BIOLOGY 664: Integrated Bioinformatics Using R for Both Wet and Dry Scientists

Problem Set 5: Linear Models Using data.frames

Due: At the beginning of class Tuesday, October 25.

Chapter 5 Exercises Using data.frames

Answer the 4 questions in the Chapter 5 Exercises **using a data.frame for every answer**. You should never refer to a matrix in your solutions (unless you are using a matrix to construct a data.frame). Also, create named rows and named columns, and use named indexing whenever possible.

Soft and Hardcopy

Submit a softcopy through blackboard and hand-in a printed hardcopy of your Problem Set. Executing the softcopy in RStudio should produce the hardcopy. Please print double-sided and use Arial font size 7 or 8 in order to save trees.

Extra Credit

I'll be giving extra credit for those Problem Sets that use Latex (pronounced “lay-tech)”. I'm also going to give extra credit for plots that have been "fancied up" to produce more publication-like figures. These plotting enhancements include colors, titles, legends, axis labels, p-values, and R^2s. Being able to create publication-quality figures in R is an important skill, and this is an opportunity to earn extra points while learning to do so. Look at the "Quick-R" and "Producing Simple Graphs with R" links on the online syllabus for good online references for how to fancy-up R plots.

Important notes and hints:

1. Use ***ALL$BT %in% c("B","B1","B2","B3","B4")*** to get the patients in just those stages when creating the data.frame in part 1a.
2. Then use ***allb.df[allb.df == "NA"] = NA*** to replace "NA" strings in the data with real NAs.
3. Use ***grep()*** to get only the columns that contain expression data, ***has.expression = grep("_at$|_st$",names(allb.df))*** gets rid of columns that have non-expression data (age, gender, remission status etc.). ("$" in the regular expression signifies end of string and “|” is logical “or”. Therefore, with this regex we’re grabbing only the columns that end in “_at” or “_st”.)
4. Also, ***not.expression = grep("_at$|_st$",names(allb.df), invert=TRUE)*** uses ***invert=TRUE*** to get the columns that DON'T match the regular expression.
5. Don't use ***featureNames()*** in Question 2. The ***featureNames()*** function is for working with the data as a ExprSet object, but we're accessing the data as a data.frame. So instead if using ***featureNames()***, use just ***names()***.

6. The ***names()*** function is also useful for Question 3.
7. You can work together, but all your written work (including R code) must be your own.

Answer to Question 1:

```
library(ALL)
data(ALL, package="ALL");

# (a) Create a data.frame from the ALL exprSet object that contains only the B-cell ALL patients. Also replace the “NA” strings with real NAs:

    allb.df = data.frame(ALL[,ALL$BT %in% c("B","B1","B2","B3","B4")])
    allb.df[ allb.df == "NA" ] = NA # replace "NA" strings with real <NA>s


# (b) Call table() to get a summary of any factor:

    table(allb.df$BT)


# (c) Instead of learning new commands (like exprs()) in order to work with the deprecated exprSet object, we have transformed the data into a data.frame so that we can analyze the data with the tools we already know. We can use grep() instead of exprs() to obtain just the gene expression (probe set) values:

    # Use grep to get the columns that contain expression data, "$" signifies end of string
    # not.expression = grep("_at$|_st$",names(allb.df), invert=TRUE) # Use invert=TRUE to get the columns that DON'T match the regular expression
    has.expression = grep("_at$|_st$",names(allb.df)) # getting rid of columns that have non-expression data (age, sex, remission status etc.)

    p.residuals = apply(allb.df[has.expression], 2, function(x) shapiro.test(residuals(lm(x ~ allb.df$BT)))$p.value)
    head(p.residuals)

# (d) Use has.expression to apply the bptest() function to just the gene expression values
    library(lmtest)
    p.bptest = apply(allb.df[has.expression], 2, function(x)
      as.numeric(bptest(lm(x ~ allb.df$BT),studentize = FALSE)$p.value))
    head(p.bptest)

# (e) Use sum() to get the totals
    sum(p.residuals > 0.05)
    sum(p.bptest > 0.05)
    sum(p.residuals > 0.05 & p.bptest > 0.05)
```