A complete analysis of peptide microarray binding data using the pepStat framework

Greg Imholte[*], Renan Sauteraud[†], Mike Jiang[‡] and Raphael Gottardo[§]

November 27, 2013

This document present a full analysis, from reading the data to displaying the results that makes use of all the packages we developped for peptide microarray.

Contents

[*]gimholte@uw.edu

[†]rsautera@fhcrc.org

[‡]wjiang2@fhcrc.org

[§]rgottard@fhcrc.org

1 Generating a peptideSet

Samples and examples are available in the data package PEP.db.

```
library(PEP.db)
library(pepStat)

## Loading required package:  Biobase
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##    clusterExport, clusterMap, parApply, parCapply, parLapply,
##    parLapplyLB, parRapply, parSapply, parSapplyLB
##
## The following object is masked from 'package:stats':
##
##    xtabs
##
## The following objects are masked from 'package:base':
##
##    anyDuplicated, append, as.data.frame, as.vector, cbind,
##    colnames, duplicated, eval, Filter, Find, get, intersect,
##    lapply, Map, mapply, match, mget, order, paste, pmax,
##    pmax.int, pmin, pmin.int, Position, rank, rbind, Reduce,
##    rep.int, rownames, sapply, setdiff, sort, table, tapply,
##    union, unique, unlist
##
## Welcome to Bioconductor
##
##    Vignettes contain introductory material; view with
##    'browseVignettes()'.  To cite Bioconductor, see
##    'citation("Biobase")', and for packages 'citation("pkgname")'.
##
## Loading required package:  IRanges
```

We first read in the .gpr files and the mapping file. For this, we use pepStat's `makePeptideSet` function:

```
mapFile <- system.file("extdata/mapping.csv", package = "PEP.db")
dirToParse <- system.file("extdata/gpr_samples", package = "PEP.db")
pSet <- makePeptideSet(files = NULL, path = dirToParse,
                        mapping.file = mapFile, log=TRUE)
```

1.1 Additional arguments

The empty spots should be listed in order to background correct the intensities. It is also useful to remove the controls when reading the data. Here we have the JPT controls, human Ig (A, E and M) and dye controls.

```
pSet <- makePeptideSet(files = NULL, path = dirToParse,
                        mapping.file = mapFile, log = TRUE,
                        rm.control.list = c("JPT-control", "Ig", "Cy3"),
                        empty.control.list= c("empty", "blank control"))
```

The mapping file must contain a 'visit' column with at least one 'Pre' and one 'Post' per ptid. For our example, we use a toy dataset of 8 samples from 4 patients and we are interested in comparing the antibody binding in placebo versus vaccinated subjects.

```
read.csv(mapFile)

##   filename ptid visit treatment
## 1     f1_1    1   Pre   PLACEBO
## 2     f1_2    1  Post   PLACEBO
## 3     f2_1    2   Pre   PLACEBO
## 4     f2_2    2  Post   PLACEBO
## 5     f3_1    3   Pre   VACCINE
## 6     f3_2    3  Post   VACCINE
## 7     f4_1    4   Pre   VACCINE
## 8     f4_2    4  Post   VACCINE
```

2 Adding peptide informations

At this point, the peptideSet contain only the peptide sequences and the associated background corrected intensities. To continue with the analysis, we need to add the position information, as well as physicochemical properties of the peptides summarized by their z-scales.

The slides used in this example are the enveloppe of HIV-1 and peptide collections are available for this in our PEP.db package (please refere to the vignette and ?pep_hxb2 for more information). However, we will pretend that this is not the case to show an example of how to build a custom peptide collection.

2.1 Creating a custom peptide collection

Here, we load a data.frame that contains the peptides used on the array as well as their start and end coordinates.

```
peps <- read.csv(system.file("extdata/pep_info.csv", package = "PEP.db"))
```

We first create a RangedData object with the available information. The region of interest is the enveloppe of the virus so we add the space information to the object.

```
rd <- RangedData(ranges=IRanges(start = peps$start, end=peps$end),
                 peptide=peps$peptide, space="gp160")
```

The z-scores are added afterward, using the create_db fuction.

```
pep_custom <- create_db(rd=rd)
```

2.2 Summarize the information

With the newly constructed peptide collection, the information can be passed on to the existing peptideSet.

```
psSet <- summarizePeptides(pSet, summary = "mean", position = pep_custom)

## Some peptides have no match in the RangedData object rownames and are removed from
the peptideSet!
```

Now that all the required information is available, we can proceed with the analysis.

3 Normalization

```
pnSet <- normalizeArray(psSet)
```

4 Data smoothing

```
psmSet <- slidingMean(pnSet, width = 9)
```

5 Making calls

```
V_calls <- makeCalls(psmSet, freq = TRUE, group = "treatment",
                     cutoff = .1, method = "FDR", verbose = TRUE)

## You have paired PRE/POST samples

## The selected threshold T is  0.7502
```

The new FDR method automatically selected an appropriate threshold.