Test1 FDA

Drew Remmenga

2022-10-02

- You are encouraged to use web resources and class materials
- Please work alone and hand in your own work.
- Be spare in what you include and you will lose credit if you include too much extraneous output or information. All questions count for an equal number of points.
- Any subproblems marked GRAD are required for 500 level students but will serve as an extra credit question for the 400 level students.
- Please send me email if you have questions or any concerns. `nychka@mines.edu`
- Hand in your work in pdf format in Gradescope. You can keep the questions as part of what you hand in but you should begin your *answer* on a separate page. You can use `\newpage` to create a page break in your work.
- To comment out the answers without just deleting them use the html commenting format

```
<!--
This text is now commented out and will not be part of the rendered output.
-->
```

and check this Rmarkdown document for more details.

Points

1 (a) 10, (b) 10 (c) 10, (d) 20, (e) 10

2 (a) 10, (b) 10, (c) 10, (d) 10, (e)[GRAD] 20

400 total 100, 500 total 120.

Setup up libraries you might need:

```
suppressMessages(library(fields))
```

```
## Warning: package 'fields' was built under R version 4.1.3
```

```
## Warning: package 'spam' was built under R version 4.1.3
```

```r
suppressMessages(library(fda))
```

```
## Warning: package 'fda' was built under R version 4.1.3

## Warning: package 'fds' was built under R version 4.1.3

## Warning: package 'rainbow' was built under R version 4.1.3

## Warning: package 'pcaPP' was built under R version 4.1.3

## Warning: package 'RCurl' was built under R version 4.1.3

## Warning: package 'deSolve' was built under R version 4.1.3
```

Problem 1

This problem is a review of functional boxplots and also tests your skill in R.

The handwritten digits data set is a classic test problem used in machine learning. The goal is to identify the digit written based on a 28X28 pixel image. For this problem we will just consider the digit "6" and only the first 3000 cases. This is the **justSix** data set. The code below reads in these data and creates images of the first 10 cases in this data set. These data are a matrix of 3000X784 grey levels. That is rows index cases and the columns index the different pixels. To draw a nice image of each case you need to reshape the vector of 784 values into a 28X28 matrix (see example below and the code for **tempResphape**). It is convenient to work with the long vector for computation (e.g. applying the **fbplot** function) and the image for interpretation.

Read in the data

```
load("justSix.rda")
dim( justSix)
```

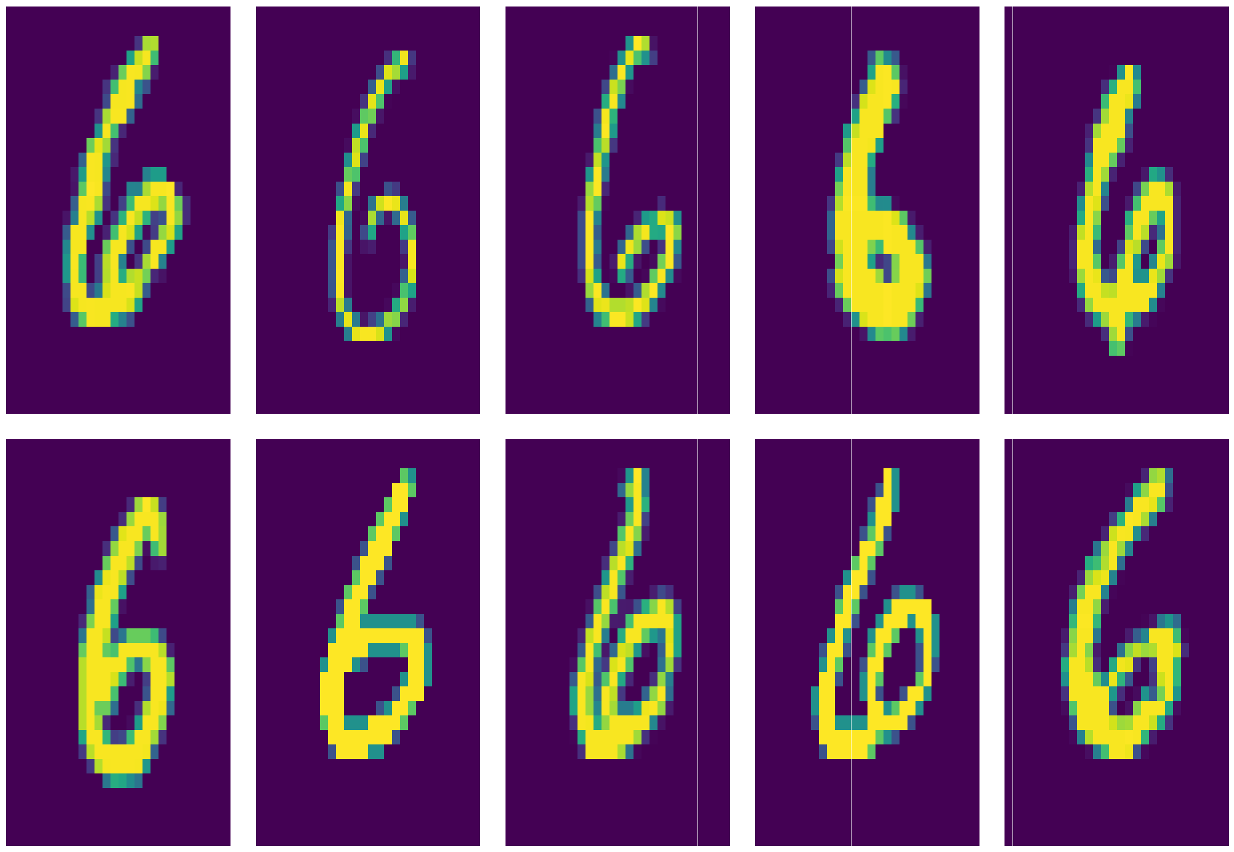
```
## [1] 3000  784
```

Take a look at the first 10 cases

```
set.panel( 2,5)
```

```
## plot window will lay out plots in a 2 by 5 matrix
```

```
zr<- range(c(justSix) )
par( mar=c(0,1,1,0))
for ( k in 1:10){
  tempReshape<- matrix(justSix[k,],28,28)
image(tempReshape,
       col=viridis(256), xlab="", ylab="", axes=FALSE,
       zlim=zr)
}
```



Problem 1

We want to apply the **fbplot** function to these data to find the depth of each case. Since these are 2D images it is not necessary to make a 1D plot.

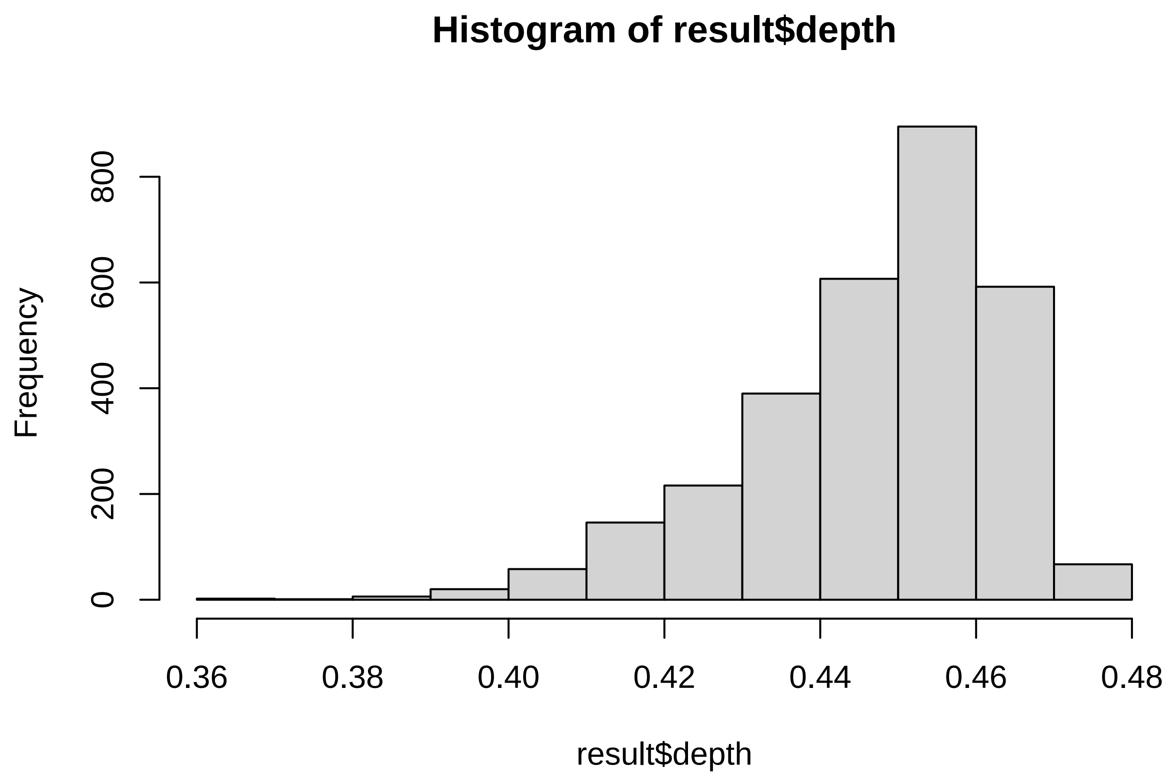
1(a)

Explain why we should use `result<- fbplot(t(justSix), plot=FALSE)` instead of `result<- fbplot(justSix, plot=FALSE)`. and what is the effect of the *plot* argument being set to FALSE.

We turn off the plots and just get the depths. We want it transposed so that we receive the data properly.

##1(b)

```
suppressWarnings(result<- fbplot(t(justSix), plot=FALSE))
hist(result$depth)
```



Find the depths of the 3000 cases and make a histogram of the depths. (When I run fbplot I get some warnings but I think these are due to all the zero valued pixels on the image borders.)

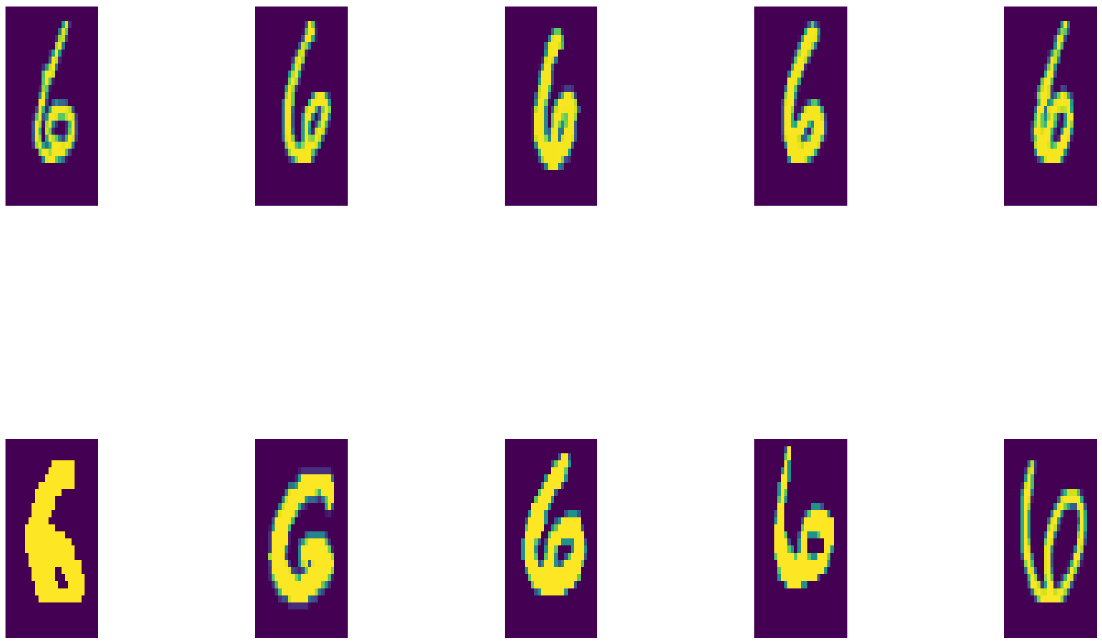
##1(c) Make a plot similar to the panel of the 10 example images above but for the first row plot the 5 *deepest* cases and for the second row plot the 5 "*shallowest*" cases. That is, the cases that have the 5 smallest depths. (Finding these two groups is a bit tricky but the **order** function is helpful to find the indexes of these cases, or you can re-sort the data by the d

order(result$depth, decreasing = TRUE)order(result$depth,decreasing = FALSE)

```r
set.panel(2,5)
```

```
## plot window will lay out plots in a 2 by 5 matrix
```

```r
x = c(164,2652,2960,137,1194,2885,352,2232,2506,1732)
for (k in x) {
tempReshape<- matrix(justSix[k,],28,28)
image(tempReshape,
       col=viridis(256), xlab="", ylab="", axes=FALSE,
       zlim=zr)
}
```



1(d)

Find the mean image for these data:

```r
meanSix<- apply(justSix, 2, mean )
invisible(rbind(justSix,meanSix))
dim(justSix)
```

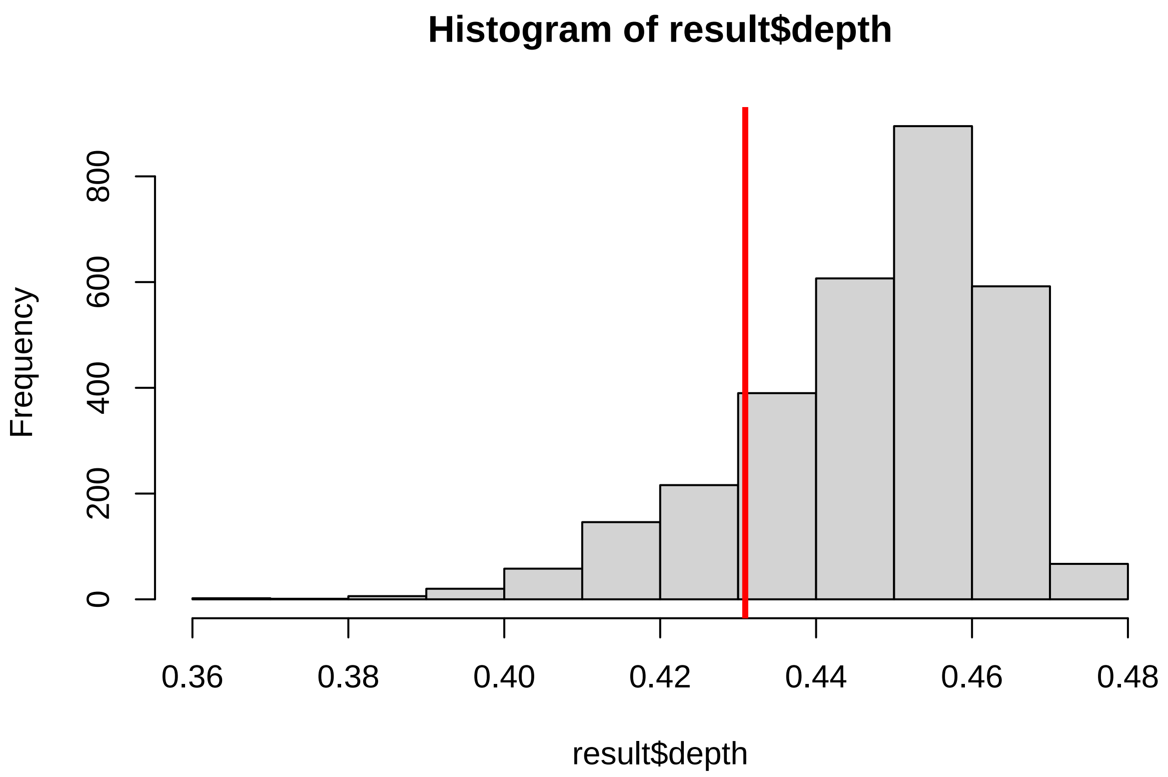
```
## [1] 3000  784
```

Add this as the 3001 row to **justSix** and use this augmented dataset to find the depth of the mean image relative to the 3000 original cases. Add the depth of the mean image as a vertical line to your depth histogram.

```r
suppressWarnings(result<- fbplot(t(justSix), plot=FALSE))
hist(result$depth)
result$depth[3000]
```

```
## [1] 0.4309094
```

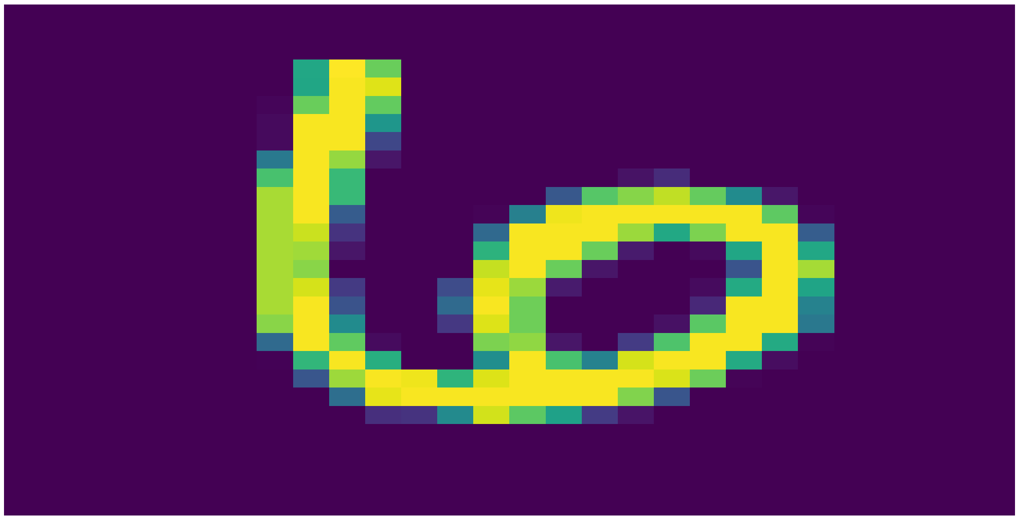
```r
abline(v = result$depth[3000],                              # Add line for mean
       col = "red",
       lwd = 3)
```



1(e)

Create an image plot of the mean image and compare it to your deepest image from (c) Its rotated for one thing. It also is not as slender.

```r
tempReshape<- matrix(justSix[3000,],28,28)
image(tempReshape,
       col=viridis(256), xlab="", ylab="", axes=FALSE,
       zlim=zr)
```



Give a reason why the mean image is not the deepest and count the number of cases that are in fact deeper than the mean image. 164 The mean image is the mean of the individual points and not necessarily the deepest graph.

```
options(max.print = 100)
order(result$depth)
```

```
##   [1] 2885  352 2232 2506 1732  796 1312  960 2339 1116 2439 1146 1335 2374  750
##  [16] 2262 1242 1119 1407 1039 1896 1502 2924 1900  138 1400 1410 2353 1714  851
##  [31] 1246 2775 2495  342 2970  565 2417 1854 1019 1483  437 2278 2492 2006  379
##  [46] 2178 2794 2007  868 1433 1060  710 1253 2528 1984  621 1416 2594  581 1784
##  [61] 1016 2136 1723  488 2619 2870  865 1040 2373 2264 1364  788  912  119  835
##  [76] 1049 1290  888  373 1807 1032 1379 2905 1404  620  873 1616 2712 2220 2817
##  [91]  487  867 1753 1261  569 2272 1334  825  553 2614
##  [ reached getOption("max.print") -- omitted 2900 entries ]
```

Problem 2

Load the Golden ozone data frame with

```r
load("GOzone2021.rda")
names( GOzone2021)
```

```
## [1] "O3"    "date"  "hour"  "day"   "month"
```

O3 are hourly, surface ozone measurements in parts per billion (PPB). **date** is a date *object*, **hour** hour of day for measurment, **day** the day in the year (1 through 365) and **month** the month.

These data are one of 34 stations and serve as the primary source for measuring the ozone levels in Colorado. The monitoring network used to determine whether Colorado is out of compliance with the EPA pollution standards for ozone (O_3). High levels ($> 80 PPB$) are associated with respiratory health problems among other issues. It is well known that ozone tends to have daily cycle based on sunlight and use of cars and typically peaks in the afternoon. The data here is the station that is closest to Golden.

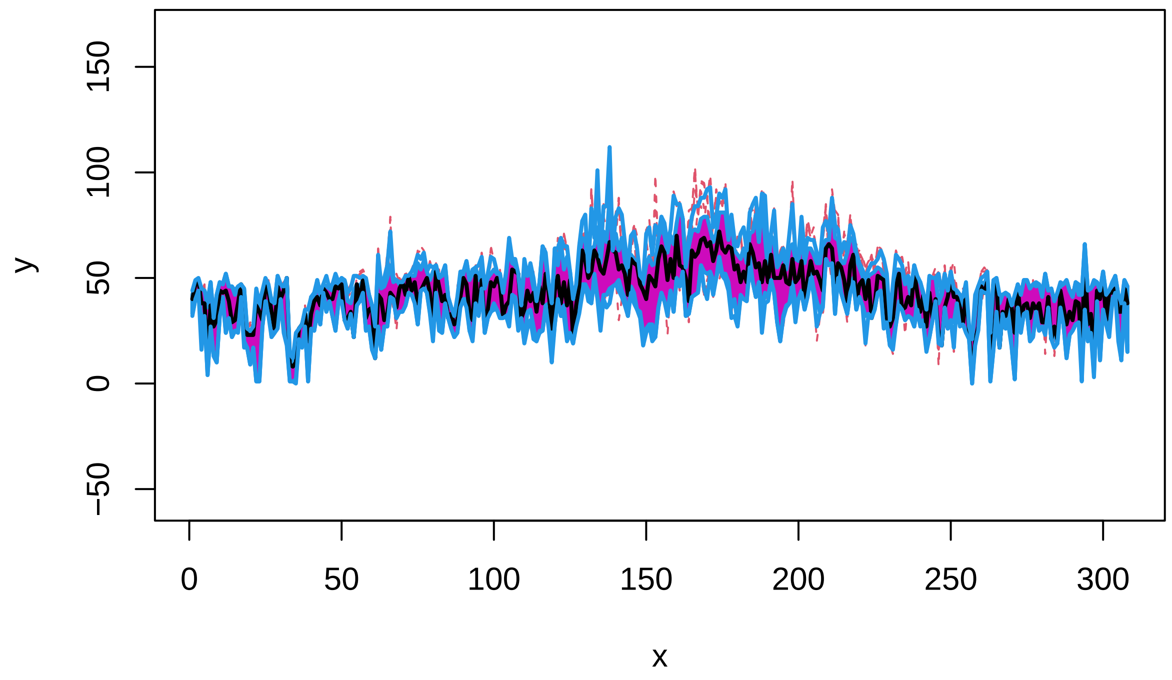
You may need this code for some questions to reshape this time series as a handy **data.frame** with days and hours. The rows are days and columns hours. Note that hours are recorded as 0 to 23 – not 1 to 24.

```r
O3<- matrix( NA, nrow=365, ncol=24)
O3[ cbind(GOzone2021$day, GOzone2021$hour+1) ]<- GOzone2021$O3
O3<- data.frame( O3)
names( O3)<- paste0("hour",0:23)
```

2(a)

Create a functional boxplot that summarizes the shape of the daily curves (the diurnal cycle of ozone) across the different days in this year. Are there any days that could be considered outliers based on the functional boxplot computation? Keep in mind that the fbplot function requires complete data so you will have to omit days with NAs.

```r
O3na <- na.omit(O3)
fbplot(O3na)
```



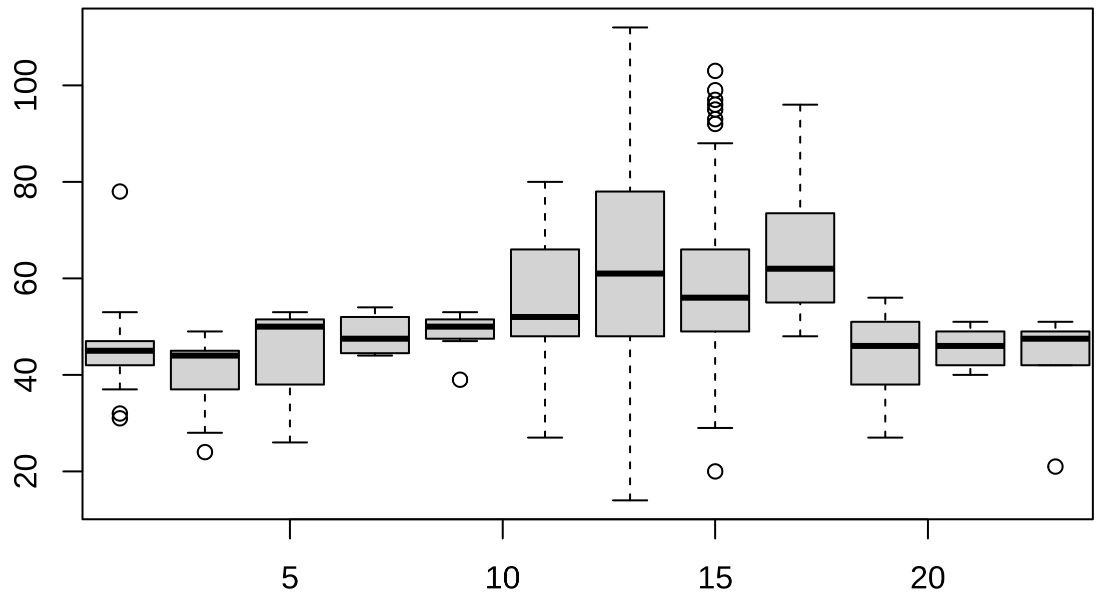
```
## $depth
##     hour0     hour1     hour2     hour3     hour4     hour5     hour6     hour7
## 0.3996624 0.4142463 0.4203428 0.4139022 0.3902015 0.3535020 0.3427266 0.3765293
##     hour8     hour9    hour10    hour11    hour12    hour13    hour14    hour15
## 0.4143463 0.4492048 0.4355767 0.3952216 0.3452822 0.3178583 0.3095415 0.3196670
##    hour16    hour17    hour18    hour19    hour20    hour21    hour22    hour23
## 0.3527874 0.3958127 0.4273598 0.4447434 0.4439906 0.4243130 0.4040620 0.3886134
##
## $outpoint
## [1]  7  8 14 15 16 17
##
## $medcurve
## hour9
##    10
```

2(b)

Find the maximum ozone value for each day and also the hour that it occurs. Make a plot over time for both of these statistics and comment on any patterns – or lack of pattern. Consider using the fields **bplot.xy** function if many points are plotted on top of each other.

Hint: The function **which.max** is a handy R function that gives the index of where the maximum occurs.

```r
O3[, "max"] <- apply(O3[, 1:23], 1, max)
O3[, "maxhour"] <- apply(O3[,1:23],1,which.max)
bplot.xy(O3$maxhour,O3$max)
```



## 2(c) For the month of July fit the daily cycle using a natural spline basis and by OLS. Select the number of basis functions by minimizing the GCV criterion. Create a figure of the GCV function and locate its minimum. (See the solution to Problem 2 from Homework 3 as a guide for this analysis.)

```r
naturalSplinefit=function(s, y, sKnots, sGrid=NULL, nGrid=80) {
  require(fda)
  sKnots <- sort(sKnots)
  N <- length(y)
  P <- length(sKnots)
  Phi <-  naturalSplineBasis(s, sKnots)
  if( is.null(sGrid)){
    sGrid<- seq( min( s), max(s), length.out=nGrid)
  }
  # omit intercept beecause it is already included as part of the
  # basis
  df<- data.frame(Phi)
  fit <- lm(y ~ . - 1, data=df)
  # fit$coefficents has the OLS coefficients to represent the function
  GCV <- (1 / N) * sum(fit$residuals ^ 2) / (1 - P / N) ^ 2

  # the GCV criterion note how the "lm" object is used to
  # get the residuals -- they do not have to be recomputed.
  GCV <- (1 / N) * sum(fit$residuals ^ 2) / (1 - P / N) ^ 2
  # prediction standard errors on the grid
  # predicted values at the grid -- this can be different
  # from the data locations
  # explicitly this is fHat <- PhiGrid %*% fit$coefficients
  PhiGrid <- naturalSplineBasis(sGrid, sKnots)
  SE <- predict( fit, newdata = data.frame(PhiGrid),
                 se.fit=TRUE )

  outObject <- list(lmObject = fit,
                    fHat = SE$fit, SE= SE$se.fit,
                    GCV = GCV)

  return(outObject)
}
naturalSplineBasis <- function(sGrid,
                               sKnots,
                               degree = 3,
                               derivative = 0) {
  boundaryKnots<- c( min(sKnots),max(sKnots))
  sKnots0<- c( rep( boundaryKnots[1],degree),sort(sKnots),
               rep( boundaryKnots[2],degree) )
  testRight<- sGrid < min(sKnots)
  testLeft <- sGrid > max(sKnots)

  basis <- splineDesign(sKnots0, sGrid,
                        ord= degree+1, outer.ok=TRUE,
                        derivs=derivative)
  # set up constraints to enforce natural BCs.
  const <- splineDesign(sKnots0, boundaryKnots, ord = degree+1,
                        derivs = c(2,2))
  qr.const <- qr(t(const))
  QBasis<- t(qr.qty( qr.const, t(basis) ))
  basis <- QBasis[,-(1:2)]
  basis

  return( basis )

}
```

```r
O3<- matrix( NA, nrow=365, ncol=24)
O3[ cbind(GOzone2021$day, GOzone2021$hour+1) ]<- GOzone2021$O3
O3<- data.frame( O3)
names( O3)<- paste0("hour",0:23)

y <- unlist(O3[182:212,])
s <- 1:(31*24)
b <- 2:20
nGCV = c()
for (x in b)
{
sKnots = seq(1,30,length=x)
fit=naturalSplinefit(s,y,sKnots, sGrid = NULL, nGrid=80)

nGCV[x]=fit$GCV
}
```

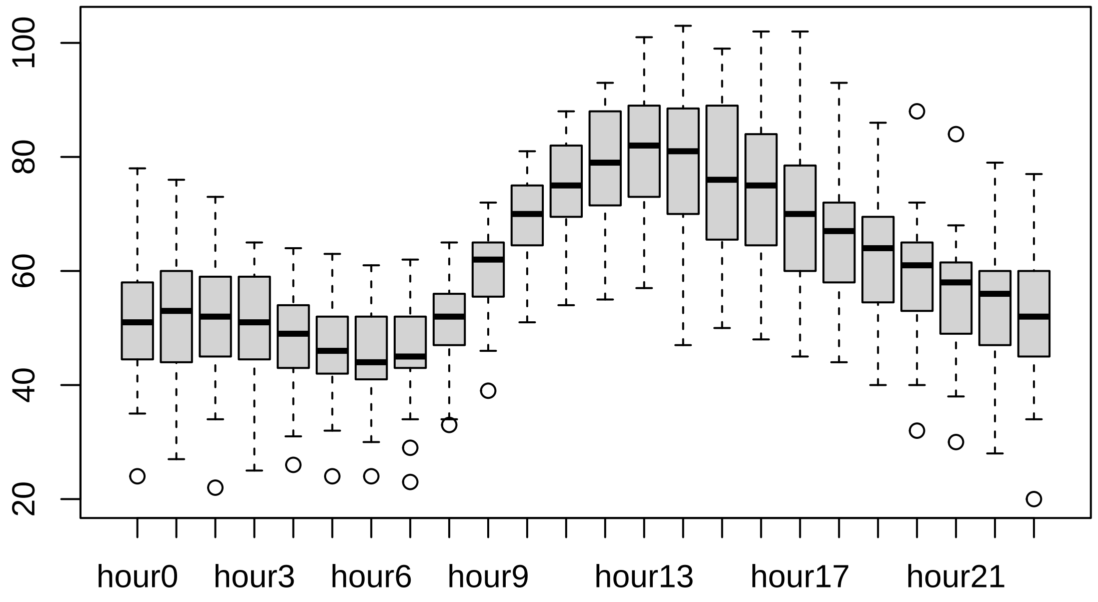
2(d)

Create a set of boxplots for the hours of the day and showing the ozone measurements for July. Add to this plot the curve that you fit to the daily cycle from part (c) and using the GCV estimate of the number of basis function. That is, evaluate your OLS model at the number of basis functions chosen by GCV and add this to your boxplot figure. This is a simlar style as the figure on page 9 of the HW3 solutions and the R code used for that figure might be helpful here.

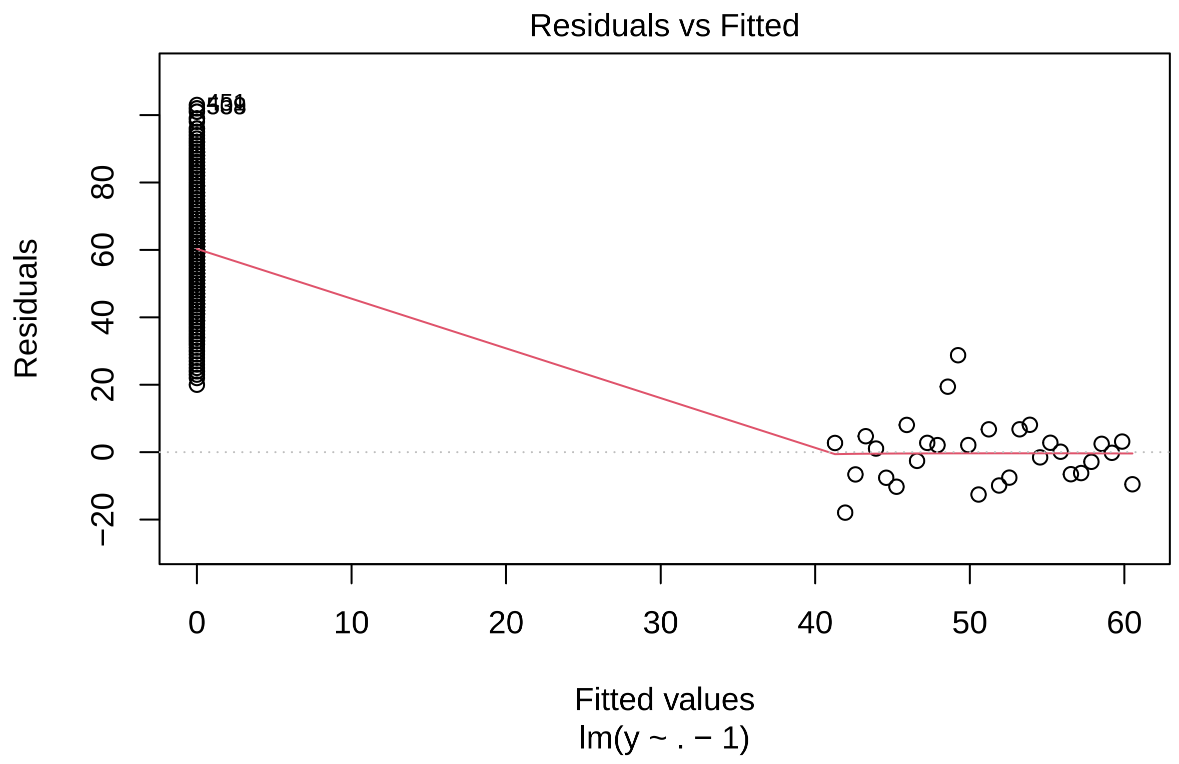
Make sure there is a boxplot for each hour. If you need to specify the breaks argument to do this use `(0:24)-.5` to get the right bin spacing.

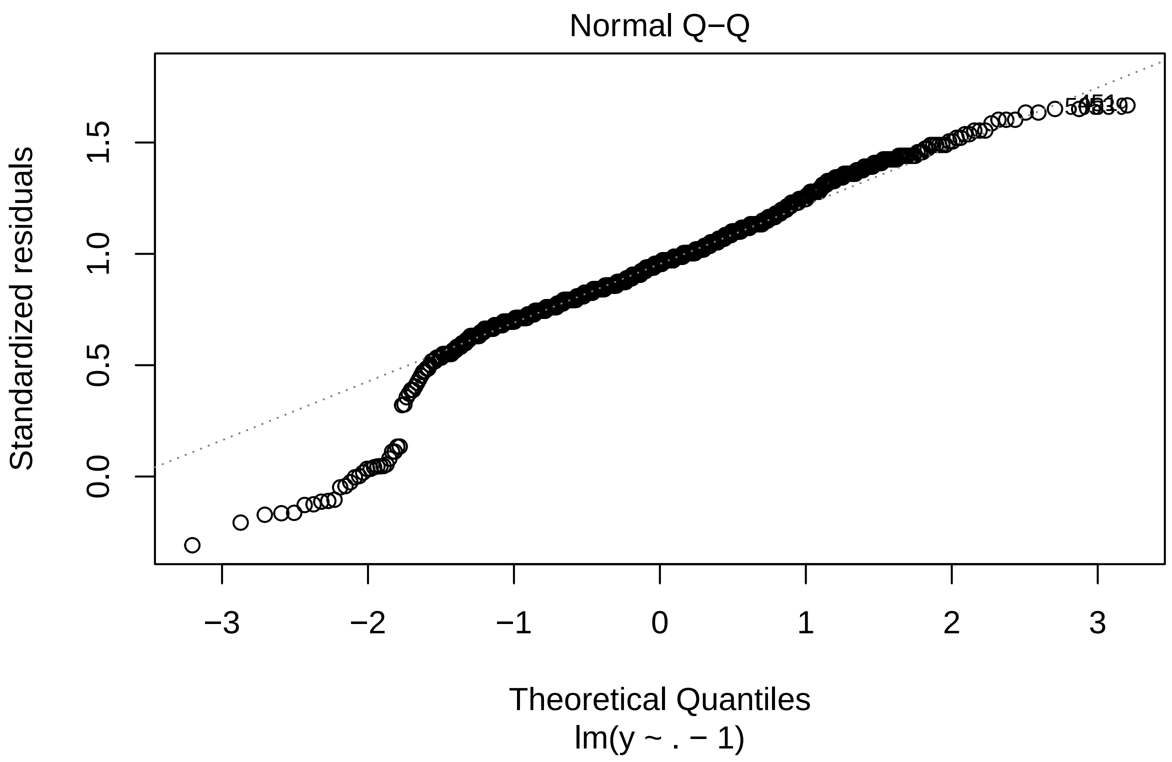
If you are unable to complete (c) then just use 7 basis functions for this question.

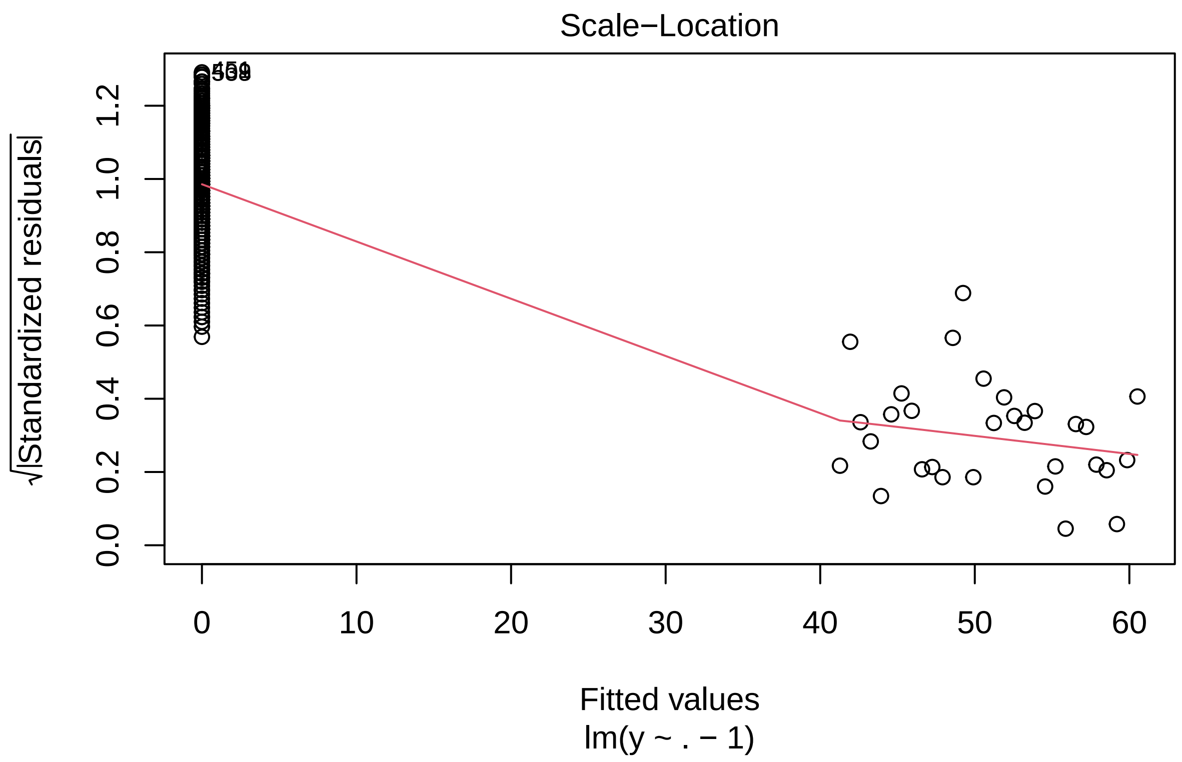
```
boxplot(O3[182:212,])
```

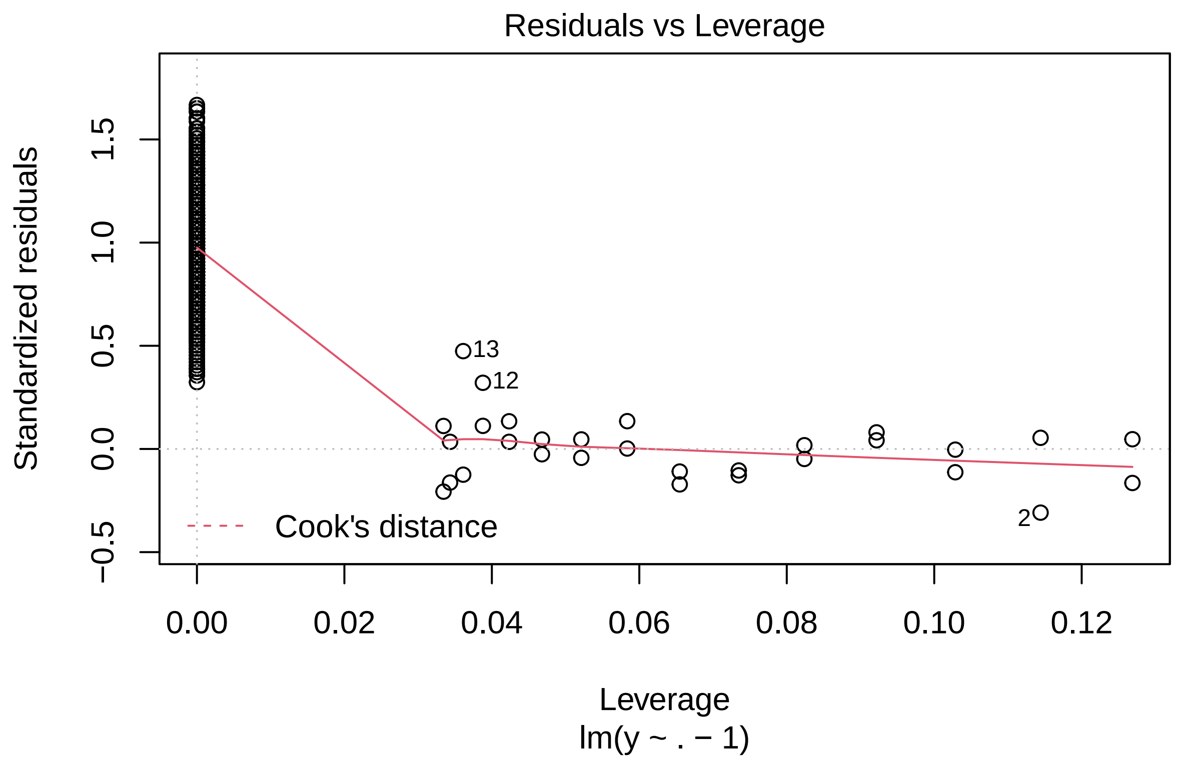


```
sKnots= seq(1,30,length=2)
fit=naturalSplinefit(s,y,sKnots,sGrid=NULL,nGrid = 80)
plot(fit$lmObject)
```


Residuals vs Fitted
Residuals
-20
0
20
40
60
80
0
10
20
30
40
50
60
Fitted values
lm(y ~ . − 1)
451
509
508


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ . − 1)
0.0
0.5
1.0
1.5
−3
−2
−1
0
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
10
20
30
40
50
60
Fitted values
lm(y ~ . − 1)



## 2(e) GRAD Add to your figure in 2(d) a 95% confidence envelope for the diurnal cycle. (See the R script and pdf *Lecture3.3CIExample.R* for the details how to do this and use the Bonferroni method.)