Working with substrate information in opm

Lea A.I. Vaas
Leibniz Institute DSMZ

Johannes Sikorski
Leibniz Institute DSMZ

Markus Göker
Leibniz Institute DSMZ

Abstract

This is the substrate-information tutorial of **opm** in the version of October 4, 2013.

Keywords: Cell Lines, Metadata, Microbiology, Respiration Kinetics, Pathway, KEGG, **pathview**.

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 OmniLog® Phenotype MicroArray 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 concept and structure of the different object-classes should be made clear before starting with this tutorial.

In addition to visual inspection or statistical comparative analyses of Phenotype Microarray data, as described in the main manual, 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 these information directly, this tutorial introduces strategies for visualization of the measured PM results by mapping on pathway-maps and introduces analysis methods for modelling and identification of informative and less informative substrates.

2. Preparation

Before starting, install the package **pathview** from Bioconductor (`http://bioconductor.org/packages/2.12/bioc/html/pathview.html`) and load it together with the **opm** package, into an R session. Please note, that it is important to load **pathview** before **opm**, since otherwise some functions are not visible and the package does not work properly. In this vignette this is catched by the `if`-construct.

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

3. Available plate information

Currently substrate layouts of various plates are available within **opm**. An overview about 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.

4. Accessing substrate information

The **opm** package contains a number of functions suitable for accessing precomputed information on the substrates within certain wells and entire plates. In the manual and help pages these functions are explained within 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 results 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 *substrings* 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 wildcards, 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; see the manual for details, entry `?find_substrate`.

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 wildcard 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. 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 CAS numbers (http://www.cas.org/content/chemical-substances/faqs), but MeSH names (useful for conducting PubMed queries; see http://www.ncbi.nlm.nih.gov/mesh/) (Coletti and Bleich 2001), 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) and MetaCyc IDs (Caspi, Altman, Dreher, Fulcher, Subhraveti, Keseler, Kothari, Krummenacker, Latendresse, Mueller, Ong, Paley, Pujar, Shearer, Travers, Weerasinghe, Zhang, and Karp 2012) IDs 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** is available and converted into customized 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. See the manual for further details, help page `?substrate_info`.

5. Visualisation by integration in pathway maps

In conjunction with other packages, it is possible to visualize PM results directly in pre-existing pathway maps as, for example, from the KEGG database. Those maps are essentially manually drawn pathway maps representing the currently available knowledge about genes, substrates and their connection in pathways. Depending on availability of genome and gene-annotation information within KEGG about a certain organism, individual maps are allocateable (Kanehisa *et al.* 2010).

The mapping itself is simply the introduction of PM-measurement data as colour-coding of nodes (here, representing the substrates) into those maps, as it can be done similarly with several other types of OMICS-data. For details, please see the description on the KEGG-homepage: (http://www.genome.jp/kegg/).

Here we will use the function `pathview` from the package of same name (Luo and Brouwer 2013). This function downloads the user-defined pathway map and subsequently maps and renders the given PM-data into it. Please note, that it has to be loaded *before* loading package **opm** into an R session.

5.1. suitable input-data

The workflow starts with either an `OPMX` object containing the agregated values, or the result from an `opm_mcp` analysis. The first step in both cases is to bring the PM-results in a suitable format, which is a named vector created by `annotated`.

```
R> # here provide the annotated vector from vaas_1
R> xx <- annotated(vaas_1)
R> head(xx)
```

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

The resulting vector basically contains the numeric values (selected parameter estimates or `opm_mcp` results, as explained below) as well as an annotation of the according substrates. With the `what`-argument, passed as eponymous argument to `substrate_info`, the user can indicate which kind of information should be used for the annotation. `annotated` works directly on `OPMX`-objects containing aggregated data for single plates or bundles of plates. However, please note, that the output allows for only one value per substrate. When applying `annotated` to a set of plates, make sure, that only one experimental group is comprised, since the resulting values are averages per well over all plates. Using the `output`-argument, one is able to choose which parameter should be adressed, for example AUC instead of maximum height:

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

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

Further options allow for indication of categorical (`different`) and for the `opm_glht` method, binary outcomes (`smaller`, `larger` or `equal`).

But visualization 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. The results from an `opm_mcp` procedure are treated with `annotated` as shown before with an `OPMX` object:

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

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 contrast if applied to the wells.

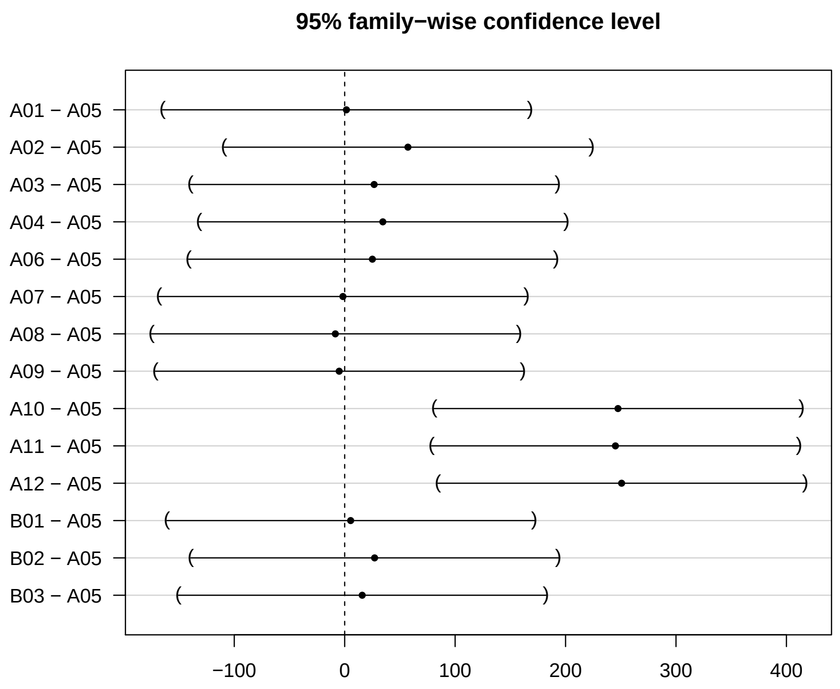

Figure 1: Point estimates and 95% confidence intervals in a manually defined comparison of group means for a specifically selected set of wells (A01 to B03) from the `vaas_4` exemplar object. In this procedure each well is compared against A05. The picture was obtained by running `opm_mcp()` and then the plotting function for the resulting `opm_glht`-object. See the main tutorial for details.

With this example, the options allowing for categorical output of `annotated` will be illustrated. Obviously, only three comparisons exhibit a statistically significant difference, namely the comparisons A10 - A05, A11 - A05 and A12 - A05, all in the way that A05 is smaller than A10, A11 and A12, respectively.

Using the `output`-argument users are able to get various categorical results beside simple numerical output of the respective point estimator. The options `upwards` and `downwards` result classifications in three categories (FALSE, NA, or TRUE), giving an indication, if the CI is located in the positive or negative range with resulting NA, if zero is included and thus no decission possible. The options `different`, `smaller`, `larger` and `equal` work similarly, but use only the two categories TRUE and FALSE.

```
R> anno.nu <- annotated(x, output = "numeric")
R> anno.up <- annotated(x, output = "upwards")
R> anno.do <- annotated(x, output = "downwards")
R> anno.di <- annotated(x, output = "different")
R> anno.eq <- annotated(x, output = "equal")
R> anno.sm <- annotated(x, output = "smaller")
R> anno.la <- annotated(x, output = "larger")
```

A comprehensive overview of the results with the above stated example is facilitated by bundling the results of all possible options into one dataframe:

```
R> annotab <- data.frame(Comparison = rownames(confint(x)$confint),
     numeric = anno.nu, upwards = anno.up, downward = anno.do,
     different = anno.di, equal = anno.eq, smaller = anno.sm, larger = anno.la)
R> annotab
```

```
   Comparison    numeric upwards downward 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
```

Using the argument `how = "value"` a numeric matrix containing the chosen computed values together with their abundance in pathway maps can be investigated.

5.2. Visualisation in pathway-maps using pathview

Basically, `pathview` maps and renders data provided by `annotated` on pathway graphs. The user has to specify the pathway, provide the PM-data and all necessary steps (download of pathway graph data, parsing of data file, maping user's data to the pathway, and rendering of pathway graph with the mapped data) are executed automatically by `pathview`.

Before you start with this section, please make sure, that you have installed the **pathview** package including all its dependencies from Bioconductor (`http://bioconductor.org/packages/2.12/bioc/html/pathview.html`).

To provide more convenient functionality for mapping PM-data on KEGG-pathway maps, we show a wrapper for the `pathview`-function. All graphics below are produced using this wrapper, however, the user is free to use the original `pathview`-function with the respectively changed arguments.

```
R> opm_path <- function(cpd.data, high = list(gene = "green4", cpd = "magenta4"),
     low = list(gene = "white", cpd = "white"),
     mid = list(gene = "lightsteelblue1", cpd = "lightsteelblue1"),
     pathway.id = "00052", species = "ko",  out.suffix = "non-native",
     kegg.native = F, key.pos = "bottomright", afactor = 1000,
     limit = list(gene = 2, cpd = 400), bins = list(gene = 0.5, cpd = 20),
     both.dirs = list(gene = FALSE, cpd = FALSE), cex = 0.7,
     sign.pos = "bottomleft", cpd.lab.offset = 0, same.layer = F, rank.dir = "TB",
     pdf.size = c(10,7), gene.data = NULL, ...) {

     pathview(gene.data = gene.data, cpd.data = cpd.data, na.col ="grey",
              low = low,
              mid = mid,
              high = high,
              pathway.id = pathway.id,
              species = species,
              out.suffix = out.suffix,
              kegg.native = kegg.native,
              key.pos = key.pos,
              afactor = afactor,
              limit = limit,
              bins = bins,
              both.dirs = both.dirs,
              cex = cex, sign.pos = sign.pos, cpd.lab.offset = cpd.lab.offset,
              same.layer = same.layer,
              pdf.size = pdf.size,
              rank.dir = rank.dir,
              ...)
   }
```

ko - KEGG orthology: manually defined ortholog groups (KO entries) for all proteins and functional RNAs that correspond to KEGG pathway nodes, BRITE hierarchy nodes, and KEGG module nodes.

```
R> xx <- annotated(vaas_1)
R> test.pdf <- opm_path(cpd.data = xx)
R> str(test.pdf)
```

```
List of 2
 $ plot.data.gene:'data.frame':        44 obs. of  9 variables:
```

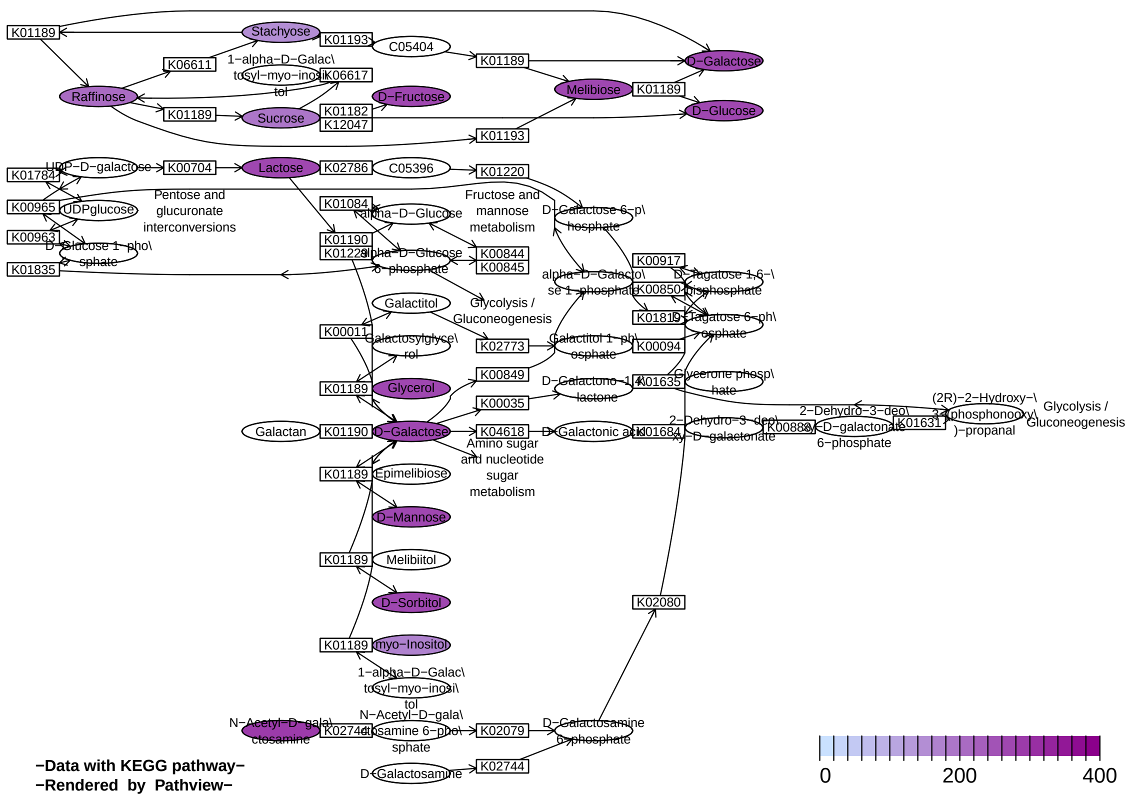

Figure 2: Map of galactose metabolisms in the native KEGG representation available in xxx.

```
  ..$ kegg.names: chr [1:44] "K00094" "K02773" "K01189" "K00917" ...
  ..$ labels    : chr [1:44] "K00094" "K02773" "K01189" "K00917" ...
  ..$ type      : chr [1:44] "ortholog" "ortholog" "ortholog" "ortholog" ...
  ..$ x         : num [1:44] 848 582 603 1027 977 ...
  ..$ y         : num [1:44] 755 754 458 561 561 310 153 586 548 487 ...
  ..$ width     : num [1:44] 46 46 46 46 46 46 46 46 46 46 ...
  ..$ height    : num [1:44] 17 17 17 17 17 17 17 17 17 17 ...
  ..$ kegg.names: num [1:44] NA NA NA NA NA NA NA NA NA NA ...
  ..$ mol.col   : Factor w/ 1 level "#BEBEBE": 1 1 1 1 1 1 1 1 1 1 ...
 $ plot.data.cpd :'data.frame':        43 obs. of  9 variables:
  ..$ kegg.names: chr [1:43] "C06311" "C01216" "C00137" "C00095" ...
  ..$ labels    : chr [1:43] "Galactitol 1-phosphate" "2-Dehydro-3-deoxy-D-galactonate" "myo-Inosit
  ..$ type      : chr [1:43] "compound" "compound" "compound" "compound" ...
  ..$ x         : num [1:43] 741 682 98 800 379 169 384 518 518 519 ...
  ..$ y         : num [1:43] 754 152 520 642 152 266 330 266 205 152 ...
  ..$ width     : num [1:43] 8 8 8 8 8 8 8 8 8 8 ...
  ..$ height    : num [1:43] 8 8 8 8 8 8 8 8 8 8 ...
  ..$ mol.data  : num [1:43] NA NA 178 271 NA ...
  ..$ mol.col   : Factor w/ 7 levels "#9C3BAA","#9F47B0",..: 7 7 5 2 7 2 7 7 7 7 ...
```

6. Model-building approach using random forest

Using the argument `how= "value"` a numeric matrix containing the chosen computed values together with their abundance in pathway maps can be investigated. In analogy to the download argument of `substrate_info` the information are convertet into a numeric matrix. Please note, that this option is not available for all values of `what` and requires additional libraries. See `substrate_info` for details.

```
R> anno.glht.mat <- annotated(x, how = "values")
R> anno.glht.mat
```

```
                 Value map00052 map00500 map01100 map02010 map02060 map04973
Negative Control   1.6       NA       NA       NA       NA       NA       NA
Dextrin           57.3        0        1        1        0        0        0
D-Maltose         26.7        0        1        1        1        1        1
D-Trehalose       34.5        0        1        1        1        1        0
b-Gentiobiose     25.1        0        0        0        0        0        0
Sucrose           -1.6        1        1        1        1        1        1
Turanose          -8.5        0        0        0        0        0        0
Stachyose         -5.0        1        0        0        0        0        0
Positive Control 247.5       NA       NA       NA       NA       NA       NA
pH 6             245.2       NA       NA       NA       NA       NA       NA
pH 5             250.8       NA       NA       NA       NA       NA       NA
D-Raffinose        5.4        1        0        0        0        0        0
a-D-Lactose       27.1        1        0        1        1        1        1
D-Melibiose       15.9        1        0        0        0        0        0
                 Carbohydrates Disaccharides Oligosaccharides exact_mass
Negative Control            NA            NA               NA         NA
Dextrin                      0             0                0         NA
D-Maltose                    1             1                1        342
D-Trehalose                  1             1                1        342
b-Gentiobiose                1             1                1        342
Sucrose                      1             1                1        342
Turanose                     1             1                1        342
Stachyose                    0             0                0        666
Positive Control            NA            NA               NA         NA
pH 6                        NA            NA               NA         NA
pH 5                        NA            NA               NA         NA
D-Raffinose                  0             0                0        504
a-D-Lactose                  1             1                1        342
D-Melibiose                  1             1                1        342
```

The obtained information about the pathway certain substrates are involved in can be used to determine manually which pathways are actually of interest or can serve as input for model-building procedures as, for example, random forest approaches, which is demonstrated below.

7. Acknowledgements

The integration of missing OmniLog® substrates into ChEBI by the ChEBI staff is gratefully acknowledged.

References

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**(D1), 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**(4), 317–323. doi:10.1136/jamia.2001.0080317.

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**(D1), D456–D463. doi:10.1093/nar/gks1146.

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**(suppl 1), D355–D360. doi:10.1093/nar/gkp896.

Luo W, Brouwer C (2013). "Pathview: an R/Biocondutor package for pathway-based data integration and visualization." *Bioinformatics*. doi:10.1093/bioinformatics/btt285. URL http://bioinformatics.oxfordjournals.org/content/29/14/1830.full.pdf+html.

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