HW3\_DataCleaningAndTransformation

NolanBentley

9/13/2021

Table of Contents

[Q1 and Q2 1](#_Toc95396285)

[ggplot introduction 2](#_Toc95396286)

[Q3 3](#_Toc95396287)

[Q4 4](#_Toc95396288)

[Q5 4](#_Toc95396289)

[Q6 5](#_Toc95396290)

[Q7 6](#_Toc95396291)

[Q8 6](#_Toc95396292)

[Q9 6](#_Toc95396293)

[Q10 7](#_Toc95396294)

[Q11 8](#_Toc95396295)

[EC 8](#_Toc95396296)

# Q1 and Q2

Run and read through the following code. Try to understand understand the code, annotate it with comments, look up unfamiliar functions, and learn what the being used. This is how you learn to look at new and unfamiliar code and approach it like a data scientist.  
  
When you understand the code below. Go to canvas and answer Q1 and Q2.

url <- "https://utexas.box.com/shared/static/rrtbkan08hl7vgmffip87iv96splwq15.zip"  
fileName <- "chr4.depth.out"  
if (!file.exists(fileName)) {  
 zipName <- paste0(fileName,".zip")  
 download.file(url,destfile = zipName)  
 unzip(zipName,files = fileName)  
}  
samtools.depth <- read.table(fileName,stringsAsFactors = F)  
samtools.depth$V2 <- as.numeric(samtools.depth$V2)  
samtools.depth$V3 <- as.numeric(samtools.depth$V3)

# ggplot introduction

In this question, we will see our first instance of using a non-base R package. This package is called ggplot2 ([<https://ggplot2.tidyverse.org>](https://ggplot2.tidyverse.org/)) and it is enormously popular among biologist and the data scientist community at large. Consequently, it’s structure, grammar, and aesthetics has inspired software developed in many other languages.

ggplot2 (often shortened to just ggplot) can be installed on a new R installation using install.packages("ggplot2"). This will likely install several other packages. Learning to read the text printed during installation is unfortunately beyond the scope of of this course.

When successfully installed, loading it using library("ggplot2") will not produce an error (however, it may still produce warning text).

I don’t expect you to learn ggplot *today*, but I want to start exposing you to its syntax. Play with this plot! Change the parameters. Have some fun for ONCE in life…

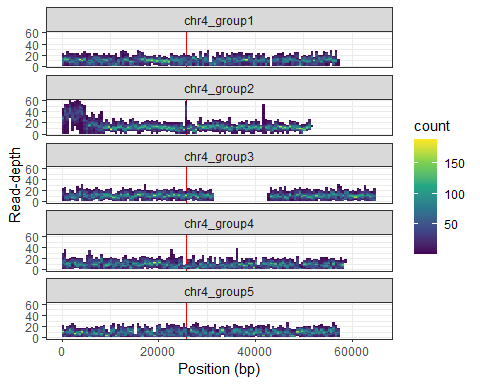
ggplot is like a mini-language unto itself… It has its own syntax.

* The input data is a data.frame
* You tell it what your input data is using the data argument.
* Then you “map” variables to different “aesthetics” using aes(). Note the strange lack of quotation marks around column names! This is like how column names are used with the $ extractor.
* After that, you add (+) layers to your plot (ex: geom\_bin\_2d is what actually adds the colored boxes).

In the following example,

* position is mapped to the x-axis position
* and read depth to the y-axis position.
* Later on we use V1 to “Facet” (split into multiple plots) based on the contig.
* The coloring indicates how many combinations of that range of position and row occurred in the dataset. Thus, its a two-dimensional histogram where coloring indicates the count.

# install.packages("ggplot2") # Run if needed  
library("ggplot2")  
  
# Tells ggplot the data.frame to use samtools.depth and which columns  
p1<-ggplot(data = samtools.depth, mapping = aes(x = V2,y = V3))+   
 # Draws a red vertical line  
 geom\_vline(xintercept = 25730,color="red")+  
 # Plots a 2d-histogram of RD vs position colored by frequency.  
 geom\_bin\_2d(binwidth = c(500,2))+  
 # Seperates the plots based on the contig  
 facet\_wrap(facets = vars(V1),ncol = 1)+  
 # Changes the fill color scaling to my favorite color-blind friendly palette  
 scale\_fill\_viridis\_c()+  
 # Changes various formatting to my favorite settings  
 theme\_bw()+  
 # Modifies the axis labels  
 labs(x="Position (bp)",y="Read-depth")  
  
# Saves the plot to file  
ggsave(filename = "samToolPosVsRd.png",plot = p1,  
 width = 15,height = 15,units = "in",dpi = 300)  
p1



# Q3

Note, this data resulted from the alignment of more than 2.7 million reads of short (~125bp) sequences of DNA to a *Drosophila obscura* genome reference! The third column in the input data indicates how many reads aligned at a given position (column 2). Go to Canvas an fill in the blanks. Look up unfamiliar terms where needed.

# Q4

Write code that returns a data.frame with only rows with 0 reads using the [ , ] method.

For the record, I have no concrete knowledge why these rows have been retained in this data set. My guess is that they occur where reads aligned, but a sequence deletion relative to the reference had occurred. Although, I could be wrong.

Example output:

## V1 V2 V3  
## 30054 chr4\_group1 33752 0  
## 31087 chr4\_group1 34824 0  
## 31233 chr4\_group1 34970 0  
## 35608 chr4\_group1 39864 0  
## 86674 chr4\_group2 37622 0  
## 113592 chr4\_group3 14910 0  
## 147006 chr4\_group3 61776 0  
## 163051 chr4\_group4 14164 0  
## 176307 chr4\_group4 29538 0  
## 181576 chr4\_group4 36038 0  
## 199941 chr4\_group4 58456 0  
## 204410 chr4\_group5 5113 0  
## 204411 chr4\_group5 5114 0  
## 204412 chr4\_group5 5115 0  
## 212241 chr4\_group5 13919 0  
## 225665 chr4\_group5 29728 0  
## 225666 chr4\_group5 29729 0  
## 225667 chr4\_group5 29730 0  
## 225949 chr4\_group5 30023 0  
## 226536 chr4\_group5 30808 0

# > samtools.depth[samtools.depth$V3=="0", ]

# Q5

Write code that removes rows with less than 11 reads using the [ , ] method.

Example output after assigning the isolated rows to a new object:

nrow(samtools.depth.gte11RD)

## [1] 121955

head(samtools.depth.gte11RD)

## V1 V2 V3  
## 29 chr4\_group1 30 11  
## 30 chr4\_group1 31 11  
## 31 chr4\_group1 32 12  
## 32 chr4\_group1 33 12  
## 33 chr4\_group1 34 13  
## 34 chr4\_group1 35 14

tail(samtools.depth.gte11RD)

## V1 V2 V3  
## 249973 chr4\_group5 57272 12  
## 249974 chr4\_group5 57273 11  
## 249975 chr4\_group5 57274 11  
## 249976 chr4\_group5 57275 11  
## 249977 chr4\_group5 57276 11  
## 249978 chr4\_group5 57277 11

> samtools.depth.gte11RD <- samtools.depth[samtools.depth$V3 > 10, ]

# Q6

Write code that produces a logical vector that isolates chromosome 4 group 2 at positions below or equal to 5,000 bp.

Example output after creating the vector:

class(chr4grp2Lte5k.logic)

## [1] "logical"

length(chr4grp2Lte5k.logic)

## [1] 250000

sum(chr4grp2Lte5k.logic)

## [1] 4969

head(samtools.depth[chr4grp2Lte5k.logic,])

## V1 V2 V3  
## 50001 chr4\_group2 23 1  
## 50002 chr4\_group2 24 2  
## 50003 chr4\_group2 25 2  
## 50004 chr4\_group2 26 2  
## 50005 chr4\_group2 27 2  
## 50006 chr4\_group2 28 2

tail(samtools.depth[chr4grp2Lte5k.logic,])

## V1 V2 V3  
## 54964 chr4\_group2 4995 27  
## 54965 chr4\_group2 4996 27  
## 54966 chr4\_group2 4997 29  
## 54967 chr4\_group2 4998 30  
## 54968 chr4\_group2 4999 30  
## 54969 chr4\_group2 5000 31

> chr4grp2Lte5k.logic <- samtools.depth$V1=="chr4\_group2" & samtools.depth$V2 < 5001

# Q7

How many positions on chr4\_group2 at positions less than or equal to 5000 bp are included in this data set?

4969

# Q8

Use logic statements to extract the positions before and after the region with zero read depth on chr4\_group3. Then look at the head or tail to find where edges of the regions are.

**Hint:** You can estimate the middle position of this assembled region. Then you can use logic to subset to positions **on this chromosome** that are less than it (look at the bottom of this subset). Then you can use logic to subset to positions **on this chromosome** that are greater than it (look at the top of the subset).

You can also just scroll through the crazy long data set like someone who doesn’t want to learn computational biology… :( … Don’t be this person. This person is boring and make Nolan sad.

The width of this region is:

#After calculating the start and end position  
endPosition - startPosition

## [1] 11602

# Q9

Install if needed and load the maps package.

Write code that uses a grepl function to extract and print only city names beginning with “San” (you will need to use a special character to indicate the beginning of the string).

# install.packages("maps")  
library("maps")  
class(us.cities)

## [1] "data.frame"

dim(us.cities)

## [1] 1005 6

us.cities[1:5,] #Visualize this data

## name country.etc pop lat long capital  
## 1 Abilene TX TX 113888 32.45 -99.74 0  
## 2 Akron OH OH 206634 41.08 -81.52 0  
## 3 Alameda CA CA 70069 37.77 -122.26 0  
## 4 Albany GA GA 75510 31.58 -84.18 0  
## 5 Albany NY NY 93576 42.67 -73.80 2

Example output:

## [1] "San Angelo TX" "San Antonio TX" "San Bernardino CA"   
## [4] "San Bruno CA" "San Buenaventura CA" "San Clemente CA"   
## [7] "San Diego CA" "San Francisco CA" "San Gabriel CA"   
## [10] "San Jose CA" "San Leandro CA" "San Luis Obispo CA"   
## [13] "San Marcos CA" "San Marcos TX" "San Mateo CA"   
## [16] "San Rafael CA" "San Ramon CA"

> san\_city <- grepl(pattern = "^San .\*", us.cities$name)

> us.cities[san\_city, 1]

# Q10

The us.cities dataset only contains cities with populations above 40,000 people. What 2 city names are most common in the us.cities dataset?

Lakewood and Springfield

Tips:

* Use a gsub function to remove the space and the state from the strings in the cities name column.
* Use the table and sort function to determine which 2 cities are the most frequent names in this dataset.

Descriptive outputs after using gsub to create a column in us.cities:

head(us.cities)

## name country.etc pop lat long capital trimmedNames  
## 1 Abilene TX TX 113888 32.45 -99.74 0 Abilene  
## 2 Akron OH OH 206634 41.08 -81.52 0 Akron  
## 3 Alameda CA CA 70069 37.77 -122.26 0 Alameda  
## 4 Albany GA GA 75510 31.58 -84.18 0 Albany  
## 5 Albany NY NY 93576 42.67 -73.80 2 Albany  
## 6 Albany OR OR 45535 44.62 -123.09 0 Albany

maxCount <- max(sortedTableOfTrimmedNames)  
#Here is the frequency of the two most frequent  
maxCount

## [1] 5

#Here are the next most frequent names  
tail(sortedTableOfTrimmedNames[sortedTableOfTrimmedNames<maxCount],10)

##   
## Wilmington Albany Arlington Bloomington Columbia Columbus   
## 2 3 3 3 3 3   
## Concord Fairfield Lawrence Newark   
## 3 3 3 3

> no\_state\_city <- gsub('.{3}$','', us.cities$name)

> tail(sort(table(no\_state\_city)))

# Q11

How many unique city names are in this dataset? Than answer is not 1005.

937

Descriptive example after trimming the names and finding the unique values:

#Print head an tail to visualize difference  
rbind(cbind(Original= head(sort(us.cities$trimmedNames)),  
 Unique = head(sort(uniCityNames))),  
 "...",  
 cbind(Original= tail(sort(us.cities$trimmedNames)),  
 Unique = tail(sort(uniCityNames))))

## Original Unique   
## [1,] "Abilene" "Abilene"   
## [2,] "Akron" "Akron"   
## [3,] "Alameda" "Alameda"   
## [4,] "Albany" "Albany"   
## [5,] "Albany" "Albuquerque"  
## [6,] "Albany" "Alexandria"   
## [7,] "..." "..."   
## [8,] "Yonkers" "Yonkers"   
## [9,] "Yorba Linda" "Yorba Linda"  
## [10,] "Youngstown" "Youngstown"   
## [11,] "Yuba City" "Yuba City"   
## [12,] "Yucaipa" "Yucaipa"   
## [13,] "Yuma" "Yuma"

> unique(no\_state\_city)

# EC

**Answer the following:**

1. What is the lowest longitude city in us.cities?

Honolulu, Hi

1. What is the highest longitude city in us.cities?

Augusta, ME

1. How many cities have populations above 200 thousand?

99

1. How many city names end in “in” not including the state abbreviation (only lowercase)?

9

1. How many city names contain an “a” followed by an “n” anywhere in the name (only lowercase)?

Examples of extra credit:

#Head of cities ending in "in"  
head(citiesEndingInIn,1)

## name country.etc pop lat long capital trimmedNames  
## 50 Austin TX TX 683404 30.31 -97.75 2 Austin

#Head of cities with an "a" then eventually an "n"  
head(citiesWithAthenN)

## name country.etc pop lat long capital trimmedNames  
## 4 Albany GA GA 75510 31.58 -84.18 0 Albany  
## 5 Albany NY NY 93576 42.67 -73.80 2 Albany  
## 6 Albany OR OR 45535 44.62 -123.09 0 Albany  
## 8 Alexandria LA LA 44933 31.29 -92.46 0 Alexandria  
## 9 Alexandria VA VA 127159 38.82 -77.09 0 Alexandria  
## 15 Altadena CA CA 43280 34.19 -118.13 0 Altadena