**Assignment 3 (worth 10% of MB6300)**

**Course co-ordinator: Dr Marcus Claesson (**[**m.claesson@ucc.ie**](mailto:m.claesson@ucc.ie)**)**

**Demonstrator: Julia Eckenberger (**[**116223742@umail.ucc.ie**](mailto:116223742@umail.ucc.ie)

**Student: James Henry Hehir (**[**120224791@umail.ucc.ie**](mailto:120224791@umail.ucc.ie)**)**

**For each question, please record your answer in a Word document, and also what you typed on the command-line to get this answer. When finished, upload the document (with your name in the title) with your answers to CANVAS under Assignments as MB6300 Assignment 3.**

**Deadline: End of 18th of April**

If you cannot meet this deadline you will have to fill in and submit the Late Submission Form (see Assignment folder on CANVAS) and provide any supporting documentation (e.g. medical cert).

All students are expected to work individually on assignments; those found collaborating with others will receive a score of zero for their work.

Include intermediate steps that you used to get your answer. If you include intermediate steps, you will obtain marks even if your final answer is incorrect.

Do not use the edit() function in any parts of this assignment.

All R code must include comments briefly describing how it works, and what it is doing.

**Q.1 The file assignment3.RData on CANVAS contains an R environment with a Microarray dataset from a human breast cancer study with both diseased and healthy samples.**

**(a) Perform a differential expression analysis with limma (use rma, quantiles, pmonly and medianpolish). Which samples contain expression information from diseased samples? How many genes are differentially expressed with an adjusted p-value less than 0.05? How many of the significantly differentially expressed genes are upregulated (i.e. expressed higher) in the diseased samples over the healthy?**

eset <- expresso(assignment3,bgcorrect.method="rma",

normalize.method="quantiles",

pmcorrect.method="pmonly",

summary.method="medianpolish")

pData(eset)

sample

GSM757755.CEL.gz healthy

GSM757756.CEL.gz healthy

GSM757757.CEL.gz healthy

GSM757758.CEL.gz diseased

GSM757759.CEL.gz diseased

GSM757760.CEL.gz diseased

So **GSM757758.CEL.gz, GSM757759.CEL.gz** and **GSM757760.CEL.gz** are the 3 samples that contain expression information from diseased samples.

annot <- pData(eset)

design <- model.matrix(~0 +annot[, "sample"])

colnames(design) <- c("diseased", "healthy")

cm <- makeContrasts(DiseasedVsHealthy = diseased-healthy, levels=design)

fit <- lmFit(eset, design)

fit2 <- contrasts.fit(fit, cm)

fit2 <- eBayes(fit2)

results <- topTable(fit2, "DiseasedVsHealthy", p.value = 0.05)

head(results)

logFC AveExpr t P.Value adj.P.Val B

231577\_s\_at 3.131748 6.995678 41.47926 4.401322e-09 0.0001268132 9.215275

225803\_at 3.451597 11.467331 41.14197 4.638800e-09 0.0001268132 9.198143

232573\_at 2.171975 9.163857 32.24667 2.221742e-08 0.0003660157 8.589701

241762\_at 3.571207 8.247122 31.32270 2.677756e-08 0.0003660157 8.503793

222457\_s\_at 1.540784 9.732538 28.24943 5.194529e-08 0.0005127289 8.174490

225328\_at 2.573676 10.006275 27.89953 5.626654e-08 0.0005127289 8.132172

**(b) In your own words, explain the 4 main parameters of the expresso() function, bgcorrect.method, normalize.method, pmcorrect.method and summary.method. Be sure to explain what each of them do, and why they are needed.**

**bgcorrect.method:** This is a background adjustment value. It is used to do background corrections.

**normalize.method:** This is used to normalize Affymetrix Probe Level Data. It is used to rescale values to arrive at values that are relative to a size value.

**pmcorrect.method:** This is a the name of the pm adjustment method. It is used to make adjustments to the pm values.

**summary.method:** This is the method used for the computation of expression values. It is used to do summarization.

**(c) In your own words, briefly describe the advantages and disadvantages of using a Microarray for measuring gene expression.**

**Advantages:** The advantages include the ability to do primary analysis. The data can be verified, and low-quality data can also be removed. Scaling and normalization can be used to make the comparison better. In-depth analysis can be used to filter and cluster relevant data to make better results.

**Disadvantages:** The high cost of doing experiments is a huge problem. The number of probe designs based on sequences of low specificity is another issue. The lack of control as most platforms use probes designed by a manufacturer.

**Q.2 The file RNA\_Seq\_assignment3.txt on CANVAS contains RNA-Seq data for a mouse sequencing experiment. Samples 1 to 4 are from condition A and the rest are from condition B.**

**(a)Perform a differential expression analysis with DESeq2. How many genes are differentially expressed with an adjusted p-value cut off of 0.05? Out of the differentially expressed genes, which have the 5 highest and 5 lowest fold changes? (Just return the gene names).**

library("DESeq2")

count\_data <- read.table("RNA\_Seq\_assignment3.txt")

condition <- c("A", "A", "A", "A", "B", "B","B", "B")

sample <- c("sample1", "sample2","sample3","sample4","sample5","sample6","sample7","sample8")

metadata <- data.frame(sample, condition)

dds <- DESeqDataSetFromMatrix(countData=count\_data,

colData=metadata,

design=~condition)

dds <- DESeq(dds)

res <- results(dds)

resOrdered = res[order(res$pvalue),]

resSig = subset(resOrdered, padj<0.05)

summary(res,alpha=0.05)

**OUTPUT:**

out of 25937 with nonzero total read count

adjusted p-value < 0.05

LFC > 0 (up) : 33, 0.13%

LFC < 0 (down) : 48, 0.19%

outliers [1] : 12, 0.046%

low counts [2] : 9211, 36%

(mean count < 4)

[1] see 'cooksCutoff' argument of ?results

[2] see 'independentFiltering' argument of ?results

select <- rownames(resSig)

select

[1] "ENSMUSG00000043091" "ENSMUSG00000081049" "ENSMUSG00000098178" "ENSMUSG00000098973"

[5] "ENSMUSG00000095041" "ENSMUSG00000030048" "ENSMUSG00000003545" "ENSMUSG00000058126"

[9] "ENSMUSG00000091383" "ENSMUSG00000088609" "ENSMUSG00000045257" "ENSMUSG00000027940"

[13] "ENSMUSG00000041453" "ENSMUSG00000020912" "ENSMUSG00000056536" "ENSMUSG00000038560"

[17] "ENSMUSG00000051497" "ENSMUSG00000036073" "ENSMUSG00000024026" "ENSMUSG00000067017"

[21] "ENSMUSG00000083557" "ENSMUSG00000087881" "ENSMUSG00000097583" "ENSMUSG00000029319"

[25] "ENSMUSG00000058488" "ENSMUSG00000062647" "ENSMUSG00000070583" "ENSMUSG00000028970"

[29] "ENSMUSG00000023236" "ENSMUSG00000043259" "ENSMUSG00000030096" "ENSMUSG00000078490"

[33] "ENSMUSG00000036915" "ENSMUSG00000070357" "ENSMUSG00000038242" "ENSMUSG00000035202"

[37] "ENSMUSG00000032332" "ENSMUSG00000053332" "ENSMUSG00000087692" "ENSMUSG00000029001"

[41] "ENSMUSG00000082274" "ENSMUSG00000070532" "ENSMUSG00000078870" "ENSMUSG00000027435"

[45] "ENSMUSG00000028645" "ENSMUSG00000027737" "ENSMUSG00000039195" "ENSMUSG00000038351"

[49] "ENSMUSG00000021098" "ENSMUSG00000044258" "ENSMUSG00000022247" "ENSMUSG00000055775"

[53] "ENSMUSG00000001506" "ENSMUSG00000099980" "ENSMUSG00000067916" "ENSMUSG00000091400"

[57] "ENSMUSG00000041577" "ENSMUSG00000074280" "ENSMUSG00000002504" "ENSMUSG00000010021"

[61] "ENSMUSG00000097164" "ENSMUSG00000019505" "ENSMUSG00000076617" "ENSMUSG00000078875"

[65] "ENSMUSG00000100890" "ENSMUSG00000041801" "ENSMUSG00000039167" "ENSMUSG00000058441"

[69] "ENSMUSG00000093483" "ENSMUSG00000039106" "ENSMUSG00000035954" "ENSMUSG00000033282"

[73] "ENSMUSG00000036078" "ENSMUSG00000025823" "ENSMUSG00000090223" "ENSMUSG00000078794"

[77] "ENSMUSG00000031431" "ENSMUSG00000028703" "ENSMUSG00000058135" "ENSMUSG00000059775"

[81] "ENSMUSG00000036896"

81 genes have an adjusted p-value cutoff of 0.05.

resOrdered = res[order(res$log2FoldChange),]

resSig = subset(resOrdered, padj<0.05)

bottom5 <- tail(resSig, n=5)

top5 <- head(res, n=5)

merged <- rbind(bottom5, top5)

**(b) Create a heatmap based on the 10 genes listed above. How do the samples cluster? Do the samples from condition A and B form discrete clusters? Also create a heatmap with the 10 genes which have the highest standard deviation. Are any of the genes with the highest standard deviation found to be differentially expressed by DESeq2? Are the clusters the same as the previous heatmap?**

library("pheatmap")

merged1 <- rownames(merged)

rlog1 <- rlog(dds, blind = F)

dataframe <- as.data.frame(colData(dds)[,c("condition")])

colnames(df) = "condition"

heatmap(assay(rlog1)[merged1,])

A picture containing diagram

Description automatically generated

resOrdered1 = res[order(res$lfcSE),]

resSig1 = subset(resOrdered1, padj<0.05)

top10 <- head(resSig1,n=10)

sd\_top10 <- rownames(top10)

rlog2 <- rlog(dds, blind = F)

dataframe1 <- as.data.frame(colData(dds)[,c("condition")])

colnames(dataframe1) = "condition"

heatmap(assay(rlog2)[sd\_top10,])

A picture containing chart

Description automatically generated

**(c) Use biomaRt to retrieve annotation information (i.e. what each gene does) for each of the 10 genes you listed in Q2(a). Insert a table with the annotation information below. (5)**

library(biomaRt)

mart <- useMart(dataset="merged",biomart='ensembl')

**(d) For each of the 10 genes listed in Q2(a) make individual graphs with two boxplots, one showing the expression in condition A and one for condition B. Ensure to label everything appropriately.**

boxplot()

**(e) The output of results() function from the DESeq2 library returns several columns.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| baseMean | log2FoldChange | lfcSE | stat | pvalue | padj |

**In your own words, briefly describe what each one of these is. What method is used for generating the padj? Should you use the p-value or the padj when deciding what genes are significant? Why?**

**baseMean:** This is the mean of the normalized counts of all samples. Used only for getting an estimate of the dispersion of a gene.

**log2FoldChange:** This is the effect size estimate. It informs us of how the gene changed through DPN and the control.

**lfcSE:** This is the stand error estiate for the log2FoldChange. This is to show if the result is truly different to zero.

**stat:** This is the difference in deviance between the reduced model and the full model.

**pvalue:** This shows us the probability that a fold change that is stronger than the obsevered one would be described by a null hypothesis.

**padj:** This is an adjusted value of the pvalue using Benjamini-Hochberg adjustment.

The method for generating the padj is the Benjamini-Hochberg adjustment. This basically calculates the adjusted pvalue which shows us if there is significant false positives.

You should use padj as it can take into account more values. This can show if genes are more significant than others.

**Q.3 No add-on packages are required for this question. It tests your knowledge of basic R functions and how they can be combined to solve a problem. There will be multiple ways to arrive at solutions and an emphasis on automation will receive higher marks. Remember to include all intermediate steps and output. Each part of Q.3 consists of multiple questions so make sure you answer them all. For large output, just include a subset of the data to show that your code worked (head, tail, etc.)**

**NOTE: you will have to use several new functions in Q3. These are hinted at after each question. Familiarize yourself with each question by reading it several times and study the vignettes for specific functions where necessary.**

1. Create a matrix of 1000 rows and 50 columns where each row is the result of using sample() to randomly select, with replacement, 50 letters out of the alphabet (uppercase). You will have to run sample() 1000 times so use an iterative method to create your matrix. Label the rows as row\_1, row\_2, etc. and label the columns as col\_1, col\_2, etc.

r <- 1000

c <- 50

randomLetters <- matrix(data = NA, nrow = r, ncol = c)

new.rownames<-NULL

for (i in 1:r) {

randomLetters[i,] <- sample(LETTERS, c, TRUE)

new.rownames[i]<-paste("row\_",i,sep="")

}

rownames(randomLetters)<-new.rownames

colnames(randomLetters) <- paste("col\_", 1:50, sep = "")

**NOTE: Due to the random nature of letter selection, each person will have a matrix with a different composition of letters. Make sure you understand the sample() function before attempting the question – the vignette should remove any confusion about how 50 letters can be sampled from a 26-letter alphabet.**

**HINT: For row and column labelling, there is a specific function that allows you to join strings, integers, etc. Find it!**

**(B)** Change all letters in the matrix created in part **Q.3 (A)** to lowercase. Change all occurrences of the letter “h” in the matrix to “hi”. Count the number of times “hi” occurs in each column. Count the number of times “hi” occurs in the entire matrix. Extract all occurrences of “hi” from the matrix as a character vector and append “\_X” to each one where X equals the index of the vector (i.e. “hi\_1” “hi\_2” “hi\_3” ….)

lowerRandomLetters <- tolower(randomLetters)

lowerRandomLettersHi <- sub("h","hi",lowerRandomLetters)

columnsCount <- colCounts(lowerRandomLettersHi, value = "hi", dim. = dim(lowerRandomLettersHi))

totalCount<- length(grep("hi", lowerRandomLettersHi))

extractHi <- grep("hi", lowerRandomLettersHi, value = TRUE)

updated\_num <- paste0("\_", 1:2003, sep="")

paste(extractHi,updated\_num, sep = "")

**OUTPUT:**

[1] "hi\_1" "hi\_2" "hi\_3" "hi\_4" "hi\_5" "hi\_6" "hi\_7" "hi\_8"

[9] "hi\_9" "hi\_10" "hi\_11" "hi\_12" "hi\_13" "hi\_14" "hi\_15" "hi\_16"

[17] "hi\_17" "hi\_18" "hi\_19" "hi\_20" "hi\_21" "hi\_22" "hi\_23" "hi\_24"

[25] "hi\_25" "hi\_26" "hi\_27" "hi\_28" "hi\_29" "hi\_30" "hi\_31" "hi\_32"

[33] "hi\_33" "hi\_34" "hi\_35" "hi\_36" "hi\_37" "hi\_38" "hi\_39" "hi\_40"

[41] "hi\_41" "hi\_42" "hi\_43" "hi\_44" "hi\_45" "hi\_46" "hi\_47" "hi\_48"

[49] "hi\_49" "hi\_50" "hi\_51" "hi\_52" "hi\_53" "hi\_54" "hi\_55" "hi\_56"

[57] "hi\_57" "hi\_58" "hi\_59" "hi\_60" "hi\_61" "hi\_62" "hi\_63" "hi\_64"

[65] "hi\_65" "hi\_66" "hi\_67" "hi\_68" "hi\_69" "hi\_70" "hi\_71" "hi\_72"

[73] "hi\_73" "hi\_74" "hi\_75" "hi\_76" "hi\_77" "hi\_78" "hi\_79" "hi\_80"

[81] "hi\_81" "hi\_82" "hi\_83" "hi\_84" "hi\_85" "hi\_86" "hi\_87" "hi\_88"

[89] "hi\_89" "hi\_90" "hi\_91" "hi\_92" "hi\_93" "hi\_94" "hi\_95" "hi\_96"

[97] "hi\_97" "hi\_98" "hi\_99" "hi\_100" "hi\_101" "hi\_102" "hi\_103" "hi\_104"

[105] "hi\_105" "hi\_106" "hi\_107" "hi\_108" "hi\_109" "hi\_110" "hi\_111" "hi\_112"

[113] "hi\_113" "hi\_114" "hi\_115" "hi\_116" "hi\_117" "hi\_118" "hi\_119" "hi\_120"

[121] "hi\_121" "hi\_122" "hi\_123" "hi\_124" "hi\_125" "hi\_126" "hi\_127" "hi\_128"

[129] "hi\_129" "hi\_130" "hi\_131" "hi\_132" "hi\_133" "hi\_134" "hi\_135" "hi\_136"

[137] "hi\_137" "hi\_138" "hi\_139" "hi\_140" "hi\_141" "hi\_142" "hi\_143" "hi\_144"

[145] "hi\_145" "hi\_146" "hi\_147" "hi\_148" "hi\_149" "hi\_150" "hi\_151" "hi\_152"

[153] "hi\_153" "hi\_154" "hi\_155" "hi\_156" "hi\_157" "hi\_158" "hi\_159" "hi\_160"

[161] "hi\_161" "hi\_162" "hi\_163" "hi\_164" "hi\_165" "hi\_166" "hi\_167" "hi\_168"

[169] "hi\_169" "hi\_170" "hi\_171" "hi\_172" "hi\_173" "hi\_174" "hi\_175" "hi\_176"

[177] "hi\_177" "hi\_178" "hi\_179" "hi\_180" "hi\_181" "hi\_182" "hi\_183" "hi\_184"

[185] "hi\_185" "hi\_186" "hi\_187" "hi\_188" "hi\_189" "hi\_190" "hi\_191" "hi\_192"

[193] "hi\_193" "hi\_194" "hi\_195" "hi\_196" "hi\_197" "hi\_198" "hi\_199" "hi\_200"

[201] "hi\_201" "hi\_202" "hi\_203" "hi\_204" "hi\_205" "hi\_206" "hi\_207" "hi\_208"

[209] "hi\_209" "hi\_210" "hi\_211" "hi\_212" "hi\_213" "hi\_214" "hi\_215" "hi\_216"

[217] "hi\_217" "hi\_218" "hi\_219" "hi\_220" "hi\_221" "hi\_222" "hi\_223" "hi\_224"

[225] "hi\_225" "hi\_226" "hi\_227" "hi\_228" "hi\_229" "hi\_230" "hi\_231" "hi\_232"

[233] "hi\_233" "hi\_234" "hi\_235" "hi\_236" "hi\_237" "hi\_238" "hi\_239" "hi\_240"

[241] "hi\_241" "hi\_242" "hi\_243" "hi\_244" "hi\_245" "hi\_246" "hi\_247" "hi\_248"

[249] "hi\_249" "hi\_250" "hi\_251" "hi\_252" "hi\_253" "hi\_254" "hi\_255" "hi\_256"

[257] "hi\_257" "hi\_258" "hi\_259" "hi\_260" "hi\_261" "hi\_262" "hi\_263" "hi\_264"

[265] "hi\_265" "hi\_266" "hi\_267" "hi\_268" "hi\_269" "hi\_270" "hi\_271" "hi\_272"

[273] "hi\_273" "hi\_274" "hi\_275" "hi\_276" "hi\_277" "hi\_278" "hi\_279" "hi\_280"

[281] "hi\_281" "hi\_282" "hi\_283" "hi\_284" "hi\_285" "hi\_286" "hi\_287" "hi\_288"

[289] "hi\_289" "hi\_290" "hi\_291" "hi\_292" "hi\_293" "hi\_294" "hi\_295" "hi\_296"

[297] "hi\_297" "hi\_298" "hi\_299" "hi\_300" "hi\_301" "hi\_302" "hi\_303" "hi\_304"

[305] "hi\_305" "hi\_306" "hi\_307" "hi\_308" "hi\_309" "hi\_310" "hi\_311" "hi\_312"

[313] "hi\_313" "hi\_314" "hi\_315" "hi\_316" "hi\_317" "hi\_318" "hi\_319" "hi\_320"

[321] "hi\_321" "hi\_322" "hi\_323" "hi\_324" "hi\_325" "hi\_326" "hi\_327" "hi\_328"

[329] "hi\_329" "hi\_330" "hi\_331" "hi\_332" "hi\_333" "hi\_334" "hi\_335" "hi\_336"

[337] "hi\_337" "hi\_338" "hi\_339" "hi\_340" "hi\_341" "hi\_342" "hi\_343" "hi\_344"

[345] "hi\_345" "hi\_346" "hi\_347" "hi\_348" "hi\_349" "hi\_350" "hi\_351" "hi\_352"

[353] "hi\_353" "hi\_354" "hi\_355" "hi\_356" "hi\_357" "hi\_358" "hi\_359" "hi\_360"

[361] "hi\_361" "hi\_362" "hi\_363" "hi\_364" "hi\_365" "hi\_366" "hi\_367" "hi\_368"

[369] "hi\_369" "hi\_370" "hi\_371" "hi\_372" "hi\_373" "hi\_374" "hi\_375" "hi\_376"

[377] "hi\_377" "hi\_378" "hi\_379" "hi\_380" "hi\_381" "hi\_382" "hi\_383" "hi\_384"

[385] "hi\_385" "hi\_386" "hi\_387" "hi\_388" "hi\_389" "hi\_390" "hi\_391" "hi\_392"

[393] "hi\_393" "hi\_394" "hi\_395" "hi\_396" "hi\_397" "hi\_398" "hi\_399" "hi\_400"

[401] "hi\_401" "hi\_402" "hi\_403" "hi\_404" "hi\_405" "hi\_406" "hi\_407" "hi\_408"

[409] "hi\_409" "hi\_410" "hi\_411" "hi\_412" "hi\_413" "hi\_414" "hi\_415" "hi\_416"

[417] "hi\_417" "hi\_418" "hi\_419" "hi\_420" "hi\_421" "hi\_422" "hi\_423" "hi\_424"

[425] "hi\_425" "hi\_426" "hi\_427" "hi\_428" "hi\_429" "hi\_430" "hi\_431" "hi\_432"

[433] "hi\_433" "hi\_434" "hi\_435" "hi\_436" "hi\_437" "hi\_438" "hi\_439" "hi\_440"

[441] "hi\_441" "hi\_442" "hi\_443" "hi\_444" "hi\_445" "hi\_446" "hi\_447" "hi\_448"

[449] "hi\_449" "hi\_450" "hi\_451" "hi\_452" "hi\_453" "hi\_454" "hi\_455" "hi\_456"

[457] "hi\_457" "hi\_458" "hi\_459" "hi\_460" "hi\_461" "hi\_462" "hi\_463" "hi\_464"

[465] "hi\_465" "hi\_466" "hi\_467" "hi\_468" "hi\_469" "hi\_470" "hi\_471" "hi\_472"

[473] "hi\_473" "hi\_474" "hi\_475" "hi\_476" "hi\_477" "hi\_478" "hi\_479" "hi\_480"

[481] "hi\_481" "hi\_482" "hi\_483" "hi\_484" "hi\_485" "hi\_486" "hi\_487" "hi\_488"

[489] "hi\_489" "hi\_490" "hi\_491" "hi\_492" "hi\_493" "hi\_494" "hi\_495" "hi\_496"

[497] "hi\_497" "hi\_498" "hi\_499" "hi\_500" "hi\_501" "hi\_502" "hi\_503" "hi\_504"

[505] "hi\_505" "hi\_506" "hi\_507" "hi\_508" "hi\_509" "hi\_510" "hi\_511" "hi\_512"

[513] "hi\_513" "hi\_514" "hi\_515" "hi\_516" "hi\_517" "hi\_518" "hi\_519" "hi\_520"

[521] "hi\_521" "hi\_522" "hi\_523" "hi\_524" "hi\_525" "hi\_526" "hi\_527" "hi\_528"

[529] "hi\_529" "hi\_530" "hi\_531" "hi\_532" "hi\_533" "hi\_534" "hi\_535" "hi\_536"

[537] "hi\_537" "hi\_538" "hi\_539" "hi\_540" "hi\_541" "hi\_542" "hi\_543" "hi\_544"

[545] "hi\_545" "hi\_546" "hi\_547" "hi\_548" "hi\_549" "hi\_550" "hi\_551" "hi\_552"

[553] "hi\_553" "hi\_554" "hi\_555" "hi\_556" "hi\_557" "hi\_558" "hi\_559" "hi\_560"

[561] "hi\_561" "hi\_562" "hi\_563" "hi\_564" "hi\_565" "hi\_566" "hi\_567" "hi\_568"

[569] "hi\_569" "hi\_570" "hi\_571" "hi\_572" "hi\_573" "hi\_574" "hi\_575" "hi\_576"

[577] "hi\_577" "hi\_578" "hi\_579" "hi\_580" "hi\_581" "hi\_582" "hi\_583" "hi\_584"

[585] "hi\_585" "hi\_586" "hi\_587" "hi\_588" "hi\_589" "hi\_590" "hi\_591" "hi\_592"

[593] "hi\_593" "hi\_594" "hi\_595" "hi\_596" "hi\_597" "hi\_598" "hi\_599" "hi\_600"

[601] "hi\_601" "hi\_602" "hi\_603" "hi\_604" "hi\_605" "hi\_606" "hi\_607" "hi\_608"

[609] "hi\_609" "hi\_610" "hi\_611" "hi\_612" "hi\_613" "hi\_614" "hi\_615" "hi\_616"

[617] "hi\_617" "hi\_618" "hi\_619" "hi\_620" "hi\_621" "hi\_622" "hi\_623" "hi\_624"

[625] "hi\_625" "hi\_626" "hi\_627" "hi\_628" "hi\_629" "hi\_630" "hi\_631" "hi\_632"

[633] "hi\_633" "hi\_634" "hi\_635" "hi\_636" "hi\_637" "hi\_638" "hi\_639" "hi\_640"

[641] "hi\_641" "hi\_642" "hi\_643" "hi\_644" "hi\_645" "hi\_646" "hi\_647" "hi\_648"

[649] "hi\_649" "hi\_650" "hi\_651" "hi\_652" "hi\_653" "hi\_654" "hi\_655" "hi\_656"

[657] "hi\_657" "hi\_658" "hi\_659" "hi\_660" "hi\_661" "hi\_662" "hi\_663" "hi\_664"

[665] "hi\_665" "hi\_666" "hi\_667" "hi\_668" "hi\_669" "hi\_670" "hi\_671" "hi\_672"

[673] "hi\_673" "hi\_674" "hi\_675" "hi\_676" "hi\_677" "hi\_678" "hi\_679" "hi\_680"

[681] "hi\_681" "hi\_682" "hi\_683" "hi\_684" "hi\_685" "hi\_686" "hi\_687" "hi\_688"

[689] "hi\_689" "hi\_690" "hi\_691" "hi\_692" "hi\_693" "hi\_694" "hi\_695" "hi\_696"

[697] "hi\_697" "hi\_698" "hi\_699" "hi\_700" "hi\_701" "hi\_702" "hi\_703" "hi\_704"

[705] "hi\_705" "hi\_706" "hi\_707" "hi\_708" "hi\_709" "hi\_710" "hi\_711" "hi\_712"

[713] "hi\_713" "hi\_714" "hi\_715" "hi\_716" "hi\_717" "hi\_718" "hi\_719" "hi\_720"

[721] "hi\_721" "hi\_722" "hi\_723" "hi\_724" "hi\_725" "hi\_726" "hi\_727" "hi\_728"

[729] "hi\_729" "hi\_730" "hi\_731" "hi\_732" "hi\_733" "hi\_734" "hi\_735" "hi\_736"

[737] "hi\_737" "hi\_738" "hi\_739" "hi\_740" "hi\_741" "hi\_742" "hi\_743" "hi\_744"

[745] "hi\_745" "hi\_746" "hi\_747" "hi\_748" "hi\_749" "hi\_750" "hi\_751" "hi\_752"

[753] "hi\_753" "hi\_754" "hi\_755" "hi\_756" "hi\_757" "hi\_758" "hi\_759" "hi\_760"

[761] "hi\_761" "hi\_762" "hi\_763" "hi\_764" "hi\_765" "hi\_766" "hi\_767" "hi\_768"

[769] "hi\_769" "hi\_770" "hi\_771" "hi\_772" "hi\_773" "hi\_774" "hi\_775" "hi\_776"

[777] "hi\_777" "hi\_778" "hi\_779" "hi\_780" "hi\_781" "hi\_782" "hi\_783" "hi\_784"

[785] "hi\_785" "hi\_786" "hi\_787" "hi\_788" "hi\_789" "hi\_790" "hi\_791" "hi\_792"

[793] "hi\_793" "hi\_794" "hi\_795" "hi\_796" "hi\_797" "hi\_798" "hi\_799" "hi\_800"

[801] "hi\_801" "hi\_802" "hi\_803" "hi\_804" "hi\_805" "hi\_806" "hi\_807" "hi\_808"

[809] "hi\_809" "hi\_810" "hi\_811" "hi\_812" "hi\_813" "hi\_814" "hi\_815" "hi\_816"

[817] "hi\_817" "hi\_818" "hi\_819" "hi\_820" "hi\_821" "hi\_822" "hi\_823" "hi\_824"

[825] "hi\_825" "hi\_826" "hi\_827" "hi\_828" "hi\_829" "hi\_830" "hi\_831" "hi\_832"

[833] "hi\_833" "hi\_834" "hi\_835" "hi\_836" "hi\_837" "hi\_838" "hi\_839" "hi\_840"

[841] "hi\_841" "hi\_842" "hi\_843" "hi\_844" "hi\_845" "hi\_846" "hi\_847" "hi\_848"

[849] "hi\_849" "hi\_850" "hi\_851" "hi\_852" "hi\_853" "hi\_854" "hi\_855" "hi\_856"

[857] "hi\_857" "hi\_858" "hi\_859" "hi\_860" "hi\_861" "hi\_862" "hi\_863" "hi\_864"

[865] "hi\_865" "hi\_866" "hi\_867" "hi\_868" "hi\_869" "hi\_870" "hi\_871" "hi\_872"

[873] "hi\_873" "hi\_874" "hi\_875" "hi\_876" "hi\_877" "hi\_878" "hi\_879" "hi\_880"

[881] "hi\_881" "hi\_882" "hi\_883" "hi\_884" "hi\_885" "hi\_886" "hi\_887" "hi\_888"

[889] "hi\_889" "hi\_890" "hi\_891" "hi\_892" "hi\_893" "hi\_894" "hi\_895" "hi\_896"

[897] "hi\_897" "hi\_898" "hi\_899" "hi\_900" "hi\_901" "hi\_902" "hi\_903" "hi\_904"

[905] "hi\_905" "hi\_906" "hi\_907" "hi\_908" "hi\_909" "hi\_910" "hi\_911" "hi\_912"

[913] "hi\_913" "hi\_914" "hi\_915" "hi\_916" "hi\_917" "hi\_918" "hi\_919" "hi\_920"

[921] "hi\_921" "hi\_922" "hi\_923" "hi\_924" "hi\_925" "hi\_926" "hi\_927" "hi\_928"

[929] "hi\_929" "hi\_930" "hi\_931" "hi\_932" "hi\_933" "hi\_934" "hi\_935" "hi\_936"

[937] "hi\_937" "hi\_938" "hi\_939" "hi\_940" "hi\_941" "hi\_942" "hi\_943" "hi\_944"

[945] "hi\_945" "hi\_946" "hi\_947" "hi\_948" "hi\_949" "hi\_950" "hi\_951" "hi\_952"

[953] "hi\_953" "hi\_954" "hi\_955" "hi\_956" "hi\_957" "hi\_958" "hi\_959" "hi\_960"

[961] "hi\_961" "hi\_962" "hi\_963" "hi\_964" "hi\_965" "hi\_966" "hi\_967" "hi\_968"

[969] "hi\_969" "hi\_970" "hi\_971" "hi\_972" "hi\_973" "hi\_974" "hi\_975" "hi\_976"

[977] "hi\_977" "hi\_978" "hi\_979" "hi\_980" "hi\_981" "hi\_982" "hi\_983" "hi\_984"

[985] "hi\_985" "hi\_986" "hi\_987" "hi\_988" "hi\_989" "hi\_990" "hi\_991" "hi\_992"

[993] "hi\_993" "hi\_994" "hi\_995" "hi\_996" "hi\_997" "hi\_998" "hi\_999" "hi\_1000"

[ reached getOption("max.print") -- omitted 1003 entries ]

**(C)** Create a list of length 20where each item in the list is the result of using sample() to randomly select, with replacement, 50 integers out of the numbers 1 to 100. You will have to run sample() 20 times so use an iterative method to create your list. Add names to the list where each name is “mean\_XX” with XX representing the mean of each integer vector (e.g. mean\_55.3). Output a summary of each integer vector in the list with names as rows and the six statistics (min., quartile 1, median, mean, quartile 3 and max.) as columns.

c <- 20

randomVector <- list()

for (i in 1:c) {

randomVector[[i]]<- sample(1:100, 50, TRUE)

}

mean1 <- sapply(randomVector, mean)

updated\_mean <- paste0("mean\_", mean1, sep="")

names(randomVector)<-updated\_mean

**OUTPUT:**

$mean\_50.38

[1] 99 87 31 41 50 57 45 91 44 99 52 33 27 10 78 59 14 69 22 87 35 71 41 38 42 1 87 47

[29] 88 33 9 19 66 63 22 55 46 84 97 32 9 89 41 99 8 80 32 6 77 7

$mean\_47.82

[1] 61 99 80 60 46 71 83 37 91 30 19 49 33 63 15 29 60 29 26 95 36 72 33 99 25 39 42 7

[29] 12 74 83 9 66 9 63 77 41 36 26 12 23 10 28 89 78 87 9 31 33 66

$mean\_50.28

[1] 70 12 33 32 99 79 25 3 73 67 81 52 72 4 96 14 96 56 41 31 38 14 25 63 11 40 86 17

[29] 82 21 86 33 72 68 91 15 41 60 1 50 75 50 97 78 92 21 61 41 34 15

$mean\_46.68

[1] 62 70 72 15 18 38 9 16 51 90 51 64 87 59 14 6 3 52 30 71 7 42 17 11 14 33 92 32

[29] 5 71 47 29 39 7 54 95 99 8 93 90 56 81 30 62 94 75 4 81 52 36

$mean\_49.22

[1] 4 77 39 43 35 91 81 55 4 18 29 28 71 50 43 70 32 58 48 95 44 26 71 61 71 86 93 6

[29] 92 24 46 36 63 28 70 2 38 45 37 25 73 11 8 96 98 91 34 66 9 40

$mean\_53.24

[1] 64 73 22 7 11 31 75 50 12 93 99 88 4 89 61 59 67 82 36 42 8 48 51 99 11 37 82 8

[29] 16 65 4 45 63 76 30 70 45 81 94 69 53 76 92 85 18 37 66 96 42 30

$mean\_50.36

[1] 66 92 7 18 11 92 91 80 78 28 7 85 32 4 73 1 3 69 99 20 86 67 21 8 85 24 92 62

[29] 52 58 28 61 86 43 88 80 59 91 22 62 56 10 92 17 19 19 60 31 63 20

$mean\_48.5

[1] 54 27 81 85 34 54 43 9 63 20 30 22 43 36 22 88 30 48 75 49 67 10 72 65 36 32 83 76

[29] 33 36 75 54 8 5 71 28 68 25 12 74 43 91 25 34 49 76 91 89 66 18

$mean\_53.94

[1] 3 50 63 65 70 97 60 51 13 90 20 30 66 56 55 1 81 100 100 47 84

[22] 7 91 16 96 17 73 80 12 8 82 73 30 80 83 90 79 79 42 23 89 56

[43] 10 19 4 50 72 28 88 18

$mean\_55.26

[1] 61 65 29 78 94 2 89 48 25 84 13 62 91 88 15 96 62 46 70 19 21 50 91 99 16 73 63 88

[29] 30 29 12 32 6 39 90 12 93 71 89 30 76 55 54 98 93 82 40 63 28 3

$mean\_48.98

[1] 40 72 62 90 62 50 5 89 1 41 8 33 14 15 3 82 77 77 39 20 92 28 59 30 72 99 60 15

[29] 89 62 88 93 16 98 23 37 14 49 65 60 58 20 86 92 38 1 63 30 28 4

$mean\_50.4

[1] 63 13 49 74 61 51 13 86 33 89 21 70 23 20 87 31 40 52 49 76 97

[22] 16 11 40 18 33 86 76 39 53 7 23 35 39 51 16 69 63 55 53 38 48

[43] 38 95 66 66 59 95 34 100

$mean\_51.04

[1] 4 94 96 18 2 58 25 48 88 64 83 33 34 79 88 30 3 70 85 56 12 5 15 34 81 96 98 28

[29] 62 91 77 81 54 89 71 5 94 43 13 5 38 31 29 11 58 20 54 24 87 88

$mean\_44.9

[1] 33 6 32 26 24 19 46 50 93 72 68 49 73 38 21 69 95 13 85 57 18 37 12 72 81 3 22 85

[29] 92 34 12 50 19 57 28 6 67 68 27 10 44 60 18 89 57 53 86 16 26 27

$mean\_47.96

[1] 62 43 82 33 9 7 15 72 78 39 74 12 13 8 45 60 7 5 20 18 41 81 53 9 77 57 53 1

[29] 6 71 3 40 51 93 40 99 51 48 22 68 55 76 85 96 14 97 40 71 99 99

$mean\_50.7

[1] 9 18 34 69 80 29 83 82 40 52 36 60 20 84 98 16 69 29 14 68 20

[22] 26 89 63 84 45 76 83 4 4 10 85 83 75 15 7 40 39 78 71 8 83

[43] 79 100 39 5 43 18 91 82

$mean\_51.34

[1] 38 77 47 38 82 46 25 74 81 85 60 30 26 16 92 17 62 53 87 85 12 57 94 46 93 44 60 38

[29] 23 50 14 64 21 9 36 92 74 48 52 3 30 92 85 48 5 61 13 61 56 65

$mean\_52.88

[1] 26 74 40 4 91 32 31 85 18 69 42 44 12 58 22 79 74 70 97 23 94

[22] 5 76 12 53 17 46 71 75 98 90 61 78 11 18 15 75 41 69 59 87 80

[43] 58 44 100 30 45 27 87 31

$mean\_53.86

[1] 78 39 73 62 26 7 7 83 86 99 62 6 5 4 66 62 5 71 50 7 68 3 96 89 75 8 62 66

[29] 41 83 3 69 93 67 51 68 75 82 39 97 16 53 50 80 41 82 85 29 37 87

$mean\_52.84

[1] 37 79 77 12 69 84 31 71 86 12 2 84 81 28 67 13 21 28 73 95 56 29 63 3 96 69 83 62

[29] 17 35 55 45 63 86 42 37 17 39 40 95 58 13 15 80 98 84 63 92 31 26

Matrix\_x <- matrix(unlist(randomVector), ncol = 50, byrow = TRUE) #convert list to matrix

rownames(Matrix\_x)<-updated\_mean #give rows names

summary(Matrix\_x) # summary

**OUTPUT:**

> summary(Matrix\_x)

V1 V2 V3 V4 V5

Min. : 12.00 Min. :13.00 Min. :16.00 Min. : 3.00 Min. : 3.00

1st Qu.: 33.50 1st Qu.:35.00 1st Qu.:34.75 1st Qu.:25.25 1st Qu.:35.25

Median : 52.50 Median :43.50 Median :56.00 Median :63.50 Median :58.00

Mean : 53.80 Mean :47.75 Mean :55.05 Mean :55.15 Mean :56.30

3rd Qu.: 75.25 3rd Qu.:61.00 3rd Qu.:78.25 3rd Qu.:81.00 3rd Qu.:80.00

Max. :100.00 Max. :99.00 Max. :98.00 Max. :98.00 Max. :97.00

V6 V7 V8 V9 V10

Min. : 5.00 Min. : 10.00 Min. : 5.00 Min. : 2.00 Min. : 7.00

1st Qu.:12.75 1st Qu.: 42.25 1st Qu.:21.50 1st Qu.:14.75 1st Qu.:17.75

Median :29.00 Median : 67.00 Median :34.00 Median :40.50 Median :40.00

Mean :43.25 Mean : 59.75 Mean :40.90 Mean :44.60 Mean :46.50

3rd Qu.:72.00 3rd Qu.: 79.50 3rd Qu.:55.75 3rd Qu.:74.25 3rd Qu.:77.00

Max. :98.00 Max. :100.00 Max. :94.00 Max. :95.00 Max. :98.00

V11 V12 V13 V14 V15

Min. : 3.00 Min. : 1.00 Min. : 2.00 Min. : 12.00 Min. : 4.00

1st Qu.:21.25 1st Qu.:19.25 1st Qu.:26.75 1st Qu.: 25.75 1st Qu.: 21.50

Median :53.00 Median :49.00 Median :52.00 Median : 51.00 Median : 54.50

Mean :51.00 Mean :43.70 Mean :50.35 Mean : 54.40 Mean : 52.15

3rd Qu.:79.50 3rd Qu.:62.00 3rd Qu.:77.25 3rd Qu.: 82.00 3rd Qu.: 78.25

Max. :98.00 Max. :86.00 Max. :93.00 Max. :100.00 Max. :100.00

V16 V17 V18 V19 V20

Min. : 7.00 Min. : 4.00 Min. : 1.00 Min. : 3.00 Min. : 1.00

1st Qu.:29.25 1st Qu.: 31.25 1st Qu.:34.00 1st Qu.: 30.00 1st Qu.:23.75

Median :65.00 Median : 55.50 Median :53.50 Median : 65.50 Median :36.00

Mean :58.45 Mean : 53.30 Mean :54.65 Mean : 60.25 Mean :43.70

3rd Qu.:86.75 3rd Qu.: 77.75 3rd Qu.:77.75 3rd Qu.: 86.25 3rd Qu.:63.25

Max. :99.00 Max. :100.00 Max. :97.00 Max. :100.00 Max. :97.00

V21 V22 V23 V24 V25

Min. :13.00 Min. : 1.00 Min. : 5.0 Min. : 8.00 Min. : 6.00

1st Qu.:31.75 1st Qu.:21.00 1st Qu.: 21.5 1st Qu.:38.75 1st Qu.:30.50

Median :47.50 Median :65.00 Median : 53.5 Median :69.00 Median :48.00

Mean :51.45 Mean :53.55 Mean : 49.4 Mean :62.70 Mean :50.15

3rd Qu.:72.50 3rd Qu.:80.50 3rd Qu.: 69.0 3rd Qu.:84.75 3rd Qu.:71.75

Max. :98.00 Max. :95.00 Max. :100.0 Max. :97.00 Max. :92.00

V26 V27 V28 V29 V30

Min. : 7.00 Min. : 1.00 Min. : 1.00 Min. : 7.00 Min. :15.00

1st Qu.:20.75 1st Qu.: 21.75 1st Qu.:40.25 1st Qu.:32.25 1st Qu.:47.50

Median :39.00 Median : 68.50 Median :57.50 Median :53.50 Median :59.50

Mean :45.80 Mean : 58.10 Mean :55.70 Mean :50.90 Mean :57.65

3rd Qu.:72.00 3rd Qu.: 79.75 3rd Qu.:81.25 3rd Qu.:72.00 3rd Qu.:72.00

Max. :93.00 Max. :100.00 Max. :98.00 Max. :93.00 Max. :98.00

V31 V32 V33 V34 V35

Min. : 3.00 Min. : 9.00 Min. : 1.00 Min. : 8.00 Min. : 5.00

1st Qu.:19.25 1st Qu.: 14.00 1st Qu.:20.00 1st Qu.: 20.50 1st Qu.:32.75

Median :38.50 Median : 35.50 Median :36.50 Median : 46.00 Median :68.00

Mean :47.90 Mean : 39.75 Mean :46.85 Mean : 49.45 Mean :60.80

3rd Qu.:69.25 3rd Qu.: 55.00 3rd Qu.:69.75 3rd Qu.: 70.50 3rd Qu.:86.50

Max. :98.00 Max. :100.00 Max. :97.00 Max. :100.00 Max. :99.00

V36 V37 V38 V39 V40

Min. : 2.00 Min. : 1.00 Min. :10.00 Min. : 2.00 Min. : 7.0

1st Qu.:20.75 1st Qu.:31.25 1st Qu.:24.75 1st Qu.:20.25 1st Qu.:23.5

Median :45.50 Median :41.00 Median :39.00 Median :46.50 Median :40.0

Mean :45.85 Mean :47.60 Mean :39.60 Mean :47.20 Mean :42.9

3rd Qu.:66.50 3rd Qu.:64.75 3rd Qu.:46.50 3rd Qu.:68.25 3rd Qu.:62.5

Max. :85.00 Max. :99.00 Max. :90.00 Max. :97.00 Max. :97.0

V41 V42 V43 V44 V45

Min. : 4.00 Min. : 1.00 Min. : 2.00 Min. : 9.00 Min. :15.00

1st Qu.:24.25 1st Qu.: 13.25 1st Qu.:15.75 1st Qu.:32.75 1st Qu.:26.50

Median :46.50 Median : 36.00 Median :44.50 Median :48.00 Median :40.50

Mean :47.25 Mean : 44.00 Mean :44.50 Mean :46.35 Mean :43.85

3rd Qu.:70.25 3rd Qu.: 69.25 3rd Qu.:69.25 3rd Qu.:60.00 3rd Qu.:55.00

Max. :93.00 Max. :100.00 Max. :89.00 Max. :99.00 Max. :97.00

V46 V47 V48 V49 V50

Min. : 2.00 Min. : 3.00 Min. : 4.00 Min. : 3.0 Min. : 3.00

1st Qu.: 32.00 1st Qu.:21.00 1st Qu.:14.25 1st Qu.:25.0 1st Qu.:20.25

Median : 38.00 Median :49.00 Median :35.00 Median :40.5 Median :30.00

Mean : 49.05 Mean :48.95 Mean :42.70 Mean :44.0 Mean :42.35

3rd Qu.: 75.00 3rd Qu.:74.00 3rd Qu.:72.00 3rd Qu.:60.0 3rd Qu.:68.00

Max. :100.00 Max. :99.00 Max. :90.00 Max. :99.0 Max. :98.00

**(D)** Write a while-loop to iterate through the integers 1 to 10000 and return a numeric vector with only odd numbers. Give the index of the number 9855 in this vector. Substitute ALL occurrences of 9 with 8 and output the results as a numeric vector.

vector = 0

i=0

while (i < 10000) {

if (i %% 2 ==1)

{vector <- c(vector, i)}

i= i+1

}

vector <- vector[-1]

match(c(9855), vector)

**OUTPUT:**

[1] 4928

new\_vector <- gsub("9", "8", vector)

**OUTPUT:**

[1] "1" "3" "5" "7" "8" "11" "13" "15" "17" "18" "21"

[12] "23" "25" "27" "28" "31" "33" "35" "37" "38" "41" "43"

[23] "45" "47" "48" "51" "53" "55" "57" "58" "61" "63" "65"

[34] "67" "68" "71" "73" "75" "77" "78" "81" "83" "85" "87"

[45] "88" "81" "83" "85" "87" "88" "101" "103" "105" "107" "108"

[56] "111" "113" "115" "117" "118" "121" "123" "125" "127" "128" "131"

[67] "133" "135" "137" "138" "141" "143" "145" "147" "148" "151" "153"

[78] "155" "157" "158" "161" "163" "165" "167" "168" "171" "173" "175"

[89] "177" "178" "181" "183" "185" "187" "188" "181" "183" "185" "187"

[100] "188" "201" "203" "205" "207" "208" "211" "213" "215" "217" "218"

[111] "221" "223" "225" "227" "228" "231" "233" "235" "237" "238" "241"

[122] "243" "245" "247" "248" "251" "253" "255" "257" "258" "261" "263"

[133] "265" "267" "268" "271" "273" "275" "277" "278" "281" "283" "285"

[144] "287" "288" "281" "283" "285" "287" "288" "301" "303" "305" "307"

[155] "308" "311" "313" "315" "317" "318" "321" "323" "325" "327" "328"

[166] "331" "333" "335" "337" "338" "341" "343" "345" "347" "348" "351"

[177] "353" "355" "357" "358" "361" "363" "365" "367" "368" "371" "373"

[188] "375" "377" "378" "381" "383" "385" "387" "388" "381" "383" "385"

[199] "387" "388" "401" "403" "405" "407" "408" "411" "413" "415" "417"

[210] "418" "421" "423" "425" "427" "428" "431" "433" "435" "437" "438"

[221] "441" "443" "445" "447" "448" "451" "453" "455" "457" "458" "461"

[232] "463" "465" "467" "468" "471" "473" "475" "477" "478" "481" "483"

[243] "485" "487" "488" "481" "483" "485" "487" "488" "501" "503" "505"

[254] "507" "508" "511" "513" "515" "517" "518" "521" "523" "525" "527"

[265] "528" "531" "533" "535" "537" "538" "541" "543" "545" "547" "548"

[276] "551" "553" "555" "557" "558" "561" "563" "565" "567" "568" "571"

[287] "573" "575" "577" "578" "581" "583" "585" "587" "588" "581" "583"

[298] "585" "587" "588" "601" "603" "605" "607" "608" "611" "613" "615"

[309] "617" "618" "621" "623" "625" "627" "628" "631" "633" "635" "637"

[320] "638" "641" "643" "645" "647" "648" "651" "653" "655" "657" "658"

[331] "661" "663" "665" "667" "668" "671" "673" "675" "677" "678" "681"

[342] "683" "685" "687" "688" "681" "683" "685" "687" "688" "701" "703"

[353] "705" "707" "708" "711" "713" "715" "717" "718" "721" "723" "725"

[364] "727" "728" "731" "733" "735" "737" "738" "741" "743" "745" "747"

[375] "748" "751" "753" "755" "757" "758" "761" "763" "765" "767" "768"

[386] "771" "773" "775" "777" "778" "781" "783" "785" "787" "788" "781"

[397] "783" "785" "787" "788" "801" "803" "805" "807" "808" "811" "813"

[408] "815" "817" "818" "821" "823" "825" "827" "828" "831" "833" "835"

[419] "837" "838" "841" "843" "845" "847" "848" "851" "853" "855" "857"

[430] "858" "861" "863" "865" "867" "868" "871" "873" "875" "877" "878"

[441] "881" "883" "885" "887" "888" "881" "883" "885" "887" "888" "801"

[452] "803" "805" "807" "808" "811" "813" "815" "817" "818" "821" "823"

[463] "825" "827" "828" "831" "833" "835" "837" "838" "841" "843" "845"

[474] "847" "848" "851" "853" "855" "857" "858" "861" "863" "865" "867"

[485] "868" "871" "873" "875" "877" "878" "881" "883" "885" "887" "888"

[496] "881" "883" "885" "887" "888" "1001" "1003" "1005" "1007" "1008" "1011"

[507] "1013" "1015" "1017" "1018" "1021" "1023" "1025" "1027" "1028" "1031" "1033"

[518] "1035" "1037" "1038" "1041" "1043" "1045" "1047" "1048" "1051" "1053" "1055"

[529] "1057" "1058" "1061" "1063" "1065" "1067" "1068" "1071" "1073" "1075" "1077"

[540] "1078" "1081" "1083" "1085" "1087" "1088" "1081" "1083" "1085" "1087" "1088"

[551] "1101" "1103" "1105" "1107" "1108" "1111" "1113" "1115" "1117" "1118" "1121"

[562] "1123" "1125" "1127" "1128" "1131" "1133" "1135" "1137" "1138" "1141" "1143"

[573] "1145" "1147" "1148" "1151" "1153" "1155" "1157" "1158" "1161" "1163" "1165"

[584] "1167" "1168" "1171" "1173" "1175" "1177" "1178" "1181" "1183" "1185" "1187"

[595] "1188" "1181" "1183" "1185" "1187" "1188" "1201" "1203" "1205" "1207" "1208"

[606] "1211" "1213" "1215" "1217" "1218" "1221" "1223" "1225" "1227" "1228" "1231"

[617] "1233" "1235" "1237" "1238" "1241" "1243" "1245" "1247" "1248" "1251" "1253"

[628] "1255" "1257" "1258" "1261" "1263" "1265" "1267" "1268" "1271" "1273" "1275"

[639] "1277" "1278" "1281" "1283" "1285" "1287" "1288" "1281" "1283" "1285" "1287"

[650] "1288" "1301" "1303" "1305" "1307" "1308" "1311" "1313" "1315" "1317" "1318"

[661] "1321" "1323" "1325" "1327" "1328" "1331" "1333" "1335" "1337" "1338" "1341"

[672] "1343" "1345" "1347" "1348" "1351" "1353" "1355" "1357" "1358" "1361" "1363"

[683] "1365" "1367" "1368" "1371" "1373" "1375" "1377" "1378" "1381" "1383" "1385"

[694] "1387" "1388" "1381" "1383" "1385" "1387" "1388" "1401" "1403" "1405" "1407"

[705] "1408" "1411" "1413" "1415" "1417" "1418" "1421" "1423" "1425" "1427" "1428"

[716] "1431" "1433" "1435" "1437" "1438" "1441" "1443" "1445" "1447" "1448" "1451"

[727] "1453" "1455" "1457" "1458" "1461" "1463" "1465" "1467" "1468" "1471" "1473"

[738] "1475" "1477" "1478" "1481" "1483" "1485" "1487" "1488" "1481" "1483" "1485"

[749] "1487" "1488" "1501" "1503" "1505" "1507" "1508" "1511" "1513" "1515" "1517"

[760] "1518" "1521" "1523" "1525" "1527" "1528" "1531" "1533" "1535" "1537" "1538"

[771] "1541" "1543" "1545" "1547" "1548" "1551" "1553" "1555" "1557" "1558" "1561"

[782] "1563" "1565" "1567" "1568" "1571" "1573" "1575" "1577" "1578" "1581" "1583"

[793] "1585" "1587" "1588" "1581" "1583" "1585" "1587" "1588" "1601" "1603" "1605"

[804] "1607" "1608" "1611" "1613" "1615" "1617" "1618" "1621" "1623" "1625" "1627"

[815] "1628" "1631" "1633" "1635" "1637" "1638" "1641" "1643" "1645" "1647" "1648"

[826] "1651" "1653" "1655" "1657" "1658" "1661" "1663" "1665" "1667" "1668" "1671"

[837] "1673" "1675" "1677" "1678" "1681" "1683" "1685" "1687" "1688" "1681" "1683"

[848] "1685" "1687" "1688" "1701" "1703" "1705" "1707" "1708" "1711" "1713" "1715"

[859] "1717" "1718" "1721" "1723" "1725" "1727" "1728" "1731" "1733" "1735" "1737"

[870] "1738" "1741" "1743" "1745" "1747" "1748" "1751" "1753" "1755" "1757" "1758"

[881] "1761" "1763" "1765" "1767" "1768" "1771" "1773" "1775" "1777" "1778" "1781"

[892] "1783" "1785" "1787" "1788" "1781" "1783" "1785" "1787" "1788" "1801" "1803"

[903] "1805" "1807" "1808" "1811" "1813" "1815" "1817" "1818" "1821" "1823" "1825"

[914] "1827" "1828" "1831" "1833" "1835" "1837" "1838" "1841" "1843" "1845" "1847"

[925] "1848" "1851" "1853" "1855" "1857" "1858" "1861" "1863" "1865" "1867" "1868"

[936] "1871" "1873" "1875" "1877" "1878" "1881" "1883" "1885" "1887" "1888" "1881"

[947] "1883" "1885" "1887" "1888" "1801" "1803" "1805" "1807" "1808" "1811" "1813"

[958] "1815" "1817" "1818" "1821" "1823" "1825" "1827" "1828" "1831" "1833" "1835"

[969] "1837" "1838" "1841" "1843" "1845" "1847" "1848" "1851" "1853" "1855" "1857"

[980] "1858" "1861" "1863" "1865" "1867" "1868" "1871" "1873" "1875" "1877" "1878"

[991] "1881" "1883" "1885" "1887" "1888" "1881" "1883" "1885" "1887" "1888"

[ reached getOption("max.print") -- omitted 4000 entries ]

**NOTE: we never covered while-loops in the Practicals, but they are similar to for-loops. Make sure you understand how they work before attempting the question.**

**HINT: an if-statement can be used within the while-loop.**

**HINT: there are specific functions for substituting in R.**