead, level = 0)

train.pred7 <- forecast(train.ar7, h = stepsAhead, level = 0)

#printaccuracy

print(accuracy(train.pred1))

print(accuracy(train.pred2))

print(accuracy(train.pred3))

print(accuracy(train.pred4))

print(accuracy(train.pred5))

print(accuracy(train.pred6))

print(accuracy(train.pred7))

# plot all the results

plot(train.pred6,xlab="Time",ylab="retailsales",bty="l",xlim = c(2010,2020))

lines(valid.ts)

lines(train.pred$fitted,col="red")

grid()

**Trend & Seasonal Filter**

# First, estimate addiive trend/seasonal filter

train.lm.season <- tslm(train.ts ~ trend + season)

# Now, build forecasts for validation periods (uses no data there)

train.lm.season.pred <- forecast(train.lm.season, h=stepsAhead, level=0)

# plot all the results

plot(train.lm.season.pred,xlab="Time",ylab="retailsales",bty="l",xlim = c(2010,2020))

lines(valid.ts)

grid()

print(accuracy(train.lm.season.pred,valid.ts))

**ACF for all models**

#ACF for Trend model controlled for seasonality

Acf(train.lm.season.pred$residuals)

#ACF for Trend AutoAurima Model --> train.ts,d=0,ic="bic",seasonal=TRUE--> ARIMA (3,0,1) (1,1,1) [12] with drift

Acf(train.pred6$residuals)

#ACF for ARIMA (2,0,0)

Acf(train.pred1$residuals)

**Cointegration**

# Cointegration model for interest rates

# USe government bond yields for 10 year and 3 month treasuries

library(urca)

library(forecast)

PI.data <- read.csv("PI.csv")

# Convert to TS, note this is monthly

PI.ts <- ts(PI.data$PI,start=c(1959,1), freq=12)

PI.ts <- window(PI.ts, start=c(1992,1), end=c(2020,1))

retail.ts <- ts(retail.data$retailales, start=c(1992,1), freq=12)

retail.ts <- window(retail.ts, end=c(2020,1))

# Cointegration model and residuals

creg.model <- lm( PI.ts ~ retail.ts)

creg.res <- residuals(creg.model)

creg.ts <- ts(creg.res,start=c(1992,1),freq=12)

ret.ts <- diff(retail.ts)

PIN.ts <- diff(PI.ts)

retlag.ts <- lag(ret.ts,-1,na.pad=TRUE)

PINLAG.ts <- lag(PIN.ts,-1,na.pad=TRUE)

creglag.ts <- lag(creg.ts,-1,na.pad=TRUE)

veclags.ts <- cbind( ret.ts, PIN.ts, retlag.ts, PINLAG.ts, creg.ts, creglag.ts)

vecseriestrain.ts <- window(veclags.ts, end=c(2019,3))

vecseriesvalid.ts <- window(veclags.ts, start=c(2019,4))

# Full sample

ec.model <- lm( ret.ts ~ retlag.ts + PINLAG.ts + creglag.ts, data=veclags.ts )

print("Full sample: EC model")

print(summary(ec.model))

#now benchmark model without error correction

ecbench.model <- lm( ret.ts ~ retlag.ts + PINLAG.ts, data=veclags.ts )

print("Full sample: EC benchmarkmodel")

print(summary(ecbench.model))

# Do training and validation predictions

# first full model

ectrain.model <- lm(ret.ts ~ retlag.ts + PINLAG.ts + creglag.ts, data = vecseriestrain.ts )

pred <- predict(ectrain.model,vecseriesvalid.ts)

print("in and out of sample: ec model")

print(accuracy(ectrain.model))

print(accuracy(pred,as.vector(vecseriesvalid.ts[,1])))

# now benchmark model w/o error correction component

ectrain.model <- lm(ret.ts ~ retlag.ts + PINLAG.ts, data = vecseriestrain.ts )

pred <- predict(ectrain.model,vecseriesvalid.ts)

print("in and out of sample: benchmark model")

print(accuracy(ectrain.model))

print(accuracy(pred,as.vector(vecseriesvalid.ts[,1])))