Homework 3

Jia-Ru Chung

October 12, 2017

**Note:**  
All answers have to be obtained using R code!

### Question 1

Let's look at the *RESULTS* object similar to the one we made at the end of Lecture6, and create an object called RES1 containing just the first element of RESULTS.

# your R code  
load("Predict\_RESULTS\_50iter.Rdata")  
class(RESULTS)

## [1] "list"

length(RESULTS)

## [1] 50

RES1 = RESULTS[[2]]

**1.1**  
1. Use *str()* to show the structure of *RES1* dataset. 2. Create the object *Preds1* containing just the Test Predictions 'data.frame' from *RES1* (check your answer using class(Preds1)) 2. Create the new column *PatNum* - consisting of the last number of *PatID* variable. Hint: use *gsub()* function (see Help for how to use *gsub*).  
3. Create new column *PredBinary*, if model predicts probability >= 0.5 for a patient being a responder (R) then *PredBinary* = R (< 0.5 = NR). {10 pts}

str(RES1)

## List of 3  
## $ TrainNames: chr [1:28] "Patient\_2" "Patient\_3" "Patient\_5" "Patient\_6" ...  
## $ TestNames : chr [1:12] "Patient\_1" "Patient\_4" "Patient\_8" "Patient\_14" ...  
## $ TestPreds :'data.frame': 12 obs. of 4 variables:  
## ..$ NR : num [1:12] 0.323 0.719 0.629 0.3 0.435 ...  
## ..$ R : num [1:12] 0.677 0.281 0.371 0.7 0.565 ...  
## ..$ PatID : chr [1:12] "Patient\_1" "Patient\_4" "Patient\_8" "Patient\_14" ...  
## ..$ Y.Test: chr [1:12] "R" "R" "R" "R" ...

Preds1 <- data.frame(RES1$TestPreds)  
class(Preds1)

## [1] "data.frame"

Preds1$PatNum <- gsub("Patient\_", "", Preds1$PatID)  
  
  
Preds1$PredBinary <- ""  
Preds1[(Preds1$R >= 0.5), "PredBinary"] <- "R"  
Preds1[(Preds1$R < 0.5), "PredBinary"] <- "NR"  
  
  
Preds1

## NR R PatID Y.Test PatNum PredBinary  
## 1 0.3228732 0.67712677 Patient\_1 R 1 R  
## 2 0.7188428 0.28115717 Patient\_4 R 4 NR  
## 3 0.6285662 0.37143382 Patient\_8 R 8 NR  
## 4 0.2995221 0.70047786 Patient\_14 R 14 R  
## 5 0.4348569 0.56514306 Patient\_15 R 15 R  
## 6 0.5131471 0.48685290 Patient\_18 R 18 NR  
## 7 0.8790314 0.12096865 Patient\_22 NR 22 NR  
## 8 0.1322514 0.86774856 Patient\_31 NR 31 R  
## 9 0.6772474 0.32275255 Patient\_33 NR 33 NR  
## 10 0.9346382 0.06536184 Patient\_34 NR 34 NR  
## 11 0.1278266 0.87217338 Patient\_37 NR 37 R  
## 12 0.2996724 0.70032761 Patient\_40 NR 40 R

**1.2** Using your code from **1.1**, write a function called extractPreds that can take RES1 as its argument. The function should return an object called Preds that contains the same columns as Preds1 you made in **1.1**. Exectute this function on RES1 to demonstrate it works correctly

extractPreds <- function(x) {  
 x$PatNum <- gsub("Patient\_", "", x$PatID)  
 x$PredBinary <- ""  
 x[(x$R >= 0.5), "PredBinary"] <- "R"  
 x[(x$R < 0.5), "PredBinary"] <- "NR"  
 return(x)  
}  
extractPreds(Preds1)

## NR R PatID Y.Test PatNum PredBinary  
## 1 0.3228732 0.67712677 Patient\_1 R 1 R  
## 2 0.7188428 0.28115717 Patient\_4 R 4 NR  
## 3 0.6285662 0.37143382 Patient\_8 R 8 NR  
## 4 0.2995221 0.70047786 Patient\_14 R 14 R  
## 5 0.4348569 0.56514306 Patient\_15 R 15 R  
## 6 0.5131471 0.48685290 Patient\_18 R 18 NR  
## 7 0.8790314 0.12096865 Patient\_22 NR 22 NR  
## 8 0.1322514 0.86774856 Patient\_31 NR 31 R  
## 9 0.6772474 0.32275255 Patient\_33 NR 33 NR  
## 10 0.9346382 0.06536184 Patient\_34 NR 34 NR  
## 11 0.1278266 0.87217338 Patient\_37 NR 37 R  
## 12 0.2996724 0.70032761 Patient\_40 NR 40 R

### Question 2 .

**1.3** Using the function you made in **1.2** extract and process the Test Predictions from all elements of *RESULTS*. A - Using a *for()* loop (check the speed using *Sys.time()*)

# your R code  
start <- Sys.time()  
result <- data.frame()  
for (i in seq\_along(RESULTS)) {  
 x <- as.data.frame(RESULTS[[i]]$TestPreds)  
 result <- rbind(result, cbind(Test\_No = i, extractPreds(x)))  
}  
Sys.time() - start

## Time difference of 0.05826283 secs

**Answer:**

and B - using lapply or mclapply. Your final answers should be a single 'data.frame'

start <- Sys.time()  
result <- do.call(rbind, lapply(seq\_along(RESULTS), function(x) cbind(Test\_No = x,   
 extractPreds(as.data.frame(RESULTS[[i]]$TestPreds)))))  
Sys.time() - start

## Time difference of 0.04869103 secs

### Question 3.

**3.1** Using the output of your function in Q2 (you can also just load *Q2\_Out.rdata*), create a new dataframe with the following columns.

Mean: Mean model prediction of responder probability for each patient SEM: Standard error of Mean probability of response, *sem=sd(x)/sqrt(length(x))* ConfUp: The upper 95% confidence interval of Mean, Mean + 2*sem ConfDown: The lower 95% confidence interval of Mean, Mean - 2*sem PatNum: Patient Number Y.Test: Actual response for each patient (the same values as Y.Test in Q2\_Out.rdata)

Using this new dataframe, recreate the plot *Q3\_plot.pdf* that is on blackboard.

Hint1: use *geom\_point()* function from *ggplot2* package Hint2: factors are your friends.

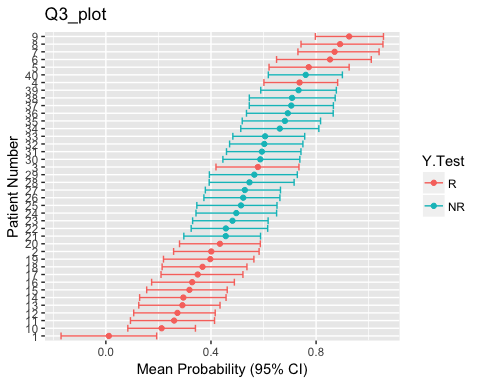
load("Q2\_Out.rdata")  
# your R code  
result <- Q2\_Out  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(forcats)  
more\_results <- data.frame()  
  
split\_patients <- split(result, as.factor(Q2\_Out$PatID))  
for (i in seq\_along(split\_patients)) {  
 Mean <- mean(split\_patients[[i]]$R)  
 SEM <- sd(split\_patients[[1]]$R)/sqrt(length(split\_patients[[i]]$R))  
 ConfUp <- Mean + 2 \* SEM  
 ConfDown <- Mean - 2 \* SEM  
 PatNum <- unique(split\_patients[[i]]$PatNum)  
 Y.Test <- unique(split\_patients[[i]]$Y.Test)  
 more\_results <- rbind(more\_results, cbind(Mean, SEM, ConfUp, ConfDown, PatNum,   
 Y.Test))  
}  
  
# more\_results <- as\_tibble(more\_results)  
more\_results$Mean <- as.numeric(as.character(more\_results$Mean))  
# more\_results$SEM <- as.numeric(as.character(more\_results$SEM))  
more\_results$ConfUp <- as.numeric(as.character(more\_results$ConfUp))  
more\_results$ConfDown <- as.numeric(as.character(more\_results$ConfDown))  
# more\_results$PatNum <- as.numeric(as.character(more\_results$PatNum))  
  
library(ggplot2)  
  
# Define the top and bottom of the errorbars  
  
p <- ggplot(more\_results, aes(x = sort(Mean), y = sort(as.factor(PatNum)), colour = Y.Test))  
p + geom\_point() + geom\_errorbarh(aes(xmax = sort(ConfUp), xmin = sort(ConfDown))) +   
 labs(x = "Mean Probability (95% CI)", y = "Patient Number", title = "Q3\_plot")



### Question 4 .

You have a matrix *seq.mat* that has 4 columns and 20 rows. The first column is SampleID, 2nd (*Sequence*) - sequence, 3rd (*RNASeq*) - RNA sequence, and 4th (*Seq.Count*) - number of times sequence "AGU" (in that specific order!) appears in RNA sequence. Columns *RNASeq* and *Seq.Count* have all values missing and you need to populate them.

seq.mat <- matrix(NA, nrow = 20, ncol = 2)  
colnames(seq.mat) <- c("SmapleID", "Sequence")  
set.seed(123)  
for (i in 1:nrow(seq.mat)) {  
 seq.mat[i, 1] <- paste0("Sample\_", i)  
 seq.mat[i, 2] <- paste(sample(c("C", "G", "A", "T"), size = 1000, replace = TRUE),   
 collapse = "")  
}  
seq.mat <- as.matrix(cbind(seq.mat, RNASeq = NA, Seq.Count = NA))

**4.1** Write a function that will detect whether a given sequence is DNA (only has "C", "G", "A", and "T" letters) or RNA (only has "C", "G", "A", and "U" letters).

If it is DNA sequence your function should:  
1. Print "Seq is DNA, converting to RNA..";  
2. Convert it to RNA by substituting letter "T" to "U" and save the result to *RNASeq* column of *seq.mat* object.

If it is RNA sequence your function should:  
1. Print "Seq is RNA";  
2. Save value of *Sequence* to *RNASeq* column of *seq.mat* object.

Use *apply()* or *sapply()* to apply your function to your data.  
Hint: to test whether a string contains a cetain letter you may use *grepl()*; and *gsub()* to substitute letters.

findingRNA <- function(sequence) {  
 if (grepl("T", sequence)) {  
 print("Seq is DNA, converting to RNA..")  
 rna <- gsub("T", "U", sequence)  
 } else {  
 if (grepl("U", sequence)) {  
 print("Seq is RNA")  
 rna <- sequence  
 }  
 }  
 return(rna)  
}  
  
seq.mat[, 3] <- sapply(seq.mat[, 2], findingRNA)

## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."  
## [1] "Seq is DNA, converting to RNA.."

**4.2** Using the *str\_count()* function populate the *Seq.Count* column of matrix *seq.mat* with the number of times the sequence "AGU" (in this specific order!) appears in *RNASeq* column for each sample. In how many samples does the "AGU" sequence appear more than 15 times?

library(stringr)  
  
seq.mat[, 4] <- sapply(seq.mat[, 3], function(x) str\_count(x, "AGU"))  
  
# In how many samples does the 'AGU' sequence appear more than 15 times?  
sum(seq.mat[, 4] > 15)

## [1] 11

**Answer**