20.109 Publically Available Big Data

Amanda Kedaigle, Ernest Fraenkel

1/11/2018

Public Data Sources

Today, there is a wealth of data being created and analyzed in biology - and most of it is available for free to the scientific community. Sharing data accelerates biomedical research, allows for better communication and collaboration between scientists, and increases the efficiency of scientific funding. Our analyses of our own experiments become much more powerful when we can directly compare our results to previous experiments.

Some popular data resources include The Cancer Genome Atlas (TCGA), The Encyclopedia of DNA Elements (ENCODE), and The NIH Roadmap Epigenomics Mapping Consortium (Roadmap). All of these resources represent huge collaborative efforts between dozens of labs to create libraries of well-controlled data, including gene expression, DNA mutations, epigenomics, and more. They include experiments done on popular cell lines, human tissues, and tumors. We'll compare some of those data with the RNAseq analysis of the colorectal cancer cell line that you performed in the last lab exercise.

The Cancer Genome Atlas - Gene Expression

We'll begin with one of the most popular data resources for cancer studies: The Cancer Genome Atlas. This project was started in 2005, and is run by the National Cancer Institute and the National Human Genome Research Institute. They have sequenced and analyzed thousands of human tumors and other samples across more than 30 types of cancer.

Luckily, to sort and download all of this data, we can once again turn to Bioconductor[1]. TCGA data is accessible via the NCI Genomic Data Commons (GDC) data portal, among other sources. The data provided by GDC data portal can be accessed using Bioconductor package TCGAbiolinks (Colaprico et al, 2016).

TCGAbiolinks has three main functions GDCquery, GDCdownload and GDCprepare that should sequentially be used to respectively search, download and load the data as an R object.

GDCquery uses GDC API to search the data for a given project and data category and filters the results by samples, sample type, file type and others features. This function returns an object with a summary table with the results found. After the search step, the user will be able to download the data using the GDCdownload function. The downloaded data will be saved in our working directory. Finally, GDCprepare transforms the downloaded data into a summarizedExperiment object, a data structure similar to the one you worked with in the last lab. To match our data from a colorectal cell line, let's download colon cancer data. The TCGA colon adenocarcinoma project is called "TCGA-COAD", and we want to download Gene Expression Quantification data from tumor samples, called "Primary solid Tumor" samples, and normal colon tissue, called "Solid Tissue Normal."

```r
library(TCGAbiolinks)
library(SummarizedExperiment)

query1 <- GDCquery(project = "TCGA-COAD",
                  data.category = "Transcriptome Profiling",
                  data.type = "Gene Expression Quantification",
                  workflow.type = "HTSeq - Counts",
                  sample.type = "Primary solid Tumor")
```

[1] Much of this analysis is based on the Bioconductor TCGA Workflow, by Tiago C. Silva, Antonio Colaprico, Catharina Olsen, Fulvio D'Angelo, Gianluca Bontempi, Michele Ceccarelli, and Houtan Noushmehr.

```
query2 <- GDCquery(project = "TCGA-COAD",
                  data.category = "Transcriptome Profiling",
                  data.type = "Gene Expression Quantification",
                  workflow.type = "HTSeq - Counts",
                  sample.type = "Solid Tissue Normal")

# We will use only 40 samples to make the analysis faster - otherwise we'd
# be computing on >500 samples!
query1$results[[1]] <-  query1$results[[1]][1:30,]
query2$results[[1]] <-  query2$results[[1]][1:10,]
GDCdownload(query1)
GDCdownload(query2)

coad.exp <- GDCprepare(query1,
                      save = TRUE,
                      summarizedExperiment = TRUE)
norm.exp <- GDCprepare(query2,
                      save = TRUE,
                      summarizedExperiment = TRUE)

#Bind the Normal Samples and Tumor samples into one summarizedExperiment
coad.exp = cbind(coad.exp,norm.exp)
```

One of the powerful things about the human tumor data we just downloaded is that TCGA also provides clinical data about the patients with these tumors. Check out the colData of the summarizedExperiment we just made to see what kind of metadata we have access to. The column “definition” holds the sample.types we downloaded, “Solid Tissue Normal”, and “Primary solid Tumor.”

To start off, we’ll perform analyses similar to what we did last time: find significantly differentially expressed genes in these samples. First, we’ll convert the summarizedExperiment object to the data structure you are more familiar with: a DESeqDataSet object. We set the “design” of the DESeq experiment to “~definition” to tell it to compare tumor to normal.

```
library(DESeq2)

#Make DESeqDataSet and explore it
coad.dds <- DESeqDataSet(coad.exp, design = ~definition)
```

Using the commands you learned in the last lab, find out how many genes we have information for in this coad.dds object. Print out the top few genes in the counts matrix to see what the rows look like.

```
#Write your code here
```

You should notice that the gene names look weird. They aren’t Gene Symbols that you might be used to, but Ensembl IDs. We will convert these to the more readily understood Gene Symbols later.

To explore our DESeqDataSet, make a PCA plot as we did in the last lab session. Color the points according to the “definition” column. The rlog calculation takes a very long time to run on such a big matrix, so we’ve done it for you and stored it in the data file “coad.rld.rda”.

```
#Instead of running this line:
#coad.rld = rlog(coad.dds)
#we load the RData object we already made
load("~/Desktop/RNA-seq data analysis/coad.rld.rda")

#Write PCA code here
```

Save the PCA plot, and discuss anything you notice about this spread of samples. There is something that shold be fixed before we go on with the data - what is it?

```
#This is the code that will fix the problem we noticed in the first PCA plot
coad.rld <- coad.rld[ , coad.rld$sample != "TCGA-A6-6650-01B" ]
coad.dds <- coad.dds[ , coad.dds$sample != "TCGA-A6-6650-01B" ]

#Now use the same command to make a new PCA plot - write here
```

Save this PCA plot as well. How is this better? Note that until now, we've been using "definition" as the interesting group to color the nodes. Try coloring it by different metadata, such as "tumor_stage" to see if gene expression correlates with other clinical data.

```
#Write PCA code here
```

Now we'll run DESeq on this DESeqDataSet like we did last time. Use the instructions from last time to write code that will run DESeq, get the results for tumor vs normal comparison, and tell you how many genes were significantly changed.

```
#Write code here
```

Comparing TCGA Data to our data

Thus far, we've analyzed TCGA data on its own, comparing cancerous tissue to nearby normal tissue. This is important - but it was already done by the TCGA scientists! In your experiments, you'd like to compare this available data to your data. Let's load the RData object we made in the last lab. To make a heatmap with all of these samples together, we'll need to concatenate our r-logged dds object from last time, which we named rld, with our TCGA r-logged object, called coad.rld

```
#load the data from our experiment
load("~/Desktop/RNA-seq data analysis/afterAnalysis.RData")

#We want to combine two r-logged data structures, but they have different row names
head(assay(rld))
head(assay(coad.rld))[,1:4]
```

Here is where we run into the problem with the two datasets using different gene names, and including different numbers of genes. We need to convert the names of the gene names in coad.rld to Gene Symbols, which we can do with the AnnotationDbi library.

```
library("AnnotationDbi")
library("org.Hs.eg.db")
coad.symbol = mapIds(org.Hs.eg.db,
                     keys=row.names(coad.rld),
                     column="SYMBOL",
                     keytype="ENSEMBL",
                     multiVals="first")
```

Now, we need to deal with the fact that coad.rld includes a lot more rows than rld does. We want to pare down coad.rld so that only rows that appear in rld remain.

```
#coad.rld.sub will have rows of coad.rld such that the symbol
#for that genes appears in rld
coad.rld.sub = coad.rld[coad.symbol %in% row.names(rld),]

#Change rownames of coad.rld.sub to use the symbols
rownames(coad.rld.sub) = coad.symbol[coad.symbol %in% row.names(rld)]
```

```r
#Only keep genes that are not duplicated
coad.rld.sub = coad.rld.sub[!duplicated(rownames(coad.rld.sub)),]

#Pare down rld as well so they have the same rows
rld.sub = rld[row.names(rld) %in% row.names(coad.rld.sub),]

#Bind the two r-logged data structures together
rld_all = cbind(assay(rld.sub), assay(coad.rld.sub))
```

And finally, we want to make a heatmap using this rld_all object. First we'll make a data.frame called def so we can add annotations to the heatmap based on whether a sample is tumor or normal, or comes from our cancer cell line data.

```r
def = c(rep("cancer cell line",8), as.character(colData(coad.rld)$definition))
names(def) = colnames(rld_all)
def = as.data.frame(def)

library("pheatmap")
trld = t(rld_all)
sampleDists = dist(trld, method = "euclidean")
pheatmap(sampleDists, labels_row=colnames(rld_all), annotation_row=def)
```

Save this image. Does this heatmap look like you were expecting? Why might it look like this?

Comparing pathway activity in TCGA data and our data

Instead of expecting values of gene expression to be the same across these experiments, let's focus on some genes of interest. For example, let's compare known pathways in these datasets.

In addition to sharing data in public databases, biologists also have databases of gene functions and annotations, such as the Gene Ontology we looked at during the last lab. KEGG (http://www.genome.jp/kegg/) contains a database of pathway information. To look at differential expression of genes within KEGG pathways, the Bioconductor package pathview can be used.

We need to give pathview a named vector of genes with their expression level, the ID of the pathway we are interested in which can be found in KEGG database, the species ('hsa' for Homo sapiens) and the limits for the gene expression. To make the named vector of gene expression levels, pathview expects our genes to be named with yet a third type of gene name - an Entrez ID. We'll use the AnnotationDbi package again to get a list of Entrez IDs for our genes.

```r
#get Entrez IDs
entrez = mapIds(org.Hs.eg.db,
                keys=row.names(coad.res),
                column="ENTREZID",
                keytype="ENSEMBL",
                multiVals="first")

#Create a named vector of gene expression fold change values
logFCvector = as.numeric(coad.res$log2FoldChange)
names(logFCvector) = entrez

library(pathview)
# pathway.id: hsa04110 is the cell cycle pathway
hsa04110 <- pathview::pathview(gene.data  = logFCvector,
                                   pathway.id = "hsa04110",
                                   species    = "hsa",
                                   limit = list(gene=as.integer(max(abs(logFCvector),na.rm=TRUE))),
                                   out.suffix="TCGA")
```

This wrote three files to your computer: hsa04110.png and hsa04110.xml are KEGG's depiction of the cell cycle pathway. The file hsa04110.TCGA.png shows the data overlaid onto the pathway by coloring in the genes.

Now, we can do the same thing for our DLD1 cell line data. Since we've already loaded the RData object from the last lab, the object 'res' contains the differential gene expression data.

```
#get Entrez IDs
dld1.entrez = mapIds(org.Hs.eg.db,
                 keys=row.names(res),
                 column="ENTREZID",
                 keytype="SYMBOL",
                 multiVals="first")

#Create named vector & run pathview
logFCvector = as.numeric(res$log2FoldChange)
names(logFCvector) = dld1.entrez
hsa04110_DLD1 <- pathview::pathview(gene.data  = logFCvector,
                                   pathway.id = "hsa04110",
                                   species = "hsa",
                                   limit = list(gene=as.integer(max(abs(logFCvector),na.rm=TRUE))),
                                   out.suffix="DLD1")
```

This created a new file hsa04110.DLD1.png Compare these results to the ones using the TCGA data. What can we learn here?

Drug Signatures

Another source of publically available data is from big screens of cell lines treated with many drugs. Connectivity mapping uses gene expression profiling to connect genes, diseases, and drugs. It is a process whereby gene expression profiles of cell-line response to drugs are compared to query gene signatures, in order to see if your gene expression signature is similar to known drug signatures, and identify potential drugs able to alter the gene signature's expression. We can access connectivity map data with the Bioconductor package gCMAP (Sandmann et. al., 2014).

To compare our experimental data to the data in gCMAP, let's get a "signature" for our experiment. We'll use the simplest one we can think of: logFC for significantly differentially expressed genes. Normally, we would want several experiments to confirm a gene signature, not just one.

For this package, we want our signature to be a named vector of logFC values, where the name values use Gene Symbols.

```
#Get a named vector of only signficant genes (with padj<0.05)
resSig = resslim[resslim$padj<0.05 ,]
signature = as.numeric(resSig$log2FoldChange)
names(signature) = rownames(resSig)

library(ccmap)
library(ccdata)

data(cmap_es)
```

```r
top_cmap  <- query_drugs(signature, cmap_es)
head(top_cmap, 3)
```

Look these top 3 drugs up in PubChem (https://pubchem.ncbi.nlm.nih.gov/). What are the mechanisms of action of these drugs? Look up etoposide, the drug that we used in our expression experiment as well. Are you surprised by the results we got?

The data that we accessed using the "data(cmap_es)" command was from the first version of the connectivity map. We can also access the newer L1000 dataset.

```r
data(l1000_es)
top_l1000 <- query_drugs(signature, l1000_es)
head(top_l1000, 3)
```

The top results from l1000 are Broad submitted drugs with no known mechanism of action. Based on these results, what might you suggest as the possible mechanism of action of some of these drugs? How would you test your theory?

Explore TCGA mutation data

You don't always need to program to find cool information in the TCGA! For the rest of this lab, go to the Genomic Data Commons Portal, another way to access TCGA data, and explore the data using their website. You can find it at https://portal.gdc.cancer.gov/ .

On this website, click on the "Exploration" tab, and select "TCGA-COAD" under "Project". That's the colon cancer project we've been exploring in this lab so far. This page will show you information about all of the cases of colon cancer sequenced in the TCGA-COAD project. Explore this page and select one of the top most mutated genes in colon cancer to find out more about. Get creative and see if you can learn something cool from these data!

Here are a few ideas for questions to explore about your gene. Create a Kaplan-Meier plot to see how patients with mutations in your gene fare. Are mutations in your gene protective or harmful, or have little effect? What kind of mutations are usually found in this gene? Back on the "Explore" page, click the "Genes" tab on the left and search for your gene. Are mutations in this common in many other kinds of cancer? Click on the gene name to learn more about the gene. Can you come up with a hypothesis about why mutations in this gene might act this way?

Epigenetic data

Another advantage of available data in resources like the TCGA is that there is more than gene expression data in these databases. You can find relevant data of other types, such as epigenetic, proteomic, or DNA mutation data, even if you have limited resources or time to do the experiments yourself.

We'll focus on epigenetic data today - specifically, DNA methylation. DNA methylation is an important component in numerous cellular processes, such as embryonic development, genomic imprinting, X-chromosome inactivation, and preservation of chromosome stability. In mammals DNA methylation is found sparsely but globally, distributed in definite CpG sequences throughout the entire genome; however, there is an exception. CpG islands (CGIs) which are short interspersed DNA sequences that are enriched for GC. These islands are normally found in sites of transcription initiation and their methylation can lead to gene silencing.

Thus, the investigation of the DNA methylation is crucial to understanding regulatory gene networks in cancer as the DNA methylation represses transcription. Therefore, the DMR (Differentially Methylated Region) detection can help us investigate regulatory gene networks.

This section describes the analysis of DNA methylation using the Bioconductor package TCGAbiolinks. For this project, we'll get samples from the colon adenocarcinoa (COAD) project that have both gene expression and DNA methylation data. We start by checking the mean DNA methylation of different groups of samples, then look for DMRs in which we search for regions of possible biological significance, (e.g., regions that are methylated in one group and unmethylated in the other). After finding these regions, they can be visualized using heatmaps.

The DNA methylation data are from the 450k array platform. So, instead of a sequencing assay like RNA-seq, where all the RNA in the cell is fragmented and sequenced, an array was created with 450,000 "probes". That is, DNA matching sites for potential methylation are on a chip. Running DNA from your sample over the chip, and then using staining & imaging to reveal which sites are bound by methylated vs unmethylated cytosines, reveals which probes are methylated in your sample.

```
# Samples that have both expression and methylation data
coad.met.samples <- matchedMetExp("TCGA-COAD")

#Download data
query <- GDCquery(project = "TCGA-COAD",
                  data.category = "DNA methylation",
                  platform = "Illumina Human Methylation 450",
                  legacy = TRUE,
                  sample.type = "Primary solid Tumor",
                  barcode = coad.met.samples)
#Limit number of samples to make it faster
query$results[[1]] <-  query$results[[1]][1:20,]
GDCdownload(query)
met <- GDCprepare(query, save = FALSE)

query <- GDCquery(project = "TCGA-COAD",
                  data.category = "DNA methylation",
                  platform = "Illumina Human Methylation 450",
                  legacy = TRUE,
                  sample.type = "Solid Tissue Normal",
                  barcode = coad.met.samples)
#Limit number of samples to make it faster
query$results[[1]] <-  query$results[[1]][1:10,]
GDCdownload(query)
metNorm <- GDCprepare(query, save = FALSE)

met <- cbind(met,metNorm)

# remove probes with NA
met <- subset(met,subset = (rowSums(is.na(assay(met))) == 0))
```

After using the function TCGAvisualize_meanMethylation function, we can look at the mean DNA methylation of each patient in each group. It receives as argument a SummarizedExperiment object with the DNA methylation data, and the arguments groupCol and subgroupCol which should be two columns from the sample information matrix of the SummarizedExperiment object (accessed by the colData function) (see listing below lines 70-74).

```
#plot bar chart for mean level of methylation
TCGAvisualize_meanMethylation(met,
                              groupCol = "definition",
                              group.legend  = "Groups",
                              filename = NULL,
                              print.pvalue = TRUE)
```

Do these cancers have significantly different levels of methylation?

The next step is to define differentially methylated CpG sites between the two groups. This can be done using the TCGAanalyze_DMR function. The DNA methylation data is presented in the form of beta-values that uses a scale ranging from 0.0 (probes completely unmethylated) up to 1.0 (probes completely methylated).

To find these differentially methylated CpG sites, first, the function calculates the difference between the mean DNA methylation (mean of the beta-values) of each group for each probe. Second, it tests for differential expression between two groups, adjusting for multiple hypotheses. Arguments of TCGAanalyze_DMR was set to require a minimum absolute beta-values difference of 0.15 and an adjusted p-value of less than 0.05.

After these tests, a volcano plot (x-axis: difference of mean DNA methylation, y-axis: statistical significance) is created to help users identify the differentially methylated CpG sites and return the object with the results in the rowRanges. This step will take a while.

```
met <- TCGAanalyze_DMR(met,
                       groupCol = "definition", # a column in the colData matrix
                       group1 = "Primary solid Tumor",
                       group2 = "Solid Tissue Normal",
                       p.cut = 0.05,
                       diffmean.cut = 0.15,
                       save = FALSE,
                       legend = "State",
                       plot.filename = "COAD_metvolcano.png",
                       cores = 4 # if set to 1 there will be a progress bar
)
```

Describe how to read this volcano plot.

Integrating Gene Expression & Epigenetic Data

Our analyses of -omic data (i.e. transcriptomic, or gene expression, and epigenomic) can get even more powerful when we integrate them together. You'll learn more about that in a future lecture! Let's start with a basic integration of these two data types.

After finding differentially methylated CpG sites, one might ask whether nearby genes also undergo a change in expression either an increase or a decrease. DNA methylation at promoters of genes has been shown to be associated with silencing of the respective gene. We use the TCGAvisualize_starburst function to create a starburst plot. The log10 (corrected p-value) for DNA methylation is plotted on the x-axis, and for gene expression on the y-axis, for each gene. The horizontal black dashed line shows the adjusted p-value of 0.1 for the expression data and the vertical ones shows the adjusted P value of 0.1 for the DNA methylation data. While the argument met.p.cut and exp.p.cut control the black dashed lines, the arguments diffmean.cut and logFC.cut will be used to highlight the genes that respect these parameters (circled genes in Figure below). For the example below we set higher p.cuts trying to get the most significant list of pair gene/probes. But for the next sections we will use exp.p.cut = 0.01 and logFC.cut = 1 as the previous sections.

```
#Make our DEGs look like what the function is expecting
DEGs = coad.res[complete.cases(coad.res$padj),]
DEGs = DEGs[DEGs$padj<0.05,]
datafDEGs <- as.data.frame(DEGs)
colnames(datafDEGs)[6] <- "FDR"
colnames(datafDEGs)[2] <- "logFC"
# We will use only chr9 to make the graph readable
met9 <- subset(met,subset = as.character(seqnames(met)) %in% c("chr9"))
# Starburst Plot
starburst <- TCGAvisualize_starburst(met9,    # DNA methylation with results
                                     datafDEGs,    # DEG results
                                     group1 = "Primary solid Tumor",
                                     group2 = "Solid Tissue Normal",
                                     filename = "starburst_chr9.png",
                                     genome = "hg19",
                                     met.platform = "450K",
                                     met.p.cut = 0.05,
                                     exp.p.cut = 10^-2,
                                     diffmean.cut = 0.15,
                                     logFC.cut = 1,
                                     width = 15,height = 10,
                                     names = TRUE)
```

There was a step when running this function that took a few seconds - “Downloading genome information”. This was followed by the step “Mapping probes to nearest TSS” (which stands for the Transcription Start Site of a gene). Why did we need to do this in order to make this plot? Describe this important step for integrating gene expression data with probe methylation data.

What do we learn from the resulting starburst plot? Look up some of the highlighted genes and describe whether their behavior is what you expected.