CquantExam

Libraries

```
#used for handiling dates more easily
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
#using for joining and more data munging
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x lubridate::as.difftime() masks base::as.difftime()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x lubridate::intersect()   masks base::intersect()
## x dplyr::lag()             masks stats::lag()
## x lubridate::setdiff()     masks base::setdiff()
## x lubridate::union()       masks base::union()
```

```
#Library for Graphing
library(ggplot2)
#library for string detecting
library(stringr)
#Used for trying to normalize data in final bonus task
library(BBmisc)
```

```
##
## Attaching package: 'BBmisc'
```

```
## The following objects are masked from 'package:dplyr':
##
##     coalesce, collapse

## The following object is masked from 'package:base':
##
##     isFALSE
```

Task 1

```
#Reading in all files, making sure strings aren't readin as  factors
Price2016 = read.csv("ERCOT_DA_Prices_2016.csv", stringsAsFactors = F)
Price2017 = read.csv("ERCOT_DA_Prices_2017.csv", stringsAsFactors = F)
Price2018 = read.csv("ERCOT_DA_Prices_2018.csv", stringsAsFactors = F)
Price2019 = read.csv("ERCOT_DA_Prices_2019.csv", stringsAsFactors = F)

#Compiling them all into one file
UltimatePrice = rbind(Price2016,Price2017,Price2018,Price2019)
#head(UltimatePrice)
```

Task 2

```
#Creating table average asked in question, using lubridate for the first part
Task2avg = UltimatePrice %>% mutate(year = year(Date), month = month(Date)) %>% group_by( month,Settleme
#head(Task2avg)
```

Task 3

```
#Creating CSV file from table above
write.csv(Task2avg, "AveragePriceByMonth.csv")
```

Task 4

```
# creating new safe datasert to edit/play with
UltimatePrice2 = UltimatePrice
#Creating column for Price Volarility
Volatility = UltimatePrice2 %>% mutate(year = year(Date), month = month(Date)) %>% group_by( month,Sett
  filter(str_detect(SettlementPoint, "HB"),Price > 0) %>% summarize(HourlyVolatility = var(log(Price)))

Volatility = Volatility[,2:4]
#Volatility
```

Task 5

```r
#Writing above table to a CSV file
write.csv(Volatility, "HourlyVolatilityByYear.csv")
```

Task 6

```r
#Grouping by year and finding highest volatility
maxVolatility = Volatility %>% group_by(year) %>% filter(HourlyVolatility == max(HourlyVolatility))
```

Task 7

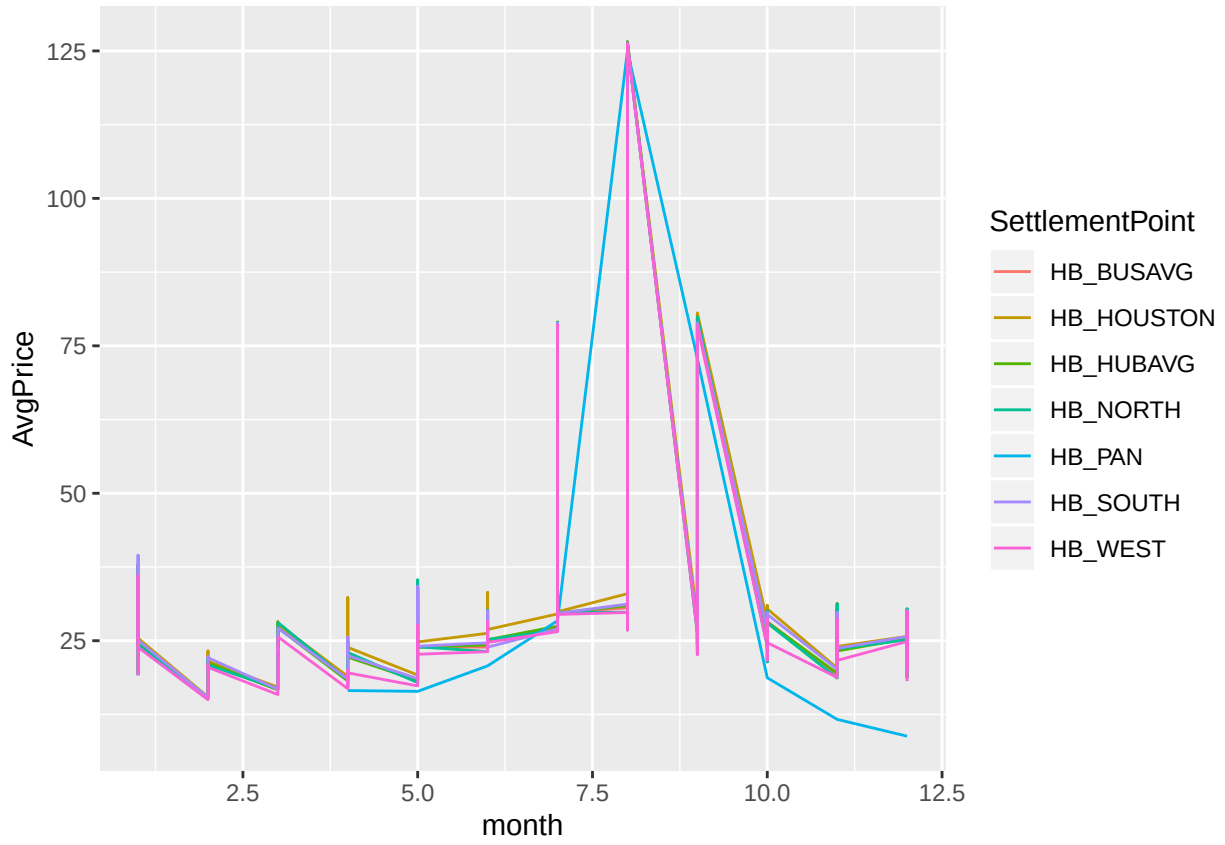
```r
#Looking at data format I'm supposed to immitate
example = read.csv("Spot_ISONE_Node1.csv", stringsAsFactors = F)
#example
#head(UltimatePrice)
#USing this to see how many settlement points there are and confirming it is 15. Then using the names f
Settlementschar = unique(UltimatePrice$SettlementPoint)
#Settlementschar[1]
#Creating for loop for all Settlement hubs
  for(i in 1:15){
    #Formatting data like the example file
    Newfile = UltimatePrice %>% filter(SettlementPoint == Settlementschar[i]) %>% mutate(hour = hour(Da
    #Writing out all the data and saving it to specified folder in question. Take out path part when u
    path = "C:/Users/paul.merica/Documents/formattedSpotHistory"
     write.csv(Newfile, file.path(path,paste0("spot_", Settlementschar[i], ".csv")))
  }
```

Bonus Task 1: Mean Plots

```r
#will come back if time permits, would just do more labelling and fix that x axis. But this stuff is ti
#Make first plot
Hub_boys= Task2avg %>% filter(str_detect(SettlementPoint, "HB"))
summary(Hub_boys)
```

```
##      month        SettlementPoint         year         AvgPrice
##  Min.   : 1.000   Length:297         Min.   :2016   Min.   :  8.817
##  1st Qu.: 4.000   Class :character   1st Qu.:2017   1st Qu.: 21.660
##  Median : 7.000   Mode  :character   Median :2018   Median : 25.086
##  Mean   : 6.545                      Mean   :2018   Mean   : 29.595
##  3rd Qu.:10.000                      3rd Qu.:2019   3rd Qu.: 29.782
##  Max.   :12.000                      Max.   :2019   Max.   :126.625
```

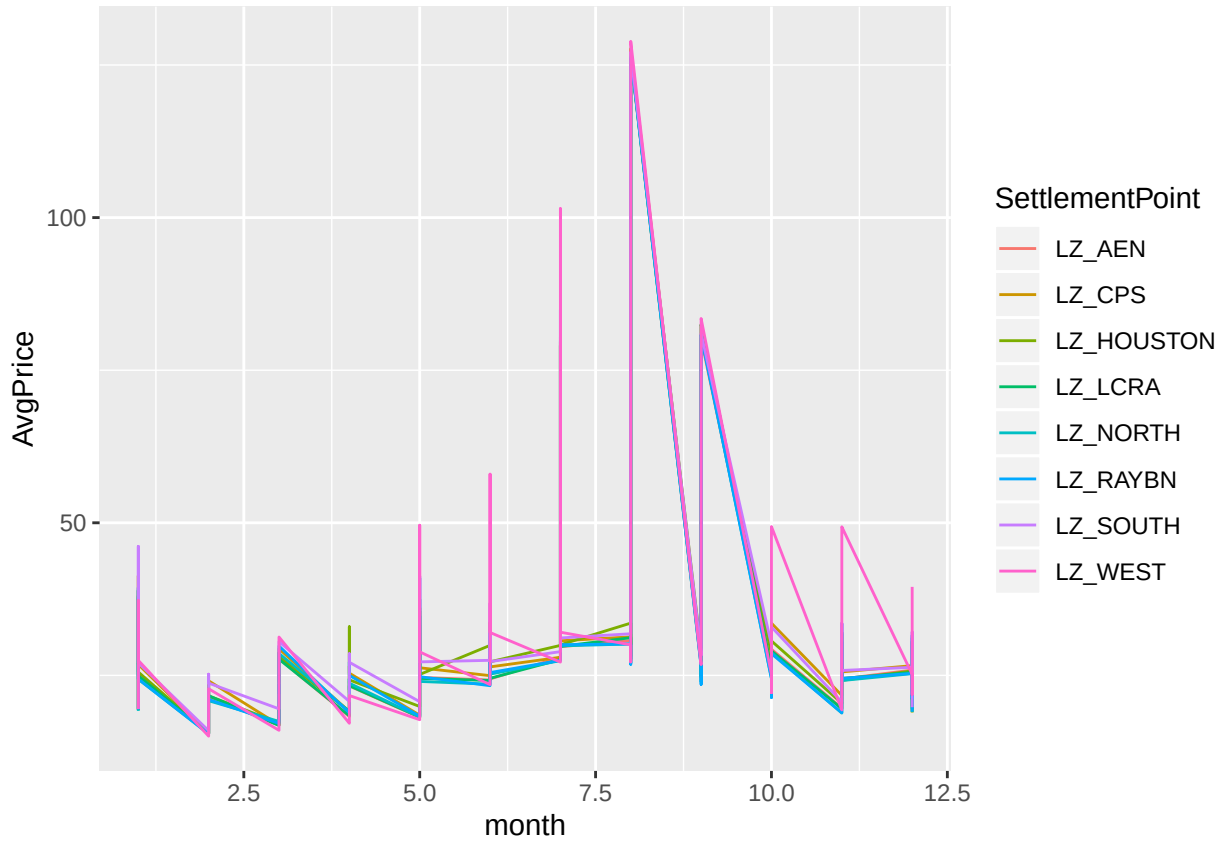
```r
ggplot(Hub_boys, aes(x = month, y = AvgPrice, color = SettlementPoint)) + geom_line()
```




```
#Making the second plot
LZ_boys= Task2avg %>% filter(str_detect(SettlementPoint, "LZ"))
summary(LZ_boys)
```

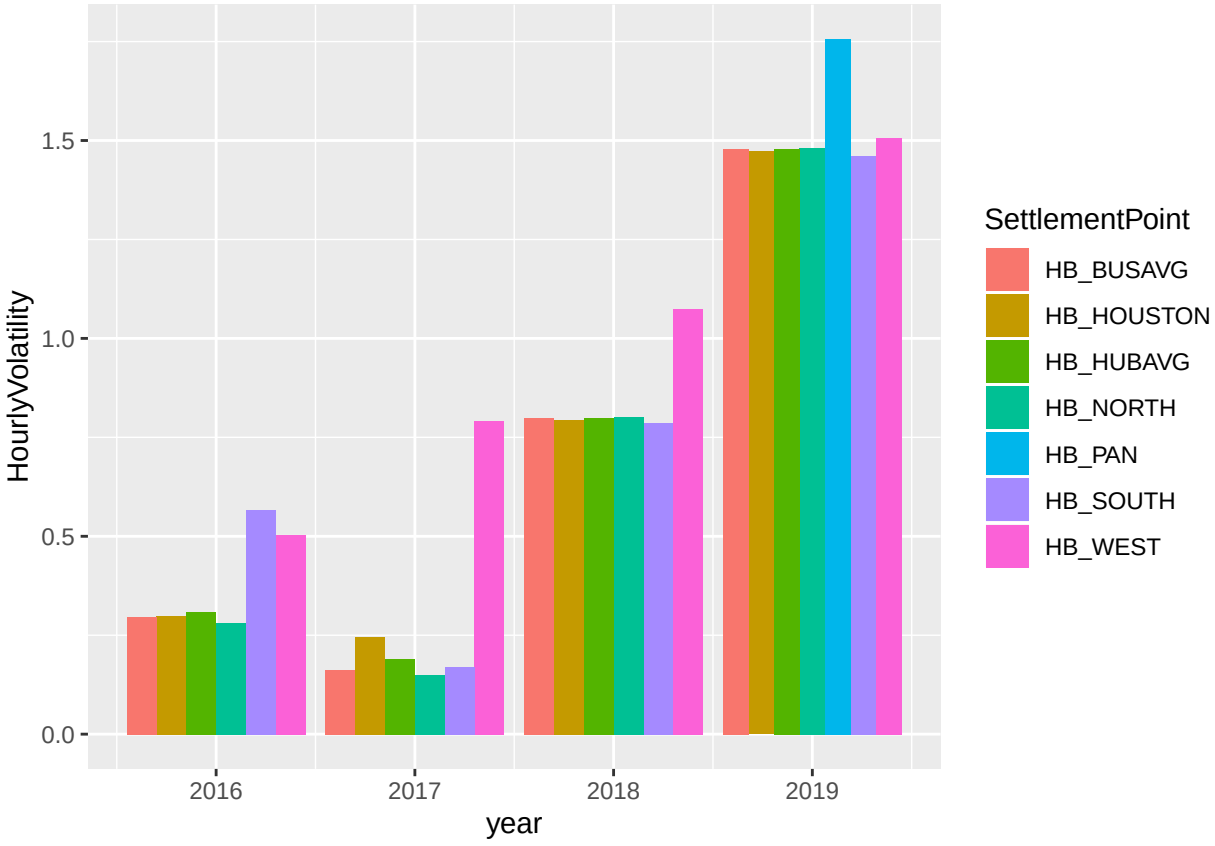
```
##      month       SettlementPoint          year         AvgPrice
##  Min.   : 1.00   Length:384         Min.   :2016   Min.   : 15.04
##  1st Qu.: 3.75   Class :character   1st Qu.:2017   1st Qu.: 22.36
##  Median : 6.50   Mode  :character   Median :2018   Median : 26.43
##  Mean   : 6.50                      Mean   :2018   Mean   : 30.74
##  3rd Qu.: 9.25                      3rd Qu.:2018   3rd Qu.: 30.68
##  Max.   :12.00                      Max.   :2019   Max.   :128.88
```

```
ggplot(LZ_boys, aes(x = month, y = AvgPrice, color = SettlementPoint)) + geom_line()
```



Bonus Task 2: Volatility Plots

```
#Same as above with new stuff aka the Volatility data set, would come back and clean it up a bit much l
#summary(Hub_boysVol)
ggplot(Volatility, aes(x = year, y = HourlyVolatility, fill = SettlementPoint)) + geom_bar(position="do
```



Bonus Task 3: Hourly Shape Profile Computation

```
# Figuring out how this thing is supposed to work. Doing it for one day first then we can translate it

Test = UltimatePrice %>% filter(SettlementPoint == "HB_BUSAVG") %>% mutate(year = year(Date), hour = hou

Test
```

```
## # A tibble: 35,060 x 8
## # Groups:   year, day, hour [2,976]
##    Date           SettlementPoint Price  year  hour month   day PriceNormal
##    <chr>          <chr>           <dbl> <dbl> <int> <dbl> <int>       <dbl>
##  1 2016-01-01 00~ HB_BUSAVG        18.4  2016     0     1     1       0.712
##  2 2016-01-01 01~ HB_BUSAVG        16.2  2016     1     1     1       0.559
##  3 2016-01-01 02~ HB_BUSAVG        15.6  2016     2     1     1       0.612
##  4 2016-01-01 03~ HB_BUSAVG        15.6  2016     3     1     1       0.677
##  5 2016-01-01 04~ HB_BUSAVG        15.9  2016     4     1     1       0.579
##  6 2016-01-01 05~ HB_BUSAVG        16.7  2016     5     1     1       0.285
##  7 2016-01-01 06~ HB_BUSAVG        19.0  2016     6     1     1      -0.114
##  8 2016-01-01 07~ HB_BUSAVG        19.2  2016     7     1     1       0.0643
##  9 2016-01-01 08~ HB_BUSAVG        19.1  2016     8     1     1       0.0852
## 10 2016-01-01 09~ HB_BUSAVG        19.7  2016     9     1     1       0.0476
## # ... with 35,050 more rows
```

```r
#Testing to see if it sums to one. Spoiler it does not.
sum(Test[1:12,8])/12
```

```
## [1] 0.2730357
```