FINAL PROJECT PHC 6068

Sumia Tahir

Due: 11:59 pm, Dec 9th, 2019

BACKGROUND/INTRODUCTION

Occasionally the need arises to plot a piecewise function and to display the individual functions separately on the plot. For example, homework assignment #2 required us to plot the Huber function, showing the two functions in different colors. In other situations, only a single function may be plotted but different segments of the function may be of interest that one may want to highlight.

This theme also extends to datasets, where one would like to highlight certain areas of the data plot and label those sections.

The **segplots** package includes two functions, **segcurve** and **segdat**.

The *segcurve* function plots a curve of the user-specified function (formula expression) and allows the user to select specific segments of interest that will be plotted in a different color. The labels for the segments are included in the legend. The maximum and minimum points of the function are also plotted and the coordinates are displayed in the legend.

The *segdat* function uses a dataset(x,y) to make the plot, instead of a formula. The output is the same as for *segcurve*.

METHODS

The "segcurve()" function

The *segcurve* function allows the user to plot a **mathematical expression** or a **piecewise function** and highlight segments of interest in a different color.

The segcurve() function takes the following arguments:

segcurve(f, xrange, title, xl, yl, nseg, segs, segcols, seglabs)

f is the formula or expression for the function. It should be specified outside of calling *segcurve*. For example, if the formula is **x^3 + x^2**, then the following line should be added before *segcurve*: **f <- function(x) {x^3 + x^2}**

The second argument, **"xrange"**, gives the vector for the domain of the function. It is written in the form **xrange <- c(x1, x2)**

The arguments, **"title"**, **"xl"** and **"yl"**, specify the **figure title**, **x-axis label** and **y-axis label**, respectively. All of these should be specified in quotes, eg. **title="Figure title"**, because they take a character string as input.

The argument, **"nseg"**, allows the user to specify the **number of segments** of interest in the plot that will be of a different color.

The argument, **"segs"**, is a vector that lists all of the **x-values that give the starting and ending** of each segment. For example, if we want to select the segments (3, 7) and (10, 13), the "segs" argument would look like this: **segs <- c(3,7,10,13)** (The length of segs should be **2*nseg.)

The **"segcols"** argument specifies the **color for each segment**. This is a vector of characters which has the same length as nseg and is specified as follows, with colors within quotes "" Example, **segcols <- c("red", "blue", "green")**

The **"seglabs"** argument specifies the label for each segment. This is a vector of characters which has the same length as nseg and is specified as follows, with labels within quotes "" Example, **seglabs <- c("segment 1", "segment 2")**

The code for the **segcurve** function is below:

```
segcurve <- function (f, xrange, title, xl, yl, nseg, segs, segcols, seglabs){

  ##plot initial curve
  par(oma = c(1, 1, 1, 8))
  curve(f, from=xrange[1], to=xrange[2], main=title, xlab=xl, ylab=yl, lwd=2)
  usr <- par("usr")

  ## minimum value of function within interval specified
  miny <- optimize(f, xrange, tol=0.00001)
  ymin <- miny$objective
  xmin <- miny$minimum
  min <- paste(round(c(xmin, ymin), 2), collapse=",")
  points(xmin, ymin, col="red", pch=19)

  ## maximum value of function within interval specified
  maxy <- optimize(f, xrange, tol=0.00001, maximum=TRUE)
  ymax <- maxy$objective
  xmax <- maxy$maximum
  max <- paste(round(c(xmax, ymax),2), collapse=",")
  points(xmax, ymax, col="blue", pch=19)

  ##create matrix for segment pairs
  segmat <- matrix(segs,nrow=nseg, ncol=2, byrow=TRUE)

  ## add segments of different colours
  for (i in 1:nseg) {
    clip(segmat[i,1], segmat[i,2], usr[3], usr[4])
    curve(f, col=segcols[i], add=T, lwd=2)
  }
  ## create empty plot to add legend
  par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0), new = TRUE)
  plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")

  legend("topright", legend=c(seglabs, paste("min:", min), paste("max:", max)),
         col=c(segcols, "red", "blue"), inset=c(0,0.1), xpd = TRUE,
         bty = "n", lty=c(rep(1, nseg), NA, NA), pch = c(rep(NA, nseg), 19, 19), lwd=2)
}
```

The main section of the code is the 'for loop' that adds the colored segments:

First, a matrix is generated from the *nseg* argument, that includes pairs of starting and ending x-values for each segment.

segmat <- matrix(segs,nrow=nseg, ncol=2, byrow=TRUE)

for (i in 1:nseg) { clip(segmat[i,1], segmat[i,2], usr[3], usr[4]) curve(f, col=segcols[i], add=T, lwd=2) }

The 'clip' function removes the segment of interest and replaces it with the function but in a different color. This plot is overlayed the initial plot.

The "segdat()" function

The *segdat* function is the same as the *segcurve* function except that it uses a **dataset (x,y)** instead of a formula expression.

The segdat() function takes the following arguments:

segdat(x, y, title, xl, yl, nseg, segs, segcols, seglabs)

x is a vector of x-values from the data. This should be specified outside of calling the *segdat* function.

y is a vector of y-values from the data. This should be specified outside of calling the *segdat* function.

Example, **y <- rnorm (30, 0, 1); x <- 1:30**

The arguments, **"title"**, **"xl"** and **"yl"**, specify the **figure title**, **x-axis label** and **y-axis label**, respectively. All of these should be specified in quotes, eg. **title="Figure title"**, because they take a character string as input.

The argument, **"nseg"**, allows the user to specify the **number of segments** of interest in the plot that will be of a different color.

The argument, **"segs"**, is a vector that lists all of the **x-values that give the starting and ending** of each segment. For example, if we want to select the segments (3, 7) and (10, 13), the "segs" argument would look like this: **segs <- c(3,7,10,13)** (The length of segs should be **2*nseg.)

The **"segcols"** argument specifies the **color for each segment**. This is a vector of characters which has the same length as nseg and is specified as follows, with colors within quotes "" Example, **segcols <- c("red", "blue", "green")**

The **"seglabs"** argument specifies the label for each segment. This is a vector of characters which has the same length as nseg and is specified as follows, with labels within quotes "" Example, **seglabs <- c("segment 1", "segment 2")**

The code for **segdat** is below:

```
segdat <- function (x, y, title, xl, yl, nseg, segs, segcols, seglabs){

  ##create initial plot
  par(oma = c(1, 1, 1, 8))
  plot(x,y, type="l", main=title, xlab=xl, ylab=yl,lwd=2)
    usr <- par("usr")

  ## minimum value
  ymin <- min(y)
  xmin <- x[which.min(y)]
  min <- paste(round(c(xmin, ymin), 2), collapse=",")
  points(xmin,ymin, col="red", pch=19)
```

```
  ## maximum value
  ymax <- max(y)
  xmax <- x[which.max(y)]
  max <- paste(round(c(xmax, ymax),2), collapse=",")
  points (xmax, ymax, col="blue", pch=19)

  ## create matrix for segment pairs
    segmat <- matrix(segs,nrow=nseg, ncol=2, byrow=TRUE)

  ## add segments to initial plot
    for (i in 1:nseg) {
    clip(segmat[i,1], segmat[i,2], usr[3], usr[4])
    lines(x,y,type="l", col=segcols[i], lwd=2)
  }

  ## create empty plot to add legend
  par(fig = c(0, 1, 0, 1), oma = c(0, 0, 0, 0), mar = c(0, 0, 0, 0), new = TRUE)
  plot(0, 0, type = "n", bty = "n", xaxt = "n", yaxt = "n")

  legend("topright", legend=c(seglabs, paste("min:", min), paste("max:", max)),
         col=c(segcols, "red", "blue"), inset=c(0,0.1), xpd = TRUE,
         bty = "n", lty=c(rep(1, nseg), NA, NA), pch = c(rep(NA, nseg), 19, 19), lwd=2)
}
```

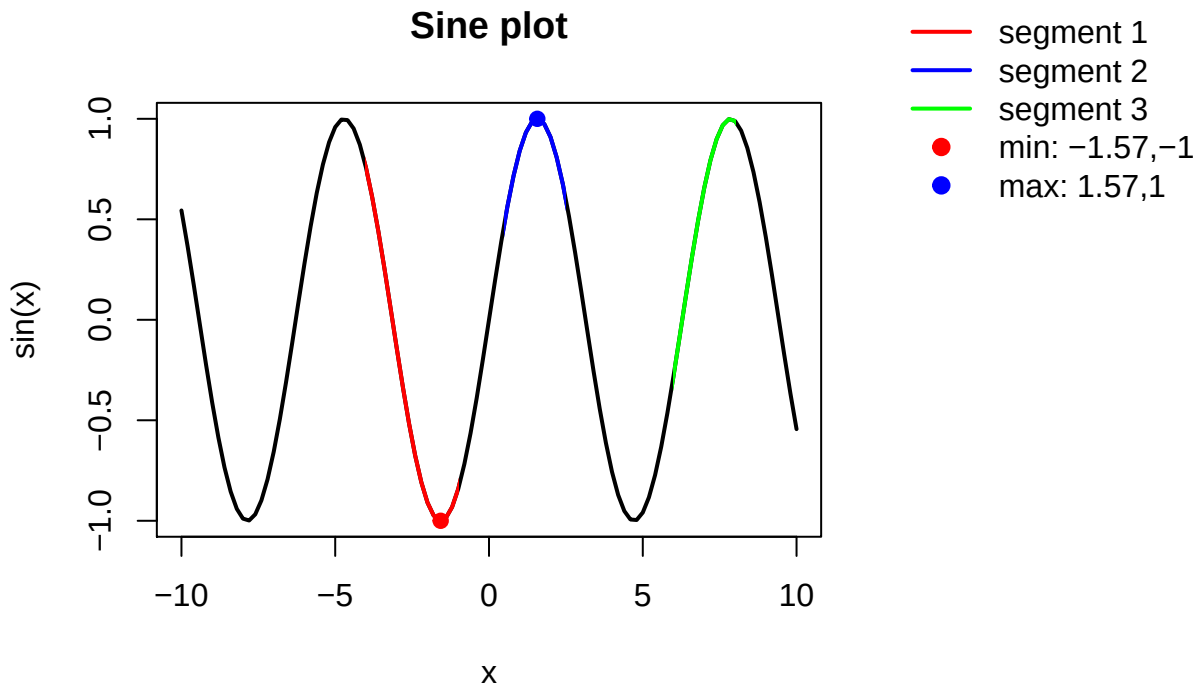
RESULTS

Examples of *segcurve()*

Example 1

We can plot the sine curve with domain (-10,10) and select three segments of interest: (-4,-1), (0.5, 2.5), (6,8). We can specify the colors of the segments to be red, blue and green, respectively, with the corresponding labels, segment 1, segment 2, segment 3.

```
library("segplots")
#>
#> Attaching package: 'segplots'
#> The following objects are masked _by_ '.GlobalEnv':
#>
#>     segcurve, segdat
f <- function(x) {sin(x)}
segcurve(f, xrange=c(-10,10), nseg=3, segs=c(-4,-1,0.5,2.5,6,8), segcols=c("red", "blue", "green"),
         seglabs=c("segment 1", "segment 2", "segment 3"), title="Sine plot", xl="x", yl="sin(x)")
```



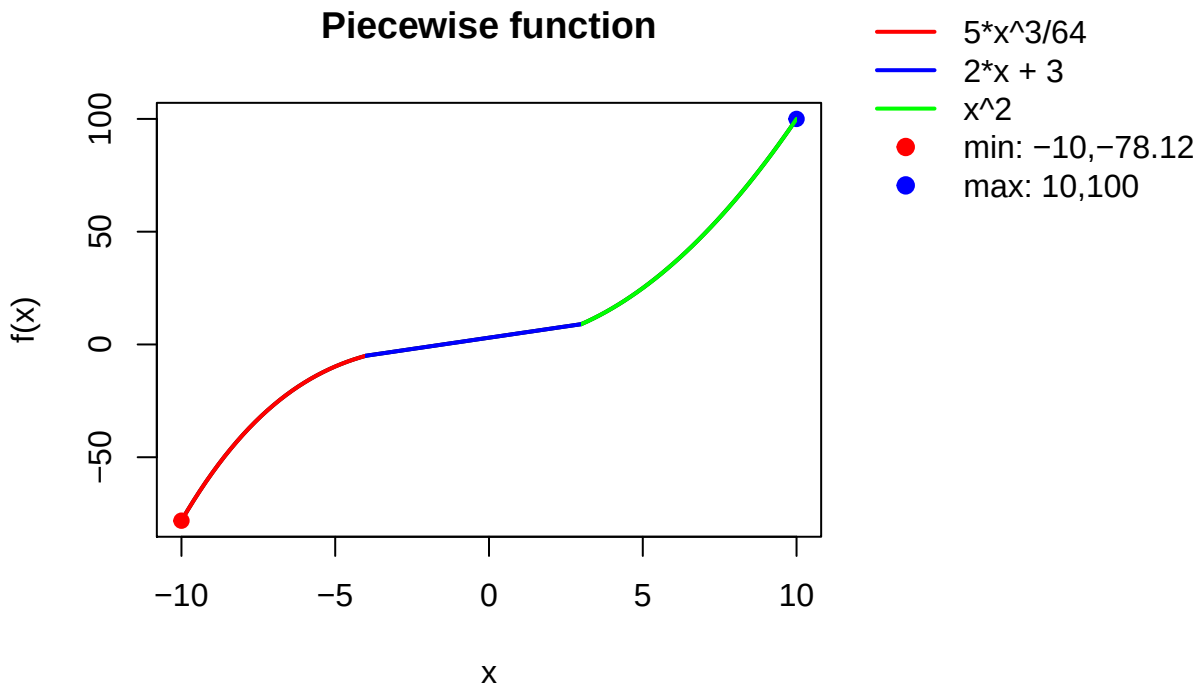
Our output shows a legend with labels corresponding to the colored segments of interest, as well as the coordinates of the maximum and minimum values of the function.

Example 2

We can also try plotting a piecewise function.

$$f(x) = \begin{cases} x^2 & \text{if } x \geq 3 \\ 2x+3 & \text{if } x > -4, x < 3 \\ \dfrac{5(x)^3}{64} & \text{if } x \leq -4 \end{cases}$$

```
f <- function (x) {ifelse(x >= 3, x^2, ifelse(x > -4 & x < 3,  2*x+3, 5*x^3/64))}
segcurve(f, xrange=c(-10,10), nseg=3, segs=c(-10,-4,-4,3,3,10),
         segcols=c("red", "blue", "green"), seglabs=c("5*x^3/64", "2*x + 3", "x^2"),
         title="Piecewise function", xl="x", yl="f(x)")
```



The output displays the function expression in the legend for each segment.

Examples of *segdat()*

Example 1

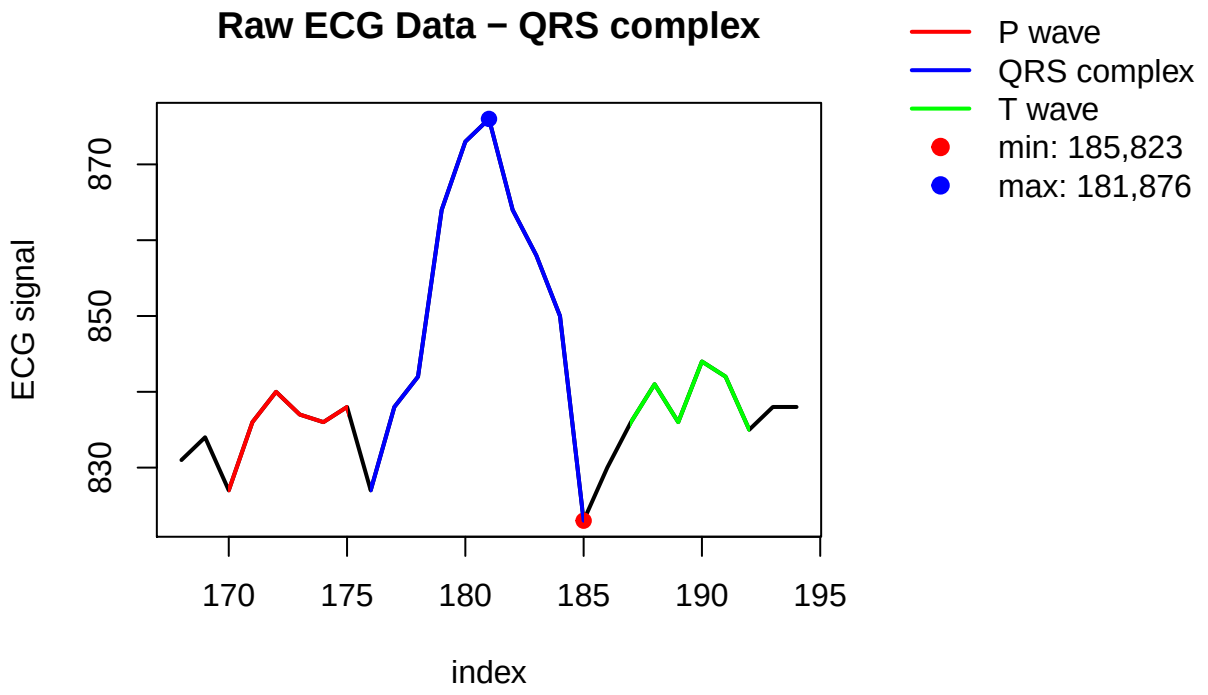
We can download the dataset data1.csv from the following webpage:

https://bioelectromagnetism.wordpress.com/2012/11/28/data-from-ecg-recording-in-todays-class/

The dataset includes 47500 samples of raw ECG recordings at a sampling frequency of 100Hz.

For our example below, we used 27 observations (index 168,194).

```
data(ecg_data)
y <- ecg_data[168:194,2]
x <- ecg_data[168:194,1]
segdat(x,y, title="Raw ECG Data - QRS complex", xl="index", yl="ECG signal", nseg=3,
    segs=c(170,175,176,185,187,192), segcols=c("red","blue","green"),
    seglabs=c("P wave","QRS complex","T wave"))
```



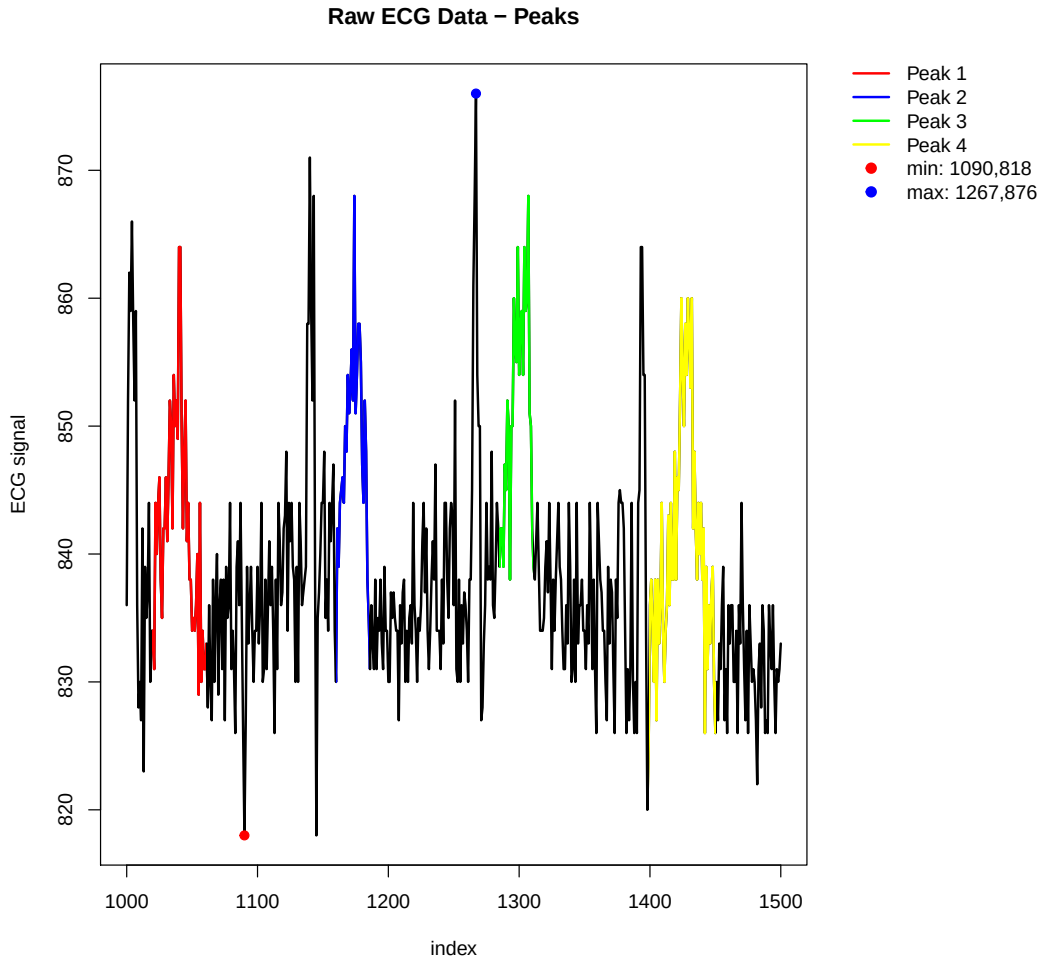
Please note: The above example is for illustrative purposes only; the ECG graph is not accurate.

Example 2

We can use the same dataset to plot a wider range of data, and highlight multiple peaks (QRS complexes). This time we will select 4 segments (nseg=4). For this graph, we used 500 observations from the index 1000:1500.

```
data(ecg_data)
y <- ecg_data[1000:1500,2]
x <- ecg_data[1000:1500,1]

segdat(x,y, title="Raw ECG Data - Peaks", xl="index", yl="ECG signal", nseg=4,
    segs=c(1020,1060,1160,1185,1285, 1310,1400,1450), segcols=c("red","blue","green", "yellow"),
    seglabs=c("Peak 1","Peak 2","Peak 3", "Peak 4"))
```



CONCLUSION

The functions **segcurve** and **segdat** serve a graphical need, where trends in data are to be visually displayed.

The **segplots package** allows the user to easily present different regions of interest in different colors.

The **segdat** function can be used for datasets such as motion graphs (space shuttle launch data) or annual sales data where certain trends in the data (peaks, drops) can be effectively labeled.

Further work on this package can include the calculation of statistics (eg. mean, median) on the selected segments. Moreover, multiple segplot graphs can be overlayed or staggered to show the change in selected regions of two samples. For example, ECG from two subjects (patient & control) can be overlayed and the regions of interest (P wave, QRS complex, T wave) can be highlighted to show how the graphs differ between patient and control.