Weighted parametric tests defined by graphs

Florian Klinglmueller

March 20, 2011

Contents

1 Introduction

This document describes how-to use `gMCP` to construct, conduct and evaluate multiple comparison procedures based on weighted parametric tests which are defined by directed graphs. In addition to sequentially rejective Bonferroni procedures `gMCP` provides functionality to construct tests that take advantage of (partial) knowledge of the the joint distribution of the p-values associated with multiple hypotheses. For the time being `gMCP` implements the case of multiple inferences based on z-tests. Assuming normally distributed test statistics with known variance one can improve on the Bonferroni test in terms of power, provided that there exists (partial) knowledge of the correlation structure between test statistics. If the correlation matrix of a given subset of test statistics is known tests for intersection hypotheses containing the associated elementary hypotheses are computed using the multivariate distribution under the respective null. The consonance condition providing the shortcut utilized by sequentially rejective tests [3, 1] can no longer be guaranteed using this approach, hence, the whole closed test has to be performed. For an exhaustive theoretical specification of the general statistical principle please see [2, Section 3.2].

As an example we will use graph based procedures introduced in Example 1 and Example 2 of [2, Sections 2 and 3.2].

2 Creating Graphs

Examples 1 and 2 of [2] are based on the same weighting scheme. Inferences on two primary and two secondary hypotheses are to be made. For example consider the comparison of two treatments to a control using a primary and secondary endpoint. Only if the the null hypotheses can be rejected for the primary hypotheses are the secondary hypotheses to be tested. If both primary and secondary hypotheses can be rejected for one treatment, then the portions of the global $\alpha-$level level reserved for this treatment is passed to the other treatment. The graph corresponding to this procedures is depicted in figure 1. We assume that all four hypotheses are tested by use of normally distributed test statistics with known variances. In Example 1 no further assumptions on the joint distribution

of test statistics is made. Example 2 assumes that both the statistics associated with the primary hypotheses as well as statistics associated with the secondary hypotheses have pairwise correlations of $\frac{1}{2}$. This would be the case if the two treatments were compared to the same control group using balanced sample sizes.

The main inputs needed for the construction of weighted parametric tests are a directed graph, initial weights for the elementary hypotheses in the global intersection hypothesis as well as a correlation matrix. The graph for both Example 1 and 2 is depicted in Figure 1. As in the article we will distribute the overall $\alpha-$level equally between the two primary hyptheses. Initially the secondary hypotheses will be given no weight. The corresponding initial weights are therefore $(\frac{1}{2}, \frac{1}{2}, 0, 0)$.

We show in Section 2.1 how these parameters can be defined using R. In Section 2.2 we demonstrate how to acchiev the same using the graphical user interface provided by `gMCP`.

2.1 Creating Graphs using the Command Line

The most basic way of defining an MTP in `gMCP` is by way of numeric matrices where each element defines the proportion of the local $\alpha-$level of the elementary hypotheses corresponding to the row index which is passed to the elementary hypotheses associated with the column index. Since no hypotheses reallocates parts of its local $\alpha-$level to itself the diagonal elements of this matrix are zero. The simple graph from our examples is defined by the matrix:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

```
> Gm <- matrix(0, nr = 4, nc = 4)
> Gm[1, 3] <- 1
> Gm[2, 4] <- 1
> Gm[3, 2] <- 1
> Gm[4, 1] <- 1
> Gm

     [,1] [,2] [,3] [,4]
[1,]    0    0    1    0
[2,]    0    0    0    1
[3,]    0    1    0    0
[4,]    1    0    0    0
```

Initial weights are set by means of a numeric vector $\omega_1, ..., \omega_m$ where m denotes the number of test statistics. For the example graph of Figure 1 the corresponding weights vector is $(\frac{1}{2}, \frac{1}{2}, 0, 0)$[1].

```
> w <- c(1/2, 1/2, 0, 0)
> w

[1] 0.5 0.5 0.0 0.0
```

Finally a correlation matrix has to be specified providing information on the pairwise correlations between test statistics. All known pairwise correlations of this matrix are set to numeric values whereas the unknown coefficients are set to `NA`. Example 1 assumes no knowledge of the correlation

[1] Note that this is different to the approach taken in `gMCP` where the local $\alpha-$level are specified for each elementary hypotheses instead of weights

structure. The corresponding 4×4 matrix would therefore have all elements set to NA except for the diagonal elements. Example 2 assumes pairwise correlations of $\frac{1}{2}$ between the statistics associated with both treatments using either the primary or the secondary endpoint. The corresponding correlation matrix has to be specified as:

$$\begin{pmatrix} 1 & \frac{1}{2} & \texttt{NA} & \texttt{NA} \\ \frac{1}{2} & 1 & \texttt{NA} & \texttt{NA} \\ \texttt{NA} & \texttt{NA} & 1 & \frac{1}{2} \\ \texttt{NA} & \texttt{NA} & \frac{1}{2} & 1 \end{pmatrix}$$

We construct the correlation matrix corresponding to Example 1 in R as Cm1 and the one corresponding to Example 2 as Cm2.

```
> Cm <- matrix(NA, nr = 4, nc = 4)
> diag(Cm) <- 1
> Cm1 <- Cm
> Cm[1, 2] <- 1/2
> Cm[2, 1] <- 1/2
> Cm[3, 4] <- 1/2
> Cm[4, 3] <- 1/2
> Cm2 <- Cm
> Cm1
```

```
     [,1] [,2] [,3] [,4]
[1,]    1   NA   NA   NA
[2,]   NA    1   NA   NA
[3,]   NA   NA    1   NA
[4,]   NA   NA   NA    1
```

```
> Cm2
```

```
     [,1] [,2] [,3] [,4]
[1,]  1.0  0.5   NA   NA
[2,]  0.5  1.0   NA   NA
[3,]   NA   NA  1.0  0.5
[4,]   NA   NA  0.5  1.0
```

Note that the diagonal elements have to be set to one. If it is known that some test statistics are uncorrelated then the corresponding elements have to be set to zero.

2.2 Creating Graphs using the gMCP GUI

Alternatively one can specify graphs using the graphical user interface provided by gMCP.

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

These commands will load the the gMCP library and open the GUI interface a screenshot of which is presented in Figure 2. Graphs defined using this tool can be saved to an R object the name of which can be specified in the line above the graph manipulation window the graph is exported by setting the cursor into this line (optionally editing the variable name) and pressing enter. By default the variable name is set to createdGraph. This creates an object of class graphMCP.

After switching back to R's interpreter the created graph can be converted to a matrix using the command graph2matrix from package gMCP. The weights set for the graph can be extracted function graph2weights.

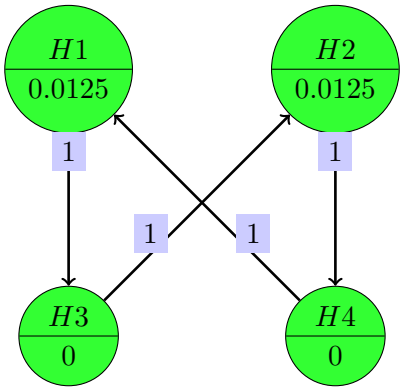

Figure 1: Graph corresponding to Examples 1 and 2 in [2]

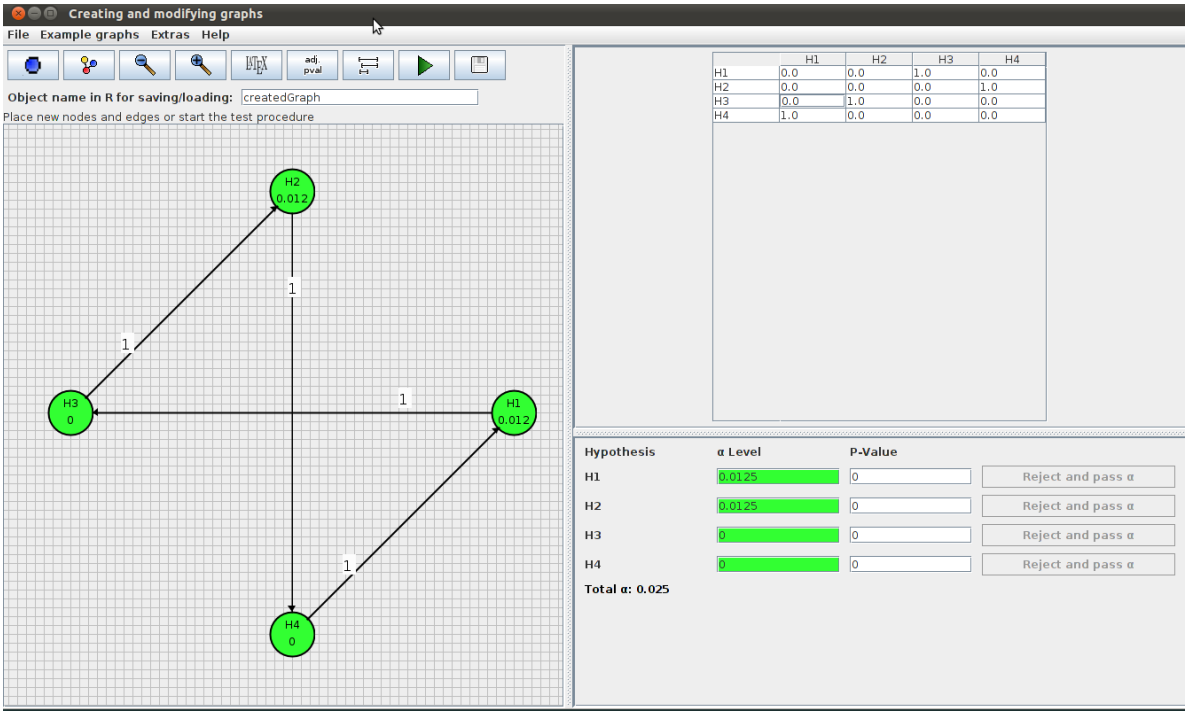

Figure 2: Screenshot of gMCP˜ GUI

```
> Gm <- graph2matrix(createdGraph)
> w <- graph2weights(createdGraph)
```

Conversely graphs defined as matrices can also be converted to `graphMCP` objects.

```
> G <- matrix2graph(Gm, alpha = 0.025 * w)
> graphGUI(G)
```

3 Testing

Performing tests using `gMCP` can be done in several ways. `gMCP` provides functions covering every step in the test procedure which can be done seperately or all together. The first step is the computation of all intersection hypotheses in the closure of the test problem together with conventional weights for the graphical approach without knowledge of any correlations. This can be done using the function `generateWeights`. For both of our examples this looks like:

```
> library(gMCP)
> generateWeights(Gm, w)

      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
 [1,]    0    0    0    1  0.0  0.0  0.0  1.0
 [2,]    0    0    1    0  0.0  0.0  1.0  0.0
 [3,]    0    0    1    1  0.0  0.0  0.5  0.5
 [4,]    0    1    0    0  0.0  1.0  0.0  0.0
 [5,]    0    1    0    1  0.0  1.0  0.0  0.0
 [6,]    0    1    1    0  0.0  0.5  0.5  0.0
 [7,]    0    1    1    1  0.0  0.5  0.5  0.0
 [8,]    1    0    0    0  1.0  0.0  0.0  0.0
 [9,]    1    0    0    1  0.5  0.0  0.0  0.5
[10,]    1    0    1    0  1.0  0.0  0.0  0.0
[11,]    1    0    1    1  0.5  0.0  0.0  0.5
[12,]    1    1    0    0  0.5  0.5  0.0  0.0
[13,]    1    1    0    1  0.5  0.5  0.0  0.0
[14,]    1    1    1    0  0.5  0.5  0.0  0.0
[15,]    1    1    1    1  0.5  0.5  0.0  0.0
```

`generateWeights` takes the graph defined as a matrix and the vector of initial weights and returns a matrix where each row corresponds to an intersection hypotheses in the closure of the test problem. The first half of each line indicates the intersection hypotheses. Hypotheses in the intersection are indicated by a 1, hypotheses not in the intersection are indicated by a 0. For example $(1, 1, 0, 0)$ would stand for the intersection between H_1 and H_2. The second half of the elements then provides weights for each hypothesis in the corresponding intersection.

In a next step critical values for all elementary hypotheses in each intersection hypothesis are computed. This can be done using the function `generateBounds`. Here for the first time the different assumptions on the correlation structure of Examples 1 and 2 are essential:

```
> generateBounds(Gm, w, Cm1, al = 0.025)

          [,1]     [,2]     [,3]     [,4]
 [1,]       NA       NA       NA 1.959964
 [2,]       NA       NA 1.959964       NA
 [3,]       NA       NA 2.241403 2.241403
 [4,]       NA 1.959964       NA       NA
 [5,]       NA 1.959964       NA      Inf
 [6,]       NA 2.241403 2.241403       NA
 [7,]       NA 2.241403 2.241403      Inf
 [8,] 1.959964       NA       NA       NA
 [9,] 2.241403       NA       NA 2.241403
[10,] 1.959964       NA      Inf       NA
[11,] 2.241403       NA      Inf 2.241403
[12,] 2.241403 2.241403       NA       NA
[13,] 2.241403 2.241403       NA      Inf
[14,] 2.241403 2.241403      Inf       NA
[15,] 2.241403 2.241403      Inf      Inf

> generateBounds(Gm, w, Cm2, al = 0.025)

          [,1]     [,2]     [,3]     [,4]
 [1,]       NA       NA       NA 1.959964
```

```
 [2,]       NA       NA 1.959964       NA
 [3,]       NA       NA 2.212125 2.212125
 [4,]       NA 1.959964       NA       NA
 [5,]       NA 1.959964       NA      Inf
 [6,]       NA 2.241403 2.241403       NA
 [7,]       NA 2.241403 2.241403      Inf
 [8,] 1.959964       NA       NA       NA
 [9,] 2.241403       NA       NA 2.241403
[10,] 1.959964       NA      Inf       NA
[11,] 2.241403       NA      Inf 2.241403
[12,] 2.212125 2.212125       NA       NA
[13,] 2.212125 2.212125       NA      Inf
[14,] 2.212125 2.212125      Inf       NA
[15,] 2.212125 2.212125      Inf      Inf
```

Alternatively we could transform these error bounds into p-values to recreate Table 2 of [2]

```
> (1 - pnorm(generateBounds(Gm, w, Cm2, al = 0.025))) * 100
```

```
          [,1]     [,2]     [,3]     [,4]
 [1,]       NA       NA       NA 2.500000
 [2,]       NA       NA 2.500000       NA
 [3,]       NA       NA 1.347901 1.347901
 [4,]       NA 2.500000       NA       NA
 [5,]       NA 2.500000       NA 0.000000
 [6,]       NA 1.250000 1.250000       NA
 [7,]       NA 1.250000 1.250000 0.000000
 [8,] 2.500000       NA       NA       NA
 [9,] 1.250000       NA       NA 1.250000
[10,] 2.500000       NA 0.000000       NA
[11,] 1.250000       NA 0.000000 1.250000
[12,] 1.347901 1.347901       NA       NA
[13,] 1.347901 1.347901       NA 0.000000
[14,] 1.347901 1.347901 0.000000       NA
[15,] 1.347901 1.347901 0.000000 0.000000
```

this function takes the graph in matrix form, the weights, the correlation matrix and the overall $\alpha -$ level as inputs in order to compute rejection bounds for all elementary hypotheses in each intersection. Bounds are computed using the multivariate distribution of test statistics whenever the whole correlation matrix is known for more than one test statistic associated with the elementary hypotheses within the intersection. For a concise explanation of the involved algorithm see [2, Section 3.2].

Once rejection bounds have been computed overall rejection of elementary hypotheses, based on actual data, needs do be determined using the closed testing principle. This means that all hypotheses which are rejected in all intersection hypotheses, of which they are a part of, are rejected at the overall $\alpha -$ level.

Since the bounds of a testing procedure is independent of the observed data `gMCP` provides a test function for that particular MTP:

```
> Example1 <- generateTest(Gm, w, Cm1, al = 0.025)
> Example2 <- generateTest(Gm, w, Cm2, al = 0.025)
```

The definition of a test function is efficient if a test is applied several times, for example in simulations.

The `myTest` function in the example above takes a vector of three z-scores and returns results in the form of a boolean vector wher `TRUE` stands for rejection of the null hypothesis and `FALSE` signifies that the null has to be retained.

```
> Example1(c(2.24, 2.24, 2.24, 2.3))

[1] FALSE FALSE FALSE FALSE

> Example2(c(2.24, 2.24, 2.24, 2.3))

[1]  TRUE FALSE FALSE FALSE
```

We see that above z-score scenario leads to an effective gain in power of the procedure when knowledge about the correlation structure is used.

*Attention:˜*Not all scenarios of partial knowledge of the correlation matrix can currently be handled by `gMCP`. The correlation matrix must have a block structure: We assume that the set of hypothesis can be partitioned into subsets such that the pairwise correlations in each subset are either known or are set to NA. For example assume that in the example given above not only correlations between the statistics for H_1 and H_2 are known but also the correlation between the statistics for H_1 and H_3 but not between those associated with H_2 and H_3. In this case it is unclear when testing the intersection hypotheses $H_1 \cap H_2 \cap H_3$ wether the bounds for the statistics associated with H_1 and H_2 or alternatively for those associated with H_2 and H_3 should be searched using their multivariate distribution. The current implementation would try to find bounds using the multivariate normal distribution of all three statistics which is not fully known and hence break and return an exception.

3.1 Testing using the gMCP interface

The function `gMCP` provides a common interface to both sequentially rejective Bonferroni procedures as well as parametric tests. `gMCP` takes objects of the type `graphMCP` as its input together with a vector of p-values and computes whether the according test procedure rejects. For Example 1 this amounts to the call:

```
> p <- 1 - pnorm(c(2.24, 2.24, 2.24, 2.3))
> G <- matrix2graph(Gm, w * 0.025)
> gMCP(G, p)

An object of class "gMCPResult"
Slot "graphs":
[[1]]
A graphMCP graph
Overall alpha: 0.025
H1 (not rejected, alpha=0.0125)
H2 (not rejected, alpha=0.0125)
H3 (not rejected, alpha=0)
H4 (not rejected, alpha=0)
Edges:
H1 -(1)-> H3
H2 -(1)-> H4
H3 -(1)-> H2
```

```
H4 -(1)-> H1


Slot "pvalues":
        H1         H2         H3         H4
0.01254546 0.01254546 0.01254546 0.01072411

Slot "rejected":
   H1    H2    H3    H4
FALSE FALSE FALSE FALSE

Slot "adjPValues":
        H1         H2         H3         H4
0.02509092 0.02509092 0.02509092 0.02509092
```

In the case of a sequentially rejective Bonferroni type procedure `gMCP` returns an object of class `graphMCP-Result` which holds information on the specific sequence the test procedure has taken through the graph and also provides adjusted p-values.

For Example 2 we use the same graph and assume normally distributed test statistics with known variances and a correlation matrix of the form:

$$\begin{pmatrix} 1 & \frac{1}{2} & \texttt{NA} & \texttt{NA} \\ \frac{1}{2} & 1 & \texttt{NA} & \texttt{NA} \\ \texttt{NA} & \texttt{NA} & 1 & \frac{1}{2} \\ \texttt{NA} & \texttt{NA} & \frac{1}{2} & 1 \end{pmatrix}.$$

This can be implemented in `gMCP` by additionally passing the correlation matrix to `gMCP`:

```
> gMCP(G, p, corr = Cm2)

An object of class "gMCPResult"
Slot "graphs":
list()

Slot "pvalues":
        H1         H2         H3         H4
0.01254546 0.01254546 0.01254546 0.01072411

Slot "rejected":
   H1    H2    H3    H4
 TRUE FALSE FALSE FALSE

Slot "adjPValues":
numeric(0)
```

which returns a similar result however without the graphs representing the rejection sequence, since the whole closed test is done in this example. A vector specifying which of the elementary hypotheses can be rejected at the overall $\alpha -$ level.

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, 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.

[3] G.˜Hommel, F.˜Bretz, and W.˜Maurer. Powerful short-cuts for multiple testing procedures with special reference to gatekeeping strategies. *Statistics in Medicine*, 26:4063–4073, 2007.