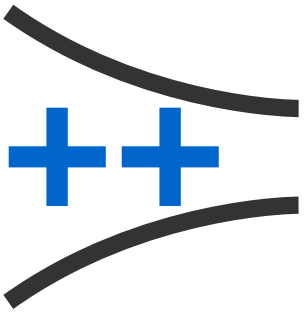


GroupSeq++

Group Sequential designs for genetic studies

Roman Pahl

Institute of Medical Biometry and Epidemiology
Philipps-Universität Marburg
Germany

May 2, 2013

Contents

List of Tables

List of listings

1 Introduction

This *package* is used to construct group sequential designs for the analysis of genome-wide association studies (GWAS). Two basic tools are provided for the construction of:

1. Optimized Multi-stage Designs (OMD) of up to five stages either minimized for the cost or maximized for the power of the study, respectively. A large number of parameters relevant for application in practice are incorporated, such as limited study budget, different gene chips (including rebate functions), varying case fractions, lower and upper bounds on sample sizes, and many more.

2. Flexible two-stage designs (CRP-Tool), enabling flexible marker selection after the first stage, that is, not only based on the statistical outcome alone but also upon biological or any other criteria, respectively. In addition, it allows the first interim analysis (stage one) to be conducted at any given time during the course of the project and to modify the initially planned sample size based on the outcome of that analysis, respectively.

The entire functionality is provided in a graphical user interface (GUI), which is started right after the package has been loaded into R, showing the main selection dialog:[1]

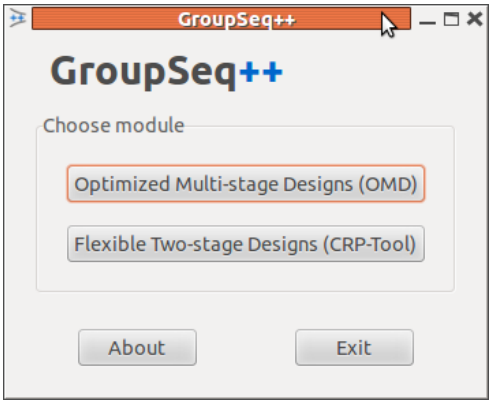


Figure 1: The GroupSeq++ window at program start (Ubuntu 12.04.1, GNOME 3.4.1).

Most of the computationally demanding calculations have been implemented in a C++ library, which is executed in the background using software mulithreading. As such, the R-console is not "blocked" during computation, but can be used as normal by the user[2].

[1] Depending on your operating system, the GUI as presented here and in all following screenshots may look different from yours in both style and colors.

[2] By default the R-console, from where GroupSeq++ has been called, is printing status/results during the computation, which of course may interfere with any work done simultaneously in that console. However, the GUI provides an option to turn this off.

1.1 IMPORTANT NOTE for Windows users

On computers running Windows, the GUI may freeze at times if a GroupSeq++ window is moved or resized. Once this did happen, tabbing into a different context (i.e., press and release Alt+Tab) usually does unfreeze the window again. If you observe this with your installation, it is recommended to avoid resizing/moving the GroupSeq++ windows, which does not affect the overall functionality by much.

2 Optimized Multi-stage Designs (OMD)

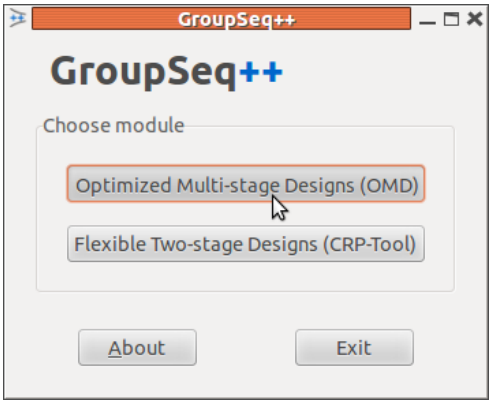


Figure 2: Selecting the OMD module from the main menu.

The OMD module enables the construction of optimized multi-stage designs for genome-wide case-control association studies (GWAS). Such multi-stage designs are used for a stage-wise but joint analyis (Skol *et al.*, 2006) of GWAS with stepwise reduction of the marker set after each stage. Given an initial setup, the program determines both an optimal allocation of the samples and an optimal choice of critical bounds for each of the different stages while still controlling the pre-specified type I error risk. The resulting optimized multi-stage design is more economical than a simple single-stage study design. The optimization can be performed for different objectives and various study parameters and constraints that typically occur in practice (see also the list of features in section 2.2).

2.1 How to read the OMD manual

For most users it is recommended to read the manual from start to end. If you are already familar with multi-stage designs in GWAS, however, you may want to skip section 2.3. Section 2.3.4 presents some valuable thoughts about the workflow when optimizing multi-stage designs. You may also skip section 2.4 about the "Parameters and options", if you instead try to start working with the GUI right away by just using the integrated tooltips. For a very quick start, you can jump to the end and check out the use cases beginning at section 2.6.

2.2 Features overview

- Optimized multi-stage design of up to 5 stages
  - Two optimzation objectives
    - Minimize study cost for desired power
    - Maximize study power for some limited budget
  - Optimal allocation of samples to the different stages
  - Optimal significance levels (while always maintaining the type I error) for marker selection at each stage including the final significance level for the final test decision
  - Optimal choice of gene chips from a pre-specified set of chips as available in practice with the possibility to specify individual discount prices depending on the number of ordered pieces
  - Optimal sample size
  - Expected number of markers per stage
  - Stage-wise cumulative type I error and power, respectively
  - On-the-fly output of currently found optimum
- Study parameters
  - Expected risk allele frequency and odds ratios of disease marker
  - Case-control fraction (possibly varying over different stages to model imbalanced study recruitment)
  - Additional cost offset per subject such as recruiting or phenotyping cost (if desired, cost can be specified individually for each subject)
  - Allowing for initial "cost-free subjects", for example, from earlier studies to construct follow-up designs
- Optimization parameters
  - Choose between three different optimization algorithms or combine them in a chain
  - Set various constraints prior to optimization
    - Maximal total sample size
    - Minimal stage size
    - Lower and/or upper sample size bounds at each stage
    - Gene chip to be used at a stage
  - Tweak optimization (advanced users)
    - Cap number of function evaluations in search process

* Maximum runtime
* Convergence tolerance
* Constraint tolerance

2.3 Method overview

2.3.1 GWAS

Genome-wide association studies (GWAS) are widely used to detect genes involved in complex diseases. Typically a huge number of, say M, genetic markers is genotyped at a large sample of, say N, subjects, which are divided into cases and controls with respect to some trait or disease. Based on the null hypothesis H_0 that no marker is associated with the trait, a statistical test is performed for each marker in order to detect a significant difference of genotype frequencies in cases versus controls. Particularly, for each marker, a test statistic summarizing the difference between cases and controls is computed and is matched against a prespecified significance threshold. Usually the threshold is chosen such that the genome-wide type I error is controlled at 5%.

2.3.2 The multi-stage framework in genomic studies

Although genotyping costs have dropped considerably over the past years, they are still a concern in GWAS, especially since not only the number of markers but also the sample sizes have been constantly increasing in order to hunt for smaller and smaller genetic effects. The multi-stage framework can be used to address this problem, because it enables cost efficient study designs. The basic idea is to save genotyping costs by performing a stage-by-stage analysis in which the set of markers will not be genotyped for all subjects at once. Instead all subjects are subdivided into several groups at the beginning of the study and the genotyping is distributed over these groups while at the same time the marker set is cut down at each stage (figure 3).

2.3.2.1 Design construction Let S be the number of groups and $n_1, \ldots, n_S$ the number of subjects per group, thus $N = \sum_{i=1}^{S} n_i$. Each group defines a stage. In stage one, all $M_1 = M$ markers are genotyped at the n_1 subjects and tested against the stage one significance threshold. Only those markers showing significant association with respect to this first threshold are carried forward to the second stage[3]. The number of markers surviving a stage is typically very small, often less than 1%. In the second stage only the remaining $M_2 \subset M_1$ markers having survived the cut after stage 1 are genotyped at the n_2 subjects and tested against the stage 2 threshold, for which the genotype data of both stages (1 and 2) is combined in a joint analysis. Similarly, the set of markers $M_3 \subset M_2$ having survived stage 2, are genotyped at n_3 subjects and tested using all relevant data of

[3] More details about testing and marker selection are given in section2.3.3

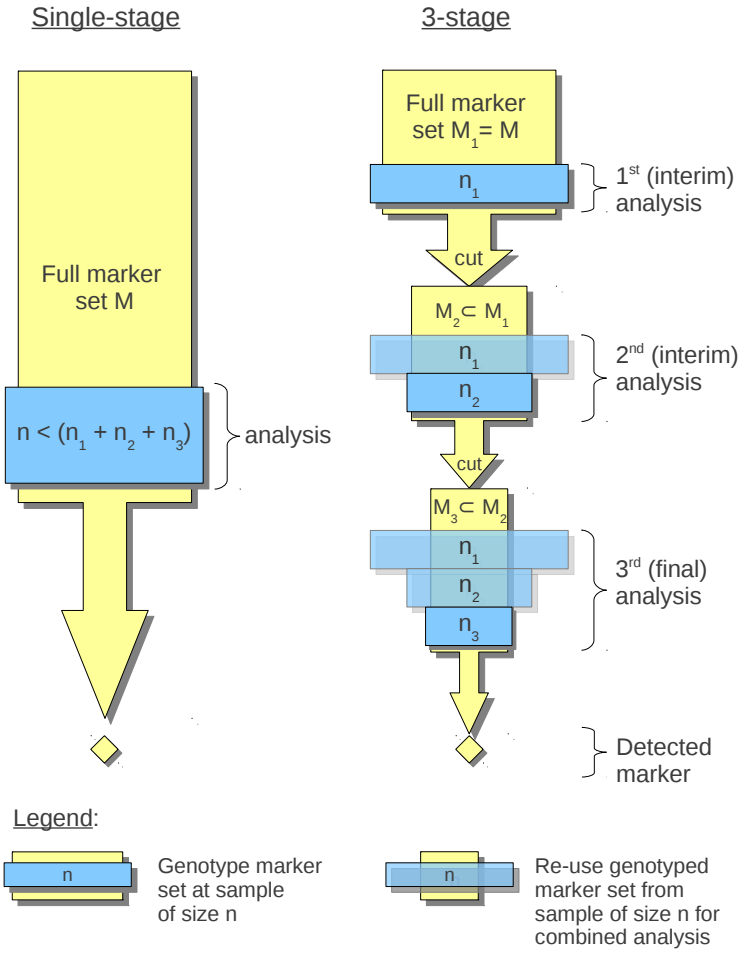


Figure 3: Standard single stage design vs. 3-stage design

stages 1, 2, and 3 cumulated up to this point (figure 3). Depending on the total number of stages, these steps are repeated until all stages have been processed. For each marker surviving all stages, the null hypothesis is rejected and the marker is declared to show significant association with the trait.

2.3.2.2 Design cost The cost of a GWAS for the most part consist of collecting the data, namely both the phenotype and the genotype data[4]. To simplify matters, we assume that the cost of phenotyping (CP) a single individual is the same for each. In contrast, the genotyping cost per individual is defined by the gene chip used to genotype the set

[4]There might be cost with respect to other aspects, for example, statistical analysis, data management, and so on. Here we assume these to be negligible small in comparison with the total cost to keep matters simple.

of markers. Since the set of markers shrinks with each stage of a multi-stage design, the applied gene chip may change as well. At stage 1 typically a "standard" gene chip is used, which contains the full marker set under investigation. After that, gene chips customizable for a specifically desired marker set are required. That is, depending on the markers selected as a result of the interim analysis after stage 1, the researcher is going to order customized gene chips, which just contain the set of selected markers. Of course, the customized chip carrying fewer markers should be lower in cost than the "standard" chip to be worthwhile. Also it is reasonable to assume that there is only one single type of chip used at a particular stage. The type of gene chip is defined by its capacity, that is, the number of markers placed at the chip. Formally, let $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ be the monotonely increasing sorted sequence of available capacities, that is, $i \leq j \iff c_i \leq c_j$. Furthermore, let $Pr(c_i)$ denote the price of the chip with capacity c_i, $i \in 1, \ldots, n$, respectively. In practice there is usually a rebate with respect to the number of ordered chips. We therefore extend the price function to $Pr(c_i, q)$, which is the price of the chip with capacity c_i when ordered in quantity $q \in \mathbb{N}$. Now for any two capacities c_i and c_j we expect the following monotonicity condition to hold:

$$c_i \leq c_j \iff Pr(c_i, q) \leq Pr(c_j, q) \tag{1}$$

In other words, the prices of gene chips are proportional to their capacity, which in practice usually does not follow a linear relationship as there is typically also a rebate regarding the number of markers placed at the chip, or formally, the ratio $Pr(c_i, q)/c_i$, which is nothing else than the "price per marker", is decreasing with an increasing capacity. In total, the cost of a multi-stage design with S stages are then computed as

$$\sum_{i=1}^{S} n_i \left(CP + Pr(c_i, n_i)\right) \tag{2}$$

2.3.3 Statistical analysis and marker selection

The program at hand computes and optimizes multi-stage designs and, with that, provides stage-wise two-sided nominal significance thresholds $\alpha_1, \alpha_2, \ldots, \alpha_S$. Then for each marker, the probability to survive stage s can be computed as

$$\int_{c_s}^{\infty} \cdots \int_{c_2}^{\infty} \int_{c_1}^{\infty} f_{\mu,\Sigma}\left(x_1, \ldots, x_s\right)\, dx_1\, dx_2 \cdots dx_s + \int_{\infty}^{-c_s} \cdots \int_{\infty}^{-c_2} \int_{\infty}^{-c_1} f_{\mu,\Sigma}\left(x_1, \ldots, x_s\right)\, dx_1\, dx_2 \cdots dx_s$$

where f is the multivariate normal distribution of the (asymptotically) normally distributed test statistics $T_1, T_2, \ldots, T_s$ obtained at the corresponding stages. The integration bounds are computed as $c_i = \Phi^{-1}(1 - \alpha_i/2)$. Figure 4 provides a schematic presentation of the integrated area for the case of two stages.

Selecting markers As soon as all subjects at some stage i are genotyped, for each marker, the researcher computes an appropriate test statistic such as Armitage's test for

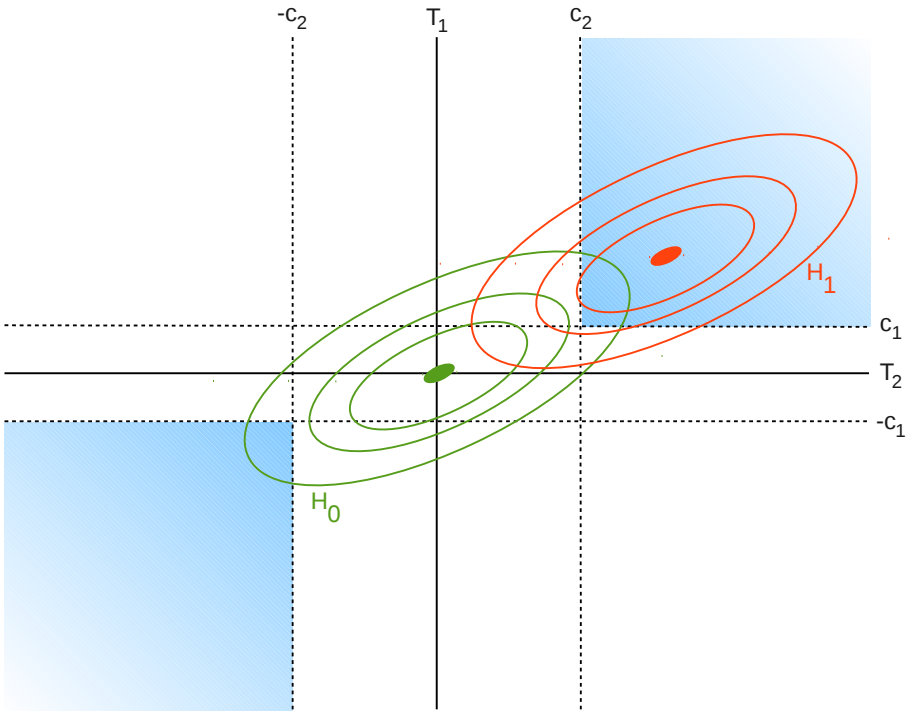


Figure 4: Schematic display of the integrated area (blue) of a two-stage design under H_0 (green) and under H_1 (red).

trend (Sasieni, 1997) and derives a corresponding two-sided p-value p_i under H_0. For example, if the test statistic is standard normal distributed under H_0 (i.e. $T \sim N(0,1)$), then $p_i = 2 \cdot (1 - \Phi(|t_i|))$, where t_i is the test statistic observed at stage i. It is important to note that the test statistic at stage i is computed by using *all* data of the particular marker that has been accumulated *up to* this stage and not only the data gained at the particular stage. This results in a joint analysis (Skol *et al.*, 2006) and can be seen as kind of updating the marker's statistic stage by stage using the additionally gained data. A marker is carried over to the next stage if and only if the following two conditions apply:

1. $p_i < \alpha_i$
2. Equal signs of the test statistics at each stage (i.e., $\text{sign}(t_1) = \text{sign}(t_2) \ldots = \text{sign}(t_i)$)

The second point basically implies that the disease causing (or risk) allele is not changing over the stages.

2.3.4 Design optimization

2.3.4.1 General notes First of all there is no easy standard method to obtain the most optimal multi-stage design in any situation. Apart from practical considerations and constraints, optimizing multi-stage designs, especially complex designs of three or more stages, requires kind of a supervised step-wise approach in order to get a good result. Thus, unless being coped with an easy 2-stage design problem, you will not be able to enter your setup, press start and let OMD do all the work for you. Instead, OMD should be rather considered a tool that helps in the search and construction of efficient multi-stage designs.

2.3.4.2 Workflow In general one should start with a single-stage design. In doing so, unrealizable parameters can be detected right at the start. For example, if the single-stage design does not yield the desired power for a given parameter set, a multi-stage design never will, because multi-stage designs are always (although only slightly) worse in power than single-stage designs, basically due to their multiple analyses at the different stages, statistically resulting in a slight loss of efficiency. On the other hand, if the given parameter setup does yield a valid single-stage design, one proceeds with the most simple multi-stage design possible, a 2-stage design. Once an optimal 2-stage design is found, the number of stages can be increased gradually. This allows monitoring the added benefit of extending the design, which is important in practice, because, for each additional stage, the extra benefit should exceed the extra effort of conducting the additional stage (Pahl *et al.*, 2009). The gradual approach moreover allows for gradual validation of the optimization results.

2.3.4.3 Gradual validation The optimum of a lower-staged design basically serves as a lower bound for a design with more stages. Using the gradual approach, non-optimal results can be catched this way. Thus, as soon as a multi-stage design did not result in a better optimum after a stage had been added, the search algorithm might not have succeeded to find the true global optimum. Besides that, often at some point, adding a stage will show no further benefit anyway. This kind of "over-staging" effect is mainly ascribed to the discrete nature of the optimization problem caused by the utilization of marker chips, which provide marker sets of only a few defined capacities. To see this, consider a simplified example: Let the available chipset consist of two chips C_1 (\$10) and C_2 (\$1) with a capacity of 500,000 and 1 marker(s), respectively. Now for some stage, unless the set of markers has been shrinked to less than ten markers, the most efficient solution will be to take chip C_1. Since a marker set thus small at most is expected to appear at the very last stage, shrinking the marker set "in between" will show no additional benefit cost-wise, and inserting more stages hence at some point most likely will not improve the objective anymore. The amount of optimization being possible hence depends to some degree on the variety of different marker chips, and, as a rule of thumb, the number of different available chips should at least exceed the target number of stages. A good indication for over-staging is when the optimized design is using the same gene

chip at two (or more) consecutive stages. Another indicator are very "small" stages, that is, stages that use comparatively few subjects in comparison to the rest of the stages. Often these two indicators fall together. Section 2.6 provides an exemplary use-case on how the above described workflow is applied in practice.

2.3.4.4 Cross validation Cross validation is done by re-optimization after constraint and objective have been swapped. For example, imagine the study budget was fixed at \$6,000,000 and the optimized multi-stage design yields a power of 82%. For cross validation the power is fixed at 82% and the design re-optimized for minimal costs, which should result in costs of about those \$6,000,000. Should the cost significantly deviate from this, one of both searches did not find the true global optimum in the first place. For example, if the second optimization results in costs of only \$5,000,000 (while maintaining the power of 82%), the first design does not exploit the budget to a full extent. To solve this problem, one could re-start the initial optimization but this time using the results of the second run as a starting point. For an example see section 2.7.1.1.

2.4 Parameters and options

This section explains the standard program parameters and options one by one and with to some extent comprehensive descriptions in the context of GWAS analysis. A short description of each parameter can be always obtained by using the GUI's tooltips, which are accessed by moving the mouse cursor over the corresponding field. Starting the OMD-module initially shows the window as depicted in figure 5 except the advanced options, which are hidden by default but can be unfolded at any time by clicks on the corresponding expander triangles. For the sake of completeness, here and in the following we depict the advanced options as well. Basically, the major part contains several fields of input parameters and options, which are divided into the four tabs Study, I/O, Optimization, and Result. If a field requires the user to manually enter some value, any invalid entry is marked immediately with a red error-symbol to the right.

The optimization is started with the large Start-button located below the tab, and at the very bottom the user will get status updates during the optimization process.

2.4.1 Study

On a general note there are a total of five mandatory parameters without default values, which therefore must be always set by the user. Four of them are located in the study tab, which can be seen by the initially appearing red error-symbols (see figure 5).

Markers

- Number of markers: total number of genetic markers (or SNPs) at the start of the study

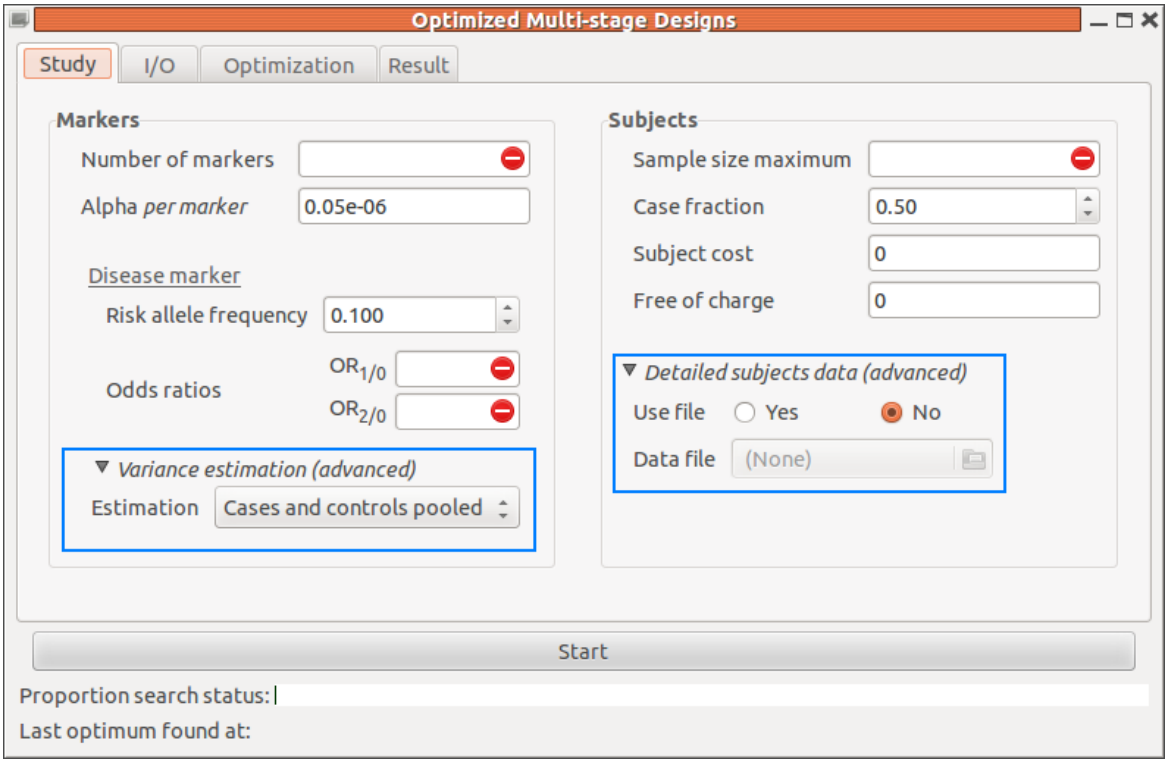


Figure 5: The OMD Study tab at program start with unfolded advanced options (blue rectangles)

- Alpha *per marker*: The program expects the user to set a marker-wise significance level, say α_m, which usually must be chosen such that the genome-wide type I error is controlled at $\alpha = 5\%$. The choice of this parameters will have a great impact on the outcome, the smaller α_m, the more costly and the lower the power of the study, respectively. One possibility is to take the Bonferroni approach and set $\alpha_m = \alpha/M$, with M being the total number of markers, which basically corrects α by the number of tests. However, this approach assumes complete independence between all test (i.e., all markers), which in practice is never the case, and thus forms the most conservative approach. Alternatively, the researcher may want to estimate the effective number of independent tests M_{eff} (Cheverud, 2001), which results in a relaxed significance level $\alpha_m = \alpha/M_{\text{eff}}$. The idea behind the effective number of independend test is to reflect the correlations between markers, which is caused by linkage disequilibrium and does lead to dependencies between the markers and thereby the tests. A widely accepted estimate for the total number of tests regarding the entire genome is $M_{\text{eff}} = 1,000,000$. This estimate might be improved for specific marker sets (Cheverud, 2001), which however requires some data that reflects the same set of markers that the estimate shall be based on.

- Disease marker: in general the program is assuming one disease (causing) marker with a certain effect on the outcome. The effect size is specified in terms of odds

ratios (see below). While in the GWAS at hand there might be more than one disease marker, the outcome of the corresponding optimal design will be basically independent from the actual number of disease markers and therefore assuming just one disease marker is sufficient. This comes from the fact that the number of disease markers is usually only a very small fraction as compared with the several hundred of thousands of non-associated null markers so that changing this small fraction basically has no effect on the design. The statistical parameters determine how the disease marker is modelled under the alternative hypothesis, which in turn has a crucial impact on the study power that is achievable by the multi-stage design.

  - Risk allele frequency: allele frequency of the risk (or disease causing) allele - the smaller this value, the lower the study power.
  - Odds Ratios: the odds ratio (OR) determines the assumed effect size – the higher the odds ratios, the higher the power to detect a true disease marker. It is defined for both one and two alleles versus no allele as follows:

$$\mathrm{OR}_{1/0} = \mathrm{Odds}(d = 1|r = 1)/\mathrm{Odds}(d = 1|r = 0) \tag{3}$$
$$\mathrm{OR}_{2/0} = \mathrm{Odds}(d = 1|r = 2)/\mathrm{Odds}(d = 1|r = 0) \tag{4}$$

    where $r = 0, 1, 2$ is the number of risk alleles and $d = 0, 1$ the disease status. Furthermore $\mathrm{Odds}(A|B) := P(A|B)/P(\text{not } A|B)$, where $P(A|B)$ denotes the conditional probability of A given B.

- *Variance estimation (advanced)*: type of variance estimation assumed in the genetic model. The user can choose between the standard pooled variance estimate (default) or to assume variance estimation from the controls only. For further reference see Zheng and Gastwirth (2007).

Subjects

- Sample size maximum: In a planning phase of the study this number represents the estimated maximum number of subjects that will be recruitable for genotyping. If recruiting is already finished, it is simply the total number of recruited subjects. Note that depending on budget restrictions or other study constraints, this number may or may not be fully used by the resulting optimized multi-stage design.

- Case fraction: this fraction is defined as: (#cases)/(#cases + #controls). The fraction is assumed to be constant over all stages. If a data file is provided (see section 2.5.2), the fraction is determined automatically depending on how the samples are distributed over the stages. Note that imbalanced fractions over different stages can be modelled this way, which can have a great impact on the study power. For example, the case fraction might be 0.5 at stage 1 and decreasing to 0.2 at stage 2. IMPORTANT NOTE: with the current version of OMD, the statistical model internally applied for such imbalanced case fraction designs, is fully correct

only for 2-stage designs. For three or more stages, at this point, it can be regarded an approximate solution to the problem, which, however, often is still better than modelling the imbalanced fractions by a constant fraction, for example, by taking the mean value of the stage-wise fractions.

- Subject cost: constant cost for each subject included in the study. This option is used to model all costs that arise *in addition* to genotyping (i.e. gene chip) cost, such as recruiting and phenotyping costs.

- Free of charge: the number of subjects that is free of charge (or has been paid already), which means they do not cause any genotyping or phenotyping costs, respectively. This option is most likely useful in a follow-up scenario, where a set of subjects have been genotyped already in an initial study and now the sample size is planned to be increased by recruiting/genotyping additional subjects, in order to be analyzed alltogether in a joined analysis. As an example, consider an initial study of 10,000 subjects and a planned sample of additional 20,000 subjects. In this case, one would set 'Sample size maximum' to 30,000 and the number 'Free of charge' to 10,000.

- *Detailed subjects data (advanced)*: this option is activated by setting 'Use file' to ⊙ Yes. The chosen file may contain detailed information about the study subjects including (recruiting) costs and phenotype. For more details see section 2.5.2. Note that activating this option invalidates all other 'Subject' options as the corrresponding values now will be derived directly from the specified data file.

2.4.2 I/O

The second tab contains all relevant parameters regarding input and output of the program (figure 6).

Input

- Chipset (*.xml): the last of the five mandatory input parameters: The chipset file contains a set of gene chips along with their cost. OMD will source this file in the process and optimally select chips to construct and plan the study. The specification of the chipset must be done in the appropriate XML-format, for which a detailed description is given in section 2.5.1. An example file named *chipset5M.xml* can be found in the folder *extdata/omd/chipsets*, which is located in the installation folder of GroupSeq++. You can get the full path by loading the package and then typing the following into the R-console:

```
file.path(path.package("GroupSeqPP"),"extdata/omd/chipsets/chipset5M.xml")
```

- *Design setup (optional)*: user-specified multi-stage design file, which can serve one or more of the following purposes:

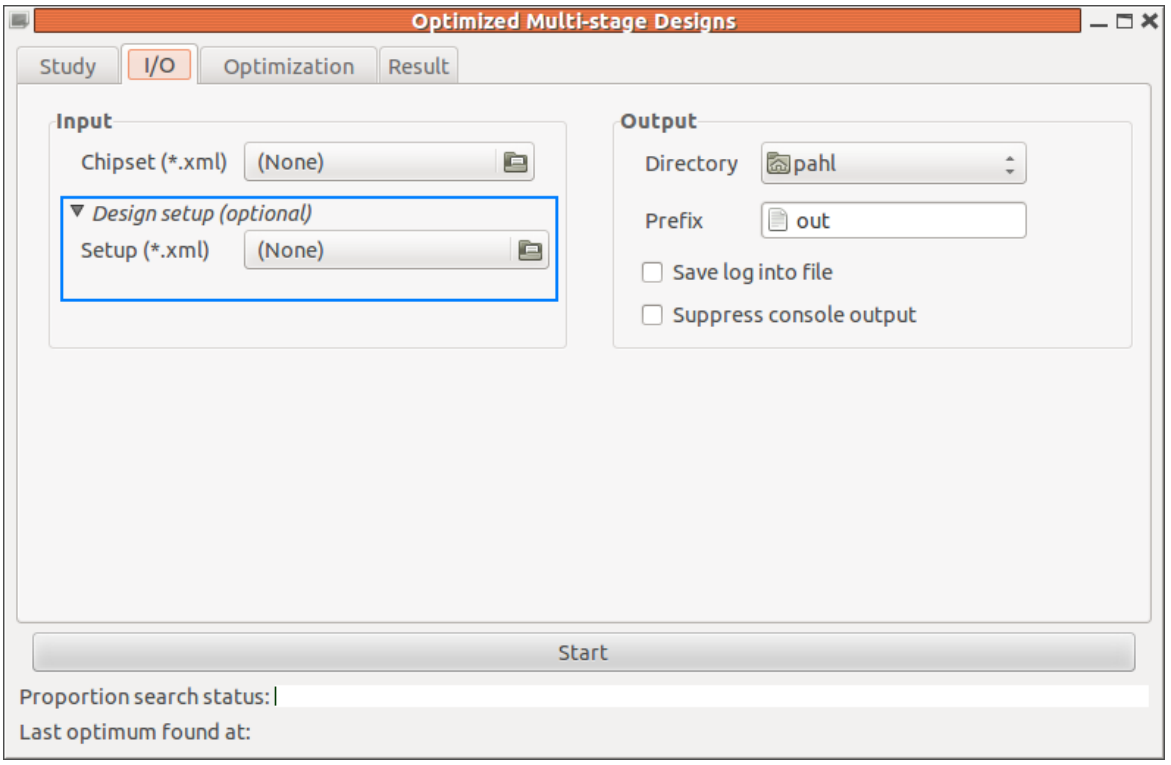


Figure 6: The OMD I/O tab at program start and with unfolded optional design setup specification (blue rectangle)

- set starting point(s) for optimization
- fix design parameters a-priori
- impose lower/upper bounds on design parameters

In particular, the *.xml output of any optimized design can be used as the input file in order to re-optimize the design with different parameters, for example, using a different optimization algorithm or targeting the opposite objective (see cross-validation, section 2.3.4.4). For more details on the file format see section 2.5.3.

Output

- Directory: the location where output files are placed
- Prefix: Since one optimization run is producing several files of output, this prefix is used to entitle the same prefix to all output files of a single run.
- ☐ Save log into file: The core of OMD is based on a C++ library, which is producing textual status output during optimization, which, in turn, is displayed in the R-console. Checking this option will save all this output into a file *[Prefix].log*, that is, by default into *out.log*.

- ☐ Suppress console output: Suppresses the textual status output in the R-console.

2.4.3 Optimization

The 'Optimization' tab (figure 7) is used to set the optimization objective, the number of stages and the search (or optimization) algorithm.

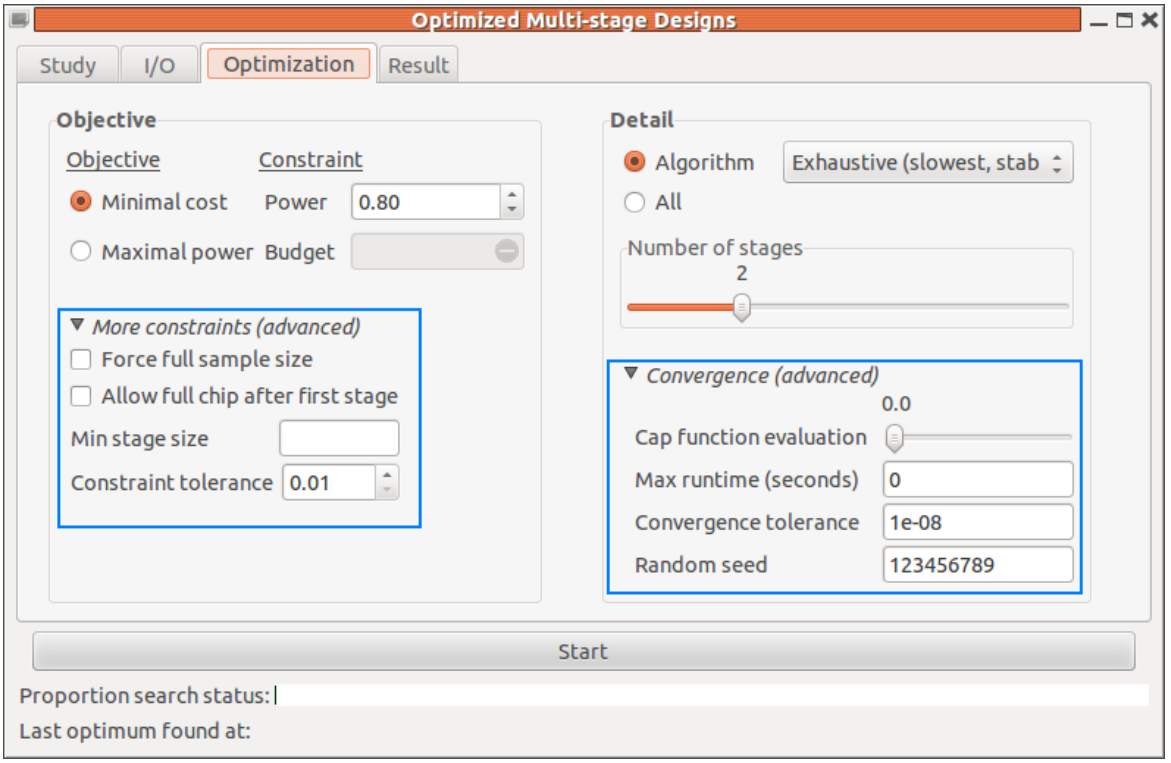


Figure 7: The OMD Optimization tab at program start with unfolded advanced options (blue rectangles)

Objective

- ⊙ Minimal cost: requires to set the desired study power. OMD will then try to determine a multi-stage design, which provides minimal cost while at the same time maintaining the desired power.
- ⊙ Maximal power: requires to specify the available study budget in dollars. OMD will then try to determine a multi-stage design that provides maximal study power while at the same time maintaining the budget.
- *More constraints (advanced)*

- □ Force full sample size: Forces the optimization to be done with the maximal available sample, that is, the last stage is fixed at the value entered in the 'Sample size maximum' field in the 'Study' tab.
- □ Allow full chip after first stage: Allow application of the chip that was used for (full) genotyping) at stage 1 also be used at stage 2 (normally a smaller chip is forced to be used assuming a shrinking marker set).
- Min stage size: the minimum sample size number that must be fulfilled in each stage. A minimum stage size can be useful for practical reasons, for example, to ensure some minimum rebate regarding the ordered gene chip. The value must be chosen such that (minsize)*(number of stages) ¡ (max sample size).
- Constraint tolerance: Relative tolerance of maintaining the constraint that is set at the top of the window. If the user sets a budget constraint of, say, 1,000,000 using the default tolerance of 0.01 (i.e., 1%), the target design may cost 1,000,000 +/- 10,000.

  Increasing the default value is most likely to be useful in situations where a budget constraint has been set but the cost function appears to be very sparse, most often caused by sparse chipset. In this case, increasing the value helps tweaking the optimization, since it allows for more valid outcomes of the function in terms of fulfilling the constraint. Note that regardless of the tolerance, OMD will still always try to fulfill the constraint as exact as possible.

Detail

- ⊙ Algorithm: OMD seeks to find both optimal alpha levels **and** an optimal allocation of sample size at each stage. While the search of alpha levels is beyond the user's control, the optimization algorithm searching for the optimal sample size allocation can be specified by the user. OMD in its current version provides three different algorithms: a systematic global search by Jones *et al.* (1993), a controlled random search algorithm as defined by Kaelo and Ali (2006), and a local variant by Powell (1994), all being implemented in the NLopt library (Johnson (2010)).

  The systematic *exhaustive* search (default) is the only algorithm that does **not** allow for user-specified starting values and is the most stable of the three, which is why one should generally start with this one. Moreover, with at least somewhat complex multi-stage designs (i.e., three or more stages), the *local* algorithm will often fail to find any reasonable (i.e., nearly globally optimal) solution to the problem. For this reason, the local algorithm is best used to sharpen an initial exhaustive search, using the initial result as a starting point (for more details see section 2.6).

  The *controlled random* search relies on a genetic algorithm and is somewhere in between the systematic exhaustive and the local search, that is, it usally converges faster than the *exhaustive* but slower than the *local* search, respectively. Since it accepts starting values, it can be also used to sharpen results of an initial systematic

search. On the other hand, it can be applied as an initial global search, especially if the default search, which is based on systematic division of the search domain into smaller and smaller subregions, is struggling, for which two basic points might can be indicative:

1. the resulting "optimum" is worse than that of an optimized design with less stages but otherwise identical setup
2. one or more of the resulting sample proportions[5] are very smooth (e.g. 0.4 instead of 0.4038) indicating failure (or at least insufficient searching time) to divide into smaller subregions

In addition, the *controlled random* search involves some randomness in its search space, so that it is possible to obtain a "non-deterministic behaviour" by changing the random seed value using the advanced convergence options (see below).

- ⊙ All: Applies all three algorithms in a subsequent chain going from top to bottom. As soon as one search has converged, the next algorithm is invoked, using the best result obtained thus far as starting values for the current run.

  Since "real" convergence can take very long with the first two search algorithms, it is recommended to set some max runtime limit (see below) with this option, which then is distributed in fractions of $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{6}$ among the three algorithms, in that order, that is, the initial search obtains half, the second search a third, and the final search, which can be considered as a local sharpening of the optimum, a sixth of the totally available time, respectively.

- Number of stages: The desired number of stages of the multi-stage design. Generally, the more stages are used, the more can be optimized but the more complex the design. Also the amount of optimization that is possible depends on the variety of gene chips available (see also section 2.3.4). In practice, often three or four stages are the best choice (Pahl *et al.*, 2009).

- *Convergence (advanced)*

  - Cap function evaluation: increasing this parameter will decrease the total runtime but at the same time the accuracy of optimization. Internally the number of function evaluations during the optimization process of the significance levels are capped. The parameter ranges from 0 (no cap) to 1 (full cap), where the latter basically means that the optimal significance levels are almost guessed. As a result, increasing this value may result in suboptimal results and therefore is best used for initial explorative optimization runs.
  - Max runtime (seconds): stop the optimization at the latest after the specified amount of time (0 = disabled). Since the optimization can be stopped manually at any time using the 'Stop' button, this option is most useful with the ⊙ All

[5] sample proportion = (cumulative number of subjects) / (total number of subjects)

option being activated in order to ensure that all algorithms are in fact applied as opposed to getting stuck with the first exhaustive search. In this case, the max runtime will be divided evenly among all search algorithms.

- Convergence tolerance: relative convergence tolerance of the search algorithm, that is, basically the optimization is stopped, when an optimization step changes the objective function value by less than the tolerance value multiplied by the absolute function value (< 0 = disabled). Especially when minimizing the cost, due to the discrete nature of the cost function (since it is based on real chipsets), it can take very long for a search algorithm to converge.
- Random seed: the random seed used for the controlled random search algorithm.

2.5 File formats

2.5.1 Chipset file (*.xml)

Any chipset file must be kept in an XML-format as follows: a chip is specified via a `<chip>` tag and each chip specification must provide the following entries/tags:

Chip specification

```
<chip>
    <name>      <!-- name of the chip -->       </name>
    <capacity>  <!-- no. of markers/SNPs -->    </capacity>
    <pricing>   <!-- price of the chip -->      </pricing>
</chip>
```

The actual cost of the chip, are set by a `<price>` tag within the `<pricing>` tag, for example:

Chip with a single price

```
<chip>
    <name> iSelect7600 </name>
    <capacity> 7600 </capacity>
    <pricing>
        <price> 174 </price>
    </pricing>
</chip>
```

If the user wants to model discount price structures, he or she must set a `<discount>` tag within the `<pricing>` tag, where each discount entry must contain a piece number along with the corresponding price, for example:

Chip with discount price structure

```
<chip>
    <name> iSelect7600 </name>
    <capacity> 7600 </capacity>
    <pricing>
        <discount>
            <piece_number> 2304 </piece_number>
            <price> 174 </price>
        </discount>
        <discount>
            <piece_number> 5760 </piece_number>
            <price> 142 </price>
        </discount>
        <discount>
            <piece_number> 11520 </piece_number>
            <price> 125 </price>
        </discount>
    </pricing>
</chip>
```

All piece numbers are treated as lower bounds for the discount, that is, in the above example, the chip is costing 174 only if 2,304 or more chips are requested and unless the number of chips reaches (or exceeds) 5,760, in which case the second discount step applies, dropping the price to 142. In consequence, this chip type will not be available at all should the number of requested pieces fall below 2,304.

2.5.2 Trait data

A trait file consists of a table with four (tab or space delimited) columns. Each row defines an individual containing a unique ID, the individual's name, costs associated with the individual (e.g. recruiting costs), and its phenotype[6]. Any line starting with `#` is considered a comment and hence ignored by OMD. The given order of entries (top to bottom) will be used as the incoming sequence in the multi-stage design. The resulting table may look as follows:

Example trait data

```
#id     name    cost    phenotype
1       A       0       1
2       B       0       1
3       C       0       1
4       D       0       1
```

[6] At this point *only* binary phenotypes implying a case-control design are supported.

```
5       E       0       0
6       F       0       1
7       G       0       1
8       H       0       1
9       I       0       1
10      J       0       0
...
```

This table is part of the file *tiny_sample.txt*, which is found in the folder *extdata/omd/trait*. Again, you can obtain the full path by loading the package and then typing the following into the R-console:

```
file.path(path.package("GroupSeqPP"),"extdata/omd/trait/tiny_sample.txt")
```

2.5.3 Design file (*.xml)

A multi-stage design file must at least contain `<stage>` tags, which determine the total number of stages. Also the `<stage>` tags must be given in the correct order, starting with stage 1 (from top to bottom). Within each stage, in turn, two items can (but must not) be specified:

Design file: stage specification

```
<stage>
    <ncum>  <!-- cumulative no. of subjects up to that stage -->    </ncum>
    <chip>  <!-- name of the marker chip applied at that stage -->  </chip>
</stage>
```

2.5.3.1 Setting bounds Brackets allow imposing lower and/or upper bounds on the `<ncum>` values, In particular, there are four different kinds of possible brackets:

`] x`	x is *upper* bound of the *former* stage
`[x`	x is *lower* bound of the *current* stage
`x]`	x is *upper* bound of the *current* stage
`x [`	x is *lower* bound of the *next* stage

While all brackets can be used in combination, the above order must be preserved, for example, `"][x]["` and `"][x"` are both valid, while `"[]x"` is not. If x is lower and upper bound at the same time (i.e. `"[x]"`), it is treated as a fixed parameter and will not be subject to optimization at all. Furthermore note that both `<ncum>` and `<chip>` tags can be fixed but single specific lower or upper bounds are not supported for chips and therefore ignored if used with values from `<chip>` tags. In general all items left unspecified or specified but not fixed represent free parameters, which are going to be optimized by OMD. Values being specified without any bound are valid as well and just serve as starting

points in the optimization routine. As an example consider the file *5-stage-example.xml* located in *extdata/omd/designs*, which contains the design setup displayed in listing 1. Again, it is important to note that any sample size set with an `<ncum>` tag is provided

5-stage-example.xml

```
<multistage_design>
    <stage>
        <!-- Stage 1: lower subject bound, fixed gene chip -->
        <ncum> [10000  </ncum>  <!-- At least 10,000 subjects at this stage -->
        <chip> [HumanOmni5] </chip> <!-- Use HumanOmni5 chip (see chipset5M.xml)-->
    </stage>

    <stage>
        <!-- Stage 2: subject bounds, unknown/free gene chip -->
        <ncum> ][25000 </ncum>  <!-- Maximally 25,000 subjects up to former stage
                                    and at least 25,000 up to this stage -->
    </stage>

    <stage>
        <!-- Stage 3: no specific constraints -->
        <!-- The <stage>-tags are still needed to set the total no. of stages -->
    </stage>

    <stage>
        <!-- Stage 4: just suggest starting value (no constraint) -->
        <ncum> 40000 </ncum>    <!-- Start optimization using 40,000 subjects -->
    </stage>

    <stage>
        <!-- Stage 5: fixed subject number, unknown/free gene chip -->
        <ncum> [50000] </ncum>  <!-- Exactly 50,000 subjects up to this stage) -->
    </stage>
</multistage_design>
```

Listing 1: Example of user-defined 5-stage design setup

as the *cumulative* sample size, that is, the total number of subjects being analyzed up to (and including) the stage at hand. Going from bottom to top in listing 1, the sample size at stage 5 has been fixed and thus will be not part of optimization. In contrast, at stage 4, the given value (40,000) will be just considered as a starting value by OMD and has no further effect otherwise. For stage 3 there is no additional information at all but at stage 2, the cumulative sample size is set to be at least 25,000 while stage 1 should not exceed those 25,000. Finally, stage 1 shall process at least 10,000 subjects using the chip denoted 'HumanOmni5'.

2.5.3.2 Special case - fixing values The motivation behind fixing values is twofold. First, it simply models limitations occuring in practice. For this, imagine an initial study

that was completed with 10,000 subjects and which is planned to be extended by another 30,000 subjects. If the type of chip used in that initial study is still available, one just sets a lower bound for stage 1, namely `[10000`. However, if this type of chip is not available anymore or due to other practical limitations that require the stage to be regarded as completed, the first stage needs to be fixed (i.e., `[10000]`). The second motivation is that each fixed parameter does not need to be optimized anymore and therefore reduces the required computational effort in the process of finding the optimal design. Of course, care must be taken, since binding or fixing parameters can result in suboptimal designs. A special case in this context is to fix the final sample at the maximal available sample size, which, instead of specifying it in the design file, also can be achieved by activating the option 'Force full sample size' in the 'Optimization' tab (see section 2.4.3). For example, if an optimized 2-stage design already spends the entire sample, a design using more than two stages will always do so as well, so that fixing the last sample at the full size becomes a logical step in this case.

2.6 Use case - minimize study cost

At first we will seek to minimize the study cost as much as possible under the constraint that the study has to maintain a certain power. Most of the results presented in the following sections were derived with restricted running times (i.e. using the max runtime limit option). The actual values may thus differ slightly in what you obtain when recalculating the examples on your own, depending on the speed of your computer for the most part.

2.6.1 Study setup

We start the OMD-module and enter some values into the Study tab as shown in figure 8. That is, we plan a GWAS using not more than 40 000 subjects and a marker set consisting of 5 000 000 SNPs. Furthermore, the significance level ($\alpha = 0.05 \times 10^{-6}$) relies on the assumption of 1 000 000 effective tests as opposed to the 5 000 000 markers planned to be investigated (for more details on the choice of α see section 2.4.1). The fraction of cases throughout the entire sample is set to 25% and the allele frequency of the sought risk allele is assumed to be 10% with an odds ratio of 1.2 and 1.4 for observing the trait if the case has one or two risk alleles, respectively. To specify our chipset, we switch to the I/O tab and select the accompanying chipset file *chipset5M.xml* (figure 9). Again, the chipset file *chipset5M.xml* used here is found in the folder *extdata/omd/chipsets*, which is located in the installation folder of GroupSeq++, and you can get the full path by typing

```
file.path(path.package("GroupSeqPP"),"extdata/omd/chipsets/chipset5M.xml")
```

into the R-console. To explore the chips specified in the file, you can either open it with a text editor, your standard web browser, or some other XML-compatible viewer. In total,

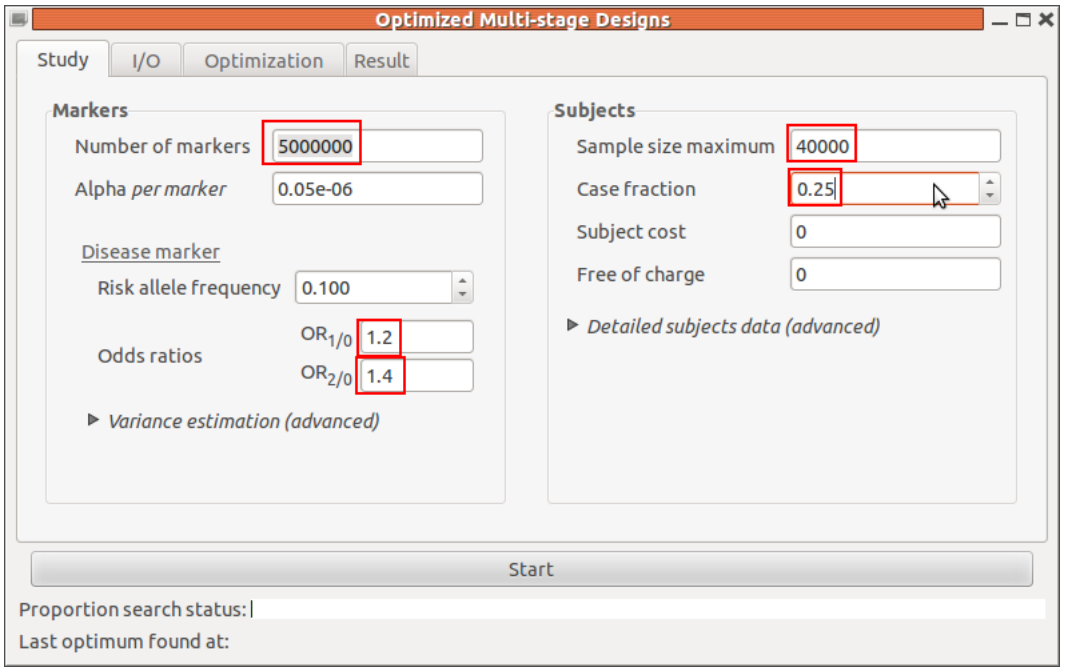


Figure 8: OMD use-case study setup (entered values marked by red rectangles)

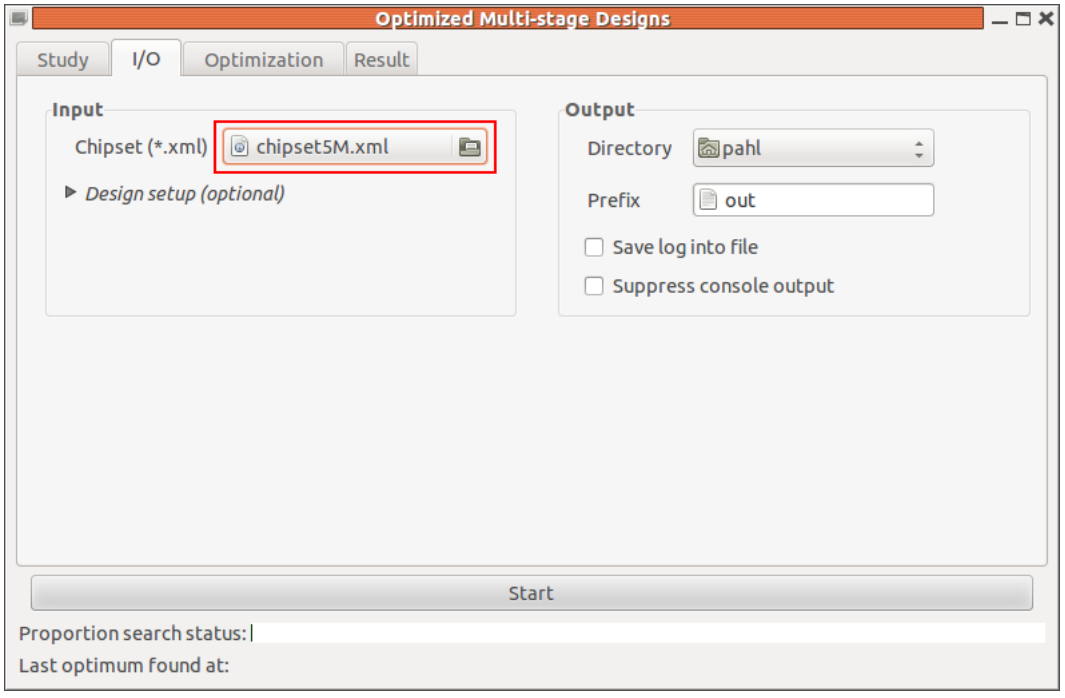


Figure 9: Select the chipset file.

the file lists seven different chips, with capacities of 5 100 000, 50 000, 16 700, 7 600, 384, 96, and 1 SNP(s), respectively. Most of the chip prices have been determined from the JHU SNP Center (see http://snpcenter.grcf.jhmi.edu/pricing.html).

2.6.2 1-stage design

As described in section 2.3.4.2, we initially try for a simple single-stage (or 1-stage) design to get an idea of the possible outcome. We start by requesting a rather high power of 95%. For this, we switch to the Optimization tab, and set the parameters accordingly (figure 10). We hit the Start button, which results in an error message, stating that the "desired

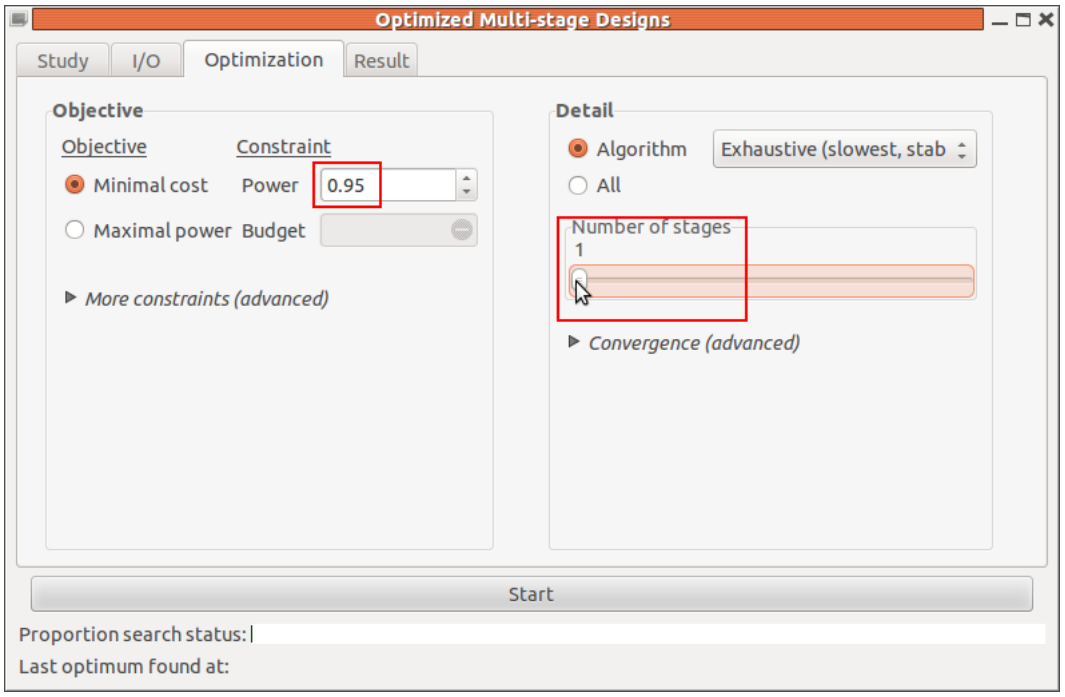


Figure 10: Select 1-stage design with 95% power.

power (0.95) is not achievable" with our sample size (figure 11). Apparently, the maximal

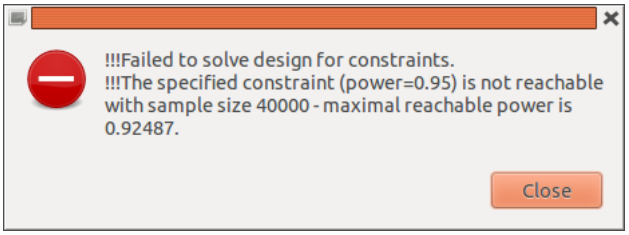


Figure 11: OMD error dialog due to unreachable power.

achievable power with the specified parameter setup is 92.487% (see figure 11). In order to increase this value, we now could adjust the initial setup, for example, by increasing the (maximal) sample or the assumed effect size (i.e., higher odds ratios or higher risk allele frequency), respectively. Here we proceed by just lowering the power requirement instead, and set the desired power below those 92.487% back to 80% (figure 12). Hitting

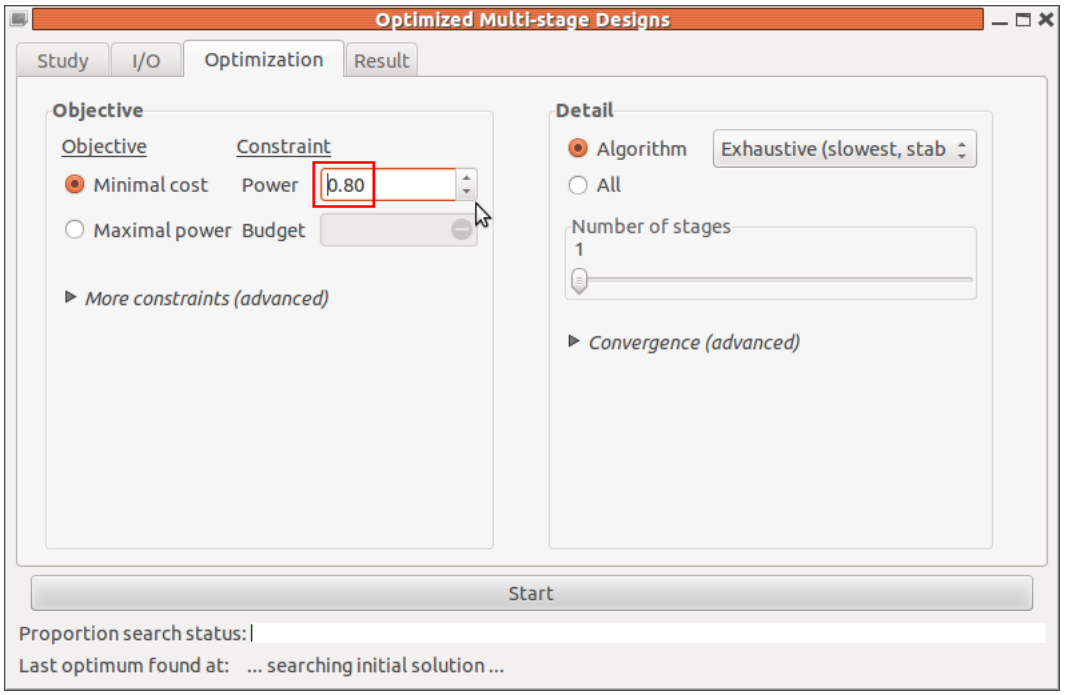


Figure 12: Set 80% power in the Optimization tab.

'Start' again will show a new dialog (figure 13), this time, containing a valid result[7]. The

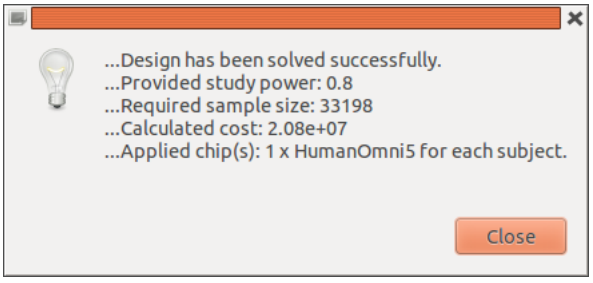


Figure 13: Result notification of 1-stage design solved for 80% power.

first line states that the "Design has been solved successfully". In fact, when calculating 1-stage designs, there is no optimization involved, because the design constraints (here: $0.05 \cdot 10^{-6}$ type I error and 80% power condition) leave no parameters free to optimize. That is, given the chosen constraints provide a solvable condition, there will be exactly one correct 1-stage design for any setup, The new result implies the need of 33 198 out of the maximal available 40 000 subjects in order to reach 80% study power. On a side note, it had not been advisable to set the desired power to match the aforementioned 92.487% exactly, because the single-stage design would exactly use all available individuals in this case, but since multi-stage designs need to compensate for their multiple analyses, their

[7]The 1-stage result output being displayed as an error message dialog at this point is not ideal and will be improved in a future version of GroupSeq++.

required sample size always exceeds that of a single-stage design with identical power. As a result, there should be always some sample size cushion with the single-stage design in order to be able to construct multi-stage designs, or in other words, the single-stage design must never be constructed to fully exhaust the maximal available sample size. A further result is that we are using the HumanOmni5 chip for all the 33,198 subjects leading to costs of $2.08 \cdot 10^7$. While OMD will always select the cheapest chip among all available chips with sufficient capacity, here the HumanOmni5 is the only chip that can hold 5 000 000 SNPs anyway (see the chipset file).

2.6.3 2-stage design

2.6.3.1 First run Compared with the above obtained 1-stage result, for the 2-stage design, we expect a slightly larger sample size but much lower study cost. Since the optimization process can take some time until full convergence, for the sake of this tutorial, we will limit the maximal running times in most cases. We start by setting a maximum of 30 seconds[8] under Detail→Advanced→Max runtime (seconds) in the Optimization tab (see figure 14). In the scaler, we set the 'Number of stages' to 2 and start the calculation.

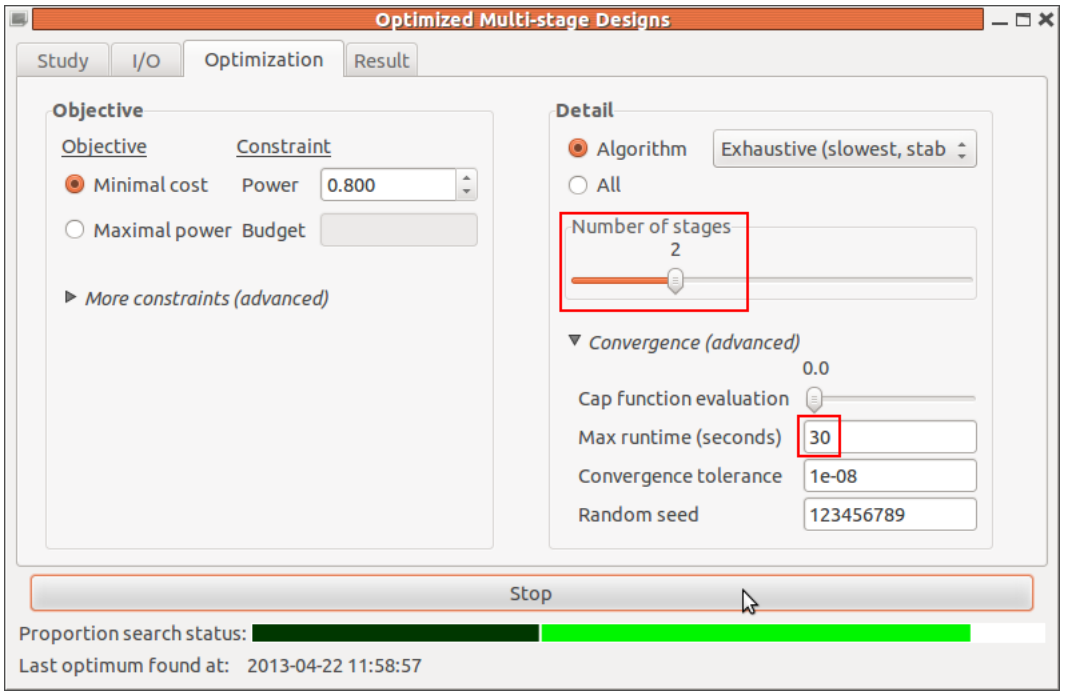


Figure 14: Setting up first run of the 2-stage design optimization.

At first, you will now notice the green bars at the bottom of the window, which show the current search status in the optimization process. The two different bars in this case represent the two stage-wise proportions of samples from the total available sample size.

[8] A fully converging run took ~1000 seconds for the example at hand, using an Intel® Core™ i5 CPU 650 @ 3.20GHz x 4

The color codes the number of markers expected to be genotyped at the corresponding stage – the darker the color, the more markers. Below that at the very bottom, OMD tells us the last point of time, when the optimum could be improved, which is useful when optimizing more complex designs of three or more stages, where the optimization will take a lot longer to converge. After about 30 seconds, the computation should stop showing the dialog as depicted in figure 15. As stated there, the "optimization has finished regularly"

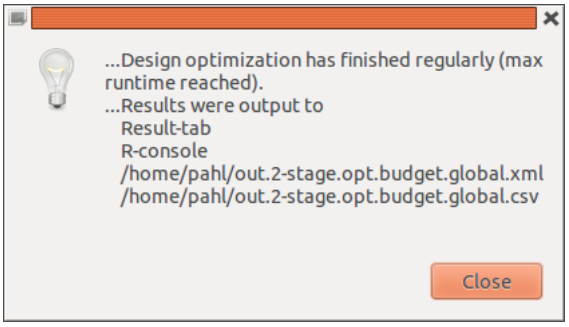


Figure 15: Result notification of the first 2-stage design optimization run.

being halted due to reaching the maximum runtime limit we had set before. Furthermore, the "results were output to" various places:

- Result-tab
- R-console
- *.xml file
- *.csv file

The "Result-tab" output is covered in the next section 2.6.3.2 followed by the R-console output in section 2.6.3.3. Finally, the file output will be explained in section 2.6.3.4.

2.6.3.2 Output: 'Result' tab The main source for immediate inspection of results during the run is the 'Result' tab (figure 16). The results presented therein are updated on the fly. In particular, all relevant outcomes of the optimized multi-stage designs are shown, divided into the four higher level categories Sample size, Probability, Marker chip, and Cost. For each of these categories, the output can be folded/unfolded by clicking on the ▶/▼ placed next to them. From the 3rd column on, the outcome is displayed successively for each stage, from top to bottom:

- Sample size
  - Stage-wise: sample size at that stage
  - Cumulative: sample size accumulated up to this stage

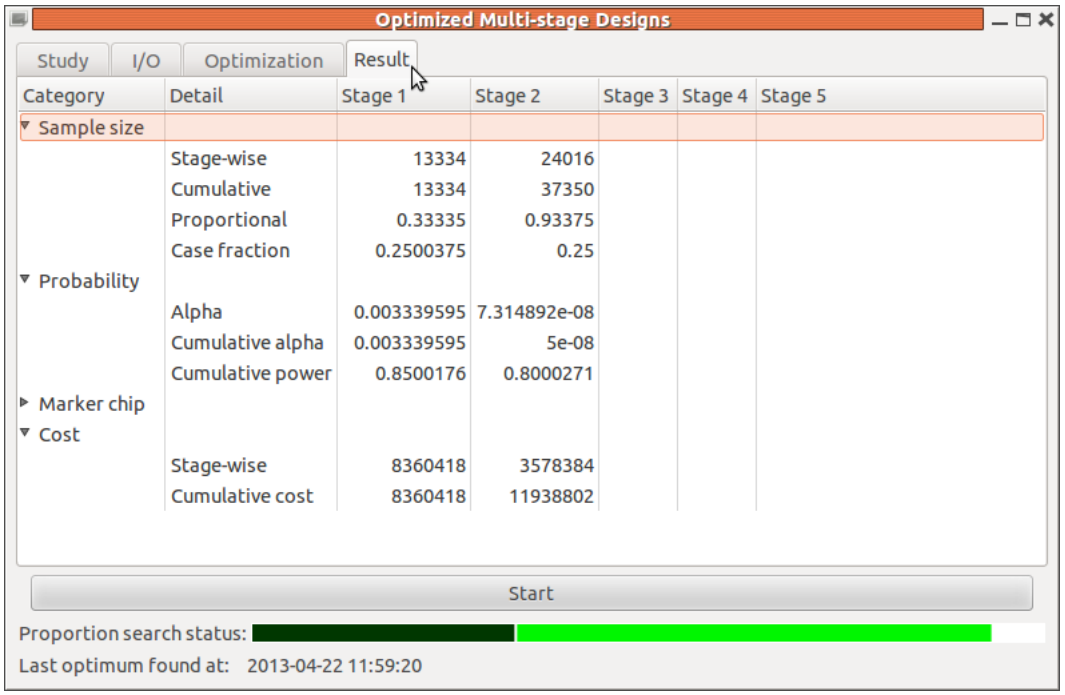


Figure 16: Result tab after the first optimization run (30s runtime limit).

  - Proportional: sample proportion relative to maximal sample size
  - Case fraction: fraction of cases in that stage

- Probability
  - Alpha: nominal significance level at this stage. Against this level the interim p-value of a marker must be compared in order to decide whether it will be carried forward to the next stage (see also section 2.3.3)
  - Cumulative alpha: the accumulated type I error. At the final stage, this should always be equal to the total type I error risk as specified by the user prior to optimization in the 'Alpha per marker' field.
  - Cumulative power: the cumulative power to detect the disease marker. The total power to detect a disease marker using this design, again, can be derived from the value obtained at the final stage.

- Marker chip
  - Type: type (or name) of the chip
  - Capacity: marker capacity of the chip used in that stage
  - Expected no. of markers: the expected number of markers at that stage, when applying the above significance thresholds to decide which markers are carried forward to a next stage

  - Price per chip: the cost per chip used at this stage
  - No. used per subject: the number of chips needed for each subject (if the capacity of a particular chip is smaller than the expected number of markers, more than one chip might be required)

- Cost
  - Stage-wise: cost of the design at that stage
  - Cumulative cost: cost accumulated up to this stage

Figure 17 again presents the Result tab, this time also highlighting the most important parts of the outcome[9]. The cumulative sample size (blue rectangle) determines the size

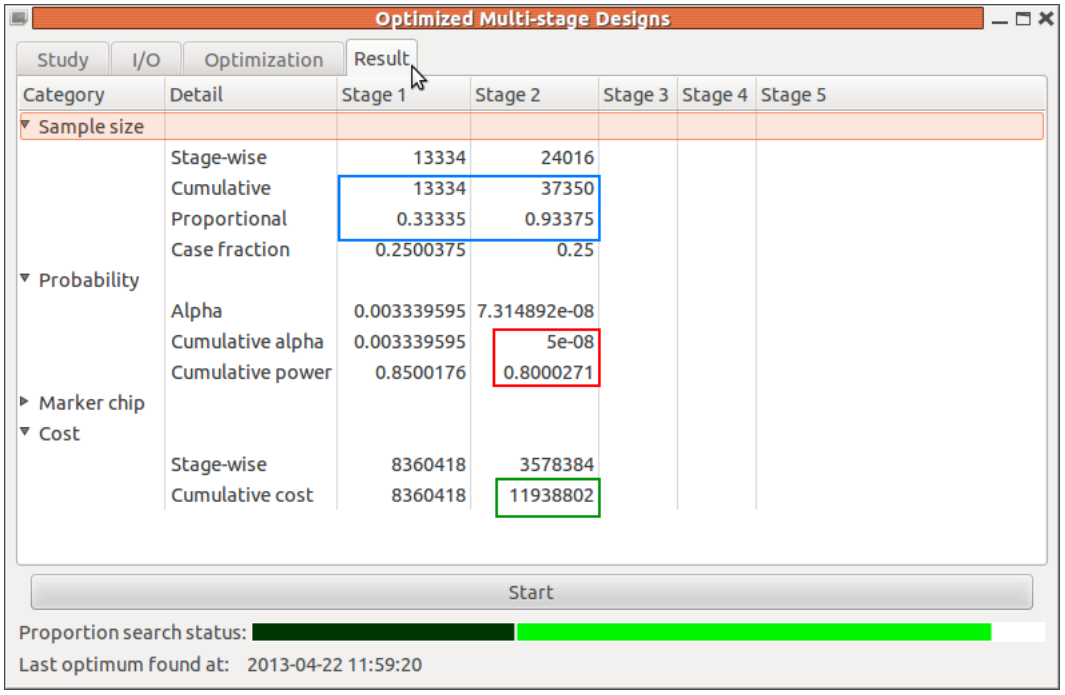


Figure 17: Highlighted results of the cost-minimized 2-stage design

of each group and basically forms the "shape" of the obtained multi-stage design. Under Probability you should always double-check that your pre-specified constraints – in our case 'Alpha per marker' = 0.05e-06 and Power = 0.80 – are maintained (red rectangle), which is the case here. Finally, the optimized (i.e., minimized) objective, namely the cost of the design, result to about 11 940 000 (green rectangle), which is 8 860 000 less than (or 57% of) the 1-stage design. With a total sample size of 37,350, as expected, it exceeds the 1-stage sample by 4,152 subjects, although still not consuming all of the maximally

[9]All results being derived with a runtime limit can vary from machine to machine depending on their CPU speed, so if you re-run the presented examples, your outcomes may differ slightly from those presented in this tutorial

available 40,000 subjects. This can be seen also from the 'Proportion search status' at the bottom, leaving some white space at the most right of the green bars. On a side note, considering the case fraction at stage 1, we observe that it does not exactly equal the requested value of 0.25 (or 1/4), which just arrises from the fact that the stage 1 sample size of 13 334 including cases *and* controls, has a non-zero remainder when divided by 4.

Next, we examine the gene chips applied with the optimized design by unfolding the 'Marker chip' category and hiding 'Sample size' and 'Cost' (figure 18).

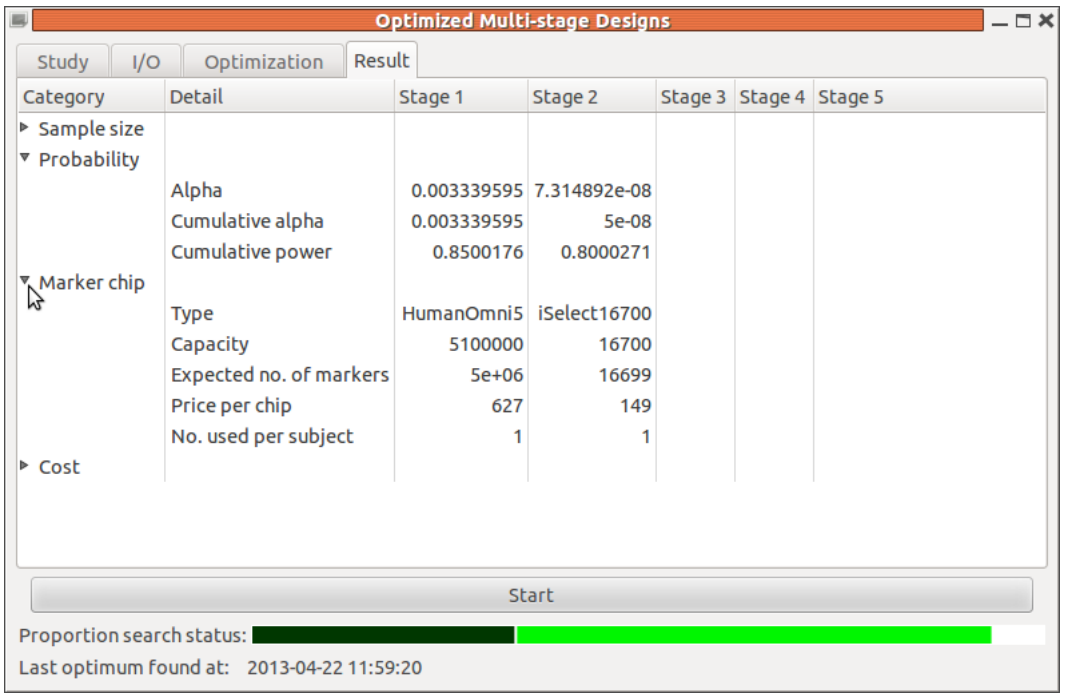


Figure 18: Applied gene chips of the cost-minimized 2-stage design

Of course, stage one is using the Human Omni 5 chip again, being the only chip that covers the starting marker set of 5 000 000 SNPs. However, the low significance threshold at stage one (Alpha = 0.003339595, figure 18) basically lets only those markers through, which show the top 0.3% highest test statistics after stage one, so that the number of markers is expected to have been dropped down to 16 699 at stage 2. Since the Illumina® iSelect chip with a capacity of 16 700 proves to be the most cost effective chip in this case, it is chosen for stage 2. In particular, the next bigger chip, the iSelect50000 (see file *chipset5M.xml)*, would be filled only for about 33% and costing more than twice. The next smaller chip, the iSelect7600, is costing only 125 given the requested piece number (i.e., 24 016 subjects, re-unfold the Sample size category to see this number), but we would actually need three (differently) customized chips per subject to cover the expected number of markers, resulting in more than twice the cost as well. Of course, the fact that the expected number of markers is fitting nearly perfectly to the capacity of the iSelect16700 chip is no accident but a result of the optimized significance thresholds.

2.6.3.3 Output: R-console Unless suppressed by the user (see section 2.4.2), OMD also puts an on-the-fly output into the R-console, which can be used (a) as a logging output and (b) to follow the optimization process, thereby providing information about the status of convergence of the search[10]. Too see this, we take a look of the console output of the previous run (~3 pages long):

2-stage optimization run - R-console output)

```
   ############
   # OMD-Tool #
   ############

+-----------------+-----------------+------------------+
|       OMD       |     v0.1.1      |    19/Apr/2013   |
|           Optimized Multi-stage Designs              |
+-----------------+-----------------+------------------+
|   Copyright (c) 2011-2013 Roman Pahl, IMBE Marburg   |
|  Distributed under the Boost Software License, v1.0  |
+------------------------------------------------------+
...Writing parameter config to '/home/pahl/out.2-stage.opt.budget.global.cfg':
   # Parameter values set by user:
   alg = global
   alpha = 5e-08
   cap = 0
   chipset = /home/pahl/chipset5M.xml
   cost = 0
   ctol = 0.01
   frac = 0.25
   markers = 5000000
   nfree = 0
   nmax = 40000
   or1 = 1.2
   or2 = 1.4
   out = /home/pahl/out
   power = 0.8
   raf = 0.1
   seed = 123456789
   stage = 2
   tmax = 30
   tol = 1e-08
   var = pooled

...Started at Fri Apr 26 14:19:39 2013
...Start optimizing 2-stage design...
...Results are updated/written on the fly in
    Result-tab
    R-console
    /home/pahl/out.2-stage.opt.budget.global.xml
    /home/pahl/out.2-stage.opt.budget.global.csv

...Searching initial solution...
```

[10] It is planned to provide a graphical presentation of the search process in a future version of OMD.

```
...Time: Fri Apr 26 14:19:39 2013
...Optimized budget: 1.453e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             20000 (20000)  1 x HumanOmni5 (5100000)       1.254e+07
2             13334 (33334)  1 x iSelect16700 (16700)       1.453e+07

...Time: Fri Apr 26 14:19:39 2013
...Optimized budget: 1.452e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             20000 (20000)  1 x HumanOmni5 (5100000)       1.254e+07
2             17778 (37778)       2 x Custom384 (384)       1.452e+07

...Time: Fri Apr 26 14:19:39 2013
...Optimized budget: 1.253e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             15556 (15556)  1 x HumanOmni5 (5100000)       9.754e+06
2             22222 (37778)    1 x iSelect7600 (7600)       1.253e+07

...Time: Fri Apr 26 14:19:40 2013
...Optimized budget: 1.236e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             14074 (14074)  1 x HumanOmni5 (5100000)       8.824e+06
2             23704 (37778)  1 x iSelect16700 (16700)       1.236e+07

...Time: Fri Apr 26 14:19:40 2013
...Optimized budget: 1.235e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             15556 (15556)  1 x HumanOmni5 (5100000)       9.754e+06
2             20740 (36296)    1 x iSelect7600 (7600)       1.235e+07

...Time: Fri Apr 26 14:19:40 2013
...Optimized budget: 1.214e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             14074 (14074)  1 x HumanOmni5 (5100000)       8.824e+06
2             22222 (36296)  1 x iSelect16700 (16700)       1.214e+07

...Time: Fri Apr 26 14:19:41 2013
...Optimized budget: 1.197e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13581 (13581)  1 x HumanOmni5 (5100000)       8.515e+06
2             23209 (36790)  1 x iSelect16700 (16700)       1.197e+07

...Time: Fri Apr 26 14:19:44 2013
...Optimized budget: 1.197e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13416 (13416)  1 x HumanOmni5 (5100000)       8.412e+06
2             23868 (37284)  1 x iSelect16700 (16700)       1.197e+07

...Time: Fri Apr 26 14:19:44 2013
...Optimized budget: 1.196e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
```

```
1             13562 (13562)  1 x HumanOmni5 (5100000)        8.503e+06
2             23228 (36790)  1 x iSelect16700 (16700)        1.196e+07

...Time: Fri Apr 26 14:19:46 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13361 (13361)  1 x HumanOmni5 (5100000)        8.377e+06
2             23923 (37284)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:19:49 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13343 (13343)  1 x HumanOmni5 (5100000)        8.366e+06
2             23996 (37339)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:19:49 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13359 (13359)  1 x HumanOmni5 (5100000)        8.376e+06
2             23925 (37284)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:19:51 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13343 (13343)  1 x HumanOmni5 (5100000)        8.366e+06
2             23990 (37333)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:19:54 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13343 (13343)  1 x HumanOmni5 (5100000)        8.366e+06
2             23988 (37331)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:19:57 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13342 (13342)  1 x HumanOmni5 (5100000)        8.365e+06
2             23989 (37331)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:19:58 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13342 (13342)  1 x HumanOmni5 (5100000)        8.365e+06
2             23988 (37330)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:20:00 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13335 (13335)  1 x HumanOmni5 (5100000)        8.361e+06
2             24016 (37351)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:20:03 2013
```

```
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13334 (13334)  1 x HumanOmni5 (5100000)         8.36e+06
2             24017 (37351)  1 x iSelect16700 (16700)        1.194e+07

...Time: Fri Apr 26 14:20:04 2013
...Optimized budget: 1.194e+07
   sample size (cumulative)    marker chip (capacity)  cumulative cost
1             13334 (13334)  1 x HumanOmni5 (5100000)         8.36e+06
2             24016 (37350)  1 x iSelect16700 (16700)        1.194e+07

   ############
   # 2-stage #
   ############

...Design optimization has finished regularly (max runtime reached).
...Results were output to
    Result-tab
    R-console
    /home/pahl/out.2-stage.opt.budget.global.xml
    /home/pahl/out.2-stage.opt.budget.global.csv


...Finished at Fri Apr 26 14:20:09 2013
...Total runtime: 30 seconds
```

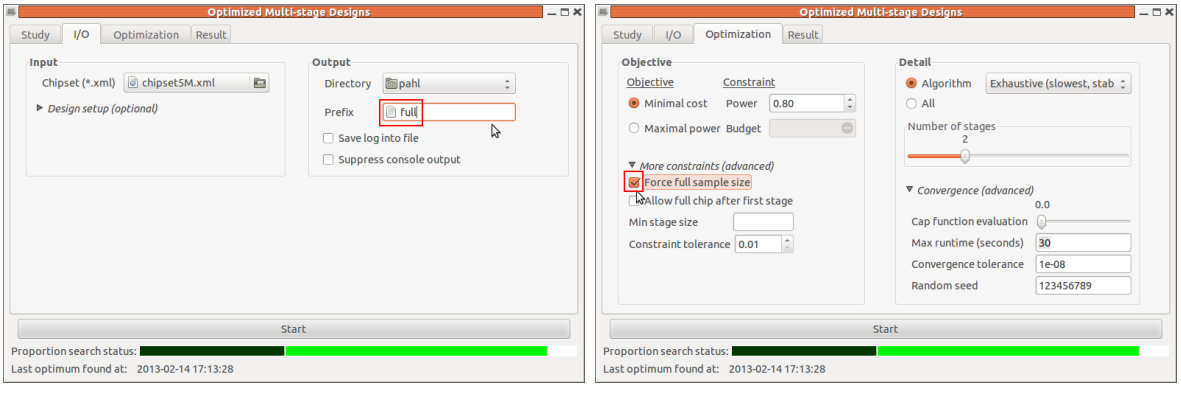
After some general information and parameter configuration at the top, the on-the-fly output follows. Each time a new optimum is found, a small table is printed displaying the interim result regarding three important design parameters: the sample size, the applied marker chip, and the cumulative cost of the design. In contrast to the 'Result' tab, here the stages are put row by row (i.e., not by column), so that the row numbers denote the corresponding stage. Thus, the cumulative sample size and cost in the last row equal the total sample size and the total cost of the design, respectively. In addition, the optimized objective (here "Optimized budget") is put at the top of each table, allowing to quickly follow the optimization process. As can be seen, when reaching convergence, the improvements of the objective value get smaller and smaller. Towards the end of the run, they in fact have fallen beyond the accuracy of the depicted (rounded) value, indicating that the search has nearly converged to the actual optimum. As soon as computation has stopped, some status information is placed at the very end, here again stating that the optimization was halted by reaching the specified maximum runtime.

To sum up, the R-console output serves as a easy to read logging about the *progress* of convergence. If this is not desired by the user and/or if he or she wants to use the R-console for different things while leaving OMD running in the background, the output can be turned off at any time by activating the "Suppress console output" checkmark in the 'I/O' tab.

2.6.3.4 Output: *.xml and *.csv files For every optimization run, two output files are generated from the start. More precisely, as soon as OMD has found a first solution to the optimization problem, it starts writing out the current best result on the fly into these two files, which are always of type *.xml and *.csv, respectively. Each time OMD can improve the optimum, it will then update (i.e., overwrite) these files, which allows for stopping the optimization process at any time and still have access to the most recently found solution in these files. The *.xml file contains the last found optimal design according to the OMD design file format (see section 2.5.3) and therefore is suitable to be used as a design setup for future runs (see also 2.7.1.2). The *.csv file provides a comprehensive and human readable description of the optimization result, similar to that displayed in the 'Result' tab (section 2.6.3.2) and is best used for comparing and storing different runs.

The names of the output files are generated automatically indicative of the task they were generated from. The file **out.2-stage.opt.budget.global.xml** in this context implies the **out**put of a **2-stage opt**imization run where the **budget** was optimized using the **global** search algorithm. Besides its informative character this also eases the usage of the program, for example, when performing the optimization with a different algorithm so that the user does not have to specify new output names for every single run.

2.6.3.5 Forcing the full sample Earlier we have calculated the 1-stage design, for which there exists exactly one solution for the sample size in order to fulfill both the $\alpha = 0.05 \times 10^{-6}$ significance threshold and the 80% power condition, namely a sample size of 33,198. In contrast, there are many possible 80% power 2-stage designs using different thresholds and sample sizes, of which we had just tried to find the most cost efficient one using the optimization functionality of OMD. To further illustrate this, we re-run the optimization but this time will force OMD to use the full (i.e. maximal) sample of 40,000. For this, we set a checkmark in the corresponding box (figure 19b) and change the output prefix to "ffull" (figure 19a), the latter in order to not overwrite the result of the previous run. Note that if you want to compare the results within the GUI, you could have also



(a) Enter "ffull" as output prefix (b) Check 'Force full sample size' box

Figure 19: Prepare 2-stage design optimization with forced full sample size.

opened another OMD-window from the main menu, re-enter the parameter setup[11] and proceed from there. We start the optimization and immediately notice that this time the proportion search status is filled completely by the green bars. After about 30 seconds again, your 'Result' tab should look similar to figure 20. The resulting costs of about

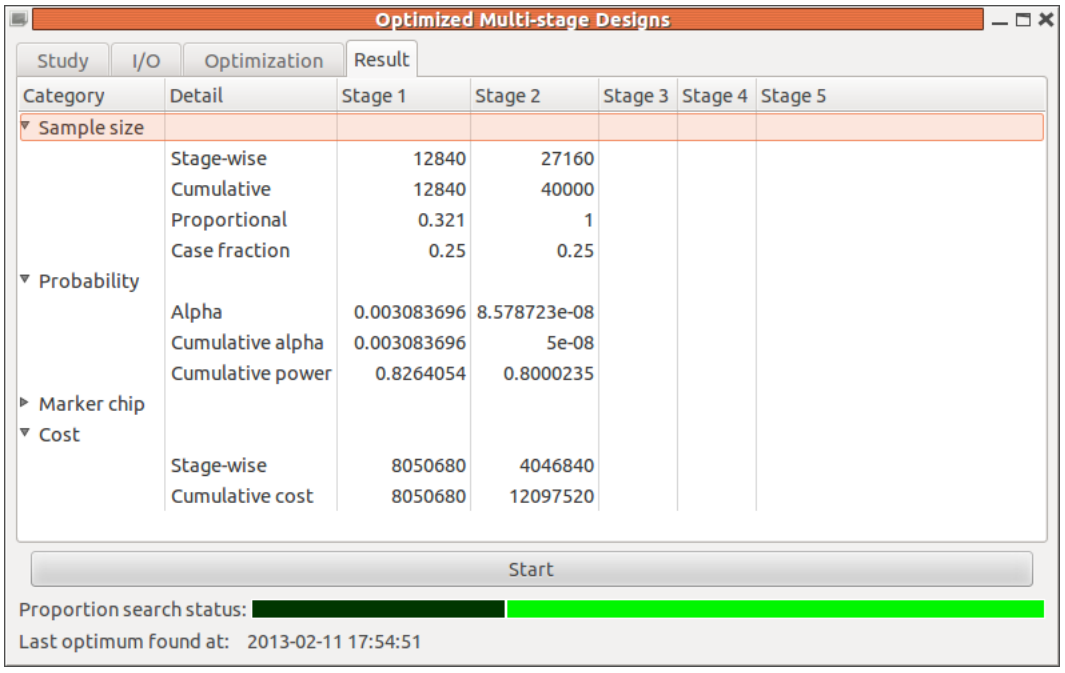


Figure 20: Result tab after the second run with forced full sample (30s runtime limit).

$1.21 \cdot 10^7$ are just about 160,000 worse than that of our initial 2-stage design. While the applied marker chips remained the same for both designs, the number of subjects at stage 1 has decreased from 13,334 to 12,840. This is reasonable, because the marker set must be shrinked significantly after stage 1 and since the second variant is using the full sample, it can (or must) drop markers earlier in the study and still retains 80% power. The signififance (or Alpha) threshold went down a notch as well in this context, namely from ~0.33% to ~0.31% so that stage 1 not only is smaller but also lets fewer markers (15 420 in expectation) pass through to stage 2. As a result, the cumulative power at stage 1 has dropped from 85% to 82.6% for the second design. On the other hand, the full sample design has a higher power at stage 2 due to a larger sample size, at the end yielding a total power of 80% as well. We recognize that the term "cumulative" power is somewhat counter-intuitive, considering that the power values are decreasing from stage to stage. It can be understood more intuitively in terms of a cumulative type II error (1-power). In this context, the type II error of the second design is higher (17.4% vs. 15%), which basically means there is a higher chance of throwing the disease marker away after stage 1. On the other hand, should the disease marker survive stage 1, there will be a higher chance to detect this marker at the final analysis, which, again, makes up for the

[11]The option to save/load parameters is planned to be integrated in a next version of GroupSeq++.

initial loss of power. All in all, expecting 15,420 markers at stage 2 while the chosen chip could carry 16 700, is the reason why this design is not optimal.

2.6.3.6 Set sample size bounds Forcing the full sample is just a special case of imposing constraints on the sample size allocation prior to design optimization, in this case fixing the sample size for the last stage at the maximum. Beyond that OMD accepts lower and upper bounds or fixed sample sizes at every stage as well as the applied chip to be used (see also section 2.5.3). To illustrate this, we start with a simple case, where the number of subjects at stage 1 is fixed at 5 000. A corresponding design specification can be found in file *2-stage-fix5k.xml* (see listing 2). We tell OMD to use this design

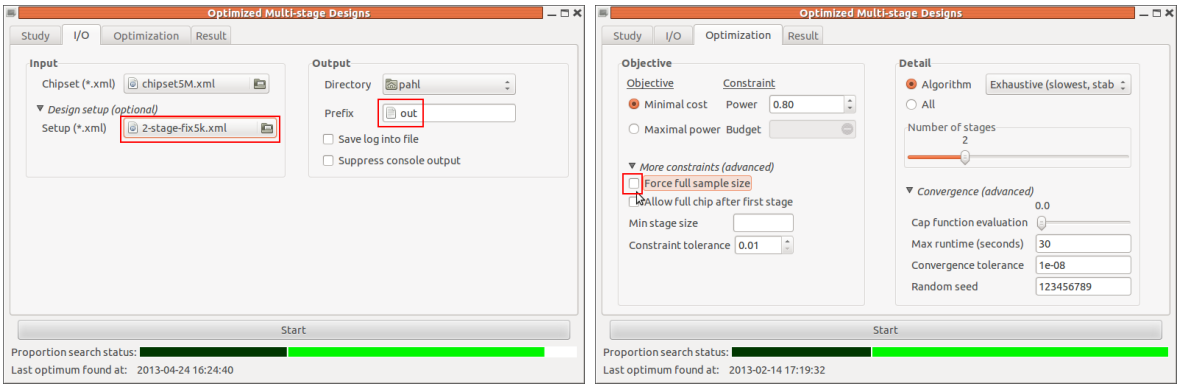
2-stage-fix5k.xml

```
<multistage_design>
    <!-- Stage 1 -->
        <stage>
        <ncum>[5000]</ncum>  <!-- Use 5,000 exactly subjects at stage 1 -->
        </stage>

        <stage>
        <!-- Stage 2 -->
        </stage>
</multistage_design>
```

Listing 2: Example 2-stage design with fixed sample size at stage 1

specification using the optional 'Design setup' option in the 'I/O' tab (figure 21a) and change the output prefix from 'full' back to 'out' again. In addition, we un-check the 'Force full sample size' option, because otherwise both sample sizes would be fixed, which would leave nothing to optimize anymore. (figure 21b). Rerunning optimization should result in the design displayed in figure 22. It becomes immediately apparent that the total costs have increased massively, which is basically a direct consequence of the imposed design restriction in combination with the available gene chips. More precisely, since the first analysis is using a much smaller sample (5 000) than the optimal design (13 334), it must not drop too many markers at this early stage to be able to retain the desired study power. The next smaller available chip (see *chipset5M.xml*) can carry 50 000 markers, which, however, is not enough to retain 80% power at the end. That is, the 5 000 subjects at stage 1 provide comparatively little information, which in fact requires to carry about 16% (799 500 markers, see 'Expected no. of markers', figure 22) of the full marker set forward to stage 2 in order to not drop any possible disease marker too early. In addition, the total sample size had to be increased to almost the full sample (39 808, not shown in the figure). Now covering those 799 500 markers at stage 2, requires 16 of the iSelect50000 gene chips, which is so costly, that the resulting 2-stage design in fact is almost ten times as expensive than the 1-stage design, and thus highly suboptimal.



(a) Select design setup file (b) Clear 'Force full sample size' box

Figure 21: Prepare 2-stage design optimization with a fixed stage 1 sample size.

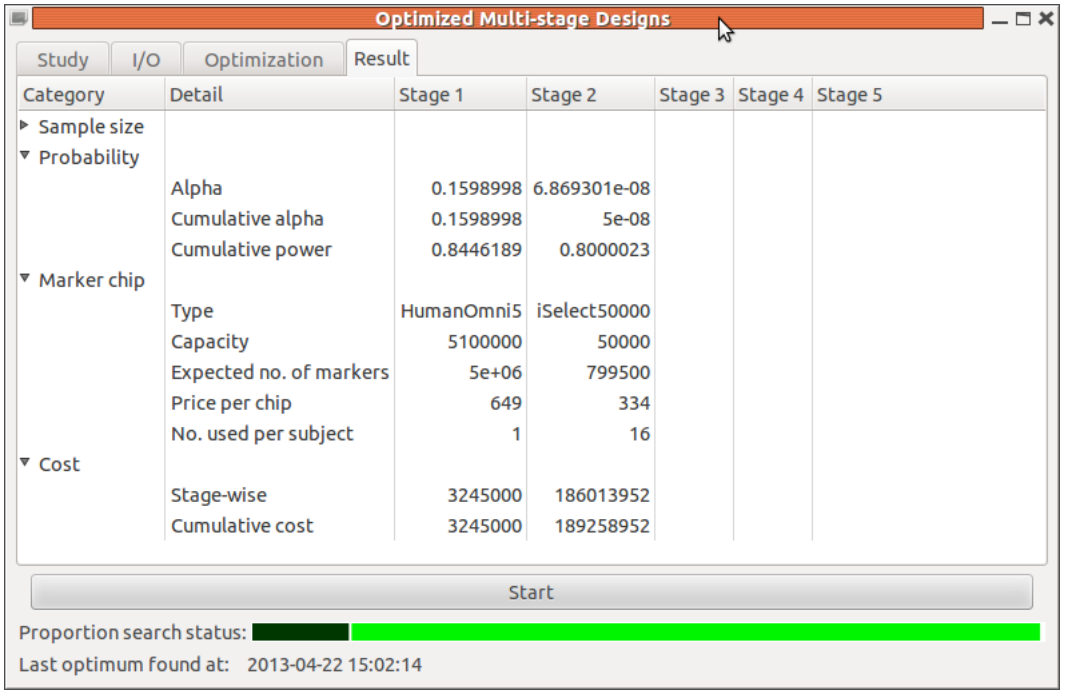


Figure 22: Result tab after 2-stage design optimization with stage 1 fixed at sample size 5 000 (30s runtime limit).

2.6.3.7 Allow full chip at stage 2 Since two chips of the iSelect50000 type are already more expensive than the initially used HumanOmni5 chip and more than 50 000 markers have to be carried forward to stage 2 in the above example, the cheapest choice would be to use the HumanOmni5 chip again at stage 2, so why did OMD not went for this option in the last optimization? The answer is: OMD by default is not allowed to use the full chip after stage 1 as this will usually not yield the optimal design, because we always could ommit the interim analysis and put both stages together to a single stage. Now fixing the first stage this early as done in the above example, has resulted in a scenario,

where ommitting the full chip at stage 2 does not yield the "best" suboptimal design. To see this, we temporary activate an option, which allows OMD to use the full chip at stage 2 (figure 23). Re-running optimization results in the design displayed in figure 24.

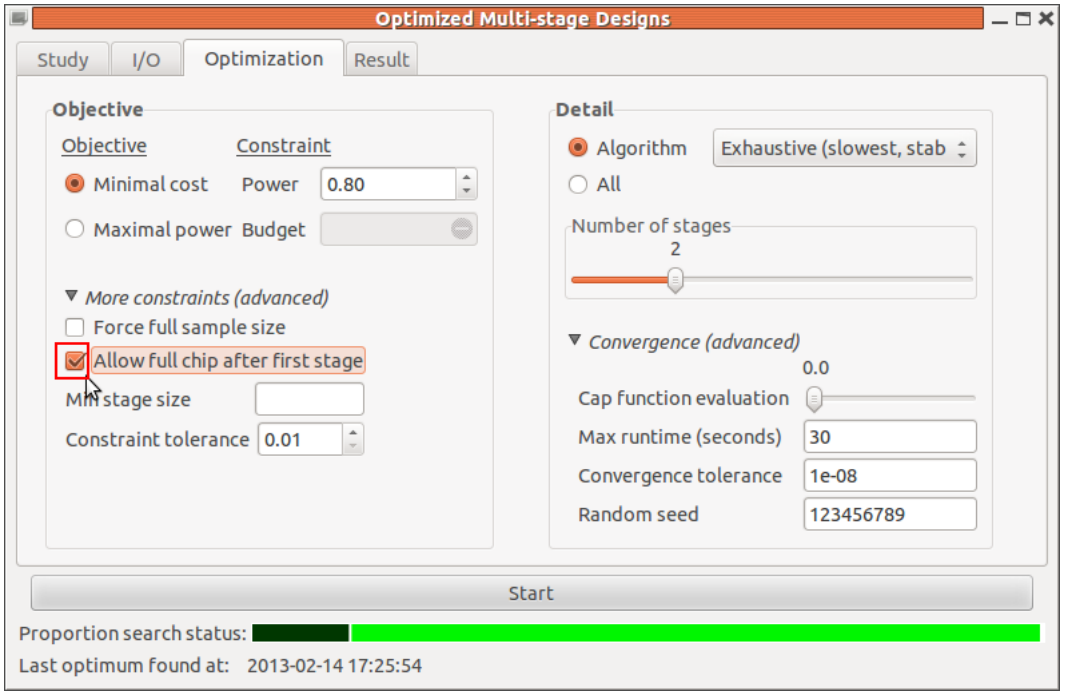


Figure 23: Activate option to allow the full marker chip at all stages.

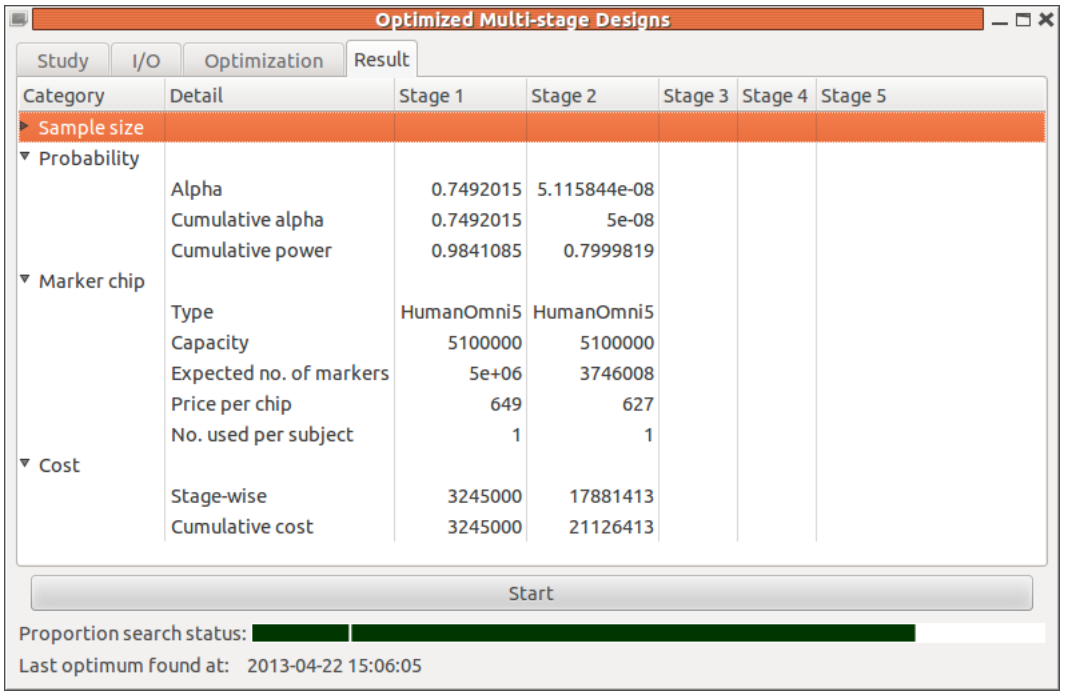


Figure 24: Re-run of 2-stage design optimization with stage 1 fixed at sample size 5 000 but using the full chip at stage 2 (30s runtime limit).

This time OMD is expecting to carry about 75% of all markers forward to stage 2 ("Alpha" at stage 1: 0.749, "Expected no. of markers" at stage 2: 3 746 008, figure 24), which retains the power easily, because a true disease marker is now unlikely to get discarded at stage 1. However, with $2.1 \cdot 10^7$ the resulting "optimized" 2-stage design is still slightly more expensive than the 1-stage design ($2.08 \cdot 10^7$). Chip-wise, of course, there is no more difference of this 2-stage design to the 1-stage design, both using the same chip on all subjects. However, since the 2-stage design has "spent some alpha" at the first interim analysis and for which it must compensate to maintain the total type I error, it requires a slightly greater total sample size than the 1-stage design (here 33 519 vs. 33 198). Thus, $33\,519 - 33\,198 = 321$ more of the HumanOmni5 chips are required, resulting in those slightly higher cost at the end.

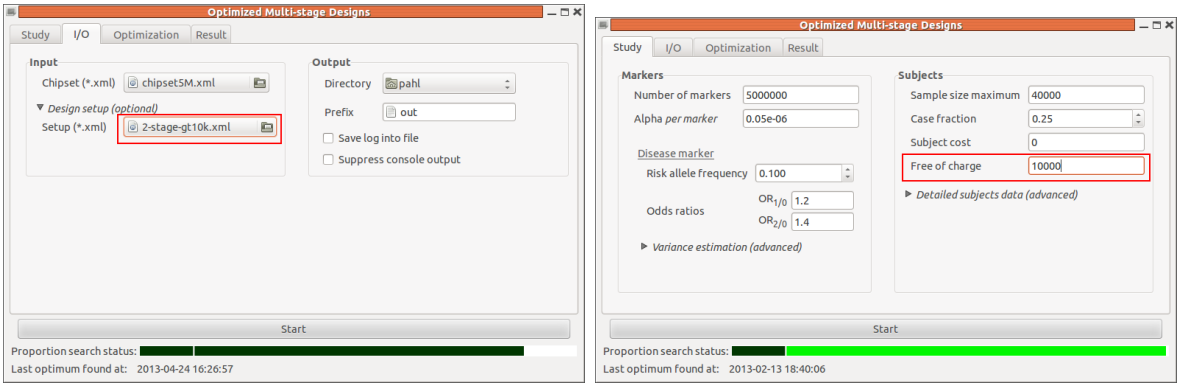
2.6.3.8 Follow-up Scenario In our next example we want to impose a lower bound of 10,000 subjects at stage one. In practice this may often occur with a kind of follow-up design, in which there is an existing sample and the researcher wants to add more data in order to further increase the study power or hunt for smaller effects. Now if there is an existing sample, genotyping has been done already for the first 10,000 subjects so that the applied marker chip is fixed as well. We therefore use the design specification from file *2-stage-gt10k.xml* (see listing 3), which is again "loaded" in the 'I/O' tab (figure 25a). More importantly, there will be no cost for the 10,000 subjects of the existing sample because they are already processed. This information can be passed to OMD in

2-stage-gt10k.xml

```
<multistage_design>
        <stage>
        <!-- Stage 1 -->
        <ncum>[10000 </ncum>          <!-- Use at least 10,000 subjects -->
        <chip>[HumanOmni5]</chip>     <!-- Use HumanOmni5 chip -->
        </stage>

        <stage>
        <!-- Stage 2 -->
        </stage>
</multistage_design>
```

Listing 3: Example 2-stage design with lower bound at stage 1



(a) Select design setup file (b) Clear 'Force full sample size' box

Figure 25: Prepare 2-stage design optimization with 10 000 cost-free subjects.

the 'Study' tab by entering the number of cost-free subjects into the 'Free of charge' field (figure 25b). We perform the optimization as usual and inspect the result (figure 26). Of course, the total cost have dropped dramatically now since the first 10,000 subjects were free of charge. However, the resulting design parameters are basically identical to the very first optimized 2-stage design (figure 16), which follows from the fact that setting the initial 10,000 subjects free of charge just works as a negative offset to the optimization problem but does not change the shape of the underlying cost function. In fact, we could have omitted the optimization and just re-calculate the cost as follows:

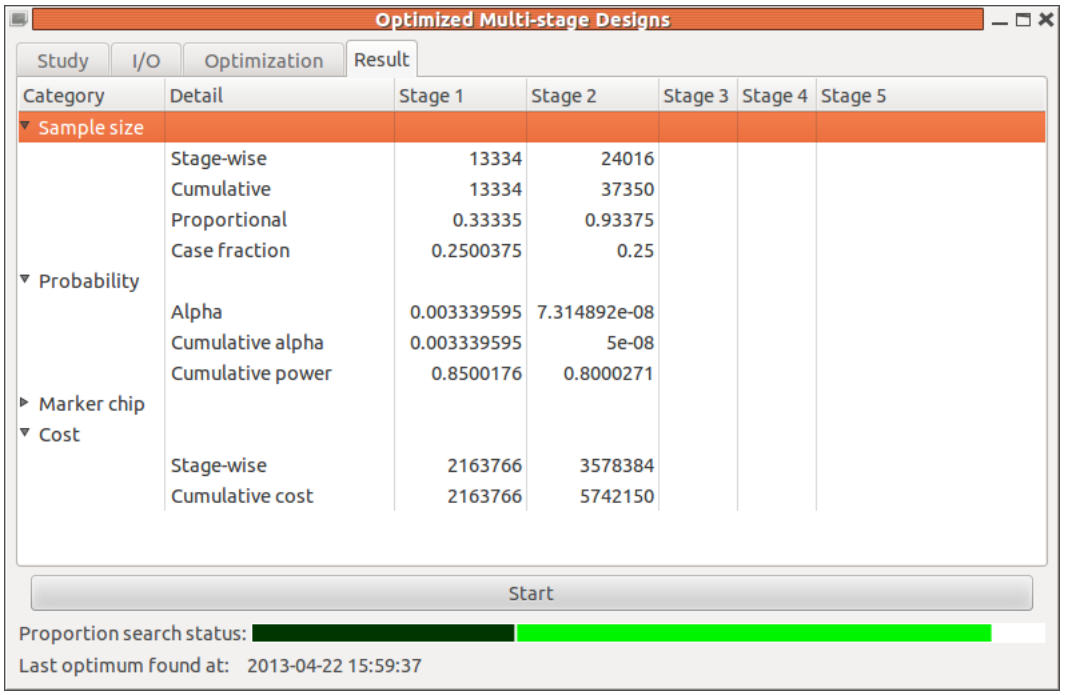


Figure 26: Result tab after 2-stage design optimization with 10 000 initial subjects free of charge.

$$\begin{array}{rrl} & 1.194 \cdot 10^7 & \text{total cost of first design} \\ - & 10000 \cdot 627 & \text{stage 1 cost of first design}^\dagger \\ + & 3334 \cdot 649 & \text{stage 1 cost of new design}^\ddagger \\ = & 5.743 \cdot 10^6 & \end{array}$$

†$627 per HumanOmni5 chip for >10000 chips
‡$649 per HumanOmni5 chip for >1881 but less <10000 chips

Note how the price for the gene chip at stage 1 has been increased due to a lower rebate, in turn being caused by just needing to genotype 3 334 additional subjects at stage 1.

2.6.4 3-stage design

The optimized 2-stage design is expecting to genotype ~16 700 markers at the second stage, which still is quite a lot. To further break this down, we carry on with the follow-up scenario and will add another stage using the the design setup contained in file *3-stage-gt10k.xml* (listing 4).

We again set the above design setup file (figure 27a), undo the 'Allow full chip after first stage' option (figure 27b), and increase the number of stages accordingly (figure 27b).

A first search still using the 30 seconds runtime limit should yield a result similar to figure 28, which presents a 3-stage design costing $7.4 \cdot 10^6$, that is, about 29% more

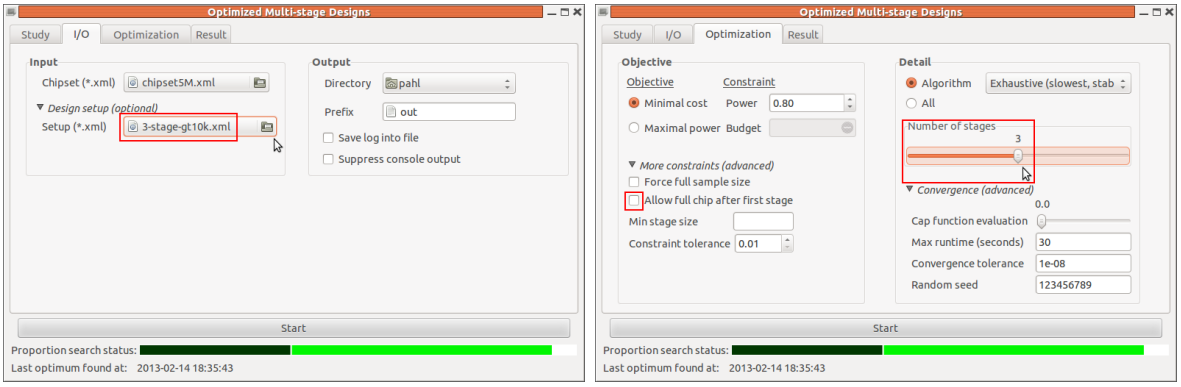
3-stage-gt10k.xml

```
<multistage_design>
        <stage>
        <!-- Stage 1 -->
        <ncum>[10000 </ncum>          <!-- Use at least 10,000 subjects -->
        <chip>[HumanOmni5]</chip>     <!-- Use HumanOmni5 chip -->
        </stage>

        <stage>
        <!-- Stage 2 -->
        </stage>

        <stage>
        <!-- Stage 3 -->
        </stage>
</multistage_design>
```

Listing 4: Example 3-stage design for a 10 000 subjects follow-up scenario



(a) Select design setup file

(b) Clear 'Force full sample size' box

Figure 27: Prepare 3-stage design optimization with 10 000 cost-free subjects.

expensive than the optimized 2-stage design. Since we expect to improve with the added stage (see also section 2.3.4.3), the design at hand cannot present a reasonable optimum. This must be attributed to the 30 seconds maximum runtime, which apparently is not enough time for a reasonable 3-stage design optimization. Considering the very smooth stage-wise sample sizes (15 000, 10 000, and 10 000) also reveals that the search was likely in an early phase of the optimization process. Another indication might be the use of two customized chips for each subject at stage 2 (see "No. used per subject" under the ▼ Marker chip category), which seems to be a rather expensive option to allocate the available chips. Before we increase the runtime, however, we will first tweak the optimization a-priori by eliminating one optimization parameter based on the following

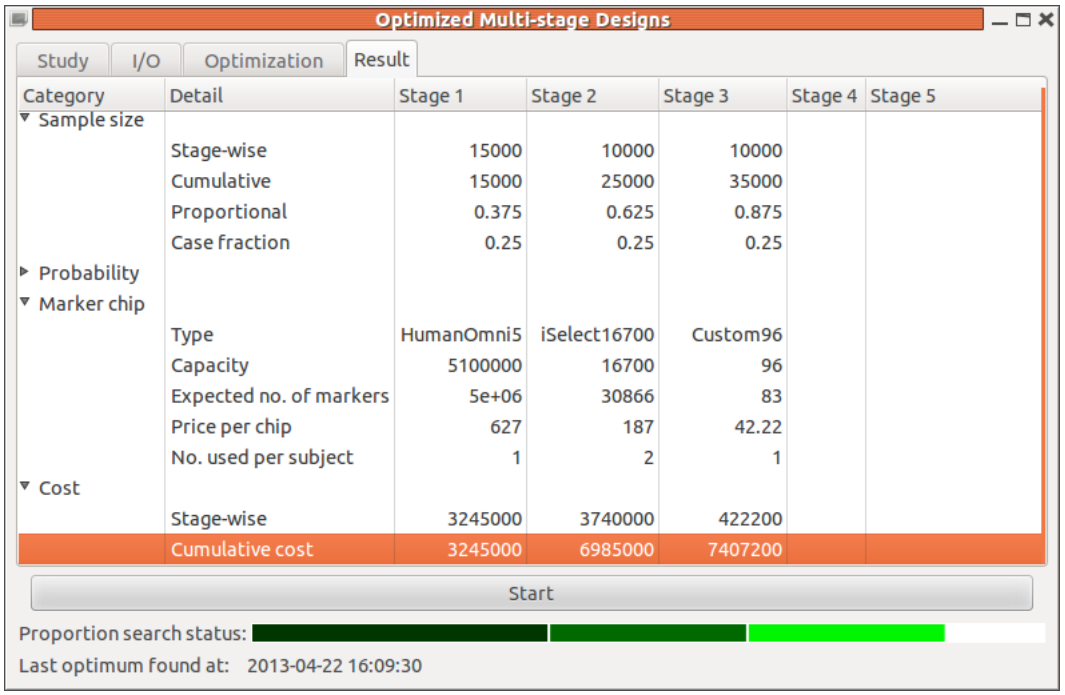
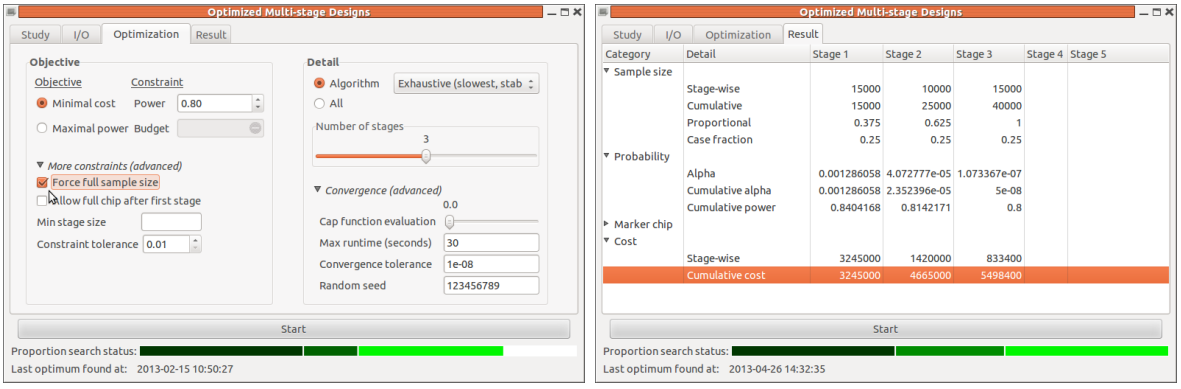


Figure 28: Result tab after 3-stage design optimization with 10 000 initial subjects free of charge.

consideration: Since the optimal 2-stage design is almost using the full sample (37 350 of 40 000), and since higher staged designs – unless bounded – always demand further increased sample sizes, we can basically assume that the optimal 3-stage design will use the full 40 000 subjects sample. Thus, we can eliminate one parameter from the search space by fixing the full sample again (figure 29a) and re-running the search now should show an improved result similar as to presented in figure 29b. In particular, the cost could



(a) Force full sample size for 3-stage design optimization.

(b) Result tab after 30 seconds of optimization.

Figure 29: Optimize 3-stage design with fixed full sample.

be further decreased to ~$5.498 \cdot 10^6$, thereby saving already $5.743 \cdot 10^6 - 5.498 \cdot 10^6 = 245\,000$ over the 2-stage design. However, the sample size numbers are as smooth as before, still indicating that the search was aborted early. If we therefore increase the runtime limit to 1 hour (figure 30), we obtain an optimized design as depicted in figure 31. This

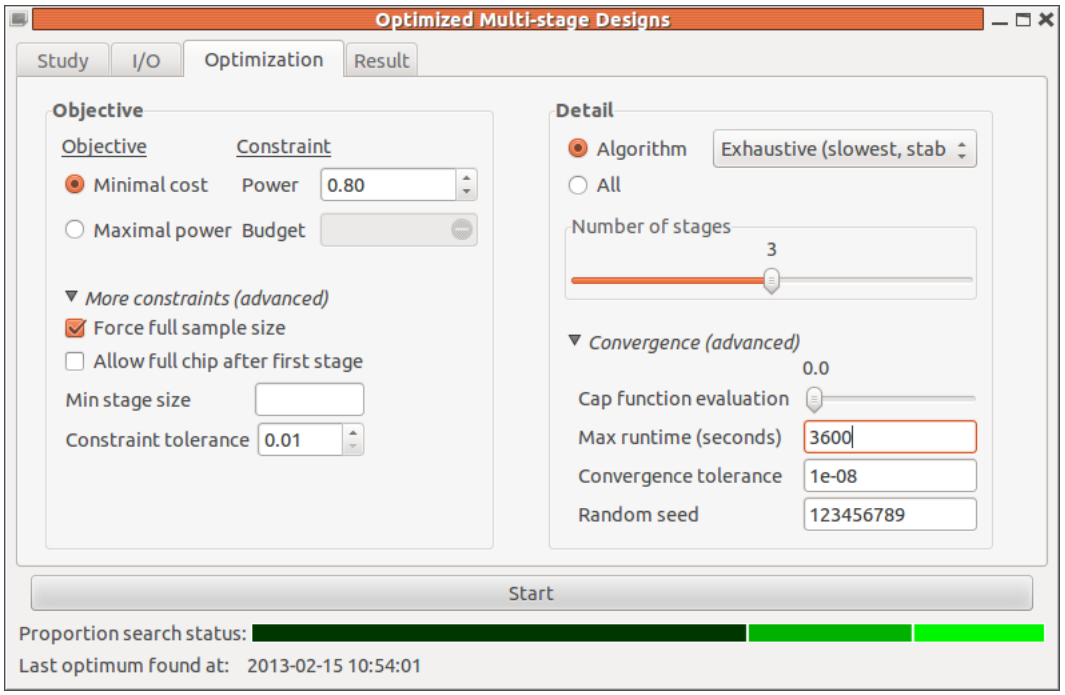


Figure 30: Increase runtime limit to one hour.

time, the cost could be decreased already to $4.8 \cdot 10^6$, thus by almost a further 1 000 000 (or ~17%) as compared with the optimized 2-stage design. As a result of the longer run, the sample sizes are finer differentiated this time. Moreover, they are basically distributed equidistantly among the stages. Inspecting the applied marker chips shows ~16 700 markers to be expected at stage 2 again, but only 22 being carried forward to stage 3, which is the major cause for the cost improvement over the 2-stage design, that is, all in all, the marker set could be decreased more gradually.

In general a 3-stage design optimization should be run at least several hours to get a reasonable result[12], depending on the problem at hand and the variety of available gene chips. Comparing the 3-stage design with the optimized 2-stage design, checking the smoothness of sample sizes and, last but not least, inspecting the convergence process from the R-console output, respectively, helps to evaluate the goodness of the optimization.

[12] After 24 hours of optimization (using one core of our Intel® Core™ i5 CPU 650 @ 3.20GHz x 4), the resulting 3-stage design showed a further cost reduction down to $4.65 \cdot 10^6$

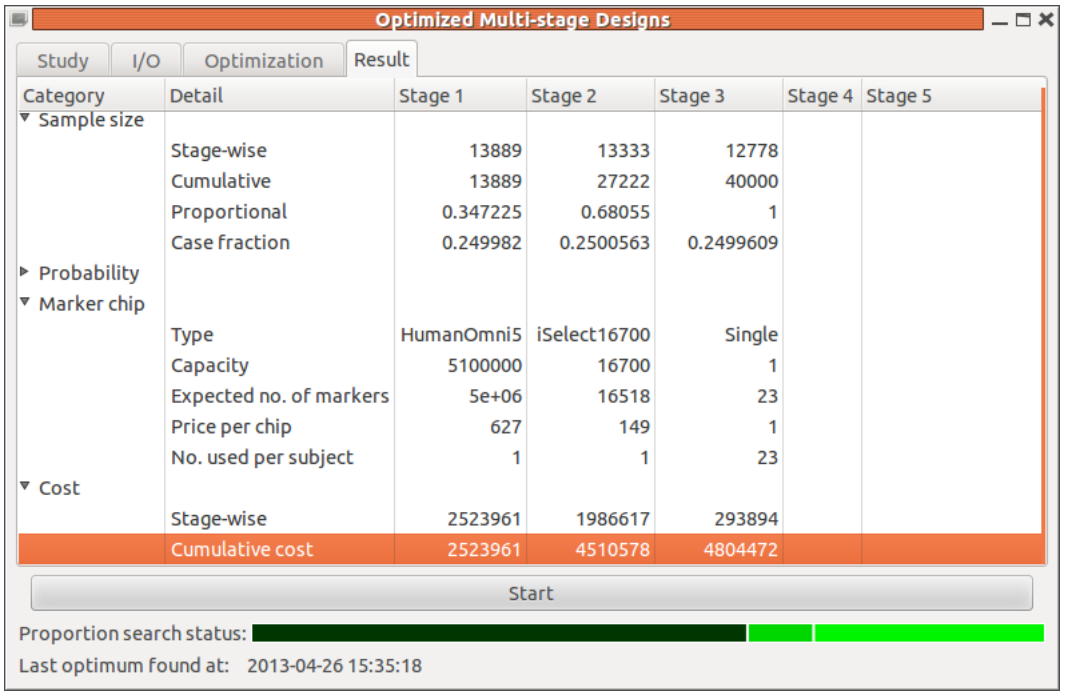


Figure 31: Resulting 3-stage design after one hour of optimization.

2.7 Use case - maximize study power

More often than not, in practice the researcher is faced with a fixed budget rather than optimizing it as a free parameter. In this scenario, he or she will be interested in gaining the maximum (study power) out of the available budget, which leads to our second optimization task: maximize the study power given a fixed budget. From a numerical point of view, the power function is "nicer" than the cost function, because it can be considered continuous, whereas the design's costs appear as a discrete step-wise function, due to the limited amount of different marker chips, as shown in figure 32. Optimizing discrete functions is notoriously difficult, because it is generally not possible to calculate gradients and OMD therefore is using derivative-free algorithms (see section 2.4.3). The smoothness of the objective function also plays an important role in that if large "flat" areas with the same function value are part of the funcion's range (figure 32), the optimization in these areas is difficult for derivative-free algorithms as well. The smoothness of the cost function in particular depends on the variety of available gene chips, that is, the more chips available, the smoother the function.

With the power function being continuous and thereby smooth, the second task to optimize the study power, at a first glance, might appear easier than optimizing the cost. However, the cost function actually is also part of the power optimization problem, namely in terms of the budget constraint. In particular, it will sometimes not be possible to match the (budget) constraint exactly, in which case the objective function will not be defined.

As a simplified example, consider the second stage of a 2-stage design with 10 000 individu-

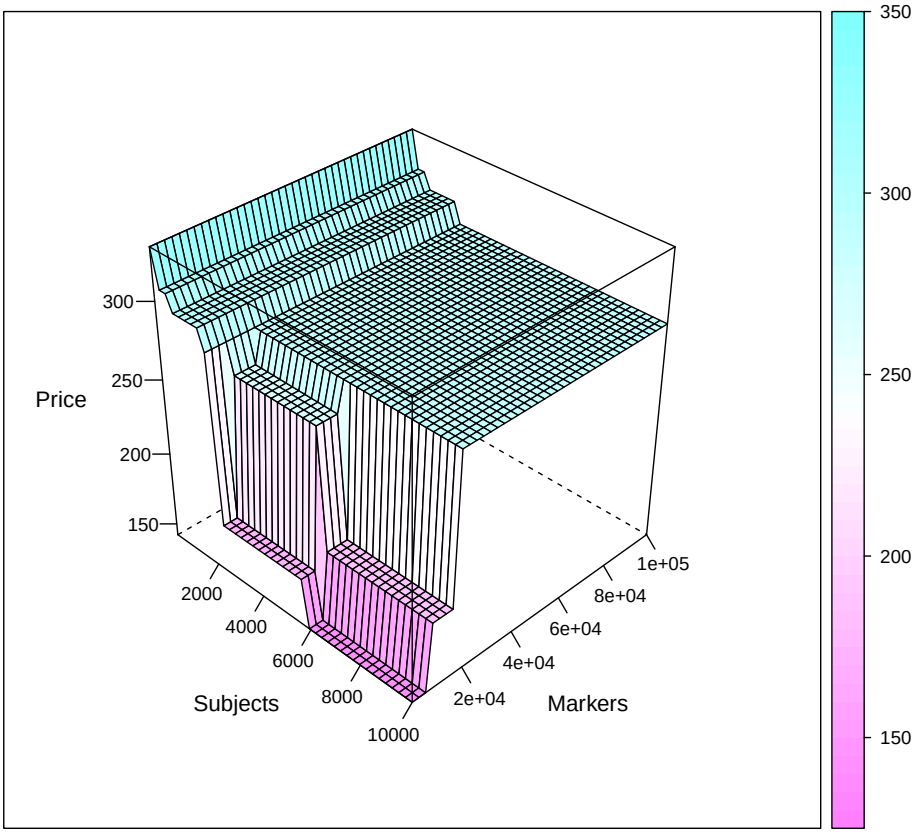


Figure 32: Price per subject as a function of both the number of subjects (determining the amount of rebate) and the number of markers (determining the chip type).

als at stage 2 and available chip capacities as displayed in table 1. As a consequence, there

Table 1: Example of available capacities and prices of a corresponding gene chip.

Capacity	96	384	7 600	16 700	50 000
Chip price [$]	40	55	140	190	420

are only five possible values for the total genotyping costs of that second stage, namely $40,00, $55 000, $140 000, $190 000, and $420 000, respectively. If we further assume that stage 1 comes at no cost, these will be the only costs obtainable by this particular 2-stage design. Now if for example the demanded budget was $300,000, there is no exact solution to this requirement for the 2-stage design at hand. Of course, the adherence to the constraint can be relaxed somewhat, but here even the closest choice (i.e., capacity of 16 700 with $190 000 total cost) would deviate by $110 000 (or about 36%) from the specified budget. The design using 10 000 individuals at stage 2 therefore has no valid chip

configuration for this budget constraint regardless of the study power it would provide.

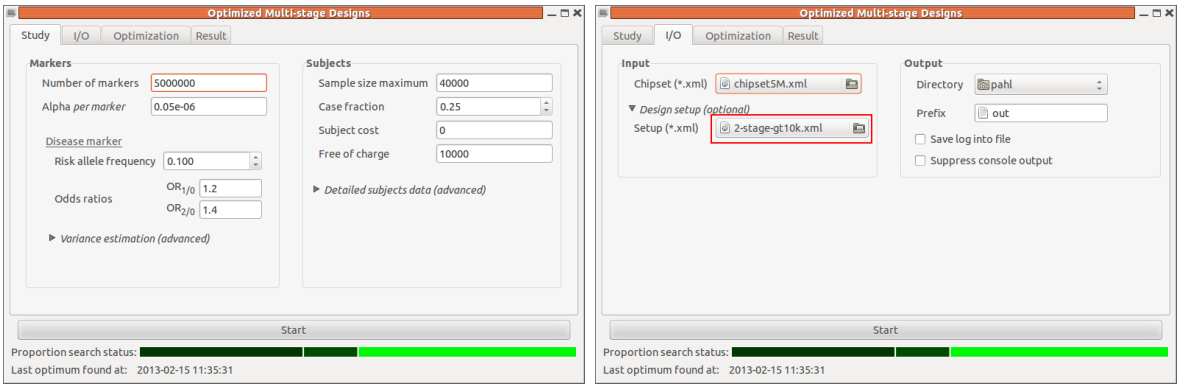
The above example shows that the power function, although being smooth in general, may not be defined for a wide range of parameters. Since this is even less information than a flat area of function values, our second task, the power maximization, therefore in fact is more difficult than the minimization of the design's cost as done in the previous section. In the upcoming sections, we present another use case and provide some options of how to deal with the constraint problem.

2.7.1 2-stage design

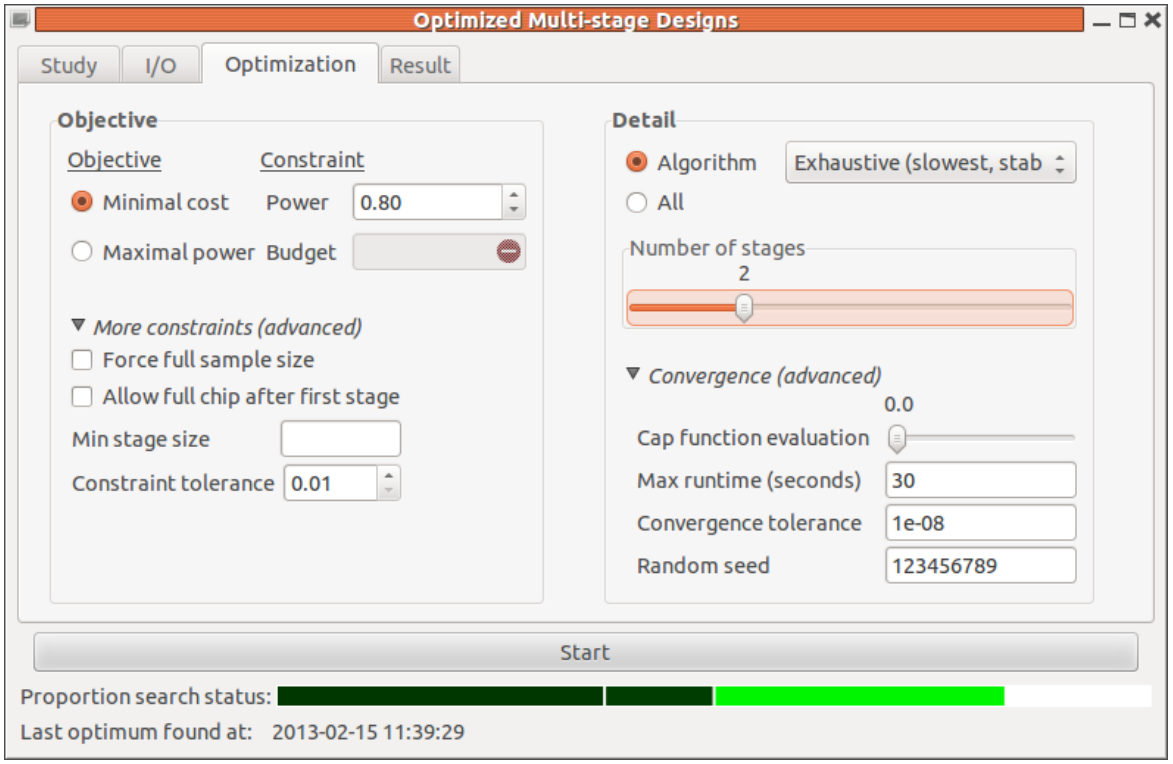
We will rely on the study setup used in section 2.6.3.8. Moreover, since we have optimized this setup already, we can rely on the concept of cross validation as described in section 2.3.4.4.

2.7.1.1 Cross validation When minimizing the study cost before, we fixed the power at 80% and obtained an optimized 2-stage design with cost of $5.742 \cdot 10^6$ (figure section 26). So if we reverse this problem by fixing the budget at $5.742 \cdot 10^6$, we expect a similar 2-stage design with 80% power. If we arrive at a design with a power significantly lower than 80%, we will know that this is not the true optimum. On the other hand, if we arrive at more than 80% power, the optimization in the last section was not truly optimal. To make sure that your parameter setup is correct, we first reproduce the corresponding 2-stage cost optimization from scratch with the parameter setup as displayed in figure 33. We run the optimization and verify that we re-obtain a design with cost of $5.742 \cdot 10^6$. Next we switch the objective and enter the obtained cost as the budget constraint (figure 34). Restarting the optimization now should finish after 30 seconds with a result as displayed in figure 35. First of all, double-checking the total cost (5 738 282, green rectangle), we see that the specified budget constraint is maintained, and so is the significance level of $0.05 \cdot 10^{-6}$ (see 'Cumulative alpha'). However, with 77.9% (red rectangle) the study power falls short by about 2% to the expected 80%.

2.7.1.2 Using a previous result as starting point Before we explain how this problem can be tackled using some of the advanced constraint options (section 2.7.1.4), we will first present a different algorithm, the 'Controlled random' search. This algorithm can take starting values, which we want to utilize in the next run by using the result of the previous one. Since the output prefix was set to 'out', the result of the previous run was saved automatically into a file called *out.2-stage.opt.power.global.xml*, which upon inspection should present as shown in listing 5. The format follows the requirements of a design file (see secion 2.5.3) and as such can be used as a design setup. Note that there are no brackets possibly defining any bounds so that all values will be merely considered as suggested values by OMD to start the optimization with. After selecting this file (figure 36a) and picking the algorithm (figure 36b), we restart the search and obtain a new result (figure 37). Again the budget constraint has been preserved and the objective



(a) Study tab (b) I/O tab



(c) Optimization tab

Figure 33: Initial parameter setup still using the cost objective.

could be increased to 79%, which is still too low with respect to the target 80% power.

The name of the algorithm implies that the 'Controlled random' search involves some randomness in its calculations. However, since OMD is using a fixed random seed for this algorithm, the search will still show deterministic behaviour. On the other hand, the user has the possibility to modify this seed, which is set to 123456789 by default. To see the effect of this, we change this value to 1234 (figure 38) and restart the optimization leaving all other parameters unchanged. As a result, the search is taking a different "path" now and the obtained design (figure 39) not only has changed but this time also is arriving at

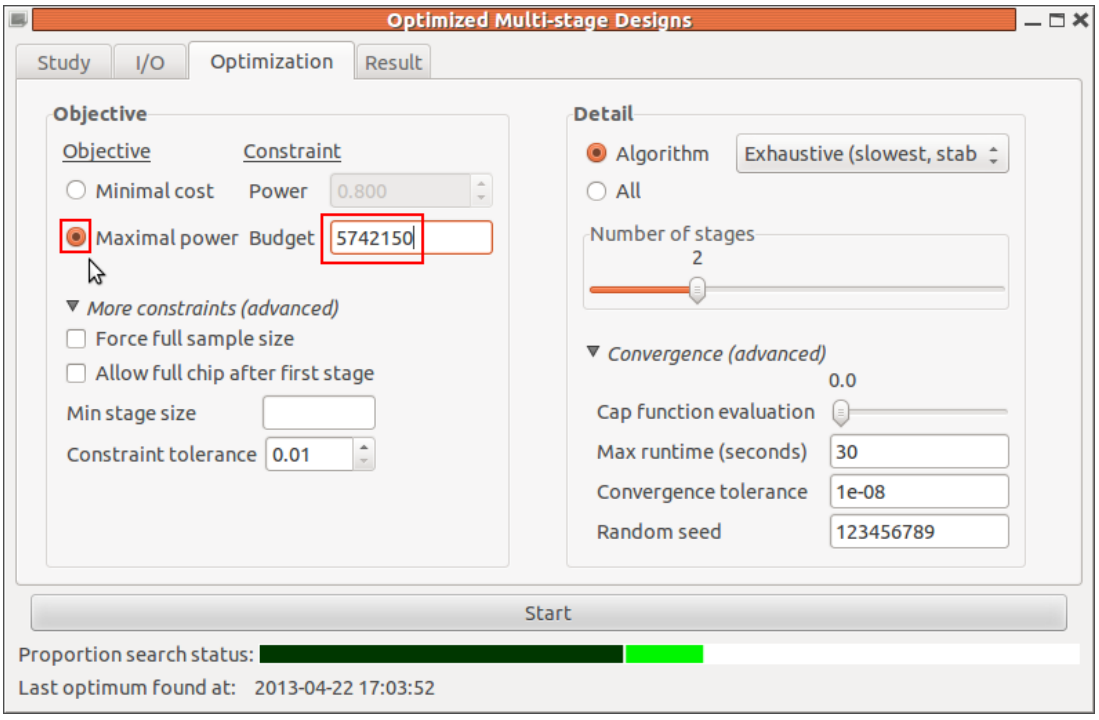


Figure 34: Prepare power objective for 2-stage cross validation.

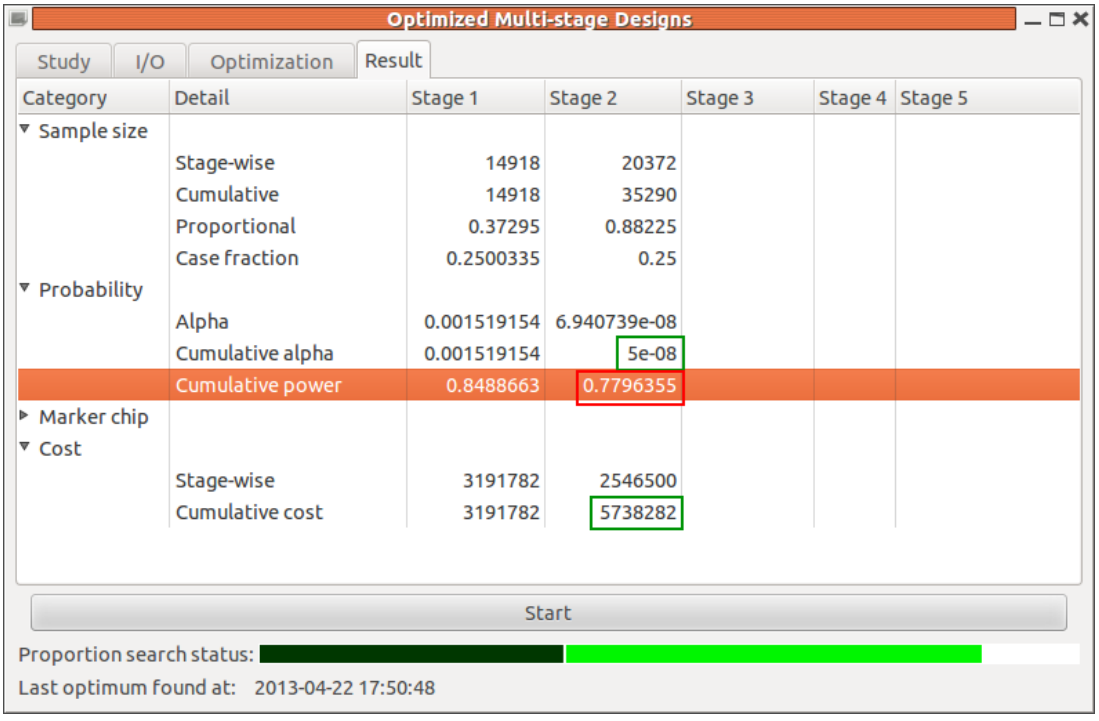
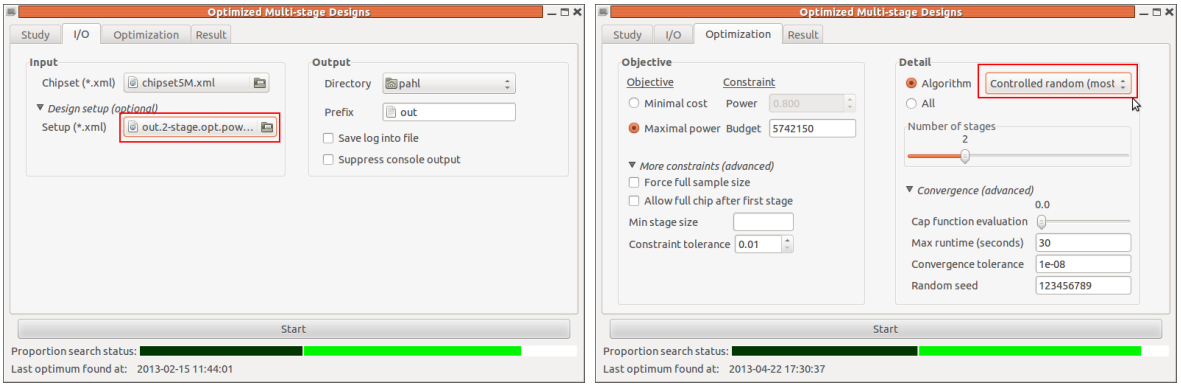


Figure 35: Result of power-optimized 2-stage design (first run, 30s runtime limit).

out.2-stage.opt.power.global.xml

```
<?xml version="1.0" encoding="utf-8"?>
<multistage_design>
        <stages>2</stages>
        <stage>
                <ncum>14918</ncum>
                <chip>HumanOmni5</chip>
        </stage>
        <stage>
                <ncum>35290</ncum>
                <chip>iSelect7600</chip>
        </stage>
</multistage_design>
```

Listing 5: Result output of optimized 2-stage design.



(a) Select design setup file of previous run.

(b) Pick 'Controlled random' search algorithm.

Figure 36

the target 80% power, with nearly exact exploitation of the pre-specified budget constraint. At this point we have therefore at least cross-validated the 2-stage design obtained in the previous section as a result of the cost-optimization. For a final verification we will apply the third available 'Local' search algorithm, possibly to locally refine the result obtained so far. To do so, we first "load" the optimized design of the last 'Controlled random' search, which was saved into a file called *out.2-stage.opt.power.genetic.xml* (figure 40a). and then pick the 'Local' search algorithm (figure 40b). Starting the search, depending on the speed of your CPU, the entire run this time will probably not make use of the full 30 seconds runtime limit but abort earlier instead with the message displayed in figure 41, stating that the "optimization has finished regularly", meaning a valid design has been obtained, but that on the other hand "no optimization" has taken place. The latter basically implies that the local search could not further improve the design of the previous 'Controlled

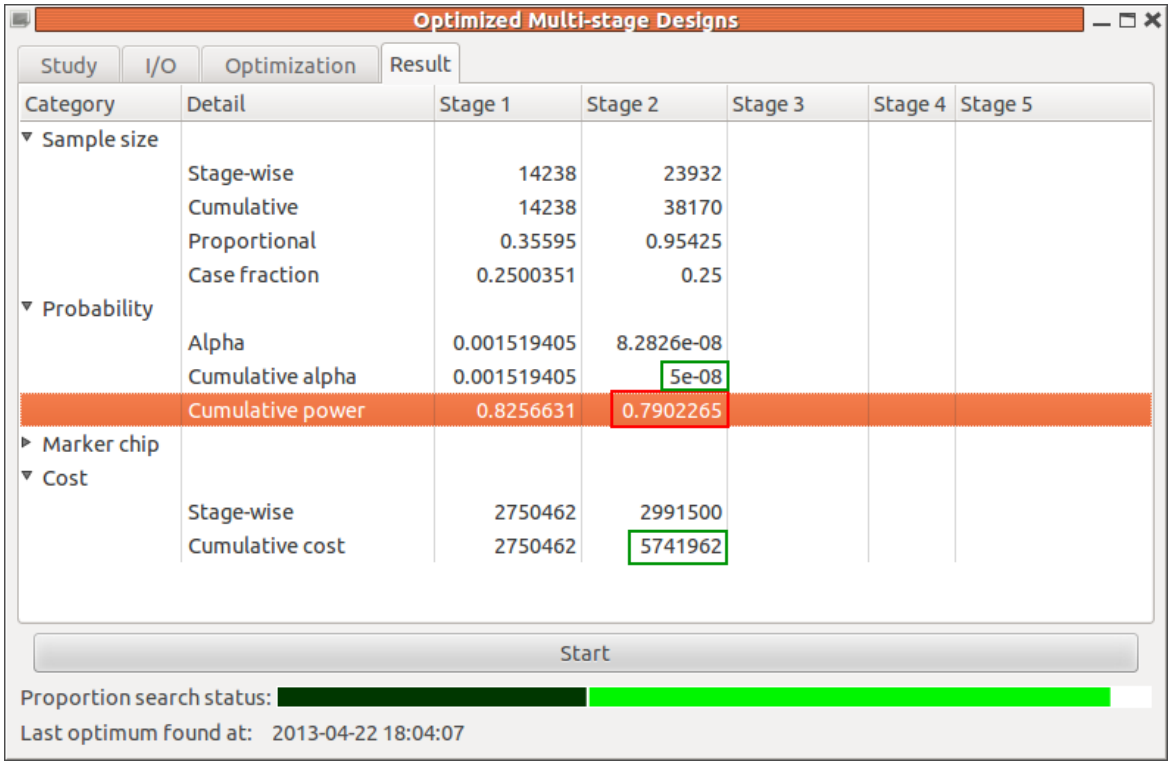


Figure 37: Result of re-run using previous result as starting point with search algorithm changed to 'Controlled random'.

random' search, therefore ending up with the same result as in figure 39.

2.7.1.3 Algorithm chain The step-wise approach presented on the last couple of pages consisting of hierarchically build and re-calculated designs based on the results of the different algorithms can be somewhat tedious and error-prone when performed manually. For this reason, OMD provides the option to do 'All' steps at once automatically (figure 42a). In particular, using this option, OMD applies all three algorithms in a subsequent chain and as soon as one search has converged, the next algorithm is invoked, using the best result obtained thus far as starting values for the next run. To make sure that the entire chain of algorithms is processed, that is, each algorithm gets applied, it is recommended to always set a runtime limit when using this option. The total runtime limit then is distributed in fractions of $\frac{1}{2}$, $\frac{1}{3}$, and $\frac{1}{6}$ among the three algorithms (for more details see section 2.4.3). Since this would only leave 15, 10, and 5 seconds for our algorithms, we increase the total runtime limits to 60 seconds (figure 42a). In addition, we assume that we have no information of previous runs yet and therefore restore the initial design setup by selecting the very first design specification (figure 42b). After about 60 seconds of optimization you should obtain a result like the one displayed in figure 43, Apparently the target power has not been achieved again and the result is deviating significantly from the one obtained by the "manual chain" before. The reason behind this is that the

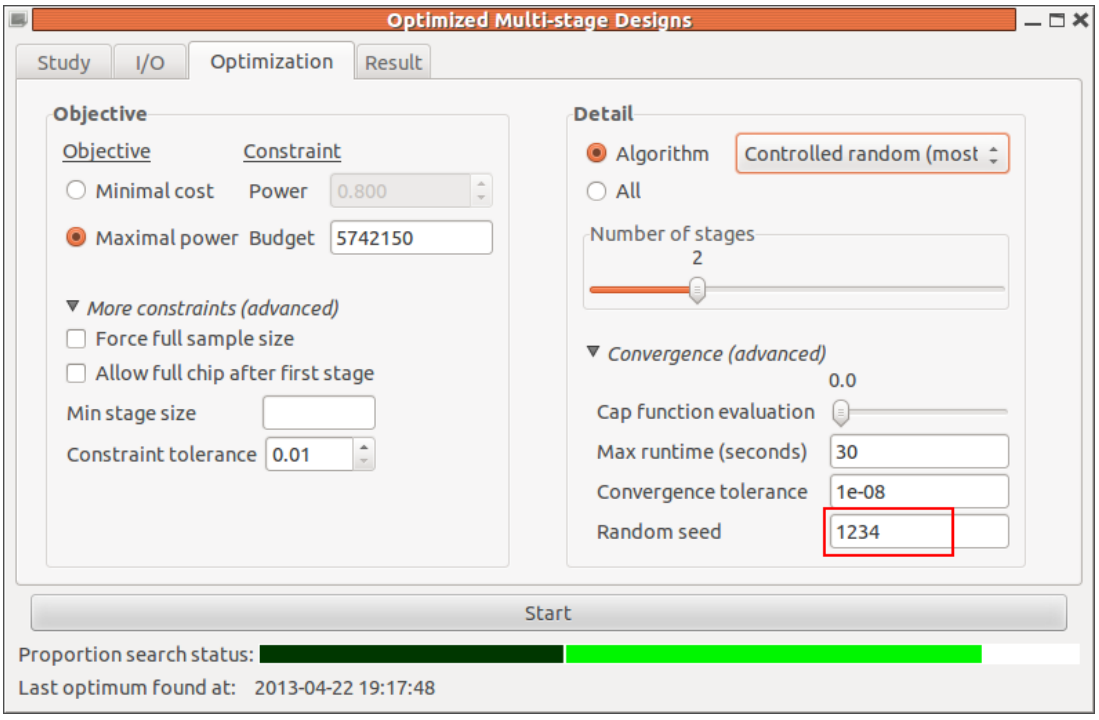


Figure 38: Change random seed for 'Controlled random' search algorithm.

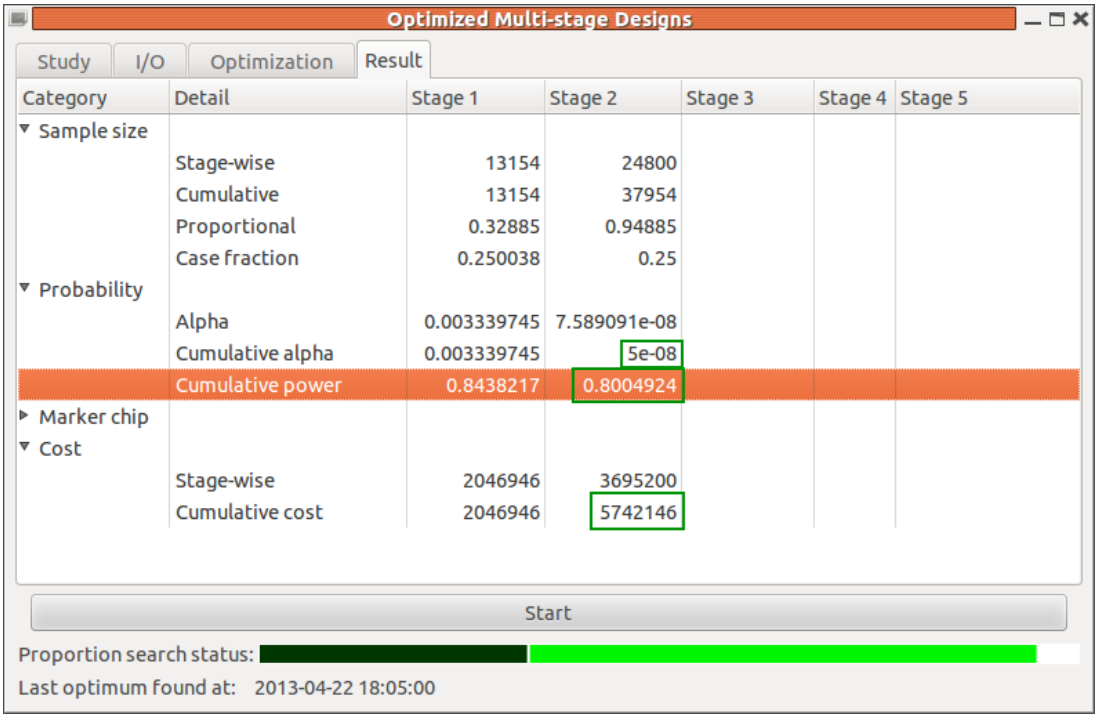
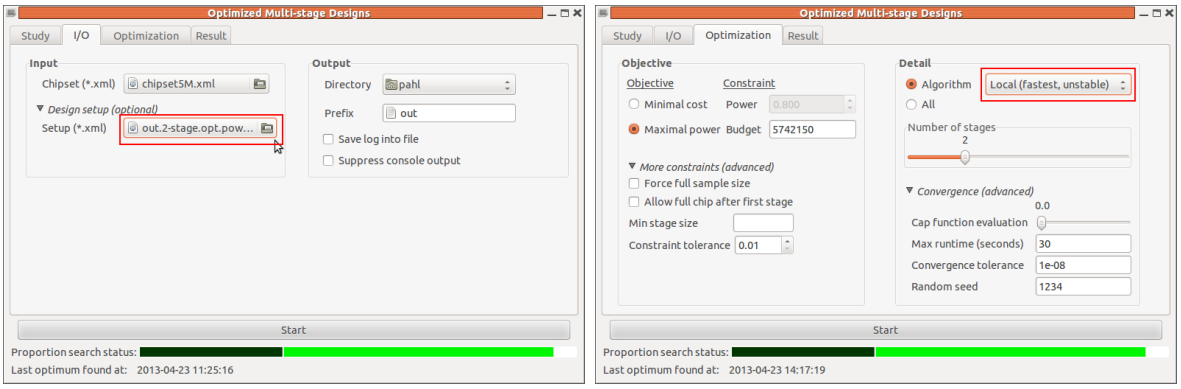


Figure 39: Result of the second run using the 'Controlled random' search algorithm with a changed seed.



(a) Select design setup file *out.2-stage.opt.power.genetic.xml* of previous run.

(b) Pick 'Local' search algorithm.

Figure 40

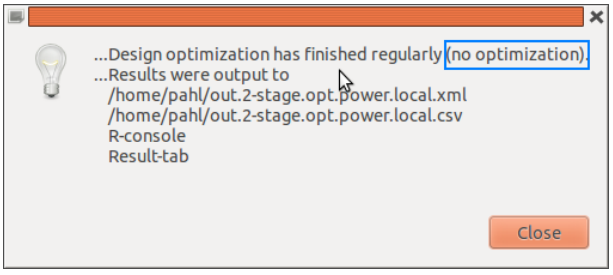
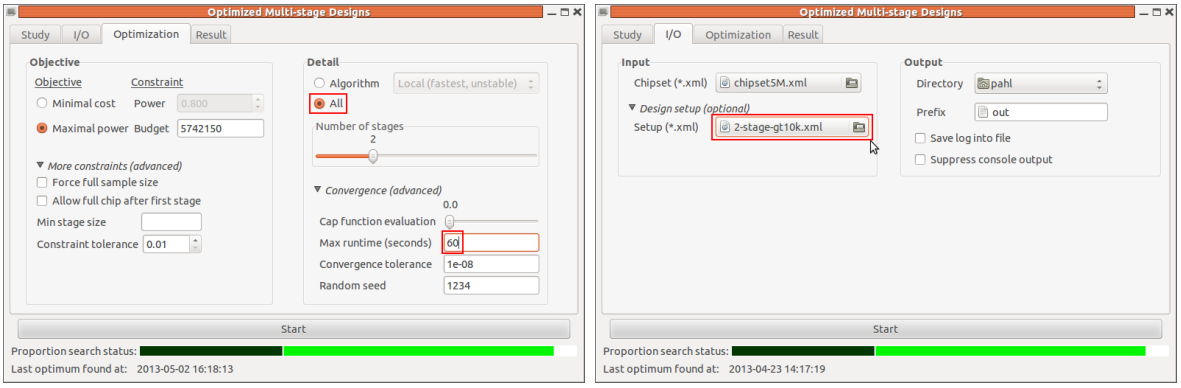


Figure 41: Dialog after finishing the 'Local' search.



(a) Pick algorithm chain option 'All' and increase runtime limit

(b) Select initial design setup.

Figure 42

seed for the "Controlled random" search, although being still set to the same value in the GUI, internally will be different, because the program already has run for some time when starting with the second search.

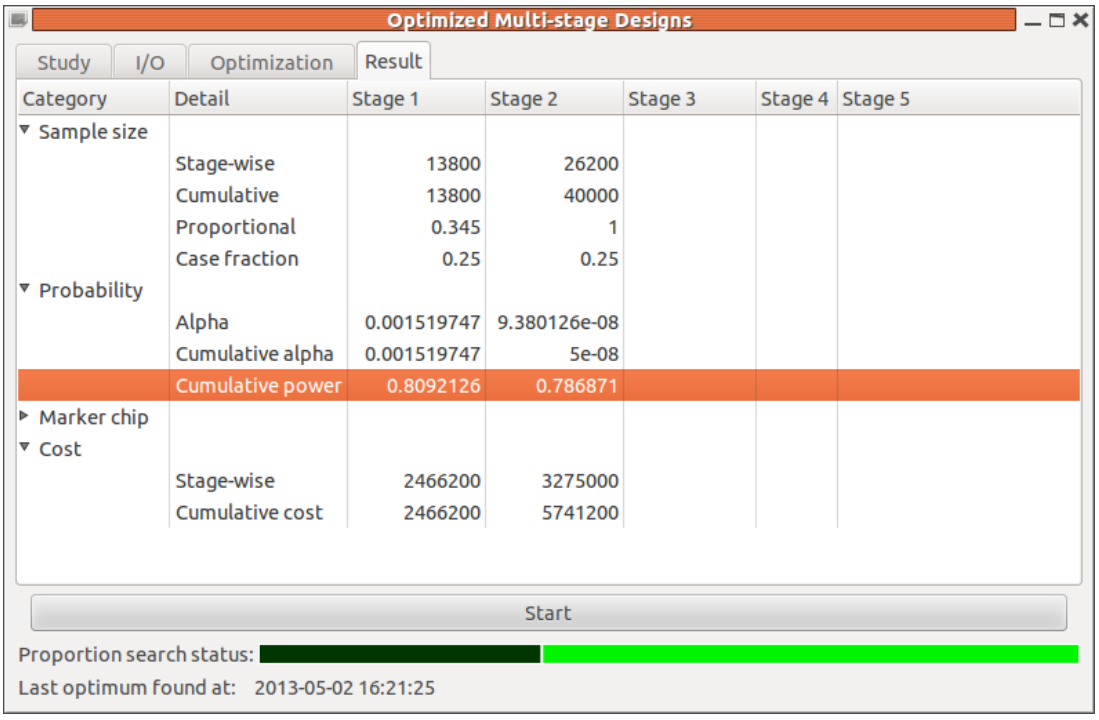


Figure 43: Result after run of the entire algorithm chain (option 'All', runtime limit 60s).

2.7.1.4 Relaxing the constraint tolerance As noted earlier the total cost of the design can be thought of as a step-wise function, because the actual cost per subject depend on both the number of markers, which determines the chip type, and the number of subjects at that stage, which determines the amount of rebate (figure 32). Due to this discrete structure, some price levels cannot be obtained exactly, which transfers to the total design cost as well, that is, a certain budget constraint may not be achievable with pinpoint accuracy. For this reason, OMD by default is using a constraint tolerance of 1%, allowing for a relative deviation of the constraint by ±1%. This is of particular importance for the optimization algorithm, because if the constraint is not fulfilled, the optimizer will get little to no information about where the objective function value is located for the computed parameter vector, typically obtaining either "undefined" or $\pm\infty$, and although there may exist a solution to the constraint near the true optimum, the optimizing algorithm may fail to find its way to this optimum, which was exactly what happened in some of the previous calculations.

The number of undefined function values to some degree can be controlled via the constraint tolerance. To see this, we increase the constraint tolerance to 10% and re-run the optimization using the entire algorithm chain (figure 44). We see that the target 80% power has been achieved again and the design overall is similar to that of figure 39.

Generally there is no problem with increasing the constraint tolerance, even by more than 10%, because OMD regardless will try to solve the constraint as accurately as possible. Basically the space of valid parameters is widened this way, which may both slightly

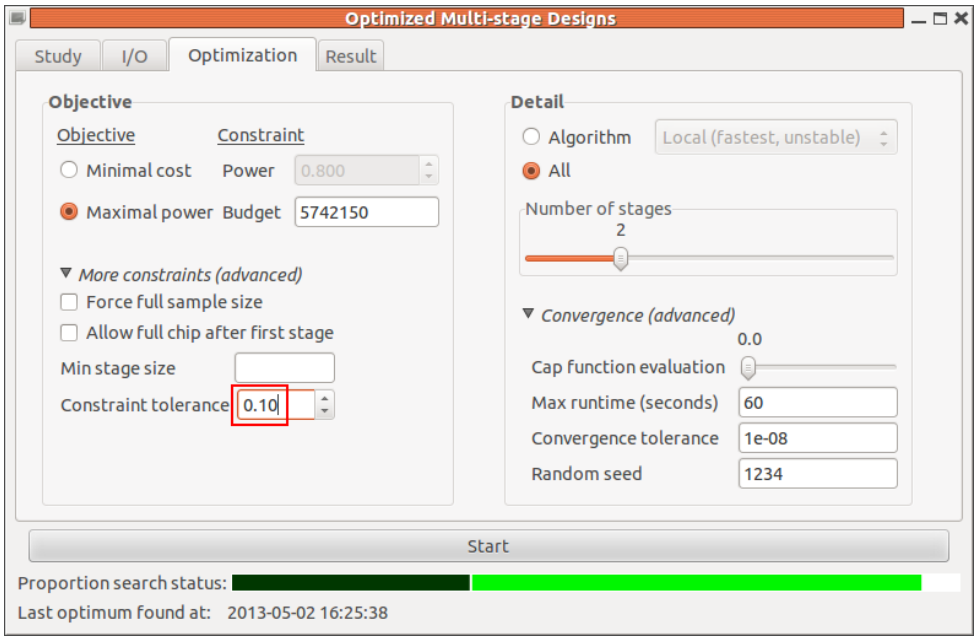


Figure 44: Increase constraint tolerance to 10%.

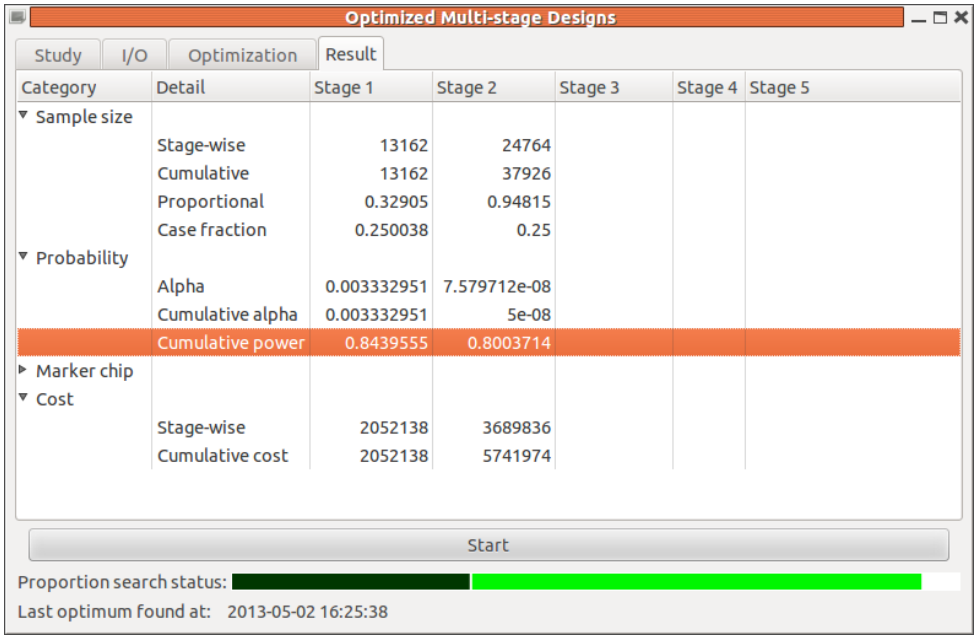


Figure 45: Result of re-run using default algorithm but with relaxed constraint tolerance.

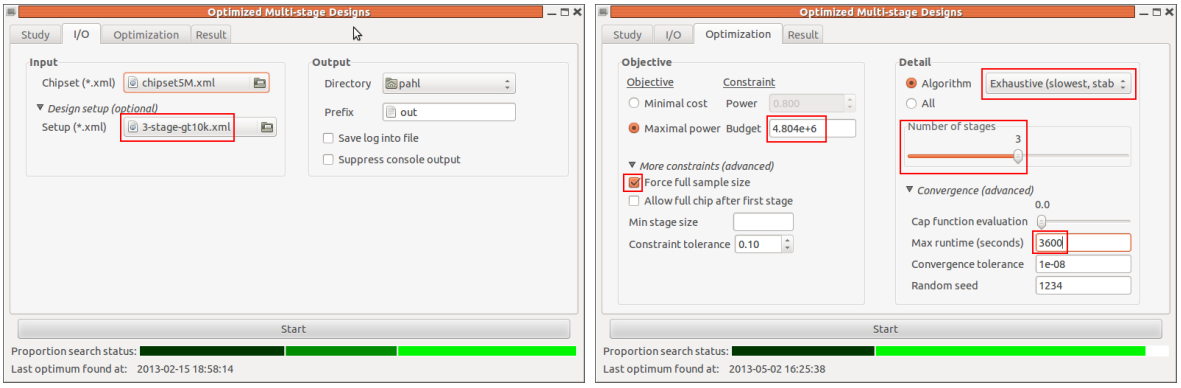
prolong the optimization process and at times make it more difficult to find the true optimum. In any case, you should always pay extra attention as to how accurate the constraint has been maintained for the final result.

To summarize cross validation should be always used to validate optimization results, especially when opting for more complex designs of three or more stages. In this context

note that the most obvious cross validation would have been to just take the design parameters that were obtained with the initial cost minimized design in section 2.6.3.8 and use them as starting values. In general, however, one should proceed as shown here and start the cross validation from fresh, because this way it is more likely to detect cases where the first optimum failed to converge to (or at least near) the true global optimum.

2.7.2 3-stage design

Next we want to cross-validate the 3-stage design that was obtained in section 2.6.4 after one hour of optimization. First, we again load the corresponding design setup (figure 46a). Second, we enter the minimized budget ($4.804 \cdot 10^6$) as the constraint, activate the 'Force



(a) Select design setup file (b) Optimization setup for 3-stage design.

Figure 46: Initial 3-stage parameter setup.

full sample size' option, and increase the runtime limit to one hour. We start optimization by using the default "Exhaustive" search algorithm (46b). Figure 47 displays the result after one hour of running time. With 77.2% (red rectangle) the resulting power is well below the cross validation target of 80%. In addition, the cost of the design ($4.394 \cdot 10^6$, orange rectangle) is about 400 000 lower than the requested budget constraint, which is almost 10%. The stage-wise sample sizes are not smooth but not very differentiated either. Alltogether, the optimization process seems to have been stopped at a rather early stage, which was kind of expected since the power maximization is the more difficult task of both. Instead of increasing the runtime, we next present another possibility to tweak the optmization process using one of the advanced options.

2.7.2.1 Set minimum stage size Considering the result of the cost minimization in figure 31, we see that the sample sizes are almost equidistantly distributed over the different stages. While the optimal design may deviate from that more or less, it is very unlikely that an optimal design contains a very tiny stage just because the smaller a stage, the less information it will provide. In practice, the benefit of a tiny stage also would often be rather small in relation to the effort of conducting that stage. With this in mind, we

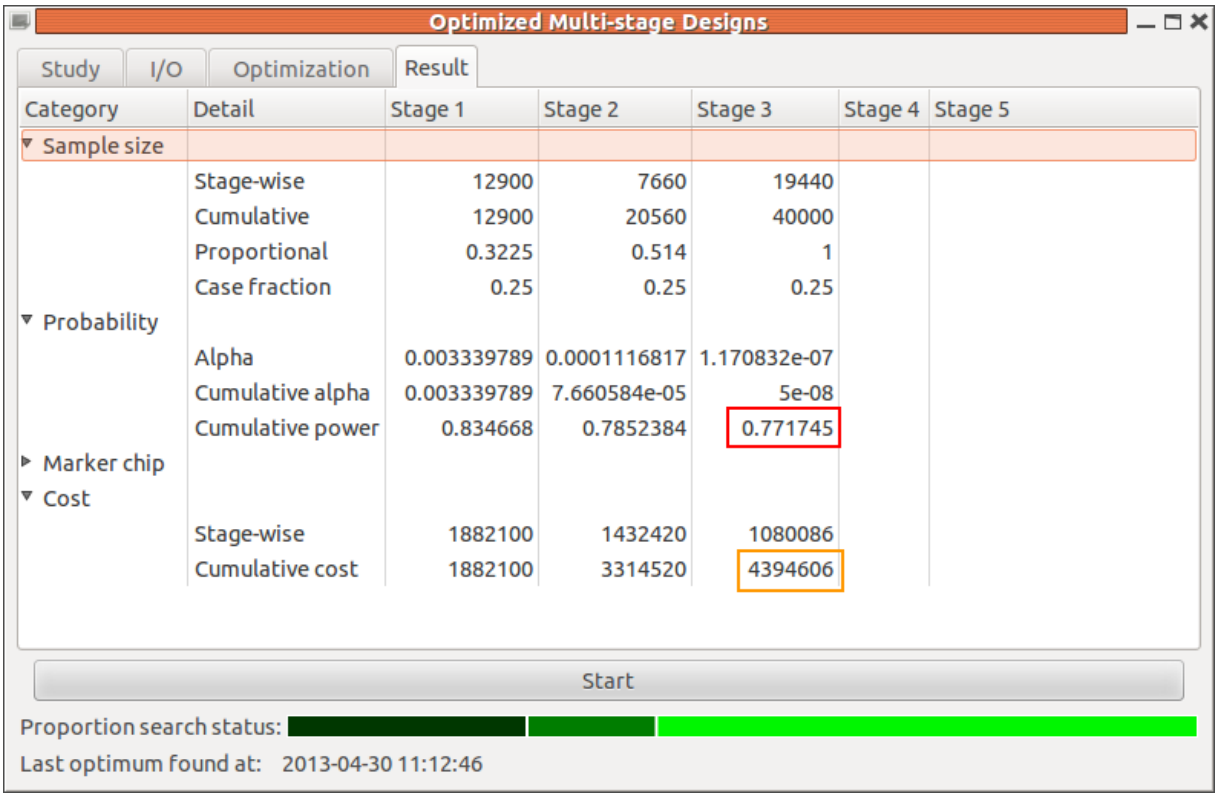


Figure 47: Power-optimized 3-stage design (1h runtime)

set a minimum stage size of 5 000 individuals (see figure 48) for our optimization problem, which is 37.5% of 13 333, the size if the total sample would be divided equidistantly over the three stages. This will limit the range of possible parameters and therefore should enhance the optimization process. Figure 49 displays the result obtained after another one hour of running time. This time with cost of $4.804 \cdot 10^6$ not only the budget constraint has been maintained accurately, but also the power is even slightly greater than 80%. We therefore would conclude that the 3-stage design obtained by one hour of cost minimization (section 2.6.4) was not optimal. This, of course, comes at no surprise since we already mentioned that 24 hours of cost minimization resulted in a budget of only $4.65 \cdot 10^6$. Thus one hour in general is too short to get stable results, but if we pretend for a moment that we did not have this information, in a next step one could go back and increase the runtime or use the same minimum stage size constraint and then re-run the cost minimization again. Another possibility would be to instead lower the budget constraint, say to $4.5 \cdot 10^6$, and restart the power optimization. We leave it to the user to try this out and develop his or her own workflow based on the presented use cases.

On a final note, as opposed to the presented use-cases, which all were started with a runtime limit, in practice is is often more convenient to start the search process without such a limit and instead use the 'Stop'-Button as soon as the user wants to abort the optimization.

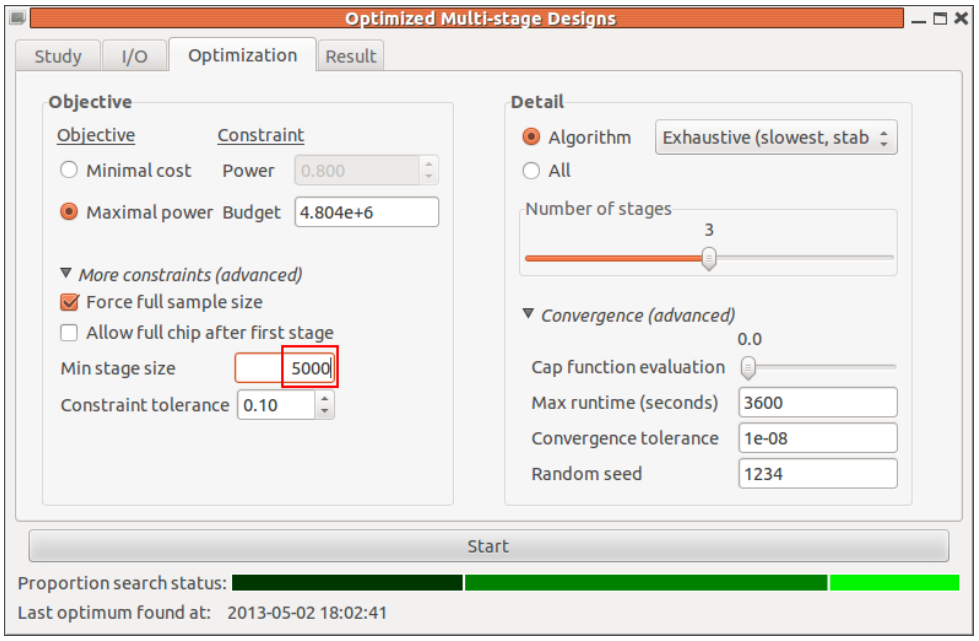


Figure 48: Enter minimum stage size to tweak optimization.

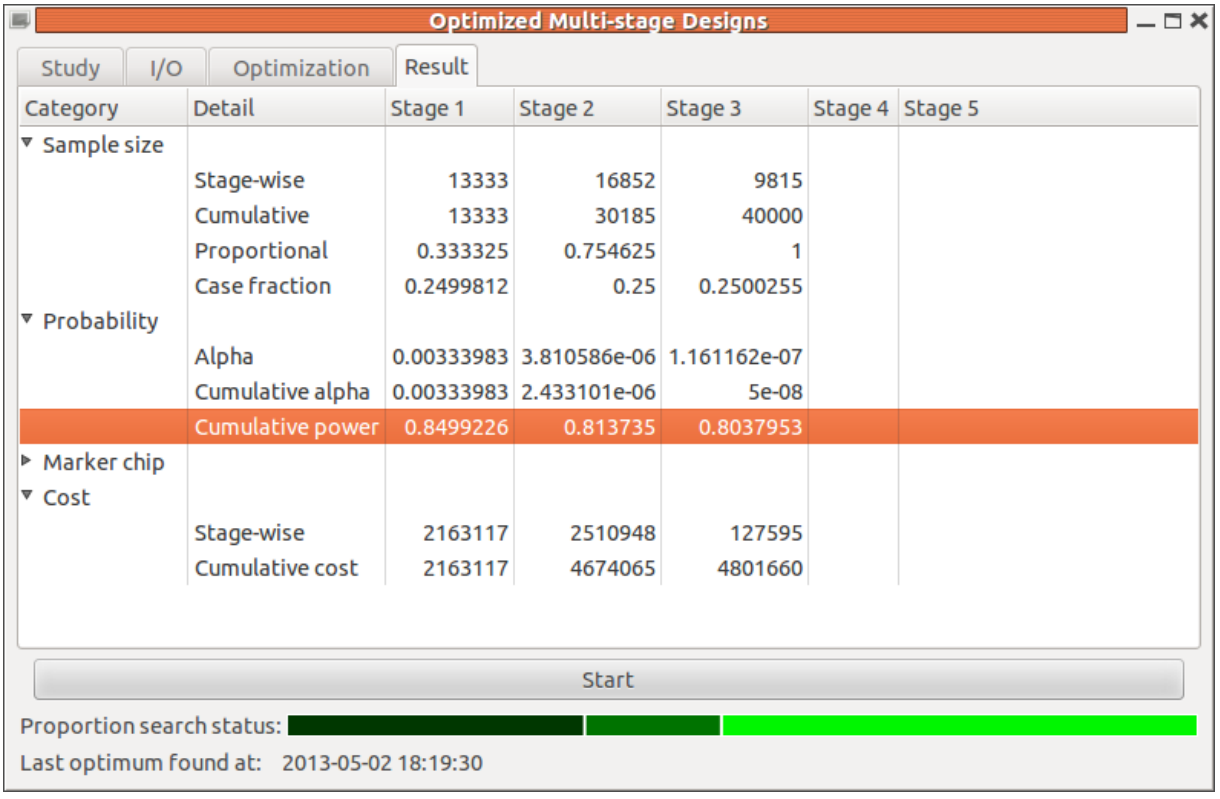


Figure 49: Optimization result using a minimum stage size of 5000 (1h runtime)

3 Flexible Two-stage Designs (CRP-Tool)

The CRP-Tool module enables the construction of flexible two-stage design for GWAS based on the conditional rejection principle (Müller and Schäfer, 2004). Again, an initially large number of markers is examined in a subsample of all available subjects, and the most promising markers shall be genotyped on the rest, combining both samples at the end in a joint analysis. When deriving optimized multi-stage designs created with the OMD module (section 2), the decision which markers are promising and thus carried forward to the next stage is solely based on the p-value of each interim analysis, that is, at each stage only those markers with the smallest p-values survive. In addition, both the size of each stage and the corresponding p-value thresholds will be fixed in advance prior to the start of the study. In contrast, the flexible design approach as provided by the CRP-Tool not only allows the first interim analysis (stage one) to be conducted at any given time during the course of the project but also the selection of markers to be based upon arbitrary attributes, particularly, not only based on the first stage's statistical outcome (e.g., p-value) alone but also upon biological (e.g. candidate gene regions) or any other criteria. Even new markers, which were *not yet* part of the study, can be included into the study after stage one in this context. Last but not least, the researcher has the option to adjust the sample size based on the first stage outcome, thereby being able to respond to outcomes not being anticipated when planning the sample size at the start of the study. Using the CRP-Tool, the family-wise error rate (fwer) is controlled in the strong sense for any selection and/or modifications being applied. Furthermore, a loss of efficiency is avoided by redistributing conditional type I error rates of all discarded markers among the markers being carried forward. Last but not least, this approach can be combined with initially optimized designs, that is, one starts with an optimized design using the OMD module and may after the first interim analysis decide to modify the design using the CRP-Tool module.

3.1 How to read the CRP-Tool manual

For most users it is recommended to read the manual from start to end. Section 3.3 provides a basic introduction to the method. For a comprehensive description we refer to Scherag *et al.* (2009). Section 3.4 explains the standard program parameters and options one by one with some extented descriptions where necessary. For a very quick start, you can jump to the last section 3.5, which provides some basic examples accompanied by further details about both the output of the CRP-Tool and flexible designs constructed this way in general.

3.2 Features overview

- Flexible marker selection
  - Arbitrary selection of markers from/after stage 1, for example, using (in any combination)
    - p-value threshold
    - biological relevance
    - markers reported elsewhere in the literature
  - Inclusion of new markers for stage 2, which were not part of stage 1
  - One-/two-sided testing and test direction both customizable individually for each marker
- Sample size modification based on stage 1 result
- Automatic choice of promising test direction for one-side tested markers
- Control of the family-wise error rate (fwer) in the strong sense, independent of the marker selection process or any other modification after stage 1
- Support of PLINK analysis format

3.3 Method overview

3.3.1 Motivation

First and foremost, the motivation behind using flexible 2-stage designs is similar to that of general multi-stage designs discussed earlier (section 2.3.2), namely saving genotyping cost by cutting down the marker set in a stage-by-stage analysis. In contrast to optimized multi-stage designs mainly targeting efficiency, the flexible design approach is emphasizing flexibility as well, of course, at the cost of dropping some efficiency in the process.

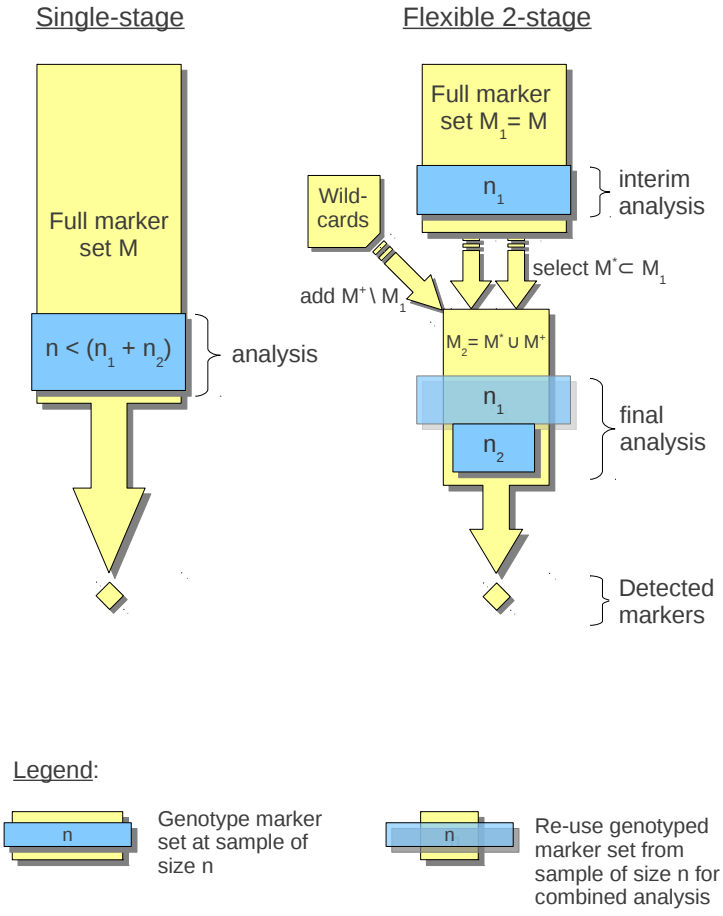


Figure 50: Standard single stage design vs. flexible 2-stage design

3.3.2 The flexible design framework in genomic studies

Using the CRP-Tool, the association study (GWAS) is conducted in two stages. First, the overall sample size of the study is fixed and a number of n_1 individuals (cases + controls) is genotyped on the initial ("full") marker set $M_1 = M$ at stage 1 (figure 50). Based on the interim results such as the p-values obtained from testing for association on the n_1 individuals or any other information observed so far, respectively, *any* subset of markers $M^* \subset M_1$ can be selected to be continued at stage 2. A set of new marker loci $M^+ \setminus M_1$ *not* being part of the initial marker set may now be included into stage 2 as well. This, however, requires to have had reserved a number of placeholders, so-called *wildcard tests* (section 3.4.2.4), before inspection of the first stage data. Othwerwise, that is, if the number of wildcard tests are chosen using the information gained from stage 1, type I error inflation is likely to occur. Similarly you will not be allowed to use genotype

information of the stage 1 sample for the final statistical test (after stage 2) of the new markers being just included this way. At stage 2, then another sample of n_2 individuals is genotyped on all selected markers $M_2 = M^* + M^+$. The samples of both stages must be disjoint while their case-control ratios should be identical[13]. The final p-values at stage 2 are calculated from the pooled sample of both stages.

3.4 Parameter and options

This section explains the standard program parameters and options one by one with some extented descriptions where necessary. A short description of each parameter can be always obtained by using the GUI's tooltips, which are accessed by moving the mouse cursor over the corresponding field. Starting the CRP-tool initially shows the window as depicted in figure 51.

3.4.1 Input/Output

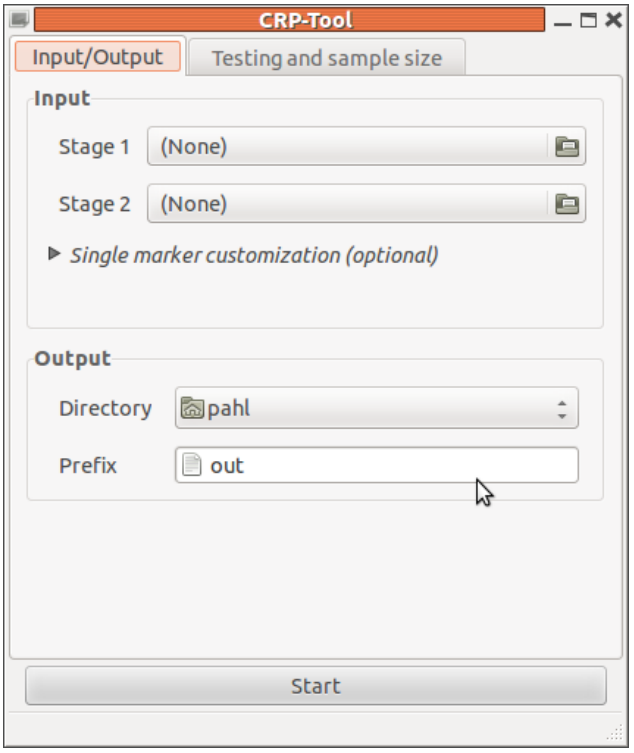


Figure 51: The Input/Output tab at program start

On a general note there are a total of five mandatory parameters without default values,

[13]The statistical model underlying the CRP-Tool is assuming identical ratios in both stages - the more this assumption is violated, the worse the fit of this model, which in most cases results in a loss of efficiency.

which therefore must be always set by the user. Two of them are located in the 'Input/Output' tab, namely both input files ('Stage 1' and 'Stage 2', see figure 51). The rest of them are located in the 'Testing and sample size' tab and initially marked with a red error-symbol in the corresponding entry field (figure 52). The input data files will contain the analyzed marker data including allele frequencies and p-values observed after stage 1 and stage 2, respectively. If the stage 2 analysis is not yet available, still interim results can be calculated for the (sub)set of markers that was chosen to be continued at stage 2 (for an example see section 3.5.1). For this, the stage 1 data of all these markers can be re-used with the stage 2 input by simply removing all markers from the file, that are *not* continued and optionally adding any new markers at the end of the list (for more details see section 3.4.1.3). By combining both data files, the CRP-Tool can determine automatically, which markers were discarded and which have been (or will be) added newly after stage 1, respectively, and also whether the second input file contains "real" stage 2 data or just a repeated subset of stage 1. If stage 2 data is provided, the CRP-Tool, in addition to the interim analysis, which is always part of the result output (section 3.5.2.1), will also provide a list with the final test decision for each marker (section 3.5.3).

3.4.1.1 Stage 1 data This file contains the analysis of the initial ("full") marker set $M_1 = M$ (see also figure 50) with one line per SNP locus including allele frequencies and unadjusted[14] The p-values can be calculated using either the Cochrane Armitage trend (Sasieni, 1997) or the allelic test statistic, respectively. In particular, the file must provide the following columns (in any order):

- SNP: name of the marker locus
- A1, A2: names of the alleles at the marker locus, A1 may be both the major or the minor allele
- F_A, F_U: frequency of allele A1 in the cases (A=affected) and in the controls (U=unaffected), respectively, in the stage 1 subsample
- P: the p-value corresponding to CHISQ
- CHISQ: (optional) the 1 degrees of freedom χ^2 statistics for association at the marker locus specified in column SNP, calculated from the stage 1 subsample

At this point, the column naming convention was chosen to follow the PLINK (Purcell *et al.*, 2007) output format of a "Basic case/control association test". PLINK is producing *.assoc files in this context which may contain some more columns not needed by the CRP-Tool (e.g. BP, OR, L95, etc.). These files can still be readily used as input files, in which case all redundant columns are simply ignored. Listing 6 displays an example file, which

[14] By "unadjusted" we mean that the p-values have *not* been adjusted with regard to the multiple testing issue of treating M markers at the same time, for example, by applying a Bonferroni correction or the like.

s1.dat

```
SNP     A1  F_A     F_U     A2  CHISQ       P
rs1     G   0.100   0.085   C   1.38418948  0.2393888135
rs2     A   0.116   0.166   T   10.33160313 0.0013077157
rs3     G   0.370   0.290   C   14.81481481 0.0001186001
rs3b    G   0.290   0.370   C   14.81481481 0.0001186001
rs4     T   0.183   0.177   A   0.11525016  0.7342442935
rs5     C   0.146   0.144   G   0.01653603  0.8976800982
rs6     A   0.515   0.482   T   2.17842697  0.1399574455
rs7     T   0.047   0.039   A   0.73954241  0.3898075005
rs8     G   0.234   0.222   C   0.41152263  0.5211976866
rs9     A   0.289   0.307   T   0.74920224  0.3867289244
rs10    C   0.328   0.274   G   6.99817606  0.0081592810
```

Listing 6: Example stage 1 data

can be found in the folder *extdata/crp* located in the installation folder of GroupSeq++. You can get the full path to the file by loading GroupSeq++ and then typing the following into the R-console:

```
file.path(path.package("GroupSeqPP"),"extdata/crp/s1.dat")
```

Note that the column headers must be exactly as indicated. For example, the following header will produce error messages:

bad.dat

```
snap    B1  FA      FU      B2  CHI         PVAL
rs1     G   0.1190  0.0890  C   9.899029895 1.653659e-03
```

3.4.1.2 Stage 2 data For this you select the file containing the data analysis after stage 2. The format is identical to the format of the stage 1 file before: The stage 2 file usually at least contains a selected subset $M^* \subset M_1$. All other markers (i.e., $M_1 \setminus M_*$) were discarded after stage 1 and therefore cannot be part of the list in the stage 2 file. In addition, a set of new markers M^+ may have been added after stage 1 (e.g., rs100 and rs101 in listing 7). Thus, in total $M_2 = M^* + M^+$ comprises all markers that have been genotyped at the second stage (see also figure 50). For each marker $m \in M^*$ thereby being genotyped in both stages the columns A1, F_A, F_U, A2, CHISQ, and P must refer to the data pooled together from stage 1 *and* stage 2. For example, for each marker $m \in M^*$, column F_A contains the allele frequency of cases as observed in the pooled data of both stages. In contrast, for each marker $m \in M^+$ added after stage 1 and thereby *not* being genotyped at stage 1 but only at stage 2, the columns refer to stage 2 data only. On a side note, carrying forward all markers from stage 1 also at stage 2 would lead to the maximal

s2.dat

```
SNP     A1  F_A     F_U     A2  CHISQ           P
rs2     A   0.1120  0.1595  T   19.489788646    1.011389e-05
rs3     G   0.3760  0.2950  C   30.290998573    3.718476e-08
rs3b    G   0.2950  0.3760  C   30.290998573    3.718476e-08
rs4     T   0.2040  0.1970  A   0.297736594     5.853049e-01
rs5     C   0.1525  0.1590  G   0.329530409     5.659357e-01
rs6     A   0.4960  0.4945  T   0.008956873     9.246002e-01
rs10    C   0.3090  0.2975  G   0.635015143     4.255217e-01

rs100   G   0.4360  0.3080  C   37.744194619    8.065639e-10
rs101   T   0.0980  0.1010  A   0.050170021     8.227677e-01
```

Listing 7: Example stage 2 data

achievable study power but, of course, at the same time result in the most ineffcient (i.e., most expensive) study design.

3.4.1.3 Using stage 1 data only (stage 2 data not yet available) Since the choice of the subset of markers that shall be continued at stage 2 is (and must be) solely based on the stage 1 data analysis, a first interim result can be calculated without any stage 2 data being actually available. The most straight-forward way to specify the desired subset is to copy the stage 1 data and then simply remove all markers from the copied file, which are going to be discared after stage 1. In addition, new markers can be added at any place in the file, which in the end will be set as the "stage 2" input file. For more details, please see the example in section 3.5.1.

3.4.1.4 Interpretation of the data Basically there are three possible setups for a marker locus – each locus has been genotyped either

1. in both stages, thereby carried over to stage 2
2. only in stage 1, thereyby discarded after stage 1
3. only in stage 2, thereby added after stage 1

No test for association will be performed for any marker that falls into the second case, that is, if the marker is not included in the stage 2 file. These marker loci have been stopped for futility and no evidence of marker disease association can be ever claimed for those. By combining the data of both files, the program can determine the status of each marker locus in the process. Table 2 shows this for the earlier presented example data files. In the above example, the set of M^* therefore consists of rs2, rs3, rs3b, rs4,

Table 2: Interpretation of example stage 1 and stage 2 data

s1.dat	s2.dat	M^*	M^+	M_2
rs1				
rs2	rs2	✓		✓
rs3	rs3	✓		✓
rs3b	rs3b	✓		✓
rs4	rs4	✓		✓
rs5	rs5	✓		✓
rs6	rs6	✓		✓
rs7				
rs8				
rs9				
rs10	rs10	✓		✓
	rs100		✓	✓
	rs101		✓	✓

rs5, rs6, and rs10. Two marker loci, rs100 and rs101, were included at stage 2 so that $M^+ = \{\text{rs100}, \text{rs101}\}$. Marker loci rs1, rs7, rs8, and rs9 were stopped after stage 1 and thus will not be tested for association.

3.4.1.5 Single marker customization In order to understand the following you should first read section 3.4.2.3, which describes the testing of markers as done by default. This section describes how these default settings can be overwritten on a marker-wise basis by including a corresponding line in the customization file. For example, if the testing is globally set to one-sided, you may want to include markers into the customization file to be tested two-sided, or vice versa. The customization file format requires five columns:

- SNP: name of the marker locus
- A1, A2: names of the alleles at the marker locus, A1 may be both the major or the minor allele
- s.A1: select allele A1 to be tested (=1) or not tested (=0)
- s.A2: select allele A2 to be tested (=1) or not tested (=0)

Consider the example file presented in listing 8. For SNP loci rs3b and rs101, the marker allele A1 is selected (column 's.A1). This means that at these loci the null hypothesis of no marker disease association will be tested against the one-sided alternative of an excess of the allele A1 in cases only (see also the section about testing 3.4.2.3). For marker loci rs4 and rs8, two-sided tests will be performed. If you set s.A1=0 and s.A2=0, no test will be performed for the respective marker locus, which has the same effect as deleting the marker locus from the stage 2 file.

custom.dat

```
SNP     A1  A2  s.A1    s.A2
rs3b    G   C   1       0
rs4     T   A   1       1
rs8     G   C   1       1
rs101   T   A   1       0
```

Listing 8: Example marker test customization

3.4.1.6 Output

- Directory: the location where output files are placed
- This prefix is used to entitle the same prefix to all output files of a single run.

3.4.2 Testing and sample size

This tab (figure 52) is used to set values regarding both the general design and statistical setup. As said before, it contains three mandatory non-default parameters to enter from the start, therefore marked by red error symbols initially.

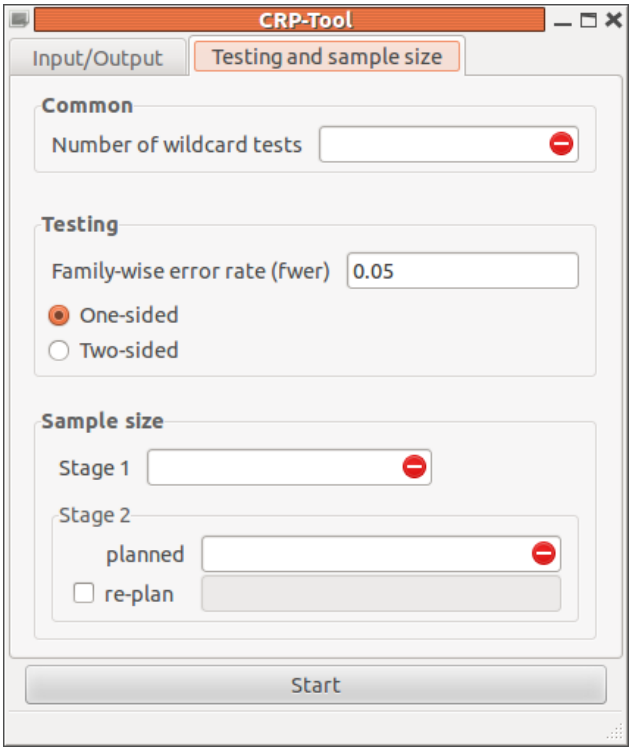


Figure 52: The Testing and sample size tab at program start

3.4.2.1 Sample size The sample sizes of both stages are entered here. Since the total sample size, n must have been planned and specified at study start before genotyping the first individual, the stage 2 sample size n_2 can be derived as soon as the stage 1 sample size n_1 is known, namely $n_2 = n - n_1$. To be exact, the sample size entered in field 'Stage 2' only refers to the number of samples used in that second stage and *not* to the total sample size n, which is computed internally by adding the values entered in both fields ('Stage 1' and 'Stage 2').

3.4.2.2 Re-planning the sample size After completion of stage 1 and prior to starting stage 2, relying on the CRP principle by Müller and Schäfer (2004), the CRP-Tool allows adjusting the sample size for stage 2, and with that the total sample size, based on the results of the interim result obtained from stage 1. For example, there might be evidence from the stage 1 analysis that the genetic effects of the promising markers are smaller (greater) than anticipated before study start, in which case one may wish to increase (decrease) the sample size in order to increase the chance of observing a true effect (decrease the study cost) at the end. Re-planning the sample size this way, as any modification of the design, preserves strong type I error control but at the same time is associated with a loss of efficiency, that is, of the ratio of power and sample size. Hence, using this option does *not* come at no cost, which makes it important to still carefully specify the target genetic effect sizes prior to study start and plan the sample size accordingly in order to avoid the need for sample size adjustment after stage 1, thereby maintaining a high statistical efficiency in the process.

3.4.2.3 One-sided vs. Two-sided testing

One-sided Using the default 'One-sided' option, a finite upper critical limit c_+ is calculated for each marker $m \in M^* = M_2 \setminus M^+$ (see also figure 50), that is, for each marker being genotyped in both stages, while the lower critical limit c_- is always set to $-\infty$. All calculation regarding the critical limits is based solely on the stage 1 data, which is why they are presented as part of the interim result output (for an example see 3.5.2.1). After stage 2, the final test decision of each marker is then made by comparing their final test statistic to their critical limits. The critical limit $(-\infty, c_+)$ as produced by the 'One-sided' option directly implies a one-sided (one-directional) test in this case. As usual, the null hypothesis is not rejected unless the test statistic falls outside the interval $(-\infty, c_+)$ of the respective marker.

For one-sided tests, the CRP-Tool will automatically determine the most promising test direction after stage 1 and set the direction of test accordingly (see section 3.5.2.2). In particular, if based on the stage 1 data the allele frequency of A1 is greater in the cases than in the controls, the one-sided null-hypothesis H_0 := frequency of A1 in cases $\leq$ frequency of A1 in controls (or `F_A` $\leq$ `F_U`, see listing 6) will be tested against the alternative hypothesis H_1 := frequency of A1 in cases $>$ frequency of A1 in controls (e.g., `F_A` $>$

F_U). If otherwise the allele frequency of A2 exceeds in cases over controls, A2 will be tested instead.

The one group of markers not covered so far in this section, namely all markers belonging to the set M^+, that is, markers not being part of stage 1 but newly included after stage 1, will never get one-sided testing by default. Instead the CRP-Tool by default will calculate both an upper *and* lower critical limit for those markers. Doing so takes account for the fact that, since they were just added after stage 1, at this point, there is usually no information available as to which allele is the most promising, in which case excluding one side of the test makes no sense. Should the user want to overwrite this default setting for a particular marker $m \in M^+$, for example, relying on some additional external biological information, he or she can use the customization option to do so (see section 3.4.1.5).

Two-sided If the 'Two-sided' option is selected, the program calculates upper and a lower critical limits for each marker $m \in M_2$, implying a two-sided test for all markers being genotyped at stage 2. Again, this setup can be customized on a marker-wise basis using the customization option (section 3.4.1.5).

On a general note, the advantage of one-sided tests is the possibility of avoiding unnecessary alpha spending, thereby gaining power for the remaining tests. Unfortunately, this gain will tend to be lower for marker loci which show marked differences of the allele frequencies between cases and controls on stage 1 than for marker loci where the difference between cases and controls is smaller. On the contrary, suppose that for some marker with alleles C and T you observe an excess of, say, the allele C in the stage 1 sample. As a consequence, you select one-sided testing for this marker. Should the situation reverse, however, that is, after stage 2 in the pooled stage 1 and stage 2 sample, the C allele now turns out to be significantly more frequent in controls than in cases, one will not be allowed to claim marker disease association at this locus since the possibility to test for this "direction" was excluded irreversibly earlier by setting the one-sided test.

3.4.2.4 Number of wildcard tests One important feature of the CRP-Tool is that the researcher is allowed to include additional markers into the study based on the results of stage 1, that is, *after* the stage 1 data has been observed. However, there is one condition in this context: the maximal number of markers that are going to be (possibly) added must be reserved beforehand, at least before the stage 1 results are known. To be exact, the researcher must reserve the number of *one-sided* tests[15] for association he or she may possibly add. These reserved tests are called wildcard (or placeholder) tests. At stage 2, there can be (but must not) added as many additional tests as wildcards have been reserved "earlier" and you will usually need two wildcards for each new marker in order to perform a two-sided test, or put another way, you will usually need to reserve two (*one-sided*) tests per "wildcard-marker". For example, in Table 2, two markers (rs100

[15] A one-sided test corresponds to a test for excess of a pre-specified marker allele at a certain locus, that is, there are two directional alternatives with either an excess of the allele A1 in cases or an excess of the allele A2 in cases, respectively (see also section 3.4.2.3).

and rs101) have been added after stage 1. In order to be able to include both markers for testing, at least $2 \cdot 2 = 4$ wildcard tests should have been reserved beforehand. As an exception, if the test direction of interest is known to you before starting stage 2 genotyping, meaning that you a priori exclude an excess of one of the alleles in cases, you will need only one wildcard for this particular locus. In this case, you will have to explicitly mark this exception by adding a line to the customization file (see also 3.4.1.5).

It is important to note that statistically some "alpha is spended" for each wildcard test being reserved. As such, any wildcard not being used at stage 2 is simply dropped from the final analysis but still has "consumed some alpha". Thus, each reserved wildcard that is not going to be used, leads to a *slight* loss of power so the number of wildcard tests should be chosen with some care.

To summarize, including tests after stage 1 means that the respective marker locus was not included in the stage 1 file, i.e., that it was not intended to perform an association test at this locus in the initial study design. If you decide to genotype this marker locus in the stage 2 individuals, you will perform an association test for this marker locus based on the stage 2 data only. Moreover, while you are allowed to select the markers depending on the results of stage 1, you are not allowed to include genotypic information obtained from the stage 1 sample at any of these newly included markers in the final p-values, which is why, in contrast to markers being part of the study from the start, only the data at stage 2 will be considered for computing the p-values of markers added after stage 1.

3.4.2.5 Family-wise error rate (FWER) A value between 0 and 1 (typically 0.05) defining the desired probability of erroneously claiming a marker disease association (type I error) for at least one of the markers out of all markers investigated in the study. The CRP-Tool will provide control of the FWER in the strong sense by initially applying the Bonferroni method to adjust for (the option of) testing all markers included in the initial marker set M_1 plus the number of wildcard test reserved a-priori. The CRP method then basically is re-adjusting the marker-wise Bonferroni levels depending on how many markers are continued after stage 1. The resulting marker-wise significance levels will be greater than the initial Bonferroni levels but smaller, of course, than if the Bonferroni levels were just derived for the set of marker M_2 at stage 2 in isolation, pretending stage 1 did never happen. As such, the CRP-Tool will *not* eliminate the conservatism immanent due to the intial Bonferroni adjustment.

Finally, it is important to note that, as usual, controlling the type I error cannot be ensured by statistical methods such as the CRP-Tool alone but also depends on an appropriate (prospective) study design and study procedure from the start. In this particular case, most importantly including

- announcement of the stage 1 and the planned stage 2 sample size *prior to* starting stage 1 genotyping
- announcement of the initial marker panel *prior to* starting stage 1 genotyping

- full inclusion of this initial marker panel in the stage 1 analysis file
- announcement of the maximal number of new markers to be possibly added at stage 2 *prior to* starting stage 1 genotyping, and the correct communication of this number to the program via the entry box 'Number of wildcard tests'
- selection of all continued markers (stage 2 file) *prior to* starting stage 2 genotyping

More aspects in this regard are discussed in Scherag *et al.* (2009).

3.5 Example

In what follows, we will use the data files placed in folder *extdata/crp*, which in turn is located in the installation folder of the GroupSeq++ package. We assume to have initially planned a study with a sample size of 1,000 subjects per stage (totalling 2,000) and that we have reserved six wildcard tests for potential inclusion of new markers after stage 1. The file *s1.dat* (see listing 6) constitutes our stage 1 data and based on this, we select some markers for stage 2.

3.5.1 Markers selection

First, we sort the markers at stage 1 according to their p-values (Table 3). We decide

Table 3: Example stage 1 data sorted by p-values

SNP	P
rs3	**0.0001186001**
rs3b	**0.0001186001**
rs2	**0.0013077157**
rs10	**0.0081592810**
rs6	**0.1399574455**
rs1	0.2393888135
rs9	0.3867289244
rs7	0.3898075005
rs8	0.5211976866
rs4	0.7342442935
rs5	0.8976800982

to carry forward any marker to stage 2 with a p-value < 0.2. Using external biological information, we furthermore decide to additionally carry over rs4 and rs5 regardless of their p-value thus far. That is, the SNPs rs1, rs7, rs8, and rs9 are stopped after stage 1 and thus will neither be genotyped at stage 2 nor considered for final analysis after stage 2. We simply eliminate the corresponding lines from listing 6, this way obtaining a reduced

s1-select.dat

```
SNP     A1  F_A     F_U     A2  CHISQ       P
rs2     A   0.116   0.166   T   10.33160313 0.0013077157
rs3     G   0.370   0.290   C   14.81481481 0.0001186001
rs3b    G   0.290   0.370   C   14.81481481 0.0001186001
rs4     T   0.183   0.177   A   0.11525016  0.7342442935
rs5     C   0.146   0.144   G   0.01653603  0.8976800982
rs6     A   0.515   0.482   T   2.17842697  0.1399574455
rs10    C   0.328   0.274   G   6.99817606  0.0081592810
```

Listing 9: Stage 1 data after removing all non-carried forward markers

set of markers as depicted in listing 9. Finally, we decide to add two new markers at stage 2, rs100 and rs101, which will consume four of the six wildcard tests we had reserved initially[16]. We correspondingly add two new lines to our existing selection (listing 10). Note that we must specify the allele names, but since we have not observed any data yet

s1-select-add.dat

```
SNP     A1  F_A     F_U     A2  CHISQ       P
rs2     A   0.116   0.166   T   10.33160313 0.0013077157
rs3     G   0.370   0.290   C   14.81481481 0.0001186001
rs3b    G   0.290   0.370   C   14.81481481 0.0001186001
rs4     T   0.183   0.177   A   0.11525016  0.7342442935
rs5     C   0.146   0.144   G   0.01653603  0.8976800982
rs6     A   0.515   0.482   T   2.17842697  0.1399574455
rs10    C   0.328   0.274   G   6.99817606  0.0081592810

rs100   G   NA      NA      C   NA          NA
rs101   T   NA      NA      A   NA          NA
```

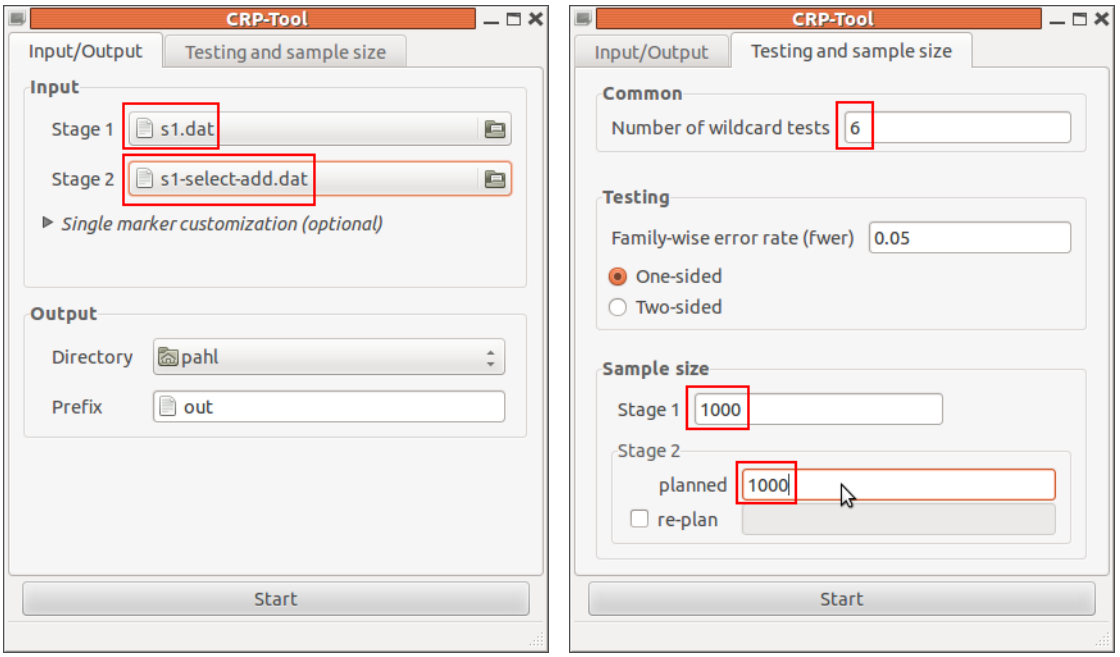
Listing 10: Adding two new markers to the existing data selection

for rs100 and rs101, all other values remain undefined, thereby being set to "NA".

3.5.2 First run

We start the CRP-Tool and select the data files *s1.dat* and *s1-select-add.dat* (figure 53a). In the 'Testing and sample size' tab, we set the number of wildcard tests and the sample size accordingly (figure 53b). Starting the computation using the 'Start' button, the status output as shown in listing 11 should appear in the R-console. Initially some information

[16]It is important to remember that the number of wildcard tests must be specified *prior* to inspection of the interim result, or in other words, it must *not* depend on the observed data.



(a) Input parameter (b) Study parameter

Figure 53: First run parameter setup

about the data input is given and how the found data has been interpreted (see also section 3.4.1.4), followed by further status output and a bar showing the progress of computation of the critical limits[17]. According to the output, the stage 1 (or interim) result has been written into the file *out.interim.csv*, which we will inspect next.

3.5.2.1 Interim result output The *.csv-format is best viewed with standard spreadsheet software such as LibreOffice Calc or Microsoft Excel. Alternatively, it can be read into R data frames using the `read.csv`-function in R:

```
> df = read.csv("out.interim.csv")
> print(df)
```

which in our case prints a table as depicted in listing 12. The first six columns are just taken over from the stage 1 input data file, however, filtered for the set of markers M_2 that were selected to be continued at stage 2, that is, only for those markers that were listed in the second input file. As a result, all other markers have been dropped from the list. Moreover, the remaining markers are now sorted by their p-values (column 'P') and the column names have changed in parts, denoting the following:

- SNP: name of the marker locus

[17] If a reasonably large marker set is analyzed, the process of computing the critical limits will take much longer and consume the major part of the total runtime.

```
...reading /home/pahl/s1.dat - found 11 markers (stage 1)...
...reading /home/pahl/s1-select-add.dat - found 9 markers (stage 2)...
...at stage 2 there are:
...7 markers continued from stage 1 and...
...2 markers added newly at stage 2...

...marking alleles to be tested - done.
...preparing interim data - done.
...preparing allele tests - done.

...Computing critical limits...
...Started at Mon Mar  4 12:20:26 2013
0%   10   20   30   40   50   60   70   80   90   100%
|----|----|----|----|----|----|----|----|----|----|
**************************************************
...Finished at Mon Mar  4 12:20:26 2013
...User runtime: 0 s

...writing results...
...interim result was written to '/home/pahl/out.interim.csv'
```

Listing 11: Status output of CRP-Tool into R-console

out.interim.csv

	SNP	A1	fCa	fCo	A2	P	A.pivot	z	crit.	crit..1
1	rs3	G	0.370	0.290	C	0.0001186001	G	3.849000	-Inf	2.90984
2	rs3b	G	0.290	0.370	C	0.0001186001	C	3.849000	-Inf	2.90984
3	rs2	A	0.116	0.166	T	0.0013077157	T	3.214280	-Inf	2.90984
4	rs10	C	0.328	0.274	G	0.0081592810	C	2.645410	-Inf	2.90984
5	rs6	A	0.515	0.482	T	0.1399574455	A	1.475950	-Inf	2.90984
6	rs4	T	0.183	0.177	A	0.7342442935	T	0.339485	-Inf	2.90984
7	rs5	C	0.146	0.144	G	0.8976800982	C	0.128593	-Inf	2.90984
8	rs100	G	NA	NA	C	NA	G	0.000000	-2.90984	2.90984
9	rs101	T	NA	NA	A	NA	T	0.000000	-2.90984	2.90984

Listing 12: Interim result of example calculation (*out.interim.csv).*

- A1, A2: names of the alleles at the marker locus, A1 may be both the major or the minor allele
- fCa, fCo: frequency of allele A1 in the cases (fCa) and in the controls (fCo), respectively.

- P: The p-value of the test of association

On the other hand, there are four new columns, which are explained in the following sections.

3.5.2.2 Pivot allele and z-score The pivot allele (column 'A.pivot') is one of A1 or A2 and defines the sign of the z-score (column 'z'), whereas the absolute value of the z-score is $\sqrt{\text{CHISQ}}$, the square root of the 1 df χ^2 statistic for association. Once the pivot allele has been set, the z-score must be always calculated relative to this allele. More precisely, if the allele frequency of the pivot allele is greater in cases than in controls (fCa > fCo), the z-score must be positive, otherwise (fCa < fCo) negative. Consider SNP rs3 at the top of listing 12. Since A1=G is set as its pivot allele and fCa = 0.37 > 0.29 = fCo, the z-score (3.849) is positive. If the second ranked SNP rs3b had also set A1 as its pivot allele, its z-score would amount to −3.849, because in this case fCa = 0.29 < 0.37 = fCo. However, using A2=C as its pivot allele, we see that the frequency relation gets reversed ($1 - 0.29 > 1 - 0.37$) so that the z-score is positive again. The actual choice of the pivot allele in general is arbitrary but for one-sided tested markers the CRP-Tool automatically chooses the pivot allele such that the z-score is positive at the interim analysis. In case of two-sided testing, simply A1 is set to be the pivot allele. Note that in both cases the choice of the pivot allele solely is based on the allele frequency observed in the stage 1 data. Using the pivot allele for the calculation of the z-score then ensures that the *direction of testing* is maintained throughout both stages. For example, consider SNP rs3 in listing 12 again. If the pooled data of stage 1 and stage 2 resulted in fCa = 0.1 < 0.7 = fCo for allele A1=G, there would be a significant difference regarding the frequency of A1 in cases versus controls, but the z-score would have turned to be negative by then, which in case of one-sided testing would not lead to rejecting the null hypthesis (see also section 3.5.2.3).

3.5.2.3 The critical limits crit- and crit+ The critical limits refer to standard normal z-score test statistics and first of all provide an interim status of each marker after stage 1, that is, the more the z-score of the marker is lying outside the interval given by the critical limits, the higher the chance of the marker to be significant at the end, and vice versa. For example, the top marker in listing 12, SNP rs3, has a z-score of 3.849, which is well above the upper critical limit (crit+[18] = 2.9098) and thereby a promising candidate to obtain a final significant association after stage 2. Second, the presented limits can (and will) also be used for the test decisicion in the final analysis of stage 2. For this, the final z-score, which then is calculated from the pooled stage 1 and stage 2 data (or from stage 2 data only in case the marker was added just after stage 1) will be compared against these critical limits, declaring significant association if the z-score is (still) located outside them. In any case, you will not have to compute the test decision on your own, because it will be delivered with the final output (see section 3.5.3).

[18]Due to the R data frame column naming conventions, crit- and crit+ are displayed as crit. and crit..1 in the data frame output, respectively

For one-sided tested markers, the critical limits together with the pivot allele (section 3.5.2.2) determine the "test direction". In particular, the lower critical limit is always set to $-\infty$ in the one-sided case, thereby aiming at a significant excess of the pivot allele frequency among the cases after the final analysis. An example of critical limits is provided in listing 12, where the critical limits of the top seven markers imply that one-sided testing is applied to these SNPs. In contrast, a two-sided test will be applied to the last two markers in the list, rs100 and rs101, recognizable by their symmetric critical limits, which is done automatically for all markers just being added after stage 1 and for which therefore no interim information is yet available (see also section 3.4.2.3).

The calculation of the crit-/crit+ values depends on the various study parameters such as the number of markers, the sample size at stage 2, the number of wildcard tests, the family-wise error rate, and so on. To see this, we decrease the number of wildcard tests from 6 to 10 (figure 54) re-run the analysis, and obtain the interim output shown in

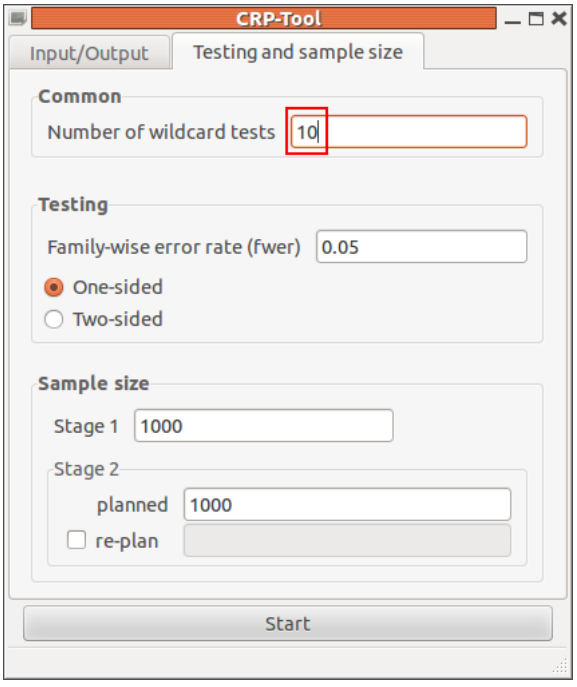


Figure 54: Set the number of wildcards to 10.

listing 13. Since the calculation relied on the same data set as before, only the critical limits have changed compared to listing 12, increasing from 2.90984 to 2.94794. These wider limits make sense as we have increased the number of wildcards and with that the total number of tests as well. As a result, the new limits have to be "more stringent" to still control the total family-wise error rate, or in other words, with an increased number of tests, the CRP-Tool must spend less alpha on average for each test.

3.5.2.4 Customizing markers The default global, either one- or two-sided testing is a reasonable choice in most standard setups but sometimes external information might

out.interim.csv

	SNP	A1	fCa	fCo	A2	P	A.pivot	z	crit.	crit..1
1	rs3	G	0.370	0.290	C	0.0001186001	G	3.849000	-Inf	2.94794
2	rs3b	G	0.290	0.370	C	0.0001186001	C	3.849000	-Inf	2.94794
3	rs2	A	0.116	0.166	T	0.0013077157	T	3.214280	-Inf	2.94794
4	rs10	C	0.328	0.274	G	0.0081592810	C	2.645410	-Inf	2.94794
5	rs6	A	0.515	0.482	T	0.1399574455	A	1.475950	-Inf	2.94794
6	rs4	T	0.183	0.177	A	0.7342442935	T	0.339485	-Inf	2.94794
7	rs5	C	0.146	0.144	G	0.8976800982	C	0.128593	-Inf	2.94794
8	rs100	G	NA	NA	C	NA	G	0.000000	-2.94794	2.94794
9	rs101	T	NA	NA	A	NA	T	0.000000	-2.94794	2.94794

Listing 13: Interim result output with number of wildcard tests increased to 10.

require additional customization of individual tests in order to be more effective with regard to the spended alpha. As an example, we will customize the way some markers are tested. For this, we "load" the file *custom.dat* (figure 55), which contains the customiza-

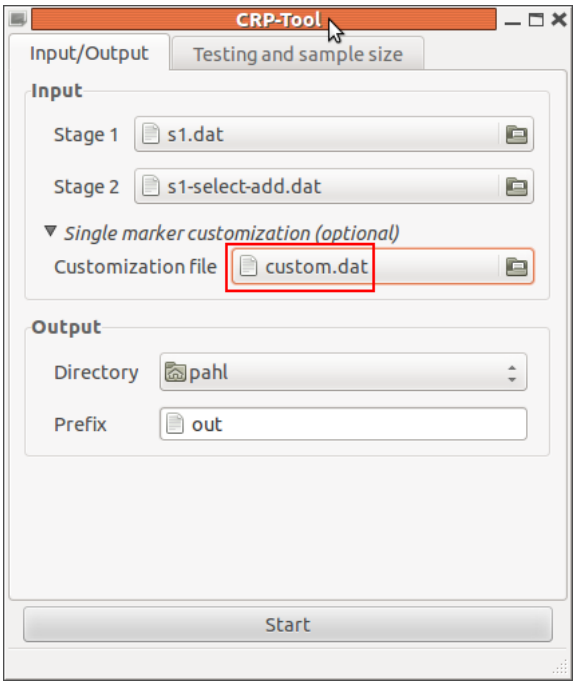


Figure 55: Load extra customization file.

tions displayed in listing 14. Particularly, for marker rs3b, an excess of allele A1 (G) in cases shall be tested and allele A2 (C) not. Since allele A1 has a lower frequency in cases (0.29) than allele A2 (0.37) at stage 1, we force to test in the less promising direction this way (see also listing 12). For rs4 and rs8 we set a two-sided test and for rs101 we omit

custom.dat

```
SNP     A1  A2  s.A1    s.A2
rs3b    G   C   1       0
rs4     T   A   1       1
rs8     G   C   1       1
rs101   T   A   1       0
```

Listing 14: Example marker test customization

testing allele A2. Re-running the tool now, besides the standard information, should show up with a warning in the R-console output:

```
Warning: Customized test file has 1 markers that won't occur at Stage 2,
which are hence ignored.
```

Reviewing listing 10, marker rs8 indeed is not part of the list for stage 2 and consequently ignored by the CRP-Tool in the process. The resulting interim output now appears as displayed in listing 15. We immediately observe a change of the sign of the z-score of

out.interim.csv

```
  X   SNP A1   fCa   fCo A2           P A.pivot         z    crit. crit..1
1 5   rs3  G 0.370 0.290  C 0.0001186001       G  3.849000     -Inf 2.93289
2 6  rs3b  G 0.290 0.370  C 0.0001186001       G -3.849000     -Inf 2.93289
3 4   rs2  A 0.116 0.166  T 0.0013077157       T  3.214280     -Inf 2.93289
4 1  rs10  C 0.328 0.274  G 0.0081592810       C  2.645410     -Inf 2.93289
5 9   rs6  A 0.515 0.482  T 0.1399574455       A  1.475950     -Inf 2.93289
6 7   rs4  T 0.183 0.177  A 0.7342442935       T  0.339485 -2.93289 2.93289
7 8   rs5  C 0.146 0.144  G 0.8976800982       C  0.128593     -Inf 2.93289
8 2 rs100  G    NA    NA  C           NA       G  0.000000 -2.93289 2.93289
9 3 rs101  T    NA    NA  A           NA       T  0.000000     -Inf 2.93289
```

Listing 15: Interim result output after single marker customization.

marker rs3b to the negative and that the z-score is now lying well within the critical limits. Thus, by choosing to test the unpromising direction as specified in the customization file, the marker has become very unlikely to be significant at the end, because after the applied customization it has to "beat" the upper limit 2.93289 while this time "starting at" -3.849 at the beginning of stage 2. For marker rs4 now two finite limits are defined reflecting the testing of both directions, while the "wildcard marker" rs101 now obviously is tested only in one direction, as was also requested in the customization file. Probably the most interesting observation, however, comes with the changed critical upper limit being

relaxed from 2.94794 down to 2.93289, which for the most part in fact can be contributed to switching the test direction of marker rs3b[19]. An intuitive explanation is that the marker had a very high z-score under the null hypothesis before, which is not the case anymore so that now "less alpha has to be spent" for this marker, which in turn can be distributed among the rest.

A similar albeit smaller effect can be observed when modifying the number of tests via the One-sided/Two-sided option. To demonstrate this, we switch to two-sided testing (figure 56) and once again restart the analysis, which delivers the results as displayed in

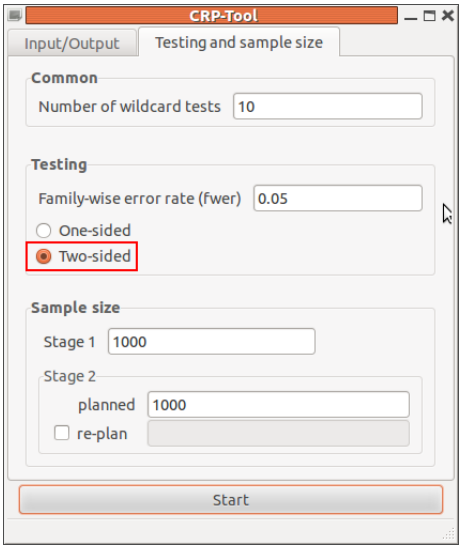


Figure 56: Set the number of wildcards to 10.

listing 16. Again only the critical limits have changed but only slightly so, increasing

out.interim.csv

1	rs3	G	0.370	0.290	C	0.0001186001	G	3.849000	-2.9329	2.9329
2	rs3b	G	0.290	0.370	C	0.0001186001	G	-3.849000	-Inf	2.9329
3	rs2	A	0.116	0.166	T	0.0013077157	A	-3.214280	-2.9329	2.9329
4	rs10	C	0.328	0.274	G	0.0081592810	C	2.645410	-2.9329	2.9329
5	rs6	A	0.515	0.482	T	0.1399574455	A	1.475950	-2.9329	2.9329
6	rs4	T	0.183	0.177	A	0.7342442935	T	0.339485	-2.9329	2.9329
7	rs5	C	0.146	0.144	G	0.8976800982	C	0.128593	-2.9329	2.9329
8	rs100	G	NA	NA	C	NA	G	0.000000	-2.9329	2.9329
9	rs101	T	NA	NA	A	NA	T	0.000000	-Inf	2.9329

Listing 16: Interim result output using two-sided testing by default.

[19] If you want investigate the influence of each customization in detail, try to modify the customization file to only contain one of the specifications a time.

barely from 2.93289 (listing 15) to 2.9329. Since each marker (except the customized ones) is tested in both directions, alpha is now spended at *both* ends (or tails) of the null distribution. Consequently, less alpha can be spended at each one side than in the one-sided case, resulting in slightly more conservative limits.

3.5.2.5 Re-plan the sample size Finally, we will investigate what happens if we re-plan the sample size. For this, we increase the stage 2 sample size from 1 000 to 1 200 (figure 57) and again re-run, yielding a new interim output as shown in listing 17. In

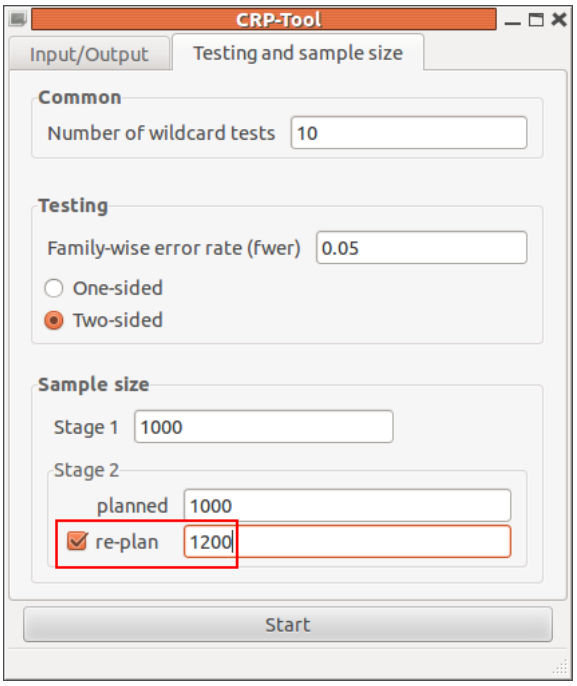


Figure 57: Modify sample size at stage 2.

contrast to all results obtained earlier, using the sample size modification, the critical limits have become individual to each marker and are no longer symmetric. An exception to this are marker loci with an interim z-score of z=0.0, which is always the case for markers added just after stage 1. As can be seen in listing 17, the more the z-score is deviating from 0 at the interim result at stage 1, the more asymmetric the critical limit turns out to be, namely in the direction that is most promising according to the interim result, or in other words, more alpha is spended in the promising test direction. As an example consider SNP rs5 with a z-score only slightly greater than 0, which reflects the fact that the frequencies of the pivot allele C do not differ by much in cases versus controls (fCa and fCo) at stage 1. As a result, the critical limits are *almost* symmetric, slightly in favor of the pivot allele regarding the possibility to declare a significant excess yet still at the end of stage 2. On the other hand consider the top, most promising, SNP rs3, which is spending the most alpha out of all SNPs on the upper (promising) and the least alpha on the lower critical limit, respectively.

out.interim.csv

	SNP	A1	fCa	fCo	A2	P	A.pivot	z	crit.	crit..1
1	rs3	G	0.370	0.290	C	0.0001186001	G	3.849000	-3.31099	2.81563
2	rs3b	G	0.290	0.370	C	0.0001186001	G	-3.849000	-Inf	3.31099
3	rs2	A	0.116	0.166	T	0.0013077157	A	-3.214280	-2.85647	3.27015
4	rs10	C	0.328	0.274	G	0.0081592810	C	2.645410	-3.23354	2.89308
5	rs6	A	0.515	0.482	T	0.1399574455	A	1.475950	-3.15829	2.96833
6	rs4	T	0.183	0.177	A	0.7342442935	T	0.339485	-3.08515	3.04146
7	rs5	C	0.146	0.144	G	0.8976800982	C	0.128593	-3.07158	3.05503
8	rs100	G	NA	NA	C	NA	G	0.000000	-2.93290	2.93290
9	rs101	T	NA	NA	A	NA	T	0.000000	-Inf	2.93290

Listing 17: Interim result output after using sample size re-plan option.

Should you want to use the asymmetrical critical limits for test decisions on your own, it is *crucial* to determine the correct sign of the z-score test statistic at every single marker locus (see section 3.5.2.2). Otherwise, the CRP-Tool is taking care for you already in this regard and includes the correct test decision in the final output, which is explained in the next section 3.5.3.

3.5.3 Stage 2 analysis and final test decision

For stage 2 we select the accompanying data file *s2.dat* (figure 58) and leave everything else as is. Running the CRP-Tool once again, we notice one additional line at the end

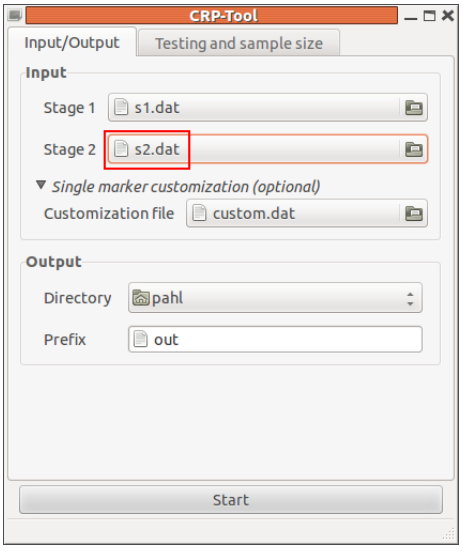


Figure 58: Load stage 2 data.

of the status output in the R-console, stating that the final ouput has been written to a

file *out.final.csv*. The content of this file is shown in listing 18. First and foremost, all

out.final.csv

	SNP	A1	fCa	fCo	A2	P	crit.	z	crit..1	assoc
1	rs100	G	0.4360	0.3080	C	8.0656e-10	-2.9329	6.143600	2.9329	TRUE
2	rs3	G	0.3760	0.2950	C	3.7185e-08	-3.3110	5.503700	2.8156	TRUE
3	rs3b	G	0.2950	0.3760	C	3.7185e-08	-Inf	-5.503700	3.3110	FALSE
4	rs2	A	0.1120	0.1595	T	1.0114e-05	-2.8565	-4.414700	3.2702	TRUE
5	rs10	C	0.3090	0.2975	G	4.2552e-01	-3.2335	0.796880	2.8931	FALSE
6	rs5	C	0.1525	0.1590	G	5.6594e-01	-3.0716	-0.574050	3.0550	FALSE
7	rs4	T	0.2040	0.1970	A	5.8530e-01	-3.0852	0.545650	3.0415	FALSE
8	rs101	T	0.0980	0.1010	A	8.2277e-01	-Inf	-0.223990	2.9329	FALSE
9	rs6	A	0.4960	0.4945	T	9.2460e-01	-3.1583	0.094641	2.9683	FALSE

Listing 18: Final result output of joint stage 1 and stage 2 analysis.

values (except those in columns denoted by 'crit') now refer to the pooled data sample of stage 1 and stage 2. Again, the list of markers is ordered by their p-values in column 'P', which, however, can now be different compared to the interim output, because the p-values may have changed with the stage 2 data, as is the case in our example. The critical limits have been taken over from the interim result output (see listing 17) and using the updated z-score (column 'z'), the test decision can now be derived directly by determining whether the z-score is located within (yes significant association) or outside (no significant association) the critical limits. The CRP-Tool provides this information in the last column named 'assoc', that is, whether (TRUE) or not (FALSE) the marker is significantly associated with the trait.

In our example, rs2 and rs3, the two most promising markers according to the interim result (listing 17), show a significant association after stage 2. The marker rs3b, for which the test direction was switched via the customization file, has the same p-value as rs3 but "unfortunately" was tested in the wrong direction so that the null hypothesis cannot be rejected for this marker. The marker rs100, although added after stage 1 and thus genotyped only at stage 2, has rised to the top of the list showing the smallest p-value of the entire set in stage 2 and a significant association as well.

References

Cheverud, J. M. (2001), A simple correction for multiple comparisons in interval mapping genome scans. *Heredity,* , **87**, 52–58.

Johnson, Steven G. (2010), The NLopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt.

Jones, D. R., Perttunen, C. D., Stuckmann, B. E. (1993), Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications*, **79**(1), 157–181.

Kaelo, P., Ali, M. M. (2006), Some variants of the controlled random search algorithm for global optimization. *Journal of Optimization Theory and Applications*, **130**(2), 253–264.

Müller, HH., Schäfer, H. (2004), A general statistical principle for changing a design any time during the course of a trial. *Statistics in Medicine*, **23**, 2497–2508.

Pahl, R., Schäfer, H., Müller, HH. (2009), Optimal multistage designs – a general framework for efficient genome-wide association studies. *Biostatistics*, **10**(2), 297–309.

Purcell, S., Neale, B., Todd-Brown, K., Thomas, L., Ferreira, M. A. R., Bender, D., Maller, J., Sklar, P., de Bakker, P. I. W., Daly, M. J., and Sham, P. C. (2007). Plink: a tool set for whole-genome association and population-based linkage analyses. *Am J Hum Genet*, **81**(3), 559–575.

Powell, M. J. D. (1994), A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in Optimization and Numerical Analysis: proceedings of the Sixth Workshop on Optimization and Numerical Analysis, Oaxaca, Mexico*, 51–67.

Sasieni, P. D., (1997), From genotypes to genes: doubling the sample size. *Biometrics* , **53**, 1253–1261.

Scherag, A., Hebebrand, J., Schäfer, H., Müller, HH. (2009), Flexible designs for genome-wide association studies. *Biometrics*, **65**(3), 815–821.

Skol, A. D., Scott, L. J., Abecasis, G. R., Boehnke, M. (2006) Joint analysis is more efficient than replication-based analysis for two-stage genome-wide association studies. *Nature Genetics* , **38**, 209–213.

Zheng, G. and Gastwirth, J. L. (2007), On estimation of the variance in Cochran-Armitage trend tests for genetic association using case-control studies. *Statistics in Medicine*, **25**, 3150–3159.