Drug versus Disease (DrugVsDisease) package

1 Introduction

The Drug versus Disease (DrugVsDisease) package provides a pipeline for the comparison of drug and disease gene expression profiles where negatively correlated (enriched) profiles can be used to generate hypotheses of drug-repurposing and positively correlated (enriched) profiles may be used to infer side-effects of drugs. The package implements a number of processing options (for deriving ranked expression profiles) and different options for determining gene sets to use in Gene Set Enrichment Analysis. Currently the DrugVsDisease package can automatically process Affymetrix platforms hgu133a, hgu133a2 and hgu133plus2, which includes converting the probes to genes using the Biomart package. The DrugVsDisease package depends on the DvDdata package which contains a default set of reference drug and disease genomic expression profiles with corresponding clusters and Cytoscape compatible files for visualisation of the networks.

2 A Simple Example

As an example we use DrugVsDisease to download and automatically process an experiment from Array Express (which makes use of the ArrayExpress R package). This experiment (with accession E-GEOD-22528) compares patients with and without asthma. Using DrugVsDisease, and the default data provided by the DvDdata package, we compare the ranked list of differentially expressed genes to those of cell lines treated with 1309 different drug compounds. The output from the package gives the ranked lists of differential expression with associated p-values and the scores using Gene Set Enrichment Analysis to the reference data set. To do the analysis we first download the data from Array Express, convert the probes to genes using BiomaRt annotations, and calculate differential expression of genes using limma.

```
    > library(DrugVsDisease)
    > profileAE<-generateprofiles(input="AE",accession="E-GEOD-22528")
The downloaded source packages are in
        '/tmp/RtmpaA7Axj/downloaded_packages'
Reading in : /tmp/RtmpaA7Axj/GSM559378.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559379.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559377.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559376.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559380.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559374.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559373.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559375.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559372.CEL
Reading in : /tmp/RtmpaA7Axj/GSM559371.CEL
Background correcting
Normalizing
Calculating Expression
Note: Ensembl genes do not match genelist in reference data. Consider uploading pre-processed I
[1] "Fitted linear models: "
[1] "factorasthma-factornormal:all"
[2] "factornormal-factorasthma:all"
```

We can see the data that has been generated, the names of profileAE are ranklist for the differential expression profiles and pvalues for the corresponding pvalues

```
> names(profileAE)
[1] "ranklist" "pvalues"
```

Within the ranklist or pvalues list, there is a matrix containing all fitted contrasts. DrugVs-Disease expects that the profiles to be generated will contain either drug perturbations or a set of disease profiles. Using this information, DrugVsDisease searches the annotation files to find an allocate a main factor. This is disease.state for a disease profile or compound for a drug profile (based on the Array Express Factor values). The main factor is used as the explanatory factor in the regression model. In this example this is the only factor which can be fitted. In some cases, there will be additional factors, such as sex or time. Where these are available a second (additive) factor is added to the linear model automatically (in addition to the single factor model based on the main factor). The main factor in this example is given by "disease state" and the fitted contrasts are shown below.

```
> colnames(profileAE$ranklist)
[1] "factorasthma-factornormal:all"
[2] "factornormal-factorasthma:all"
```

As the above output shows, DrugVsDisease identifies the main factor but not which of the levels in the factor is the control. Consequently, all possible pairwise contrasts are fitted for the given levels of the factor. This means that there will be redundant contrasts which are not interesting to the user, in this example, asthma-normal is interesting whilst normal-asthma is not. The next step is therefore, to select those contrasts to be analysed further which is done using the selectrankedlists function.

```
> selprofiles<-selectrankedlists(profileAE,1)
```

Given the data to be analysed, we now use the classifyprofile function to calculate enrichment scores between the input profiles and the data available from the DvDdata package (this can be changed to a users custom data). This data is based on the connectivity map drug screening. Here we use the default options to analyse the data. Note as this is an example, the signif.fdr value for the false discovery rate is set to 1.1, this is simply to restrict the situation where the process stops due to their being no significant matches, in practice the user should select a value greater than 0 and less than 1, 0.05 is a common choice.

```
> default<-classifyprofile(data=selprofiles$ranklist, signif.fdr=1.1)
```

Number of Significant results greater than 10 Using top 10 hits - consider using average linkage instead

The table shows the output from classifyprofile. The distance between the input profile and all scores found to be statistically significant are given. They also include the cluster number of each of the drug profiles the input profile is connected to as well as the sign of the connection. A positive (1) for Running sum Peak Sign (RPS) denotes a positive correlation and (-1) a negative correlation.

	ES Distance	Cluster	RPS
foliosidine	0.86	2	-1.00
pyridoxine	0.86	57	1.00
nordihydroguaiaretic_acid	0.87	22	-1.00
sertaconazole	0.87	88	1.00
staurosporine	0.87	11	1.00
gibberellic_acid	0.87	59	1.00
decitabine	0.87	44	-1.00
benzthiazide	0.87	29	1.00
5151277	0.87	5	-1.00
fenspiride	0.87	73	1.00

3 Importing Data

DrugVsDisease has four options for processing Affymetrix microarray data, the options are selected using the "input" argument of generateProfiles. The options are "AE" for downloading raw CEL files from the Array Express data base, which uses the functionality of the ArrayExpress package, "localAE" allows the processing of Array Express CEL and SDRF files which have been downloaded and stored locally. From Gene Expression Omnibus, processed GDS files can be downloaded and analysed using the GEOquery package by selecting the "GEO" option. Finally the "local" option is used for any other Affymetrix files stored locally where the customfile option is used to supply the associated experimental design.

Array Express Downloads In this first example we process data (where DrugVsDisease makes use of the ArrayExpress package), to analyse the experiment E-GEOD-22528 which looks at differences in patients with and without asthma.

```
> profileAE<-generateprofiles(input="AE",accession="E-GEOD-22528")
```

The output shows the contrasts which have been fitted by limma. The generateprofiles function automatically detects and fits all possible pairwise contrasts, giving contrasts asthma-normal and normal-asthma. To fit these contrasts DrugVsDisease uses the information in the expression set object created by the normalisation procedure from the affy package, which uses the information from the Array Express database to annotate the experimental design. The experimental design is contained in a standard SDRF file format and experimental factors should be prefixed with "Factor Value". DrugVsDisease uses this information to identify factors to include as explanatory variables in the regression. If a factor of interest has not been labelled as a "Factor Value" in the SDRF annotation it may be necessary to download the Array Express files and provide a manual annotation of the data using the option in DrugVsDisease for locally stored data.

Locally stored Array Express files Files downloaded from Array Express that have been stored locally can be processed using the input="localAE" option. Inputs required are the path to the CEL files and allows the sdrf file to be used for providing factor information. The path to the folder containing the CEL files is passed to R using the celfilepath parameter and the corresponding sdrf file path using parameter sdrfpath. Both these parameters therefore expect a string as input.

Locally stored files To process a local data (not from Array Express or GEO), provide DrugVsDisease with a path to the folder containing CEL files and one to the file containing the experimental design. The file containing the experimental design should have one column giving the names of the CEL files contained in the folder 'celfilepath' and should have column name "CEL" as well as either a "disease" column giving the disease state factor, or "compound" column for drug profiles. Any

other factors can be provided in additional columns. The option input="local" is used to specify local files. Similar to the localAE option the path to the folder containing the CEL files is given by parameter celfilepath and an additional character string containing the experimental design via parameter customfile.

GEO download For downloading GDS files the input parameter is set to "GEO" and the relevant database reference on GEO is passed via the accession parameter. With GDS downloads sometimes the platform annotation is not available in the metadata, in this case it can be passed to DrugVsDisease using the annotation parameter as shown below (annotation="hgu133a"). Case is specified as "disease" (this is the default option). The case parameter tells the package which factor values to search for in the meta data e.g. disease.state for a disease profile or dose for a drug profile. In some instances the metadata associated with the GDS file may not have the main factor as one of "disease state" or "disease.state" for a disease profile, or "dose" for a compound profile. For example the GDS profile 1479 has a main factor labelled as "specimen" in the GEO database. Using this information and the optional factorvalue parameter we can process the data by providing DrugVsDisease with the name of the factor in the experimental design.

```
> gds1479<-generateprofiles("GEO",accession="GDS1479",
+ annotation="hgu133a",factorvalue="specimen",case="disease")
```

4 Normalisation

Where raw CEL files as opposed to processed GDS files are analysed the normalisation can be done in DrugVsDisease using one of rma (default) or mas5. For example, to use mas5 normalisation set normalisation="mas5".

5 Calculating Differential Expression

Ranked lists of differential expression (either log fold change, difference or t-statistic) and p-values are automatically calculated using limma. Differential expression is calculated between case and controls where ćaseís either a disease or compound factor. For example to change to the t statistic instead of the default log fold change, add the option statistic="t" to the generateprofiles command.

The main 'case' factor is either identified from the phenotype (pData) of the expression set (eset) object produced following normalisation (see affy package), or it is specified by the user in the customfile option (see previous section). DrugVsDisease uses the metadata from GEO or Factor values of Array Express to automatically detect additional factors such as sex or time. Where the number of replicates allows, the data is stratified by the second factor. The differential expression for disease or compound factor is then calculated for a subset of the data split according to the different levels of a second factor e.g. male and female.

6 Selecting Ranked Lists

As the generateprofiles function automatically generates profiles for every possible pairwise contrast there are likely to be some redundant proflies which the user does not wish to analyse further. The selectrankedlists function allows the user to select those profiles to be further analysed. To select just the 3rd and 18th profile profile from gds1479 and the first from the Array Express profile E-GEOD-22528:

```
> sel1479<-selectrankedlists(gds1479,c(3,18))

> sel22528<-selectrankedlists(profileAE,1)
```

7 Classifying Profiles

The function *classifyProfile* has four main phases of, calculating Enrichment Scores, determining significant Enrichment Scores, assigning profiles to clusters and producing a network visualisation.

7.1 Enrichment Scores:

The Enrichment Scores (ES) are calculated for up-regulated and down-regulated genes separately then averaged.The ES is the maximum value of the running sum statistic for input profiles compared to "inverted" reference profiles. For example, when using a disease profile of differentially expressed genes for case versus control, a significant score would indicate a potential therapeutic relationship. Conversely differentially expressed genes of control versus case (the inverse) would indicate a potential side-effect of that drug. Using the default options for classifyprofile the data generated from generateprofiles can be analysed as follows (where the cytoout=FALSE option indicates that no Cytoscape compatible format should be produced):

```
> default<-classifyprofile(data=sel22528$ranklist, signif.fdr=1.1)
```

A two sided test is used by DvD whereby a gene set of the same size is also generated from the reference data and compared to the input profile. The gene sets for enrichment analysis, can be selected by one of three methods which is specified using the type parameter to classifyprofile The options are:

- type="fixed": A fixed set size (default 100).
- type="pvalues": P-values can be used to determine the set size by selecting probes for a given false discovery rate (default 5%).
- type="range": A fixed range of set sizes (default is between 100 and 2000 at intervals of 100).

The classifyprofile function requires the ranklist of differential expression to be passed to the data parameter. The output from generateprofiles is a list with one element containing the ranked lists (ranklist) which is passed as the data option to classifyprofile. The second element contains the associated p-values which should be used as the pvalue input in classifyprofile when the type is set to dynamic.

Example classifying the profile to default drug clusters

```
> c1<-classifyprofile(data=sel22528$ranklist,type="fixed",lengthtest=100,
+ cytoout=FALSE,noperm=100)
```

Alternatively, classify the profile using a range of different gene set sizes (100 and 200).

```
> c2<-classifyprofile(data=sel22528$ranklist,type="range",range=c(100,200),
+ cytoout=FALSE,noperm=100)
```

Finally, using the dynamic option:

```
> c3<-classifyprofile(data=sel22528$ranklist,pvalues=sel22528$pvalues,
+ cytoout=FALSE,type="dynamic",dynamic.fdr=0.5,adj="BH")
```

7.2 Significant Enrichment Scores and Multiple Hypothesis Correction:

Significance of Enrichment Scores (ES) is determined by comparing the scores for a given input profile to a set of randomly generated reference profiles. These scores are used to define an empirical p-value. The number of random profiles generated can be set via the noperm option (default is 2000), see above examples with noperm=100.

Multiple hypothesis correction is done when assessing the significance of the enrichment scores which have been calculated, the maximum number of significant results to output can be specified using the no.signif parameter (default 10). Two methods are available for multiple hypothesis correction (MHC) of ES q-value or Benjamini-Hochberg and the False discovery rate is set using the signif.fdr parameter. MHC is also used when the dynamic option is selected to correct genome-wide p-values of differential expression, this false discovery rate is set using dynamic.fdr. Example. Classifying using 20% F.D.R for significant results and 15% F.D.R for the significance of differential expression (only required when type="dynamic"). Further to change to use the Benjamini Hochberg correction method instead of q-value approach of Tibshirani (this is applied to correction for differential expression where type="dynamic" and to correct for multiple hypothesis testing of significant enrichment scores) we have adj="BH".

```
> dynamicfdr<-classifyprofile(data=sel22528$ranklist,pvalues=sel22528$pvalues,
+ cytoout=FALSE,type="dynamic",dynamic.fdr=0.5,signif.fdr=0.15,adj="BH")
```

7.3 Clustering:

Two methods for assigning profiles to clusters are implemented in DrugVsDisease. One is single linkage, whereby an edge is drawn between the input and every significant match to the reference data. The second method is average linkage where the profile is assigned to the cluster with the largest average enrichment score over all nodes in the cluster, the 'average' score is set as either the arithemetic mean or the median via the avgstat parameter.

Example changing to average clustering (single is default) and using median as the statistic (mean is the default).

```
> clusteraverage<-classifyprofile(data=sel22528$ranklist,cytoout=FALSE,
+ clustermethod="average",avgstat="median")
```

7.4 Custom Clusters

As well as the default disease and drug clusters contained in the DvDdata package, DrugVsDisease offers the option to use custom reference data sets and clusters provided by the user. To use the custom cluster options the user needs to provide genome wide expression profiles for the reference data set, using the customRefDB parameter. The rank profiles should be in rank decreasing order such that the highest up-regulated has the lowest numerical rank. This can be either an R matrix containing the expression profiles or a string containing the path to a file containing the reference data. Where a path to a file is provided the file will be read using the read.table option in R and will expect rownames in the first column and the remaining columns should contain the rank profiles and the column names will be used as the names for each of the profiles in the reference data set. These names should match the node names for the customClusters parameter. As with the reference data, the custom Cluster option can take either an R matrix or a path specifying the filename for the custom Clusters data. The first column should be either "Drug" or "Disease" depending

on whether the profiles are drug or disease profiles respectively. The second column should be headed "Cluster" and contain the cluster number of the corresponding node in the network. If a cytoscape output is to be produced it is also necessary to provide a corresponding SIF and edge attribute file via parameters customsif and customedge. The DrugVsDisease package contains example data for the custom clustering option, which are derived from a subset of the reference drug data from the DvDdata package. This is shown in the example below, with the example data: customDB,customClust,customedge and customsif.

```
> data(customDB,package="DrugVsDisease")
> data(customClust,package="DrugVsDisease")
> data(customedge,package="DrugVsDisease")
> data(customsif,package="DrugVsDisease")
> customRefData<-classifyprofile(data=sel22528$ranklist, lengthtest=100,
+ customRefDB=customDB, customClusters=customClust,
+ customedge=customedge, customsif=customsif, cytoout=FALSE,signif.fdr=1.1)
Number of Significant results greater than 10 Using top 10 hits - consider using average linkag
```

7.5 Visualisation:

From R a Cytoscape compatible file is produced where the edge attribute is 1-ES, to give a distance metric of the input profile to the reference node. Nodes are the input profiles and the corresponding profiles (nodes) they are assigned to. Using the Cytoscape plug-in these results are automatically displayed after calculation. To produce Cytoscape compatible SIF and Edge Attribute files, the cytoout=TRUE option should be used as well as providing a filename via the cytofile parameter which is used to previx the name of the SIF and Edge attribute files.

```
> CytoOut<-classifyprofile(data=sel22528$ranklist,cytoout=TRUE,
+ cytofile="CytoscapeOutput",adj="BH")
```

References

[Hu *et al.*, 2009] Hu G, Agarwal P (2009) Human Disease-Drug Network Based on Genomic Expression Profiles, *PLoS ONE*, **4(8)**: e6536.

[Shigemizu *et al.*, 2012] Shigemizu D, Hu Z, Hung J-H, Huang C-L, Wang Y, et al. (2012) Using Functional Signatures to Identify Repositioned Drugs for Breast, Myelogenous Leukemia and Prostate Cancer. *PLoS Comput Biol* **8(2)**: e1002347.

[Sirota *et al.*, 2011] Sirota M *et al.* (2011) Discovery and Preclinical Validation of Drug Indications Using Compendia of Public Gene Expression Data. *Sci Trans Med*, **3:96ra77**.

[Subramanian *et al.*, 2005] Subramanian A *et al.* (2005) Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *PNAS*, **102**(43), 15545-15550.

[Lamb *et al.*, 2006] Lamb J *et al.* (2006) The Connectivity Map: Using Gene-Expression Signatures to Connect Small Molecules, Genes, and Disease. *Science*, **313**(5795), 1929-1935.

[Gentleman *et al.*, 2004] Gentleman R *et al.* (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, **5**(10), R80.

[Parkinson *et al.*, 2010] Parkinson et al. (2010) ArrayExpress update—an archive of microarray and high-throughput sequencing-based functional genomics experiments. *Nucl. Acids Res.*,doi: 10.1093/nar/gkq1040.

[Edgar *et al.*, 2002] Edgar R, Domrachev M, Lash AE. (2002) Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucl. Acids Res*, **30**(1):207-10

[R 2008] R Development Core Team (2008). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0

[Cline *et al.*, 2007] Cline *et al.* (2007) Integration of biological networks and gene expression data using Cytoscape. *Nature Protocols*, **2**, 2366-2382.