Differential Expression (DE) Analysis with Proteus

Jessica Randall

Contents

Briefly, Proteus provides users with tools to build both 2-d and 3-D interactive plots and other summaries of their proteomics output (proteinGroups.txt) from MaxQuant and implements the linear models and empirical Bayes estimation methods from limma (Linear Models for Microarray Data) to determine differential expression between comparisons of two or more groups.

These methods work for protemics data because they account for small sample sizes often found in these types of projects and the potential effects of outlier proteins or samples on estimates of variance within and between groups. One protein or one sample could have the potential to skew results but the empirical Bayes test moderates these effects resulting in improved sensitivity and specificty over popular tools like Perseus.

Linked is the original 2018 Proteus paper from Gierlinski, Gastaldello, and Barton introducing the concepts implemented in Proteus.

Here is the original link to the 2015 limma paper from Ritchie, Phipson, Wu, Hu, Law, Shi, and Smyth. There are also several additional papers describing methods implemented in limma which include the a 2016 paper describing the differential expression methodologies from Phipson, Lee, Majewski, Alexander, and Smyth, a 2014 article describing methods behind the lmFit function described below from Law, Chen, Shi, and Smyth, and a 2015 article from Liu, Holik, Su, Jansz, Chen, Leong, Blewitt, Asselin-Labat, Smyth, and Ritchie describing the eBayes function also implemented below.

You may recognize some of the names of the authors on limma papers as authors on related tools like edgeR for differential expression of RNASeq data and GOSeq specifically for GO term analysis.

We will be using example data from the Proteus Label Free package. These data come from an unpublished project by the Barton Group examining the proteomics of yeast with 2 experimental groups with seven replicates in each group.

Proteus has wide range of capabilities for proteomics data. Our simplified example describes how to read in data from the protein groups file from MaxQuant, and perform and visualize differentially expressed proteins with a volcano plot. We strongly encourage you to reach out to EICC with questions regarding more advanced options available to you with Proteus. Check out some of the graphs from previous projects here.

Definition of terms

0.0.0.1 Empirical Bayes test:

The empirical Bayes test works well for protemics data because it accounts for small sample sizes often found in these types of projects and the potential effects of outlier proteins or samples on estimates of variance within and between groups. One protein or one sample has the potential to skew results but the empirical Bayes test moderates these effects resulting in improved sensitivity and specificty over popular tools like Perseus.

Compared to Perseus it has been our experience that Proteus is less stringent in its calling of proteins as significantly DE. If your study is exploratory and you are not sure which or how many DE proteins you are expecting to find, Proteus may be better suited to your project goals.

0.0.0.2 Imputation

One distinct feature of Proteus is its implementation of imputation methods. Since proteomics data often has a high proportion of missing values, the choice of how to impute those values becomes crucial to accurate estimation of differences within and between groups of samples. Proteus estimates missing values by using values in the existing data rather than drawing random values from the normal distribution as is implemented in Perseus.

0.0.0.3 Unadjusted p values vs adjusted p-values/(FDR):

In DE analysis, a single p-value tells you how likely it is that a single protein is differential expressed between at least two groups (ex: a control and a treatment group) due to some actual difference between the groups as opposed to random chance. False Discovery Rate (FDR) tells you how likely it is that all proteins identified as DE are false positives. A FDR of 5% means that among all proteins called DE, an average of 5% of those are truly not DE. DE proteins are only considered significantly so if they meet the adjusted p value, not only the unadjusted p-value. FDRs for each individual protein are called q-values or local FDRs.

Loading data

Our very first step is to load the libraries required for analysis and graphing. Please see Bioconductor for information about initial installation and use of Bioconductor and its packages. We also set the minimal theme in R's gglot2 package for all graphs to have the same aesthetic features by default.

Here we will demonstrate importing the protein group data from the proteusLabelFree package.

We also use the here package to specify relative file paths rather than absolute file paths, the janitor package for cleaning up data formatting, the RColorBrewer package for custom color palettes, and prefer a tidy syntax from the tidyverse R package collective.

Please reach out to EICC if you would like to compare 3 or more groups as this is a simplified example. It may also be the case you will need more than 7 samples per experimental group or that you may need to adjust filtering criteria. Please see our PROPER walk-through for an example of our of power and sample size analysis for RNASeq data as an example of considerations we take into account for those types of experiments.

```
pacman::p_load("readr", "here", "knitr", "janitor",
               "tidyverse", "proteus", "EnhancedVolcano")

files <- list(metadata =
                here("ProteomicsMethods/Proteus/input/metadata.txt"),
              pgroups =
                here("ProteomicsMethods/Proteus/input/proteinGroups.txt"))

meta <- read_tsv(files$metadata)

# import data and filter out reversed and contaminant proteins
prodat <- readProteinGroups(files$pgroups, meta,
                            measure.cols =
                              setNames(paste("Intensity",
                                             meta$experiment),
                                       meta$experiment))
```

Annotation

Annotate with UniProt being careful to choose the right delimiter to parse the protein names correctly. Since We're using proteus example data, this was taken directly from the proteus vignette but in cases where names are delimited with other characters like semicolons or commas, this process will look a bit different. Choosing how to parse the names of your list of proteins will take some trial and error. Once we have a list of protein names we can remove duplicates that result from names changing overtime as discoveries are made in the field.

It may also be the case that protein IDs are too long to be legible on your graphs later on so you have the option of annotating with gene names at this step instead.

Getting annotation from UniProt with fetchUniProt can take a long time depending on the size of your dataset and your computing speed but it will tell you when it's done.

```
luni <- lapply(as.character(prodat$proteins), function(prot) {
 if(grepl("sp\\|", prot)) {
   uniprot <- unlist(strsplit(prot, "|", fixed=TRUE))[2]
   c(prot, uniprot)
 }
})
ids <- as.data.frame(do.call(rbind, luni))
names(ids) <- c("protein", "uniprot")

annotations <- fetchFromUniProt(ids$uniprot, verbose=TRUE)
```

```
## Batch 1 out of 10
## Batch 2 out of 10
## Batch 3 out of 10
## Batch 4 out of 10
## Batch 5 out of 10
## Batch 6 out of 10
## Batch 7 out of 10
## Batch 8 out of 10
## Batch 9 out of 10
## Batch 10 out of 10
```

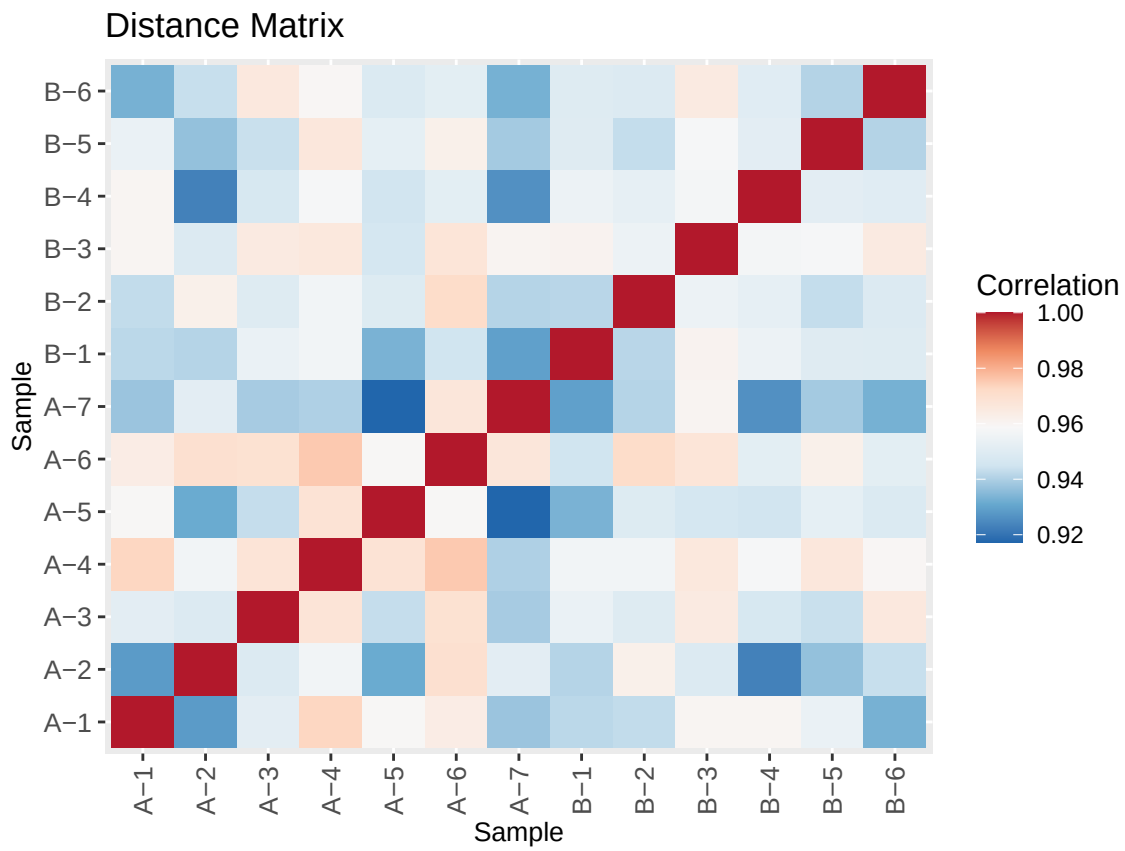
```
# remove duplicate IDs

ann_id <- merge(ids, annotations, by.x="uniprot", by.y="id")
ann_id<- unique(ann_id)

# annotate proteus object with protein IDs

prodat_ann <- annotateProteins(prodat, ann_id)
```

```
## Annotated 3744 proteins.
```

Exploratory Visualizations

To plot the Pearson coorelations of each sample compared to other samples in the dataset we use a distance matrix. Here we have customized the plotDM function in proteus to use a preferred color scheme and increase the font size and called it plotDM_custom. These are small cosmetic changes we can accomodate if you have different preferences in color scheme, titles, or other labels.

```
plotDM_custom(prodat_ann)
```

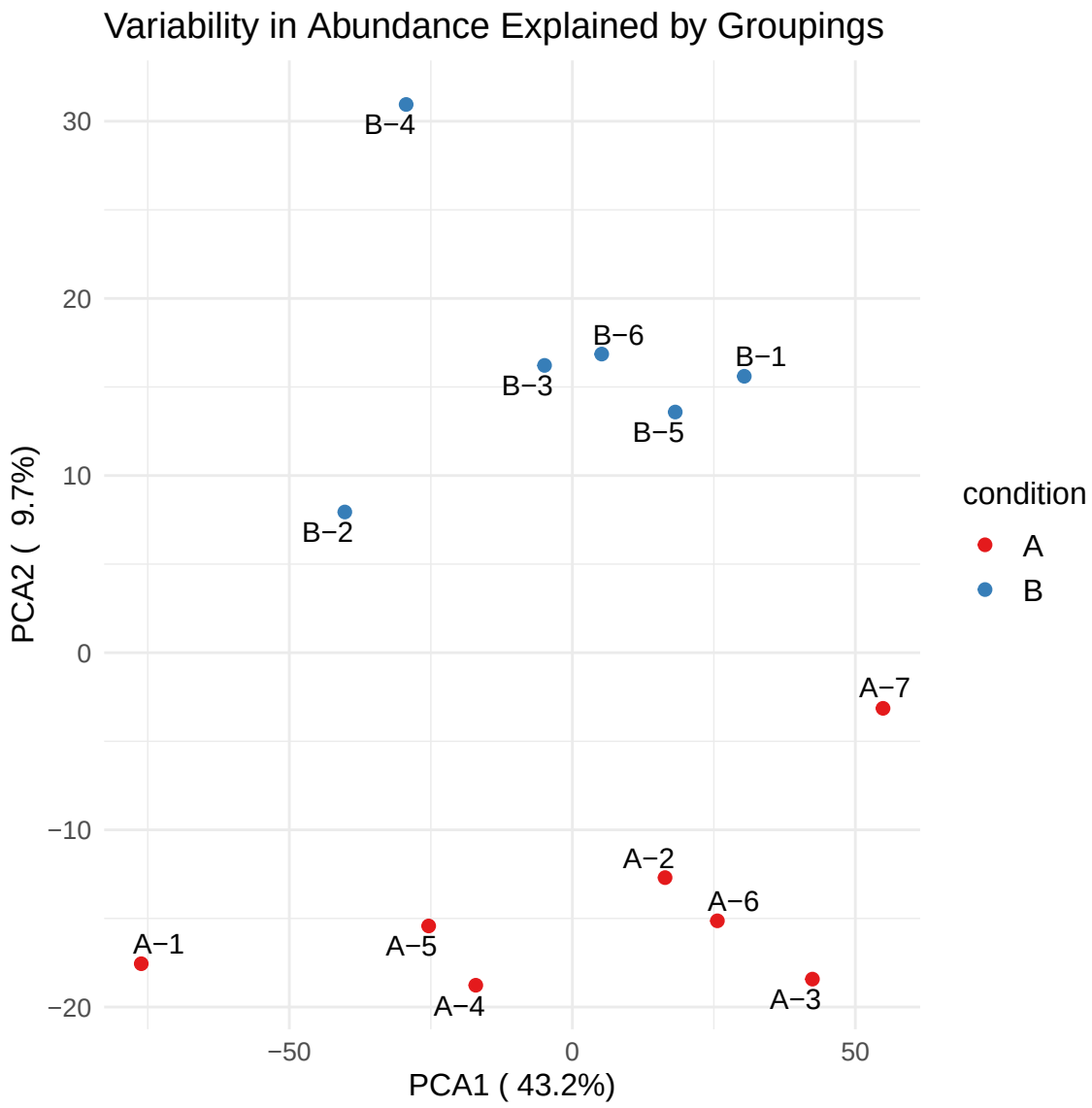


This graph tells us how correlated each sample is with other samples. The darker red, the more correlated and the darker blue, the less correlated. Samples will always be most highly correlated with themselves (darkest red) and we would expect to see higher correlations within experimental groups than we would expect to see between them. This is not the case in this example data set but is more clearly visible in very clean data with truly biologically different groups.

If we had any additional data to add about the samples that we wanted to include in our analysis we would add it next but since this is a simplified example, we are only comparing treated and control samples without taking into account any additional information about them.

At this point we generate our second exploratory visualization, the principal components analysis (or, PCA) plot. This will show us how your data cluster or how similar each sample is to others of the same group.

Here we have also customized the plotPCA function in proteus to use a preferred color scheme, add a title, change legend labels, and increase the font size, we call it plotPCA_custom.



This plot shows that our samples are somewhat clustering together and we would interpret the x-axis as saying that 43% of the variability between these samples is due to some true difference in expression between the groups and a similarity in expression between samples of the same group.

The y-axis tells us how much variability between these samples is due to other factors in our model or if we have none, sources of variability we may not have accounted for like sex or ethnicity which are often leading contributors of variability between samples and should be accounted for in experimental design if you wish to control for their effects.

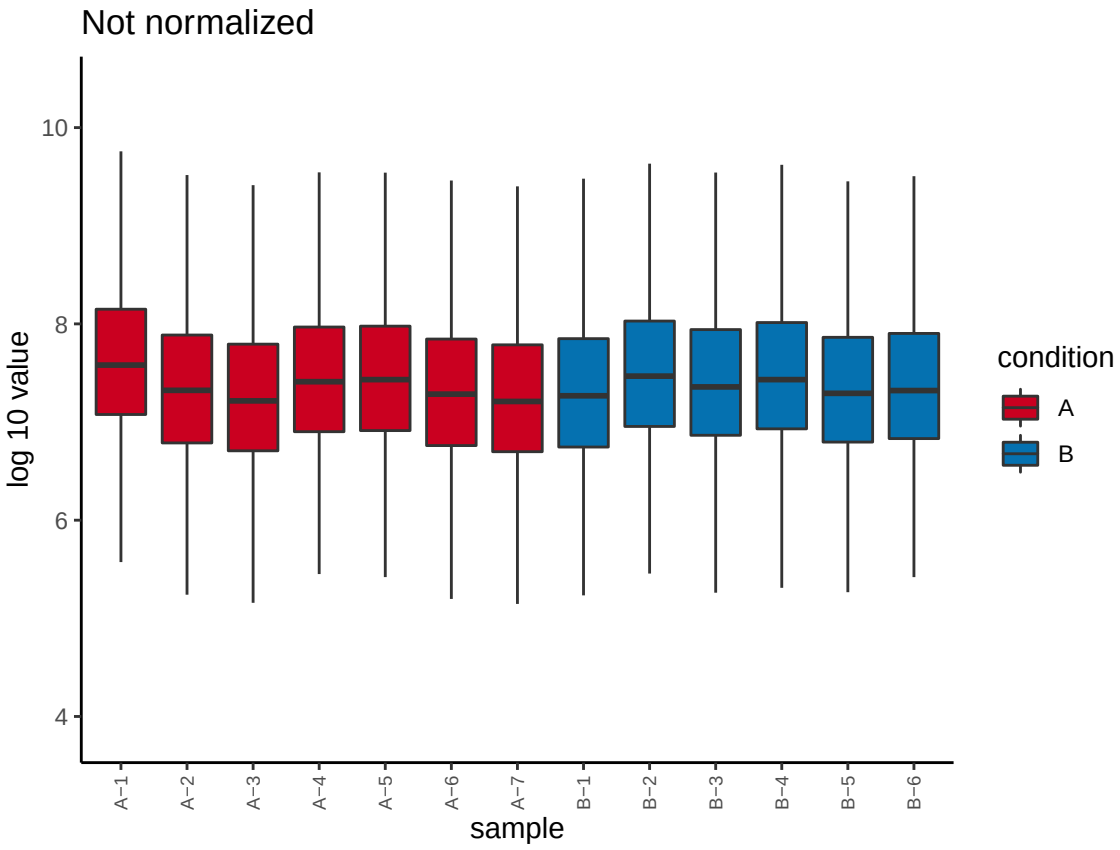
Normalizing data

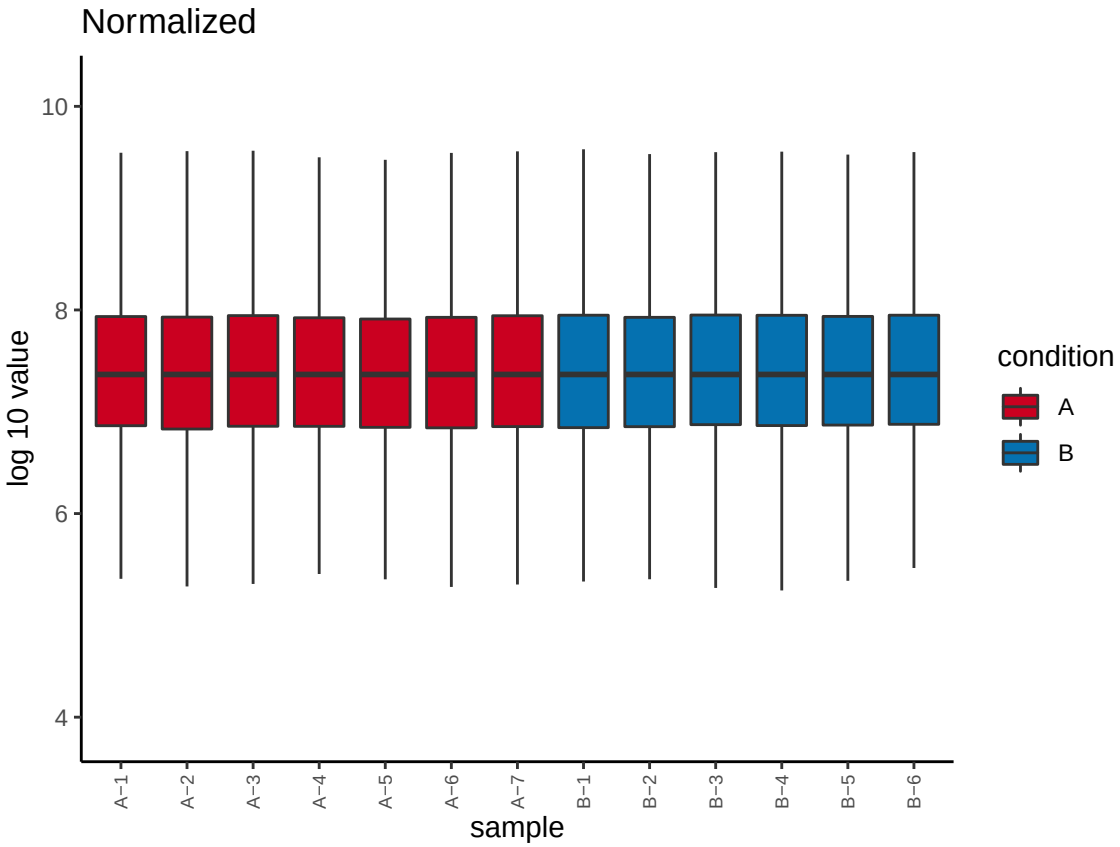
There are a number of reasons we may want to normalize data in Proteus and the normalizeData functions offer the options to normalize by the median or by quantiles as desired. The default method in the function is median normalization rather than mean normalization to mitigate the effects of very small or very large values. This means that median intensity in every sample will be the same.

We normalize using this method because we assume the data we have collected are a sample from a population of possible protein expression values which follow a normal or bell-curve shaped distribution around the

median. We normalize to make our data better match the shape of this curve rather than findings being driven by very high or very low values.

We can see the effects of median normalization in the two graphs below, the first representing the log10 transformed values of the un-normalized data and the second showing log10 transformed values of the median-normalized data.





Differential Expression Setup and Testing

At this point we have prepared all of the data to be input into our linear model. In this simplified example we are only making 1 pairwise comparison between two groups. While Proteus is designed to handle these types of analyses, the authors advise that for more complicated study designs than this, that limma be used directly.

Please contact EICC if would like to discuss your experimental goals so we can assist you in choosing the appropriate analysis tool.

Recall that the hypothesis we are testing is that there is no difference in protein expression between the treatment and control samples. Here we use the limmaDE function in Proteus to examine that hypothesis with our FDR set at 0.05. Below are the 10 proteins with the lowest FDRs in our results.

```
res <- limmaDE(prodat_norm, sig.level=0.05)
```

Results

```
##  [1] "sp|P19145|GAP1_YEAST"
##  [2] "sp|P40472|SIM1_YEAST"
##  [3] "sp|Q02516|HAP5_YEAST"
##  [4] "sp|P0CI39|STE2_YEASX;sp|D6VTK4|STE2_YEAST"
##  [5] "sp|P09435|HSP73_YEAST"
##  [6] "sp|P22202|HSP74_YEAST"
##  [7] "sp|P26263|PDC6_YEAST"
```

```
##  [8] "sp|Q03178|PIR1_YEAST;sp|B3LQU0|PIR1_YEAS1;sp|A6ZZG0|PIR1_YEAS7"
##  [9] "sp|Q12298|YD061_YEAST"
## [10] "sp|P40893|REE1_YEAST"
```

In this example, while controlling the adjusted p/FDR threshold at <0.05, we have 22 DE proteins between these groups, 12 are more expressed in the A samples or up-regulated and 10 are more expressed in the B samples or down-regulated.Up and down regulation refers to the group you have set as the control or reference group. In this case, we are comparing the B samples to the A samples so the B samples are our reference group.

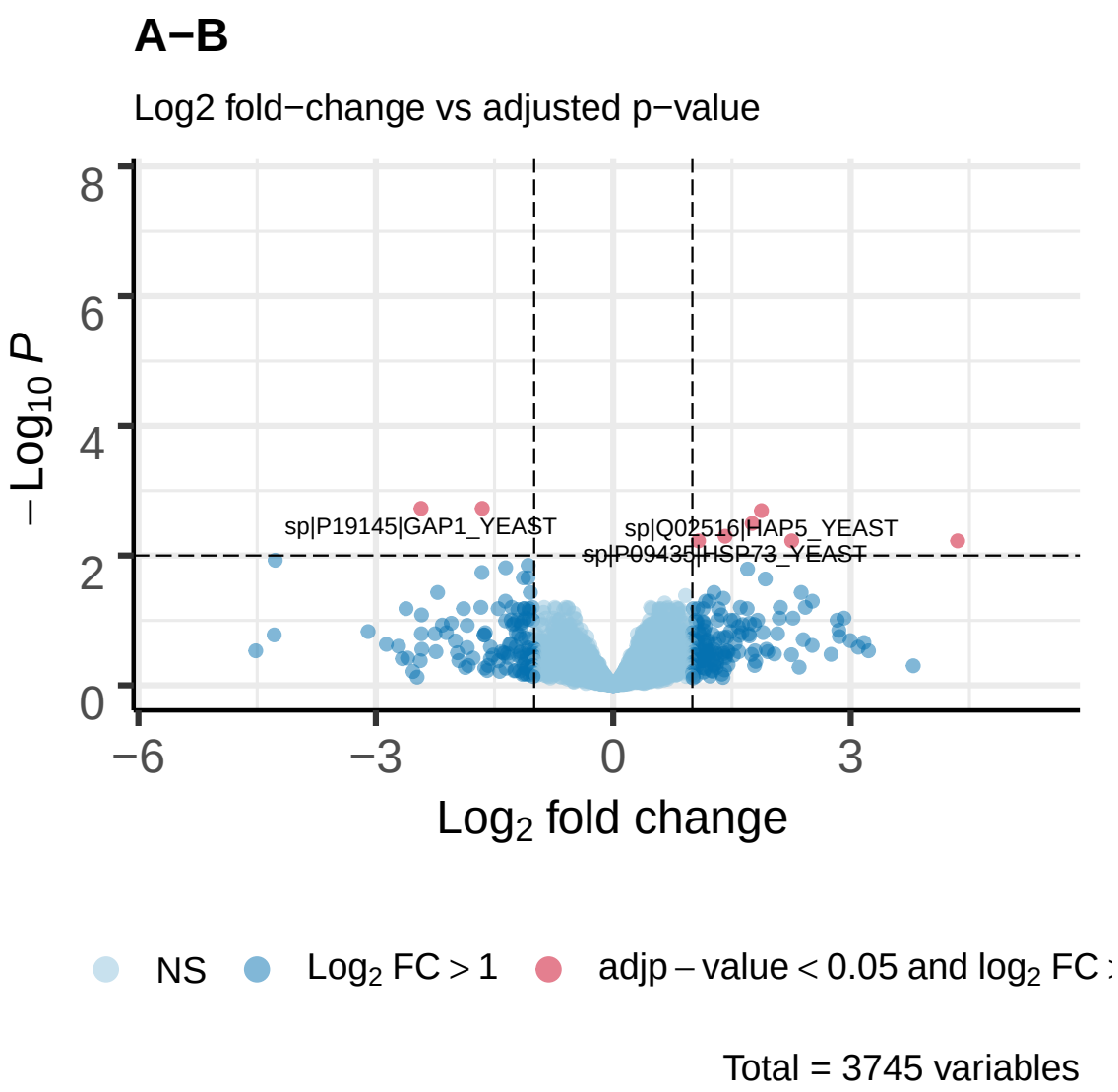
FDR thresholds can also range from 0.05-0.2 and it is much better to have the option to lower your threshold during analysis rather than have to increase it because your study was not sufficiently powered to find anything but the most extremely highly or lowly differentially expressed proteins. In proteomics experiments it is also common to report the nominal p-value instead of the FDR. Please reach out to EICC if you would like assistance in planning your experimental design for your proteomics project and in setting appropriate p-value or FDR thresholds.

Visualizing Results

We now generate a common data visualization, a volcano plot. Typically we provide a distance matrix, PCA plot, and volcano plots with proteomics projects but we would be happy to work with you to customize these and additional visualizations for publication. Please see our Data Visualization menu for examples from previous projects.

A volcano plot is a scatter plot where the x-axis has the log2 transformed fold changes between the compared samples and the y axis has the nominal or adjusted p-values for each protein. Here we have also labelled the proteins with FDR < 0.05 as that is the threshold we set for our results.

Using the EnhancedVolcano package with a custom color palette from ColorBrewer2, the points in red are those which meet the threshold for statistical significance with an adjusted p value less than or equal to 0.05 and a log2 fold change of 1.0 or greater. Points in dark blue are those with only log2 fold changes >1.0 and those in light blue are not statistically significant nor have a log2 fold change > 1.0. All of these parameters can be adjusted based on cutoffs and thresholds relevant to your data.



There are many more capabilities and functions available in Proteus than are used here in order to show a simplified example of one of the tools we use for differential expression analysis for proteomics data. Obtaining specific, actionable, and publication quality results from analysis requires a deeper understanding of your specific data set and we would love the opportunity to discuss these options with you.

While we encourage clients to reach out prior to sequencing so that we can collaborate to design the experiment to answer your specific questions, we look forward to hearing from you at any stage of your proteomics project. Please find our contact information available on our website and check out some of the graphs we've made for previous clients here.

References and Session information

Hadley Wickham, Jim Hester and Romain Francois (2018). readr: Read Rectangular Text Data. R package version 1.3.1. https://CRAN.R-project.org/package=readr

Kevin Blighe, Sharmila Rana and Myles Lewis (2020). EnhancedVolcano: Publication-ready volcano plots with enhanced colouring and labeling. R package version 1.6.0.https://github.com/kevinblighe/EnhancedVolcano

Marek Gierlinski (2020). proteus: Downstream analysis of the MaxQuant output. R package version 0.2.14.https://github.com/bartongroup/Proteus

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

Yihui Xie (2014) knitr: A Comprehensive Tool for Reproducible Research in R. In Victoria Stodden, Friedrich Leisch and Roger D. Peng, editors, Implementing Reproducible Computational Research. Chapman and Hall/CRC. ISBN 978-1466561595

Yihui Xie (2015) Dynamic Documents with R and knitr. 2nd edition. Chapman and Hall/CRC. ISBN 978-1498716963

Yihui Xie (2016). bookdown: Authoring Books and Technical Documents with R Markdown. Chapman and Hall/CRC. ISBN 978-1138700109

Yihui Xie (2020). knitr: A General-Purpose Package for Dynamic Report Generation in R. R package version 1.28.

```
## [1] "Thu May 28 12:44:51 2020"
```

```
## R version 4.0.0 (2020-04-24)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Debian GNU/Linux 10 (buster)
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/libopenblasp-r0.3.5.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] EnhancedVolcano_1.6.0 ggrepel_0.8.2         proteus_0.2.14
##  [4] forcats_0.5.0         stringr_1.4.0         dplyr_0.8.5
##  [7] purrr_0.3.4           tidyr_1.0.2           tibble_3.0.1
## [10] ggplot2_3.3.0         tidyverse_1.3.0       janitor_2.0.1
## [13] knitr_1.28            here_0.1              readr_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] viridis_0.5.1      httr_1.4.1         jsonlite_1.6.1     viridisLite_0.3.0
##  [5] modelr_0.1.7       shiny_1.4.0.2      assertthat_0.2.1   cellranger_1.1.0
##  [9] yaml_2.2.1         pillar_1.4.4       backports_1.1.7    lattice_0.20-41
## [13] limma_3.44.1       glue_1.4.1         digest_0.6.25      RColorBrewer_1.1-2
## [17] promises_1.1.0     rvest_0.3.5        snakecase_0.11.0   colorspace_1.4-1
## [21] htmltools_0.4.0    httpuv_1.5.3.1     plyr_1.8.6         pkgconfig_2.0.3
## [25] broom_0.5.6        haven_2.2.0        bookdown_0.18      xtable_1.8-4
## [29] scales_1.1.1       later_1.0.0        generics_0.0.2     farver_2.0.3
```

```
## [33] ellipsis_0.3.1     pacman_0.5.1       withr_2.2.0        cli_2.0.2
## [37] magrittr_1.5       crayon_1.3.4       readxl_1.3.1       mime_0.9
## [41] evaluate_0.14      fs_1.4.1           fansi_0.4.1        nlme_3.1-147
## [45] xml2_1.3.2         tools_4.0.0        hms_0.5.3          lifecycle_0.2.0
## [49] munsell_0.5.0      reprex_0.3.0       compiler_4.0.0     rlang_0.4.6
## [53] grid_4.0.0         rstudioapi_0.11    labeling_0.3       rmarkdown_2.1
## [57] gtable_0.3.0       DBI_1.1.0          reshape2_1.4.4     R6_2.4.1
## [61] gridExtra_2.3      lubridate_1.7.8    fastmap_1.0.1      rprojroot_1.3-2
## [65] stringi_1.4.6      parallel_4.0.0     Rcpp_1.0.4.6       vctrs_0.3.0
## [69] dbplyr_1.4.3       tidyselect_1.1.0   xfun_0.13
```