# **optimgui**: A GUI program to assist optimization problems in R

Yixuan Qiu*

August 22, 2011

## 1 Introduction

Optimization methods are intensively used in statistical computations such as maximum likelihood estimation and non-linear regression. R also has numerous packages and functions to solve optimization problems as those listed in the CRAN Task Views of Optimization[1]. However, the number and variety of these methods often confuse users which one to select, so it is quite necessary to provide a guidance of how to choose appropriate optimizers and set up the control parameters.

**optimgui** is an R package to assist the preparation and solution of optimization problems in R. This package has only one exported function, `optimgui()`, to accomplish most operations via a GUI (Graphical User Interface) environment, built with the **RGtk2[Lawrence and Temple Lang, 2010]**and **gWidgets[Verzani, 2011]**packages. The main features of **optimgui** are:

➤ User-friendly interface and operations;

➤ Built-in templates of optimization problems to get started with;

➤ More convenient approach to express constraints in constrained optimization problems;

➤ Automatically provides suggestions and choice of optimizers and parameters;

➤ Auto-generate R code of optimization problems.

*Department of Statistics, Renmin University of China. Email: yixuan.qiu@cos.name

[1] `http://cran.r-project.org/web/views/Optimization.html`

# 2 Rop file

**optimgui** uses the "Rop file" to describe an optimization problem. An Rop file is a combination of XML document and R code, wich stores the functions, data and related infomation of an optimization problem. Users are unlikely to edit the Rop file directly, since the creation and modification of Rop file could all be done by the **optimgui** program.

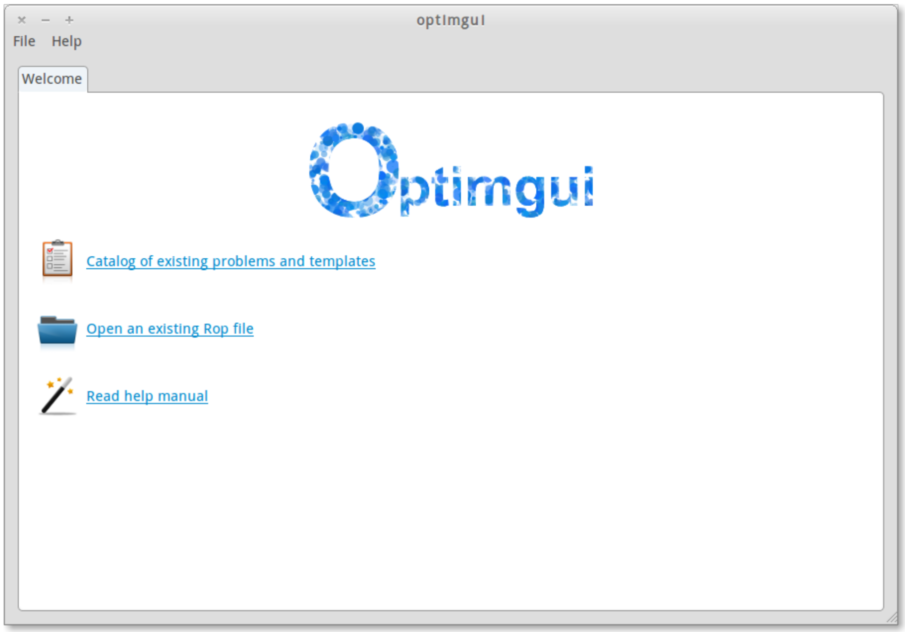
**optimgui** has several built-in Rop files under the `<package dir>/resources/Rop` directory, which serve as the templates to create a new one. Users should avoid modifying these built-in Rop files in case of a program crash. The usage of Rop file is discussed in the next section.

# 3 Usage

This section describes the usage of **optimgui** and the work you can do with it. To run the program, type the following two commands:

```
library(optimgui)
optimgui()
```

## 3.1 Welcome page and main menu



× – +
optimgui
File
Help
Welcome
Optimgui
Catalog of existing problems and templates
Open an existing Rop file
Read help manual

Figure 1: Welcome page

The first interface you'll see is a Welcome page with the following three links:

- **Catalog of existing problems and templates** Clicking this link will open the Catalog page, which shows the built-in templates of optimization problems. The Catalog displays some "characteristics" (we call them "Catalog entries" in the following sections) of each template, such as the number of parameters, property of gradient function, type of constraints, etc. Then you can select one that most fits your problem to get started. Click the "OK" button or double click the item to open the Rop editor.
- **Open an existing Rop file** Directly open an Rop file from your filesystem.
- **Read help manual** Open this document.

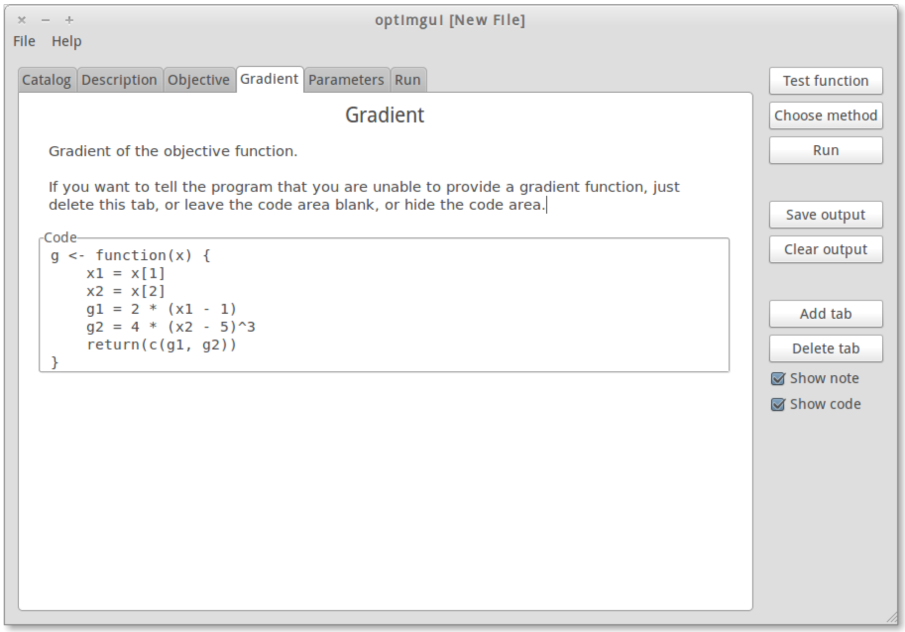
There is also a menu bar on the top of the window. The menu items and their functions are as follows:

- **The "File" menu**
  - **Open File** Open an existing Rop file from filesystem.
  - **Save/Save As** Save the current Rop file to filesystem.
  - **Save As Template** Save the current Rop file to the package directory, so that this file will appear in the Catalog page as a template.
  - **Set User Repository** Set the working directory of Rop file. When you have a new Rop file to be saved, this will be the default directory to store the file. Note that this will not change the working directory of R.
  - **Close File** Close the current file.
  - **Exit optimgui** Exit optimgui program.
- **The "Help" menu**
  - **optimgui Help** Open this document.
  - **About** Additional information of this package.

If this is the first time you use **optimgui**, you probably want to create a new Rop file from the template, and the next time you could open it directly from the filesystem, either by the Welcome page or by the menu.

## 3.2 The Rop editor

The Rop file will then be opened by the Rop editor, which consists of several tabs and a column of buttons. In the Rop editor, each tab represents a "section" of the Rop file, e.g., the objective function, the gradient function, etc. Typically each tab has three parts: the title, the note and the code area, all editable. However, both the note and the code area could be hided by unchecking the corresponding checkboxes in the button column.



Gradient

Gradient of the objective function.

If you want to tell the program that you are unable to provide a gradient function, just delete this tab, or leave the code area blank, or hide the code area.

```
g <- function(x) {
    x1 = x[1]
    x2 = x[2]
    g1 = 2 * (x1 - 1)
    g2 = 4 * (x2 - 5)^3
    return(c(g1, g2))
}
```

Figure 2: A tab in the Rop editor

You can add a new tab by clicking the "Add tab" button, or delete an existing one by clicking "Delete tab", but note that there are several indispensable tabs that could not be deleted. They are "Catalog", "Objective", "Parameters" and "Run".

- **The Catalog tab** This tab stores the Catalog entries of the optimization problem. The entries could be added, edited or deleted.
- **The Objective tab** The Objective tab stores the objective function, and the code area is not allowed to be hided.
- **The Parameters tab** This tab describes the initial value and constraints (optional) of the optimizer. Details are discussed in section 3.3.
- **The Run tab** This tab holds the generated R code and output report. Details are discussed in section 3.4.

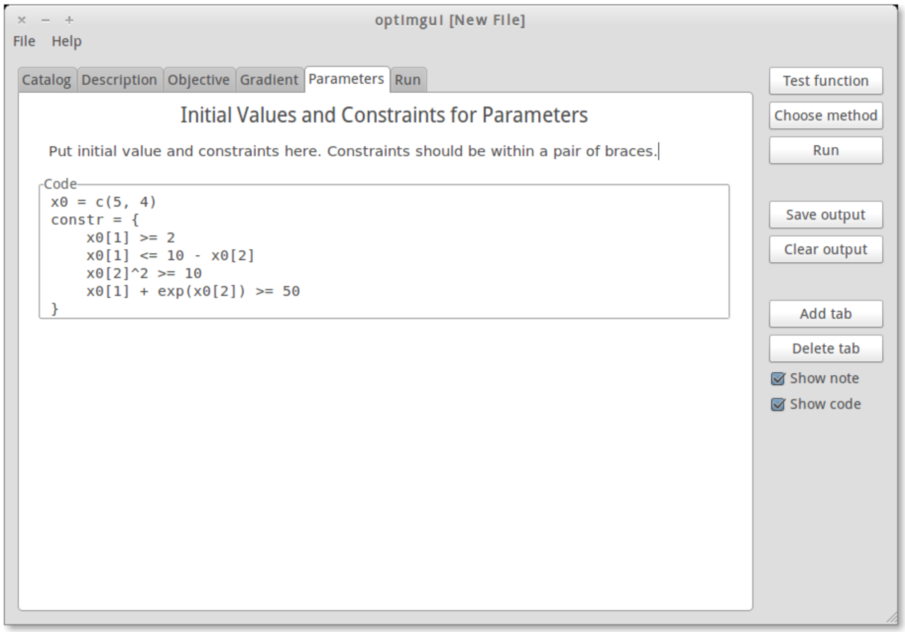
## 3.3 About constrained optimization

In many cases constraints of the following form should be added to the parameters in an optimization problem:

$$\begin{aligned} x_L \le x \le x_U & \quad \text{(box constraint)} \\ Ax \le b & \quad \text{(linear constraint)} \\ f(x) \le 0 & \quad \text{(non-linear constraint)} \end{aligned}$$

Here $x$ is often a vector.

In general, functions that allow constraints will ask users to specify the coefficients in the formulas above. However, this is often troublesome and not intuitive. In **optimgui**, users could express the constraints using a more convenient approach, that is, to input the constraints with respect to the elements of the parameter vector in an inequality form, as is shown in Figure 3.



Initial Values and Constraints for Parameters

Put initial value and constraints here. Constraints should be within a pair of braces.

Code

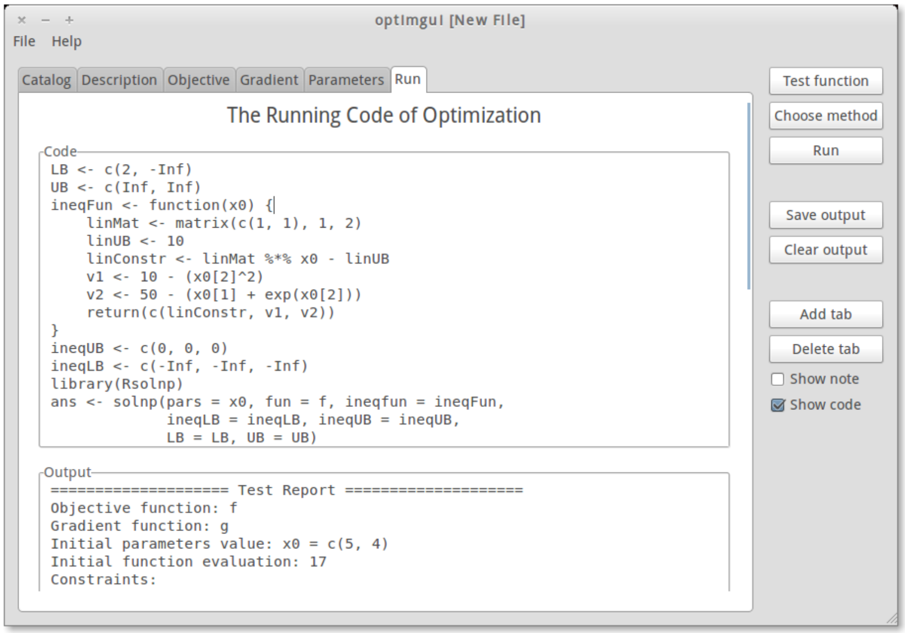
```
x0 = c(5, 4)
constr = {
    x0[1] >= 2
    x0[1] <= 10 - x0[2]
    x0[2]^2 >= 10
    x0[1] + exp(x0[2]) >= 50
}
```

Figure 3: Input constraints

Then in the next step (described in section 3.4), the program will analyze and classify these constraints and automatically determine the related coefficients needed by the optimizer.

## 3.4 Testing and running

When you have prepared all the needed functions, you can click the "Test function" button to obtain an overview of the optimization problem, including the name of objective function and gradient function, the initial parameters value, the type and coefficients of constraints, etc. After doing this, the program could then generate appropriate running code for the current problem when the "Choose method" button is clicked. Currently, there are three optimizers used as alternatives: `optimx()` in **optimx[Nash and Varadhan, 2011]**package for unconstrained and box constrained optimization, `constrOptim()` in **stats[R Development Core Team, 2011]** package for linear constrained problem, and `solnp()` in **Rsolnp[Ghalanos and Theussl, 2011]** package for non-linear constrained programming. Finally, users could hit the "Run" button to run the optimizer and obtain the result in the output box.



The Running Code of Optimization

Code

```
LB <- c(2, -Inf)
UB <- c(Inf, Inf)
ineqFun <- function(x0) {
    linMat <- matrix(c(1, 1), 1, 2)
    linUB <- 10
    linConstr <- linMat %*% x0 - linUB
    v1 <- 10 - (x0[2]^2)
    v2 <- 50 - (x0[1] + exp(x0[2]))
    return(c(linConstr, v1, v2))
}
ineqUB <- c(0, 0, 0)
ineqLB <- c(-Inf, -Inf, -Inf)
library(Rsolnp)
ans <- solnp(pars = x0, fun = f, ineqfun = ineqFun,
             ineqLB = ineqLB, ineqUB = ineqUB,
             LB = LB, UB = UB)
```

Output

```
==================== Test Report ====================
Objective function: f
Gradient function: g
Initial parameters value: x0 = c(5, 4)
Initial function evaluation: 17
Constraints:
```

Figure 4: Testing and running

# 4 TODO

This package is still under development and user may experience unexpected errors and crashes. The author also has a TODO list including the following:

- More templates of optimization problems;
- Support for more optimizers in R;
- Support for constraints in a vectorized form, e.g., user could input

  ```
  x >= 0
  ```

  instead of

```
x[1] >= 0
x[2] >= 0
...
x[length(x)] >= 0
```

- ...

Any comments and suggestions of this package are appreciated.

# References

Alexios Ghalanos and Stefan Theussl. *Rsolnp: General Non-linear Optimization Using Augmented Lagrange Multiplier Method*, 2011. R package version 1.11.

Michael Lawrence and Duncan Temple Lang. RGtk2: A graphical user interface toolkit for R. *Journal of Statistical Software*, 37(8):1–52, 2010. URL `http://www.jstatsoft.org/v37/i08/`.

John C Nash and Ravi Varadhan. *optimx: A replacement and extension of the optim() function*, 2011. URL `http://CRAN.R-project.org/package=optimx`. R package version 2011-2.29.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

John Verzani. *gWidgets: gWidgets API for building toolkit-independent, interactive GUIs*, 2011. URL `http://CRAN.R-project.org/package=gWidgets`. R package version 0.0-45.

Yinyu Ye. *Interior Algorithms for Linear, Quadratic, and Linearly Constrained Non-Linear Programming*. PhD thesis, Department of ESS, Stanford University, 1987.