Working with substrate information in opm

Lea A.I. Vaas Leibniz Institute DSMZ
Benjamin Hofner IMBE, FAU Erlangen-Nürnberg
Markus Göker Leibniz Institute DSMZ

Abstract

This is the substrate-information tutorial of **opm** in the version of March 3, 2016. The stored information on the known Phenotype Microarray (PM) substrates is explained, as well as the methods available to query this information. IDs for a variety of databases are stored within **opm** and can be used to conduct web queries to obtain comprehensive information on the substrates of interest. We show how these data can be used to visualise results from PM experiments, including the outcome from advanced multiple-comparison statistics, within biochemical pathway maps. Visually comparing genome annotation and PM results is easily possible in that manner. Moreover, methods are described to automatically detect the substrate features that potentially explain a given experimental outcome. This includes determining the relevant pathways to be used in the visualisations. The final chapters of this tutorial provide examples for the powerful feature-selection approaches available within R with a focus on state-of-the-art boosting implementations.

Keywords: Respiration Kinetics, Pathways, Feature Selection, Boosting, CAS, MeSH, ChEBI, MetaCyc, KEGG, SEED, **pathview**, **mboost**.

1. Introduction

A detailed description of the OmniLog® Phenotype Microarray (PM) system, its measuring procedure and data characteristics are found in the vignette "**opm**: An R Package for Analysing Phenotype Microarray and Growth Curve Data" (called "main tutorial" in the following). The description of the methods below presupposes that the user is familiar with the usage of **opm** and has studied the main tutorial as well as the entries of the **opm** manual relevant to her or his research. Especially the concepts behind, and the methods available for, the different classes of **opm** objects should be known before starting with this tutorial.

In addition to visual inspection or statistical comparative analyses of PM data, as described in the main tutorial, users might be interested in specific information on the substrates used in PM assays. The **opm** package contains a large variety of additional data on PM substrates. Beside methods for assessing this information directly, this tutorial introduces strategies for visualising the measured PM results by mapping them on pathway maps. Furthermore, analysis methods are described for modelling the effect of substrate features on the PM results and thus, e.g., for the identification of those pathways that are particularly suitable for visualising the PM results in pathway graphs.

2. Preparation

For just downloading information from Kyoto Encyclopedia of Genes and Genomes (KEGG) (see Section 3.1), install the Bioconductor package **KEGGREST**. It needs not be loaded into your R session. For also drawing PM information into KEGG pathway maps (see Section 4), install the Bioconductor package **pathview** and load it into your session. Note that it is important to load **pathview** before **opm**, which is needed throughout this tutorial, since otherwise some methods are not visible and the package does not work properly. In this vignette this is enforced by optionally detaching **opm** and loading it (again) as follows:

```
R> suppressPackageStartupMessages(library("pathview"))
R> if ("package:opm" %in% search())
     detach("package:opm", unload = TRUE)
R> library("opm")
R> data(vaas_et_al, package = "opmdata")
```

3. Accessing plate and substrate information

The **opm** package contains a number of functions suitable for accessing stored information on entire plates and on the substrates within certain wells.

3.1. Available plate information

Currently substrate layouts of various plates are available within **opm**. An overview of the plate types available in the respective version of **opm** is obtained by entering

```
R> plate_type(full = TRUE)
```

The resulting vector of names does not only include OmniLog® plates; see the manual and the main tutorial for further details. Using other values for `full`, or additional arguments, distinct spelling variants of the plate names can be obtained.

3.2. Available substrate information

In the manual these functions are assigned to the family "naming-functions" with according cross-references.

One usually would start a search by determining the exact spelling of an internally used name with `find_substrate`:

```
R> substrates <- find_substrate(c("Glutamine", "Glutamic acid"))
R> substrates
```

The result is a list (of the S3 class "substrate_match") containing character vectors with the results for each query name as values. Surprisingly, nothing was found for "Glutamic acid" but several values for "Glutamine". The default `search` argument is "exact", which is exact (case-sensitive) matching of *parts* of the names. One might want to use "glob" searching mode:

```
R> substrates <- find_substrate(c("L-Glutamine", "L-Glutamic acid"), "glob")
R> substrates
```

But with so-called wild-cards, i.e. "*" for zero to many and "?" for a single arbitrary character the search is more flexible:

```
R> substrates <- find_substrate(c("*L-Glutamine", "*L-Glutamic acid"), "glob")
R> substrates
```

This fetches all terms that end in either query character string, and does so case-insensitively. Advanced users can apply the much more powerful `regex` and `approx` search modes, which are described in the manual.

Note that **opm** appends a concentration (or just repetition) indicator as a number after a hash sign ("#") to the substrate names wherever necessary. Thus a wild-card "*" at the end of a name might often by the most useful search pattern.

Once the internally used names (which are not guaranteed to be stable between distinct **opm** releases) have been found, information on the substrates can be queried such as their occurrences and positions on plates:

```
R> positions <- find_positions(substrates)
R> positions
```

This yields a nested list containing two-column matrices with plate names in the first and well coordinates in the second column. Using the `type` argument, search can be restricted to a plate type of interest, which would yield a named vector. References to external data resources for each substrate name can be obtained using `substrate_info`:

```
R> subst.info <- substrate_info(substrates)
R> subst.info
```

By default this yields Chemical Abstracts Service (CAS) numbers (http://www.cas.org/content/chemical-substances/faqs), but Medical Subject Headings (MeSH) names (useful for conducting PubMed queries; see http://www.ncbi.nlm.nih.gov/mesh/) (Coletti and Bleich 2001), Chemical Entities of Biological Interest (ChEBI) IDs (Hastings, de Matos, Dekker, Ennis, Harsha, Kale, Muthukrishnan, Owen, Turner, Williams, and Steinbeck 2013), KEGG compound IDs, KEGG drug IDs (Kanehisa, Goto, Furumichi, Tanabe, and Hirakawa 2010), MetaCyc compound IDs (Caspi, Altman, Dreher, Fulcher, Subhraveti, Keseler, Kothari, Krummenacker, Latendresse, Mueller, Ong, Paley, Pujar, Shearer, Travers, Weerasinghe, Zhang, and Karp 2012) and SEED compound IDs (Overbeek, Begley, Butler, Choudhuri, Chuang, Cohoon, de Crecy-Lagard, Diaz, Disz, Edwards, Fonstein, Frank, Gerdes, Glass, Goesmann, Hanson, Iwata-Reuyl, Jensen, Jamshidi, Krause, Kubal, Larsen, Linke, McHardy, Meyer, Neuweger, Olsen, Olson, Osterman, Portnoy, Pusch, Rodionov, Rueckert, Steiner, Stevens, Thiele, Vassieva, Ye, Zagnitko, and Vonstein 2005) have also been collected for the majority of the substrates. Using the "browse" argument, full URLs can be created and optionally also directly opened in the default web browser. Using the "download" argument, if KEGG drug or compound IDs have been selected, these can be downloaded from the KEGG server (if the **KEGGREST** package is available) and converted into customised objects. It is possible to nicely display all available information at once:

```
R> subst.info <- substrate_info(substrates, "all")
R> subst.info
```

Another use of `substrate_info` is to convert substrate names to lower case but protecting name components such as abbreviations or chemical symbols, as described in the manual.

4. Visualisation of PM information within pathway maps

In conjunction with other R packages, it is possible to visualise PM results directly in already existing pathway maps as, for example, those from KEGG. These maps are essentially manually drawn biochemical pathway maps representing the currently available knowledge on substrates, enzymes and genes and their connections within pathways. Depending on the availability of genome and gene-annotation information within KEGG, organism-specific, individual maps can be obtained (Kanehisa *et al.* 2010).

The mapping itself works by using information produced by **opm** for a colour coding of the nodes (here, representing the substrates) within those maps, as can be done similarly with several other types of "OMICS" data such as transcriptomics or proteomics data. For details, see the description on the KEGG website (`http://www.genome.jp/kegg/`).

4.1. Providing suitable input data

The work flow starts with either an `OPMX` object containing the aggregated values or the result from an `opm_mcp` analysis. The first step in both cases is to convert the data into a suitable format, which is a named vector created by the function `annotated`.

```
R> x <- annotated(vaas_1)
R> head(x)

    <NA>    C00721    C00208    C01083    C00185    C08240
123.4558 248.1809 284.0994 269.7548 180.7536 287.7959
```

The resulting vector contains the numeric values (selected parameter estimates or `opm_mcp` results, as explained below) as well as an annotation of the according substrates. For substrates such as "Positive Control" or "pH 5" no KEGG Identifier (ID) is available, which results in `NA` values in the vector. Accordingly, those substrates cannot be marked within pathway maps (see section 4.2.1). The `what` argument, passed as eponymous argument to `substrate_info`, selects the kind of information to be used for the annotation. With `annotated` used with `how = "value"` a numeric matrix including the substrate names as row names and first column indicating the mean of chosen computed values is provided. Further columns indicate the occurrence of a certain substrate in a pathway map or the affiliation to a certain class, e.g. "Carbohydrates". This information is in analogy to the download argument of `substrate_info` but with conversion to a numeric matrix. For usage of argument `how`, please refer to section 5.

Although `annotated` works directly on `OPMX` objects containing aggregated data for single plates or bundles of plates, please note, that the output allows for only one value per substrate. Thus, when applying `annotated` to a set of plates, make sure that only one experimental group is comprised, since the resulting values are averaged per well over all plates in the input object. Using the `output` argument, one can select the parameter of interest, for example area under the curve instead of maximum height:

```
R> x <- annotated(vaas_1, output = param_names()[4])
R> head(x)
```

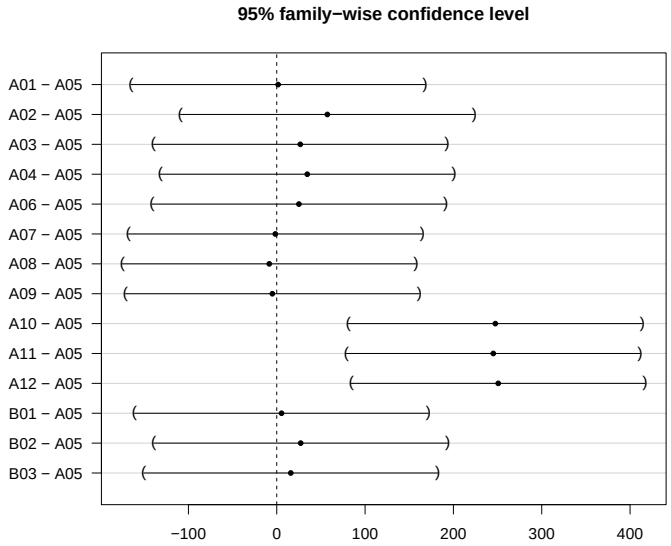

Figure 1: Point estimates and 95% confidence intervals in a manually defined comparison of group means for a specifically selected set of wells from the `vaas_4` exemplar object. In this procedure each selected well is compared against A05. The picture was obtained by running `opm_mcp` and then the plotting function for the resulting `opm_glht` object. See the main tutorial for details.

```
     <NA>     C00721     C00208     C01083     C00185     C08240
 8918.137 18391.590 21960.080 18531.180 11831.150 19254.160
```

Visualisation of the results of an `opm_mcp` analysis is also possible, which offers more (statistically interesting) opportunities for making sense of the PM data in the context of pathways. However, this method only makes sense if each coefficient estimated by `opm_mcp` can be linked to a single substrate. This is usually only possible for the "Dunnett" and "Pairs" type of contrasts if applied to the wells (see Section 4.2.3). See the main tutorial and the manual for details on this restriction.

The results from an `opm_mcp` procedure are treated with `annotated` as shown before with an `OPMX` object, but additional options are available. In the following example, first an `opm_glht` object is generated from the `vaas_4` exemplar object by performing a Dunnett-type multiple comparison of the selected 13 wells against well A05 as control group. The comparison applies to the default parameter given by `opm_opt("curve.param")`; see the manual for details.

```
R> x <- opm_mcp(vaas_4[, , 1:15], output = "mcp", model = ~ Well,
     linfct = c(Dunnett.A05 = 1), full = FALSE)
```

The resulting 95% confidence intervals for the difference of means are plotted in Figure 1.

Using the above generated `opm_glht` object, the options modifying the output of `annotated` will be illustrated. Apparently only three comparisons exhibit a statistically significant difference, namely the comparisons A10 - A05, A11 - A05 and A12 - A05, all showing that the reactions in A05 are weaker than those in A10, A11 and A12, respectively.

Using the `output` argument, users are able to obtain various statistically relevant categorical results instead of the simple numerical output of the respective point estimator. The options `upwards` and `downwards` result in a classification into three categories (`FALSE`, `NA`, or `TRUE`). These indicate whether or not the cut-off (zero per default) is included in the confidence interval (`NA`) and thus a decision possible. If not, the category indicates the direction of the shift relative to the cut-off. The options `different`, `smaller`, `larger` and `equal` work similarly, but use only the two categories `TRUE` and `FALSE`. Please note that the underlying test seeks for differences and thus the results always have to be interpreted regarding the significance (and magnitude) of these differences; "insignificantly different" does not mean "significantly equal"!

Short-cuts are available for all `output`-options, enabling the user to set the cut-off together with the kind of output. See the manual for details.

A comprehensive overview of the possible results for object `x` is shown in the following data frame:

```
   Comparison    numeric upwards downwards different equal smaller larger
1   A01 - A05   1.577317      NA        NA     FALSE  TRUE   FALSE  FALSE
2   A02 - A05  57.274661      NA        NA     FALSE  TRUE   FALSE  FALSE
3   A03 - A05  26.661023      NA        NA     FALSE  TRUE   FALSE  FALSE
4   A04 - A05  34.537328      NA        NA     FALSE  TRUE   FALSE  FALSE
5   A06 - A05  25.078779      NA        NA     FALSE  TRUE   FALSE  FALSE
6   A07 - A05  -1.606236      NA        NA     FALSE  TRUE   FALSE  FALSE
7   A08 - A05  -8.458879      NA        NA     FALSE  TRUE   FALSE  FALSE
8   A09 - A05  -4.975284      NA        NA     FALSE  TRUE   FALSE  FALSE
9   A10 - A05 247.470724    TRUE     FALSE      TRUE FALSE   FALSE   TRUE
10  A11 - A05 245.163382    TRUE     FALSE      TRUE FALSE   FALSE   TRUE
11  A12 - A05 250.763650    TRUE     FALSE      TRUE FALSE   FALSE   TRUE
12  B01 - A05   5.417463      NA        NA     FALSE  TRUE   FALSE  FALSE
13  B02 - A05  27.079437      NA        NA     FALSE  TRUE   FALSE  FALSE
14  B03 - A05  15.870312      NA        NA     FALSE  TRUE   FALSE  FALSE
```

All these results are obtained with the setting `how = "ids"`; for the usage of `how = "value"` see Section 5.

4.2. Visualisation in pathway maps using pathview

4.2.1. Visualisation of group means in pathway maps

Here we will use the function `pathview` from the package of the same name (Luo and Brouwer 2013). This function can download a user-defined pathway graph from KEGG, optionally integrate additional data from other sources, and render the result. For integrating experimental data from other "OMICS" approaches (such as transcriptomics and proteomics), see the corresponding chapter in the **pathview** vignette for details.

Here the `pathview` function serves for integrating and visualising information produced by **opm** and provided by `annotated`. The user only has to specify the pathway and provide the **opm** data. All other necessary steps (download of pathway graph data as XML file from KEGG, parsing of this data file, integrating user-defined data into the pathway representation,

and rendering of final output graphics) are automatically conducted by `pathview`. See the **pathview** vignette for technical details.

In the case of KEGG, a pathway map is described as "a molecular interaction/reaction network diagram represented in terms of the KEGG Orthology (KO) groups" (see http://www.genome.jp/kegg/kegg3a.html for further details). KEGG contains a collection of distinct types of pathway maps identified by a code containing between two and four letters as a prefix, followed by five digits.

The prefixes have the following meanings:

map - Reference pathway

ko - Reference pathway (KO)

ec - Reference pathway (Enzyme Commission (EC))

rn - Reference pathway (Reaction)

org - Organism-specific pathway map (*org* is a wild-card for the organism-specific abbreviation composed of two to four letters)

Only the first reference pathway map is drawn manually; all other maps are computationally generated. The `ko` maps contain the manually defined ortholog groups (`ko` entries) for all proteins and functional RNA molecules that correspond to KEGG pathway nodes, BRITE hierarchy nodes, and KEGG module nodes. The `ko` entries are converted to gene ID if organism-specific pathways maps are generated. A list of the existing maps and their corresponding numbers are available on the KEGG homepage (see above).

`pathview` allows only for using KEGG orthology (the `ko` maps) or species-specific letter codes. See http://www.genome.jp/kegg/catalog/org_list.html for an up-to-date list of organisms with complete genome information in KEGG.

A vector as returned by `annotated` (see Section 4.1) serves as input for the visualisation procedure based on `pathview`. For demonstration purposes, we use subsets of `vaas_et_al` containing the *Escherichia coli* strains from the first biological repetition.

```
R> coli.sub <- subset(vaas_et_al, list(Species = "Escherichia coli",
     Experiment = "First replicate"))
R> coli.k12 <- subset(coli.sub, list(Strain = "DSM18039"))
R> coli.type <- subset(coli.sub, list(Strain = "DSM30083T"))
```

Afterwards we create the annotated vectors containing the average maximum curve heights for the two groups separately:

```
R> anno.k12 <- annotated(coli.k12)
R> anno.type <- annotated(coli.type)
```

For a more convenient drawing of **opm** data on KEGG pathway maps, we suggest a wrapper for the `pathview` function, providing other default settings for some arguments. All graphics below are produced using this wrapper, but the user is of course free to use the original `pathview` function or write an alternative wrapper.

```
R> opm_path <- function(cpd.data, gene.data = NULL,
       high = list(gene = "green4", cpd = "blue"),
       mid = list(gene = "lightsteelblue1", cpd = "yellow"),
       low = list(gene = "white", cpd = "yellow"),
       species = "ko", out.suffix = "non-native",
       key.pos = "topright", afactor = 1000,
       limit = list(gene = 2, cpd = 400),
       bins = list(gene = 0.5, cpd = 40),
       both.dirs = list(gene = FALSE, cpd = FALSE),
       sign.pos = "topleft", cpd.lab.offset = 0,
       same.layer = FALSE,
       na.col = "white", ...) {
     pathview(cpd.data = cpd.data, gene.data = gene.data,
       high = high, mid = mid, low = low,
       species = species, out.suffix = out.suffix, key.pos = key.pos,
       afactor = afactor, limit = limit, bins = bins,
       both.dirs = both.dirs, sign.pos = sign.pos,
       cpd.lab.offset = cpd.lab.offset, same.layer = same.layer,
       na.col = na.col, ...)
   }
```

The data for the two strains are shown on the correspondingly separated maps in Figure 2 and 3.

```
R> coli.map.k12 <- opm_path(cpd.data = anno.k12, species = "ko",
     out.suffix = "k12.ko", pathway.id = "00052")
R> coli.map.type <- opm_path(cpd.data = anno.type, species = "ko",
     out.suffix = "type.ko", pathway.id = "00052")
```

Note particularly the substrates Raffinose, Stachyose and Sucrose (in the middle of the map), which exhibit large respiratory differences, while Sorbitol (on the very left of the map) yields only small respiratory differences between the two strains.

Using the default settings, `pathview` yields a raster image in Portable Network Graphics (PNG) format, which is stored in the current working directory and shown in Figure 2 and 3. For demonstration purposes the pathway number "00052", which encodes for the Galactose metabolism pathway map, is chosen. Genes (boxes) are annotated with KEGG ontology numbers (set by choosing `species = "ko"`), where available or, alternatively, with EC numbers. Note that the `species` arguments offers the possibility to use species-specific genome information available in the KEGG directory; see above for the letter codes and below for an application example. The substrates (circles) in the maps get standard compound names, which are automatically retrieved from the ChEMBL database using the compound IDs.

The data for the two strains can be shown analogously using the categorical output of `annotated`, see Figure 4 and 5. This works because the underlying OPMX object contains discretised data. Whereas `annotated` would by default return a logical vector in that case, the `lmap` argument can be used to create a numeric vector on the fly. See the manual for applications of `lmap`.

```
R> anno.k12.bin <- annotated(coli.k12, output = param_names("disc.name"),
     lmap = 1:3)
```

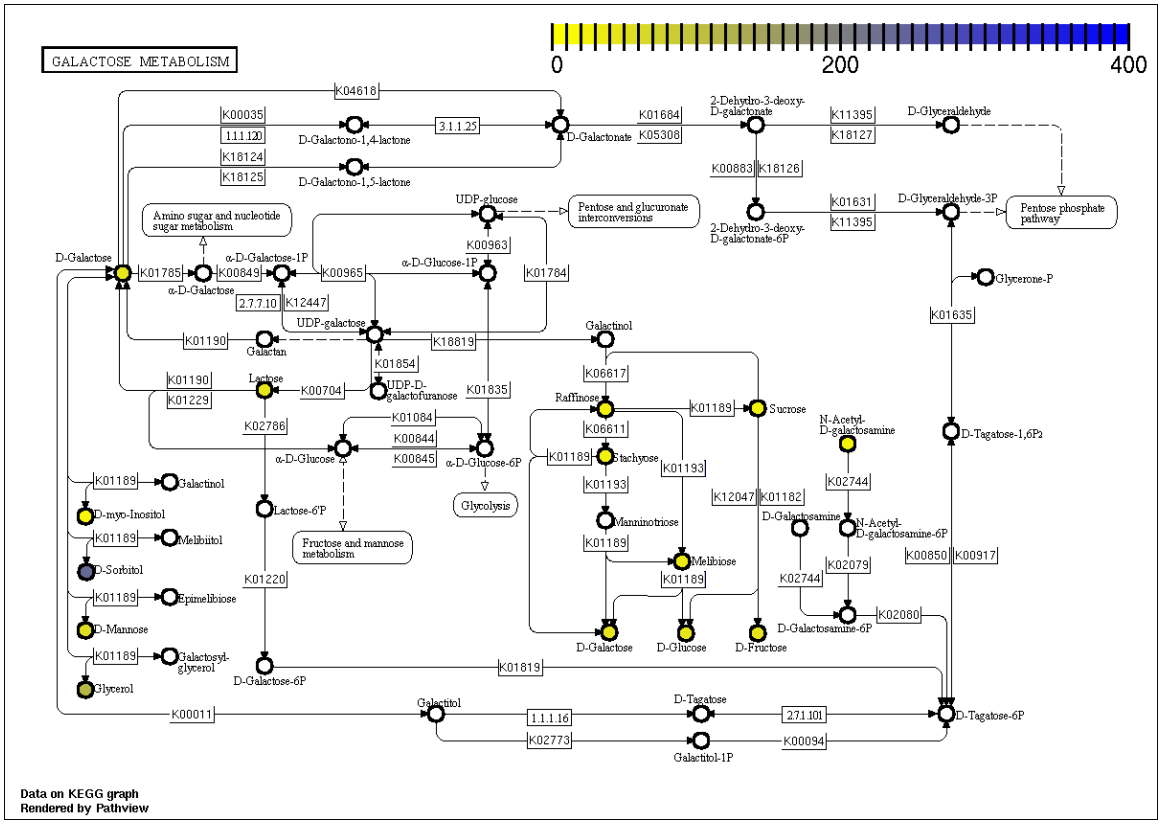

Figure 2: Respiratory data (mean maximum height) of *Escherichia coli* strain K12 mapped on the KEGG Galactose pathway. The aggregated measurement data are represented by according colours and mapped on the corresponding substrates (circles) in the graph. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

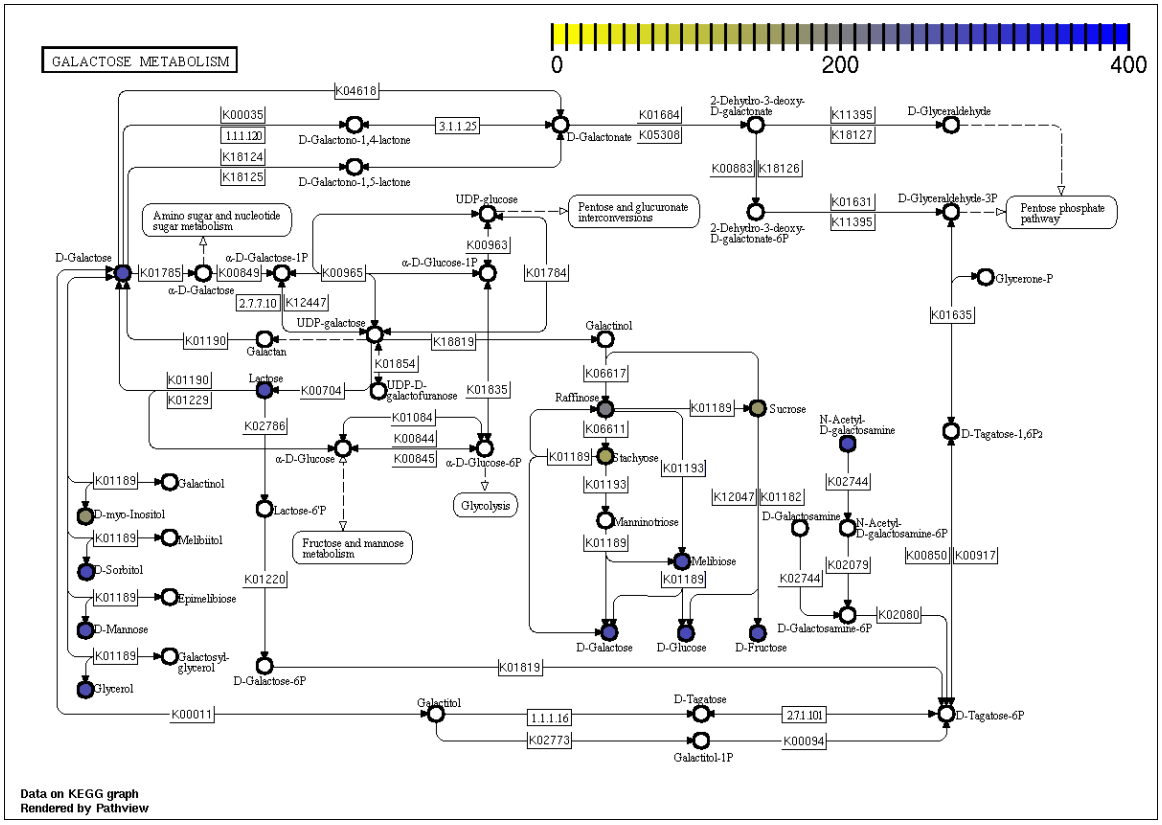

Figure 3: Respiratory data (mean maximum height) of *Escherichia coli* type strain DSM 30083^{T} mapped on the KEGG Galactose pathway. The aggregated measurement data are represented by according colours and mapped on the corresponding substrates (circles) in the graph. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

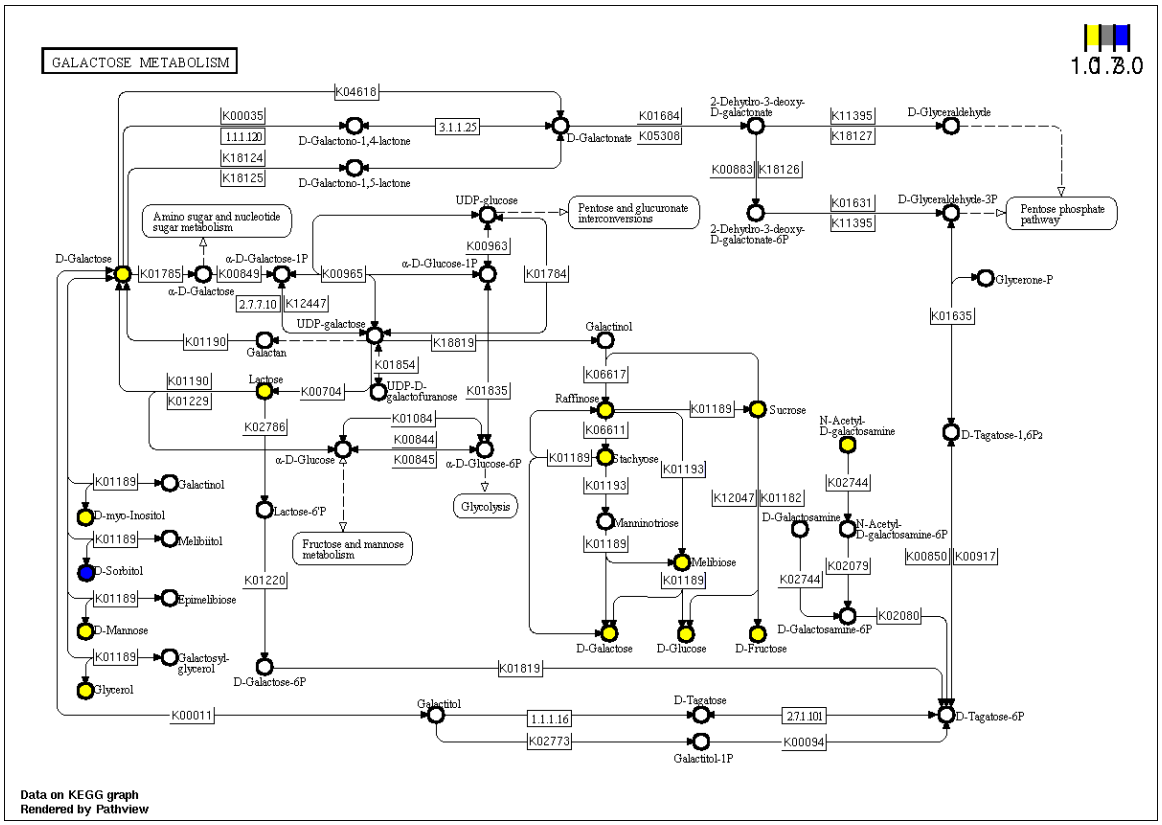

Figure 4: Respiratory data (mean maximum height, discretised) of *Escherichia coli* strain K12 mapped on the KEGG Galactose pathway. The aggregated and discretised measurement data are represented by according colours and mapped on the corresponding substrates (circles) in the graph. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

```
R> anno.type.bin <- annotated(coli.type, output = param_names("disc.name"),
     lmap = 1:3)
R> coli.map.k12.bin <- opm_path(cpd.data = anno.k12.bin, species = "ko",
     out.suffix = "k12.ko.bin", pathway.id = "00052",
     limit = list(gene = 2, cpd = c(1, 3)), bins = list(gene = 0.5, cpd = 3))
R> coli.map.type.bin <- opm_path(cpd.data = anno.type.bin, species = "ko",
     out.suffix = "type.ko.bin", pathway.id = "00052",
     limit = list(gene = 2, cpd = c(1, 3)), bins = list(gene = 0.5, cpd = 3))
```

By using the `species` argument it is possible to include the genome annotations available from KEGG (see above for details). Since in KEGG no genome for the *E. coli* type strain is available, we will demonstrate the usage of this argument only for strain K12 (= DSM 18039). One could choose the annotation of strain *E. coli* "K-12 MG1655" from the year 1997, which corresponds to `species = "eco"` (other K12 variants are available in KEGG). In the corresponding figure, genes (boxes) without annotation information in the chosen genomes remained white without any labelling. But when using `na.col`, entries without annotation information are highlighted with the colour of choice.

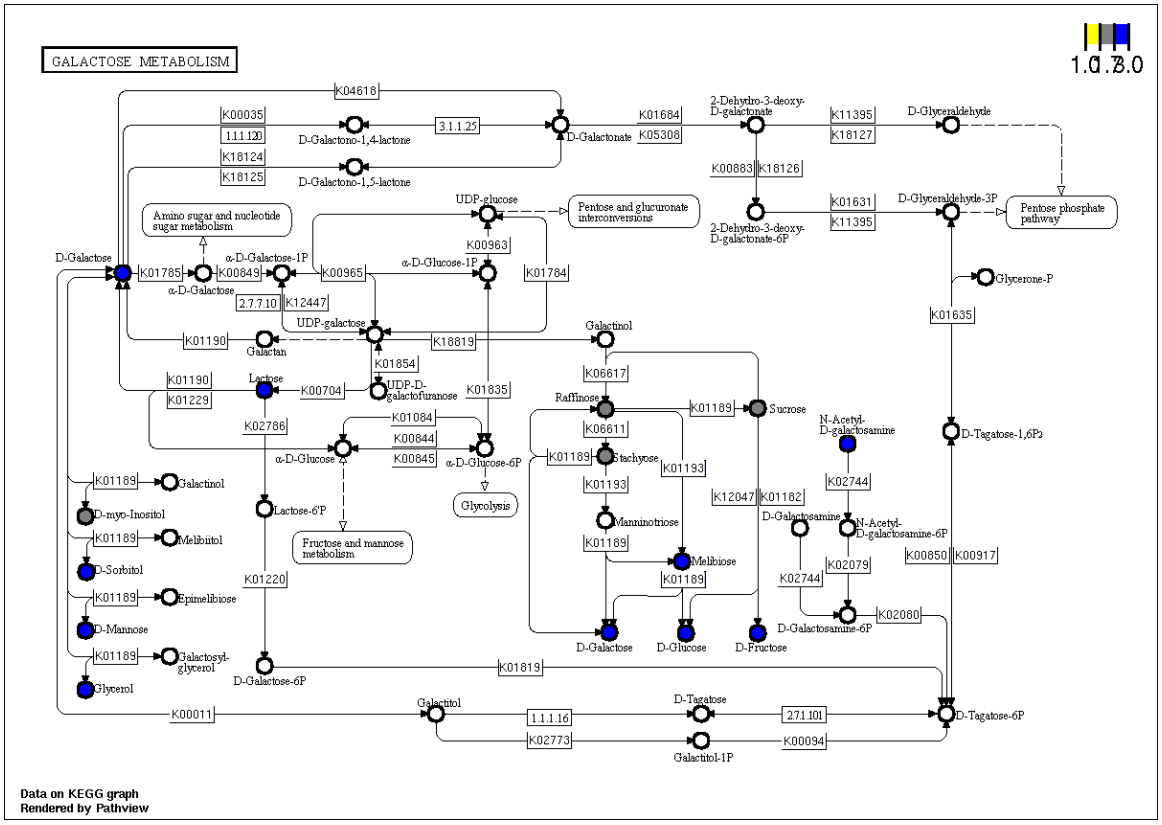

Figure 5: Respiratory data (mean maximum height, discretised) of *Escherichia coli* type strain DSM 30083$^{\mathrm{T}}$ mapped on the KEGG Galactose pathway. The aggregated and discretised measurement data are represented by according colours and mapped on the corresponding substrates (circles) in the graph. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

```
R> coli.map.k12.eco <- opm_path(cpd.data = anno.k12, species = "eco",
     out.suffix = "k12.eco", pathway.id = "00052", na.col = "pink")
```

We do not run this code (and show the resulting figure) here because it involved installing a Bioconductor package that includes the KEGG annotation for `species = "eco"`. This package would be selected and downloaded automatically, so this is rather convenient for the user, but we refrain from modifying user libraries within a vignette.

With this type of visualisation users can detect metabolic steps for which no genes are annotated, but PM data indicate metabolic activity. This can help improving genome annotation. In the chosen example, the failure of the strain to metabolise Galactose is in accordance with the lack of some genes in the genome annotation, causing gaps in the pathway for Galactose catabolism.

4.2.2. Finding substrates within pathways

Note that from the annotation objects `anno.k12` or `anno.type`, respectively, comprising 96 substrates, only 13 are represented in the shown pathway map in Figure 2 and 3. This is no wonder because the PM plates are not arranged according to their affiliation to KEGG pathways. It often makes sense to restrict the considered substrates beforehand if the pathway of interest is already known. This particularly saves running time in the calls to `opm_mcp` and the `annotated` method for opm_glht objects.

When using the option `how = "value"`, `annotated` yields a numeric matrix with substrate names as row names and pathway ID as column names. Whereas the main use of such a matrix is described in Section 5, it can also be used simply to show the distribution of substrates over pathways. Ones and zeros indicate whether or not a certain substrate (row) is contained in a certain pathway (column). `NA`s indicate that a substrate has no KEGG ID, as for example well A01 which harbours the (pseudo-)substrate "Negative Control".

By summing up the columns and sorting the resulting vector, the user gets a ranking of the columns (pathways) indicating how many substrates are covered.

```
R> anno.k12.mat <- annotated(coli.k12, how = "value")
R> col.sums <- sort(colSums(anno.k12.mat, na.rm = TRUE), decreasing = TRUE)
R> col.sums[1:10]
```

```
                     exact_mass                           Value
                       17697.97                        14820.75
                       map01100                     Carbohydrates
                          45.00                           26.00
                       map02010                        Chemicals
                          22.00                           21.00
                       map01120             Third-class OTC drugs
                          19.00                           19.00
Inorganic and organic chemicals                   Monosaccharides
                          18.00                           18.00
```

In the next example we search for the substrates represented in pathway number "00052", which is Galactose metabolism. Then we extract the positions of these substrates (for the plate type of interest) with `find_positions`:

```
R> e.subs <- rownames(anno.k12.mat)[!is.na(anno.k12.mat[, "map00052"]) &
     anno.k12.mat[, "map00052"] > 0]
R> e.subs.pos <- find_positions(e.subs, type = "Gen III")
R> e.subs.pos
```

```
                Sucrose                Stachyose              D-Raffinose
                  "A07"                    "A09"                    "B01"
          a-D-Lactose              D-Melibiose N-Acetyl-D-Galactosamine
                  "B02"                    "B03"                    "B08"
            D-Glucose                D-Mannose               D-Fructose
                  "C01"                    "C02"                    "C03"
          D-Galactose               D-Sorbitol             myo-Inositol
                  "C04"                    "D01"                    "D04"
             Glycerol
                  "D05"
```

4.2.3. Visualisation of differences of group means in pathway maps

Next, we show the maximum-height values from the 13 substrates represented in the Galactose pathway map (number "00052") in Figure 6 to demonstrate the sizes of their differences. Remember that the vector `e.subs.pos` contains the positions of the substrates of interest as a character string. It can thus directly be used to subset `OPMX` objects.

```
R> old.mar <- par(mar = c(3, 2, 2, 2)) # adapt plotting settings
R> ci_plot(object = coli.sub[, , e.subs.pos],
     as.labels = list("Species", "Strain"), subset = "A", x = "bottomright",
     draw.legend = TRUE, crr = 1.33, legend.field = NULL)
R> par(old.mar) # reset plotting settings
```

Straightforwardly, we compute a multiple comparison between only the 13 substrates included in the Galactose metabolism pathway map. Our example compares the type strain with K12; each corresponding 95% Confidence Interval (CI) for differences of means for the chosen substrates is shown in Figure 7.

```
R> coli.comp <- opm_mcp(coli.sub[, , e.subs.pos],
     output = "mcp", model = ~ J(Well + Strain), linfct = c(Pairs = 1))
```

The annotation vector for the differences of means can be obtained by simply applying `annotated` to the `opm_glht` object. The direct mapping of these differences between the two strains on the Galactose pathway is shown in Figure 8.

```
R> coli.comp.map <- opm_path(cpd.data = annotated(coli.comp), species = "ko",
     out.suffix = "coli.comp.ko", pathway.id = "00052")
```

In analogy to the last example, the function `annotated` can be used to produce categorical annotation vectors for the differences of means. Such vectors are very useful because they can specifically highlight the statistically significant differences, and particularly those that are significantly larger than a certain biologically relevant minimum effect size. Thus the full

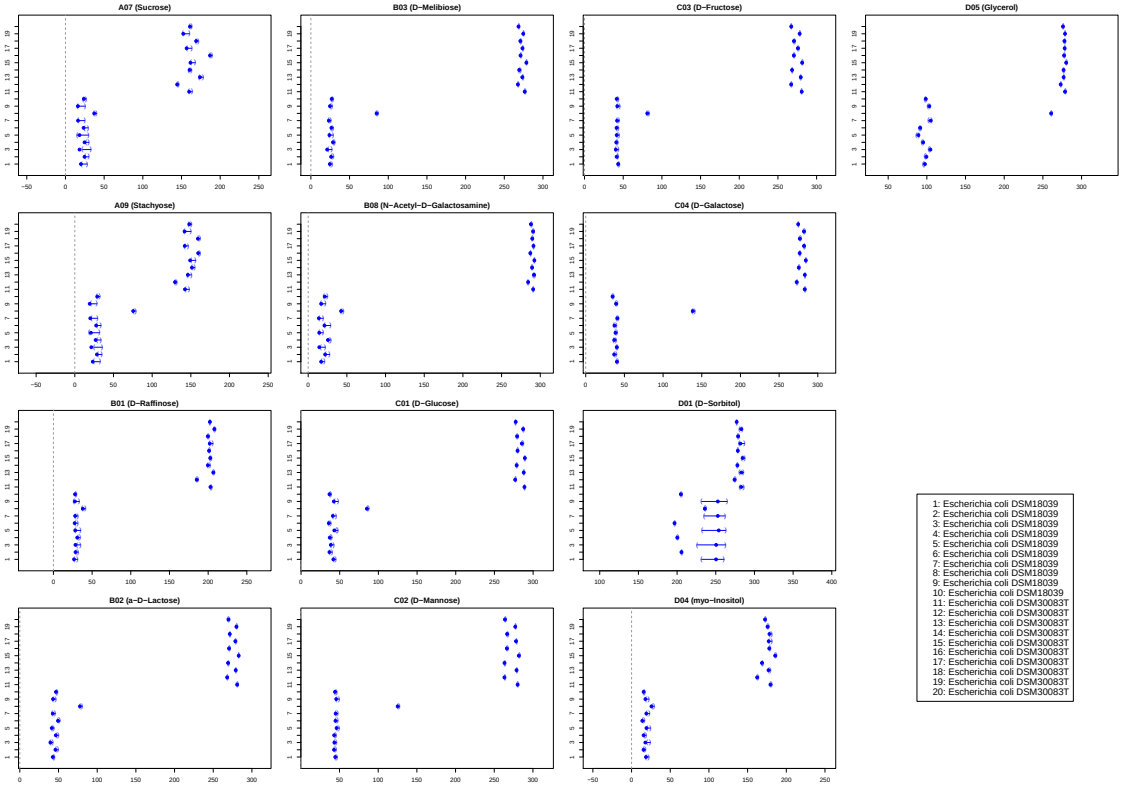

Figure 6: Point estimates and 95% confidence intervals for the single maximum-height values of the two *E. coli* strains for the subset of substrates represented in the Galactose pathway map.

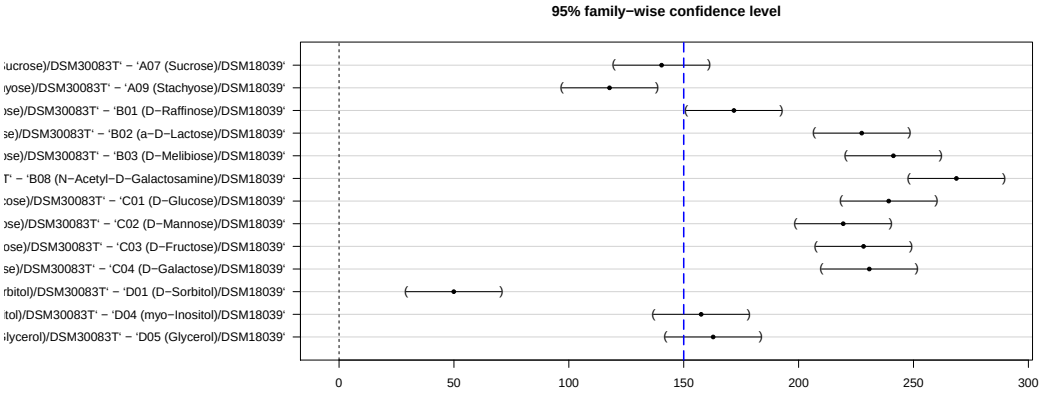

Figure 7: Point estimates and 95% confidence intervals for the differences of means between the two *E. coli* strains for the subset of substrates represented in the Galactose pathway map. The blue dashed line indicates a minimal effect size of 150 as used in Figure 10. The default is 0, as used in Figure 9.

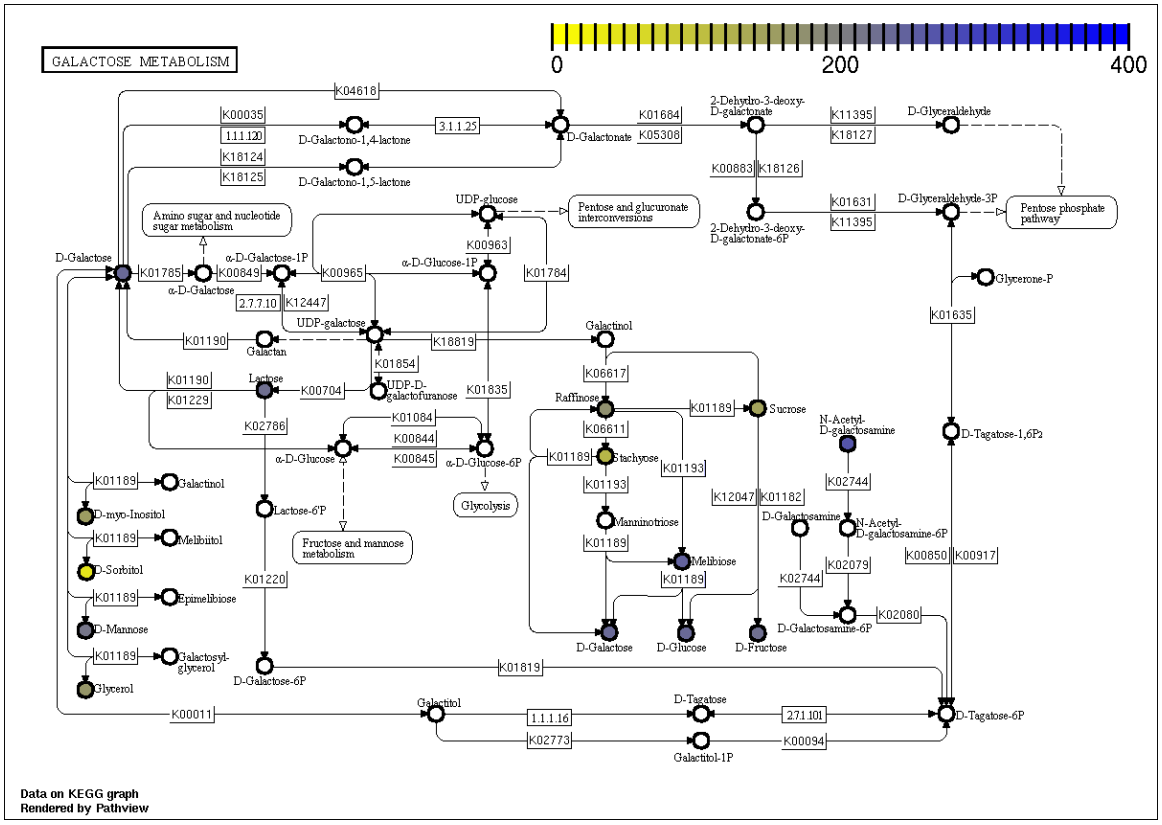

Figure 8: Respiration differences between *E. coli* K12 and DSM 30083^{T} regarding the maximum-height values mapped on the Galactose pathway. Compare also Figure 7, which directly depicts the magnitude of the differences. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

power of the functions from the **multcomp** package underlying `opm_mcp` (see the main tutorial and the manual for details) is available when visualising PM data in pathway graphs.

The mapping of indicators of the significance of the differences between the two strains regarding the Galactose pathway is shown in Figure 9 and 10. Because we chose the "upwards" running mode, up to three colours are used in the map, indicating whether the performance difference is significantly larger than the minimum effect size, insignificantly different from the minimum effect size, or significantly smaller than the minimum effect size. In Figure 9, the default minimum effect size of zero is chosen. Figure 10 shows the results for a user-defined minimum effect size. This can be set using a short-cut notation indicating both the minimum effect size and the kind of comparison to be conducted. The manual lists all possible short-cuts. Remember the use of the `lmap` argument.

```
R> # 'upwards' comparison, default minimum effect size
R> cat.coli.comp.0 <- opm_path(
     cpd.data = annotated(coli.comp, output = "upwards", lmap = 1:3),
     species = "ko", out.suffix = "cat-coli-comp-0", pathway.id = "00052",
     limit = list(gene = 2, cpd = c(1, 3)), bins = list(gene = 0.5, cpd = 3))
R> # 'upwards' comparison, 150 units as minimum effect size
R> cat.coli.comp.150 <- opm_path(
     cpd.data = annotated(coli.comp, output = "'150", lmap = 1:3),
     species = "ko", out.suffix = "cat-coli-comp-150", pathway.id = "00052",
     limit = list(gene = 2, cpd = c(1, 3)), bins = list(gene = 0.5, cpd = 3))
```

4.2.4. Visualisation of pathway maps in Graphviz layout

In addition to the native KEGG visualisation, `pathview` can use the Graphviz library for an alternative visualisation approach. As return value, the function always generates a list containing two data frames. The data frame "plot.data.gene" contains the data for mapping the genes and, analogously, "plot.data.cpd" stores the compound-related data.

In the examples detailed above, a variety of such objects have already been generated, e.g., `coli.map.k12` or the `coli.map.types` (both described in Section 4.2.1 and visualised in Figure 2 and 3 therein).

Next, we show how to produce graphics from these objects instead of directly from KEGG.

```
R> coli.graphvizmap.k12 <- opm_path(cpd.data = anno.k12, species = "ko",
     afactor = 1500, kegg.native = FALSE, out.suffix = "graphvizk12.ko",
     pathway.id = "00052", sign.pos = "bottomleft", key.pos = "bottomright")
```

Although `pathview` offers quite a number of arguments specifically for `kegg.native=FALSE`, there are, unfortunately, only limited options for tuning the size of the nodes or the font sizes in this visualisation. The scaling of text and symbols can be tuned with the usual `cex` argument (for `kegg.native = FALSE` the default is `cex = 0.5`). The `text.width` argument can be used to specify the length for text wrapping. In principle, the node size can be fine-tuned with `afactor`, however even the help page of the `pathview` function emphasises that "its effect is subtler than expected". Together with the automated rendering by the `graphviz` layout engine it might be difficult to obtain a visually satisfying map of the pathway of interest.

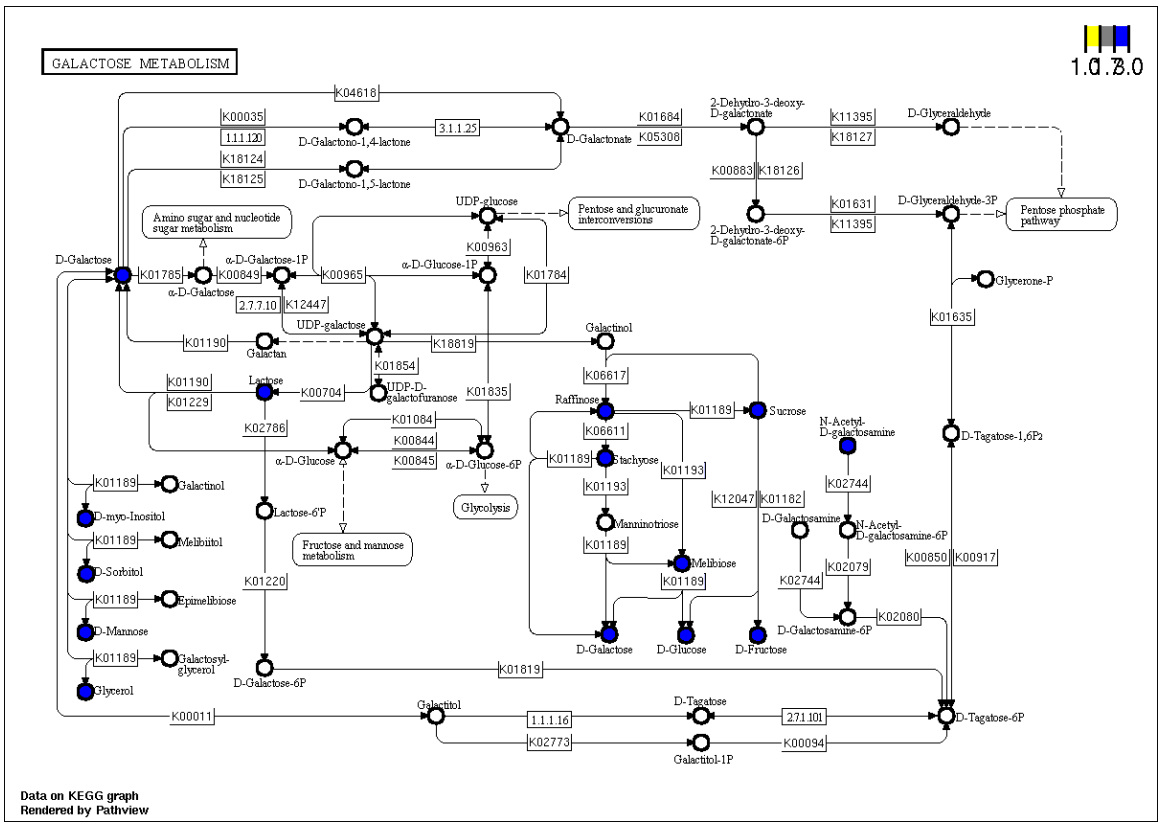

Figure 9: Indicators of the significance of the respiration differences between *E. coli* K12 and *E. coli* DSM 30083$^{\mathrm{T}}$ regarding the maximum-height values mapped on the Galactose pathway. Blue, significantly larger than the chosen minimum effect size of 0 (default); grey, insignificantly different from the chosen minimum effect size; yellow, significantly smaller than the chosen minimum effect size. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

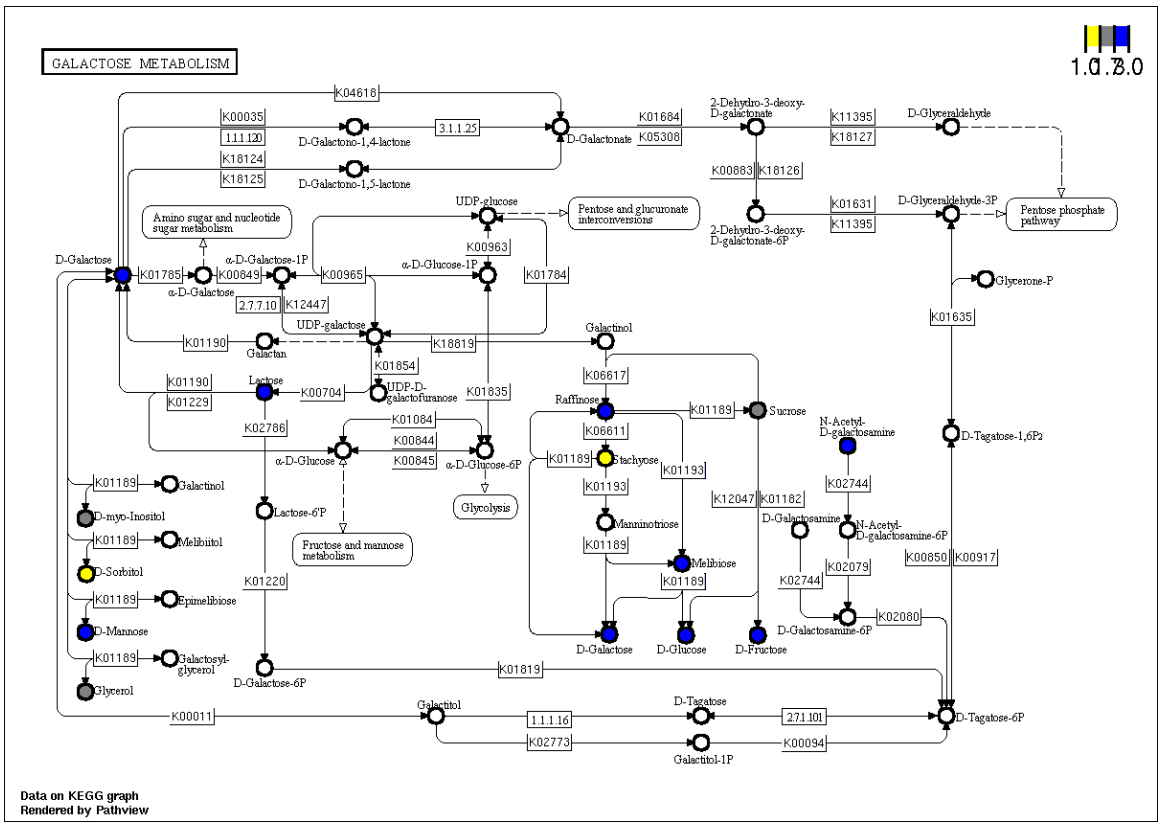

Figure 10: Indicators of the significance of the respiration differences between *E. coli* K12 and *E. coli* DSM 30083^{T} regarding the maximum-height values mapped on the Galactose pathway. Blue, significantly larger than the chosen minimum effect size of 150; grey, insignificantly different from the chosen minimum effect size; yellow, significantly smaller than the chosen minimum effect size. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

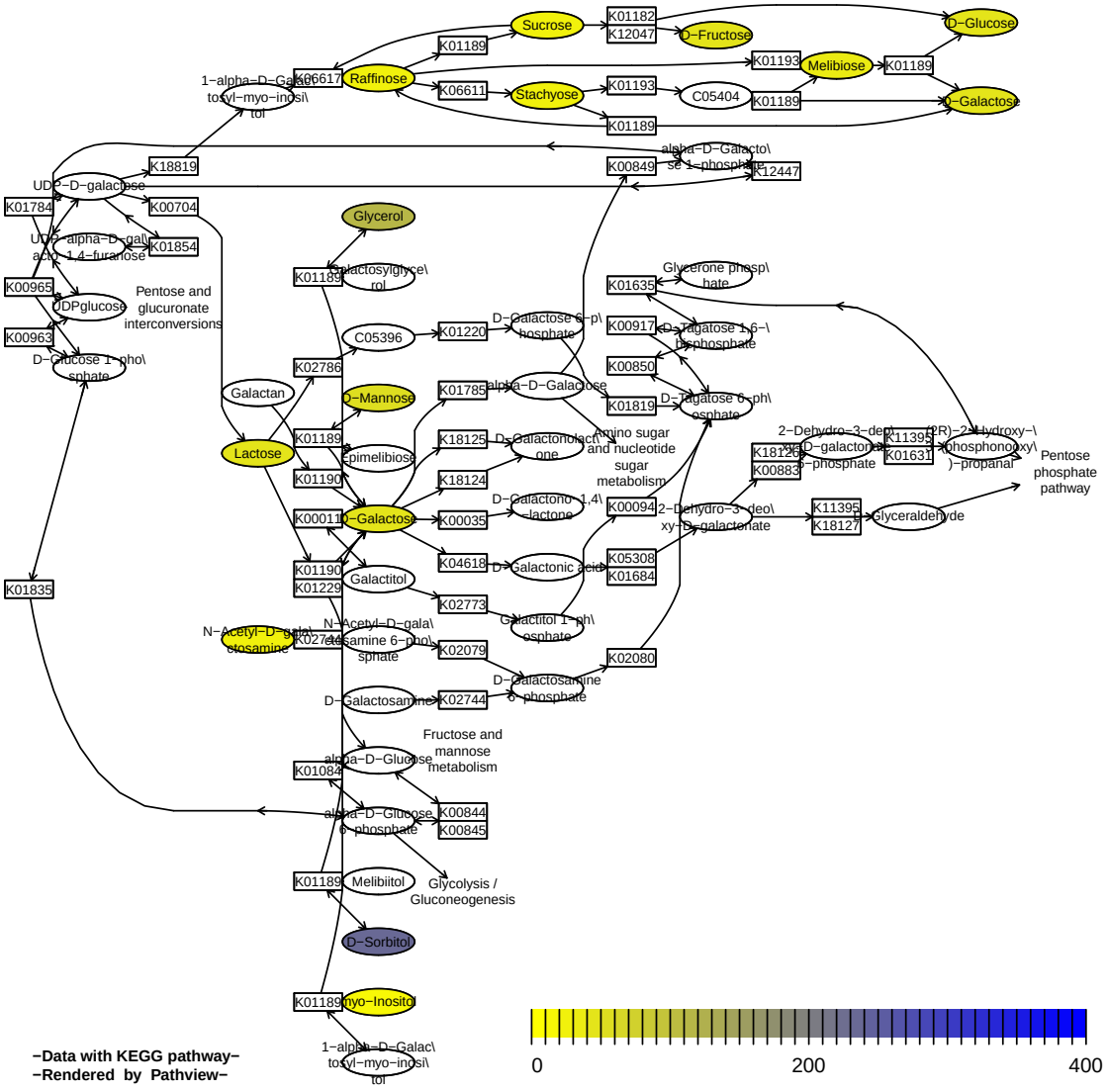

Figure 11: Respiratory data (mean maximum height) of *Escherichia coli* strain K12 mapped on the KEGG Galactose pathway and rendered by **pathview** using the Graphviz layout engine. Compare Figure 2, which shows the native KEGG rendering of these data. See the main text for a description of some of the limitations when creating graphics in this way. (If the resolution of this figure is insufficient, this is due to PDF compaction, not due to the **pathview** graphics.)

5. Finding the pathways of interest

In many experimental approaches, the identification of the pathway(s) of interest is one of the main subjects of research.

The information accessible *via* `annotated` can serve as input for suitable statistical procedures in order to assess and rank the observed metabolic capabilities (i.e. respiration on the substrates) with respect to their ability to predict a given response. Beside the identification of pathways of interest to be used when drawing the **opm** results in a graph, as detailed in the previous section, such results can further be used as a starting point when searching for a causal explanation of the given **opm** results.

5.1. Boosting as a tool to identify important substrates

In the following example we use boosting to estimate a linear model in order to determine important substrates that describe the differences between the two strains. Boosting (Bühlmann and Yu 2003; Hothorn, Bühlmann, Kneib, Schmid, and Hofner 2010) is a method to fit a statistical model that additionally selects the most important predictor variables. Optimal stopping of the algorithm can be done via cross-validation. For a detailed hands-on introduction to boosting methods using the package **mboost** (Hothorn, Bühlmann, Kneib, Schmid, and Hofner 2013) we refer to Hofner, Mayr, Robinzonov, and Schmid (2014).

At first, we need to load the package:

```
R> ## load package mboost
R> library("mboost")
R> ## set seed (for reproducible results)
R> set.seed(1907)
```

Now we need to extract and restructure the data using custom methods available in **opm** and a few further adaptations. We use all available biological and technical replicates and obtain the annotation from KEGG:

```
R> ## use multiple biological replicates:
R> coli.sub1 <- subset(vaas_et_al, list(Species = "Escherichia coli",
     Experiment = c("First replicate", "Second replicate")))
R> ## reformat and add substrate information
R> data <- annotated(opm_mcp(coli.sub1,
     ~ Strain + Experiment + Slot + `Plate number`,
     output = "data", in.parens = FALSE), how = "data.frame")
R> data <- data[complete.cases(data),
     names(data) %in% c("Plate.number", "Strain", "Experiment",
       "Slot", "Value") | grepl("^map", names(data))]
R> ## add an ID per biological / technical replicate for usage of
R> ## brandom (subject-specific random effect, see below):
R> data$ID <- extract_columns(data, c("Plate.number", "Strain", "Experiment"),
     direct = TRUE, factor = TRUE)
R> ## now the plate number is not needed any more
R> data$Plate.number <- NULL
R> ## have a look at a random subset of the resulting data set
R> data[sample(1:nrow(data), size = 5), 1:7]
```

```
        Strain       Experiment Slot     Value map00010 map00020 map00030
113   DSM30083T  First replicate    A 286.02512    FALSE    FALSE    FALSE
3044   DSM18039 Second replicate    B  38.97180    FALSE     TRUE    FALSE
802    DSM18039 Second replicate    B 275.82110    FALSE    FALSE    FALSE
2130   DSM18039 Second replicate    B  54.33588    FALSE    FALSE    FALSE
2157  DSM30083T  First replicate    A 213.73781    FALSE    FALSE    FALSE
```

Each row of the resulting data set describes the measurement of *one well, for one replicate, within one strain*. Starting from `map00010` we find the annotations that indicate whether or not this well can be mapped to the corresponding substrate class or pathway. Note that a well can potentially be found in multiple substrate classes or pathways.

In the next step, we want to model the differences of the measured values between the two strains. Therefore we set up a log-linear model of the following form:

$$\begin{aligned}\log(y) =& \beta_0 + \beta_1 \text{strain} + \beta_2 \text{experiment} + \beta_3 \text{slot} + b_{\text{ID}} + \\ &+ \beta_{4,1} I_{\text{PW1}} + \beta_{4,2} I_{\text{PW2}} + \ldots + \\ &+ X(\text{strain}) \cdot \beta_{5,1} I_{\text{PW1}} + X(\text{strain}) \cdot \beta_{5,2} I_{\text{PW2}} + \ldots,\end{aligned}$$

where y is the measured PM value, β_0 is an overall intercept, β_1 is the overall strain effect (the difference between strains irrespective of the pathway or substrate), and β_2 and β_3 are the effects of the experiment and slot. These are used to correct for possible confounding. Additionally, we use a random effect for the replicate (b_{ID}) to account for subject-specific effects. The pathway-specific effects $\beta_{4,j}$ represent the differences of the PM values between pathways and substrates, where $I_{\text{PW}j}$ is an indicator function, which is 0 if the well does not belong to the pathway and 1 if it does, i.e., $I_{\text{PW}j}$ corresponds to the annotations discussed above.

The most interesting part is given by $\beta_{5,j}$: $X(\text{strain})$ is a strain-specific function which is either -1 (strain DSM30083T) or 1 (strain DSM18039). The coefficients $\beta_{5,j}$ hence represent the deviation of the strains from the global effect. If $\beta_{5,j} = 0$, no strain-specific effect is present, i.e., the pathway j does not differ between strains. If $\beta_{5,j} \neq 0$, the difference between the two strains is twice this effect, i.e., $X(\text{strain}_1) \cdot \beta_{5,j} - (X(\text{strain}_2) \cdot \beta_{5,j}) = 1 \cdot \beta_{5,j} - (-1 \cdot \beta_{5,j}) = 2\beta_{5,j}$.

Now we can set up and estimate the model. Be careful as this takes a while.

```
R> ## re-define linear base-learner to have 2 degrees of freedom
R> bols2 <- function(a, ...)
     bols(a, df = 2, ...)
R> ## fit offset model with main effects only (and use very many iterations):
R> options(contrasts = c("contr.treatment", "contr.poly"))
R> offsetmod <- gamboost(log(Value) ~ ., data = data, baselearner = bols2,
     control = boost_control(mstop = 6000, nu = 0.2))
R> ## now start from the offset model and add interaction effects (which
R> ## represent differences between strains).
R> options(contrasts = c("contr.sum", "contr.poly"))
R> ## set up model formula
R> fm <- as.formula(paste(c("log(Value) ~ bols(Strain, intercept = FALSE, df = 1)",
       ## add variables as main effects
       sprintf("bols(%s, intercept = FALSE, df = 1)", names(data)[-c(1, 4)]),
       ## add the interaction terms (i.e. strain-specific effects)
       sprintf("bols(%s, by = Strain, intercept = FALSE, df = 1)",
```

```
        names(data)[-c(1, 4)])),
    collapse = " + "))
R> ## estimate model
R> mod <- gamboost(fm, data = data, offset = fitted(offsetmod),
    control = boost_control(mstop = 250, nu = 0.2))
R> options(contrasts = c("contr.treatment", "contr.poly"))
```

The offset model fits all main effects, i.e. differences in the maximum curve height with respect to different substrate groups or pathway maps while neglecting possible differences in substrate/map effects between strains. Additionally, we specify effects for the strain, the experiment and the slot to account for overall differences between strains, experiments and slots. To account for repeated measurements we include a random effects term in the model. (Currently we use *one* random effect for both technical and biological replicates; this is appropriate as we currently do not really want to differentiate between the two types of variability. However, more complex random effect terms might be used as well.)

In a second step, we start from the offset model and we allow for interactions between strain and the substrate/map effects and check if any are present. These interactions represent differential PM expressions between strains.

Boosting is an iterative procedure where the major tuning parameter is the number of boosting steps (`mstop`). To find the optimal model, we need to use cross-validation techniques (Hofner *et al.* 2014; Mayr, Hofner, and Schmid 2012). This can be easily done using:

```
R> ## to speed up computation on linux systems set number of
R> ## cores in cvrisk: e.g. mc.cores = 25
R> cvr <- cvrisk(mod, grid = 1:2000)
```

Note that this might take some time depending on your system.

Now, we subset the model to the optimal number of stopping iterations:

```
R> ## optimal number of iterations
R> mstop(cvr)
R> ## plot the cross-validation results
R> plot(cvr)
R> ## finally set model iterations to this number
R> mstop(mod) <- mstop(cvr)
```

If one uses cross-validation techniques to find the optimal stopping iteration, boosting intrinsically selects variables that best predict the outcome, i.e., here the log(PM) values that differ between the two *E. coli* strains with respect to certain substrates. However, cross-validation tends to select too many variables to enter the model (here: 13.3% of the base-learners where selected to enter the model; this makes 37 out of 279). Thus, to refine the selection process, we use stability selection to determine which of the (interaction) effects are selected (while controlling the per-family error rate).

5.2. Boosting with stability selection

Stability selection (Meinshausen and Bühlmann 2010) is an additional screening tool that allows the user to determine which of the selected variables are "stable" variables, i.e. which

are variables that always "pop-up" in random subsets of the data. Additionally, stability selection has a built-in error control (for the per-family error rate, see Dudoit, Shaffer, and Boldrick 2003). The application of boosting with stability selection to PM data has been explored in (Hofner, Boccuto, and Göker 2015).

In our example the maximum acceptable error is set to 1.5, i.e. we accept at maximum 1.5 noise variables to be chosen by stability selection by chance. The cut-off value determines which inclusion frequency is at least required to term a variable stable. In our example this is chosen to be 0.6, but other values in, say $(0.6, 0.9)$, are possible as well. In general, stability selection is relatively insensitive to the choice of the cut-off.

```
R> ## use stability selection to extract important pathways / substrate classes
R> ## to speed up computation on linux systems set number of
R> ## cores in stabsel: e.g. mc.cores = 25
R> stab <- stabsel(mod, cutoff = 0.75, PFER = 1.5, assumption = "unimodal")
```

Plotting the results of stability selection shows which of the pathways and substrate groups are differentially expressed between the two strains (see Figure 12). The stability paths (Figure 12, left) show the inclusion frequencies per variable dependent on the number of boosting iterations. This graphical display mainly helps to verify whether or not the number of boosting iterations was sufficient. In this case, the selection paths stay constant after a certain point, indicating a sufficiently large number of boosting iterations. As an alternative to the stability paths we can plot the maximum inclusion frequency (Figure 12, right). Higher inclusion frequencies indicate that the substrate is more important. Furthermore, we see that only the top 15 variables are stable and thus can be considered to play a substantial role. This is a much sparser model than the one obtained from cross-validation.

```
R> ## extract variable names
R> nms <- gsub(", ", " by ", variable.names(mod))
R> ## adapt plotting settings
R> old.par <- par(mar = c(4, 0.1, 0.1, 11.5), mfrow = c(1, 2))
R> plot(stab, type = "paths", main = "", labels = nms)
R> par(mar = c(4, 11.5, 0.1, 0.1))
R> plot(stab, type = "maxsel", main = "", np = 20, labels = nms)
R> par(old.par) # reset plotting settings
```

In the next step we extract the coefficient estimates of the stable variables. We set the stopping iteration to the optimal iteration obtained by cross-validation and extract only the coefficients of the stable variables:

```
R> mstop(mod) <- mstop(cvr)
R> cf <- coef(mod, which = selected(stab))
R> ## make cf a vector
R> beta <- unlist(cf)
R> old.par <- par(mar = c(4, 15, 0.1, 0.1))# adapt plotting settings
R> ## plot 2 * beta as this is the difference between the two strains
R> plot(2 * sort(beta), 1:length(beta),
     type = "n", yaxt = "n",
     ylab = "", xlab = expression(paste("Difference between strains (",
       2 * hat(beta), ")")))
```

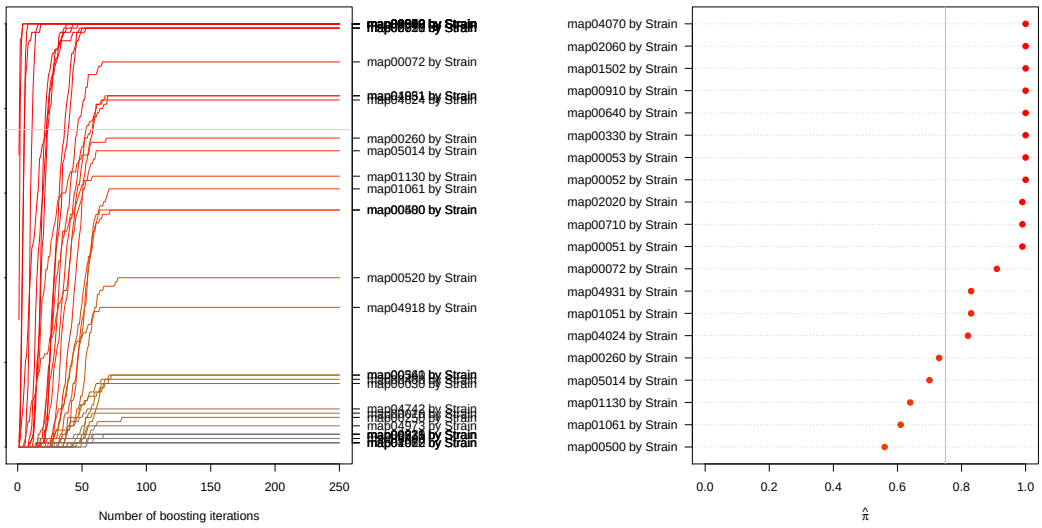

Figure 12: Selected pathways and substrate classes. From the stability selection paths plot one can deduce that the number of iterations was sufficiently large as all selection paths stop to increase after approx. 100 iterations (left). All variables that cross the threshold value (horizontal grey line; $\hat{\pi}_{\text{threshold}} = 0.75$) at some point are termed "stable", i.e., are chosen by stability selection. The maximum inclusion frequency $\hat{\pi}$ for the top 20 variables as determined by stability selection is displayed in the right figure. All variables with inclusion frequencies $\hat{\pi} \geq \hat{\pi}_{\text{threshold}}$, i.e., right of the grey vertical line are stable variables.

```
R> abline(h = 1:length(beta), lty = "dotted", col = "grey")
R> points(2 * sort(beta), 1:length(beta), pch = 19)
R> nms <- gsub(", ", " by ", variable.names(mod, which = selected(stab)))
R> axis(2, at = 1:length(beta), nms[order(beta)], las = 2)
R> abline(v = 0, col = "grey")
R> par(old.par) # reset plotting settings
```

From Figure 13, we can see that `map04070` explains the largest difference between the two strains (as the absolute value is biggest) followed by `Phenylpropanoids`, `map00072` and `map00053`. While the first three variables have a negative sign, indicating that DSM30083T is increased compared to strain DSM18039, `map00053` has a positive sign, indicating an increase in strain DSM18039.

5.2.1. Evaluation of results

In the next step we will augment the boosting approach with an `opm_mcp` analysis for confirming whether the reactions on carbohydrate wells are indeed more strongly different between the two strains than those on other wells.

First, we prepare a data matrix with the differences between the two *E. coli* strains (as computed in section 4.2.3) and add potential explanatory variables from KEGG. Please refer to the manual and Section 4.1 for an explanation of the usage of `annotated` with the argument `how = "data.frame"`. Again, the comparison applies to the default parameter given by `opm_opt("curve.param")`; see the manual for details.

```
R> coli.comp <- opm_mcp(coli.sub, output = "mcp",
```

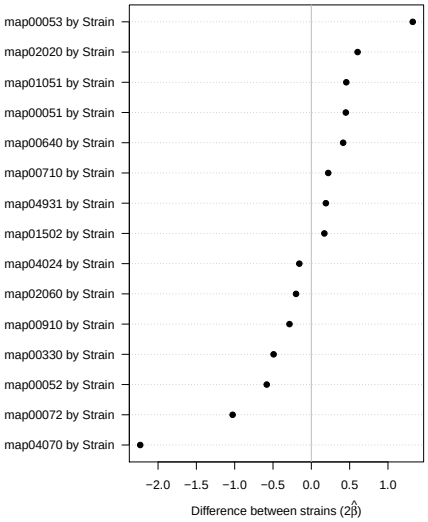

Figure 13: Coefficients of stable variables (in the optimal boosting iteration). Positive signs indicate an increase in strain DSM18039 compared to DSM30083T, and negative signs vice versa.

```
     model = ~ J(Well + Strain), linfct = c(Pairs = 1))
R> coli.anno <- annotated(coli.comp, how = "data.frame")
R> coli.anno <- coli.anno[complete.cases(coli.anno), ]
```

Now, the positions of the substrates either belonging to the category "Carbohydrates" or not to it are determined and afterwards stored in two vectors.

```
R> carb.pos <- rownames(coli.anno)[coli.anno$Carbohydrates == TRUE]
R> noncarb.pos <- rownames(coli.anno)[coli.anno$Carbohydrates == FALSE]
R> carb.pos <- find_positions(carb.pos, plate_type(coli.sub))
R> noncarb.pos <- find_positions(noncarb.pos, plate_type(coli.sub))
```

We can now generate the contrast matrix needed for the computation of the multiple comparisons between the two strains for precisely these two subsets of substrates. As recommended in the main tutorial, this should be done by first using `opm_mcp` to generate auxiliary contrast matrices, which here provide the dimensions and column names for the final contrast matrix.

```
R> contr.carb <- opm_mcp(coli.sub[, , carb.pos], linfct = c(Dunnett = 1),
     model = ~ J(Well + Strain), output = "contrast")[[1]]
R> contr.noncarb <- opm_mcp(coli.sub[, , noncarb.pos], linfct = c(Dunnett = 1),
     model = ~ J(Well + Strain), output = "contrast")[[1]]
```

The final contrast matrix is set up by concatenating the necessary numeric vectors, setting labels for the comparisons and adding the column names.

```
R> contr <- rbind(
     "Carbohydrates/DSM18039 - Carbohydrates/DSM30083T" = c(
       rep(c(2 / ncol(contr.carb), -2 / ncol(contr.carb)),
```

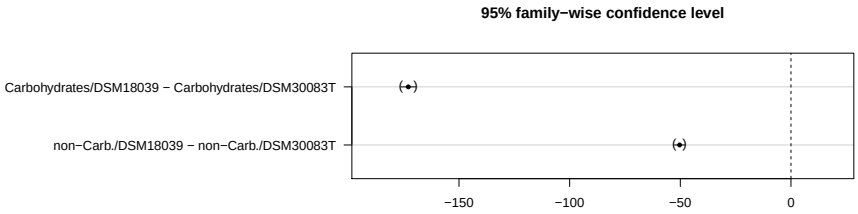

Figure 14: Point estimates and 95% confidence intervals for the comparison of the two *E. coli* strains pooled over substrates classified as "Carbohydrates" and "non-Carbohydrates". The difference between the two strains is much larger regarding the carbohydrate substrates than regarding the remaining ones.

```
        times = ncol(contr.carb) / 2),
      rep(0, times = ncol(contr.noncarb))
    ),
    "non-Carb./DSM18039 - non-Carb./DSM30083T" = c(
      rep(0, times = ncol(contr.carb)),
      rep(c(2 / ncol(contr.noncarb), -2 / ncol(contr.noncarb)),
        times = ncol(contr.noncarb) / 2)
    )
  )
R> colnames(contr) <- c(colnames(contr.carb), colnames(contr.noncarb))
```

This contrast matrix `contr` can now directly be fed into `opm_mcp`.

```
R> carb.test <- opm_mcp(coli.sub[, , c(carb.pos, noncarb.pos)], linfct = contr,
     m.type = "lm", model = ~ J(Well + Strain))
```

The results of the two comparisons can be plotted as usual, see Figure 14. According to the interpretation provided in the figure caption the analysis shows that one of the strains performs stronger than the other one particularly regarding carbohydrate utilisation.

As an alternative to boosting, one could also apply Random Forest (RF) for variable selection, e.g. implemented in the **randomForest** package (Liaw and Wiener 2002). See Touw, Bayjanov, Overmars, Backus, Boekhorst, Wels, and van Hijum (2012) for a review of RF properties that allow for maximising the biological insights that can be extracted from complex OMICS data. Boulesteix, Janitza, Kruppa, and König (2012) emphasise the applications of RF to computational biology with special attention to practical aspects such as parameter selection and major pitfalls and biases of RF and its Variable Importance Measure (VIM).

6. Acknowledgements

We are grateful to Barry Bochner (BIOLOG Inc.) for providing substrate and plate information on PM assays. The integration of missing OmniLog® substrates into ChEBI by the ChEBI staff is gratefully acknowledged. We thank Weijun Luo for providing hints regarding the use of the **pathview** package.

References

Boulesteix AL, Janitza S, Kruppa J, König IR (2012). "Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics." *WIREs Data Mining Knowl Discov*, **2**, 493–507. doi:10.1002/widm.1072.

Bühlmann P, Yu B (2003). "Boosting with the L_2 Loss: Regression and Classification." *Journal of the American Statistical Association*, **98**, 324–339. doi:10.1198/016214503000125.

Caspi R, Altman T, Dreher K, Fulcher CA, Subhraveti P, Keseler IM, Kothari A, Krummenacker M, Latendresse M, Mueller LA, Ong Q, Paley S, Pujar A, Shearer AG, Travers M, Weerasinghe D, Zhang P, Karp PD (2012). "The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases." *Nucleic Acids Research*, **40**, D742–D753. doi:10.1093/nar/gkr1014.

Coletti MH, Bleich HL (2001). "Medical Subject Headings Used to Search the Biomedical Literature." *Journal of the American Medical Informatics Association*, **8**, 317–323. doi:10.1136/jamia.2001.0080317.

Dudoit S, Shaffer JP, Boldrick JC (2003). "Multiple Hypothesis Testing in Microarray Experiments." *Statistical Science*, **18**, 71–103.

Hastings J, de Matos P, Dekker A, Ennis M, Harsha B, Kale N, Muthukrishnan V, Owen G, Turner S, Williams M, Steinbeck C (2013). "The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013." *Nucleic Acids Research*, **41**, D456–D463. doi:10.1093/nar/gks1146.

Hofner B, Boccuto L, Göker M (2015). "Controlling false discoveries in high-dimensional situations: Boosting with stability selection." *BMC Bioinformatics*, **16**, 144. doi:10.1186/s12859-015-0575-3.

Hofner B, Mayr A, Robinzonov N, Schmid M (2014). "Model-based Boosting in R – A Hands-on Tutorial Using the R Package mboost." *Computational Statistics*, **29**, 3–35. doi:10.1007/s00180-012-0382-5.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2010). "Model-Based Boosting 2.0." *Journal of Machine Learning Research*, **11**, 2109–2113.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B (2013). *Model-Based Boosting*. R package version 2.2-3, URL http://CRAN.R-project.org/package=mboost.

Kanehisa M, Goto S, Furumichi M, Tanabe M, Hirakawa M (2010). "KEGG for representation and analysis of molecular networks involving diseases and drugs." *Nucleic Acids Research*, **38**, D355–D360. doi:10.1093/nar/gkp896.

Liaw A, Wiener M (2002). "Classification and Regression by randomForest." *R News*, **2**, 18–22. URL http://CRAN.R-project.org/doc/Rnews/.

Luo W, Brouwer C (2013). "Pathview: an R/Bioconductor package for pathway-based data integration and visualization." *Bioinformatics*, **29**, 1830–1831. doi:10.1093/bioinformatics/btt285.

Mayr A, Hofner B, Schmid M (2012). "The importance of knowing when to stop – a sequential stopping rule for component-wise gradient boosting." *Methods of Information in Medicine*, **51**, 178–186. doi:10.3414/ME11-02-0030.

Meinshausen N, Bühlmann P (2010). "Stability Selection." *Journal of the Royal Statistical Society. Series B*, **72**, 417–473. doi:10.1111/j.1467-9868.2010.00740.x.

Overbeek R, Begley T, Butler RM, Choudhuri JV, Chuang HY, Cohoon M, de Crecy-Lagard V, Diaz N, Disz T, Edwards R, Fonstein M, Frank ED, Gerdes S, Glass EM, Goesmann A, Hanson A, Iwata-Reuyl D, Jensen R, Jamshidi N, Krause L, Kubal M, Larsen N, Linke B, McHardy AC, Meyer F, Neuweger H, Olsen G, Olson R, Osterman A, Portnoy V, Pusch GD, Rodionov DA, Rueckert C, Steiner J, Stevens R, Thiele I, Vassieva O, Ye Y, Zagnitko O, Vonstein V (2005). "The Subsystems Approach to Genome Annotation and its Use in the Project to Annotate 1000 Genomes." *Nucleic Acids Research*, **33**, 5691–5702. doi:10.1093/nar/gki866.

Touw WG, Bayjanov JR, Overmars L, Backus L, Boekhorst J, Wels M, van Hijum SAFT (2012). "Data mining in the Life Sciences with Random Forest: A walk in the park or lost in the jungle?" *Briefings in Bioinformatics*, **14**, 315–326. doi:10.1093/bib/bbs034.

Affiliation:

Markus Göker
Leibniz Institute DSMZ – German Collection of Microorganisms and Cell Cultures
Braunschweig

Telephone: +49/531-2616-272
Fax: +49/531-2616-237
E-mail: markus.goeker@dsmz.de
URL: www.dsmz.de