Stat 240 - Lab 04

Dr. Lloyd T. Elliott

February 3, 2022

The goal of this lab is to get more understanding and experience with density estimation, and moving averages.

Writing a histogram function

We will write a version of the histogram function almost from scratch (some basic plotting functions will be assumed). If you struggle with this question, please read the O'Reilly text *Learning R*, or chapter 11 of the course textbook *Automated Data Collection with R*. **Question 1a, (4 points):** Create a function in R named **counts**. This function should take as parameters a numeric vector x and also a number indicating a number of bins n. The function will consider the range $[\min(x), \max(x)]$, and then consider a partition of this interval into n equally sized non-overlapping intervals (the first interval will be closed, and the remaining intervals will be half open on the left): $I_1 = [\min(x), b_1]$, $I_2 = (b_1, b_2]$, ..., $I_n = (b_{n-1}, \max(x)]$. Note that since the intervals are equally sized, the value of b_i is constrained. The function will then return a vector of length n such that the j-th element of this vector is the number of elements of x that lie in the interval I_j. Provide the code for this function: It should be of the following form:

```
counts = function( ... ) {
  ...
  return(...)
}
```

Question 1b, (3 points): Create a function in R called **histo** and provide the R code. This function should take the same x and n parameters as the **count** function you wrote in the previous part of the question, and it should plot a histogram of the vector x with the n bins defined in the previous part of the question. The only plotting functions you may make use of are the *plot* function and the *lines* function ('just add an egg'). You may not make use of the *hist* function. Provide your code for this function. Hint: there are several ways to do this, one way would be to create a new and empty plot with:

```
plot(1,
  type = 'n',
  xlab = 'x',
  ylab = 'counts',
  xlim = ...,
  ylim = ...)
```

Then make a *for* loop through the bins and call *lines* in the body of the for loop so that three lines (delinating the left, top, and right of the rectangle for that bin) are drawn. As before, your code should be in the following form:

```
histo = function( ... ) {
  ...
}
```

Question 1c, (2 points): We will now test the **histo** function. Create a vector of 200 random variables such that the first 100 random variables are independent draws from a normal distribution with mean -1 and variance 1 and the second 100 elements are independent draws from a normal distribution with mean 1 and variance 1. Call your **histo** function on this vector with 10 bins, and provide a graphic including the resulting plot.

Question 1d, (2 points): We will now test a corner case of the **histo** function. Call **histo** on the vector $x = (0, 0, 0, 1, 1, 2)$ with 3 bins, and provide a graphic including the resulting plot. Note that this is a corner case because the middle values lie at the boundary between two bins, and so their contribution to the histogram's height requires that the half-open nature of the intervals be respected.

Moving Averages with SQL

In statistics, a moving average (rolling average or running average) is a calculation to analyze data points by creating series of averages of different subsets of the full data set. Usually the mean is taken from an equal number of data on either side of a central value. Sometimes we use the average of the previous few years since we do not yet have information moving forward. Let's look at how to obtain moving with SQL Queries. Lets try to obtain moving averages for total number of athletes who obtained medals regardless of which country they are from. We will make use of the *sqlite* table provided in the previous lab. Throughout this section, please refer to Chapter 7 of the textbook of this course or to `http://www.dev.mysql.com/doc/refman/8.0/en` for more detail about the functions used (not all of them were covered in the SQL lectures). First we need to create a Frequency table where we need a unique list of year with its corresponding total medal counts. To do this we will create a new virtual table as an interim step. We'll split up the task into small pieces so that you can see what is happening.

- Obtain the medal count within each year for the Olymp_meds table.

```
mov_avg1 = "SELECT Edition, Count(Edition) AS TotalNumber
  FROM Olymp_meds GROUP BY Edition"
out = dbGetQuery(con, mov_avg1)
```

Make a nice plot of the number of athletes who obtained medals per year. Use the option type="p". Make sure axes and the title are nice and clear.

```
plot(out$Edition, out$TotalNumber,
main="Total number of athletes who obtained Olympic medals",
     xlab="year",ylab="Count",type="p",
     col="red",lwd=2,las=2)
```

- Create a virtual table (called a *VIEW*) which we will call *tot_meds* to house that output from the previous query without bringing the results into R. This will let us do other queries on that virtual table.

```
mov_avg2 = "CREATE VIEW tot_meds AS SELECT Edition,
Count(Edition) AS TotalNumber
FROM Olymp_meds GROUP BY Edition"
dbSendQuery(con, mov_avg2)
```

Check that the above worked by finding the new virtual table inside the database.

```
dbListTables(con)
```

- Do this to delete the temporary table so that you don't get stuck with errors when you rerun your code. If you run this you will need to re-run the previous step to re-build the virtual table.

```
#dbSendQuery(con, "drop view tot_meds")
```

- Query the virtual table to extract year and total medal.

```
mov_avg3 = "SELECT Edition, TotalNumber FROM tot_meds"
out = dbGetQuery(con, mov_avg3)
```

Add a line plot of the total number of athletes who received medals per year from this table as an addition to the plot in part 1a. Make sure you chose a different colour for the line.

```
#remaking the plot from 1a (not necessary for students):
plot(out$Edition, out$TotalNumber,
main="Total number of athletes who obtained Olympic medals",
     xlab="year",ylab="Count",type="p",
     col="red",lwd=2,las=2)

lines(out$Edition, out$TotalNumber,col=1,lwd=2)
# Note that the line must go through the points
# to show that it is from "lines" instead of type="b".
```

- Now we will compute a moving average by defining an index within the year and averaging everything within that window. This requires us to define two new tables t1 and t2. Remember that we can specify where a variable comes from by using ".". So *t1.year* means that we use the variable *year* from table *t1*.

  We will use table t1 to define an index over year and use that index to average everything within the window:

  `WHERE t2.Year BETWEEN (t1.Year-4) AND (t1.Year+4) .`

  Note that the moving average will contain more or less Olympics within the window depending on the year. The reason is that the Olympics were occasionally cancelled due to world wars.

```
check = "SELECT * FROM tot_meds AS t,
        (SELECT t1.Edition, AVG(t2.TotalNumber) AS mavg
            FROM tot_meds AS t1, tot_meds AS t2
  WHERE t2.Edition BETWEEN (t1.Edition-4) AND (t1.Edition+4)
  GROUP BY t1.Edition) sq  WHERE (t.Edition = sq.Edition)"
movingAvg = dbGetQuery(con, check)
```

  Put a line with the moving average on the figure and make a legend(total 4 points; 2 points for the line, 2 points for the legend).

```
#remaking the plot from 1a (not necessary for students):
plot(out$Edition, out$TotalNumber,
main="Total number of athletes who obtained Olympic medals",
     xlab="year",ylab="Count",type="p",
     col="red",lwd=2,las=2)

lines(out$Edition, out$TotalNumber,col=1,lwd=2)
### New part:
lines(movingAvg$Edition, movingAvg$mavg,type="l",col=3,lwd=2)
legend("topleft",lwd=2,lty=c(NA,1,1),pch=c(1,NA,NA),
        col=c(2,1,3),
       c("medals per year points",
              "medals per year line",
              "moving average"))
```

  Now, make sure that you delete the temporary table.

Two dimensional density plots

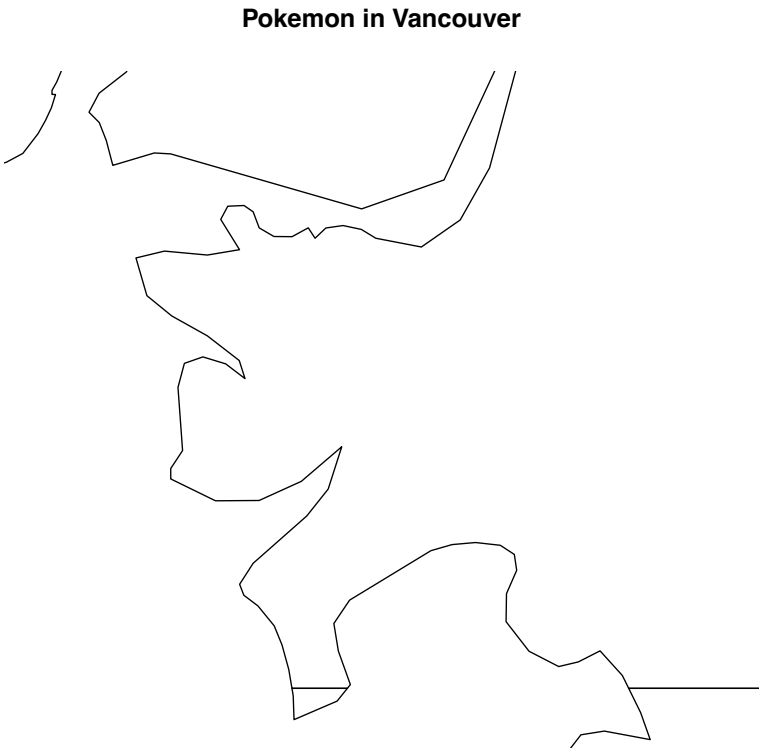
Question 2a, (4 points): Plot the locations of all of the pokemon in the `Vanpoke` table provided in the previous lab, overlayed on a map of Vancouver. Provide the resulting graphic. The following code segment may be useful:

```
library(rworldmap)

## Loading required package:  sp
## ### Welcome to rworldmap ###
## For a short introduction type :  vignette('rworldmap')

library(rworldxtra)

worldmap = getMap(resolution = "high")
NrthAm = worldmap[which(worldmap$REGION =="North America"),]
plot(NrthAm, xlim=c(-123.35,-122.65),
  ylim=c(49,49.35), main = "Pokemon in Vancouver")
```



Question 2b, (6 points): Make a two dimensional density plot with contours of the pokemon locations and overlay it onto a map of Vancouver, and provide the graphic. The following functions may be useful: *kde2d*, and *contour*. These functions are provided in the libraries *MASS* and *sp*. **Question 2c, (2 points):** In three sentences or fewer, answer these questions: Where are the peaks of this two dimensional density plot? Why are the peaks in those locations? **Question 2d, (2 bonus points):** Provide a graphic of the contour plot constructed in Question 2c, but with a more detailed depiction of Vancouver.