HW4 (95 points): DUE 11/24/2021

STAT 131A Fall 2021

For the first part of the HW, we are going to use the fitbit data again. However, some of the column names have changed, 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.

```
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 we explicitly give the function `factor` the levels to expect. This allows us to define what order they will be in, so we can force them to be in a proper order for days/months (to save on typing and possible typos, we have used built in functions in R to find the names of months and weeks in the right order, but you 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`.

```
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
```

My answer:
Line 3 will first subset fitbit by taking the columns in the 'absoluteMinNames' vector and then divide each column by the total number of minutes in activities, i.e. 'totalNumberOfMinutes', to get the percentage of each activity with respect to the whole time in activities.
Line 4 will replace the 'Minutes' string in column names of fitbit by the 'pctMin' string, since the corresponding column now represents the percentage instead of the exact minutes spent in that activity.
Line 5 will abbreviate the column names of fitbit to the minimum length of 10, such that they remain unique.

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`.

```
colWeek<-palette()[1:nlevels(fitbit$Day)]
names(colWeek)<-levels(fitbit$Day)
```

```
colWeek
```

```
##    Monday   Tuesday Wednesday  Thursday    Friday  Saturday    Sunday
##   "black" "#DF536B" "#61D04F" "#2297E6" "#28E2E5" "#CD0BBC" "#F5C710"
```

```
#Code for pairs plot here.
library(gplots)
```

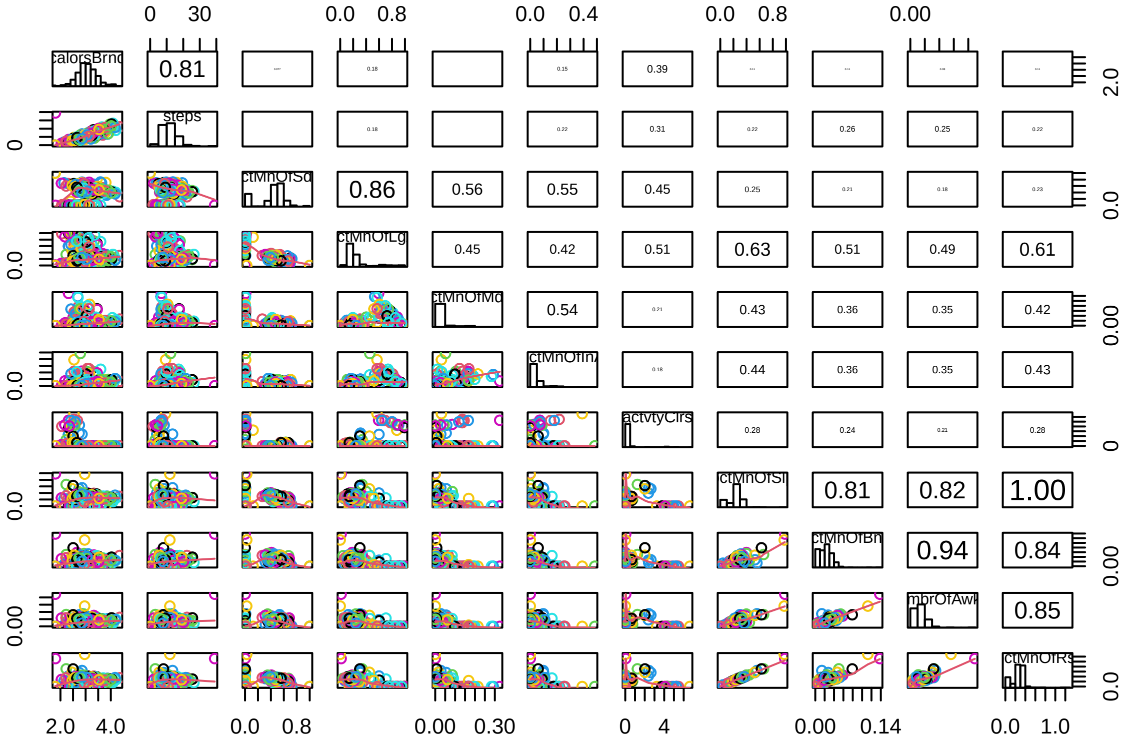
```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```
head(fitbit)
```

```
##         Date calorsBrnd  steps distance plans pctMnOfSdA pctMnOfLgA pctMnOfMdA
## 1 2016-01-01      2.992 10.460     7,92     0  0.5119581  0.2802691 0.00000000
## 2 2016-03-01      3.117 11.618     8,66    12  0.5169887  0.1445703 0.03930713
## 3 2016-04-01      2.814 11.130     8,61     8  0.6181319  0.1112637 0.00206044
## 4 2016-05-01      3.331 14.262     10,6     5  0.4637883  0.2513928 0.01462396
## 5 2015-06-01      3.354 16.836    12,51     8  0.4293040  0.1816850 0.02564103
## 6 2015-06-01      3.354 16.836    12,51     8  0.4293040  0.1816850 0.02564103
##   pctMnOfInA   actvtyClrs pctMnOfSlp pctMnOfBnA NmbrOfAwkn pctMnOfRst     Day
## 1 0.00000000 0.0011644245  0.2077728 0.01121076 0.01046338  0.2204783  Friday
## 2 0.01932045 0.0010552965  0.2798135 0.02065290 0.01732179  0.3004664 Tuesday
## 3 0.03708791 0.0008076923  0.2314560 0.03502747 0.01579670  0.2664835  Friday
## 4 0.02855153 0.0013488858  0.2416435 0.02855153 0.01392758  0.2701950  Sunday
## 5 0.05128205 0.0013641026  0.3120879 0.03956044 0.02417582  0.3626374  Monday
## 6 0.05128205 0.0013641026  0.3120879 0.03956044 0.01978022  0.3626374  Monday
##      Month
## 1 January
## 2   March
## 3   April
## 4     May
## 5    June
## 6    June
```

```
fitbit.continuous <- fitbit[,-c(1, 4, 5, 15, 16)]
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
    #from help of pairs
    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)
}
panel.hist <- function(x, ...)
{
    #from help of pairs
    usr <- par("usr"); on.exit(par(usr))
    par(usr = c(usr[1:2], 0, 1.5) )
    h <- hist(x, plot = FALSE)
    breaks <- h$breaks; nB <- length(breaks)
    y <- h$counts; y <- y/max(y)
    rect(breaks[-nB], 0, breaks[-1], y)
}
pairs(fitbit.continuous, col = colWeek, lower.panel = panel.smooth, upper.panel = panel.cor, diag.panel
```



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

My answer:
From the correlation of variables in the upper panel, we can say that only 'steps' have a strong pairwise relationship with `calorsBrnd`, which has a correlation coefficient of 0.81, while other variables all have correlation coefficients lower than 0.7.

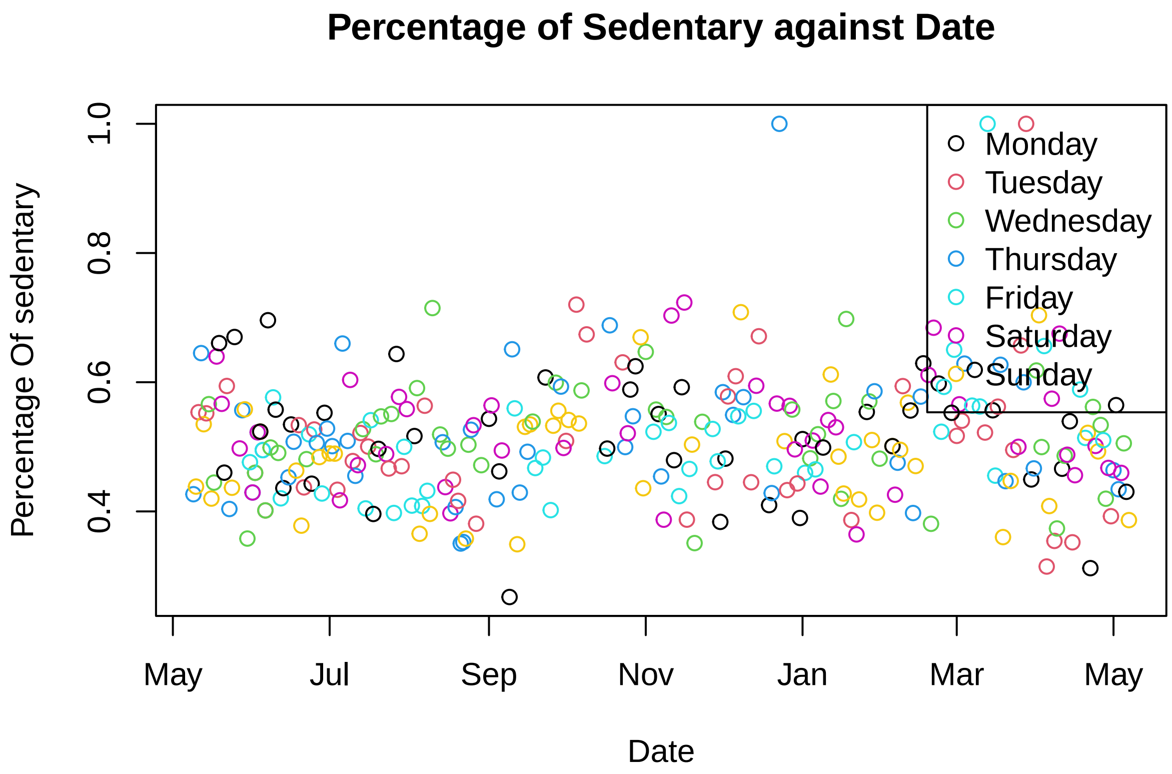
Question 3:

(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
head(fitbit)
```

```
##         Date calorsBrnd  steps distance plans pctMnOfSdA pctMnOfLgA pctMnOfMdA
## 1 2016-01-01      2.992 10.460     7,92     0  0.5119581  0.2802691 0.00000000
## 2 2016-03-01      3.117 11.618     8,66    12  0.5169887  0.1445703 0.03930713
## 3 2016-04-01      2.814 11.130     8,61     8  0.6181319  0.1112637 0.00206044
## 4 2016-05-01      3.331 14.262     10,6     5  0.4637883  0.2513928 0.01462396
## 5 2015-06-01      3.354 16.836    12,51     8  0.4293040  0.1816850 0.02564103
## 6 2015-06-01      3.354 16.836    12,51     8  0.4293040  0.1816850 0.02564103
##   pctMnOfInA   actvtyClrs pctMnOfSlp pctMnOfBnA NmbrOfAwkn pctMnOfRst     Day
## 1 0.00000000 0.0011644245  0.2077728 0.01121076 0.01046338  0.2204783  Friday
## 2 0.01932045 0.0010552965  0.2798135 0.02065290 0.01732179  0.3004664 Tuesday
## 3 0.03708791 0.0008076923  0.2314560 0.03502747 0.01579670  0.2664835  Friday
## 4 0.02855153 0.0013488858  0.2416435 0.02855153 0.01392758  0.2701950  Sunday
## 5 0.05128205 0.0013641026  0.3120879 0.03956044 0.02417582  0.3626374  Monday
## 6 0.05128205 0.0013641026  0.3120879 0.03956044 0.01978022  0.3626374  Monday
##     Month
## 1 January
## 2   March
## 3   April
## 4     May
## 5    June
## 6    June
```

```r
plot(fitbit$Date[fitbit$pctMnOfSdA >= 0.2], fitbit$pctMnOfSdA[fitbit$pctMnOfSdA >= 0.2],
      main = "Percentage of Sedentary against Date",
      ylab = "Percentage Of sedentary",
      xlab = "Date", col = colWeek)
legend("topright", legend=levels(fitbit$Day), pch = 1, col = colWeek)
```

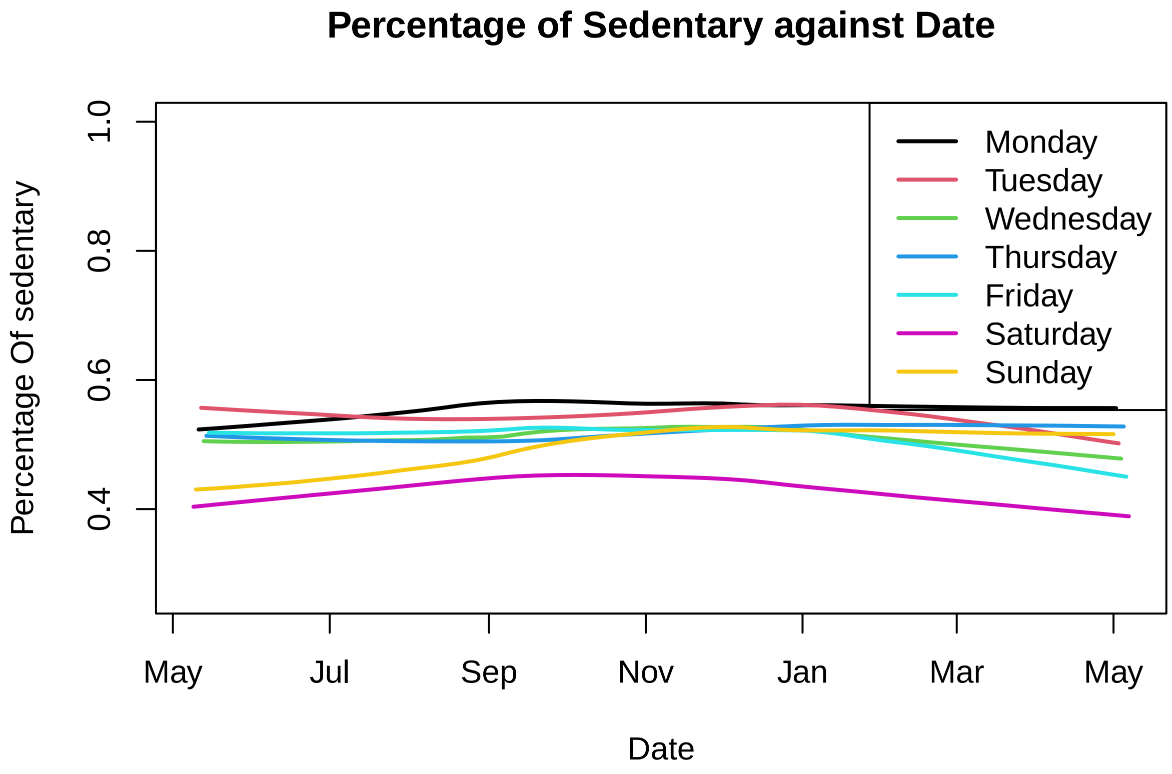


(b) (10 points) Draw loess lines for for each day of the week that fit the percent of sedentary activity as a function of date. The loess lines should all be on the same plot. Note that you will need to do as.numeric for the fitbit$Date object for loess to convert it from a date to a number for fitting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

```
# code to plot loess lines
fitbit.remove <- fitbit[which(fitbit$pctMnOfSdA >= 0.2),]
plot(fitbit.remove$Date, fitbit.remove$pctMnOfSdA,
      main = "Percentage of Sedentary against Date",
      ylab = "Percentage Of sedentary",
      xlab = "Date", col = colWeek, type="n")
legend("topright", legend=levels(fitbit$Day), lwd = 2, col=colWeek)
fitbit.remove$Date <- as.numeric(fitbit.remove$Date)
for (i in levels(fitbit.remove$Day)) {
  fitbit.remove.day = fitbit.remove[fitbit.remove$Day == i,]
  lines(loess.smooth(fitbit.remove.day$Date, fitbit.remove.day$pctMnOfSdA), col=colWeek[i], lwd = 2)
}
```



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

My answer:
The percentage of sedentary differs in different days of the week and the order of days does not stay the same through out the year.
In general, Saturday has the lowest percentage of sedentary through out the year. In May, Tuesday has the highest percentage of sedentary, following Mon, Fri, Thur and Wed. The percentage of sedentary dropped in Saturday and Sunday. Later on the percentage of sedentary in Sunday goes up. In the end of May in the next year, the order of days becomes Mon, Thurs, Sun, Tues, Wed, Fri and Sat.

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

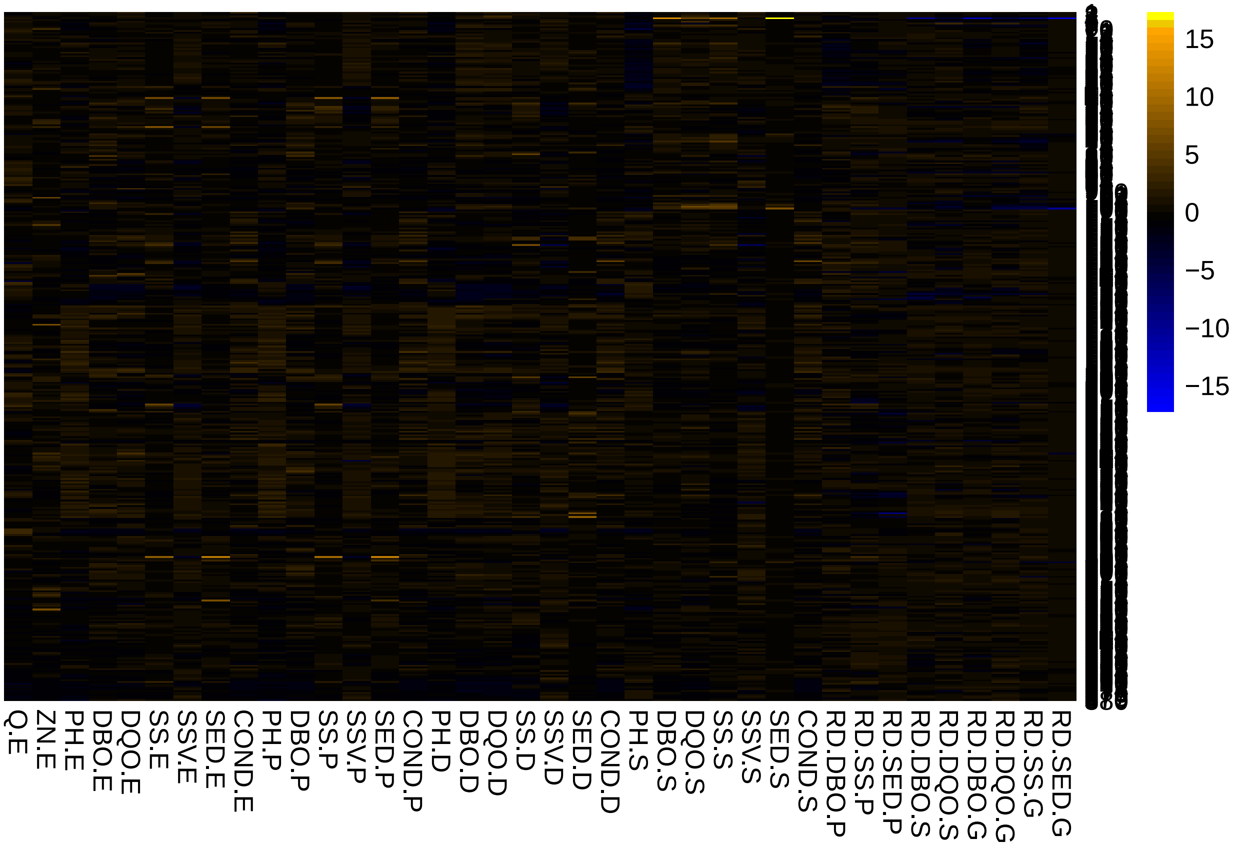
(a) (5 points) Create a simple heatmap of this data using `pheatmap` with: the color scale given by `seqPal2` (created above in the code you were given) and with `scale="column"` so that the variables are centered and scaled to be comparable. Make sure you install the package `pheatmap` if needed. [Hint: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

```
##         Date Year Month     Day Season   Q.E ZN.E PH.E DBO.E DQO.E SS.E SSV.E
## 1 1990-03-05 1990 March  Monday Winter 35023  3.5  7.9   205   588  192  65.6
## 2 1990-03-11 1990 March  Sunday Winter 29156  2.5  7.7   206   451  194  69.1
## 3 1990-03-12 1990 March  Monday Winter 39246  2.0  7.8   172   506  200  69.0
## 4 1990-03-13 1990 March Tuesday Winter 42393  0.7  7.9   189   478  230  67.0
## 5 1990-03-18 1990 March  Sunday Spring 40923  3.5  7.6   146   329  188  57.4
## 6 1990-03-19 1990 March  Monday Spring 43830  1.5  7.8   177   512  214  58.9
##   SED.E COND.E PH.P DBO.P SS.P SSV.P SED.P COND.P PH.D DBO.D DQO.D SS.D SSV.D
## 1   4.5   2430  7.8   236  268  73.1   8.5   2280  7.8   158   376   96  77.1
## 2    4.5    1249  7.7    206   220  61.8    4.0    1219  7.7    111    282  124  77.4
## 3    5.0    1865  7.8    208   248  66.1    6.5    1929  7.8    164    463  100  78.0
## 4    5.5    1410  8.1    173   192  62.5    5.0    1406  7.7    172    412  104  71.2
## 5    2.5    1300  7.6    162   132  63.6    2.0    1324  7.6    109    243   88  81.8
## 6    5.5    1605  7.7    164   256  71.9    5.5    1599  7.7    118    320   70  88.6
##   SED.D COND.D PH.S DBO.S DQO.S SS.S SSV.S SED.S COND.S RD.DBO.P RD.SS.P
## 1   0.4   2060  7.6    20   104   20  96.7  0.00   1840     33.1    64.2
## 2   0.3   1233  7.5    16   118   19  84.2  0.03   1338     46.1    43.6
## 3   0.6   1825  7.6    19   157   27  87.0  0.02   1616     21.2    59.7
## 4   0.4   1562  7.6   152   306  131  79.6  3.50   1575      0.6    45.8
## 5   0.2   1467  7.5    19    94   41  82.9  0.02   1545     32.7    33.3
## 6   0.4   1401  7.6    25   203   20  85.0  0.00   1110     28.0    72.7
##   RD.SED.P RD.DBO.S RD.DQO.S RD.DBO.G RD.DQO.G RD.SS.G RD.SED.G
## 1     95.3     87.3     72.3     90.2     82.3    89.6    100.0
## 2     92.5     85.6     58.2     92.2     73.8    90.2     99.4
## 3     90.8     88.4     66.1     89.0     69.0    86.5     99.6
## 4     92.0     11.6     25.7     19.6     36.0    43.0     36.4
## 5     90.0     82.6     61.3     87.0     71.4    78.2     99.2
## 6     92.7     78.8     36.6     85.9     60.4    90.7    100.0
```

```
water.continuous <- water[, -c(1:5)]
pheatmap(water.continuous, 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)

```r
# Add code here for `waterScaled` and `waterCat`
waterScaled<-scale(water.continuous, 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
```

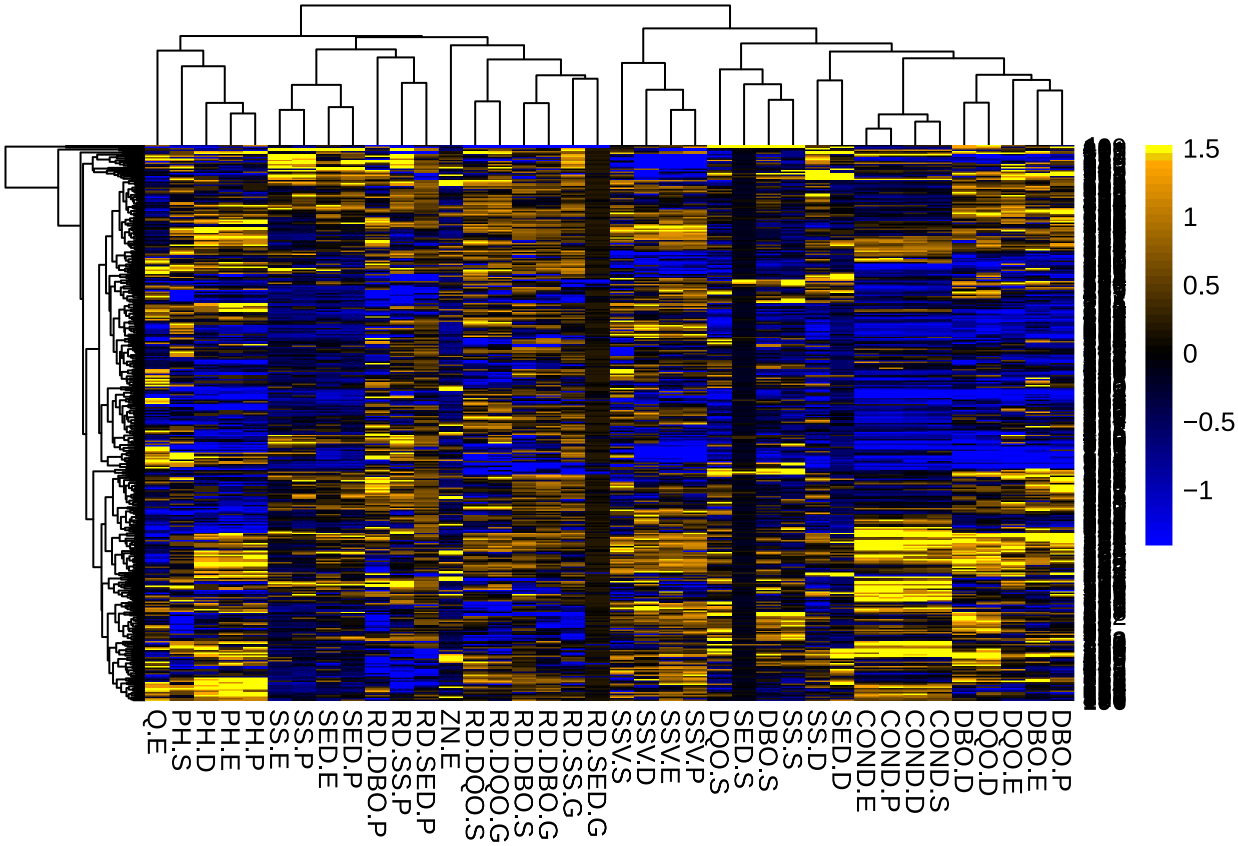
```r
summary(waterCat)
```

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

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

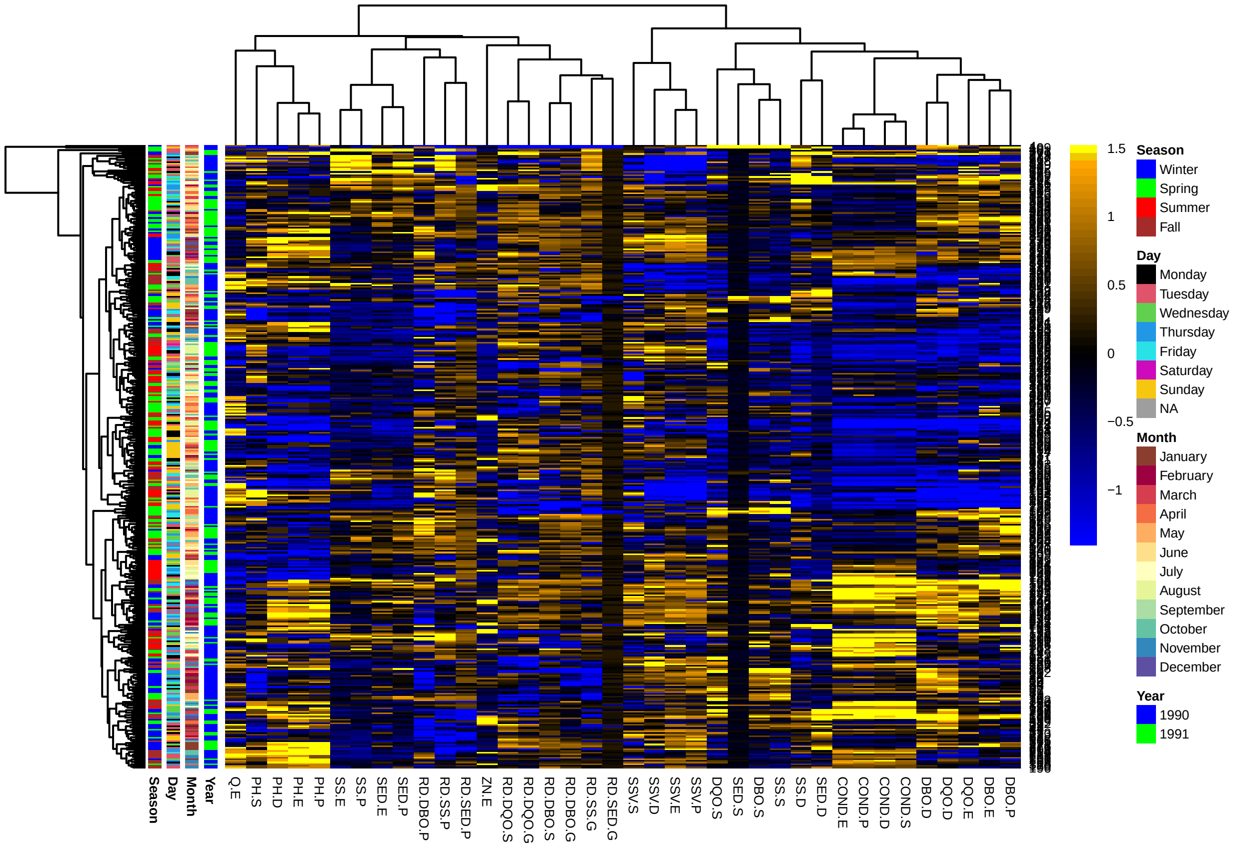
Use these breakpoints to create a better heatmap for the data. Note that the length of the breaks vector needs to be *one longer* than the length of the color vector (see `?pheatmap`).

```r
# Add code here for breakpoints
qnt<-quantile(as.numeric(data.matrix(waterScaled)),c(0.05,.95))
brks<-seq(qnt[1],qnt[2],length=length(seqPal2)+1)
fullHeat<-pheatmap(waterScaled,cluster_rows = TRUE, cluster_cols = TRUE,
                   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=waterCat[,2:5],
                   annotation_colors = list("Year"=colYear, "Month"=colMonths,"Day"=colDays, "Season"=c
                   fontsize = 5)
```



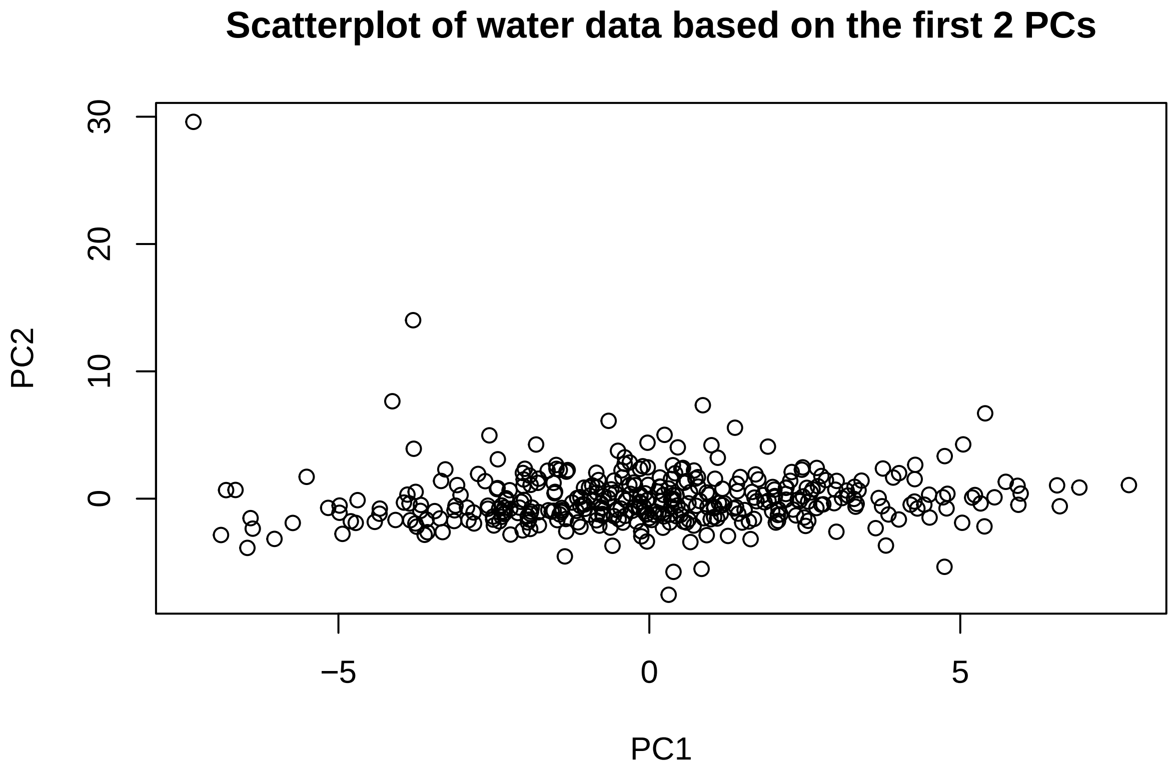
(e) (5 points) Comment on the results of your heatmap? Does it help you find patterns in the variables? In the samples/days? Describe what patterns you see. You can look at the variable descriptions if you find it helpful, but you mainly need to describe the patterns you see in the heatmap.

My answer is:
In the heatmap, yellow represents that the variable has a value higher than average and blue represents that the variable has a value lower than average.
From the clustering tree of columns, we can see that variables that share the same prefix, such as COND. and RD.DQO., tend to have a similar value in the same sample and are clustered to the same group.
From the clustering tree of rows, it is comparably hard to see the pattern since the sample size is large. However, we can roughly see that PH values are higher in winter and fall than in summer and spring ,and SED.S as well as RD.SED.G seem to be stable through out the year.

Question 5: PCA

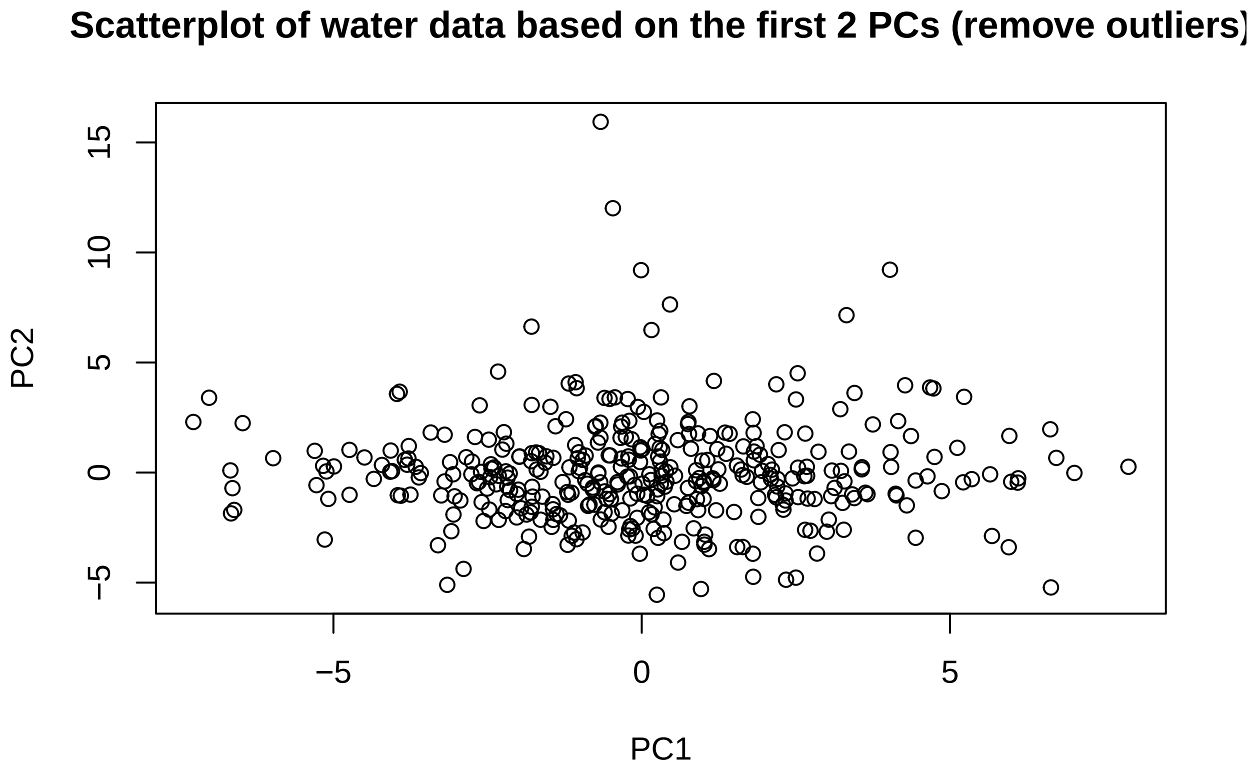
(a) (15 points) Perform a PCA of this data and plot a scatterplot of the samples based on the first 2 principal coordinates.

```r
# add code here for pca and scatterplot
pcaWater<-prcomp(waterScaled, center=TRUE, scale=TRUE)
plot(pcaWater$x[,1:2], main = "Scatterplot of water data based on the first 2 PCs")
```



(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
outlier <- pcaWater$x[,1][which(pcaWater$x[,2]>=10)]
waterScaled.remove <- waterScaled[-c(as.numeric(names(outlier))),]
pcaWater.remove<-prcomp(waterScaled.remove, center=TRUE, scale=TRUE)
outlier.date <- water[as.numeric(names(outlier)), 1]
plot(pcaWater.remove$x[,1:2], main = "Scatterplot of water data based on the first 2 PCs (remove outlie
```



```
# print the date
outlier.date
```

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