Level 1: Module 3 – Quality Control

Table of Contents

DATA CURATION AND PRE-PROCESSING *2*
QUALITY CONTROL (QC) *2*
simpleaffy package **3**
arrayQualityMetrics package **4**
affyPLM package **5**
BACKGROUND CORRECTION AND NORMALIZATION *6*
BATCH CORRECTION *7*
DATA VISUALIZATIONS *8*
Boxplots **8**
PCA **8**
Correlation Heatmaps **10**
All Deliverables: *12*

Guo et al., 2020 uses three GEO datasets and TCGA data for ceRNA network construction and analysis. We are aiming to acquire basic microarray data analysis techniques using two of these datasets from GEO, GSE32323 (Khamas et al., 2012) consisting of 17 normal and 17 tumor samples, and GSE8671 (Sabates-Bellver et al., 2007) consisting of 32 normal and 32 tumor samples making 98 samples in total.

Before starting with the analysis:
Please refer to previous modules for more information.

Technical Background

- Familiarize yourself with R programming and the RStudio environment (Module 1)
- Familiarize yourself with the data and the Gene Expression Omnibus (GEO) database (Module 2)

Biology Background

- Read the Guo paper (Module 1)
- Familiarize yourself with the concept of the microarray sequencing (Module 1)
- Get a better understanding of the data from these two datasets, e.g. (Module 2)
    - What type of experiment/protocol was used to prepare the samples?
    - What instrumentation was used?
    - For which genome was the data generated?

DATA CURATION AND PRE-PROCESSING

Follow the steps outlined in Module 2 to prepare the required data, metadata, and packages.

QUALITY CONTROL (QC)

Deliverable 1: Analyze output from at least one of the following QC methods. I recommend you run all methods and learn about the different metrics used to measure data quality. Your deliverable should include the output and a brief summary of your interpretation.

The purpose of QC is to check whether the data and any conclusions drawn from it can be considered reliable. It also helps us detect outliers and improve the signal-to-noise ratio. If our QC shows that there are many outliers in our data (or a low signal-to-noise ratio), the data may be too variable to find meaningful results. There are several packages that can be

used for assessing the quality of our raw data. Try the following packages and compare their results.

simpleaffy package

This package's functions produce a statistical analysis report that can be nicely visualized.

QC Analysis Report

Function: `qc()`

Input: AffyBatch Object (output generated by `ReadAffy()`)

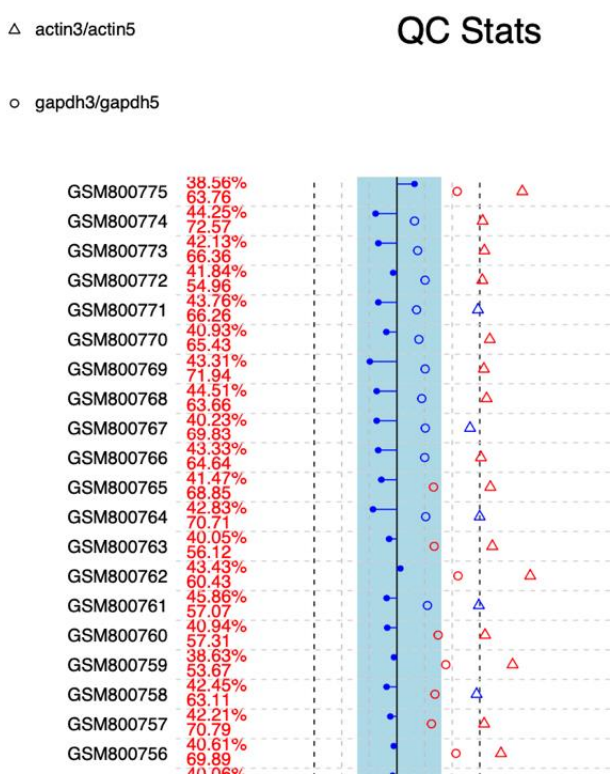
Visualization of the Result

Function: `plot()`

Input: QC Stats Object (output generated by `qc()`)

Note: affyQCReport package produces a plot similar to this one, so if you want to save time, you can skip this package. However, the QC plot in the affyQCReport is often squished and less readable since it is confined to one page.

Desired Output: Your visualization should look similar to the partial plot below



Note: If you have a plot in the plot window (lower right panel in RStudio by default) that is squished or doesn't look "correct," try exporting the plot and setting the dimensions to something larger. For the image to the left, I exported the plot as a pdf with dimensions of 25x8.

Interpretation: simpleaffy Vignette
Feel free to look at other resources, too! I found the one linked above particularly helpful.

arrayQualityMetrics package

This package's function will take a long time to run and requires a lot of memory

The function produces an HTML report, containing many visualizations of different QC methods.

HTML report

Function: `arrayQualityMetrics()`

Input: AffyBatch Object (output generated by `ReadAffy()`)

Additional Arguments:

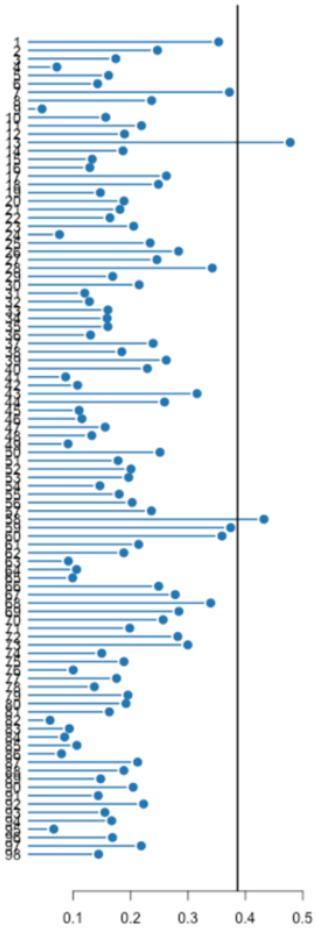
`Outdir = ""` specify a directory for storing the results

`force = T` overwrite existing files in the specified directory

`do.logtransform = T` log transform the intensities

Note: arrayQualityMetrics() typically takes a long time to run and uses a lot of memory. If you get an error message that mentions "cannot allocate vector of size…", try increasing the memory you're allocating to RStudio using the function: memory.limit()

- Figure 5: Outlier detection for Boxplots.



Interpretation:

Since there are multiple different plots in this report, I advise you to research the specific plots and their methods. Figuring out how to interpret each plot is fairly straight-forward and simplified thanks to the handy "outlier detection" plots produced in the report (see left). Please ask any questions you have on the forum.

In the figure on the left, we can see that samples 13 and 58 cross the vertical line and are therefore classified as outliers based on Boxplot analysis. If we look at the order of the samples using:

```
colnames(affyBatchData@assayData$exprs)
```

we can conclude that these numbers correspond to samples GSM215063 and GSM215109.

affyQCReport package

This package's function might take a long time, depending on your computer.

The function ran fairly quickly on my 1 year old Windows Surface, but took a much longer time on my 5 years old MacBook Pro.

QC report

Function: `QCReport()`

Input: AffyBatch Object (output generated by `ReadAffy()`)

Additional Arguments:

`file = "output.pdf"` specify output file's name

Desired Output: A 6-page report in pdf format. It's okay if the first page isn't readable. You should focus on the 6th page for interpreting results.

Interpretation: QCReport Figure Details

Feel free to look at other resources, too! I found the one linked above particularly helpful.

affyPLM package

This package's functions compute Relative Log Expression (RLE) and Normalized Unscaled Standard Error (NUSE) scores of the microarray samples.

Fit Probe Level Model (PLM)

Function: `fitPLM()`

Input: AffyBatch Object (output generated by `ReadAffy()`)

Additional Arguments:

`normalize = T` normalize data using quantile normalization

`background = T` background correct using RMA method

Desired Output: Two boxplots – one showing median RLE scores and the other showing median NUSE scores.

Interpretation: Bioconductor Packages Presentation (STEM-Away) – slides 12-15

- RLE In-Depth Explanation – page 15
- NUSE In-Depth Explanation – page 14

Feel free to look at other resources, too! I found the one linked above particularly helpful.

BACKGROUND CORRECTION AND NORMALIZATION

Background correction removes signal from nonspecific hybridization (i.e., signal emitted by things other than a sample hybridizing to a probe). Normalization corrects for systematic biases due to environment variations such as different absorption rates, spatial heterogeneity in a microarray array chip, and others. These two processes are sometimes performed separately, but the packages we're using perform these two methods in one function. There are 3 different methods of background correction and therefore 3 possible functions you can use.

You only need to use <u>one</u> of the following methods.

All the following methods take an AffyBatch Object as their argument.

A. **mas5()**
B. **rma()**
C. **gcrma()**

You should spend some time researching these background correction methods to see how they're different or similar and if there's certain cases where one might be preferred over another.

Note 1: Both RMA and GCRMA do log2 transformation on the background corrected/ normalized data. MAS5 does not, so you will need to do this additional step before moving on with you MAS5 corrected data.

Note 2: RMA correction runs the fastest.

Note 3: `rma()` is a special packaging of expresso(). So, you can get the same result by using:

```
expresso(data.raw, bgcorrect.method="rma",
          normalize.method="quantiles",
          pmcorrect.method="pmonly",
          summary.method="medianpolish")
```

Tip: Any data submitted as part of your deliverables will not be reviewed. However, it's advised that you save a copy of relevant data as a CSV for your convenience. (You can read in the CSV as opposed to rerunning all of your code).

BATCH CORRECTION

Use the ComBat() function, your background/normalized corrected data, metadata, and the object from model.matrix() to correct for batch effects while preserving features of interest (which include cancer and normal labels). Batch effects occur when you combine data from two different sources (e.g., comparing feet to meters). Batch correction makes the data comparable between the different "batches" (i.e., data sets) so that we can use them in the same analysis.

Build a model preserving features of interest

Function: `model.matrix()`

Input:

Factor of Feature of Interest (whether the sample is normal or cancer)

- this should be a column in your metadata

Data Frame (metadata)

Perform batch correction

Function: `ComBat()`

Input:

Data Frame (background corrected/normalized data)

Vector containing Batch Information (from metadata)

Model from `model.matrix()`

For more information on how to do batch correction:

SVA Package Tutorial – Section 7

Tip: Any data submitted as part of your deliverables will not be reviewed. However, it's advised that you save a copy of relevant data as a CSV for your convenience. (You can read in the CSV as opposed to rerunning all of your code).

DATA VISUALIZATIONS

Deliverable 2: 3 visualizations from each of the methods below using batch corrected (or background corrected/normalized data).

Boxplots – To determine the existence of potential outliers and check whether the data clusters as expected, it is important to see how much the samples are comparable based on their quartile distribution. Hence, we look for samples that stand out in terms of interquartile ranges (IQR) by looking at boxplots. **You can use standard R plots or ggplot2 to create your boxplots.**

PCA – PCA is mostly used as a tool in exploratory data analysis (EDA) and for making predictive models. It is often used to visualize genetic distance and relatedness between populations. The statistics and math behind PCA are based on linear algebra concepts. If you're interested in learning more, I recommend you watch some of StatQuest's videos on the process. You can find them on YouTube. Additional resources will be linked on the STEM-Away Level 1: Module 3 post. **You can use standard R plots or ggplot2 to plot PC1 vs. PC2.**

Build a PCA model

Function: `prcomp()`
Input: Data Frame
Additional Arguments:

`scale=F`	don't scale variables to have unit variance
`center=F`	don't shift variables to be centered at zero

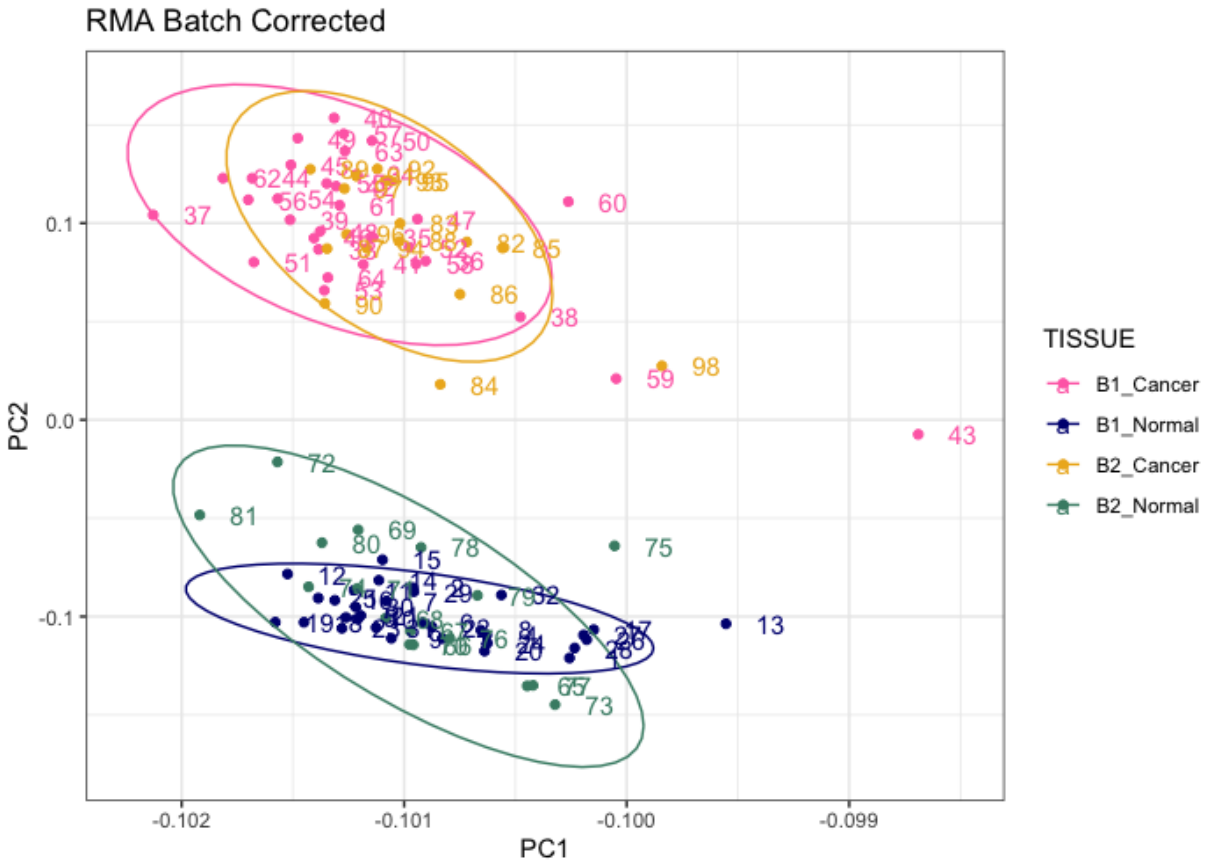
Note: These arguments are basically telling the function to leave the data as it is. If these values were true, the function would normalize the data to have a normal distribution. Since we've already normalized the data, we don't want any further modifications.

View relevant PCA data

- Access the `$rotation` attribute of your PCA model (output of `prcomp()`) to view the PC values. You will be using the columns for PC1 and PC2 to make your plot
- Access the `$importance` attribute to examine the percent of variability explained by each principal component

Plot PC1 vs. PC2 using standard R plots or ggplot2.
You should notice that cancer and normal data points show definitive clusters.

Desired Output: Your plot should look similar to the figure below [I used ggplot2]. You don't need the ellipses or numbers; they just make the image easier to interpret quickly. **You should have the different colors representing 4 groups: batch 1 cancer, batch 1 normal, batch 2 cancer, batch 2 normal.** To do this, I added a column to my metadata. Colors should represent 2 groups if using 1 data set: Normal, Cancer.



Correlation Heatmaps – Heatmaps use color gradients to represent the intensity of a certain metric. Typically, the metric represents how similar or dissimilar two items are. For our heatmap, we'll be using a dissimilarity metric representing *1-correlation* between two samples. Samples that are the same will have a value of 0 and samples that are the most different will have values closer to 1 or -1. **You should use the pheatmap package and function to create this plot.**

The heatmaps made with the pheatmap also annotate the plot based on hierarchical clustering. Hierarchical clustering is an unsupervised learning technique that tries to cluster similar things together with no prior knowledge using dis/similarity metrics. The branching structure can be interpreted much like a family tree. Samples connected by "lower" branches have a "closer relationship" meaning they are more similar than those with further branches. If you want to know more about this clustering method, I recommend you watch some of StatQuest's videos on the topic. You can find them on YouTube. Additional resources will be linked on the STEM-Away Level 1: Module 3 post.

Calculate correlation

Function: `cor()`
Input: Data Frame

Plot heatmap

Function: `pheatmap()`
Input: Data Frame (1-cor(data))*
Additional Arguments:

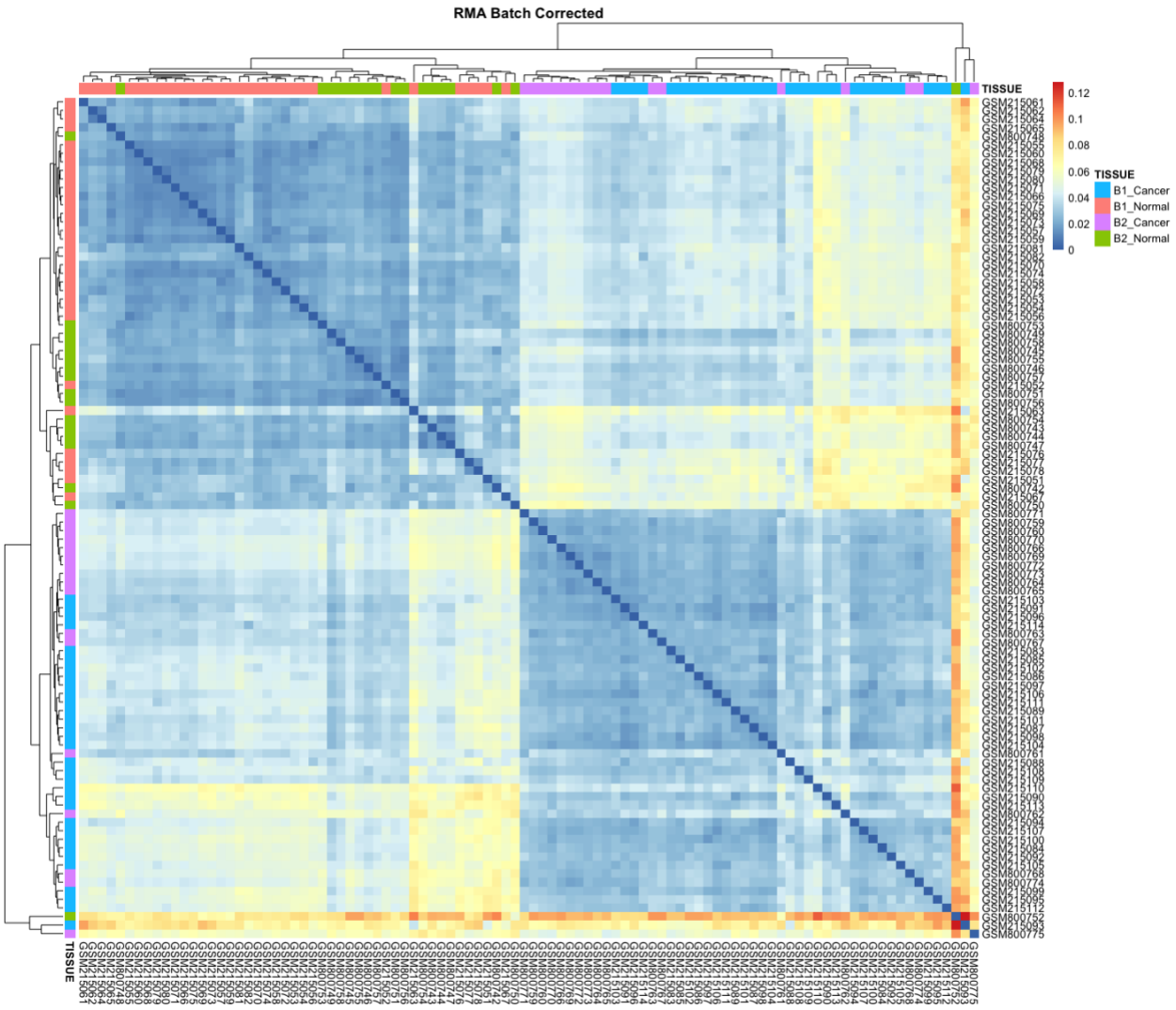
`annotation_row=[T]`	annotate rows
`annotation_col=[T]`	annotate columns

`[T]` should be the name of a data frame representation of your 4 groups with sample names as the row names. A portion of an example is shown below:

	TISSUE
GSM215051	B1_Normal
GSM215052	B1_Normal

* Since we are measuring dissimilarity, the data we are plotting will actually be 1-correlation.

Desired Output: Your plot should look similar to the figure below. **You should have the different colors representing 4 groups: Batch 1 Cancer, Batch 1 Normal, Batch 2 Cancer, Batch 2 Normal.** To do this, I added a column to my metadata. Colors should represent 2 groups if using 1 data set: Normal, Cancer.



Deliverable 3: 3 (or 2) visualizations for at least one of the methods above. These 3 (or 2) visualizations should represent your raw data, data after background correction and normalization, and data after batch correction. In these plots, you will find the importance of background correction, normalization, and batch correction

Deliverable 4: Determine which samples you will consider outliers, if any, and post your decision on the forums. This information will be used in Module 4. There will not

be a consensus on which samples are outliers since everyone has different criteria and allowances. This is okay!

All Deliverables:

1. **QC results.**
2. **Data visualization I. 1 visualization for each method using your batch corrected (or background corrected/normalized)**
    a. **Boxplots**
    b. **PCA (Principal Component Analysis)**
    c. **Correlation Heatmaps**
3. **Data visualization II. 3 (or 2) visualizations for at least one of the methods above. These 3 (or 2) visualizations should represent your raw data, data after background correction and normalization, and data after batch correction.**
4. **Identification of outliers as a post on the forums.**