Chapter 10

Analyzing Peptide Microarray Data with the R *pepStat* Package

Gregory Imholte, Renan Sauteraud, and Raphael Gottardo

Abstract

In this chapter we demonstrate the use of *R* Bioconductor packages *pepStat* and *Pviz* on a set of paired peptide microarrays generated from vaccine trial data. Data import, background correction, normalization, and summarization techniques are presented. We introduce a sliding mean method for amplifying signal and reducing noise in the data, and show the value of gathering paired samples from subjects. Useful visual summaries are presented, and we introduce a simple method for setting a decision rule for subject/peptide responses that can be used with a set of control peptides or placebo subjects.

Key words Normalization, False discovery rate, Data visualization, Baseline correction, Decision rule, Sliding mean, Smoothing, Background correction

1 Introduction

The peptide microarray assay has proven useful in immunology studies, where understanding antibody reactions has helped characterize vaccine responses, allergic reactions, and autoimmune disorders. A common slide design is the peptide tiling array, in which slide peptides have partially overlapping amino acid sequences drawn from a larger protein. In this chapter, we demonstrate how the *R* statistical computing environment [1] and the Bioconductor project [2] can be used to analyze peptide microarray data. In particular, we will demonstrate the *pepStat* package [3], which can be used to normalize, analyze, and visualize peptide microarray data. The *pepStat* package is an open-source software, freely available from the Bioconductor project (http://www.bioconductor.org), an open-source repository of *R* packages for bioinformatics.

1.1 Example Data

We use a specific dataset comprising 200 slides, prepared with pre- and post-treatment samples drawn from 100 subjects to illustrate the pepStat package. Among these 100 subjects, 80 subjects received an experimental vaccine regimen targeting human immunodeficiency

Marina Cretich and Marcella Chiari (eds.), *Peptide Microarrays: Methods and Protocols*, Methods in Molecular Biology, vol. 1352, DOI 10.1007/978-1-4939-3037-1_10,

virus (HIV) and the remaining 20 subjects received a placebo (see below for details). Thus we have 100 slides separately measuring each subject's baseline antibody responses, and 100 slides separately measuring each subject's antibody responses to treatment stimulus, for a total of 200 slides.

1.2 Slide Design

The proteins gp120 and gp41 are surface proteins on the HIV virion and are frequent targets of antibody response in naturally infected individuals. The trial vaccine consisted of a recombinant canarypox vector with a fabricated gp120 AIDSVAX B/E core boost [4], and the slides were designed to interrogate antibody responses against gp120 and gp41. To understand vaccine responses against a diverse panel of HIV strains, the slide design tiled consensus gp120/gp41 amino acid sequences from six HIV clades (A, B, C, D, CRF01, CRF02) and a consensus sequence, M [5]. The seven sequences were aligned against the HxB2 standard reference sequence to facilitate analysis and visualization. Following this alignment, each sequence was assigned a position number, corresponding to where the peptide's central amino acid aligned with the HxB2 reference sequence.

Each slide contains three identical subarrays printed in four-by-four grids of blocks. Each block was an 11 by 11 grid of probes. Thus, each subarray had 1936 total probes. Figure 1 illustrates the layout of blocks and probes for a single slide. Our HIV tiling array design contained 1423 unique peptide sequences representing gp120/gp41 proteins, each printed once per subarray. The remaining 513 subarray probes consisted of human-Ig controls, Cy3 landmarks, control peptides, and empty spots.

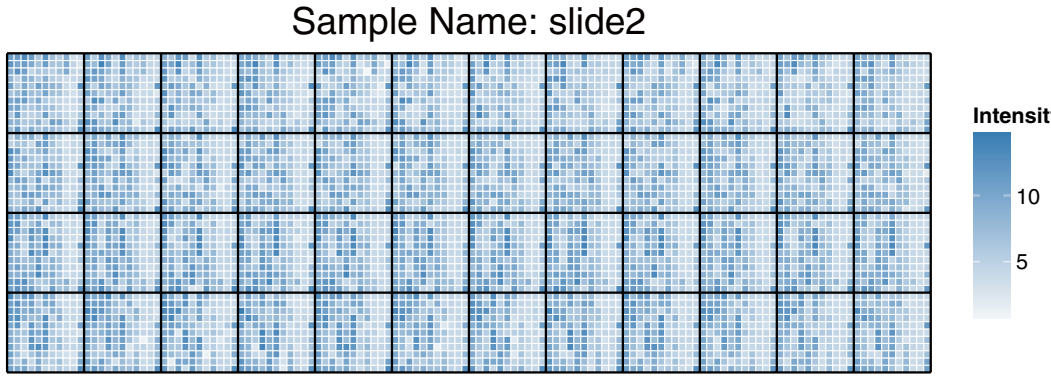

Fig. 1 Slide image for a single subject. Single boxes are colored according to $\log_2$ intensity. *Black horizontal* and *vertical lines* separate slide blocks

2 Data Preprocessing

We outline the analysis of paired peptide microarray data with the *R* package *pepStat*. Our demonstration requires the installation of the free *R* statistical computing environment and the Bioconductor tool (http://bioconductor.org/install/). The software package *pepStat* and its dependencies are available from Bioconductor.

2.1 Reading GPR Files

The core of the *pepStat* package is the *peptideSet* class, which unites a variety of data sources such as expression information, sequence information, and slide metadata. Although *peptideSet* objects may be manually created, the *pepStat* package implements a function, *makePeptideSet*, to read in peptide microarray data stored in the GPR file format created by GenePix Pro software [6], connect the slide expression data with relevant slide metadata, and create a *peptideSet* class object. Given a directory containing GPR files, and a *mapping file* containing information about slide metadata, we can create a *peptideSet* object:

```
> library(pepStat)
> my_gpr_directory <- "~/directory/containing/gpr_files"
> my_mapping_file <- "~/filepath/to/mapping_file.csv"
> pset <- makePeptideSet(path=my_gpr_directory, mapping.file=my_mapping_file)
```

Detailed information about how to structure a mapping file can be found in the help entry for *makePeptideSet*.

The function *makePeptideSet* extracts the F635 Median and B635 Median columns from GPR files, representing the median signal intensity of foreground pixels from probe pixels and local background pixels, respectively. The default background correction method is the *normexp* method, which has the benefit of always estimating a strictly positive corrected foreground intensity and has been shown to have favorable performance [7]. Background-corrected probe intensities are transformed to the $\log_2$ scale by default to reduce skewness, stabilize variance, and improve interpretation of results.

2.2 Accessing Data from peptideSet Objects

The *peptideSet* class combines the *GRanges* class from the package *GenomicRanges* [8] and the *ExpressionSet* class from the package *Biobase* [1]. The *peptideSet* class implements several *methods* (i.e., convenience functions) to access various data fields in a *peptideSet* object. Table 1 lists several convenient methods for the *peptideSet* class. The *exprs* method returns a matrix of probe intensities, where rows correspond to probes and columns correspond to slides. Information about probes can be accessed with the methods *ranges* and *values*. By default, *values* will access probes' names, sequences, and spatial information. Rows in the data frame returned by *values*

Table 1
***peptideSet* class methods and their descriptions**

Method	Description
pData	Access slide metadata.
exprs	Access probe/peptide intensities.
ranges	Access *GRanges* probe/peptide sequence data and metadata.
values	Access *GRanges* probe/peptide metadata.
clade	Return logical matrix indicating peptide clade membership.
start, end, width	Return start, end, or width of peptides based on sequence data.
position	Return position of peptides, defined as *floor((start+end)/2)*.
peptide	Access peptide amino acid sequence for probes/peptides.
featureID	Access peptide/probe names.
"[i, j]"	Subset object by slides *i*, probes *j*.

correspond to rows in the matrix of probe intensities. The *pData* method returns a data frame with rows corresponding to slides, matching the columns in the matrix of probe intensities.

2.3 Plotting Slide Images

Visual inspection of microarray data can reveal spatial flaws that may not be obvious from data tables alone. The function *makePeptideSet* stores probes' spatial locations in the returned object, and the *pepStat* package provides two functions to plot probes' intensity values against slide spatial positions. The function *plotArrayImage* plots selected slides in a two-dimensional grid with color intensity proportional to background-corrected probe intensity. The function *plotArrayResiduals* exploits within-slide replication to give a clearer view of spatial trends among probe intensities. Points are colored according to a probe's deviation from the within-slide mean of replicates for that probe's type. Spatial patterns appear as continuous areas having similar color. The optional argument *smooth = TRUE* applies a two-dimensional smoother on the slide residuals to help identify broader spatial trends. Figure 1 demonstrates a slide image from a single pre-treatment subject.

```
> plotArrayImage(pset, array.index = 1:2)
> plotArrayResiduals(pset, array.index = 1:2)
> plotArrayResiduals(pset, array.index = 1:2, smooth = TRUE)
```

3 Data Summarization and Normalization

With microarray data, it is important to estimate and remove technical effects to improve the comparability of signal intensities across multiple slides. The extensive experimental protocol for peptide microarrays introduces several sources of variability. When we fail to account for these effects, we risk confounding true biological signals with technical effects arising from plate and sample preparation, such as batch effects, slide effects, and nonspecific binding effects. We can also exploit the overlapping nature of peptides in tiling array designs to reduce the noise and boost signal in our results. In this section we describe how to incorporate peptide sequence information and how to use the normalization procedure employed by *pepStat*.

3.1 Probe Summarization

With probe replication, we observe multiple fluorescence intensities against the same peptide sequence. Before further analysis, we summarize replicates within slide by their median with the function *summarizePeptides*. The median summary is robust against outliers and questionable observations. We refer to the median of probe intensities as the peptide intensity. During summarization we also add additional sequence information about our peptides. Our slide design incorporates peptides from seven aligned amino acid sequences representing different clades of HIV. Using the alignment information, we constructed a *GRanges* object containing information about each peptide's alignment position and clade membership. Our slide design is represented in the *pep_hxb2* data set contained in the package *pepDat*, available on Bioconductor. The following command executes the prescribed operations:

```
> pset_sum <- summarizePeptides(pset, summary="median",
position=pep_hxb2)
```

Users can incorporate their own custom sequence information following instructions provided in the *pepStat* vignette.

3.2 Normalization

Batch and slide effects may arise in a variety of ways, such as subtle differences in sample concentrations, analyst procedures, scanner properties, or even plate manufacturing. Figure 2 shows box plots of the median peptide intensities on ten slides from pre-treatment subjects. In the absence of treatment effects, these box plots should approximately center around the same value. Instead we see differences as great as fourfold between the slides, indicating the presence of tremendous slide effects. Nonspecific binding refers to antibody binding events occurring against peptides unrelated to an antibody's epitope. Labeled secondary antibodies and subjects' primary antibodies can contribute nonspecific binding effects, and

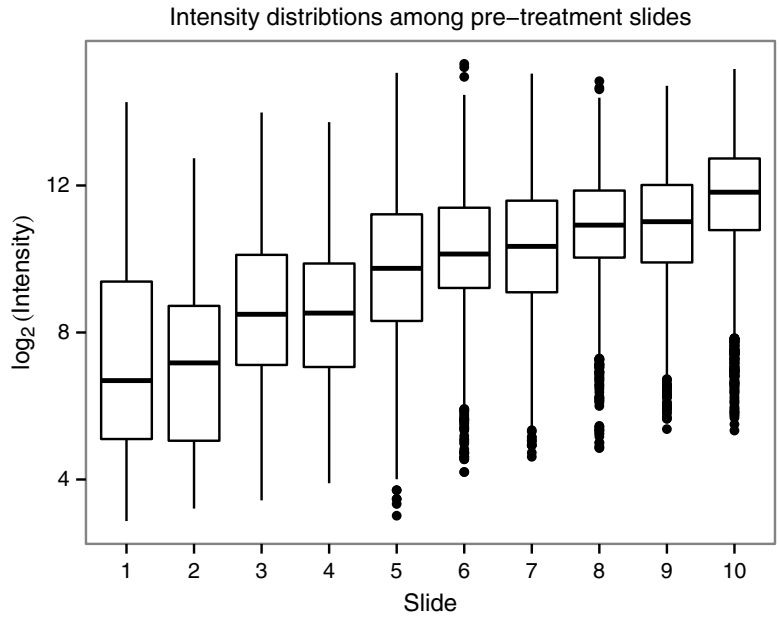

Fig. 2 Intensity distributions for summarized peptide intensities from pre-treatment subjects. Slides are ordered by median intensity. Large shifts in intensity distributions indicate a slide technical effect

pepStat normalization models nonspecific antibody binding as a function of peptide physiochemical properties. We use the amino acid z-scales [9], which are weighted combinations of 26 physiochemical properties. A peptide z-scale is obtained by summing the z-scale values of the amino acids it comprises. Figure 3 shows the nonlinear relationships between each peptide z-scale and intensity value.

The function *normalizeArray* implements a linear model that estimates peptide intensity as a function of a slide effect, quadratic z-scale effects, and a Student-*t*-distributed error term. Using *t-distributed* errors reduces the influence of outliers and signal-carrying peptides on the estimation of slide effects and nonspecific binding effects. This normalization method performs favorably compared to several other methods [10–12]. To accommodate differences in nonspecific binding among subjects, this model is fit on a per-slide basis. We execute z-scale normalization with the command

```
> pset_norm <- normalizeArray(pset_sum)
```

and we refer to the resulting intensity values as normalized peptide intensities.

3.3 Sliding Mean Method

In a tiled slide design, peptides are drawn from a linear protein amino acid sequence in short, overlapping segments. The overlapping segments share amino acid sequences, and thus are similarly susceptible to binding from the same antibodies. If we order peptide

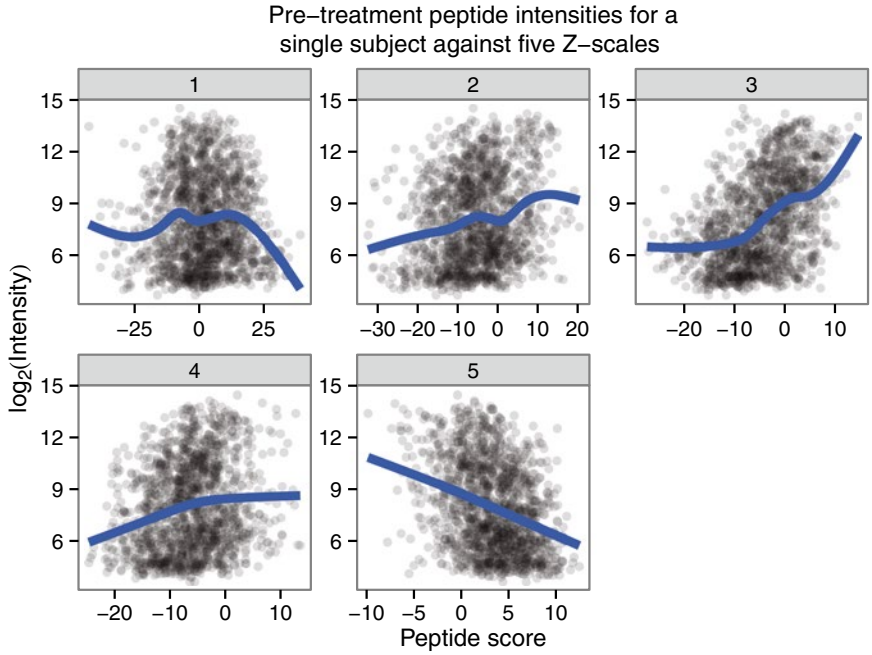

Fig. 3 Peptide intensity values from a pre-treatment slide are plotted against peptide z-scale values. Considerable trends indicate nonspecific binding effects related to peptide physiochemical properties

intensities and normalized peptide intensities by their position in the protein sequence, we detect strong patterns in estimated autocorrelation functions. Strong autocorrelation implies that neighboring peptides tend to share similar intensity values. Figure 4 demonstrates the reduction in autocorrelation induced by z-scale normalization. Although autocorrelation is reduced, the remaining spikes at lags one and two likely represent biological effects rather than technical effects. For our slide design, a position lag of one or two between peptides represents, on average, sharing a common sequence 12 or 9 amino acids in length. We exploit this remaining information by applying a sliding mean technique to boost signal and reduce noise.

The *pepStat* function *slidingMean* applies the mean function over a window generated by peptide positions. A peptide position can refer to a peptide's alignment within a baseline reference sequence, or to the relative ordering of a peptide within a sequence of overlapping peptides drawn from a single amino acid sequence. Peptide positions in our slide design are assigned according to alignment with the HIV HxB2 standard reference sequence. Within *slidingMean*, position is calculated as the rounded average of the *start* and *end* values from the *featureRange* slot of a *peptideSet* object. The *position* method from Table 1 returns the values used in *slidingMean*. The *width* argument of *slidingMean* controls the width of the sliding mean window.

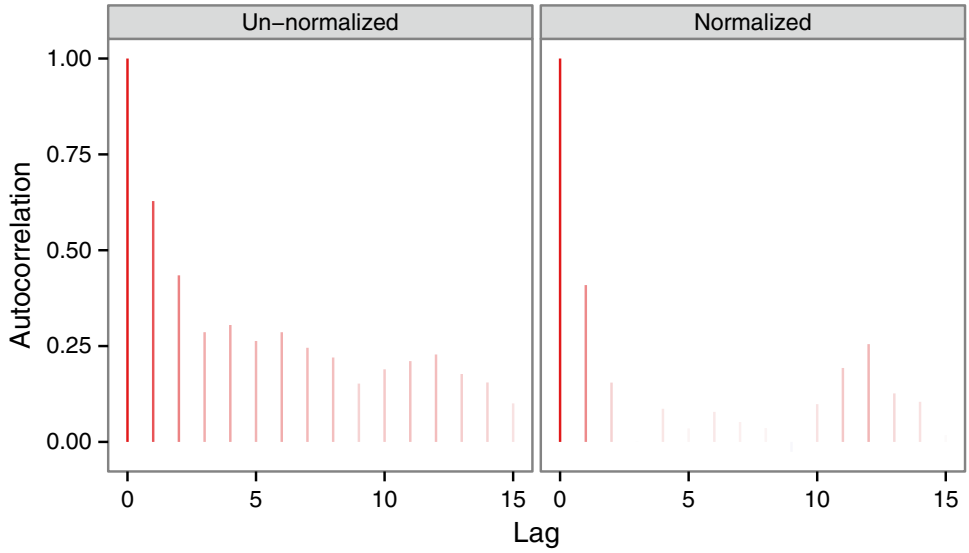

Fig. 4 Autocorrelation functions of peptide intensities from a single clade, before and after normalization. High autocorrelation in un-normalized data indicates similar binding effects for similar peptides. Normalization reduces autocorrelation, but does not eliminate the first few lags

The *slidingMean* function optionally smooths peptides by groups. Our slide design is based on seven amino acid sequences representing six HIV clades and a consensus sequence. Peptide clade membership is stored in the character vector *clade* of the *featureRange* slot, and membership in multiple clades is indicated with clade names separated by commas:

```
> ranges(pset_norm)$clade[74:76]
[1] "A" "C" "M,B,D"
```

Particular applications may choose to group peptides by factors other than clades, which may be accomplished by assigning the desired grouping factor to *clade* in the *featureRanges* slot of the *peptideSet* object. The *clade* method returns a logical matrix whose columns indicate group membership for each level of the grouping factor:

```
> clade(psset_rv)[74:76,]
A CRF01 CRF02 D B C M
WVTVYYGVPVWKDAE TRUE FALSE FALSE FALSE FALSE FALSE FALSE
WVTVYYGVPVWKEAK FALSE FALSE FALSE FALSE FALSE TRUE FALSE
WVTVYYGVPVWKEAT FALSE FALSE FALSE TRUE TRUE FALSE TRUE
```

The argument *split.by.clade* controls whether smoothing is divided by clade or performed in aggregate.

```
pset_sm_all <- slidingMean(pset_norm, width=9, split.by.clade=FALSE)
pset_sm_split <- slidingMean(pset_norm, width=9, split.by.clade=TRUE)
```

4 Data Analysis

After normalizing slides and applying our sliding mean technique, we are prepared to analyze our data to discover binding trends induced by treatment. Additionally, we are interested in discovering which peptides experienced increased binding as a result of treatment. We discuss methods for setting detection thresholds, and give a few examples for useful visual summaries of results. The *Pviz* package [13], which builds upon *Gviz* [14], available on Bioconductor, provides a convenient framework for visualizing the results of peptide microarray experiments. *Pviz* and *Gviz* have many configuration options and we refer the reader to their respective user guides for details.

4.1 Baseline Correction for Nonspecific Binding

Nonspecific binding effects vary among subjects, but they are highly correlated for the same subject. We demonstrate this correlation using our placebo subjects. Placebo subject slide measurements contain no treatment-related biological signal, so we examine correlations between pre- and post-treatment smoothed intensities to check for consistent nonspecific binding effects. Figure 5 is a density plot of pre-/post-treatment pairs of smoothed peptide intensities among placebo subjects. The dashed line is the $y=x$ identity line. We see a strong linear association between the pre- and post-treatment measurements, indicating the presence of weak but consistent binding effects. Figure 6 shows the correlation of pre- and post-treatment smoothed intensities, calculated separately across 1423 peptides. All correlations are positive, well over half exceed a correlation of 0.6, suggesting that a difference of post-treatment minus pre-treatment measurements will often have lower variance than either, separately. For each subject-peptide combination, we subtract pre-treatment smoothed peptide intensity from the post-treatment value, and call the resulting value a response index.

4.2 Visualizing Antibody Responses

The *Pviz* defines plotting elements called tracks, which can carry information such as annotations, data summaries, or sequences. We show two visual summaries of our data using the *heatmap* and *line* types in the *DTrack* class, along with annotations formed with the *ATrack* class and an axis formed with the *ProteinAxisTrack*.

Figure 7 is a heatmap of response indices for all subjects. For this figure, we plot *aggregate* response indices created by smoothing without respect to clade membership. Rows represent observations

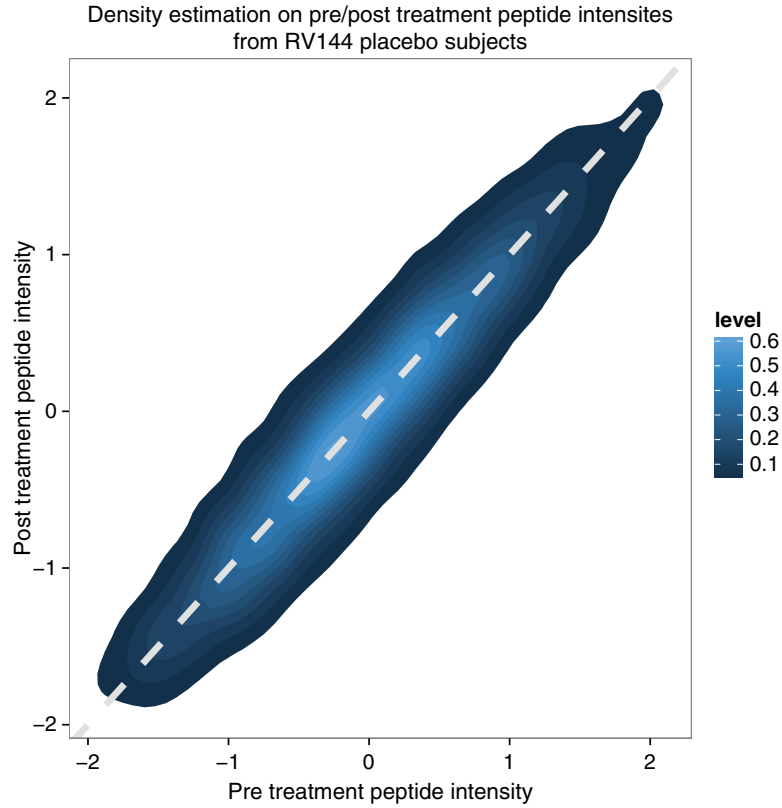

Fig. 5 A two-dimensional density estimation of the relationship between pre- and post-treatment smoothed peptide intensities

from a single subject, while columns are lined against position. The intensity of color corresponds to the positive magnitude of the response index for peptides at a given position. We detect a pattern of vertical bands prior to the V1 loop, inside the V2 and V3 loops, and near the end of gp120 among vaccinated subjects. Response indices among gp41 peptides and placebo subjects tend to be near zero, indicated by white or mild colors. Figure 8 uses the *clade* response indices created by smoothing with respect to clade membership. For the seven clades, we plot the average clade response index among vaccinated subjects against position. From such figures, we can detect differences in response indices among the clades. Here we observe relatively weak response indices in the V2 loop of the B clade, the gp120 terminus of the CRF02 clade, and the V3 loop of the CRF01/D/M clades.

4.3 Classifying Antibody Responses Within Subjects

Beyond aggregate summaries, subject level inference for peptide response may be of interest. Classifying subjects as responders or non-responders against a given peptide can enable further examination of relationships such as correlation between response status and other covariates. We opt to control the false discovery rate

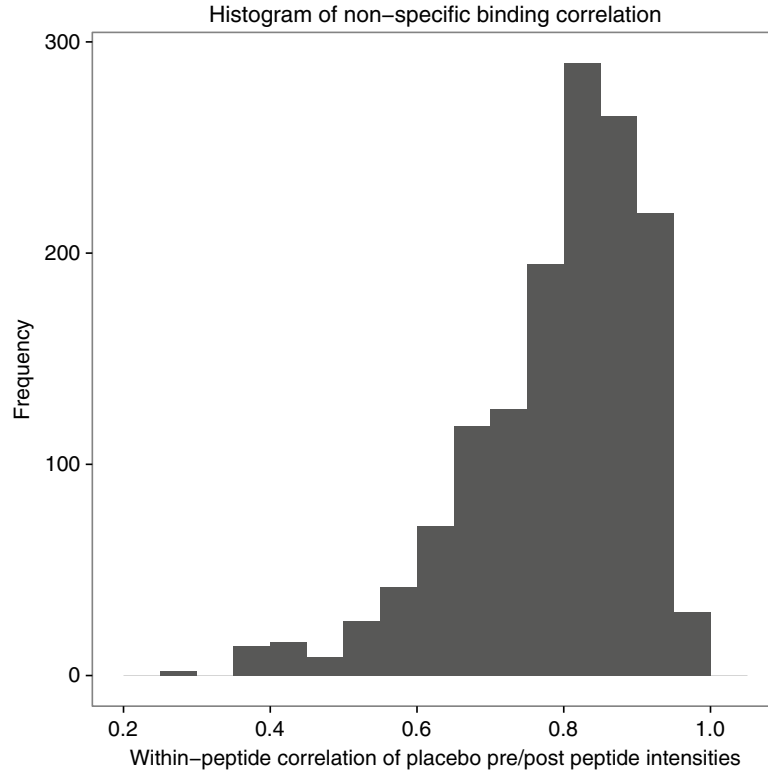

Fig. 6 Correlation between pre- and post-treatment smoothed peptide intensities estimated from 20 placebo subjects for each unique peptide

(FDR) in our classification [15]. We refer to a response classification as a call, and the FDR gives the expected proportion of calls that are false. We make calls by setting a threshold on response indices, and we have a few options available for setting thresholds that, on average, control the FDR to a desired level.

The *pepStat* function *makeCalls* implements a symmetry-based thresholding method, accessible with the argument *method = "FDR"*. This method is further described in (Imholte et al.). An alternative threshold may be set with the use of known placebo slides or control peptides. We illustrate choosing a threshold based on control peptides. For a given threshold *t*, we calculate the proportion of control peptides that we falsely call positive, and call this *pt*. We estimate that the proportion of false calls in the control peptides is similar to the proportion of false calls made among non-control peptides. At a given threshold *t* we estimate the proportion of false discoveries FDR*t* as

$$\mathrm{FDR}_t = \frac{\left(\#\ \text{non-control peptide} \times n\text{subject}\right) \times p_t}{\left(\#\ \text{non-control response indices above threshold } t\right)}$$

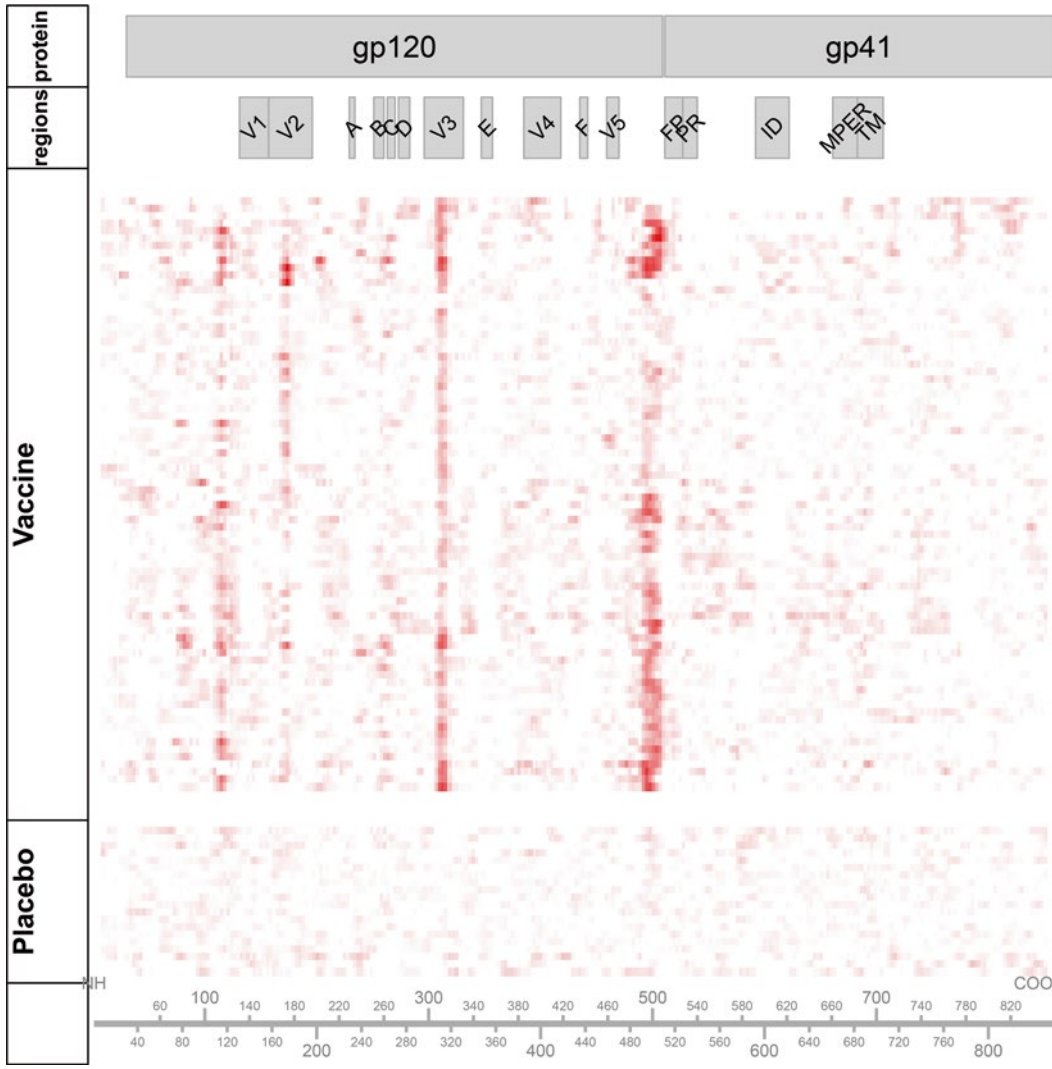

Fig. 7 A heatmap of response indices, plotted by position. *Rows* represent subjects. *Darker colors* represent higher response index values

In summary, the numerator estimates the number of false calls among non-control peptide calls, while the denominator is the total number of non-control calls. Searching over a range of t values, we choose a threshold t such that FDRt is closest to our desired FDR. We use gp41 peptides as control peptides, because the vaccine contained no gp41 insert. Using vaccine subjects only, we calculate that a response index threshold of $t = 1.8$ controls FDR at approximately 10 %. Figure 9 plots the proportion of calls among vaccinated subjects at each position, split by clade. Notably, call frequencies are relatively low in V2 positions for the B clade, V3 positions in clade CRF01, and C-terminus positions in CRF02.

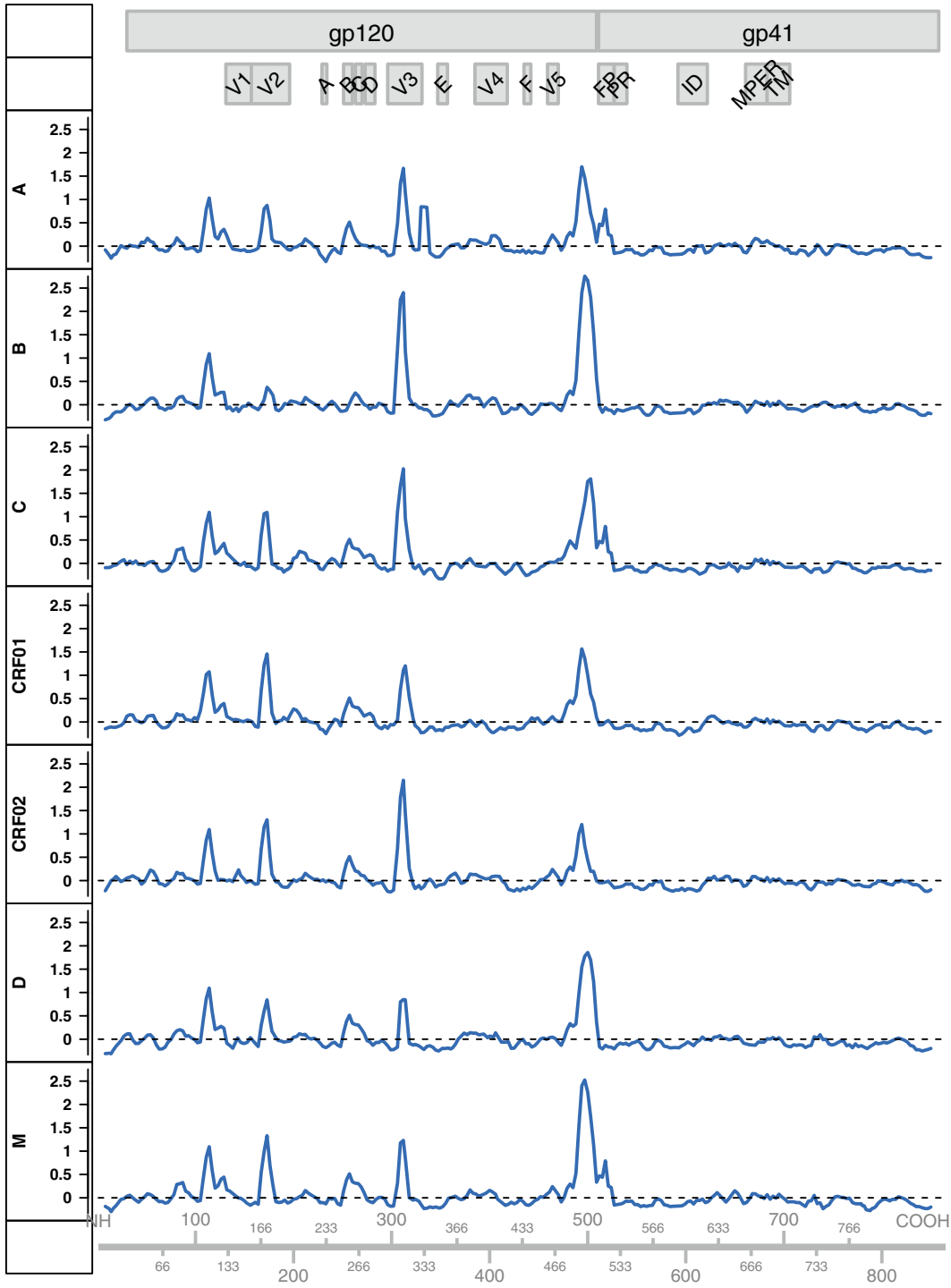

Fig. 8 Average response indices from vaccinated subjects, plotted by position

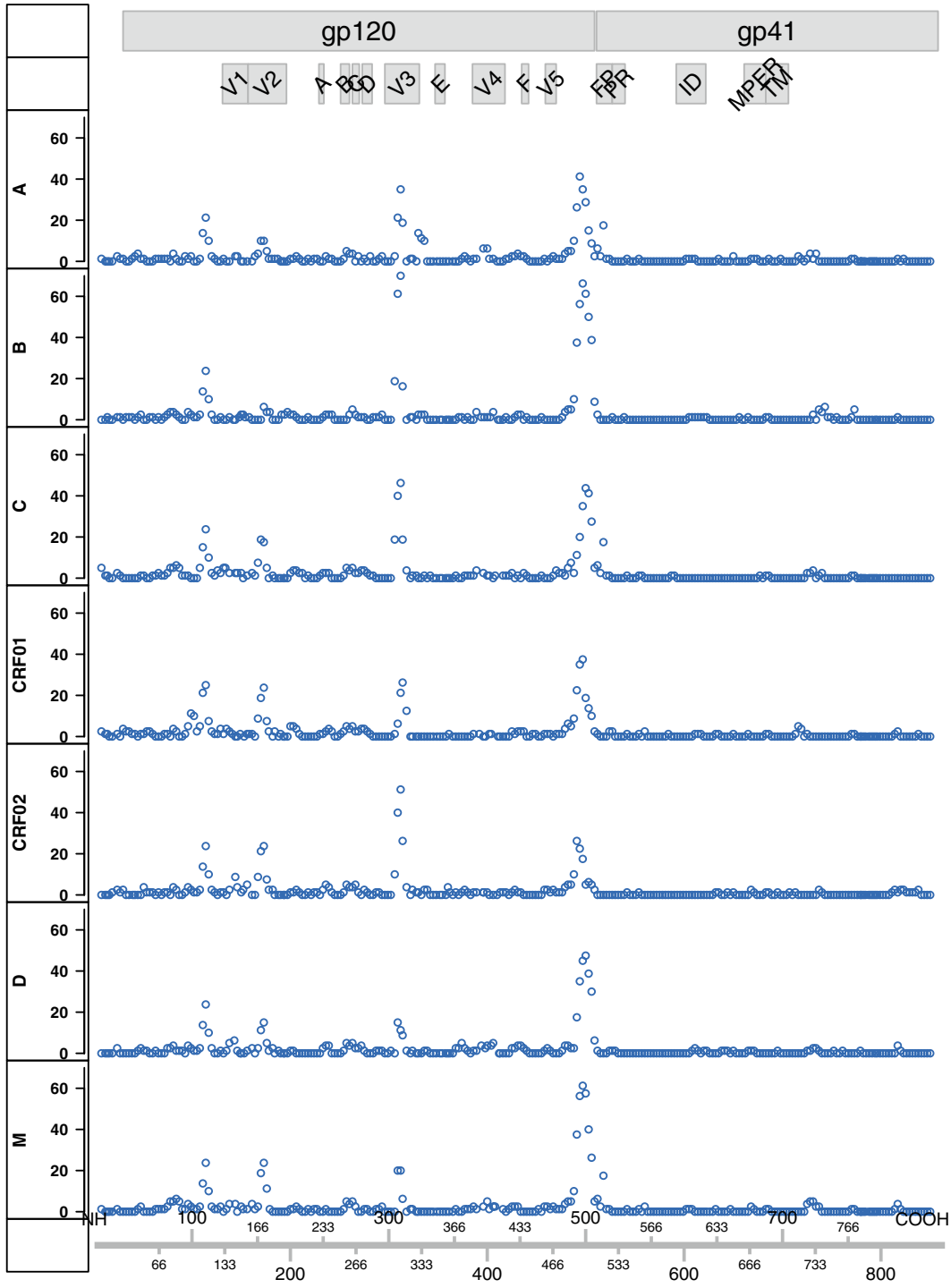

Fig. 9 Proportion of vaccinated subjects estimated to have responded against each position in different clades

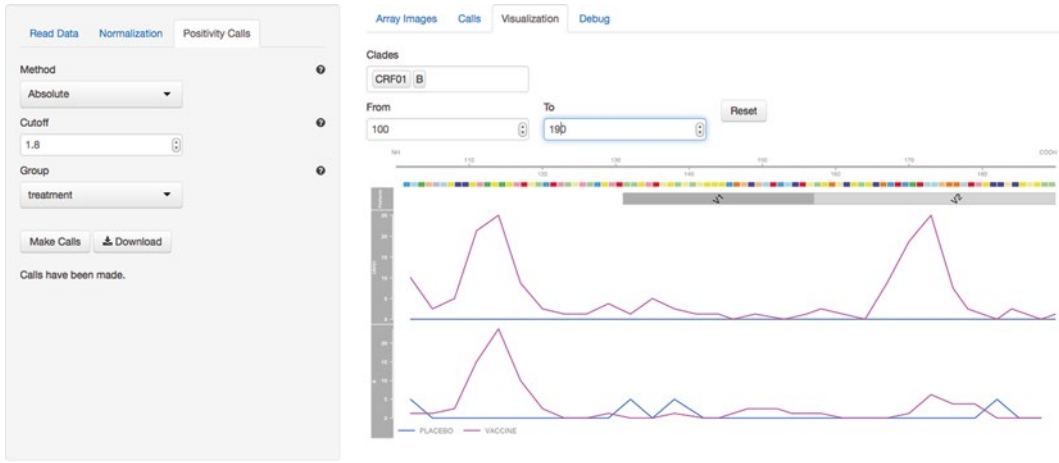

Fig. 10 A screenshot of *pepStat's* Shiny application running on our data

4.4 pepStat Shiny Application

The R *shiny* package [16] is a web application framework for creating graphical user interfaces to conduct interactive data analysis in R. Shiny applications are built on top of R packages, and provide users with a graphical alternative to traditional command line R scripting. The *pepStat* package includes a Shiny application to assist users with analysis of their own data sets, tracing the steps detailed above. The application allows users to quickly read and normalize data, generate calls, and plot response frequencies split by various metadata variables such as treatment. Figure 10 shows a screenshot of the *pepStat* Shiny application analyzing our data. We view a close up of B clade and CRF01 clade response rates against V1 and V2 loop positions. The application automatically draws *Pviz* tracks from our calls analysis, easily allowing us to compare response frequencies among different grouping factors, and to focus our attention on specific positions. We run the *pepStat* Shiny application with the following command:

```
> shinyPepStat()
```

References

1. R Core Team (2014) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria, http://www.R-project.org/
2. Gentleman R, Carey VJ, Bates DM et al (2004) Bioconductor: open software development for computational biology and bioinformatics. Genome Biol 5:R80
3. Gottardo R, Imholte G, Sauteraud R et al. (2014) pepStat: statistical analysis of peptide microarrays. R package version 1.1.0
4. Rerks-Ngarm S, Pitisuttithum P, Nitayaphan S et al (2009) Vaccination with ALVAC and AIDSVAX to prevent HIV-1 infection in Thailand. N Engl J Med 361(23):2209–2220
5. Gottardo R et al (2013) Plasma IgG to linear epitopes in the V2 and V3 regions of HIV-1 gp120 correlate with a reduced risk of infection in the RV144 vaccine efficacy trial. PLoS One 8:e75665
6. Molecular Devices (2014) GenePix Pro. http://mdc.custhelp.com/app/answers/detail/a_id/18792/~/genepix%C2%AE-pro-7-

microarray-acquisition-%26-analysis-software-download-page

7. Ritchie M, Silver J, Oshlack A et al (2007) A comparison of background correction methods for two-colour microarrays. Bioinformatics 23(20):2700–2707
8. Lawrence M, Huber W, Pagès H et al (2013) Software for computing and annotating genomic ranges. PLoS Comput Biol 9(8): e1003118. doi:10.1371/journal.pcbi.1003118
9. Hellberg S, Sjöström M, Skagerberg B et al (1987) Peptide quantitative structure-activity relationships, a multivariate approach. J Med Chem 30(7):1126–1135
10. Imholte G, Sauteraud R, Korber B et al (2013) A computational framework for the analysis of peptide microarray antibody binding data with application to HIV vaccine profiling. J Immunol Methods 395(1–2):1–13
11. Nahtman T, Jernberg A, Mahdavifar S et al (2007) Validation of peptide epitope microarray experiments and extraction of quality data. J Immunol Methods 328:1–13
12. Bolstad B, Irizarry R, Astrand M et al (2003) A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. Bioinformatics 19:185
13. Sauteraud R, Jiang M, Gottardo R. Pviz (2014) peptide annotation and data visualization using Gviz. R package version 1.1.0
14. Hahne F, Durinck S, Ivanek R et al. Gviz (2012) plotting data and annotation information along genomic coordinates. R package version 1.8.0
15. Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. J R Stat Soc Series B 57(1):289–300
16. RStudio and Inc (2014) shiny: Web application framework for R. R package version 0.10.2.1. http://CRAN.R-project.org/package=shiny