Vignette for DiseaseCellTypes

Alex J. Cornish

Introduction

This vignette explains how to use the functions within the **DiseaseCellTypes** package to identify disease-associated cell types, create cell type-specific interactomes and build cell type-based diseasomes.

Two method are implemented within the package:

- Gene set compactness (GSC)
- Gene set overexpression (GSO)

The performance of these two methods is compared in the accompanying paper (in preparation). The code contained within this vignette, along with the data provided within the **DiseaseCellTypes** package, also allows the user to reproduce the results shown in the accompanying paper. Due to the time required to reproduce the results in full, much of this code is commented.

Overview of the GSC method

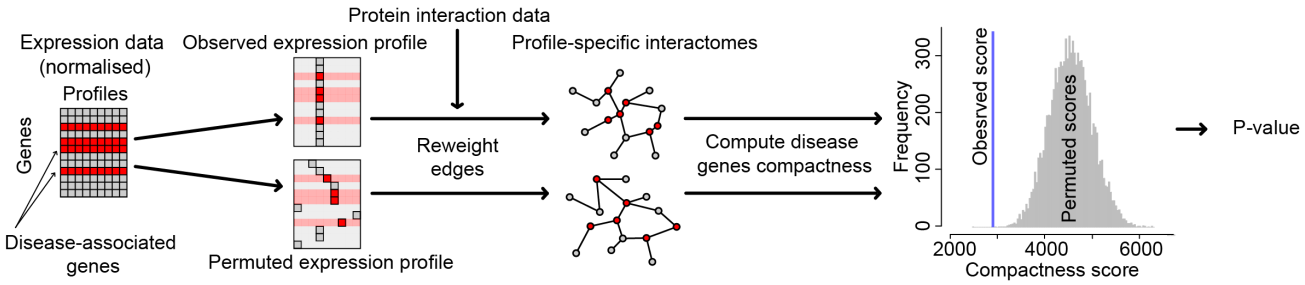
As described in the accompanying paper, the GSC method identifies cell types associated with diseases by measuring the clustering of disease-associated genes on cell type-specific interactomes. While we only identify disease-associated cell types in the paper and in this vignette, the GSC method is not restricted to cell types and could be used to identify other diseases-associated biological entities, such as tissues.

The method is based on the compactness score defined by Glaab et al. (2010) as the mean distance between pairs of vertices in a set of vertices on a network. Through a permutation-based approach, it is possible to use the compactness score to identify networks upon which sets of vertices cluster (Cornish et al. 2014). Here we use random walks with restart (RWR) to compute the distance between vertex pairs, similar to the method described by Kohler et al (2008).

Context-specific interactomes are created by integrating expression data with protein-protein interaction (PPI) data. Edges are re-scored using the product of the percentile-normalised relative expression scores (Hu et al. 2011). This is similar to the edge reweight method of Magger et al (2012). However, rather than using cutoff expression scores, this method uses continuous relative expression scores. In this vignette, we use gene expression data from 73 cell types from the FANTOM5 project

(Forrest et al. 2014). PPI data is taken from the STRING database (Franceschini et al. 2013). We also use disease-gene associations from the DisGeNET database (Bauer-Mehren et al. 2010).

A cell type-specific interactome is created for each of the cell types. These are referred to as the observed interactomes. In order to determine whether the genes associated with a disease are significantly more compact on the observed interactomes, permuted interactomes are created by permuting the percentile-normalized expression scores and using these profiles to create the interactomes. The compactness score of the disease-associated gene set is computed for each observed and permuted interactome and an empirical p-value produced by computing the number of permuted compactness scores smaller than each observed score.



Guide to identifying disease-associated cell types

In this section, we will describe how to transform raw gene expression values to percentile-normalized relative gene expression scores, create a global interactome using the **igraph** package (Csardi et al. 2006) and identify disease-associated cell types using the GSC and GSO methods.

Formatting the global interactome

The PPI data used by the GSC method should be in the format of an **igraph** object. Gene names should be present under the vertex attribute **name**. It is not necessary to create the cell type-specific interactomes to identify the disease-associated cell types, as they are created within the **gene.set.compactness** function. However, cell type-specific interactomes can be created separately using the **score.edges** function for various other uses. PPI data from the STRING database has been saved within the **DiseaseCellTypes** package.

```
# load DiseaseCellTypes package
set.seed(999) # for reproducibility
require(DiseaseCellTypes)
```

```
## Loading required package: DiseaseCellTypes
## Loading required package: igraph
```

```
# load STRING PPI data
data(edgelist.string)
edgelist.string[1:2, ]
```

```
##                idA             idB weight
## 1 ENSG00000049245 ENSG00000072818    808
## 2 ENSG00000010256 ENSG00000099624    878
```

```
# create igraph object
g <- graph.edgelist(as.matrix(edgelist.string[, c("idA", "idB")]), d
```

Normalizing gene expression data

Before running the GSC or GSO methods, it is first necessary to produce a matrix of percentile-normalized relative expression scores. Each row in this matrix should represent a gene and each column a cell type.

The FANTOM5 project make their gene expression data available for download. This data is provided as expression counts for transcription start sites (TSS), rather than genes. We convert the TSS-wise expression values to gene-wise values by summing the scores of the TSSs associated with each gene (Sardar et al. 2014). This data has been saved within the **DiseaseCellTypes** package.

We will now load the gene-wise expression values and use the **expression.transform** function to transform these values to percentile-normalized relative expression scores.

```
# load FANTOM5 gene expression data
data(expression.fantom5)
expression.fantom5[1:3, 1:2]
```

```
##                  acinar cell alveolar epithelial cell
## ENSG00000000003  18.61736713                 30.82841
## ENSG00000000005   0.09612222                  0.00000
## ENSG00000000419 112.06766580                 56.34010
```

```
# transform to percentile-normlized relative expression scores
expression <- expression.transform(expression.fantom5)
expression[1:3, 1:2]
```

```
##                 acinar cell alveolar epithelial cell
## ENSG00000000003   0.3794111                0.6573934
## ENSG00000000005   0.2691600                0.1177473
## ENSG00000000419   0.7221260                0.5336653
```

Identifying disease-associated genes

Disease-associated genes contain be obtained from a number of databases, including DisGeNET and OMIM (Hamosh et al. 2005). The **DiseaseCellTypes** package contain genes associated with diseases from the DisGeNET database.

```
# load disease-gene associations
data(disease.genes)
disease.genes[["stomach ulcer"]]
```

```
## [1] "ENSG00000165841" "ENSG00000106366" "ENSG00000104368" "ENSG0(
## [5] "ENSG00000137673"
```

Running the GSC method

We will now apply the GSC method to genes known to be associated with bipolar disorder. Due to the time required to run the code (about 10 minutes on a 2.5 GHz Intel Core i5 processor with 8GB of RAM), the script is commented out.

```
# identify cell types associated with panic disorder-
disease <- "panic disorder"
genes <- disease.genes[[disease]]

# remove genes not contained within the network
genes <- genes[genes %in% V(g)$name]

# run the gene set compactness function
#res.gsc <- gene.set.compactness(expression, genes, g, n.perm=100)
```

The results of this code has been saved within the package.

```
# load the results of the gene.set.compactness function
data(res.gsc)
```

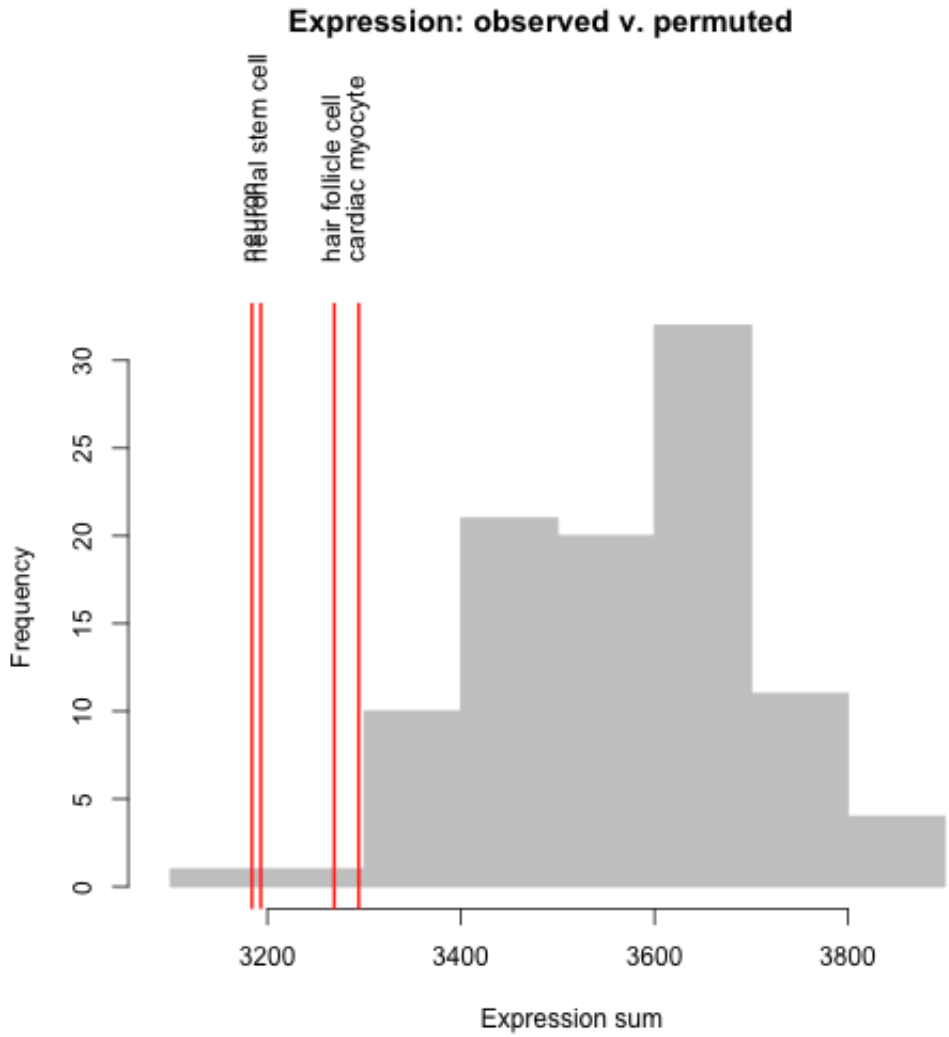
Each p-value contained within the **dct** object output by the **gene.set.compactness** function represents the likelihood of observing clustering at least as strong as that observed, given the PPI data and that the context is not associated with the disease.

By plotting the results, the observed scores can be compared to the permuted scores. The gray histogram represents the distribution of the permuted scores. The red lines represent the observed scores of contexts with p-values less than a given cutoff. By default, this cutoff is **0.05**. When using these results, p-values should be adjusted for multiple testing. The p-values output by the function have not been adjusted.

```
# look at the p-value of the neuron
res.gsc$pval[["neuron"]]
```

[1] 0.01

```
# plot the results to compare permuted and observed scores
plot(res.gsc)
```



Running the GSO method

The GSO method is similar to the GSC method, although it does not use cell type-specific interactomes and instead identifies cell types in which the mean expression of the disease-associated genes is higher than expected. Like the GSC method, the GSO method computes an empirical p-value by permuting the gene expression profiles and comparing the observed mean expression score of the disease associated genes to the distribution of permuted mean expression scores.

Here we will apply the GSO method to the gene expression data and disease-associated genes used in the previous section.

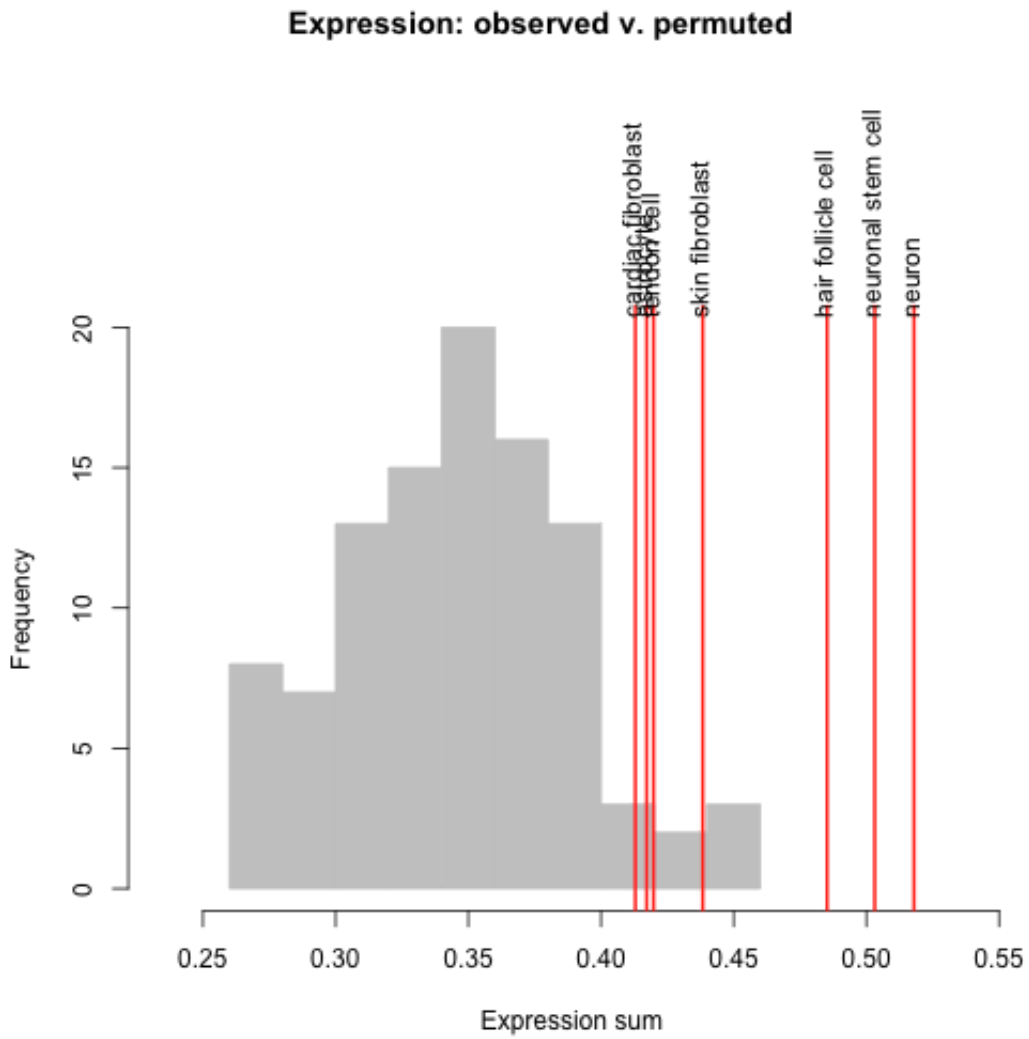
```
# run the GSO method
res.gso <- gene.set.overexpression(expression, genes, n.perm=100)
```

Like the **gene.set.compactness** results, we can plot the results of the **gene.set.overexpression** function to compare the observed and permuted mean expression scores.

```
# look at the p-value of the neuron
res.gso$pval[["neuron"]]
```

[1] 0.01

```
# plot the results to compare permuted and observed scores
plot(res.gso)
```



Creating cell type-specific interactomes

Cell type-specific interactomes are created within the **gene.set.compactness** function. However, they can be created outside of the function using the **score.edges** function. In this example, we will create a neuron-specific interactome:

```
# create global network
data(edgelist.string)
g <- graph.edgelist(as.matrix(edgelist.string[, c("idA", "idB")]), d

# load and transform expression data
data(expression.fantom5)
expression <- expression.transform(expression.fantom5)

# create neuron-specific interactome
g.neuron <- score.edges(g, expression.fantom5[, "neuron"])
```

By default, the edge weights are saved under the edge attribute **score** and the gene expression scores under the vertex attribute **expression**.

```
# edge scores
E(g.neuron)$score[1:5]
```

```
## [1]     5.502241 83568.995179 40957.857558 57139.770651 23717.583
```

```
# vertex expression scores
V(g.neuron)$expression[1:5]
```

```
## [1]  68.72772208   0.08005853 244.74958234 341.44693682 167.34597
```

Guild to recreating the results of the accompanying paper

In this section, we will provide details of how to recreate the results given in the accompanying paper.

The data required to recreate the results, such as the FANTOM5 gene expression data, STRING PPI data and DisGeNET disease-gene associations are provided within the package. Where this data is obtained from, the license agreements and any formatting or changes made are detailed in the accompanying manual. Due to software required to conduct the text mining, it has not been possible to include the code required to rerun the text-mining. However, further details about how to recreate the text mining results can be obtained by e-mailing the author of this vignette (**a.cornish12 at imperial.ac.uk**). The results of the text-mining are included within the package.

Identifying disease-associated cell types

Both the GSC and GSO method were used to identify the disease-associated cell types. This section contains the code required to rerun this analysis. Due to the time required to run the code, it is commented out.

```
# # parameters
# n.diseases <- 100 # number of diseases
# n.perm <- 10000 # number of permutations
# rwr.r <- 0.7 # the RWR restart probability
# rwr.cutoff <- 1e-5 # the RWR iteration termination cutoff
# parallel <- 10 # the number of cores to use
#
# # load and create global network
# data(edgelist.string)
# g <- graph.edgelist(as.matrix(edgelist.string[, c("idA", "idB")]),
# genes.g <- V(g)$name
```

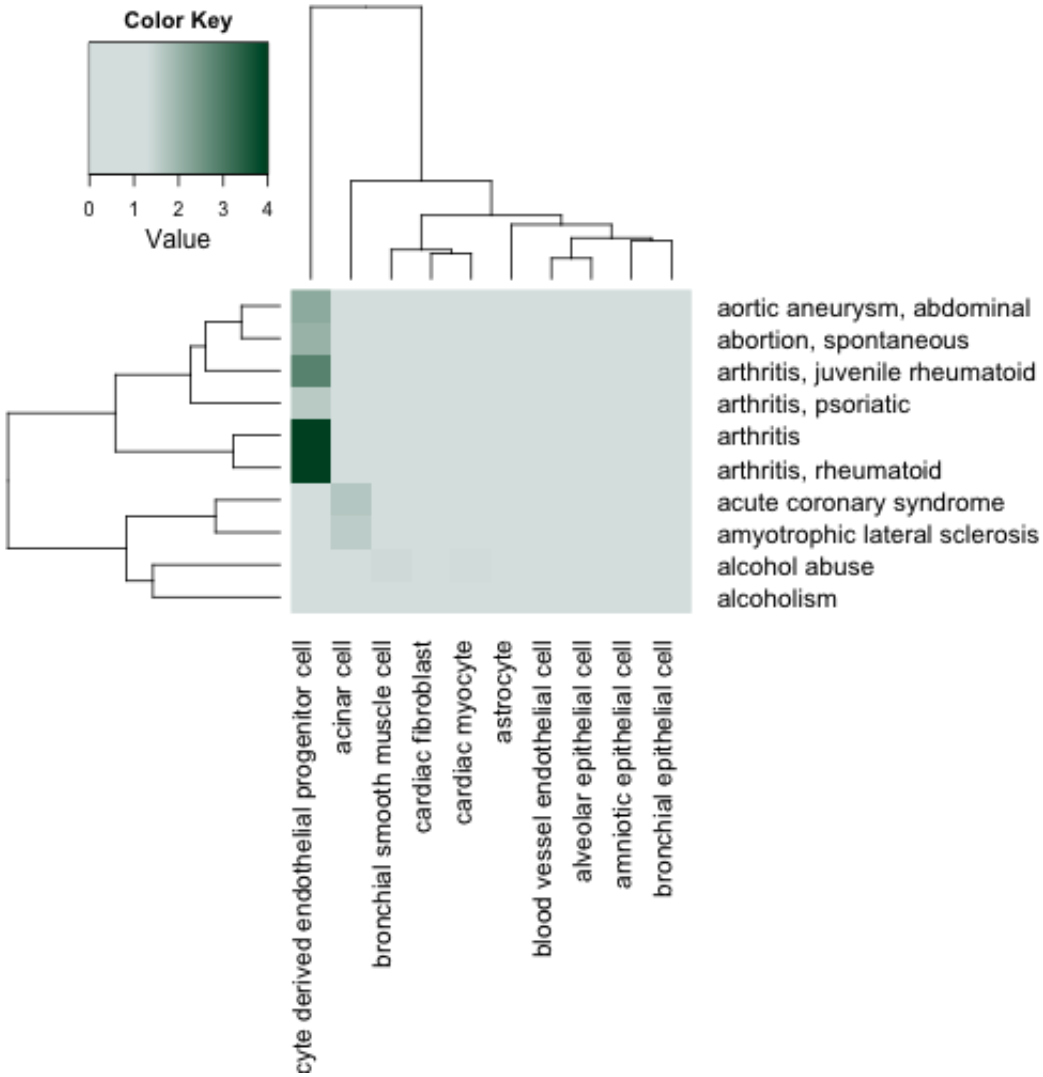
```
#
# # load and format gene expression data
# data(expression.fantom5)
# expression <- expression.transform(expression.fantom5)
# expression <- expression[rownames(expression) %in% genes.g, ]
# genes.expression <- rownames(expression)
#
# # load disease-associated genes
# data(disease.genes)
#
# # remove disease-associated genes not found in network or expressi
# disease.genes <- sapply(disease.genes, function(genes) genes[genes
# disease.genes <- sapply(disease.genes, function(genes) genes[genes
#
# # remove diseases not identified as diseases in MeSH
# diseases.to.remove <- c("dna damage", "oxidative stress", "impaire
# disease.genes <- disease.genes[!names(disease.genes) %in% diseases
#
# # select the 100 diseases with the most disease-associated genes i
# disease.genes <- disease.genes[order(sapply(disease.genes, length)
#
# # for each disease, apply the GSC method
# res.full.gsc <- list()
# for (disease in names(disease.genes)) {
#     res.full.gsc[[disease]] <- gene.set.compactness(expression, di
# }
#
# # for each disease, apply the GSO method
# res.full.gso <- list()
# for (disease in names(disease.genes)) {
#     res.full.gso[[disease]] <- gene.set.overexpression(expression,
# }
#
# # convert to patrix of p-values
# pvalues.gsc <- t(sapply(res.full.gsc, function(disease) disease$pv
# pvalues.gso <- t(sapply(res.full.gso, function(disease) disease$pv
```

The results produced using this script were used in the accompanying paper and are saved within the package. We will now load these results and produce a heatmap showing the associations. The heatmap shown in the accompanying paper contains the diseases and cell types involved in at least 2 associations ($p < 0.1$). Here, for simplicity, we will include only 10 diseases and cell types.

```
# load the associations
data(pvalues.gsc)

# compute the -log10 of the p-values
pvalues.ml10 <- -log10(pvalues.gsc)

# plot heatmap
require(gplots)
```

```
## Loading required package: gplots
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##     lowess
```

```
n.cols <- 500
cols <- colorRampPalette(c("#D3DDDC", "#02401B"))(n.cols)
col.breaks <- c(0, seq(-log10(0.05), max(pvalues.ml10), length.out=r
heatmap.2(pvalues.ml10[1:10, 1:10], Rowv=TRUE, Colv=TRUE, dendrogran
```



Comparing the results with text mining

Next, we will compare the associations identified by the GSC and GSO methods with the associations identified by the text mining. Since the text-mining identifies cell types affected by a disease, as well as cell types causing a disease, these associations cannot be considered gold standards. However, by comparing the results of the methods, we are able to compare the methods efficacy.

First, we load the associations.

```
# load the associations
data(pvalues.gsc)
data(pvalues.gso)
data(pvalues.text)

# ensure that the rows and columns of the matrices are ordered the s
diseases <- rownames(pvalues.gsc)
cell.types <- colnames(pvalues.gsc)
pvalues.gso <- pvalues.gso[diseases, cell.types]
pvalues.text <- pvalues.text[diseases, cell.types]
```

We next adjust the p-values using the Benjamini-Hochberg (BH) procedure for multiple testing:

```
# adjust p-values for multiple testing
pvalues.adj.gsc <- array(p.adjust(as.vector(pvalues.gsc), method="BH
pvalues.adj.gso <- array(p.adjust(as.vector(pvalues.gso), method="BH
pvalues.adj.text <- array(p.adjust(as.vector(pvalues.text), method='
```

We can then compare the association overlap at a false discovery rate (FDR) of 10%:

```
# count the number of associations supported by text
res <- matrix(NA, 2,2, dimnames=list(c("supported by text", "not sup
cutoff <- 0.10
res["supported by text", "GSO"] <- sum(pvalues.adj.gso < cutoff & pv
res["supported by text", "GSC"] <- sum(pvalues.adj.gsc < cutoff & pv
res["not supported by text", "GSO"] <- sum(pvalues.adj.gso < cutoff
res["not supported by text", "GSC"] <- sum(pvalues.adj.gsc < cutoff
res
```

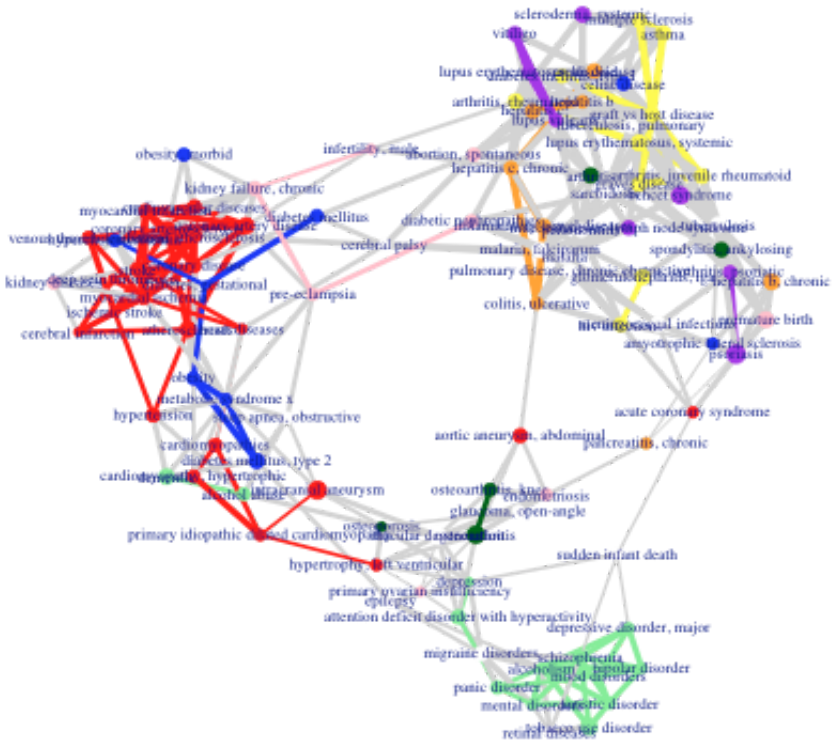
```
##                         GSO GSC
## supported by text       202 205
## not supported by text   283 234
```

Creating the cell type diseasome

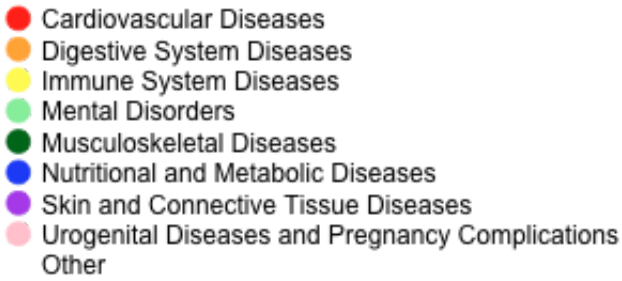
Using the associations identified by the GSC method (or the other methods), we are able to create a cell type-based diseasome. This is done by measuring the correlation between pairs of diseases, with respect to the -log10 of their cell type association p-values, then connecting each disease to the four diseases with which it correlates most strongly. A value of four was chosen to aid in visual interpretation of the results. However, other values can also be used.

In order to visualise the clustering of similar diseases, we colour the diseases by their disease class, obtained from the MeSH databases and saved within the package.

```
# load the disease classes
data(disease.classes)
col.other <- "white"

# set colours for each of the classes
classes <- sort(unique(disease.classes))
classes.cols <- structure(rep(col.other, length(classes)), names=cla
classes.cols["Cardiovascular Diseases"] <- "red"
classes.cols["Digestive System Diseases"] <- "orange"
classes.cols["Immune System Diseases"] <- "yellow"
classes.cols["Mental Disorders"] <- "lightgreen"
classes.cols["Musculoskeletal Diseases"] <- "darkgreen"
classes.cols["Nutritional and Metabolic Diseases"] <- "blue"
classes.cols["Skin and Connective Tissue Diseases"] <- "purple"
classes.cols["Urogenital Diseases and Pregnancy Complications"] <- '

# identify the class color for each disease
disease.cols <- classes.cols[disease.classes]
names(disease.cols) <- names(disease.classes)

# create cell type diseasome
project.network(pvalues.gsc, col.vert=disease.cols, col.other=col.ot
```




```
project.network.legend(classes.cols, col.other)
```



Extracting a cell type-specific disease gene subgraph

In this section, we will extract a subgraph of psoriasis-associated genes from a monocyte-specific interactome. This will include the psoriasis-associated genes and their interacting partners. We first construct the monocyte-specific interactome.

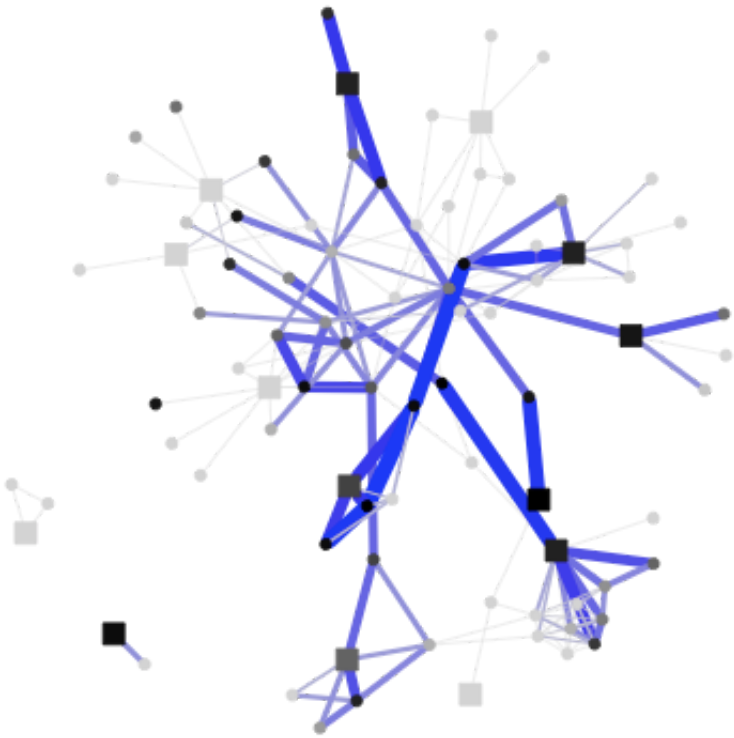
```
# create global network
data(edgelist.string)
g <- graph.edgelist(as.matrix(edgelist.string[, c("idA", "idB")]), c

# load expression data
data(expression.fantom5)
expression <- expression.transform(expression.fantom5)

# create monocyte-specific interactome
g.monocyte <- score.edges(g, expression[, "monocyte"])
```

Next, we identify the psoriasis-associated genes. To aid in visual interpretation of the subgraph, only those psoriasis-associated genes with 15 or fewer interacting partners will be included.

```
# identify all psoriasis-associated genes
data(disease.genes)
psoriasis.genes <- disease.genes[["psoriasis"]]

# remove genes not present in network and for which we have no expr
psoriasis.genes <- psoriasis.genes[psoriasis.genes %in% V(g.monocyte
psoriasis.genes <- psoriasis.genes[psoriasis.genes %in% rownames(exp

# remove genes with more than 15 interacting partners
psoriasis.genes <- psoriasis.genes[degree(g.monocyte)[psoriasis.gene
```

We can now use the **disease.subgraph** function to produce a monocyte-specific psoriasis subgraph.

```
# produce psoriasis subgraph
# in the paper, 500 bins, rather than 20 are used
g.subgraph <- disease.subgraph(g.monocyte, psoriasis.genes, n.bins=2
plot(g.subgraph)
```



In this plot, edges are weighted and coloured by their score, so that edges that are more likely to occur in monocytes are thick and dark blue. Vertices are coloured by the expression score of their respective gene, so that genes that are highly overexpressed in monocytes are coloured black. Square vertices represent disease genes and circle vertices represent their interacting partners.

In this plot, many of the disease-associated genes are strongly connected by monocyte-upweighted interactions, suggesting a pathway through which they may effect the development of the disease in monocytes.

To determine whether this subgraph is enriched with high-weight edges, we can permute the expression data, recreate the subgraph and compare the number of high-weight edges in the observed subgraph to the permuted subgraphs. In the accompanying paper, we used 10,000 permutations. However, to save time, we will use only 20 here.

```
# setup
edge.attr <- "score"
n.perm <- 20
edge.cutoff <- 0.9 # edges with weight higher than 0.9 are within t
proportion.perm <- rep(NA, n.perm)

# create observed subgraph
g.subgraph.obs <- disease.subgraph(g.monocyte, psoriasis.genes, edge
proportion.obs <- mean(get.edge.attribute(g.subgraph.obs, edge.attr)

for (n in 1:n.perm) {
    # create permuted subgraph
    expression.cell <- structure(sample(expression[, "monocyte"]), r
    g.monocyte.perm <- score.edges(g, expression.cell)
    g.subgraph.perm <- disease.subgraph(g.monocyte.perm, psoriasis.g

    # compute proportion of edges with weights created than edge.cut
    proportion.perm[n] <- mean(get.edge.attribute(g.subgraph.perm, e
}

# compute empirical p-value
# we set the lower limit of the p-value as 1/n.perm
max(mean(proportion.perm >= proportion.obs), 1/n.perm)
```

```
## [1] 0.05
```

Session information

This document was produced using:

```
sessionInfo()
```

```
## R version 3.1.1 (2014-07-10)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## locale:
## [1] C/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   b
##
## other attached packages:
## [1] gplots_2.14.2          DiseaseCellTypes_0.9.0 igraph_0.7.1
## [4] formatR_1.0            knitr_1.7
##
## loaded via a namespace (and not attached):
##  [1] KernSmooth_2.23-13 Matrix_1.1-4       bitops_1.0-6
##  [4] caTools_1.17.1     evaluate_0.5.5     gdata_2.13.3
##  [7] grid_3.1.1         gtools_3.4.1       lattice_0.20-29
## [10] psych_1.4.8.11     snow_0.3-13        stringr_0.6.2
## [13] tools_3.1.1
```

References

Bauer-Mehren A, Rautschka M, Sanz F, Furlong LI (2010) DisGeNET: a Cytoscape plugin to visualize, integrate, search and analyze gene-disease networks. Bioinformatics 26(22):2924-2926.

Cornish AJ, Markowetz F (2014) SANTA: Quantifying the Functional Content of Molecular Networks. PLoS Comput Biol 10(9):e1003808.

Csardi G, Napusz T (2006) The igraph software package for complex network research. Inter Journal Complex Systems p. 1695.

Forrest ARR et al. (2014) A promoter-level mammalian expression atlas. Nature 507(7493):462-470.

Franceschini A et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. Nucleic Acids Res 41:D808-D815.

Glaab E, Baudot A, Krasnogor N, Valencia A (2010) Extending pathways and processes using molecular interaction networks to analyse cancer genome data. BMC Bioinformatics 11:597.

Hamosh A, Scott AF, Amberger JS, Bocchini Ca, McKusick Va (2005) Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. Nucleic Acids Res 33:514-7.

Hu X et al. (2011) Integrating autoimmune risk loci with gene-expression data identifies specific pathogenic immune cell subsets. Am J Hum Genet 89(4):496-506.

Kohler S, Bauer S, Horn D, Robinson P (2008) Walking the interactome for prioritization of candidate disease genes. Am J Hum Genet 82:949-958.

Magger O, Waldman YY, Ruppin E, Sharan R (2012) Enhancing the prioritization of disease-causing genes through tissue specific protein interaction networks. PLoS Comput Biol 8(9):e1002690.

Sardar AJ et al. (2014) The evolution of human cells in terms of protein innovation. Mol Biol Evol 31(6):1364-1374.