ShotgunFunctionalizeR - an R-package for functional comparison of metagenomes

User's Guide

Erik Kristiansson, Philip Hugenholtz and Daniel Dalevi

October 18, 2010

Contents

1 Introduction

This is the User's Guide to ShotgunFunctionalizeR, an R-package for functional comparison of metagenomes. The aim of this document is to provide an easy but comprehensive introduction to ShotgunFunctionalizeR and show how it can be used to analyze metagenomic data. ShotgunFunctionalizeR is applicable to data generated by shotgun high-throughput sequencing annotated as any of the supported gene-families which currently are Cluster of Orthologous Genes (COGs) (Tatusov et al., 1997; Tatusov et al., 2003) and Enzyme Commission (EC) numbers (Bairoch, 2000), PFAM and TIGRFAM protein domains.

Metagenomic data can in ShotgunFunctionalizeR be examined using two different types of approaches: *gene-centric* and *pathway-centric analysis*. The gene-centric analysis is focused on individual gene families, while pathway-centric analysis deals with pathways and other groups of functionally related gene-families. ShotgunFunctionalizeR contains several for these types of functional analysis.

Several statistical procedures have been implemented into ShotgunFunctionalizeR. Procedures for gene-centric analysis contains traditional exact tests like binomial, hypergeometric and the corresponding Gaussian approximation. In addition, we have developed and implemented a Poisson linear model, which offers a higher flexibility than previous approaches. With this model, experimental design such as direct comparisons using multiple samples, regression models and factor designs becomes possible.

Pathway-centric analysis also offers a number of different alternatives. The Gaussian sum statistics, which is currently in use at the JGI MGI/M web portal (Markowitz et al., 2008) is implemented. In addition, other common procedures, such as Fisher's exact test and enrichment test using a contingency tables is also available. The novel Poisson model can also be used for pathway-centric analysis.

2 Contact information

ShotgunFunctionalizeR is maintained by Erik Kristiansson (erik.kristiansson@neuro.gu.se) and Daniel Dalevi (dalevi@chalmers.se). Any suggestions and ideas how to make the ShotgunFunctionalizeR package even more appealing are greatly appreciated.

As most software, ShotgunFunctionalizeR may contain bugs. If you encounter a bug, we would appreciate if you could send us a bug report, describing both the itself bug, how it can be reproduced and what version of R and Bioconductor you are using. The ShotgunFunctionalizeR web page is available at http://shotgun.zool.gu.se and contain the latest new and updates. Before you submit any bug report, make sure that you are using the latest version of the ShotgunFunctionalizeR package.

3 Citing ShotgunFunctionlizeR

If you ShotgunFunctionalizeR in your scientific work, please cite the following article

Kristiansson, E. and Dalevi, D. (2009). ShotgunFunctionlizeR: An R package for functional comparison of metagenomes. *Bioinformatics*. 25(20).

4 Prerequisites and installation

ShotgunFunctionalizeR is dependent on the `multtest` package (Gilbert et al., 2009), which is a part of the Bioconductor software project (Gentleman et al., 2004). This means that the Bioconductor needs to be installed on your computer before ShotgunFunctionalizeR can function properly. To install Bioconductor, start R and type:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

After the installation of Bioconductor have finished, you can install ShotgunFunctionalizeR by writing:

```
> install.packages("ShotgunFunctionalizeR",
  contriburl="http://shotgun.zool.gu.se")
```

5 Analysis of metagenomic data using ShotgunFunctionalizeR

5.1 Data sets used in this tutorial

ShotgunFunctionalizeR package contains several example data sets. Two of these will be used in this tutorial and are therefore described below. Information on how you import your own data is available in Section 5.8.

5.1.1 The Ocean's interior data

The Ocean's interior data set comes from a study performed by Delong and coauthors a few years ago (DeLong et al., 2006). In this study, microbial metagenomes were samples at different depths (10, 70, 130, 200, 500, 770 and 4000 meters) at the Hawaii Ocean Time-series (HOT) station in the Pacific Ocean. The ocean's interior data is available with annotations according to the gene families COG, EC numbers, PFAM domains and TIGRfam domains (the data objects named `OceanCOG`, `OceanEC`, `OceanPFAM` and `OceanTIGR` respectively).

5.1.2 The Mouse obesity data

The Mouse obesity data was generated by Turnbaugh et al. (2006). In this study, the gut microbiota from three lean and two obese mouse were compared. This dataset is also available both annotated using both COG and EC (`MouseCOG` and `MouseEC` respectively). This data was downloaded from the IMG/M web service (Markowitz et al., 2008).

5.2 Data visualization and basic statistics

We will start with the basics. To load the ShotgunFunctionalizeR, type

```
> library(ShotgunFunctionalizeR)
```

To list the available functions and variables, you can use the function `ls`

```
> ls("package:ShotgunFunctionalizeR")
```

We will now take a look at the Ocean's interior data set. To load this data set annotated as COGs, type

```
> data(OceanCOG)
```

The loaded object `OceanCOG` is the standard data structure used by ShotgunFunctionalizeR and is simply list containing both the raw data in the form of a matrix

```
> OceanCOG$Data[1:10,]
   X10m X70m X130m X200m X500m X770m X4000m
1     1    0     1     0     0     0      3
2     0    2     0     1     1     0      4
3     4    0     2     4     8     9      6
4     0    0     1     1     1     1      0
5     1    1     1     1     1     0      0
6     0    0     0     0     1     0      0
7     0    0     0     0     1     0      0
8     0    0     0     0     0     0      1
9     4    1     0     3     0     3      3
10    0    0     1     0     0     0      0
```

and the corresponding annotation

```
> OceanCOG$Annotation[1:10,]
   GeneFamily
1     COG2940
2     COG3264
3     COG1894
4     COG0731
5     COG0239
6     COG3125
```

```
7     COG5429
8     COG3657
9     COG0122
10    COG2913
                                                                  Annotation
1                                             Proteins containing SET domain
2                                 Small-conductance mechanosensitive channel
3            NADH:ubiquinone oxidoreductase, NADH-binding (51 kD) subunit
4                                                      Fe-S oxidoreductases
5   Integral membrane protein possibly involved in chromosome condensation
6                     Heme/copper-type cytochrome/quinol oxidase, subunit 4
7                                           Uncharacterized secreted protein
8                              Uncharacterized protein conserved in bacteria
9            3-methyladenine DNA glycosylase/8-oxoguanine DNA glycosylase
10                                         Small protein A (tmRNA-binding)
```

To get a quick summary over the data, you can use the `sumarizeData` function. Simply type

```
> summarizeData(OceanCOG)

Summary

Gene family type: COG
Number of samples: 7
Samples: X10m,X70m,X130m,X200m,X500m,X770m,X4000m

Sample               1      2      3      4      5      6      7
Total counts      5667   6716   5126   6164   7172   8543   8691
Mean counts        1.7    2.0    1.5    1.8    2.1    2.5    2.5
Standard dev.      3.6    4.2    7.5    6.5    4.9    4.9    5.2
Zeros             1612   1430   1777   1652   1446   1345   1297
Zeros (%)         28.4   21.3   34.7   26.8   20.2   15.7   14.9

Quantiles      Min    25%    50%    75%    90%    Max
Sample  1:     0.0    0.0    1.0    2.0    4.0   97.0
Sample  2:     0.0    0.0    1.0    2.0    5.0   78.0
Sample  3:     0.0    0.0    0.0    2.0    3.0  380.0
Sample  4:     0.0    0.0    1.0    2.0    4.0  294.0
Sample  5:     0.0    0.0    1.0    2.0    5.0  155.0
Sample  6:     0.0    0.0    1.0    3.0    7.0   88.0
Sample  7:     0.0    0.0    1.0    3.0    6.9   78.0
```

We can see that there are in total 5667 COGs in sample 1 and 6716 in sample 2 and so on. The output also shows the mean counts per COG, the standard deviation and the number of COGs not having a count in the sample. Note however that only COGs that have at least one count in at least one of the samples

are included in this dataset. The quantiles table shows information about how the counts are distributed among the COGs. In sample 1, for example, 90% of the COGs have less than or equal to 4 counts and no COG has more than 97 counts.

A subset of samples can be extracted from a data object using the `getSamples` command.

```
> OceanCOG.subsample<-getSamples(OceanCOG, samples=c(1,2,6,7))
> summarizeData(OceanCOG.subsample)
Summary

Gene family type: COG
Number of samples: 4
Samples: X10m,X70m,X770m,X4000m

Sample               1      2      3      4
Total counts      5667   6716   8543   8691
Mean counts        1.7    2.0    2.5    2.5
Standard dev.      3.6    4.2    4.9    5.2
Zeros             1612   1430   1345   1297
Zeros (%)         47.1   41.8   39.3   37.9

Quantiles      Min    25%    50%    75%    90%    Max
Sample  1:       0      0      1      2      4     97
Sample  2:       0      0      1      2      5     78
Sample  3:       0      0      1      3      7     88
Sample  4:       0      0      1      3      7     78
```

The `getSamples` can also be used to change the order of the samples in a data object.

```
> OceanCOG.2<-getSamples(OceanCOG, samples=c(7,6,5,4,3,2,1))
```

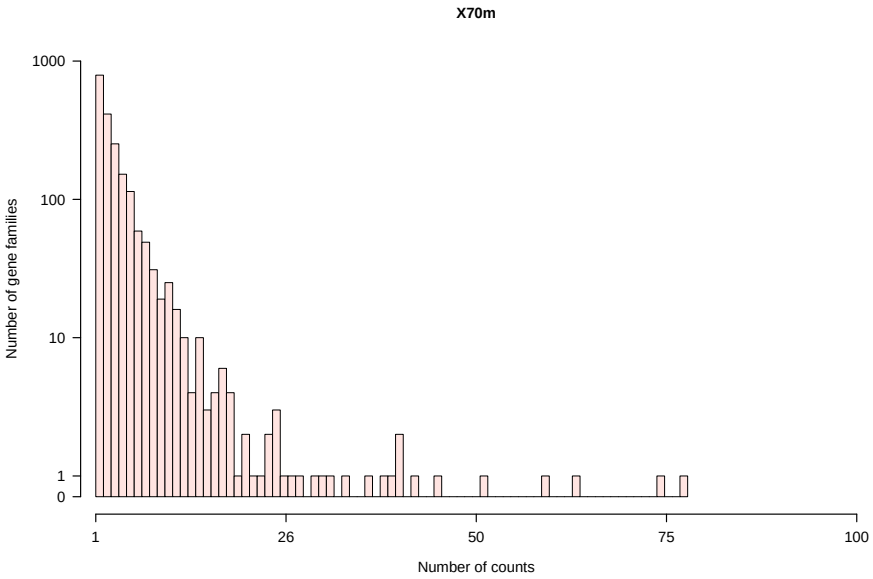
Two data object with a two disjunct set of samples can be merged using `mergeSamples`.

```
> OceanCOG.subsample.1<-getSamples(OceanCOG, samples=c(1,2,6,7))
> OceanCOG.subsample.2<-getSamples(OceanCOG, samples=c(3,4,5))
> OceanCOG.merged<-mergeSamples(OceanCOG.subsample.1, OceanCOG.subsample.2)
> summarizeData(OceanCOG.merged)
Summary

Number of samples: 7
Samples: X10m,X70m,X770m,X4000m,X130m,X200m,X500m

Sample               1      2      3      4      5      6      7
Total counts      5667   6716   8543   8691   5126   6164   7172
Mean counts        1.7    2.0    2.5    2.5    1.5    1.8    2.1
```

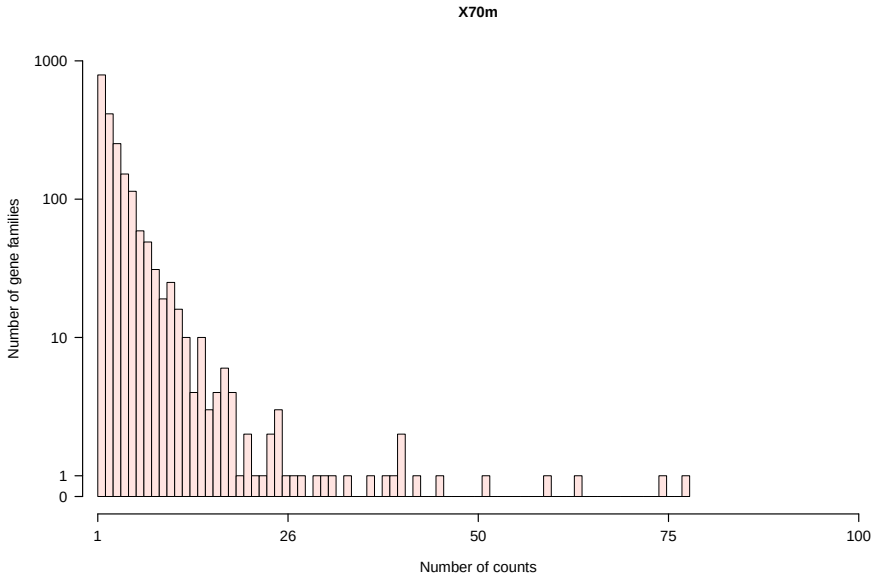
```
Standard dev.      3.6    4.2    4.9    5.2    7.5    6.5    4.9
Zeros             1612   1430   1345   1297   1777   1652   1446
Zeros (%)         47.1   41.8   39.3   37.9   51.9   48.3   42.3

Quantiles       Min    25%    50%    75%    90%    Max
Sample  1:        0      0      1      2      4     97
Sample  2:        0      0      1      2      5     78
Sample  3:        0      0      1      3      7     88
Sample  4:        0      0      1      3      7     78
Sample  5:        0      0      0      2      3    380
Sample  6:        0      0      1      2      4    294
Sample  7:        0      0      1      2      5    155
```

The function `densityPlot` can be used to take a closer look at the data distribution.

```
> densityPlot(OceanCOG, sample=1)
```

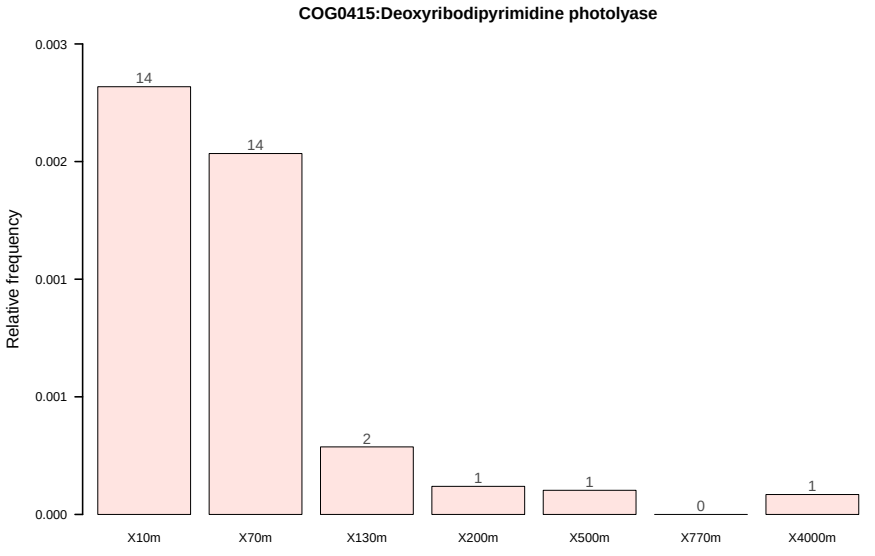



```
> densityPlot(OceanCOG, sample=2)
```



It is also possbile to visualize the relative frequences for a single gene family using the function `freqPlot.family`.

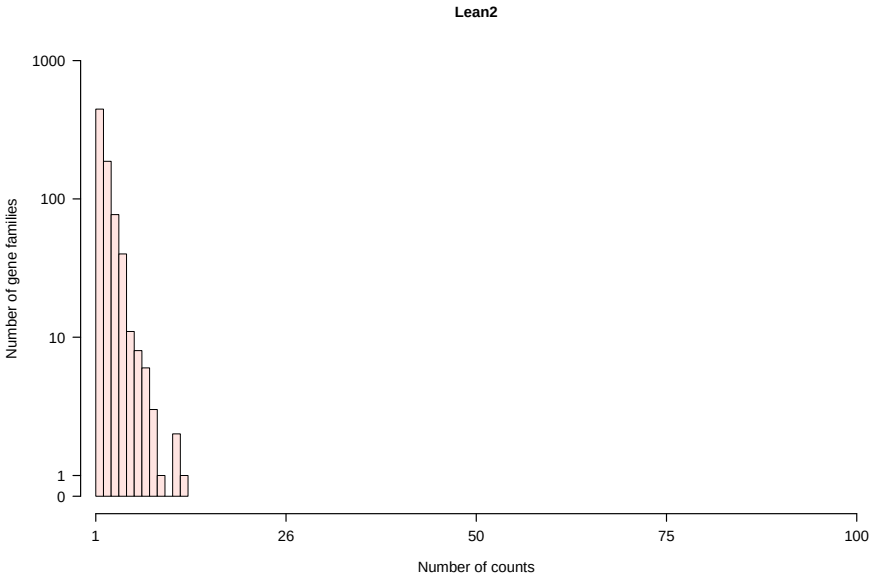
```
>freqPlot.family(OceanCOG, family="COG0415")
```



The numbers above the bars shows the total number of counts and the relative frequences are calculated based on the total observations in each sample.

A similar analyis of the Mouse obesity data shows

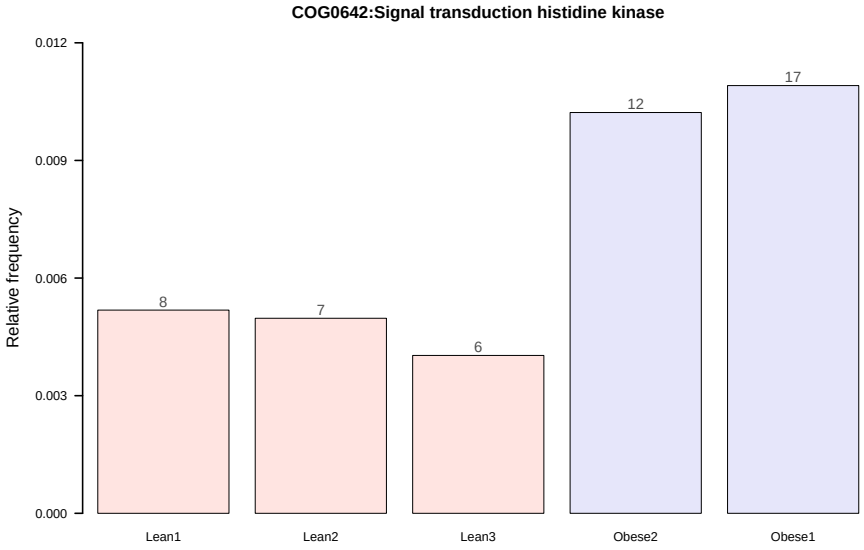
```
> data(MouseCOG)
> summarizeData(MouseCOG)
Summary

Gene family type: COG
Number of samples: 5
Samples: Lean1,Lean2,Lean3,Obese2,Obese1

Sample               1      2      3      4      5
Total counts      1562   1424   1508   1188   1577
Mean counts        1.0    0.9    0.9    0.7    1.0
Standard dev.      2.9    1.3    3.9    2.5    3.5
Zeros              781    815    810    928    822
Zeros (%)         48.9   51.0   50.7   58.1   51.5

Quantiles      Min    25%    50%    75%    90%    Max
Sample  1:       0      0      1      1      2    104
Sample  2:       0      0      0      1      2     13
Sample  3:       0      0      0      1      2    139
Sample  4:       0      0      0      1      2     81
Sample  5:       0      0      0      1      2    117

> densityPlot(MouseCOG, sample=1)
```



The `freqPlot.family` can also used to color groups of samples differently.

```
>freqPlot.family(MouseCOG, family="COG0642", groups=c(1,1,1,2,2),
```

```
groups.color=c("red", "blue"))
```



5.3 Managing samples

ShotgunFunctionalizeR contains functions for swift manipulation of data. The `getSamples` can extract or remove samples from a data object.

```
> OceanCOGShallow<-getSamples(OceanCOG,c("X10m", "X70m", "X130m"))
> summarizeData(OceanCOGShallow)
Summary

Gene family type: COG
Number of samples: 3
Samples: X10m,X70m,X130m

Sample              1      2      3
Total counts     5667   6716   5126
Mean counts       1.7    2.0    1.5
Standard dev.     3.6    4.2    7.5
Zeros            1612   1430   1777
Zeros (%)        47.1   41.8   51.9

Quantiles      Min    25%    50%    75%    90%    Max
Sample  1:       0      0      1      2      4     97
Sample  2:       0      0      1      2      5     78
Sample  3:       0      0      0      2      3    380
```

```
> OceanCOGDeep<-getSamples(OceanCOG,c("X10m", "X70m", "X130m"), exclude=T)
> summarizeData(OceanCOGDeep)
Summary

Gene family type: COG
Number of samples: 4
Samples: X200m,X500m,X770m,X4000m

Sample               1      2      3      4
Total counts      6164   7172   8543   8691
Mean counts        1.8    2.1    2.5    2.5
Standard dev.      6.5    4.9    4.9    5.2
Zeros             1652   1446   1345   1297
Zeros (%)         48.3   42.3   39.3   37.9

Quantiles      Min    25%    50%    75%    90%    Max
Sample  1:       0      0      1      2      4    294
Sample  2:       0      0      1      2      5    155
Sample  3:       0      0      1      3      7     88
Sample  4:       0      0      1      3      7     78
```

5.4 Gene-centric analysis

5.4.1 Direct comparison of two groups with single samples

ShotgunFunctionalizeR contains several tests for direct comparisons of two metagenomic samples: the binomial, the hypergeometric and the Gaussian test. These tests can be applied by using the `testGeneFamilies` function. Note however that these tests only work on one pair of samples, so you need to specify which to compare. For example, performing a binomial test between the samples located at 10 and 4000 meter in Ocean's interior dataset, type:

```
> stat.bt<-testGeneFamilies(OceanCOG, method="Binomial",
  samples=c(1,7))
```

The resulting data structure (`stat.bt`) is a list and to view the 5 entries with lowest p-values (most significant gene families) we look at the first 5 rows in the `Ranking` data.frame:

```
> stat.bt$Ranking[1:5,]
     GeneFamily Metagenome 1 Metagenome 2 Total Count      P-value
1584    COG0810           97           78         175 2.842310e-05
3064    COG0415           14            1          15 4.218974e-05
33      COG2141           17           69          86 1.536201e-04
3062    COG3728            0           18          18 2.380096e-04
2936    COG5178           58           42         100 2.740724e-04
```

```
     Adjusted P-value (BH)
1584            0.07218664
3064            0.07218664
33              0.17522928
3062            0.18757516
2936            0.18757516
                                                        Annotation
1584  Periplasmic protein TonB, links inner and outer membranes
3064                            Deoxyribodipyrimidine photolyase
33                         ... tetrahydromethanopterin reductase
3062                              Phage terminase, small subunit
2936                                U5 snRNP spliceosome subunit
```

This list shows the name of the gene family, the p-value and corresponding p-value adjusted for multiple testing and the annotation. These numbers all always returned in ShotgunFunctionalizeR. In addition, the total count as well as the count in each of the samples is shown. These number are specific for the binomial tests and other procedures may return other types of data. You can specify other depths by changing the `samples` parameter, e.g. $c(1, 2)$ will be 10 versus 70 meters.

ShotgunFunctionalizeR uses the `multtest` R-package (Gilbert et al., 2009) to correct the P-values for multiple testing. This implies that all implemented in `multtest` is available in ShotgunFunctionalizeR. The default method is the Benjamini-Hochberg False Discovery Rate and the adjusted P-values are in this case the q-values described in Benjamini and Hochberg (1995). This can, however, be changed with the `multtest` argument. Please see the manual page for the function `mt.rawp2adjp` for information about the available algorithms.

There are also other alternatives for direct comparison of two metagenomes implemented in in ShotgunFunctionalizeR. These can be used by changing the `method` parameter to either "Hypergeometric" or "Gaussian". Please refer to (Kristiansson et al., 2009b) and the manual page of `testGeneFamilies` for further information.

5.4.2 Direct comparisons of two groups with multiple samples

Direct comparison between two groups, each containing multiple samples, can be performed by the Poisson model. To demonstrate this feature, we will used the Mouse obesity dataset. This dataset contains five samples, three from lean mice and two from obese mice. To compare the lean with the obese group, we can use the `testGeneFamilies.dircomp` function

```
> po.stat<-testGeneFamilies.dircomp(MouseCOG,
           groups=c("Lean", "Lean", "Lean", "Obese", "Obese"))
```

The most significant gene families can be accessed as before

```
> po.stat$Ranking[1:5,]
     GeneFamily Coefficient       AIC      P-value Adjusted P-value (BH)
```

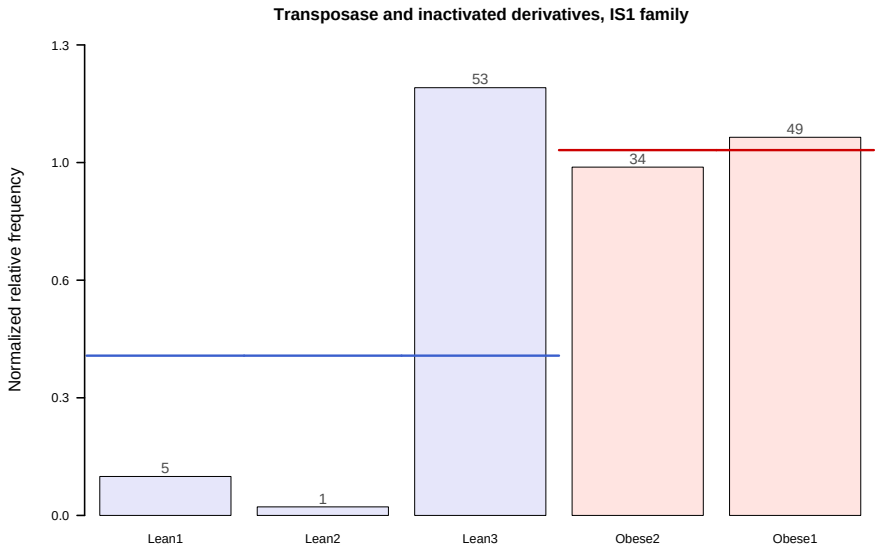
```
1497    COG1662  -0.8270057 110.94030 1.194384e-06           0.005820232
4103    COG4845  -0.2727114 205.99440 4.320546e-03           0.999999317
636     COG0642  -0.8084759  24.69728 4.778929e-03           0.999999317
831     COG0840  -1.3330004  27.22481 6.298196e-03           0.999999317
975     COG1131  -0.9452349  25.95773 1.036351e-02           0.999999317
                                                      Annotation
1497   Transposase and inactivated derivatives, IS1 family
4103                    Chloramphenicol O-acetyltransferase
636                   Signal transduction histidine kinase
831                    Methyl-accepting chemotaxis protein
975  ABC-type multidrug transport system, ATPase component
```

As before, the ranking list contains annotations and both the original and adjusted p-value. In addition, a coefficient from the Poisson model is available. This coefficient is the estimated difference between the two groups. In this case, the five most significant gene families are all lower abundant in the samples from lean than obese mice.

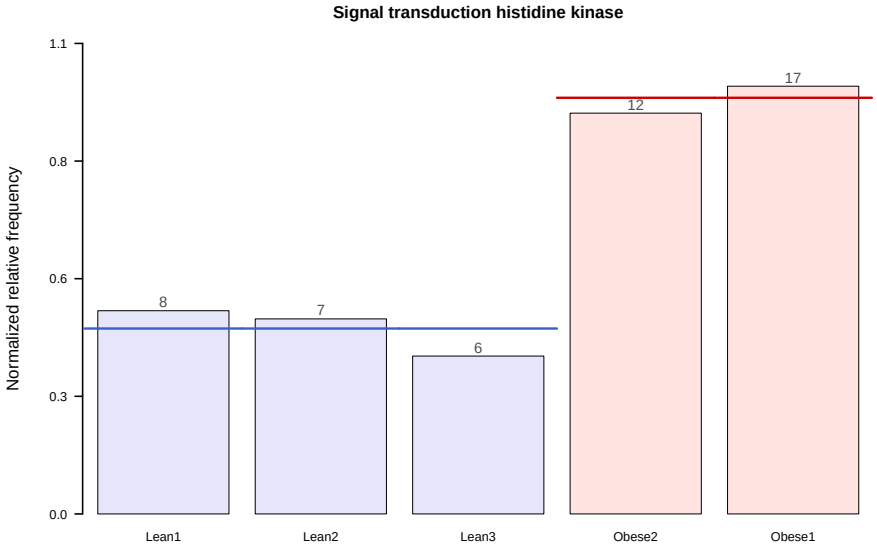
The column named AIC contains the Akaike information criterion (AIC) which is a measure of the fit of the model. A lower value means a better correspondence between the data and the Poisson model.

The `trendPlot` can be used to visualize the difference between the groups

```
> trendPlot(po.stat, family="COG1662")
```




```
> trendPlot(po.stat, family="COG0642")
```



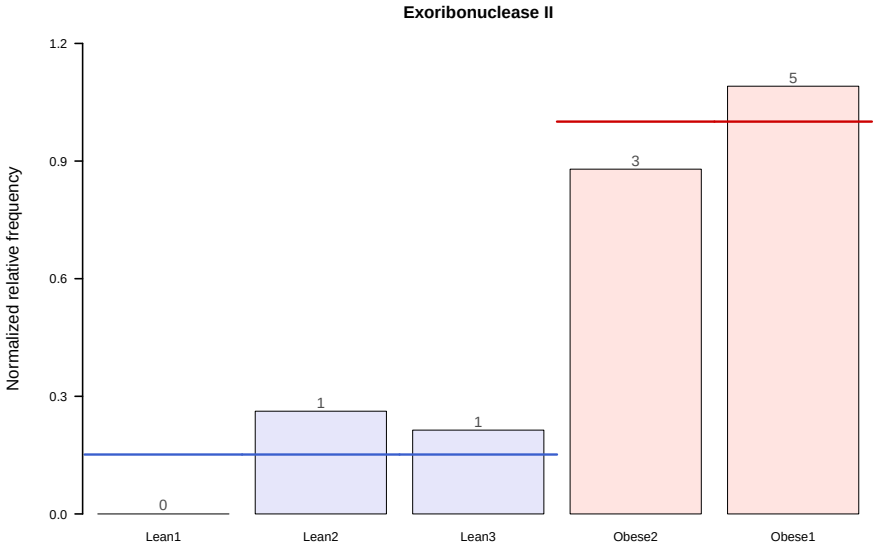
We can clearly see why the AIC is higher for `COG1662` than `COG0642`, but the difference is more significant for the former due to the high number of counts.

A similar analysis is easy to do for the Mouse obesity data annotated as EC numbers.

```
> data(MouseEC)
> po.dircomp<-testGeneFamilies.dircomp(MouseEC,
             groups=c("Lean", "Lean", "Lean", "Obese", "Obese"))
> po.dircomp$Ranking[1:5,]
    GeneFamily Coefficient       AIC     P-value Adjusted P-value (BH)
131 EC2.3.1.28  -0.2731415 158.86894 0.004038305             0.9996222
258  EC2.7.8.6  -1.8875920  16.34440 0.016957113             0.9996222
396 EC3.6.3.28  -0.8496043  23.84310 0.024235458             0.9996222
278 EC3.1.21.3  -2.2930571  14.58038 0.033757342             0.9996222
393 EC3.6.3.24  -0.9532827  20.47751 0.048649018             0.9996222
                        Annotation
131 Sinapate 1-glucosyltransferase
258             Exoribonuclease II
396         Methylisocitrate lyase
278            Phosphodiesterase I
393          Anthranilate synthase
```

As before, the result for individual gene families can be shown using the `trendPlot` function

```
> trendPlot(po.dircomp, family="EC2.7.8.6")
```



5.4.3 Regression analysis

The Ocean's interior dataset is sampled at a number of different depths. In such situations, the Poisson model can be used to identify gene families that change depending on the depth. Regression analysis can be performed with the `testGeneFamilies.regression` function

```
> stat.po<-testGeneFamilies.regression(OceanCOG,
           covariates=c(10,70,130,200,500,770,4000),
   log.covariate=T)
```

The depths are supplied using the `covariate` argument which can be log-transformed by adding `log.covariate=T`. The `testGeneFamilies.regression` function will perform a regression analysis for each COG and as before, return a list with the corresponding p-values. The calculations may take a few minutes depending on what kind of computer you have. Since the Ocean's Interior dataset is very sparse, the Poisson model may have problems with convergence for some of the gene families and hence generate a few warnings. These gene families will still be included in the ranking list but will, in most situation, result in a very high p-value.

To view the 5 entries with lowest p-values (most significant gene families):

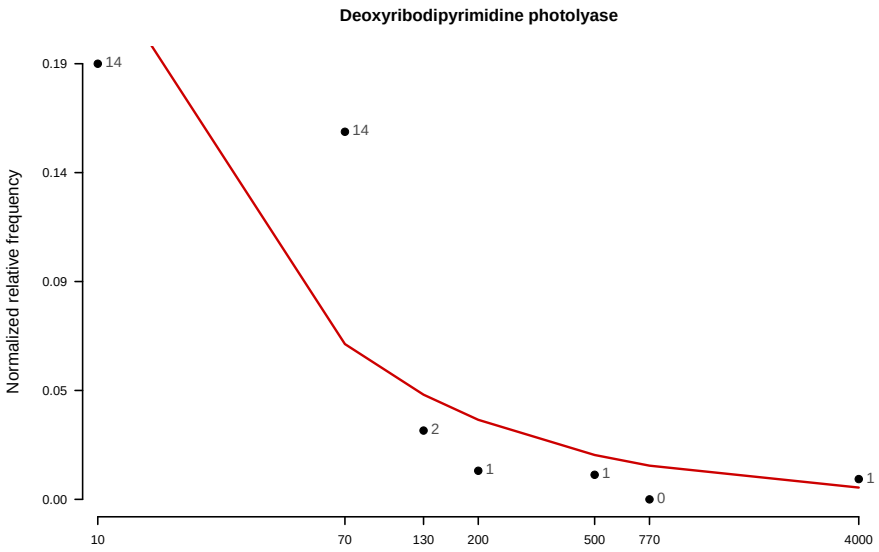
```
> stat.po$Ranking[1:5,]
     GeneFamily Coefficient       AIC      P-value Adjusted P-value (BH)
1584    COG0810  -0.3656694 955.50059 4.467316e-20          1.528716e-16
3064    COG0415  -1.4673033  36.06359 4.664631e-09          7.981184e-06
33      COG2141   0.4569334  61.28127 2.821974e-07          3.218932e-04
```

```
815     COG2801   1.1389283  47.58137 5.175972e-07          3.596107e-04
2936    COG5178  -0.3605901 112.42801 5.254394e-07          3.596107e-04

Annotation
1584             Periplasmic protein TonB, links inner and outer membranes
3064             Deoxyribodipyrimidine photolyase
33               Coenzyme F420-dependent N5,N10-methylene ...
815              Transposase and inactivated derivatives
2936             U5 snRNP spliceosome subunit
```

This list shows the name of the gene family, the regression-coefficient (slope), the Akaike information criterion (AIC) for the fitted Poisson model, the p-value, the corresponding p-value adjusted for multiple testing and the annotation. The sign of the coefficient tells you whether the trend is positive or negative with respect to the covariate.

The fitted regression model can be visualized in a plot together with the data points. For example, if we wish to look at the photolyase and how its abundance depends on the depth:

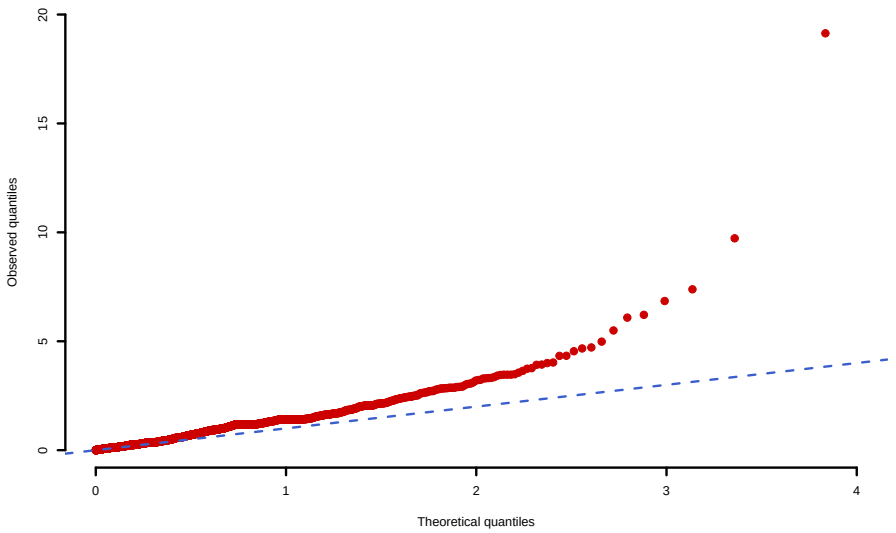
```
> trendPlot(stat.po, family="COG0415")
```



The `trendPlot` shows the data from the different depths as well as the fitted regression line. To be able to compare the different counts between the different samples, the normalized rates are plotted and the number of counts are added at each dot. Note for example, that the 14 counts at a depths of 10 meter get a higher rate than those at 70 meters since the total number of counts was higher at 70 (6716) than 10 (5667).

ShotgunFunctionalizeR provides means for general model validation. The function `ppPlot` plots the resulting p-values against the theoretical p-values under the null hypothesis that there is no effect (in this case, no change depending on depth). Under the null-hypothesis, the p-values should be uniformly distributed between 0 and 1, which should be true for many of the gene families. To create a `ppPlot`, type

```
> ppPlot(stat.po)
```

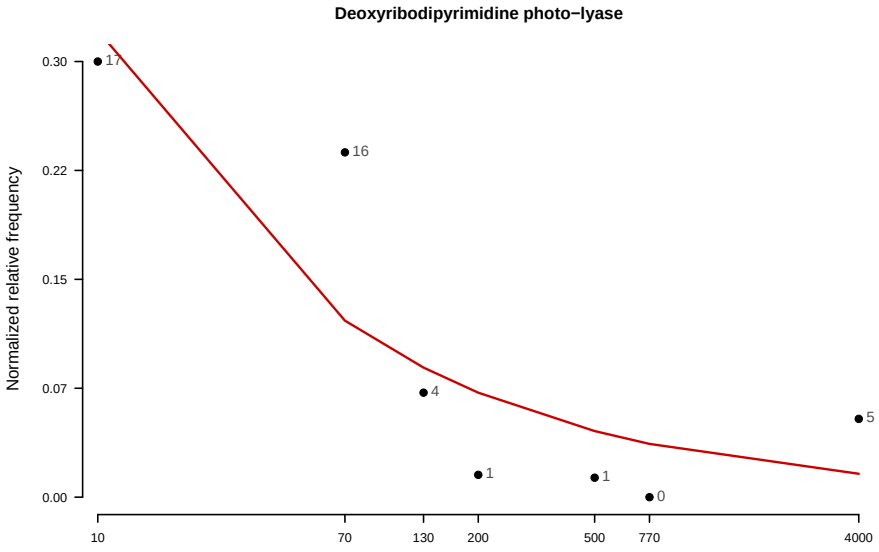


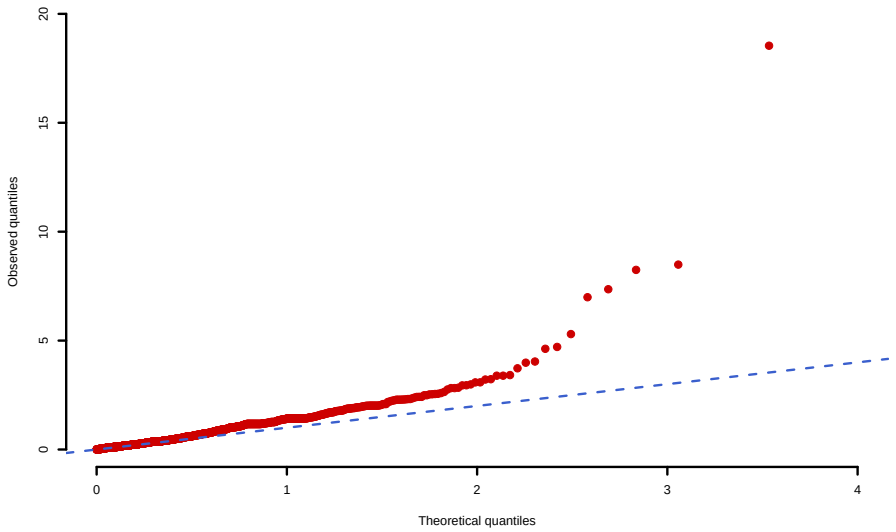
The `ppPlot` shows that many of the gene families lie close to the diagonal line (dashed blue), which indicates a satisfactory model fit under the null-hypothesis.

The Ocean's interior dataset is also provided in a EC version, which can be loaded and analyzed with

```
> data(OceanEC)
> po.stat<-testGeneFamilies.regression(OceanEC,
          covariates=c(10,70,130,200,500,770,4000),
   log.covariate=T)
>
> po.stat$Ranking[1:5,]
     GeneFamily Coefficient       AIC      P-value Adjusted P-value (BH)
26  EC3.4.21.72  -0.4106573 524.03989 1.884177e-19          3.225710e-16
411  EC4.1.99.3  -1.1502569  47.97085 1.223340e-08          1.047179e-05
605  EC2.7.7.49   1.0194988  36.94930 6.479597e-08          3.697690e-05
475 EC3.4.21.10  -0.1191125 627.00743 9.252169e-08          3.959928e-05
581   EC4.3.2.5   0.9737575  43.40352 3.325821e-07          1.138761e-04
                          Annotation
26  IgA-specific serine endopeptidase
```

```
411 Deoxyribodipyrimidine photo-lyase
605       RNA-directed DNA polymerase
475                          Acrosin
581      Peptidylamidoglycolate lyase
>
> trendPlot.family(po.stat, family="EC4.1.99.3")
> ppPlot(stat.po)
```





5.4.4 More complicated situations - a 2×2 factor design

ShotgunFunctionalizeR can also analyze data in more complicated situations using the more general `testGeneFamilies` function for the Poisson model. This function requires a design matrix, which is a powerful way to describe experimental setups. The design matrix has one row for each sample and one column for each coefficient hence linking the samples to the experimental setup. More information regarding linear models and design matrices can be found in Arnold (1980) and Radhakrishna and Toutenburg (1999).

In this example we will be working with a 2 by 2 factor design and we show how such an experimental setup can be analyzed using ShotgunFunctionalizeR. For simplicity, we will use simulated data from four assumed groups, A, B, C and D, each with two repliactes. The aim is to compare group A and B versus C and D (A and B can for example be lean and obese mice before an antibiotic treatment and C and D be lean and obese mice after antibiotic treatment).

We start by creating a ShotgunFunctionalizeR data object containing eight samples with ten gene families

```
> Data<-matrix(rpois(n=80, lambda=rep((1:8)*10, each=10)),
        nr=10, nc=8, byrow=F)
> colnames(Data)<-c("A1", "A2", "B1", "B2", "C1", "C2", "D1", "D2")
> Annotation<-data.frame(GeneFamily=paste("GF", 1:10),
              Annotation=paste("GF", 1:10))
```

We add an effect to the first four samples

```
> Data[3,1:2]<-Data[3,1:2]+10
```

```
> Data[3,3:4]<-Data[3,3:4]+40
```

This effect is higher for group B (even relative to the number of total counts). We can now create the data object used by ShotgunFunctionalizeR and take a look at the data

```
> SimData<-list(Data=data.frame(Data), Annotation=Annotation,
           Type="Custom")
> summarizeData(SimData)
Summary

Gene family type: Custom
Number of samples: 8
Samples: A1,A2,B1,B2,C1,C2,D1,D2

Sample               1      2      3      4      5      6      7      8
Total counts        99    217    338    492    514    625    698    823
Mean counts        9.9   21.7   33.8   49.2   51.4   62.5   69.8   82.3
Standard dev.      3.5    6.3   13.6   10.8    6.2   10.1    8.4    8.1
Zeros                0      0      0      0      0      0      0      0
Zeros (%)          0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0

Quantiles      Min    25%    50%    75%    90%    Max
Sample  1:       5      9      9     10     12     19
Sample  2:      14     18     20     22     27     37
Sample  3:      22     27     30     36     40     70
Sample  4:      38     44     47     50     54     78
Sample  5:      43     48     50     56     58     63
Sample  6:      49     57     62     66     69     86
Sample  7:      55     65     70     74     78     85
Sample  8:      68     76     85     87     89     95
```

To test group A and B versus group C and D we use the following design matrix

```
> Design<-cbind(rep(1,8), c(rep(1,6), rep(0,2)),
          c(rep(1,4), rep(0,4)), c(rep(1,2), rep(0,6)))
> Design
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    1    1    1    1
[3,]    1    1    1    0
[4,]    1    1    1    0
[5,]    1    1    0    0
[6,]    1    1    0    0
[7,]    1    0    0    0
[8,]    1    0    0    0
```

As mentioned abobe, the design matrix have the same number of row as samples and each column indicates a coefficient. In this case, the first column

corresponds to the intercept (baseline), the second column to the coefficient for group A+B+C, the third column to the coefficient for group A+B and the final column for the coefficient for group A. Since we want to test group A+B versus C+B we will test whether the coefficient for column three is significantly different from zero (we want it to be positive). Note that the design matrix is not unique and will look different depending on the type of test you want to perform (e.g. testing A+C versus B+D or A versus B+C+D will result in different design matrices).

To perform the test we used the general function `testGeneFamilies` and type

```
> po.stat<-testGeneFamilies(SimData, method="poisson",
  design.matrix=Design, coef=3)
> po.stat$Ranking
   GeneFamily Coefficient      AIC      P-value Adjusted P-value (BH)
3        GF 3  0.55578838 57.32251 4.823406e-06          4.823406e-05
1        GF 1 -0.24138646 52.95610 1.222456e-01          4.871056e-01
2        GF 2 -0.22588227 56.45127 1.461317e-01          4.871056e-01
7        GF 7 -0.13233710 56.64227 3.785373e-01          8.729182e-01
6        GF 6 -0.12190419 57.93180 4.525199e-01          8.729182e-01
9        GF 9 -0.07875286 53.37376 6.040613e-01          8.729182e-01
5        GF 5 -0.06072557 56.56852 6.822884e-01          8.729182e-01
4        GF 4 -0.04136533 51.98878 7.868985e-01          8.729182e-01
10      GF 10  0.03364803 56.67389 8.140259e-01          8.729182e-01
8        GF 8 -0.02508444 55.33650 8.729182e-01          8.729182e-01
```

As expected, the third gene family is highly significant. Note that we get no significant gene family if we for example test the coefficient corresponding to the fourth column of the design matrix (A versus B+C+D).

```
> po.stat<-testGeneFamilies(SimData, method="poisson",
  design.matrix=Design, coef=4)
> po.stat$Ranking
    GeneFamily Coefficient      AIC   P-value Adjusted P-value (BH)
2        GF 2  0.29042079 56.45127 0.1549711             0.4971404
8        GF 8  0.28199485 55.33650 0.1558989             0.4971404
1        GF 1  0.27925673 52.95610 0.1763180             0.4971404
3        GF 3 -0.18932543 57.32251 0.2289104             0.4971404
10      GF 10 -0.24813352 56.67389 0.2485702             0.4971404
7        GF 7 -0.19134768 56.64227 0.4029851             0.5917524
6        GF 6  0.16582342 57.93180 0.4461670             0.5917524
9        GF 9 -0.15360735 53.37376 0.4960745             0.5917524
5        GF 5 -0.13621561 56.56852 0.5325772             0.5917524
4        GF 4  0.01629169 51.98878 0.9388769             0.9388769
```

5.5 Pathway-centric analysis

ShotgunFunctionalizeR can also analyse sets of functionally related gene families, called gene categories. This is sometimes refereed to as pathway-centric analysis. ShotgunFunctionalizeR has currently support for gene categories based on both COG and EC number annotated data. For data annotated with COGs, the COG Pathways and COG Categories from JGI MGI/M (Markowitz et al., 2008) are available. For data annotated with EC numbers, KEGG pathways are available (Kanehisa et al., 2002). These are regularly updated against the corresponding databases so be sure to use the latest version of ShotgunFunctionalizeR.

5.5.1 Basic pathway-centric analysis

The gene category annotation data is available in the ShotgunFunctionalizeR system data object named `ShotgunFunctionalizeRAnnotationData`. To list the available pathways for COGs and EC numbers we can type

```
> names(ShotgunFunctionalizeRAnnotationData$PATHWAYS$COG)
[1] "PATHWAY2COG"  "COG2PATHWAY"  "CATEGORY2COG" "COG2CATEGORY"
> names(ShotgunFunctionalizeRAnnotationData$PATHWAYS$KEGG)
[1] "PATHWAY2EC" "EC2PATHWAY"
```

To list the available categories under COG Pathways, we can type

```
> names(ShotgunFunctionalizeRAnnotationData$PATHWAYS$COG$PATHWAY2COG)
 [1] "Archael/Vacuolar-type H+ ATPase subunits"
 [2] "F0F1-type ATP synthase subunits"
 [3] "Glyoxylate bypass"
 [4] "NA+-transporting NADH:Ubiquinone oxireductase"
 ...
[73] "Intracellular trafficking, secretion, and"
[74] "Defense mechanisms"
[75] "Extracellular structures"
[76] "Nuclear structure"
[77] "Cytoskeleton"
```

To check which gene families that associated to a certain pathway, we can simply take a look at the element with the corresponding number

```
> ShotgunFunctionalizeRAnnotationData$PATHWAYS$COG$PATHWAY2COG[1]
$`Archael/Vacuolar-type H+ ATPase subunits`
[1] "COG0636" "COG1155" "COG1156" "COG1269" "COG1390" "COG1394" "COG1436"
[8] "COG1527" "COG2811"
```

One procedure used to identify difference in abundant gene categories between two pair of metagenomes is the Gaussian Sum statistic (also known as d-score, Markowitz et al. (2008). This procedure calculates a Z-score (Gaussian approximation) for each gene family in the gene category, which are then summed

(further details are available in Kristiansson et al. (2009b)). To make a pathway-centric analysis of the the Ocean's interior data using the Gaussian Sum statistic (10 vs 4000 meters) type

```
> stat.gauss<-testGeneCategories(OceanCOG, category="cogpathways",
  samples=c(1,7))
> stat.gauss$Ranking[1:5,]
   Category size Gene families found Gaussian sum (Z-score)     P-value
62           245                 209               3.161505 0.001569563
15             5                   5              -3.107277 0.001888197
56            72                  43               2.700404 0.006925530
29             9                   8              -2.642839 0.008221412
77            12                   6               2.642425 0.008231460
   Adjusted P-value (BH)                                      Category
62            0.07269558 Translation, ribosomal structure and biogenesis
15            0.07269558                          Threonine biosynthesis
56            0.11151276   Cell cycle control, cell division, chromosome
29            0.11151276                        Coenzyme A biosynthesis
77            0.11151276                                   Cytoskeleton
```

The ranking list contains the Gaussian Z-score, p-value, adjusted p-value and the annotation for each gene category. In addition, the number of gene families in category and the number of these present in the data set is shown.

A pathway-centric analysis of the Ocean's interior using the KEGG gene categories can be done by typing

```
> stat.gauss<-testGeneCategories(OceanEC, category="kegg", samples=c(1,7))
> stat.gauss$Ranking[1:5,]
    Category size Gene families found Gaussian sum (Z-score)      P-value
122             7                   2               3.501710 0.0004622822
158            65                  42               3.227124 0.0012504125
37             41                  28               2.691977 0.0071029759
12             54                  19               2.550119 0.0107686048
129            22                  10              -2.451584 0.0142228942
    Adjusted P-value (BH)                       Category
122            0.07350287  Tetrachloroethene degradation
158            0.09940779          Pyrimidine metabolism
37             0.37645772         Aminosugars metabolism
12             0.42805204             Lysine degradation
129            0.45228804 Toluene and xylene degradation
```

ShotgunFunctionalizeR contains several procedures for pathway-centric analysis. One other example is the enrichment analysis, which can be performed on any ranked list of gene families. To use enrichment analysis, you have to specify the method to rank the gene families and the p-value cut-off to discriminate significant gene families from non-significant. The enrichment is then tested using Fisher's exact test (Agresti, 2002; Kristiansson et al., 2009b). To use en-

richment analysis on the Ocean's interior data ranked according to the binomial procedure and with a p-value cut-off we can type

```
> stat.gauss<-testGeneCategories(OceanCOG, category="cogpathways",
  samples=c(1,7), method="enrichment", enrichment.method="binomial",
  enrichment.p=0.05)
> stat.gauss$Ranking[1:5,]
   Category size Gene families found Sig. gene families in cat.
55           258                 218                          10
56            72                  43                           3
27             6                   5                           1
6              7                   7                           1
44            11                  10                           1
   Sig. gene families not in cat. Non-sig. gene families in cat
55                             61                           208
56                             68                            40
27                             70                             4
6                              70                             6
44                             70                             9
   Non-sig. gene families not in cat    P-value Adjusted P-value (BH)
55                              3143 0.01335793                     1
56                              3311 0.05851833                     1
27                              3347 0.09957986                     1
6                               3345 0.13661516                     1
44                              3342 0.18936940                     1
                                        Category
55              Energy production and conversion
56 Cell cycle control, cell division, chromosome
27                            Biotin biosynthesis
6                       Pyruvate decraboxylation
44                    Basal transription factors
```

The resulting ranking list contains, as before, p-values, annotations and information about the size of the categories. In addition, information about the enrichment test is also provided (significant gene families in the category, significant gene families not in the category, non-significant gene families in the category and non-significant gene families not in the category).

5.5.2 Pathway-centric analysis using the Poisson model

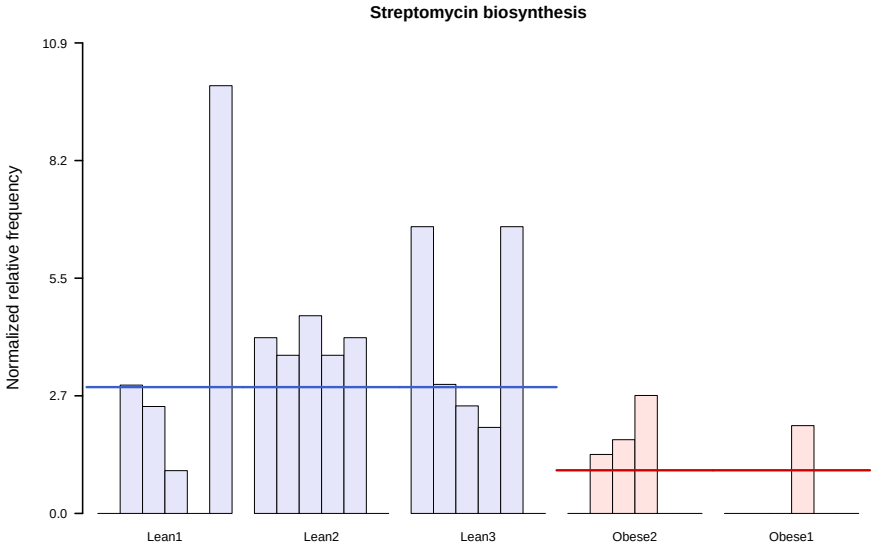
As for gene-centric analysis, the Poisson model can be used for pathway-centric analysis. As before, the Poisson model can be used in three different ways: direct comparison (`testGeneCategories.dircomp`), regression (`testGeneCategories.regression`) and with supplied design matrix (`testGeneCategories`).

We will start by performing an analysis using the Mouse obesity data annotated with EC numbers. Type

```
> po.cat.dircomp<- testGeneCategories.dircomp(MouseEC, category="KEGG",
                   groups=c("Lean", "Lean", "Lean", "Obese", "Obese"))
> po.cat.dircomp$Ranking[1:5,]
    Category size Gene families found Coefficient       AIC    P-value
86             47                   9  -1.0121232  94.30718 0.01637442
60             14                   6   1.0742387  78.73192 0.01661083
112            30                   2  -1.5999099  27.25366 0.05005643
158            65                  28   0.3598413 369.75908 0.05339128
56             22                  20  -0.3912967 261.86570 0.06238529
    Adjusted P-value (BH)                              Category
86                      1 Nicotinate and nicotinamide metabolism
60                      1              Streptomycin biosynthesis
112                     1           Phenylpropanoid biosynthesis
158                     1                Pyrimidine metabolism
56                      1            Aminoacyl-tRNA biosynthesis
```

The ranking list have similarities to the results from `testGeneFamilies.dircomp` but in this case the coefficient shows the difference for the entire category. You can still use `trendPlot` to look at the result but now you need to specify the category number, i.e.

```
> trendPlot(po.cat.dircomp, category=60)
```



In this case `trendPlot` shows one bar for each gene family available in the data (6 of 14 were found). A missing bar indicates a count of zero.

Similarly, for the Mouse obesity data annotated as COG,

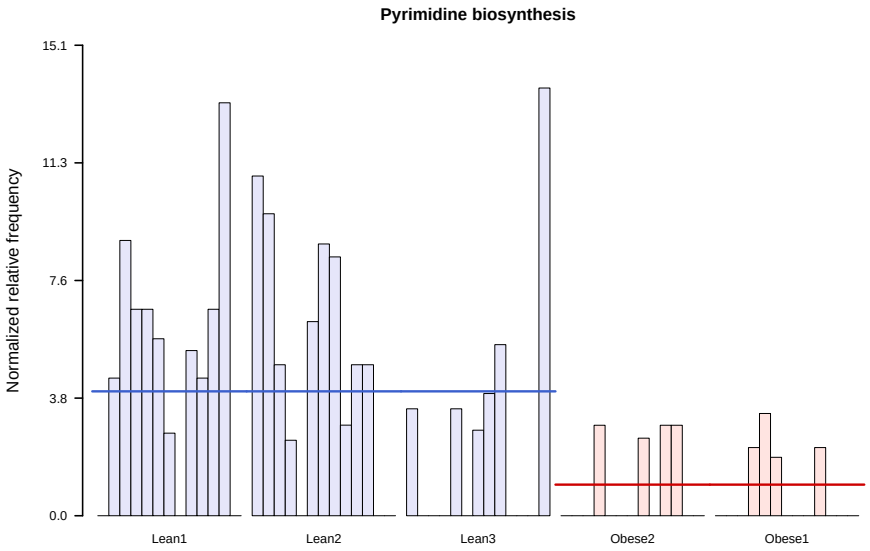
```
> po.cat.dircomp<- testGeneCategories.dircomp(MouseCOG, category="COGPATHWAYS",
                  groups=c("Lean", "Lean", "Lean", "Obese", "Obese"))
```

As before, the ranking list can be examined by

```
> po.cat.dircomp$Ranking[1:5,]
   Category size Gene families found Coefficient       AIC      P-value
64           238                 113  -0.2691967 1662.9297 7.132977e-05
72           152                  60  -0.3959325  763.1691 1.180633e-04
20            14                  13   1.3860996  171.9987 2.624334e-04
63           230                  72  -0.2955768  981.6928 8.221396e-04
58            95                  64   0.4194149  787.1437 2.673642e-03
   Adjusted P-value (BH)                             Category
64           0.004545438 Replication, recombination and repair
72           0.004545438        Signal transduction mechanisms
20           0.006735792               Pyrimidine biosynthesis
63           0.015826188                         Transcription
58           0.038065104   Nucleotide transport and metabolism
```

And a plot created by

```
> trendPlot(po.cat.dircomp, category=20)
```



Pathway analysis can also be performed for gene categories using the `testGeneCategories.regression` function. To perform regression analysis for the Ocean's interior dataset annotated using COGs, type

```
> po.cat.reg<-testGeneCategories.regression(OceanCOG,
              category="COGPATHWAYS",
```

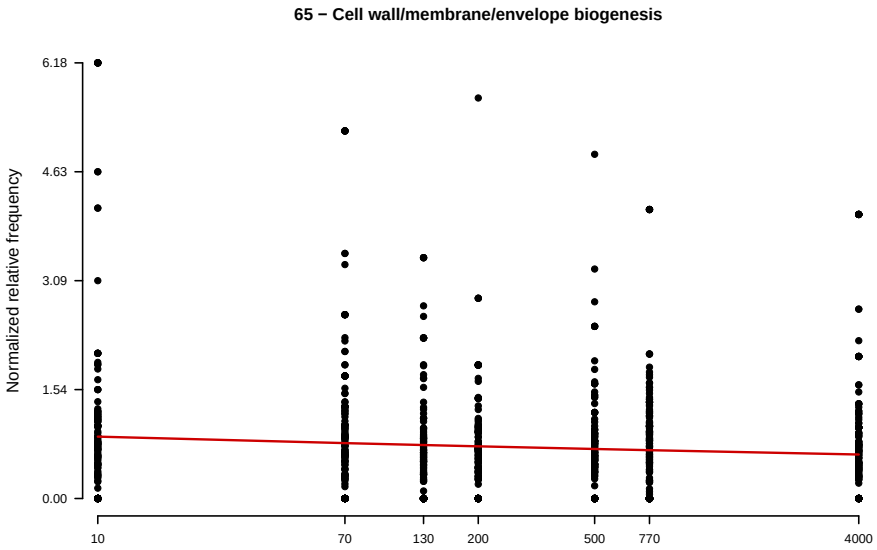
```
      covariates=c(10,70,130,200,500,770,4000),
      log.covariates=T)
```

We have thus performed a pathway-centric regression analysis using the COG Pathway gene category with the depths as covariates. We have also used a log-transform of the covariates before the curve is fitted (`log.covaraites=T`). The resulting ranking list reveals several highly significant gene categories

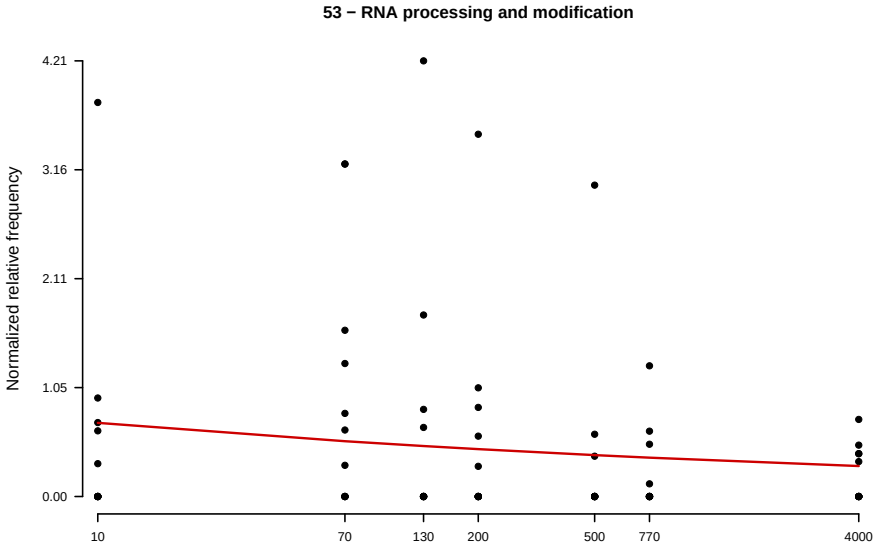
```
> po.cat.reg$Ranking[1:5,]
   Category size Gene families found Coefficient       AIC      P-value
65           188                 166 -0.13101955 4895.0967 3.564303e-11
53            25                  12 -0.33911985  265.9712 4.846582e-07
56            72                  43 -0.19096608 1042.7716 5.285793e-05
55           258                 218  0.08079222 4827.7422 6.302852e-05
57           270                 244  0.06381631 6040.7912 2.689587e-04
   Adjusted P-value (BH)                                  Category
65          2.744514e-09        Cell wall/membrane/envelope biogenesis
53          1.865934e-05              RNA processing and modification
56          1.213299e-03 Cell cycle control, cell division, chromosome
55          1.213299e-03             Energy production and conversion
57          4.141963e-03           Amino acid transport and metabolism
```

The trend can be plotted using the `trendPlot` function,

```
>trendPlot(po.cat.reg, category=65)
```

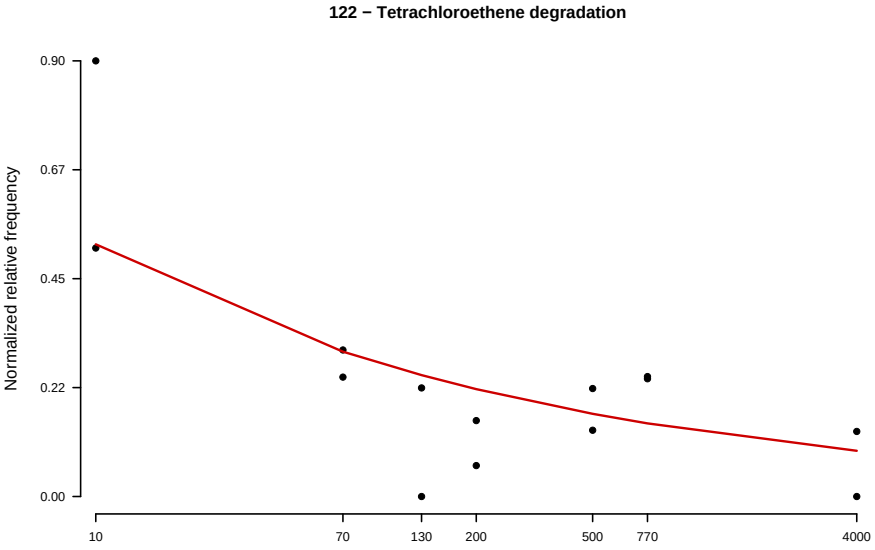



```
>trendPlot(po.cat.reg, category=53)
```



A similar analysis of the EC number annotated data results in

```
> po.cat.reg<-testGeneCategories.regression(OceanEC, category="KEGG",
              covariates=c(10,70,130,200,500,770,4000), log.covariates=T)
> po.cat.reg$Ranking[1:5,]
    Category size Gene families found Coefficient        AIC      P-value
12              54                  19  -0.1794115  600.86801 3.916582e-05
61              12                   3  -0.2606291  114.02120 7.897390e-04
157             39                  19   0.1640067  451.31840 9.163898e-04
122              7                   2  -0.6562776   51.08901 1.435028e-03
158             65                  42  -0.1089667 1054.37196 1.583618e-03
    Adjusted P-value (BH)                     Category
12             0.006227365           Lysine degradation
61             0.043110155          Geraniol degradation
157            0.043110155       Glycerolipid metabolism
122            0.043110155 Tetrachloroethene degradation
158            0.043110155         Pyrimidine metabolism
> trendPlot(po.cat.reg, category=122)
```



5.6 Functional diversity

ShotgunFunctionalizeR also contains tools for estimation and comparison of the functional diversity of metagenomic data. The function `estimateDiversity` calculates three common diversity indices for any number of specified samples.

```
> estimateDiversity(OceanTIGR, sample=c(1,7))
[[1]]
[[1]]$Chao
[1] 1845.568

[[1]]$Simpson
[1] 0.9985473

[[1]]$Shannon
[1] 6.804437


[[2]]
[[2]]$Chao
[1] 2061.151

[[2]]$Simpson
[1] 0.999155

[[2]]$Shannon
```

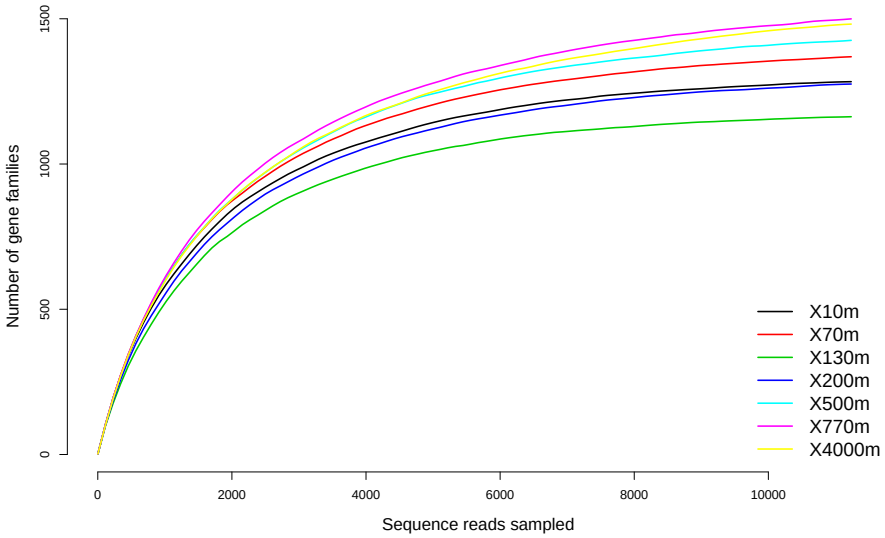
```
[1] 6.881281
```

Here, `Chao` corresponds to Chaos estimator for species (functional) richness, `Simpson` the Simpson index and `Shannon` the Shannon diversity from information theory. It is also possible to add these indices to the data summary generated by `summarizeData`,

```
> summarizeData(OceanTIGR, print.diversity=TRUE)
Summary

Gene family type: TIGRFAM
Number of samples: 7
Samples: X10m,X70m,X130m,X200m,X500m,X770m,X4000m

Sample               1      2      3      4      5      6      7
Total counts      3694   4170   3144   3954   4758   5674   5668
Mean counts        1.5    1.7    1.3    1.6    2.0    2.3    2.3
Standard dev.      2.7    3.2    4.1    4.2    3.6    3.9    4.2
Zeros             1127   1030   1262   1132    952    858    877
Zeros (%)         46.2   42.2   51.8   46.4   39.0   35.2   36.0

Quantiles      Min    25%    50%    75%    90%    Max
Sample  1:       0      0      1      2      4     41
Sample  2:       0      0      1      2      5     71
Sample  3:       0      0      0      1      3    107
Sample  4:       0      0      1      2      4    117
Sample  5:       0      0      1      2      5     59
Sample  6:       0      0      1      3      6     70
Sample  7:       0      0      1      3      6     76

Diversity indices
Sample       1       2       3       4       5       6       7
Chao      1846    1847    1808    1859    2074    2058    2061
Simpson 0.9985  0.9984  0.9958  0.9971  0.9984  0.9986  0.9984
Shannon 6.8044  6.8521  6.4607  6.6273  6.8587  6.9386  6.8813
```

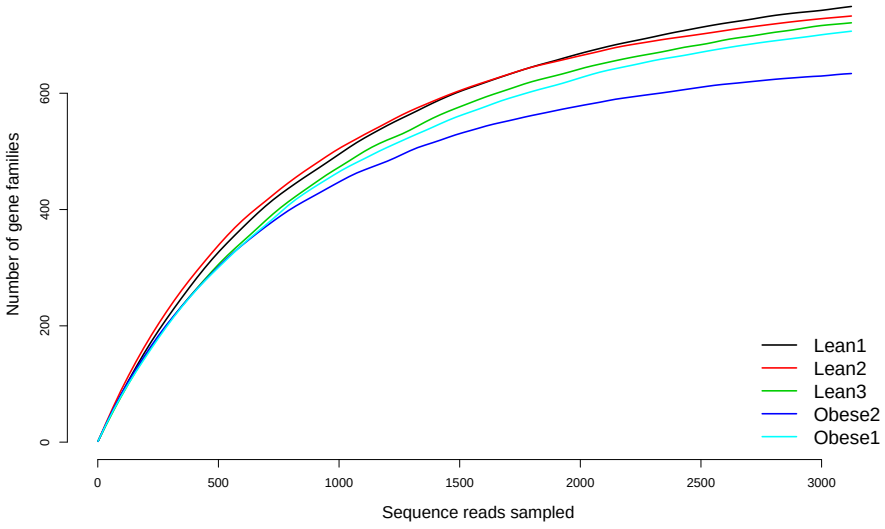
In addition to Chao's, Simplson's and Shannon's diversity indices, ShotgunFunctionalizeR can also perform rarefaction curve analysis. The function `diversityPlot.family` calculates and visualizes such curves for any number of samples. For the Ocean's interior dataset annotated as TIGRFAM the rarefaction curves can be shown using

```
> diversityPlot.family(OceanTIGR, sample=1:7)
```



A similar plot for the MouseCOG data looks like,

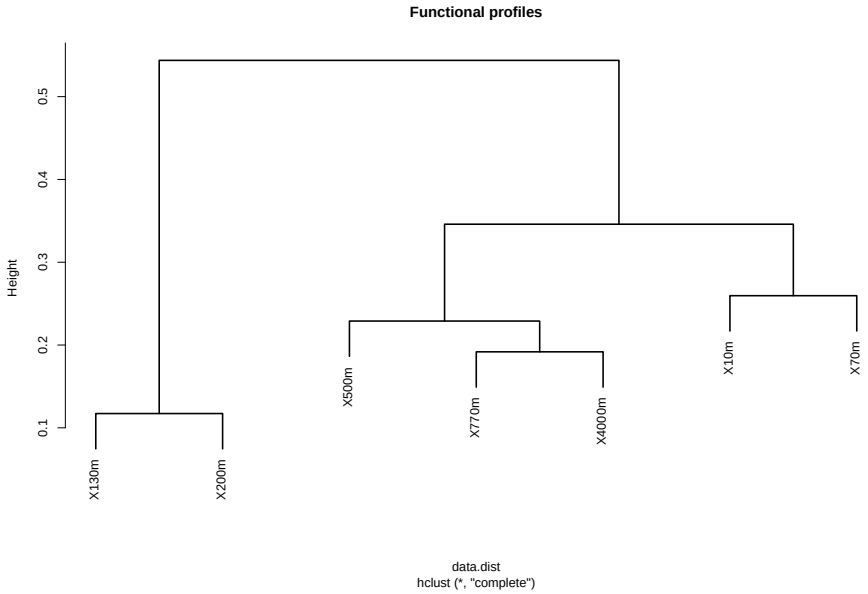
```
> diversityPlot.family(MouseCOG, sample=1:5)
```



ShotgunFunctionalizeR also contains methods for hierarchical clustering of samples based on their functional profile. The clustering is performed using the `hclust` command with a correlation distance metric, which makes it independent to the total number of observations in each sample. The clustering can

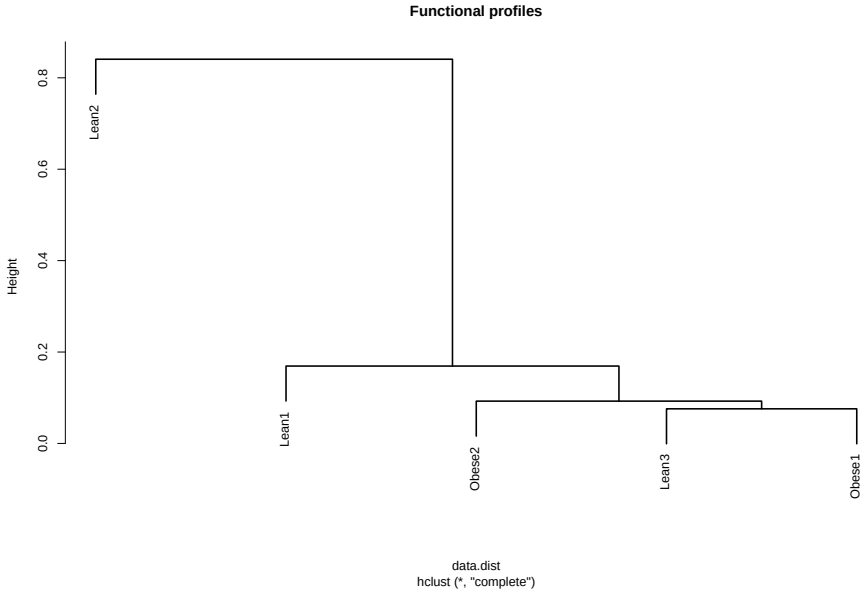
performed and visualized using

```
> clusterSamples(OceanTIGR, sample=1:7)
```



For the MouseCOG data we get

```
> clusterSamples(MouesCOG, sample=1:5)
```



5.7 Summary of methods in ShotgunFunctionalizeR

This section summarizes the methods currently implemented in ShotgunFunctionalizeR together with a brief description. For details of the methods we refer to Kristiansson et al. (2009a,b) and the reference manual accompanying the package.

5.7.1 Methods for gene-centric analysis

- Binomial test (`testGeneFamilies`): Tests whether the relative frequency of a gene family in sample 1 is different from that in sample 2.
- Hypergeometric test (`testGeneFamilies`): Tests whether the relative frequency of a gene family in sample 1 is different from that in sample 2. Slightly slower than the binomial test but derived from slightly different data assumptions.
- Gaussian test (`testGeneFamilies`): Tests whether the relative frequency of a gene family in sample 1 is different from that in sample 2. A normal approximation is performed and should only be used when having many counts.
- Poisson direct comparison (`testGeneFamilies.dircomp`): Direct comparison using the Poisson model. Used to compare two groups of samples.
- Poisson regression (`testGeneFamilies.regression`): Regression using the Poisson model. Used to identify gene families that follows a trend.
- General Poisson model (`testGeneFamilies`): Interface to the general Poisson model for gene families. Needs a design matrix.

5.7.2 Methods for pathway-centric analysis

- Gaussian sum (`testGeneCategories`): Uses a normal approximation (Z-score) to combine the results from all gene families in the category. Should be interpreted with care when the category have few counts. Works only for direct comparison of two samples.
- Independence (`testGeneCategories`): Using a $2 \times k$ contingency table (where k is the number of gene families in the category) to test if the abundance profile in metagenome 1 different from the profile in metagenome 2. Works only for direct comparison of two samples.
- Enrichment (`testGeneCategories`): Tests whether there is an enrichment of gene families from the gene category among the top list. Can be combined with any of the methods for ranking gene families.
- Poisson direct comparison (`testGeneCategories.dircomp`): Direct comparison using the Poisson model. Used to compare two groups of samples.

- Poisson regression (`testGeneCategories.regression`): Regression using the Poisson model. Used to identify gene categories for which the gene families follows a trend.
- General Poisson model (`testGeneCategories`): General Poisson model for gene categories. Needs a design matrix.

5.7.3 Functional diversity

- Rarefaction curve analysis (`densityPlot.family`): Comparison of the functional diversity of different samples based on resampling. The samples can have different number of reads.
- Hierarchical clustering (`clusterSamples`): Performs a hierarchical clustering based on the functional profiles.

5.8 Importing your own data

If you wish to analyze your own data in ShotgunFunctionalizeR you may do so by loading data from a text file. The text-file has a header which the first column is named `GeneFamily` and the subsequent specifies the sample names. Each column should then be **tab-separated**. The proceeding lines have the name of the gene family in the first column and then the counts in each metagenome. The first 10 lines of the Ocean dataset looks like this:

GeneFamily	10m	70m	130m	200m	500m	770m	4000m
COG2940	1	0	1	0	0	0	3
COG3264	0	2	0	1	1	0	4
COG1894	4	0	2	4	8	9	6
COG0731	0	0	1	1	1	1	0
COG0239	1	1	1	1	1	0	0
COG3125	0	0	0	0	1	0	0
COG5429	0	0	0	0	1	0	0
COG3657	0	0	0	0	0	0	1
COG0122	4	1	0	3	0	3	3

Similarly for the EC numbers:

GeneFamily	10m	70m	130m	200m	500m	770m	4000m
3.1.2.1	0	2	0	1	4	2	2
1.7.7.1	1	1	1	0	1	0	1
2.4.99.3	2	1	13	5	3	0	2
2.4.1.56	0	1	0	1	1	1	2
3.6.3.28	17	31	21	21	32	30	41
2.1.1.103	0	1	0	1	0	1	2
2.4.1.22	4	7	7	5	10	8	11
1.1.1.69	0	1	0	0	0	0	0
3.8.1.3	1	2	0	1	2	0	4

You may then read the file using:

```
> myData<-readGeneFamilies("MyCOGData.txt")
```

myData will now be your data-object used as the OceanCOG or OceanEC. If your file is in a different directory to where you started R you may either specify the absolute path:

```
> myData<-readGeneFamilies("/home/sven/data/MyCOGData.txt")
```

or change the working directory to the one where the file is:

```
> setwd("/home/sven/data/")
> myData<-readGeneFamilies("MyCOGData.txt")
```

5.8.1 Using custom annotations

In addition to the internally supported gene families (COG, EC, PFAM, TIGRFAM), ShotgunFunctionalizeR also have support for any custom-made annotation. The following example shows how to import such data using the `readGeneFamilies` function. First we start with a ShotgunFunctionalizeR data file custom-made gene family names. These can be named anything as long as they are unique.

```
GeneFamily      Sample1 Sample2 Sample3 Sample4
ID1     0       0       0       0
ID2     1       0       1       2
ID3     0       0       1       1
ID4     0       0       0       0
ID5     0       0       0       0
ID6     0       0       2       3
ID7     1       1       3       2
ID8     0       1       2       1
ID9     2       2       0       3
ID10    0       1       1       1
```

We will also need a cusotm annotation, which has to be in the form of a `data.frame` with two columns, one containign the custom-made gene family names and one the annotation.

```
> custom.annotation<-data.frame(GeneFamily=paste("ID", 1:10, sep=""),
                      Annotation=paste("Annotation number", 1:10))
> custom.annotation
   GeneFamily           Annotation
1         ID1  Annotation number 1
2         ID2  Annotation number 2
3         ID3  Annotation number 3
4         ID4  Annotation number 4
5         ID5  Annotation number 5
6         ID6  Annotation number 6
```

```
7          ID7  Annotation number 7
8          ID8  Annotation number 8
9          ID9  Annotation number 9
10        ID10 Annotation number 10
```

The data can now be imported using the following command

```
> data.custom<-readGeneFamilies(file="Files/CustomData.txt", type.family="custom",
                                  custom.annotation=custom.annotation)
```

5.9 Saving your plot in R

If you wish to save a plot to a pdf-file, use:

```
> x=0:314;
> pdf()
> plot(x, sin(x/50))
> dev.off()
```

The second line should be changed to which ever plot you want displayed in your file (you repeat the plot command you used to generate the plot you want to save). A file is created called `RPlot.pdf` in the directory your running R. You may specify your own file name with the `file` parameter in the `pdf` command. So if you wish to save the trend plot made in Section 5.4.3, use:

```
> pdf(file="mytrendplot.pdf")
> trendPlot.family(stat.po, family="COG0415")
> dev.off()
```

You may also use other formats such as postscript, `postscript`, by substituting the `pdf` command.

5.10 Scalability

The rapid development of high-throughput sequencing technology hmmhhmmhhmhm on handling large amount of data. When using ShotgunFunctionalizeR for functional analysis of metagenomes, the first step is to annotate the sequence reads. This is typically done by comparing each read against a reference database using tools such as BLAST or RPSBLAST (Altschul et al., 1997) and then assign the read to a gene familiy based on the best match.

Given a successful annotation, ShotgunFunctionalizeR offers excellent scalability. The data imported and analyzed by ShotgunFunctionalizeR constists gene families and their observations. Increasing the number of reads will not increase the number of gene family, only the number of times a gene family was observed. The computational complexity therfore almost independent to the number of reads.

The follow simple examples shows that this indeed the case. The time needed to compare the Lean groups versus the Obese group in the Mouse gut data was (R 2.8.1 running on a Dell Latitude E6400 with a 2.4 Ghz Intel Core2 Duo)

```
> system.time(po.stat<-testGeneFamilies.dircomp(MouseCOG,
                       groups=c("Lean", "Lean", "Lean", "Obese", "Obese")))
   user  system elapsed
  23.23    0.00   23.02
```

If we now multiple the number of observations with 10, i.e. assumes that we have ten times as many reads, we get the following result

```
> MouseCOG10<-MouseCOG
> MouseCOG10$Data<-MouseCOG10$Data*10
> system.time(po.stat10<-testGeneFamilies.dircomp(MouseCOG10,
                         groups=c("Lean", "Lean", "Lean", "Obese", "Obese")))
   user  system elapsed
  22.27    0.04   22.64
```

The time needed is more or less the same (it is actually faster, which is probably due to that the fact that Poisson model can, in some cases, converge faster when we have more observations).

When the number of samples increases, the scalability of ShotgunFunctionalizeR is not as good. In most situations, such as Poisson regression, the computational complexity should not increase faster than a linear function of the number of samples. In fact, running a similar analysis as on the Mouse Gut data as above but with twice as many samples actually takes only slightly longer time.

```
MouseCOG2$Data<-cbind(MouseCOG$Data, MouseCOG$Data*2)
> system.time(po.stat2<-testGeneFamilies.dircomp(MouseCOG2,
              groups=c("Lean", "Lean", "Lean", "Obese", "Obese",
               "Lean", "Lean", "Lean", "Obese", "Obese")))
   user  system elapsed
  23.79    0.00   23.61
```

References

Agresti, A. (2002). *Categorical data analysis*. Wiley series in probability and mathematical statistics.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*, 25(17):3389–3402.

Arnold, S. (1980). *The Theory of Linear Models and Multivariate Analysis*. John Wiley & Sons.

Bairoch, A. (2000). The enzyme database in 2000. *Nucleic Acids Res*, 28:304–5.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57.

DeLong, E., Preston, C., Mincer, T., Rich, V., Hallam, S., Frigaard, N., Martinez, A., Sullivan, M., Edwards, R., Brito, B., Chisholm, S., and Karl, D. (2006). Community genomics among stratified microbial assemblages in the ocean's. *Science*, 311:496–503.

Gentleman, R., Carey, V., Bates, D., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Li, F. L. C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y. H., and Zhang, J. (2004). Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80.

Gilbert, H., Pollard, K., van der Laan, M., and Dudoit, S. (2009). Resampling-based multiple hypothesis testing with applications to genomics: New developments in the r/bioconductor package multtest. *U.C. Berkeley Division of Biostatistics Working Paper Series*, Working Paper 249.

Kanehisa, M., Goto, S., Kawashima, S., and Nakaya, A. (2002). The KEGG databases at GenomeNet. *Nucleic Acids Res*, 30:42–6.

Kristiansson, E., Hugenholtz, P., and Dalevi, D. (2009a). ShotgunFunctionalizeR: An R-package for functional analysis of metagenomic data. *Bioinformatics*.

Kristiansson, E., Hugenholtz, P., and Dalevi, D. (2009b). ShotgunFunctionalizeR: An R-package for functional analysis of metagenomic data - Supplement. *Bioinformatics*.

Markowitz, V., Ivanova, N., Szeto, E., Palaniappan, K., Chu, K., Dalevi, D., Chen, I., Grechkin, Y., Dubchak, I., Anderson, I., Lykidis, A., Mavromatis, K., Hugenholtz, P., and Kyrpides, N. (2008). IMG/M: a data management and analysis system for metagenomes. *Nucleic Acids Res*, 36:D534–8.

Radhakrishna, R. and Toutenburg, H. (1999). *Linear Models: Least Squares and Alternatives*. Springer.

Tatusov, R., Fedorova, N., Jackson, J., Jacobs, A., Kiryutin, B., Koonin, E., Krylov, D., Mazumder, R., Mekhedov, S., Nikolskaya, A., Rao, B., Smirnov, S., Sverdlov, A., Vasudevan, S., Wolf, Y., Yin, J., and Natale, D. (2003). The cog database: an updated version includes eukaryotes. *BMC Genomics*, 4.

Tatusov, R., Koonin, E., and Lipman, D. (1997). A genomic perspective on protein families. *Science*, 278:631–7.

Turnbaugh, P., Ley, R., Mahowald, M., Magrini, V., Mardis, E., and Gordon, J. (2006). An obesity-associated gut microbiome with increased capacity for energy harvest. *Nature*, 444:1027–131.