Introduction to PepSeq

Derek Sonderegger

2018-12-18

PepSeq is a package developed to facilitate the exploration of PMI’s PepSeq data. Given a pulldown, we want to explore how the cleaved-uncleaved signal varies across different locations in the proteins.

To begin, we will install the package via the following R commands:

library(devtools)  
install\_github('dereksonderegger/PepSeq') # install latest version of Derek's PepSeq tools

library(tidyverse) # load the usual set of data processing tools  
library(PepSeq) # load Derek's library of routines

The first step is to set the working directory to wherever the pulldown .csv file is located. This can be done via the point and click interface (Session -> Set Working Directory) or using the command setwd. Once the working directory is set to where ever the data file lives, you can define the input and output files.

setwd('~/GitHub/PepSeq/inst/extdata')  
input\_file <- 'PepSeq\_vs\_NN.csv'  
output\_file <- 'PepSeq\_vs\_NN.pdf'

Alternatively, you could just list the full pathname to the input and output files and ignore the working directory business.

input\_file <- '~/GitHub/PepSeq/inst/extdata/PepSeq\_vs\_NN.csv'  
output\_file <- '~/GitHub/PepSeq/inst/extdata/PepSeq\_vs\_NN.pdf'

PepSeq assumes that a pulldown .csv file is arrange in a format with one protein/location combination per line and then several sets of cleaved/uncleaved pairs and possibly columns of data that aren’t cleaved/uncleaved but rather just some other signal (e.g. output from another model as to the likelihood of binding).

read.csv(input\_file) %>% colnames()  
#> [1] "Protein\_ID" "Start\_Loc" "wTEV\_Cleaved"   
#> [4] "wTEV\_Uncleaved" "ModelPrediction" "Experimental\_Class"

Notice that in the example pulldown file, there is just one experiment (wTEV) and two other added columns model of neural net model predictions and if the number of literature citations. The orderings of all the columns doesn’t matter, but you need to be able to identify the protein and position columns and each of the cleaved/uncleaved pairs need to have the same prefix followed by an ’\_Cleaved’ or ’\_Uncleaved’. If there is no ‘Cleaved’ or ‘Uncleaved’ suffix on a column then we assume the column is a measurement and will be data to be visualized.

In order to allow there to be many other interesting columns that should be ignored in the visualization, PepSeq requires you to indicate which columns are to be plotted by either having an initial suffix indicating it is data or to indicate the responses by column location (e.g. columns 3 to 6).

# There is a function to import from a data file, either an Excel file or .csv  
data\_input <- import\_pulldown(  
 input\_file,   
 standardization\_method = 'additive',  
 protein\_column = 'Protein\_ID',   
 position\_column = 'Start\_Loc',  
 read\_indicator = 3:6)

data\_input <- import\_pulldown(  
 file = input\_file, # input file  
 standardization\_method = 'additive',  
 protein\_column = 'Protein\_ID', # which is the protein name  
 position\_column = 'Start\_Loc', # which is the starting location  
 read\_indicator = c(3,4,5,6)) # column range to be visualized

Notice that in each of the calls, we also had to specify which column specified the protein name and which column denotes the position along the protein.

Once the data has been read in, it can then be passed to the visualization program. In the visualization program, you can control the final height and width of the resulting figure as well as the maximum and minimum values for the y-axis. By including a peaks=TRUE option, the peaks will be highlighted. The default peak fitting procedure is ‘Peaks over Threshold’ and requires a theshold parameter, which can be set via the peak\_param option.

plot\_pulldown(   
 data\_input, # input data  
 output\_file = output\_file, # output file name  
 peaks = TRUE, # Show the peaks  
 peak\_method = 'PoT', # Use "Peaks over Threshold"  
 peak\_param = c(.5, 90, 10) ) # thresholds are by row

Finally if we want to look at the interactive Shiny version, all the options are the same, but we don’t need the output\_file.

plot\_pulldown\_Shiny(   
 data\_input, # input data  
 peaks = TRUE, # Show the peaks  
 peak\_method = 'PoT', # Use "Peaks over Threshold"  
 peak\_param = c(.5, 90, 10) ) # thresholds are by row

Notice in the Shiny application there are options to rescale the min and max of the y-axis, the window width (to squish or stretch the data points apart), as well as to select which rows are shown. While it is possible to resize the window and have the x-axis expand with the window, the same is frustratingly not possible for the y-axis. To get around this, I’ve added a height option that can be set to create a larger or smaller graph.

plot\_pulldown\_Shiny(   
 data\_input, # input data  
 peaks = TRUE, # Show the peaks  
 peak\_method = 'PoT', # Use "Peaks over Threshold"  
 peak\_param = c(.5, 90, 10), # thresholds are by row   
 height = 500 ) # The graph height (in pixels)

A similar trick can be done for the scaling the height and width of the pdf output. This is particularly handy for stretching the x-axis to get a bit more readability.

plot\_pulldown(   
 data\_input, # input data  
 output\_file = output\_file, # output file name  
 peaks = TRUE, # Show the peaks  
 peak\_method = 'PoT', # Use "Peaks over Threshold"  
 peak\_param = c(.5, 90, 10), # thresholds are by row   
 height=8, width = 120 )

The reason I’ve allowed for a way to separate the data import from the visualization is that it can be useful to do a little pre-processing. For example, perhaps we want to only look at a few of the experiments instead of all of them.

input\_file <- '~/Dropbox/NAU/Research/PepSeq/Pulldown Visualization/Example\_SeveralRows\_3-27.csv'  
output\_file <- '~/Dropbox/NAU/Research/PepSeq/Pulldown Visualization/Example\_SeveralRows\_3-27.pdf'  
  
# What columns are we looking at in this example?  
read.csv(input\_file) %>% colnames()  
#> [1] "library\_member"   
#> [2] "peptide\_seq"   
#> [3] "protein\_ID"   
#> [4] "position"   
#> [5] "protein\_seq"   
#> [6] "protein\_length"   
#> [7] "NAU.peptide.number"   
#> [8] "locus.tag"   
#> [9] "NAU.Protein.set"   
#> [10] "Peptides..Jason.Sahl.s.original.naming.and.peptides"  
#> [11] "X04.04\_KTX\_uncleaved"   
#> [12] "X14.01\_KTX\_uncleaved"   
#> [13] "X04.07\_KTX\_uncleaved"   
#> [14] "X03.01\_KTX\_cleaved"   
#> [15] "X01.02\_KTX\_cleaved"   
#> [16] "X04.06\_KTX\_cleaved"   
#> [17] "X01.01\_KTX\_cleaved"   
#> [18] "X04.01\_KTX\_uncleaved"   
#> [19] "X04.04\_KTX\_cleaved"   
#> [20] "X14.01\_KTX\_cleaved"   
#> [21] "X04.01\_KTX\_cleaved"   
#> [22] "X14.03\_KTX\_uncleaved"   
#> [23] "X04.05\_KTX\_uncleaved"   
#> [24] "X04.07\_KTX\_cleaved"   
#> [25] "X14.03\_KTX\_cleaved"   
#> [26] "X04.05\_KTX\_cleaved"   
#> [27] "X11.01\_KTX\_uncleaved"   
#> [28] "X15.01\_KTX\_uncleaved"   
#> [29] "X09.01\_KTX\_uncleaved"   
#> [30] "X11.01\_KTX\_cleaved"   
#> [31] "X15.01\_KTX\_cleaved"   
#> [32] "X09.01\_KTX\_cleaved"   
#> [33] "X13.02\_KTX\_uncleaved"   
#> [34] "X15.02\_KTX\_uncleaved"   
#> [35] "X03.01\_KTX\_uncleaved"   
#> [36] "X13.02\_KTX\_cleaved"   
#> [37] "X15.02\_KTX\_cleaved"   
#> [38] "X01.02\_KTX\_uncleaved"   
#> [39] "X04.06\_KTX\_uncleaved"   
#> [40] "X01.01\_KTX\_uncleaved"

# Read in this second example set of data  
data\_input <- import\_pulldown(  
 file = input\_file, # input file  
 protein\_column = 'protein\_ID', # which is the protein name  
 position\_column = 'position', # which is the starting location  
 read\_indicator = 'X') # column range to be visualized  
  
# Now make a smaller set of data that only includes X01.01 and X04.04  
data\_small <- data\_input%>%  
 filter( Group %in% c('X01.01\_KTX', 'X04.04\_KTX') ) # Just graph these!

plot\_pulldown\_Shiny(   
 data\_small, # input data  
 peaks = TRUE, # Show the peaks  
 peak\_method = 'PoT', # Use "Peaks over Threshold"  
 peak\_param = c(10, 10) ) # thresholds are by row