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

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

March 27, 2014

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 Introduction

The pepStat package offers a complete analytical framework for the analysis of peptide microarray data. It includes a novel normalization method to remove non-specific peptide binding activity of antibodies, a data smoothing reducing step to reduce background noise, and subject-specific positivity calls.

1.1 Requirements

The pepStat package requires GSL, an open source scientific computing library. This library is freely available at http://www.gnu.org/software/gsl/.

In this vignette, we make use of the samples and examples available in the data package PEP.db.

2 Generating a peptideSet

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

2.1 Reading in .gpr files

The reading function, makePeptideSet, takes a path as its argument and parses all the *.gpr* files in the given directory. Alternatively, one may specify a character vector of paths to individual *.gpr* files.

By default channels F635 Median and B635 Median are collected, and the 'normexp' method of the backgroundCorrect function in the limma package corrects probe intensities for background fluorescence. Other methods may be selected, see documentation.

```
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)
```

While optional, it is strongly recommended to provide a mapping.file giving annotations data for each slide, such as treatment status or patient information. If provided, the mapping.file should be a .csv file. It must include columns labeled filename, ptid, and visit. Elements in column filename must correspond to the filenames of slides to be read in, without the .gpr extension. Column ptid is a subject or slide identifier. Column visit indicates a case or control condition, such as pre/post vaccination, pre/post infection, or healthy/infected status. Control conditions must be labelled *pre*, while case conditions must be labelled *post*. Alternatively, one may input a data.frame satisfying the same requirements.

This minimal information is required by pepStat's functions further in the analysis. Any additional information (column) will be retained and can be used as a grouping variable.

If no mapping file is included, the information will have to be added later on to the peptideSet object.

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
```

2.2 Additional arguments

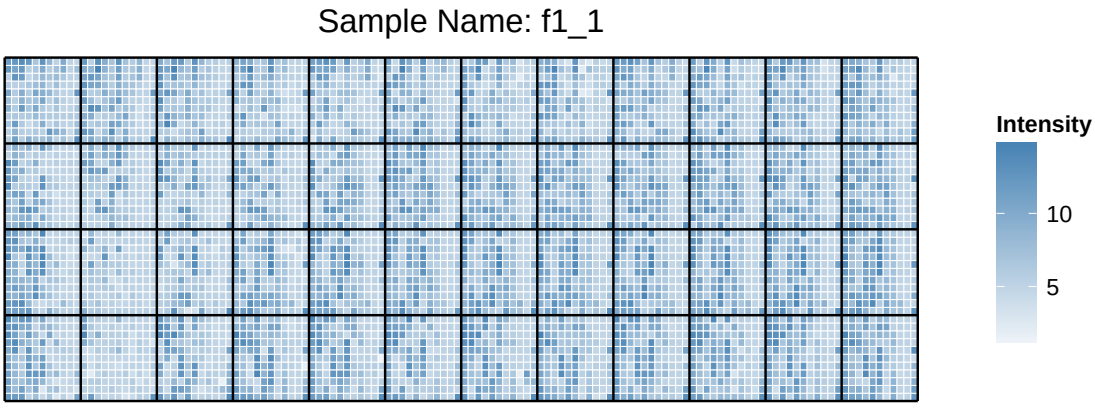
The empty spots should be listed in order to background correct the intensities. It is also useful to remove the controls when reading the data. Here we have the JPT controls, human Ig (A, E and M) and dye controls.

```
pSetNoCtrl <- 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"))
```

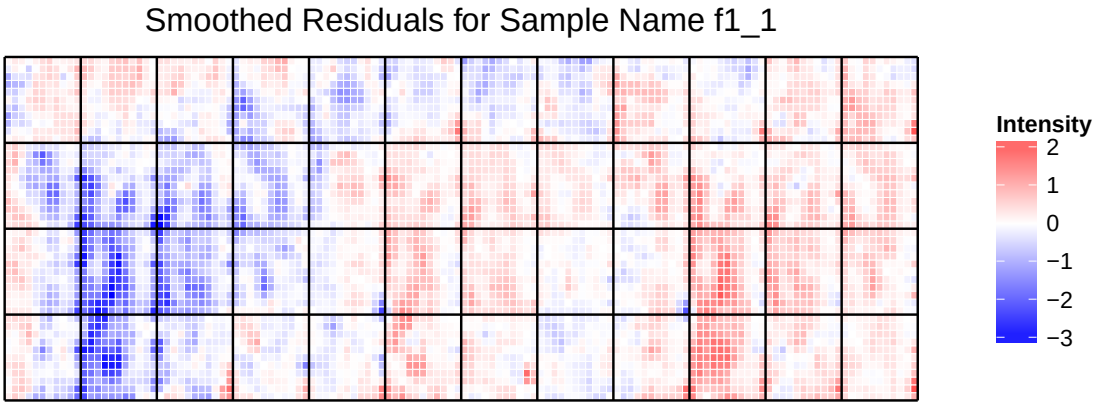
2.3 Visualize slides

We include two plotting functions to detect possible spatial slide artifacts. Since the full plate is needed for this visualization, the functions will work best with rm.contol.list and empty.control.list set to NULL in makePeptideSet.

```
plotArrayImage(pSet, array.index = 1)
```




```
plotArrayResiduals(pSet, array.index = 1, smooth = TRUE)
```



3 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.

3.1 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"))
head(peps)

##   start end         peptide
## 1     1  16 MRVKETQMNWPNLWK
## 2     1  16 MRVMGIQKNYPLLWR
## 3     1  16 MRVMGIQRNCQHLWR
## 4     1  16 MRVKGIRKNYQHLWR
## 5     1  16 MRVRGILRNWQQWWI
## 6     1  16 MRVRGIERNYQHLWR
```

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)
```

3.2 Summarize the information

The function `summarizePeptides` summarizes within-slide replicates by either their mean or median. Additionaly, with the newly constructed peptide collection, peptides positions and annotations 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.

4 Normalization

The primary goal of the data normalization step is to remove non-biological source of bias and increase the comparability of true positive signal intensities across slides. The method developped for this package uses physiochemical properties of individual peptides to model non-specific antibody binding to arrays.

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

An object of class `peptideSet` containing the corrected peptides intensities is returned.

5 Data smoothing

The optional data smoothing step takes advantage of the overlapping nature of the peptides on the array to remove background noise caused byexperimental variation. It is likely that two overlapping peptides will share common binding signal, when present. `pepStat` use a sliding mean technique technique to borrow strength across neighboring peptides and to reduce signal variability. This statistic increases detection of binding *hotspots* that noisy signals might otherwise obscure. Peptides are smoothed according to their sequence alignment position, taken from `position(psSet)`.

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

6 Making calls

The final step is to make the positivity calls. The function `makeCalls` automatically uses information provided in the mapping file, accessed via `pData(pSet)`. It detects whether samples are paired or not. If samples are paired, POST intensities are subtracted from PRE intensities, then thresholded. Otherwise, PRE samples are averaged, and then subtracted from POST intensities. These corrected POST intensities are thresholded.

The `freq` argument controls whether we return the percentage of responders against each peptide, or a matrix of subject specific call. When `freq` is `TRUE`, we may supply a `group` variable from `pData(psmSet)` on which we split the frequency calculation.

```
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
```

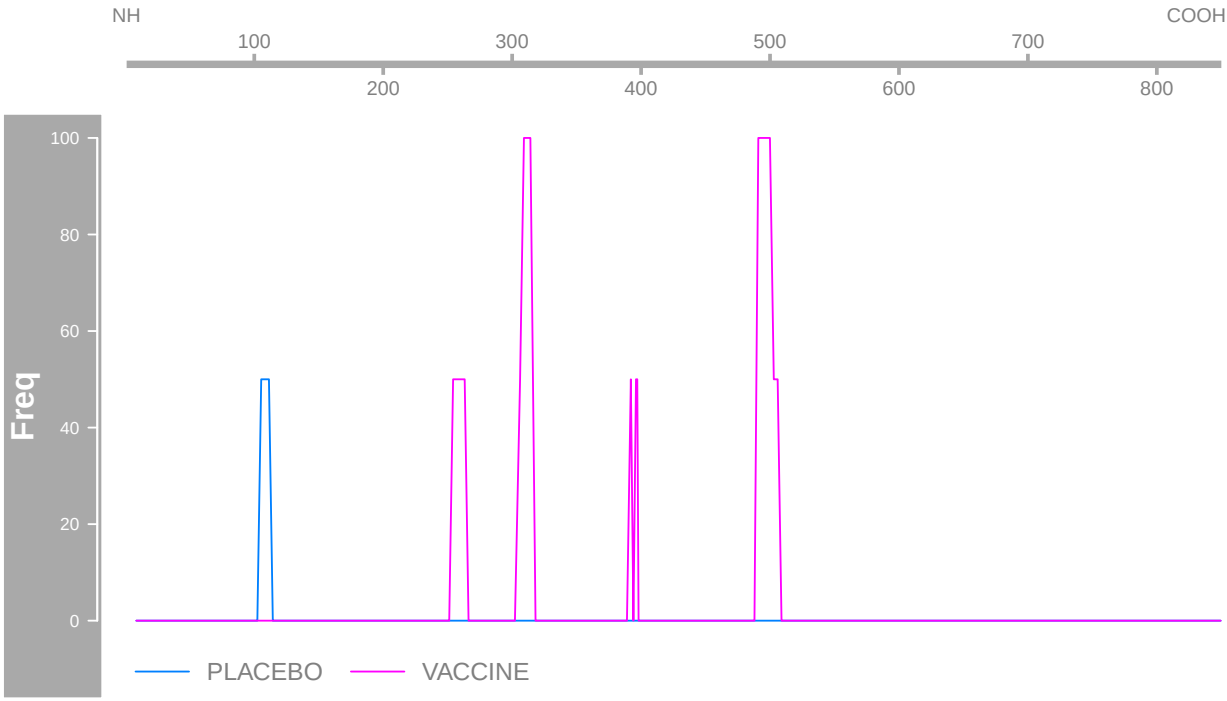
The function automatically selected an appropriate FDR threshold.

7 Results

`pepStat` extends the visualisation package `Gviz` and adapts it for peptide and protein sequences.

The most simple plots can display an overview of the difference between groups. In this case, comparing placebo and vaccine.

```
plot_inter(psmSet, V_calls)
```



However, this plot does not show differences that may exist within a group. Values are averaged for the group so this plot cannot detect when a single clade shows a much higher or lower frequency of binding than the others for a given group.

To detect these changes, we plot all peptides individually. This is best visualised with a track for each group.

```
plot_clade(psmSet, V_calls, clade="M")

## [1] "HELLO"

## Error:  The mandatory argument 'NA' is missing with no default
```

8 Summary

Here we showcase a quick analysis of peptide microarray data for HIV-1 gp160. This displays the minimal amount of code required to go from raw data file to antibody binding positivity call.

```
library(pepStat)
library(PEP.db)
mapFile <- system.file("extdata/mapping.csv", package = "PEP.db")
dirToParse <- system.file("extdata/gpr_samples", package = "PEP.db")
ps <- makePeptideSet(files = NULL, path = dirToParse, mapping.file = mapFile)
data(pep_hxb2)
ps <- summarizePeptides(ps, summary = "mean", position = pep_hxb2)
ps <- normalizeArray(ps)
ps <- slidingMean(ps, width = 9)
V_calls <- makeCalls(ps, freq = TRUE, group = "treatment")
```

9 sessionInfo

```
sessionInfo()

## R Under development (unstable) (2013-10-02 r64018)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## 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] grid      parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
## [1] pepStat_0.99.7     Pviz_0.99.1        Gviz_1.6.0
## [4] data.table_1.9.2   IRanges_1.20.7     Biobase_2.22.0
## [7] BiocGenerics_0.8.0 PEP.db_0.99.5      knitr_1.5.21
##
## loaded via a namespace (and not attached):
##  [1] AnnotationDbi_1.24.0   biomaRt_2.18.0         Biostrings_2.30.1
##  [4] biovizBase_1.10.8      bitops_1.0-6           BSgenome_1.30.0
##  [7] cluster_1.15.1         codetools_0.2-8        colorspace_1.2-4
## [10] DBI_0.2-7              dichromat_2.0-0        digest_0.6.4
## [13] evaluate_0.5.1         fields_6.9.1           formatR_0.10
## [16] Formula_1.1-1          GenomicFeatures_1.14.5 GenomicRanges_1.14.4
## [19] ggplot2_0.9.3.1        gtable_0.1.2           highr_0.3
## [22] Hmisc_3.14-3           labeling_0.2           lattice_0.20-27
## [25] latticeExtra_0.6-26    limma_3.18.13          maps_2.3-6
## [28] MASS_7.3-30            munsell_0.4.2          plyr_1.8.1
## [31] proto_0.3-10           RColorBrewer_1.0-5     Rcpp_0.11.1
## [34] RCurl_1.95-4.1         reshape2_1.2.2         Rsamtools_1.14.3
## [37] RSQLite_0.11.4         rtracklayer_1.22.6     scales_0.2.3
## [40] spam_0.41-0            splines_3.1.0          stats4_3.1.0
## [43] stringr_0.6.2          survival_2.37-7        tools_3.1.0
## [46] XML_3.98-1.1           XVector_0.2.0          zlibbioc_1.8.0
```