Seoul Bike Data

William K Davis III Pei-Yin Yang Max Kutschinski

2022-06-20

http://archive.ics.uci.edu/ml/bikesets/Seoul+Bike+Sharing+Demand

Read Data

```
bike <- readr::read_csv("SeoulBikeData.csv",
                        col_names = c("Date",
                                      "BikeCount",
                                      "Hour",
                                      "Temperature",
                                      "Humidity",
                                      "WindSpeed",
                                      "Visibility",
                                      "Dewpoint",
                                      "SolarRadiation",
                                      "Rainfall",
                                      "Snowfall",
                                      "Seasons",
                                      "Holiday",
                                      "FunctionalDay"),
                        skip = 1,
                        col_types = cols("Hour" = col_time(format = "%H"),
                                         Seasons = "f",
                                         Holiday = "f",
                                         FunctionalDay = "f"))
```

Data Cleaning

```
bikets <- bike %>%
  mutate(
    Hour = parse_date_time(
      paste(Date, Hour),
      orders = c("dmy HMS", "dmY HMS"),
      tz = "Asia/Seoul"
    ),
    .before = everything(),
    Date = NULL
  ) %>%
```

```
  as_tsibble(index = Hour)

bikets_tall <- bikets %>%
  select(Hour, where(is.numeric)) %>%
  pivot_longer(cols = -Hour,
               names_to = "Measure",
               values_to = "Value")

bikets %>% count_gaps()
```

```
## # A tibble: 0 x 3
## # ... with 3 variables: .from <dttm>, .to <dttm>, .n <int>
```

EDA

Data Summary

```
colnames(bikets)
```

```
##  [1] "BikeCount"      "Hour"           "Temperature"    "Humidity"
##  [5] "WindSpeed"      "Visibility"     "Dewpoint"       "SolarRadiation"
##  [9] "Rainfall"       "Snowfall"       "Seasons"        "Holiday"
## [13] "FunctionalDay"
```

```
anyNA(bikets)
```

```
## [1] FALSE
```

```
dim(bikets)
```

```
## [1] 8760   13
```

```
str(bikets)
```

```
## tbl_ts [8,760 x 13] (S3: tbl_ts/tbl_df/tbl/data.frame)
##  $ BikeCount     : num [1:8760] 254 204 173 107 78 100 181 460 930 490 ...
##  $ Hour          : POSIXct[1:8760], format: "2017-12-01 00:00:00" "2017-12-01 01:00:00" ...
##  $ Temperature   : num [1:8760] -5.2 -5.5 -6 -6.2 -6 -6.4 -6.6 -7.4 -7.6 -6.5 ...
##  $ Humidity      : num [1:8760] 37 38 39 40 36 37 35 38 37 27 ...
##  $ WindSpeed     : num [1:8760] 2.2 0.8 1 0.9 2.3 1.5 1.3 0.9 1.1 0.5 ...
##  $ Visibility    : num [1:8760] 2000 2000 2000 2000 2000 ...
##  $ Dewpoint      : num [1:8760] -17.6 -17.6 -17.7 -17.6 -18.6 -18.7 -19.5 -19.3 -19.8 -22.4 ...
##  $ SolarRadiation: num [1:8760] 0 0 0 0 0 0 0 0 0.01 0.23 ...
##  $ Rainfall      : num [1:8760] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Snowfall      : num [1:8760] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Seasons       : Factor w/ 4 levels "Winter","Spring",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Holiday       : Factor w/ 2 levels "No Holiday","Holiday": 1 1 1 1 1 1 1 1 1 1 ...
##  $ FunctionalDay : Factor w/ 2 levels "Yes","No": 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  - attr(*, "key")= tibble [1 x 1] (S3: tbl_df/tbl/data.frame)
##   ..$ .rows: list<int> [1:1]
##   .. ..$ : int [1:8760] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..@ ptype: int(0)
##  - attr(*, "index")= chr "Hour"
##   ..- attr(*, "ordered")= logi TRUE
##  - attr(*, "index2")= chr "Hour"
##  - attr(*, "interval")= interval [1:1] 1h
##   ..@ .regular: logi TRUE
```

```
summary(bikets) # Investigate BikeCount = 0 and Humidity = 0
```

```
##    BikeCount           Hour                       Temperature
##  Min.   :   0.0   Min.   :2017-12-01 00:00:00   Min.   :-17.80
##  1st Qu.: 191.0   1st Qu.:2018-03-02 05:45:00   1st Qu.:  3.50
##  Median : 504.5   Median :2018-06-01 11:30:00   Median : 13.70
##  Mean   : 704.6   Mean   :2018-06-01 11:30:00   Mean   : 12.88
##  3rd Qu.:1065.2   3rd Qu.:2018-08-31 17:15:00   3rd Qu.: 22.50
##  Max.   :3556.0   Max.   :2018-11-30 23:00:00   Max.   : 39.40
##     Humidity       WindSpeed       Visibility      Dewpoint
##  Min.   : 0.00   Min.   :0.000   Min.   :  27   Min.   :-30.600
##  1st Qu.:42.00   1st Qu.:0.900   1st Qu.: 940   1st Qu.: -4.700
##  Median :57.00   Median :1.500   Median :1698   Median :  5.100
##  Mean   :58.23   Mean   :1.725   Mean   :1437   Mean   :  4.074
##  3rd Qu.:74.00   3rd Qu.:2.300   3rd Qu.:2000   3rd Qu.: 14.800
##  Max.   :98.00   Max.   :7.400   Max.   :2000   Max.   : 27.200
##  SolarRadiation      Rainfall          Snowfall         Seasons
##  Min.   :0.0000   Min.   : 0.0000   Min.   :0.00000   Winter:2160
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.00000   Spring:2208
##  Median :0.0100   Median : 0.0000   Median :0.00000   Summer:2208
##  Mean   :0.5691   Mean   : 0.1487   Mean   :0.07507   Autumn:2184
##  3rd Qu.:0.9300   3rd Qu.: 0.0000   3rd Qu.:0.00000
##  Max.   :3.5200   Max.   :35.0000   Max.   :8.80000
##        Holiday     FunctionalDay
##  No Holiday:8328   Yes:8465
##  Holiday   : 432   No : 295
##
##
##
##
```

```
sum(bike$BikeCount==0) # 295 NAs
```

```
## [1] 295
```

Notice there are no missing values. However there are some observations with BikeCount = 0, which correspond to non-functional days. Can these be treated as missing values and be imputed? - Humidity has a minimum value of 0. Is this realistic? (see feature plot for further investigation)

```
bike %>% group_by(FunctionalDay) %>% summarise(bc = sum(BikeCount))
```

```
## # A tibble: 2 x 2
```

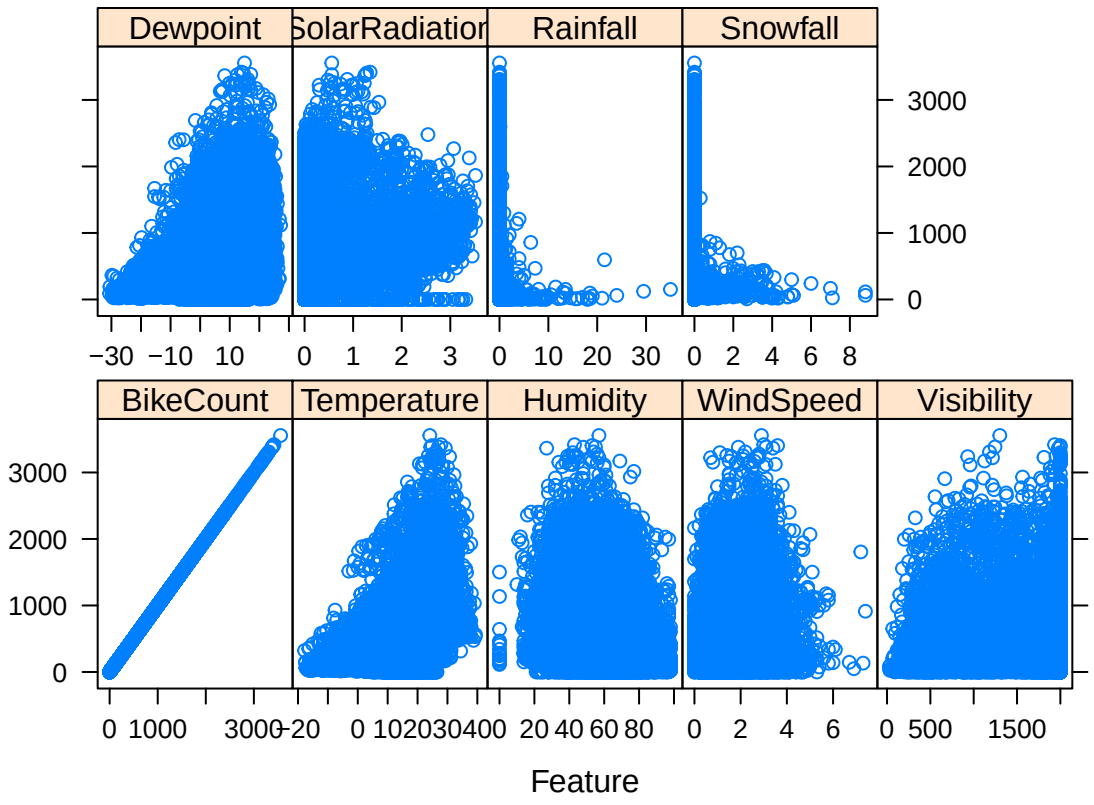
```
##   FunctionalDay      bc
##   <fct>           <dbl>
## 1 Yes           6172314
## 2 No                  0
```

```
bike %>% filter(FunctionalDay == "Yes", BikeCount == 0)
```

```
## # A tibble: 0 x 14
## # ... with 14 variables: Date <chr>, BikeCount <dbl>, Hour <time>,
## #   Temperature <dbl>, Humidity <dbl>, WindSpeed <dbl>, Visibility <dbl>,
## #   Dewpoint <dbl>, SolarRadiation <dbl>, Rainfall <dbl>, Snowfall <dbl>,
## #   Seasons <fct>, Holiday <fct>, FunctionalDay <fct>
```

It looks like FunctionalDay=No only has BikeCount = 0, and there are no FunctionalDay=Yes with BikeCount > 0. Based on this I think it makes sense to treat those days as missing data (i.e. set `BikeCount=NA`). The imputation should probably happen during the modeling phase, as different modeling methods might handle missing data differently.

Feature Plots



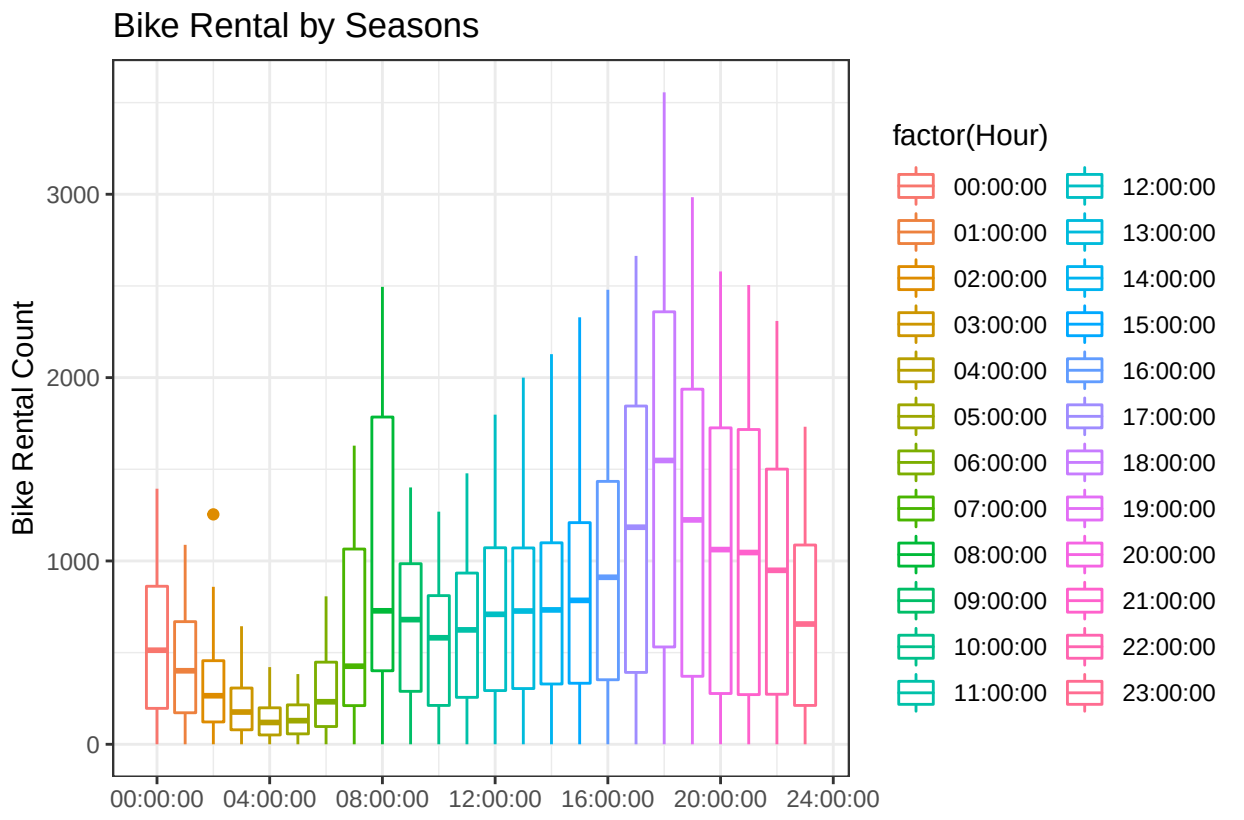
Something seems to be going on with humidity = 0 values.

Scatter plots for numerical variables and box plot for categorical variables

```
library(gridExtra)

level_order <- c('Spring', 'Summer', 'Autumn', 'Winter')

season <- bike %>%
  ggplot(aes(factor(Seasons, levels = level_order), BikeCount, fill = Holiday)) +
  geom_boxplot() +
  labs(x = "",y = 'Bike Rental Count',
       title = 'Bike Rental by Seasons') +
  theme_bw()

# rentals by hour, split by holiday (or not)
hour <- bike %>% group_by(Hour, Holiday) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(x = Hour, y = AvgBikeCount, col = Holiday)) +
  labs(x = "",y = 'Average Bike Rental',
       title = 'Average Bike Rental by Hours') +
  geom_line() + theme_bw()
```

```
## `summarise()` has grouped output by 'Hour'. You can override using the
## `.groups` argument.
```

```
bike %>%
  ggplot(aes(Hour, BikeCount, col = factor(Hour))) +
  geom_boxplot() +
  labs(x = "",y = 'Bike Rental Count',
       title = 'Bike Rental by Seasons')
```




```
bike %>% group_by(FunctionalDay) %>%
  summarise(SumBikeCount = sum(BikeCount))
```

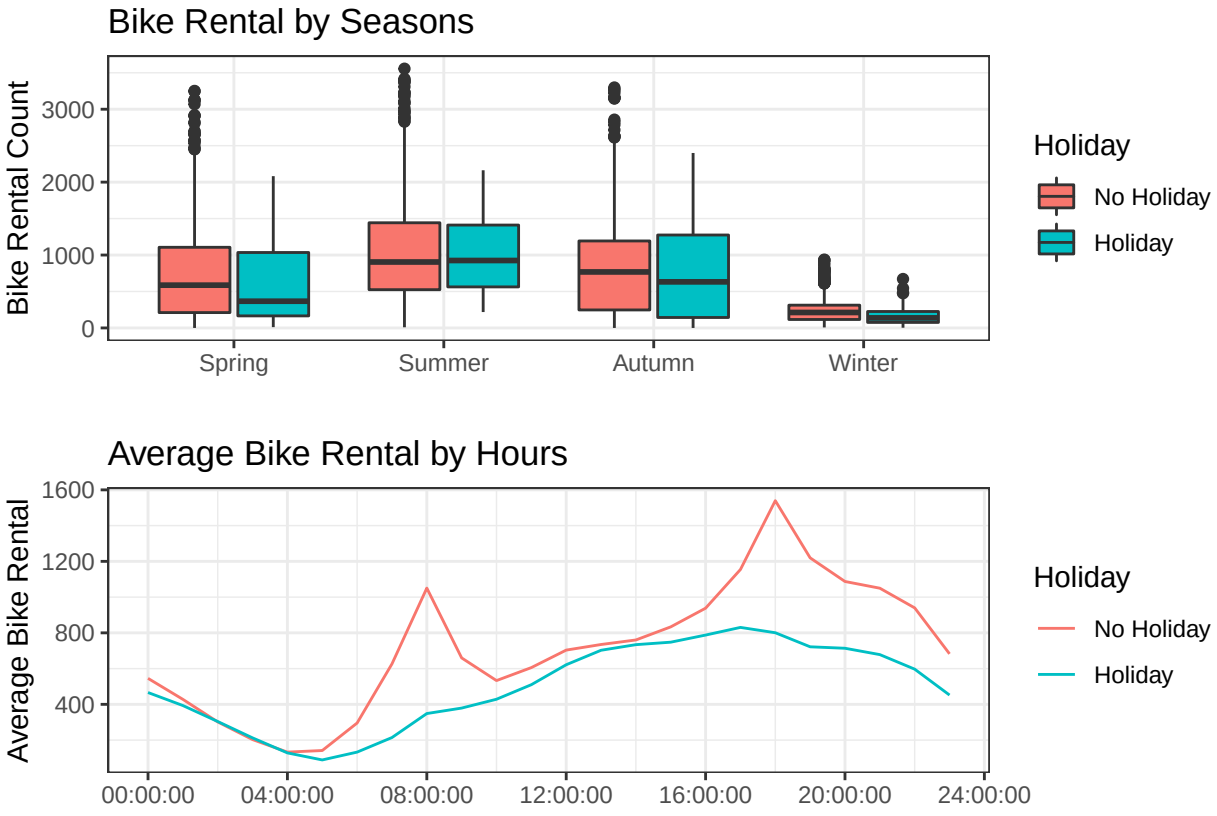
```
## # A tibble: 2 x 2
##   FunctionalDay SumBikeCount
##   <fct>                <dbl>
## 1 Yes                6172314
## 2 No                       0
```

```
unique(bike$FunctionalDay)
```

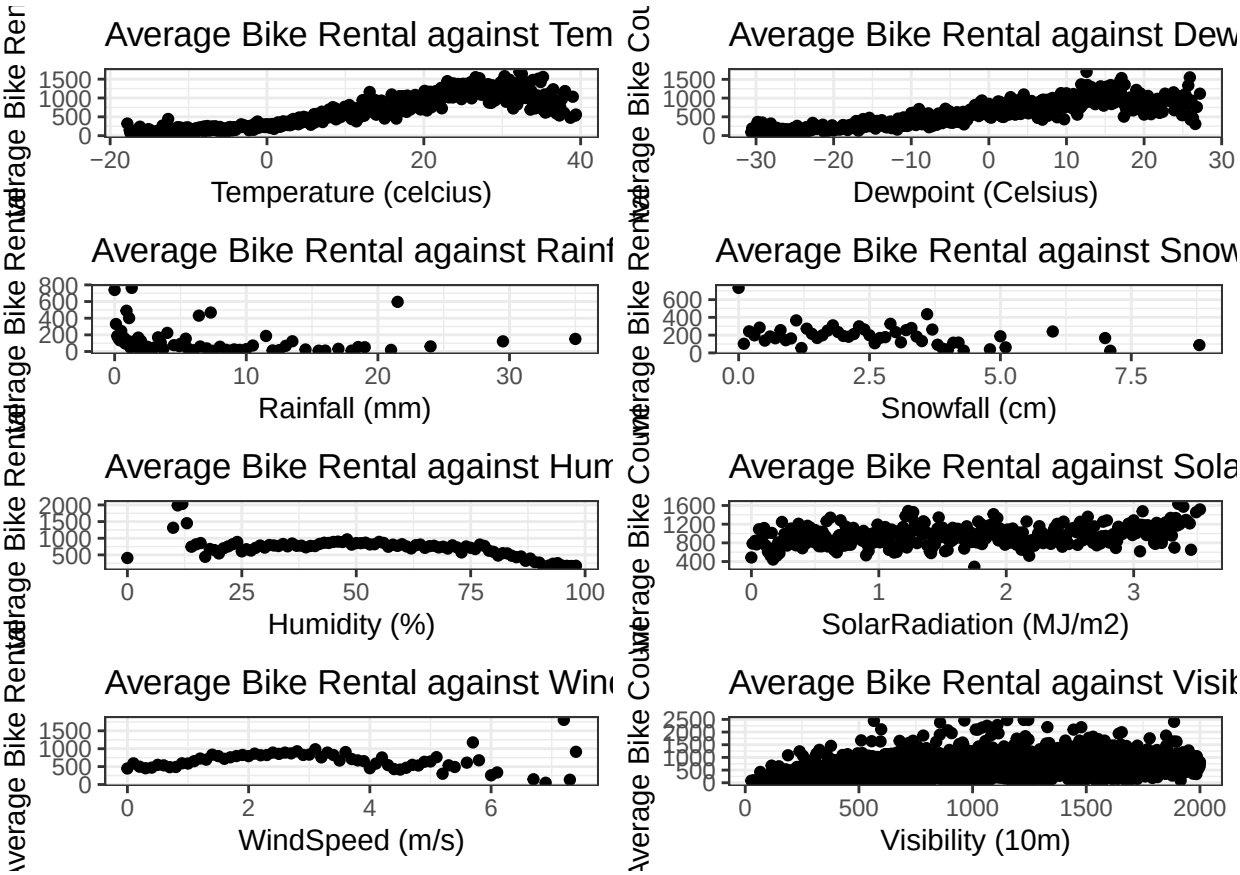
```
## [1] Yes No
## Levels: Yes No
```

```
#bike count of non-Functional day = 0. Agreed with treating them as NA.

rainfall <- bike %>% group_by(Rainfall) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(Rainfall, AvgBikeCount)) +
  geom_point() +
  labs(x = "Rainfall (mm)", y = 'Average Bike Rental',
       title = 'Average Bike Rental against Rainfall') +
  theme_bw()
```

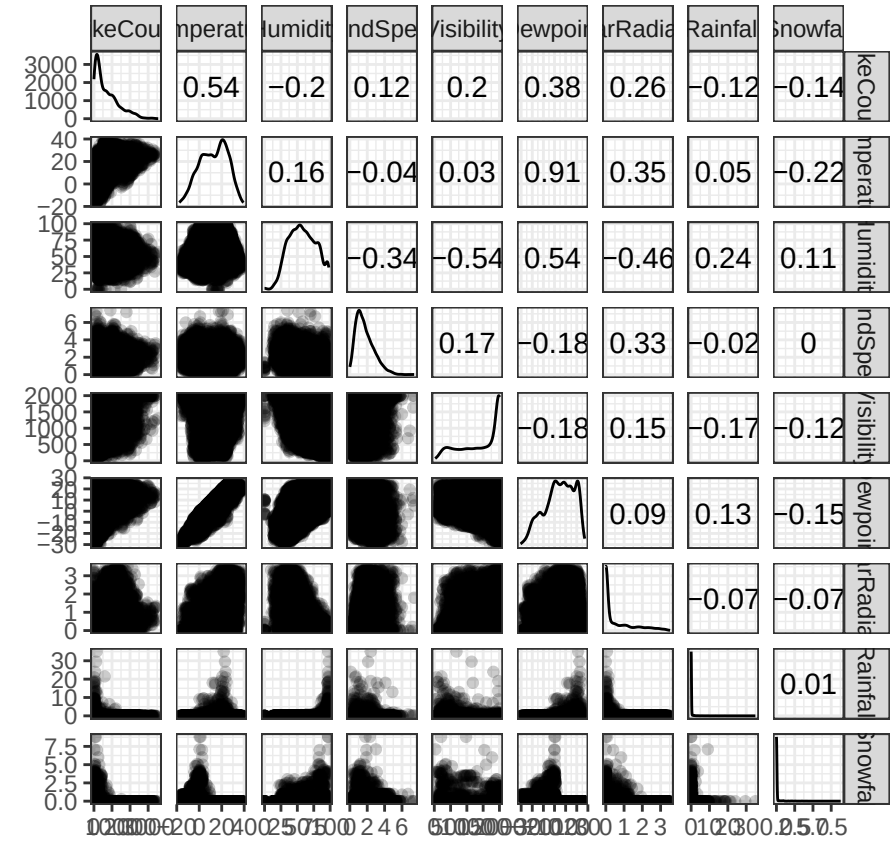
```r
snowfall <- bike %>% group_by(Snowfall) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(Snowfall, AvgBikeCount)) +
  geom_point() +
  labs(x = "Snowfall (cm)", y = 'Average Bike Rental',
       title = 'Average Bike Rental against Snowfall') + theme_bw()

temp <- bike %>% group_by(Temperature) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(Temperature, AvgBikeCount)) +
  geom_point() +
  labs(x = "Temperature (celcius)", y = 'Average Bike Rental',
       title = 'Average Bike Rental against Temperature') + theme_bw()

humidity <- bike %>% group_by(Humidity) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(Humidity, AvgBikeCount)) +
  geom_point() +
  labs(x = "Humidity (%)", y = 'Average Bike Rental',
       title = 'Average Bike Rental against Humidity') + theme_bw()

wp <- bike %>% group_by(WindSpeed) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(WindSpeed, AvgBikeCount)) +
  geom_point() +
  labs(x = "WindSpeed (m/s)", y = 'Average Bike Rental',
       title = 'Average Bike Rental against Wind Speed') + theme_bw()

vis <- bike %>% group_by(Visibility) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(Visibility, AvgBikeCount)) +
  geom_point() +
  labs(x = "Visibility (10m)", y = 'Average Bike Count',
       title = 'Average Bike Rental against Visibility') + theme_bw()

dew <- bike %>% group_by(Dewpoint) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(Dewpoint, AvgBikeCount)) +
  geom_point() +
  labs(x = "Dewpoint (Celsius)", y = 'Average Bike Count',
       title = 'Average Bike Rental against Dewpoint') + theme_bw()

solar <- bike %>% group_by(SolarRadiation) %>%
  summarise(AvgBikeCount = mean(BikeCount)) %>%
  ggplot(aes(SolarRadiation, AvgBikeCount)) +
  geom_point(stat='identity') +
  labs(x = "SolarRadiation (MJ/m2)", y = 'Average Bike Count',
       title = 'Average Bike Rental against Solar Radiation') + theme_bw()

grid.arrange(season, hour, nrow=2)
```

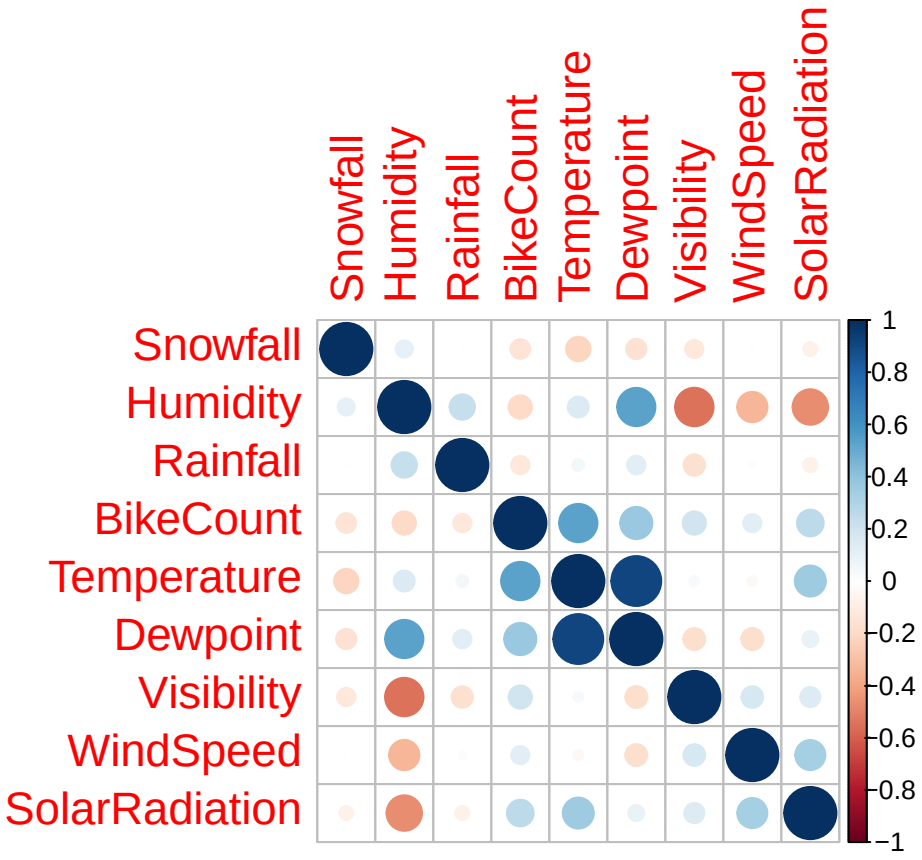



```
grid.arrange(temp, dew, rainfall, snowfall, humidity, solar, wp, vis, nrow=4, ncol = 2)
```


Average Bike Rental against Tem
Temperature (celcius)
Average Bike Rental against Dew
Dewpoint (Celsius)
Average Bike Rental against Rainf
Rainfall (mm)
Average Bike Rental against Snow
Snowfall (cm)
Average Bike Rental against Hum
Humidity (%)
Average Bike Rental against Sola
SolarRadiation (MJ/m2)
Average Bike Rental against Wind
WindSpeed (m/s)
Average Bike Rental against Visib
Visibility (10m)

Correlation Plot



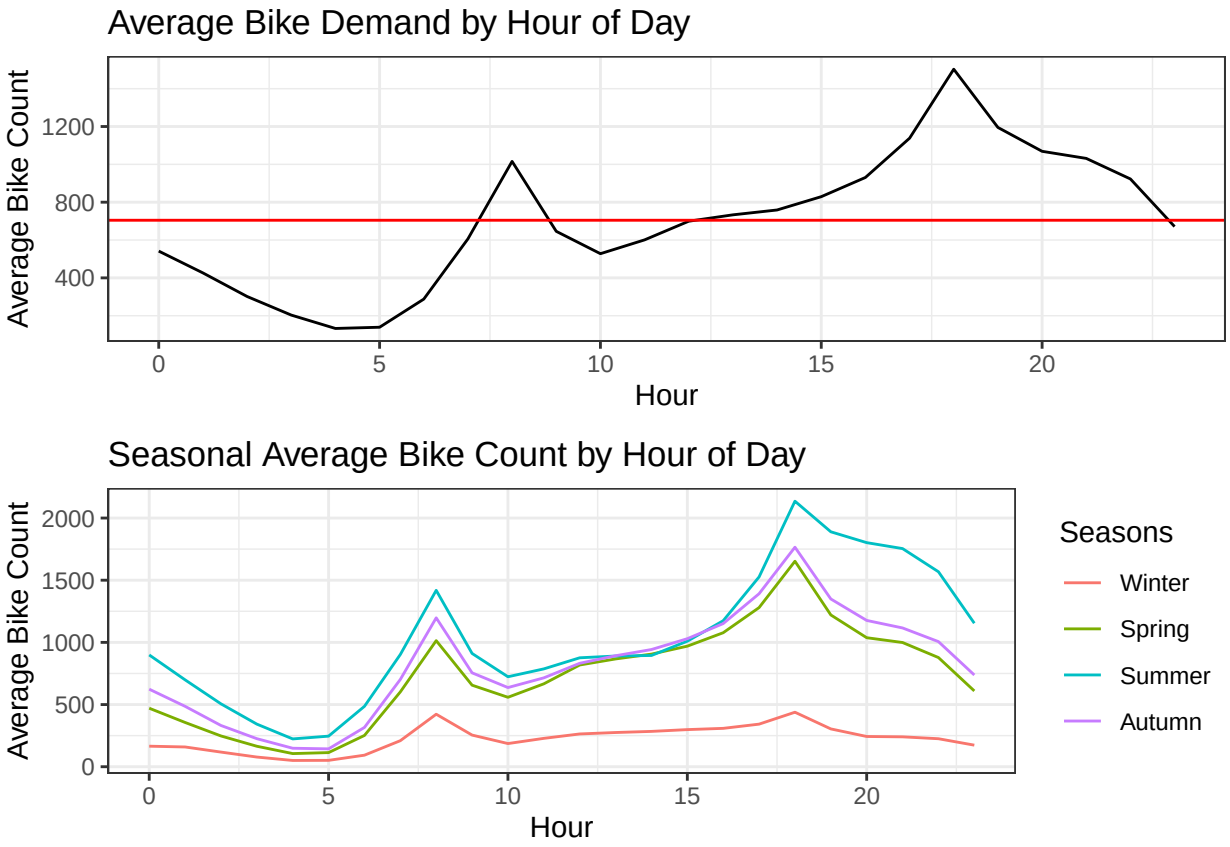


```
## Compare row 6  and column  2 with corr  0.913
##   Means:  0.319 vs 0.208 so flagging column 6
## All correlations <= 0.85
```

```
## [1] "Dewpoint"
```

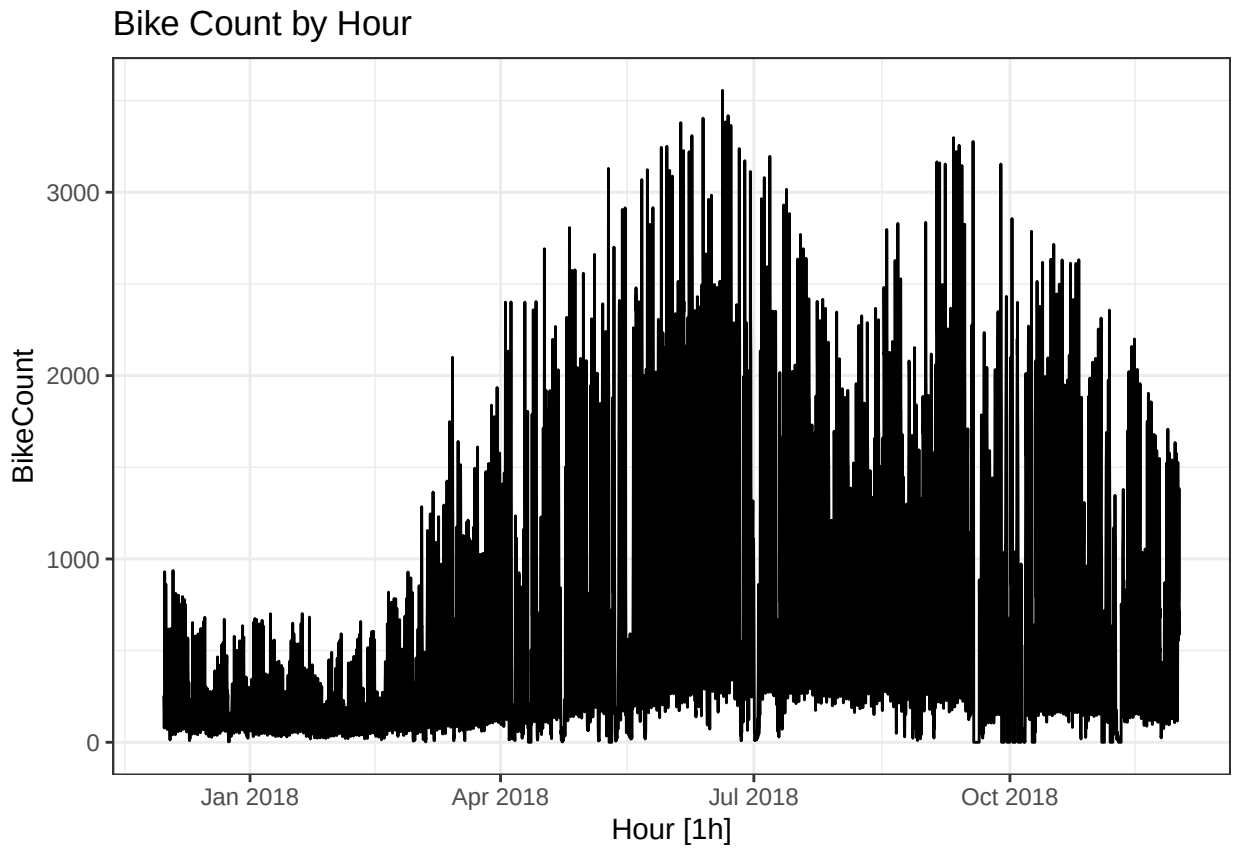
Dewpoint and Temperature potentially problematic

Bike Demand by Hour of Day

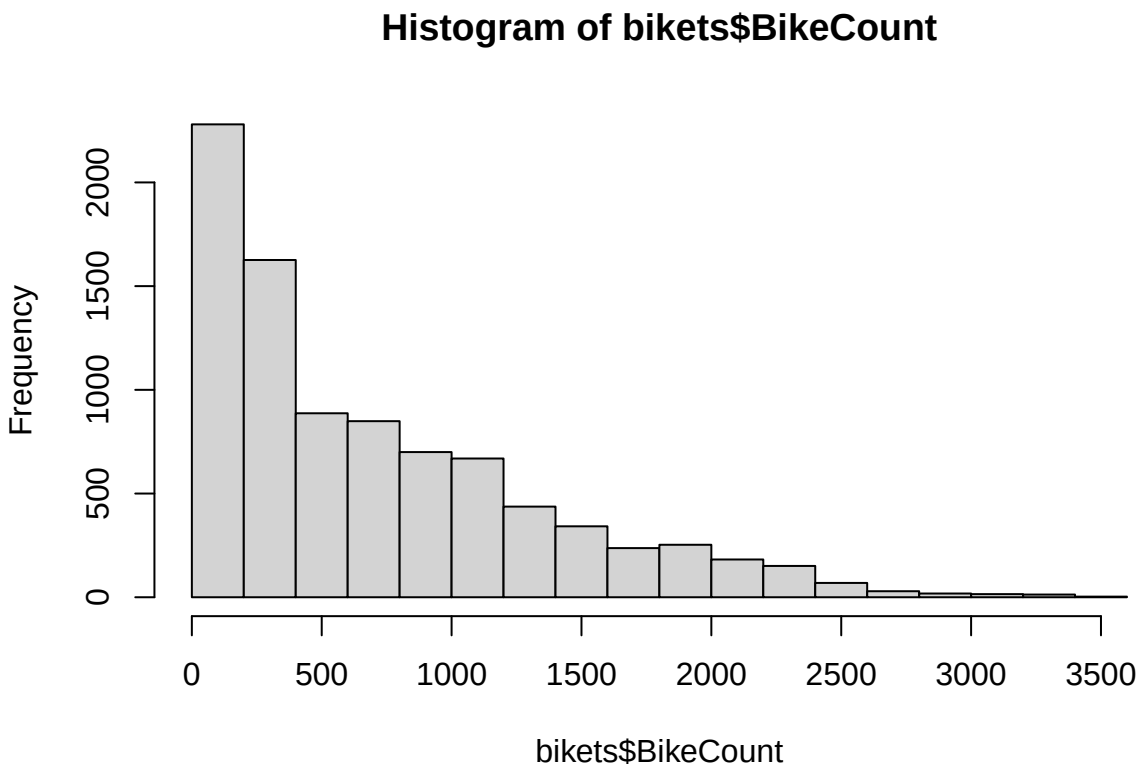


Time Series Plots

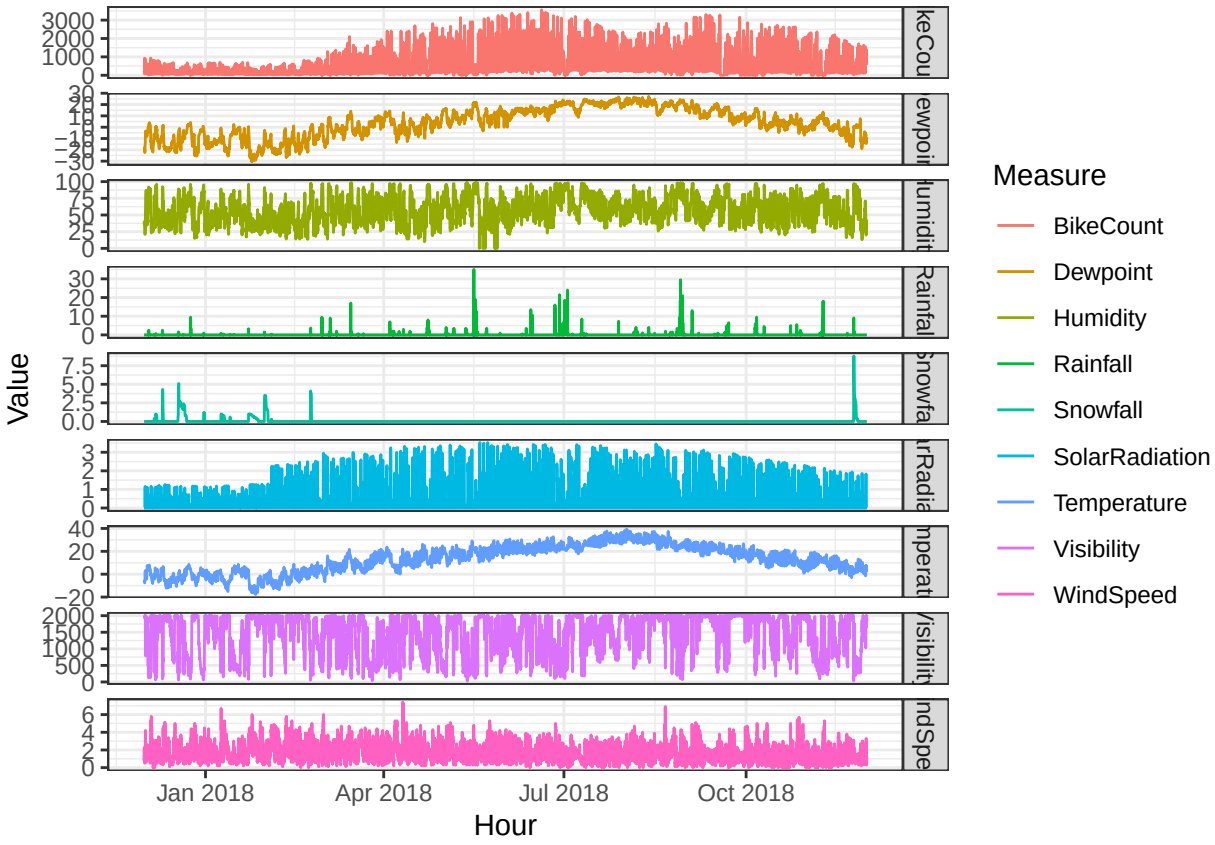
```
autoplot(bikets, .vars = BikeCount) + labs(title = "Bike Count by Hour")
```



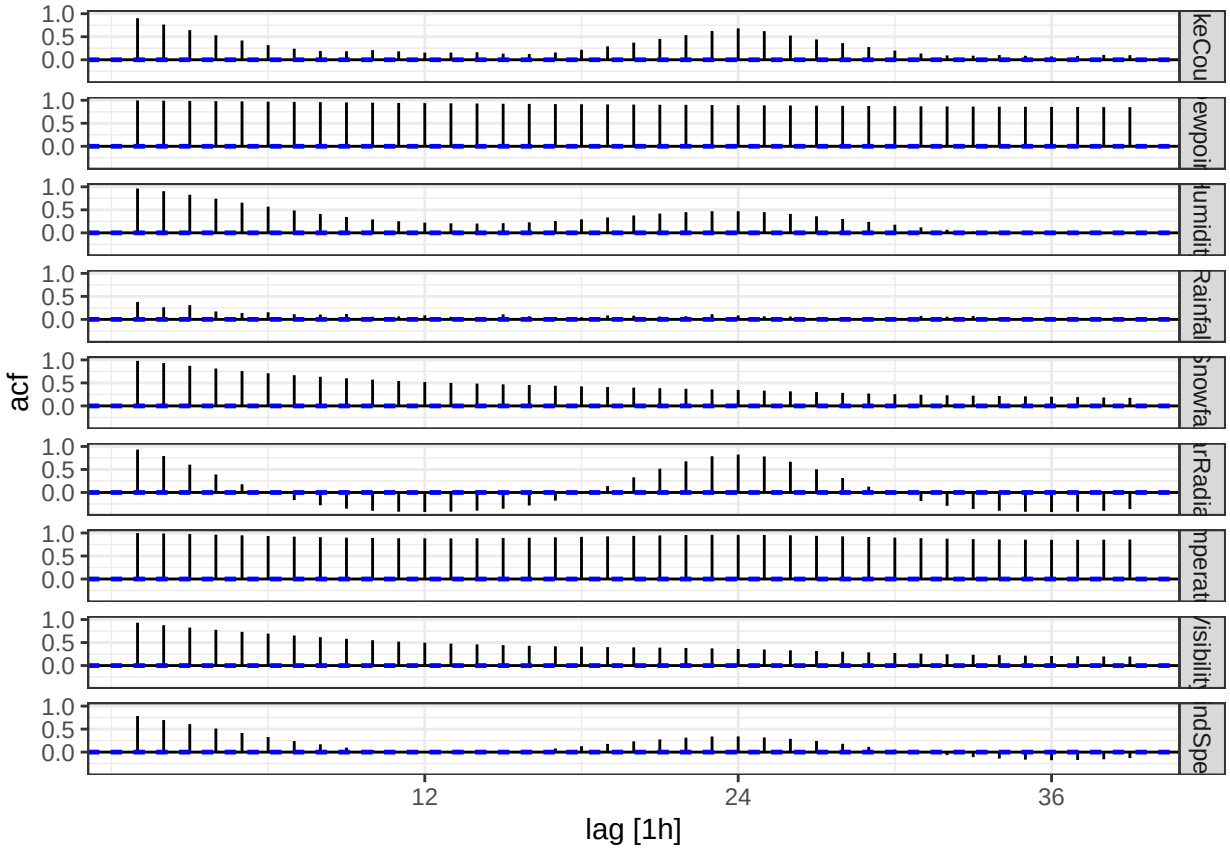

```
hist(bikets$BikeCount)
```




```
bikets_tall %>%
  ggplot(aes(x = Hour, y = Value, color = Measure)) +
  geom_line() +
  facet_grid(rows = vars(Measure), scales = "free_y")
```




```
ACF(bikets_tall, Value) %>% autoplot()
```




```
features(bikets_tall, Value,
         c(unitroot_kpss, unitroot_ndiffs, unitroot_nsdiffs, ljung_box))
```

```
## Warning: 9 errors (1 unique) encountered for feature 1
## [9] The `urca` package must be installed to use this functionality. It can be installed with install
```

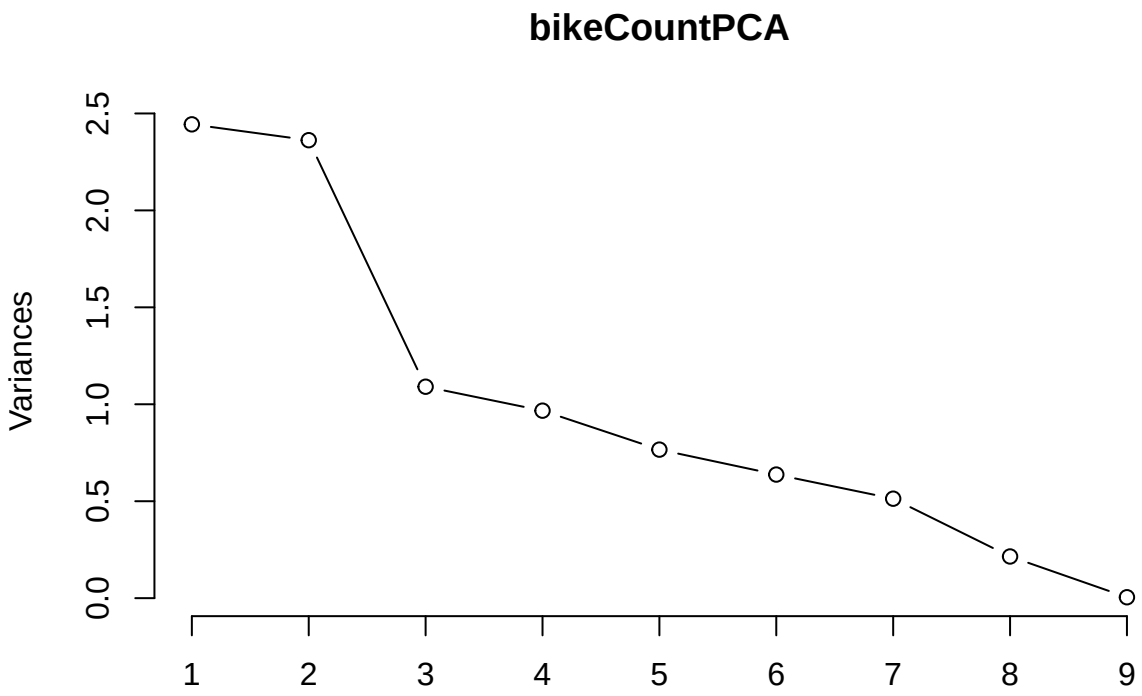
```
## Warning: 9 errors (1 unique) encountered for feature 2
## [9] The `urca` package must be installed to use this functionality. It can be installed with install
```

```
## # A tibble: 9 x 4
##   Measure        nsdiffs lb_stat lb_pvalue
##   <chr>            <int>   <dbl>     <dbl>
## 1 BikeCount            1   7152.         0
## 2 Dewpoint             0   8693.         0
## 3 Humidity             1   8128.         0
## 4 Rainfall             0   1243.         0
## 5 Snowfall             0   8384.         0
## 6 SolarRadiation       1   7616.         0
## 7 Temperature          1   8706.         0
## 8 Visibility           0   7599.         0
## 9 WindSpeed            0   5375.         0
```

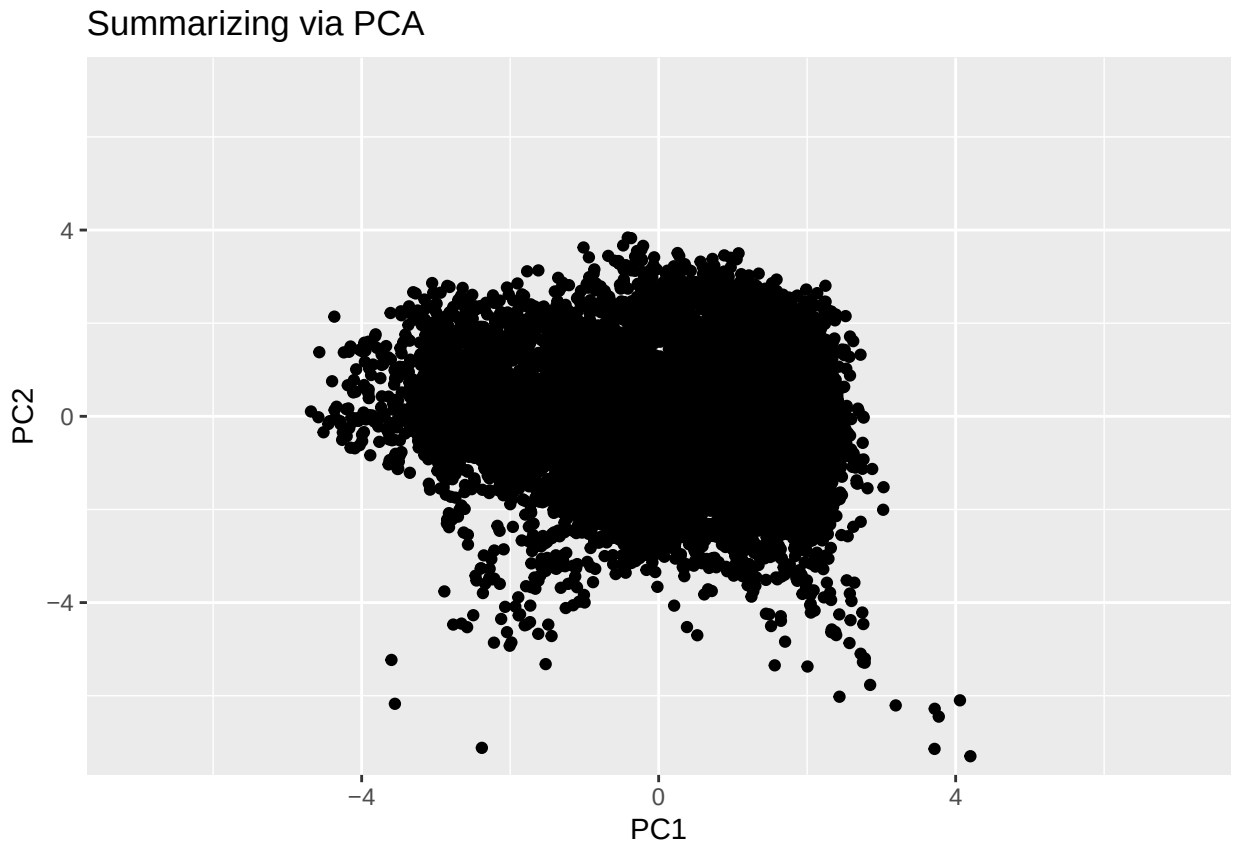
KPSS and Ljung-Box Test both indicate (auto)correlation within each variable.

PCA

```
bike$Hour = as.factor(bike$Hour)

# Correct for skewness for PCA purposes
quantData = bike %>%
  select_if(is.numeric)
skewnessVec = quantData %>% sapply(., e1071::skewness)

skewnessCriterion = abs(skewnessVec)> 1

quantDataSkewedYJ = quantData %>%
  select_if(skewnessCriterion) %>%
  preProcess(method = 'YeoJohnson') %>%
  predict(quantData %>% select_if(skewnessCriterion)) # apply Yeo-Johnson transformation

quantDataNotSkewed = quantData %>%
  select_if(!skewnessCriterion)

quantDataCombined = cbind(quantDataSkewedYJ, quantDataNotSkewed)
#

# start PCA
bikeCountPCA = prcomp(quantDataCombined,scale=TRUE,center=TRUE)
XtransformPC = data.frame(bikeCountPCA$x)

screeplot(bikeCountPCA,type='lines') # First two PCs are informative
```




```
ggplot(data = XtransformPC, aes(x = PC1, y = PC2)) +
  geom_point() +
  coord_cartesian(xlim = c(-7,7), ylim = c(-7,7))+
  ggtitle("Summarizing via PCA")
```




```
# Extreme observations
filter(bike,XtransformPC$PC1 == min(XtransformPC$PC1)) # Bike Count v low, v high Humidity, v low Visib
```

```
## # A tibble: 1 x 14
##   Date       BikeCount Hour  Temperature Humidity WindSpeed Visibility Dewpoint
##   <chr>          <dbl> <fct>       <dbl>    <dbl>     <dbl>      <dbl>    <dbl>
## 1 24/01/2018        34 5           -15.6       29       3.7       2000    -29.6
## # ... with 6 more variables: SolarRadiation <dbl>, Rainfall <dbl>,
## #   Snowfall <dbl>, Seasons <fct>, Holiday <fct>, FunctionalDay <fct>
```

```
filter(bike,XtransformPC$PC2 == max(XtransformPC$PC2)) # v high Humididty
```

```
## # A tibble: 1 x 14
##   Date    BikeCount Hour  Temperature Humidity WindSpeed Visibility Dewpoint
##   <chr>       <dbl> <fct>       <dbl>    <dbl>     <dbl>      <dbl>    <dbl>
## 1 10/4/18       913 14           21.2       35       7.4       1992      5.1
## # ... with 6 more variables: SolarRadiation <dbl>, Rainfall <dbl>,
## #   Snowfall <dbl>, Seasons <fct>, Holiday <fct>, FunctionalDay <fct>
```

Feature Engineering

```
# Extract features from date
bike$Day = factor(format(bikets$Hour, "%d"))
bike$Month = factor(months(bikets$Hour, abbreviate= T))
bike$WeekDay = factor(wday(bikets$Hour, label =F, week_start = 1), ordered = F)

# drop last 3 rows since we're not using those. (Similar to Fold 51 in create_cv_folds.R)
bike = slice(bike, 1:(n() - 3))

# convert categorical features to numeric encoding
factors = c("Seasons", "Holiday", "FunctionalDay", "Day", "Month", "WeekDay")
bike[,factors] = sapply(bike[,factors], unclass)
bike[,factors] <- lapply(bike[,factors], as.factor)
```

Imputation

```
# Imputing values for Humidity using KNN
# BikeCount NA values should be predicted rather than imputed since it is the supervisor and not a feat
###

bike = bike %>%
  mutate(Humidity = ifelse(Humidity == 0, NA, Humidity))%>% # convert 0 to NA before imputing
  select(Humidity) %>%
  preProcess(method='knnImpute') %>% # Note that this automatically centers and scales Humidity
  predict(newdata = bike)
```

Modeling

Time Series

- Prophet
- fasster

```
library(fable.prophet)

holidays <- bikets %>%
  filter(Holiday == "Holiday") %>%
  mutate(ds = as.Date(Hour)) %>%
  distinct(ds) %>%
  mutate(holiday = "holiday")

fit <- bikets %>%
  model(
    mdl = prophet(BikeCount ~ season(period = "day", type = "multiplicative") + season(period = "week",
  )
components(fit) %>% autoplot()
```

Machine Learning

XGBoost

```
#sanity check
#timeSlices <- createTimeSlices(1:nrow(bike),
#                   initialWindow = nrow(bike)-(50*24), horizon = 24, fixedWindow = T, skip = 23)

#tail(bike[timeSlices$train$Training7581,],1)
#bike[timeSlices$test$Testing7557,]
```

- LSTM
- RNN