Using Turnstile Data To Forcast Student Worker Staffing

Chad Madding, Shane Weinstock

April 11, 2020

### Clients

This report is being preparied for the management of the Dedman Center for Lifetime Sports at Southern Methodist University.

### Directive

We have been tasked with providing general useage of the facilities in the form of entry data. The main goal will be to use ID swipes as a response variable to assist in hyring patters for the facilities’ student staff. It is the goal of the report to assist in the spotting of trends and thus correctly identify staffing needs. This will help also with budgeting needs.

### Data

This is a time series data set collected from three turnstiles at the Dedman Center for Lifetime Sports on the Southern Methodist University campus. The data set is a record of the time of swipe, the turnstile used and an anonymized student ID number. This was collected from January 2nd, 2019 through March 11th, 2020 and consists of 414156 entries. Entry and error swipes are included in the data. It should be noted that the turnstiles are only used to enter the facility and ID swipe is not required to exit the building. We have also collected hourley weather data for the same time period.

**Read in the data**

## Warning: Duplicated column names deduplicated: 'REPORT\_TYPE' =>  
## 'REPORT\_TYPE\_1' [95], 'SOURCE' => 'SOURCE\_1' [96]

## Parsed with column specification:  
## cols(  
## .default = col\_logical(),  
## STATION = col\_double(),  
## DATE = col\_datetime(format = ""),  
## REPORT\_TYPE = col\_character(),  
## SOURCE = col\_character(),  
## DailyWeather = col\_character(),  
## HourlyAltimeterSetting = col\_double(),  
## HourlyDewPointTemperature = col\_double(),  
## HourlyDryBulbTemperature = col\_double(),  
## HourlyPrecipitation = col\_double(),  
## HourlyPresentWeatherType = col\_character(),  
## HourlyRelativeHumidity = col\_double(),  
## HourlySkyConditions = col\_character(),  
## HourlyStationPressure = col\_double(),  
## HourlyVisibility = col\_double(),  
## HourlyWetBulbTemperature = col\_double(),  
## HourlyWindDirection = col\_character(),  
## HourlyWindGustSpeed = col\_character(),  
## HourlyWindSpeed = col\_double(),  
## REM = col\_character(),  
## REPORT\_TYPE\_1 = col\_character()  
## # ... with 3 more columns  
## )

## See spec(...) for full column specifications.

## Warning: 57 parsing failures.  
## row col expected actual file  
## 1258 HourlyVisibility a double \* '2085029.csv'  
## 1535 HourlyDewPointTemperature no trailing characters s '2085029.csv'  
## 1536 HourlyDewPointTemperature no trailing characters s '2085029.csv'  
## 1537 HourlyDewPointTemperature no trailing characters s '2085029.csv'  
## 1999 HourlyVisibility a double \* '2085029.csv'  
## .... ......................... ...................... ...... .............  
## See problems(...) for more details.

**Data Cleaning**

Compactly Display the Structure of the data.

We had to clean up the hours for the merge.

#Group hours  
SMUSwipe$TempTime <- format(SMUSwipe$LDT,"%Y-%m-%d")  
SMUSwipe$Hour <-ifelse(SMUSwipe$Hour <= '04','04',  
 ifelse(SMUSwipe$Hour <= '08','08',  
 ifelse(SMUSwipe$Hour <= 12, 12,  
 ifelse(SMUSwipe$Hour <= 16, 16,  
 ifelse(SMUSwipe$Hour <= 20, 20,  
 ifelse(SMUSwipe$Hour <= 24, 24))))))  
SMUSwipe$TempTime<-paste(SMUSwipe$TempTime, SMUSwipe$Hour, sep=" ")

We had to clean up the hours for the merge.

#Group all ID swipes by 15 minutes (quarter hour)  
#Create a TempTime in HourlyWeather to merge Temperature data  
HourlyWeather$TempTime <- format(HourlyWeather$DATE,"%Y-%m-%d")  
HourlyWeather$Hour <- format(HourlyWeather$DATE,"%H")  
  
HourlyWeather$Hour <-ifelse(HourlyWeather$Hour <= '04','04',  
 ifelse(HourlyWeather$Hour <= '08','08',  
 ifelse(HourlyWeather$Hour <= 12, 12,  
 ifelse(HourlyWeather$Hour <= 16, 16,  
 ifelse(HourlyWeather$Hour <= 20, 20,  
 ifelse(HourlyWeather$Hour <= 24, 24))))))  
HourlyWeather$TempTime<-paste(HourlyWeather$TempTime, HourlyWeather$Hour, sep=" ")  
#Recreate the Hour data  
SMUSwipe$Hour <- format(SMUSwipe$LDT,"%H")

#Merge on TempTime to pull in hourly weather data  
SMUSwipe = merge(SMUSwipe, HourlyWeather, by.x='TempTime', by.y='TempTime',all.x = TRUE, all.y = TRUE)  
#Remove dup after the left join  
SMUSwipe = distinct(SMUSwipe, LDT, .keep\_all = TRUE)  
SMUSwipe = SMUSwipe[which(!is.na(SMUSwipe$`LDT`)),]  
  
#sum(is.na(SMUSwipe$HourlyDryBulbTemperature))  
  
#Fill in NA's from Temperature data with number close to the mean  
SMUSwipe$HourlyDryBulbTemperature[is.na(SMUSwipe$HourlyDryBulbTemperature)] <- 65  
SMUSwipe$HourlyAltimeterSetting[is.na(SMUSwipe$HourlyAltimeterSetting)] <- 30  
SMUSwipe$HourlyDewPointTemperature[is.na(SMUSwipe$HourlyDewPointTemperature)] <- 49  
SMUSwipe$HourlyRelativeHumidity[is.na(SMUSwipe$HourlyRelativeHumidity)] <- 58  
SMUSwipe$HourlyWindSpeed[is.na(SMUSwipe$HourlyWindSpeed)] <- 0  
  
#Renaming some information for ease of use  
SMUSwipe$Temperature <- SMUSwipe$HourlyDryBulbTemperature  
SMUSwipe$Hour <- SMUSwipe$Hour.x  
  
#Dropping some data we no longer need  
drop <- c("Message Type","Hour.x","Hour.y", "TempTime", "DATE", "HourlyPressureChange", "HourlyPressureTendency", "HourlySeaLevelPressure", "HourlyWetBulbTemperature", "HourlyDryBulbTemperature")  
SMUSwipe = SMUSwipe[,!(names(SMUSwipe) %in% drop)]

Create and export hourley datasets for time series studies

#Read in the data  
SMU = read.csv('SMUSwipe.csv',header = TRUE)  
#Look at the top of the data  
  
HourSwipes = dplyr::count(SMU,Hours)  
  
HourSwipesTemp = merge(HourSwipes,SMU, by='Hours')  
HourSwipesTemp = distinct(HourSwipesTemp, Hours, .keep\_all = TRUE)  
  
colnames(HourSwipesTemp)[2] = "IDSwipes"  
  
#Dropping some data we no longer need  
drop <- c("Secondary.Object.Name", "LDT", "ID.", "Time", "Minutes", "Minutes\_15", "interval\_15")  
HourSwipesTemp = HourSwipesTemp[,!(names(HourSwipesTemp) %in% drop)]  
  
#Export the dataset  
write.csv(HourSwipesTemp,"DedmanHourlySwipe.csv", row.names = FALSE)  
  
head(HourSwipesTemp)

## Hours IDSwipes Day Date Month HourlyAltimeterSetting  
## 1 2019-01-02 06 11 Wednesday 2019-01-02 2019-01 30.23  
## 2 2019-01-02 07 12 Wednesday 2019-01-02 2019-01 30.23  
## 3 2019-01-02 08 7 Wednesday 2019-01-02 2019-01 30.23  
## 4 2019-01-02 09 12 Wednesday 2019-01-02 2019-01 30.27  
## 5 2019-01-02 10 11 Wednesday 2019-01-02 2019-01 30.27  
## 6 2019-01-02 11 20 Wednesday 2019-01-02 2019-01 30.27  
## HourlyDewPointTemperature HourlyPrecipitation HourlyPresentWeatherType  
## 1 30 NA -RA:02 |RA |RA  
## 2 30 NA -RA:02 |RA |RA  
## 3 30 NA -RA:02 |RA |RA  
## 4 32 NA -RA:02 |RA |RA  
## 5 32 NA -RA:02 |RA |RA  
## 6 32 NA -RA:02 |RA |RA  
## HourlyRelativeHumidity HourlySkyConditions HourlyStationPressure  
## 1 93 OVC:08 7 29.54  
## 2 93 OVC:08 7 29.54  
## 3 93 OVC:08 7 29.54  
## 4 93 OVC:08 5 29.57  
## 5 93 OVC:08 5 29.57  
## 6 93 OVC:08 5 29.57  
## HourlyVisibility HourlyWindDirection HourlyWindGustSpeed HourlyWindSpeed  
## 1 4 020 18 13  
## 2 4 020 18 13  
## 3 4 020 18 13  
## 4 2 350 <NA> 9  
## 5 2 350 <NA> 9  
## 6 2 350 <NA> 9  
## Temperature Hour WeekdayCount  
## 1 32 6 2019-01-02 Wednesday  
## 2 32 7 2019-01-02 Wednesday  
## 3 32 8 2019-01-02 Wednesday  
## 4 34 9 2019-01-02 Wednesday  
## 5 34 10 2019-01-02 Wednesday  
## 6 34 11 2019-01-02 Wednesday

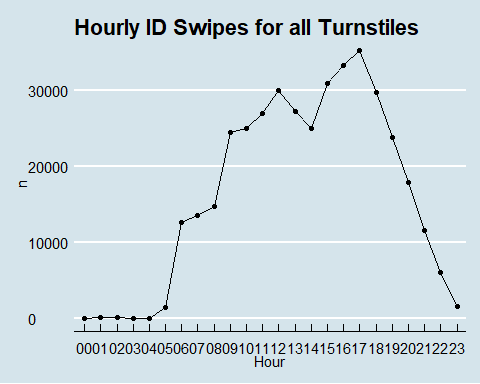
We now have a dataset with weather data along with card swipes.

head(SMUSwipe)

## LDT ID# Secondary Object Name Time Day Date  
## 2 2019-01-02 07:14:08 1751 DEDM (101.2LB) 07:14:08 Wednesday 2019-01-02  
## 3 2019-01-02 07:20:53 6472 DEDM (101.2LB) 07:20:53 Wednesday 2019-01-02  
## 4 2019-01-02 07:11:36 4341 DEDM (101.3LB) 07:11:36 Wednesday 2019-01-02  
## 5 2019-01-02 06:16:37 247 DEDM (101.3LB) 06:16:37 Wednesday 2019-01-02  
## 6 2019-01-02 07:08:26 1752 DEDM (101.2LB) 07:08:26 Wednesday 2019-01-02  
## 7 2019-01-02 07:08:32 2515 DEDM (101.3LB) 07:08:32 Wednesday 2019-01-02  
## Hours Month HourlyAltimeterSetting HourlyDewPointTemperature  
## 2 2019-01-02 07 2019-01 30.23 30  
## 3 2019-01-02 07 2019-01 30.23 30  
## 4 2019-01-02 07 2019-01 30.23 30  
## 5 2019-01-02 06 2019-01 30.23 30  
## 6 2019-01-02 07 2019-01 30.23 30  
## 7 2019-01-02 07 2019-01 30.23 30  
## HourlyPrecipitation HourlyPresentWeatherType HourlyRelativeHumidity  
## 2 NA -RA:02 |RA |RA 93  
## 3 NA -RA:02 |RA |RA 93  
## 4 NA -RA:02 |RA |RA 93  
## 5 NA -RA:02 |RA |RA 93  
## 6 NA -RA:02 |RA |RA 93  
## 7 NA -RA:02 |RA |RA 93  
## HourlySkyConditions HourlyStationPressure HourlyVisibility  
## 2 OVC:08 7 29.54 4  
## 3 OVC:08 7 29.54 4  
## 4 OVC:08 7 29.54 4  
## 5 OVC:08 7 29.54 4  
## 6 OVC:08 7 29.54 4  
## 7 OVC:08 7 29.54 4  
## HourlyWindDirection HourlyWindGustSpeed HourlyWindSpeed Temperature Hour  
## 2 020 18 13 32 07  
## 3 020 18 13 32 07  
## 4 020 18 13 32 07  
## 5 020 18 13 32 06  
## 6 020 18 13 32 07  
## 7 020 18 13 32 07

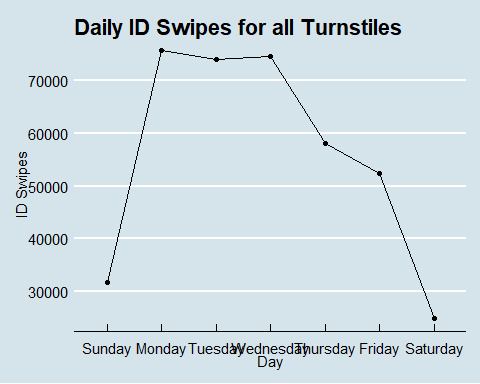
Hourly ID Swipes for all Turnstiles?

#Group all badge swipes by the hour  
  
hourplot = dplyr::count(SMUSwipe,Hour)  
  
hourplot %>%  
 ggplot(aes(x=Hour,y=n, group=1))+  
 geom\_line()+  
 geom\_point()+  
 theme\_economist()+  
 scale\_colour\_economist()+  
 ggtitle('Hourly ID Swipes for all Turnstiles')



What are the Daily ID Swipes for all Turnstiles?

dayCount=dplyr::count(SMUSwipe, Day)  
dayCount$Day <- factor(dayCount$Day, levels= c("Sunday", "Monday",   
 "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))  
  
dayCount=dayCount[order(dayCount$Day), ]  
   
dayCount %>%  
 ggplot(aes(x=Day,y=n, group=1))+  
 geom\_line()+  
 geom\_point()+  
 theme\_economist()+  
 scale\_colour\_economist()+  
 theme\_economist()+  
 ggtitle('Daily ID Swipes for all Turnstiles')+  
 ylab('ID Swipes')



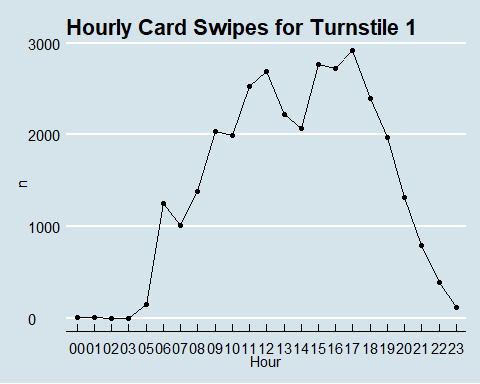
We want to group all ID swipes by 15 minutes (quarter hour).

#Group all ID swipes by 15 minutes (quarter hour)  
SMUSwipe$Minutes <- format(SMUSwipe$LDT,"%M")  
SMUSwipe$interval\_15 <-ifelse(SMUSwipe$Minutes <= 15, 1, ifelse(SMUSwipe$Minutes <= 30, 2, ifelse(SMUSwipe$Minutes <= 45, 3, ifelse(SMUSwipe$Minutes <= 60, 4))))  
SMUSwipe$Minutes\_15<-paste(SMUSwipe$Hours, SMUSwipe$interval\_15, sep=" ")

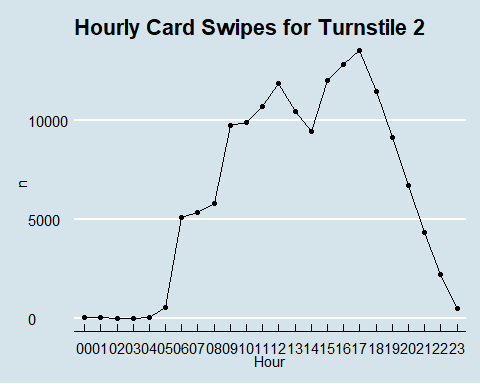
Look at turnstile information.  
Create indivual turnstile data

#Create a dataset for each turnstyle  
SMUSwipeTurn1 = SMUSwipe[SMU[,3]=="DEDM (101.1LB)",]  
SMUSwipeTurn2 = SMUSwipe[SMU[,3]=="DEDM (101.2LB)",]  
SMUSwipeTurn3 = SMUSwipe[SMU[,3]=="DEDM (101.3LB)",]

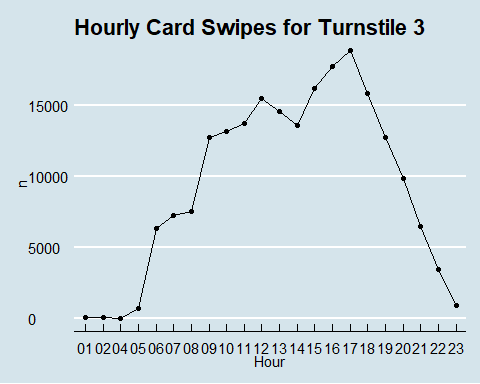
df = dplyr::count(SMUSwipeTurn3, Hours)  
  
HTurn1 = dplyr::count(SMUSwipeTurn1, Hour)  
HTurn1 %>%  
 ggplot(aes(x=Hour,y=n, group=1))+  
 geom\_line()+  
 geom\_point()+  
 theme\_economist()+  
 scale\_colour\_economist()+  
 ggtitle('Hourly Card Swipes for Turnstile 1')



HTurn2 = dplyr::count(SMUSwipeTurn2, Hour)  
HTurn2 %>%  
 ggplot(aes(x=Hour,y=n, group=1))+  
 geom\_line()+  
 geom\_point()+  
 theme\_economist()+  
 scale\_colour\_economist()+  
 ggtitle('Hourly Card Swipes for Turnstile 2')



HTurn3 = dplyr::count(SMUSwipeTurn3, Hour)  
HTurn3 %>%  
 ggplot(aes(x=Hour,y=n, group=1))+  
 geom\_line()+  
 geom\_point()+  
 theme\_economist()+  
 scale\_colour\_economist()+  
 ggtitle('Hourly Card Swipes for Turnstile 3')



Looks like turnstile 3 gets used more than the others but the pattern of usage looks the same accross each.

This data is anamonized but we can still see who are some of the top users.

#This will count up the times users swiped in.   
IDCount = dplyr::count(SMUSwipe, `ID#`)  
#Print off the top 10 users ordering nuency in decending order  
head(IDCount[order(IDCount$n, decreasing = TRUE),],10)

## # A tibble: 10 x 2  
## `ID#` n  
## <dbl> <int>  
## 1 1445 429  
## 2 1168 413  
## 3 1429 390  
## 4 1469 369  
## 5 1140 361  
## 6 912 354  
## 7 1778 349  
## 8 104 339  
## 9 152 338  
## 10 214 335

Look at some user data.

#Breakdown user numbers  
summary(IDCount$n)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.00 7.00 21.00 35.99 48.00 429.00

#how many users  
dplyr::count(IDCount)

## # A tibble: 1 x 1  
## n  
## <int>  
## 1 10873

Convert the hourly data to time series.

#Hourley data  
#AnonDataTurnstile <- ts(df$n, frequency=6195, start=c(2019,1), end=c(2020,3))  
AnonDataTurnstile = ts(df$n)  
head(AnonDataTurnstile)

## Time Series:  
## Start = 1   
## End = 6   
## Frequency = 1   
## [1] 5 3 3 4 6 9

Create different data sets for forcasting.

#### Some information on frequency from the web

<http://manishbarnwal.com/blog/2017/05/03/time_series_and_forecasting_using_R/>

**Daily** data There could be a weekly cycle or annual cycle. So the frequency could be 7 or 365.25.

Some of the years have 366 days (leap years). So if your time series data has longer periods, it is better to use frequency = 365.25. This takes care of the leap year as well which may come in your data.

**Weekly** data There could be an annual cycle. frequency = 52 and if you want to take care of leap years then use frequency = 365.25/7

**Monthly** data Cycle is of one year. So frequency = 12

**Quarterly** data Again cycle is of one year. So frequency = 4

**Yearly** data Frequency = 1

#### How about frequency for smaller interval time series

**Hourly** The cycles could be a day, a week, a year. Corresponding frequencies could be 24, 24 X 7, 24 X 7 X 365.25

**Half-hourly** The cycle could be a day, a week, a year. Corresponding frequencies could be 48, 48 X 7, 48 X 7 X 365.25

**Minutes** The cycle could be hourly, daily, weekly, annual. Corresponding frequencies would be 60, 60 X 24, 60 X 24 X 7, 60 X 24 X 365.25

**Seconds** The cycle could be a minute, hourly, daily, weekly, annual. Corresponding frequencies would be 60, 60 X 60, 60 X 60 X 24, 60 X 60 X 24 X 7, 60 X 60 X 24 X 365.25

MonthCount <- dplyr::count(SMUSwipe, Month)  
MonthCount <- ts(MonthCount$n, start = c(2019,1), end = c(2020,3), frequency = 12)  
MonthCount

## Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec  
## 2019 26754 39223 32565 37192 17963 12281 11419 20547 41423 34837 28224 15030  
## 2020 24732 35541 13547

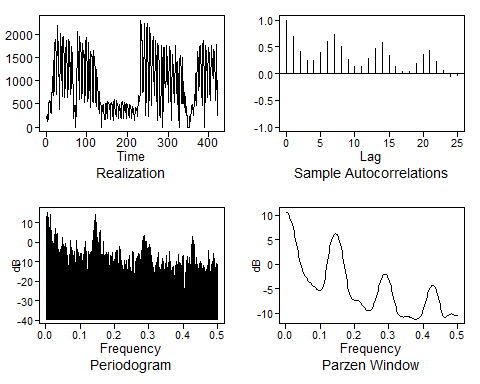
DayCount <- dplyr::count(SMUSwipe, Date)  
DayCount <- ts(DayCount$n, start = c(2019,2), frequency = 365.25)  
DayCount

## Time Series:  
## Start = 2019.00273785079   
## End = 2020.15263518138   
## Frequency = 365.25   
## [1] 235 123 267 172 156 480 584 570 534 552 342 290 705 679 816  
## [16] 752 1439 684 714 768 1783 1882 1642 1619 709 868 2180 1783 1838 1588  
## [31] 1548 642 385 1904 1997 2018 1469 1328 652 953 1924 1947 1809 1396 1422  
## [46] 638 1005 1722 1525 1861 1434 1332 635 952 1895 1677 1853 1300 1382 695  
## [61] 942 1727 1626 1597 1072 804 328 184 452 471 466 1 460 236 456  
## [76] 2053 1823 1723 1557 1416 601 756 1956 1709 1798 1424 1436 607 807 1790  
## [91] 1509 1650 1488 1351 541 795 1857 1780 1939 1445 1407 551 653 1821 1552  
## [106] 1395 1148 427 458 1626 1516 1525 1526 1264 575 522 1659 1422 1514 1341  
## [121] 1283 590 455 1233 928 794 885 683 465 296 759 595 486 533 439  
## [136] 3 186 463 360 492 422 391 213 166 169 517 424 510 368 221  
## [151] 186 543 536 506 483 454 284 166 492 523 520 446 480 284 151  
## [166] 538 604 538 476 345 286 239 542 582 497 481 429 266 183 490  
## [181] 457 382 247 217 196 517 549 501 458 450 178 518 537 496 465  
## [196] 424 181 511 478 463 437 371 300 206 486 428 476 393 376 240  
## [211] 196 406 438 385 360 313 205 152 419 387 375 396 387 327 240  
## [226] 667 671 678 616 554 787 688 2300 1855 2204 1548 1228 756 774 809  
## [241] 2177 2225 1629 1336 249 768 2134 1977 2221 1532 1261 220 907 2144 1811  
## [256] 2014 1416 1243 459 764 1852 1787 1942 1363 1141 842 685 1741 1797 1866  
## [271] 1248 955 154 609 1803 1641 1763 1158 632 284 209 585 715 1756 1259  
## [286] 1004 1 631 1710 1702 1723 1005 846 986 988 1760 1569 1499 979 833  
## [301] 527 650 1739 1480 1658 1136 945 141 666 1495 1505 1555 1165 920 537  
## [316] 566 1747 1587 1656 1166 876 561 484 1417 925 286 1 337 1440 1550  
## [331] 1511 1131 891 508 508 1506 505 675 724 756 534 324 667 555 390  
## [346] 261 245 3 2 3 2 2 223 217 161 131 446 461 485 489  
## [361] 442 290 244 590 675 709 575 1164 696 466 743 1630 1589 1432 1258  
## [376] 782 913 1734 1732 1751 1403 1301 784 471 1760 1871 1686 1447 1208 2  
## [391] 1051 1756 1637 1656 1307 1103 633 1057 1719 1660 1569 1289 834 581 944  
## [406] 1631 1752 1667 1316 1150 1048 1616 1663 1620 1308 1078 584 1036 1558 1786  
## [421] 250

SMUSwipe$WeekdayCount<-paste(SMUSwipe$Date, SMUSwipe$Day, sep=" ")  
WeekdayCount <- dplyr::count(SMUSwipe, WeekdayCount)  
WeekdayCount <- ts(WeekdayCount$n)  
WeekdayCount

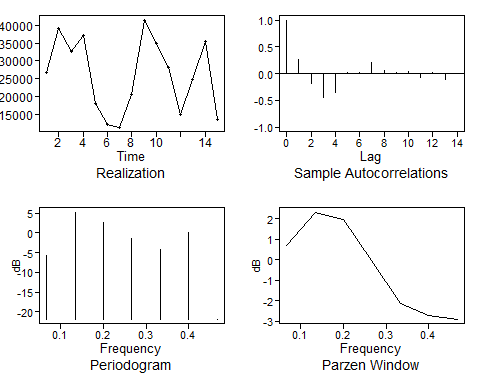
## Time Series:  
## Start = 1   
## End = 421   
## Frequency = 1   
## [1] 235 123 267 172 156 480 584 570 534 552 342 290 705 679 816  
## [16] 752 1439 684 714 768 1783 1882 1642 1619 709 868 2180 1783 1838 1588  
## [31] 1548 642 385 1904 1997 2018 1469 1328 652 953 1924 1947 1809 1396 1422  
## [46] 638 1005 1722 1525 1861 1434 1332 635 952 1895 1677 1853 1300 1382 695  
## [61] 942 1727 1626 1597 1072 804 328 184 452 471 466 1 460 236 456  
## [76] 2053 1823 1723 1557 1416 601 756 1956 1709 1798 1424 1436 607 807 1790  
## [91] 1509 1650 1488 1351 541 795 1857 1780 1939 1445 1407 551 653 1821 1552  
## [106] 1395 1148 427 458 1626 1516 1525 1526 1264 575 522 1659 1422 1514 1341  
## [121] 1283 590 455 1233 928 794 885 683 465 296 759 595 486 533 439  
## [136] 3 186 463 360 492 422 391 213 166 169 517 424 510 368 221  
## [151] 186 543 536 506 483 454 284 166 492 523 520 446 480 284 151  
## [166] 538 604 538 476 345 286 239 542 582 497 481 429 266 183 490  
## [181] 457 382 247 217 196 517 549 501 458 450 178 518 537 496 465  
## [196] 424 181 511 478 463 437 371 300 206 486 428 476 393 376 240  
## [211] 196 406 438 385 360 313 205 152 419 387 375 396 387 327 240  
## [226] 667 671 678 616 554 787 688 2300 1855 2204 1548 1228 756 774 809  
## [241] 2177 2225 1629 1336 249 768 2134 1977 2221 1532 1261 220 907 2144 1811  
## [256] 2014 1416 1243 459 764 1852 1787 1942 1363 1141 842 685 1741 1797 1866  
## [271] 1248 955 154 609 1803 1641 1763 1158 632 284 209 585 715 1756 1259  
## [286] 1004 1 631 1710 1702 1723 1005 846 986 988 1760 1569 1499 979 833  
## [301] 527 650 1739 1480 1658 1136 945 141 666 1495 1505 1555 1165 920 537  
## [316] 566 1747 1587 1656 1166 876 561 484 1417 925 286 1 337 1440 1550  
## [331] 1511 1131 891 508 508 1506 505 675 724 756 534 324 667 555 390  
## [346] 261 245 3 2 3 2 2 223 217 161 131 446 461 485 489  
## [361] 442 290 244 590 675 709 575 1164 696 466 743 1630 1589 1432 1258  
## [376] 782 913 1734 1732 1751 1403 1301 784 471 1760 1871 1686 1447 1208 2  
## [391] 1051 1756 1637 1656 1307 1103 633 1057 1719 1660 1569 1289 834 581 944  
## [406] 1631 1752 1667 1316 1150 1048 1616 1663 1620 1308 1078 584 1036 1558 1786  
## [421] 250

plotts.sample.wge(DayCount)



## $autplt  
## [1] 1.00000000 0.70215951 0.42324532 0.25110163 0.24537802 0.38890810  
## [7] 0.59940311 0.73274869 0.50042359 0.26253004 0.13417670 0.13517610  
## [13] 0.28431957 0.47112224 0.57721389 0.34840851 0.13815428 0.03215917  
## [19] 0.04316582 0.19659812 0.35731162 0.43329390 0.21962520 0.05292005  
## [25] -0.04562156 -0.03182862  
##   
## $freq  
## [1] 0.002375297 0.004750594 0.007125891 0.009501188 0.011876485 0.014251781  
## [7] 0.016627078 0.019002375 0.021377672 0.023752969 0.026128266 0.028503563  
## [13] 0.030878860 0.033254157 0.035629454 0.038004751 0.040380048 0.042755344  
## [19] 0.045130641 0.047505938 0.049881235 0.052256532 0.054631829 0.057007126  
## [25] 0.059382423 0.061757720 0.064133017 0.066508314 0.068883610 0.071258907  
## [31] 0.073634204 0.076009501 0.078384798 0.080760095 0.083135392 0.085510689  
## [37] 0.087885986 0.090261283 0.092636580 0.095011876 0.097387173 0.099762470  
## [43] 0.102137767 0.104513064 0.106888361 0.109263658 0.111638955 0.114014252  
## [49] 0.116389549 0.118764846 0.121140143 0.123515439 0.125890736 0.128266033  
## [55] 0.130641330 0.133016627 0.135391924 0.137767221 0.140142518 0.142517815  
## [61] 0.144893112 0.147268409 0.149643705 0.152019002 0.154394299 0.156769596  
## [67] 0.159144893 0.161520190 0.163895487 0.166270784 0.168646081 0.171021378  
## [73] 0.173396675 0.175771971 0.178147268 0.180522565 0.182897862 0.185273159  
## [79] 0.187648456 0.190023753 0.192399050 0.194774347 0.197149644 0.199524941  
## [85] 0.201900238 0.204275534 0.206650831 0.209026128 0.211401425 0.213776722  
## [91] 0.216152019 0.218527316 0.220902613 0.223277910 0.225653207 0.228028504  
## [97] 0.230403800 0.232779097 0.235154394 0.237529691 0.239904988 0.242280285  
## [103] 0.244655582 0.247030879 0.249406176 0.251781473 0.254156770 0.256532067  
## [109] 0.258907363 0.261282660 0.263657957 0.266033254 0.268408551 0.270783848  
## [115] 0.273159145 0.275534442 0.277909739 0.280285036 0.282660333 0.285035629  
## [121] 0.287410926 0.289786223 0.292161520 0.294536817 0.296912114 0.299287411  
## [127] 0.301662708 0.304038005 0.306413302 0.308788599 0.311163895 0.313539192  
## [133] 0.315914489 0.318289786 0.320665083 0.323040380 0.325415677 0.327790974  
## [139] 0.330166271 0.332541568 0.334916865 0.337292162 0.339667458 0.342042755  
## [145] 0.344418052 0.346793349 0.349168646 0.351543943 0.353919240 0.356294537  
## [151] 0.358669834 0.361045131 0.363420428 0.365795724 0.368171021 0.370546318  
## [157] 0.372921615 0.375296912 0.377672209 0.380047506 0.382422803 0.384798100  
## [163] 0.387173397 0.389548694 0.391923990 0.394299287 0.396674584 0.399049881  
## [169] 0.401425178 0.403800475 0.406175772 0.408551069 0.410926366 0.413301663  
## [175] 0.415676960 0.418052257 0.420427553 0.422802850 0.425178147 0.427553444  
## [181] 0.429928741 0.432304038 0.434679335 0.437054632 0.439429929 0.441805226  
## [187] 0.444180523 0.446555819 0.448931116 0.451306413 0.453681710 0.456057007  
## [193] 0.458432304 0.460807601 0.463182898 0.465558195 0.467933492 0.470308789  
## [199] 0.472684086 0.475059382 0.477434679 0.479809976 0.482185273 0.484560570  
## [205] 0.486935867 0.489311164 0.491686461 0.494061758 0.496437055 0.498812352  
##   
## $db  
## [1] 11.2847941 15.5067474 10.9560916 1.6235540 1.3568184 14.5778854  
## [7] -1.7089159 0.4505457 -4.9269921 4.0768507 -6.7571705 6.9980703  
## [13] 1.8856330 4.5599246 -12.4918372 -3.4611109 -6.3136038 -3.7522049  
## [19] -6.3074071 2.4000345 0.3072035 -2.1692535 -2.9897525 -0.4679117  
## [25] -4.3862522 -8.3588419 -10.7701306 -12.0698076 -13.4045551 0.8930276  
## [31] -5.3123233 -5.5995212 -3.9852508 -8.1546005 -4.7983107 -15.1198330  
## [37] -9.9150998 0.7183291 -7.1743432 -5.1904638 -8.1960083 -9.5271481  
## [43] -5.5103254 -5.4647008 -11.9901686 -7.2767437 -5.0568756 0.2748102  
## [49] -9.6172227 -4.4328088 -23.3225437 -4.6634461 -6.6158486 -6.8069861  
## [55] -5.3965427 -8.5961329 -0.2372156 2.3971323 9.7161980 8.5685465  
## [61] 14.4783780 6.6187684 -0.9522843 -1.1588099 -3.4709094 3.9475735  
## [67] 6.0617632 -17.5739314 -13.4936035 -12.6910751 0.6757239 -12.9278483  
## [73] -6.8159710 -5.3346167 -6.0899327 -8.0003709 -5.0158509 -18.1907995  
## [79] -6.9752083 -16.8785168 -9.2732758 -5.2093852 -13.5045068 -10.7254471  
## [85] -6.2925537 -4.2256425 -13.5303404 -4.0379063 -19.0080250 -2.2513431  
## [91] -4.3996792 -17.5066753 -9.2492975 -13.4326787 -17.9906754 -9.2559905  
## [97] -12.3008952 -13.7671283 -9.4383037 -17.6742159 -10.3950188 -6.5224114  
## [103] -11.6227410 -14.0191163 -24.7762648 -4.7952393 -4.9557287 -14.6200889  
## [109] -20.1083452 -11.4214865 -8.1076372 -8.9048028 -5.7974599 -14.6009092  
## [115] -6.2891288 -13.5428761 -4.1323042 -20.3909451 -0.4578085 2.5489357  
## [121] 3.4435578 -2.1051302 0.3207886 -5.6134128 -13.0799918 -4.3984414  
## [127] -1.1786120 -3.3737642 -9.2943776 -15.1260881 -10.5360892 -15.1166896  
## [133] -16.7707860 -13.5038571 -10.2410752 -13.1601035 -10.5305182 -5.1459030  
## [139] -20.3366947 -13.7442660 -19.7043546 -14.1994119 -8.4586252 -10.9297475  
## [145] -16.6755830 -13.8428736 -15.1468545 -6.7014550 -18.7666130 -15.2491490  
## [151] -4.6903388 -12.0292788 -9.6304152 -18.3902396 -13.0355382 -25.5831583  
## [157] -12.2283692 -21.3118864 -13.5408760 -9.9327649 -12.6322285 -18.2231393  
## [163] -6.2958947 -13.4669444 -10.7223127 -16.5341138 -7.2354109 -17.2048177  
## [169] -6.5067185 -39.4744135 -19.7014171 -11.2751862 -7.3502282 -10.9776086  
## [175] -14.1029391 -7.5526964 -21.4722086 -1.3051011 -11.9124106 3.0672540  
## [181] -0.4598960 -1.1336313 -20.6719350 -7.8854835 -13.2778123 -11.3166831  
## [187] -15.1852807 -11.1185861 -9.6325735 -12.3494982 -10.9031888 -15.0914169  
## [193] -11.7931954 -12.8928300 -18.1943385 -15.6051038 -13.7288878 -19.6565613  
## [199] -4.2342780 -12.8377287 -14.5505371 -18.2215718 -11.3785099 -13.5070947  
## [205] -5.5170419 -11.9160412 -8.5389449 -17.0406682 -10.7968233 -11.4887658  
##   
## $dbz  
## [1] 10.6009371 10.4779115 10.2731478 9.9871285 9.6206643 9.1750459  
## [7] 8.6522631 8.0553075 7.3885763 6.6583836 5.8735638 5.0460924  
## [13] 4.1915612 3.3292119 2.4811318 1.6702670 0.9172899 0.2370638  
## [19] -0.3639313 -0.8879511 -1.3438897 -1.7437441 -2.0991614 -2.4191885  
## [25] -2.7095648 -2.9732968 -3.2119684 -3.4271828 -3.6216188 -3.7993767  
## [31] -3.9655621 -4.1253214 -4.2827028 -4.4397394 -4.5960646 -4.7492041  
## [37] -4.8954670 -5.0310547 -5.1527044 -5.2570089 -5.3377116 -5.3808915  
## [43] -5.3592411 -5.2288875 -4.9347526 -4.4291003 -3.6970366 -2.7697124  
## [49] -1.7126954 -0.6005614 0.5029075 1.5518326 2.5167813 3.3804126  
## [55] 4.1332990 4.7708318 5.2911768 5.6939979 5.9796896 6.1489309  
## [61] 6.2024405 6.1408655 5.9647644 5.6746728 5.2712561 4.7555727  
## [67] 4.1294897 3.3963139 2.5617209 1.6350658 0.6311055 -0.4280290  
## [73] -1.5111974 -2.5782263 -3.5833028 -4.4828515 -5.2459476 -5.8617083  
## [79] -6.3385821 -6.6963185 -6.9566276 -7.1378612 -7.2547277 -7.3208031  
## [85] -7.3509957 -7.3622789 -7.3725750 -7.3986834 -7.4542621 -7.5484386  
## [91] -7.6851409 -7.8629953 -8.0756261 -8.3122967 -8.5589484 -8.7997185  
## [97] -9.0189102 -9.2031358 -9.3430707 -9.4341710 -9.4759539 -9.4699684  
## [103] -9.4170850 -9.3149947 -9.1568229 -8.9316167 -8.6270848 -8.2341523  
## [109] -7.7517455 -7.1895759 -6.5674767 -5.9117652 -5.2505680 -4.6100033  
## [115] -4.0120489 -3.4739382 -3.0085086 -2.6249620 -2.3296902 -2.1269940  
## [121] -2.0196392 -2.0092417 -2.0964949 -2.2812509 -2.5624560 -2.9379314  
## [127] -3.4039738 -3.9547470 -4.5814427 -5.2712402 -6.0062131 -6.7625644  
## [133] -7.5108816 -8.2183006 -8.8530980 -9.3909180 -9.8201098 -10.1432392  
## [139] -10.3738602 -10.5305754 -10.6315462 -10.6914371 -10.7209351 -10.7278902  
## [145] -10.7189333 -10.7007261 -10.6804422 -10.6654763 -10.6626350 -10.6771428  
## [151] -10.7117491 -10.7661282 -10.8366932 -10.9169086 -10.9981509 -11.0710733  
## [157] -11.1272457 -11.1606248 -11.1682882 -11.1500040 -11.1065883 -11.0374653  
## [163] -10.9381934 -10.7988986 -10.6045846 -10.3380253 -9.9850511 -9.5405165  
## [169] -9.0121089 -8.4198936 -7.7919904 -7.1588588 -6.5486777 -5.9848691  
## [175] -5.4855019 -5.0638074 -4.7291254 -4.4878744 -4.3443572 -4.3013509  
## [181] -4.3604755 -4.5223518 -4.7865416 -5.1512368 -5.6126286 -6.1638496  
## [187] -6.7933716 -7.4828113 -8.2044081 -8.9191697 -9.5778708 -10.1278142  
## [193] -10.5262229 -10.7551207 -10.8278002 -10.7817021 -10.6634023 -10.5155318  
## [199] -10.3703686 -10.2489920 -10.1630723 -10.1171803 -10.1106922 -10.1391051  
## [205] -10.1949060 -10.2682455 -10.3476846 -10.4212219 -10.4776807 -10.5083206

plotts.sample.wge(MonthCount)

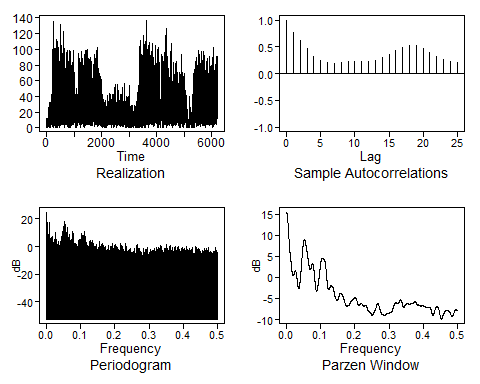


## $autplt  
## [1] 1.000000000 0.261454211 -0.188479130 -0.445834792 -0.342773654  
## [6] 0.024137505 0.015989290 0.207816302 0.052793951 0.028515320  
## [11] 0.035530323 -0.067522542 0.027542217 -0.103680277 -0.005488726  
##   
## $freq  
## [1] 0.06666667 0.13333333 0.20000000 0.26666667 0.33333333 0.40000000 0.46666667  
##   
## $db  
## [1] -5.80979143 5.22065184 2.50253719 -1.38490680 -4.21777975  
## [6] 0.08182939 -21.84593228  
##   
## $dbz  
## [1] 0.68763129 2.33094720 1.98370063 -0.04846235 -2.13675186 -2.73256000  
## [7] -2.90625332

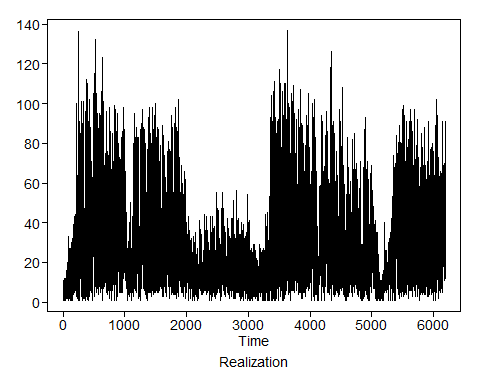
#### Visualize what we have so far.

**Plot the hourly data for all three turnstyles**

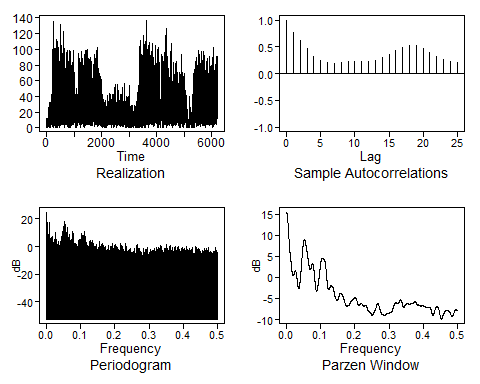
#Plot the hourly data  
TurnPlotHour=plotts.sample.wge(AnonDataTurnstile)



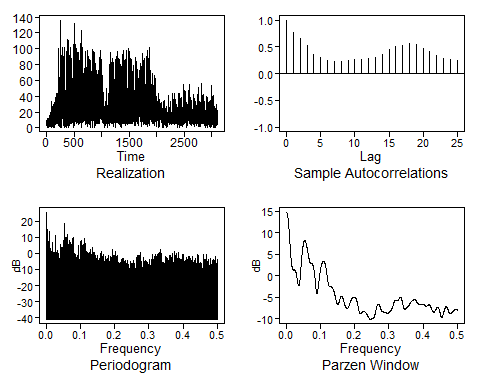
# Test for Conditions  
x = AnonDataTurnstile  
tswge::plotts.wge(x)



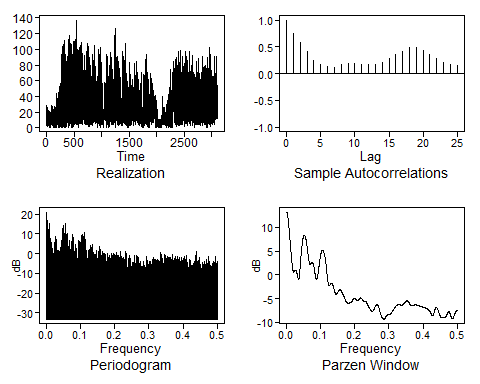
plotts.sample.wge(x)



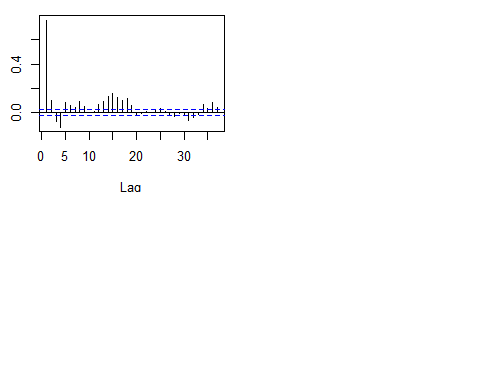
plotts.sample.wge(head(x,length(x)/2))



plotts.sample.wge(tail(x,length(x)/2))

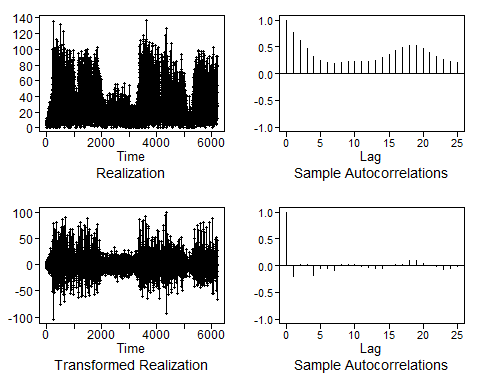


pacf(x)



Start ID’ing the data by using the Box and Jeakins method with first differcing the data

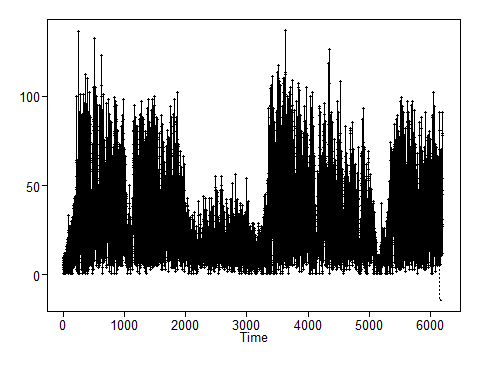
#Differcing the data to use the Box and Jeakins method  
AnonDataTurnstile\_diff=artrans.wge(AnonDataTurnstile,phi.tr=1)



p=4;q=4;s=0  
es=tswge::est.arma.wge(x,p=p,q=q)

##   
## Coefficients of Original polynomial:   
## 0.1191 0.1304 0.3401 -0.1073   
##   
## Factor Roots Abs Recip System Freq   
## 1+0.8694B+0.5241B^2 -0.8294+-1.1046i 0.7240 0.3525  
## 1-0.6930B 1.4431 0.6930 0.0000  
## 1-0.2955B 3.3846 0.2955 0.0000  
##   
##

ase = function(f,x){mean((f - tail(x,length(f)))^2)}  
m = tswge::fore.aruma.wge(x,phi = es$phi,theta = es$theta,s=s,n.ahead = 48,lastn = T)

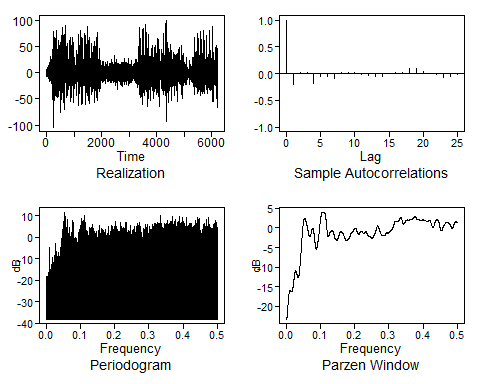


ase = ase(m$f,x)  
message("ASE is: ",ase)

## ASE is: 583.835392223577

ase1=ase

#Plot the differeaced hourly data - diff  
plotts.sample.wge(AnonDataTurnstile\_diff)



Use aic5.wge() to identify estimates of p and q on the differanced data.

#AIC of the hourly data  
aic5.wge(AnonDataTurnstile\_diff)

## ---------WORKING... PLEASE WAIT...   
##   
##   
## Five Smallest Values of aic

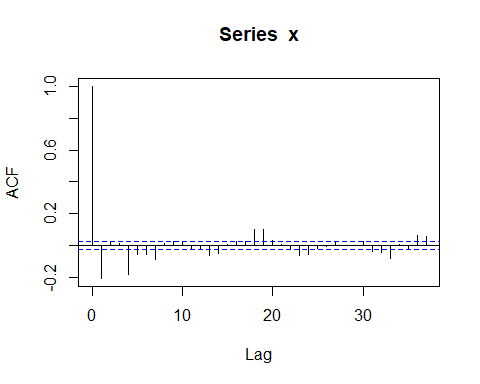
## p q aic  
## 14 4 1 5.412712  
## 15 4 2 5.413000  
## 17 5 1 5.413007  
## 18 5 2 5.413308  
## 12 3 2 5.418775

aic5.wge(AnonDataTurnstile\_diff, type = "bic")

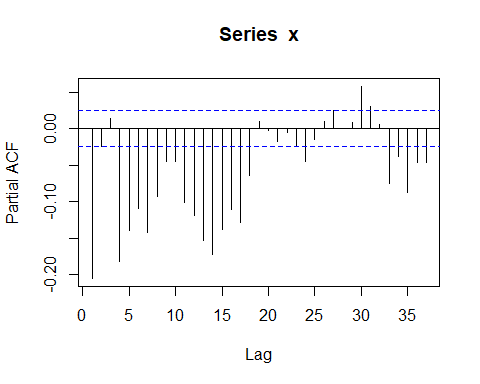
## ---------WORKING... PLEASE WAIT...   
##   
##   
## Five Smallest Values of bic

## p q bic  
## 14 4 1 5.419232  
## 15 4 2 5.420608  
## 17 5 1 5.420615  
## 18 5 2 5.422002  
## 12 3 2 5.425295

x = AnonDataTurnstile\_diff  
acf(x)



pacf(x)



Use the estimate of p and q to get estimates of the phis and thetas.

#Use the estimate of p and q to get estimates of the phis and thetas.  
est = est.arma.wge(AnonDataTurnstile\_diff, p = 4, q = 2)

##   
## Coefficients of Original polynomial:   
## 0.6377 0.1247 -0.0120 -0.1937   
##   
## Factor Roots Abs Recip System Freq   
## 1-1.3922B+0.6059B^2 1.1489+-0.5749i 0.7784 0.0738  
## 1+0.7544B+0.3197B^2 -1.1800+-1.3175i 0.5654 0.3662  
##   
##

#Use the estimated model to forecast and so on.   
phi=est$phi  
phi

## [1] 0.63773997 0.12473899 -0.01204978 -0.19367013

theta=est$theta  
theta

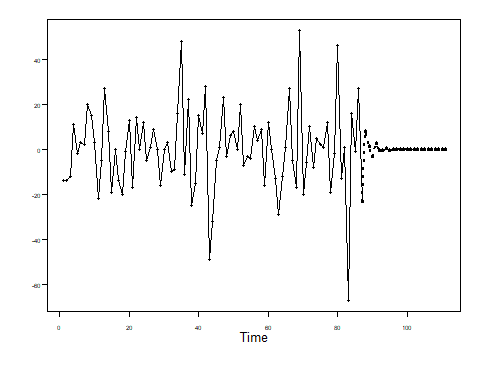
## [1] 1.00540183 -0.03410593

wnv=est$avar  
wnv

## [1] 223.7972

#### Final ARMA Model

#forcast hour  
foreHour\_diff=fore.arma.wge((tail(x,length(x)/72)), phi = est$phi, theta = est$theta, lastn = F, n.ahead = 24, limits=F)



TESTING Look at some of the weather data.

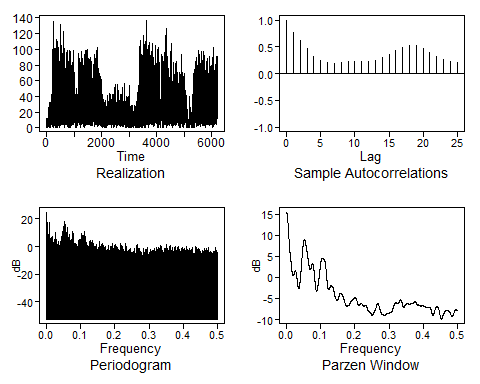
#Conver to a time series.  
#WeatherTS <- ts()  
#plotts.sample.wge(WeatherTS)

#### Seasonal Model

Lets see what a Seasonal Model might look like

Look at the data again.

plotts.sample.wge(AnonDataTurnstile)



There are a few peeks we can look at for a sign of seasonality in the data.  
This data is from a collage so we might expect to find a nine month pattern accounting for the three months of Summer break.  
We found the factors at 36 seemed to match up. This would coinside with a 9 month weekly pattern.

#Factor tables  
factAnonData=est.ar.wge(AnonDataTurnstile,p=36,type='burg')

##   
## Coefficients of Original polynomial:   
## 0.5782 0.1554 0.0111 -0.1663 0.0308 0.0017 -0.0423 0.0479 0.0433 0.0021 -0.0273 0.0104 -0.0011 0.0305 0.0687 0.0377 0.0107 0.0832 0.0633 -0.0215 -0.0123 0.0121 -0.0123 -0.0128 0.0327 0.0172 -0.0076 -0.0325 0.0107 0.0403 -0.0380 -0.0270 -0.0579 0.0366 -0.0153 0.0820   
##   
## Factor Roots Abs Recip System Freq   
## 1-0.9899B 1.0102 0.9899 0.0000  
## 1-1.8495B+0.9574B^2 0.9658+-0.3340i 0.9785 0.0530  
## 1-1.5285B+0.9542B^2 0.8009+-0.6376i 0.9769 0.1070  
## 1-1.6815B+0.8968B^2 0.9375+-0.4859i 0.9470 0.0761  
## 1-0.9835B+0.8950B^2 0.5495+-0.9030i 0.9460 0.1630  
## 1-0.6276B+0.8908B^2 0.3522+-0.9992i 0.9438 0.1961  
## 1+0.9427B -1.0608 0.9427 0.5000  
## 1-1.2557B+0.8859B^2 0.7087+-0.7916i 0.9412 0.1338  
## 1+1.8507B+0.8854B^2 -1.0451+-0.1930i 0.9410 0.4709  
## 1+1.7504B+0.8833B^2 -0.9908+-0.3879i 0.9398 0.4406  
## 1+1.5822B+0.8688B^2 -0.9105+-0.5674i 0.9321 0.4113  
## 1+0.9000B+0.8659B^2 -0.5197+-0.9406i 0.9305 0.3303  
## 1+0.2322B+0.8563B^2 -0.1356+-1.0721i 0.9254 0.2700  
## 1+1.3776B+0.8522B^2 -0.8082+-0.7212i 0.9232 0.3840  
## 1-0.2916B+0.8499B^2 0.1716+-1.0711i 0.9219 0.2247  
## 1+1.1677B+0.8477B^2 -0.6888+-0.8398i 0.9207 0.3593  
## 1+0.6209B+0.8320B^2 -0.3731+-1.0309i 0.9121 0.3053  
## 1-1.7853B+0.8131B^2 1.0979+-0.1568i 0.9017 0.0226  
## 1-0.0094B+0.7257B^2 0.0065+-1.1739i 0.8519 0.2491  
##   
##

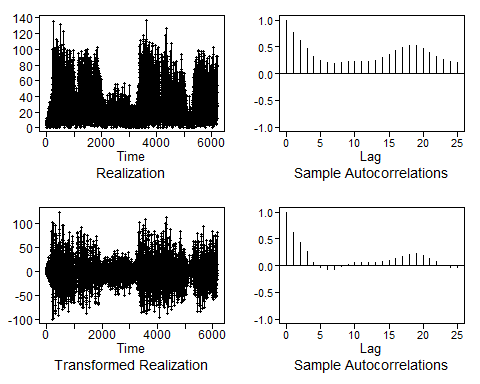
factor.wge(c(rep(0,35),1)) #(1-B^36)

##   
## Coefficients of Original polynomial:   
## 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 1.0000   
##   
## Factor Roots Abs Recip System Freq   
## 1-1.9696B+1.0000B^2 0.9848+-0.1736i 1.0000 0.0278  
## 1-1.0000B+1.0000B^2 0.5000+-0.8660i 1.0000 0.1667  
## 1+1.0000B -1.0000 1.0000 0.5000  
## 1+1.5321B+1.0000B^2 -0.7660+-0.6428i 1.0000 0.3889  
## 1-1.2856B+1.0000B^2 0.6428+-0.7660i 1.0000 0.1389  
## 1-1.7321B+1.0000B^2 0.8660+-0.5000i 1.0000 0.0833  
## 1-1.5321B+1.0000B^2 0.7660+-0.6428i 1.0000 0.1111  
## 1-0.3473B+1.0000B^2 0.1736+-0.9848i 1.0000 0.2222  
## 1+1.2856B+1.0000B^2 -0.6428+-0.7660i 1.0000 0.3611  
## 1+0.3473B+1.0000B^2 -0.1736+-0.9848i 1.0000 0.2778  
## 1-1.0000B 1.0000 1.0000 0.0000  
## 1+1.0000B+1.0000B^2 -0.5000+-0.8660i 1.0000 0.3333  
## 1-1.8794B+1.0000B^2 0.9397+-0.3420i 1.0000 0.0556  
## 1-0.6840B+1.0000B^2 0.3420+-0.9397i 1.0000 0.1944  
## 1+0.0000B+1.0000B^2 0.0000+-1.0000i 1.0000 0.2500  
## 1+0.6840B+1.0000B^2 -0.3420+-0.9397i 1.0000 0.3056  
## 1+1.9696B+1.0000B^2 -0.9848+-0.1736i 1.0000 0.4722  
## 1+1.7321B+1.0000B^2 -0.8660+-0.5000i 1.0000 0.4167  
## 1+1.8794B+1.0000B^2 -0.9397+-0.3420i 1.0000 0.4444  
##   
##

#factor.wge(c(0,0,0,0,0,0,0,0,1)) #(1-B^9)

Use artrans.wge() to get y. This is to remove the seasonality in the data.

#Use artrans.wge() to get y. This is to remove the seasonality in the data.  
#In this data we are checking for nine months weekley number (36).  
y = artrans.wge(AnonDataTurnstile,phi.tr=c(c(rep(0,35),1)))



# y is the transformed data  
#aic5.wge(y,p=0:15,q=0:6,type='bic') #picked a ARMA(9,6)

Based on the decision to fit an ARMA(9,6) model, we use the est.ar.wge command to obtain ML estimates.

AnonDataTurnstile.est155=est.arma.wge(y,p=9, q=6)

##   
## Coefficients of Original polynomial:   
## 0.6868 -0.1084 0.2612 -0.4556 -0.5855 0.4482 0.0591 0.0516 -0.1038   
##   
## Factor Roots Abs Recip System Freq   
## 1-1.5829B+0.9874B^2 0.8015+-0.6085i 0.9937 0.1033  
## 1+0.6782B+0.9241B^2 -0.3670+-0.9734i 0.9613 0.3074  
## 1+0.8313B -1.2030 0.8313 0.5000  
## 1-1.2378B+0.4377B^2 1.4140+-0.5342i 0.6616 0.0575  
## 1+0.6244B+0.3126B^2 -0.9987+-1.4838i 0.5591 0.3443  
##   
##

AnonDataTurnstile.est155$phi

## [1] 0.68679676 -0.10839164 0.26121944 -0.45561104 -0.58554610 0.44821701  
## [7] 0.05911743 0.05155061 -0.10377254

AnonDataTurnstile.est155$theta

## [1] 0.11103637 -0.19253158 0.16064671 -0.24447955 -0.78060402 0.04999182

AnonDataTurnstile.est155$avar

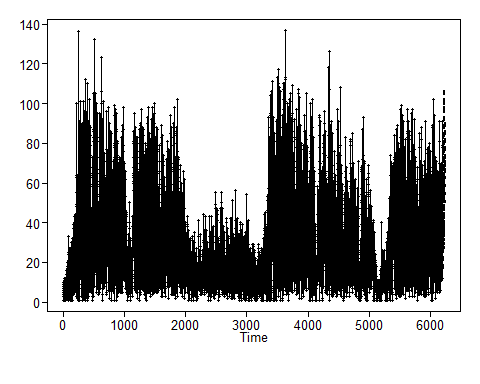
## [1] 411.9237

mean(AnonDataTurnstile)

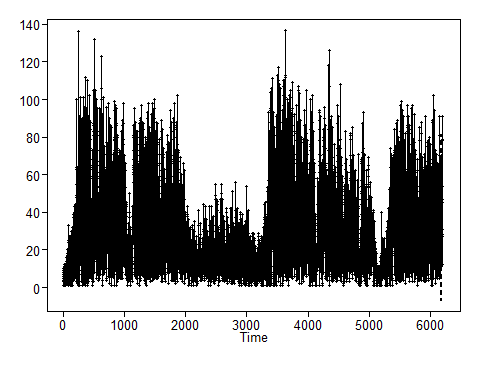
## [1] 33.37821

#### Final Seasonal Model

seasonFore=fore.aruma.wge(AnonDataTurnstile,phi=c(factAnonData$phi),s=36,n.ahead=48,limits=F)



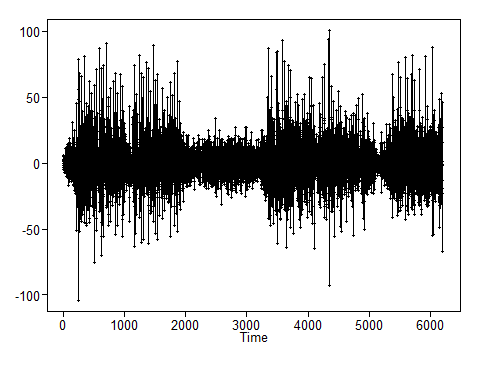
seasonFore$f  
  
fore.aruma.wge(AnonDataTurnstile,phi=c(factAnonData$phi),s=36,n.ahead=48,plot=T,lastn=T,limits=F)



p=15;q=9;s=3  
es=tswge::est.arma.wge(x,p=p,q=q)

##   
## Coefficients of Original polynomial:   
## -0.6601 -0.2037 0.0504 0.2138 -0.5360 -0.8913 -0.3991 -0.1656 0.1943 -0.1544 -0.2146 -0.2537 -0.2508 -0.2023 -0.0809   
##   
## Factor Roots Abs Recip System Freq   
## 1+1.8705B+0.9577B^2 -0.9765+-0.3009i 0.9786 0.4524  
## 1+0.1990B+0.9551B^2 -0.1042+-1.0179i 0.9773 0.2662  
## 1-1.5381B+0.9421B^2 0.8163+-0.6286i 0.9706 0.1044  
## 1-1.7387B+0.8680B^2 1.0016+-0.3859i 0.9317 0.0585  
## 1+0.7397B+0.7439B^2 -0.4971+-1.0474i 0.8625 0.3205  
## 1-0.4321B+0.5335B^2 0.4050+-1.3078i 0.7304 0.2022  
## 1+0.9666B+0.4592B^2 -1.0525+-1.0344i 0.6777 0.3764  
## 1+0.5933B -1.6856 0.5933 0.5000  
##   
##

ase = function(f,x){mean((f - tail(x,length(f)))^2)}  
m = tswge::fore.aruma.wge(x,phi = es$phi,theta = es$theta,s=s,n.ahead = 48,lastn = T, limits = F)



ase = ase(m$f,x)  
message("ASE is: ",ase)

## ASE is: 1063.06802306848

ase2=ase

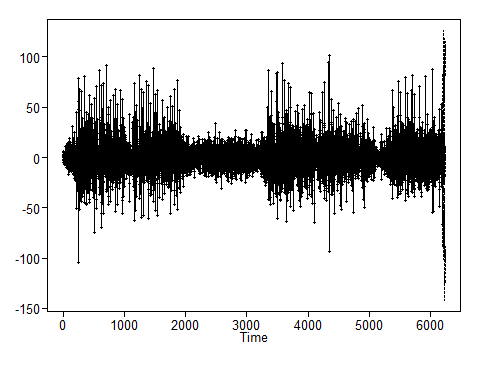
Trying the forecasting package on the hourly data

#Trying the forcasting package  
  
IDSwipes <- read\_csv("DedmanDailySwipe.csv")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## Date = col\_date(format = ""),  
## LDT = col\_datetime(format = ""),  
## Turnstile = col\_character(),  
## Day = col\_character(),  
## Hours = col\_datetime(format = ""),  
## Month = col\_character(),  
## HourlyPrecipitation = col\_logical(),  
## HourlyPresentWeatherType = col\_character(),  
## HourlySkyConditions = col\_character(),  
## HourlyWindDirection = col\_character(),  
## HourlyWindGustSpeed = col\_character(),  
## WeekdayCount = col\_character(),  
## Week = col\_character(),  
## End\_of\_Week = col\_date(format = ""),  
## DayTemperature.Date = col\_date(format = "")  
## )

## See spec(...) for full column specifications.

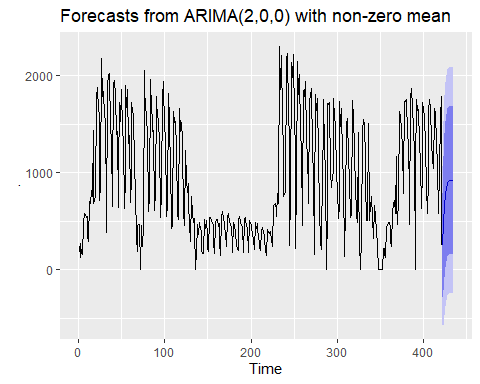
#Convert to a time series at 365 days  
#IDSwipes = ts(IDSwipes$IDSwipes, start = c(2019,2), frequency = 365)  
  
IDSwipes = ts(IDSwipes$IDSwipes)  
  
#Divide dataset into training and test set. The last 36 months in the test set.  
IDSwipes1st = (IDSwipes[1:100])  
IDSwipes2nd = (IDSwipes[101:421])  
  
AnonDataTurnstile.f = auto.arima(IDSwipes, stepwise=FALSE, approximation=FALSE, stationary = FALSE, seasonal = TRUE, trace = FALSE,max.p = 13,max.q = 5,max.P = 13,max.Q = 5)  
  
f=fore.aruma.wge(x, phi = c(1.195, 0.0297, 0.0842, 0.187, -1.169, .524, .076, .217, -.197, -.011,.023,-.033,.078,-.075,.0318), theta = c(.614, .302, .305, .624, -.993), s = 36, n.ahead = 48, plot = TRUE)



DayForcast=forecast::forecast(AnonDataTurnstile.f, h=7, level = 95)  
DayForcast

## Point Forecast Lo 95 Hi 95  
## 422 323.9133 -419.59031 1067.417  
## 423 558.9822 -340.25921 1458.224  
## 424 1131.4703 193.76856 2069.172  
## 425 1302.1159 363.80427 2240.427  
## 426 1192.2846 252.90445 2131.665  
## 427 930.7972 -11.69285 1873.287  
## 428 785.1916 -182.57260 1752.956

# Automatic ARIMA forecasts  
IDSwipes %>%  
 auto.arima() %>%  
 forecast(h=14) %>%  
 autoplot()



#### NN

Fit the model with the default settings forcasting 14 days out.

df <- read\_csv("DedmanDailySwipe.csv")

## Parsed with column specification:  
## cols(  
## .default = col\_double(),  
## Date = col\_date(format = ""),  
## LDT = col\_datetime(format = ""),  
## Turnstile = col\_character(),  
## Day = col\_character(),  
## Hours = col\_datetime(format = ""),  
## Month = col\_character(),  
## HourlyPrecipitation = col\_logical(),  
## HourlyPresentWeatherType = col\_character(),  
## HourlySkyConditions = col\_character(),  
## HourlyWindDirection = col\_character(),  
## HourlyWindGustSpeed = col\_character(),  
## WeekdayCount = col\_character(),  
## Week = col\_character(),  
## End\_of\_Week = col\_date(format = ""),  
## DayTemperature.Date = col\_date(format = "")  
## )

## See spec(...) for full column specifications.

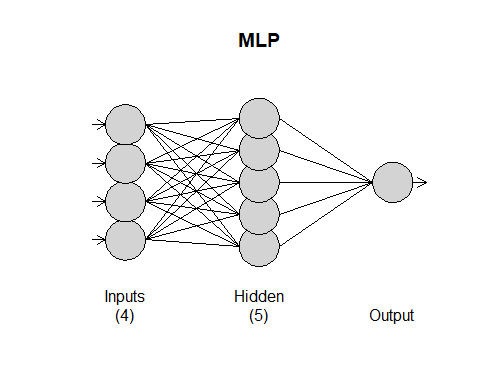
#Convert to a time series at 365 days  
df = ts(df$IDSwipes, start = c(2019,2), frequency = 365)  
  
df

## Time Series:  
## Start = c(2019, 2)   
## End = c(2020, 57)   
## Frequency = 365   
## [1] 235 123 267 172 156 480 584 570 534 552 342 290 705 679 816  
## [16] 752 1439 684 714 768 1783 1882 1642 1619 709 868 2180 1783 1838 1588  
## [31] 1548 642 385 1904 1997 2018 1469 1328 652 953 1924 1947 1809 1396 1422  
## [46] 638 1005 1722 1525 1861 1434 1332 635 952 1895 1677 1853 1300 1382 695  
## [61] 942 1727 1626 1597 1072 804 328 184 452 471 466 1 460 236 456  
## [76] 2053 1823 1723 1557 1416 601 756 1956 1709 1798 1424 1436 607 807 1790  
## [91] 1509 1650 1488 1351 541 795 1857 1780 1939 1445 1407 551 653 1821 1552  
## [106] 1395 1148 427 458 1626 1516 1525 1526 1264 575 522 1659 1422 1514 1341  
## [121] 1283 590 455 1233 928 794 885 683 465 296 759 595 486 533 439  
## [136] 3 186 463 360 492 422 391 213 166 169 517 424 510 368 221  
## [151] 186 543 536 506 483 454 284 166 492 523 520 446 480 284 151  
## [166] 538 604 538 476 345 286 239 542 582 497 481 429 266 183 490  
## [181] 457 382 247 217 196 517 549 501 458 450 178 518 537 496 465  
## [196] 424 181 511 478 463 437 371 300 206 486 428 476 393 376 240  
## [211] 196 406 438 385 360 313 205 152 419 387 375 396 387 327 240  
## [226] 667 671 678 616 554 787 688 2300 1855 2204 1548 1228 756 774 809  
## [241] 2177 2225 1629 1336 249 768 2134 1977 2221 1532 1261 220 907 2144 1811  
## [256] 2014 1416 1243 459 764 1852 1787 1942 1363 1141 842 685 1741 1797 1866  
## [271] 1248 955 154 609 1803 1641 1763 1158 632 284 209 585 715 1756 1259  
## [286] 1004 1 631 1710 1702 1723 1005 846 986 988 1760 1569 1499 979 833  
## [301] 527 650 1739 1480 1658 1136 945 141 666 1495 1505 1555 1165 920 537  
## [316] 566 1747 1587 1656 1166 876 561 484 1417 925 286 1 337 1440 1550  
## [331] 1511 1131 891 508 508 1506 505 675 724 756 534 324 667 555 390  
## [346] 261 245 3 2 3 2 2 223 217 161 131 446 461 485 489  
## [361] 442 290 244 590 675 709 575 1164 696 466 743 1630 1589 1432 1258  
## [376] 782 913 1734 1732 1751 1403 1301 784 471 1760 1871 1686 1447 1208 2  
## [391] 1051 1756 1637 1656 1307 1103 633 1057 1719 1660 1569 1289 834 581 944  
## [406] 1631 1752 1667 1316 1150 1048 1616 1663 1620 1308 1078 584 1036 1558 1786  
## [421] 250

fit.mlp.Dedman = mlp(df)  
  
fit.mlp.Dedman

## MLP fit with 5 hidden nodes and 20 repetitions.  
## Series modelled in differences: D1.  
## Univariate lags: (188,226,295,351)  
## Forecast combined using the median operator.  
## MSE: 22194.0002.

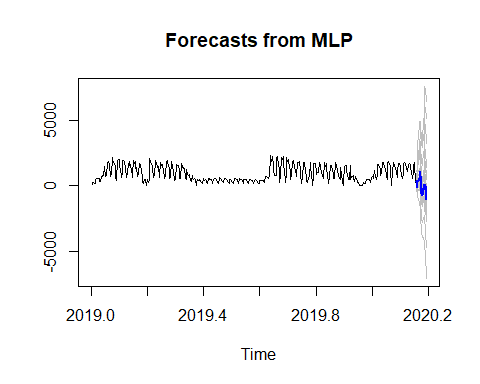
plot(fit.mlp.Dedman)



fore.mlp.Dedman = forecast(fit.mlp.Dedman, h = 14)  
fore.mlp.Dedman

## Point Forecast  
## 2020.1562 306.86795  
## 2020.1589 -117.41611  
## 2020.1616 394.33445  
## 2020.1644 342.60104  
## 2020.1671 557.88732  
## 2020.1699 1092.97407  
## 2020.1726 802.53221  
## 2020.1753 -564.46478  
## 2020.1781 -733.72729  
## 2020.1808 -242.62416  
## 2020.1836 -262.88367  
## 2020.1863 60.22653  
## 2020.1890 10.02422  
## 2020.1918 -1044.93161

plot(fore.mlp.Dedman)



ASE\_Dedman\_NNid = mean((df[((421-14)+1):421] - fore.mlp.Dedman$mean)^2)  
ASE\_Dedman\_NNid

## [1] 1665657