HW4

STAT 131A

For the first part of the HW, we are going to use the fitbit data again. However, I have changed some of the column names, so make sure you use the dataset attached with this homework, not the previous one.

Below is code to read in the data. The rest of the code will convert the `Date` variable into a standard date class used by R, and then create variables that give the day of the week and the month of the date.

```r
fitbit<-read.csv("fitbit.csv",header=TRUE)
fitbit$Date<-as.Date(as.character(fitbit$Date),format="%d-%m-%Y")
fitbit$Day<-factor(weekdays(fitbit$Date),levels=weekdays(x=as.Date(seq(7), origin="1950-01-01")))
fitbit$Month<-factor(months(fitbit$Date),levels=month.name)
```

Notice how I explicitly give the function `factor` the levels to expect. This allows me to define what order they will be in, so I can force them to be in a proper order for days/months (to save myself typing and possible typos, I used built in functions in R to find the names of months and weeks in the right order, but I could have just typed them out too).

Question 1: (5 points) The below code changes the dataset to convert the minutes to percentages of the total, like the last homework, in addition to other changes. Describe what lines 3-5 do. You may need to look at the `help` of the function `gsub` and `abbreviate`.

```r
totalNumberOfMinutes <- fitbit$MinutesOfSleep + fitbit$MinutesOfLightActivity + fitbit$MinutesOfModerate
absoluteMinNames<-c("MinutesOfSedentaryActivities","MinutesOfLightActivity",
    "MinutesOfModerateActivity" , "MinutesOfIntenseActivity" , "activityCalories","MinutesOfSleep",
    "MinutesOfBeingAwake" , "NumberOfAwakings","MinutesOfRest") #LINE 2
fitbit[,absoluteMinNames]<-fitbit[,absoluteMinNames]/totalNumberOfMinutes  #LINE 3
names(fitbit)<-gsub("Minutes","pctMin",names(fitbit)) #LINE 4
names(fitbit)<-abbreviate(names(fitbit),10) #LINE 5
```

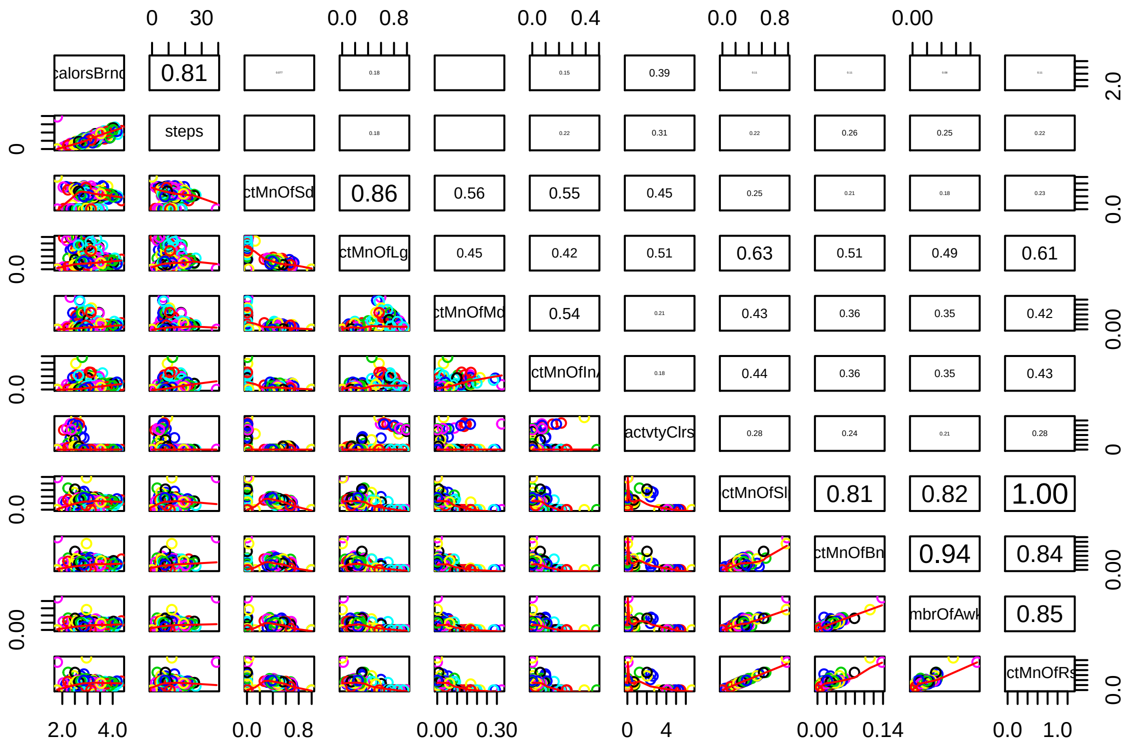
Line 3 will take the subset of fitbit at the specific columns in the list of absoluteMinNames and divide the values in these columns by totalNumberOfMinutes to get their respective percentage ratios out of the totals. Since the line 3 changes each minutes to now a percentage it becomes important to also change the name of each column respectively from "Minutes" of to "PctMin" of. Thus, Line 4 will make replacements based on the pattern character string given as "Minutes" with the character string "pctMin" of all the names of the fitbit data. For example, "MinutesofModerateActivity becomes"pctMinofModeratActivity". Line 5 will take the vector of names (names(fitbit)) and shorten each of the names to the minimum length of characters which is given as 10.

In future questions make sure you use this modified data.frame.

Question 2: (10 points) Create a pairs plot of the continous variables in the dataset, except for `distance` and `plans`, and color code by points by the day of the week. We have defined the colors for you in the following chunk with the vector `colWeek`.

```r
colWeek<-palette()[1:nlevels(fitbit$Day)]
names(colWeek)<-levels(fitbit$Day)
#Code for pairs plot here.

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r <- abs(cor(x, y,use="pairwise.complete.obs"))
    txt <- format(c(r, 0.123456789), digits = digits)[1]
    txt <- paste0(prefix, txt)
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex = cex.cor * r)
}
newpair<-fitbit[,-c(1,4,5, 15, 16)]
pairs(newpair, col=colWeek, lower.panel = panel.smooth, upper.panel = panel.cor)
```



Comment which variables appear to have a strong pairwise relationship with `calorsBrnd`.

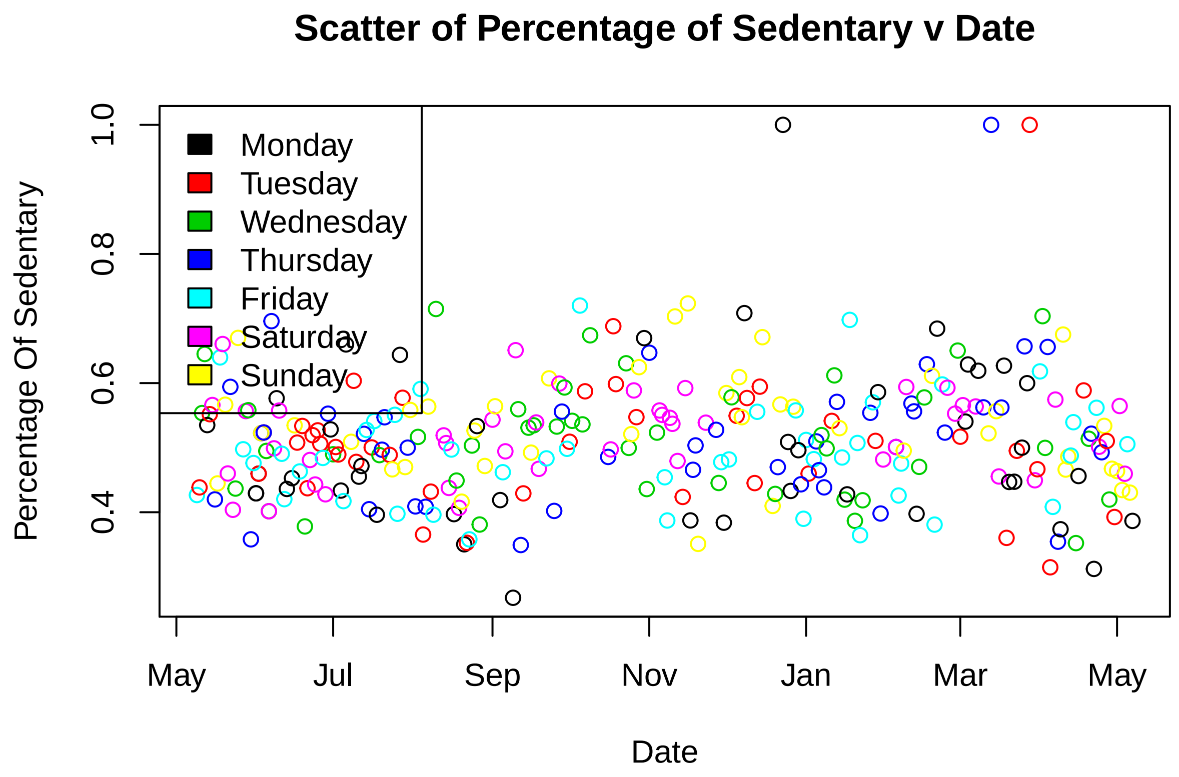
Calories burned's pair plot relationships can be viewed amongst the first row. The variable that has the most relation is steps since the matrix shows that they are correlated at a high value of 0.81.

Question 3:

(a) (5 points) Plot the percent of sedentary activity as a function of `Date`, making sure to remove the (practically) zero values as in HW3 – you should repeat your code from HW3 [Use the solutions of HW3 if you were not able to do this successfully yourself]. Color the points according the day of the week and give a legend.

```
# code to plot sedentary against date
remove=subset(fitbit, pctMnOfSdA>=0.2)
plot(remove$Date, remove$pctMnOfSdA, ylab = "Percentage Of Sedentary", xlab = "Date", col=colWeek[fitbi

legend("topleft",c("Monday","Tuesday", "Wednesday",  "Thursday " ,  "Friday",  "Saturday ",   "Sunday" 
```



(b) (10 points) Draw loess lines for for each day of the week that fit the percent of sedentary activity as a function of date. The loess lines should all be on the same plot. Note that you will need to do `as.numeric` for the `fitbit$Date` object for loess to convert it from a date to a number for fitting the loess line.

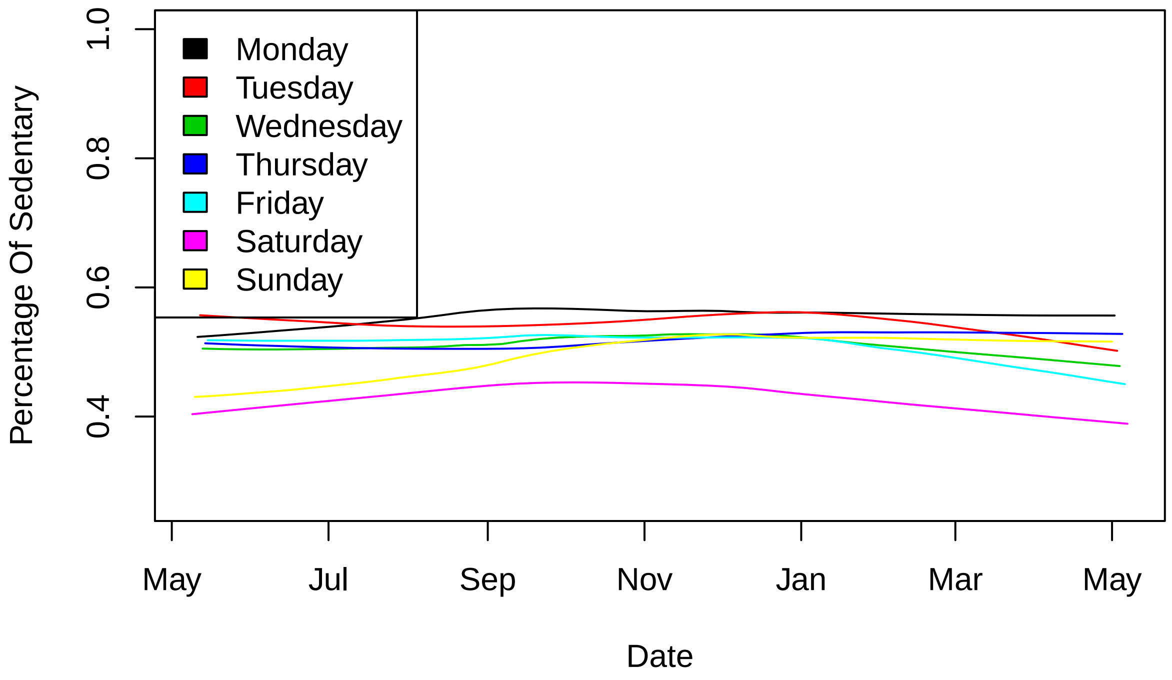
Color each line by the day of the week, as in the previous plot, and include a legend. Use either a for-loop or the `by` function to do this.

The first line of your code should set up a blank plot. Do this by reusing your code from (a) above (i.e. that plots sedentary against the data), but include the argument `type="n"` in your plot command. This instructs R to set up the axes, etc. but not actually plot the points (it could also be interesting to draw the loess lines *with* the points as well, but for this assignment draw just the lines).

```r
# code to plot loess lines
plot(remove$Date, remove$pctMnOfSdA, ylab = "Percentage Of Sedentary", xlab = "Date", col=colWeek[fitbi
fitbit$Date=as.numeric(fitbit$Date)
for (i in c("Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday")){
      fit=remove[remove$Day==i,]
      lines(loess.smooth(fit$Date,fit$pctMnOfSdA),col=colWeek[i],lwd=1)

  }

legend("topleft",c("Monday","Tuesday", "Wednesday",  "Thursday " ,  "Friday",  "Saturday ",   "Sunday" 
```



(c) (5 points) Comment on whether there are any difference due to the day of the week.

There is difference over days of the week for percentage of sedentary. In the beginning, the highest level of PCA Sedentary is for Tuesday close to 60% and Saturday is lowest at close to 40%. We also see that in the beginning of the recorded time period, May, the weekdays fall in the order of Tuesday, Monday, Friday, Thursday, Wednesday, Sunday, and Saturday for levels of percentage sedentary. At the end of one full year of recording in May of the following year, the order becomes Monday, Thursday, Tuesday, Wednesday, Friday, and Saturday. The days did not stay consistent with their levels of sedentary over time and to begin with there were already differences depending on the day of the week.

For the last questions in this HW, we are going to use multiple variable visualization tools (heatmaps and PCAs) to learn about an unknown dataset. This dataset comes from the daily measures of sensors in a urban waste water treatment plant [https://archive.ics.uci.edu/ml/datasets/Water+Treatment+Plant]. The measurements are all continuous, and are described in `VariableDescriptions.txt` file that comes with the homework. However, these are not "intuitive" variables, since they are measurements of various chemical properties of the water, so we don't expect you to understand the measurements. But we will use some of our visualization techniques to get a sense of the data, even though we don't have an intuitive sense of the meaning of the variables.

There are also variables that are related to the date in which the observations were collected (e.g. `Date`, `Month`, `Season`). For simplicity, we have removed days with NA values in any of the sensors, though this is not ideal for data analysis.

First we will provide you with some code to read in the data, and we will set up some factors and colors for the date-related variables.

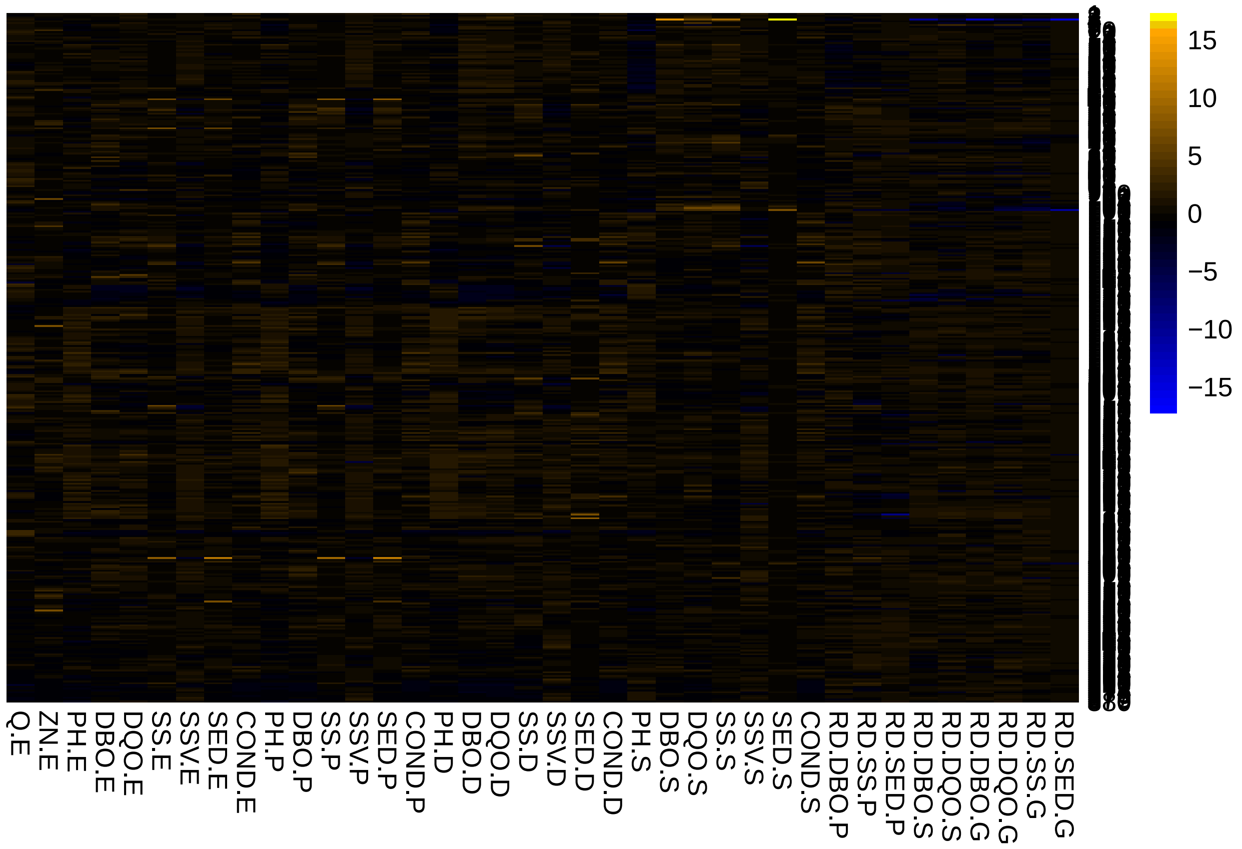
```r
water<-read.csv(file = "water-treatment-cleaned.csv", header = TRUE, stringsAsFactors = FALSE)
water$Month<-factor(water$Month,levels=month.name)
water$Day<-factor(water$Day,levels=weekdays(x=as.Date(seq(7), origin="1950-01-01")))
water$Year<-factor(water$Year,levels=c(90,91),labels=c("1990","1991"))
colDays<-palette()
names(colDays)<-levels(water$Day)
library(RColorBrewer)
colMonths<-c("coral4",brewer.pal(11, "Spectral"))
names(colMonths)<-levels(water$Month)
colYear<-c("blue","green")
names(colYear)<-levels(water$Year)
colSeason<-c("Blue","Green","Red","Brown")
names(colSeason)<-c("Winter","Spring","Summer","Fall")
# to be used for the colors of the heatmap:
seqPal2<- colorRampPalette(c("orange","black","blue"))(50)
seqPal2<-(c("yellow","gold2",seqPal2))
seqPal2<-rev(seqPal2)
```

Question 4: Heatmaps

(a) (5 points) Create a simple heatmap of this data using `pheatmap` with: the color scale given by `seqPal2` (created above in the code you were given) and with `scale="columns"` so that the variables are centered and scaled to be comparable. Make sure you install the package `pheatmap` if needed. [Hint: you need to subset your data to only the numeric variables].

You should easily see in this simple heatmap that this will not be a useful visualization without limiting the influence of outlier entries.

```r
library(pheatmap)
# add code here for basic heatmap
data=(water[,-c(1:5)])
pheatmap(data,cluster_rows = FALSE, cluster_cols = FALSE, color=seqPal2, scale="column" )
```



Next you are going to make a nice heatmap. Namely, the next questions are going to walk you through fixing the color scale like we did in class for the breast cancer data, and add information about the day, month, season, and year to the heatmap. [Hint: look at the .Rmd/.html file from the 04Chapter and the Lab that went through this]

(b) (5 points) Create a scaled version of the continuous variables of this dataset using the function `scale`, i.e. where the variables are on the same scale (centered with same st. dev). Call the new scaled dataset `waterScaled`. Save the categorical variables (`Month`,`Day`,`Year`,`Season`) into separated datatset called `waterCat`. Once you have done that, uncomment the summary commands to demonstrate that you were successful. The commented code also puts the row names onto the data (which for some reason `scale` deletes)

```
# Add code here for `waterScaled` and `waterCat`
waterScaled<-scale(water[,-c(1:5)],center=TRUE,scale=TRUE)
waterCat<-water[,c(1:5)]
# Uncomment this code
row.names(waterScaled)<-row.names(water)
summary(waterScaled[,1:5])
```

```
##       Q.E                ZN.E               PH.E               DBO.E
##  Min.   :-3.9876   Min.   :-0.9350   Min.   :-2.2200   Min.   :-2.3076
##  1st Qu.:-0.6390   1st Qu.:-0.5915   1st Qu.:-0.5294   1st Qu.:-0.6783
##  Median :-0.1868   Median :-0.3338   Median :-0.1068   Median :-0.1080
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.6137   3rd Qu.: 0.3104   3rd Qu.: 0.7385   3rd Qu.: 0.5804
##  Max.   : 3.3144   Max.   : 7.2244   Max.   : 2.8517   Max.   : 4.0469
##      DQO.E
##  Min.   :-2.53677
##  1st Qu.:-0.67855
##  Median :-0.03369
##  Mean   : 0.00000
##  3rd Qu.: 0.56238
##  Max.   : 4.55669
```

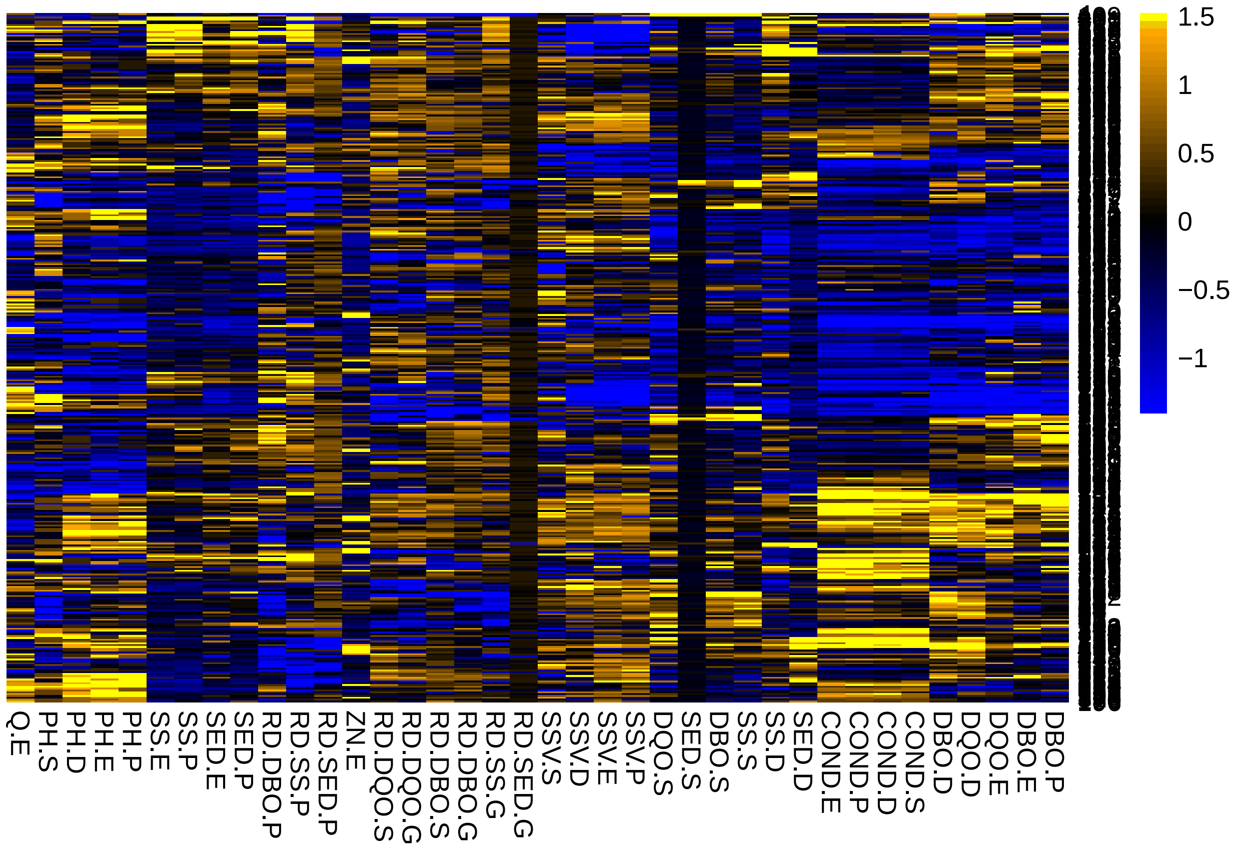
```r
summary(waterCat)
```

```
##      Date               Year         Month             Day
##  Length:380         1990:220   May     : 47   Monday   :62
##  Class :character   1991:160   January: 43   Tuesday  :72
##  Mode  :character              June    : 43   Wednesday:61
##                                April   : 41   Thursday :66
##                                October: 35   Friday   :51
##                                March   : 33   Saturday : 2
##                                (Other):138   Sunday   :66
##     Season
##  Length:380
##  Class :character
##  Mode  :character
##
##
##
##
```

(c) (10 points) Find new breakpoints for the data that span only the 0.05 and 0.95 quantiles of *all the scaled data*.

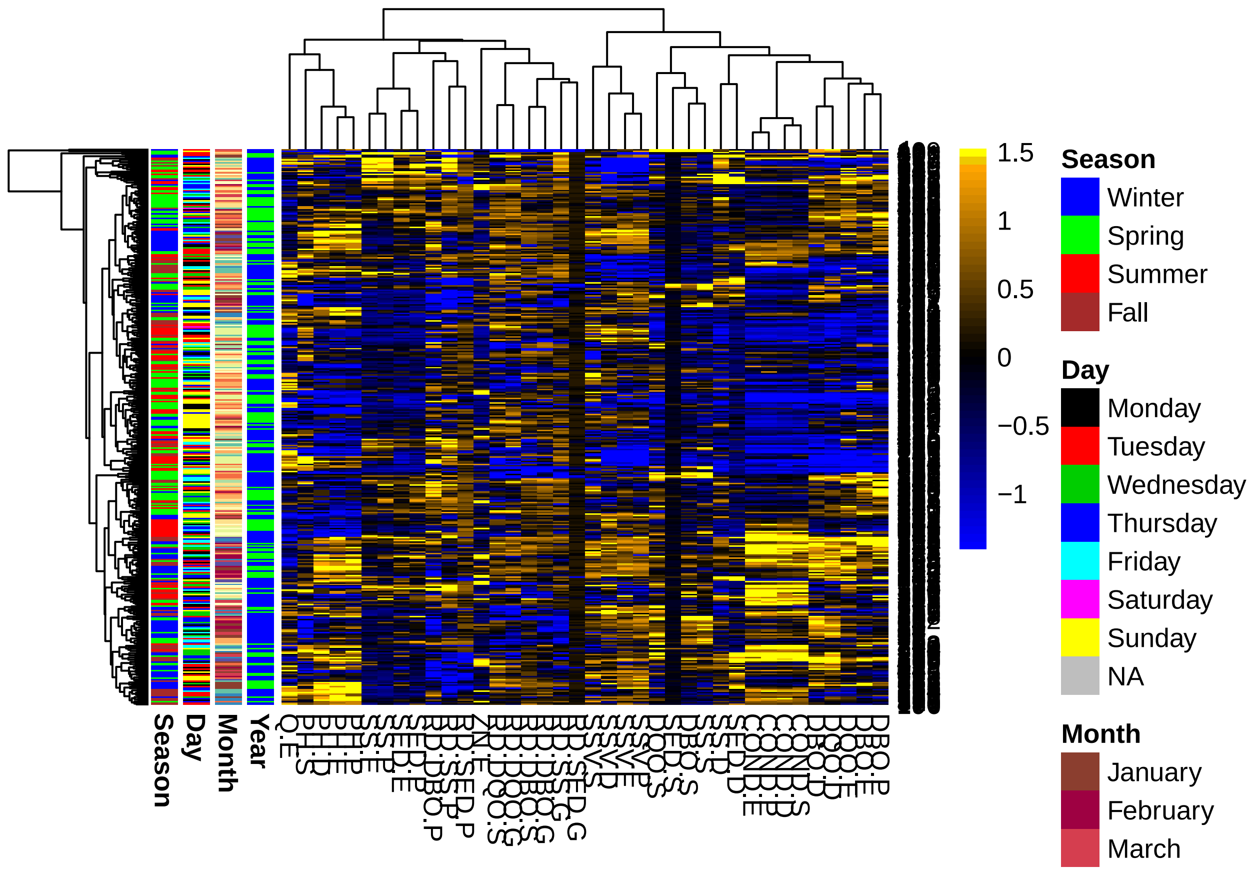
Use these breakpoints to create a better heatmap for the data. Note that the length of the breaks vector needs to be *one longer* than the length of the color vector (see `?pheatmap`).

```r
# Add code here for breakpoints
qnt<-quantile(as.numeric(data.matrix(waterScaled)),c(0.05,.95))
brks<-seq(qnt[1],qnt[2],length=53)
fullHeat<-pheatmap(waterScaled,cluster_rows = TRUE, cluster_cols = TRUE,
    treeheight_row =0, treeheight_col = 0,
    color=seqPal2,
    breaks=brks)
```



(d) (10 points) Add annotation on the samples/days corresponding to `Month`,`Day`,`Year`,`Season` using the colors created above.

```
# Add code here and fix existing code to get better heatmap
fullHeat<-pheatmap(waterScaled,cluster_rows = TRUE, cluster_cols = TRUE,
    color=seqPal2,
    breaks=brks,
    annotation_row=water[,2:5],
    annotation_colors = list("Month"=colMonths,"Day"=colDays,"Year"=colYear, "Season"=colSeason))
```



(c) (5 points) Comment on the results of your heatmap? Does it help you find patterns in the variables? In the samples/days? Describe what patterns you see. You can look at the variable descriptions if you find it helpful, but you mainly need to describe the patterns you see in the heatmap.

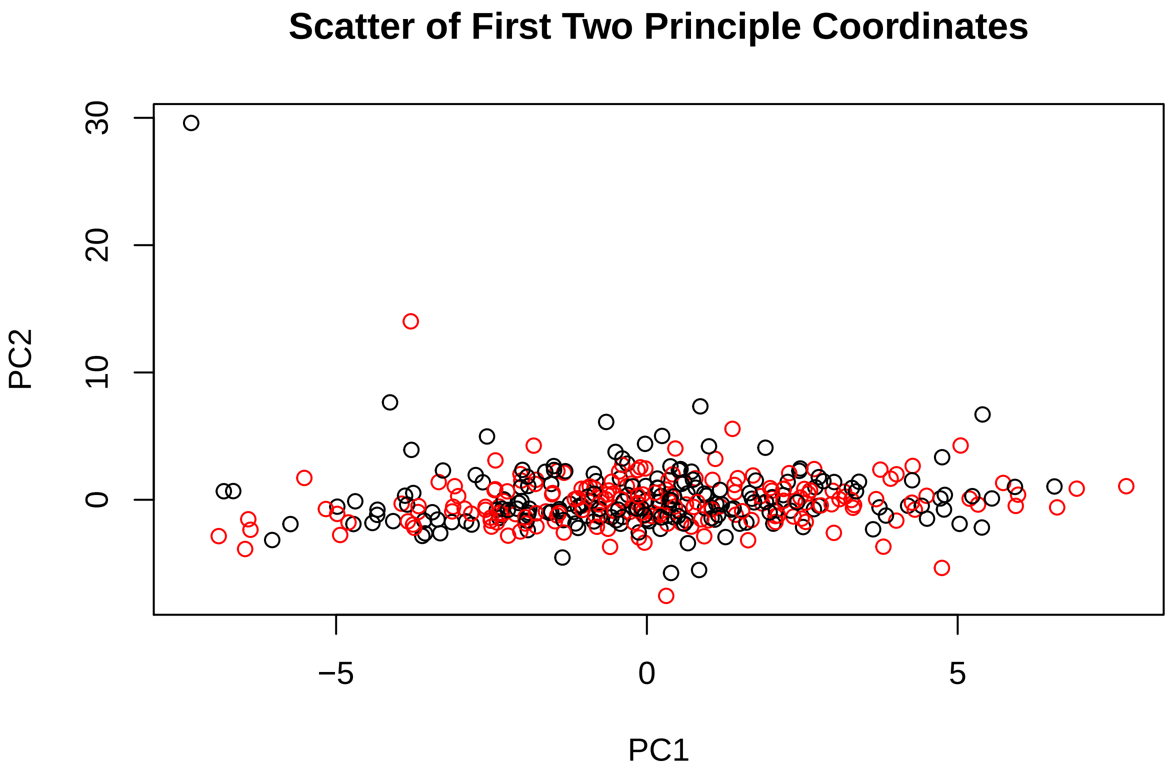
The heatmap is helpful in showing us the comparisons amongst levels of the measurements of various chemical properties of the water. Since we actually don't know what these measurements are it is easier to visualize with tree clusters and the color visualizations of the heatmap. Based on the patterns of the heatmap, blue values would be negatively correlated and yellow would show positively correlated values of the measurements of chemical properties against different variation of year, month, day and season. At first glance, the bulk of the data appears negatively correlated or not correlated represented by blue and brown. Only the bottom half of COND, PH, DBO appear to be highly correlated. Two rows of RD.SED.G and SED.S appear to be not correlated at all since they have a pattern of strictly black all along the heat map. however, the rest of the RD and SS appear to have more of the tan color meaning they are at a value of close to 1 for high correlation. Looking at the tree clusters it is easier to see that different groupings of variables are clustered if they are similar to one another. For example, different variations of DBO such as DBO.P, DBO.E, DQO.E, DQO.D, and DBO.D are in a cluster together. Therefore, families of DBO, COND, SSV, RD, SS, PH will all respectively be closer clustered to one another.

Question 5: PCA

(a) (15 points) Perform a PCA of this data and plot a scatterplot of the samples based on the first 2 principal coordinates.

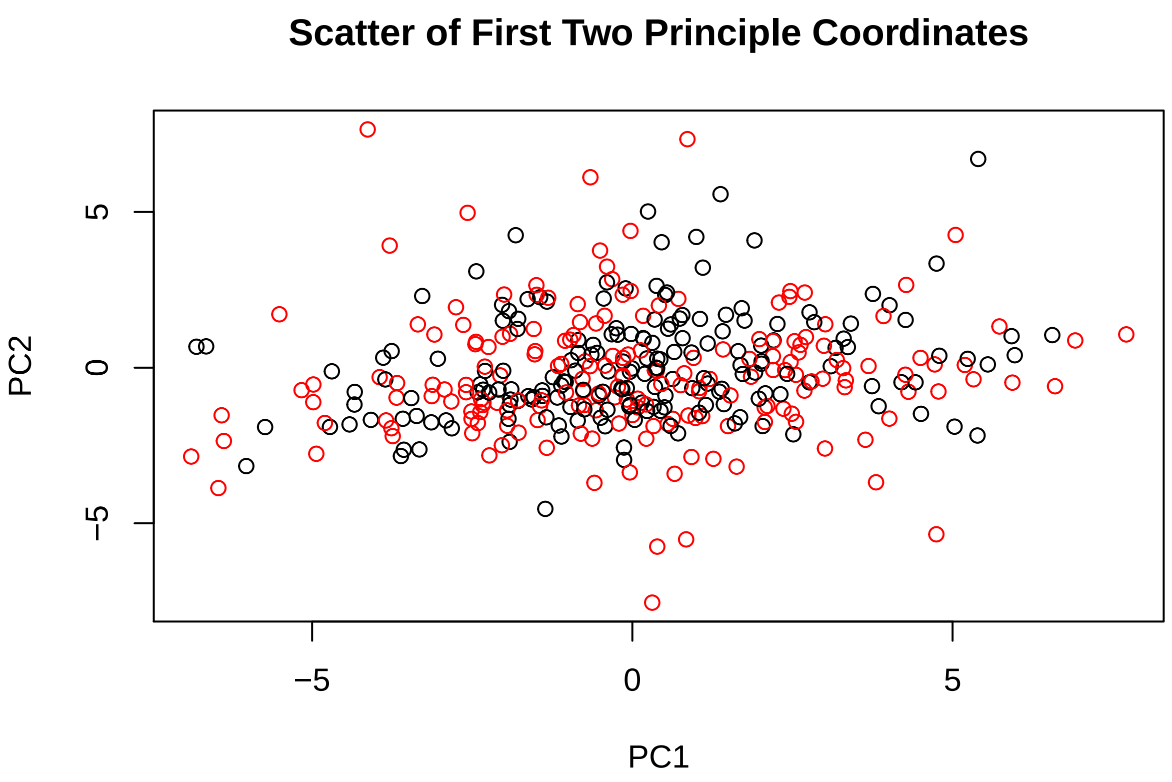
```
# add code here for pca and scatterplot

pcawater<-prcomp(waterScaled,center=TRUE,scale=TRUE)
plot(pcawater$x[,1], pcawater$x[,2],col=c("red","black"),xlab = "PC1", ylab = "PC2", main="Scatter of Fi
```



(b) (10 points) There are 1-2 observations that seem perhaps far away from the other points and might be influencing our visualization or PCA. Identify them, and remove them and redo the PCA and the scatterplot. In your R code, print out the date of the observation(s) you remove.

```
# add code here for pca and scatterplot
pcanew=pcawater$x[,1:2]
outlier=pcanew[,2]>=10
WO2=pcanew[,2][-c(4,109)]
WO1=pcanew[,1][-c(4,109)]
plot(WO1, WO2,col=c("red","black"),xlab = "PC1", ylab = "PC2", main="Scatter of First Two Principle Coor
```



```
# print the date outlying points
water[4,1]
```

```
## [1] "1990-03-13"
```

```
water[109,1]
```

```
## [1] "1990-04-29"
```

If you are interested, you can use the function `identify` to find these points (see help of `identify`). This is an interactive feature in R, but you can use it to find the points, and then once you find them, you can hard-code in your code which ones they are. This is just for interest – you do not have to find them in this way.