Lab 1: Basic R syntax/plots with data solutions

Ryan Yancey (ryancey3)

For this lab, we will be using some basic data manipulation and plotting commands in R. We are working with a data set that is comparing the transcript profiles from peripheral B lymphocytes between patients with systemic lupus erythematosus (SLE) and normal healthy controls. The GEO summary of the data set is as follows:

> Systemic lupus erythematosus (SLE) is an autoimmune disease with an important clinical and biological heterogeneity. B lymphocytes appear central to the development of SLE which is characterized by the production of a large variety of autoantibodies and hypergammaglobulinemia. In mice, immature B cells from spontaneous lupus prone animals are able to produce autoantibodies when transferred into immunodeficient mice, strongly suggesting the existence of intrinsic B cell defects during lupus. In order to approach these defects in humans, we compared the peripheral B cell transcriptomes of quiescent lupus patients to normal B cell transcriptomes.

---

1) Go to class website under Course Documents > Data Sets and download the SLE B cell data set (from Garaud et al).

Done. Downloaded data set as **sle_b_cell.7z** in my working directory.

2) Unzip the text file, and read into R (Hint: using the `read.table()` function with a `header = T` argument and `row.names = 1` argument is one method to do this).

After downloading the file, we'll look at my directory to see what files are available to work with.

```
dir()
```

```
## [1] "ryancey3-gedav-lab1.Rmd" "sle_b_cell.7z"
```

We're interested in working with **sle_b_cell.7z** but we need to extract its contents first. We can do this by using the 7zip extractor, p7zip. I'm on a UNIX system, so I'll run the command in R as I would in my terminal, but modified to work within R.

```
# decompress sle_b_cell.7z
system(command = "7z -y e sle_b_cell.7z")

# view the new file in directory
dir()
```

```
## [1] "ryancey3-gedav-lab1.Rmd" "sle_b_cell.7z"
## [3] "sle_b_cell.txt"
```

Now, we can import **sle_b_cell.txt** into our environment.

```
# read data into R
garaud <- read.table(file = "sle_b_cell.txt",
                     header = TRUE,
                     row.names = 1)

# view the first few rows to preview data
head(x = garaud)
```

```
##              sle.1    sle.2    sle.3    sle.4    sle.5    sle.6    sle.7
## 1007_s_at 7.201054 6.946044 6.722417 6.478136 7.657807 7.458431 7.128307
## 1053_at   6.329305 6.525640 6.323421 5.788303 7.064163 6.475584 6.394097
## 117_at    4.943542 4.075428 5.173628 5.161637 5.879714 5.207273 4.860286
## 121_at    5.964546 5.959280 5.956043 5.891120 6.207989 6.155414 5.936915
## 1294_at   8.798413 9.027502 9.478432 9.069233 9.210925 9.060505 8.605030
## 1316_at   4.272086 3.659428 3.673992 3.822420 3.735879 4.164235 3.724233
##              sle.8    sle.9   sle.10   sle.11   sle.12   sle.13   sle.14
## 1007_s_at 6.893233 7.417972 6.931443 7.043239 6.477673 7.257562 7.196866
## 1053_at   6.708787 6.457130 6.407173 6.124894 6.181405 6.471388 6.504525
## 117_at    4.589202 5.545966 5.397000 5.204737 5.510464 5.551032 4.594924
## 121_at    5.647587 6.175860 5.905099 5.619636 5.917675 6.130143 5.899389
## 1294_at   9.088856 9.002307 9.386652 9.434617 9.329697 8.469365 8.794065
## 1316_at   3.669170 4.107480 3.806881 3.715541 3.825745 3.875756 3.803949
##             sle.15   sle.16   sle.17 control.1 control.2 control.3 control.4
## 1007_s_at 7.215563 6.975056 6.946701  7.001722  7.674048  7.207480  7.237135
## 1053_at   6.638673 6.619250 6.427475  6.417270  6.583484  6.304779  6.631808
## 117_at    4.974176 4.452682 4.879310  6.204755  5.462083  5.162292  5.182004
## 121_at    6.665831 6.567842 5.931885  5.819153  5.634927  6.063086  5.618122
## 1294_at   9.255156 9.297987 8.527558  9.156431  9.028355  9.878348  8.880848
## 1316_at   3.870495 3.978532 4.068268  3.889457  3.725608  4.024489  3.778687
##           control.5 control.6 control.7 control.8 control.9
## 1007_s_at  6.819833  7.261240  6.934885  7.003605  7.275157
## 1053_at    6.496153  6.584563  6.376337  6.355599  6.419475
## 117_at     5.214210  4.657415  5.199447  5.796467  4.687595
## 121_at     5.627996  6.050202  5.744575  6.366819  5.910187
## 1294_at    9.025513  8.503446  8.619609  9.285734  8.710406
## 1316_at    3.826248  3.826856  3.811117  3.990947  3.830298
```

Each row appears to be a probeset in the microarray, and each column appears to be a sample from the experiment.

3) Look at the dimensions of the data There should be 26 samples. If you have 27 samples, you still have the row names in the first data column, so retry 2 to set the row names to these.

```
# view number of rows (probesets), columns (samples)
dim(garaud)
```

```
## [1] 34853    26
```

There's 34,853 probesets and 26 samples.

4) Print the sample names to screen.

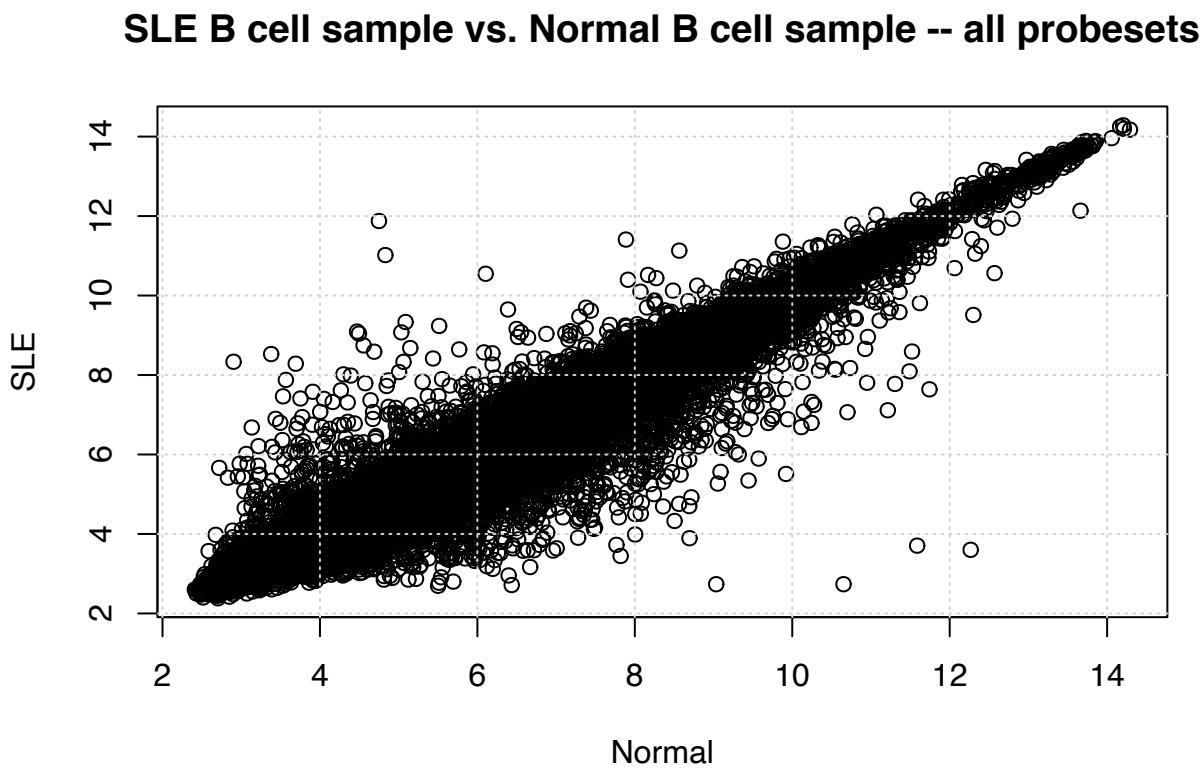
```
# samples are columns so we will view those
colnames(garaud)
```

```
##  [1] "sle.1"     "sle.2"     "sle.3"     "sle.4"     "sle.5"     "sle.6"
##  [7] "sle.7"     "sle.8"     "sle.9"     "sle.10"    "sle.11"    "sle.12"
## [13] "sle.13"    "sle.14"    "sle.15"    "sle.16"    "sle.17"    "control.1"
## [19] "control.2" "control.3" "control.4" "control.5" "control.6" "control.7"
## [25] "control.8" "control.9"
```

Of the 26 samples, the first 17 are SLE samples, and the remaining 9 are control samples

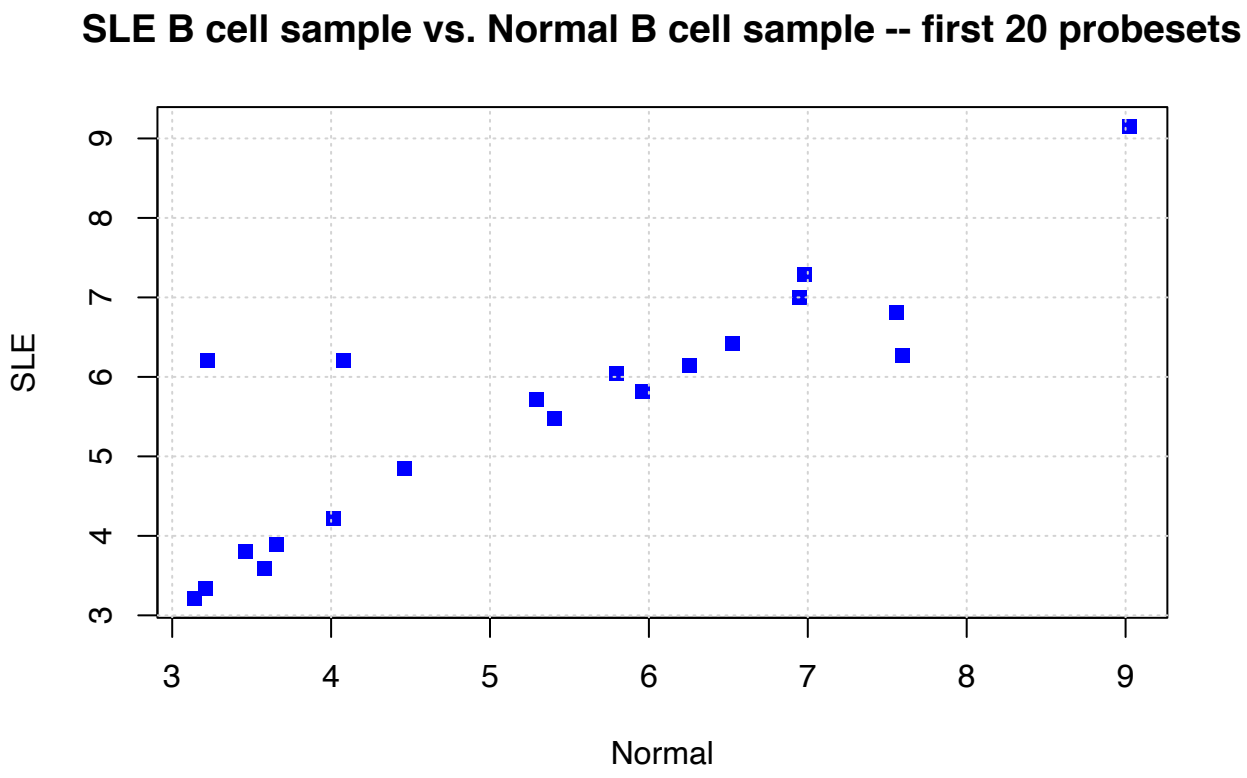
5) Plot the second SLE patient sample versus the first normal control samples in an xy scatter plot. Remember that the first argument is the x vector. Label the x and y-axes as 'Normal' and 'SLE', respectively. Title the plot, 'SLE B cell sample vs. Normal B cell sample – all probesets'. Add grey grid lines with the function `grid()`.

```
# plot probesets
plot(
    x = garaud[, "sle.2"],
    y = garaud[, "control.1"],
    xlab = "Normal",
    ylab = "SLE",
    main = "SLE B cell sample vs. Normal B cell sample -- all probesets"
)

# add grid to plot
grid()
```



6.) Now do the same plot but pick only the first 20 probesets. Use the `pch = 15` argument to change the shape and color the points blue with the `col` argument.

```
# plot probesets
plot(
    # subset rows 1 thru 20
    x = garaud[1:20, "sle.2"],
    y = garaud[1:20, "control.1"],
    xlab = "Normal",
    ylab = "SLE",
    # update title to reflect subset
    main = "SLE B cell sample vs. Normal B cell sample -- first 20 probesets",
    # change shape and color of points
    pch = 15,
    col = "blue"
)

# add grid to plot
grid()
```

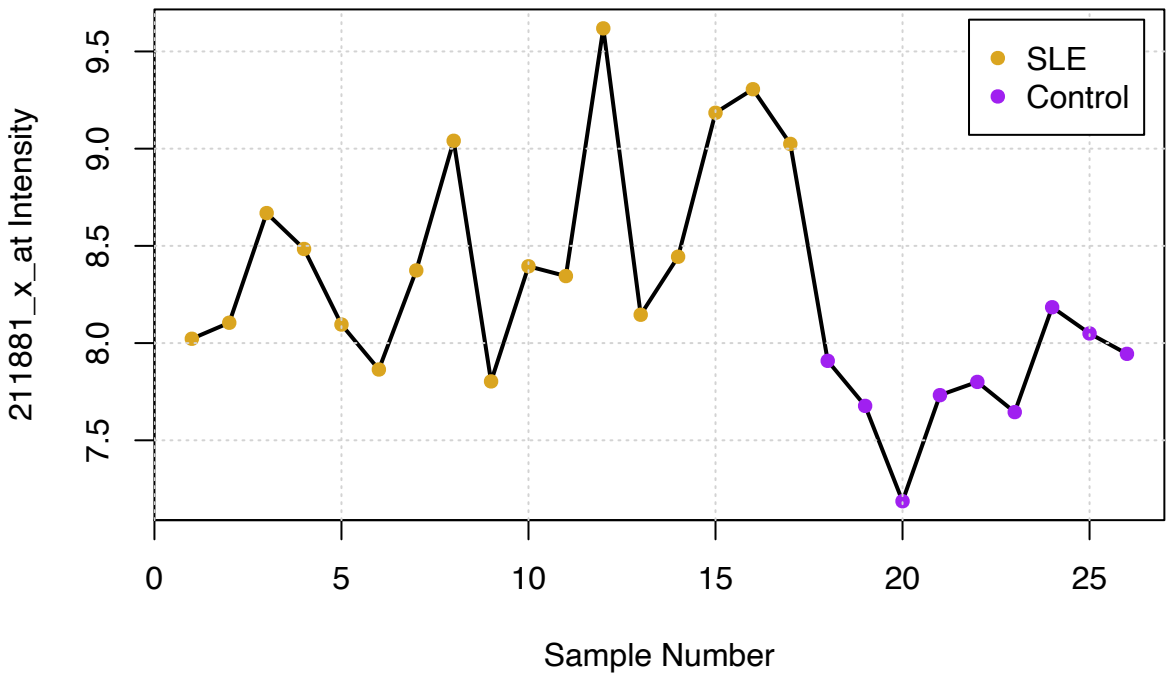


7.) Now plot the following gene in a gene profile plot, IGLJ3 (immunoglobulin lambda joining 3), which is probeset ID 211881_x_at. This type of plot has the sample indices across the x-axis and the intensities on the y-axis, so you can see a profile of the gene across experiments or arrays. First plot the ranges using the `type="n"` argument and the `plot()` function, then add the genes with the `lines()` function call. Add grid lines. Hint: to plot just ranges of x and y vectors, use the `range()` function like so:

```
plot(range(1:26),range(dat[geneX,]),...
```

Be sure to cast the gene vector to numeric before plotting.

```r
# subset the rows based on the probeset ID
IGJL3_allsamples <- as.numeric(garaud["211881_x_at", ])

# plot an empty canvas with labels
plot(
    x = range(1:26),
    y = range(IGJL3_allsamples),
    type = "n", # empty plot
    xlab = "Sample Number",
    ylab = "211881_x_at Intensity",
    main = "Gene profile plot of IGJL3 (211881_x_at) across all samples"
)

# add line to plot
lines(x = IGJL3_allsamples, lwd = 2)

# add points and color based on sample type (17: SLE, 9: control)
points(
    x = IGJL3_allsamples,
    pch = 16,
    col = c(rep("goldenrod", 17), rep("purple", 9))
)

# add grid to plot
grid()

# add figure legend for clarity
legend(
    "topright",
    bg = "white",
    inset = 0.02,
    legend = c("SLE", "Control"),
    col = c("goldenrod", "purple"),
    pch = 16
)
```


Gene profile plot of IGJL3 (211881_x_at) across all samples
211881_x_at Intensity
7.5
8.0
8.5
9.0
9.5
0
5
10
15
20
25
Sample Number
SLE
Control

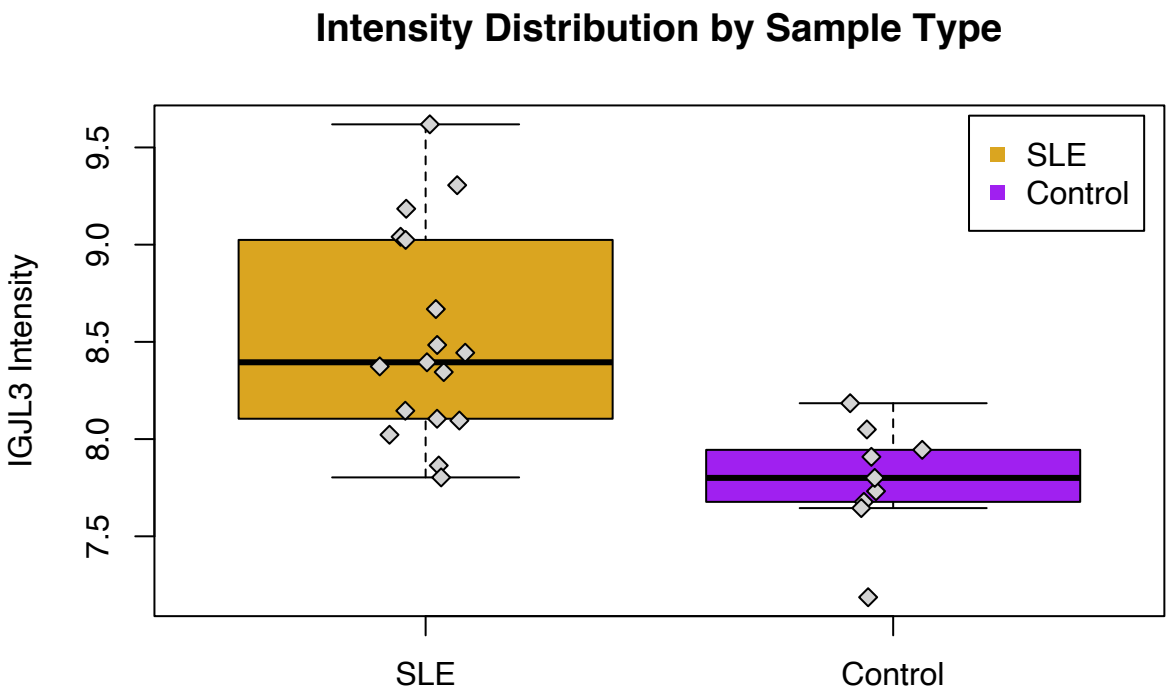
8.) Finally, another way to visualize a gene profile across conditions is to graph a boxplot with a single distribution box per condition. To do this, we need to create a factor vector that indicates the disease or normal condition like so:

```
f <- c(rep("SLE",17),rep("Control",9))
```

Then use this vector with the expression vector for IGLJ3 in the `boxplot()` function to create the graph.

Not required, but you can increase the plot info by using the `with()` function and `stripchart()` function to add points.

```
 # factor vector for samples
samples <- factor(c(rep("SLE", 17), rep("Control", 9)), levels = c("SLE", "Control"))

# create a data frame for the IGJL3 subset with factors included
df.8 <- data.frame(Sample = samples, IGJL3 = IGJL3_allsamples)

# makes jitter from stripchart reproducible
set.seed(1234)

# plot boxplot and add stripchart on top
with(df.8, {
    boxplot(
        # formula: y ~ group
        IGJL3 ~ Sample,
        col = c("goldenrod", "purple"),
        xlab = "", # remove x-axis label
        ylab = "IGJL3 Intensity",
        outcol = "white", # hides duplicate points (outliers + stripchart)
        main = "Intensity Distribution by Sample Type"
    )
    # add stripchart on top
    stripchart(
        IGJL3 ~ Sample,
        method = "jitter",
        pch = 23,
        bg = "lightgray",
        vertical = TRUE,
        add = TRUE
    )
    # add legend for clarity
    legend(
      "topright",
      inset = 0.02,
      legend = c("SLE", "Control"),
      col = c("goldenrod", "purple"),
      pch = 15
    )
})
```


Intensity Distribution by Sample Type
IGJL3 Intensity
9.5
9.0
8.5
8.0
7.5
SLE
Control
SLE
Control

Session info

```
sessionInfo()
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.1.0    magrittr_2.0.1    tools_4.1.0       htmltools_0.5.1.1
##  [5] yaml_2.2.1        stringi_1.6.2     rmarkdown_2.8     knitr_1.33
##  [9] stringr_1.4.0     xfun_0.23         digest_0.6.27     rlang_0.4.11
## [13] evaluate_0.14
```