13

Gene Set Enrichment Analysis

R. Gentleman, M. Morgan, and W. Huber

Abstract

Gene Set Enrichment Analysis (GSEA) is an important method for analyzing gene expression data. It is useful for finding biological themes in gene sets, and it can help to increase the statistical power of analyses by aggregating the signal across groups of related genes. In this chapter, we introduce tools available in the **Category** and **GSEABase** packages for carrying out gene set enrichment analysis.

13.1 Introduction

In this case study, we see how to use gene set enrichment analysis (GSEA); Subramanian et al. (2005); Tian et al. (2005); Jiang and Gentleman (2007). We primarily concentrate on the two-sample problem, where the data can be divided into two distinct groups, and we want to understand the set of differentially expressed genes between the two groups. We use a subset of the **ALL** dataset as described in Chapter 1.

The basic idea behind GSEA is to use predefined sets of genes, usually derived from functional annotation or from results of prior experiments, in order to better interpret the experiment that we are analyzing. The idea is similar to the one of Hypergeometric testing of $2x2$ contingency tables (see Chapter 14 for more details on that approach). Perhaps the most important difference is that in GSEA there is no need to make a hard cutoff between genes that are differentially expressed and those that are not. Rather, we determine a continuous-valued score, for example, the t-statistic, and see whether its values are associated with the gene sets of interest.

Any collection of gene sets can be used, and in many cases users will avail themselves of predefined gene sets, such as those available from GOA or KEGG. In this chapter we concentrate on KEGG, chromosome bands, and protein domains, because many other examples have been presented of applying these procedures to GO annotations.

F. Hahne et al., *Bioconductor Case Studies*, DOI: 10.1007/978-0-387-77240-0_13,

In release 2.1 of Bioconductor the **GSEABase** package was introduced. It contains software infrastructure for performing GSEA analyses. The basic concepts are simple: the *GeneSet* class represents a set of genes that are all described by a common set of identifiers such as EntrezGene IDs. A *GeneSetCollection* object is a collection of gene sets. **GSEABase** provides tools for performing set operations on gene sets, such as union and intersection, as well as ways to construct specific gene set collections, such as those for GO and KEGG, and using different types of inputs, such as a Bioconductor annotation package or an *ExpressionSet*.

There are a number of other Bioconductor packages that provide GSEA-type analyses. They include **PGSEA**, **sigPathways**, and **GlobalAncova**. For the Hypergeometric testing approach, the packages **GOstats** and **topGO** provide specialized methods that try to remove redundancy by taking advantage of the nested structure of the Gene Ontologies (The Gene Ontology Consortium, 2000).

13.1.1 Simple GSEA

Prior to conducting a GSEA analysis, we recommend making a basic data quality assessment followed by filtering genes to remove those that do not show much variation across samples (and hence have little ability to discriminate any samples). It is essential that sample annotation, such as phenotypic characteristics, not be used in this filtering step.

Next we compute a test statistic for each remaining gene, possibly using some form of moderation or regularization with an empirical Bayes method. It does not really matter how this step is carried out, but it is important that you make a choice that you are comfortable with, and that the resulting quantity have a similar distribution, and interpretation, for all genes being used. In particular, one should avoid statistics that have a different baseline for each gene, for example, statistics that are proportional to the observed intensity. In the following examples, which consider a two-group comparison, we use the t-statistic.

Now, the basic idea is that under the null hypothesis of no difference in mean expression between the two groups, the per-gene t-statistics t_k have a t-distribution. If we further assume that the genes, and hence the observed t-statistics, are independent, their sum, taken over the genes in a gene set, divided by the square root of the number of genes, should have an approximately Normal distribution with mean zero and variance one, by the central limit theorem. So that with

$$z_K = \frac{1}{\sqrt{|K|}} \sum_{k \in K} t_k, \tag{13.1}$$

where K denotes the gene set, and $|K|$ the number of genes in the gene set, the z_k have approximately a standard Normal distribution. And so,

one can potentially identify interesting gene sets by comparing their z_K to the quantiles of a standard Normal distribution. The assumption that the the genes are independent is quite strong, but in practice the test seems to lead to reasonable results.

An alternative approach is to make use of a permutation test to assess which gene sets have an unusually large absolute value of z_K. The usual null hypothesis is that the sample labels are not related to the observed values of gene expression, and hence by permuting labels we can generate a reference distribution for any test statistic of interest. When there are relatively few observations one will typically compute all possible permutations, but even for modest sample sizes it is not practical to compute all permutations and instead we sample some large number (typically 1000) of permutations. The observed value of the z_K-statistic is then compared to the reference distribution to obtain a p-value.

13.1.2 Visualization

The use of graphical tools can help to better understand how well the data are being modeled and they provide diagnostic checks on various assumptions you might have made.

A Q–Q plot of the observed test statistics (or p-values) versus an appropriate reference distribution will help to visually identify how extreme some of the per gene set statistics are. Once specific gene sets are identified as being of interest, heatmaps of the gene set data can be very informative. The functions `KEGG2heatmap` and `GO2heatmap` can easily be extended to other situations.

For two-sample comparisons we use plots that show the mean expression value in each group. Two specialized functions are the `GOmnplot` and `KEGmnplot`. It would also be useful to produce side-by-side boxplots on a per gene basis in situations where there are sufficient samples.

13.1.3 Data representation

The **GSEABase** package provides basic software tools for dealing with gene sets. A gene set is a set of genes that someone (possibly you) has determined are of interest. These gene sets can be grouped together into collections, and most analyses will be performed on collections. **GSEABase** provides tools for performing the usual set operations such as unions and intersection on gene sets.

A gene set in the *GeneSet* class is represented, essentially, as a *character* vector of identifiers together with information denoting the identifier system to which the identifiers refer. An alternative representation, which may be especially useful in the context of gene set collections, is as an incidence matrix, where, say, the rows correspond to the different gene sets, and there is one column for each gene. The entries in the matrix $A[i, j]$ are

either 1 or 0, depending on whether gene j is in gene set i. The incidence matrix is very useful for many of the computations we want to carry out and is used in most of the examples. The `incidence` function from the **GSEABase** package can be used to compute the incidence matrix from a gene set collection. Simple operations such as computing column or row sums will tell you how many gene sets a gene is in, or how many genes are in a particular gene set.

When an analysis identifies multiple gene sets as significant, we have found it valuable to characterize the amount of overlap between them. This can easily be computed from the incidence matrix. Gene sets that overlap substantially may be complementary, or one may be able to determine that the effect seems to be due to only one of the gene sets. Some examples and more explicit discussion were given by Jiang and Gentleman (2007).

13.2 Data analysis

We begin by describing the preprocessing steps that are used on the data, and follow that with applications using KEGG pathways and chromosome location. We note that the use of chromosome location is sensible for these data, as they are derived from cancer cells, and it is well known that some of the genomic alterations that are related to cancer tend to cluster by genomic location, perhaps due to amplification or deletion of DNA, or due to methylation or demethylation events.

13.2.1 Preprocessing

For this chapter, we use the `ALL` data, which have been obtained in a microarray study of B- and T-cell leukemia. We want to find genes that are differentially expressed between two distinct types of B-cell leukemia.

```
> library("ALL")
> data("ALL")
```

The data and the following steps with which we construct the subset of interest, `ALL_bcrneg`, are described in more detail in Chapter 1. Briefly, we select samples with B-cell leukemia harboring the BCR/ABL translocation and those samples with no observed cytogenetic abnormalities (NEG).

```
> bcell = grep("^B", as.character(ALL$BT))
> moltyp = which(as.character(ALL$mol.biol)
      %in% c("NEG", "BCR/ABL"))
> ALL_bcrneg = ALL[, intersect(bcell, moltyp)]
> ALL_bcrneg$mol.biol = factor(ALL_bcrneg$mol.biol)
```

The last line in the code above is used to drop unused levels of the *factor* variable `mol.biol`. Nonspecific filtering removes the probe sets that we believe are not sufficiently informative, so that there is little point in considering them further. We use the function `nsFilter` from the **genefilter** package to apply a number of different criteria. For instance, by default the function removes the control probes on Affymetrix arrays, which can be identified by their `AFFX` prefix. It also excludes probe sets without EntrezGene ID, and for instances of multiple probe sets mapping to the same EntrezGene ID, only the probe set with the largest variability is retained. The choice of the `var.cutoff` parameter indicates that we select the top 50% of probe sets on the basis of variability.

```
> library("genefilter")
> ALLfilt_bcrneg = nsFilter(ALL_bcrneg, var.cutoff=0.5)$eset
```

Exercise 13.1

a *How many samples are in our subset? How many are BCR/ABL and how many NEG?*

b *How many probe sets have been selected for our analysis?*

13.2.2 Using KEGG

KEGG (Kanehisa and Goto, 2000) provides mappings of genes to pathways and this information is included in most Bioconductor annotation packages. Many investigators are interested in whether there is some indication that certain pathways are implicated in their analysis. One way to make that assessment is to perform a gene set analysis on the KEGG pathways.

We use code from the **GSEABase** package to compute the incidence matrix that maps between probes and the pathways. The function `GeneSetCollection` can produce gene sets for a number of different inputs, including an instance of the *ExpressionSet* class, which is what we use.

```
> library("GSEABase")
> gsc = GeneSetCollection(ALLfilt_bcrneg,
                    setType=KEGGCollection())
> Am = incidence(gsc)
> dim(Am)
[1]  194 1666
```

We next compute a reduced *ExpressionSet* object `nsF` that retains only those features (genes) that are mentioned in the incidence matrix `Am` and whose features are in the same order as the columns of `Am`.

```
> nsF = ALLfilt_bcrneg[colnames(Am),]
```

Exercise 13.2
How many gene sets and how many genes are represented by the incidence matrix? How many gene sets have fewer than ten genes in them? What is the largest number of gene sets in which a gene can be found?

Next we compute the per gene test statistics using the `rowttests` function from the **genefilter** package. There are several other functions in **genefilter** for performing fast rowwise statistics (e.g., `rowFtests`).

```
> rtt = rowttests(nsF, "mol.biol")
> rttStat = rtt$statistic
```

Exercise 13.3
How many test statistics are positive? How many are negative? How many have a p-value less than 0.01*?*

In the next code segment, we reduce the incidence matrix by removing all gene sets that have fewer than ten genes in them.

```
> selectedRows = (rowSums(Am)>10)
> Am2 = Am[selectedRows, ]
```

Finding general trends requires that you use gene sets with a reasonable number of genes, and here we have operationalized that by setting our cutoff at ten. This cutoff is arbitrary, and in any analysis you should think about whether to do this, and if so, what value might be used on the basis of what you are interested in finding.

Now it is fairly easy to compute the per gene set test statistics and to produce a Normal Q–Q plot; see Figure 13.1.

```
> tA = as.vector(Am2 %*% rttStat)
> tAadj = tA/sqrt(rowSums(Am2))
> names(tA) = names(tAadj) = rownames(Am2)
```

```
> qqnorm(tAadj)
```

We see that there is one pathway that has a remarkably low observed value (less than -5) so we take a closer look at this pathway.

```
> library("KEGG.db")
> smPW = tAadj[tAadj < (-5)]
> pwName = KEGGPATHID2NAME[[names(smPW)]]
> pwName
[1] "Ribosome"
```

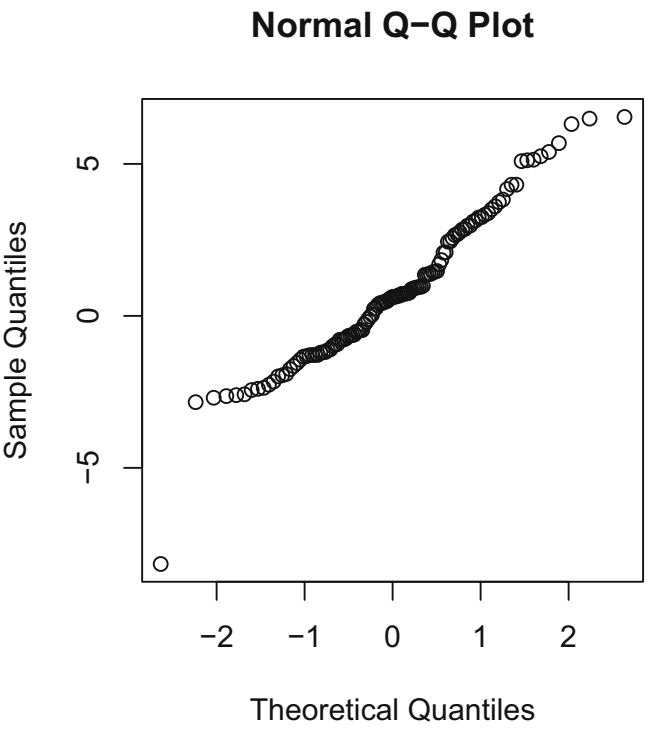

Figure 13.1. The per gene set Q–Q plot.

Now we can produce some summary plots based on the genes annotated at this pathway. The mean plot presents a comparison of the average expression value for each of our two groups, for each gene in the specified pathway.

```
> KEGGmnplot(names(smPW), nsF, "hgu95av2", nsF$"mol.biol",
    pch=16, col="darkblue")
```

That is, each point in the left panel of Figure 13.2 represents one gene and the value on the x-axis is the mean in the BCR/ABL samples whereas the value on the y-axis is the mean value in the NEG samples.

Exercise 13.4
Many of the points in Figure 13.2 appear to lie above the diagonal. Is this to be expected?"

And finally we can produce a heatmap for the genes in the ribosome pathway. We use the `KEGG2heatmap` function to do most of the hard work, and the result is shown in the right panel of Figure 13.2.

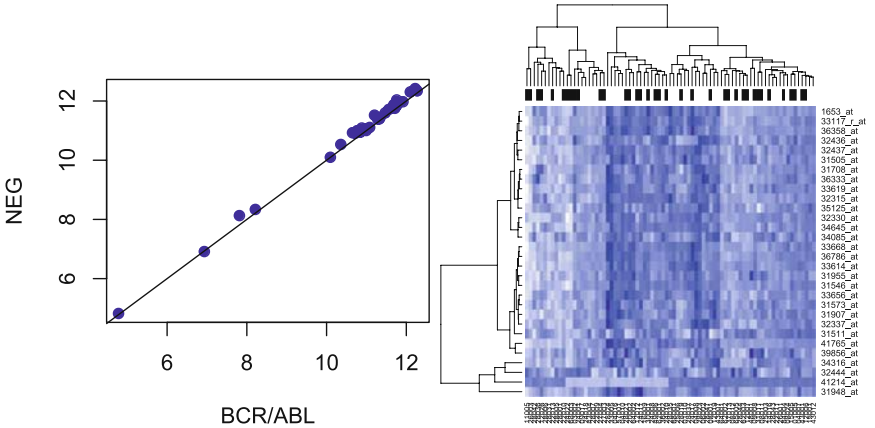

Figure 13.2. The left panel shows the scatterplot of within-group means for the genes in the ribosome pathway. The right panel shows a heatmap for these genes. The black and white bars at the top indicate the two disease types, BCR/ABL and NEG.

```
> sel = as.integer(nsF$mol.biol)
> KEGG2heatmap(names(smPW), nsF, "hgu95av2",
    col=colorRampPalette(c("white", "darkblue"))(256),
    ColSideColors=c("black", "white")[sel])
```

Exercise 13.5
What sorts of things do you notice in the heatmap? The gene labeled `41214_at` *has a very distinct pattern of expression. Can you guess what is happening? Hint: look at which chromosome it is on.*

As a further exercise, produce corresponding plots for the pathway with the largest positive average t-statistic.

13.2.3 Permutation testing

The assumptions on which we based the test above are somewhat strong, and it is of some interest to consider alternative approaches. For GSEA it is straightforward to compute a permutation test. The `gseattperm` function in the **Category** package can be used to compute the permutation test. It takes as inputs the gene expression data, phenotypic data for the samples, and the incidence matrix representing the gene sets of interest. The value returned by `gseattperm` is a matrix with columns `Lower` and `Upper`. For each row (gene set) the `Lower` column gives the proportion of permutation t-statistics that were smaller than the observed t; the `Upper` column gives the proportion of the permutation t-statistics that were larger than the observed t-statistic.

In the code chunk below we compute the permutation distribution based on 1000 permutations. We use a p-value cutoff of 0.025 that corresponds to a two-sided hypothesis test at the 0.05 level. Because we are carrying out a permutation test and want the results to be reproducible we begin by setting the seed for the random number generator. For users that are going to use permutation distributions, we recommend familiarizing themselves with the different pseudo-random number generators in R and the role of the seed, and using these tools with care.

```
> set.seed(123)
> NPERM = 1000
> pvals = gseattperm(nsF, nsF$mol.biol, Am2, NPERM)
> pvalCut = 0.025
> lowC  = names(which(pvals[, 1]<=pvalCut))
> highC = names(which(pvals[, 2]<=pvalCut))
```

In the next code chunk we print some of the resulting pathway names. There is one in `lowC` and 21 in `highC`.

```
> getPathNames(lowC)
```

```
[1] "03010: Ribosome"
```

```
> getPathNames(highC)
```

```
[1] "04360: Axon guidance"
[2] "05130: Pathogenic Escherichia coli infection - EHEC"
[3] "05131: Pathogenic Escherichia coli infection - EPEC"
[4] "04520: Adherens junction"
[5] "04510: Focal adhesion"
```

Exercise 13.6
What permutation-base p-value is the most extreme? What does the heatmap look like for this gene set?

Exercise 13.7
Compare the p-values from the parametric analysis to those from the permutation analysis.

13.2.4 Chromosome bands

Another interesting application is to use the mapping of genes to chromosome bands as the gene sets. In doing this, you are studying whether there

are anomalies in the pattern of gene expression that relate to chromosomal location. In many cancers there are underlying genomic changes, such as amplifications or deletions that cause coordinated changes in gene expression as a function of chromosomal location. In other settings, a separate justification for using chromosomal location would be needed. The chromosome band information can be obtained from the metadata variable with the suffix `MAP`, so for us it is `hgu95av2MAP`.

Exercise 13.8
What does the manual page say is the interpretation of the MAP position `17p33.2`*?*

Exercise 13.9
Just as we did for the KEGG analysis, we need to remove probe sets that have no chromosome band annotation. Follow the approach used in the KEGG analysis and create a new ExpressionSet object `nsF2`*. We will use it later in this chapter.*

Relevant MAP positions can be computed using the `MAPAmat` function from the **Category** package.

```
> EGtable = toTable(hgu95av2ENTREZID[featureNames(nsF2)])
> entrezUniv = unique(EGtable$gene_id)
> chrMat = MAPAmat("hgu95av2", univ=entrezUniv)
> rSchr = rowSums(chrMat)
```

Exercise 13.10
How many genes were selected? How many map positions?

Exercise 13.11
Further reduce `chrMat` *so that only bands with at least five genes are retained. Produce a Q–Q plot and identify the interesting bands. Use Google or some other search engine to determine what might be interesting about these bands.*
Do an analysis similar to the one we performed on the KEGG pathways. Produce mean plots, heatmaps, and so on. Try to identify a set of interesting bands.

Exercise 13.12
Reorder the columns of `chrMat` *so that they are in the same order as the corresponding features in the ExpressionSet object* `nsF2`*.*

13.3 Identifying and assessing the effects of overlapping gene sets

Once the potentially interesting gene sets have been identified, it is useful to consider how much overlap there is between them. In some cases the list can be reduced by identifying gene sets that are redundant, or that perhaps have been identified because they overlap with another gene set that really is important. It is possible to consider higher levels of overlap, say between three gene sets, but here we restrict our attention to two gene sets at a time.

To assess the amount of overlap we suggest using a simple statistic which is the number of genes in common, divided by the number of genes in the smaller of the two gene sets. This statistic is 0 when there are no genes in common and 1 if one of the two gene sets is a subset of the other. Computation can be performed using the methods in the **GSEABase** package, but is easy enough using generic algebra on the incidence matrix.

We consider the permutation analysis reported in Section 13.2.3 and consider all gene sets with permutation p-values less than 0.025 as interesting. To compute our index, we simply take the submatrix of the incidence matrix that corresponds to the interesting gene sets (as identified earlier) and then take the matrix product with its transpose. The diagonal elements of `Amx` are the sizes of each gene set, and the off-diagonal elements are the number of genes in common.

```
> Ams = Am2[union(lowC, highC),]
> Amx = Ams %*% t(Ams)
> minS = outer(diag(Amx), diag(Amx), pmin)
> overlapIndex = Amx/minS
```

The resulting matrix, `overlapIndex`, is plotted in Figure 13.3.

Exercise 13.13
How many genes are in each of the four pathways, `04512`, `04940`, `04510`, `04514`? How many are in the overlap for each pair?

In order to determine whether there is evidence that both of the gene sets in a pair are involved in the observed relationship with the t-statistic, or whether one of the two is perhaps only implicated due to the shared genes, we consider a number of different linear models.

First we fit a linear model to each of the two gene sets separately, and observe that the corresponding p-values are less than 0.05. But when both are included in the model, only one remains significant.

```
> P04512 = Ams["04512",]
> P04510 = Ams["04510",]
```

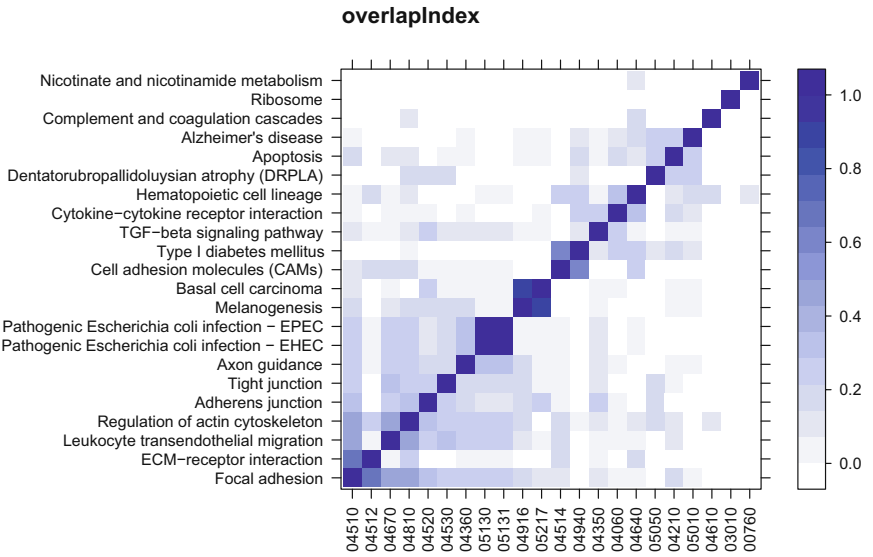

Figure 13.3. Overlap between the gene sets as measured by `overlapIndex`.

```
> lm1 = lm(rttStat ~ P04512)
> summary(lm1)$coefficients
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.0948     0.0378    2.51   0.0123
P04512        0.5783     0.2774    2.08   0.0372
> lm2 = lm(rttStat ~ P04510)
> summary(lm2)$coefficients
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.0715     0.0385    1.86 0.063565
P04510        0.5913     0.1604    3.69 0.000235
> lm3 = lm(rttStat ~ P04510+P04512)
> summary(lm3)$coefficients
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    0.070     0.0386   1.815  0.06971
P04510         0.541     0.1724   3.139  0.00173
P04512         0.237     0.2973   0.796  0.42627
```

We next divide the genes into three groups: those that are in `04512` only, those that are in both sets, and those that are in `04510` only. Using these variables in our linear regression shows that only the second and third variables are significant. This suggests that the pathway `04512` only appears to be interesting because of the genes it shares with the `04510` pathway.

```
> P04512.Only = ifelse(P04512 != 0 & P04510 == 0, 1, 0)
> P04510.Only = ifelse(P04512 == 0 & P04510 != 0, 1, 0)
> Both        = ifelse(P04512 != 0 & P04510 != 0, 1, 0)
> lm4 = lm(rttStat ~ P04510.Only + P04512.Only + Both)
> summary(lm4)
Call:
lm(formula = rttStat ~ P04510.Only + P04512.Only + Both)

Residuals:
   Min     1Q Median     3Q    Max
-4.167 -1.034 -0.183  0.884  7.211

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.0689     0.0386    1.78   0.0749 .
P04510.Only   0.5650     0.1804    3.13   0.0018 **
P04512.Only   0.4088     0.4842    0.84   0.3985
Both          0.6972     0.3353    2.08   0.0377 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
 ' ' 1

Residual standard error: 1.53 on 1662 degrees of freedom
Multiple R-Squared: 0.0086,     Adjusted R-squared: 0.00
681
F-statistic: 4.81 on 3 and 1662 DF,  p-value: 0.00245
```

We also recommend checking the sign of the t-statistic, to make sure that effects are concordant (as they are here).

Exercise 13.14
Repeat this analysis for the other pair of pathways, 04940 and 04514.