Final Team Project - Team 4

Uyen Pham, Jacqueline Urenda, and Renetta Nelson

December 5, 2022

Loading Libraries

```
library(tidyverse)
library(fpp2)
library(readr)
library(forecast)
library(ggplot2)
library(gridExtra)
library(reshape2)
library(dplyr)
library(lubridate)
library(RColorBrewer)
library(corrplot)
library(Hmisc)
library(ggpubr)
set.seed(506)
```

Loading the Dataset

```
#Load the data
power <- read_csv("Tetuan City power consumption.csv",
    col_types = cols(DateTime = col_datetime(format = "%m/%d/%Y %H:%M")))
```

The columns in the data set are renamed. This was done for easy reference to variables.

```
colnames(power) <- c('DateTime', 'Temperature', 'Humidity','Wind_Speed',
                     'Gen_Diffuse_Flows', 'Diffuse_Flows', 'Zone1', 'Zone2',
                     'Zone3')
head(power)
```

```
## # A tibble: 6 x 9
##   DateTime            Temperature Humidity Wind_~1 Gen_D~2 Diffu~3  Zone1  Zone2
##   <dttm>                    <dbl>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>
## 1 2017-01-01 00:00:00        6.56     73.8   0.083   0.051   0.119 34056. 16129.
## 2 2017-01-01 00:10:00        6.41     74.5   0.083   0.07    0.085 29815. 19375.
## 3 2017-01-01 00:20:00        6.31     74.5   0.08    0.062   0.1   29128. 19007.
## 4 2017-01-01 00:30:00        6.12     75     0.083   0.091   0.096 28229. 18361.
## 5 2017-01-01 00:40:00        5.92     75.7   0.081   0.048   0.085 27336. 17872.
## 6 2017-01-01 00:50:00        5.85     76.9   0.081   0.059   0.108 26625. 17416.
## # ... with 1 more variable: Zone3 <dbl>, and abbreviated variable names
## #   1: Wind_Speed, 2: Gen_Diffuse_Flows, 3: Diffuse_Flows
```

Exploratory Data Analysis

```r
print("Missing Values: ")
```

```
## [1] "Missing Values: "
```

```r
sum(is.na(power)) #no missing values
```

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

Statistical Data Analysis

The data set consists of 364 days total, taking data from January 1, 2017 to December 30, 2017. The time window is every ten minutes. The temperature is measured in Celsius with a mean and median around 18 degrees. The humidity column displays the percentage of the humidity. The average humidity is 68.26%. The wind speed is measured in km/h, and the power consumption is measured in KiloWatts.

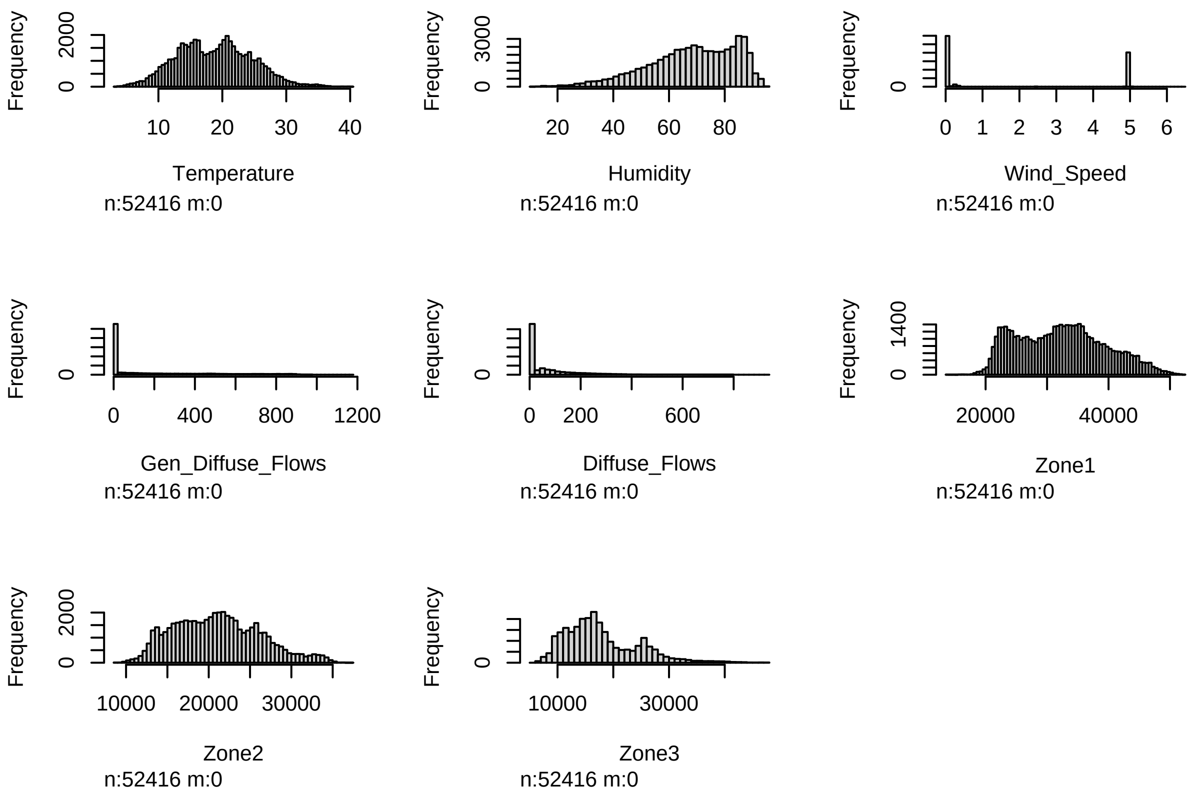
```r
summary(power)
```

```
##     DateTime                      Temperature       Humidity       Wind_Speed
##  Min.   :2017-01-01 00:00:00   Min.   : 3.247   Min.   :11.34   Min.   :0.050
##  1st Qu.:2017-04-01 23:57:30   1st Qu.:14.410   1st Qu.:58.31   1st Qu.:0.078
##  Median :2017-07-01 23:55:00   Median :18.780   Median :69.86   Median :0.086
##  Mean   :2017-07-01 23:55:00   Mean   :18.810   Mean   :68.26   Mean   :1.959
##  3rd Qu.:2017-09-30 23:52:30   3rd Qu.:22.890   3rd Qu.:81.40   3rd Qu.:4.915
##  Max.   :2017-12-30 23:50:00   Max.   :40.010   Max.   :94.80   Max.   :6.483
##  Gen_Diffuse_Flows  Diffuse_Flows         Zone1           Zone2
##  Min.   :   0.004   Min.   :  0.011   Min.   :13896   Min.   : 8560
##  1st Qu.:   0.062   1st Qu.:  0.122   1st Qu.:26311   1st Qu.:16981
##  Median :   5.035   Median :  4.456   Median :32266   Median :20823
##  Mean   : 182.697   Mean   : 75.028   Mean   :32345   Mean   :21043
##  3rd Qu.: 319.600   3rd Qu.:101.000   3rd Qu.:37309   3rd Qu.:24714
##  Max.   :1163.000   Max.   :936.000   Max.   :52204   Max.   :37409
##      Zone3
##  Min.   : 5935
##  1st Qu.:13129
##  Median :16415
##  Mean   :17835
##  3rd Qu.:21624
##  Max.   :47598
```

Histograms

Temperature appears to have a normal distribution. Humidity is left skewed- as mentioned in summary statistics, humidity is pretty high. Wind Speed is interesting as well, looks like either little to low wind speeds or ~4.5 km/h wind speed . General and Diffuse flows are right skewed. Flow tends to be on a lower level. Zone 1 and Zone 2 have a normal distribution. Zone 2 looks normal as well but looks a bit right skewed.

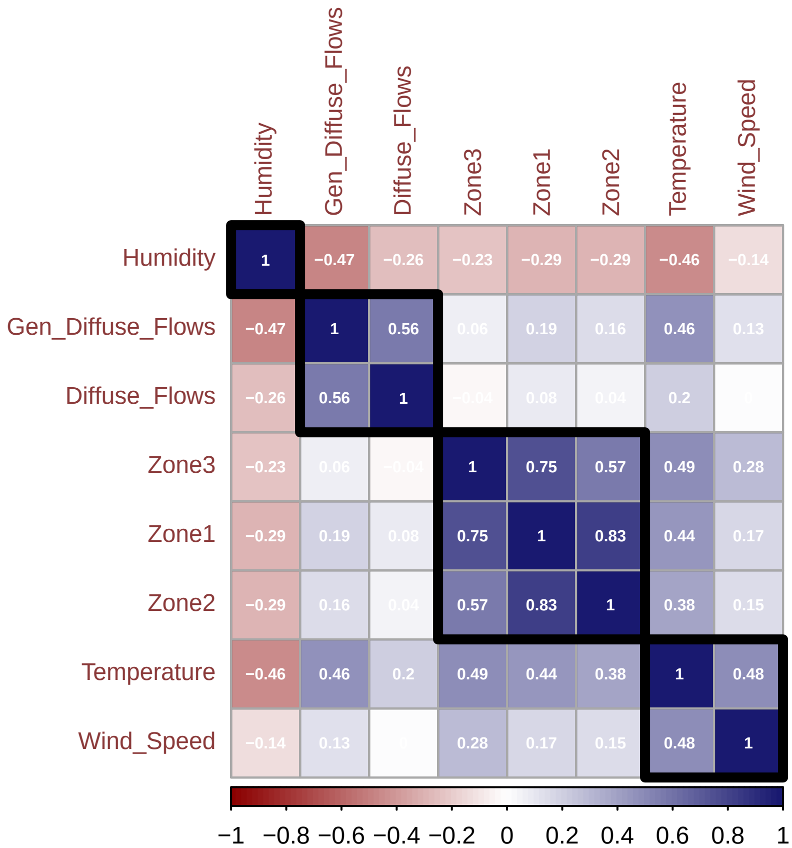
```r
#Histograms of all variables
hist.data.frame(power[,2:9])
```



Correlations

All three zones have strong correlations between each other. Temperature and wind speed have a correlation of 0.48. Humidity and temperature have a moderate but noticeable correlation as well. General diffuse flows and Diffuse flows have a correlation of 0.56. Temperature seems to show highest correlation with power consumption.

```r
corr <- round(x = cor(power[, 2:9]), digits = 2)
corrplot(corr, method = "color", outline = T, addgrid.col = "darkgray",
         order="hclust", addrect = 4, rect.col = "black", rect.lwd = 5,
         cl.pos = "b", tl.col = "indianred4", tl.cex = .75, cl.cex = .75,
         addCoef.col = "white", number.digits = 2, number.cex = 0.5, col =
           colorRampPalette(c("darkred","white","midnightblue"))(100))
```

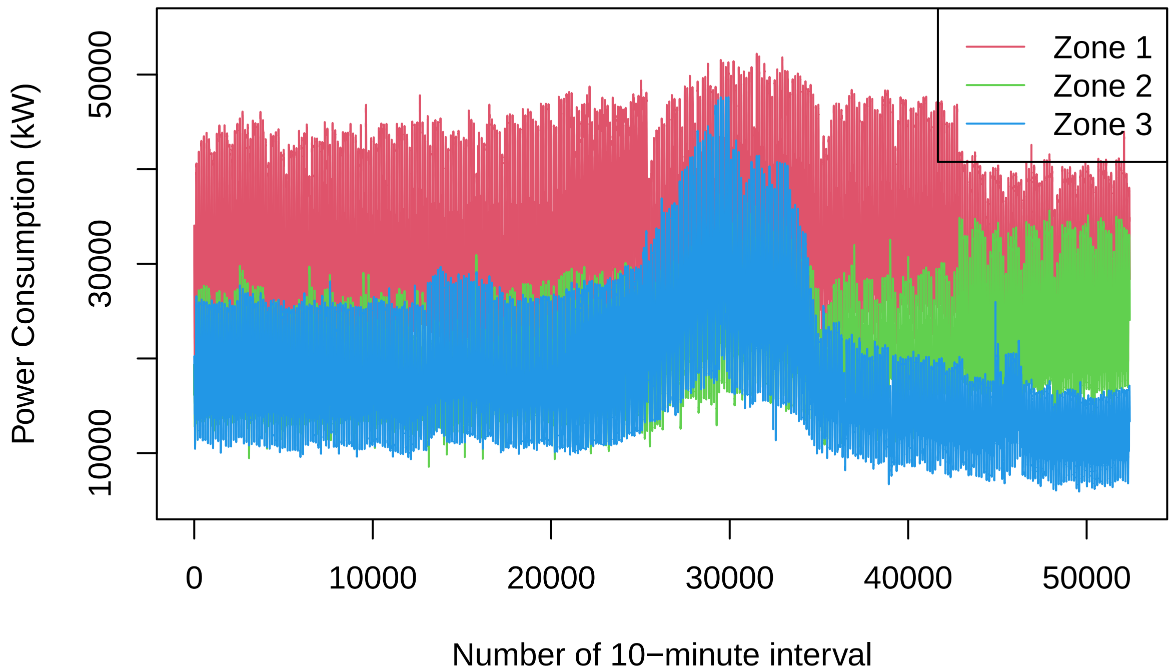



```
#Observe the the pair plots between variables
#plot(power[, 2:9],  pch = 16, cex= 0.3)
```

Comparisons of the Power Consumption Zones

Looking at all three zone's power consumption, it looks like Zone 1 has more power consumption than both Zone 2 and Zone 3.

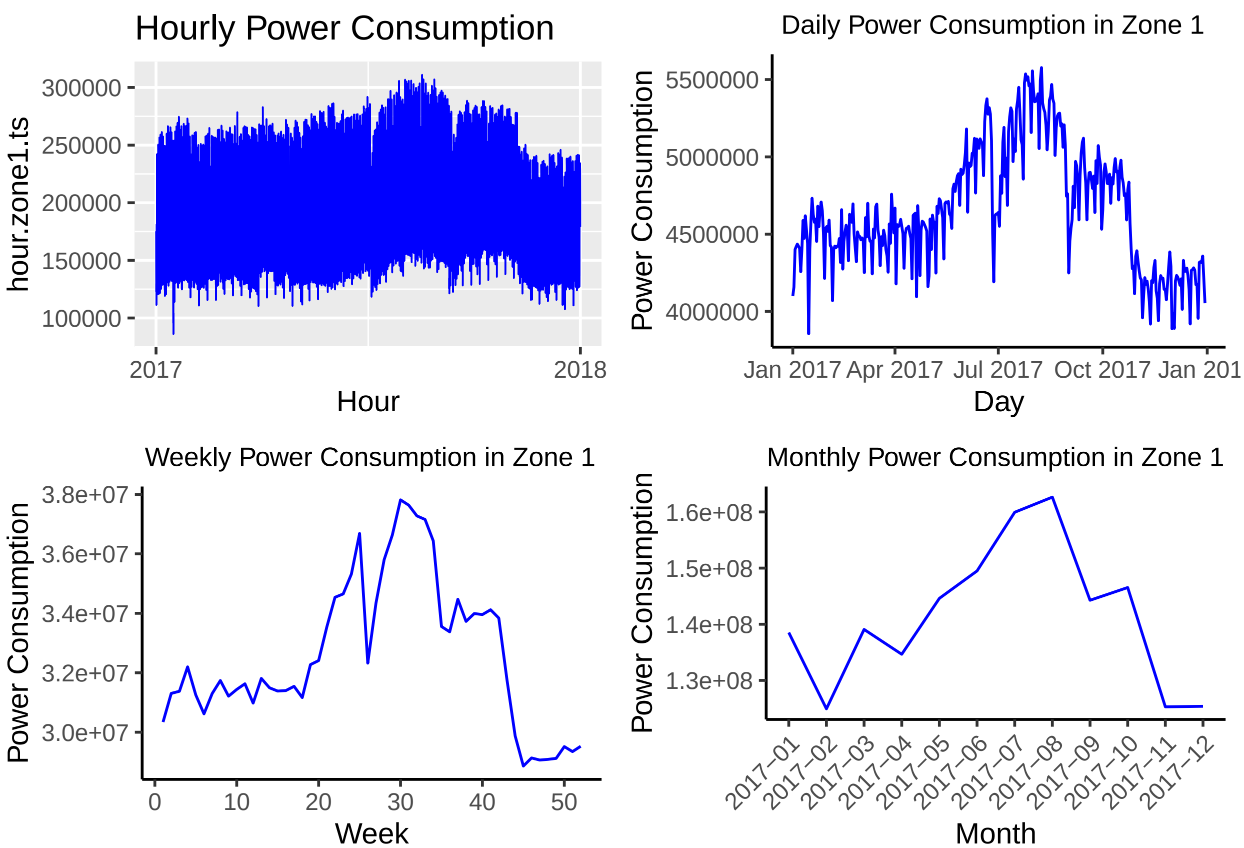
```
plot(power$Zone1,
     type="l",
     col =2,
     ylim = c(5000,55000),
     xlab = "Number of 10-minute interval",
     ylab = "Power Consumption (kW)")
lines(power$Zone2,
       type="l",
     col =3)
lines(power$Zone3,
       type="l",
     col =4)
legend("topright", c("Zone 1", "Zone 2", "Zone 3"), lty = 1, col = 2:4)
```



Explore Power Consumption in different time intervals

```
#Hourly Power consumption
power_hourly <- power %>%
 group_by(Hour= format(DateTime, "%Y-%m-%d %H")) %>%
 summarise(Total= sum(Zone1))

  #Convert Zone1_Consumption to time series
hour.zone1.ts <- ts(power_hourly$Total, start= c(2017,1), end=c(2017, 8376), frequency=8376)
  #Plot the ts
p1 <- autoplot(hour.zone1.ts, color="blue", x = "Hour", main = "Hourly Power Consumption", cex =0.3)

#Daily  Power consumption
power$Date <- as.Date(power$DateTime, format = "%m/%d/%Y")
power.daily.z1 <- power %>% group_by(Date) %>% summarise(Total= sum(Zone1))

p2 <- ggplot(power.daily.z1, aes(x = Date, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Daily Power Consumption in Zone 1",
       x = "Day",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.3, size=10))

#Weekly Power consumption
power_weekly <- power %>%
 group_by(week = lubridate::week(Date)) %>% summarise(Total= sum(Zone1))

p3 <- ggplot(power_weekly, aes(x = week, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Weekly Power Consumption in Zone 1",
```

```r
       x = "Week",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.3, size=10))

#Monthly Power consumption
power_monthly <- power %>%
  group_by(Month = format(Date, "%Y-%m")) %>%
  summarise(Total = sum(Zone1))

p4 <- ggplot(power_monthly, aes(x = Month, y = Total, group = 1)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Monthly Power Consumption in Zone 1",
       x = "Month",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.3, size=10))+
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
grid.arrange(p1, p2, p3, p4, ncol = 2)
```

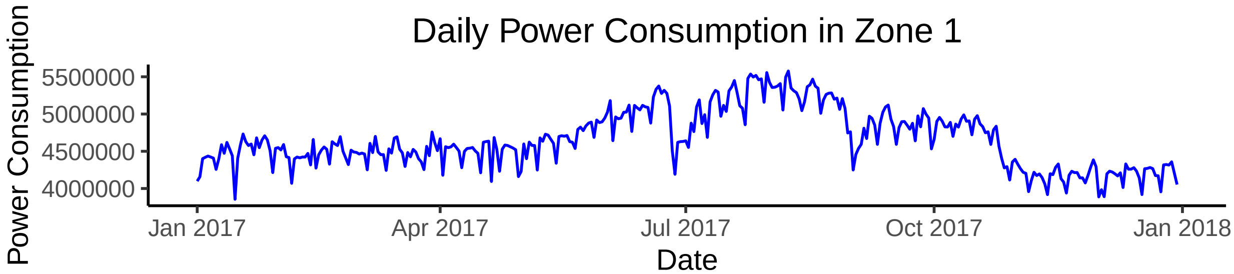


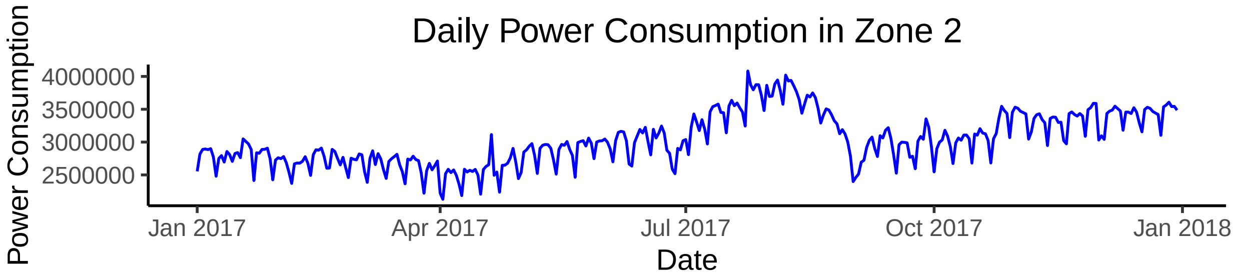
It is more interesting to explore the daily data, since we would be more interesting to forecast the power each Zone needed daily, so the power providers could increase or decrease power supply to meet the demand.

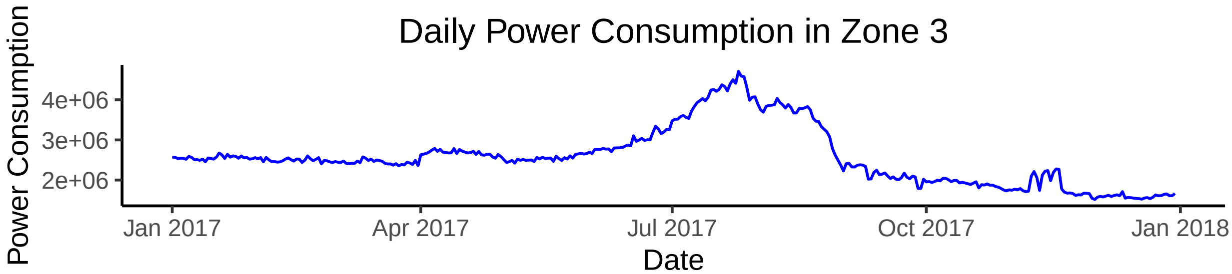
Daily Power Consumption for all three zones

All three zones for power consumption have similar trending paths. The power consumption increases up until it reaches a peak followed by a decline. There appears to be some seasonality. There are some sharp declines on random days, perhaps indicating a presence of power outages. Zone 1 and Zone 2 seem to have big daily fluctuation while Zone 3 seem less fluctuated. Also, They all seem to increase around June to September which could be due to more power consumption during Summer time.

```
power$Date <- as.Date(power$DateTime, format = "%m/%d/%Y")
power.daily.z1 <- power %>% group_by(Date) %>% summarise(Total= sum(Zone1))
power.daily.z2 <- power %>% group_by(Date) %>% summarise(Total= sum(Zone2))
power.daily.z3 <- power %>% group_by(Date) %>% summarise(Total= sum(Zone3))
p1<- ggplot(power.daily.z1, aes(x = Date, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Daily Power Consumption in Zone 1",
       x = "Date",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.5))
p2 <- ggplot(power.daily.z2, aes(x = Date, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Daily Power Consumption in Zone 2",
       x = "Date",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.5))
p3 <- ggplot(power.daily.z3, aes(x = Date, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Daily Power Consumption in Zone 3",
       x = "Date",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.5))
grid.arrange(p1, p2, p3, ncol = 1)
```





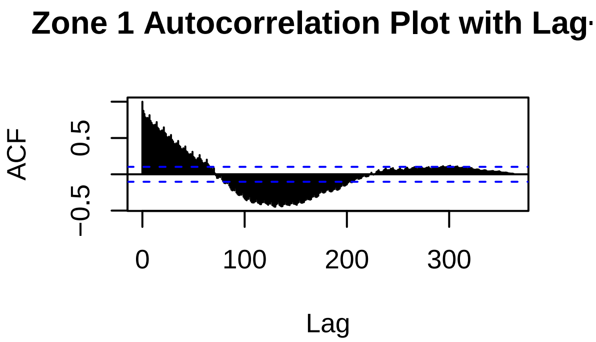


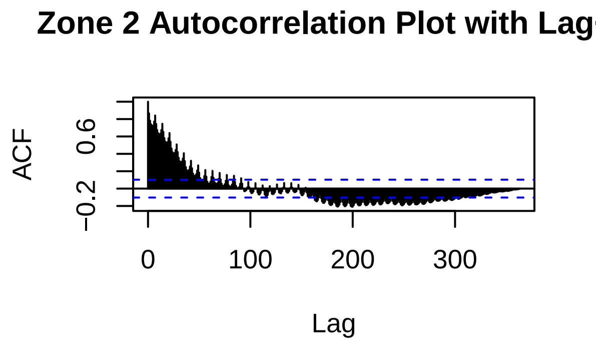
Check for random walk

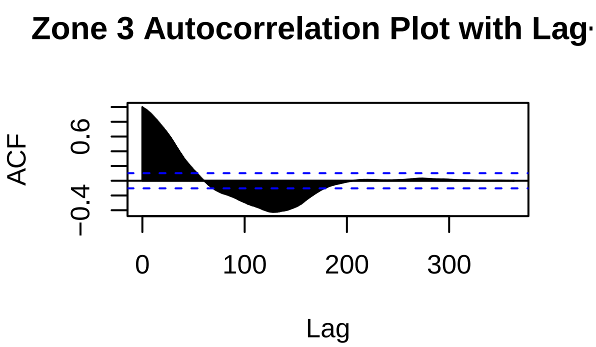
```
#Convert daily power in 3 zones into time series object
power.daily.z1.ts <- ts(power.daily.z1$Total)
power.daily.z2.ts <- ts(power.daily.z2$Total)
power.daily.z3.ts <- ts(power.daily.z3$Total)
```

Autocorrelation show high lag 1 correlation in all three zones with zone 1 very close to random walk while zone 2 and 3 also show correlation at multiple lags which could be weekly seasonality.

```
#Compute and plot the autocorrelation and produce an ACF plot
par(mfrow=c(2,2))
par(mar=c(5,5,5,2))
#To adjust the plot size and not get the title cut off
acf_value1 <- acf(power.daily.z1.ts, lag.max=364,
                  main='Zone 1 Autocorrelation Plot with Lag-1')
acf_value2 <- acf(power.daily.z2.ts, lag.max=364,
                  main='Zone 2 Autocorrelation Plot with Lag-1')
acf_value3 <- acf(power.daily.z3.ts, lag.max=364,
                  main='Zone 3 Autocorrelation Plot with Lag-1')
```








```r
#Get value at lag-1
print(acf_value1[1])
```

```
##
## Autocorrelations of series 'power.daily.z1.ts', by lag
##
##     1
## 0.875
```

```r
print(acf_value2[1])
```

```
##
## Autocorrelations of series 'power.daily.z2.ts', by lag
##
##     1
## 0.866
```

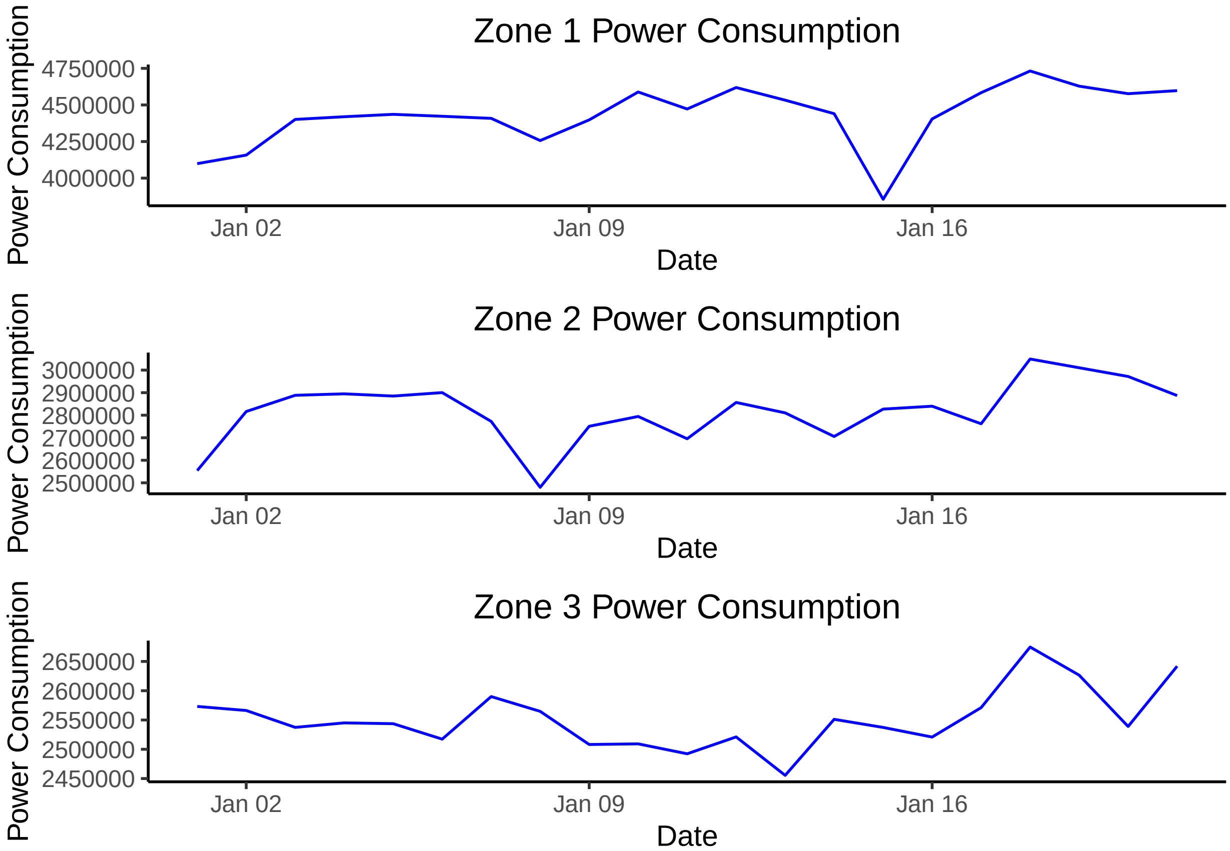
```r
print(acf_value3[1])
```

```
##
## Autocorrelations of series 'power.daily.z3.ts', by lag
##
##     1
## 0.989
```

Examine the first Three Weeks of the Different Zones

The following shows the first few weeks of power consumption for all the zones. Zone 1 and 2 starts low on Sunday, goes high during the weekdays, and then low again on Saturday. Zone 3 seems to be have the opposite behavior and the difference between days are less pronounced.

```r
#Zoom in the first 3 weeks of Zone 1
p1 <- ggplot(power.daily.z1[0:21, ], aes(x = Date, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Zone 1 Power Consumption",
       x = "Date",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.5))

#Zoom in the first 3 weeks of Zone 1
p2 <- ggplot(power.daily.z2[0:21, ], aes(x = Date, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Zone 2 Power Consumption",
       x = "Date",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.5))

#Zoom in the first 3 weeks of Zone 3
p3 <- ggplot(power.daily.z3[0:21, ], aes(x = Date, y = Total)) +
  geom_line(size = 0.5, color = "blue") +
  labs(title = "Zone 3 Power Consumption",
       x = "Date",
       y = "Power Consumption") +
  theme_classic()+
  theme(plot.title=element_text(hjust=0.5))
grid.arrange(p1, p2, p3, ncol = 1)
```



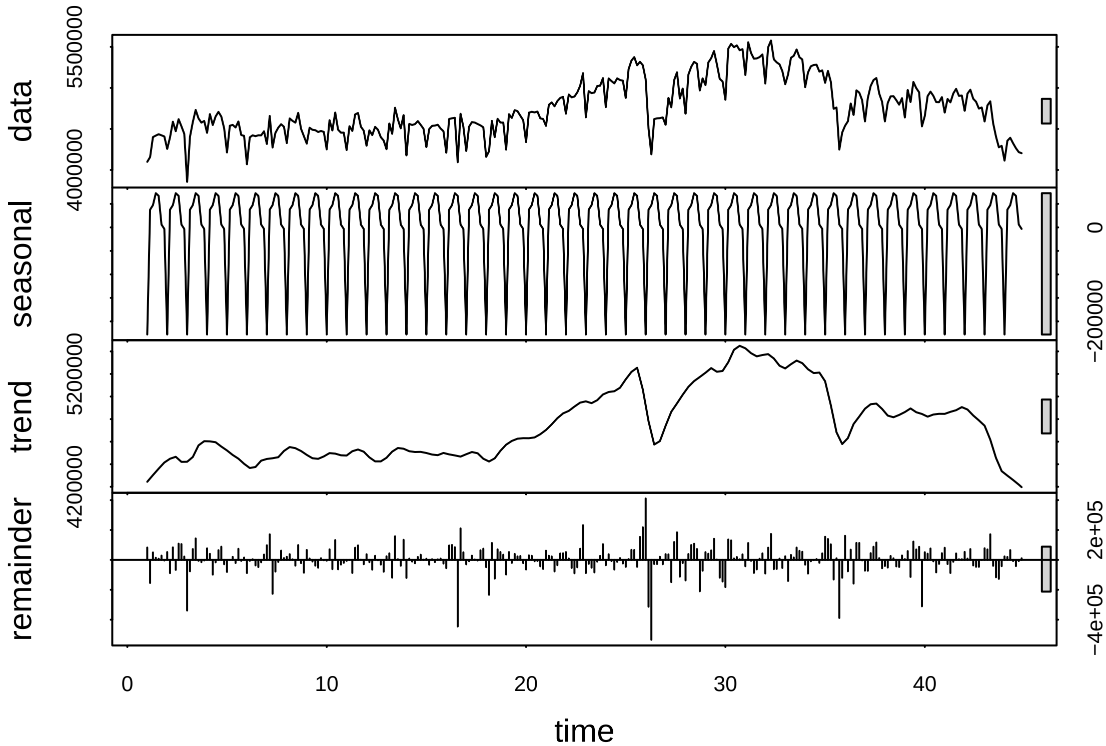
Partitioning the Data

```
#Convert power data frame to daily data including all the variables
daily.sum <- power[ ,2:10] %>%
   group_by(Date) %>%
   summarise(across(everything(), sum), .groups = 'drop')
#Compute daily mean of the variables (Temperature, Humidity, Wind_Speed, Gen_Diffuse_Flows, Diffuse_Flo
daily.var.mean <- (daily.sum[ ,2:6])/144
#Create day of the week column
daily.sum$DOW <- wday(daily.sum$Date, label = TRUE)
#create new frames with values DOW converted to dummies
DOW.dummies <- model.matrix(~ 0 + DOW, data = daily.sum)
#Rename each dummies column without  "Dow" in front
colnames(DOW.dummies) <- gsub("DOW", "", colnames(DOW.dummies))
#Combine the data frames, exclude Tues to avoid dummy trap
X <- as.data.frame(cbind(daily.var.mean, DOW.dummies[, -3]))

#Create y series with power values from daily data frame for zone 1
y1 <- daily.sum$Zone1
y2 <- daily.sum$Zone2
y3 <- daily.sum$Zone3
#Split data
nTotal <- length(y1) #could use length zone 1 for all 3 zones
nValid <- 56
nTrain <- nTotal - nValid
xTrain <- X[1:nTrain, ]
```

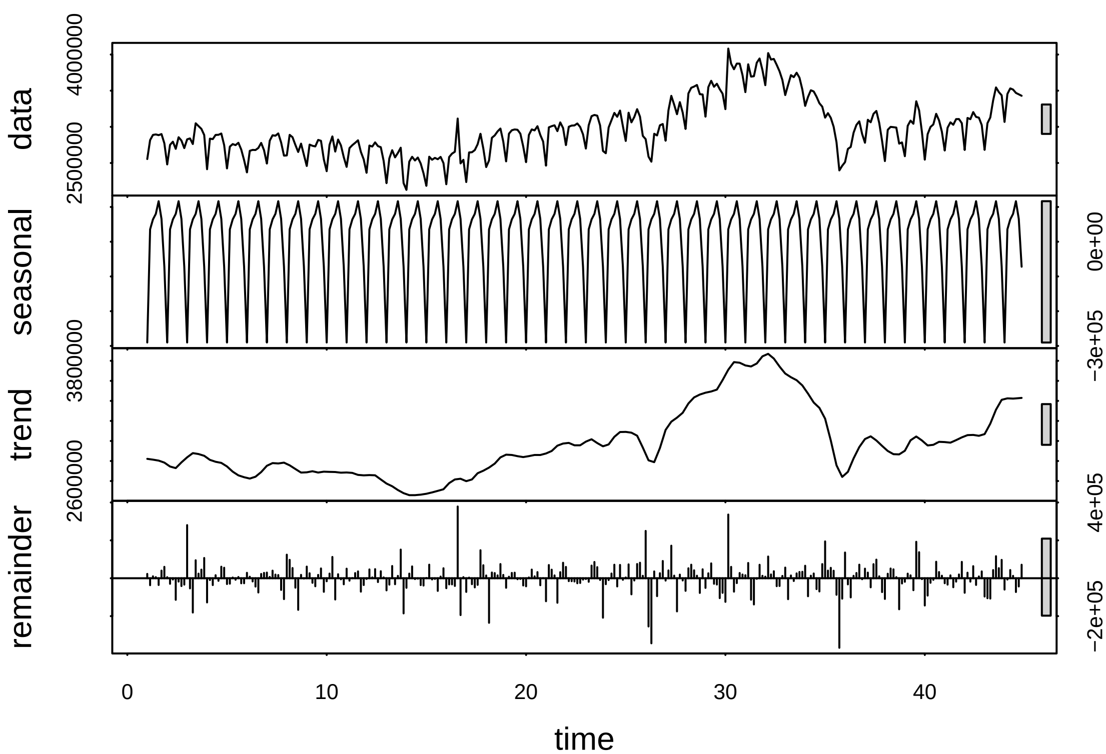
```r
xValid <- X[(nTrain + 1):nTotal, ]
#
##Response variable for train and valid sets
    # Zone 1
yTrain1 <- y1[1:nTrain]
yValid1 <- y1[(nTrain + 1):nTotal]

yTrain2 <- y2[1:nTrain]
yValid2 <- y2[(nTrain + 1):nTotal]

yTrain3 <- y3[1:nTrain]
yValid3 <- y3[(nTrain + 1):nTotal]

#Convert y train and validation into time series object
yTrain1.ts <- ts(yTrain1, start = c(1, 1), end= c(44, 7), frequency = 7)
yValid1.ts <- ts(yValid1, start = c(45, 1), end= c(52, 7), frequency = 7)

yTrain2.ts <- ts(yTrain2, start = c(1, 1), end= c(44, 7), frequency = 7)
yValid2.ts <- ts(yValid2, start = c(45, 1), end= c(52, 7), frequency = 7)

yTrain3.ts <- ts(yTrain3, start = c(1, 1), end= c(44, 7), frequency = 7)
yValid3.ts <- ts(yValid3, start = c(45, 1), end= c(52, 7), frequency = 7)
```

Decomposition of the Time Series

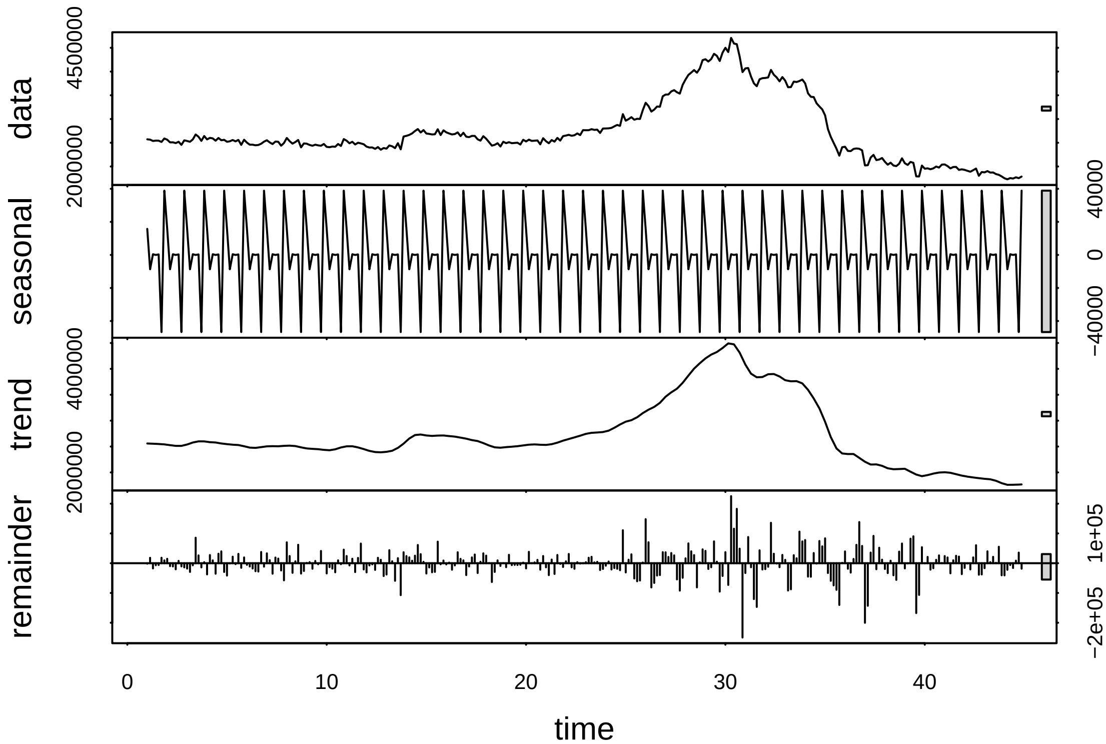
```r
#Decompose time series
stl1.run <- stl(yTrain1.ts, s.window = "periodic")
stl2.run <- stl(yTrain2.ts, s.window = "periodic")
stl3.run <- stl(yTrain3.ts, s.window = "periodic")

plot(stl1.run)
```




```
plot(stl2.run)
```




```
plot(stl3.run)
```




```
# autoplot(yTrain1.ts ) + autolayer(yValid1.ts, color = "blue") + theme_classic() +
#   labs(title = "Training and Validation Sets", y = "Power Consumption (kW)")
# autoplot(yTrain2.ts ) + autolayer(yValid2.ts, color = "blue") + theme_classic() +
#   labs(title = "Training and Validation Sets", y = "Power Consumption (kW)")
# autoplot(yTrain3.ts ) + autolayer(yValid3.ts, color = "blue") + theme_classic() +
#   labs(title = "Training and Validation Sets", y = "Power Consumption (kW)")
```

Modeling

Mean Model

```
#Average model function
mean_predict <- function(yTrain.ts, yValid.ts, titl) {
  mean_power = meanf(yTrain.ts, h=nValid )
  mean_power.ts <- ts( mean_power$mean, start = c(45, 1), end= c(52, 7),
                       frequency = 7)
  p <- autoplot(yTrain.ts) +
    autolayer(mean_power$mean, color = 'red')+
    autolayer(yValid.ts, color = "blue")+
    labs(title =titl,x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))+
    geom_hline(yintercept = mean_power$mean, color = "green", size = 0.5)
  acc <- accuracy(mean_power$mean, yValid.ts)
  lst <- list(acc,p)
  return(lst)}
#call model output
zone1.mean<- mean_predict(yTrain1.ts, yValid1.ts, "Zone 1")
zone2.mean<- mean_predict(yTrain2.ts, yValid2.ts, "Zone 2")
zone3.mean<- mean_predict(yTrain3.ts, yValid3.ts, "Zone 3")
```

```r
zone1.mean[[1]]
```

```
##                    ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -573375.5 586831.7 573375.5 -13.84628 13.84628 0.2941203  3.904904
```

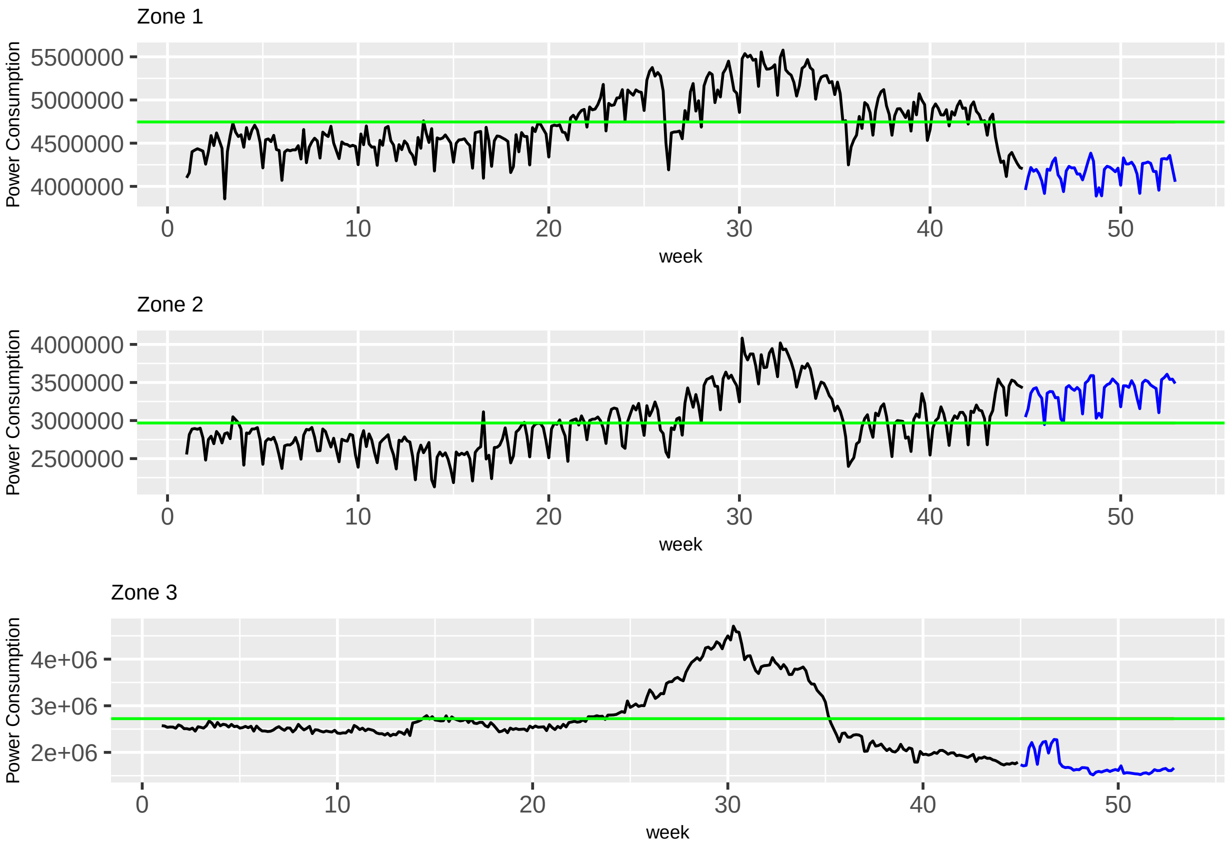
```r
zone2.mean[[1]]
```

```
##                   ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 404226.2 441012.8 404983.5 11.73235 11.75804 0.3748593  2.172706
```

```r
zone3.mean[[1]]
```

```
##                   ME    RMSE     MAE       MPE     MAPE      ACF1 Theil's U
## Test set -1004672 1028633 1004672 -60.70393 60.70393 0.8385883  9.670353
```

```r
ggarrange(zone1.mean[[2]], zone2.mean[[2]], zone3.mean[[2]], ncol=1)
```



Naive Forecast

```r
# Naive Forecast function for all 3 zones
naive <- function(yTrain.ts, yValid.ts, titl) {
  naive_model = forecast::naive(yTrain.ts, h=nValid)
  #naive_model.pred<- forecast(naive_model, h = nValid, level = c(.95))
  naive_model.ts <- ts(naive_model$mean, start = c(45, 1), end= c(52, 7),
                      frequency = 7)
  p <- autoplot(yTrain.ts) +
    autolayer(naive_model$mean, color = 'red')+
    autolayer(yValid.ts, color = "blue")+
    labs(title =titl,x = "week", y = "Power Consumption")+
```

```
    theme (title =element_text(size=7))
    #geom_hline(yintercept = (naive_model$mean, color = "green", size = 0.5))
  acc <- accuracy(naive_model$mean, yValid.ts)
  lst <- list(acc,p)
  return(lst)}
#call model output
zone1.naive<- naive(yTrain1.ts, yValid1.ts, "Zone 1")
zone2.naive<- naive(yTrain2.ts, yValid2.ts, "Zone 2")
zone3.naive<- naive(yTrain3.ts, yValid3.ts, "Zone 3")
```

```
zone1.naive[[1]]
```

```
##                 ME     RMSE      MAE        MPE    MAPE      ACF1 Theil's U
## Test set -31758.88 128921.2 95225.98 -0.8537643 2.32781 0.2941203 0.8467079
```

```
zone2.naive[[1]]
```

```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -57086.56 185343.7 130814.6 -1.987304 4.073163 0.3748593 0.9106244
```

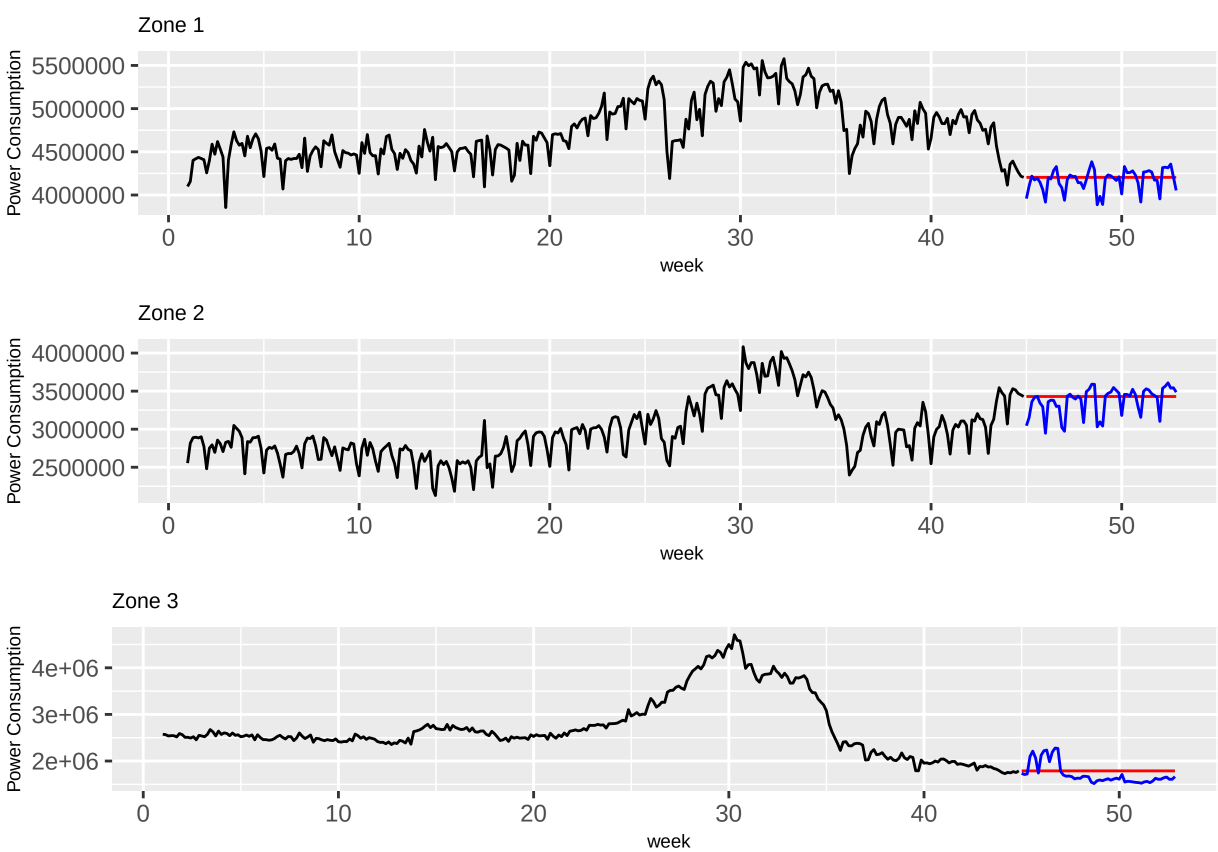
```
zone3.naive[[1]]
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set -70039.93 231569.8 205595.1 -5.54176 11.74121 0.8385883  2.004269
```

```
(zone1.naive[[1]])[2]
```

```
## [1] 128921.2
```

```
ggarrange(zone1.naive[[2]], zone2.naive[[2]], zone3.naive[[2]], ncol=1)
```



Seasonal Naive Forecast

```r
# Build seasonal Naive Forecast function for all 3 zones
snaive <- function(yTrain.ts, yValid.ts, titl) {
  snaive_model = forecast::snaive(yTrain.ts, h=nValid)
  snaive_model.ts <- ts(snaive_model$mean, start = c(45, 1), end= c(52, 7),
                        frequency = 7)
  p <- autoplot(yTrain.ts) +
    autolayer(snaive_model$mean, color = 'red')+
    autolayer(yValid.ts, color = "blue")+
    labs(title =titl,x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))
    #geom_hline(yintercept = (naive_model$mean, color = "green", size = 0.5))
  acc <- accuracy(snaive_model$mean, yValid.ts)
  lst <- list(acc,p)
  return(lst)}
#call model output
zone1.snaive<- snaive(yTrain1.ts, yValid1.ts, "Zone 1")
zone2.snaive<- snaive(yTrain2.ts, yValid2.ts, "Zone 2")
zone3.snaive<- snaive(yTrain3.ts, yValid3.ts, "Zone 3")
```

```r
zone1.snaive[[1]]
```

```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -95450.4 127311.1 105459.7 -2.332443 2.563294 0.3962262  0.857502
```

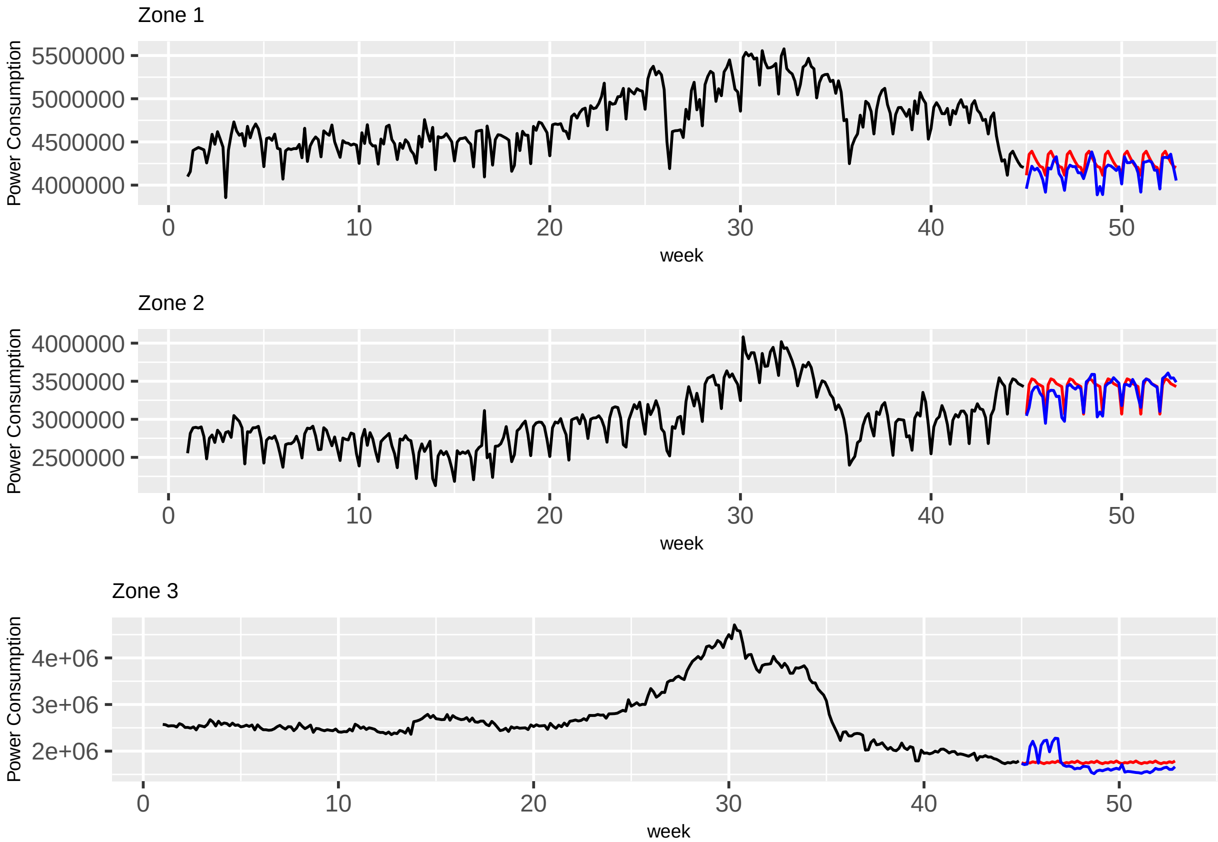
```r
zone2.snaive[[1]]
```

```
##                  ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set -44567.12 125935.4 87784.12 -1.432863 2.683949 0.4514328 0.6385044
```

```r
zone3.snaive[[1]]
```

```
##                  ME   RMSE      MAE      MPE    MAPE      ACF1 Theil's U
## Test set -38740.71 224690 186269.9 -3.69554 10.4615 0.8313098  1.890542
```

```r
ggarrange(zone1.snaive[[2]], zone2.snaive[[2]], zone3.snaive[[2]], ncol=1)
```



Holt Winters

```
#Built function for Holt Winters model
holtW<- function(yTrain.ts, yValid.ts, titl) {
  holt.model <- ets(yTrain.ts, model = "ZAA", alpha = .2, gamma = .05)
  holt.model.pred <- forecast(holt.model, h = nValid)
  holt.model.pred.ts <- ts(holt.model.pred$mean, start = c(45, 1), end= c(52, 7), frequency = 7)
  model <- holt.model
  p <- autoplot(yTrain.ts) +
    autolayer( holt.model.pred, color = 'red')+
    autolayer(yValid.ts, color = "blue")+
    labs(title =titl,x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))
  acc <- accuracy(holt.model.pred, yValid.ts)
  lst <- list(acc,p, model)
  return(lst)}
#call model output
zone1.holt<- holtW(yTrain1.ts, yValid1.ts, "Zone 1")
zone2.holt<- holtW(yTrain2.ts, yValid2.ts, "Zone 2")
zone3.holt<- holtW(yTrain3.ts, yValid3.ts, "Zone 3")
```

```
zone1.holt[[1]]
```

```
##                     ME      RMSE       MAE         MPE     MAPE      MASE
## Training set -1564.421 151970.29 101378.05 -0.1093762 2.156048 0.6155733
## Test set      8045.769  76833.47  57968.65  0.1542896 1.391692 0.3519890
##                   ACF1 Theil's U
```

```
## Training set 0.4544233        NA
## Test set     0.4423987 0.5159244
```

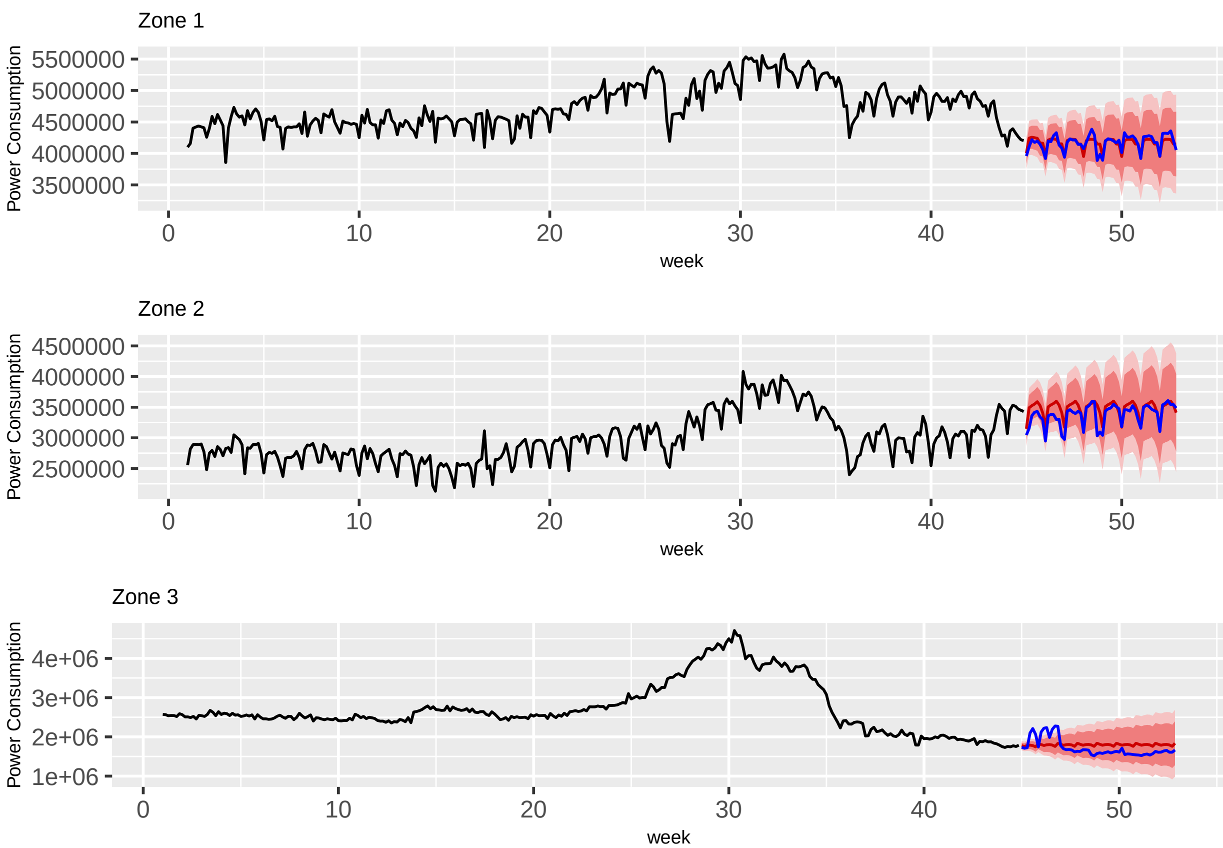
```r
zone2.holt[[1]]
```

```
##                      ME     RMSE       MAE         MPE     MAPE      MASE
## Training set   5126.657 133642.4  90436.87  0.01413649 3.077795 0.5900849
## Test set     -101729.396 151507.5 112031.94 -3.14159657 3.434808 0.7309890
##                   ACF1 Theil's U
## Training set 0.4763998        NA
## Test set     0.5174548 0.7628992
```

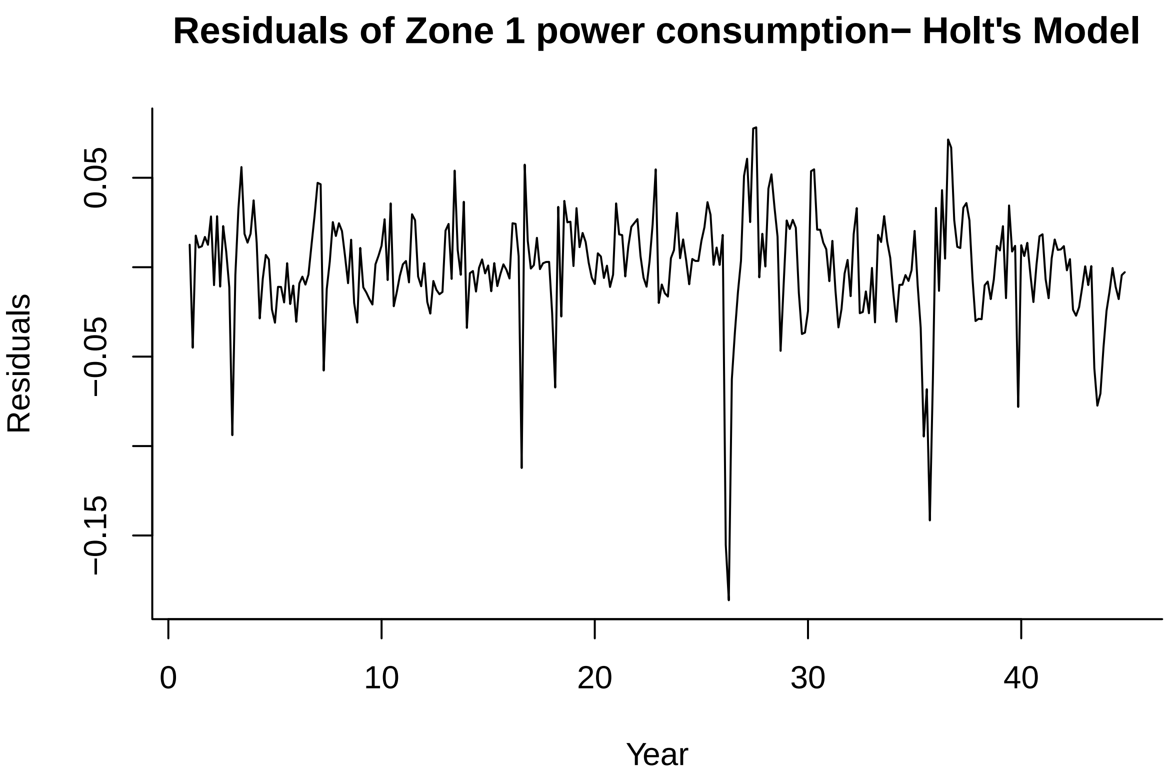
```r
zone3.holt[[1]]
```

```
##                     ME     RMSE       MAE        MPE      MAPE      MASE
## Training set  -2443.093  96697.0  63366.07 -0.1357273  2.290272 0.4209035
## Test set     -78124.597 238432.9 213787.46 -6.0393713 12.245072 1.4200642
##                   ACF1 Theil's U
## Training set 0.6069677        NA
## Test set     0.8287440  2.077576
```

```r
ggarrange(zone1.holt[[2]], zone2.holt[[2]], zone3.holt[[2]], ncol=1)
```




```r
#Residuals plot for Holt model
plot(zone1.holt[[3]]$residuals, xlab = "Year", ylab = "Residuals", bty = "l",
    lwd = 1, main = "Residuals of Zone 1 power consumption- Holt's Model")
```



TSLM

```
#Built function for TSLM
tslm<- function(yTrain.ts, yValid.ts, titl) {
  #Create the formula for the regression model
  (formula <- as.formula(paste("yTrain1.ts", paste(c("trend", "season",
                                                 colnames(xTrain)),
                                               collapse = "+"), sep = "~")))
  tslm.model <- forecast::tslm(formula, data = xTrain)
  tslm.model.pred <- forecast(tslm.model, newdata = xValid)
  tslm.model.pred.ts <- ts(tslm.model.pred$mean, start = c(45, 1), end= c(52, 7), frequency = 7)
  p <- autoplot(yTrain.ts) +
    autolayer( tslm.model.pred, color = 'red')+
    autolayer(yValid.ts, color = "blue")+
    labs(title =titl,x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))
  acc <- accuracy(tslm.model.pred, yValid.ts)
  lst <- list(acc,p)
  return(lst)}
#call model output
zone1.tslm<- tslm(yTrain1.ts, yValid1.ts, "Zone 1")
```

```
## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, :
## prediction from a rank-deficient fit may be misleading
```

```r
zone2.tslm<- tslm(yTrain2.ts, yValid2.ts, "Zone 2")
```

```
## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, :
## prediction from a rank-deficient fit may be misleading
```

```r
zone3.tslm<- tslm(yTrain3.ts, yValid3.ts, "Zone 3")
```

```
## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(predict_object, newdata = newdata, se.fit = TRUE, :
## prediction from a rank-deficient fit may be misleading
```

```r
zone1.tslm[[1]]
```

```
##                         ME     RMSE      MAE        MPE     MAPE     MASE
## Training set  2.421659e-11 214543.3 168725.1 -0.2015387 3.569670 1.024508
## Test set     -2.389210e+05 298700.5 247018.1 -5.7833709 5.973748 1.499908
##                   ACF1 Theil's U
## Training set 0.7255617        NA
## Test set     0.6972350  2.001817
```

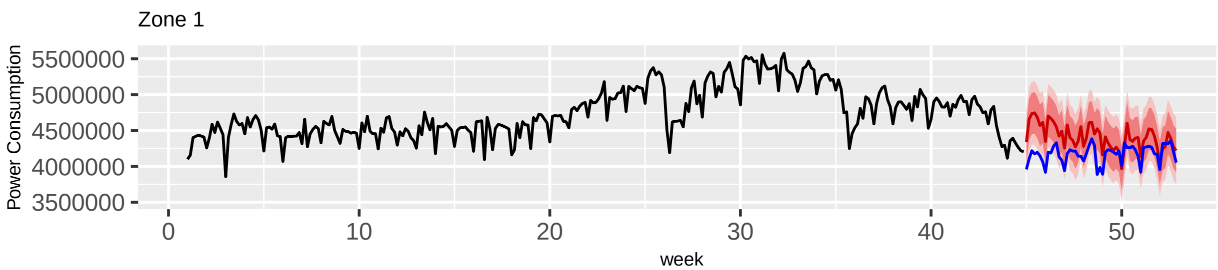
```r
zone2.tslm[[1]]
```

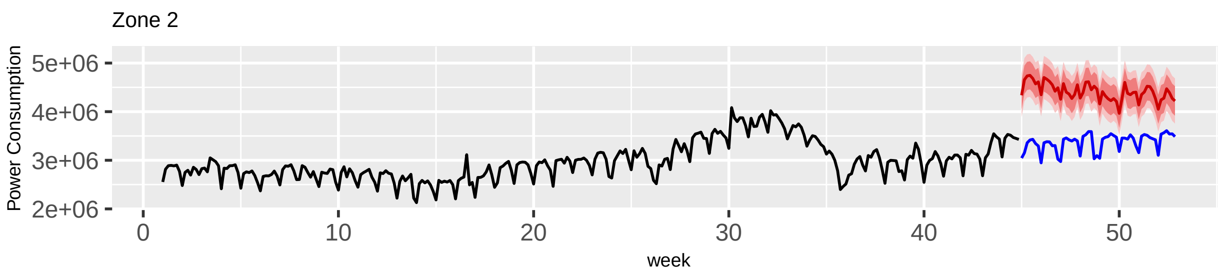
```
##                         ME      RMSE       MAE         MPE     MAPE     MASE
## Training set  2.421659e-11  214543.3  168725.1  -0.2015387  3.56967 1.024508
## Test set     -1.039274e+06 1064187.8 1039274.0 -31.1550282 31.15503 6.310531
##                   ACF1 Theil's U
## Training set 0.7255617        NA
## Test set     0.7220227  5.294553
```

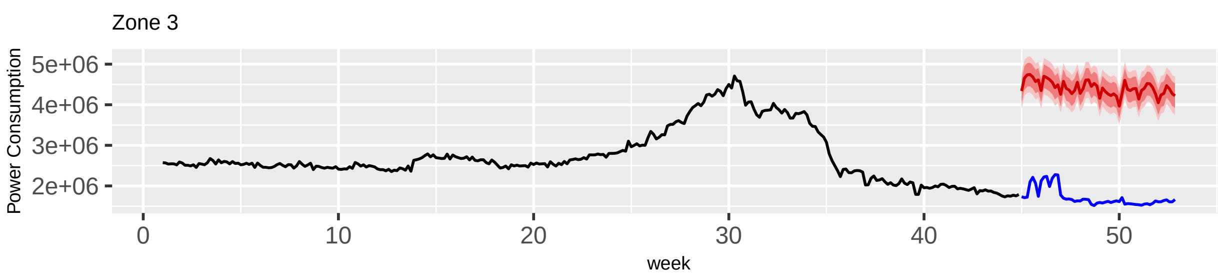
```r
zone3.tslm[[1]]
```

```
##                         ME      RMSE       MAE          MPE      MAPE      MASE
## Training set  2.421659e-11  214543.3  168725.1   -0.2015387   3.56967  1.024508
## Test set     -2.693242e+06 2701385.0 2693241.5 -159.8616239 159.86162 16.353516
##                   ACF1 Theil's U
## Training set 0.7255617        NA
## Test set     0.4921255  24.84325
```

```r
ggarrange(zone1.tslm[[2]], zone2.tslm[[2]], zone3.tslm[[2]], ncol=1)
```







Auto Arima Model

```
#Built function for TSLM model in 3 zones
autoArima<- function(yTrain.ts, yValid.ts, titl) {
  predictor_train <- as.matrix(xTrain)
  predictors_test <- as.matrix(xValid)
  autoArima.model <- forecast::auto.arima(yTrain1.ts, xreg =predictor_train)
  autoArima.model.pred <- forecast(autoArima.model, h = nValid, xreg= predictors_test)
  autoArima.model.pred.ts <- ts(autoArima.model.pred$mean, start = c(45, 1), end= c(52,7), frequency =
  model <- autoArima.model
  p <- autoplot(yTrain.ts) +
    autolayer( autoArima.model.pred, color = 'red')+
    autolayer(yValid.ts, color = "blue")+
    labs(title =titl,x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))
  acc <- accuracy(autoArima.model.pred, yValid.ts)
  lst <- list(acc,p,model)
  return(lst)}
#call model output
zone1.autoArima<- autoArima(yTrain1.ts, yValid1.ts, "Zone 1")
zone2.autoArima<- autoArima(yTrain2.ts, yValid2.ts, "Zone 2")
zone3.autoArima<- autoArima(yTrain3.ts, yValid3.ts, "Zone 3")
```

```
#Output auto arima performance
zone1.autoArima[[1]]
```

```
##                          ME       RMSE        MAE          MPE       MAPE       MASE
```

```
## Training set    3100.921 130277.8  88525.01 -0.01165145 1.887858 0.5375289
## Test set     -260420.338 274318.3 260420.34 -6.26871282 6.268713 1.5812871
##                      ACF1 Theil's U
## Training set -0.008193179        NA
## Test set      0.456826861   1.84643
```

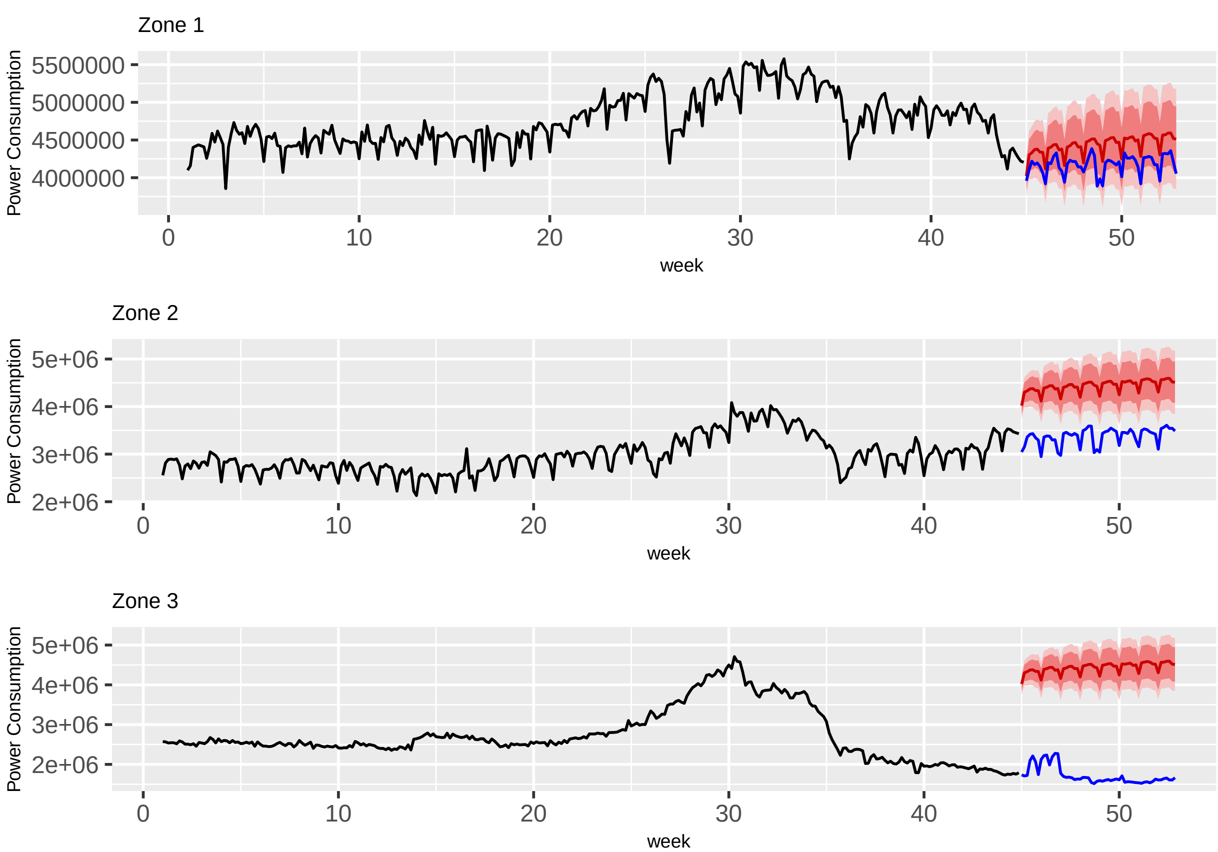
```r
zone2.autoArima[[1]]
```

```
##                        ME      RMSE        MAE          MPE      MAPE      MASE
## Training set     3100.921  130277.8   88525.01  -0.01165145  1.887858 0.5375289
## Test set     -1060773.339 1065558.5 1060773.34 -31.66728795 31.667288 6.4410762
##                      ACF1 Theil's U
## Training set -0.008193179        NA
## Test set      0.349214670  5.291881
```

```r
zone3.autoArima[[1]]
```

```
##                        ME      RMSE        MAE           MPE       MAPE
## Training set     3100.921  130277.8   88525.01   -0.01165145   1.887858
## Test set     -2714740.835 2730131.0 2714740.84 -161.93420553 161.934206
##                    MASE         ACF1 Theil's U
## Training set  0.5375289 -0.008193179        NA
## Test set     16.4840611  0.798755988  25.37145
```

```r
ggarrange(zone1.autoArima[[2]], zone2.autoArima[[2]], zone3.autoArima[[2]], ncol=1)
```




```r
#Output auto arima parameters
zone1.autoArima[[3]]
```

```
## Series: yTrain1.ts
```

```
## Regression with ARIMA(2,0,1)(2,0,1)[7] errors
##
## Coefficients:
##          ar1      ar2      ma1     sar1     sar2    sma1  intercept
##       1.3438  -0.3589  -0.6620  -0.6534  -0.0508  0.7051  4561988.8
## s.e.  0.1925   0.1832   0.1638   0.2624   0.0713  0.2591   225397.9
##       Temperature  Humidity  Wind_Speed  Gen_Diffuse_Flows  Diffuse_Flows
##          1796.499  1096.416    1847.980           190.3560       -89.6163
## s.e.     6503.588  1005.791    6224.514           169.1815       298.2813
##             Sun        Mon       Wed       Thu        Fri       Sat
##       -277573.6  -10841.20  25232.93  19962.89  -42749.64  -52445.9
## s.e.    24722.6   20558.59  20566.58  24826.10   26363.81   26395.2
##
## sigma^2 = 1.803e+10:  log likelihood = -4065.63
## AIC=8169.26   AICc=8171.9   BIC=8240.13
```

```
zone2.autoArima[[3]]
```

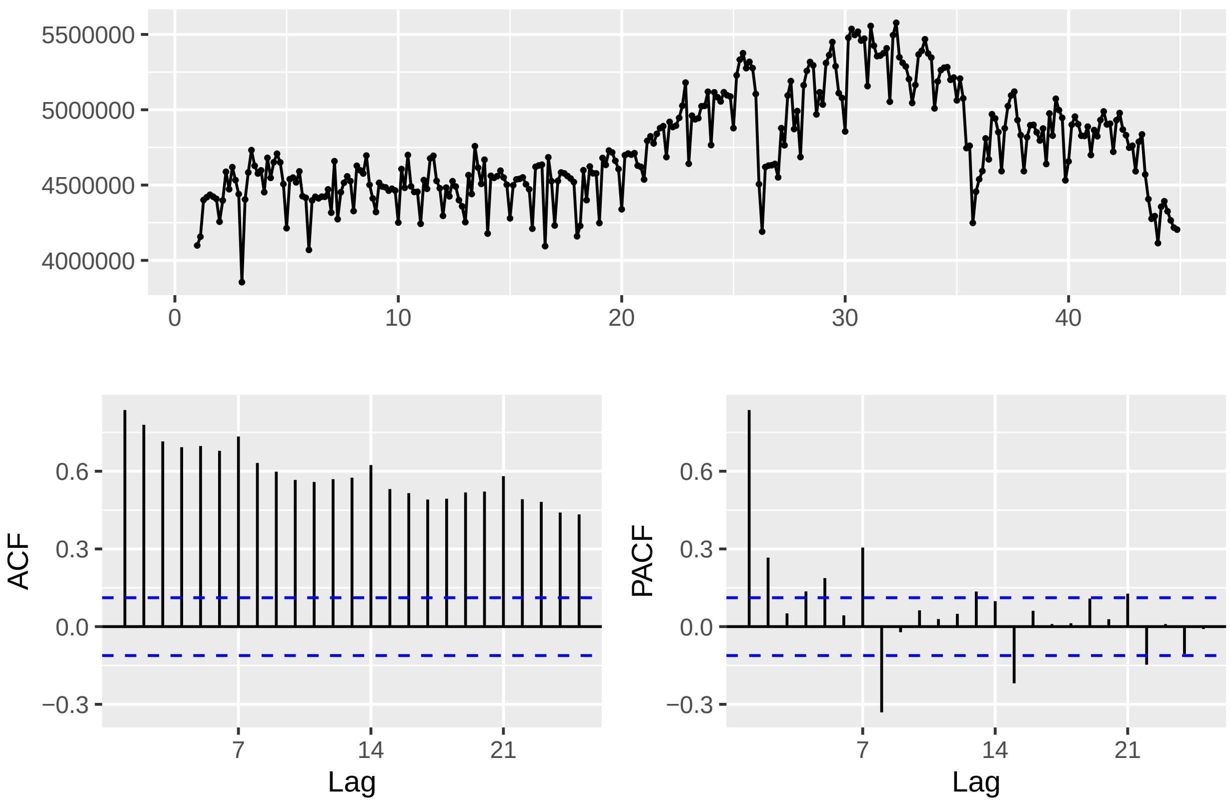
```
## Series: yTrain1.ts
## Regression with ARIMA(2,0,1)(2,0,1)[7] errors
##
## Coefficients:
##          ar1      ar2      ma1     sar1     sar2    sma1  intercept
##       1.3438  -0.3589  -0.6620  -0.6534  -0.0508  0.7051  4561988.8
## s.e.  0.1925   0.1832   0.1638   0.2624   0.0713  0.2591   225397.9
##       Temperature  Humidity  Wind_Speed  Gen_Diffuse_Flows  Diffuse_Flows
##          1796.499  1096.416    1847.980           190.3560       -89.6163
## s.e.     6503.588  1005.791    6224.514           169.1815       298.2813
##             Sun        Mon       Wed       Thu        Fri       Sat
##       -277573.6  -10841.20  25232.93  19962.89  -42749.64  -52445.9
## s.e.    24722.6   20558.59  20566.58  24826.10   26363.81   26395.2
##
## sigma^2 = 1.803e+10:  log likelihood = -4065.63
## AIC=8169.26   AICc=8171.9   BIC=8240.13
```

```
zone3.autoArima[[3]]
```

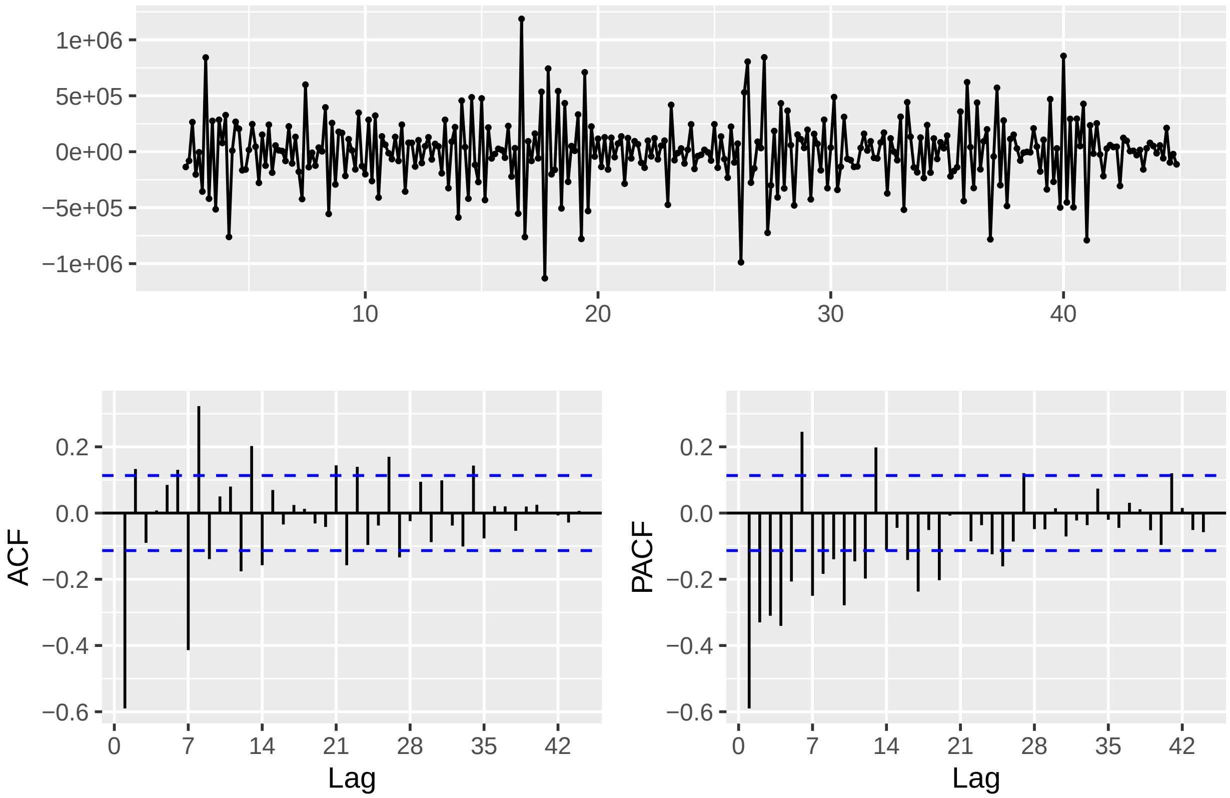
```
## Series: yTrain1.ts
## Regression with ARIMA(2,0,1)(2,0,1)[7] errors
##
## Coefficients:
##          ar1      ar2      ma1     sar1     sar2    sma1  intercept
##       1.3438  -0.3589  -0.6620  -0.6534  -0.0508  0.7051  4561988.8
## s.e.  0.1925   0.1832   0.1638   0.2624   0.0713  0.2591   225397.9
##       Temperature  Humidity  Wind_Speed  Gen_Diffuse_Flows  Diffuse_Flows
##          1796.499  1096.416    1847.980           190.3560       -89.6163
## s.e.     6503.588  1005.791    6224.514           169.1815       298.2813
##             Sun        Mon       Wed       Thu        Fri       Sat
##       -277573.6  -10841.20  25232.93  19962.89  -42749.64  -52445.9
## s.e.    24722.6   20558.59  20566.58  24826.10   26363.81   26395.2
##
## sigma^2 = 1.803e+10:  log likelihood = -4065.63
## AIC=8169.26   AICc=8171.9   BIC=8240.13
```

Optimized ARIMA model on Zone 1

```
#Plot autocorrelation of yTrain1.ts
ggtsdisplay(yTrain1.ts)
```




```
#Plot autocorrelation of differenced yTrain1.ts after two non-seasonal differences and 1 seasonal diffe
yTrain1.ts %>% diff %>% diff%>% diff(lag = 7) %>% ggtsdisplay(lag=44)
```



After taking two non-seasonal differences and 1 seasonal difference,the autocorrelation seem improved and has removed some of the seasonality.

```
#Built function for my Arima models in 3 zones
myArima<- function(yTrain.ts, yValid.ts, titl) {
  predictor_train <- as.matrix(xTrain)
  predictors_test <- as.matrix(xValid)
  myArima.model <- forecast::Arima(yTrain1.ts,  order = c(1,2,1),
                                   seasonal=c(1,0,0), xreg =predictor_train)
  myArima.model.pred <- forecast(myArima.model, h = nValid, xreg= predictors_test)
  myArima.model.pred.ts <- ts(myArima.model.pred$mean, start = c(45, 1), end= c(52,7), frequency = 7)
  model <- myArima.model
  p <- autoplot(yTrain.ts, series='Train', color = "black") +
    autolayer( myArima.model.pred, color = 'red', series='Forecast')+
    autolayer(yValid.ts, color = "blue", series = 'Test')+
    labs(title =titl,x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))+
    guides(color = guide_legend(title = "Data Series")) +
    scale_color_manual(values = c(Train = "black", Forecast = "red",
                                  Test = "blue"))
    #coord_cartesian(xlim = c(15, 54))
  acc <- accuracy(myArima.model.pred, yValid.ts)
  lst <- list(acc,p, model)
  return(lst)}
#call model output
zone1.myArima<- myArima(yTrain1.ts, yValid1.ts, "Zone 1")
zone2.myArima<- myArima(yTrain2.ts, yValid2.ts, "Zone 2")
zone3.myArima<- myArima(yTrain3.ts, yValid3.ts, "Zone 3")
```

```r
#Output auto arima performance
zone1.myArima[[1]]
```

```
##                      ME      RMSE      MAE        MPE     MAPE      MASE
## Training set  -4872.857 133783.69 88708.39 -0.1518214 1.894779 0.5386424
## Test set     -32504.793  80930.79 61207.14 -0.8143030 1.484744 0.3716533
##                     ACF1 Theil's U
## Training set -0.02427314        NA
## Test set      0.38306457 0.5460399
```

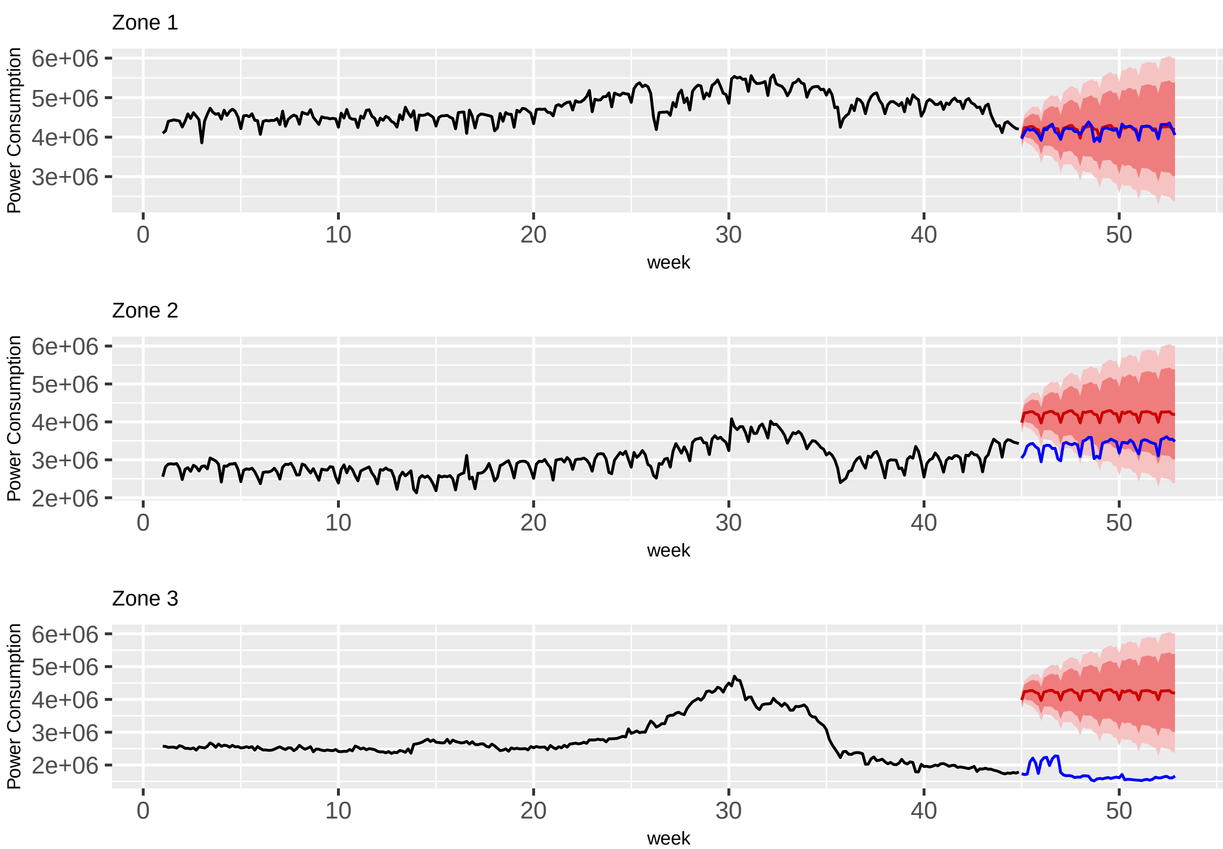
```r
zone2.myArima[[1]]
```

```
##                       ME     RMSE       MAE         MPE      MAPE      MASE
## Training set   -4872.857 133783.7  88708.39  -0.1518214  1.894779 0.5386424
## Test set     -832857.795 841332.3 832857.79 -24.9363272 24.936327 5.0571600
##                     ACF1 Theil's U
## Training set -0.02427314        NA
## Test set      0.52757630  4.180559
```

```r
zone3.myArima[[1]]
```

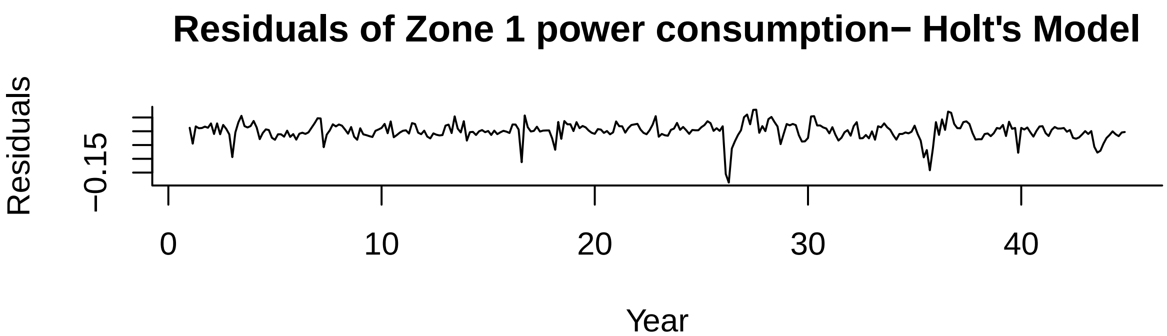
```
##                        ME      RMSE        MAE          MPE       MAPE
## Training set    -4872.857  133783.7   88708.39   -0.1518214   1.894779
## Test set     -2486825.290 2498228.4 2486825.29 -148.1704264 148.170426
##                    MASE        ACF1 Theil's U
## Training set  0.5386424 -0.02427314        NA
## Test set     15.1001449  0.73316989  23.14676
```

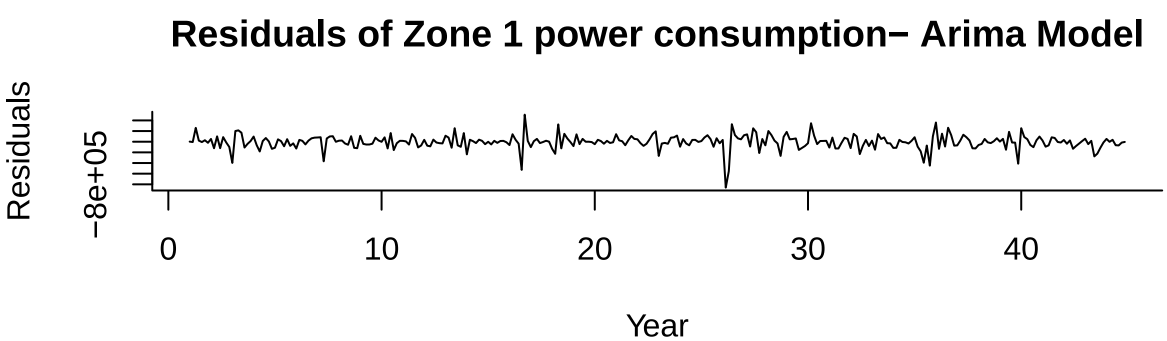
```r
grid.arrange(zone1.myArima[[2]], zone2.myArima[[2]], zone3.myArima[[2]], ncol=1)
```



Residual plots for Holt's and Arima model

```
par(mfrow=c(2,1))
#Residuals plot for Holt model
plot(zone1.holt[[3]]$residuals, xlab = "Year", ylab = "Residuals", bty = "l",
    lwd = 1, main = "Residuals of Zone 1 power consumption- Holt's Model")
#Residuals plot for arima model
plot(zone1.myArima[[3]]$residuals, xlab = "Year", ylab = "Residuals", bty = "l",
    lwd = 1, main = "Residuals of Zone 1 power consumption- Arima Model")
```





Neuro Network

```
#Built function for my Arima models in 3 zones
nnetar<- function(yTrain.ts, yValid.ts, titl) {
  nnetar.model <- forecast::nnetar(yTrain1.ts)
  nnetar.model.pred <- forecast(nnetar.model, h = nValid)
  #nnetar.model.pred.ts <- ts(nnetar.model.pred, start = c(45, 1), end= c(52,7), frequency = 7)
  p <- autoplot(yTrain.ts, series='Train', color = "black") +
    autolayer( nnetar.model.pred, color = 'red', series='Forecast')+
    autolayer(yValid.ts, color = "blue", series = 'Test')+
    labs(title =titl,x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))+
    guides(color = guide_legend(title = "Data Series")) +
    scale_color_manual(values = c(Train = "black", Forecast = "red",
                                  Test = "blue"))
    #coord_cartesian(xlim = c(15, 54))
  acc <- accuracy(nnetar.model.pred, yValid.ts)
  lst <- list(acc,p)
  return(lst)}
```

```r
#call model output
zone1.nnetar<- nnetar(yTrain1.ts, yValid1.ts, "Zone 1")
zone2.nnetar<- nnetar(yTrain2.ts, yValid2.ts, "Zone 2")
zone3.nnetar<- nnetar(yTrain3.ts, yValid3.ts, "Zone 3")
```

```r
#Output auto arima performance
zone1.nnetar[[1]]
```

```
##                        ME      RMSE       MAE         MPE      MAPE      MASE
## Training set     225.8251  54156.38  35419.41 -0.02358917 0.7549447 0.2150687
## Test set     -289644.0365 313655.66 289644.04 -7.02972900 7.0297290 1.7587351
##                    ACF1 Theil's U
## Training set 0.07300538        NA
## Test set     0.31786933  2.118797
```

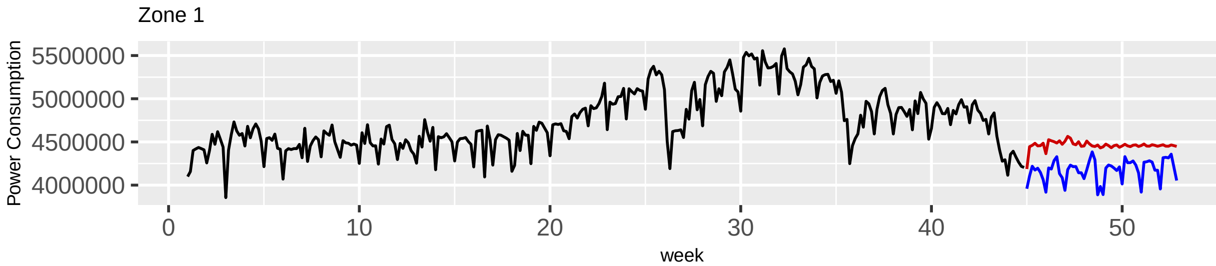
```r
zone2.nnetar[[1]]
```

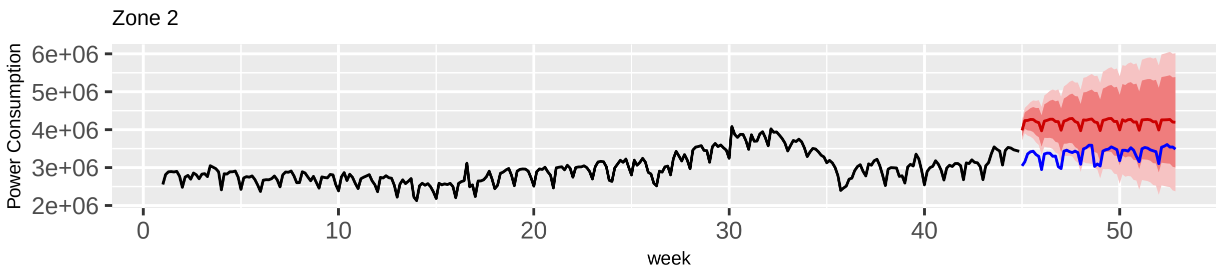
```
##                         ME      RMSE         MAE         MPE       MAPE
## Training set     -108.2002   52347.5    33996.09  -0.0303586  0.7224266
## Test set     -1091461.1810 1104931.1 1091461.18 -32.7190310 32.7190310
##                   MASE       ACF1 Theil's U
## Training set 0.2064262 0.03018241        NA
## Test set     6.6274145 0.41419058  5.505867
```

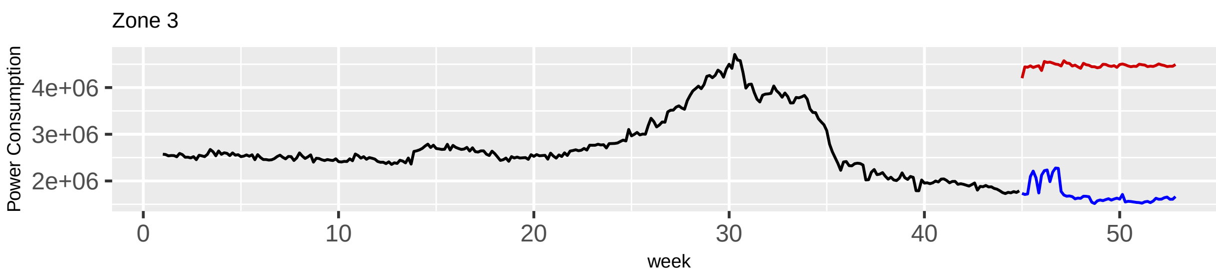
```r
zone3.nnetar[[1]]
```

```
##                         ME       RMSE         MAE           MPE        MAPE
## Training set      178.2742    50509.29    33859.56   -0.02358424   0.7190109
## Test set     -2751192.6832 2759897.98 2751192.68 -163.73761505 163.7376151
##                    MASE      ACF1 Theil's U
## Training set  0.2055972 0.1017592        NA
## Test set     16.7053988 0.7822626  25.52062
```

```r
grid.arrange(zone1.nnetar[[2]], zone2.myArima[[2]], zone3.nnetar[[2]], ncol=1)
```

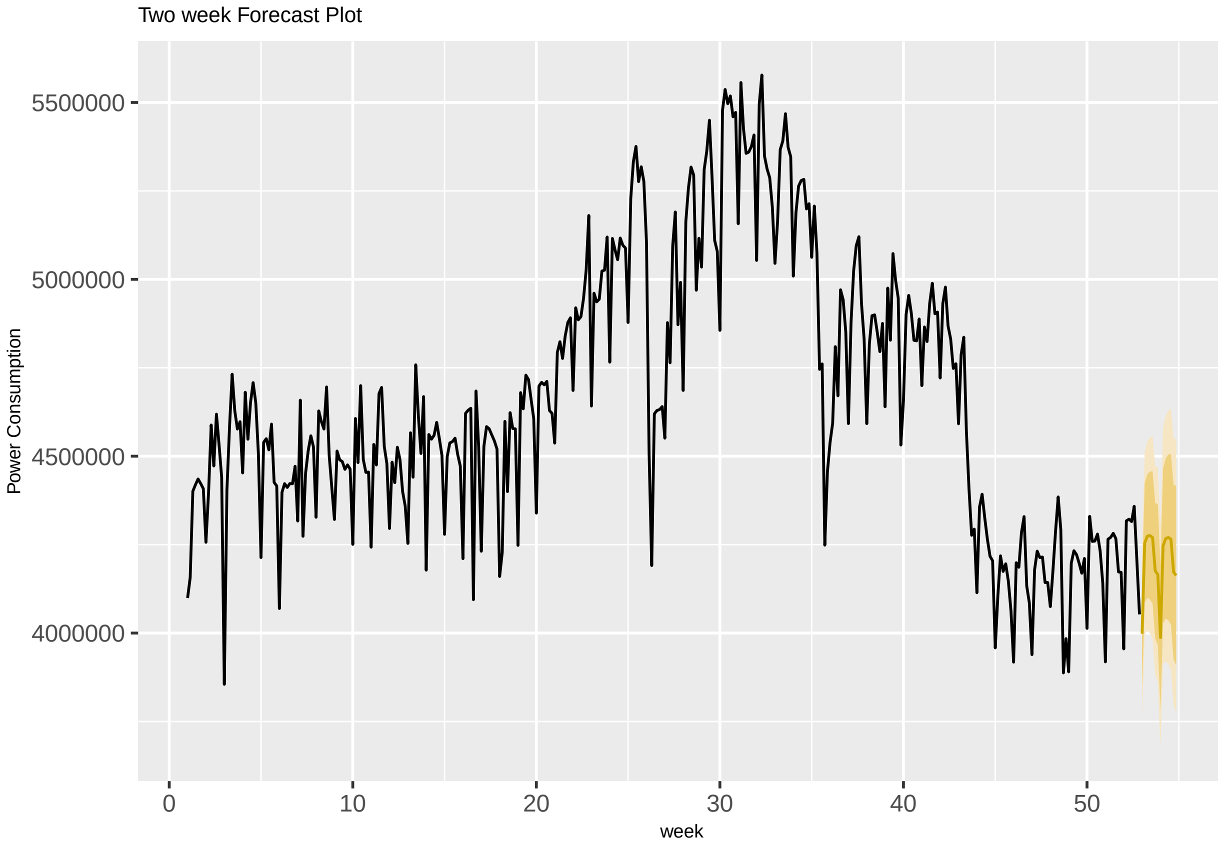






Provide two weeks future forecast

```
fc.period = 14
#Convert full data to ts
y1.ts <- ts(y1, start = c(1, 1), end= c(52, 7), frequency = 7)
full.holt.model <- ets(y1.ts, model = "ZAA", alpha = .2, gamma = .05)
full.holt.model.pred <- forecast(full.holt.model, h = fc.period)
full.holt.model.pred.ts <- ts(full.holt.model.pred$mean, start = c(45, 1), end= c(52, 7), frequency = 7
    autoplot(y1.ts) +
    autolayer( full.holt.model.pred, color = 'orange')+
    labs(title = "Two week Forecast Plot",x = "week", y = "Power Consumption")+
    theme (title =element_text(size=7))
```




```
#Forecast Value
full.holt.model.pred$mean
```

```
## Time Series:
## Start = c(53, 1)
## End = c(54, 7)
## Frequency = 7
##  [1] 3998163 4254857 4274085 4275799 4269856 4176025 4165649 3987405 4246251
## [10] 4267200 4270291 4265449 4172500 4162828
```

```
Zone1 <- c(
  format(as.integer((zone1.mean[[1]])[2]), scientific = FALSE),
  format(as.integer((zone1.naive[[1]])[2]), scientific = FALSE),
  format(as.integer((zone1.snaive[[1]])[2]), scientific = FALSE),
  format(as.integer((zone1.holt[[1]])[4]), scientific = FALSE),
  format(as.integer((zone1.tslm[[1]])[4]), scientific = FALSE),
  format(as.integer((zone1.myArima[[1]])[4]), scientific = FALSE),
  format(as.integer((zone1.nnetar[[1]])[4]), scientific = FALSE))
Zone2 <- c(
  format(as.integer((zone2.mean[[1]])[2]), scientific = FALSE),
  format(as.integer((zone2.naive[[1]])[2]), scientific = FALSE),
  format(as.integer((zone2.snaive[[1]])[2]), scientific = FALSE),
  format(as.integer((zone2.holt[[1]])[4]), scientific = FALSE),
  format(as.integer((zone2.tslm[[1]])[4]), scientific = FALSE),
  format(as.integer((zone2.myArima[[1]])[4]) , scientific = FALSE),
  format(as.integer((zone2.nnetar[[1]])[4]) , scientific = FALSE))
```

```r
Zone3 <- c(
  format(as.integer((zone3.mean[[1]])[2]), scientific = FALSE),
  format(as.integer((zone3.naive[[1]])[2]), scientific = FALSE),
  format(as.integer((zone3.snaive[[1]])[2]), scientific = FALSE),
  format(as.integer((zone3.holt[[1]])[4]), scientific = FALSE),
  format(as.integer((zone3.tslm[[1]])[4]), scientific = FALSE),
  format(as.integer((zone3.myArima[[1]])[4]), scientific = FALSE),
  format(as.integer((zone3.nnetar[[1]])[4]), scientific = FALSE))
Models <- c("Mean", "Naive", "Seasonal Naive", "Holt's Winter", "TSLM", "Arima", "Neural Network" )
```

```r
data.frame(Models, Zone1, Zone2, Zone3)
```

Generate a data frame for the result from all models

```
##           Models  Zone1   Zone2   Zone3
## 1           Mean 586831  441012 1028632
## 2          Naive 128921  185343  231569
## 3 Seasonal Naive 127311  125935  224689
## 4  Holt's Winter  76833  151507  238432
## 5           TSLM 298700 1064187 2701385
## 6          Arima  80930  841332 2498228
## 7 Neural Network 313655 1104931 2759897
```