Comp 4442 Project: Paul O'Leary

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```
knitr::opts_chunk$set(echo = TRUE)
#install.packages("ggseas")
#install.packages("tseries")
#install.packages("forecast")
#install.packages("Metrics")
#install.packages("outliers")
#install.packages("sarima")
#install.packages("fpp2")
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(ggpubr)
library(HistData)
library(boot)
library(ggseas)
library(tseries)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(stats)
library(forecast)
```

```
##
## Attaching package: 'forecast'
```

```
## The following object is masked from 'package:ggpubr':
##
##     gghistogram
```

```
library(fpp2)
```

```
## -- Attaching packages ---------------------------------------------- fpp2 2.4 --
```

```
## v fma       2.4     v expsmooth 2.3
```

```
## -- Conflicts ------------------------------------------------ fpp2_conflicts --
## x forecast::gghistogram() masks ggpubr::gghistogram()
```

```
library(Metrics)
```

```
##
## Attaching package: 'Metrics'
```

```
## The following object is masked from 'package:forecast':
##
##     accuracy
```

```
library(outliers)
library(sarima)
```

```
## Loading required package: FitAR
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'lattice'
```

```
## The following object is masked from 'package:boot':
##
##     melanoma
```

```
## Loading required package: leaps
```

```
## Loading required package: ltsa
```

```
## Loading required package: bestglm
```

```
##
## Attaching package: 'FitAR'
```

```
## The following object is masked from 'package:forecast':
##
##     BoxCox
```

```
## Loading required package: stats4
```

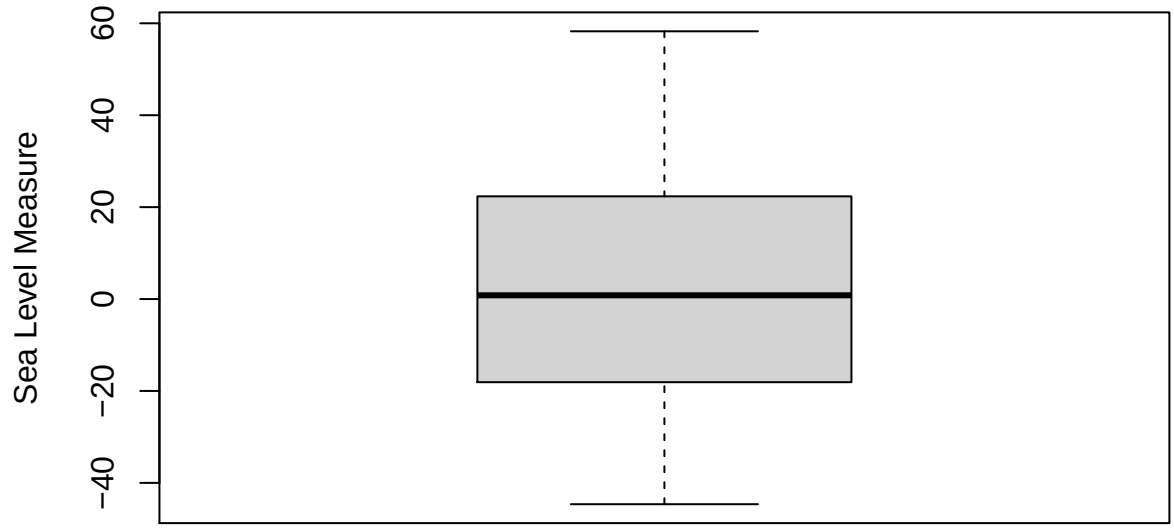
Note to Dr. Christensen. This R Notebook contains a lot of commented out stuff. I have left a lot of the various attempts I made to get my head around everything SARIMA. Apologies if the mess is too much.

Time Series analysis of sea level data.

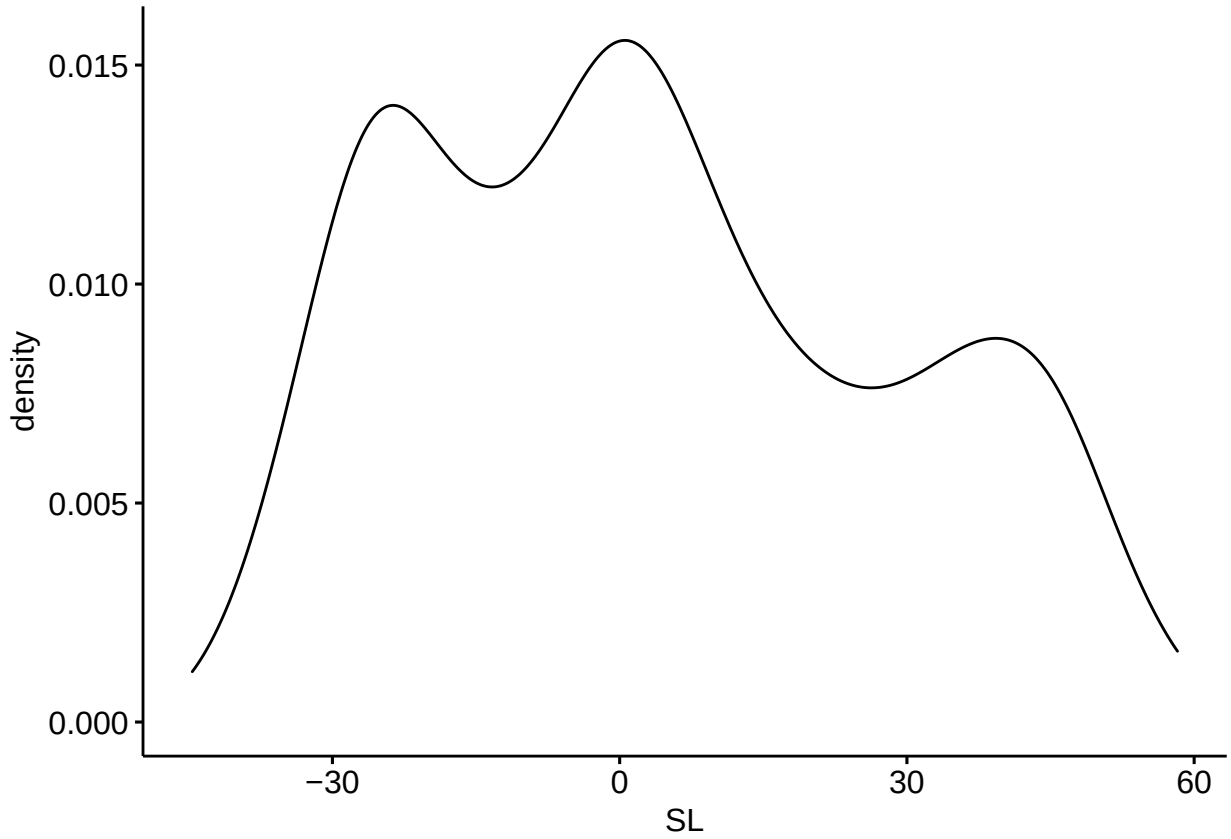
Code to import file. Basic data file analysis.

```
# Code to pull the file in, and do a basic plot ALL data.  Data Manipulations.

wths <- c(3,5,14,10,10,10,10,10,10,10,10,10)

datSL <- read.fwf("GMSL_DATA.txt", wths, header = FALSE)

# datSL2 <- datSL[3,6]
datSL <- data.frame(datSL$V3,datSL$V6)

names(datSL)[1] <- "Yr"
names(datSL)[2] <- "SL"

# Basic data analysis
```

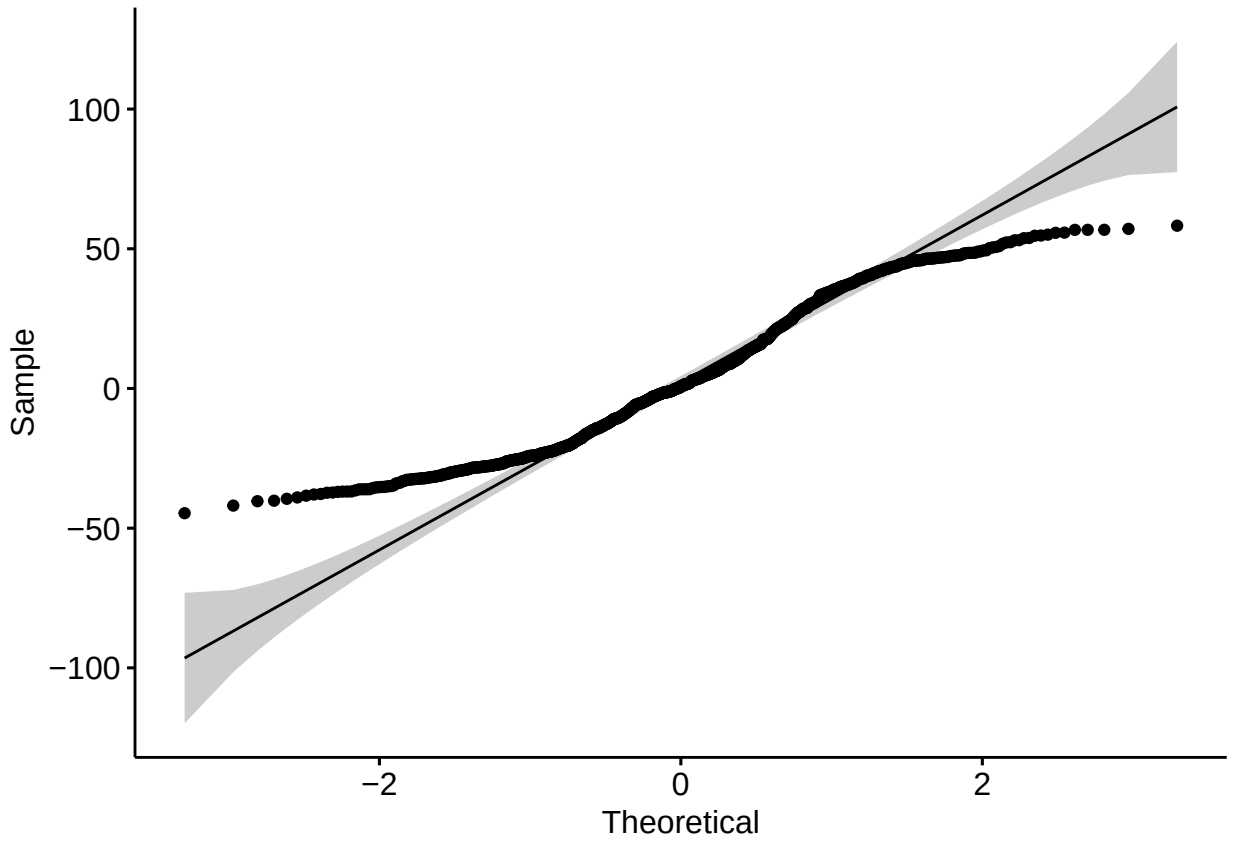
```r
boxplot(datSL$SL, ylab = "Sea Level Measure")
```




```r
# qqplot(datSL$Yr, datSL$SL)

ggdensity(datSL, x = "SL")
```




```r
ggqqplot(datSL, x = "SL")
```




```
# Any missing data
# Missing or BAD data flag in data file is 99900.000
sum(datSL == 99900.000)
```

```
## [1] 0
```

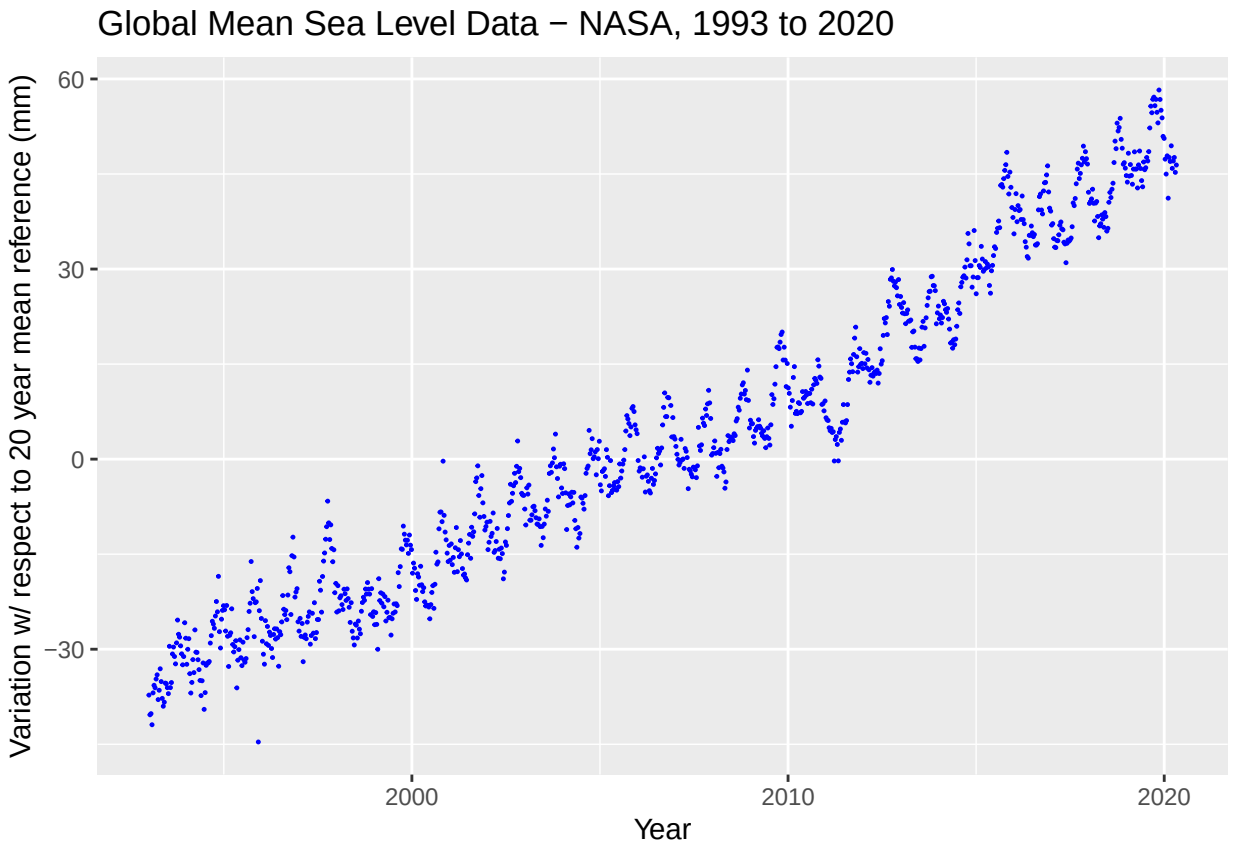
```
sum(is.na(datSL))
```

```
## [1] 0
```

```
# No missing or bad data
```

I found conflicting information on the web about the importance of Normality in Time Series Data. I wonder if the fact that the data is centered around a 20 year mean reference affects the plots. This could be addressed by shifting the data all into the positive?

```
# Basic plot of raw data

gSL <- ggplot(datSL, aes(x=Yr, y=SL))
gSL <- gSL + geom_point(size = .2, color="blue") + ggtitle("Global Mean Sea Level Data - NASA, 1993 to 1
gSL <- gSL + labs(x="Year", y="Variation w/ respect to 20 year mean reference (mm)")
gSL
```



Select the appropriate ARIMA model? acf() and pacf() with plot="false"

Code from here through line 187 covers my experimentation with figuring out how SARIMA works.

```
# First passes at trying to determine the components of the SARIMA analysis.

# acf(datProjDiffs1)
# acf(datProjDiffs2)

# # Values of the AutoCorrelation
# datCorr <- acf(datProjDiffs4)
#
# # Values of partial AutoCorrelations
# datCorrPartial <- pacf(datProjDiffs4)
```

Tail Off to 0.

Looks like for ACF it comes into the boundaries of the CI at 15 for 3 OR 4 differences For PACF, looks like 12 for 4 differences.

ARMA(13, 15) ???

Try auto-arima:

```
# Just to check that it is aware of number of seasonal changes.  ??
# auto.arima(datProj$V6, seasonal = TRUE, stationary = FALSE)
#
# auto.arima(datProjDiffs1)
```

```
#let this run for a while:
#auto.arima(datProjTS)
```

#RUN on the TS data The time series frequency has been rounded to support seasonal differencing.Series: datProjTS ARIMA(5,0,2)(2,1,1)[36] with drift

Coefficients: NaNs produced ar1 ar2 ar3 ar4 ar5 ma1 ma2 sar1 sar2 sma1 drift -0.0973 0.496 0.0952 0.2230 0.1038 0.6379 -0.1079 -0.4491 -0.2088 -0.1699 0.0863 s.e. 0.0017 NaN 0.0042 0.0029 0.0030 0.0266 0.0297 0.0073 NaN 0.0124 0.0105

sigma^2 estimated as 7.536: log likelihood=-2358.37

OLD STUFF ******* Auto-ARIMA returns ARIMA(3,1,3), with mean=0 when run on SL

Run on Diffs1, mean is 0, and (3,0,3) (But this is after already running a DIFF - makes sense)

(Run on Diffs2-4, the mean is non-zero, and get (3,0,0), (1,0,5) and (1,0,0) respectively) ********** Website, Forecasting with long seasonal data: https://robjhyndman.com/hyndsight/longseasonality/

```
# METHOD = LM ??

#arimaResults <- arima(datProjTS, order = c(5,0,2), seasonal = c(2,1,1), method = "ML")
#
```

RUNS SLOW

OLD (Going with (3,1,3) for now, and method "ML")

These got he lowest AICc and BIC close to the values determined by Auto.ARIMA. We KNOW

What about Seasonal. Arima

```
#sarimaResults <- Arima(datProjTS, order = c(3,1,3), seasonal = c(3,1,3), method = "ML")
#sarimaResults
```

Back to Arima results: Check the residuals

```
#checkresiduals(arimaResults)
```

Forecast:

Showing no cyclical movement. Model bad?

```
#arimaResults %>% forecast() %>% autoplot()
```

Everything Above was my first attempts to make sense of all this.

****

REAL WORK IS BELOW:

****

Split the data, 70% into training data, 30% into test data. For Time Series data this must be sequential - meaning the first 70% is the data from 1993 through 2012, and the 30% test data is 2012 through 2020.

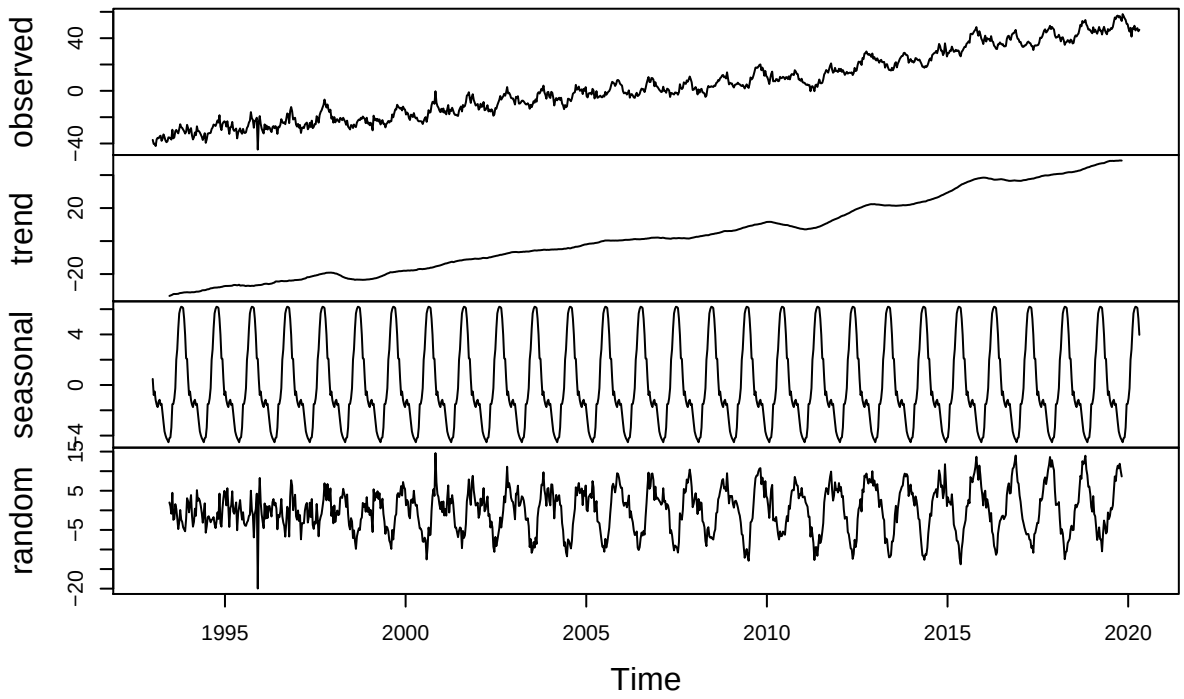
```
# Create Time Series data, and split into 70%/30%

datSL_TS <- ts(datSL$SL, frequency = 36.8, start = c(1993, 1))

# Split into 70% / 30%  SEA LEAVEL DATA

trainTS <- head(datSL_TS, round(length(datSL_TS) * 0.7))
x <- length(datSL_TS) - length(trainTS)
testTS <- tail(datSL_TS, x)

# Always experimenting.
# sarima(SL~Temp, data = trainTS, ss.method = "sarima", use.symmetry = FALSE, SSinit = "Rossignol2011")
```

Is the data Additive or Multiplicative? Additive.

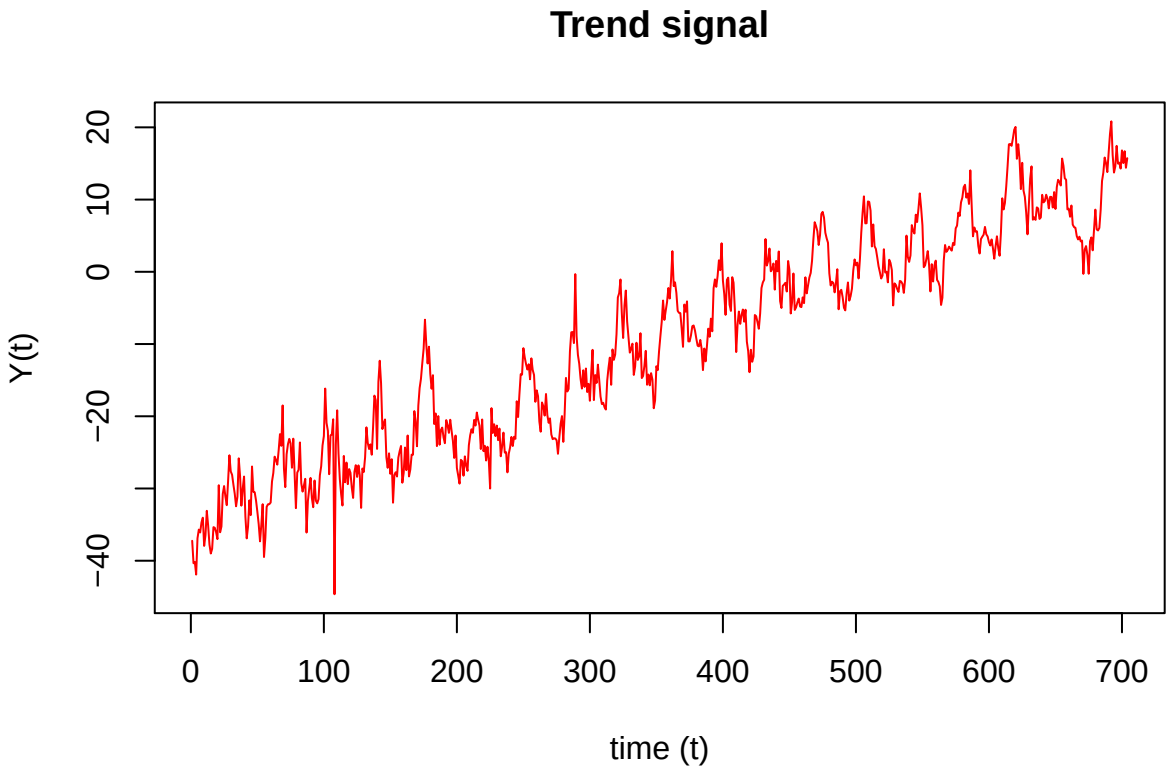
Is the data stationary?

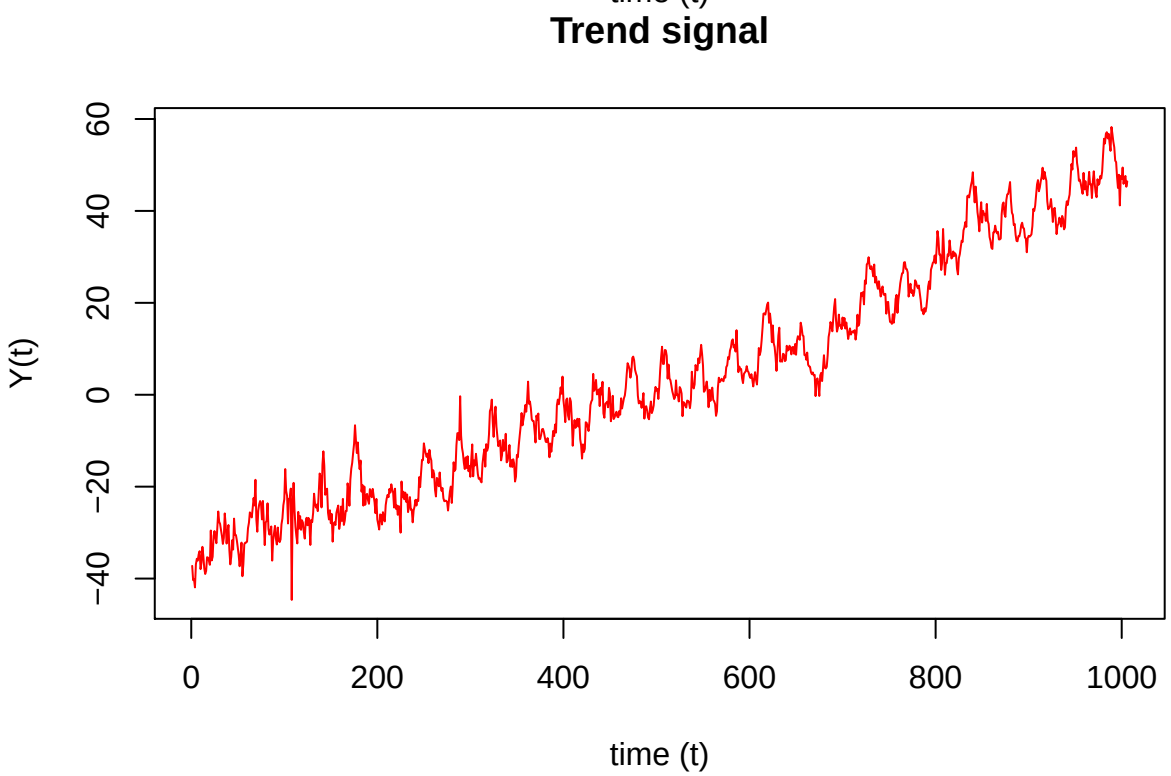
Use the Augmented Dickey-Fuller test to check.

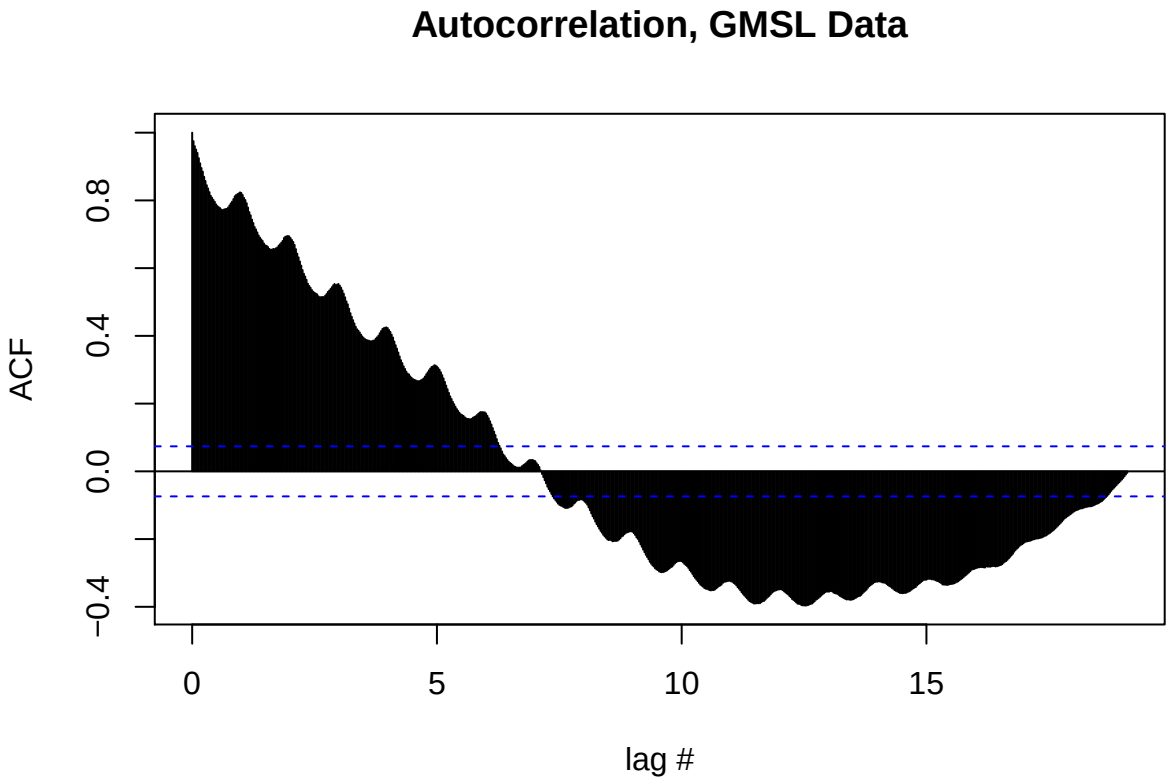
Then run Autocorrelation of a univariate time series to check as well.

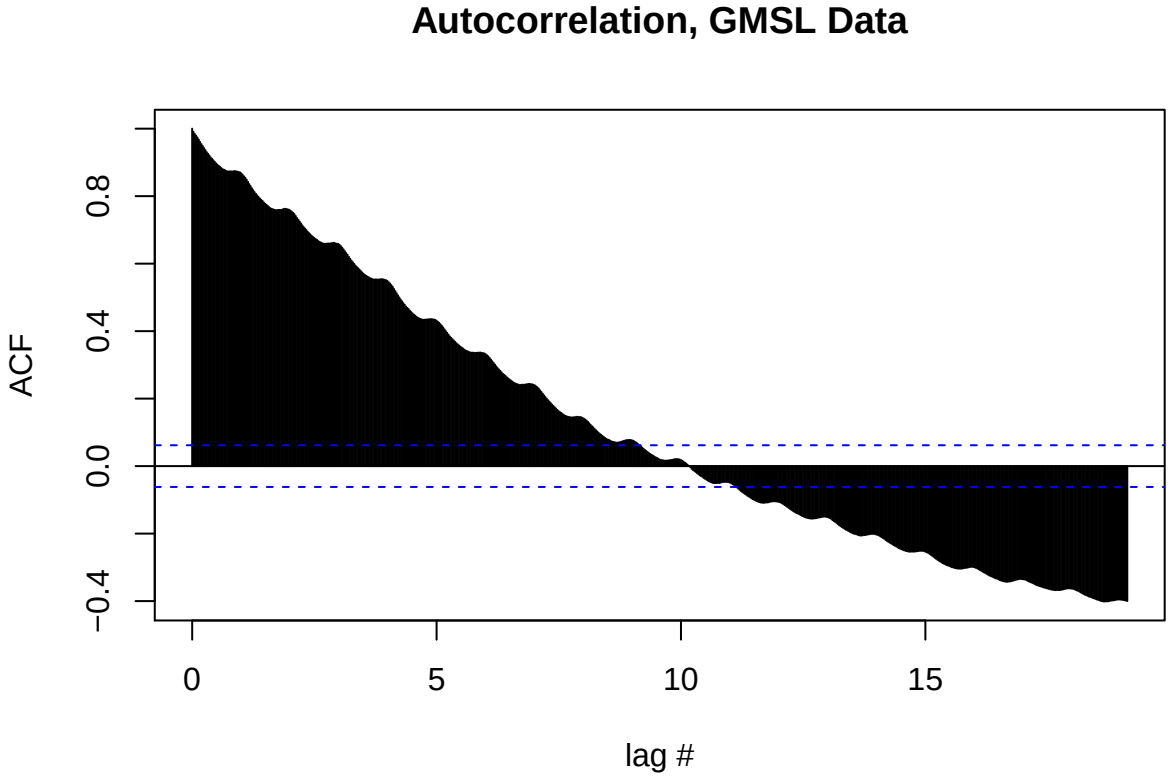



```
## Warning in adf.test(trainTS): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  trainTS
## Dickey-Fuller = -8.0222, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(datSL_TS): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  datSL_TS
## Dickey-Fuller = -6.7844, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(datSL_TEST): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  datSL_TEST
## Dickey-Fuller = -6.7844, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```


Trend signal
Y(t)
20
10
0
−20
−40
0
100
200
300
400
500
600
700
time (t)


Trend signal
Y(t)
60
40
20
0
−20
−40
0
200
400
600
800
1000
time (t)






```
## [1] -44.64
```

```
## Time Series:
## Start = 1993
## End = 2012.10326086957
```

```
## Frequency = 36.8
##   [1] -1.818959410 -2.028742079 -2.016590342 -2.134787044 -1.795372448
##   [6] -1.714534768 -1.741475254 -1.647879656 -1.603460301 -1.866820531
##  [11] -1.767747406 -1.539546896 -1.675480446 -1.854010862 -1.937634546
##  [16] -1.894466898 -1.691639521 -1.697699659 -1.740128102 -1.803458925
##  [21] -1.300940478 -1.739454532 -1.685579649 -1.382229943 -1.309000569
##  [26] -1.409110945 -1.487090692 -1.264674495 -1.023072593 -1.173368902
##  [31] -1.204246846 -1.296238925 -1.380886005 -1.497177051 -1.409783025
##  [36] -1.051245205 -1.214988141 -1.491797579 -1.332511247 -1.220359013
##  [41] -1.591348284 -1.796720159 -1.684233046 -1.442046095 -1.579910070
##  [46] -1.127060892 -1.362743883 -1.366103390 -1.443390528 -1.548291334
##  [51] -1.663362362 -1.824351318 -1.666728392 -1.477677339 -1.970695136
##  [56] -1.792677067 -1.502556708 -1.478349703 -1.477677339 -1.462213753
##  [61] -1.266689084 -1.188136011 -1.035816911 -1.066003702 -1.110285194
##  [66] -0.980149528 -0.826626160 -0.935893183 -0.560662319 -1.145851282
##  [71] -1.319076193 -1.013682497 -0.920472375 -0.870863859 -0.913767968
##  [76] -1.139140232 -0.868852900 -1.194848698 -1.514661613 -1.184108509
##  [81] -1.157931727 -0.903711686 -1.281463408 -1.360728210 -1.304970478
##  [86] -1.241844008 -1.742148834 -1.448096129 -1.335870170 -1.233115437
##  [91] -1.421880936 -1.505919087 -1.257287856 -1.445407197 -1.470281524
##  [96] -1.430618867 -1.209617419 -1.124376690 -0.931870273 -0.844722545
## [101] -0.403982198 -0.722091105 -0.796468074 -1.198876420 -0.838690286
## [106] -0.829977254 -0.688593643 -2.318752776 -0.923824644 -0.605533831
## [111] -1.006304832 -1.247215637 -1.384917850 -1.490452740 -1.029109286
## [116] -1.274076098 -1.090827053 -1.291537494 -1.151220280 -1.186122250
## [121] -1.324450092 -1.419864556 -1.175382534 -1.114982255 -1.224387266
## [126] -1.116995312 -1.208946089 -1.512644063 -1.145851282 -1.178067409
## [131] -1.042524735 -0.763632782 -0.908404568 -0.965396521 -0.921813277
## [136] -1.018377497 -0.758272256 -0.469594267 -0.511777939 -0.964055383
## [141] -0.342393978 -0.146281018 -0.355782215 -0.779714925 -0.726111030
## [146] -0.690603396 -1.041183154 -1.139811327 -1.004963463 -1.195519979
## [151] -1.059295172 -1.464230658 -1.206260793 -1.180752319 -1.220359013
## [156] -1.044537121 -0.984173232 -0.936563675 -1.278105503 -1.186793501
## [161] -0.951314942 -1.161287534 -0.839360531 -1.219687646 -1.155247122
## [166] -1.017036058 -1.017706777 -0.612901283 -0.707351805 -0.939245658
## [171] -0.562001696 -0.399295920 -0.313610143 -0.168367027 -0.035854625
## [176]  0.233857251  0.004299661 -0.171713422 -0.016446706 -0.266085728
## [181] -0.405990614 -0.280141957 -0.733481023 -0.635004499 -0.937904663
## [186] -0.659118610 -0.925165565 -0.787756319 -0.766313080 -0.859468619
## [191] -0.911086254 -0.696632727 -0.744201316 -0.815232734 -0.693283085
## [196] -0.795127787 -0.887622444 -1.047891138 -0.839360531 -1.141153524
## [201] -1.212302762 -1.286836185 -1.065332840 -1.077408674 -1.212974103
## [206] -1.033133837 -1.119679420 -1.167999315 -0.929858842 -0.837349802
## [211] -0.781725253 -0.814562547 -0.694622938 -0.747551482 -0.626297016
## [216] -0.695962796 -0.748221520 -0.965396521 -0.690603396 -0.984843856
## [221] -0.935222694 -1.072041554 -0.943939199 -1.068687172 -1.332511247
## [226] -0.585441489 -0.816573114 -0.734821039 -0.845392804 -0.747551482
## [231] -0.882929934 -0.775024212 -1.032463074 -0.940586661 -0.812551994
## [236] -0.998256736 -0.994232792 -1.182094788 -1.009658287 -0.950644411
## [241] -0.852765760 -0.937234168 -0.850754934 -0.870193538 -0.521822232
## [246] -0.668496759 -0.457542458 -0.270101770 -0.275456522 -0.029831473
## [251] -0.114825702 -0.178406236 -0.227934204 -0.179075519 -0.318295806
## [256] -0.124195314 -0.231280769 -0.278133910 -0.527179280 -0.420719217
## [261] -0.474950719 -0.710031628 -0.804509923 -0.533206036 -0.568028943
```

```
## [266] -0.655099480 -0.454194785 -0.649740709 -0.722091105 -0.680554749
## [271] -0.829977254 -0.873545162 -0.866171644 -0.866841955 -0.884940999
## [276] -1.009658287 -0.859468619 -0.730801006 -0.659118610 -0.899018888
## [281] -0.644382018 -0.305577648 -0.431431172 -0.407999037 -0.058608812
## [286]  0.117401960  0.119409733  0.017015178  0.656338326  0.083939351
## [291] -0.092740352 -0.175059824 -0.315618281 -0.404651669 -0.234627345
## [296] -0.389254022 -0.217894575 -0.436787228 -0.360468162 -0.517804488
## [301] -0.258723032 -0.045223984 -0.511108327 -0.278803259 -0.352435131
## [306] -0.182421940 -0.323650885 -0.478298531 -0.547938457 -0.535884621
## [311] -0.577404836 -0.598836269 -0.331683578 -0.209862939 -0.117502729
## [316] -0.369840157 -0.041877782 -0.138249820 -0.090063354  0.099332092
## [321]  0.440033983  0.479536847  0.606772512  0.293428277  0.063861989
## [326]  0.365722776  0.504311444  0.215785884  0.071892919 -0.071324441
## [331] -0.033846907  0.011661275 -0.278133910 -0.200492770  0.020361368
## [336] -0.138919085 -0.106125387  0.108701634 -0.306916391 -0.287504838
## [341] -0.192461257 -0.054593360 -0.369170725 -0.274117831 -0.374526207
## [346] -0.261400369 -0.320303956 -0.586780946 -0.515795629 -0.197146300
## [351] -0.233288713 -0.056601085  0.081931609  0.214447277  0.411915103
## [356]  0.231849306  0.316856254  0.396517264  0.527748089  0.429991360
## [361]  0.602084192  0.869422289  0.545158967  0.579982864  0.482215120
## [366]  0.316186874  0.297444452  0.294767000  0.150195861 -0.017785182
## [371]  0.374425411  0.308154351  0.403881397  0.033076902  0.032407663
## [376]  0.089293338  0.173620490  0.180313313  0.132794939  0.058508046
## [381] -0.007746615 -0.013100517  0.046461699 -0.035185386 -0.233288713
## [386] -0.033177668 -0.152304443 -0.008415853  0.150865131  0.075239145
## [391]  0.243897039  0.126102324  0.525739186  0.607442277  0.538462394
## [396]  0.638252732  0.784974932  0.692512292  0.942497221  0.596726181
## [401]  0.472841228  0.278702476  0.603423706  0.623516982  0.373086536
## [406]  0.314848114  0.628205566  0.577303999  0.321541937 -0.065970487
## [411]  0.187675459  0.309493099  0.195037648  0.279371825  0.328235822
## [416]  0.215116581  0.326227650  0.030399946 -0.057939570 -0.252699052
## [421] -0.044554743 -0.156320072 -0.108133149  0.277363780  0.265984947
## [426]  0.211100766  0.149526592  0.292758917  0.527748089  0.579313146
## [431]  0.604763225  0.982060398  0.735390168  0.776263513  0.894225236
## [436]  0.682463588  0.702561294  0.754821086  0.512346729  0.778943922
## [441]  0.682463588  0.866741027  0.406559286  0.342293187  0.547167960
## [446]  0.559891800  0.575964572  0.494937120  0.778273818  0.696531856
## [451]  0.291420198  0.397186727  0.662367060  0.324219484  0.348317860
## [456]  0.395178341  0.429991360  0.356350841  0.350326096  0.440703498
## [461]  0.387144864  0.626865964  0.477528152  0.552525319  0.625526368
## [466]  0.667726021  0.778943922  0.975354320  1.138368094  1.102132306
## [471]  1.054498317  0.926405541  1.018276507  1.214887058  1.234357173
## [476]  1.181322486  1.040411367  0.988096029  0.947190823  0.663036926
## [481]  0.551185973  0.586010377  0.573955441  0.486902133  0.575964572
## [486]  0.701891351  0.330244000  0.496945888  0.511007505  0.442712047
## [491]  0.344301405  0.320203168  0.475519465  0.579313146  0.410576144
## [496]  0.452085381  0.521721407  0.691172447  0.792346331  0.738070223
## [501]  0.762861820  0.616149324  0.796367171  1.039740582  1.225628946
## [506]  1.378768943  1.127630909  1.128301966  1.331066548  1.327035957
## [511]  1.245771615  0.912326162  1.117565301  0.915678325  0.889532609
## [516]  0.812451086  0.729360125  0.675094727  0.615479544  0.642941503
## [521]  0.887521508  0.671075414  0.679114086  0.579982864  0.789665801
## [526]  0.760851617  0.694522068  0.365722776  0.570606912  0.555873704
## [531]  0.518373286  0.490919610  0.593377461  0.587349837  0.565249321
```

```
## [536]  0.482215120  0.606772512  1.013581509  0.816472204  0.769562593
## [541]  0.829206118  1.111526169  1.057852484  1.033032841  1.208173681
## [546]  1.139710281  1.263230338  1.406321459  1.273303424  1.108171170
## [551]  0.720650264  0.737400207  0.799047762  0.870092609  0.739410259
## [556]  0.495606708  0.748790672  0.587349837  0.736730193  0.781624356
## [561]  0.602753948  0.583331471  0.542480325  0.370408795  0.436686421
## [566]  0.779614029  0.926405541  0.862719185  0.879477259  0.911655735
## [571]  0.892214100  0.874114561  0.945849785  0.927076007  1.081333082
## [576]  1.106158196  1.227643117  1.196090189  1.322333718  1.367346043
## [581]  1.464129430  1.486317049  1.366674131  1.404977329  1.307556061
## [586]  1.620855874  1.299496035  1.006874535  1.089384156  1.049802569
## [591]  1.053156665  0.916348763  0.847302670  0.982060398  1.003521120
## [596]  1.022300868  1.094751706  1.025654555  1.005533163  0.951213981
## [601]  0.921712329  0.978036729  0.896906775  0.799047762  0.915007889
## [606]  1.007545223  0.896906775  0.826525247  1.041082155  1.360627045
## [611]  1.256515252  1.316959883  1.471524960  1.657202353  1.863347925
## [616]  1.866719014  1.849190334  1.918645272  1.999613400  2.025939932
## [621]  1.727902934  1.864696350  1.727229424  1.445978210  1.690864807
## [626]  1.431189828  1.374065303  1.227643117  1.025654555  1.298152732
## [631]  1.542808803  1.657875523  1.161186478  1.183336199  1.159844149
## [636]  1.275989673  1.265916441  1.168569446  1.182664959  1.392880644
## [641]  1.322333718  1.335097230  1.393552660  1.357939519  1.266587972
## [646]  1.374065303  1.372721430  1.274646543  1.418419117  1.261215787
## [651]  1.464801737  1.532719474  1.513215317  1.479593201  1.730597007
## [656]  1.663261000  1.546844722  1.533392075  1.256515252  1.264573385
## [661]  1.188706199  1.293451252  1.115552238  1.092738859  1.086700431
## [666]  1.009557301  0.978036729  1.003521120  0.959931050  0.967977851
## [671]  0.658347893  0.884840065  0.916348763  0.833227473  0.659687610
## [676]  0.960601606  0.996814434  0.876795897  1.067915290  1.255172264
## [681]  1.071269660  1.061206701  1.085358582  1.255172264  1.521285704
## [686]  1.599321534  1.740026684  1.688844831  1.604031899  1.787185016
## [691]  1.961146730  2.077935553  1.762930076  1.599994434  1.657202353
## [696]  1.849190334  1.683458373  1.692884812  1.637008727  1.806726968
## [701]  1.688171513  1.799314160  1.644412725  1.733964674

##
##  chi-squared test for outlier
##
## data:  trainTS
## X-squared = 5.3433, p-value = 0.0208
## alternative hypothesis: lowest value -44.64 is an outlier
```

The Data is NOT Stationary.

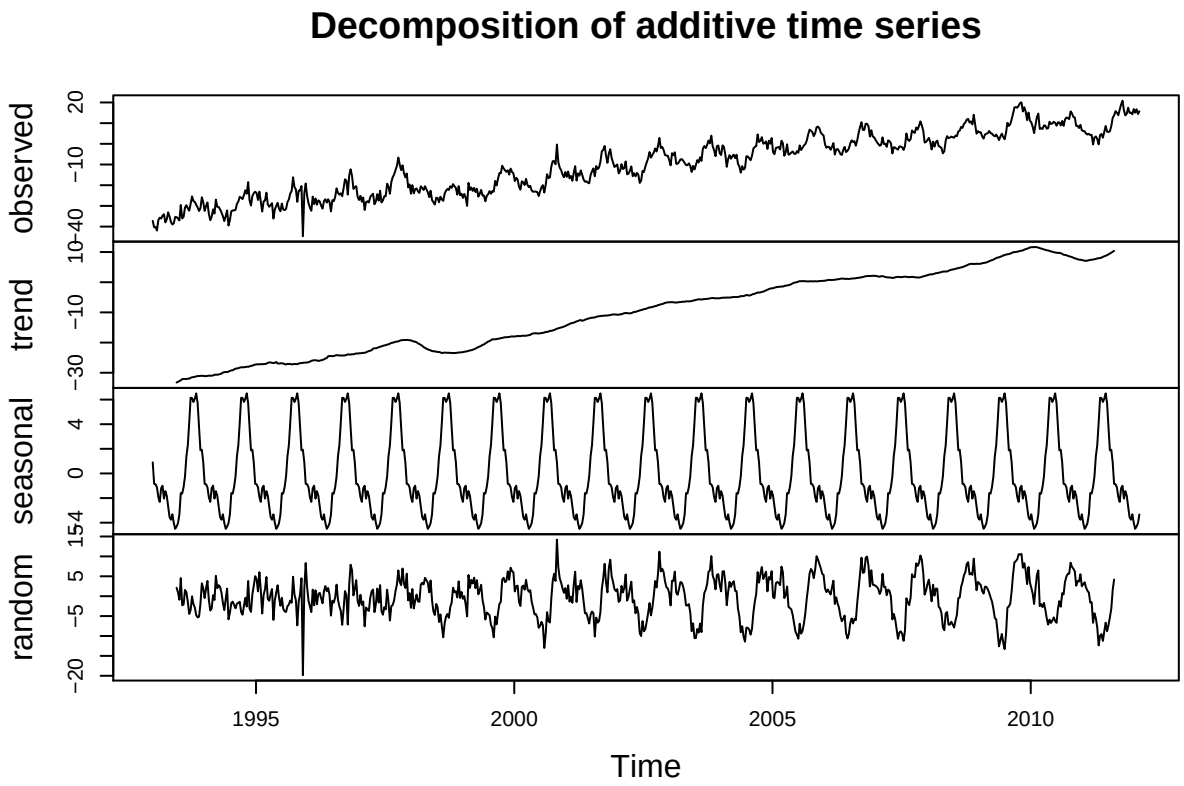
The Dickey Fuller test returned a p-value of less than .01, meaning to reject the null hypothesis that the data is stationary.

The serial correlation, or autocorrelation calculates the correlation with observations from previous time steps - these are called lags. Almost all of the lags for this data is outside the 95% confidence interval.meaning the data is NOT stationary - the MEAN is changing over time.

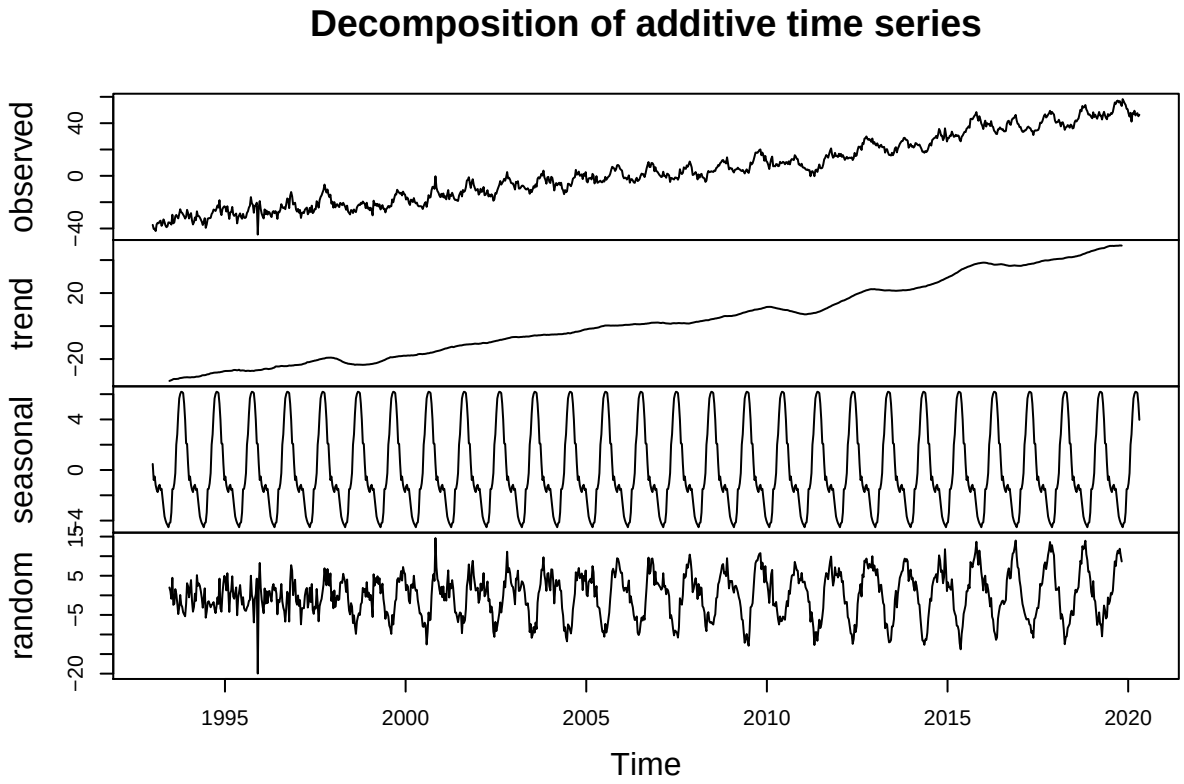
Decomposition

The data looks to be Seasonal. We can decompose it, and look at some graphs to get a better idea.

```r
#Decomposed
decTrainTS <- decompose(trainTS, type = "additive")
plot(decTrainTS)
```




```r
decFullData <- decompose(datSL_TS, type = "additive")
plot(decFullData)
```




```
# I attempted some seasonal adjusting to get the forecasting to work.
# # Seasonally adjusted
# decTrainTSSeasAdjusted <- trainTS - decTrainTS$seasonal
# plot(decTrainTSSeasAdjusted)
```

The Decomposition of the trainTS data is shown above.

observed: shows the raw data for the training Time Series set

trend: shows that the data is definitely trending upwards

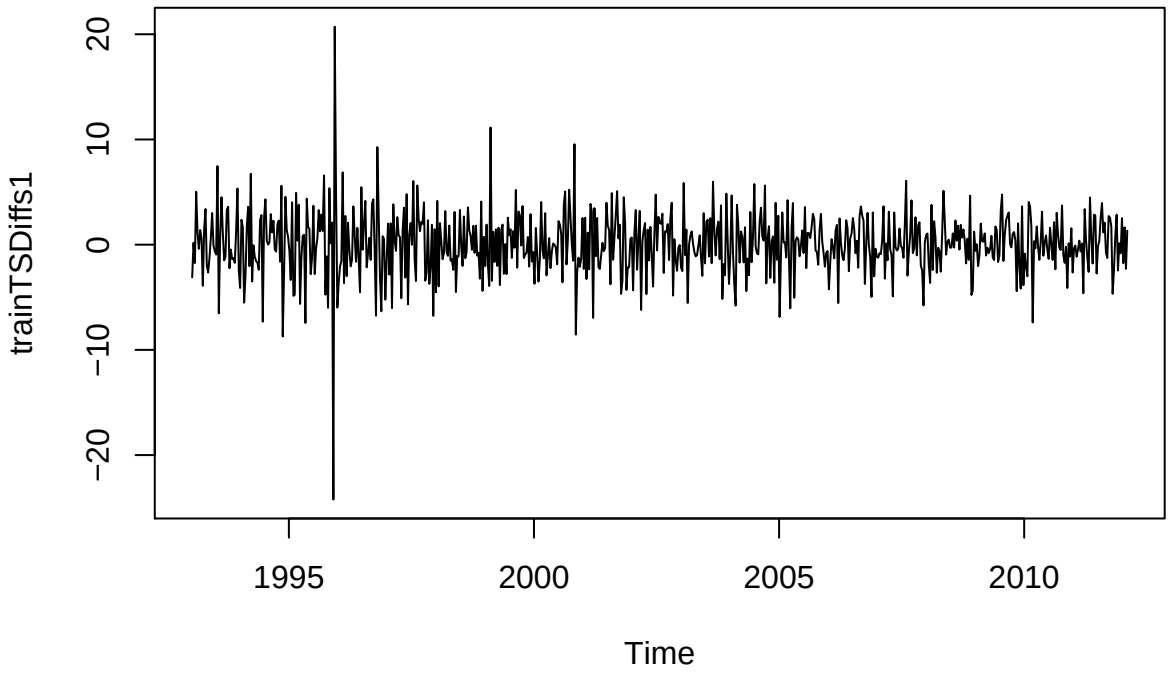
seasonal: The data shows a definite cyclical season

random: describes the "noise" in the data. This represents the remainder after the other components have been removed.

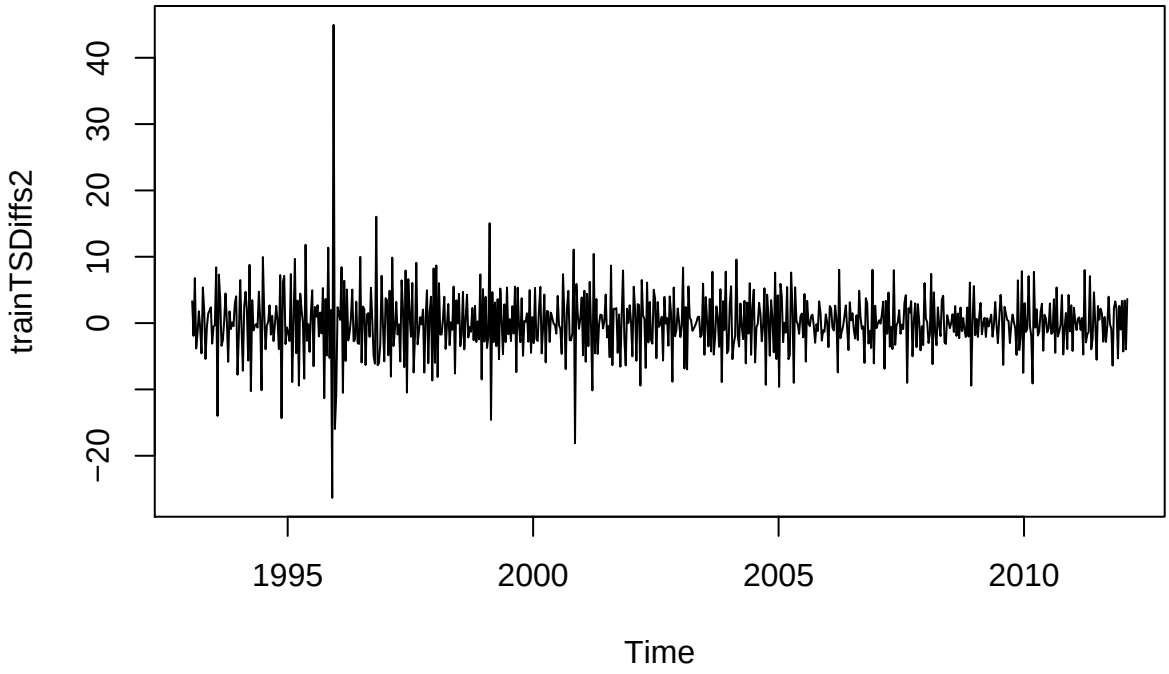
website: https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html

Differencing the data, because it is Seasonal

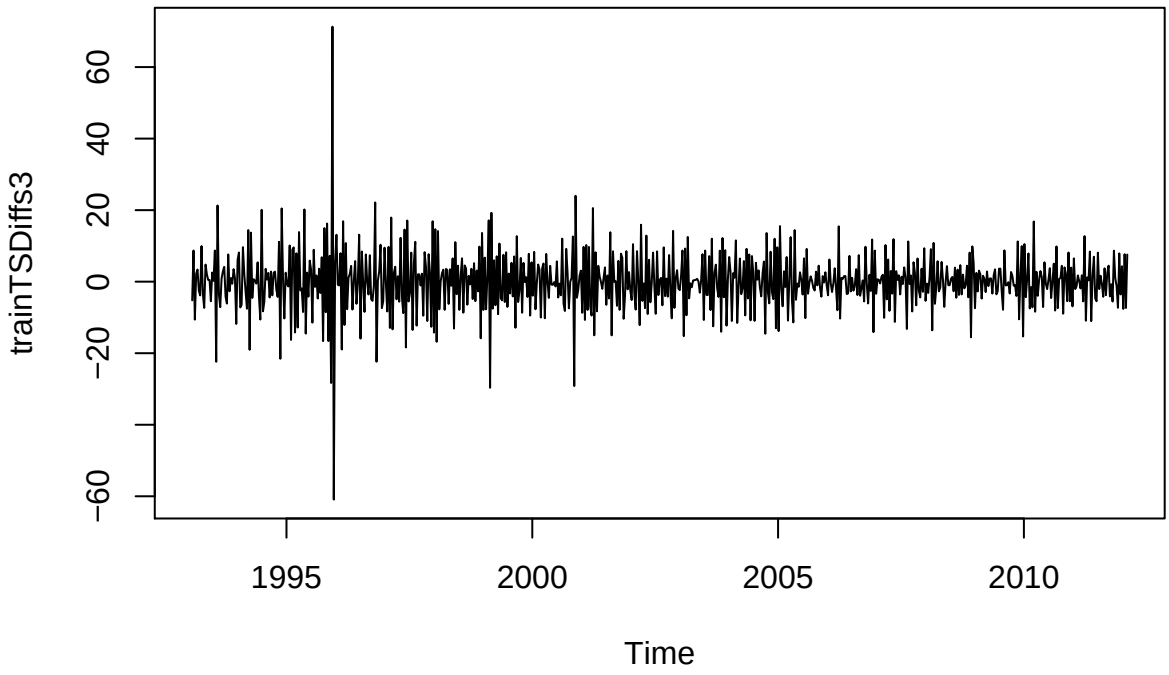
```
# Calc various Difference levels in R, to REMOVE the Seasonal component of the data:
trainTSDiffs1 <- diff(trainTS, differences = 1)

plot.ts(trainTSDiffs1)
```



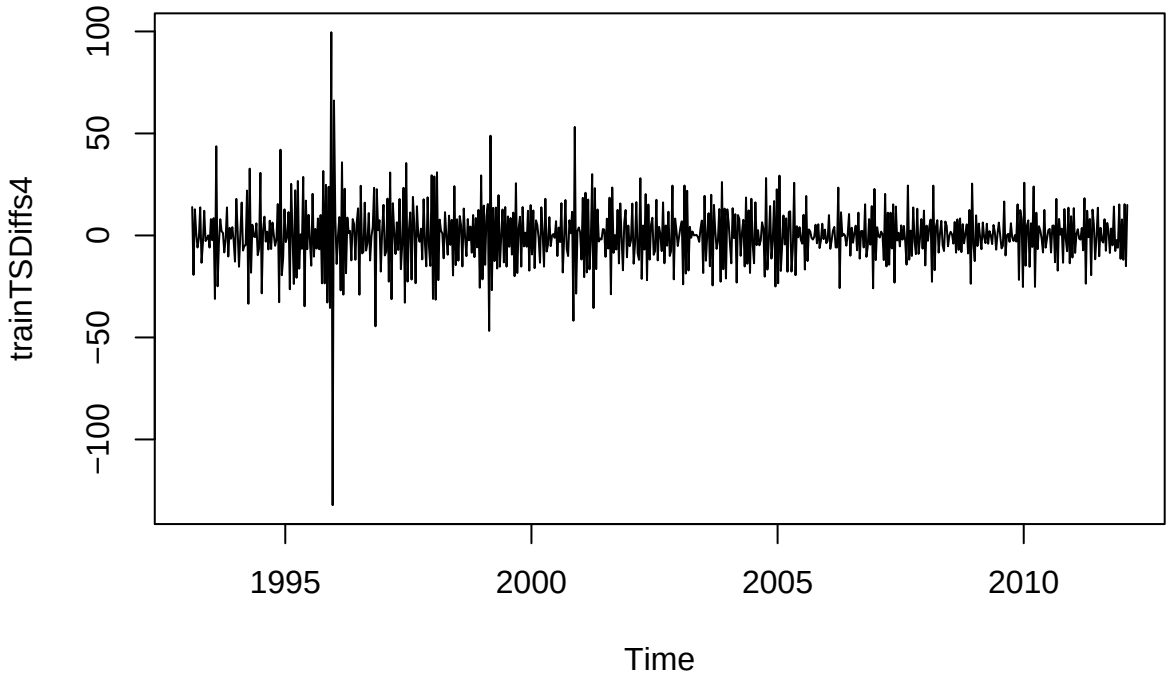

```
#Try 2:
trainTSDiffs2 <- diff(trainTS, differences = 2)
plot.ts(trainTSDiffs2)
```



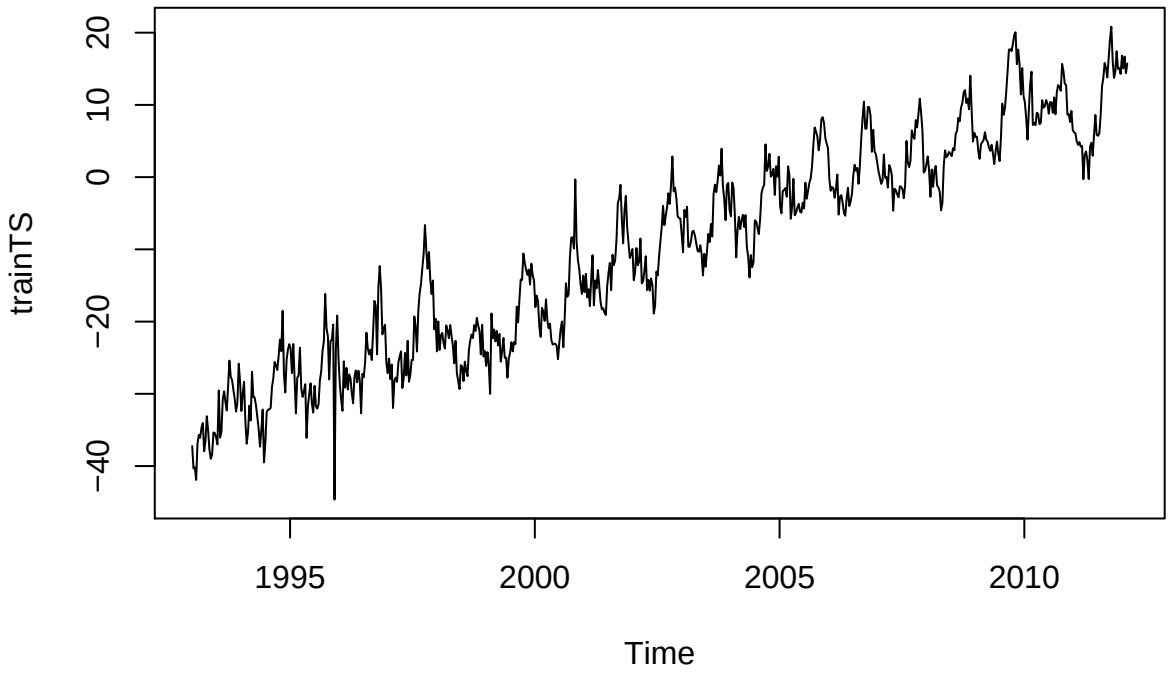

```
#Maybe for snicks - lets do 3
trainTSDiffs3 <- diff(trainTS, differences = 3)
plot.ts(trainTSDiffs3)
```




```
#lets do 4
trainTSDiffs4 <- diff(trainTS, differences = 4)
plot.ts(trainTSDiffs4)
```




```
plot.ts(trainTS)
```



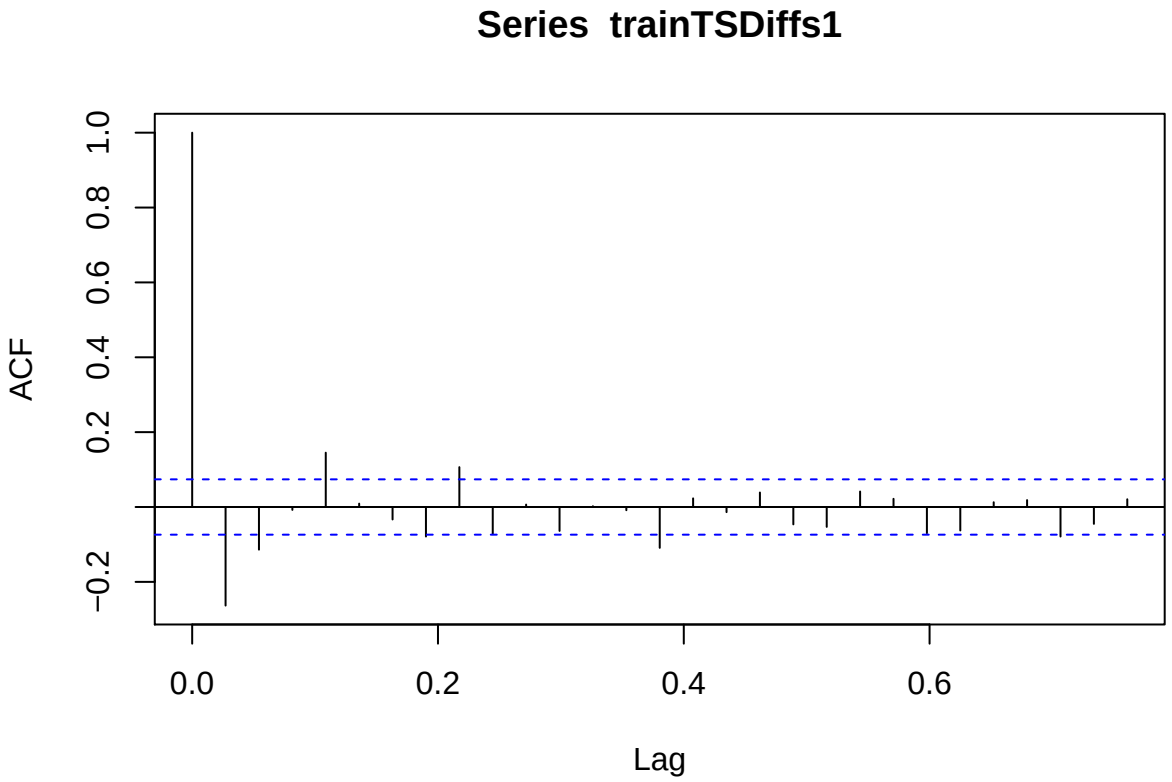

```
# Attempt at a log differencing.  Did not make improve the model.
trainTSDiffsLOG <- diff(log(trainTS + 50))
```

Differencing results

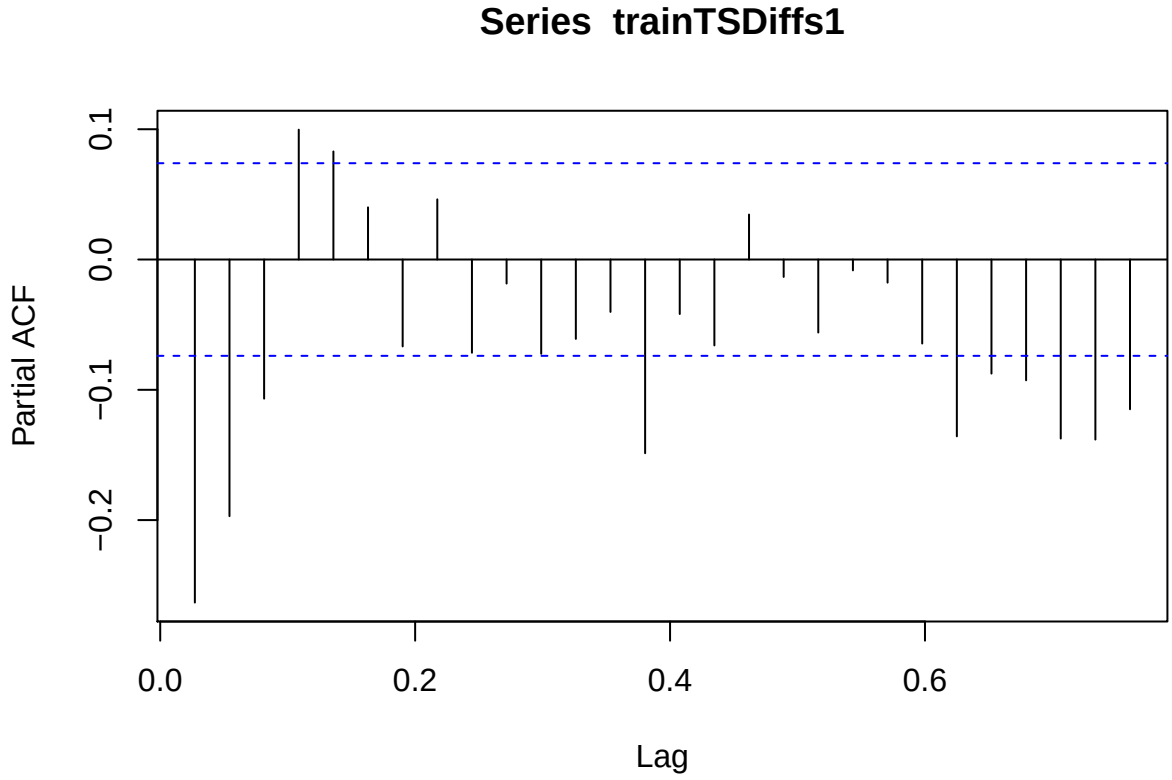
One difference seemed to smooth the seasonality of the SL data. However, when I began to have problems with the ARIMA model, I tried other levels of differencing.

Attempted to determine the ARIMA modeling numbers to use, by looking at the autocorrelation and partial autocorrelation of the differences data.

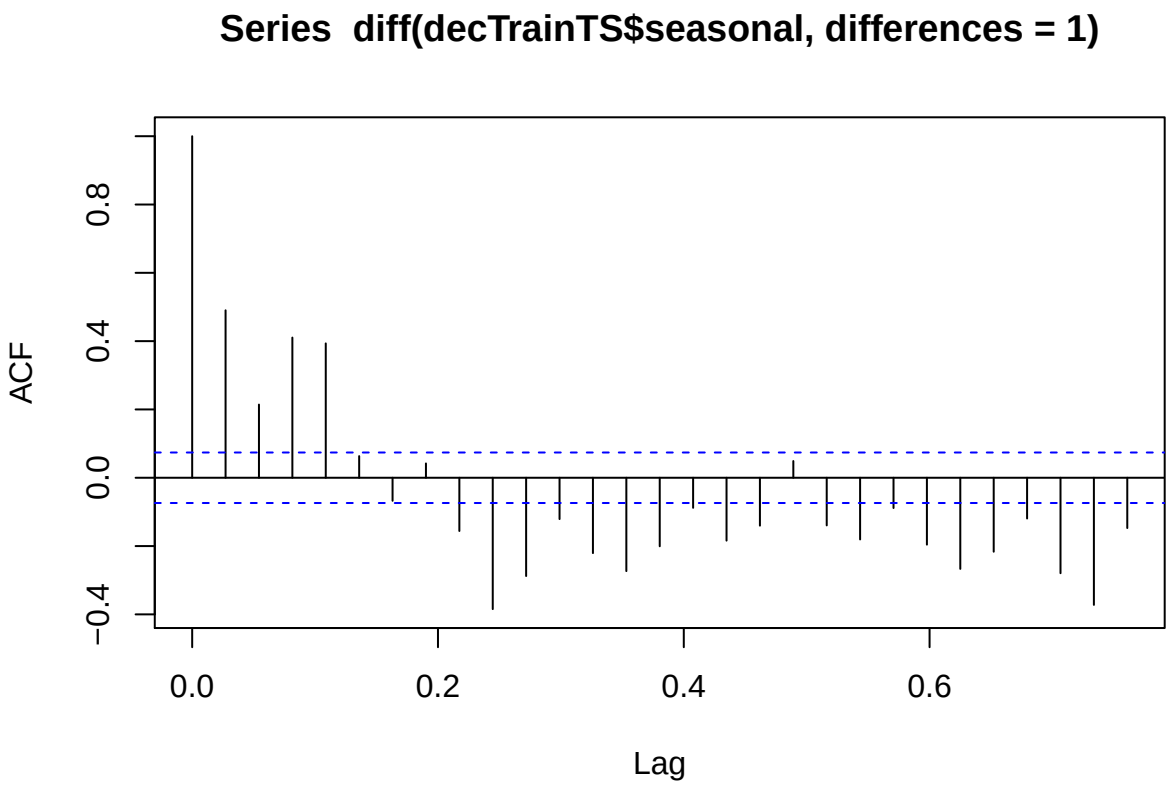
```
# Values of the AutoCorrelation
datCorr <- acf(trainTSDiffs1)
```



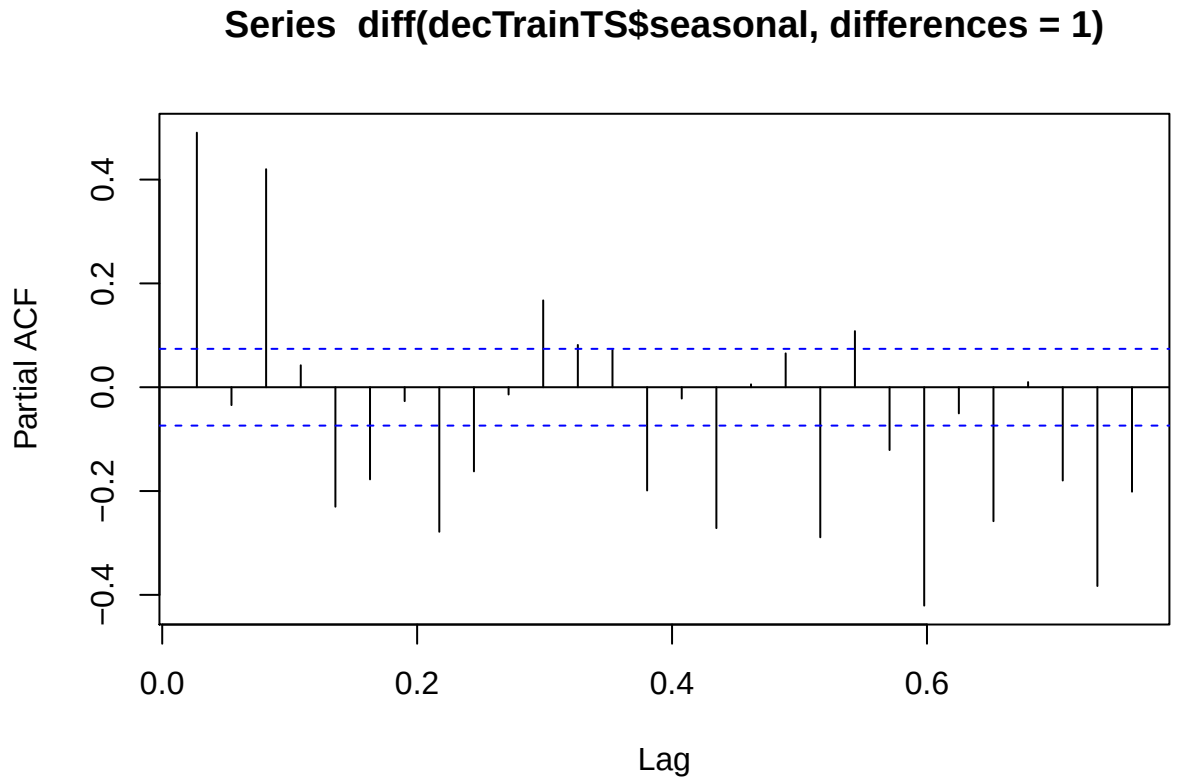

```
# datCorrRaw <- acf(datSL_TS)

# Values of partial AutoCorrelations
datCorrPartial <- pacf(trainTSDiffs1)
```



```r
# Check ACF and pacf of the seasonal component of the decomposed data

datSeasCorr <- acf(diff(decTrainTS$seasonal, differences = 1))
```



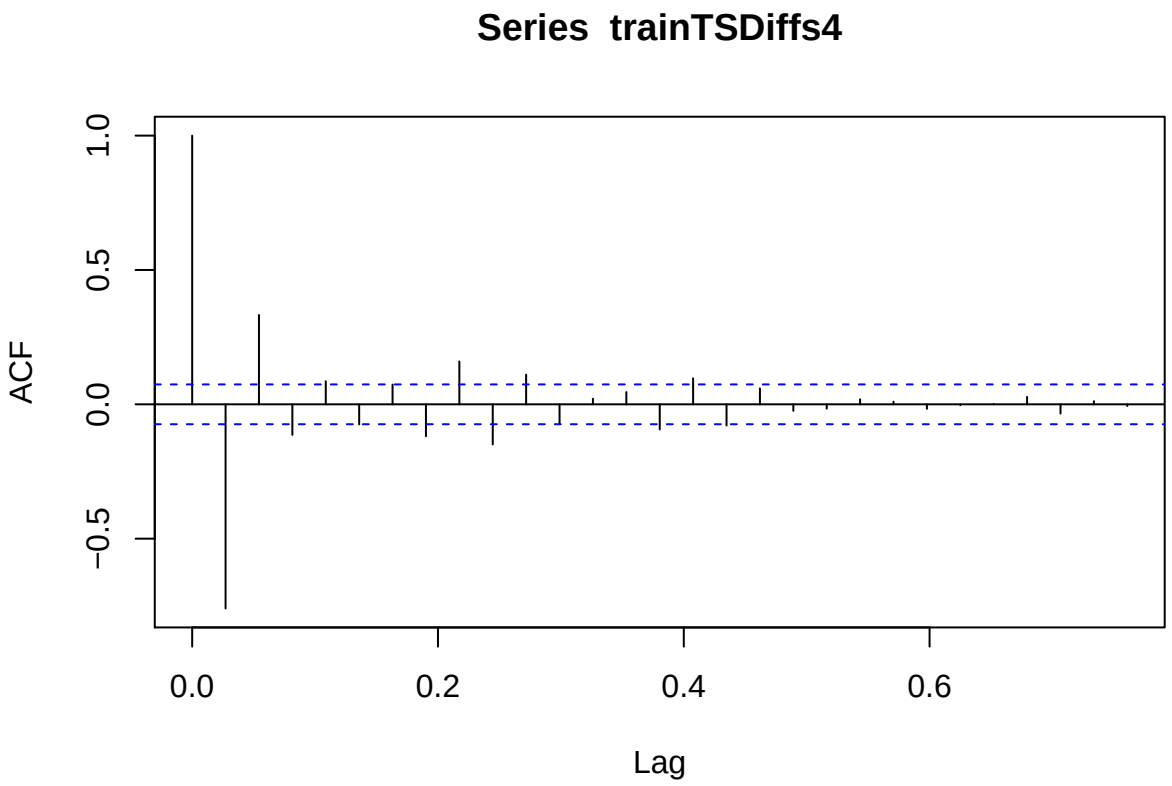

```r
datSeasCorrPartial <- pacf(diff(decTrainTS$seasonal, differences = 1))
```



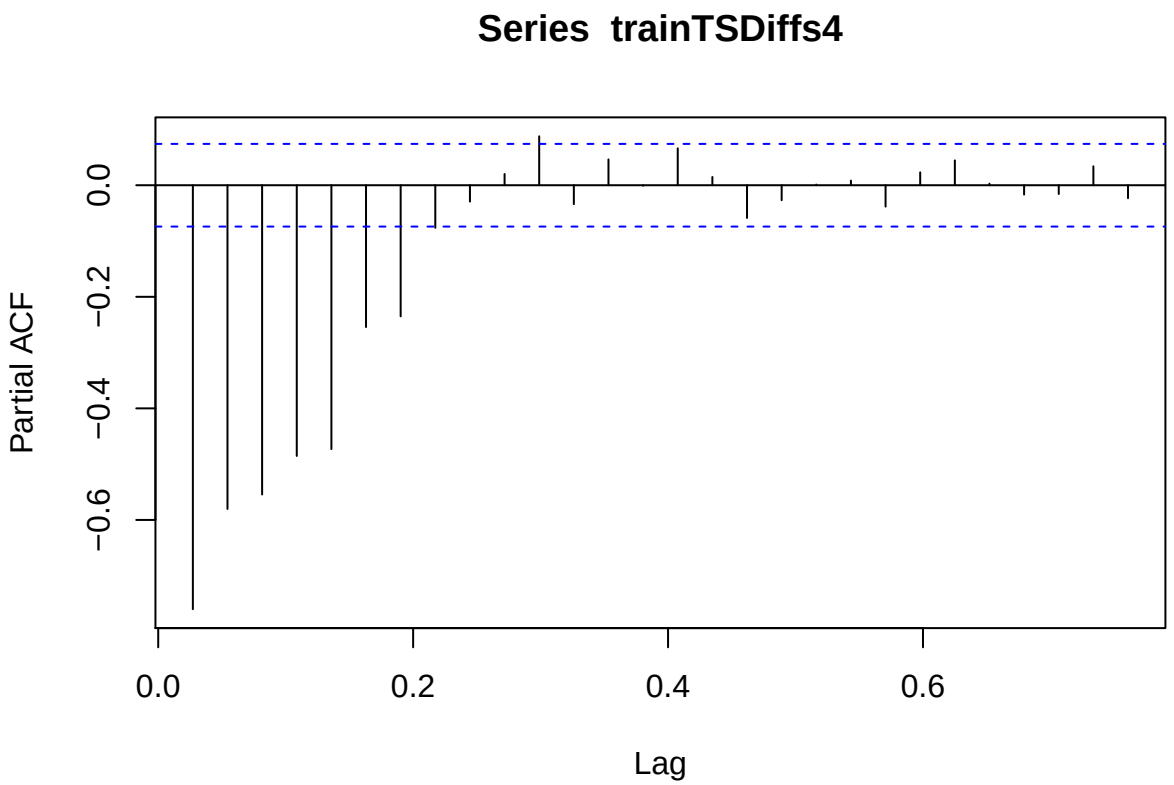
##

The ACF graphs above should show a trending to 0. The ACF does, but the PACF does not. So I tried different number of differences. For brevity, I will show d = 4:

```
# # Values of the AutoCorrelation
datCorr <- acf(trainTSDiffs4)
```



```
#
# # Values of partial AutoCorrelations
datCorrPartial <- pacf(trainTSDiffs4)
```



I attempted to estimate the values needed for Arima(p, d, q).

Arima(p,d,q): Arima is a non-seasonal predictive model. p is the number of autoregressive terms, d is the number of differences calculated, and q is the number of lagged forecast errors. One method to estimate these: for p, count the number of entries outside the confidence interval on the ACF graph before it trends to 0. We know d from above. For q, count the number of entries outside the confidence interval on the PACF graph before it trends to 0.

My manual method of doing all this was tedious, and netted me no good results.

And I discovered auto.arima, which programmatically attempts multiple p, d, q combinations, and also finds seasonal P, D, Q, m values to get much more accurate results by incorporating the seasonal components of SARIMA..

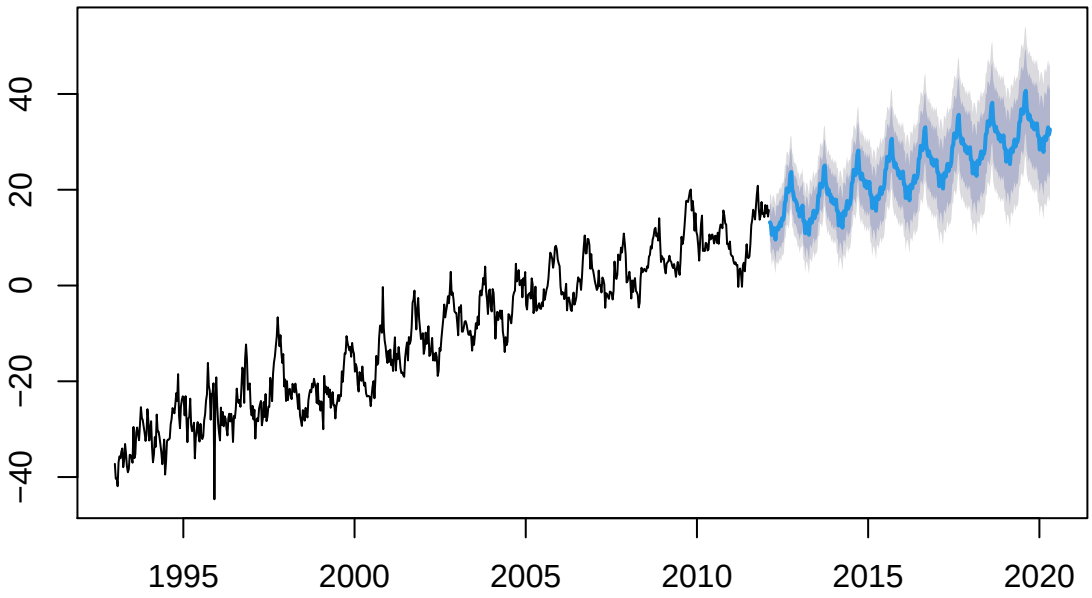
auto.arima

Took a while to determine the entries necessary for the best prediction for the testTS data.

```
#

SARIMA.model <- auto.arima(trainTS, allowdrift = TRUE, stationary = FALSE, seasonal = TRUE, method = "M
```

```
## Warning: The time series frequency has been rounded to support seasonal
## differencing.
```

```
plot(forecast(SARIMA.model, h = 302))
```




```
summary(SARIMA.model)
```

```
## Series: trainTS
## ARIMA(1,0,1)(2,1,0)[36] with drift
##
## Coefficients:
##          ar1      ma1     sar1     sar2   drift
##       0.9202  -0.5835  -0.5888  -0.2636  0.0700
## s.e.  0.0228   0.0481   0.0392   0.0419  0.0092
##
## sigma^2 estimated as 8.869:  log likelihood=-1681.27
## AIC=3374.54   AICc=3374.67   BIC=3401.57
##
## Training set error measures:
##                       ME     RMSE      MAE      MPE     MAPE      MASE
## Training set -0.01255144 2.888394 2.114557 29.33882 82.33086 0.5103135
##                     ACF1
## Training set 0.06926022
```

*** Auto-arima took 3.5 minutes to run

Below is the results from the auto.arima.

The time series frequency has been rounded to support seasonal differencing.Series: trainTS ARIMA(1,0,1)(2,1,0)[36] with drift

Coefficients: ar1 ma1 sar1 sar2 drift 0.9202 -0.5835 -0.5888 -0.2636 0.0700 s.e. 0.0228 0.0481 0.0392 0.0419 0.0092

sigma^2 estimated as 8.869: log likelihood=-1681.27 AIC=3374.54 AICc=3374.67 BIC=3401.57

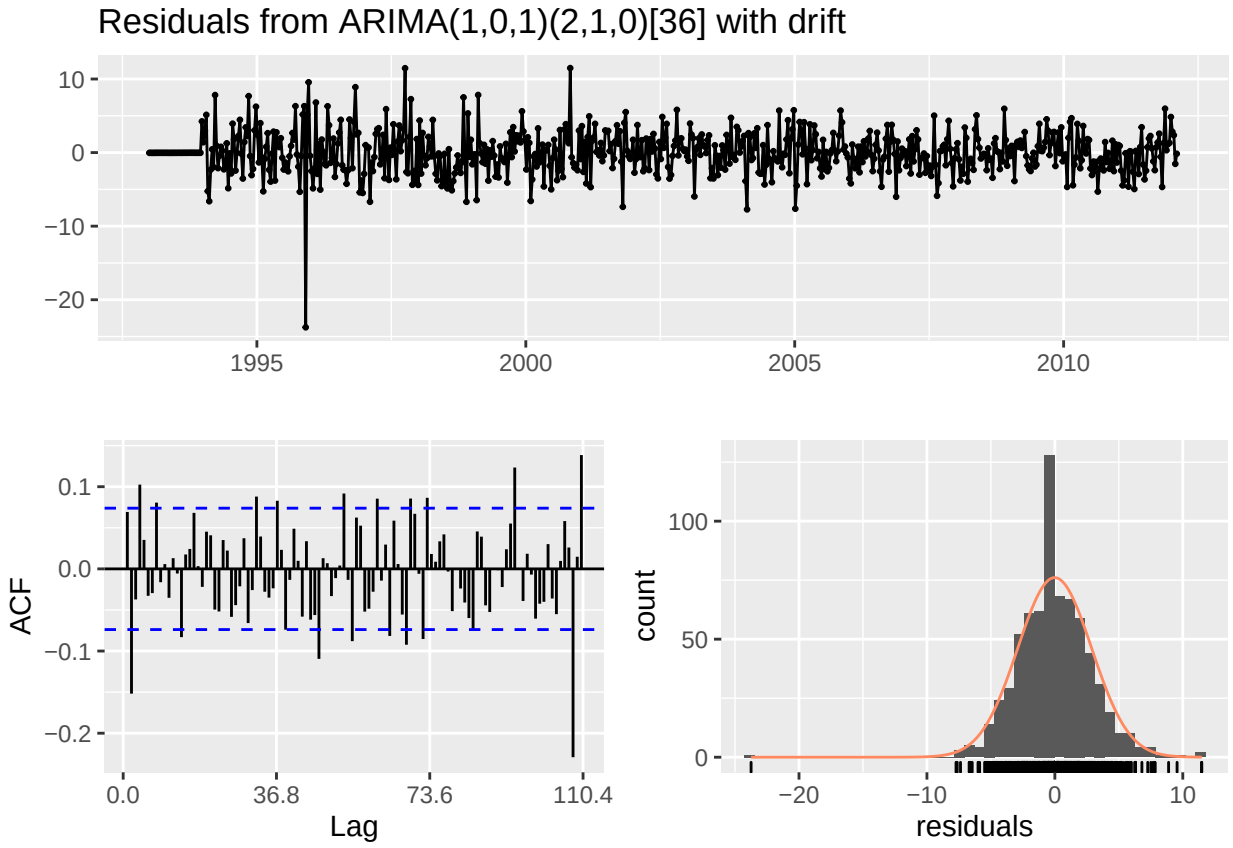
Based on these findings, I will run Arima.

Note: Many attempts were made using arima (lower-case a), as opposed to Arima (capital a). Arima seems to have the necessary parameters to make it work as expected.

```
# This one worked!!
arimaResults <- Arima(trainTS, order = c(1,0,1), seasonal = c(2,1,0), method = "ML",
                      include.drift = TRUE)
```

Check the residuals:

```
# checkresiduals(arimaResults)

checkresiduals(SARIMA.model)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(2,1,0)[36] with drift
## Q* = 167.96, df = 68.6, p-value = 2.632e-10
##
## Model df: 5.   Total lags used: 73.6
```

```
res <- resid(arimaResults)

Box.test(res, type="Ljung-Box")
```

```
##
##  Box-Ljung test
```

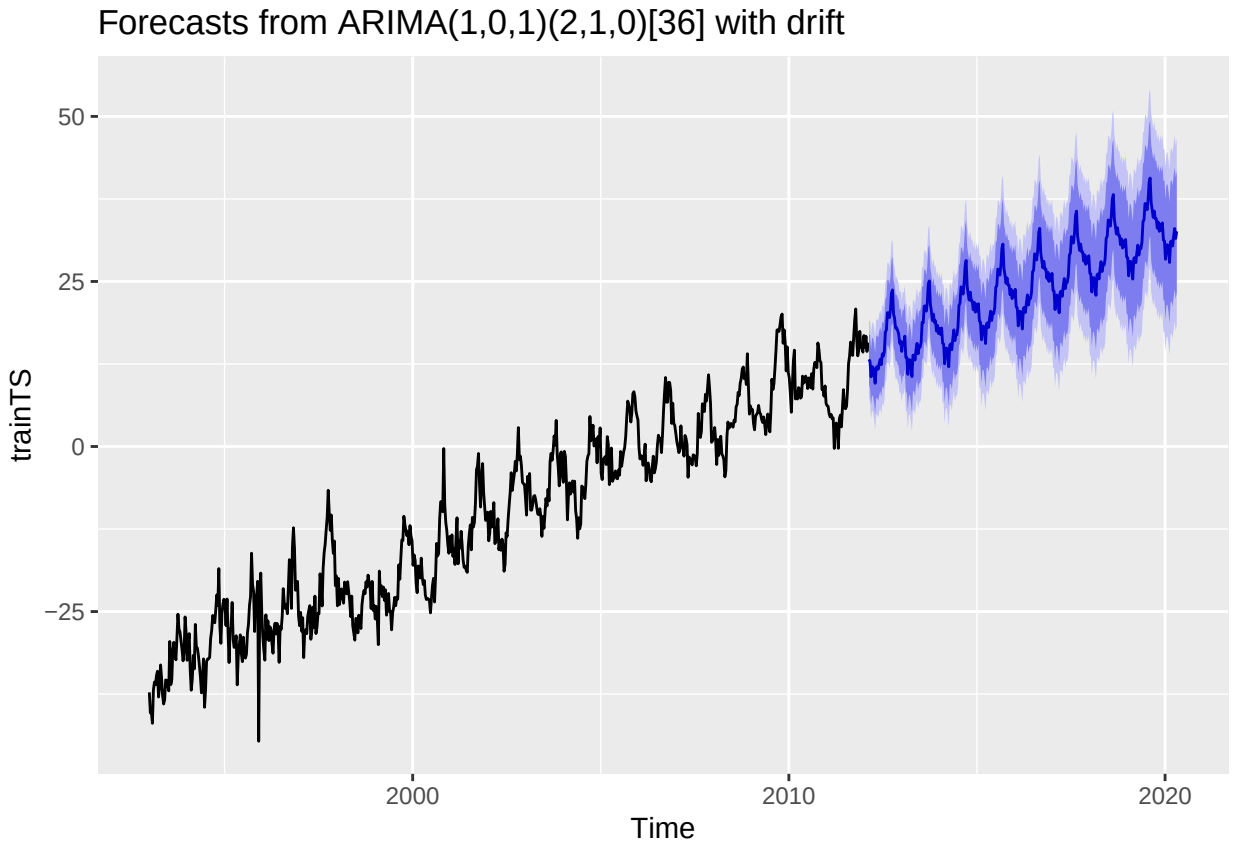
```
##
## data:  res
## X-squared = 3.3915, df = 1, p-value = 0.06553
```

```
# The null for Box.test is that our model DOES NOT show a lack of fit.
# I guess it is close.
```

I guess it is close, but we can't reject the null based on the Ljung-Box test.

See what a prediction looks like:

```
arimaResults %>% forecast(h = length(testTS)) %>% autoplot()
```




```
# ** Tried all the following things before finding the right mix for auto.arima above **
#--------------------------------------

# #arimaresults2 %>% forecast(h = 302) %>% autoplot()
#
# #accuracy(arimaresults2)
#
# # tbatsR <- tbats(trainTS)
#
# # tbatsR %>% forecast() %>% autoplot()
#
# pred <- predict(arimaResults, n.ahead = length(testTS))
#
# ts.plot(trainTS, pred$pred)
#
# ###Try something:
```

```
#
# dif <- diff(trainTS)
# fit <- auto.arima(dif)
#
# # The following shows an upward trend, but the seasonality vanishes.
# pred <- predict(fit, n.ahead = length(testTS))
# ts.plot(dif, pred$pred)
# trainTemp_pred<-trainTS[length(trainTS)]
# for (i in 1:length(pred$pred)) {
#   trainTemp_pred[i+1]<-trainTemp_pred[i]+pred$pred[i]
# }
# plot(c(trainTS, trainTemp_pred), type='l')
#
# tsdisplay(residuals(arimaResults))
```

##Now let's see how the results stack up.

I tried a number of prediction analysis measures.

```
testForecast <- forecast(arimaResults, h = length(testTS))

# Had to install Library "Metrics" for mse:

#testForecast$mean
#testTS

# Mean Square Error
cat("Mean Square Error: ", mse(testTS, testForecast$mean), "\n")
```

```
## Mean Square Error:  157.4686
```

```
# In general, for Mean Square Error, the smaller the better.  The squaring
# can heavily weight larger differences.  May not be a good test for TS data.

# Mean Absolute Error
cat("Mean Absolute Error: ", mae(testTS, testForecast$mean), "\n")
```

```
## Mean Absolute Error:  10.99024
```

```
# Root Mean Square Error
cat("Root Mean Square Error: ", rmse(testTS, testForecast$mean), "\n")
```

```
## Root Mean Square Error:  12.54865
```

```
# Relative Squared Error
cat("Relative Squared Error: ", rse(testTS, testForecast$mean), "\n")
```

```
## Relative Squared Error:  1.284211
```

```
# Root Relative Squared Error
cat("Root Relative Squared Error: ", rrse(testTS, testForecast$mean), "\n")
```

```
## Root Relative Squared Error:  1.13323
```

```
# R-Squared
cat("R-Squared test: ", cor(testTS, testForecast$mean)^2, "\n")
```

```
## R-Squared test:  0.7589007
```

```r
# Mean Absolute Percentage Error
cat("Mean Absolute Percentage Error: ", mape(testTS, testForecast$mean), "\n")
```

```
## Mean Absolute Percentage Error:  0.2961975
```

```r
# Symmetric Mean Absolute Percentage Error
cat("Symmetric Mean Absolute Percentage Error: ", smape(testTS, testForecast$mean), "\n")
```

```
## Symmetric Mean Absolute Percentage Error:  0.3583701
```

Websites:

https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html#decomposing-time-series

https://otexts.com/fpp2/

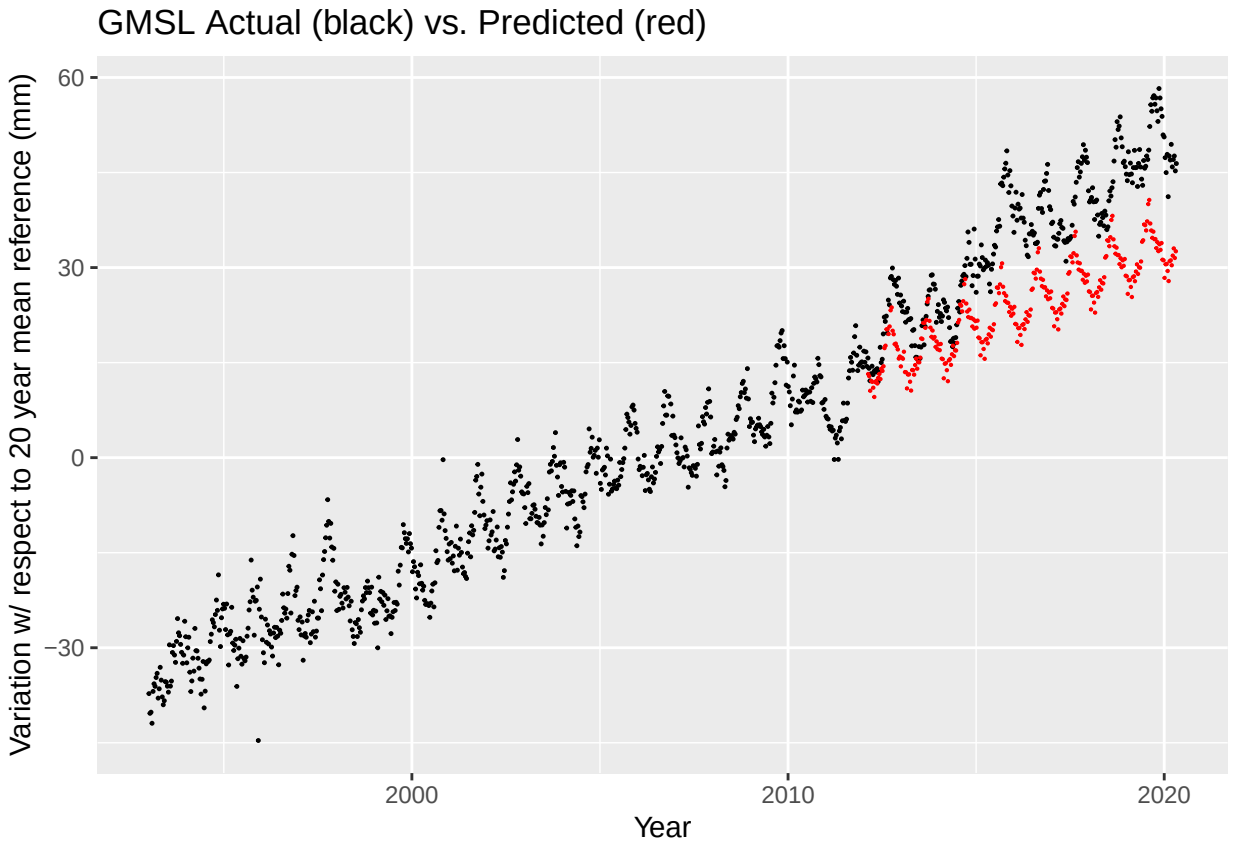
Try to get an overlay to work:

```r
# gP <- ggplot(datProj, aes(x=V3, y=V6))
# gP <- gP + geom_point(size = .2, color="blue") + ggtitle("Global Mean Sea Level Data - NASA, 1993 to
# gP <- gP + labs(x="Year", y="Variation w/ respect to 20 year mean reference (mm)")
# gP

foreDF <- testForecast$mean
foreDF
```

```
## Time Series:
## Start = 2012.13043478261
## End = 2020.3097826087
## Frequency = 36.8
##   [1] 13.199831 12.860314 10.560798 12.109585 11.963001 11.041760  9.577827
##   [8] 11.920923 12.131883 11.705764 12.619498 13.219857 12.404157 14.034515
##  [15] 13.691306 14.423902 17.289934 17.673921 20.270622 20.249456 19.522196
##  [22] 20.706962 23.258076 23.710974 20.029846 19.474130 17.917970 18.001450
##  [29] 17.557889 17.083597 15.636261 15.968860 14.405485 15.764311 15.610543
##  [36] 16.718432 13.506683 13.481810 10.931050 13.113626 13.157900 11.985560
##  [43] 10.589143 13.831029 13.810380 13.097281 14.623892 15.633571 14.044072
##  [50] 15.083256 15.050114 15.785894 18.813133 18.728607 21.271974 21.279766
##  [57] 20.492167 21.730074 24.599319 25.082843 21.592671 20.522858 19.072130
##  [64] 20.063690 18.782220 19.042763 17.478225 18.344033 17.104170 17.895695
##  [71] 16.959844 17.947465 15.631933 15.542346 12.527422 14.818981 14.912944
##  [78] 13.804033 12.070711 15.348889 15.551012 14.647135 16.322939 17.668462
##  [85] 16.027179 16.945006 16.914445 18.121888 21.342369 21.738068 24.179156
##  [92] 23.975049 23.082455 24.693584 27.383034 28.185021 24.321712 23.063174
##  [99] 22.184453 23.370415 22.050790 22.040519 20.707639 21.835955 20.353750
## [106] 21.555993 20.514933 21.631871 18.968264 18.833826 16.158384 18.218714
## [113] 18.233098 17.152999 15.600228 18.620140 18.752125 18.036195 19.462617
## [120] 20.502511 19.095644 20.240720 20.126910 21.055798 24.120006 24.356428
## [127] 26.871786 26.784804 25.969925 27.347297 30.058757 30.665163 26.971219
## [134] 25.959287 24.715988 25.548158 24.471829 24.427338 22.988947 23.822172
## [141] 22.397388 23.507320 22.734325 23.806819 21.111949 21.020973 18.267981
## [148] 20.435730 20.483872 19.370091 17.799807 20.962216 21.076783 20.300473
## [155] 21.834406 22.965735 21.434495 22.477754 22.412280 23.380866 26.486155
## [162] 26.689791 29.188376 29.088279 28.255315 29.671954 32.417841 33.055468
## [169] 29.360147 28.252757 27.073360 28.062601 26.853079 26.900110 25.462788
```

```
## [176] 26.400572 25.005850 26.061886 25.158788 26.223367 23.638641 23.533896
## [183] 20.737092 22.902536 22.951775 21.850232 20.242674 23.389250 23.532558
## [190] 22.742264 24.278622 25.436666 23.916873 24.960188 24.888195 25.906822
## [197] 29.029110 29.294027 31.782914 31.659670 30.816872 32.271999 34.991815
## [204] 35.662606 31.923459 30.807279 29.686344 30.676347 29.481125 29.483288
## [211] 28.073145 29.027139 27.599583 28.711681 27.814536 28.895489 26.254151
## [218] 26.146058 23.395489 25.533979 25.573675 24.473804 22.892808 26.011150
## [225] 26.142127 25.375980 26.882575 28.000791 26.507037 27.577154 27.496260
## [232] 28.474961 31.576413 31.813890 34.312908 34.206749 33.374507 34.796624
## [239] 37.522717 38.165753 34.452773 33.366927 32.194729 33.142886 31.974349
## [246] 31.978808 30.552381 31.469273 30.053127 31.146421 30.280060 31.353458
## [253] 28.716424 28.613931 25.847686 28.002654 28.047679 26.943598 25.356787
## [260] 28.495926 28.626588 27.849909 29.373389 30.508014 28.995912 30.050234
## [267] 29.976298 30.965319 34.074558 34.312041 36.807650 36.697532 35.861666
## [274] 37.293075 40.022344 40.672981 36.956145 35.854755 34.697332 35.669926
## [281] 34.481909 34.496842 33.072839 34.007304 32.593093 33.682684 32.796628
## [288] 33.870159 31.245511 31.140603 28.371403 30.523773 30.568175 29.466132
## [295] 27.875744 31.010080 31.144177 30.367335 31.888718 33.024178 31.516017
## [302] 32.572574
```

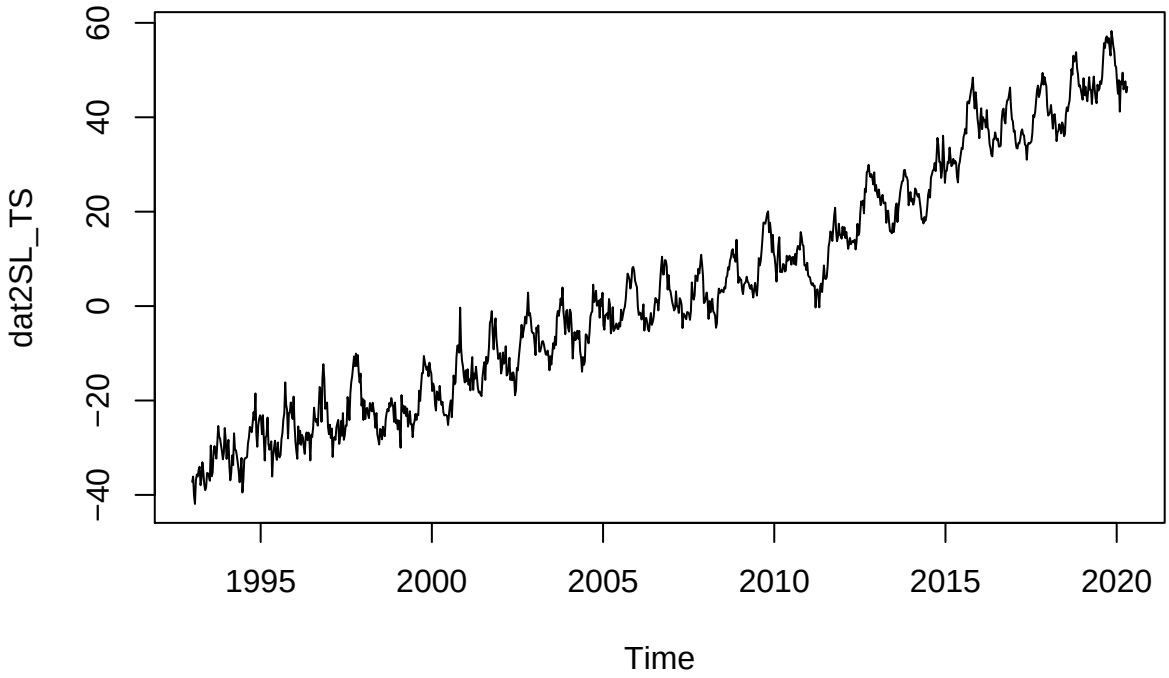
```
gP <- ggplot(datSL, aes(x=Yr, y=SL))
gP <- gP + geom_point(size = .2) + ggtitle("GMSL Actual (black) vs. Predicted (red)")
gP <- gP + labs(x="Year", y="Variation w/ respect to 20 year mean reference (mm)")


gP = gP + geom_point(data = foreDF, aes(x=x, y=y), color="red", size = .1)
gP
```



Overlay graph worked!

Down the rabbit hole of whether or not to remove the outlier.

```
#Just remove it first:

dat2SL_TS <-  tsclean(datSL_TS)

plot(dat2SL_TS)
```




```
outlier(dat2SL_TS)
```

```
## [1] 58.26
```

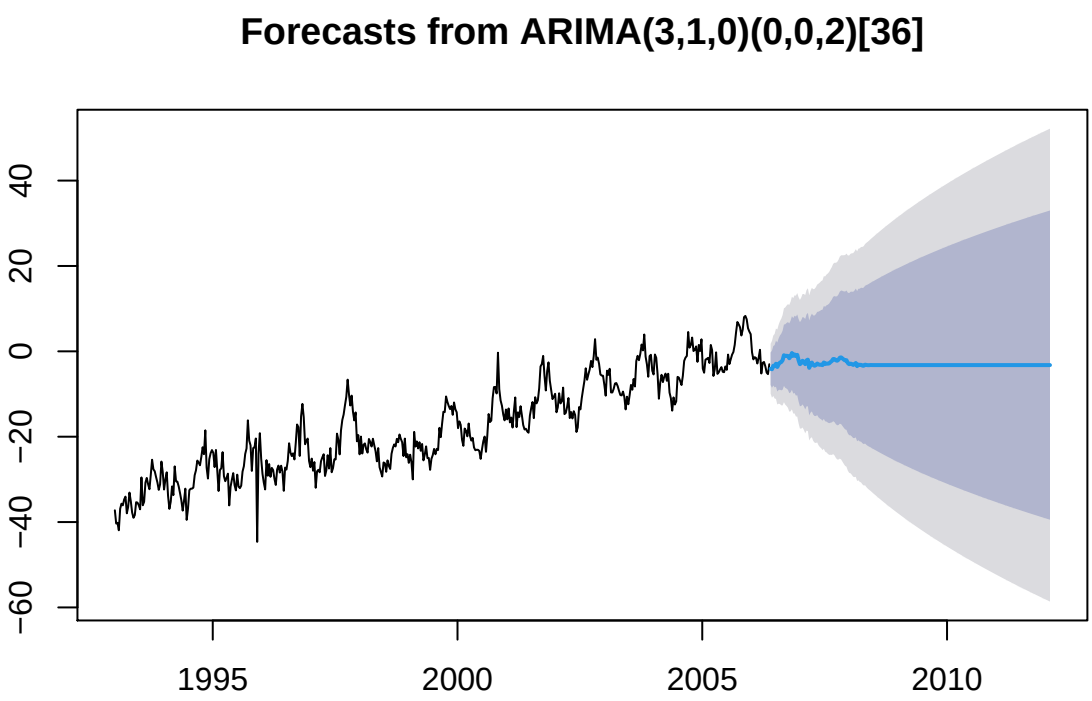
```
chisq.out.test(dat2SL_TS, variance=var(trainTS), opposite = FALSE)
```

```
##
##  chi-squared test for outlier
##
## data:  dat2SL_TS
## X-squared = 13.492, p-value = 0.0002396
## alternative hypothesis: highest value 58.26 is an outlier
```

```
train2TS <- head(dat2SL_TS, round(length(dat2SL_TS) * 0.7))
x <- length(dat2SL_TS) - length(trainTS)
test2TS <- tail(dat2SL_TS, x)

# auto.arima(train2TS, allowdrift = TRUE, stationary = FALSE, seasonal = TRUE, method = "ML")
# Takes a long time to run, so commented out.

# Results in the same settings as before the outlier removal.
```

```
# Test of the prediction if I split the original trainTS into 70%/30% to see how the prediction goes.

# trainTS <- head(datSL_TS, round(length(datSL_TS) * 0.7))
```

```r
# x <- length(datSL_TS) - length(trainTS)
# testTS <- tail(datSL_TS, x)

train2 <- head(trainTS, round(length(trainTS) * 0.7))
x2 <- length(trainTS) - length(train2)
test2 <- tail(trainTS, x2)

SARIMA2.model <- auto.arima(train2, allowdrift = TRUE, stationary = FALSE, seasonal = TRUE, method = "M

plot(forecast(SARIMA2.model, h = 211))
```




```r
summary(SARIMA2.model)
```

```
## Series: train2
## ARIMA(3,1,0)(0,0,2)[36]
##
## Coefficients:
##           ar1      ar2      ar3    sma1    sma2
##       -0.4584  -0.3379  -0.1695  0.1936  0.1645
## s.e.   0.0460   0.0482   0.0456  0.0463  0.0488
##
## sigma^2 estimated as 8.893:  log likelihood=-1234.85
## AIC=2481.7   AICc=2481.87   BIC=2506.89
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.09886555 2.963932 2.202987 -14.29377 49.06782 0.5235231
##                      ACF1
## Training set 0.005282316
```

```r
# With Time I would continue to play with this to experiment.  But right now, time is too precious.
```