PathWave v2.1 – Usage example

March 22, 2014

[Important note: this is a quick guide; see the full manual for further details]

This document describes a step-by-step analysis of breast cancer gene expression data as a practical example and quick guide for the usage of the PathWave R package, version 2.1.

For installation and a detailed description of the single commands, please see the PathWave manual.

PathWave authors:
Gunnar Schramm, Rosario M. Piro, Stefan Wiesberg

Availability:
http://www.ichip.de/software/pathwave.html

Document author:
Rosario M. Piro
DKFZ, Heidelberg, Germany

1. Gene expression data

The gene expression dataset used for demonstration purpose is provided by the Bioconductor package 'breastCancerNKI'.
(see http://www.bioconductor.org/packages/release/data/experiment/html/breastCancerNKI.html for details)

Install the package using the following commands in R:

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

Then load Biobase and the expression data package and make sure the data is accessible as an eSet object (named 'nki'):

```
library(Biobase)
library(breastCancerNKI)
data(nki)
```

We recommend to explore the nki dataset and its structure in order to better understand the following steps (see the breastCancerNKI manual for more information):

```
experimentData(nki)
abstract(nki)
head(fData(nki))
head(pData(nki))
exprs(nki)[1:8,1:8]
```

Since PathWave requires the expression data to be passed as a data.frame object and a mapping from samples to sample classes, we need to preprocess the data from the eSet:

First, we have to determine which probes are associated with an Entrez gene ID (required by PathWave if using pathways from KEGG or BiGG) and extract this subset of expression profiles:

```
probesWithEntrezID = !is.na(fData(nki)[,"EntrezGene.ID"])
nkiExprsEntrez = as.data.frame(exprs(nki)[probesWithEntrezID,])
```

Now we add an initial column containing the probe name and replace it by the corresponding Entrez gene ID:

```
nkiExprsEntrez = cbind(rownames(nkiExprsEntrez),nkiExprsEntrez)
nkiExprsEntrez[,1] = fData(nki)[probesWithEntrezID,"EntrezGene.ID"]
colnames(nkiExprsEntrez)[1] = "EntrezGene.ID"
```

(Note: the probes/rows in fData(nki) and exprs(nki) have the same order, therefore we didn't need any additional sorting of the Entrez IDs obtained from fData).

The data.frame containing the expression profiles should now have the following format and content:

```
nkiExprsEntrez[1:6,1:6]

               EntrezGene.ID  NKI_4  NKI_6  NKI_7  NKI_8  NKI_9
Contig45645_RC         64388 -0.215  0.071  0.182 -0.343 -0.134
Contig44916_RC        140883 -0.207  0.055  0.077  0.302  0.051
J00129                  2244 -0.819 -0.391 -0.624 -0.528 -0.811
Contig29982_RC        286133 -0.267 -0.310 -0.120 -0.447 -0.536
D25274                  5879  0.059 -0.135  0.067  0.032 -0.192
D49958                  2823  0.152  0.037 -0.215  0.251  0.183

dim(nkiExprsEntrez)

[1] 14960   338
```

Since several probes can map to the same Entrez gene ID and we need one expression profile per gene, we have to aggregate the data by gene ID, averaging all expression profiles for each gene:

```
     nkiExprsEntrez = aggregate(x=nkiExprsEntrez[,2:ncol(nkiExprsEntrez)],
by=list(EntrezGene.ID=nkiExprsEntrez$EntrezGene.ID), FUN="mean", na.rm=T)
```

This may take a while… The result should look as follows:

```
     nkiExprsEntrez[1:6,1:6]

  EntrezGene.ID  NKI_4  NKI_6  NKI_7  NKI_8  NKI_9
1             1 -0.050  0.125  0.426 -0.016  0.307
2             2 -0.384 -0.314  0.053 -0.035 -0.169
3             9  0.087  0.491 -0.771 -0.507 -0.537
4            10  0.067 -0.027 -0.019 -0.129  0.098
5            12 -0.533  0.695  0.089  0.093  0.166
6            13  0.033  0.037 -0.131  0.019 -0.204
```

To finalize the expression data matrix, we use the Entrez gene IDs as row names, keeping only the expression values in the columns of the data.frame:

```
     rownames(nkiExprsEntrez) = nkiExprsEntrez$EntrezGene.ID
     nkiExprsEntrez = nkiExprsEntrez[,2:ncol(nkiExprsEntrez)]
```

The final gene expression matrix should have expression profiles for 337 samples and 13,120 unique Entrez gene IDs:

```
     nkiExprsEntrez[1:6,1:6]

    NKI_4  NKI_6  NKI_7  NKI_8  NKI_9 NKI_11
1  -0.050  0.125  0.426 -0.016  0.307 -0.310
2  -0.384 -0.314  0.053 -0.035 -0.169 -0.116
9   0.087  0.491 -0.771 -0.507 -0.537 -0.136
10  0.067 -0.027 -0.019 -0.129  0.098  0.001
12 -0.533  0.695  0.089  0.093  0.166 -0.759
13  0.033  0.037 -0.131  0.019 -0.204  0.031

     dim(nkiExprsEntrez)

[1] 13120   337
```

For demonstrating how to use external files for PathWave, we additionally save the expression matrix to a file:

```
     write.table(nkiExprsEntrez, file="test-expression-data.tsv", sep="\t",
quote=FALSE)
```

IMOPRTANT NOTE: For the use with PathWave, gene expression data should, like in this case, be log-transformed! This has to be particularly kept in mind when using RNA-seq expression data because many RNA-seq datasets report plain read counts or RPKM values which are not yet log-transformed.

2. Sample classes for comparison

We need to define two distinct sample classes for comparison. For this purpose, we divide the samples into one group of estrogen receptor-positive (ER+, here: er=1) breast cancer and one group of ER- (here: er=0) breast cancer. Please note that a clinically more relevant distinction would also consider other tumor characteristics, like the status of further receptors (PR, HER2), the lymph node status, luminal A/B, BRCA1 and BRCA2 mutations and others. Since not all clinically relevant sample annotation is available for this dataset, here we just show a comparison for demonstrative purpose: ER+ versus ER- breast cancer.

The following creates a factor indicating the sample classes of the single samples (this is already in the correct order):

```
      sampleClasses = as.factor(c("no_er","er")[pData(nki)$er+1])
```

(Here, we use “er” and “no_er” as class labels instead of “1” and “0”).

For demonstrating how to use external files for PathWave, we additionally create a mapping of sample IDs to sample classes, and save it to a file:

```
      sample2classMap = cbind(pData(nki)$samplename,
as.character(sampleClasses))

      head(sample2classMap)

     [,1]     [,2]
[1,] "NKI_4"  "er"
[2,] "NKI_6"  "er"
[3,] "NKI_7"  "no_er"
[4,] "NKI_8"  "no_er"
[5,] "NKI_9"  "er"
[6,] "NKI_11" "er"


      write.table(sample2classMap, file="test-sample2class-mapping.tsv",
sep="\t", quote=FALSE, col.names=FALSE, row.names=FALSE)
```

3. Running PathWave

Running PathWave on this data is now straight forward. Here, we will run PathWave in two different ways: 1) using the R objects created (expression matrix and sample factor), and 2) using the saved files which contain the same information. But first, we have to load the PathWave library:

```
      library(PathWave)
```

3.1. Running from R objects

To run PathWave on the R objects (nkiExprsEntrez, sampleClasses), we select the pathway set we are interested in. Let's use human KEGG pathways (KEGG.hsa) because we have human expression data

and would like to obtain colored pathway maps from KEGG.

Additionally, we choose to perform 1000 random permutations (default) of the sample classes to determine statistical significance (via fitting of a Gumbel distribution to the randomly obtained values). We restrict the analysis to metabolic pathways (param.kegg.only_metabolism=TRUE) and set the *P*-value threshold to 0.001. Correction for multiple testing will be done using the Bonferroni method (default) over the number of pathways.

Finally, we limit our results to pathways for which at least 6 reactions/proteins involving at least 6 genes are up- or downregulated (param.filter.size=6). This allows to filter out cases in which, for example, few enzymes are deregulated but used several times in the metabolic network of a pathway (i.e. involved in several reactions/network nodes).

```
      pwres = pathWave.run(preprocessed.tag="KEGG.hsa",
input.exprdata=nkiExprsEntrez, input.sampleclasses=sampleClasses,
param.numperm=1000, param.pvalue.threshold=0.001,
param.kegg.only_metabolism=TRUE, param.filter.size=6, verbose=TRUE)
```

(For more details on parameters and possible settings, please see ?pathWave.run and the manual).

Important note: PathWave ignores all gene expression profiles that have zero variance (and thus offer no useful information and do not allow to compute z-scores). Expression profiles containing missing values (NA), as in the case of this example, are handled in the same way because they do not allow to produce comparable pathway maps for all samples (which are required to identify significant pattern changes by applying Haar wavelet transforms: see the PathWave publications for more details).

If verbose=TRUE is specified, running this command will produce an output like the following:

```
No configuration file: pathwave.run.conf
Using preprocessed pathway data: KEGG.hsa
Using metabolic KEGG pathways only: TRUE
Using expression data from data frame passed to function
Using sample class definition from factor passed to function
Using prefix for output files:
Using p-value threshold: 0.001
Using gene filter size: 6
Warning: no meaningful correction method for param.pvalue.correction.method;
using default: Bonferroni
Using correction method for multiple testing: Bonferroni
Warning: no logical (TRUE/FALSE) value for param.ztransform; using default:
TRUE
Applying z-transformation to expression data: TRUE
Number of permutations to perform: 1000

[...]

Summary for results before filtering and correction for multiple testing:

        pw.pathWave

        Pathways: 79 pathways with 3086 reactions
        Source url: /home/piro/bioinfo/data/KEGG/2011-04-14/hsa/
        Version of 2012-05-18
```

```
        Expression data: 2458 reactions
        Samples: 337
        Classes: er,no_er

        Overlap between expression and pathway reactions: 2391

        Permutations: 1000

        Number of features generated: 18793
        Number of pathways with p.value <= 0.01: 74 and p.value <= 0.05: 74

        Significance of regulation patterns (first 5):

                pathway p.value(uncorrected)
                hsa00010 < 2.2e-16
                hsa00030 < 2.2e-16
                hsa00040 < 2.2e-16
                hsa00051 < 2.2e-16
                hsa00052 < 2.2e-16

Summary of significant and filtered results:

        pw.result

        P-value cutoff p <= 0.001
        Filtering was used: TRUE
        Size of filter: 6
        Multiple testing correction: Bonferroni
        Number of pathways with significant pattern: 42
```

The function call returns a list object composed of three elements:

`pwres$results.all`: results for all pathways, whether significant or not
`pwres$results.filtered`: only filtered, significant pathways
`pwres$results.table`: human readable table with a summary of filtered, significant results

Depending on the number of permutations requested, running PathWave may take a while. More than 1000 permutations should best be performed on systems with a physical memory of 16 GB or more.

3.2. Running from data files and writing output files

Exactly the same results can be obtained using the saved data files (instead of the R objects) simply by specifying the respective file names for input.exprdata and input.sampleclasses:

```
     pwres = pathWave.run(preprocessed.tag="KEGG.hsa", input.exprdata="test-
expression-data.tsv", input.sampleclasses="test-sample2class-mapping.tsv",
param.numperm=1000, param.pvalue.threshold=0.001,
param.kegg.only_metabolism=TRUE, param.filter.size=6, verbose=TRUE,
output.file.prefix="pathwave-results-breastcancer")
```

By additionally specifying output.file.prefix two output files are produced, containing the results; in this case:

1) “pathwave-results-breastcancer-pw-results.rda”: This file contains the pwres object for later use as an R data file.

2) “pathwave-results-breastcancer-pw-results_table.tsv”: This file contains the result table as a tab-separated vector (TSV), which can also be loaded as a spreadsheet using Excel and similar applications.

Note: when running PathWave on external data files, instead of specifying the parameters on the R command line for pathWave.run(), they can alternatively be specified in a configuration file using as a command:

```
pwres <- pathWave.run(configfile="<my_conf_file>")
```

Please see the full manual for further information on the format of configuration files.

3.3. Analyzing the output

The most important output provided by PathWave is the table containing the list of significantly deregulated pathways:

```
head(pwres$results.table)

        pathway.name                                  p.value reactions.up
hsa00010 "Glycolysis / Gluconeogenesis"              "0"     "4"
hsa00030 "Pentose phosphate pathway"                 "0"     "3"
hsa00040 "Pentose and glucuronate interconversions" "0"     "3"
hsa00051 "Fructose and mannose metabolism"           "0"     "2"
hsa00052 "Galactose metabolism"                      "0"     "9"
hsa00100 "Steroid biosynthesis"                      "0"     "14"
         reactions.nochange reactions.down regulation.direction
hsa00010 "6"                "17"           "er"
hsa00030 "3"                "10"           "er"
hsa00040 "1"                "5"            "er"
hsa00051 "6"                "8"            "er"
hsa00052 "5"                "10"           "er"
hsa00100 "6"                "6"            "er"
```

(Note: since the pathways' *P*-values are determined empirically via random sampling, they may be slightly different for every run.)

While a pathway's *P*-value (p.value) is corrected for multiple testing, the reported counts of up- or down-regulated reactions (reactions.up, reactions.down) are determined from the uncorrected *P*-values for the single reactions (t-tests). Reactions are declared up-/down-regulated, if their *P*-value is lower than the *P*-value threshold specified in the function call pathWave.run(param.pvalue.threshold=...).

Finally, we compute URLs for each significantly deregulated pathway (except hsa01100 which is too complex because it represents the entire metabolism):

```
keggurls <- pathWave.getColorKEGGMapURLs(pwres$results.filtered,
preprocessed.tag="KEGG.hsa", col=c("green","grey","red"),
col.sign.pattern=c("red","red","green"))
```

These URLs can be used to request colored pathway maps from the KEGG web server using any web browser; example:

```
     keggurls[1]

 hsa00010
"http://www.kegg.jp/kegg-bin/show_pathway?
map00010/rn:R04780%09red,green/rn:R04779%09green,red/rn:R03321%09green,red/rn:R0
1015%09green,black/rn:R01070%09green,black/rn:R00014%09grey,black/rn:R00703%09gr
een,black/rn:R03270%09grey,black/rn:R01518%09grey,black/rn:R01061%09green,black/
rn:R01662%09green,black/rn:R01512%09green,black/rn:R02740%09green,black/rn:R0178
8%09grey,black/rn:R00959%09green,black/rn:R00658%09green,black/rn:R00200%09gree
n,black/rn:R01516%09green,black/rn:R00710%09red,black/rn:R00711%09grey,black/rn:
R02739%09green,black/rn:R01786%09green,black/rn:R01600%09green,black/rn:R01602%0
9green,black/rn:R00754%09grey,black/rn:R00431%09red,black/rn:R00726%09red,black/
rn:R00235%09%23bfffbf,black/rn:R00746%09%23bfffbf,black/rn:R02569%09%23bfffbf,b
lack/rn:R07618%09%23bfffbf,black/rn:R09085%09%23bfffbf,black/rn:R09086%09%23bfff
bf,black"
```

The obtained KEGG pathway map for this URL is shown in Fig. 1. This shows, for example, that glycolysis is less pronounced in ER+ than in ER- breast cancer (which does not imply that it is downregulated with respect to normal tissue). The most significant regulatory pattern seen in Fig. 1 (from which the pathway's *P*-value is computed) suggests that ER+ breast tumors show less conversion of beta-D-Fructose 6-phosphate to beta-D-Fructose 1,6-bisphosphate by 6-phosphofructokinase 1 [EC:2.7.1.11] than ER- breast tumors. Instead, they have a higher expression of fructose-1,6-bisphosphatase I [EC:3.1.3.11], responsible for the opposite reaction.

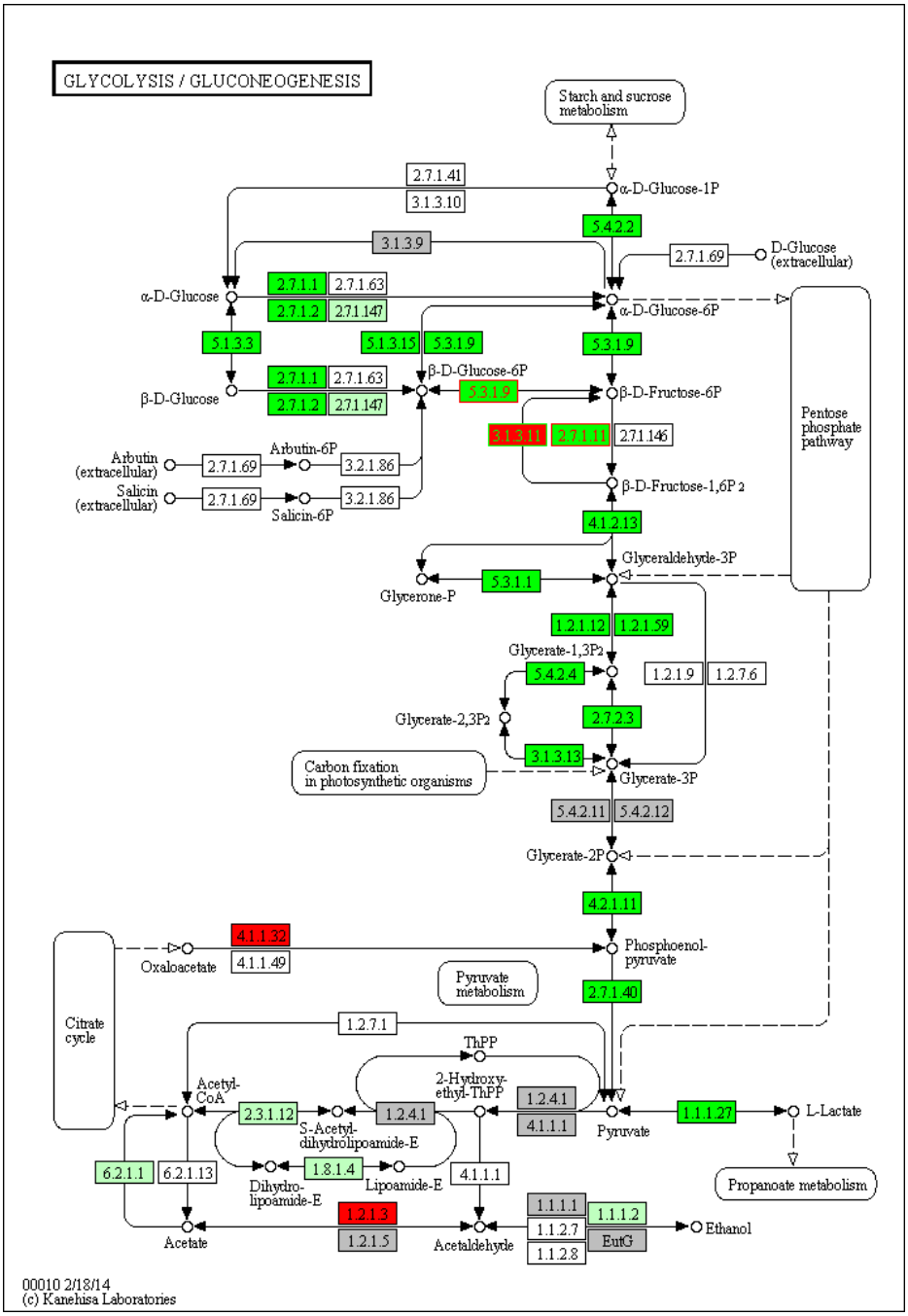

Figure 1: Glycolysis (hsa00010) in ER+ versus ER- breast cancer (green nodes=lower expression; red nodes=higher expression; gray nodes=no notable difference; white nodes=not present in human; the most significant pattern is indicated by red/green font).

The color code to be used for drawing PathWave results can be personalized using the function arguments “col” and “col.sign.pattern”, as in the example above. Colors for both nodes and font can also be arbitrarily chosen and specified as hexadecimal RGB code (e.g. “#FF0000” for red).

Important:

1. Keep in mind that the preprocessed pathway information used with PathWave may be older than the pathway map currently available from the KEGG web server, hence the map may have been changed since pathway information was downloaded/preprocessed. The date of last

modification of the obtained pathway map can be found in its lower left corner (see Fig. 1).

2. Attention: signaling pathways from KEGG can also be drawn, but should be verified thoroughly because they do not involve metabolic reactions with well defined reaction IDs. Therefore, the URL generated by PathWave, instead, communicates the KEGG web server which gene IDs should be colored. If such a gene is involved in multiple protein complexes (i.e. network nodes), all of them will be colored.

The short examples described here reflect most use cases for PathWave and can serve as an easy and quick guide to get started. Nonetheless, we recommend to read the entire manual that describes the options for running PathWave and the structure of the returned results in greater detail. Additionally, it provides information on how to preprocess pathway information from KEGG and BiGG and lists further already preprocessed pathway sets (appart from KEGG.hsa) that come with the PathWave package.