INTERFACING R AND ArcGIS WITH PYTHON

Germán Carrillo
gcarrillo@uni-muenster.de
Institut für Geoinformatik (ifgi)
University of Münster, Germany

Aim

The aim of this project is to interface R[1] and ArcGIS by providing an automated way of offering R scripts as ArcGIS Geoprocessing Tools. In that sense, the work of Mark Janikas[2] is taken as reference point for evaluating how the proposed approach makes it easier for R users to publish their algorithms allowing ArcGIS users to run them from a graphic environment.

Introduction

The interface between R and ArcGIS is based on Python[3], a powerful multi-platform Open Source programming language increasingly adopted by ESRI products. Taking advantage of Python add-ins, an ArcMap Toolbar is given for loading R scripts and generating Python toolboxes with a corresponding GUI that lets the user enter input and output parameters and run the algorithm. The R scripts have to be annotated to provide parameters with semantics and thus ease the selection of the most suitable ArcGIS graphic control, e.g. to show a check box for a Boolean parameter in the R script.

The Figure 1 shows the workflow required to expose an R script as a Python toolbox and how the toolbox communicate with the original R script in order to run the algorithm.

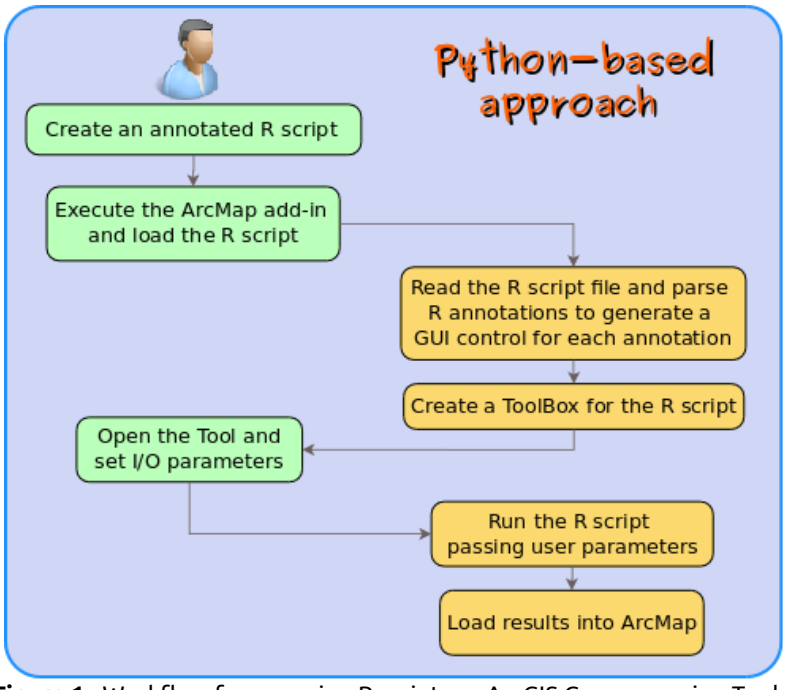

Figure 1. Workflow for exposing R scripts as ArcGIS Geoprocessing Tools.

From the workflow, it is clear that there will be at least two potential users of the proposed interface, namely, an R user, who writes R scripts and an ArcGIS user, who is not familiar with the R language but is get used to process spatial data through higher-level interfaces such as ArcGIS Geoprocessing tools.

1 http://www.r-project.org/
2 http://resources.arcgis.com/gallery/file/geoprocessing/details?entryID=F855D6D1-1422-2418-A0B2-643E624A8925
3 http://python.org/

Architecture

The interface between R and ArcGIS runs on Windows as a Python add-in for ArcMap, which offers a toolbar for loading an R script through a simple interface build on the top of the Python toolkit wxPython. Such a toolkit allows to create GUIs with Python and was used because it will be included in next releases of ArcGIS[4].

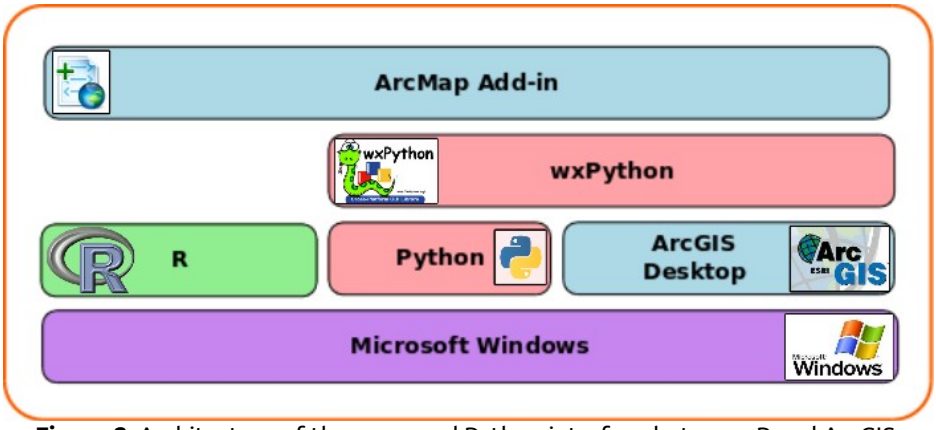

Figure 2. Architecture of the proposed Python interface between R and ArcGIS.

Annotations

Since R and ArcGIS data types differ and since it may be useful to provide R script parameters with semantics for interpreting and representing them well enough in ArcGIS, there must be a mechanism to describe R script parameters. Annotations are such a mechanism. The idea has been taken from Matthias Hinz and his work on WPS4R[5], they have been slightly adjusted and extended, though.

Basically, annotations are comments for describing R script parameters. An annotation is always a single line in the script and as a comment, it starts with the character '#'. As well as comments, annotations can be located in the same line of an R statement.

The most basic annotation is for describing what the purpose of the R script is. It starts with the word *des*, a colon and a string with no length restriction. For example:

```
# des: Creates a histogram from a sample with standard normal distribution
```

Besides the aforementioned annotation, there are two main kinds of annotations: for input and output parameters. Annotations for input parameters start with the word *in*, whereas output parameters are represented by the word *out*. Both of them require at least an identifier and a data type:

```
# in: sizeOfSample, integer
# out: histogramPath, png
```

However, it is strongly recommended providing a display text, which the ArcGIS user will read when setting the tool's parameters:

```
# in: sizeOfSample, integer, Size of the sample
# out: histogramPath, png, Histogram location
```

A very simple example using these annotations is presented in the Figure 3.

4 http://forums.arcgis.com/threads/41809-wxPython-hooked-to-arcmap?p=142208&viewfull=1#post142208
5 http://52north.org/communities/geoprocessing/wps/backends/52n-wps-r.html#R_scripts_and_WPS-annotations

```
# des: Creates a histogram from a sample with standard normal distribution
# in: sizeOfSample, integer, Size of the sample
# out: histogramPath, png, Histogram location
args = commandArgs( trailingOnly = TRUE )
a = rnorm( args[1] )
png( args[2] )
hist( a )
dev.off()
```

Figure 3. Example of annotated script.

The identifier of each annotated parameter can differ from the identifier of the corresponding R script parameter, e.g. in Figure 3 the annotated parameter *sizeOfSample* corresponds to the parameter *a* in the R script. The order of annotations matters when associating them to R script parameters.

Supported ArcGIS data types are shown in Table 1. For a better understanding of annotations, see the example provided bellow in this document.

Data type	Annotation key
Double	double
Long	integer
String	string
Boolean	boolean
dBaseTable	dbf
Raster Dataset	geotiff
Raster Dataset	geotiff-x
Raster Dataset	img
Raster Dataset	img-x
Raster Dataset	geotiff_image
Raster Dataset	gif
Raster Dataset	jpeg
Raster Dataset	png
Raster Dataset	tiff
Feature Layer	dgn
Shapefile	shp
Feature Layer	kml
Shapefile	shp_x
Text File	text
Field	field
Value list (string)	strings

Table 1. ArcGIS data types supported by annotations.

Configuration

In order to interface R and ArcGIS, the following system requirements have to be fulfilled:3

- ArcGIS Desktop v.10.1 with Python support.
- Python v.2.7 (32 bits).
- wxPython 2.8 (32 bits, for Python v.2.7).[6]
- R (tested with v.2.13.x).[7]

6 http://wxpython.org/download.php
7 http://cran.r-project.org/bin/windows/base/

The path to R.exe has to be included in the environment variable PATH, see the official R's FAQ (Frequently Asked Questions) web page[8] for details.

Additionally, download this ZIP file[9] which includes the Python add-in and a template file. Install the add-in by double-clicking on it and copy the template to the same folder where the R scripts are located.

The template file is a toolbox template that will be the basis for generating new Python toolboxes.

The Python add-in adds a new ToolBar to ArcMap. The ToolBar is called *R scripts* and contains a button called *R script to ToolBox* (see Figure 4).

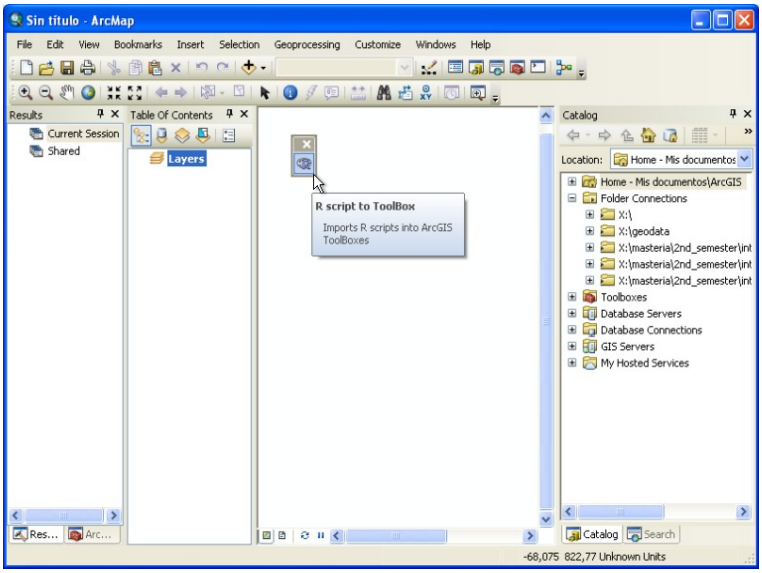

Figure 4. Parameters used in the R script PointCLusters.r

Example: Point Clusters

Since this project is intended to extend the Mark Janikas' work, a natural example would be to take one of his R scripts.

The R script

Download the ZIP file published by Mark Janikas from here[10]. Extract the folder *RTools*. Go to the sub-folder *Scripts* and copy the R script *PointClutsers.r* into a folder you create on another location for this example, let's say, a folder called *demo*.

Copy the template file (see configuration) into *demo*, as it has to be in the same folder as the R script.

Install R packages required by the R script

The *PointClusters.r* is based on external functionality provided by R packages. Unless the user has those packages installed, it would not be possible to run the script from R nor as an ArcGIS Geoprocessing Tool.

Therefore, the following packages must be installed:

- clustTool: Clustering with spatial information.
- maptools: Tools for reading and handling spatial objects.

For installing R packages see [11].

8 http://cran.r-project.org/bin/windows/base/rw-FAQ.html#How-do-I-set-environment-variables_003f
9 http://ifgibox.de/g_carr02/R_script_loader_add-in.zip
10 http://resources.arcgis.com/gallery/file/geoprocessing/details?entryID=F855D6D1-1422-2418-A0B2-643E624A8925
11 http://cran.r-project.org/doc/manuals/R-admin.html#Installing-packages

Annotating the R script

Open the *PointClusters.r* file and identify the parameters it requires (lines 7-12).

```
5   #### Get/Parse Arguments ####
6   Args = commandArgs()
7   inputFC = sub(".shp", "", Args[5], ignore.case = TRUE)
8   outputFC = sub(".shp", "", Args[6], ignore.case = TRUE)
9   numClusters = as.integer(Args[7])
10  clusterMethod = Args[8]
11  fields = Args[9]
12  useLocation = as.integer(Args[10])
13  useFields = FALSE
14
```

Figure 5. Parameters used in the R script PointCLusters.r

There are five (5) parameters required and they must be annotated. As stated in the section Annotations, each parameters needs at least an identifier and a data type. The display text is optional but recommended. In the following lines an explanation on how to annotate each parameter will be given.

Parameter **`inputFC`**:

The identifier can be either the same parameter name given in the R script or a new one, for instance, *shp1*. The type of this parameter is Shapefile, which is specified as *shp_x* (see Table 1). A self-descriptive display text could be ”Input Shapefile”. This parameter's annotation is:

```
# in: shp1, shp_x, Input Shapefile
```

Parameter **`outputFC`**:

Since this parameter is for specifying the output path, it needs to have the word *out* before the colon. The rest is similar to the previous parameter, so the annotation is:

```
# out: out, shp_x, Output Shapefile
```

Parameter **`numClusters`**:

This is an Integer parameter, its annotation is:

```
# in: numClus, integer, Number of Clusters
```

Parameter **`clusterMethod`**:

This parameter is a bit more complex. It represents the name of the cluster method to be applied, which has to be selected from a list of options (see the file *RTools/Scripts/PointClusters.py* in the folder recently extracted). Because the options are all strings (they are just names), a list of string values must be created. This data type has been accordingly called *strings*. The display text is *Cluster Method* and the list of string values is given as a period-separated list. Moreover, the default value *kmeansHartigan* is be specified. The annotation for this parameter is:

```
# in: method, strings, Cluster Method,
kmeansHartigan.clara.bclust.Mclust.kccaKmeans.cmeans, kmeansHartigan
```

Parameter **`fields`**:

This is another not so simple parameter. It is for selecting a field from the **`inputFC`**

parameter, i.e. from the input Shapefile, which has been given the identifier shp1 in its annotation. Because this parameter deals with fields, it has been called *field*. The display text is *Attribute Field(s)*. It has to be specified whether multiple selection is allowed, if so, put a *1*, otherwise a *0*. Because the cluster could be applied on multiple fields (i.e., clustering may be multivariate), in this example the multiple selection of fields is enabled, thus a *1* is written. Next, it is possible to set a list of data types to filter the fields retrieved, for instance, if only numeric fields have to be retrieved, the filter list is *short.long.double.float* (see ArcGIS field data types[12]). Finally, the Shapefile containing the fields has to be included, namely *shp1*, the identifier of the **`inputFC`** parameter.

```
# in: attributes, field, Attribute Field(s), 1, short.long.double.float, shp1
```

Parameter **`useLocation`**:

This is actually a Boolean parameter (see lines 33-36 in *PointCLusters.r*), its annotation, with a default value *true* is:

```
# in: location, boolean, Use Location, true
```

Once all annotations have been defined, they can be located anywhere in the script and since they are comments they do not alter the behavior of the R script. For this example, they will be located at the very beginning of the script, as a header:

```
1  # des: Creates spatial clusters (groups) of points based on their locations and their attributes. Uses R for
2  # in: shp1, shp_x, Input Shapefile
3  # out: out, shp_x, Output Shapefile
4  # in: numClus, integer, Number of Clusters
5  # in: method, strings, Cluster Method, kmeansHartigan.clara.bclust.Mclust.kccaKmeans.cmeans, kmeansHartigan
6  # in: attributes, field, Atribute Field(s), 1, short.long.double.float, shp1
7  # in: location, boolean, Use Location, true
8
```

Figure 6. Annotations added to the R script PointCLusters.r

Note that a description has been added for informing the ArcGIS user what the script does. The annotation for doing so is (everything in one line):

```
# des: Creates spatial clusters (groups) of points based on their locations and their
attributes. Uses R for the calculations. For additional information, see the following
documentation of the R tools: http://bm2.genes.nig.ac.jp/RGM2/pkg.php?p=clustTool
http://bm2.genes.nig.ac.jp/RGM2/R_current/library/clustTool/man/clustTool-package.html
http://cran.r-project.org/web/packages/clustTool/clustTool.pdf
```

Finally, save the annotated script as annotated_PointClusters.R in the folder *demo*.

Generating the Python toolbox from the R script

Start ArcMap and run the *R script to ToolBox* tool (see configuration). In the window *R script loader* select the R script *annotated_PointClusters.R* and click *Ok*.

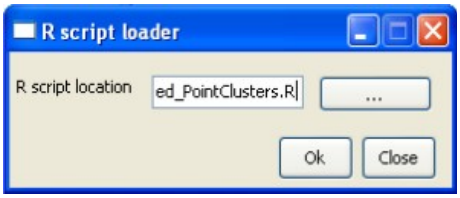

Figure 7. R script loader.

If everything went fine an information message is shown.

12 http://resourcesbeta.arcgis.com/en/help/main/10.1/index.html#//003n0000001m000000

Figure 8. Information message when a ToolBox is created.

A new Python toolbox is created with extension *pyt*, which can be easily shared to ArcGIS users and can be accessed from ArcCatalog from the folder *demo*.

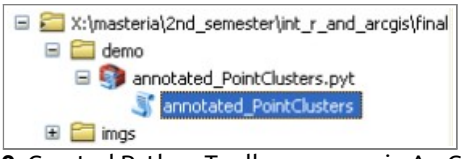

Figure 9. Created Python Toolbox as seen in ArcCatalog.

It can also be manually added to ArcToolbox.

Figure 10. Created Python Toolbox added to ArcToolbox.

An ArcGIS user would run the tool as any other ArcGIS Geoprocessing Tool, just by double-clicking on it.

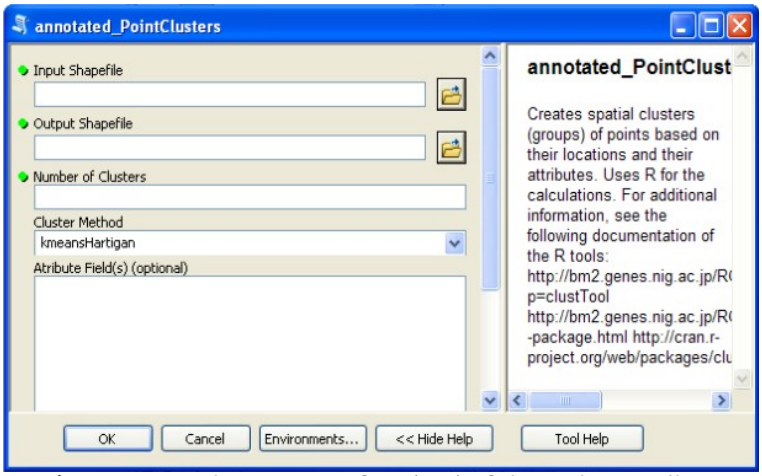

Figure 11. Graphic User Interface (GUI) of the Python Toolbox.

The GUI has been created from the R script annotations. Have a look at each control's type (e.g. ComboBox, MultiValue, CheckBox, among others) and its default values.

Compared to the Mark Janikas' work, this approach avoids two main steps for R users wanting to share their work:

- Defining a toolbox from scratch in ArcToolbox (where each parameter has to be set by hand)
- Creating a Python script (see PointClusters.py in the folder Rtools/Scripts/) with the logic of the created toolbox.

Therefore, R users will not have to learn yet another language (Python) as well as a proprietary package (ArcPy) for publishing ArcGIS Geoprocessing Tools.

Running the PointClusters tool

In the *annotated_PointClusters* tool load the Shapefile *Votepoints.shp* from the folder *RTools/ToolData/*, set an output path, set the number of clusters as 5, let the cluster method selected by default, check the fields *Conserv*, *Moderate* and *Liberal* and let the *Use location* option checked (see Figure 12). Click on *Ok* to run the algorithm.

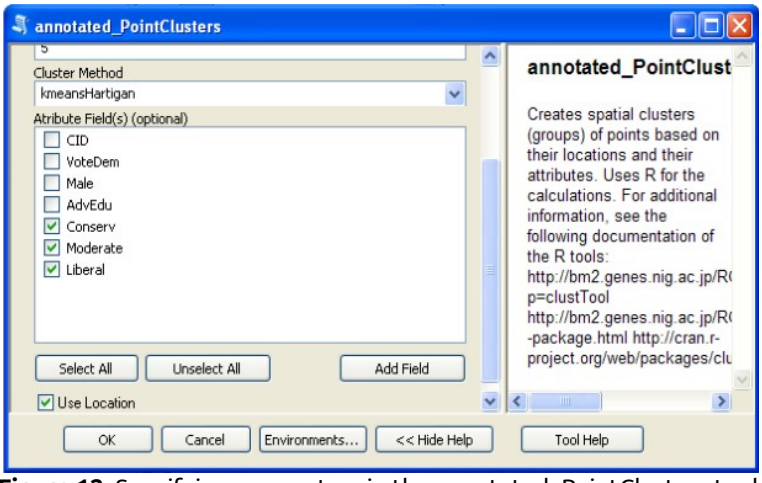

Figure 12. Specifying parameters in the annotated_PointClusters tool.

If ArcMap is running and the output will be automatically loaded in the map. Some contextual information about the process will be available afterward in the Results tab in ArcMap (see Figure 13).

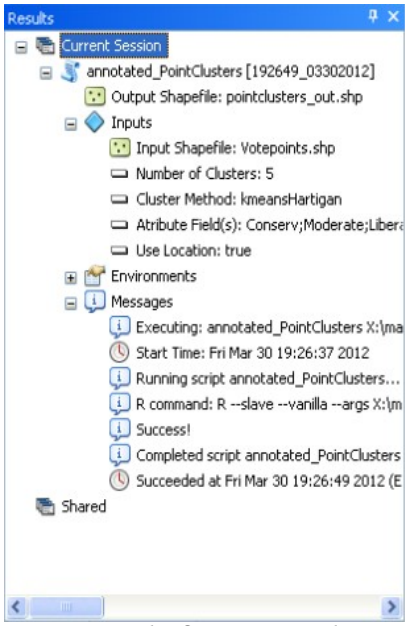

Figure 13. Contextual information in the Results tab.

From there it is possible to access the R command used to run the algorithm. To do so, right-click on the node *Messages* (Figure 13), select *Copy* and paste the content on a notepad. Because of space limitations in this document, the following figure shows each parameter of the R command in a new line, but it is actually a one-line command.

```
1  R --slave --vanilla --args
2  X:\path\RTools\ToolData\Votepoints.shp
3  X:\path\pointclusters_out.shp
4  5
5  kmeansHartigan
6  Conserv;Moderate;Liberal
7  1
8  <
9  X:\path\demo\annotated_PointClusters.R
10
```

Figure 14. R command executed by the annotated_PointClusters tool.

It is possible to define a pre-defined symbology layer to the output file. For instance, load the file *RTools/Scripts/RenderClusters.lyr*, define the output Shapefile as Data Source and edit a bit its properties (name and symbology) to obtain the following result:

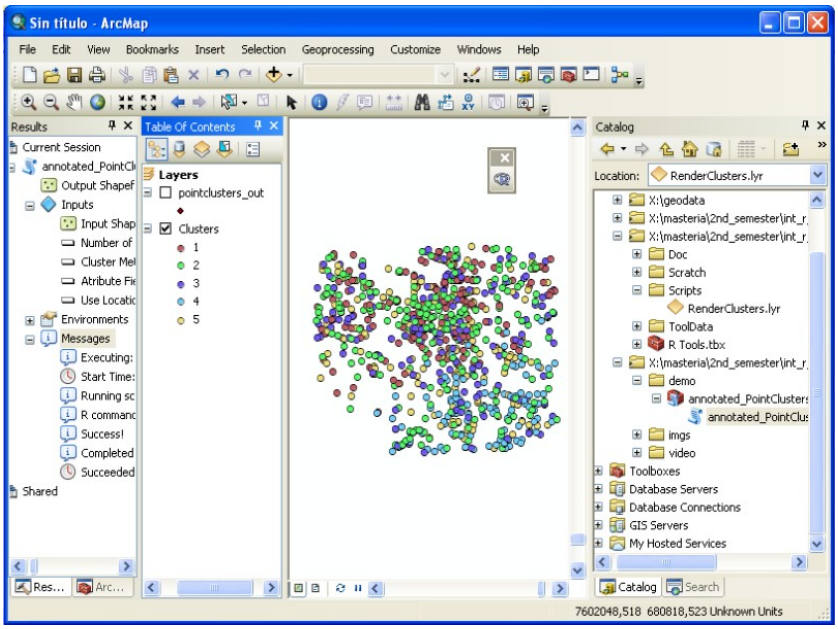

Figure 15. R command executed by the annotated_PointClusters tool.

This example can also be found as a screencast (without sound) in [13].

Source code and project files

The source code of this project can be downloaded from [14]. It is licensed under the terms of GNU/LGPL[15].

The add-in project, which consists of more files (not covered by the aforementioned license) can be downloaded from [16].

For downloading the executable add-in as well as the template file see configuration.

13 http://www.dailymotion.com/video/xpsqma_interfacing-r-and-arcgis-through-python_tech
14 http://ifgibox.de/g_carr02/source_code_R_script_loader_add-in.zip
15 http://www.gnu.org/licenses/lgpl.txt
16 http://ifgibox.de/g_carr02/add-in_R_ToolBox.zip