Forecasting Monthly German Electricity Consumption with SARIMA Models

Kelly Pham (Perm: 5004064)

December 12, 2020

Abstract

Germany is one of the world's largest consumer of electricity, and we seek to analyze and predict its consumption patterns. We will perform time series analysis on a German electricity consumption dataset from Kaggle to forecast monthly average electricity consumption in the year 2017.

In particular, we first convert time series from daily to monthy. We split our dataset into train and test sets. Then, we normalize our training data by applying a square-root transformation. We then twice-difference the data to remove seasonality and trend. We select preliminary SARIMA$(p, d, q) \times (P, D, Q)_s$ models based on ACF and PACF plots of our differenced time series and select 3 models to test based on their AICc. Analysis of residuals and diagnostic testing suggest that our transformed dataset follows a SARIMA$(0, 1, 1) \times (3, 1, 0)_{12}$ model. We use the 12-steps-ahead prediction to forecast and compare the values from the test dataset to the predicted values. The key result is that the chosen SARIMA$(0, 1, 1) \times (3, 1, 0)_{12}$ model is a good fit for our dataset, since actual values are within confidence intervals. However, our dataset is not normal and that heavy-tailed models should be considered to improve our data.

Introduction

Germany is Europe's largest, and the world's 6th largest, consumer of electricity. Additionally, Germany is a country the people think of when the phrase "sustainability" is mentioned. Therefore, it would be interesting to study and predict German electricity consumption patterns in the year 2017.

To do this, we use the Germany electricity power for 2006-2017 dataset provided by the Open Power System Data (OPSD), which I found through kaggle.com. This dataset contains 5 columns:

- Date - Daily data from January 2006 to December 2017
- Consumption — Electricity consumption in gigawatt-hours (GWh)
- Wind — Wind power production in GWh
- Solar — Solar power production in GWh
- Wind+Solar — Sum of wind and solar power production in GWh.

We use the Date and Consumption columns of this dataset and convert the data from a daily to a monthly time series by taking monthly averages. After that, we split the dataset into a train set consisting of data from 2006-2016 and a test set of data from 2017. The test set will be used to compare the predicted values.

We then analyze the plot and histogram of the training data to determine if a transformation is needed to normalize the data. After examination of histograms of various transforms, we choose to apply a square root

transform. The plot and ACF/PACF graphs of transformed data suggest a positive trend and seasonality at lag 12. So, we twice-differenced the data to make it stationary. From analysis of ACF and PACF plots of the now stationary series, we identify values p, q, P, Q for our preliminary SARIMA$(p, 1, q) \times (P, 1, Q)_{12}$ models. We then select three models to test by choosing models with the lowest AICc.

We then check if the chosen models are invertible and causal. All models have a unit root on the seasonal MA component; we then modify the models and select two, namely a SARIMA$(1, 1, 0) \times (3, 1, 0)_{12}$ and a SARIMA$(0, 1, 1) \times (3, 1, 0)_{12}$ for diagnostic testing. Both models passed all diagnostic tests except for the Shapiro-Wilk tests. The SARIMA$(0, 1, 1) \times (3, 1, 0)_{12}$ was selected as the final model based on AICc.

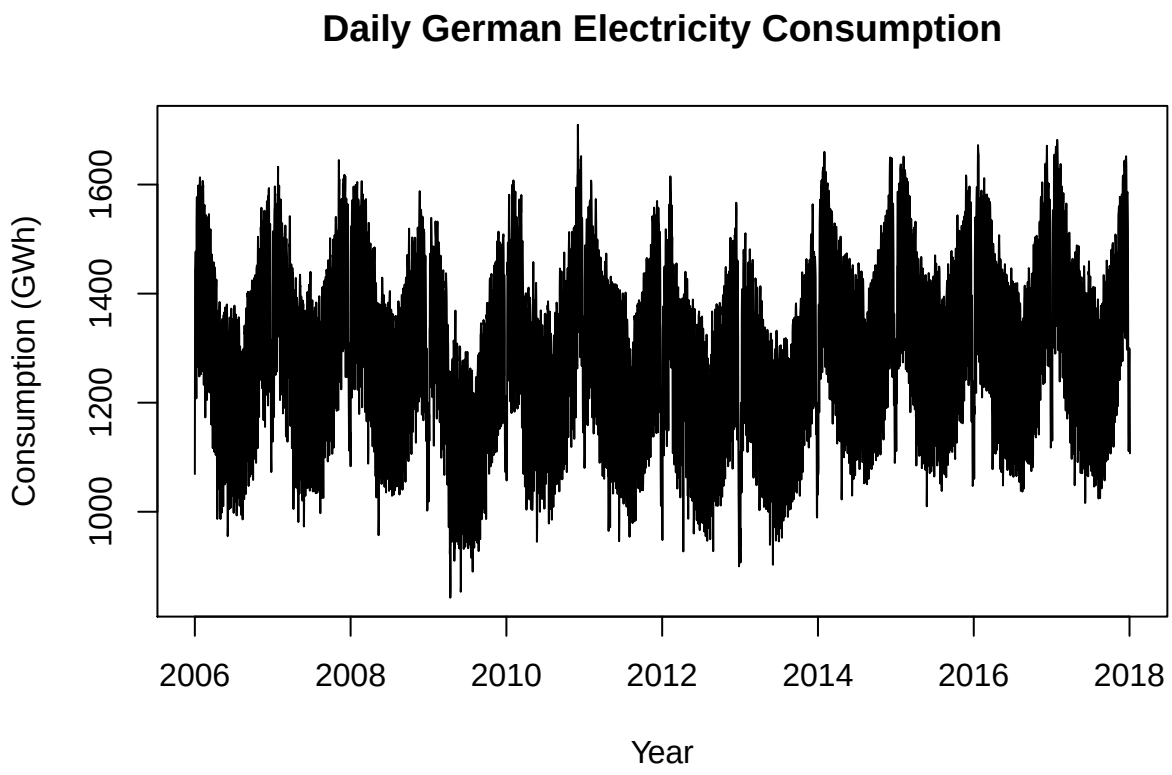
We apply a 12-steps-ahead predictor to this model and compared the predicted values to the test data. The predicted values were slight overestimates, but the actual values were within the prediction interval. Therefore, the chosen SARIMA$(0, 1, 1) \times (3, 1, 0)_{12}$ is a good model for forecasting. However, based on the Shapiro-Wilk test and Normal Q-Q plot results, the data is not normal, and heavy-tailed models should be used to improve the model.

This project was done using R software and data provided by the Open Power System Data (OPSD).

Methodology

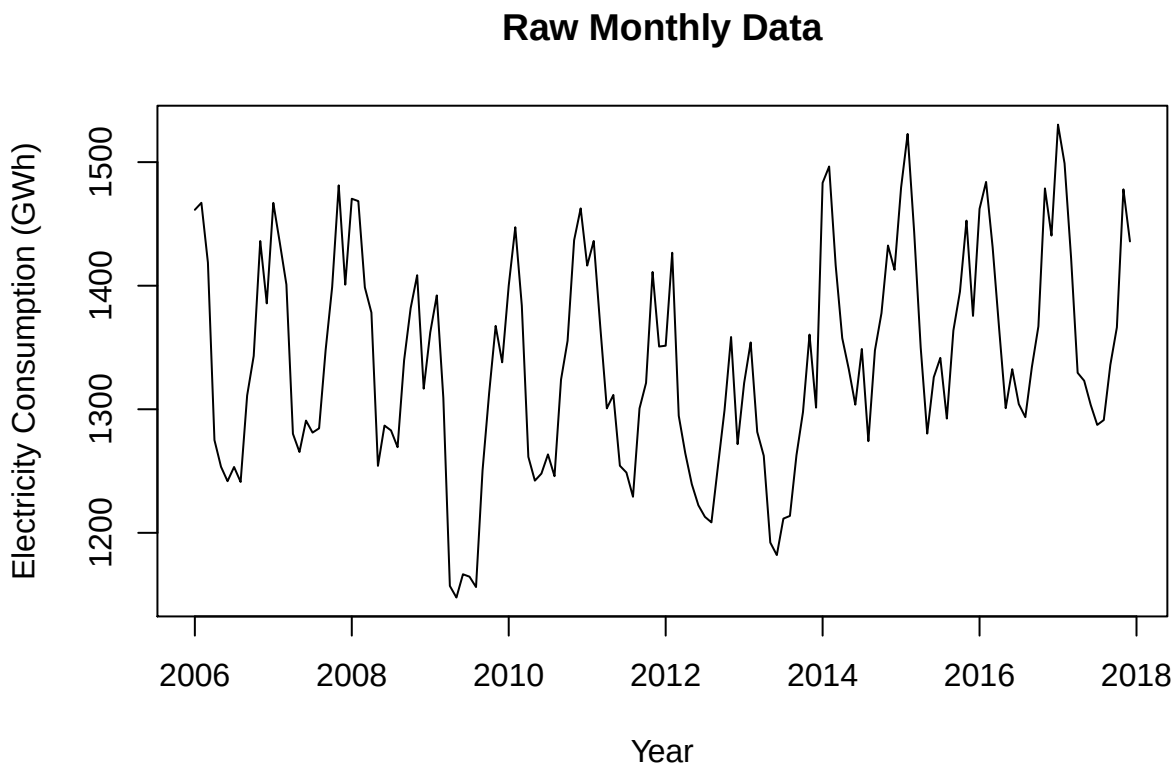
Loading the data

We load the Germany electricity power for 2006-2017 dataset in R and plot its graph:



The dataset is currently noisy. Because the goal is to forecast monthly data, we convert the time series from daily to monthly via monthly averages.

```
# Convert V1 (date column) into date format
econsumption.csv$V1 <- as.Date(econsumption.csv$V1, format="%Y-%m-%d")

# create variables for the Month of each observation:
econsumption.csv$Month <- as.Date(cut(econsumption.csv$V1,
  breaks = "month"))

# Loop for monthly average
avg <- vector()
for (mo in unique(econsumption.csv$Month))
{
  ss = subset(econsumption.csv, Month == mo)
  avg <- c(avg, mean(ss$V2))
}

# Create new dataframe of monthly averages
e_mo <- data.frame(Month = unique(econsumption.csv$Month), Consumption_Avg = avg)

# Graph Averaged Raw Data
e <- ts(e_mo[,2], start = c(2006,1,1), frequency = 12)
ts.plot(e, xlab = 'Year', ylab = 'Electricity Consumption (GWh)', main = "Raw Monthly Data")
```

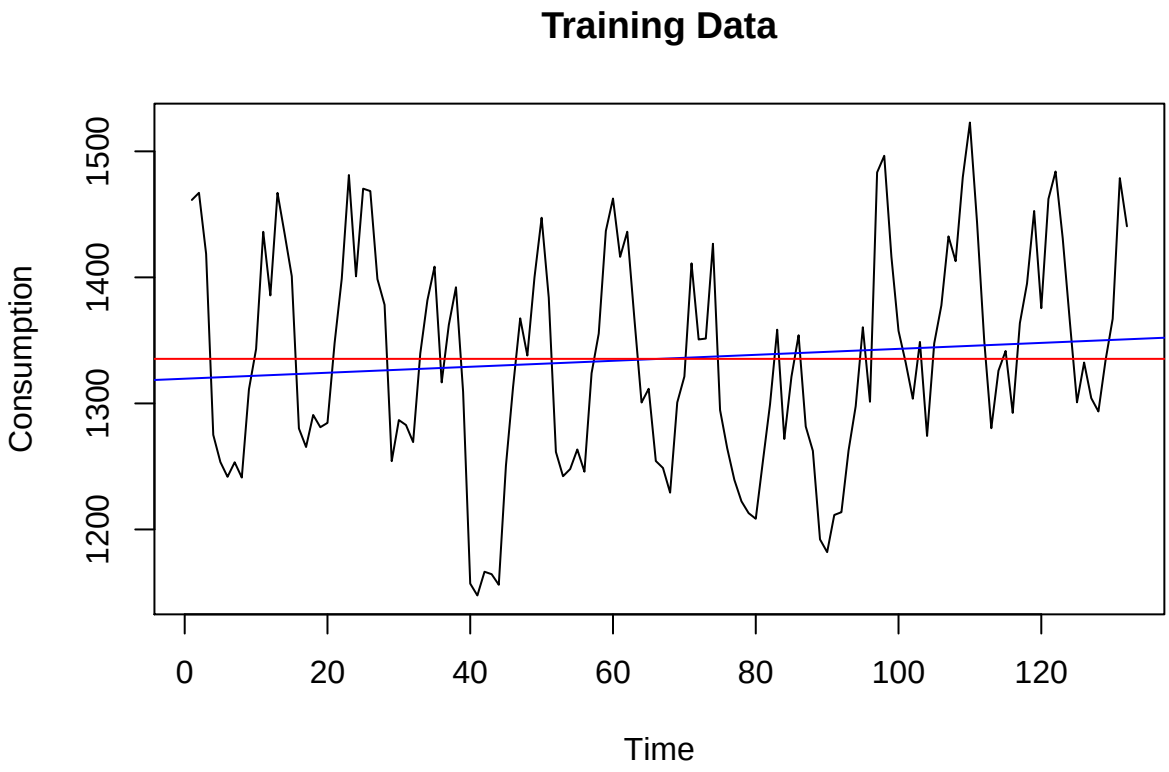


The plot of the converted time series is shown above:

Analysis of the plot shows the data is clearly seasonal, and there is a slight positive trend.

Preliminary Analysis and Model Identification

We split the raw data into training and test sets and begin preliminary analysis. Below is the plot of the training data, which we denote as U_t here and e_train in R code. The red line represents the (constant) mean, and the blue line is a trend line.



There is obvious periodicity and a positive trend. The graph seems to oscillate every year, suggesting seasonality at lag 12. There variance after time 90 seems different than the variance before time 90.

We further examine the behavior of the data by plotting a histogram.

The histogram forms a slightly right-skewed bell curve. To normalize the data, we seek to apply a Box-Cox transformation.

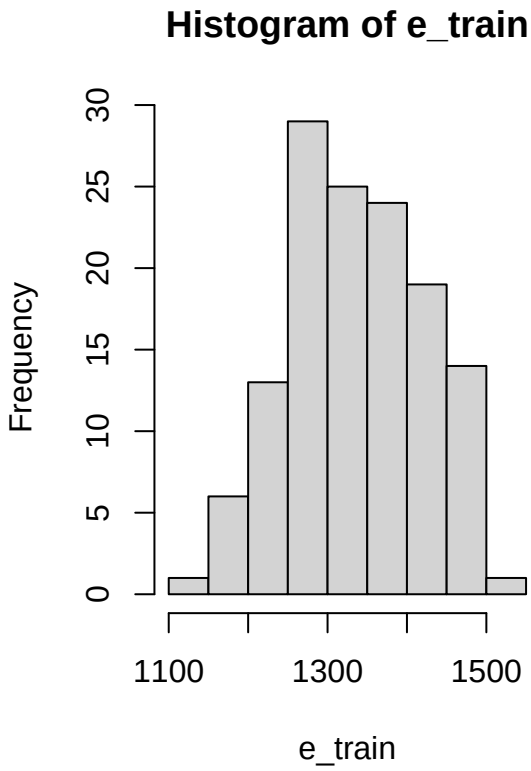
The Box-Cox transformation is given by:

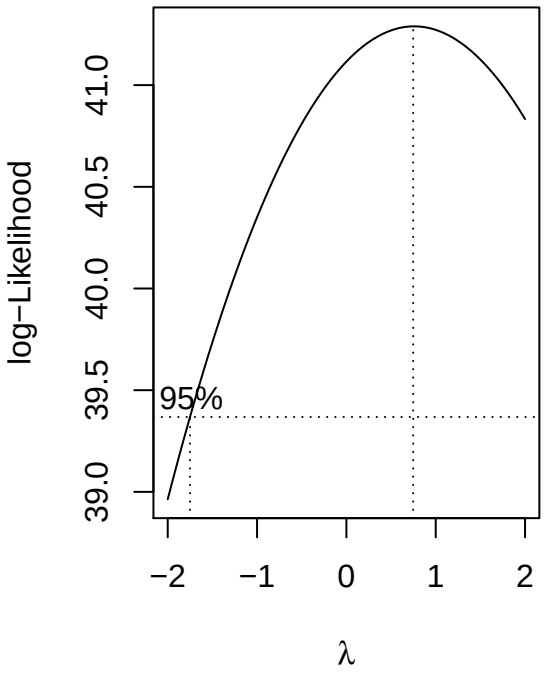
$$f_\lambda(U_t) = \ln(U_t), \text{ if } U_t > 0, \ \lambda = 0$$

and

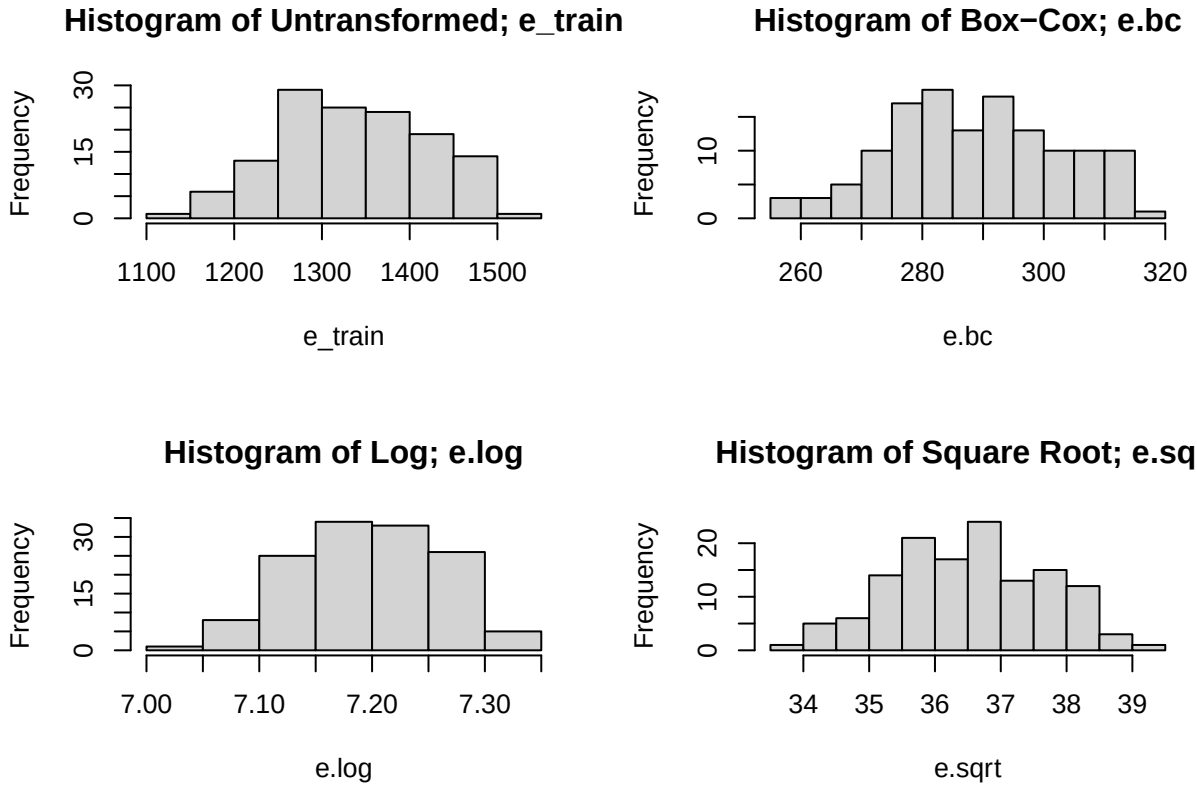
$$f_\lambda(U_t) = \lambda^{-1}(U_t^\lambda - 1), \text{ if } U_t \geq 0, \ \lambda \neq 0.$$

We determine optimal parameters of λ for a Box-Cox transformation by writing the following R code:





The confidence interval of λ is large and includes 0 and $\frac{1}{2}$, so we consider the transformations $\ln(U_t)$ and $\sqrt{U_t}$ as well. Histograms of all transformations are shown below:

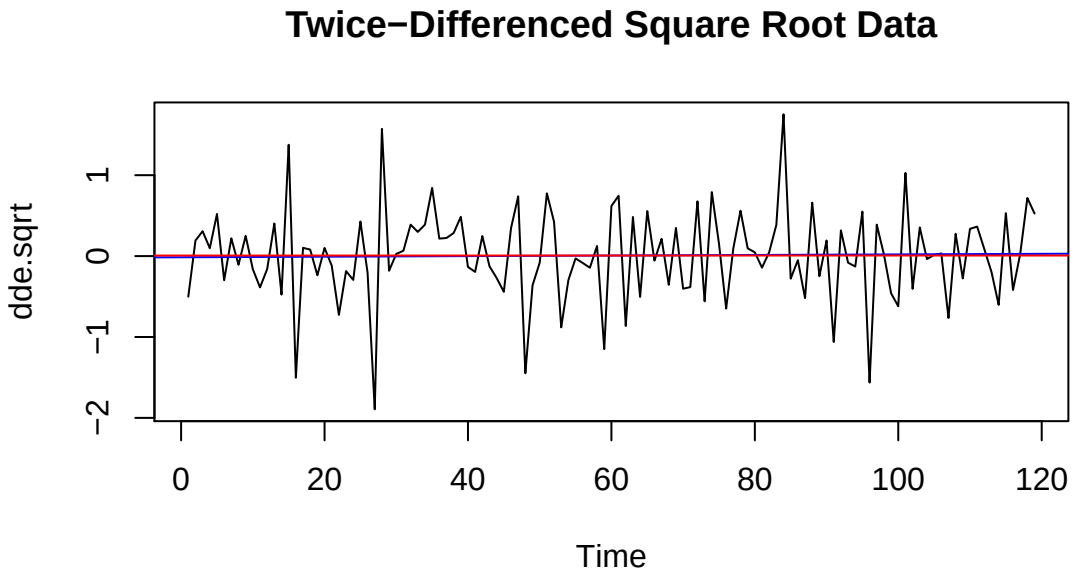


The histogram of $\sqrt{U_t}$ looks the most normal; additionally, square-root is easy to invert. Thus, we select this transformation before differencing.

We take two differences of $\sqrt{U_t}$: first a difference of lag 12 to de-seasonalize; then, a difference of lag 12 to de-trend. The R code for the differences and their standard deviations is shown below:

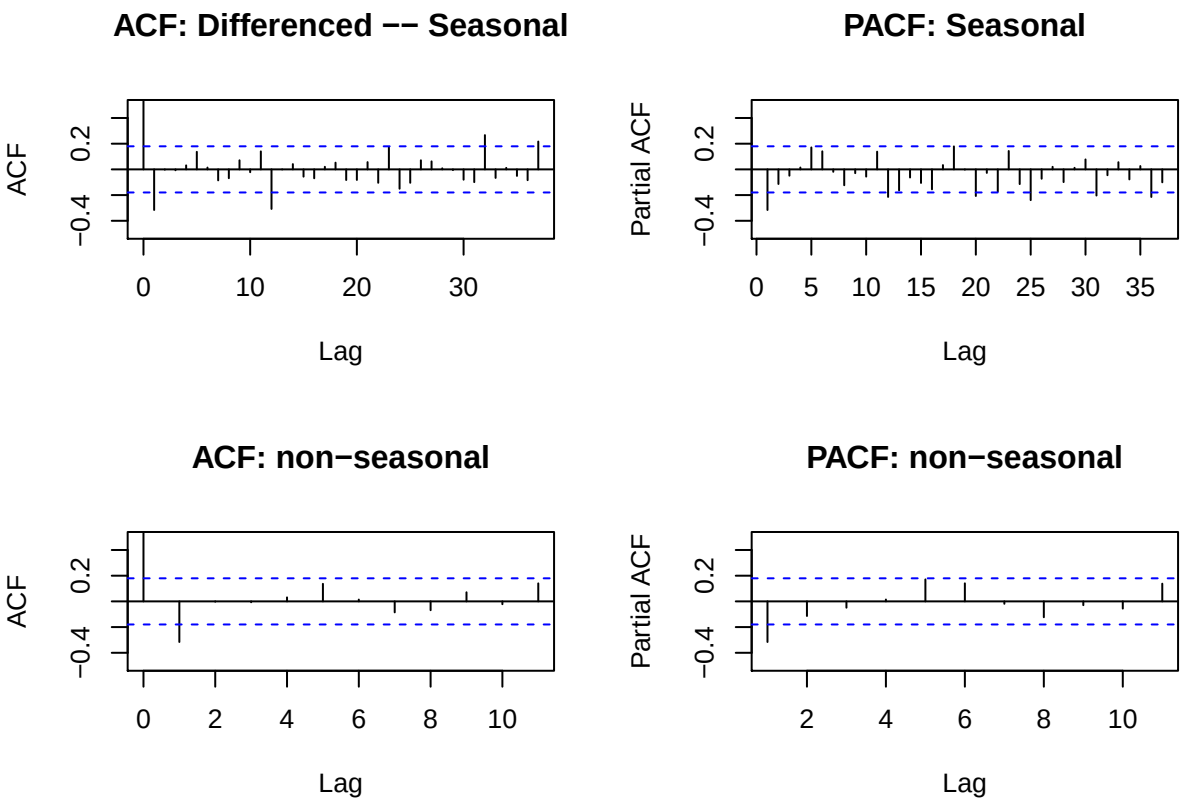
```
de.sqrt = diff(e.sqrt, lag=12, differences = 1) # difference at lag 12
dde.sqrt = diff(de.sqrt, lag=1, differences = 1) # then difference at lag 1

print(sd(e.sqrt));      #1.173245
print(sd(de.sqrt));     #0.8988376
print(sd(dde.sqrt));    #0.5723453
```

Since $\nabla_1\nabla_{12}\sqrt{U_t}$ has the lowest variance, we use that. Next, we analyze the plot of $\nabla_1\nabla_{12}\sqrt{U_t}$ to check if it resembles white noise:



The trend and mean lines of the plot coincide; thus, the series $\nabla_1\nabla_{12}\sqrt{U_t}$ does resemble white noise.

Now, we analyze the ACF and PACF of $\nabla_1\nabla_{12}\sqrt{U_t}$ to identify preliminary SARIMA$(p,1,q)\times(P,1,Q)_{12}$



The ACF of $\nabla_1\nabla_{12}\sqrt{U_t}$ has peaks at lags 0, 1, 12, and 32 as well as a slightly smaller peak at lag 37. So,

we try $Q = 1$ for our model. Since the PACF peaks at lags 1,12,20,25,31, and 36, values $P \in \{1, 3\}$ are good candidates for our model. Because the PACF cuts off after lag 36, the seasonal component of the model might be a pure AR model; hence, we also try $Q = 0$.

Now, we analyze the ACF and PACF of $\nabla_1 \nabla_{12} \sqrt{U_t}$ at lag h, $h = 1, ..., 12$ to determine p, q.

Both graphs peak at lag 1. Thus, we choose $p = q = 1$. There are a total of 8 models; we choose the 3 with the lowest AICc.

After looping through each value of P, Q, p, q to determine AICc, we test the following models:

- SARIMA$(0, 1, 1) \times (1, 1, 1)_{12}$ – AICc 1.207622
- SARIMA$(1, 1, 1) \times (1, 1, 1)_{12}$ – AICc 1.222933
- SARIMA$(1, 1, 0) \times (1, 1, 1)_{12}$ – AICc 1.220693

Model Fitting (I)

Now, we fit our SARIMA$(p, 1, q) \times (P, 1, Q)_{12}$ models onto our dataset U_t by estimating coefficients:

For the SARIMA$(0, 1, 1) \times (1, 1, 1)_{12}$ model, we estimate the coefficients by running the following R code:

```
fit1101sq <- sarima( xdata = e.sqrt,
        p = 0, d = 1, q = 1, #
        P = 1 , D = 1, Q = 1, S = 12,
        details = F); #AICc 1.207622

print(fit1101sq$ttable)
```

```
##      Estimate     SE t.value p.value
## ma1   -0.3852 0.0813 -4.7396    0.00
## sar1   0.1959 0.0989  1.9802    0.05
## sma1  -1.0000 0.1760 -5.6809    0.00
```

Unfortunately, there is a unit root on the seasonal MA term of our model. This suggests we may have overdifferenced by lag 12. We now compare the variance of $\nabla_1 \nabla_{12} \sqrt{U_t}$ and $\nabla_1 \sqrt{U_t}$:

```
te.sqrt <- diff(e.sqrt, lag = 1) # difference square-root data by lag 1

sd(te.sqrt) #std dev of data differenced by lag 1
# 0.8128187

sd(dde.sqrt) # std dev of data differenced by lag 12, then lag 1
# 0.5723453
```

Since $\text{sd}(\nabla_1 \nabla_{12} \sqrt{U_t}) < \text{sd}(\nabla_1 \sqrt{U_t})$, there was no over-differencing.

Now we test the other two models:

- SARIMA$(1, 1, 1) \times (1, 1, 1)_{12}$

```
##      Estimate     SE t.value p.value
## ar1   -0.0805 0.2166 -0.3715  0.7110
## ma1   -0.3190 0.2038 -1.5649  0.1204
## sar1   0.1951 0.0989  1.9732  0.0509
## sma1  -1.0000 0.1759 -5.6857  0.0000
```

- SARIMA$(1,1,0) \times (1,1,1)_{12}$

```
##      Estimate     SE t.value p.value
## ar1   -0.3517 0.0858 -4.1004  0.0001
## sar1   0.1990 0.0985  2.0201  0.0457
## sma1  -1.0000 0.1553 -6.4384  0.0000
```

All of these models have a unit root in the seasonal MA term and cannot be used.

Because the problem is the seasonal MA component, and there was no over-differencing, it is suggested to try AR models. Thus, we set $Q = 0$ and $P = 3$ and test these models:

- SARIMA$(1,1,0) \times (3,1,0)_{12}$ – AICc 1.282223
- SARIMA$(0,1,1) \times (3,1,0)_{12}$ – AICc 1.255449

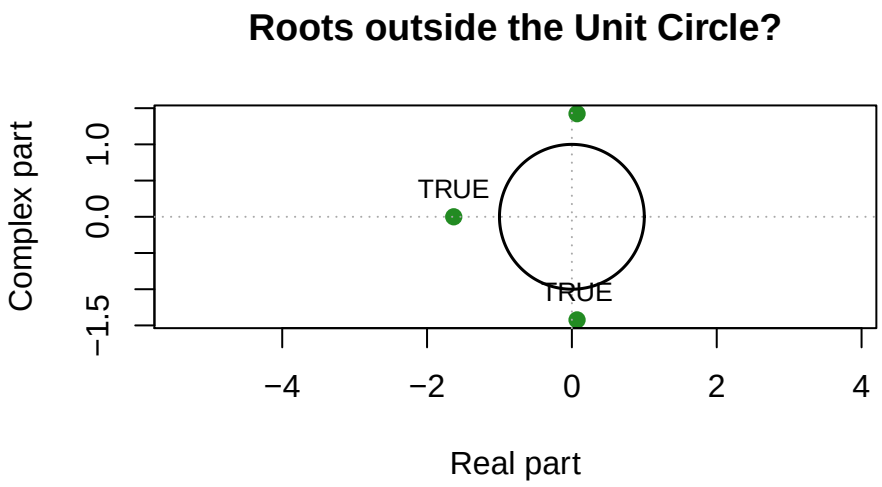
Model Fitting (II)

We start with the SARIMA$(1,1,0) \times (3,1,0)_{12}$ model: we estimate coefficients, and check if the model is causal invertible. To do that, we run the following R code:

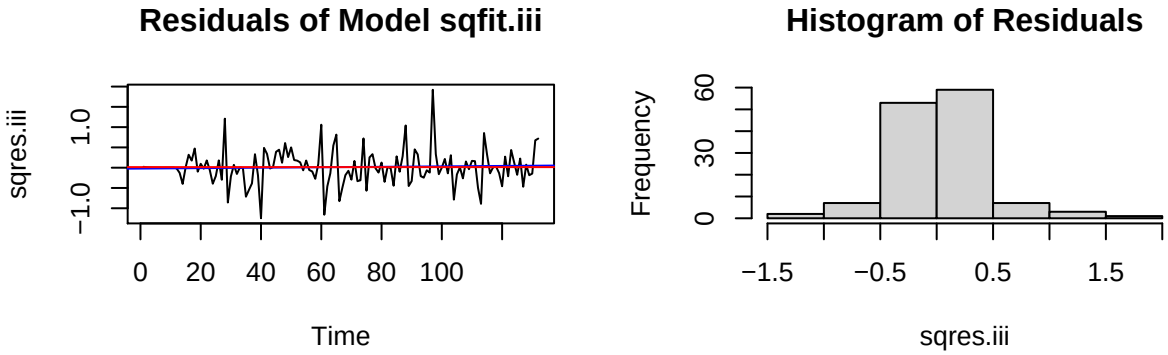
```
sqfit.iii <- sarima( xdata = e.sqrt,
        p = 1, d = 1, q = 0,
        P = 3 , D = 1, Q = 0, S = 12,
        details = F);

print(sqfit.iii$ttable)
```

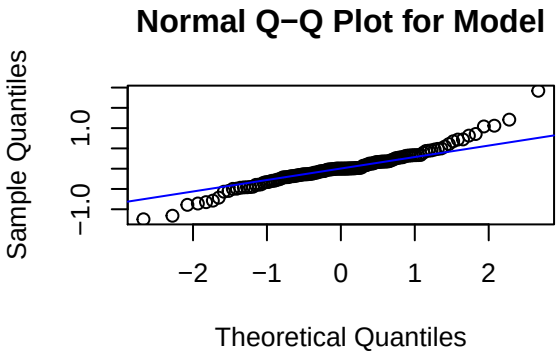
```
##      Estimate     SE t.value p.value
## ar1   -0.3528 0.0866 -4.0750  0.0001
## sar1  -0.5428 0.0894 -6.0690  0.0000
## sar2  -0.4490 0.0930 -4.8258  0.0000
## sar3  -0.3016 0.1020 -2.9573  0.0038
```

Since $|\text{ar1}| < 1$, the AR component is causal invertible. To check causality and invertibility of the seasonal AR component, it suffices to check if all roots of the seasonal AR polynomial lie outside outside the unit circle.



Since all roots are outside the unit circle, the seasonal AR component of our model is causal invertible. We proceed to analysis of residuals. We analyze the plot and histogram of the residuals, as well as the normal Q-Q plot of the model.

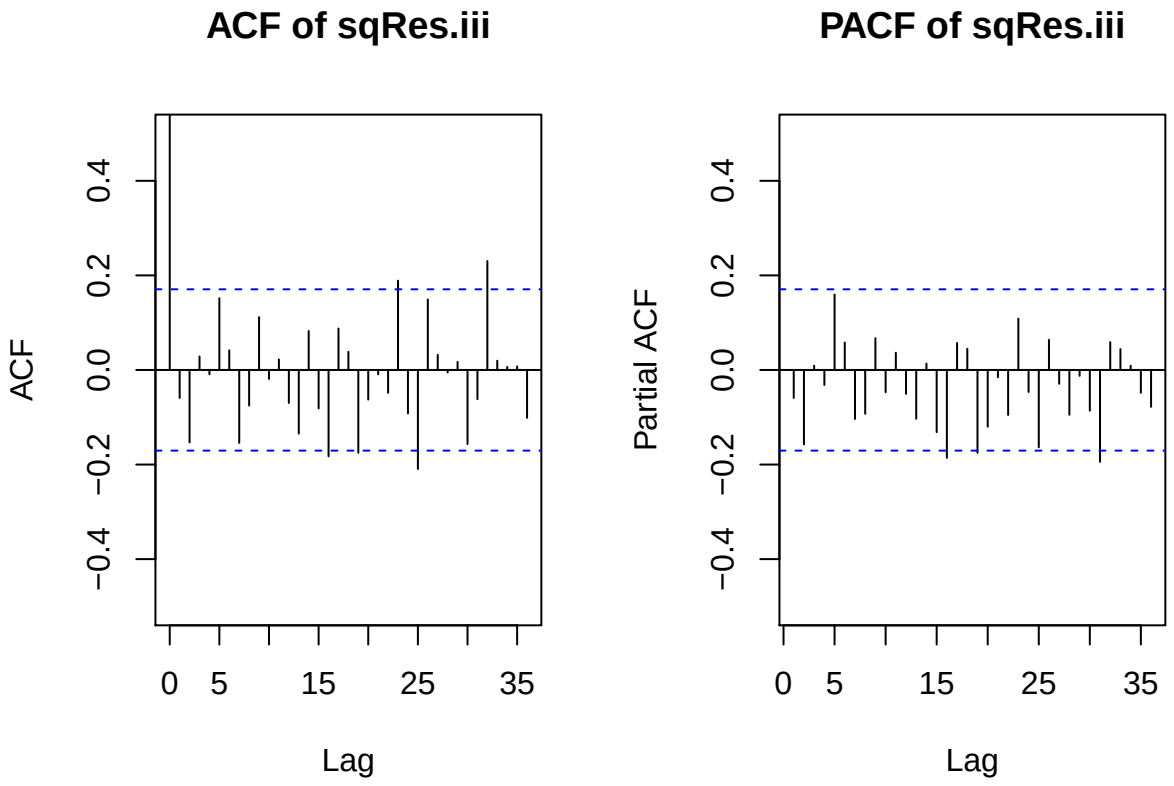




We see that the plot resembles white noise, and that the histogram looks like a slightly right-skewed bell curve.

The Normal Q-Q plot resembles a straight line in the interval $[-1, 1]$; the behavior of the points outside $[-1, 1]$ suggests the data is too peaked in the middle or that the distribution is fat-tailed on both sides.

Next, we plot and analyze the sample ACF and PACF of the residuals:



The ACF of residuals has small peaks outside the confidence interval at lags 23, 25, and 32, but they can be counted as zero by Bartlett's Formula. Similarly, the PACF peaks at lag 16 and 31 can be counted as zero. We conclude the residuals of this model resembles White Noise.

Now, we perform the Yule-Walker, Shapiro-Wilk and Portmanteau tests on the residuals at level $\alpha = 0.05$.

```
# yule walker test
ar(sqres.iii, aic = TRUE, order.max = NULL, method = c("yule-walker"))
```

```
##
## Call:
## ar(x = sqres.iii, aic = TRUE, order.max = NULL, method = c("yule-walker"))
##
##
## Order selected 0  sigma^2 estimated as  0.1857
```

```
shapiro.test(sqres.iii)
# p-value = 6.204e-05 --> fail
```

```
# Portmanteau tests

#fitdf = p+q+P+Q = 4
# Box-Pierce
Box.test(sqres.iii, lag = 11, type = c("Box-Pierce"), fitdf = 4)
```

```
# p-value = 0.08236 --> pass

# Ljung-Box
Box.test(sqres.iii, lag = 11, type = c("Ljung-Box"), fitdf = 4)
#  p-value = 0.06419 --> pass

# McLeod-Li
Box.test(sqres.iii^2, lag = 11, type = c("Ljung-Box"), fitdf = 0)
#  p-value = 0.8929 --> pass
```

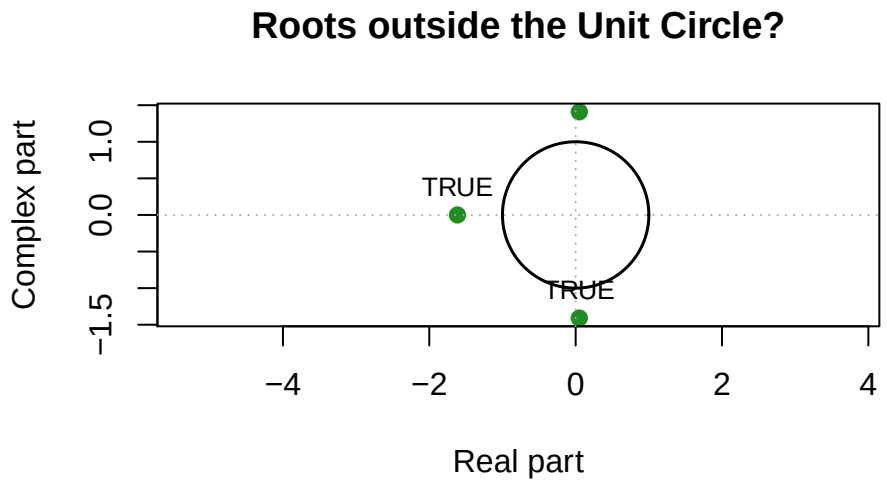
All p-values are larger than 0.05, except for the Shapiro-Wilk test. One can try to improve this model by fitting an ARMA(h,k), with $h \in \{0,16,19,25,32\}$, $k \in \{0,16,19,25\}$ in the residuals, but that will add in at least 16 more coefficients. Thus, we will keep this model.

We now test the SARIMA$(0,1,1)\times(3,1,0)_{12}$ model. The estimated coefficients are:

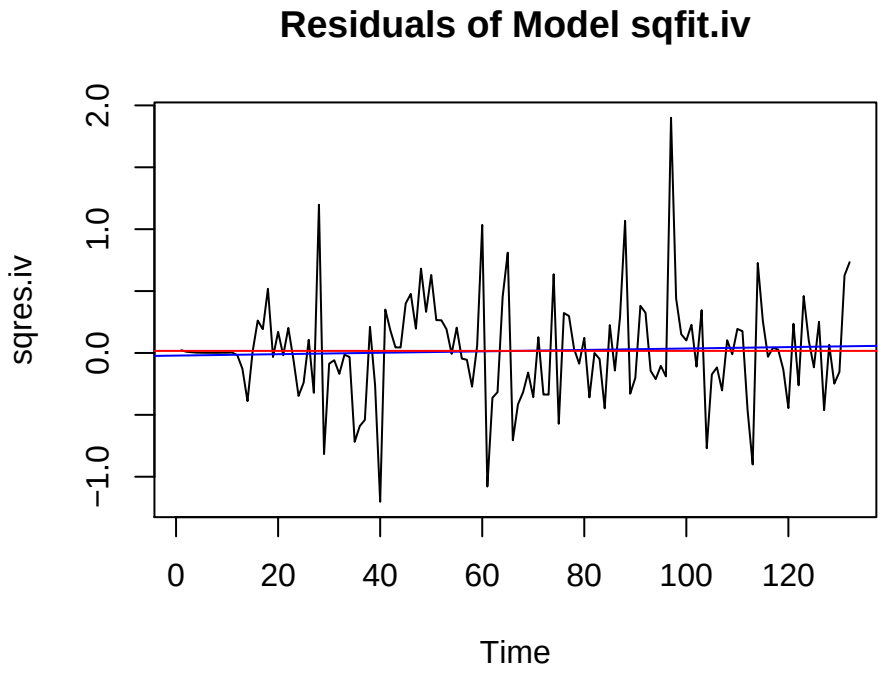
```
sqfit.iv <- sarima( xdata = e.sqrt,
        p = 0, d = 1, q = 1,
        P = 3 , D = 1, Q = 0, S = 12,
        details = F);
print(sqfit.iv$ttable)
```

```
##      Estimate     SE t.value p.value
## ma1   -0.4205 0.0829 -5.0751  0.0000
## sar1  -0.5692 0.0914 -6.2257  0.0000
## sar2  -0.4720 0.0946 -4.9906  0.0000
## sar3  -0.3116 0.1030 -3.0246  0.0031
```

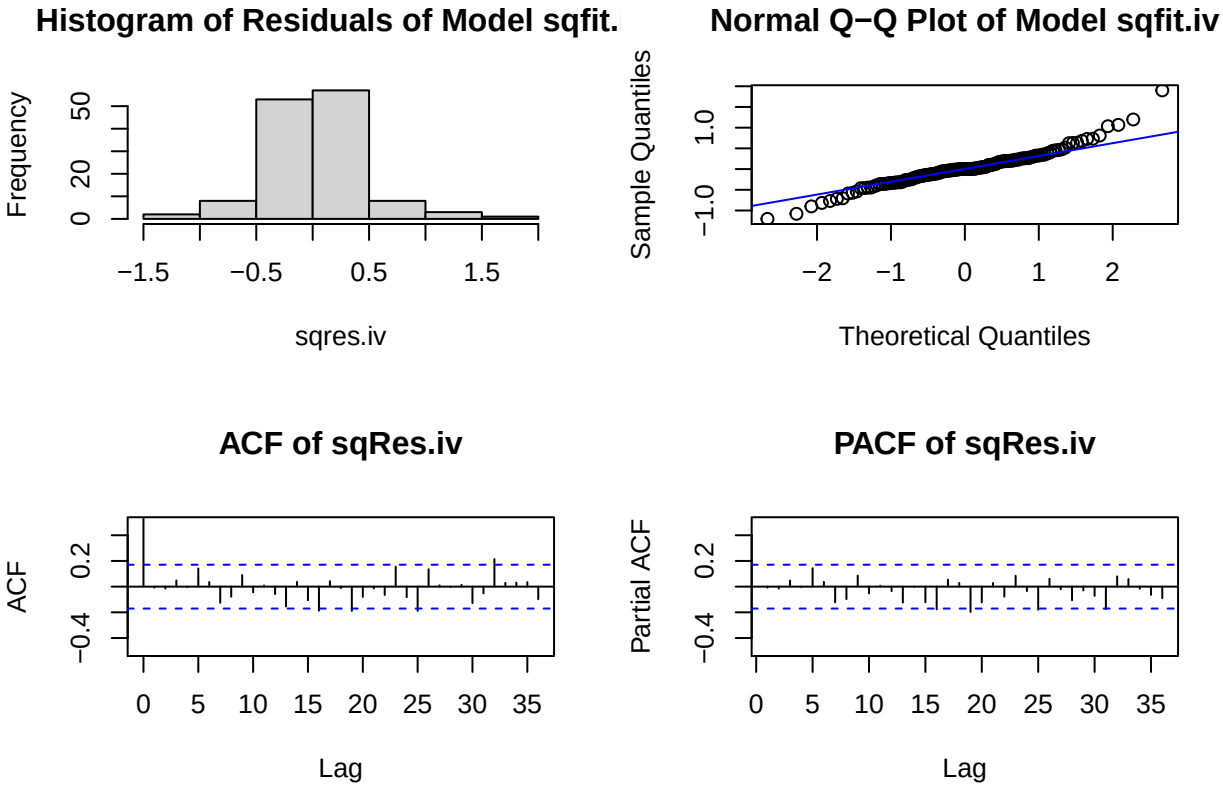
Clearly, the AR component is causal invertible. We apply the unit circle test on the sAR component:



Thus, model sqfit.iv is causal invertible. Proceed to analysis of residuals.



Plot resembles white noise, and histogram looks close to Gaussian. Q-Q plot suggests a fat-tailed distribution.



All peaks in the sample ACF/ PACF can be counted as zero.

```
# yule walker test
ar(sqres.iv, aic = TRUE, order.max = NULL, method = c("yule-walker"))
#  Order selected 0 --> pass


shapiro.test(sqres.iv)
#  p-value = 0.0001258 --> fail


# Box-Pierce
Box.test(sqres.iv, lag = 11, type = c("Box-Pierce"), fitdf = 4)
# p-value = 0.3893 --> pass


# Ljung-Box
Box.test(sqres.iv, lag = 11, type = c("Ljung-Box"), fitdf = 4)
#  p-value = 0.3416 --> pass


Box.test(sqres.iv^2, lag = 11, type = c("Ljung-Box"), fitdf = 0)
# p-value = 0.8374 --> pass
```

Residuals pass all these tests except for Shapiro-Wilk. Again, it is possible to try fitting an ARMA(h,k) in the residuals, but according to the sample ACF and PACF of the residuals, h,k are large. We keep this model and consider a few more before proceeding to the forecast.

We tried a SARIMA$(1,1,1)\times(3,1,0)_{12}$ model but discarded it because the R code produced NaN's and warnings.

We also considered SARIMA$(1,1,0)\times(p,1,q)_{12}$ with $p,q\in\{0,1\}$ but did not use them because their AICc's were higher than the models above.

```
# P=1  p=0, q=1
sqfit.vi <- sarima( xdata = e.sqrt,
        p = 0, d = 1, q = 1,
        P = 1 , D = 1, Q = 0, S = 12,
        details = F)
print(sqfit.vi$AICc)
# AICc 1.397925 --> Don't use


# P=1  p=1, q=0
sqfit.vii <- sarima( xdata = e.sqrt,
        p = 1, d = 1, q = 0,
        P = 1 , D = 1, Q = 0, S = 12,
        details = F)
print(sqfit.vii$AICc)
# AICc 1.414499 --> Don't use
```

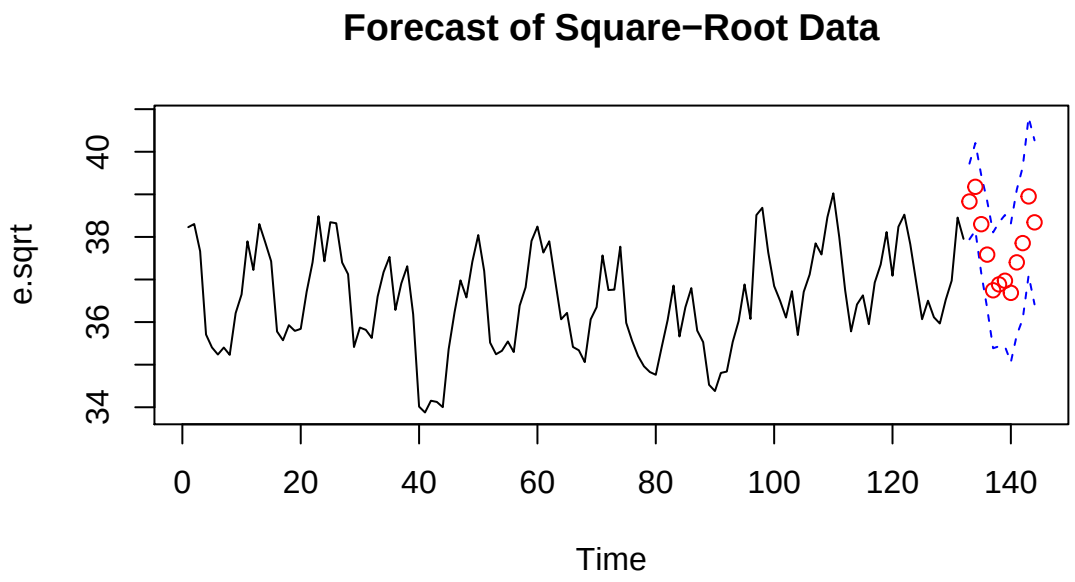
We select the SARIMA$(0,1,1)\times(3,1,0)_{12}$ for $\sqrt{U_t}$ because it had a lower AICc of the two models. We conclude that $\sqrt{U_t}$ follows a SARIMA$(0,1,1)\times(3,1,0)_{12}$ model with equation

$$(1+0.5692B^{12}+0.4720B^{24}+0.3116B^{36})(1-B)(1-B^{12})\sqrt{U_t}=(1-0.4205B)Z_t,$$

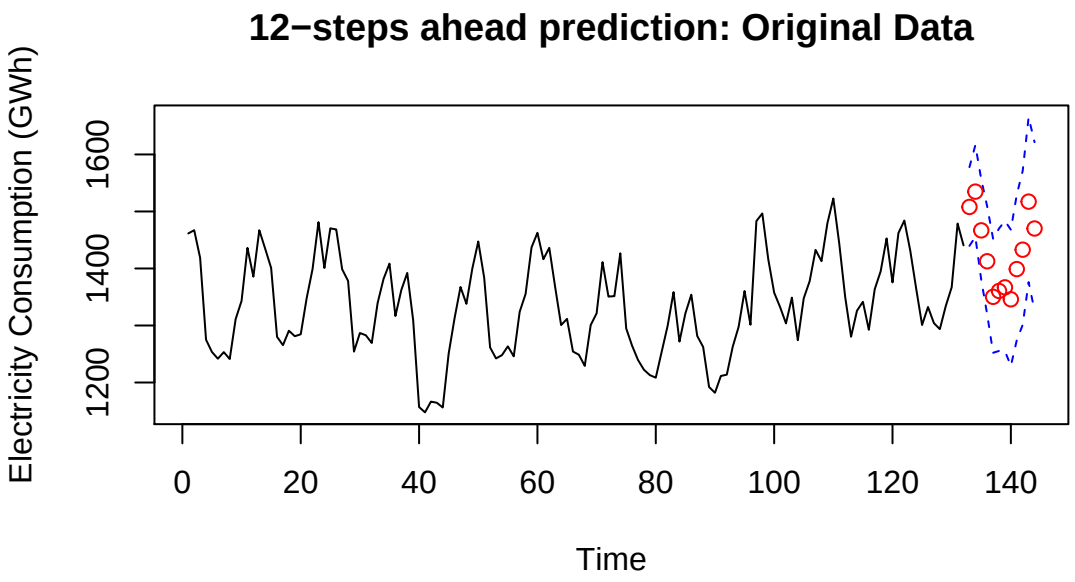
where $Z_t \sim \text{WN}(0, 0.1793)$ and B denotes the backshift operator.

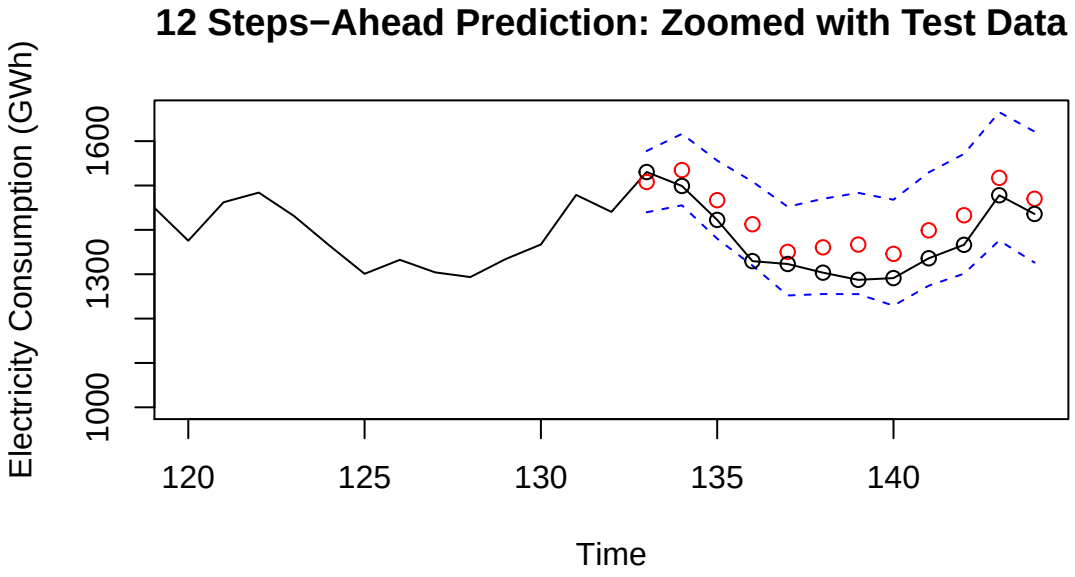
Forecasting

Now, use a 12-steps-ahead predictor to forecast values of our transformed data. All code can be found in the appendix.



Now, we plot a graph of the forecast of our original data. The first plot is a forecast of original data without the actual values. The second plot is a zoomed in version of the first with both predicted and actual vales is shown below. The red circles represent predicted values, the black circles represent actual values, and the blue dashed lines represent the upper and lower bounds of the prediction interval.





Clearly, the actual values are within prediction interval, but they are close to the lower bound.

Conclusion

The goal of this project is to forecast average monthly electricity consumption in Germany in the year 2017 via time-series methods. These goals were achieved by fitting a SARIMA$(0,1,1)\times(3,1,0)_{12}$ to square-root transformed data. The equation of this model is

$$(1+0.5692B^{12}+0.4720B^{24}+0.3116B^{36})(1-B)(1-B^{12})\sqrt{U_t}=(1-0.4205B)Z_t,$$

where $Z_t \sim \text{WN}(0, 0.1793)$. Since the actual values lie within the prediction interval, this model is a good model for this dataset. However, the results of the Normal Q-Q plot and Shapiro-Wilk tests suggest the dataset is not normal. Therefore, heavy-tailed models should be considered to improve the forecast.

Many thanks to Professor Raya Feldmann, TAs Jiamin Lin and Nicole Yang, and classmates Amiya Dutta and Jyotsana Sharma for helping me with this project. Additionally, my cat Titania (Tati) gets some credit for her attempts to debug some of my code.

References

- Feldman, R. (2020). Lecture 7: SARIMA Models [PowerPoint slides]
- Feldman, R. (2020). Week 6, Lecture 12 [PowerPoint slides]
- Feldman, R. (2020). Lecture 15: Let's Do a Time Series Project! [PowerPoint Slides]
- Kaggle. (2019). Germany electricity power for 2006-2017 From Open Power System Data [Data file]. Retrieved from https://www.kaggle.com/mvianna10/germany-electricity-power-for-20062017
- Varshney, P. (2020, April 14). Q-Q Plots Explained. Towards Data Science. https://towardsdatascience.com/q-q-plots-explained-5aa8495426c0

Appendix

```
knitr::opts_chunk$set(echo = TRUE)

path = "C:/Users/KNguyen/Dropbox/PSTAT 174/HW/HW5"
setwd(path)

econsumption.csv <- read.table("opsd_germany_daily.csv", sep=",", header=FALSE, skip=1)
# model 2006-2016 data (nrows=4018): forecast 2017 and use 2017 data as test data

# Plot the original time series -- too noisy.
electric = ts(econsumption.csv[,2], start = c(2006,1,1), frequency = 365)
ts.plot(electric,xlab="Year", ylab="Consumption (GWh)", main = 'Daily German Electricity Consumption')

# Convert V1 (date column) into date format
econsumption.csv$V1 <- as.Date(econsumption.csv$V1, format="%Y-%m-%d")

# create variables for the Month of each observation:
econsumption.csv$Month <- as.Date(cut(econsumption.csv$V1,
  breaks = "month"))

# Loop for monthly average
avg <- vector()
for (mo in unique(econsumption.csv$Month))
{
  ss = subset(econsumption.csv, Month == mo)
  avg <- c(avg, mean(ss$V2))
}
```

```r
# Create new dataframe of monthly averages
e_mo <- data.frame(Month = unique(econsumption.csv$Month), Consumption_Avg = avg)

# Graph Averaged Raw Data
e <- ts(e_mo[,2], start = c(2006,1,1), frequency = 12)
ts.plot(e, xlab = 'Year', ylab = 'Electricity Consumption (GWh)', main = "Raw Monthly Data")


# e is the raw (monthly) data
# split e into train and test
e_train <- e[1:132]
e_test <- e[133:144]
# Plot e_train
ts.plot(e_train, ylab = 'Consumption', main = 'Training Data')

# Draw a Trend Line
abline(lm(e_train ~ as.numeric(1:length(e_train))), col="blue")
# Draw a (constant) Mean Line
abline(h=mean(e_train), col='red')


par(mfrow = c(1, 2))
acf(e_train, lag.max = 48, main = "ACF: Training Data")
pacf(e_train, lag.max = 48, main = "PACF: Training Data")
par(mfrow = c(1, 1))
# ACF suggests seasonality at lag 12

# Box-Cox
library(MASS)
t = 1:length(e_train)
par(mfrow=c(1,2))
hist(e_train)
bcTransform <- boxcox(e_train~ as.numeric(t))  #plots graph

lambda = bcTransform$x[which(bcTransform$y == max(bcTransform$y))] #gives value of \lambda
# wide confidence interval for lambda.
# compare with log and sqrt


e.bc = (1/lambda)*(e_train^lambda-1) #apply box-cox
# Log
e.log <- log(e_train)
# sqrt
e.sqrt <- sqrt(e_train)


# compare plots
#ts.plot(e_train, main = 'Untransformed; e_train')
#ts.plot(e.bc, main = 'Box-Cox Data; e.bc')
#ts.plot(e.log, main = 'Log data; e.log')
#ts.plot(e.sqrt, main = 'Square Root Transform; e.sqrt')

# all plots look similar
```

```r
par(mfrow = c(2, 2))
# Compare histograms
hist(e_train, main = 'Histogram of Untransformed; e_train')
hist(e.bc, main = 'Histogram of Box-Cox; e.bc')
hist(e.log, main = 'Histogram of Log; e.log')
hist(e.sqrt, main = 'Histogram of Square Root; e.sqrt')

# e.sqrt is the easiest transform; try it.
de.sqrt = diff(e.sqrt, lag=12, differences = 1) # difference at lag 12
dde.sqrt = diff(de.sqrt, lag=1, differences = 1) # then difference at lag 1

print(sd(e.sqrt));     #1.173245
print(sd(de.sqrt));    #0.8988376
print(sd(dde.sqrt));   #0.5723453
# plot dde.sqrt
ts.plot(dde.sqrt, main = "Twice-Differenced Square Root Data")
# Draw a Trend Line
abline(lm(dde.sqrt ~ as.numeric(1:length(dde.sqrt))), col="blue")
# Draw a (constant) Mean Line
abline(h=mean(dde.sqrt), col='red')


# acf/pacf of dde.sqrt (seasonal)
par(mfrow = c(2, 2))
acf(dde.sqrt, lag.max = 37, ylim=c(-0.5, 0.5), main = "ACF: Differenced -- Seasonal")
pacf(dde.sqrt, lag.max = 37, ylim=c(-0.5, 0.5), main = "PACF: Seasonal")

# ACF peaks at lags 0,1,12,32, and a smaller one at lag 37(?)
# PACF peaks at lags 1,12,20,25,31,36. It seems to cut off.
# Try P = 1 or 3, Q = 1 or 3
# PACF seems to cut off -- so also try Q = 0

acf(dde.sqrt, lag.max = 11, ylim=c(-0.5,0.5), main = "ACF: non-seasonal")
pacf(dde.sqrt, lag.max = 11, ylim=c(-0.5,0.5), main = "PACF: non-seasonal")
# peaks at lag 1 on both. Try p=q=1.



# needed for sarima
library(astsa)
# needed for aicc
library(qpcR)

#P <- c(1,3)
#Q <- c(0,1)
#p <- c(0,1)
#q <- c(0,1)


#for (P.hat in P)
#{
#  for (Q.hat in Q)
#  {
```

```
#    for (p.hat in p)
#    {
#      for (q.hat in q)
#      {
        # Fit the model
#          fit.m <- sarima( xdata = e.sqrt,
#          p = p.hat, d = 1, q = q.hat,
#          P = P.hat , D = 1, Q = Q.hat, S = 12,
#          details = F);
#          # Print AICC's
#          midd = (min(fit.m$AICc) + max(fit.m$AICc))/2
#          models <- c(fit.m$AICc, P.hat, Q.hat, p.hat, q.hat)
#         if (fit.m$AICc <= min(midd,1.3)) {
#           print((models))
#           }

#      }
#    }
#  }
#}

# These values of P, Q, p, q have the lowest AICc:

# AICc         P.hat         Q.hat        p.hat        q.hat
# 1.207622     1.000000    1.000000      0.000000     1.000000
# 1.220693     1.000000    1.000000      1.000000     0.000000
# 1.222933     1.000000    1.000000      1.000000     1.000000

fit1101sq <- sarima( xdata = e.sqrt,
        p = 0, d = 1, q = 1, #
        P = 1 , D = 1, Q = 1, S = 12,
        details = F); #AICc 1.207622

print(fit1101sq$ttable)
te.sqrt <- diff(e.sqrt, lag = 1) # difference square-root data by lag 1

sd(te.sqrt) #std dev of data differenced by lag 1
# 0.8128187

sd(dde.sqrt) # std dev of data differenced by lag 12, then lag 1
# 0.5723453

fit1111sq <- sarima( xdata = e.sqrt,
        p = 1, d = 1, q = 1,
        P = 1 , D = 1, Q = 1, S = 12,
        details = F);  # AICc 1.222933

print(fit1111sq$ttable)
fit1110sq <- sarima( xdata = e.sqrt,
        p = 1, d = 1, q = 0,
        P = 1 , D = 1, Q = 1, S = 12,
        details = F);
```

```r
print(fit1110sq$ttable)
sqfit.iii <- sarima( xdata = e.sqrt,
        p = 1, d = 1, q = 0,
        P = 3 , D = 1, Q = 0, S = 12,
        details = F);

print(sqfit.iii$ttable)
#install.packages('UnitCircle')
library(UnitCircle)

# coefficients of SAR part
sqcoef.iiisar <- c(1,-1*sqfit.iii$ttable[2:4])
uc.check(pol_ = sqcoef.iiisar, plot_output=TRUE, print_output = FALSE)
# all roots of SAR outside the unit circle --> pass
# check residuals

sqres.iii <- sqfit.iii$fit$residuals
par(mfrow=c(2,2))

plot(sqres.iii, main = "Residuals of Model sqfit.iii")
# Draw a Trend Line
abline(lm(sqres.iii ~ as.numeric(1:length(sqres.iii))), col="blue")
# Draw a (constant) Mean Line
abline(h=mean(sqres.iii), col='red')
# mean and trend line coincide -- olot resembles white noise.

hist(sqres.iii, main = "Histogram of Residuals")
# Looks like a bell curve; slightly right-skewed
qqnorm(sqres.iii, main = "Normal Q-Q Plot for Model")
qqline(sqres.iii, col = 'blue')

par(mfrow=c(1,2))
acf(sqres.iii, lag.max = 36, ylim=c(-0.5,0.5), main = "ACF of sqRes.iii")
pacf(sqres.iii, lag.max = 36, ylim=c(-0.5,0.5), main = "PACF of sqRes.iii")

# yule walker test
ar(sqres.iii, aic = TRUE, order.max = NULL, method = c("yule-walker"))
shapiro.test(sqres.iii)
# p-value = 6.204e-05 --> fail

# Portmanteau tests

#fitdf = p+q+P+Q = 4
# Box-Pierce
Box.test(sqres.iii, lag = 11, type = c("Box-Pierce"), fitdf = 4)
# p-value = 0.08236 --> pass

# Ljung-Box
Box.test(sqres.iii, lag = 11, type = c("Ljung-Box"), fitdf = 4)
#  p-value = 0.06419 --> pass

# McLeod-Li
Box.test(sqres.iii^2, lag = 11, type = c("Ljung-Box"), fitdf = 0)
```

```
#  p-value = 0.8929 --> pass

sqfit.iv <- sarima( xdata = e.sqrt,
        p = 0, d = 1, q = 1,
        P = 3 , D = 1, Q = 0, S = 12,
        details = F);
print(sqfit.iv$ttable)
sqcoef.ivsar <- c(1,-1*sqfit.iv$ttable[2:4])
uc.check(pol_ = sqcoef.ivsar, plot_output=TRUE, print_output = FALSE)
# all roots outside --> pass
# check residuals
sqres.iv <- sqfit.iv$fit$residuals

plot(sqres.iv, main = "Residuals of Model sqfit.iv")
# Draw a Trend Line
abline(lm(sqres.iv ~ as.numeric(1:length(sqres.iv))), col="blue")
# Draw a (constant) Mean Line
abline(h=mean(sqres.iv), col='red')
# resembles white noise

par(mfrow=c(2,2))
hist(sqres.iv, main = "Histogram of Residuals of Model sqfit.iv")
# QQ Test
qqnorm(sqres.iv, main = "Normal Q-Q Plot of Model sqfit.iv")
qqline(sqres.iv, col = 'blue')
# fat-tailed

acf(sqres.iv, lag.max = 36, ylim=c(-0.5,0.5), main = "ACF of sqRes.iv")
pacf(sqres.iv, lag.max = 36, ylim=c(-0.5,0.5), main = "PACF of sqRes.iv")

# acf has small peak at lag 32 --> count as zero
# pacf has small peak at lag 19 --> count as zero
# yule walker test
ar(sqres.iv, aic = TRUE, order.max = NULL, method = c("yule-walker"))
#  Order selected 0 --> pass


shapiro.test(sqres.iv)
#  p-value = 0.0001258 --> fail


# Box-Pierce
Box.test(sqres.iv, lag = 11, type = c("Box-Pierce"), fitdf = 4)
# p-value = 0.3893 --> pass


# Ljung-Box
Box.test(sqres.iv, lag = 11, type = c("Ljung-Box"), fitdf = 4)
#  p-value = 0.3416 --> pass


Box.test(sqres.iv^2, lag = 11, type = c("Ljung-Box"), fitdf = 0)
# p-value = 0.8374 --> pass
```

```r
sqfit.v <- sarima( xdata = e.sqrt,
        p = 1, d = 1, q = 1,
        P = 3 , D = 1, Q = 0, S = 12,
        details = F)
#print(sqfit.v$AICc)
## Warning in sqrt(diag(fitit$var.coef)): NaNs produced


# P=1  p=0, q=1
sqfit.vi <- sarima( xdata = e.sqrt,
        p = 0, d = 1, q = 1,
        P = 1 , D = 1, Q = 0, S = 12,
        details = F)
print(sqfit.vi$AICc)
# AICc 1.397925 --> Don't use


# P=1  p=1, q=0
sqfit.vii <- sarima( xdata = e.sqrt,
        p = 1, d = 1, q = 0,
        P = 1 , D = 1, Q = 0, S = 12,
        details = F)
print(sqfit.vii$AICc)
# AICc 1.414499 --> Don't use

# compute \sigma_Z^2
print(sd(sqres.iv)^2) #0.1793
#install.packages("forecast")
#library(forecast)

# To produce graph with 12 forecasts on TRANSFORMEDdata:
pred.sq <- sarima.for(e.sqrt, n.ahead = 12, p=0,d= 1, q=1, P=3,D=1,Q=0,S=12, plot=FALSE)

# upper bound of prediction interval
U.sq = pred.sq$pred + 2*pred.sq$se
# upper bound
L.sq = pred.sq$pred - 2*pred.sq$se

#plot the predictions
ts.plot(e.sqrt, xlim=c(1,length(e.sqrt)+12), ylim = c(min(e.sqrt),max(U.sq)), main = "Forecast of Squar

lines(U.sq, col="blue", lty="dashed")
lines(L.sq, col="blue", lty="dashed")

points((length(e.sqrt)+1):(length(e.sqrt)+12), pred.sq$pred, col="red")


# To produce graph with forecasts on ORIGINAL data:
pred.og <- (pred.sq$pred)^2
U= (U.sq)^2
L= (L.sq)^2
```

```r
ts.plot(e_train, xlim=c(1,length(e_train)+12), ylim = c(min(e_train),max(U)), ylab = "Electricity Consu

lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")

points((length(e_train)+1):(length(e_train)+12), pred.og, col="red")

# reset the x values of e (raw dataset) to match the x values of e_train
e2 <- ts(e, start = 1, frequency = 1)

#To zoom the graph of original data, starting from entry 110:
ts.plot(e2, xlim = c(120,length(e_train)+12), ylim = c(1000,max(U)), ylab = "Electricity Consumption (G

lines(U, col="blue", lty="dashed")
lines(L, col="blue", lty="dashed")

#predicted values
points((length(e_train)+1):(length(e_train)+12), pred.og, col="red")
#actual values
points((length(e_train)+1):(length(e_train)+12), e_test, col="black")
```