NetWeAvers Vignette

Elizabeth McClellan

April 4, 2013

Contents

1 Introduction

This document describes the functions available in the `netweavers` package. The package may be used with proteomics mass spectrometry data to summarize quantified peptide data, fit linear models and test for differential expression, and perform network analysis on protein-protein interaction networks using protein *p*-values. To get started, first you need to load the package in your `R` session:

```
> library(netweavers)
```

We will use the example dataset called `vanHoof`, which contains mass spectrometry (MS) data from a phosphorylation study of human embryonic stem cells (hESCs), (Van Hoof *et al.*, 2009). The SILAC experiment measured undifferentiated hESCs and hESCs differentiated with bone morphogenetic protein 4 (BMP4) at three time points (30 minutes, 60 minutes and 240 minutes) with two biological replicates at each time point. The data were processed using PVIEW (Khan *et al.*, 2009, 2011).

The dataset that contains the main objects required for use in this vignette is loaded by

```
> data(vanHoof)
```

2 Differential Expression Test

The function `DEtest` uses functions from the package **limma** for hypothesis testing via linear models on quantified protein- or peptide-level data. The input requires the data to be in the form of an `ExpressionSet`.

We will use `ExpSetVH` to test for differential expression at the peptide level. This `ExpressionSet` contains the quantified data in the `assayData` slot, the sequences and protein IDs in the `featureData` slot, and the experimental design, `phenoDataVH`, in the `phenoData` slot:

```
> ExpSetVH

ExpressionSet (storageMode: lockedEnvironment)
assayData: 12524 features, 6 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: a30min b30min ... b240min (6 total)
  varLabels: sampleIDs groups
  varMetadata: names labelDescription
featureData
  featureNames: 1 2 ... 13096 (12524 total)
  fvarLabels: Sequence Protein
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

Note that the `phenoData` contains the label `groups` that indicates which samples belong to each of the three factor levels (time points) of the experiment:

```
> phenoData(ExpSetVH)@data

        sampleIDs groups
a30min     a30min  min30
```

```
b30min      b30min  min30
a60min      a60min  min60
b60min      b60min  min60
a240min   a240min min240
b240min   b240min min240
```

Note that the `ExpressionSet` only contains 6 samples (and not 12) because with SILAC one measures the ratio of abundances of differentiated versus undifferentiated hESCs.

The function `DEtest` takes as input the `ExpressionSet` and the name of the variable subject to testing:

```
> outDE <- DEtest(ExpSetVH, label="groups")
```

```
groups (F-test)

Warning: nestedF multiple testing scheme not carried out due to missing p-values
groups (pairwise comparison): min240 - min30
groups (pairwise comparison): min240 - min60
groups (pairwise comparison): min30 - min60
```

If there are technical replicates in the data, set `reps = TRUE`. Note the warning: the function `decideTests()` with `method = "nestedF"` is used for multiple testing across the contrasts and proteins (or peptides); it requires complete data (see the limma User's Guide for more details). One option is to filter out peptides with too many missing values and impute the remaining NAs. The resulting object is a list containing two items:

1. A list of data frames comprised of the output of `topTable` applied to the `eBayes` model fit.
2. A data frame containing (moderated) statistics from an F-test.

```
> str(outDE)
```

```
List of 2
 $ resultsTable_limma  :List of 3
  ..$ min240 - min30:'data.frame':        12524 obs. of  9 variables:
  .. ..$ Sequence : chr [1:12524] "MVMIQDGPLPTGADKPLR" "RPLEDGDQPDAK" "VMLGETNPADSKPGTI
  .. ..$ Protein  : chr [1:12524] "FUBP3" "FUBP1" "NME1" "SMARCC1" ...
  .. ..$ logFC    : num [1:12524] 62.8 -27.8 -15.5 12.1 5.9 ...
  .. ..$ AveExpr  : num [1:12524] 20.98 33.44 46.34 32.28 9.42 ...
  .. ..$ t        : num [1:12524] 481.1 -213.3 -118.5 93.1 45.2 ...
  .. ..$ P.Value  : num [1:12524] 8.76e-06 4.02e-05 1.21e-04 1.90e-04 7.33e-04 ...
  .. ..$ adj.P.Val: num [1:12524] 0.0189 0.0432 0.0867 0.1021 0.3154 ...
  .. ..$ B        : num [1:12524] 0.944 0.926 0.879 0.837 0.528 ...
  .. ..$ df       : num [1:12524] 1 0 0 0 0 0 0 0 0 0 ...
  ..$ min240 - min60:'data.frame':        12524 obs. of  9 variables:
  .. ..$ Sequence : chr [1:12524] "SLDQEIARPLENENQEFLK" "VMLGETNPADSKPGTIR" "MVMIQDGPLP
  .. ..$ Protein  : chr [1:12524] "NES" "NME1" "FUBP3" "SMARCC1" ...
  .. ..$ logFC    : num [1:12524] -274.7 -95.2 62.7 54.5 -37.2 ...
  .. ..$ AveExpr  : num [1:12524] 91.6 46.3 21 32.3 13.5 ...
  .. ..$ t        : num [1:12524] -2106 -730 481 417 -285 ...
  .. ..$ P.Value  : num [1:12524] 5.52e-07 4.02e-06 8.77e-06 1.14e-05 2.34e-05 ...
  .. ..$ adj.P.Val: num [1:12524] 0.00119 0.00432 0.00615 0.00615 0.01005 ...
  .. ..$ B        : num [1:12524] 1.22 1.22 1.21 1.21 1.2 ...
  .. ..$ df       : num [1:12524] 1 0 0 0 0 0 0 0 0 0 ...
  ..$ min30 - min60 :'data.frame':        12524 obs. of  9 variables:
  .. ..$ Sequence : chr [1:12524] "SLDQEIARPLENENQEFLK" "VMLGETNPADSKPGTIR" "EKPVDLQNFG
  .. ..$ Protein  : chr [1:12524] "NES" "NME1" "SMARCC1" "AKAP8L" ...
  .. ..$ logFC    : num [1:12524] -274.74 -79.75 42.31 -37.29 -4.02 ...
  .. ..$ AveExpr  : num [1:12524] 91.62 46.34 32.28 13.51 2.86 ...
  .. ..$ t        : num [1:12524] -2106 -611.3 324.3 -285.9 -23.4 ...
  .. ..$ P.Value  : num [1:12524] 5.52e-07 5.60e-06 1.83e-05 2.32e-05 2.57e-05 ...
  .. ..$ adj.P.Val: num [1:12524] 0.00119 0.00602 0.01106 0.01106 0.01106 ...
  .. ..$ B        : num [1:12524] 1.19 1.19 1.18 1.18 3.77 ...
  .. ..$ df       : num [1:12524] 1 0 0 0 0 0 0 0 0 0 ...
 $ resultsTable_Fpvalue:'data.frame':        12524 obs. of  7 variables:
  ..$ Sequence : chr [1:12524] "SLDQEIARPLENENQEFLK" "VMLGETNPADSKPGTIR" "MVMIQDGPLPTGA
  ..$ Protein  : chr [1:12524] "NES" "NME1" "FUBP3" "SMARCC1" ...
  ..$ F        : num [1:12524] 2956842 306746 154265 96028 54321 ...
  ..$ df1      : int [1:12524] 2 2 2 2 2 2 2 2 2 2 ...
  ..$ df2      : num [1:12524] 0 0 0 0 0 2 0 3 2 0 ...
  ..$ p.Val    : num [1:12524] 8.23e-07 6.86e-06 1.31e-05 2.04e-05 3.47e-05 ...
  ..$ adj.P.Val: num [1:12524] 0.00177 0.00738 0.00937 0.01095 0.01493 ...
```

The data frames contain any original annotation and several statistics output by the function `topTable` including *p*-values and adjusted *p*-values. These *p*-values can be used as input in `runNetweavers`, e.g. for comparing timepoints 30 and 60 minutes you could use

```
> outDE_30to60 <- outDE$resultsTable_limma[["min30 - min60"]]
```

3 Peptide to Protein Summarization

If your quantified MS data are at the peptide level, you will need to summarize the data to the protein level in order to do network analysis.

3.1 Data Frame Summarization

One way to roll-up peptides to proteins is to input a data frame of the quantified data, along with the sequences, protein IDs and experimental design, into the function `customSummarizer`. The input data look like:

```
> head(dataVH)

                      Sequence Protein    a30min    b30min   a60min b60min
1                 AAAAAAAALQAK    RPL4 0.9390492 0.9430509       NA     NA
2             AAAAAWEEPSSGNGTAR    RCC2        NA        NA       NA     NA
3 AAAAGLGHPASPGGSEDGPPGSEEEDAAR  ARID3A        NA        NA       NA     NA
4               AAAGEFADDPCSSVK  CTNNA1 0.8568542        NA       NA     NA
5          AAAGQESEGPAVGPPQPLGK  ARID1A 1.8687121        NA 2.272664     NA
6       AAALVDEGLDPEEHTADGEPSAK   TSHZ3        NA 0.4784046       NA     NA
    a240min   b240min
1 0.9122702        NA
2 0.4053151        NA
3        NA 0.4606221
4 0.9437617        NA
5 1.7914869        NA
6        NA 0.4516751

> phenoDataVH

        sampleIDs groups
a30min     a30min  min30
b30min     b30min  min30
a60min     a60min  min60
b60min     b60min  min60
a240min   a240min min240
b240min   b240min min240
```

and the following shows how to specify the parameters in the function to get the desired output:

```
> esetSum1 <- customSummarizer(pepquant=dataVH, samplecols=3:8,
+                              peptidecol=1, proteincol=2,
+                              sumprot="median", phenodata=phenoDataVH)
> esetSum1

ExpressionSet (storageMode: lockedEnvironment)
assayData: 3200 features, 6 samples
  element names: exprs
```

```
protocolData: none
phenoData
  sampleNames: a30min b30min ... b240min (6 total)
  varLabels: sampleIDs groups
  varMetadata: names labelDescription
featureData
  featureNames: 1 2 ... 3200 (3200 total)
  fvarLabels: Protein
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
```

The output is an `ExpressionSet` with the summarized, protein-level data in the `assayData` slot. The slot `phenoData` contains the `phenoDataVH` and the slot `featureData` contains the `Protein` field from `dataVH`. The output can be input into `DEtest` for differential expression testing.

In this example we used the default function `median` to aggregate all peptide values of a given protein. You may also input the function `mean`. Missing values in the feature data (in the case of peptides that have not been quantified in all samples) are allowed but ignored.

3.2 ExpressionSet Summarization

If you already have your data in an `ExpressionSet`, e.g. `ExpSetVH`, then you can simply run

```
> esetSum2 <- esetSummarizer(ExpSetVH, sumprot="median")
```

to get the summarized data. The output can be input into `DEtest` for differential expression analysis.

3.3 p-Value Summarization

Suppose you have peptide-level data and prefer to test at this level as opposed to the protein level, but still want to run the data through a network analysis. The function `pvalueSummarizer` allows you to do this: it summarizes peptide p-values (e.g. generated by `DEtest`) and the output may be used in `runNetweavers`.

The method used to aggregate p-values is Fisher's method. It returns a single p-value for each set of p-values, x, using the chi-squared distribution function on $-2\sum \ln(x)$ with degrees of freedom equal to the length of x.

For example, we can use p-values from `outDE` generated in the previous section for comparing timepoints 30 and 60 minutes:

```
> pvals3060 <- outDE_30to60[,c("Protein","P.Value")]
> names(pvals3060)[2] <- "pvalue"
> pvalsSum <- pvalueSummarizer(pvals3060)
> head(pvalsSum)

  Protein omnibus.P.Val
1    AARS    0.10947111
2   ABCF2    0.71294855
3    ABT1    0.30946672
4   ACBD3    0.36964204
5   ACIN1    0.05578688
6    ACLY    0.32122988
```

Note that Fisher's Method assumes independence, which is clearly not true for peptides from the same protein, so these *p*-values might be too optimistic.

4 Find and Score Dense Clusters

The main function of the NetWeAvers algorithm is to perform network analysis. This section describes the core network analysis functions that find and score highly connected subgraphs within a protein-protein interaction network.

The first thing we want to do is filter an entire human network down to just those proteins of interest and their first interactors. We use `networkVH` which is the Reactome version 43 protein-protein interaction network (Matthews *et al.*, 2009):

```
> filtnet <- filterNetwork(networkVH, pvalsSum$Protein)
```

Then we find all the dense clusters of this network that have at least 4 proteins (default size) by taking random walks of length 10 (see Pons and Latapy (2006) for details):

```
> ## note: this will take a couple of minutes
> ## see clust_2_proteinVH for identical output
> outDC <- findDenseClusters(filtnet, pvalsSum$Protein, min_clus_size=4,
+                              steps=10)
> denseClusters <- outDC$clust_2_protein
```

The first item in the list is the data frame of cluster information that we will use in the following example. The second two items are special classes of the **igraph** package that can be used for visualizing the network (see Visualization section).

In order to score the clusters, we first need to calculate the protein weights and measures, which requires the use of *p*-values. We will use the summarized *p*-values from the comparison of 30 and 60 minutes (see previous section):

```
> ## change name of p-values for use in measureCalc
> names(pvalsSum)[2] <- "pvalue"
> ## calculate protein weights
> proteinWts <- weightCalc(filtnet)
> ## calculate protein measures
> proteinMsr  <- measureCalc(pvalsSum, proteinWts, weightamt=3)
> head(proteinMsr)
```

```
   Protein    pvalue     weight measure
1     A1CF        NA 0.04761905       0
2      A2M        NA 0.50000000       0
3     AARS 0.1094711         NA      NA
4 AASDHPPT        NA 1.00000000       0
5    ABCC8        NA 0.50000000       0
6    ABCF2 0.7129485         NA      NA
```

Note that we used a weight amount of 3, meaning we consider the weights nearly as important as p-values. Now that we have all the necessary data in `proteinMsr` we can score the clusters contained in `outDC`. The `cscoremethod` is the function that aggregates the protein measures per cluster and here we use the default mean:

```
> cscores <- scoreClusters(denseClusters, proteinMsr, cscoremethod="mean")
```

The larger the cluster score, the more significantly differentially regulated the proteins in the cluster are. We can use this statistic to rank our clusters, but we can also run a permutation test to see if the scores are large not by chance alone. The function `permTest` will do this, and here we perform 1000 permutations (`B`) and make sure we use the exact same `cscoremethod` that we did to find the `cscores`:

```
> set.seed(1234)
> cpvals <- permTest(B=1000, prot_data=proteinMsr,
+                    clust_2_protein=denseClusters,
+                    cluster_scores=cscores, cscoremethod="mean")
```

Note that we used `set.seed` before performing the permutation, which ensures the same results will be output from the permutation test in the next section. Finally, we can combine the cluster p-values with the cluster scores and membership information:

```
> cdata <- data.frame(Cluster_ID=denseClusters$clusID,
+                     Cluster_size=denseClusters$clusSIZE,
+                     Cluster_score=cscores,
+                     Cluster_pvalue=cpvals,
+                     Cluster_symbol=denseClusters[,-c(1,2)])
```

5 All-in-one: `runNetweavers`

The function `runNetweavers` is a wrapper for the functions in the NetWeAvers package. This function performs all of the steps in the NetWeAvers algorithm including filtering the protein network, finding dense clusters (only if output of `findDenseClusters` not already provided), and calculating protein weights, protein measures, cluster scores, and cluster p-values from a permutation test (optional). The following code runs NetWeAvers using the default values and produces the same output as above (`proteinMsr`, `filtnet`, and `cdata`) in a list:

```
> set.seed(1234)
> outNW <- runNetweavers(pvalsSum, filtnet, weightamt=3)

filtering network
finding dense clusters
calculating proteins weights
calculating protein measures
calculating cluster scores
calculating permutation pvalues
producing results tables

> str(outNW,list.len=5)

List of 3
 $ proteininfo:'data.frame':       2363 obs. of  4 variables:
  ..$ Protein: chr [1:2363] "A1CF" "A2M" "AARS" "AASDHPPT" ...
  ..$ pvalue : num [1:2363] NA NA 0.109 NA NA ...
  ..$ weight : num [1:2363] 0.0476 0.5 NA 1 0.5 ...
  ..$ measure: num [1:2363] 0 0 NA 0 0 NA 0 NA 0 0 ...
 $ filt_net   :'data.frame':       4823 obs. of  2 variables:
  ..$ intA: chr [1:4823] "AGPS" "AGPS" "KIF2A" "MYO1C" ...
  ..$ intB: chr [1:4823] "FAR1" "FAR2" "CCNB1" "RAB4A" ...
 $ clusterinfo:'data.frame':       52 obs. of  280 variables:
  ..$ Cluster_ID         : Factor w/ 52 levels "c1","c10","c11",..: 1 12 23 34 45 49 50
  ..$ Cluster_size       : int [1:52] 97 88 10 276 190 6 145 146 72 21 ...
  ..$ Cluster_score      : num [1:52] 0.246 0.6413 0.7331 0.0464 1.4122 ...
  ..$ Cluster_pvalue     : num [1:52] 0.996 0.438 0.314 1 0 0.416 0.719 0.125 0.745 0.0
  ..$ Cluster_symbol.X1  : chr [1:52] "EREG" "RPA1" "ACOT8" "ADCY6" ...
  .. [list output truncated]
```

6 Visualization

6.1 Writing Files for Cytoscape

Here we show how to export the data that can be used as input in Cytoscape (Shannon *et al.*, 2003) to make the following figure. First we will create a node attributes file by adding the cluster membership to the protein list:

```
> proteininfo <- outNW$proteininfo
> clusterinfo <- outNW$clusterinfo
> proteininfo$cluster <- NA
> for(i in 1:nrow(clusterinfo)){
+   proteininfo$cluster[proteininfo[,1]%in%clusterinfo[i,-c(1:4)]] <- i
+ }
```

Then we can write the file to be uploaded into Cytoscape as node attributes. Note that Cytoscape cannot parse missing values, so we replace them with -9:

```
> ## write.table(proteininfo,"nodeAttributes.txt",
> ##             sep="\t",row.names=F,quote=F,na="-9")
```

The following writes the network file:

```
> ## write.table(filtnet,"network.txt",sep="\t",row.names=F,quote=F)
```

and after a little bit of customization in Cytoscape we get, for example, the graph of cluster 5 in Figure 1.

6.2 Using igraph

It is also possible to visualize the resulting filtered network using the functions available in the **igraph** package. As an example, we will use the output from findDenseClusters to set up the plot (but not run, since it takes a while) of the graph with the proteins colored by cluster membership:

```
> dc_member <- outDC[['denseClus']]$membership
> dc_graph  <- outDC$subGraph
> cols      <- rainbow(length(unique(dc_member)))
>
> ## make the plot
> ## plot(dc_graph, vertex.color=cols, vertex.size=3, vertex.label=NA,
> ##      edge.color="grey")
```

6.3 Using RCystocape

See RCytoscape on how to use transfer the network and attributes from R to Cytoscape.

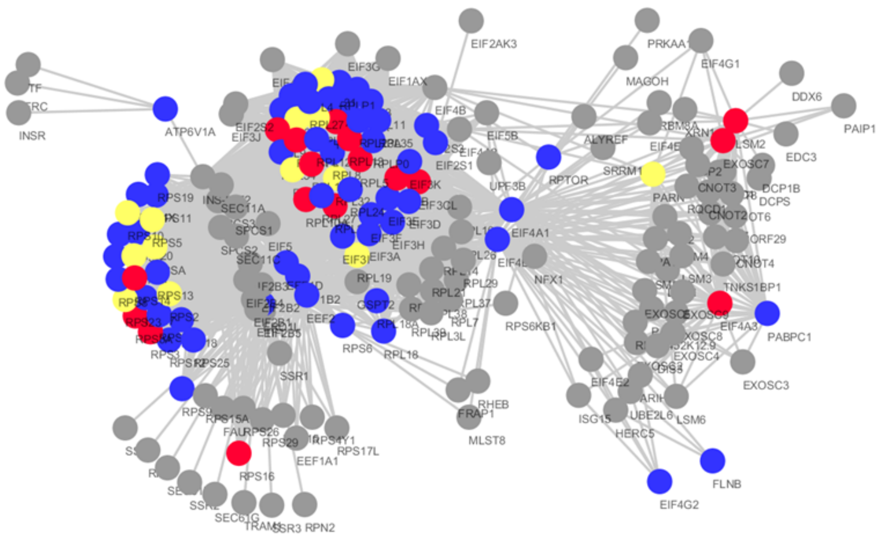

Figure 1: Image of cluster 5, generated with Cytoscape. A red node indicates a protein with a p-value less than 0.05, yellow less than 0.10, blue less than 1.00. A grey protein is one that is not in the Van Hoof dataset but is in the Reactome network.

References

Cerami EG, Gross BE, Demir E, Rodchenkov I, Babur O, Anwar N, Schultz N, Bader GD, and Sander C. (2010). Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Res.* **39(Database issue)**:D685-D690.

Khan Z, Bloom JS, Garcia BA, Singh M, and Kruglyak L. (2009). Protein Quantification Across Hundreds of Experimental Conditions. *Proc. Natl. Acad. Sci.* **06(37)**:15544-15548.

Khan Z, Amini S, Bloom JS, Ruse C, Caudy AA, Kruglyak L, Singh M, Perlman DH, and Tavazoie S. (2011). Accurate proteome-wide protein quantification from high-resolution 15N mass spectra. *Genome Biology* **12**:R122.

Matthews L, Gopinath G, Gillespie M, Caudy M, Croft D, de Bono B, Garapati P, Hemish J, Hermjakob H, Jassal B, Kanapin A, Lewis S, Mahajan S, May B, Schmidt E, Vastrik I, Wu G, Birney E, Stein L, D'Eustachio P. (2009). Reactome knowledgebase of biological pathways and processes. *Nucleic Acids Res.* **37(suppl 1)**:D619-D622.

Pons P and Latapy M (2006). Computing Communities in Large Networks Using Random Walks. *J. Graph Algorithms Appl.* **10(2)**:191-218.

Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, Amin N, Schwikowski

B, and Ideker T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research* **13(11)**:2498-2504.

Van Hoof D, Munoz J, Braam SR, Pinkse MW, Linding R, Heck AJ, Mummery CL, and Krijgsveld J. (2009). Phosphorylation dynamics during early differentiation of human embryonic stem cells. *Cell Stem Cell* **5(2)**:214-226.