HWForecastExample

Reed Thunstrom

First Read in Data

#Install forecast packages  
install.packages("forecast", repo="http://cran.us.r-project.org")

##   
## The downloaded binary packages are in  
## /var/folders/69/c32333f95yl1mmqptcszh0t80000gn/T//Rtmpx7BjkB/downloaded\_packages

library(forecast)

## Warning: package 'forecast' was built under R version 3.2.5

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.2.5

##   
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

## Loading required package: timeDate

## This is forecast 7.3

#Update link to your file location to read in data  
comp\_data<- read.csv('/Users/Thunstrom/Documents/MSPA/PREDICT 413/Midterm/competitiondata.csv', header=TRUE)  
  
#converting my dataset date value to an actual date, it is text in the .csv  
comp\_data$DATE<- as.Date(as.character(comp\_data$DATE), "%Y%m%d")  
  
#Creating a year and month variable, because my dataset did not have those  
comp\_data["Year"] <- as.numeric(substr(as.character(comp\_data$DATE),1,4))  
comp\_data["Month"] <- as.numeric(substr(as.character(comp\_data$DATE),6,7))  
  
#Aggregate precipitation by month  
comp\_data\_month\_year<- (aggregate(comp\_data[,'PRCP'], list(comp\_data$Month, comp\_data$Year), sum))  
df<- data.frame(comp\_data\_month\_year)  
  
#final dataset  
comp\_data\_df<- setNames(df, c("Month", "Year", "PRCP"))  
  
#Look at the data to make sure it's good to go  
head(comp\_data\_df)

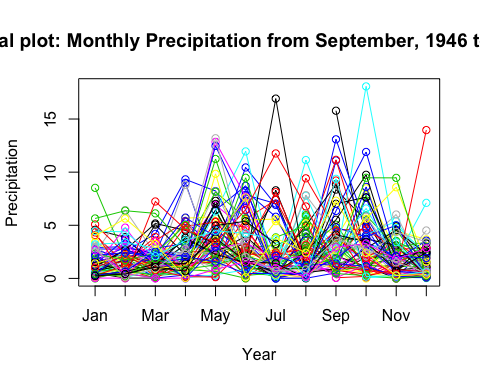
## Month Year PRCP  
## 1 9 1946 15.78  
## 2 10 1946 1.31  
## 3 11 1946 1.86  
## 4 12 1946 2.43  
## 5 1 1947 2.14  
## 6 2 1947 0.29

Then create a time series data set so we can use time series modeling

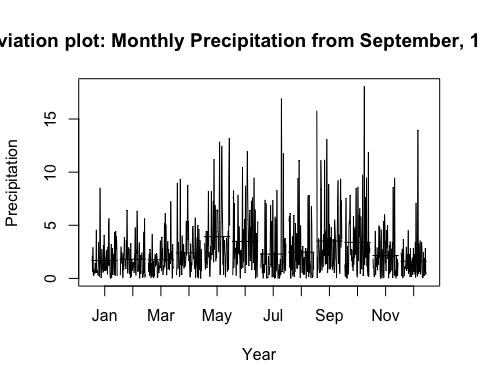
#The varialbe I'm trying to forecast is PRCP, update to yours  
#Frequency is 12 because it's monthly data, update to your specs  
#Starting on September, 1946, update to your timeframe  
monthly\_ts\_all<- ts(comp\_data\_df$PRCP, frequency = 12, start = c(1946, 9))

Creating a season and month plot to see trends year over year

#probably want to restrict the data set, this is a lot of data so it looks weird  
seasonplot(monthly\_ts\_all,ylab="Precipitation", xlab="Year",   
 main="Seasonal plot: Monthly Precipitation from September, 1946 to July, 2014", col=1:12)



monthplot(monthly\_ts\_all,ylab="Precipitation",xlab="Year",xaxt="n",  
 main="Seasonal deviation plot: Monthly Precipitation from September, 1946 to July, 2014")  
axis(1,at=1:12,labels=month.abb,cex=0.8)



Now we'll create a training and testing dataset. Training will be used to fit the model, testing set will be used to assess the error

#training set is September, 1946 until July, 2012  
train <- window(monthly\_ts\_all, start=c(1946, 9), end=c(2012, 7))  
  
#testing set is August, 2012 and the next 24 months (24 month forecasting project)  
test <- window(monthly\_ts\_all,start=c(2012, 8))  
  
#HW needs non-zero observations, so if there are any 0s, update them to 0.001  
train[train==0] <- 0.001  
test[test==0] <- 0.001  
monthly\_ts\_all[monthly\_ts\_all==0] <- 0.001

Next decide which hyperparameters in the HW model work the best for your test set

#Function allows for s and d to be variable  
#We use s for the two seasonal options and d for the dampened trend options  
getHW <- function(s, d) {  
 forecast <- hw(train, seasonal = s, damped = d, h = 24)  
 return(accuracy(forecast,test)[2,"RMSE"])  
}  
  
getHW("additive", TRUE)

## [1] 2.438169

getHW("additive", FALSE)

## [1] 2.39452

getHW("multiplicative", TRUE)

## [1] 2.384303

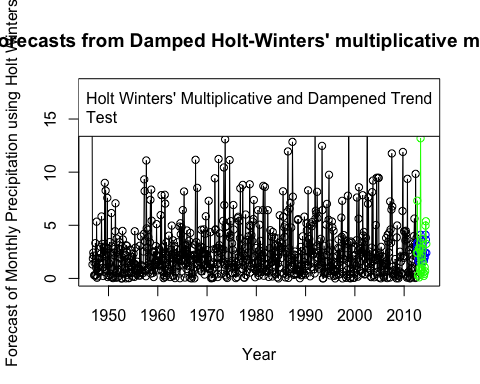
getHW("multiplicative", FALSE)

## [1] 2.420744

Chose the model with the lowest RMSE. If they are the same, you'll probably have to do some more investigating. Like plotting the prediction intervals. I'll add that code later.

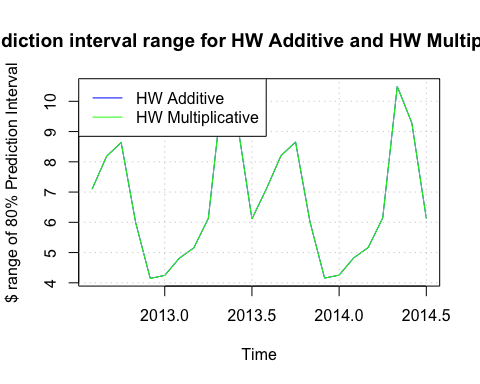
Now let's plot our training model with the next 24 month predictions, which will be used to compare against our test set. If our model is good, it should overlay the test set observations

forecastsbest<- hw(train, seasonal = "multiplicative", damped = TRUE, h = 24)  
plot(forecastsbest,ylab="Forecast of Monthly Precipitation using Holt Winters",  
 plot.conf=FALSE, type="o", fcol="white", xlab="Year")  
  
lines(forecastsbest$mean, type="o", col="blue")  
lines(test, type = "o", col = "green")  
legend("topright",lty=1, pch=1, col=c("blue", "green"),   
 c("Holt Winters' Multiplicative and Dampened Trend", "Test"))



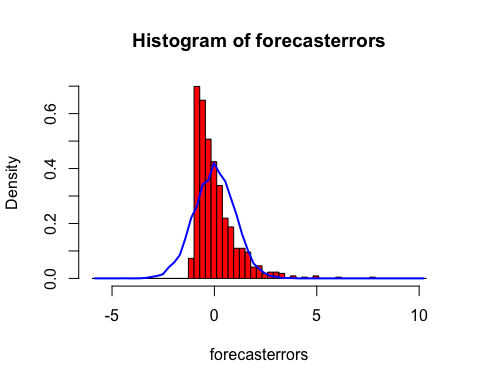
Prediction interval code if you want to check models.

#You'll need to update this code after you create your model. My model in these scenarios are called forecastsbest, so the prediction intervals are the same. You'll want to make sure you change that to the two (or more) models you wish to examine  
  
#Upper PI 80% for HW   
hw\_add\_upper<- forecastsbest$upper[,1]  
hw\_mult\_upper<- forecastsbest$upper[,1]  
  
#Lower PI 80% for HW   
hw\_add\_lower<- forecastsbest$lower[,1]  
hw\_mult\_lower<- forecastsbest$lower[,1]  
  
#The subtract the lower value from the upper to get the range.  
  
#Range of PI  
pi\_range\_add<- hw\_add\_upper - hw\_add\_lower  
pi\_range\_mult<- hw\_mult\_upper - hw\_mult\_lower  
  
#Finally, plot the range of the prediction intervals and see how they behave  
  
#Plot PI range  
plot(pi\_range\_add, ylab = "$ range of 80% Prediction Interval",   
main = "Prediction interval range for HW Additive and HW Multiplicative",  
col = "blue")  
lines(pi\_range\_mult, col = "green")  
grid()  
legend("topleft",lty=1, col=c("blue","green"),   
c("HW Additive", "HW Multiplicative"))

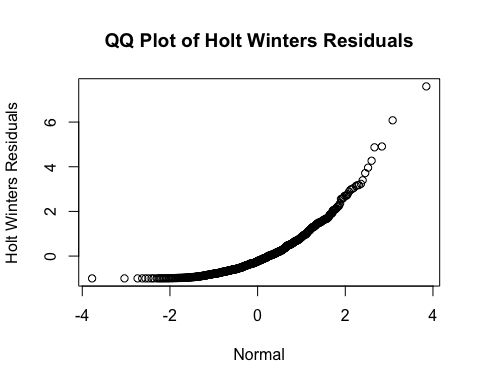


Examine residuals to make sure it looks adequate.

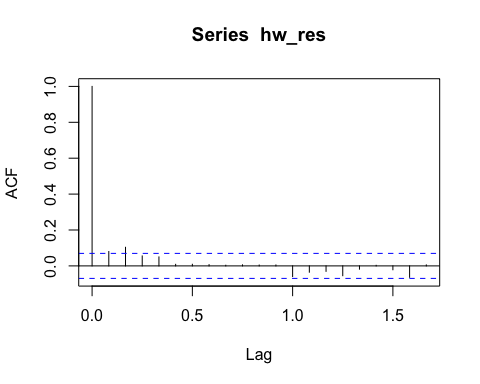
#Create a random population to make sure residuals are normal  
mynorm <- rnorm(10000, mean=0, sd=1)  
  
#Creating plotting function for a histogram  
plotForecastErrors <- function(forecasterrors)  
 {  
 # make a histogram of the forecast errors:  
 mybinsize <- IQR(forecasterrors)/4  
 mysd <- sd(forecasterrors)  
 mymin <- min(forecasterrors) - mysd\*5  
 mymax <- max(forecasterrors) + mysd\*3  
 # generate normally distributed data with mean 0 and standard deviation mysd  
 mynorm <- rnorm(10000, mean=0, sd=mysd)  
 mymin2 <- min(mynorm)  
 mymax2 <- max(mynorm)  
 if (mymin2 < mymin) { mymin <- mymin2 }  
 if (mymax2 > mymax) { mymax <- mymax2 }  
 # make a red histogram of the forecast errors, with the normally distributed data overlaid:  
 mybins <- seq(mymin, mymax, mybinsize)  
 hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)  
 # freq=FALSE ensures the area under the histogram = 1  
 # generate normally distributed data with mean 0 and standard deviation mysd  
 myhist <- hist(mynorm, plot=FALSE, breaks=mybins)  
 # plot the normal curve as a blue line on top of the histogram of forecast errors:  
 points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)  
}  
  
#Get the residuals from the model  
hw\_res<- forecastsbest$residuals  
  
#Plot them with a histogram  
plotForecastErrors(hw\_res)



#Run a QQ plot to make sure it's normal  
qqplot(mynorm, hw\_res, xlab = "Normal", ylab = "Holt Winters Residuals", main = "QQ Plot of Holt Winters Residuals")



#Run an autocorrelation correlgram to make sure not too many values fall outside the thresholds  
acf(hw\_res, lag.max = 20)



Then examine all error, not just RMSE

accuracy(forecastsbest, test)

## ME RMSE MAE MPE MAPE MASE  
## Training set 0.04363971 2.467418 1.760952 -2669.1819 2699.7969 0.7050051  
## Test set -0.09045256 2.384303 1.646927 -205.0236 227.4268 0.6593545  
## ACF1 Theil's U  
## Training set 0.046807328 NA  
## Test set -0.001270933 1.072479

If that looks good, use your model on the entire dataset to predict the next X observations. In my case I'm looking for the next 24.

#Build the model on all values not just train  
forecastsbest<- hw(monthly\_ts\_all, seasonal = "multiplicative", damped = TRUE, h = 24)  
  
#mean values are the predictions  
hw\_pred<- forecastsbest$mean  
  
#print out predictions  
hw\_pred

## Jan Feb Mar Apr May Jun Jul  
## 2014   
## 2015 1.658768 1.892947 1.815540 2.289482 4.195678 3.443395 2.391629  
## 2016 1.656335 1.890317 1.813150 2.286627 4.190722 3.439542 2.389094  
## Aug Sep Oct Nov Dec  
## 2014 2.789442 3.340206 3.414430 2.121950 1.381034  
## 2015 2.784079 3.334123 3.408539 2.118481 1.378896  
## 2016

#plot all data and then add predictions  
plot(monthly\_ts\_all, main = "Forecasted values using HW")  
grid()  
  
#Dashed line for predictions are easy to read  
lines(hw\_pred, col="green", lty=2, lwd=1.5)

