TreeGate macro, version 1.0
User manual

Thomas Brechenmacher
Dainippon Sumitomo Pharma Co., Ltd.

August 9, 2011

Contents

1 Introduction

This macro implements commonly used p-value based tree-gatekeeping procedures and allows users to perform adjusted p-values calculation as well as power computation. Gatekeeping procedures based on the following multiple comparison procedures are available:

- Bonferroni.
- Holm.
- Hochberg.
- Hommel.

An important feature of gatekeeping procedures is that logical restrictions can be specified to account for clinically relevant logical relationships among the null hypotheses of interest. The TreeGate macro implements logical restrictions that are formulated in terms of serial and parallel rejection sets (tree-gatekeeping procedures). More information on gatekeeping procedures and logical restrictions can be found in Brechenmacher et al. (2011) and Dmitrienko et al. (2007, 2008a, 2009 and 2011a, b, c).

Section 2 gives a detailed description of the arguments available with the TreeGate macro. Section 3 provides instructions on how to define the gatekeeping problem before calling the TreeGate macro and three clinical trial examples are given to illustrate the method. Section 4 describes the use of the macro to compute adjusted p-values. Section 5 and 6 discuss more advanced topics on power and familywise error rate (FWER) calculation.

2 Macro arguments

The TreeGate macro is called as follows:

```
%TreeGate(test=, gamma=, gaminf=, gamsup=, gamby=, exhaust=, alpha=, powout=,
rawp=, pvalout=);
```

Table 1 defines the common macro arguments, i.e., arguments that can be used for adjusted p-values calculation as well as power/FWER computation. Table 2 details the macro arguments specific to power/FWER computation.

Table 1. Common macro arguments.

Macro argument	Values	Default value	Description
`test`	`holm`, `hochberg`, `hommel`	`hommel`	Component procedure, i.e., multiple testing procedure used in the gatekeeping procedure. A truncated version of the component procedure is used in each family except the last one where the regular procedure is employed. More information on truncated procedure can be found in Dmitrienko et al. (2008b).
`gamma`	$0 \leq$ num < 1, `_all_`	`_all_`	Truncation parameter for each family except the last one. Values are separated by `#`. Ex: `gamma=0.5#0.9` for a gatekeeping problem with 3 families, where gamma1=0.5 and gamma2=0.9 (gamma3 is automatically set to 1). For `gamma=_all_`, results are calculated for gamma values from `gaminf` to `gamsup` and incremented by `gamby`.
`gaminf`	$0 \leq$ num < 1	`0`	Minimum value for gamma when `gamma=_all_`. Ex: for `gaminf=0.5`, results are calculated for values of gamma from 0.5 to `gamsup`.
`gamsup`	$0 \leq$ num < 1	`0.9`	Maximum value for gamma when `gamma=_all_`. Ex: for `gamsup=0.5`, results are calculated for values of gamma from `gaminf` to 0.5.
`gamby`	$0 \leq$ num < 1	`0.1`	Increment value for gamma when `gamma=_all_`. Ex: for `gamby=0.1`, `gaminf=0` and `gamsup=0.9`, results are calculated for values of gamma 0.0, 0.1, 0.2,. . . ,0.9.
`exhaust`	`yes, no`	`no`	If `exhaust=yes`, an alpha-exhaustive gatekeeping procedure is used instead of the regular gatekeeping procedure. Alpha-exhaustive gatekeeping procedures are described in Dmitrienko et al. (2011d).

Note: num refers to any numerical value.

Table 2. Macro arguments specific to power/FWER calculation.

Macro argument	Values	Default value	Description
`alpha`	$0 <$ num < 1	`0.025`	Global familywise error rate.
`powout`	`power`, `fwer`	`power`	Ouput type. `powout=power` requests power calculation and `powout=fwer` requests familywise error rate calculation.
`rawp`	dataset name, `_null_`	`_null_`	Specifies the dataset containing the raw p-values used to compute power/FWER. When `rawp=_null_`, no power/FWER calculation is performed.
`pvalout`	dataset name, `_null_`	`_null_`	Specifies the dataset to output all adjusted p-values calculated based on the raw p-values contained in dataset `rawp`. When `pvalout=_null_`, the adjusted p-values are not saved. Note that the running time of the macro is increased when adjusted p-values are requested.

Note: num refers to any numerical value.

3 Input of the gatekeeping framework

3.1 Dataset study

Before calling the TreeGate macro, the gatekeeping problem, i.e., the individual null hypotheses, hierarchical families and logical restrictions must be entered in the SAS dataset `study`. This dataset should contain all variables described in Table 3.

Table 3. Variables to be defined in the dataset `study`.

Variable name	Variable type	Description
hyp	character or numeric	Label for individual null hypothesis.
family	numeric	Family number.
parallel	character	Chain of Boolean indicators indicating which null hypotheses are in the parallel set of the current null hypothesis. The Boolean indicators are ordered based on the position of the corresponding null hypotheses in the dataset `study`. Ex: For three null hypotheses H_1, H_2 and H_3 and if H_1 and H_2 are both in the parallel set of H_3, the parallel variable for H_3 is set to `110`.
serial	character	Chain of Boolean indicators indicating which null hypotheses are in the serial set of the current null hypothesis. The Boolean indicators are ordered based on the position of the corresponding null hypotheses in the dataset `study`. Ex: For three null hypotheses H_1, H_2 and H_3 and if only H_2 is in the serial set of H_3, the serial variable for H_3 is set to `010`.

3.2 Examples

3.2.1 Diabetes trial example

This example was introduced in Dmitrienko et al. (2011a) to discuss the choice of efficient gatekeeping procedures. Consider a clinical trial in patients with Type II diabetes mellitus conducted to evaluate the efficacy and safety of two doses (Dose 1 and Dose 2) of an experimental treatment compared to an active control. The primary efficacy analyses in this trial include non-inferiority and superiority comparisons at each dose level. Let H_1 and H_2 denote the null hypotheses of inferiority for Dose 1 and Dose 2 respectively. Similarly, H_3 and H_4 are the null hypotheses of lack of superiority for Dose 1 and Dose 2 respectively. As discussed in Dmitrienko et al. (2011a), a possible gatekeeping strategy for this clinical trial is as presented in Figure 1.

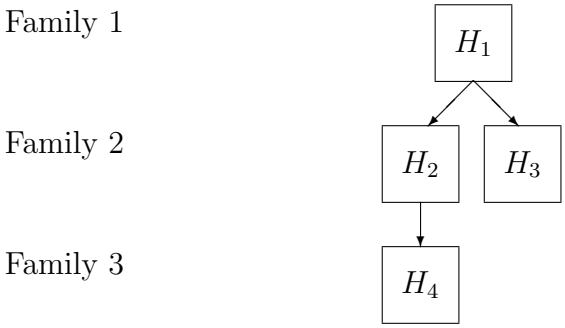

Figure 1. Decision rules in the diabetes trial example.

TThis gatekeeping problem is defined using the dataset `study` as follows:

```
data study;
    length hyp $20 family 8 parallel serial $50;
    input hyp $ family parallel $ serial $;
    datalines;
H1 1 0000 0000
H2 2 0000 1000
H3 2 0000 1000
H4 3 0000 1100
;
run;
```

The first line in the datalines statement indicates that H_1 is in family 1 and is not logically restricted by any null hypothesis. The second and third lines indicate that H_2 and H_3 are in family 2 and that H_1 is in their serial logical restriction set. The fourth line indicates that H_4 is in family 3 and that its serial restriction set is composed of H_1 and H_2. The parallel restriction set is empty for all null hypotheses.

3.2.2 Schizophrenia trial example

We now consider a more complex gatekeeping procedure with three families each composed of three null hypotheses. This example was discussed in Brechenmacher et al. (2011) to illustrate the Hommel-based gatekeeping procedure. Consider a clinical trial comparing three doses of an antipsychotic to a control for three ordered endpoints. The gatekeeping framework for this clinical trial is summarized in Figure 2.

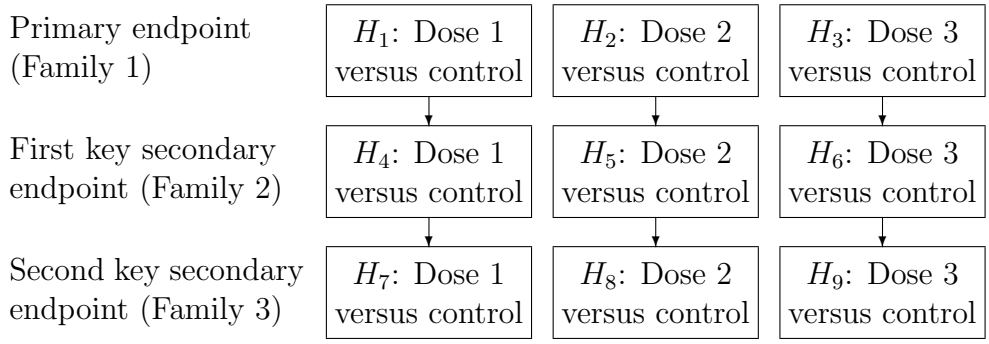

Figure 2. Decision rules in the schizophrenia trial example.

As can be seen from Figure 2, the test for one dose in a given family is only performed if that dose is shown to be superior to the control in the previous family. This gatekeeping problem is defined using the dataset `study` as follows:

```
data study;
    length hyp $20 family 8 parallel serial $50;
    input hyp $ family parallel $ serial $;
    datalines;
H1 1 000000000 000000000
H2 1 000000000 000000000
H3 1 000000000 000000000
H4 2 000000000 100000000
H5 2 000000000 010000000
H6 2 000000000 001000000
H7 3 000000000 100100000
H8 3 000000000 010010000
H9 3 000000000 001001000
;
run;
```

3.2.3 Hypertension trial example

This example was used in Brechenmacher et al. (2011) to illustrate the Hommel-based gatekeeping procedure. This example differs from the two other examples because it includes parallel logical restrictions. Consider a clinical trial in patients with hypertension in which the aim is to compare a new antihypertensive treatment to an active control with regard to four endpoints (P, S_1, S_2 and T). Although the primary comparison of this trial is non-inferiority, superiority is also tested conditionally upon

establishing non-inferiority for each endpoint. The gatekeeping framework for this clinical trial is illustrated in Figure 3.

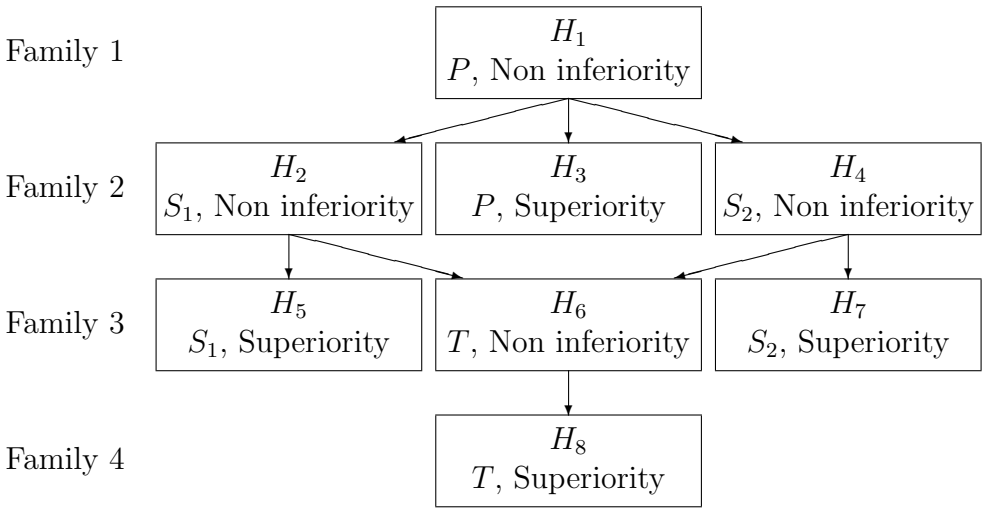

Figure 3. Decision rules in the hypertension trial example.

Only parallel restriction sets are defined in this trial (a serial rejection set consisting of only one null hypothesis is equivalent to a parallel rejection set). This gatekeeping problem is defined using the dataset `study` as follows:

```
data study;
    length hyp $20 family 8 parallel serial $50;
    input hyp $ family parallel $ serial $;
    datalines;
H1 1 00000000 000000000
H2 2 10000000 000000000
H3 2 10000000 000000000
H4 2 10000000 000000000
H5 3 01000000 000000000
H6 3 01010000 000000000
H7 3 00010000 000000000
H8 4 00000100 000000000
;
run;
```

4 Calculation of adjusted p-values

An important feature of the TreeGate macro is computation of adjusted p-values based on raw p-values pre-specified by the user. The first step in calculating adjusted p-values is to enter manually the corresponding raw p-values. This can be done easily by adding the variable *rawp* to the dataset `study`. Note that this variable should be numeric. In the hypertension trial example, the following raw p-values were obtained from an appropriate statistical test:

```
data study;
    length hyp $20 family rawp 8 parallel serial $50;
    input hyp $ family rawp parallel $ serial $;
    datalines;
H1 1 0.001 00000000 000000000
H2 2 0.008 10000000 000000000
H3 2 0.003 10000000 000000000
H4 2 0.026 10000000 000000000
H5 3 0.208 01000000 000000000
H6 3 0.010 01010000 000000000
H7 3 0.302 00010000 000000000
H8 4 0.578 00000100 000000000
;
run;
```

Once the p-values have been entered in the dataset `study`, the macro can be invoked with common macro arguments described in Section 2. In the context of the hypertension trial example, the following code requests the computation of adjusted p-values for the Hommel-based gatekeeping procedure with the truncation parameter in Family 2 gamma2=0.9 and Family 3 gamma3=0.9 (note that gamma1 for family 1 does not need to be specified since there is only one null hypothesis in Family 1 and thus gamma1 is set to 1).

```
%TreeGate(test=Hommel,gamma=1#0.9#0.9);
```

Adjusted p-values for the Bonferroni-based gatekeeping procedure can be computed using the following code (note that the Holm procedure is used in the last family):

```
%TreeGate(test=Holm,gamma=1#0#0);
```

In order to improve power of the gatekeeping procedure we can request an alpha-exhaustive gatekeeping procedure by setting `exhaust` to `yes`:

```
%TreeGate(test=Hommel,gamma=1#0.9#0.9,exhaust=yes);
```

Parallel alpha-exhaustive gatekeeeping procedures are described in Dmitrienko et al. (2011d) and can be easily generalized to the case of tree-gatekeeping procedures, i.e., intersections which contain null hypotheses from only one family are tested using the more powerful Hommel test instead of the truncated Hommel test.

Another convenient option of the macro is to perform calculations using different values of gamma. For example, to compute the adjusted p-values for the gamma ranging from 0.4 to 0.8 with an increment of 0.2, the following code can be used:

```
%TreeGate(test=Hommel,gaminf=0.4,gamsup=0.8,gamby=0.2);
```

Adjusted p-values calculated by the macro are presented in the SAS output window and are stored in the SAS dataset `adjp` in the work folder.

5 Power calculation

5.1 General considerations

A more advanced feature of the TreeGate macro is power calculation. As discussed in Brechenmacher et al. (2011) and Dmitrienko and al. (2011a), in order to select the most efficient gatekeeping procedure as well as the optimal values of gatekeeping parameters, e.g., truncation parameters, simulations can be performed to compare power of the procedures under consideration. To request power calculation, a dataset containing raw p-values obtained by simulations must be created. In this dataset, each row should correspond to one single simulation run and each column should contain raw p-values for one null hypothesis of interest. For example, to compute power based on 10,000 simulation runs in the schizophrenia trial example, a dataset of 10,000 rows and 9 columns containing the simulated raw p-values should be created.

The macro argument `rawp` allows users to specify the name of the dataset containing the simulated raw p-values. For example, the code below requests power calculation based on the raw p-values contained in the dataset `simulate`.

```
%TreeGate(test=Hommel,gamma=_all_,powout=power,rawp=simulate);
```

When requesting power calculation with `powout=power`, the default output of the macro is power for each individual null hypothesis. For example in the schizophrenia trial example, the code shown above will compute power for the 9 individual null hypotheses $H_1, \ldots, H_9$ with gamma from 0, 0.1, 0.2,..., 0.9 based on the Hommel-based gatekeeping procedure. The power values for each individual null hypothesis is also stored in the dataset `indpower`.

5.2 Power functions

In addition to power calculations for individual null hypotheses, the macro also supports power calculations based on complex power functions, see Brechenmacher et al. (2011) and Dmitrienko and al. (2011a). For example in the schizophrenia trial example, the following power function was used to determine the optimal values of the truncation parameters:

- Probability to reject at least two null hypotheses in Family 1 and at least one in Family 2.

Power functions can be defined in the dataset `powfunc` before executing the macro. This dataset should contain all variables defined in Table 4.

Table 4. Variables to be defined in the dataset `powfunc` (assuming n families and m null hypotheses).

Variable name	Variable type	Description
powlabel	character or numeric	Power function label. Power functions defined with the same label add up to form one power function.
weight	numeric	Power function weight (should lie between 0 and 1). A weight<1 should only be assigned for power functions that are defined with the same label.
F_i, $i = 1, \ldots, n$	character	Expected Number r_i of null hypotheses rejected in family i. If r_i is followed by the sign "-", power is calculated expecting exactly r_i null hypotheses rejected in family i. Otherwise, power is calculated expecting at least r_i null hypotheses rejected in family i.
H_i, $i = 1, \ldots, m$	character	If $H_i = 1$, the null hypothesis H_i is expected to be rejected when calculating power. For $H_i = 0$-, the null hypothesis is expected to be accepted when calculating power. For $H_i = 0$, no assumption is made about H_i.

Once power functions are defined in the dataset `powfunc`, the TreeGate macro is called using the macro arguments described in Section 2. The following three results are presented in the SAS output window:

- A summary of power functions defined in the dataset `powfunc`.
- Power results for the desired power functions.

- Power results for all individual null hypotheses.

Results are also stored in the datasets `power` (power functions) and `indpower` (power for individual null hypotheses).

5.3 Examples

5.3.1 Diabetes trial example

As discussed in Dmitrienko and al. (2011a), an optimal value for the truncation parameter in Family 2 can be chosen by maximizing one of the two power functions:

- Exceedence criterion: Probability to reject H_1, at least one null hypothesis in Family 2 and H_4.
- Weighted exceedence criterion: $0.7P$(reject H_1, H_2, H_3 and H_4) + $0.3P$(reject H_1, H_2 and H_4, accept H_3)

Due to the logical relationships, the first power function reduces to the probability of rejecting H_4 and thus it does not provide a good trade-off between the procedures in Family 2 and 3. The second power function is obtained by partitioning the event "reject at least one null hypothesis in Family 2" into two events and assigning weights corresponding to their relative importance. The two power functions are defined in the dataset `powfunc` as follows:

```
data POWfunc;
    input powlabel $1-19 weight F1 $ F2 $ F3 $ H1 $ H2 $ H3 $ H4 $;
/*label*/          /*Weight*/ /*Power:*/   /*Power: individual*/
                               /*Family*/   /*null hypotheses*/
    datalines;
Exceedence            1        0 1 0       1 0 0  1
Weighted exceedence   0.7      0 0 0       1 1 1  1
Weighted exceedence   0.3      0 0 0       1 1 0- 1
;
run;
```

Once the raw p-values are obtained through simulations and stored in the dataset `myrawp` (the SAS code is given in the appendix), the TreeGate macro is called:

```
%TreeGate(test=Hochberg,gamma=_all_,rawp=myrawp);
```

5.3.2 Schizophrenia trial example

In order to select optimal values of the truncation parameters in the schizophrenia trial example, the following three power functions are considered:

- Power function 1: Probability to reject at least two null hypotheses in Family 1 and at least one in Family 2.
- Power function 2: Probability to reject at least two null hypotheses in Family 1, at least two in Family 2 and at least one in Family 3.
- Power function 3: Probability to reject at least two null hypotheses in Family 1 including H_3 and reject at least H_6 in Family 2.

Power functions are defined in the dataset `powfunc` as shown below.

```
data POWfunc;
    input powlabel $1-10 weight F1 $ F2 $ F3 $ H1 $ H2 $ H3 $
          H4 $ H5 $ H6 $ H7 $ H8 $ H9 $;
/*label*/   /*Weight*/ /*Power:*/   /*Power: individual*/
                        /*Family*/   /*null hypotheses*/
    datalines;
Function 1     1         2 1 0       0 0 0 0 0 0 0 0 0
Function 2     1         2 2 1       0 0 0 0 0 0 0 0 0
Function 3     1         2 1 0       0 0 1 0 0 1 0 0 0
;
run;
```

In order to compute the power for the three power functions, 10,000 simulations are run based on the study design assumptions described in Brechenmacher et al. (2011). The `simul` macro given in the Appendix can be used to obtain a dataset with 10,000 rows each containing raw p-values for the 9 null hypotheses of interest.

```
%simul(n_Htest=3,n_fam=3,n_subj=120,n_sim=10000,out=myrawp,seed=1234,
       eff_size=0.3 0.4 0.7#0.2 0.3 0.5#0.1 0.2 0.3,
       sigma=1 0.8 0.4#0.8 1 0.3#0.4 0.3 1);
```

Finally, to compute power based on the Hommel-based gatekeeping procedure, the TreeGate macro is invoked:

```
%TreeGate(test=Hommel,gamma=_all_,rawp=myrawp);
```

6 FWER calculation

6.1 FWER definition

In some cases, it may be desirable to verify through simulations whether the FWER, i.e., the probability of erroneously rejecting at least one true null hypothesis, is controlled at the nominal level α under different sets of assumptions. This can be done by setting the argument `powout` to `FWER`, e.g.,

```
%TreeGate(test=Hommel,powout=FWER,rawp=rawp);
```

Results are presented in the SAS output window and stored in the dataset FWER.

Similarly to power calculation, FWER calculation requires that the user specify a definition for the FWER. For example, weak control of the FWER is achieved when all null hypotheses are true. A more desirable definition is strong FWER control which ensures the FWER to be protected under any configuration of the true and false null hypotheses. The FWER definition needs to be entered in the dataset `fwerdef` which should contain all variables defined in Table 5.

Table 5. Variables to be defined in the dataset `fwerdef` (assuming m null hypotheses).

Variable name	Variable type	Description
H_i, $i = 1, \ldots, m$	character	When calculating the FWER, the null hypothesis H_i is expected to be false if $H_i = 1$ and true if $H_i = 0$.

6.2 Example

Suppose that we want to compute the FWER for the following two scenarios in the schizophrenia trial example:

- Scenario 1: All null hypotheses are true.
- Scenario 2: All null hypotheses are true except H_3 and H_6.

We begin with Scenario 1 and create the dataset `fwerdef` as follows:

```
data FWERdef;
    input H1 H2 H3 H4 H5 H6 H7 H8 H9;
    datalines;
0 0 0 0 0 0 0 0 0
;
run;
```

Raw p-values are then generated through simulations using the `simul` macro given in the Appendix:

```
%simul(n_Htest=3,n_fam=3,n_subj=120,n_sim=100000,out=myrawp,seed=1234,
       eff_size=0 0 0#0 0 0#0 0 0,
       sigma=1 0.8 0.4#0.8 1 0.3#0.4 0.3 1);
```

Note that all effect sizes are set to 0 to reflect the fact that all null hypotheses are true. Finally, we call the TreeGate macro with option `powout=FWER`:

```
%TreeGate(test=Hommel,gamma=0.5#0.9,powout=FWER,rawp=myrawp);
```

For scenario 2, we proceed in the same way and create the dataset `fwerdef` as shown below.

```
data FWERdef;
    input H1 H2 H3 H4 H5 H6 H7 H8 H9;
    datalines;
0 0 1 0 0 1 0 0 0
;
run;
```

Raw p-values are simulated using the `simul` macro:

```
%simul(n_Htest=3,n_fam=3,n_subj=120,n_sim=100000,out=myrawp,seed=1234,
       eff_size=0 0 0.7#0 0 0.9#0 0 0,
       sigma=1 0.8 0.4#0.8 1 0.3#0.4 0.3 1);
```

Note that all effect sizes are set to 0 except for null hypotheses H_3 and H_6. Finally, the same SAS code as in Scenario 1 is used to call the TreeGate macro.

References

[1] Brechenmacher, T., Xu, J., Dmitrienko, A., Tamhane, A. C. (2011). A mixture gatekeeping procedure based on the Hommel test for clinical trial applications. *Journal of Biopharmaceutical Statistics*. 21, 748–767.

[2] Dmitrienko, A., Wiens, B.L., Tamhane, A.C., Wang X. (2007). Tree-structured gatekeeping tests in clinical trials with hierarchically ordered multiple objectives. *Statistics in Medicine*. 26, 2465–E478.

[3] Dmitrienko, A., Tamhane, A.C., Liu, L., Wiens, B.L. (2008a). A note on tree gatekeeping procedures in clinical trials. *Statistics in Medicine*. 27, 3446–3451.

[4] Dmitrienko, A., Tamhane, A.C., Wiens, B.L. (2008b). General multistage gatekeeping procedures. *Biometrical Journal*. 50, 667–677.

[5] Dmitrienko, A., Tamhane, A.C., (2009). Gatekeeping procedures in clinical trials. *Multiple Testing Problems in Pharmaceutical Statistics*. Dmitrienko, A., Tamhane, A.C., Bretz, F. (editors). Chapman and Hall/CRC Press, New York.

[6] Dmitrienko, A., Millen, B. A., Brechenmacher, T., Paux, G. (2011a). Development of gatekeeping strategies in confirmatory clinical trials. *Biometrical Journal*. To appear.

[7] Dmitrienko, A., Tamhane, A. C. (2011b). Mixtures of multiple testing procedures for gatekeeping applications in clinical trials. *Statistics in Medicine*. 30, 1473–1488.

[8] Dmitrienko, A., Tamhane, A. C. (2011c). Theory of mixture gatekeeping procedures for clinical trials. In preparation.

[9] Dmitrienko, A., Kordzakhia, G., Tamhane, A.C. (2011d). Multistage and mixture gatekeeping procedures in clinical trials. *Journal of Biopharmaceutical Statistics*. 21, 726–747.

Appendix

Raw p-values simulation in the diabetes trial example

The SAS code to generate raw p-values for the diabetes trial is given below. 10,000 simulations are performed assuming a multivariate normal distribution for the test statistics and under the study design assumptions described in Dmitrienko et al. (2011a) in Section 6.

```
proc IML;

    /*Define the mean vector*/
    mu= {0.35 0.35}‘;

    /*Define correlation matrix*/
    varz=j(2,2,1);
    varz[1,1]=1;varz[1,2]=0.5;
    varz[2,1]=0.5;varz[2,2]=1;
```

```
    /*generate the test statistics vector*/
    CALL vnormal(Z,(sqrt(480/2))*(1/1.2)*mu,varZ,10000,5963);
    Z2=j(10000,4,1);
    Z2[,1]=Z[,1];Z2[,2]=Z[,2];
    Z2[,3]=Z[,1]-0.15/(1.2*sqrt(2/480));
    Z2[,4]=Z[,2]-0.15/(1.2*sqrt(2/480));

    /*Corresponding raw p-values*/
    myrawp=1-probnorm(Z2);

    create myrawp from myrawp;
        append from myrawp;
quit;
```

Raw p-values simulation in the schizophrenia trial example

The `simul` macro generates raw p-values for clinical trials with several families each containing the same number of null hypotheses. Simulations are performed assuming a multivariate normal distribution for the test statistics. The `simul` macro is called as:

```
%simul(n_Htest=, n_fam=, n_subj=, eff_size=, sigma=, n_sim=, seed=, out=);
```

Arguments of the `simul` macro are described in Table 6.

Table 6. Arguments of the `simul` macro.

Macro argument	Description
`n_Htest`	Number of null hypotheses to be tested in each family.
`n_fam`	Number of families.
`n_subj`	Number of subjects in each group (only balanced designs).
`eff_size`	Effect sizes. Effect sizes for each family are separated by `#` and ordered by group within a family.
`sigma`	Correlation matrix. Correlation coefficients for each family are separated by `#` and ordered by family within a family.
`n_sim`	Number of simulations to be performed.
`seed`	Seed to be used for the random vector generating the test statistics. If seed=0 then the system time clock is used.
`out`	Name of the SAS dataset to contain the resulting raw p-values.

For example, consider a clinical trial with 2 families each composed of 2 null hypotheses. Effect sizes corresponding to the first and second null hypotheses in the first family are 0.4 and 0.6, respectively. Similarly, 0.2 and 0.3 are the effect sizes in the second family. A correlation of 0.8 is assumed between family 1 and 2 and a sample size of 120 subjects per group is planned. The `simul` macro is employed to simulate 10,000 raw p-values as shown below.

```
%simul(n_Htest=2, n_fam=2, n_subj=120, eff_size=0.4 0.6#0.2 0.3, sigma=1
0.8#0.8 1, n_sim=10000, seed=1234, out=output);
```

The SAS code for the `simul` macro is given below.

```
%macro simul(n_Htest=,n_fam=,n_subj=,eff_size=,sigma=,n_sim=,seed=1234,out=);
  proc IML;
    /*Shape the correlation matrix and effect-size vector*/
    sigma=tranwrd("&sigma","#",",");
    mu=tranwrd("&eff_size","#"," ");
    call symput('sigma2',sigma);
    call symput('mu',mu);
    free sigma mu;

    /*Define the effect-size vector*/
      mu={&mu}`;

    /*Define the covariance matrix of the group mean vector*/
    sigma= (1/&n_subj)*block(%do b=1 %to &n_Htest;{&sigma2},%end;{&sigma2});

    /*Define the contrast matrix*/
    contrast=j(&n_Htest*&n_fam,(&n_Htest+1)*&n_fam,0);
    do block=1 to &n_fam;
      do ligne=1 to &n_Htest;
        contrast[&n_Htest*(block-1)+ligne,block+ligne*&n_fam]=1;
        contrast[&n_Htest*(block-1)+ligne,block]=-1;
      end;
    end;

    /*Define the covariance matrix of test statistics*/
    varZ=(&n_subj/2)*contrast*sigma*contrast`;

    /*generate the test statistic vector*/
```

```
    CALL vnormal(Z,(sqrt(&n_subj/2))*mu,varZ,&n_sim %if &seed ne 0 %then ,&seed;);
    /*p-values*/
    rawp=1-probnorm(Z);

    create &out from rawp;
    append from rawp;
  quit;
%mend;
```