Non - linear Programming Assignment – Optimization Techniques

Author: Aziza Zhanabatyrova

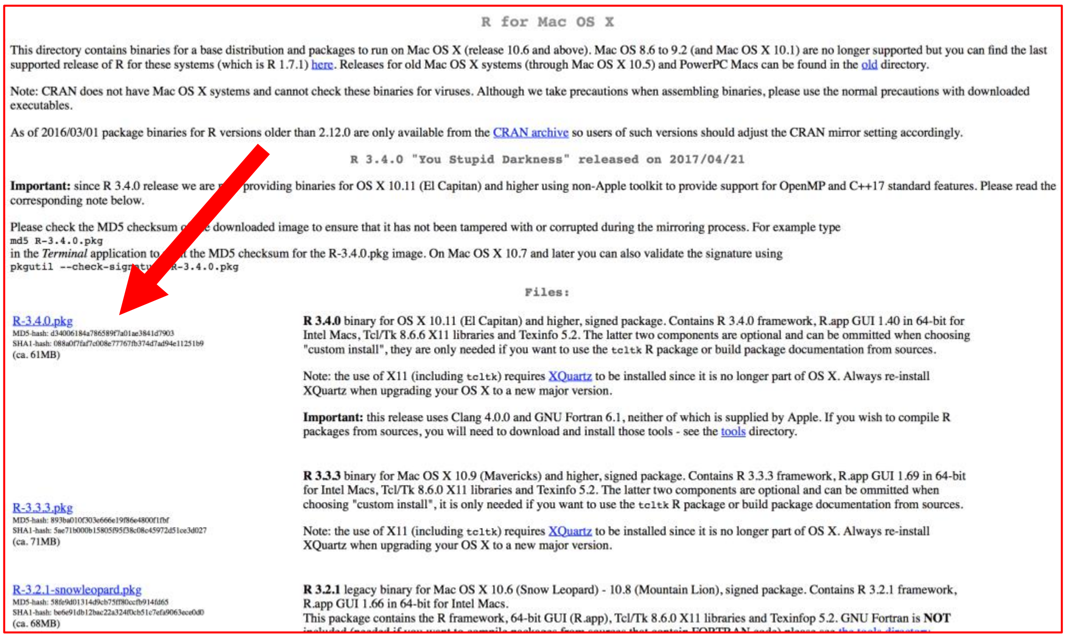
Important functions

- penalty_function: solves a constrained optimization problem with the penalty method. Every individual problem in the sequence of unconstrained problems is solved using the gradient method.

- gradient: solves an unconstrained optimization problem with the use of the gradient method.

HOW TO USE

1. Download R installation package (for mac) https://cran.r-project.org/bin/macosx/



R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) here. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the old directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

As of 2016/03/01 package binaries for R versions older than 2.12.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting accordingly.

R 3.4.0 "You Stupid Darkness" released on 2017/04/21

Important: since R 3.4.0 release we are now providing binaries for OS X 10.11 (El Capitan) and higher using non-Apple toolkit to provide support for OpenMP and C++17 standard features. Please read the corresponding note below.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type

```
md5 R-3.4.0.pkg
```

in the *Terminal* application to print the MD5 checksum for the R-3.4.0.pkg image. On Mac OS X 10.7 and later you can also validate the signature using

```
pkgutil --check-signature R-3.4.0.pkg
```

Files:

R-3.4.0.pkg
MD5-hash: d34006184a786589f7a01ae3841d7903
SHA1-hash: 088a0f7faf7c008e77767fb374d7ad94e11251b9
(ca. 61MB)

R 3.4.0 binary for OS X 10.11 (El Capitan) and higher, signed package. Contains R 3.4.0 framework, R.app GUI 1.40 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 5.2. The latter two components are optional and can be ommitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires XQuartz to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your OS X to a new major version.

Important: this release uses Clang 4.0.0 and GNU Fortran 6.1, neither of which is supplied by Apple. If you wish to compile R packages from sources, you will need to download and install those tools - see the tools directory.

R-3.3.3.pkg
MD5-hash: 893ba010f303e666e19f86e4800f1fbf
SHA1-hash: 5ae71b000b15805f95f38c08c45972d51ce3d027
(ca. 71MB)

R 3.3.3 binary for Mac OS X 10.9 (Mavericks) and higher, signed package. Contains R 3.3.3 framework, R.app GUI 1.69 in 64-bit for Intel Macs, Tcl/Tk 8.6.0 X11 libraries and Texinfo 5.2. The latter two components are optional and can be ommitted when choosing "custom install", it is only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires XQuartz to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your OS X to a new major version.

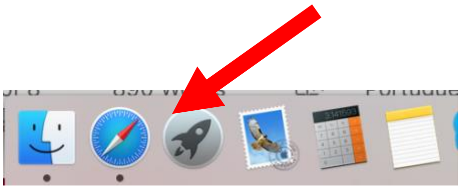
R-3.2.1-snowleopard.pkg
MD5-hash: 58fe9d01314d9cb75ff80ccfb914fd65
SHA1-hash: be6e91db12bac22a324f0cb51c7efa9063ece0d0
(ca. 68MB)

R 3.2.1 legacy binary for Mac OS X 10.6 (Snow Leopard) - 10.8 (Mountain Lion), signed package. Contains R 3.2.1 framework, R.app GUI 1.66 in 64-bit for Intel Macs.
This package contains the R framework, 64-bit GUI (R.app), Tcl/Tk 8.6.0 X11 libraries and Texinfop 5.2. GNU Fortran is **NOT** included (needed if you want to compile packages from sources that contain FORTRAN code) please see the tools directory.

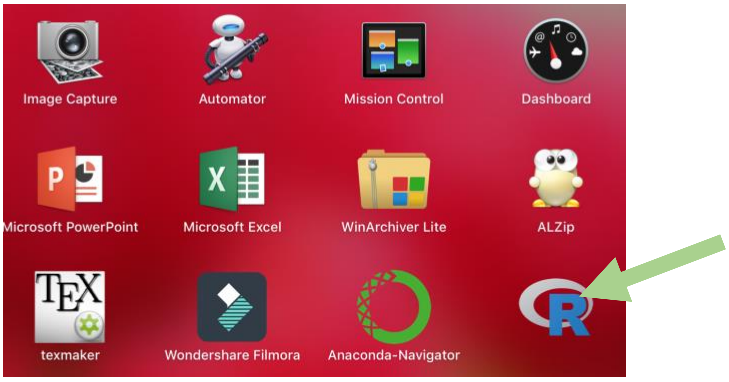
2. Install R, use default configurations

3. Open R console in the applications

3.1.



3.2.



4. Paste these commands:

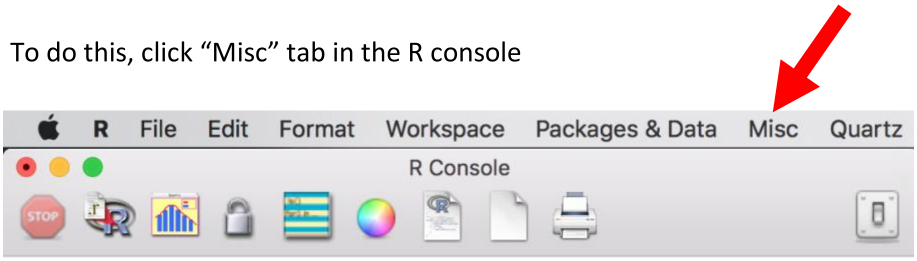
```
install.packages("shiny")

install.packages("numDeriv")
```

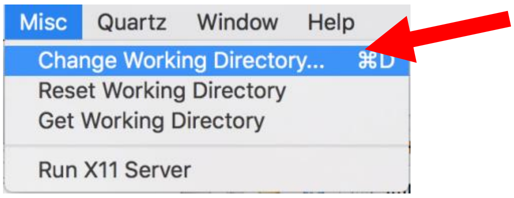
5. Download source folder “Assignment2” (ex. save on your desktop)

6. Change working directory to the directory in which the source folder “Assignment2” is located (“ex. desktop”), but not inside the source folder “Assignment2” itself.

   6.1. To do this, click “Misc” tab in the R console



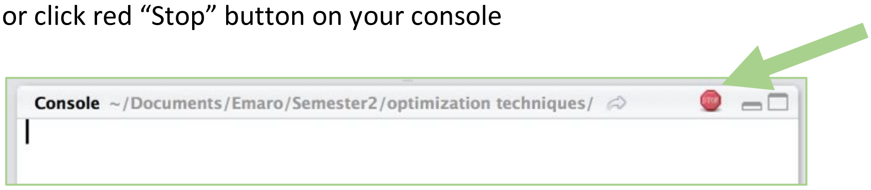
   6.2.



7. Launch the program by pasting into the console

```
runApp("Assignment2")
```

8. to stop the program, press “esc” button on your keyboard or click red “Stop” button on your console


Console ~/Documents/Emaro/Semester2/optimization techniques/
STOP

TESTS

$$\text{minimize} \;\; (x_2 - x_1^2)^2 + (x_1 - 1)^2$$

Initial solution: [5, 3]

Gradient tolerance: 1e-6

Constant step size: 0.01

```
The cost is:  1.44850369169905e-12
The solution =  1.00000111191351, 1.00000268442791
Found at iteration =  4155
Elapsed time in seconds =  0.221577882766724
```

$$\text{minimize} \;\; x_1^4 - 23x_1^2 - 18x_1 + 40$$

Initial solution: -5

Gradient tolerance: 1e-6

Constant step size: 0.01

```
The cost is:  -33.085379139164
The solution =  -3.17534758094279
Found at iteration =  16
Elapsed time in seconds =  0.0159831047058105
```

Initial solution: 1

Gradient tolerance: 1e-6

Constant step size: 0.01

```
The cost is:  -154.96045601391
The solution =  3.5720819545905
Found at iteration =  10
Elapsed time in seconds =  0.0160260200500488
```

$$\text{minimize } \ (x_2 - {x_1}^2)^2 + (x_1 - 1)^2 \ subject \ to \ {x_1}^2 \ \geq 2$$

Initial solution: [5,3]

Gradient tolerance: 1e-6

Penalty tolerance: 1e-6

Penalty max number of iterations: 15

Penalty initial Kj: 1

Penalty Kj increase rate: 2

Constant step size: 0.0001

```
The cost is:  0.171804744986702

The solution =  1.41414755474844, 2.0008234704695

Found at iteration =  5000

Iteration in penalty =  15

Elapsed time in seconds =  4.30507588386536
```

COMPARISON

In comparison with MATLAB, the accuracy depends on the parameters chosen for both penalty and gradient method. In general, the accuracy is similar. The time taken is faster than MATLAB for unconstrained cases, but constrained cases seem to last longer.