

Biological Statistics (MMG3002Y)

Assignment 3 10 Points

Assignment goals:

- Practice R concepts learned in class
- Practice regular expressions
- Become familiar with reading in and analyzing data

Note: this assignment is expected to take a long time to complete because we finally have learned enough R concepts to be able to practice real R programs. Please make sure you start right away.

Note: some R packages may be available that perform some functions asked for in this assignment. You are not allowed to use packages for this assignment so you can learn the first principles that you will need in the rest of the course.

Part I: Practice regular expressions (3 marks)

Write regular expressions for each of the following cases. Assume that the regular expression would be used in the following context, where we'd like to check if a string has a particular pattern, and if so, execute a set of statements:

```
if (grep("regular expression", stringVariable)) {  
    <do something here with a set of statements>  
}
```

a. Check to see if a string contains a valid yeast gene name, e.g. YJL045W, YKR034W, YLL023C (hint: a yeast systematic gene name consists of a Y, a letter between A and P indicating chromosome number, an R or L for right or left chromosome arm, 3 numbers indicating chromosome position, and then a W or C to designate the strand, either Watson or Crick)

b. Check to see if a yeast gene is encoded on the Crick strand (Note: YHR143W-C is not encoded on the Crick strand)

c. Check a sequence string for Mot3's (a yeast transcription factor) binding site motif: AAGG(G or T)T

d. Check a sequence string for Hap1's (a yeast transcription factor) binding site motif. '?' implies that there is a single arbitrary nucleotide.
GG???TA?CGG

e. Explain in English what the following regular expression matches:
CGG(A|C|G|T){3}T(A|C|G|T)(A|G)(A|C|G|T){8,12}CCG

E.g. "It matches CGG followed by..."

Write your answers in a text file called assignment3_part1.txt. You do not need to write

a script for this section, though you can test your regular expression in your own script that you don't submit. Each regular expression should be in a line by itself. E.g.

a)

regular expression a here

b)

regular expression b here...

Part II: Finding transcription factor binding sites in yeast DNA sequence (3 marks)

Gene transcription in eukaryotes is controlled by transcription factor proteins, which bind specific sequence motifs near the gene they regulate. In yeast, the transcription factors are well-characterized, such that we often know the specific sequence motif bound by each factor.

We would like to write a script to answer the following question:

How often does each transcription factor motif occur in gene promoter sequences across the whole genome? (you'll need to print a list of transcription factors and a corresponding count of the number of genes whose promoter contains that motif for each transcription factor)

Here are the data that we have:

(1) A tab-delimited list of a subset of yeast transcription factors and their corresponding sequence binding sites (motifs). Note: there are multiple binding sites for some transcription factors—you can count frequencies independently for each of these binding sites. [yeast_tf_motifs.txt](#)

(2) A FASTA file containing the DNA sequence of all yeast genes including 1000bp of upstream and downstream sequence originally downloaded from

<http://www.yeastgenome.org/cgi-bin/seqTools>. [orf_genomic_1000_554.fasta](#)

Note: this contains only the first 554 sequences to keep the file smaller for the purposes of the assignment)

Write R code that reads the FASTA file and converts it to a data frame where the ORF name is stored in column 1 and the corresponding upstream and downstream sequence is stored in column 2. You should be able to find the upstream and downstream sequence using code like:

```
sequenceDatabase$sequences[sequenceDatabase$sequenceNames=="YAL002W"]
```

(This would print the upstream 1000bp and downstream 1000bp DNA sequence, concatenated, for gene YAL002W)

Your finished script should output the following:

Counts for each transcription factor/motif over all DNA sequence available in the FASTA file that comes with the assignment. You should count the number of genes with at least one occurrence of each motif in its promoter sequence (defined as the 1000 bases upstream of the start site), and report this as a fraction over the total number of genes.

Part III and IV: Introduction to the rest of the assignment

We will now focus on statistical analysis of gene expression data. Specifically, we will

investigate genes that are associated with the development of metastases from primary breast tumours. The data from this lab are taken from the following study, which you can read about at <http://www.nature.com/nature/journal/v415/n6871/abs/415530a.html>:

“Gene expression profiling predicts clinical outcome of breast cancer.” Nature 415, 530-536 (31 January 2002). van 't Veer LJ, Dai H, van de Vijver MJ, He YD, Hart AA, Mao M, Peterse HL, van der Kooy K, Marton MJ, Witteveen AT, Schreiber GJ, Kerkhoven RM, Roberts C, Linsley PS, Bernards R, Friend SH.

We will focus on understanding the data, loading the data into R, and doing some simple manipulations of the data. The main goal of this assignment is to find genes that have significantly different expression in metastatic and non-metastatic samples, as these could be important for cancer progression.

Part III: Loading and manipulating gene expression data in R (4 points)

(1) We have provided two data files for this lab: The actual expression data (24481 genes X 78 samples) “assignment3_data.txt” and a simple file that contains the sample assignments of non-metastatic [1] and metastatic samples [2] “assignment3_sample_labels.txt” (one number per line, corresponding to columns in the data file i.e. the 5th line contains the label of the sample measured in the 5th data file column). First load all the data into R using the “read.table” command. You may want to transfer everything over into a matrix using a command like “cancerdata_matrix = data.matrix(cancerdata)”.

(2) Write a series of commands that will do each of the following.

- (a) Print the data for the BRCA1 gene (gene index is 12361)
- (b) Compute the mean and standard deviation of the expression data for the BRCA1 gene across all samples
- (c) Compute the mean and standard deviation of the expression of all genes in the first metastatic sample (column #45).
- (d) Create a vector called met_samples that consists of the column indices of all metastatic samples
- (e) Create a vector called nonmet_samples that consists of the column indices of all non-metastatic samples
- (f) Use the vectors created in parts (d) and (e) to compute a new vector expr_difference, which contains the difference in the mean expression for each gene between the metastatic and non-metastatic sample groups. (Hint: For each gene, this will require you to compute the mean expression for that gene for each of the non-metastatic and metastatic groups and measure the difference between these two means. You can use the “apply” function to do this.).
- (g) Using the expr_difference vector you just created, print the gene that has the largest positive difference (i.e. metastatic < non-metastatic) and the largest negative difference (i.e. metastatic > non-metastatic) in mean expression between the metastatic and non-metastatic groups.

Part IV Optional bonus section: Statistical analysis of gene expression data (2 points)

We will find a subset of the genes that is differentially expressed using the T-test. Remember, the purpose of the t-test is to statistically quantify the difference in expression for a single gene between two groups of samples (in this case, metastatic

and non-metastatic), and answer the yes/no question “is this difference significant?”

(1) Our goal is to answer the yes/no question of whether or not each gene is significant. To do this, we will need to compute a p-value that reflects the significance of the t-statistic. Small p-values reflect cases where the gene is very likely different between the two sets of samples, and typically a cutoff of 0.05 is used— genes with less than a 0.05 p-value are called significantly differentially expressed. (In a real world situation, we would have to apply multiple testing correction, such as FDR, but we will ignore this here). Use the “t.test” function in R to calculate the p-value of the t-statistic for every gene (for the difference in expression between metastatic and non-metastatic samples). You can either use a simple “for” loop, or you can use the “apply” function, but this is a bit complicated here.

(2) Answer the following questions by writing short sections of R code:

- a. Print a list of significantly differentially expressed genes (p-value < 0.05), sorted in order of their significance (most to least). Include the gene name, the t-statistic, and the corresponding p-value in your list.
- b. How many genes are significantly differentially expressed at a p-value < 0.05?
- c. How many genes are significantly under-expressed in metastatic tumors vs. non-metastatic tumors?
- d. How many genes are significantly over-expressed in metastatic tumors vs. non-metastatic tumors?

These days, RNA-Seq has mostly replaced mRNA profiling microarrays. However, once the RNA-Seq data is normalized and expression level computed using standard tools (see e.g. <http://bioconductor.org/packages/release/bioc/html/edgeR.html>), the above workflow is still applicable.

Submitting your assignment

When you’re finished with the assignment, make a report of any questions you answered plus any requested output, and gather the scripts that you modified. Place them all in a folder, and create a single archive file called `assignment3.tar.gz`. Remember, you’ll need to use a command like this:

```
tar -czf assignment3.tar.gz <your name>_Assignment3
```

When you’re finished, upload the `assignment3.tar.gz` file to the course website. You only need to upload your single `.tar.gz` file that contains all your answers and output for this assignment.

You’re finished with Assignment 3!