gMCP - an R package for a graphical approach to weighted multiple test procedures

Kornelius Rohmeyer

May 11, 2011

Contents

1 Introduction

This package provides functions and graphical user interfaces for graph based multiple test procedures. These graphs define a weighting strategy for all subsets of null hypotheses and following the closed test procedure weighted tests can be performed on these subsets leading to a multiple test procedure controlling the family wise error rate in the strong sense. In some cases shortcuts are available, for example the weighted Bonferroni procedure leads to a sequentially rejective multiple test procedure.

At all steps either graphical user interfaces or the R Console with S4 objects and methods can be used.

Please note that this is still a beta release and the API will most likely still change in future versions.

1.1 Installation

Open R and type the following commands into the R Console:

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("graph")
> install.packages("gMCP")
```

Select an arbitrary mirror and gMCP will be downloaded and installed.

From now on you can load the gMCP package by entering `library(gMCP)` into the R Console.

If you run into problems, see `http://cran.r-project.org/web/packages/gMCP/INSTALL` or write us an email at `help@small-projects.de`.

1.2 Example and diving in

Let's start with a well-known procedure and see how it fits into this graphical approach to weighted multiple test procedures: The Bonferroni-Holm-Procedure [8].

Theorem 1.1 (Bonferroni-Holm-Procedure)**.** *Let $T_1, \ldots, T_m$ be test statistics for $m \in \mathbb{N}$ null hypotheses $H_1, \ldots, H_m$ and $p_1, \ldots, p_m$ the associated p-values. Then the following test will control the familywise error rate at level $\alpha \in]0, 1[$ in the strong sense:*

Denote the ordered p-values by $p^{(1)} < p^{(2)} < \ldots < p^{(m)}$ and the corresponding hypotheses by $H^{(1)}, H^{(2)}, \ldots, H^{(m)}$.

Reject $H^{(1)}, H^{(2)}, \ldots, H^{(j)}$ such that

$$p^{(i)} \leq \frac{\alpha}{n - i + 1} \quad \text{for all } 1 \leq i \leq j.$$

The corresponding graph for the Bonferroni-Holm-Procedure for three hypotheses is given in Figure 1. We see a fully connected graph, where each node represents a hypothesis and the nodes and edges have weights.

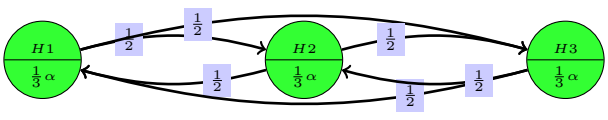

Figure 1: Graph representing the Bonferroni-Holm-Procedure for three hypotheses.

A null hypothesis can be rejected, when the p-value is less than the alpha level of the corresponding node. In this case the graph will be updated and the alpha level of this node is passed according to the edge weights.

Example 1.2. We give an example for the Bonferroni-Holm-Procedure that will be used repeatedly throughout this manual. Of course this package is made for more advanced tests (you find a selection in section 8), but since most readers are already familiar with this procedure, for a first introduction of gMCP, we stick to this simple example.

Let $p_1 = 0.01$, $p_2 = 0.07$ and $p_3 = 0.02$ be three p-values and $\alpha = 0.05$. In the first step H_1 can be rejected since $p_1 < \alpha/3$. The updated graph can be seen in figure 2 and now also H_3 can be rejected since $p_1 < \alpha/2$. Again the graph is updated, but H_2 can not be rejected.

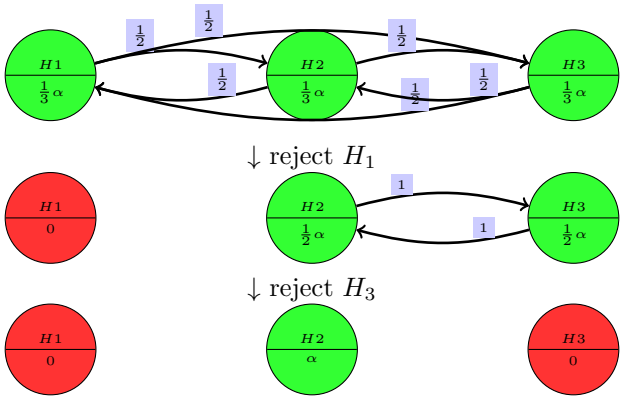

Figure 2: Example showing how two null hypotheses can be rejected with p-values $p_1 = 0.01$, $p_2 = 0.07$ and $p_3 = 0.02$.

Let's reproduce this with the `gMCP` package. We start R and enter:

```
> library(gMCP)
> graphGUI()
```

The GUI seen in Figure 4 is shown and we select from the menu "*Example graphs*" the entry "*Bonferroni-Holm Test*". We enter the three p-values in the respective fields on the right side. By clicking on the button with the green arrow we start the test procedure and can sequentially reject all three hypotheses.

If we don't want to use the GUI we can also use R:

```
> library(gMCP)
> graph <- BonferroniHolmGraph(3)
> gMCP(graph, pvalues=c(0.01,0.07,0.02), alpha=0.05)
gMCP-Result

Initial graph:
A graphMCP graph
H1 (not rejected, weight=0.3333)
H2 (not rejected, weight=0.3333)
H3 (not rejected, weight=0.3333)
Edges:
H1  -( 1/2 )->  H2
H1  -( 1/2 )->  H3
H2  -( 1/2 )->  H1
H2  -( 1/2 )->  H3
H3  -( 1/2 )->  H1
H3  -( 1/2 )->  H2


P-values:
  H1   H2   H3
0.01 0.07 0.02

Adjusted p-values:
```

```
  H1   H2   H3
0.03 0.07 0.04

Alpha: 0.05

Hypothesis rejected:
   H1    H2    H3
 TRUE FALSE  TRUE

Final graph after 2 steps:
A graphMCP graph
H1 (rejected, weight=0)
H2 (not rejected, weight=1)
H3 (rejected, weight=0)
No edges.
```

2 Creating the graph

In the first step a graph that describes the multiple test procedures must be created.

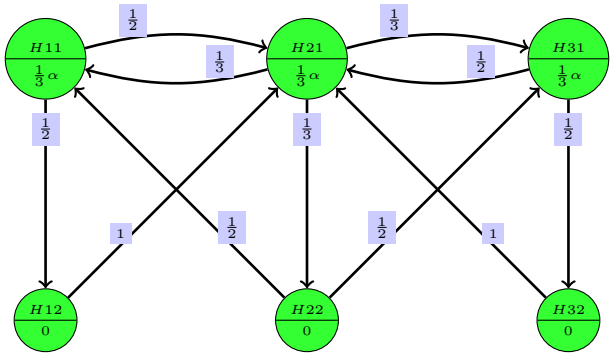

Figure 3: Example graph from [4] that we will create in this vignette.

2.1 Using R

We build upon the package `graph` [6], more precisely we declare a new class `graphMCP` that is a subclass of `graphNEL`. The `initialize` method of this subclass differs only in an extra argument `alpha`, the initial allocation of the significance level alpha to the individual hypotheses. Declaration of the nodes and edges is inherited from class `graphNEL`.

As an example we now create the graph from Bretz et al. [4] that you can see in figure 3.

```
> hnodes <- c("H11", "H21", "H31", "H12", "H22", "H32")
> weights <- c(1/3, 1/3, 1/3, 0, 0, 0)
> edges <- list()
> edges[["H11"]] <- list(edges=c("H21","H12"), weights=c(1/2, 1/2))
> edges[["H21"]] <- list(edges=c("H11","H31","H22"), weights=c(1/3, 1/3, 1/3))
> edges[["H31"]] <- list(edges=c("H21","H32"), weights=c(1/2, 1/2))
> edges[["H12"]] <- list(edges="H21", weights=1)
> edges[["H22"]] <- list(edges=c("H11","H31"), weights=c(1/2, 1/2))
> edges[["H32"]] <- list(edges="H21", weights=1)
> graph <- new("graphMCP", nodes=hnodes, edgeL=edges, weights=weights)
```

Let's print the newly created graph:

```
> print(graph)
```

```
A graphMCP graph
H11 (not rejected, weight=0.3333)
H21 (not rejected, weight=0.3333)
H31 (not rejected, weight=0.3333)
H12 (not rejected, weight=0)
H22 (not rejected, weight=0)
H32 (not rejected, weight=0)
Edges:
H11  -( 1/2 )->  H21
H11  -( 1/2 )->  H12
H21  -( 1/3 )->  H11
H21  -( 1/3 )->  H31
H21  -( 1/3 )->  H22
H31  -( 1/2 )->  H21
H31  -( 1/2 )->  H32
H12  -( 1 )->  H21
H22  -( 1/2 )->  H11
H22  -( 1/2 )->  H31
H32  -( 1 )->  H21
```

Since we also want to visualize the graph, we use the method `nodeRenderInfo` from package `graph` to set appropriate x- and y-coordinates in the renderInfo. (We are compatible to the renderInfo usage from package Rgraphviz [7].)

```
> nodeX <- c(H11=100, H21=300, H31=500, H12=100, H22=300, H32=500)
> nodeY <- c(H11=100, H21=100, H31=100, H12=300, H22=300, H32=300)
> nodeRenderInfo(graph) <- list(nodeX=nodeX, nodeY=nodeY)
```

For placement of the nodes in a matrix pattern, the function `placeNodes` is helpful. The following code does the same as the three lines of R code above.

```
> graph <- placeNodes(graph, nrow=2)
```

Coordinates are interpretated as pixels in the GUI and big points in LaTeX (72 bp = 1 inch).

Let's take a look at the graph in LaTeX rendered with TikZ [10] (you can see the compiled result in figure 3):

```
> cat(graph2latex(graph))
\begin{tikzpicture}[scale=1]
\node (H11) at (100bp,-100bp)[draw,circle split,fill=green!80] {$H11$ \nodepart{lower} $\frac{1}{3}\alpha$};
\node (H21) at (300bp,-100bp)[draw,circle split,fill=green!80] {$H21$ \nodepart{lower} $\frac{1}{3}\alpha$};
\node (H31) at (500bp,-100bp)[draw,circle split,fill=green!80] {$H31$ \nodepart{lower} $\frac{1}{3}\alpha$};
\node (H12) at (100bp,-300bp)[draw,circle split,fill=green!80] {$H12$ \nodepart{lower} $0$};
\node (H22) at (300bp,-300bp)[draw,circle split,fill=green!80] {$H22$ \nodepart{lower} $0$};
\node (H32) at (500bp,-300bp)[draw,circle split,fill=green!80] {$H32$ \nodepart{lower} $0$};
\draw [->,line width=1pt] (H11) to[bend left=15] node[near start,above,fill=blue!20] {$\frac{1}{2}$} (H21);
\draw [->,line width=1pt] (H11) to[auto] node[near start,above,fill=blue!20] {$\frac{1}{2}$} (H12);
\draw [->,line width=1pt] (H21) to[bend left=15] node[near start,above,fill=blue!20] {$\frac{1}{3}$} (H11);
\draw [->,line width=1pt] (H21) to[bend left=15] node[near start,above,fill=blue!20] {$\frac{1}{3}$} (H31);
\draw [->,line width=1pt] (H21) to[auto] node[near start,above,fill=blue!20] {$\frac{1}{3}$} (H22);
\draw [->,line width=1pt] (H31) to[bend left=15] node[near start,above,fill=blue!20] {$\frac{1}{2}$} (H21);
\draw [->,line width=1pt] (H31) to[auto] node[near start,above,fill=blue!20] {$\frac{1}{2}$} (H32);
\draw [->,line width=1pt] (H12) to[auto] node[near start,above,fill=blue!20] {$1$} (H21);
\draw [->,line width=1pt] (H22) to[auto] node[near start,above,fill=blue!20] {$\frac{1}{2}$} (H11);
\draw [->,line width=1pt] (H22) to[auto] node[near start,above,fill=blue!20] {$\frac{1}{2}$} (H31);
\draw [->,line width=1pt] (H32) to[auto] node[near start,above,fill=blue!20] {$1$} (H21);
\end{tikzpicture}
```

We can even change the position of the edge labels for further fine tuning of the graphical representation. With the following command we place the label for the edge from `H1` to `H2` at position (200, 80):

```
> edgeData(graph, "H11", "H21", "labelX") <- 200
> edgeData(graph, "H11", "H21", "labelY") <- 80
```

2.1.1 graph2matrix and matrix2graph

We can also constuct a graph from a given adjacency matrix via the command `matrix2graph`:

```
> # Bonferroni-Holm:
> m <- matrix(rep(1/3, 16), nrow=4)
> diag(m) <- c(0, 0, 0, 0)
> graph <- matrix2graph(m)
> print(graph)
A graphMCP graph
H1 (not rejected, weight=0.25)
H2 (not rejected, weight=0.25)
H3 (not rejected, weight=0.25)
H4 (not rejected, weight=0.25)
Edges:
H1  -( 1/3 )->  H2
H1  -( 1/3 )->  H3
H1  -( 1/3 )->  H4
H2  -( 1/3 )->  H1
H2  -( 1/3 )->  H3
H2  -( 1/3 )->  H4
H3  -( 1/3 )->  H1
H3  -( 1/3 )->  H2
H3  -( 1/3 )->  H4
H4  -( 1/3 )->  H1
H4  -( 1/3 )->  H2
H4  -( 1/3 )->  H3
> graph2matrix(graph)
       H1     H2     H3     H4
H1 0.0000 0.3333 0.3333 0.3333
H2 0.3333 0.0000 0.3333 0.3333
H3 0.3333 0.3333 0.0000 0.3333
H4 0.3333 0.3333 0.3333 0.0000
```

2.2 Using the GUI

The creation of `graphMCP` objects as seen in the last section with basic R commands is very straight forward, but still takes some time and typos may occur. More convenient for the average user is the use of the graphical user interface for creating and editing MCP graphs that the `gMCP` package includes.

It is called by the command `graphGUI()` and takes as optional argument a variable name, given as a character string, of the graph to edit or under which a newly created `graphMCP` object will be available from the R command line.

```
> graphGUI("graph")
```

Let's take a look at the icon panel:

 This button lets you add a new node to the graph. After pressing the button click somewhere on the graph panel and a new node will appear at this place.

 This button lets you add a new edge between two nodes. After pressing the button click on the node the edge should start and after that on the node the edge should end.

 For really big graphs the ability to zoom in and out is usefull.

Starts the testing procedure / goes back to the graph modification.

Calculates the adjusted p-values.

 Calculates simultaneous confidence intervals.

With drag and drop you can move nodes and also adjust edges.

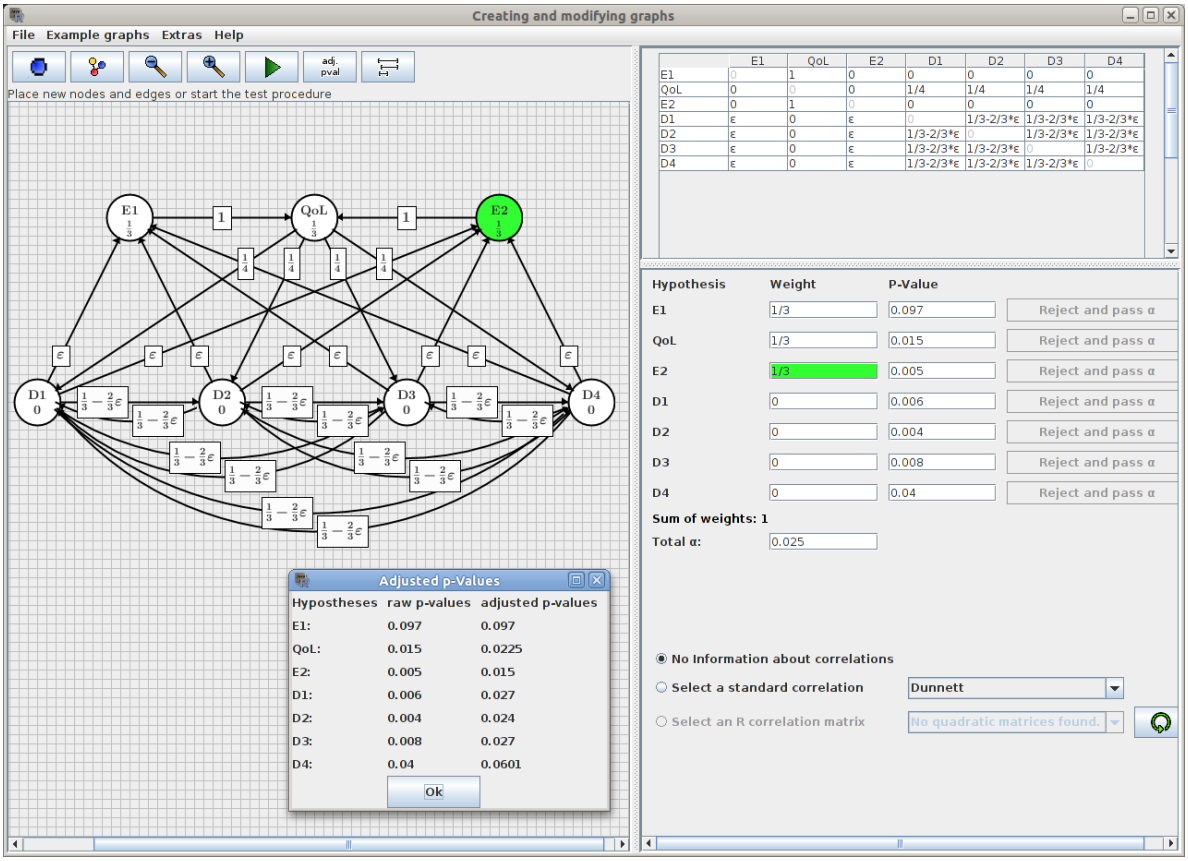

Figure 4: The graphical user interface allows testing, calculation of confidence intervals and adjusted p-values.

3 The sequentially rejective MTP

For a full description of the sequentially rejective multiple testing procedure take a look at Bretz et al. [3].

3.1 Using R

You can either specify each rejection step yourself or simply use the method `gMCP`:

```
> graph <- graphFromBretzEtAl2011()
> # We can reject a single node:
> print(rejectNode(graph, "H11"))
A graphMCP graph
H11 (rejected, weight=0)
H21 (not rejected, weight=0.5)
H31 (not rejected, weight=0.3333)
H12 (not rejected, weight=0.1667)
H22 (not rejected, weight=0)
H32 (not rejected, weight=0)
Edges:
H21  -( 2/5 )->  H31
H21  -( 2/5 )->  H22
H21  -( 1/5 )->  H12
H31  -( 1/2 )->  H21
H31  -( 1/2 )->  H32
H12  -( 1 )->  H21
H22  -( 1/2 )->  H31
H22  -( 1/4 )->  H21
H22  -( 1/4 )->  H12
H32  -( 1 )->  H21
> # Or given a vector of pvalues let the function gMCP do all the work:
```

```
> pvalues <- c(0.1, 0.008, 0.005, 0.15, 0.04, 0.006)
> result <- gMCP(graph, pvalues)
> print(result)
gMCP-Result

Initial graph:
A graphMCP graph
H11 (not rejected, weight=0.3333)
H21 (not rejected, weight=0.3333)
H31 (not rejected, weight=0.3333)
H12 (not rejected, weight=0)
H22 (not rejected, weight=0)
H32 (not rejected, weight=0)
Edges:
H11  -( 1/2 )->  H21
H11  -( 1/2 )->  H12
H21  -( 1/3 )->  H11
H21  -( 1/3 )->  H31
H21  -( 1/3 )->  H22
H31  -( 1/2 )->  H21
H31  -( 1/2 )->  H32
H12  -( 1 )->  H21
H22  -( 1/2 )->  H11
H22  -( 1/2 )->  H31
H32  -( 1 )->  H21


P-values:
  H11   H21   H31   H12   H22   H32
0.100 0.008 0.005 0.150 0.040 0.006

Adjusted p-values:
   H11    H21    H31    H12    H22    H32
0.1200 0.0160 0.0150 0.1500 0.1200 0.0225

Alpha: 0.05

Hypothesis rejected:
  H11   H21   H31   H12   H22   H32
FALSE  TRUE  TRUE FALSE FALSE  TRUE

Final graph after 3 steps:
A graphMCP graph
H11 (not rejected, weight=0.6667)
H21 (rejected, weight=0)
H31 (rejected, weight=0)
H12 (not rejected, weight=0)
H22 (not rejected, weight=0.3333)
H32 (rejected, weight=0)
Edges:
H11  -( 2/3 )->  H12
H11  -( 1/3 )->  H22
H12  -( 1/2 )->  H11
H12  -( 1/2 )->  H22
H22  -( 1 )->  H11
```

We can create a TikZ graphic from the last graph with `graph2latex(result@graphs[[4]])` that is shown in figure 5.

The command `gMCPReport` generates a full report of the testing procedure:

```
> gMCPReport(result, "Report.tex")
```

3.1.1 Adjusted p-values and simultaneous confidence intervals

Also adjusted p-values and simultaneous confidence intervals can be computed.

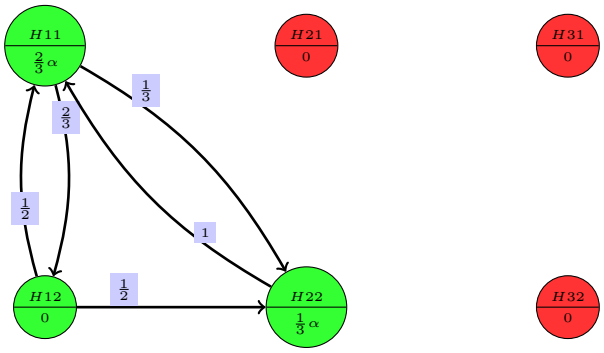

Figure 5: Final graph from the test procedure after rejection of H_{21}, H_{31} and H_{32}.

Let's assume the tests for hypotheses $H1: \theta_1 \leq 0$, $H2: \theta_2 \leq 0$ and $H3: \theta_3 \leq 0$ are three t-tests with degree of freedom 9. The estimates are $\hat{\theta}_1 = 0.981$, $\hat{\theta}_2 = 1.089$ and $\hat{\theta}_3 = 0.8706$, the sample standard deviations $s_1 = 0.876$, $s_2 = 1.291$ and $s_3 = 0.8571$ the t-statistics 3.541, 2.666 and 3.212 and the corresponding p-values 0.0063, 0.02577 and 0.01062. We want to adjust for multiple testing by using the Bonferroni-Holm-Procedure with $\alpha = 0.025$.

```
> # Estimates:
> est <- c("H1"=0.860382, "H2"=0.9161474, "H3"=0.9732953)
> # Sample standard deviations:
> ssd <- c("H1"=0.8759528, "H2"=1.291310, "H3"=0.8570892)
> pval <- c(0.01260, 0.05154, 0.02124)/2
> simConfint(BonferroniHolmGraph(3), pvalues=pval,
+                 confint=function(node, alpha) {
+                         c(est[node]-qt(1-alpha,df=9)*ssd[node]/sqrt(10), Inf)
+                 }, estimates=est, alpha=0.025, mu=0, alternative="greater")
   lower bound estimate upper bound
H1      0.0000   0.8604         Inf
H2     -0.0076   0.9161         Inf
H3      0.0000   0.9733         Inf
> # Note that the sample standard deviations in the following call
> # will be calculated from the pvalues and estimates.
> simConfint(BonferroniHolmGraph(3), pvalues=pval,
+                 confint="t", df=9, estimates=est, alpha=0.025, alternative="greater")
     lower bound estimate upper bound
[1,]    0.000000   0.8604         Inf
[2,]   -0.007581   0.9161         Inf
[3,]    0.000000   0.9733         Inf
```

3.2 Using the GUI

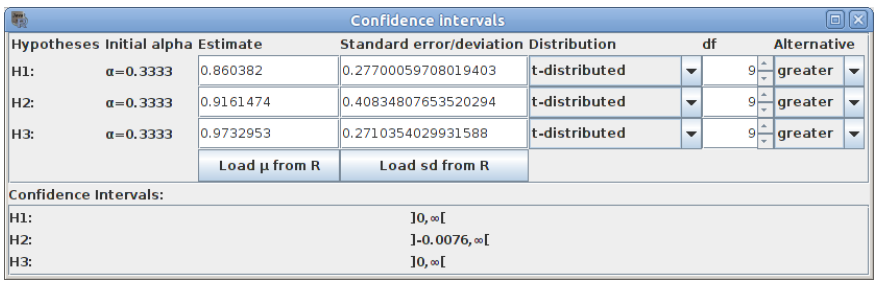

Figure 6: For normal and t-distributions simultaneous CI can be calculated by the GUI.

Use the following two buttons: 

See [5].

4 Weighted parametric tests

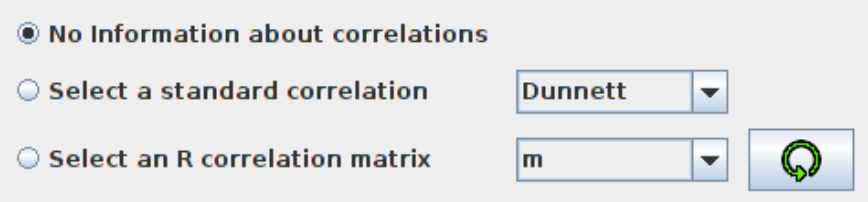

Figure 7: You can also specify a correlation between the tests.

In the lower right panel with p-values, it is also possible to specify a known correlation between these values (see figure 7).

For further information please take a look at the vignette "*Weighted parametric tests defined by graphs*".

5 Epsilon edges

The GUI supports epsilon edges. You can enter the weights in R syntax, e.g. `1-2*\epsilon+1/3*\epsilon^2` for $1 - 2\varepsilon + \frac{1}{3}\varepsilon^2$.

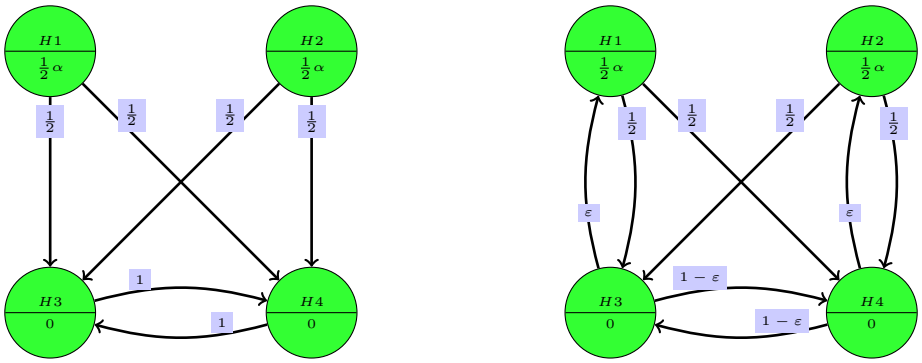

Figure 8: The Parallel Gatekeeping and the Improved Parallel Gatekeeping Procedure.

```
> graph <- graphForImprovedParallelGatekeeping()

A graphMCP graph
H1 (not rejected, weight=0.5)
H2 (not rejected, weight=0.5)
H3 (not rejected, weight=0)
H4 (not rejected, weight=0)
Edges:
H1  -( 1/2 )->  H3
H1  -( 1/2 )->  H4
H2  -( 1/2 )->  H3
H2  -( 1/2 )->  H4
H3  -( 1-\epsilon )->  H4
H3  -( \epsilon )->  H1
H4  -( 1-\epsilon )->  H3
H4  -( \epsilon )->  H2
> substituteEps(graph, eps=0.001)
A graphMCP graph
H1 (not rejected, weight=0.5)
H2 (not rejected, weight=0.5)
H3 (not rejected, weight=0)
H4 (not rejected, weight=0)
Edges:
```

```
H1  -( 1/2 )->  H3
H1  -( 1/2 )->  H4
H2  -( 1/2 )->  H3
H2  -( 1/2 )->  H4
H3  -( 999/1000 )->  H4
H3  -( 1/1000 )->  H1
H4  -( 999/1000 )->  H3
H4  -( 1/1000 )->  H2
```

```
> gMCP(graph, pvalues=c(0.02, 0.04, 0.01, 0.02), eps=0.001)
```

```
gMCP-Result

Initial graph:
A graphMCP graph
H1 (not rejected, weight=0.5)
H2 (not rejected, weight=0.5)
H3 (not rejected, weight=0)
H4 (not rejected, weight=0)
Edges:
H1  -( 1/2 )->  H3
H1  -( 1/2 )->  H4
H2  -( 1/2 )->  H3
H2  -( 1/2 )->  H4
H3  -( 1-\epsilon )->  H4
H3  -( \epsilon )->  H1
H4  -( 1-\epsilon )->  H3
H4  -( \epsilon )->  H2


P-values:
  H1   H2   H3   H4
0.02 0.04 0.01 0.02

Adjusted p-values:
  H1   H2   H3   H4
0.04 0.04 0.04 0.04

Alpha: 0.05

Hypothesis rejected:
  H1   H2   H3   H4
TRUE TRUE TRUE TRUE

Final graph after 4 steps:
A graphMCP graph
H1 (rejected, weight=0)
H2 (rejected, weight=1)
H3 (rejected, weight=0)
H4 (rejected, weight=0)
No edges.
```

6 Power Simulations

No ε-edges are allowed.

6.1 Variable edge weights

	H1	H2	H3	H4
H1	0	γ	1-\gamma	0
H2	6	0	0	1-6
H3	0	1	0	0
H4	1	0	0	0

```
> graph <- graph2FromBretzEtAl2011()
```

```
A graphMCP graph
H1 (not rejected, weight=0.5)
H2 (not rejected, weight=0.5)
```

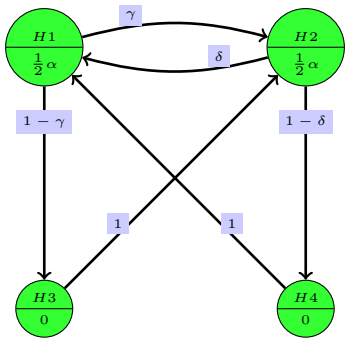

Figure 9: Graph from Bretz et al. (2009)

```
H3 (not rejected, weight=0)
H4 (not rejected, weight=0)
Edges:
H1  -( \gamma )->  H2
H1  -( 1-\gamma )->  H3
H2  -( \delta )->  H1
H2  -( 1-\delta )->  H4
H3  -( 1 )->  H2
H4  -( 1 )->  H1
```

7 Options and Import/Export

7.1 Options

This subsection is work in progress, but fortunately the options in figure 10 should be fairly self-explanatory.

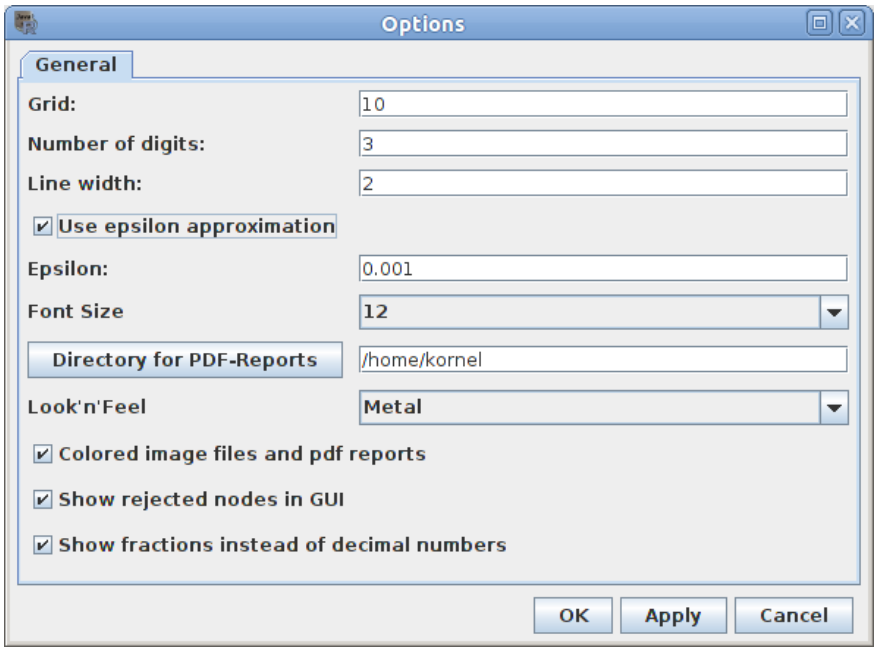

Figure 10: You can configure many things in the option dialog.

7.2 Import/Exports

This subsection is work in progress, but fortunately the menu entries in figure 11 should be fairly self-explanatory.

You can export graphs to png files. The background of these png files will be made transperant, so that they will fit into whichever document you insert them. Note that some image viewers visualize transparency with a checkerboard pattern.

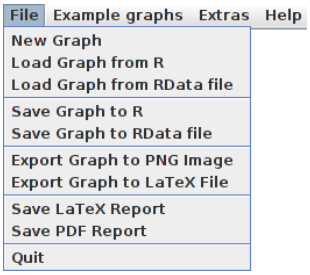

Figure 11: Import and export of graphs.

7.3 Important TikZ commands for optimizing the reports

A clear automatic placement of edges and weight labels without overlapping is a very difficult task and for complicated graphs the `gMCP` package will often fail to accomplish this. There is the possibilty to adjust the edges and labels in the GUI, but since the LaTeX graph layout is not (yet) exactly the same, there is perhaps the need for adjusting the graphs in the TikZ code. The TikZ program is very useful and we recommend it for many purposes, but perhaps you don't have the time to read the 560 pages manual [10], so here is a short overview of the most important commands for this kind of graphs.

Let's start with this graph in figure 12:

```
\begin{tikzpicture}[scale=1]
\node (H11) at (200bp,200bp) [draw,circle split,fill=green!80] {$H11$ \nodepart{lower} $0.0333$};
...
\draw [->,line width=1pt] (H11) to[bend left=15] node[near start,above,fill=blue!20] {0.667} (H12);
...
\end{tikzpicture}
```

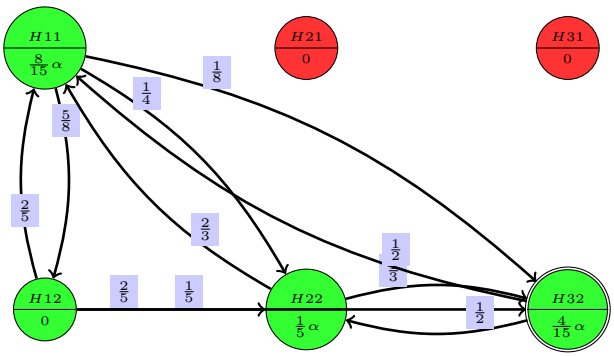

Figure 12: Graph from `graph2latex` that does not look optimal.

You can scale the TikZ graphic by changing the `[scale=1]` option. By default `graph2latex` doesn't scale TikZ graphics, but has an optional parameter `scale`.

For an explanation what `green!80` means and how you can specify other colors, please take a look at the xcolor manual [9].

You can choose between the following label positions `above, below, right, left, above right, above left, below right`, and `below left`. In addition these positions can take an optional dimension argument, so that for example `below=1pt` can be used to place a label below and additionally shift it 1pt downwards.

You can change the position where the edge weight label is placed to `at start`, `very near start`, `near start`, `midway`, `near end`, `very near end` and `at end` or simply use something like `pos=0.5`. If you add an argument `sloped`, the text label will be rotated so that a parallel line to the base line becomes a tangent to the edge.

Often it is useful to reduce the bending angle in `[bend left=15]` below 15. You could also specify and change `out=15` and `in=165` separately.

A powerful feature is the use of styles, since this will effect all objects of a given class. But for this please take a look directly at the TikZ manual [10].

8 Case Studies

This section is work in progress.

8.1 Identifying effective and/or safe doses by stepwise confidence intervals for ratios

In this subsection we show how to use gMCP to reproduce the results of the paper [2] with the same title.

8.2 Testing strategies in multi-dose experiments including active control

[1]

```
> data(hydroquinone)
> pvalues <- c()
> x <- hydroquinone$micronuclei[hydroquinone$group=="C-"]
> for (dose in c("30 mg/kg", "50 mg/kg", "75 mg/kg", "100 mg/kg", "C+")) {
+         y <- hydroquinone$micronuclei[hydroquinone$group==dose]
+         result <- wilcox.test(x, y, alternative="less", correct=TRUE)
+         pvalues <- c(result$p.value,  pvalues)
+ }
> pvalues
[1] 0.004929 0.002634 0.002634 0.004319 0.066255
> library(coin)
> pvalues <- c()
> for (dose in c("30 mg/kg", "50 mg/kg", "75 mg/kg", "100 mg/kg", "C+")) {
+         subdata <- droplevels(hydroquinone[hydroquinone$group %in% c("C-", dose),])
+         result <- wilcox_test(micronuclei ~ group, data=subdata, distribution="exact")
+         pvalues <- c(pvalue(result), pvalues)
+ }
> pvalues
[1] 0.006061 0.001263 0.001263 0.005051 0.135101
```

A Appendix - Multiple Testing Basics

This section is work in progress.

A.1 Closed testing principle

A.2 Partitioning principle

Definition A.1.

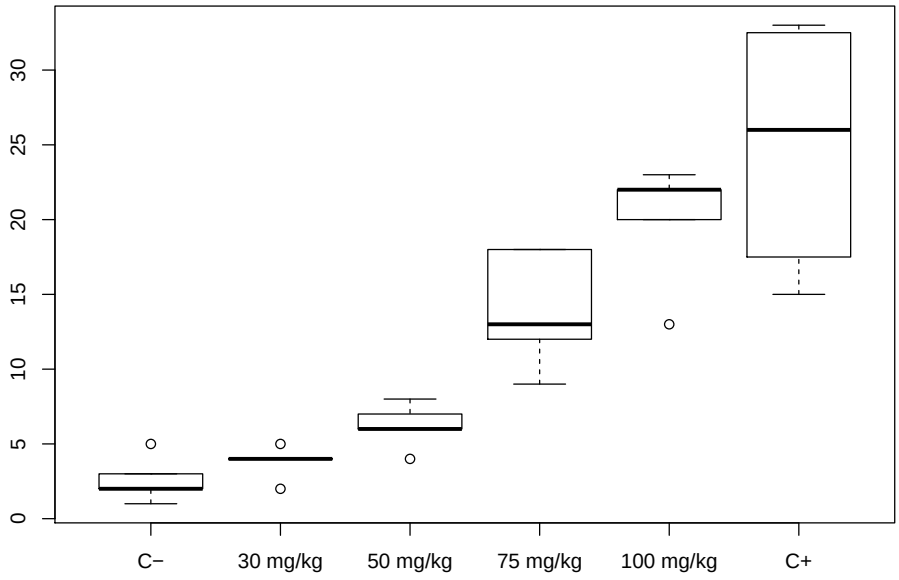

Figure 13: Boxplot of the hydroquinone data set

Index

References

[1] P. Bauer, J. R
"ohmel, W. Maurer, and L. Hothorn. Testing strategies in multi-dose experiments including active control. *Statistics in Medicine*, 17(18):2133–2146, 1998. ISSN 1097-0258.

[2] F. Bretz, L.A. Hothorn, and J.C. Hsu. Identifying effective and/or safe doses by stepwise confidence intervals for ratios. *Statistics in medicine*, 22(6):847–858, 2003. ISSN 1097-0258.

[3] F. Bretz, W. Maurer, W. Brannath, and M. Posch. A graphical approach to sequentially rejective multiple test procedures. *Statistics in medicine*, 28(4):586–604, 2009. URL `www.meduniwien.ac.at/fwf_adaptive/papers/bretz_2009_22.pdf`.

[4] F. Bretz, W. Maurer, and G. Hommel. Test and power considerations for multiple endpoint analyses using sequentially rejective graphical procedures. *Statistics in medicine*, 2010 (in press).

[5] F. Bretz, M. Posch, E. Glimm, F. Klinglmueller, W. Maurer, and K. Rohmeyer. Graphical approaches for multiple comparison problems using weighted bonferroni, simes or parametric tests. *Biometrical Journal*, page to appear, 2011.

[6] R. Gentleman, E. Whalen, W. Huber, and S. Falcon. *graph: A package to handle graph data structures*, 2010. URL `http://CRAN.R-project.org/package=graph`. R package version 1.26.0.

[7] J. Gentry, L. Long, R. Gentleman, S. Falcon, F. Hahne, D. Sarkar, and K. Hansen. *Rgraphviz: Provides plotting capabilities for R graph objects*, 2010. URL `http://www.bioconductor.org/packages/2.6/bioc/html/Rgraphviz.html`. R package version 1.26.0.

[8] S. Holm. A simple sequentially rejective multiple test procedure. *Scand. J. Statist.*, 6:65–70, 1979.

[9] U. Kern. *Extending LaTeX's color facilities: the xcolor package*, 2007. URL `http://www.ctan.org/tex-archive/macros/latex/contrib/xcolor/`.

[10] T. Tantau. *The Tik Z and PGF Packages Manual for version 2.00*, 2008. URL `http://www.ctan.org/tex-archive/graphics/pgf/base/doc/generic/pgf/pgfmanual.pdf`.