Projeto Data Mining I

Lucas Parada, Ana Catarina Monteiro, Lucas de Paula

11/16/2019

Introdução

Definição do problema

Data importation, clean-up and pre-processing

Importando os Dados

Inicialmente importamos todos as bibliotecas que utilizaresmos neste trabalho Foi importado o dataset "PRSA_Data_Aotizhongxin_20130301-20170228" - Foi utilizado o DataFrame do R para manipular os dados, pois este tipo de estrutura de dados possui um conjunto de funcionalidades e ferrantas que auxiliam neste processo

Analisando os dados

Utilizando a função "summary" da linguagem R foi feita uma análise dos dados.

```
##        No              year          month             day        
##  Min.   :    1   Min.   :2013   Min.   : 1.000   Min.   : 1.00  
##  1st Qu.: 8767   1st Qu.:2014   1st Qu.: 4.000   1st Qu.: 8.00  
##  Median :17532   Median :2015   Median : 7.000   Median :16.00  
##  Mean   :17532   Mean   :2015   Mean   : 6.523   Mean   :15.73  
##  3rd Qu.:26298   3rd Qu.:2016   3rd Qu.:10.000   3rd Qu.:23.00  
##  Max.   :35064   Max.   :2017   Max.   :12.000   Max.   :31.00  
##                                                                 
##       hour           PM2.5             PM10            SO2          
##  Min.   : 0.00   Min.   :  3.00   Min.   :  2.0   Min.   :  0.2856  
##  1st Qu.: 5.75   1st Qu.: 22.00   1st Qu.: 38.0   1st Qu.:  3.0000  
##  Median :11.50   Median : 58.00   Median : 87.0   Median :  9.0000  
##  Mean   :11.50   Mean   : 82.77   Mean   :110.1   Mean   : 17.3759  
##  3rd Qu.:17.25   3rd Qu.:114.00   3rd Qu.:155.0   3rd Qu.: 21.0000  
##  Max.   :23.00   Max.   :898.00   Max.   :984.0   Max.   :341.0000  
##                  NA's   :925      NA's   :718     NA's   :935       
##       NO2               CO              O3               TEMP        
##  Min.   :  2.00   Min.   :  100   Min.   :  0.2142   Min.   :-16.80  
##  1st Qu.: 30.00   1st Qu.:  500   1st Qu.:  8.0000   1st Qu.:  3.10  
##  Median : 53.00   Median :  900   Median : 42.0000   Median : 14.50  
##  Mean   : 59.31   Mean   : 1263   Mean   : 56.3534   Mean   : 13.58  
##  3rd Qu.: 82.00   3rd Qu.: 1500   3rd Qu.: 82.0000   3rd Qu.: 23.30  
##  Max.   :290.00   Max.   :10000   Max.   :423.0000   Max.   : 40.50  
##  NA's   :1023     NA's   :1776    NA's   :1719       NA's   :20      
##       PRES             DEWP              RAIN               wd        
##  Min.   : 985.9   Min.   :-35.300   Min.   : 0.00000   NE     : 5140  
##  1st Qu.:1003.3   1st Qu.: -8.100   1st Qu.: 0.00000   ENE    : 3950  
##  Median :1011.4   Median :  3.800   Median : 0.00000   SW     : 3359  
##  Mean   :1011.8   Mean   :  3.123   Mean   : 0.06742   E      : 2608  
##  3rd Qu.:1020.1   3rd Qu.: 15.600   3rd Qu.: 0.00000   NNE    : 2445  
```

```
##  Max.   :1042.0   Max.   : 28.500   Max.   :72.50000   (Other):17481
##  NA's   :20        NA's   :20        NA's   :20         NA's   :   81
##       WSPM                station
##  Min.   : 0.000   Aotizhongxin:35064
##  1st Qu.: 0.900
##  Median : 1.400
##  Mean   : 1.708
##  3rd Qu.: 2.200
##  Max.   :11.200
##  NA's   :14
```

Começamos por verificar se existia algum dia em falta no dataframe e vimos que nao. Samendo que o dataset possui os valores referentes a 4 anos completos especificados por hora então sevem existir $(4365+1)24 = 35064$ rows

```
  #samendo que o dataset possui os valores referentes a 4 anos completos especificados por hora então s
dim(df)
```

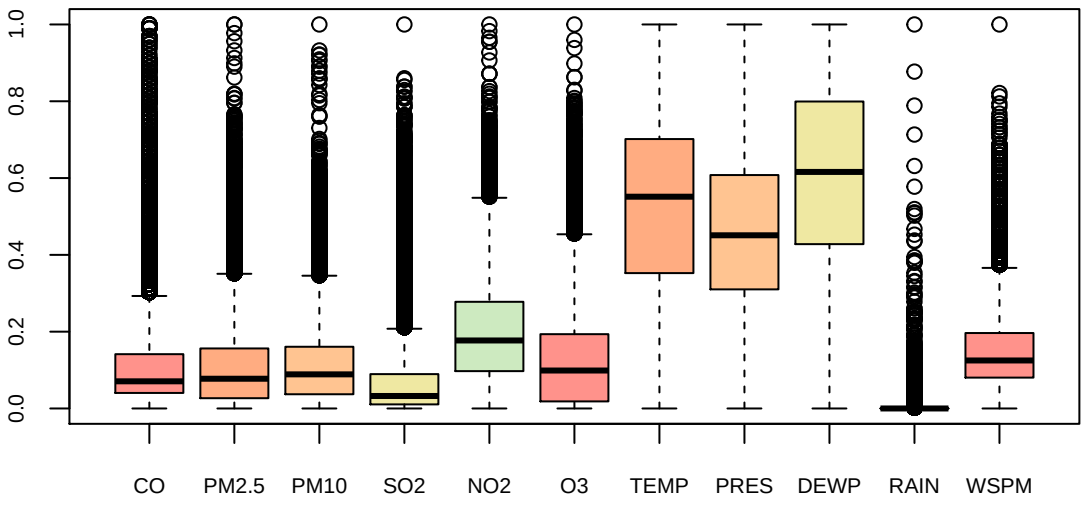
```
## [1] 35064    18
```

Outliers

Seguimos com a analise da existencia de outliers em variáveis numericas. Começando por ver fazer a análise por variavel como um todo, em seguida fizemos a análise por variavel tendo em conta a estação do ano e por último por variavel por mês.

Como um todo

```
col_list = c("CO", "PM2.5", "PM10", "SO2", "NO2", "CO", "O3", "TEMP", "PRES", "DEWP", "RAIN", "WSPM")

df_ALL_boxplot <- df %>% select(col_list)

# remove rows with missing values
df_ALL_boxplot <- df_ALL_boxplot[ complete.cases(df_ALL_boxplot), ]

preproc<-preProcess(df_ALL_boxplot, method=c("range"))
df_ALL_boxplot <- predict(preproc, df_ALL_boxplot)

boxplot(x = df_ALL_boxplot, col = color_list, cex.axis=0.7)
```

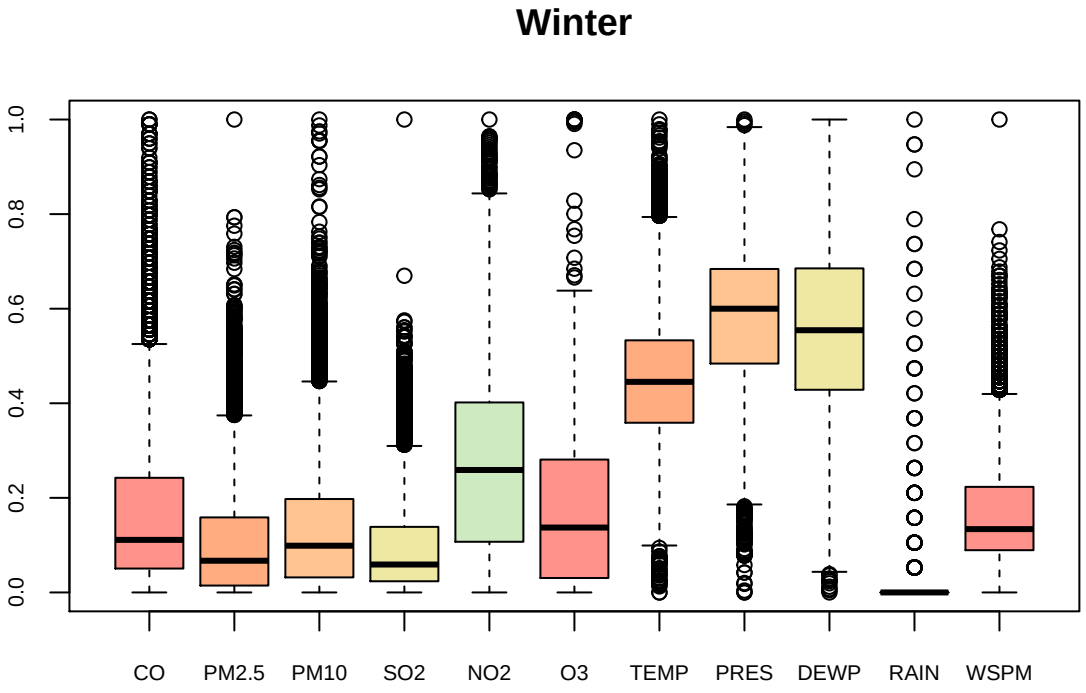


Como

Por estação do ano

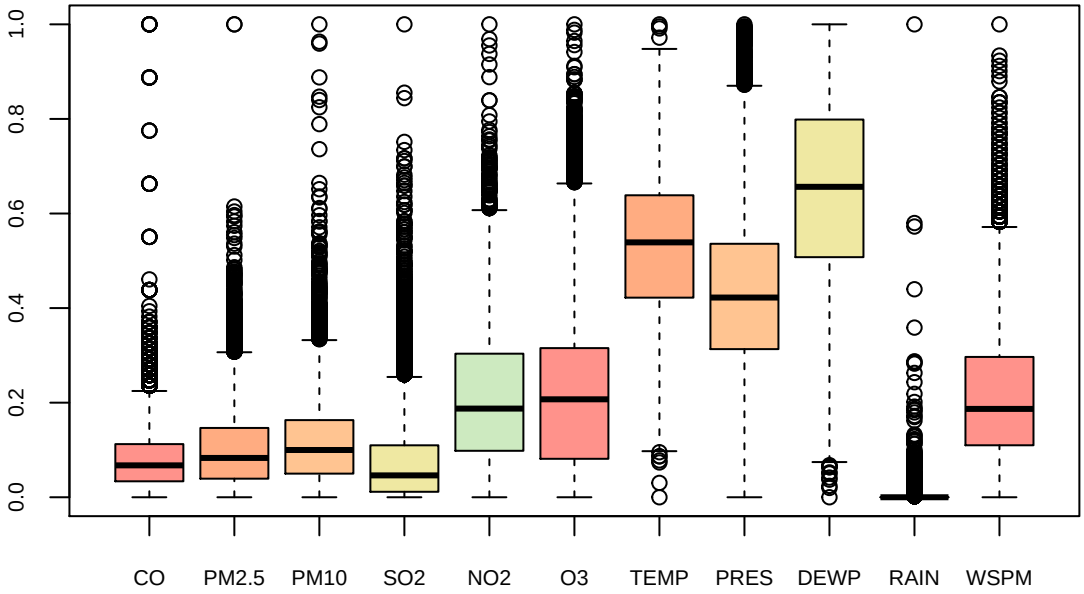
```
#season <- function(month, day){
  #print(month)
  # if(month == 12 & day >= 21)
#   return('Winter')
#  if(month == 1 | month == 2)
#   return('Winter')
#  if(month == 3 & day < 20)
#   return('Winter')
#  if(month == 3 & day >= 20)
#   return('Spring')
#  if(month == 4 | month == 5)
#   return('Spring')
#  if(month == 6 & day < 21)
#   return('Spring')
#  if(month == 6 & day >= 21)
#   return('Summer')
#  if(month == 7 | month == 8)
#   return('Summer')
#  if(month == 9 & day < 21)
#   return('Summer')
#  if(month == 9 & day >= 21)
#     return('Autumn')
#  if(month == 10 | month == 11)
#     return('Autumn')
#  if(month == 12 & day < 2)
#     return('Autumn')
#}

col_list <- c("season","CO", "PM2.5", "PM10", "SO2", "NO2", "CO", "O3", "TEMP", "PRES", "DEWP", "RAIN",

df_seasons_boxplot <- df %>% mutate(season =
  ifelse(month == 12 & day >= 21, 'Winter',
  ifelse(month == 1 | month == 2, 'Winter',
  ifelse(month == 3 & day < 20, 'Winter',

  ifelse(month == 3 & day >= 20, 'Spring',
  ifelse(month == 4 | month == 5, 'Spring',
  ifelse(month == 6 & day < 21, 'Spring',

  ifelse(month == 6 & day >= 21, 'Summer',
  ifelse(month == 7 | month == 8, 'Summer',
  ifelse(month == 9 & day < 21, 'Summer',

  ifelse(month == 9 & day >= 21, 'Autumn',
  ifelse(month == 10 | month == 11, 'Autumn',
  ifelse(month == 12 & day < 21, 'Autumn',
         0))))))))))))
) %>% select(col_list)

#df_seasons_boxplot %>% group_by(season) %>% boxplot(col = color_list, cex.axis=0.7)

for(s in df_seasons_boxplot$season %>% unique()){
```

```
  df_aux <- df_seasons_boxplot %>% filter(season == s) %>% select(-season)

  preproc<-preProcess(df_aux, method=c("range"))
  norm2 <- predict(preproc, df_aux)

  norm2 %>% boxplot(main=s, col = color_list, cex.axis=0.7)
  print('')


}
```



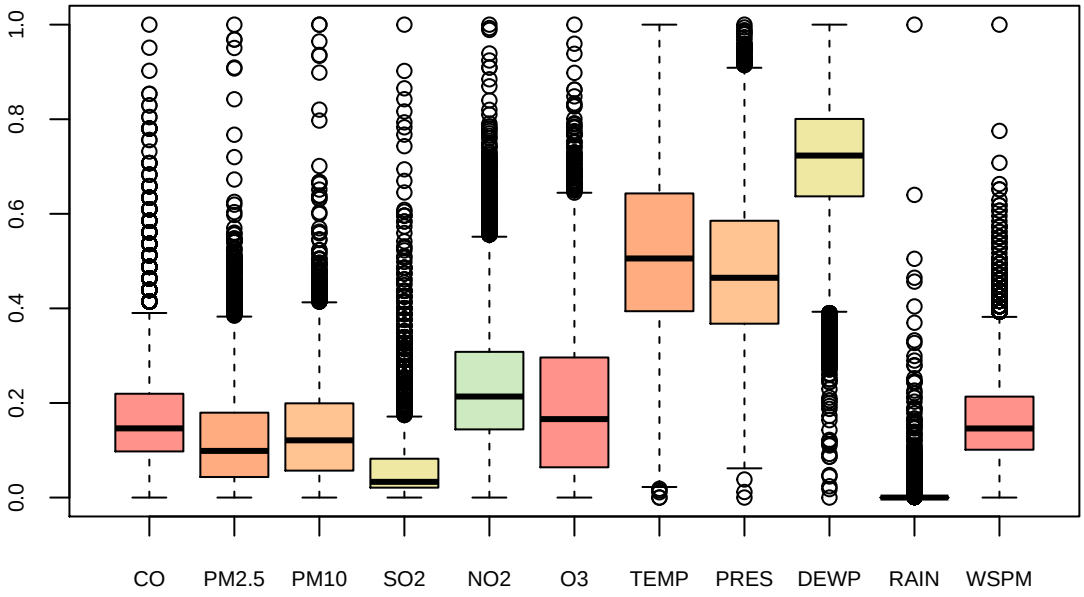

```
## [1] ""
```

Spring

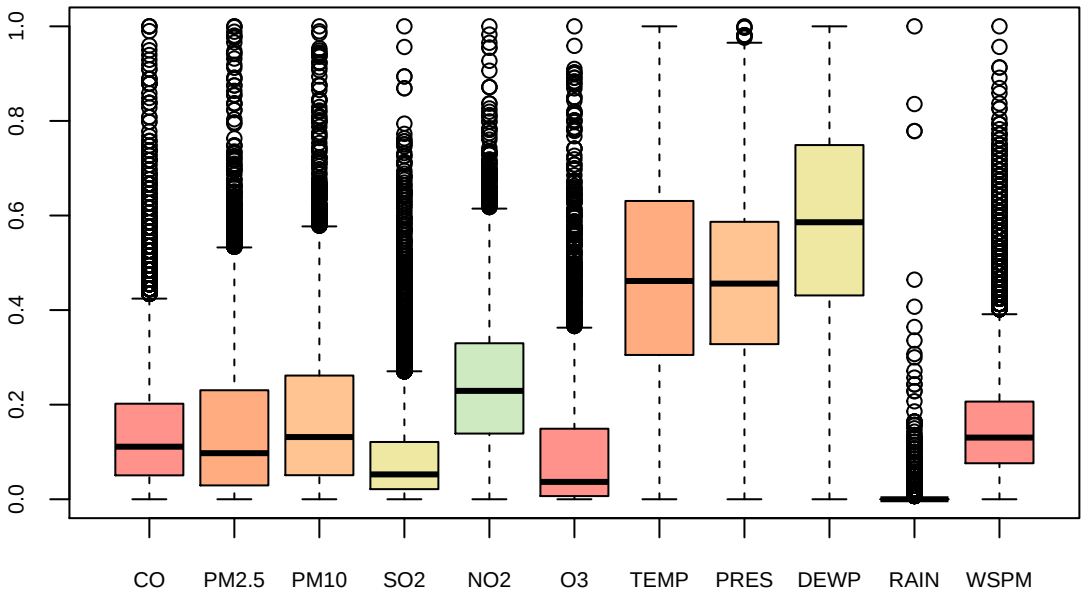



```
## [1] ""
```

Summer




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




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

Por mês

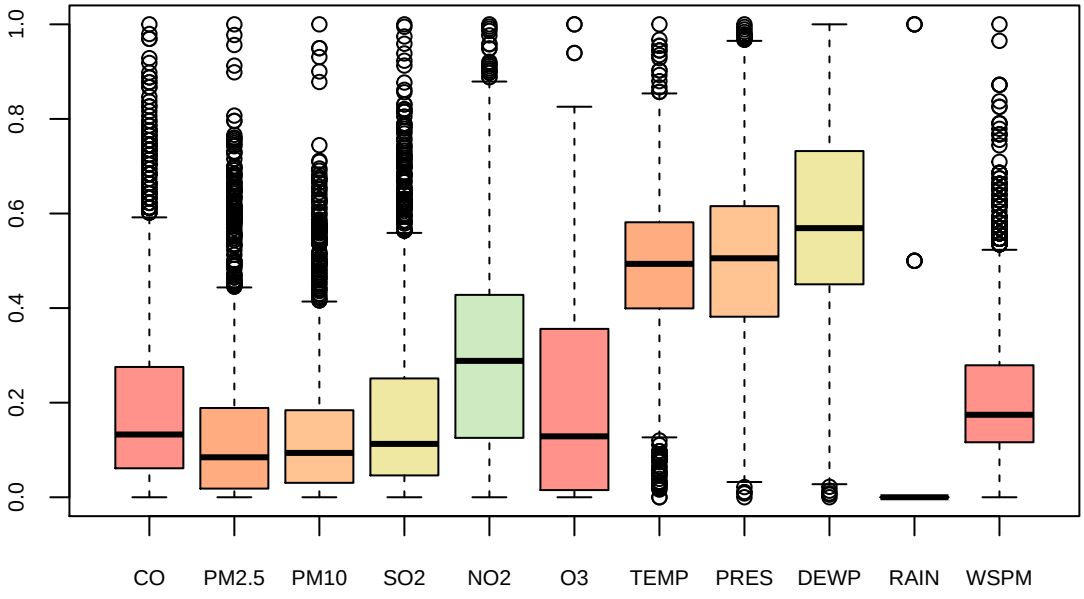
```
col_list = c("month", "CO", "PM2.5", "PM10", "SO2", "NO2", "CO", "O3", "TEMP", "PRES", "DEWP", "RAIN",
df_month_boxplot <- df %>% select(col_list)

month_names <- c("January", "February", "March", "April", "May", "June", "July", "August", "September",

for(m in df_month_boxplot$month %>% unique() %>% sort()){
  df_aux <- df_month_boxplot %>% filter(month == m) %>% select(-month)

  preproc<-preProcess(df_aux, method=c("range"))
  norm2 <- predict(preproc, df_aux)

  par(cxy = c(.5,.5))

  norm2 %>% boxplot(main=month_names[m],col = color_list, cex.axis=0.7)
  print('')
   #df_month_boxplot %>% filter(month == m) %>% select(-month) %>% scale() %>% boxplot(main=month_names
}
```

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

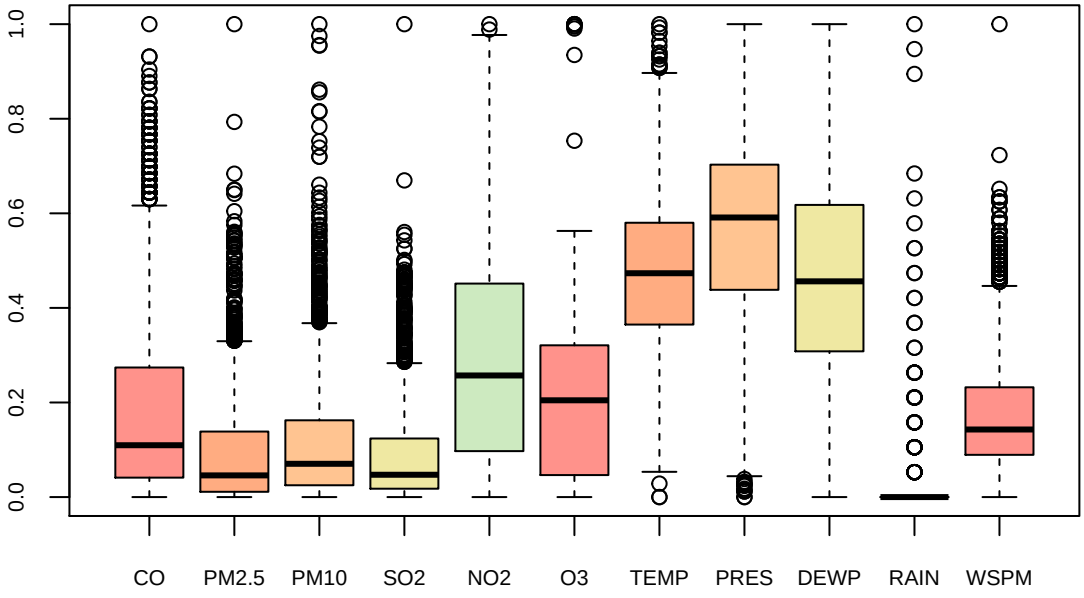
January




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

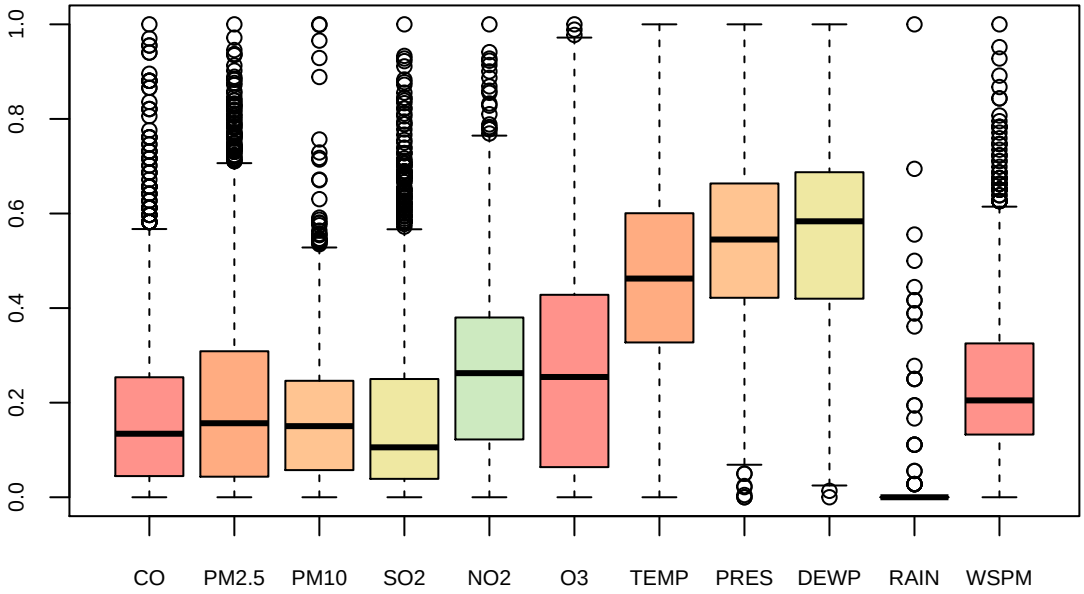
February




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

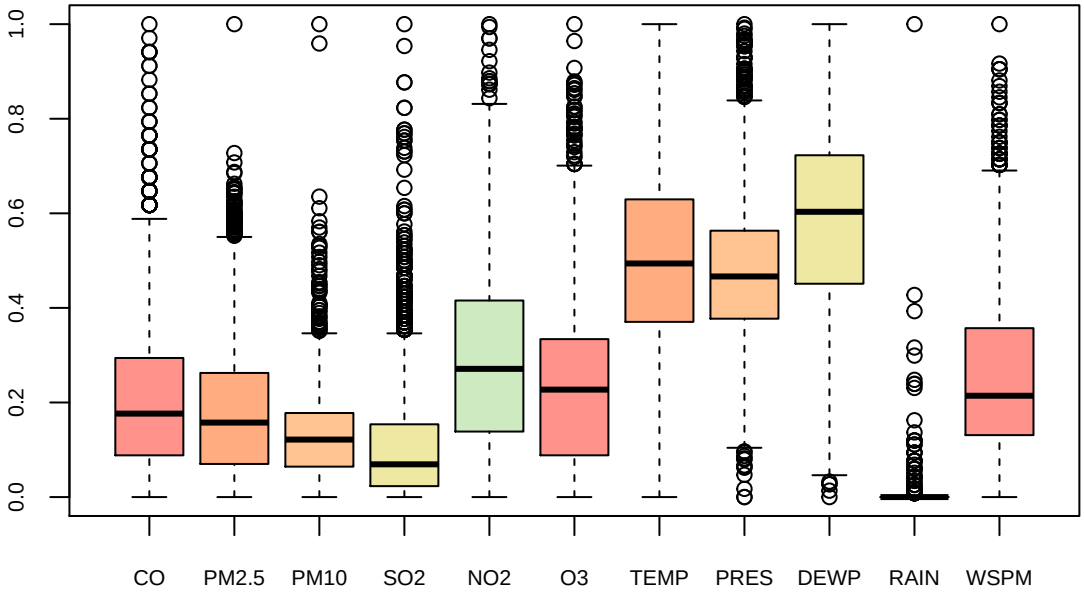
March




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

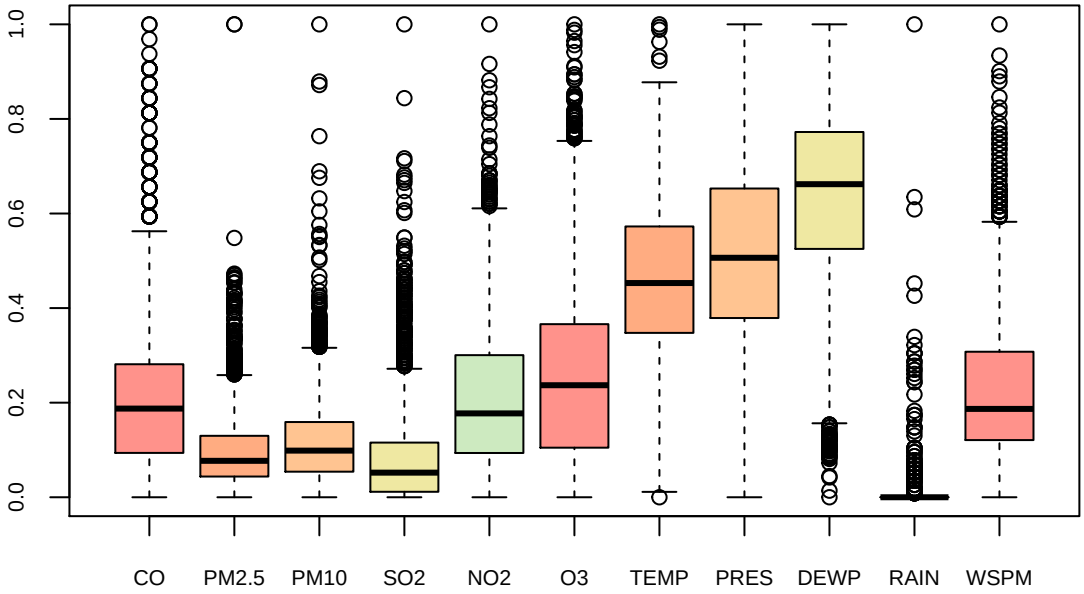
April




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

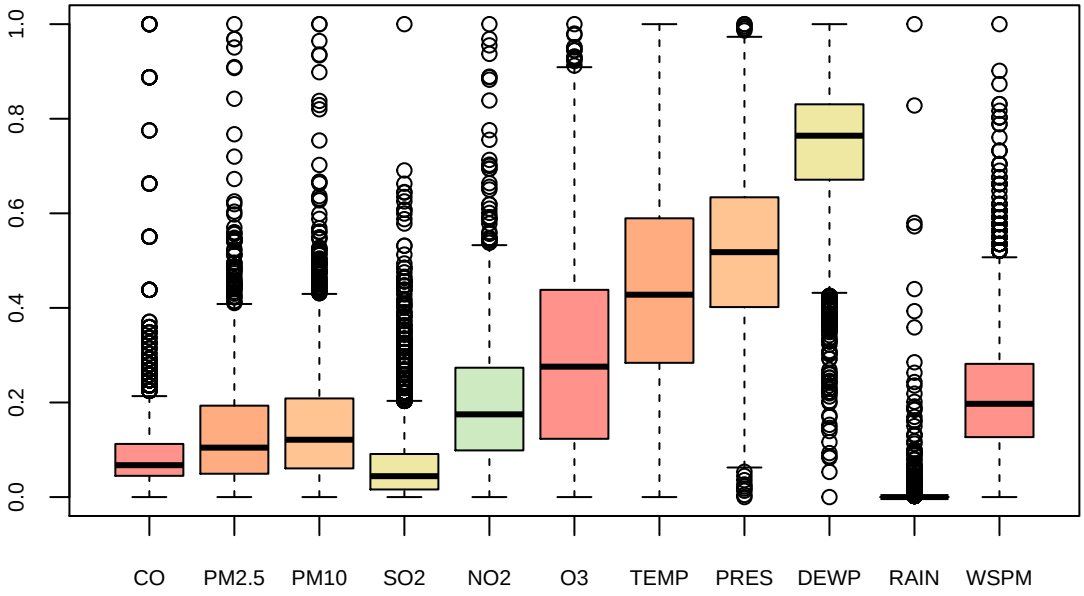
May




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

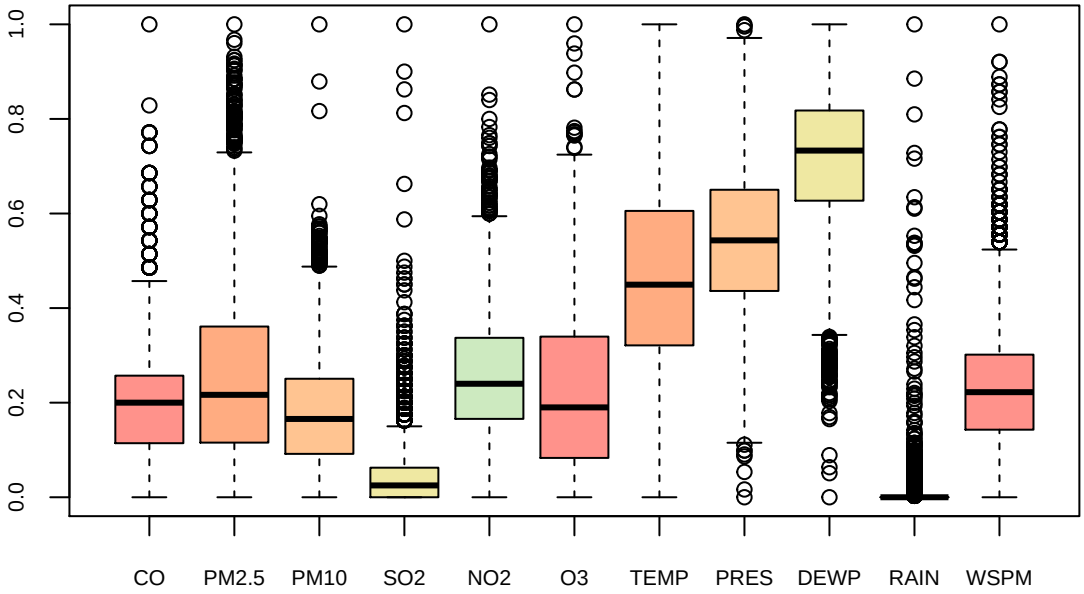
June




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

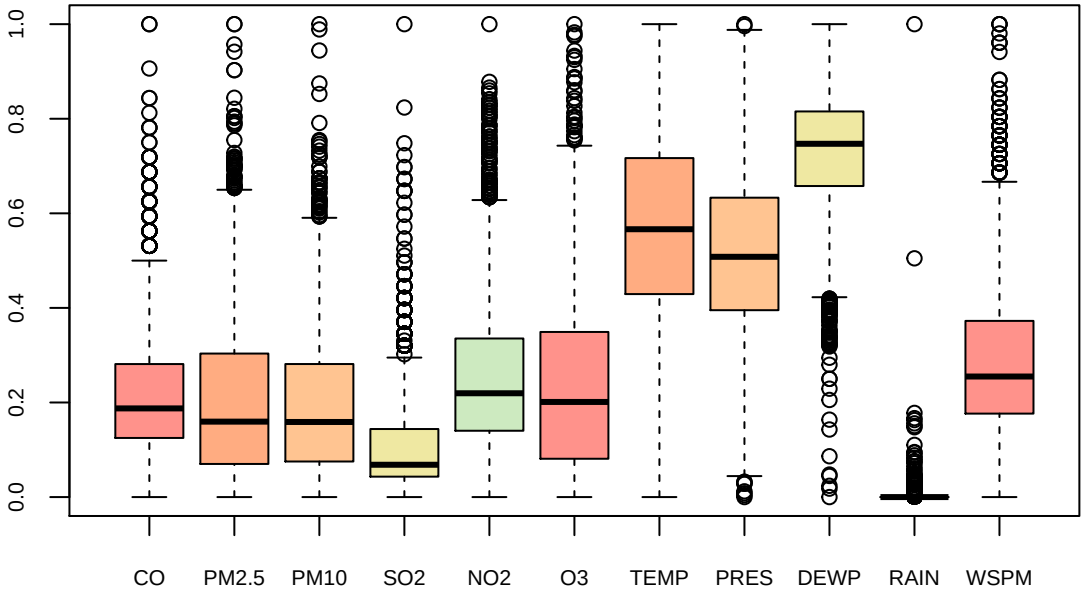
July




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

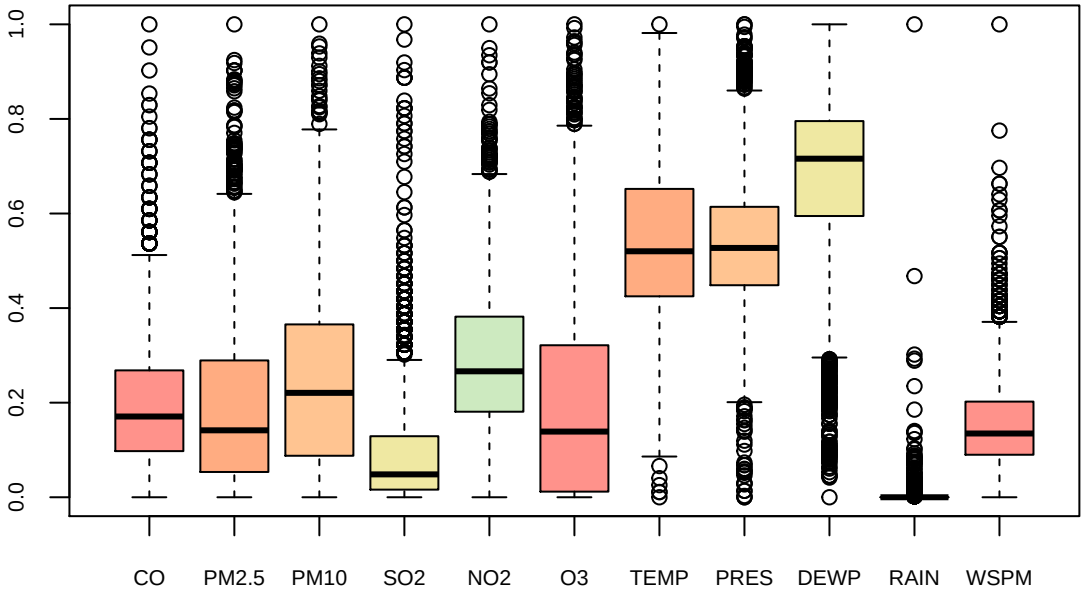
August




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

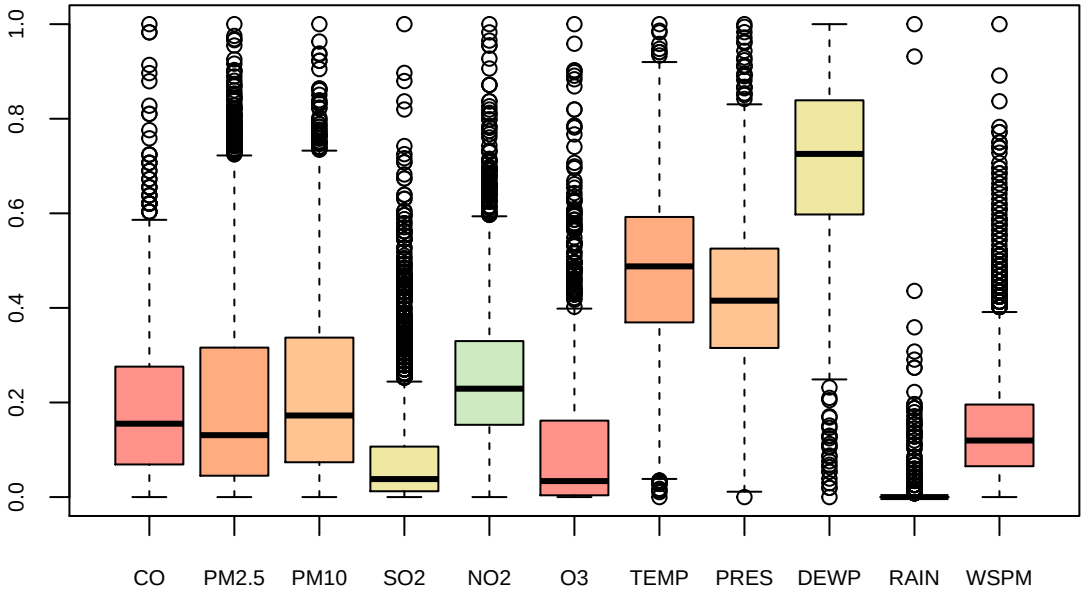
September




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

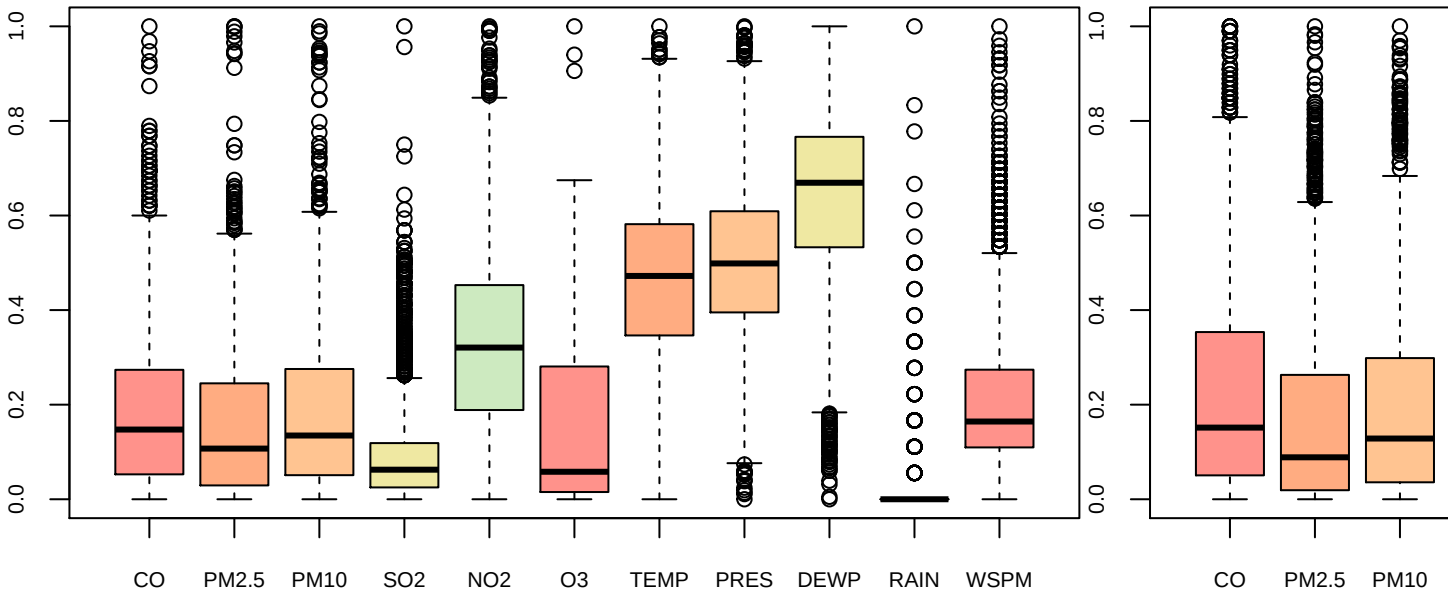
```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```

October




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

```
## Warning in par(cxy = c(0.5, 0.5)): graphical parameter "cxy" cannot be set
```




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

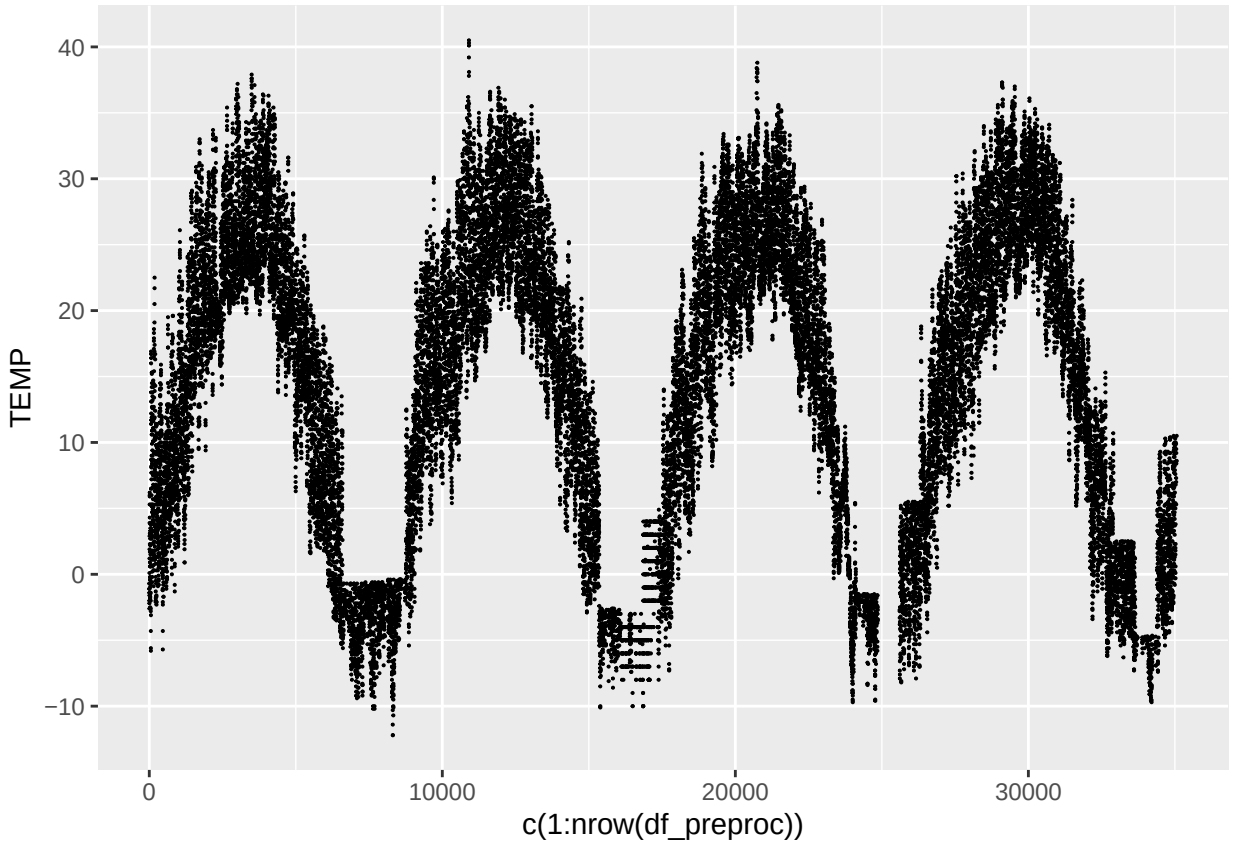
```
#df_month_boxplot %>% group_by(month) %>% select(-month) %>% scale() %>% boxplot()

#df %>% group_by(month) %>% ggplot(aes(x = month, y = TEMP)) + geom_boxplot() + facet_wrap(~year)
```

Tratamento de outliers

```
#identificar se é melhorretirar os outliers como um todo, por estação do ano ou por mês

#
func_quart <- function(val){
  qr1 <- as.numeric(summary(val)['1rd Qu.'])
  qr3 <- as.numeric(summary(val)['3rd Qu.'])
  iqr <- 1.5 * (qr3 - qr1)

  if(is.na(val) )
    return(T)

  if( val>( qr3 + iqr ) )
    return(T)

  if( val<(qr1 - iqr) )
     return(T)

  return( F )
}

df_preproc <- df
df_preproc <- df_preproc %>% group_by(year, month) %>%
```

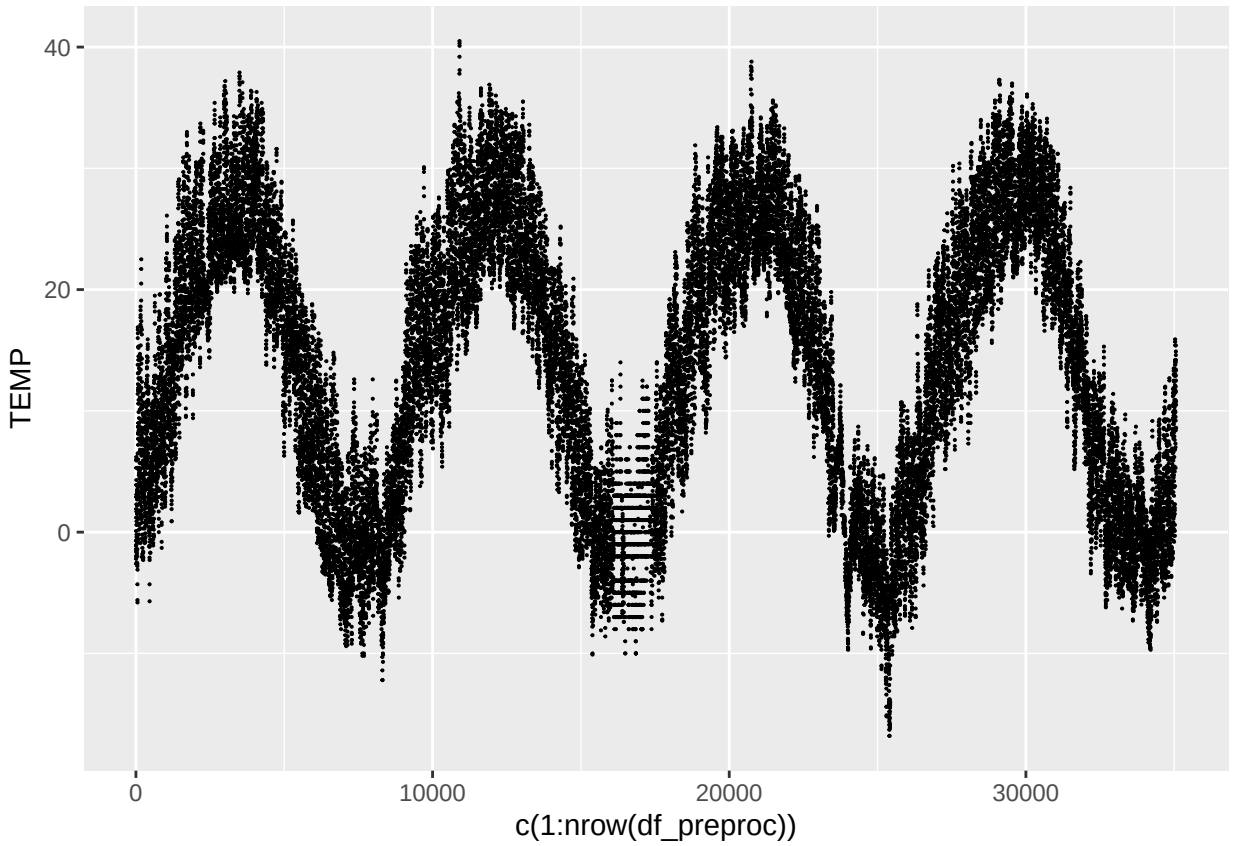
```
      mutate(CO =    ifelse(CO >    4*mean(CO, na.rm = T),   NA,  CO)) %>%
      mutate(PM2.5 = ifelse(PM2.5 > 4*mean(PM2.5, na.rm = T),NA,  PM2.5)) %>%
      mutate(PM10 =  ifelse(PM10 >  4*mean(PM10, na.rm = T), NA,  PM10)) %>%
      mutate(SO2 =   ifelse(SO2 >   4*mean(SO2, na.rm = T),  NA,  SO2)) %>%
      mutate(NO2 =   ifelse(NO2 >   4*mean(NO2, na.rm = T),  NA,  NO2)) %>%
      mutate(O3 =    ifelse(O3 >    4*mean(O3, na.rm = T),   NA,  O3)) %>%
      mutate(TEMP =  ifelse(TEMP >  4*mean(TEMP, na.rm = T), NA,  TEMP)) %>%
      mutate(PRES =  ifelse(PRES >  4*mean(PRES, na.rm = T), NA,  PRES)) %>%
      mutate(DEWP =  ifelse(abs(DEWP) >  4*abs(mean(PRES, na.rm = T)), NA,  DEWP)) %>%
      #mutate(DEWP =  ifelse(DEWP > func_quart(DEWP),  NA,  DEWP)) %>%
      #mutate(RAIN =  ifelse(RAIN >  4*mean(RAIN, na.rm = T), NA,  RAIN)) %>%
      #mutate(WSPM =  ifelse(WSPM >  4*mean(WSPM, na.rm = T), NA,  WSPM))  %>%
      ungroup()

df_preproc %>% ggplot(aes(c(1:nrow( df_preproc )), TEMP)) +
  geom_point(size=.05)
```

```
## Warning: Removed 4489 rows containing missing values (geom_point).
```




```
df %>% ggplot(aes(c(1:nrow( df_preproc )), TEMP)) +
  geom_point(size=.05)
```

```
## Warning: Removed 20 rows containing missing values (geom_point).
```




```
  #facet_wrap(~season ) +
#df_preproc %>% filter(!is.na(CO)) %>% ggplot(aes(c(1:nrow( filter(df_preproc, !is.na(CO)))*4), CO)) + 
#  facet_wrap(~month )+
  #facet_wrap( ~ month) +
#  labs(subtitle="CO", title="Jittered Points")
```

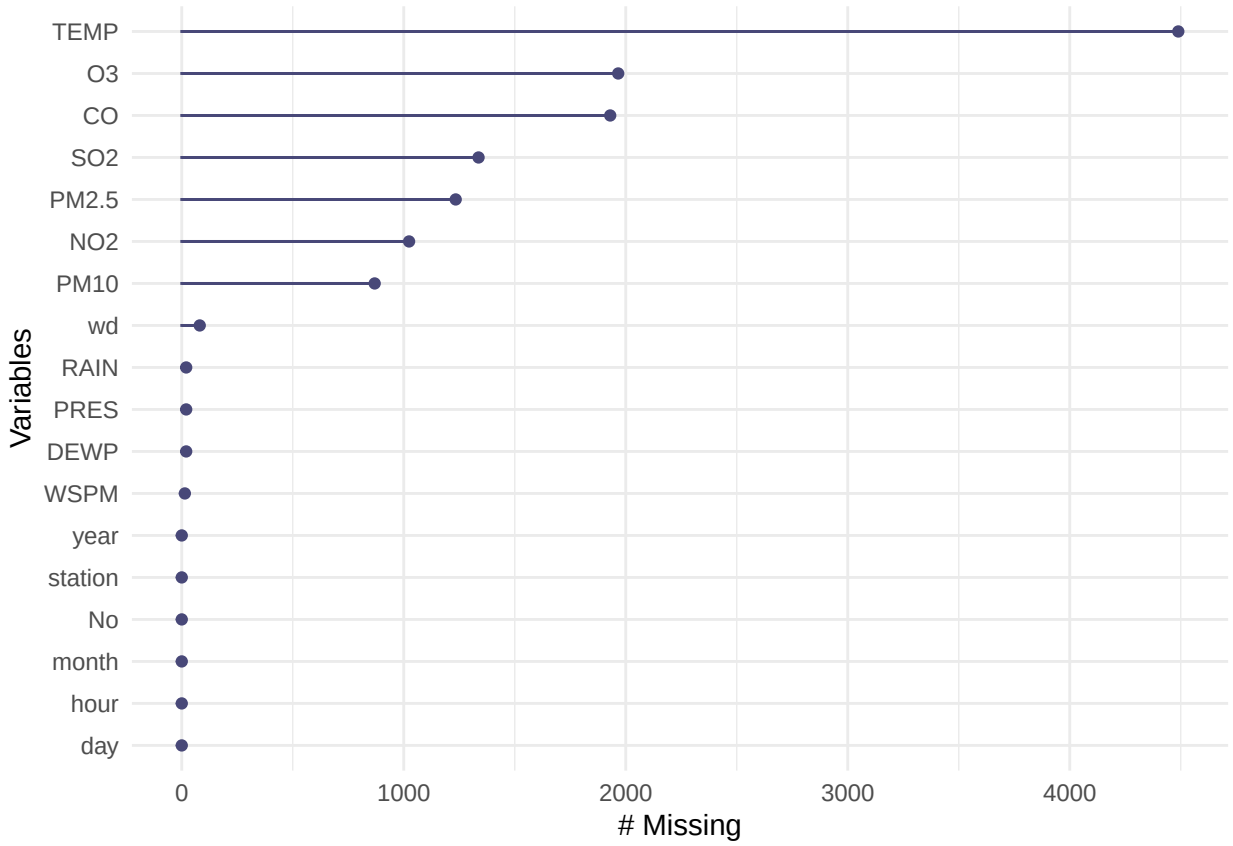
One

Apartir da análise mencionada anteriormente foi identificada que as varáveis “PM2.5”, “PM10”, “SO2”, “NO2”, “CO”, “O3”, “TEMP”, “PRES”, “DEWP”, “RAIN”, “wd”, “WSPM” possuem missing values.

Em seguida foi uma análise e foi verificado que as variáveis “TEMP”, “PRES”, “DEWP”, “RAIN” esta a faltar no mesmo dia atravez do gráfico:

Como a variável DEWP possui muitos missings values fizemos um estudo relacionando a varável DEWP e as outras variáveis climaticas para assim tentarmos identificar alguma relação entre elas. Concluimos que existe uma relação direta com a TEMP e uma relação inversa com A variável PRES.

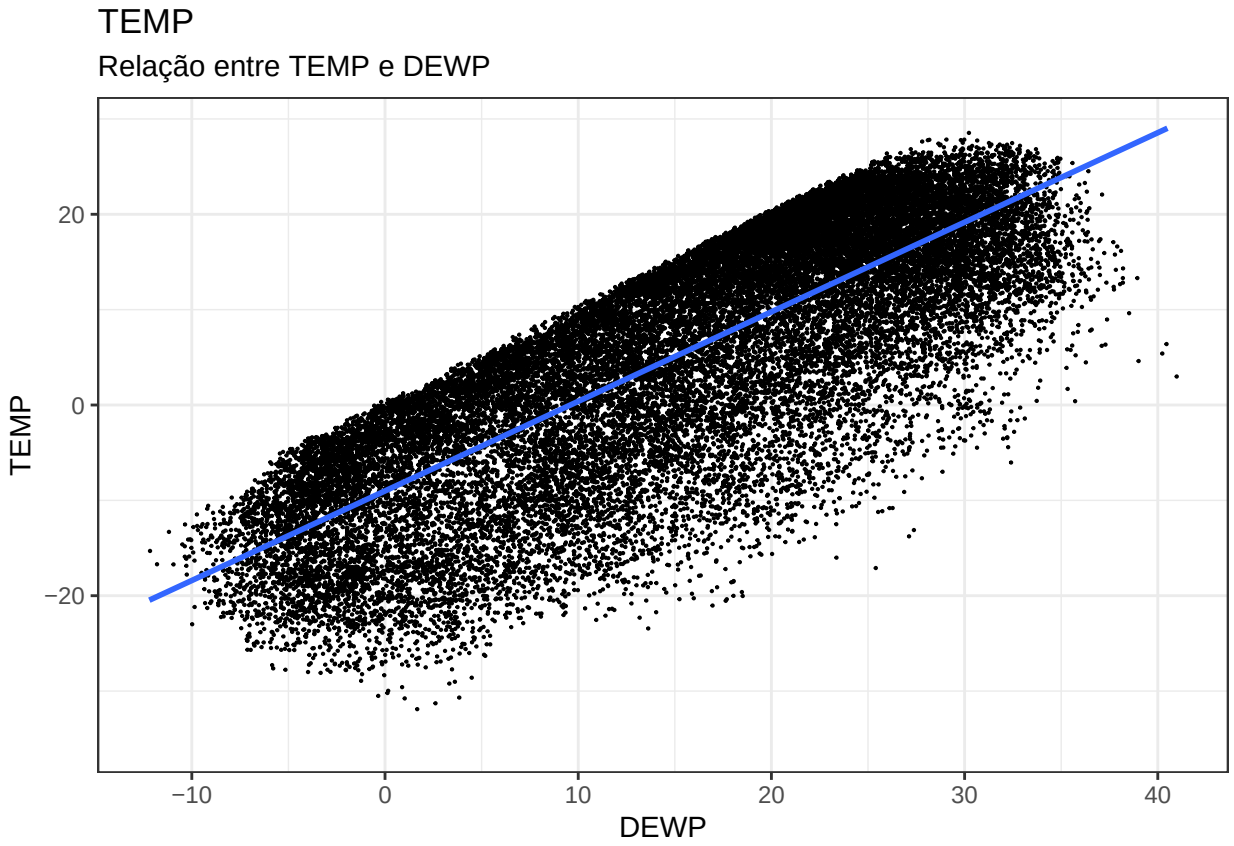
```
#check missing values
#df %>% select(RAIN, TEMP, PRES, wd, WSPM) %>% filter_any_na()
gg_miss_var(df_preproc)
```




```
atm = c("TEMP", "PRES", "DEWP", "RAIN")
# TEMP
theme_set(theme_bw())  # pre-set the bw theme.
ggplot(df_preproc, aes(TEMP, DEWP)) + geom_jitter(width = .5, size=.1) +
  labs(
    title="TEMP",
       subtitle="Relação entre TEMP e DEWP",
       x="DEWP",
       y="TEMP")+
  geom_smooth(method="lm", se=F)
```

```
## Warning: Removed 4489 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 4489 rows containing missing values (geom_point).
```

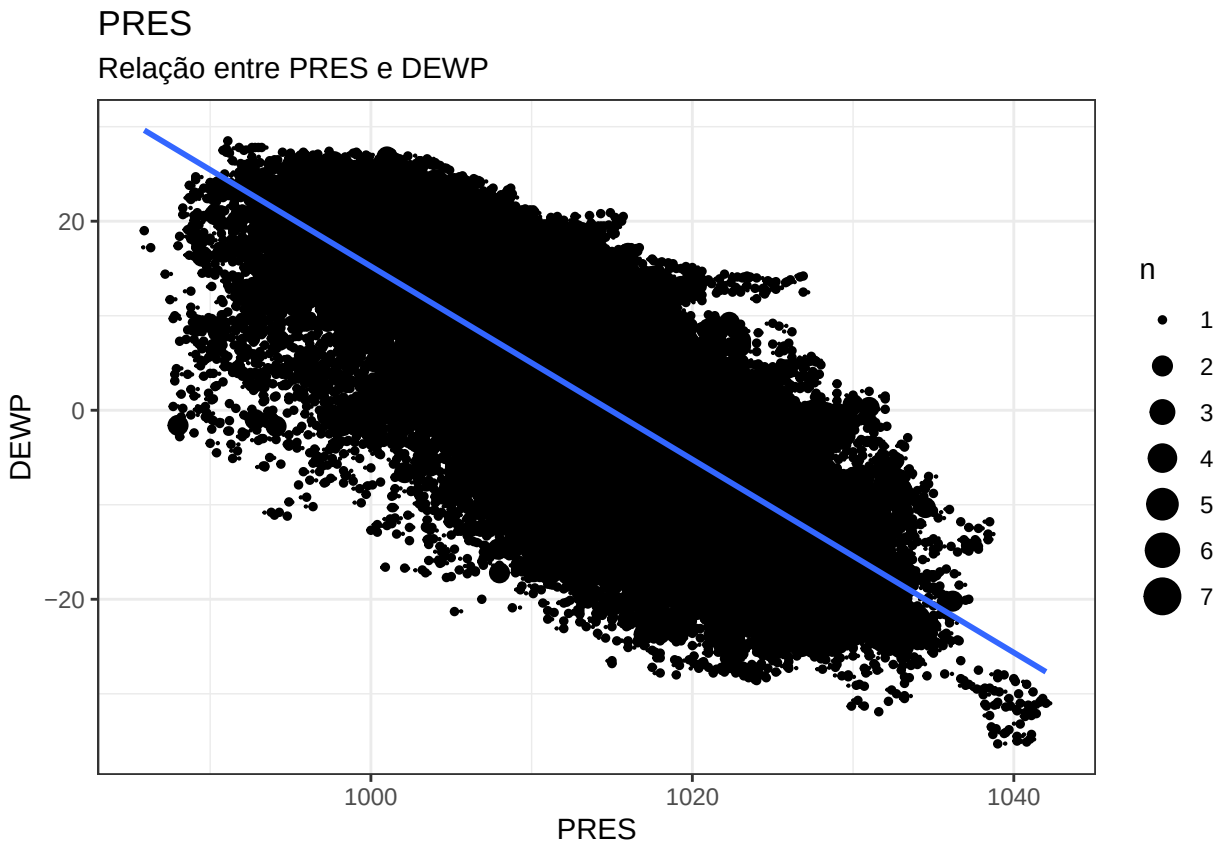



```
# PRES
ggplot(df_preproc, aes(PRES, DEWP)) + geom_jitter(width = .5, size=.1) +
  labs(
      title="PRES",
       subtitle="Relação entre PRES e DEWP",
       x="PRES",
       y="DEWP")+
  geom_count() +
  geom_smooth(method="lm", se=F)
```

```
## Warning: Removed 20 rows containing non-finite values (stat_sum).
```

```
## Warning: Removed 20 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 20 rows containing missing values (geom_point).
```

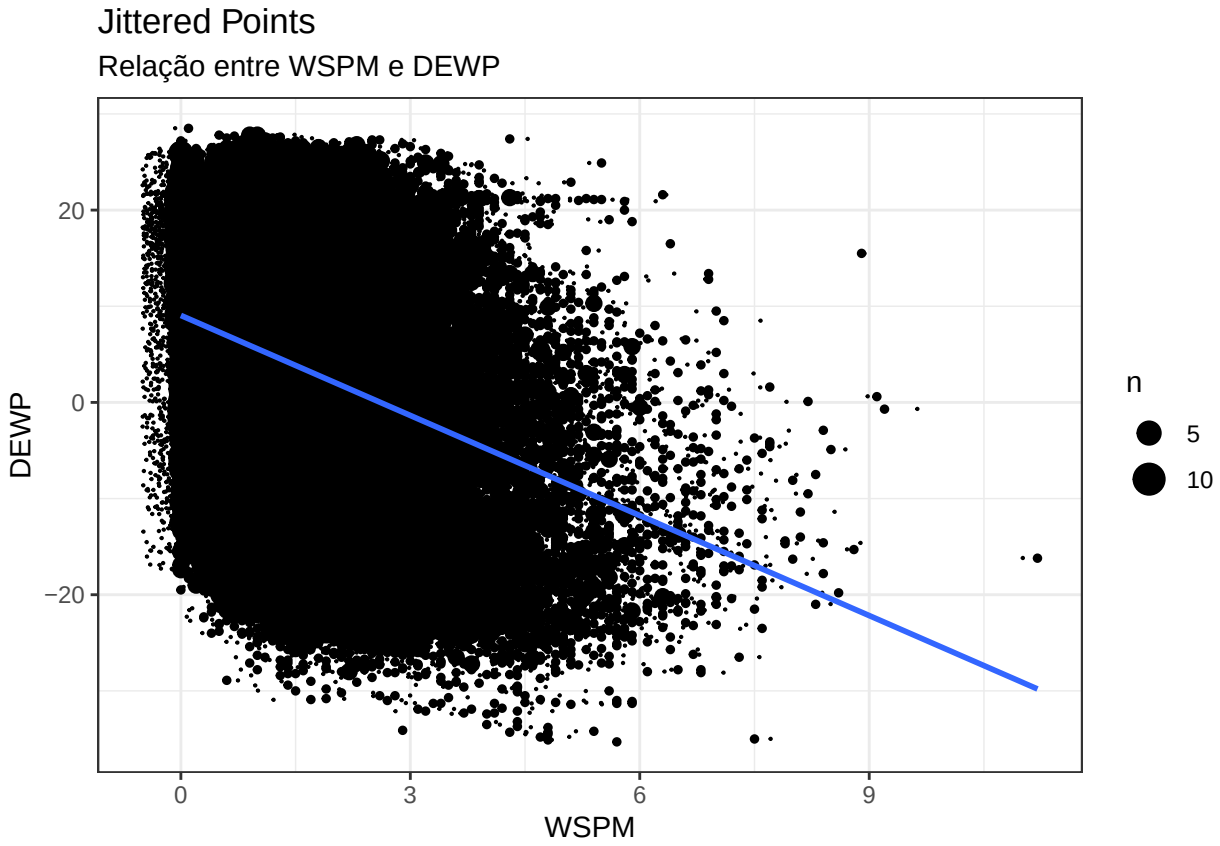



```
# DEWP
ggplot(df_preproc, aes(WSPM, DEWP)) + geom_jitter(width = .5, size=.1) +
  labs(
       title="Jittered Points",
       subtitle="Relação entre WSPM e DEWP",
       x="WSPM",
       y="DEWP")+
  geom_count() +
  geom_smooth(method="lm", se=F)
```

```
## Warning: Removed 20 rows containing non-finite values (stat_sum).
```

```
## Warning: Removed 20 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 20 rows containing missing values (geom_point).
```

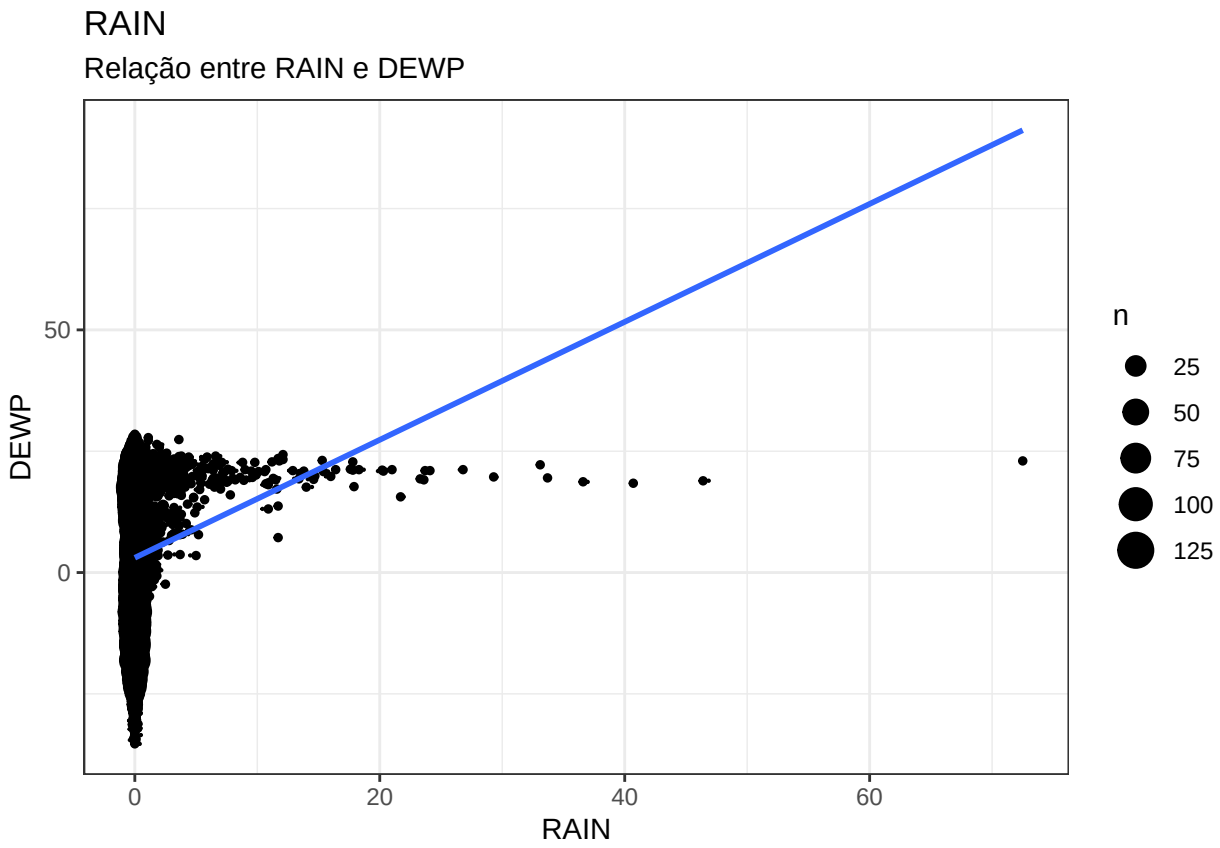



```
# RAIN
ggplot(df_preproc, aes(RAIN, DEWP)) + geom_jitter(width = .5, size=.1) +
  labs(
       title="RAIN",
      subtitle="Relação entre RAIN e DEWP",
       x="RAIN",
       y="DEWP")+
  geom_count() +
  geom_smooth(method="lm", se=F)
```

```
## Warning: Removed 20 rows containing non-finite values (stat_sum).
```

```
## Warning: Removed 20 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 20 rows containing missing values (geom_point).
```

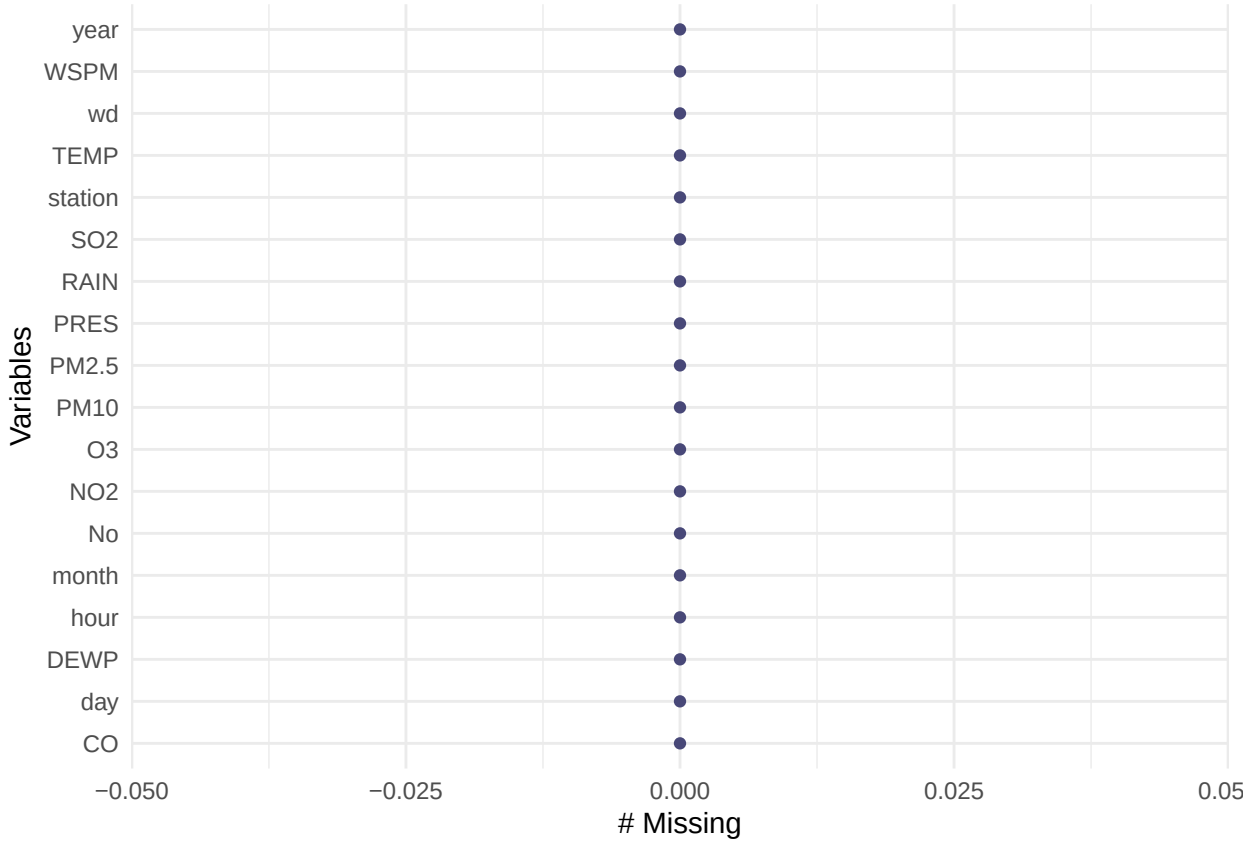



```
#for(col_name in c('TEMP', 'PRES', 'DEWP', 'RAIN')){
#  (df %>% mutate(test_col = ifelse(is_na(TEMP), 1, 0)))$test_col %>% plot(pch=".", cex=2, main=col_nam
#  abline(h = 4*mean(c(df[col_name]), na.rm=T), col="red")  # add cutoff line
#}
```

Missing Values

```
df_preproc$PM2.5 <- na.approx(df_preproc$PM2.5, rule=2)
df_preproc$PM10 <- na.approx(df_preproc$PM10, rule=2)
df_preproc$SO2 <- na.approx(df_preproc$SO2, rule=2)
df_preproc$NO2 <- na.approx(df_preproc$NO2, rule=2)
df_preproc$CO <- na.approx(df_preproc$CO, rule=2)
df_preproc$O3 <- na.approx(df_preproc$O3, rule=2)
df_preproc$TEMP <- na.approx(df_preproc$TEMP, rule=2)
df_preproc$PRES <- na.approx(df_preproc$PRES, rule=2)
df_preproc$DEWP <- na.approx(df_preproc$DEWP, rule=2)
df_preproc$WSPM <- na.approx(df_preproc$WSPM, rule=2)


#RAIN
#df_preproc$RAIN[which(is.na(df_preproc$RAIN))] <- mode(df_preproc$RAIN)
df_preproc <- df_preproc %>% mutate(RAIN = ifelse(is.na(RAIN), 0, RAIN))
#WD
df_preproc <- transform(df_preproc, wd = na.locf(wd))

#WSPM
#Precisamo verificar como tratar desta variavel(relação entre temperatura e )
```

```r
gg_miss_var(df_preproc)
```




```r
df_preproc %>% any_na()
```

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

Calcular as unidades de medida apropriadas

```r
#massa atomica de atomos utilizados
o <- 15.999
n <- 14.007
c <- 12.011
s <- 32.06

#calculando o volume a partir da pressao e da temperatura
volum_func <- function(t, p){
  return(22.41 * ((t+273.15)/273.15) * 1013/p)
}

# conversão de ug/m^3 para ppb
df_preproc <- df_preproc %>% mutate( CO  = CO  * volum_func(TEMP, PRES) / (c+o) )
df_preproc <- df_preproc %>% mutate( SO2 = SO2 * volum_func(TEMP, PRES) / (s+o*2) )
df_preproc <- df_preproc %>% mutate( NO2 = NO2 * volum_func(TEMP, PRES) / (n+o*2) )
df_preproc <- df_preproc %>% mutate( O3  = O3  * volum_func(TEMP, PRES) / (o*3) )

# conversão de ppb para ppm
```

```r
  df_preproc <- df_preproc %>% mutate( CO  = CO / 1000 )
```

Calcular a media diaria das variaveis

```r
calc_mode <- function(x){
  uniq_x <- unique(x)
  return(uniq_x[which.max(tabulate(match(x, uniq_x)))])
}

df_preproc_sample <- df_preproc %>% group_by(year, month, day) %>%
      mutate(CO =  mean(CO, na.rm = T)) %>%
      mutate(PM2.5 =  mean(PM2.5, na.rm = T)) %>%
      mutate(PM10 = mean(PM10, na.rm = T)) %>%
      mutate(SO2 = mean(SO2, na.rm = T)) %>%
      mutate(NO2 = mean(NO2, na.rm = T)) %>%
      mutate(O3 = mean(O3, na.rm = T)) %>%
      mutate(TEMP = mean(TEMP, na.rm = T)) %>%
      mutate(PRES = mean(PRES, na.rm = T)) %>%
      mutate(DEWP = mean(DEWP, na.rm = T)) %>%
      mutate(RAIN = mean(RAIN, na.rm = T)) %>%
      mutate(WSPM = mean(WSPM, na.rm = T)) %>%

      mutate(wd = calc_mode(wd)) %>%
      ungroup() %>%
      select(-No)  %>%
      select(-hour) %>%
      unique()

dim(df_preproc_sample)
```

```
## [1] 1461   16
```

calcular a estação do ano

```r
# Selecionamos uma sequencia logica para a estação do ano, criando uma escala de de 0 a 3, sendo 0 a ma

df_preproc_sample <- df_preproc_sample %>% mutate(season =
  ifelse(month == 12 & day >= 21, 3, #inverno
  ifelse(month == 1 | month == 2, 3, #inverno
  ifelse(month == 3 & day < 20, 3, #inverno

  ifelse(month == 3 & day >= 20, 1, #primavera
  ifelse(month == 4 | month == 5, 1, #primavera
  ifelse(month == 6 & day < 21, 1,  #primavera

  ifelse(month == 6 & day >= 21, 0,  #verao
  ifelse(month == 7 | month == 8, 0, #verao
  ifelse(month == 9 & day < 21, 0, #verao

  ifelse(month == 9 & day >= 21, 2, #outono
  ifelse(month == 10 | month == 11, 2, #outono
  ifelse(month == 12 & day < 21, 2, #outono
         -1))))))))))))
```

```
df_preproc_sample <- df_preproc_sample %>% mutate(season_id =
  ifelse(month == 12 & day >= 21, 'Winter',
  ifelse(month == 1 | month == 2, 'Winter',
  ifelse(month == 3 & day < 20, 'Winter',

  ifelse(month == 3 & day >= 20, 'Spring',
  ifelse(month == 4 | month == 5, 'Spring',
  ifelse(month == 6 & day < 21, 'Spring',

  ifelse(month == 6 & day >= 21, 'Summer',
  ifelse(month == 7 | month == 8, 'Summer',
  ifelse(month == 9 & day < 21, 'Summer',

  ifelse(month == 9 & day >= 21, 'Autumn',
  ifelse(month == 10 | month == 11, 'Autumn',
  ifelse(month == 12 & day < 21, 'Autumn',
         0))))))))))))
```

One Hot Encode da variavel wd

```
dmy  <- dummyVars(" ~wd", data = df_preproc_sample)
trsf <- data.frame(predict(dmy, newdata = df_preproc_sample))

df_preproc_sample <- cbind(df_preproc_sample, trsf)

df$wd %>% unique() %>% sort()
```

```
##  [1] E   ENE ESE N   NE  NNE NNW NW  S   SE  SSE SSW SW  W   WNW WSW
## Levels: E ENE ESE N NE NNE NNW NW S SE SSE SSW SW W WNW WSW
```

```
df %>% filter(wd == 'Levels:')
```

```
## # A tibble: 0 x 18
## # ... with 18 variables: No <int>, year <int>, month <int>, day <int>,
## #   hour <int>, PM2.5 <dbl>, PM10 <dbl>, SO2 <dbl>, NO2 <dbl>, CO <int>,
## #   O3 <dbl>, TEMP <dbl>, PRES <dbl>, DEWP <dbl>, RAIN <dbl>, wd <fct>,
## #   WSPM <dbl>, station <fct>
```

Calcular AQI para cada variavel

```
aqi_table <- read.csv("aqi_table.csv", header = T)
  calc_aqi <- function(c, c_h, c_l){
    ii = 1
    i_h <- aqi_table$AQI_NUM_high
    i_l <- aqi_table$AQI_NUM_low

    for(i in c(1:7)){
      if(c_h[i]>=c)
        ii = i
        break;
    }
    return( ((i_h[ii] - i_l[ii])/(c_h[ii] - c_l[ii])) * (c - c_l[ii]) + i_l[ii])
  }
```

```
  df_preproc_sample <- df_preproc_sample %>% mutate( CO_AQI    =  calc_aqi(CO,aqi_table$CO_high, aqi_tab
```

```
## Warning in if (c_h[i] >= c) ii = i: the condition has length > 1 and only
## the first element will be used
```

```
  df_preproc_sample <- df_preproc_sample %>% mutate( PM2.5_AQI =  calc_aqi(PM2.5, aqi_table$PM2.5_high,
```

```
## Warning in if (c_h[i] >= c) ii = i: the condition has length > 1 and only
## the first element will be used
```

```
  df_preproc_sample <- df_preproc_sample %>% mutate( PM10_AQI  =  calc_aqi(PM10, aqi_table$PM10_high, aq
```

```
## Warning in if (c_h[i] >= c) ii = i: the condition has length > 1 and only
## the first element will be used
```

```
  df_preproc_sample <- df_preproc_sample %>% mutate( SO2_AQI   =  calc_aqi(SO2,aqi_table$SO2_high, aqi_t
```

```
## Warning in if (c_h[i] >= c) ii = i: the condition has length > 1 and only
## the first element will be used
```

```
  df_preproc_sample <- df_preproc_sample %>% mutate( NO2_AQI   =  calc_aqi(NO2,aqi_table$NO2_high, aqi_t
```

```
## Warning in if (c_h[i] >= c) ii = i: the condition has length > 1 and only
## the first element will be used
```

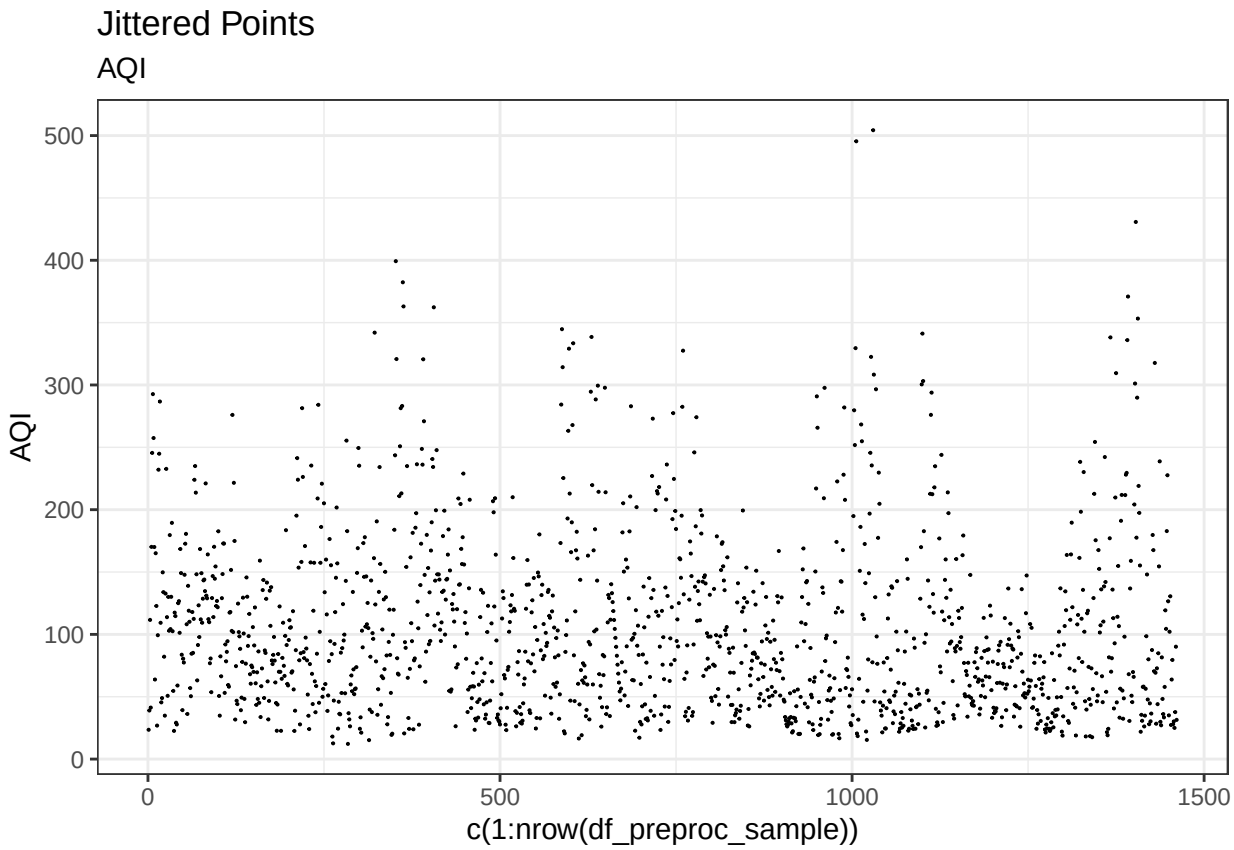
```
  df_preproc_sample <- df_preproc_sample %>% mutate( O3_AQI    =  calc_aqi(O3,aqi_table$O3_high, aqi_tab
```

```
## Warning in if (c_h[i] >= c) ii = i: the condition has length > 1 and only
## the first element will be used
```

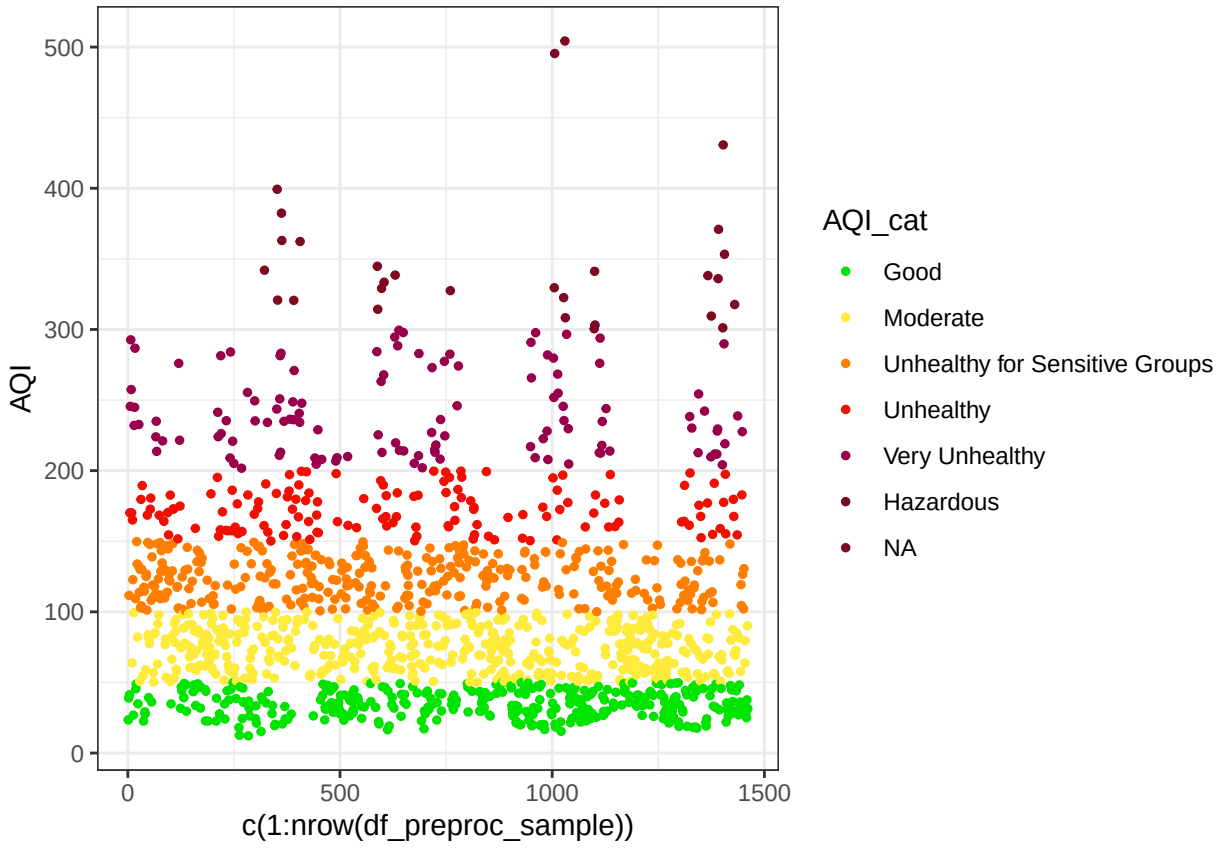
Calc a class variable AQI

```
df_preproc_sample[, "AQI"] <- apply( df_preproc_sample[c("PM10_AQI", "SO2_AQI", "NO2_AQI", "CO_AQI", "O3
```

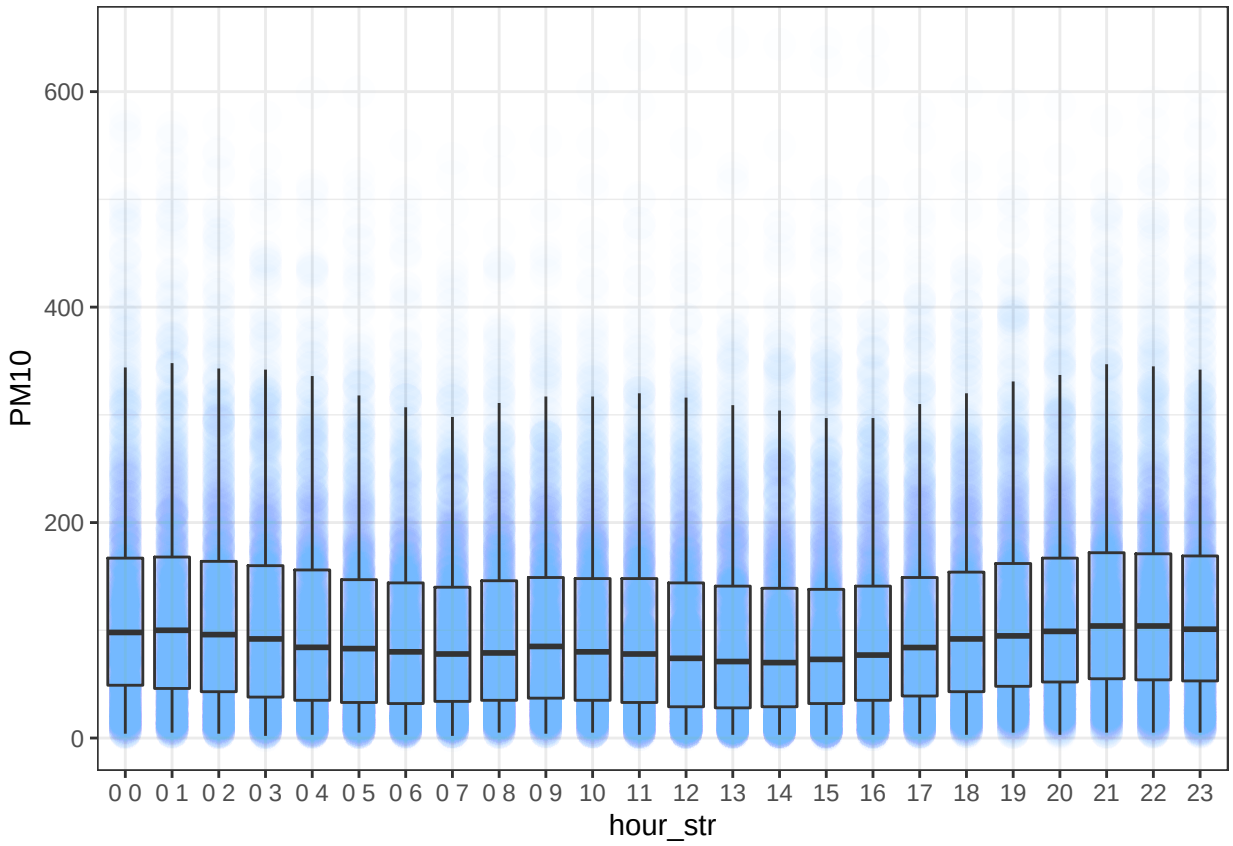
```
df_preproc_sample <- df_preproc_sample %>% mutate(AQI_cat =
                                  ifelse(AQI <= 50, 0,
                                         ifelse(AQI <= 100, 1,
                                                ifelse(AQI <= 150, 2,
                                                       ifelse(AQI <= 200, 3,
                                                              ifelse(AQI <= 300, 4, 5 ))))))


df_preproc_sample <- df_preproc_sample %>% mutate(AQI_poll =
                                  ifelse(CO_AQI  ==  AQI,  'CO',
                                  ifelse(SO2_AQI  == AQI, 'SO2',
                                  ifelse(NO2_AQI == AQI, 'NO2',
                                  ifelse(O3_AQI == AQI, 'O3',
                                  ifelse(PM10_AQI == AQI, 'PM10', 'PM2.5' ))))))



df_preproc_sample %>% ggplot(aes(c(1:nrow( df_preproc_sample )), AQI)) +
  geom_point(size=.05) +
  labs(subtitle="AQI", title="Jittered Points")
```




```
df_preproc_sample %>% ggplot(aes(c(1:nrow( df_preproc_sample )), AQI)) +
  geom_point(size=1, aes(colour = cut(AQI, c(-Inf,50, 100, 150, 200, 300, 500, Inf)))) +
  #facet_wrap(~season ) +
  scale_color_manual(name = "AQI_cat"
                     ,values = c("(-Inf,50]" = "#00e400",
                                 "(50,100]" = "#ffeb3b",
                                 "(100,150]" = "#ff7e00",
                                 "(150,200]" = "#f00f00",
                                 "(200,300]" = "#99004c",
                                 "(300,500]" = "#7e0922",
                                 "(500, Inf]" = "#7e0922")
                     ,labels = c("Good", "Moderate", "Unhealthy for Sensitive Groups", "Unhealthy", "Ve
```

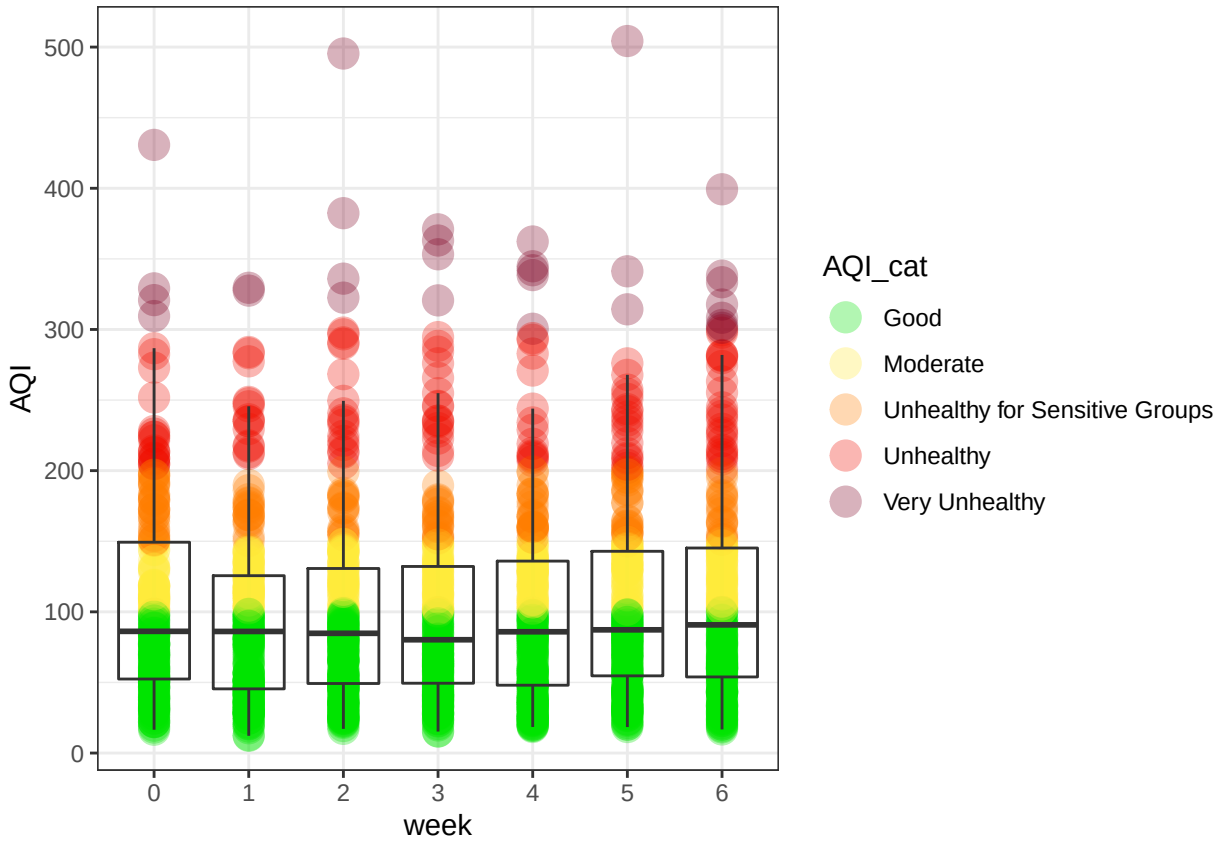



```
df_preproc %>% mutate(hour_str = ifelse(hour<10,paste("0", as.character(hour)), as.character(hour) ) ) %
  geom_point(size=5, alpha = .01, color = c('#74b9ff'))+
  geom_boxplot(alpha = 0)
```



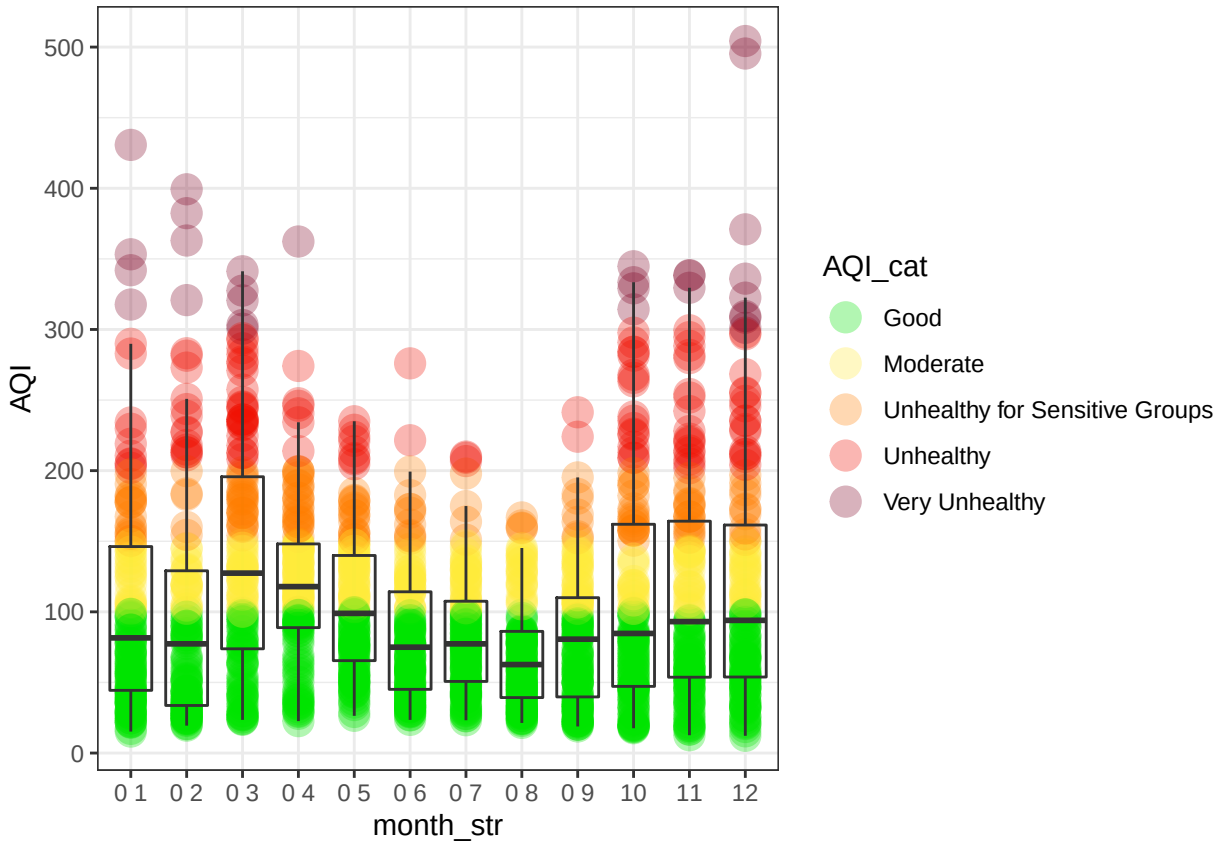
dias da semana

```
week_day = 5

df_preproc_sample <- df_preproc_sample %>% mutate(week = 0)
for(i in c(1: dim(df_preproc_sample)[1])){
  df_preproc_sample$week[i] <- week_day
  week_day <- week_day + 1
  if(week_day>6){
    week_day <-0
  }
}

df_preproc_sample %>% mutate(week = as.character(week)) %>% ggplot(aes( week, AQI ))+
  #geom_point(size=5, alpha = .1)+
geom_point(size=5, alpha = .3,aes(colour = cut(AQI_cat, c(-Inf, 1, 2, 3, 4, Inf)))) +
  #facet_wrap(~season ) +
geom_boxplot(alpha = 0) +
scale_color_manual(name = "AQI_cat"
                   ,values = c("(-Inf,1]" = "#00e400",
                               "(1,2]" = "#ffeb3b",
                               "(2,3]" = "#ff7e00",
                               "(3,4]" = "#f00f00",
                               "(4, Inf]" = "#7e0023")
                   ,labels = c("Good", "Moderate", "Unhealthy for Sensitive Groups", "Unhealthy", "Very
```

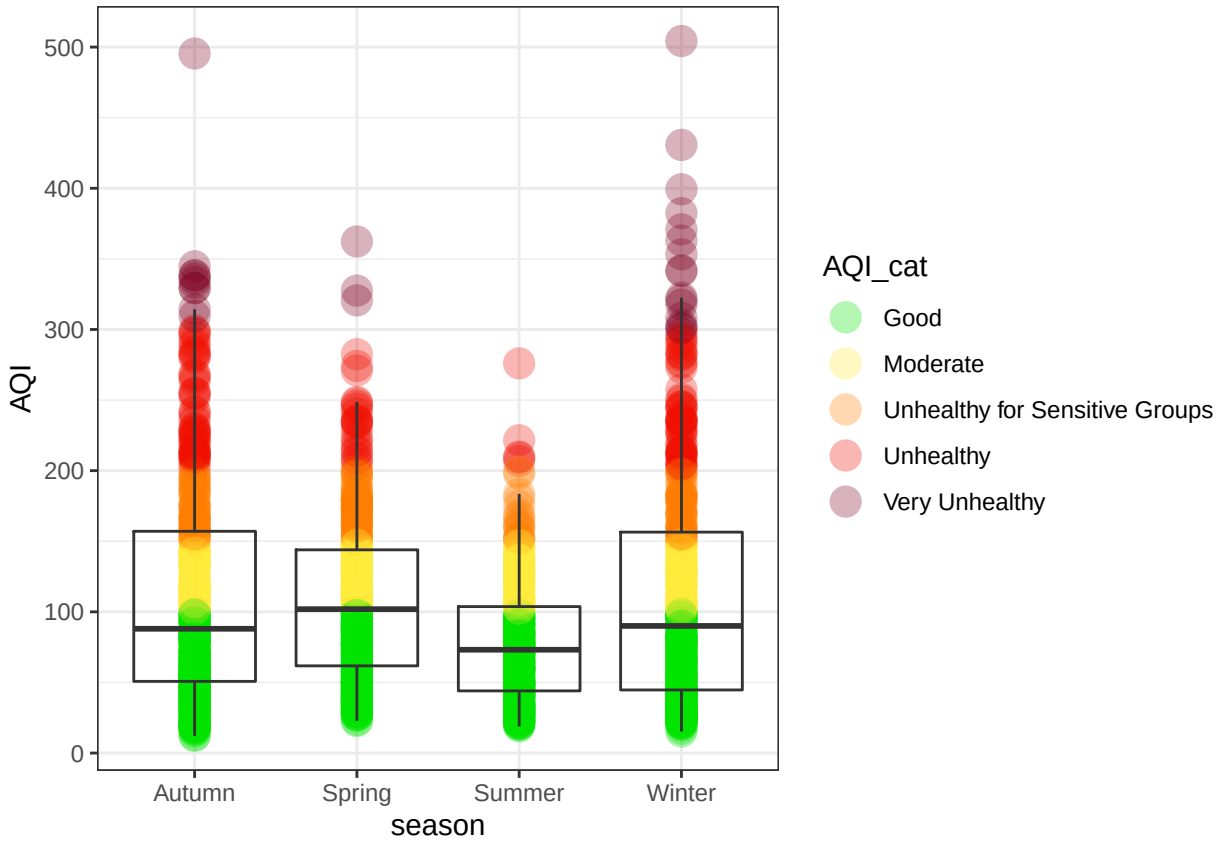


Data exploratory analysis

```
#Aumento do AQI por estação do ano
df_preproc_sample %>% mutate(month_str =
                        ifelse(month<10,paste("0", as.character(month)), as.character(month) ) ) %>% ggp
  geom_point(size=5, alpha = .3,aes(colour = cut(AQI_cat, c(-Inf, 1, 2, 3, 4, Inf)))) +

  #facet_wrap(~season ) +
  geom_boxplot(alpha = 0) +

  scale_color_manual(name = "AQI_cat"
                     ,values = c("(-Inf,1]" = "#00e400",
                                 "(1,2]" = "#ffeb3b",
                                 "(2,3]" = "#ff7e00",
                                 "(3,4]" = "#f00f00",
                                 "(4, Inf]" = "#7e0023")
                     ,labels = c("Good", "Moderate", "Unhealthy for Sensitive Groups", "Unhealthy", "Ver
```

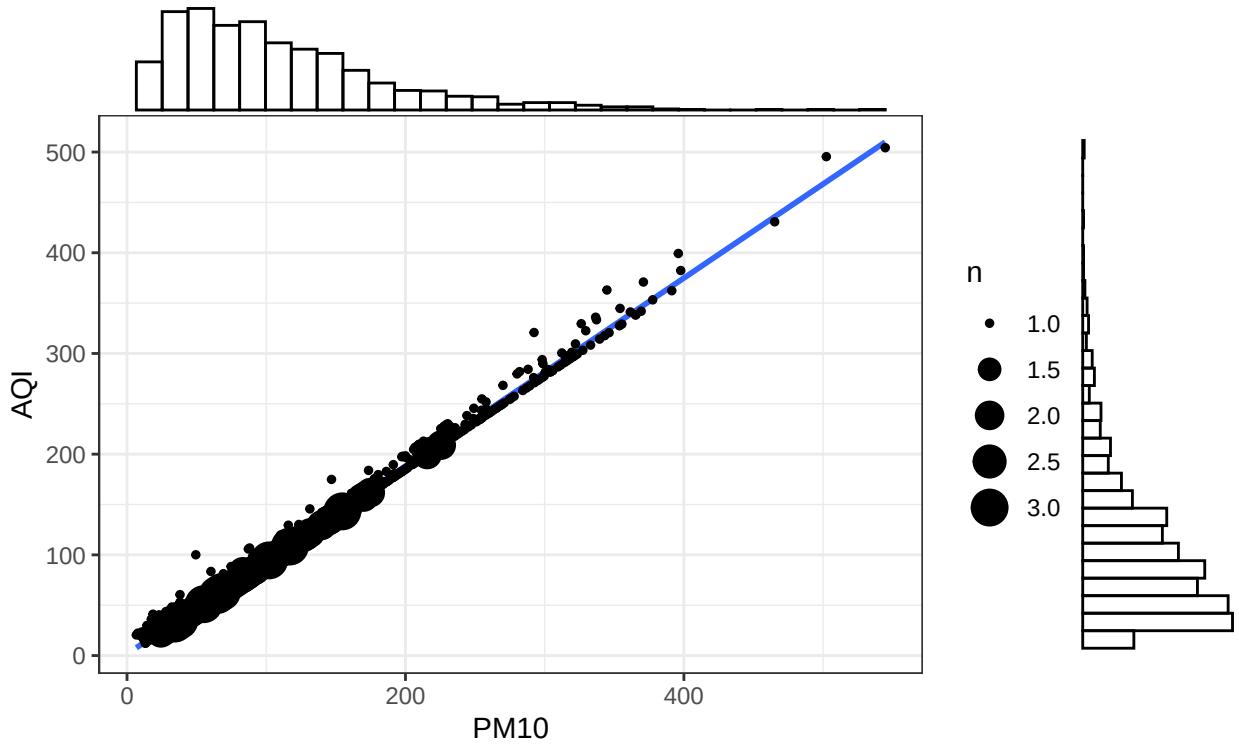



```
#Aumento do AQI por estação do ano
df_preproc_sample %>%
  mutate(season = as.character(season_id)) %>%
  ggplot(aes( season, AQI )) +
  geom_point(size=5, alpha = .3,aes(colour = cut(AQI_cat, c(-Inf, 1, 2, 3, 4, Inf)))) +
  geom_boxplot(alpha = 0) +
  scale_color_manual(name = "AQI_cat"
                     ,values = c("(-Inf,1]" = "#00e400",
                                 "(1,2]" = "#ffeb3b",
                                 "(2,3]" = "#ff7e00",
                                 "(3,4]" = "#f00f00",
                                 "(4, Inf]" = "#7e0023")
                     ,labels = c("Good", "Moderate", "Unhealthy for Sensitive Groups", "Unhealthy", "Ve
```

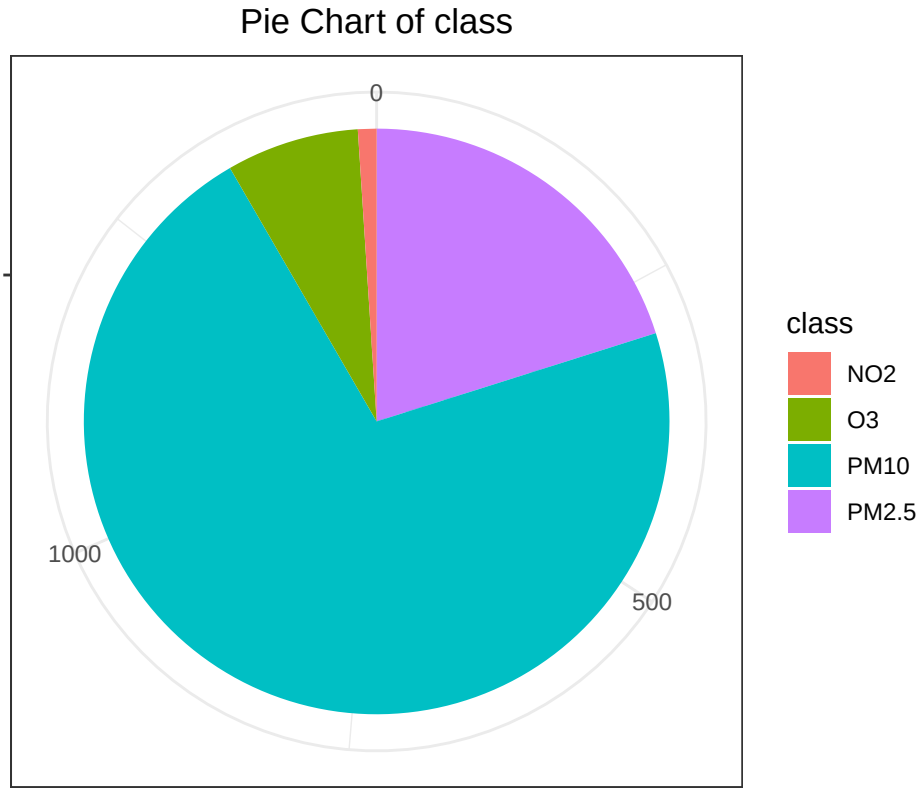


Relação AQI e PM10

```
g <- ggplot(df_preproc_sample, aes(PM10, AQI)) +
  geom_jitter(alpha=.3, size=.5) +
  labs(
       title="PM10",
       subtitle="Relação entre PM10 e AQI",
       x="PM10",
       y="AQI") +
  geom_smooth(method="lm" ) +
  geom_count()

ggMarginal(g, type = "histogram", fill="transparent")
```



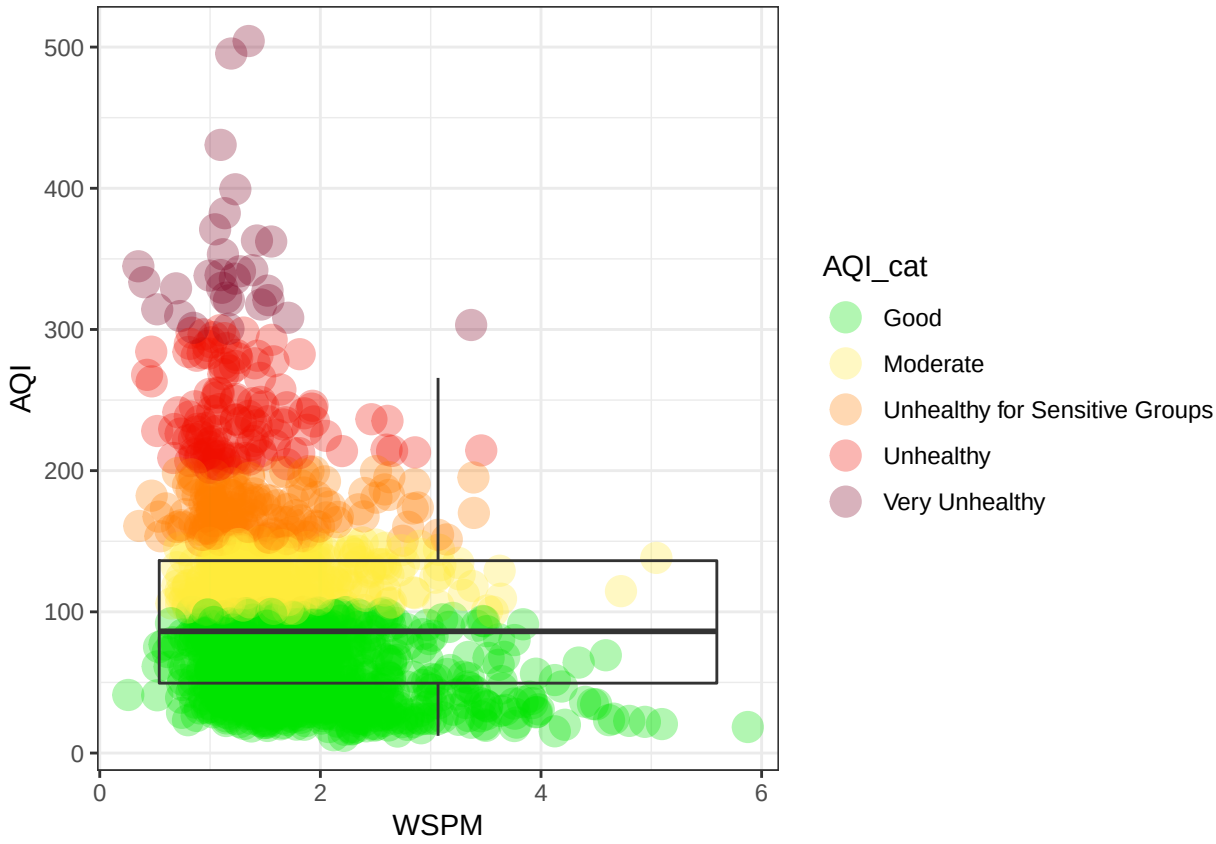

```
# Source: Categorical variable.
# mpg$class


pie <- ggplot(df_preproc_sample, aes(x = "", fill = factor(AQI_poll))) +
  geom_bar(width = 1) +
  theme(axis.line = element_blank(),
        plot.title = element_text(hjust=0.5)) +
  labs(fill="class",
       x=NULL,
       y=NULL,
       title="Pie Chart of class",
       caption="Source: mpg")

pie + coord_polar(theta = "y", start=0)
```



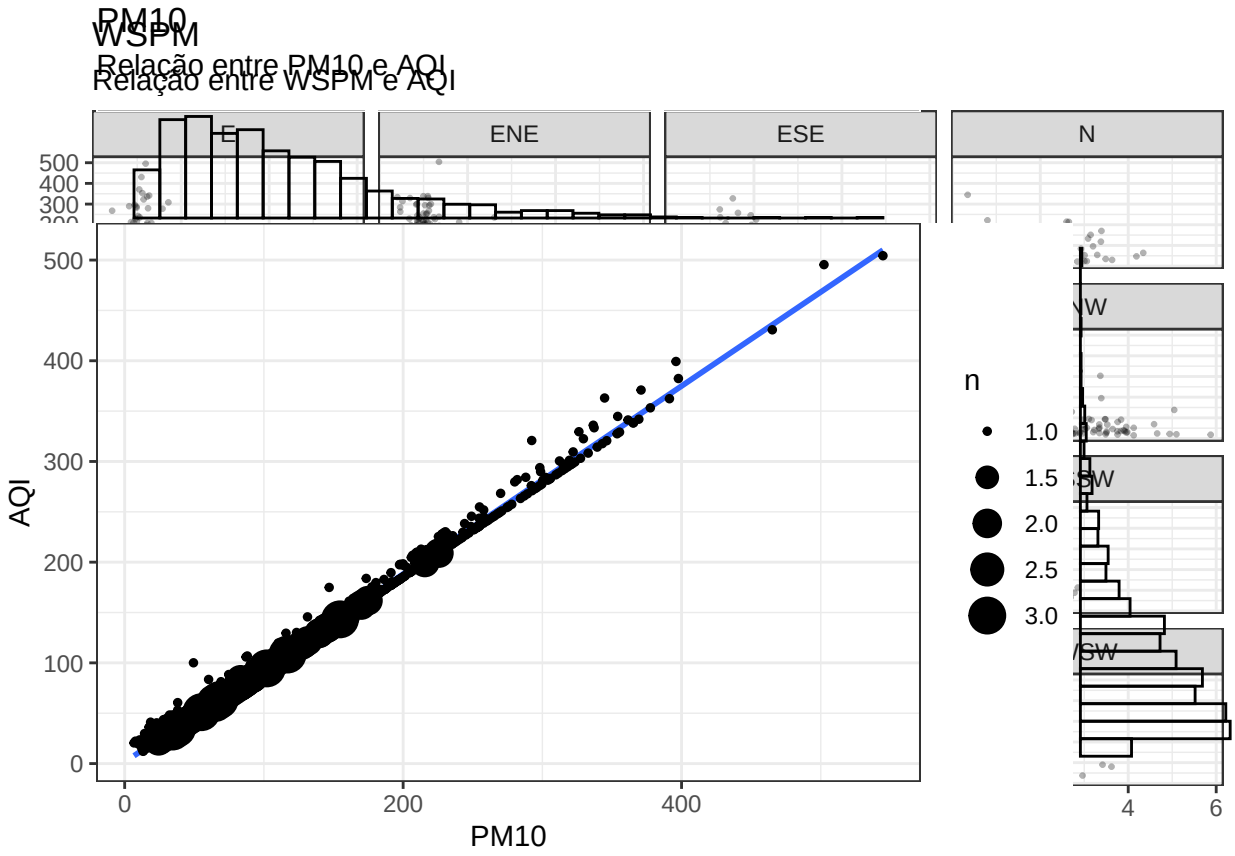
Dias de semana vs fim de semana

```
#Aumento do AQI por estação do ano
df_preproc_sample %>%
  ggplot(aes( WSPM, AQI )) +
  geom_point(size=5, alpha = .3,aes(colour = cut(AQI_cat, c(-Inf, 1, 2, 3, 4, Inf)))) +
  geom_boxplot(alpha = 0) +
  scale_color_manual(name = "AQI_cat"
                     ,values = c("(-Inf,1]" = "#00e400",
                                 "(1,2]" = "#ffeb3b",
                                 "(2,3]" = "#ff7e00",
                                 "(3,4]" = "#f00f00",
                                 "(4, Inf]" = "#7e0023")
                     ,labels = c("Good", "Moderate", "Unhealthy for Sensitive Groups", "Unhealthy", "Ve
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```

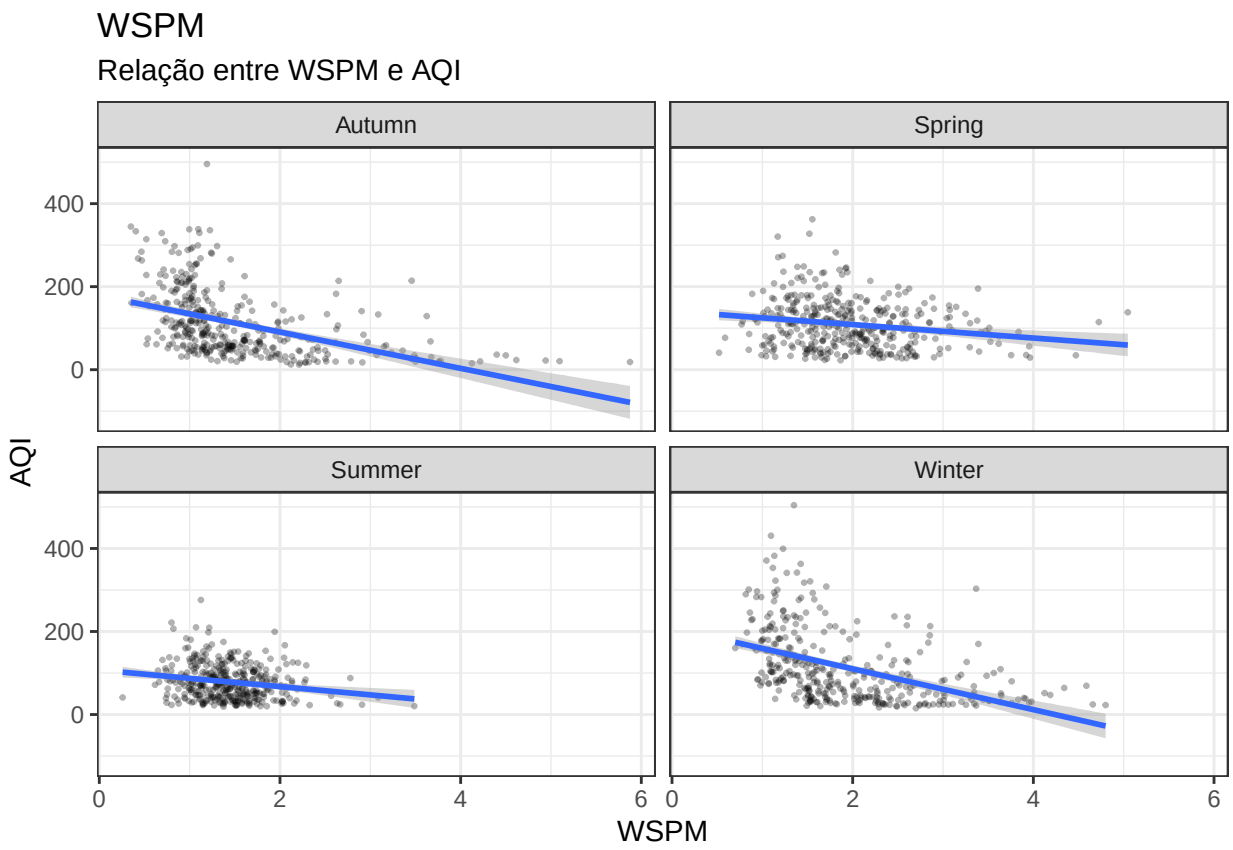



```
ggplot(df_preproc_sample, aes(WSPM, AQI)) +
  geom_jitter(alpha=.3, size=.5) +
  facet_wrap(~wd ) +
  labs(
       title="WSPM",
       subtitle="Relação entre WSPM e AQI",
       x="WSPM",
       y="AQI")

ggMarginal(g, type = "histogram", fill="transparent")
```

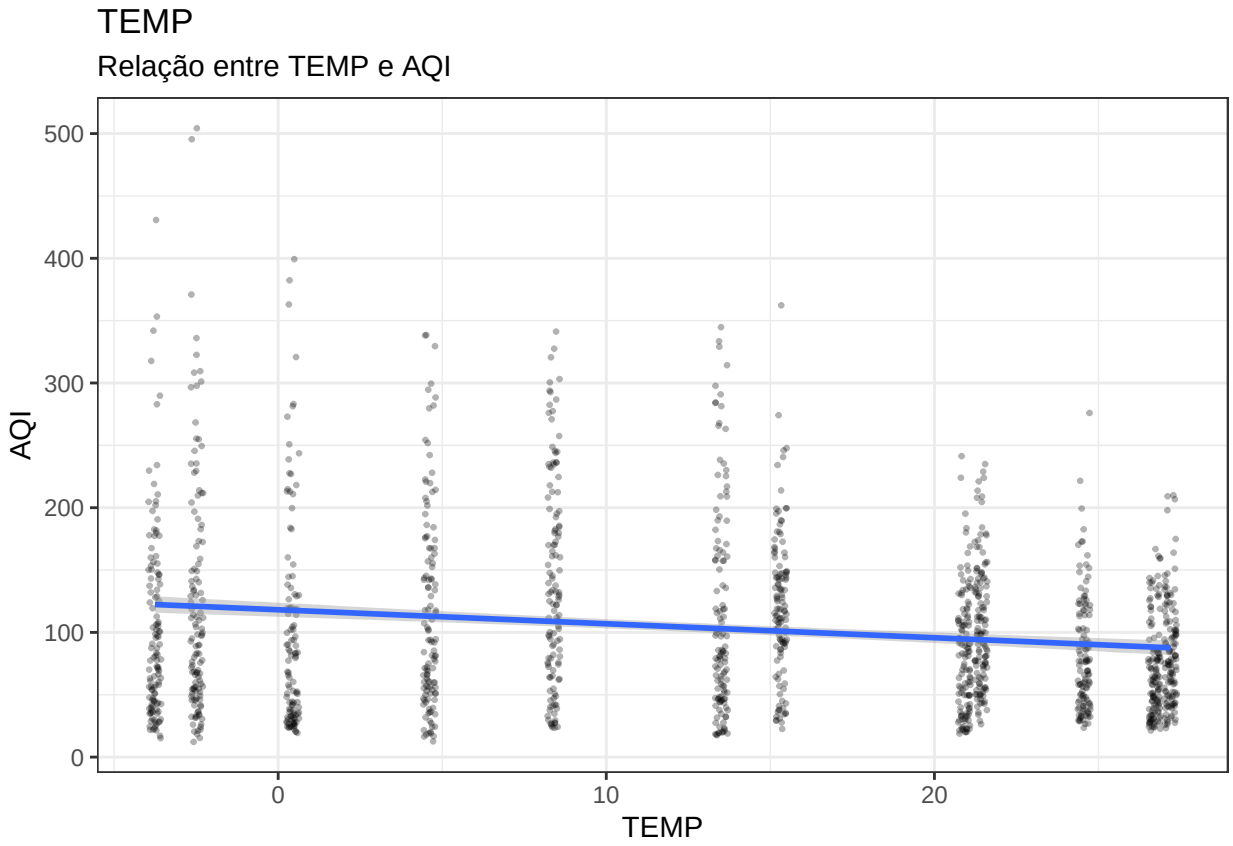


Relação AQI e TEMP

```
ggplot(df_preproc_sample, aes(WSPM, AQI)) +
  geom_jitter(alpha=.3, size=.5) +
  facet_wrap(~season_id ) +
  labs(
       title="WSPM",
       subtitle="Relação entre WSPM e AQI",
       x="WSPM",
       y="AQI") +
  geom_smooth(method="lm" )
```




```
df_preproc_sample %>%
  group_by(month) %>%
    mutate(TEMP_mean = mean(TEMP)) %>%
      ungroup() %>%
        ggplot(aes(TEMP_mean, AQI)) +
          geom_jitter(alpha=.3, size=.5) +
          labs(
               title="TEMP",
               subtitle="Relação entre TEMP e AQI",
               x="TEMP",
               y="AQI") +
          geom_smooth(method="lm" )
```



Predictive modelling

installs and imports

```
#install.packages("tensorflow")
#install.packages("keras")
#install.packages("tfdatasets")

suppressMessages(library(tensorflow))
suppressMessages(library(keras))
suppressMessages(library(tfdatasets))
```

Normalizar os dados

```
df_norm <- df_preproc_sample %>% select(c("TEMP","PRES","DEWP","RAIN","WSPM","wd.E","wd.ENE","wd.ESE","

preproc <- preProcess(df_norm, method=c("range"))
df_norm <- predict(preproc, df_norm)
#df_norm <- df_norm %>% mutate(AQI_cat = as.factor(AQI_cat))

df_norm <- df_norm %>% mutate(AQI_cat_prev1 = NA)

for(i in c(2: dim(df_norm)) ){
  df_norm$AQI_cat_prev1[i] <- df_norm$AQI_cat[i-1]
```

```
}
```

```
## Warning in 2:dim(df_norm): numerical expression has 2 elements: only the
## first used
```

```
df_norm$AQI_cat_prev1[1] <- df_norm$AQI_cat[1]

#sapply(df_preproc, class)
```

Separar o dataset in training e test

```
  #df_norm_train <- sample(1:nrow(df_norm), 0.8 * nrow(df_norm))
  #df_norm_test <- setdiff(1:nrow(df_norm), df_norm_train)

  #x_df_norm_train <- df_norm[df_norm_train, -28]
  #y_df_norm_train <- df_norm[df_norm_train, "AQI_cat"]

  #x_df_norm_test <- df_norm[df_norm_test, -28]
  #y_df_norm_test <- df_norm[df_norm_test, "AQI_cat"]
```

Criar o modelo (rede neuronal)

```
#model <- keras_model_sequential()
#model %>%
#  layer_dense(units = 28) %>%
#  layer_dense(units = 128, activation = 'relu') %>%
#  layer_dropout(0.2) %>%
#  layer_dense(units = 10, activation = 'softmax')


# COMPILE THE MODEL
#model %>% compile(
#  optimizer = 'adam',
#  loss = 'sparse_categorical_crossentropy',
#  metrics = c('accuracy')
#)
#colnames(x_df_norm_train)
# TRAIN THE MODEL
#model %>% fit(colnames(x_df_norm_train), x_df_norm_train, epochs = 5, verbose = 2)
```

Utilizando a biblioteca neuralnet

```
suppressMessages(library(neuralnet))
require(neuralnet)

## 70% of the sample size
smp_size <- floor(0.7 * nrow(df_norm))

## set the seed to make your partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(df_norm)), size = smp_size)
```

```
train <- df_norm[train_ind, ]
test <- df_norm[-train_ind, ]

nn=neuralnet(AQI~TEMP + PRES + DEWP + RAIN + WSPM + wd.E + wd.ENE + wd.ESE + wd.N + wd.NE + wd.NNE + wd
             data=train, hidden=22,act.fct = "logistic",
                linear.output = FALSE)
plot(nn)

Predict=compute(nn,test)
#Predict$net.result

#error <- data.frame(error = (abs(test$AQI - Predict$net.result)))
#error

# Calc the error
error <- data.frame(
  ERROR = (abs(test$AQI_cat - Predict$net.result)),
  real = test$AQI_cat,
  pred = Predict$net.result
  )

error <- error %>% mutate(pred = as.integer(pred) + ifelse(pred - as.integer(pred)>.5, 1, 0)) %>% mutat

print( paste("Accuracy: ", as.character(sum(error$accuracy)/dim(error)[1]*100), "%", sep = ""))
```

```
## [1] "Accuracy: 29.3849658314351%"
```

```
print(paste("Total error: ", as.character( sum(error$ERROR)/dim(error)[1]*100), sep = ""))
```

```
## [1] "Total error: 17.4565994178145"
```

```
ggplot(error, aes(c(1: dim(error)[1]), error$ERROR)) +
  geom_line(alpha=.3, size=.5) +
  labs(
       title="error",
       x="TEMP",
       y="AQI")

length(error)
```

```
## [1] 4
```

Utilizando a biblioteca tree

```
require(tree)
```

```
## Loading required package: tree
```

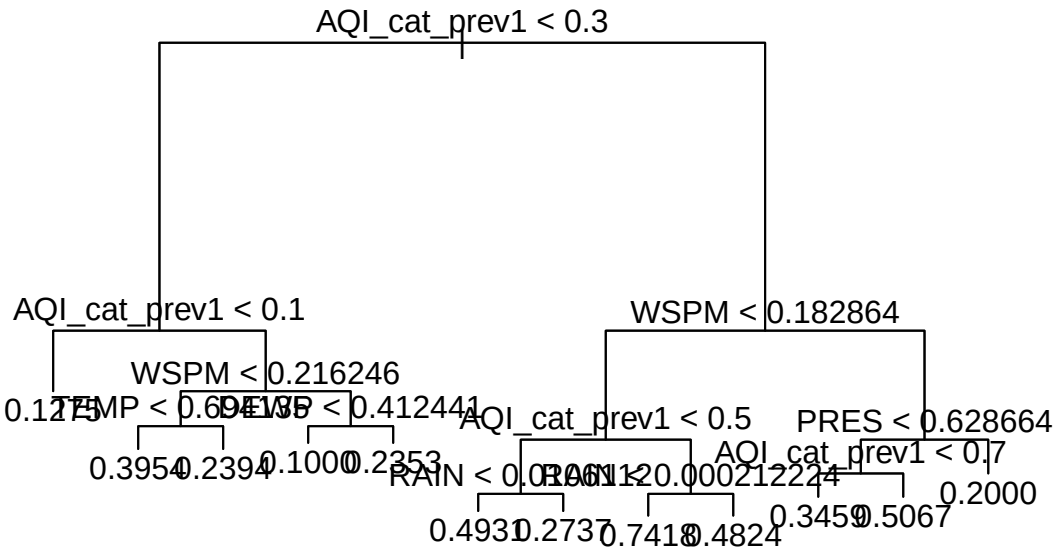
```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```
## set the seed to make your partition reproducible
set.seed(123)
smp_size <- floor(0.7 * nrow(df_norm))
train_ind <- sample(seq_len(nrow(df_norm)), size = smp_size)
```

```
train <- df_norm[train_ind, ]
test  <- df_norm[-train_ind, ]

tree_aqi <- tree(AQI_cat~ TEMP + PRES + DEWP + RAIN + WSPM + wd.E + wd.ENE + wd.ESE + wd.N + wd.NE + wd
                 data=train)
summary(tree_aqi)
```

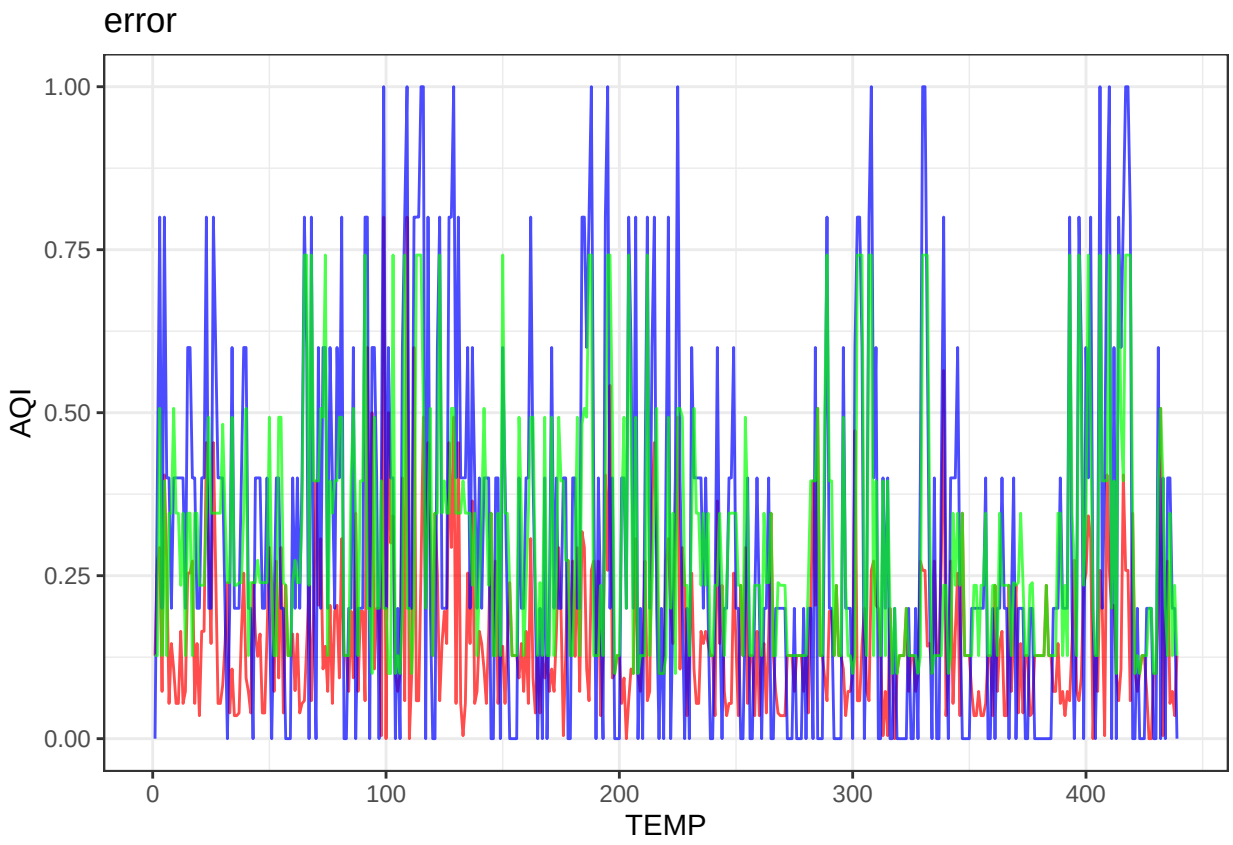
```
##
## Regression tree:
## tree(formula = AQI_cat ~ TEMP + PRES + DEWP + RAIN + WSPM + wd.E +
##     wd.ENE + wd.ESE + wd.N + wd.NE + wd.NNE + wd.NNW + wd.NW +
##     wd.S + wd.SE + wd.SSE + wd.SSW + wd.SW + wd.W + wd.WNW +
##     wd.WSW + season + AQI_cat_prev1, data = train)
## Variables actually used in tree construction:
## [1] "AQI_cat_prev1" "WSPM"          "TEMP"          "DEWP"
## [5] "RAIN"          "PRES"
## Number of terminal nodes:  12
## Residual mean deviance:  0.03599 = 36.35 / 1010
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.74180 -0.12750 -0.03529  0.00000  0.10000  0.60460
```

```
plot(tree_aqi)
text(tree_aqi, pretty = 0)
```




```
# Calc the error
error <- data.frame(
  ERROR = (abs(test$AQI_cat - predict(tree_aqi, test))),
  real = test$AQI_cat,
  pred = predict(tree_aqi, test)
  )

#plot the error
ggplot(error, aes(c(1: dim(error)[1]))) +
  geom_line(aes(y = ERROR), color = "red", alpha=.7) +
  geom_line(aes(y = real), color = "blue", alpha=.7) +
```

```
  geom_line(aes(y = pred), color = "green", alpha=.7) +
  labs(
       title="error",
       x="TEMP",
       y="AQI")
```




```
train %>% group_by(AQI_cat) %>% count()
```

```
## # A tibble: 6 x 2
## # Groups:   AQI_cat [6]
##   AQI_cat     n
##     <dbl> <int>
## 1     0     252
## 2     0.2   341
## 3     0.4   236
## 4     0.6   102
## 5     0.8    77
## 6     1      14
```

```
df_norm %>% group_by(AQI_cat) %>% count()
```

```
## # A tibble: 6 x 2
## # Groups:   AQI_cat [6]
##   AQI_cat     n
##     <dbl> <int>
## 1     0     372
## 2     0.2   474
## 3     0.4   327
```

```
## 4     0.6   142
## 5     0.8   117
## 6     1      29
```

```
# Calculate and print accuracy
error <- error %>% mutate(pred = as.integer(pred) + ifelse(pred - as.integer(pred)>.5, 1, 0)) %>% mutat

print( paste("Accuracy: ", as.character(sum(error$accuracy)/dim(error)[1]*100), "%", sep = ""))
```

```
## [1] "Accuracy: 29.8405466970387%"
```

```
print(paste("Total error: ", as.character( sum(error$ERROR)/dim(error)[1]*100), sep = ""))
```

```
## [1] "Total error: 15.9477214239685"
```