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

Kornelius Rohmeyer

November 15, 2010

Contents

1 Introduction

This package provides functions and graphical user interfaces for sequentially rejective multiple test procedures.

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

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

2 Creating the graph

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

2.1 Using R

We build upon the package `graph` [3], 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. [2] that you can see in figure 1.

```
> hnodes <- c("H11", "H21", "H31", "H12", "H22", "H32")
> alpha <- c(0.05/3, 0.05/3, 0.05/3, 0, 0, 0)
> edges <- list()
```

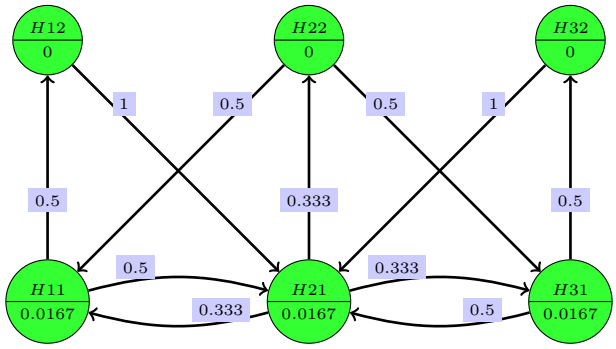

Figure 1: Example graph from [2] that we will create in this vignette.

```
> 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, alpha = alpha)
```

Let's print the newly created graph:

```
> print(graph)
```

```
A graphMCP graph
Overall alpha: 0.05
H11 (not rejected, alpha=0.01667)
H21 (not rejected, alpha=0.01667)
H31 (not rejected, alpha=0.01667)
H12 (not rejected, alpha=0)
H22 (not rejected, alpha=0)
H32 (not rejected, alpha=0)
Edges:
H11 -(0.5)-> H21
H11 -(0.5)-> H12
H21 -(0.3333)-> H11
H21 -(0.3333)-> H31
H21 -(0.3333)-> H22
H31 -(0.5)-> H21
H31 -(0.5)-> H32
H12 -(1)-> H21
H22 -(0.5)-> H11
H22 -(0.5)-> 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 [4].)

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

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 [6] (you can see the compiled result in figure 1):

```
> cat(graph2latex(graph))
```

```
\begin{tikzpicture}[scale=1]
\node (H11) at (100bp,-300bp) [draw,circle split,fill=green!80] {$H11$ \nodepart{lower} $0.0167$};
```

```
\node (H21) at (300bp,-300bp) [draw,circle split,fill=green!80] {$H21$ \nodepart{lower} $0.0167$};
\node (H31) at (500bp,-300bp) [draw,circle split,fill=green!80] {$H31$ \nodepart{lower} $0.0167$};
\node (H12) at (100bp,-100bp) [draw,circle split,fill=green!80] {$H12$ \nodepart{lower} $0$};
\node (H22) at (300bp,-100bp) [draw,circle split,fill=green!80] {$H22$ \nodepart{lower} $0$};
\node (H32) at (500bp,-100bp) [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] {0.5} (H21);
\draw [->,line width=1pt] (H11) to[auto] node[near start,above,fill=blue!20] {0.5} (H12);
\draw [->,line width=1pt] (H21) to[bend left=15] node[near start,above,fill=blue!20] {0.333} (H11);
\draw [->,line width=1pt] (H21) to[bend left=15] node[near start,above,fill=blue!20] {0.333} (H31);
\draw [->,line width=1pt] (H21) to[auto] node[near start,above,fill=blue!20] {0.333} (H22);
\draw [->,line width=1pt] (H31) to[bend left=15] node[near start,above,fill=blue!20] {0.5} (H21);
\draw [->,line width=1pt] (H31) to[auto] node[near start,above,fill=blue!20] {0.5} (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] {0.5} (H11);
\draw [->,line width=1pt] (H22) to[auto] node[near start,above,fill=blue!20] {0.5} (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
```

2.2 Using the GUI

The creation of `graphMCP` objects 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")
```

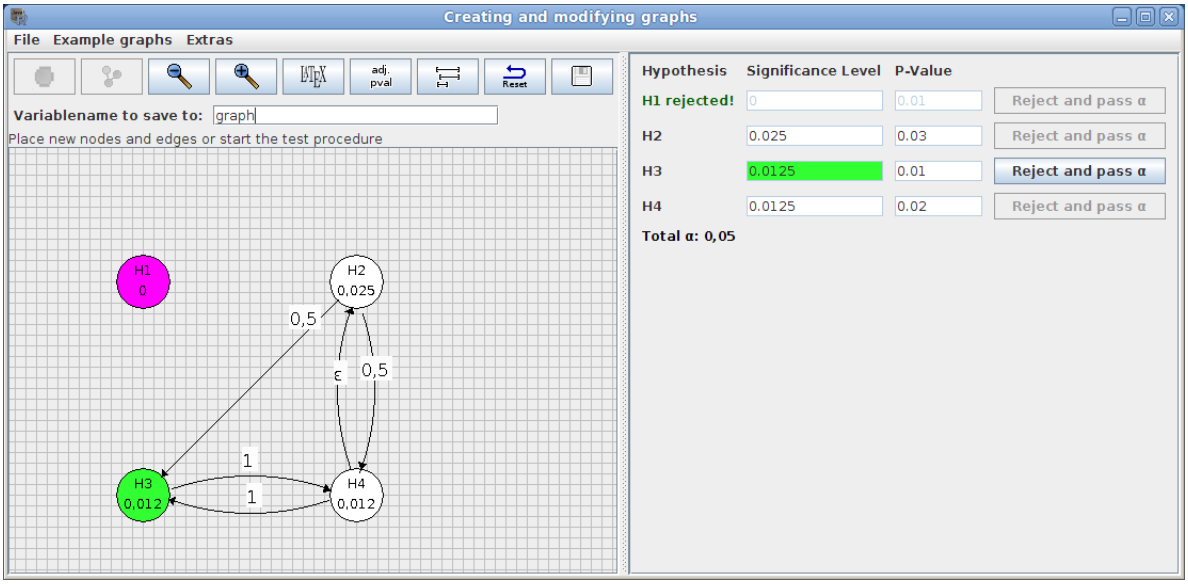

Figure 2: The graphical user interface allows testing, calculation of confidence intervals and adjusted p-values.

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.

 This button exports the graph as an LaTeX TikZ picture.

 Calculates the adjusted p-values.

 Calculates simultaneous confidence intervals.

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

 This button exports the graph back to R under the variable name in the adjacent text field.

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

3 The sequentially rejective MTP

For a full description of the sequentially rejective multiple testing procedure take a look at Bretz et al. [1].

3.1 Using R

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

```
> graph <- createGraphFromBretzEtAl()
> # We can reject a single node:
> print(rejectNode(graph, "H11"))


A graphMCP graph
Overall alpha: 0.025
H11 (rejected, alpha=0)
H21 (not rejected, alpha=0.0125)
H31 (not rejected, alpha=0.008333)
H12 (not rejected, alpha=0.004167)
H22 (not rejected, alpha=0)
H32 (not rejected, alpha=0)
Edges:
H21 -(0.4)-> H31
H21 -(0.4)-> H22
H21 -(0.2)-> H12
H31 -(0.5)-> H21
H31 -(0.5)-> H32
H12 -(1)-> H21
H22 -(0.5)-> H31
H22 -(0.25)-> H21
H22 -(0.25)-> 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

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

Initial graph:
A graphMCP graph
```

```
Overall alpha: 0.025
H11 (not rejected, alpha=0.008333)
H21 (not rejected, alpha=0.008333)
H31 (not rejected, alpha=0.008333)
H12 (not rejected, alpha=0)
H22 (not rejected, alpha=0)
H32 (not rejected, alpha=0)
Edges:
H11 -(0.5)-> H21
H11 -(0.5)-> H12
H21 -(0.3333)-> H11
H21 -(0.3333)-> H31
H21 -(0.3333)-> H22
H31 -(0.5)-> H21
H31 -(0.5)-> H32
H12 -(1)-> H21
H22 -(0.5)-> H11
H22 -(0.5)-> H31
H32 -(1)-> H21

Final graph after 3 steps:
A graphMCP graph
Overall alpha: 0.025
H11 (not rejected, alpha=0.01667)
H21 (rejected, alpha=0)
H31 (rejected, alpha=0)
H12 (not rejected, alpha=0)
H22 (not rejected, alpha=0.008333)
H32 (rejected, alpha=0)
Edges:
H11 -(0.6667)-> H12
H11 -(0.3333)-> H22
H12 -(0.5)-> H11
H12 -(0.5)-> H22
H22 -(1)-> H11
```

We can create a TikZ graphic from the last graph with `graph2latex(result@graphs[[4]])` that is shown in figure 3.

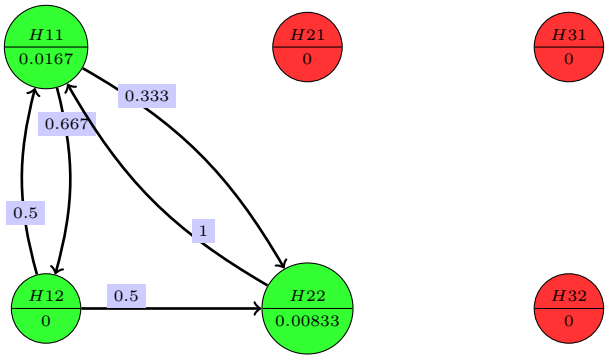

Figure 3: Final graph from the test procedure after rejection of H_{21}, H_{31} and H_{32}.

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.

Let's assume the tests for hypotheses $H1: \theta_1 = 0$, $H2: \theta_2 = 0$ and $H3: \theta_3 = 0$ are three t-tests with degree of freedom 9. The estimates are $\hat{\theta}_1 = 1.071$, $\hat{\theta}_2 = 0.5575$ and $\hat{\theta}_3 = 1.686$, the sample standard deviations $s_1 = 1.2$, $s_2 = 0.8$ and $s_3 = 1.7$ the t-statistics 2.821, 2.204 and 3.136 and the corresponding p-values $p_1 = 0.02$, $p_2 = 0.055$ and $p_3 = 0.012$. We want to adjust for multiple testing by using the Bonferroni-Holm-Procedure.

```
> est <- c("H1"=0.9101444, "H2"=0.4485153, "H3"=1.4568271)
> # Sample standard deviations:
> ssd <- c("H1"=1.2, "H2"=0.8, "H3"=1.7)
> simConfint(createBonferroniHolmGraph(3),
+                 pvalues  = c(0.02,0.055,0.012),
+                 confint = function(node, alpha) {
+                         est[node]+c(-1,1)*qt(1-alpha/2,df=9)*ssd[node]/sqrt(10)
+                 })

   lower bound upper bound
H1        -Inf         Inf
H2     -0.1238       1.021
H3        -Inf         Inf
```

3.2 Using the GUI

Use the following two buttons: 

4 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 [6], so here is a short overview of the most important commands for this kind of graphs.

Let's start with this graph in figure 4:

```
\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}
```

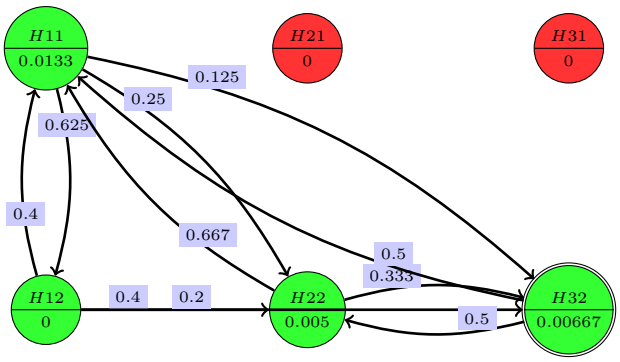

Figure 4: Graph from `graph2latex` that does not look optimal.

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 [5].

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 [6].

References

[1] 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`.

[2] 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).

[3] R.˜Gentleman, Elizabeth 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.

[4] Jeff Gentry, Li˜Long, Robert Gentleman, Seth Falcon, Florian Hahne, Deepayan Sarkar, and Kasper 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.

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

[6] Till 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`.