

# Package ‘RC’

January 2, 2012

**Type** Package

**Title** Reproducible Computing

**Version** 1.0.2.13

**Date** 2011-03-19

**Author** Patrick Wessa

**Maintainer** Send questions/complaints to <patrick@wessa.net>

**Depends** igraph, bitops

**Description** The RC package allows the user to create and use reproducible computations for the purpose of research and education. The meta data about the computations are stored in a remote repository which is hosted at [www.freeststatistics.org](http://www.freeststatistics.org)

**License** GPL-2

**LazyLoad** yes

**Repository** CRAN

**Date/Publication** 2011-03-20 19:49:22

## R topics documented:

RC-package . . . . .	2
RC.blog . . . . .	3
RC.browse . . . . .	5
RC.citation . . . . .	6
RC.credits . . . . .	6
RC.demo . . . . .	6
RC.disclaimer . . . . .	7
RC.end.plot . . . . .	7
RC.feedback.computation . . . . .	7
RC.feedback.submitted . . . . .	8

RC.hello . . . . .	9
RC.igraph . . . . .	10
RC.igraph.union . . . . .	10
RC.load . . . . .	11
RC.ls . . . . .	11
RC.ls.course . . . . .	12
RC.ls.module . . . . .	13
RC.ls.user . . . . .	14
RC.meta.data . . . . .	15
RC.mystep . . . . .	16
RC.network . . . . .	17
RC.network.list . . . . .	18
RC.parent . . . . .	19
RC.pk . . . . .	19
RC.plot.network . . . . .	20
RC.powered.by . . . . .	20
RC.prepare.input . . . . .	21
RC.print.html . . . . .	21
RC.print.tree . . . . .	22
RC.report.user . . . . .	23
RC.reproduce . . . . .	24
RC.reuse . . . . .	25
RC.save.image . . . . .	25
RC.send.feedback . . . . .	26
RC.start.plot . . . . .	26
RC.traffic . . . . .	27
RC.traffic.user . . . . .	28
RC.tree . . . . .	28
RC.utag . . . . .	29
RC.version . . . . .	29
RC.wrapper.function . . . . .	30
RCpicarr . . . . .	30
RCpicnr . . . . .	30
<b>Index</b>	<b>31</b>

---

 RC-package

*Reproducible Computing package*


---

## Description

The RC package allows the user to: - submit code snippets to the online repository at FreeStatistics.org - allow users to reproduce your computation through the internet (even if R is not installed on the client machine) - retrieve the meta information about an archived computation - reproduce a computation within the R console (on the local machine) - reuse the code, data, and parameters of a computation - query the computational repository for the purpose of educational quality control and research Have a look at the examples in the following functions: RC.ls, RC.tree, RC.blog,

RC.reproduce, RC.reuse, RC.traffic, RC.traffic.user, RC.report.user, ... The RC.demo() function demonstrates various functions in a step-wise manner.

### Details

Package: RC  
Type: Package  
Version: see RC.version()  
Date: 2011-03-20  
License: GPL v2 (only applies to client software)  
LazyLoad: yes

To start a tutorial use RC.demo(). More information can be found at <http://www.wessa.net/ReproducibleComputing/>

### Author(s)

Patrick Wessa

Maintainer: send questions/complaints to <patrick@wessa.net> or use the RC.send.feedback() function.

### References

Wessa P., A framework for statistical software development, maintenance, and publishing within an open-access business model, Computational Statistics, 2009, (DOI 10.1007/s00180-008-0107-y)

Wessa, P., Reproducible Computing: a new Technology for Statistics Education and Educational Research, IAENG Transactions on Engineering Technologies, American Institute of Physics, Eds: Ao, Sio-Iong , 2009

Wessa P., Exploring Social Networks in Reproducible Computing and Collaborative Assignments, Proceedings of the International Conference on E-Learning 2009, Toronto, Canada

---

RC.blog

*Blog an R function in the FreeStatistics.org Repository*

---

### Description

Use this function to blog any user-defined R function in the Repository (at [www.FreeStatistics.org](http://www.FreeStatistics.org)).

### Usage

```
RC.blog(title = "", keywords = "", comments = "", uid = "", pwd = "",  
typeofaccess = "public", moratoriumdate = "", rcode)
```

**Arguments**

title	Any user-specified title which briefly describes the computation to be blogged.
keywords	A comma separated list of keywords (or key phrases) which can be used in future queries. If you want to be able to retrieve your computation at a later stage in time, you should specify one (or several) unique keywords. For example, the keyword 'AS2009' could be used for computations that are associated with the Reproducible Computing workshop at the Applied Statistics 2009 conference.
comments	This is used to specify any ASCII text (of any length) which describes the computation.
uid	The User ID of the owner of the computation is required in any one of the following cases: 1. to make sure that the identity of the computation is 'verified' (this is proof that the computation was created by you - or anyone who knows your UserID and Password) 2. to hide the computation from third parties (if typeofaccess='private') 3. to hide the computation until a certain date in the future (if typeofaccess='moratorium')
pwd	The password of your account - only required in combination with uid.
typeofaccess	A string which identifies how the computation can be accessed by third parties: 1. 'public' (default value which specifies that the computation can be accessed by anyone) 2. 'private' (specifies that the computation can only be viewed by you - or anyone who knows your uid and pwd) 3. 'moratorium' (the computation is 'private' until the 'moratoriumdate')
moratoriumdate	Only required if typeofaccess='moratorium'.
rcode	The name of the function to be blogged.

**Details**

If you want to use the uid and pwd you must create an account first. To create an account you must do the following: 1. browse the <http://www.freeststatistics.org/index.php?action=8> web page 2. click the 'Create user' button 3. submit the account creation form (some fields are required) 4. within a period of 24 hours you will receive an e-mail which contains a hyperlink that you must use (in a web browser) to confirm your e-mail address 5. check that your account is working properly (login with your new uid and pwd on the <http://www.freeststatistics.org/index.php?action=8> page)

**Value**

Returns the url that uniquely identifies the statistical computation.

**Examples**

```
#define RCx data frame
RCx <- data.frame(array(rnorm(100),dim=c(50,2)))
RCxnames <- c("X1","X2")
#define wrapper function
RC.sample.fun <- function(title="my title") {
  RC.start.plot
  plot(RCx$X1,RCx$X2,main=title)
  RC.end.plot
}
```

```
    res <- cor.test(RCx$X1,RCx$X2)
    print(res) #print the result!
  }
  #blog it [not run]
  #r <- RC.blog(title="correlation test", keywords="blogtest",
  # comments="This example is used in the manual files of the
  # RC package.", uid="UseR", pwd="UseR", typeofaccess="public",
  # rcode=RC.sample.fun)
  #now we can use the returned URL to fetch the meta data [not run]
  #md <- RC.meta.data(r)
  #md
```

---

RC.browse

*View computation in a web browser*

---

### Description

Opens a web browser and fetches the web page that contains the snapshot of the statistical computation. The web page is opened through a secure callback mechanism which ensures that the content is shown within the [www.wessa.net](http://www.wessa.net) website. This effectively avoids any browser (such as IE 8/9) from (falsely) complaining about cross-site scripting when the statistical computation is reproduced.

### Usage

```
RC.browse(url)
```

### Arguments

url                    The URL of the statistical computation.

### Details

To use this function you need to specify a valid URL that identifies a statistical computation. The easiest way to obtain such an URL is through the `RC.ls()` function (see example).

### Examples

```
#search for all computations with the 'AS2009' keyword
#r <- RC.ls(keyword="AS2009")
#view the second computation from the list
#RC.browse(r$url[2])
#a browser window will be opened which displays the snapshot of the
#statistical computation
```

RC.citation

*Print citation info*

---

**Description**

Describes how the package can be cited.

**Usage**

```
RC.citation()
```

---

RC.credits

*Print credits*

---

**Description**

This function displays the credits about the RC package and the FreeStatistics.org project.

**Usage**

```
RC.credits()
```

**References**

<http://www.freeststatistics.org/>

---

RC.demo

*Run the RC demo*

---

**Description**

This function illustrates the key features about the RC package in a series of documented steps.

**Usage**

```
RC.demo()
```

**References**

<http://www.freeststatistics.org/>

---

RC.disclaimer	<i>Print disclaimer</i>
---------------	-------------------------

---

**Description**

This function displays the legal disclaimer.

**Usage**

```
RC.disclaimer()
```

**References**

<http://www.freeststatistics.org/>

---

RC.end.plot	<i>End a plot</i>
-------------	-------------------

---

**Description**

This is a tag that is used by the back engine to close the graphics device.

**Examples**

```
#see example of RC.blog()
```

---

RC.feedback.computation	<i>Fetch all feedback about a computation</i>
-------------------------	---

---

**Description**

This function fetches all feedback messages that have been submitted by any user about the computation.

**Usage**

```
RC.feedback.computation(pk = "", echo = TRUE)
```

**Arguments**

pk	primary key of the computation
echo	boolean parameter which indicates whether the number of records should be displayed

**Details**

This function submits a query to the repository at [www.freeststatistics.org](http://www.freeststatistics.org). The result is sent back (via HTTP) and returned in the form of a data frame. The feedback that is submitted may be of great importance within the context of peer review. The quality and quantity of peer review messages (generated by students within the context of statistics education) can be shown to be strongly related to exam results. Experimental research has shown that submitting peer review is highly beneficial and causes deep learning (publication forthcoming).

**Value**

The function returns a data frame with the following items:

pk	primary key of the feedback message
pk_frcomp	primary key of the computation
username	the name of the user who posted the feedback
userid	the User ID of the person who posted the feedback
text	the actual feedback message
date	the date of submission
pk_parent	primary key of 'parent' feedback message (this is zero if the message has no parent)
usernameinvisible	a boolean variable which indicates whether the user wishes to remain anonymous (if this variable is 1 then the username and userid will be 'Anonymous')

**Examples**

```
#Fetch the feedback messages that were submitted about the computation with pk = 22149 [not run]
#r <- RC.feedback.computation(pk=22149)
```

---

```
RC.feedback.submitted Fetch all feedback messages from user
```

---

**Description**

This function fetches all feedback messages that have been submitted by a user in the forums of FreeStatistics.org.

**Usage**

```
RC.feedback.submitted(id = "", echo = TRUE)
```

**Arguments**

id	user id of the person who submitted the feedback
echo	boolean parameter which indicates whether the number of records should be displayed

## Details

This function submits a query to the repository at [www.freeststatistics.org](http://www.freeststatistics.org). The result is sent back (via HTTP) and returned in the form of a data frame. The feedback that is submitted may be of great importance within the context of peer review. The quality and quantity of peer review messages (generated by students within the context of statistics education) can be shown to be strongly related to exam results. Experimental research has shown that submitting peer review is highly beneficial and causes deep learning (publication forthcoming).

## Value

The function returns a data frame with the following items:

pk	primary key of the feedback message
pk_frcomp	primary key of the computation
username	the name of the user who posted the feedback
userid	the User ID of the person who posted the feedback
text	the actual feedback message
date	the date of submission
pk_parent	primary key of 'parent' feedback message (this is zero if the message has no parent)
usernameinvisible	a boolean variable which indicates whether the user wishes to remain anonymous (if this variable is 1 then the username and userid will be 'Anonymous')

## Examples

```
#Fetch the feedback messages that were submitted by the student with User ID = 'b-r0262549' [Not run]
#r <- RC.feedback.submitted(id='b-r0262549')
```

---

RC.hello	<i>Say Hello to the RC network</i>
----------	------------------------------------

---

## Description

This function tests the network connectivity and performance of the R servers that are connected to the R Framework. This function is mainly useful whenever the user wishes to determine if errors or problems are related to the internet connection or the availability of R servers.

## Usage

```
RC.hello()
```

## Examples

```
#RC.hello()
```

---

RC.igraph                      *Convert Network into Igraph object*

---

### Description

This function converts a network object that is generated by the RC.network() function and converts it into an igraph object (from the igraph package) with additional attributes.

### Usage

```
RC.igraph(network = NULL)
```

### Arguments

network                      network object as generated by RC.network() or RC.network.list()

### Value

The function returns an igraph object (see igraph package for more information).

### See Also

[igraph](#)

### Examples

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

---

RC.igraph.union                      *Create the Union of two Sociograms*

---

### Description

This function joins two network objects and returns the corresponding union igraph object (from the igraph package).

### Usage

```
RC.igraph.union(r1, r2, dovert = FALSE)
```

### Arguments

r1                              the first network object as generated by RC.network() or RC.network.list()  
r2                              the second network object as generated by RC.network() or RC.network.list()  
dovert                          boolean

**Value**

The function returns an igraph object (see igraph package for more information).

**Examples**

```
##---- Should be DIRECTLY executable !! ----
##-- ==> Define data, use random,
##--or do help(data=index) for the standard data sets.
```

---

RC.load	<i>Load image</i>
---------	-------------------

---

**Description**

Loads an image from the repository. Unlike RC.save.image() this function is executed quickly.

**Usage**

```
RC.load(url = "", uid = "", pwd = "")
```

**Arguments**

url	The URL that uniquely identifies the image in the repository at FreeStatistics.org
uid	User ID of your account at FreeStatistics.org
pwd	Password

---

RC.ls	<i>List computations</i>
-------	--------------------------

---

**Description**

This function allows the user to search computations that are associated with a keyword or key phrase.

**Usage**

```
RC.ls(uid = "", pwd = "", keyword = "AS2009", quote = TRUE,
echo = TRUE, course = "", module = "")
```

**Arguments**

uid	User ID (only required to search for private computations)
pwd	Password (only required to search for private computations)
keyword	The keyword or key phrase to search. Use * as a wildcard.
quote	Put key phrase between quotes
echo	Echo intermediary results to console
course	Not yet implemented.
module	Not yet implemented

**Value**

The function returns a list with the following items:

url	url of the webpage that contains the meta data of the computation
key	the key of the computation (this is NOT the primary key but the URL key)
folder	folder of the computation
date	date of the computation (as registered on the server)
module	the name of the R module that generated the computation
title	user-specified title
keyword	user-specified keywords
course	id of the course or project (this is only available if the 'protag' label was used in the hyperlink that directs to the R module)
user	user id (owner of the computation)
parent	URL key of the parent computation (if any)
message	boolean variable which indicates whether the computation contains forum messages

**Examples**

```
#[not run]
#r <- RC.ls(keyword="AS2009")
#r
```

---

 RC.ls.course

*Find computations in course*


---

**Description**

This function retrieve a list of all computations that were generated within the context of a course. The course identification is automatically stored in the repository if the user clicks on a hyperlink with a tag called 'protag' (e.g. [http://.../rwasp\\_myrmodule.waps?utag=...&protag=mycourseid](http://.../rwasp_myrmodule.waps?utag=...&protag=mycourseid)). In Moodle, BlackBoard, and many other Virtual Learning Environments it is possible to specify the course id with any label in the hyperlinks. The use of the 'protag' label is not limited to courses - one may uniquely identify any project, journal, type of publication, etc...

**Usage**

```
RC.ls.course(course = "Applied Business Statistics 2008 (SHW)")
```

**Arguments**

course            course id

**Value**

The function returns a list with the following items:

url	url of the webpage that contains the meta data of the computation
key	the key of the computation (this is NOT the primary key but the URL key)
folder	folder of the computation
date	date of the computation (as registered on the server)
module	the name of the R module that generated the computation
title	user-specified title
keyword	user-specified keywords
course	id of the course or project (this is only available if the 'protag' label was used in the hyperlink that directs to the R module)
user	user id (owner of the computation)
parent	URL key of the parent computation (if any)
message	boolean variable which indicates whether the computation contains forum messages

**See Also**

[RC.ls](#)

---

RC.ls.module

*Find computations based on an R module*

---

**Description**

This function retrieves a list of all computations that were generated with a specific R module.

**Usage**

```
RC.ls.module(module = "Kernel Density Estimation")
```

**Arguments**

module            module id (this is equal to the 'exact' title of the R module)

**Value**

The function returns a list with the following items:

url	url of the webpage that contains the meta data of the computation
key	the key of the computation (this is NOT the primary key but the URL key)
folder	folder of the computation
date	date of the computation (as registered on the server)
module	the name of the R module that generated the computation
title	user-specified title
keyword	user-specified keywords
course	id of the course or project (this is only available if the 'protag' label was used in the hyperlink that directs to the R module)
user	user id (owner of the computation)
parent	URL key of the parent computation (if any)
message	boolean variable which indicates whether the computation contains forum messages

**See Also**

[RC.ls](#)

---

RC.ls.user

*Find computation from a user*

---

**Description**

This function returns information about all the computations that were blogged by a user.

**Usage**

```
RC.ls.user(id = "", echo = TRUE)
```

**Arguments**

id	user id
echo	should results be echoed to console?

**Value**

The function returns a list with the following items:

url	url of the webpage that contains the meta data of the computation
key	the key of the computation (this is NOT the primary key but the URL key)
folder	folder of the computation
date	date of the computation (as registered on the server)
module	the name of the R module that generated the computation
title	user-specified title
keyword	user-specified keywords
course	id of the course or project (this is only available if the 'protag' label was used in the hyperlink that directs to the R module)
user	user id (owner of the computation)
parent	URL key of the parent computation (if any)
message	boolean variable which indicates whether the computation contains forum messages

**See Also**

[RC.ls](#)

**Examples**

```
#fetch the list
#r <- RC.ls.user(id="UserR")
#print some useful info
#table(r$module)
```

---

RC.meta.data

*Fetch meta data of a statistical computation*

---

**Description**

Use this function to download the meta data that is associated with a stored computation. You need to know the exact URL of the computation or use the URL that is returned by the RC.ls() function (see example).

**Usage**

```
RC.meta.data(url = "")
```

**Arguments**

url                      url of the computation

**Value**

This function returns a list which contains several types of meta data about the computation. The actual elements in the list depend on several factors. In most cases the following elements will be available:

type	Indicates how the computation has been archived (e.g. 'R module', 'R console', ...)
date	Date when the computation was archived
uid	User id (owner of the computation)
title	User defined title
target	URL of the R module
rawinput	The R code (in raw form) which is sent to the R server
rawoutput	The reply from the R server (in raw form)
output	Output of the R module in HTML
ylimmax	maximum value on y-axis
ylimmin	minimum value of y-axis
chartxlab	text on x-axis of chart
chartylab	text on y-axis of chart
charheight	the height of each chart in pixels
chartwidth	the width of each chart in pixels
par<number>	value of parameter <number>
parent	id of the parent computation
data	dataset
newformula	R code as specified by the author or user before it is submitted to the R server

**Examples**

```
#search for computations with the 'AS2009' keyword [not run]
#r <- RC.ls(keyword="AS2009")
#fetch the meta data for the second computation in the list
#md <- RC.meta.data(r$url[2])
#do something useful with the data that was used in the computation
#plot(md$data,type="l",main="Data from the fetched computation")
```

---

RC.mystep

*Step counter*


---

**Description**

This is a step counter for the RC.demo() function. For internal use only.

---

RC.network	<i>Create network (sociomatrix) based on keyword</i>
------------	--

---

## Description

This function creates the sociomatrix of parent-child relationships about all computations that contain a specific keyword. The object that is returned can be used to display a sociogram (requires the 'igraph' package).

## Usage

```
RC.network(keyword = "workshop", elaborate = TRUE, computations = 1,  
quote = TRUE, startdate="1970-01-01",enddate="4000-12-31")
```

## Arguments

keyword	keyword to be used in search
elaborate	Boolean parameter which indicates if parents should have the same keyword as the children (elaborate=TRUE).
computations	Imposes a filter: only the relationships between pairs of users are displayed for which the number of reproduced computations of user A by user B is larger than (or equal to) <computations>.
quote	use a keyword search with quotes?
startdate	do not use computations that are older than startdate
enddate	do not use computations that are newer than enddate

## Value

Returns edges and verices that can be used in the igraph package to display and analyze social networks.

## See Also

[RC.network.list](#)

## Examples

```
#Note: the RC.plot.network function requires that the 'igraph' package is  
#installed on your computer  
#r <- RC.network(keyword="exercise")  
#g <- RC.plot.network(r,colors=c("red","blue"),weights=as.numeric(r$edges$V3))
```

---

RC.network.list      *Create network (sociomatrix) based on list of bloggers*

---

### Description

This function creates the sociomatrix of parent-child relationships about all computations that belong to a user-specified list of bloggers (users who archive statistical computations). The object that is returned can be used to display a sociogram (requires the 'igraph' package).

### Usage

```
RC.network.list(bloggers = NULL, elaborate = TRUE, computations = 1, quote = TRUE, startdate = "1970-01-
```

### Arguments

bloggers	character list of user identifications of the bloggers
elaborate	Boolean parameter which indicates if parents should have the same keyword as the children (elaborate=TRUE).
computations	Imposes a filter: only the relationships between pairs of users are displayed for which the number of reproduced computations of user A by user B is larger than (or equal to) <computations>.
quote	use a keyword search with quotes?
startdate	do not use computations that are older than startdate
enddate	do not use computations that are newer than enddate
vertex.attributes	apply attributes to vertices (for instance 'male' and 'female')

### Value

Returns edges and verices that can be used in the igraph package to display and analyze social networks.

### See Also

[RC.network](#)

### Examples

```
#[not run]
#RC.demo.users <- c('b-s0800252', 'b-s0800298', 'b-s0800944', 'b-s0801290', 'b-s0800032', 'b-s0510133', 'b-s08004
#RC.demo.r <- RC.network.list(RC.demo.users)
#RC.demo.g <- RC.plot.network(RC.demo.r)
#now you may analyze RC.demo.g with the igraph package
```

---

RC.parent	<i>Parent key</i>
-----------	-------------------

---

### Description

This is an object that keeps track of the parent key. For internal use only.

---

RC.pk	<i>Fetch primary key of a computation</i>
-------	---

---

### Description

This function fetches the primary key that corresponds to a record in the computational database. Some functions require the primary key as input.

### Usage

```
RC.pk(url = "", echo = TRUE)
```

### Arguments

url	the unique URL of the computation
echo	Print result

### See Also

[RC.traffic](#), [RC.traffic.user](#)

### Examples

```
#[not run]
#r <- RC.ls(keyword="AS2009")
#pk <- RC.pk(r$url[2])
```

RC.plot.network      *Plot the Sociogram of Impact*

---

### **Description**

This function uses the features of the 'igraph' package to plot the Sociogram that is based on the Impact between users. The Impact is computed based on the parent-child relationships that exist between computations.

### **Usage**

```
RC.plot.network(network = "", colors = "", weights = "")
```

### **Arguments**

network	network created by RC.network()
colors	list of color codes
weights	weights of edges

### **References**

Wessa P., Exploring Social Networks in Reproducible Computing and Collaborative Assignments, Proceedings of the International Conference on E-Learning 2009, Toronto, Canada

---

RC.powered.by      *Powered by?*

---

### **Description**

This function advertises Fedora linux as a free and robust OS. All servers of the FreeStatistics.org and Wessa.net network run Fedora, CentOS, or Ubuntu linux.

### **Usage**

```
RC.powered.by()
```

---

RC.prepare.input	<i>Prepare Raw Input</i>
------------------	--------------------------

---

**Description**

This function converts the Raw Input (as archived in the repository) into a code snippet that can be reproduced in the R console. This function is called by RC.reproduce().

**Usage**

```
RC.prepare.input(rawinput)
```

**Arguments**

rawinput	rawinput from RC.meta.data()
----------	------------------------------

**Value**

returns pre-processed R code which can be recomputed

**Examples**

```
#[not run]
#r <- RC.ls.user(id='User')
#fetching meta data about a computation that was generated in the console
#md <- RC.meta.data(r$url[21])
#prepare input
#p <- RC.prepare.input(md$rawinput)
#now it can be reproduced
#RC.texteval(p)
```

---

RC.print.html	<i>Print the HTML output of a computation</i>
---------------	---

---

**Description**

This function converts the 'a' object into a readable array for the R console. The 'a' object is created by the RC.reproduce() function whenever a computation is involved that was originally generated by an R module (published on the web).

**Usage**

```
RC.print.html(a)
```

**Arguments**

a This variable contains the raw output of the computation (only applies to computations that generate result tables in HTML format)

**Details**

This function converts the tables of the HTML output into R objects (arrays) and prints them in the console.

**Examples**

```
#[not run]
#r <- RC.ls(keyword="airline")
#RC.reproduce(r$url[40])
#convert the HTML output and print it
#RC.print.html(a)
```

---

RC.print.tree	<i>Print tree of parent-child relationships</i>
---------------	---

---

**Description**

This function is typically used after a tree (of parent-child relationships) has been generated with the RC.tree() function. RC.print.tree() outputs a hierarchical tree in text format. The 'current' computation is indicated with \*\*\*. The tree contains all children and parents of the 'current' computation. However, the brothers/sisters are not shown.

**Usage**

```
RC.print.tree(mytree = "")
```

**Arguments**

mytree tree produced by RC.tree()

**Examples**

```
#[not run]
#search computations with the 'babies' keyword
#r <- RC.ls(keyword="babies")
#fetch and print the tree about the first computation from the list
#mytree.1 <- RC.tree(r$url[1])
#RC.print.tree(mytree.1)
#fetch the tree based on the root parent
#mytree.2 <- RC.tree(mytree.1$url[1])
#print the entire tree (generates a lot of output)
#RC.print.tree(mytree.2)
```

---

RC.report.user	<i>Report about a user</i>
----------------	----------------------------

---

**Description**

Generate a report about the user activities. Various types of frequency statistics are reported.

**Usage**

```
RC.report.user(id = "UseR user ", recursive = FALSE)
```

**Arguments**

id	user id
recursive	should the search be done recursively (including children's children)?

**Value**

numcomp	Number of computations
module.table	Frequency of computations by R module
keywords.table	Frequency table of keywords
year.table	Frequency of computations by year
month.table	Frequency of computations by month of the year
day.table	Frequency of computations by day of the week
year.month.table	Frequency of computations by month and year
year.month.day.table	Frequency of computations by day, month, and year
hour.table	Frequency of computations by hour of the day
computations	list of all computations from the user (this list has the same items as the list that is produced by RC.ls())

**Examples**

```
#[not run]
#r <- RC.report.user(id="s0801036")
#r
```

---

RC.reproduce	<i>Reproduce a computation</i>
--------------	--------------------------------

---

### Description

This function fetches the meta data of a computation and reproduces the analysis on your local machine.

### Usage

```
RC.reproduce(url = "", echo = FALSE, secure = TRUE)
```

### Arguments

url	The unique URL of the computation to be reproduced
echo	Print intermediary results to screen
secure	Check R code against blacklist of potentially dangerous commands

### Value

The object 'a' contains the HTML output of any reproduced computation that was originally generated by an R module (as published on the web). To display the information in the R console it is appropriate to use the `RC.print.html(a)` command. The reproduced computation may create or change objects in the R session (see `RC.demo()` for an example).

### Warning

This function executes R code in your current R session. This implies that objects may be overwritten or changed as a result of using `RC.reproduce()`.

### Examples

```
#[not run]
#fetch info about the computations with the 'AS2009' keyword
#r <- RC.ls(keyword="AS2009")
#reproduce the second computation
#RC.reproduce(r$url[2])
#print the output table
#RC.print.html(a)
```

---

RC.reuse	<i>Reuse computation from the archive</i>
----------	---

---

### Description

This function allows you to recompute and reuse a computation from the repository. The data, parameters, and underlying R code can be changed.

### Usage

```
RC.reuse(url = "", secure = TRUE)
```

### Arguments

url	The unique URL of the computation to be reproduced
secure	Check R code against blacklist of potentially dangerous commands

### Examples

```
#[not run]
#RC.demo.r <- RC.ls(keyword='AS2009')
#RC.demo.res <- RC.reuse(RC.demo.r$url[2])
#now the RC.fun() function is available and can be executed
#op <- par(mfrow=c(4,2))
#RC.fun('1','0','0')
#RC.fun('1','1','0')
#RC.fun('1','0','1')
#RC.fun('0','1','1')
#par(op)
```

---

RC.save.image	<i>Save image</i>
---------------	-------------------

---

### Description

Saves an image in the repository. This function is extremely slow and not recommended unless the amount of data to be stored is relatively small.

### Usage

```
RC.save.image(title = "", keywords = "", comments = "", uid = "",
pwd = "", typeofaccess = "public", moratoriumdate = "")
```

**Arguments**

title	Title of your image file
keywords	Comma separated list of keywords
comments	Any comment (ASCII text)
uid	User ID of your account at FreeStatistics.org
pwd	Password
typeofaccess	A string which identifies how the computation can be accessed by third parties: 1. 'public' (default value which specifies that the computation can be accessed by anyone) 2. 'private' (specifies that the computation can only be viewed by you - or anyone who knows your uid and pwd) 3. 'moratorium' (the computation is 'private' until the 'moratoriumdate')
moratoriumdate	Only required if typeofaccess='moratorium'.

---

RC.send.feedback	<i>Send a question/complaint to the author</i>
------------------	--

---

**Description**

Use this function if you want to send a complaint or if you want to make a suggestion.

**Usage**

```
RC.send.feedback(email = "", subject = "", message = "")
```

**Arguments**

email	Your e-mail address (optional)
subject	A subject line
message	Your message in plain ASCII.

**Examples**

```
#RC.send.feedback("my.email@myisp.com", "Help!!!",
# "My computer crashed and displays a blue screen...")
#Note: you will probably get a reply to install Linux :-)
```

---

RC.start.plot	<i>Start a plot</i>
---------------	---------------------

---

**Description**

This is a tag that is used by the back engine to open the appropriate graphics device.

**Examples**

```
#see example of RC.blog()
```

---

`RC.traffic`*Fetch internet traffic about a computation*

---

**Description**

This function fetches the internet traffic of any computation in the repository. In order to identify the computation, a primary key (pk) must be provided. The pk can be obtained through `RC.pk()` which translates the URL into the corresponding primary key.

**Usage**

```
RC.traffic(pk = "", echo = TRUE)
```

**Arguments**

pk	the primary key of the computation
echo	print intermediary results?

**Value**

ip	IP addresses
count	# of visits

**See Also**

[RC.traffic.user](#)

**Examples**

```
#[not run]
#r <- RC.ls(keyword="AS2009")
#pk <- RC.pk(r$url[2])
#(traffic <- RC.traffic(pk=pk))
#result at the time of writing:
#Number of valid cases found: 5.
#           ip count
#1 84.192.102.135    9
#2 66.249.71.227    9
#3 72.30.81.186     9
#4 66.249.71.101    5
#5 84.194.80.219    5

#[not run]
#now an example with many IP addresses
#r <- RC.ls(keyword="exercise")
#tree <- RC.tree(r$url[50])
#pk <- RC.pk(tree$url[1])
#(traffic <- RC.traffic(pk=pk))
#output not shown
```

---

RC.traffic.user            *Fetch internet traffic from all computations of a specific user*

---

### Description

This function fetches statistics about the internet traffic of all computations that have been blogged by a user.

### Usage

```
RC.traffic.user(id = "", echo = TRUE)
```

### Arguments

id	user id
echo	echo to console?

### Value

A list of IP addresses. For each IP address the following sublists are available:

traffic	IP address, count, and primary key
statistics	sum and mean

### Examples

```
#[not run]
#traffic <- RC.traffic.user(id="Philippe Versluys")
#what is the 'impact' of the user?
#sum(traffic$statistics$sum)
#length(traffic$statistics$sum)
#maybe we should discount the IP addresses from search engines?
#traffic$stat[traffic$stat$mean>1,]
```

---

RC.tree                    *Fetch tree of parent & child relationships*

---

### Description

Creates a tree of parent-child relationships of all computations that correspond to a specific computation. The tree contains all parents and children of the 'current' computation. The sisters/brothers however are not displayed.

### Usage

```
RC.tree(url = "")
```

**Arguments**

url                    The unique URL of the 'current' computation

**Examples**

```
#search computations with the 'babies' keyword [not run]
#r <- RC.ls(keyword="babies")
#fetch and print the tree about the first computation from the list
#mytree.1 <- RC.tree(r$url[1])
#RC.print.tree(mytree.1)
#fetch the tree based on the root parent
#mytree.2 <- RC.tree(mytree.1$url[1])
#print the entire tree (generates a lot of output)
#RC.print.tree(mytree.2)
```

---

RC.utag	<i>User tag</i>
---------	-----------------

---

**Description**

This is an object that keeps track of the utag key. For internal use only.

---

RC.version	<i>Display version number of the RC package</i>
------------	---

---

**Description**

This function displays the version of the currently installed RC package.

**Usage**

```
RC.version(title = FALSE)
```

**Arguments**

title                    Print title?

---

RC.wrapper.function     *Fetch wrapper function*

---

**Description**

Creates the wrapper function RC.fun() which contains the R code that was stored in the repository. The function RC.wrapper.function() is called by RC.reuse().

**Usage**

```
RC.wrapper.function(metadata = "")
```

**Arguments**

metadata             the meta data object which is obtained from RC.meta.data()

**Value**

This function simply creates the function RC.fun() which contains the R code from the repository. If RC.fun() already exists, it is replaced without warning.

**See Also**

[RC.reuse](#)

---

RCpicarr             *Picture array*

---

**Description**

This is an object that keeps track of the picture array. For internal use only.

---

RCpicnr             *Picture number*

---

**Description**

This is an object that keeps track of the picture number. For internal use only.

# Index

## \*Topic **IO**

- RC.blog, 3
- RC.load, 11
- RC.meta.data, 15
- RC.reproduce, 24
- RC.reuse, 25
- RC.save.image, 25

## \*Topic **attribute**

- RC.meta.data, 15

## \*Topic **connection**

- RC.blog, 3
- RC.reproduce, 24
- RC.reuse, 25

## \*Topic **database**

- RC.blog, 3
- RC.meta.data, 15
- RC.reproduce, 24
- RC.reuse, 25

## \*Topic **debugging**

- RC.send.feedback, 26

## \*Topic **feedback**

- RC.feedback.computation, 7
- RC.feedback.submitted, 8

## \*Topic **file**

- RC.load, 11
- RC.save.image, 25

## \*Topic **forum**

- RC.feedback.computation, 7
- RC.feedback.submitted, 8

## \*Topic **graphs**

- RC.network, 17
- RC.network.list, 18
- RC.plot.network, 20
- RC.tree, 28

## \*Topic **manip**

- RC.prepare.input, 21

## \*Topic **misc**

- RC.browse, 5
- RC.demo, 6

- RC.end.plot, 7
- RC.igraph, 10
- RC.igraph.union, 10
- RC.mystep, 16
- RC.parent, 19
- RC.start.plot, 26
- RC.utag, 29
- RCpicarr, 30
- RCpicnr, 30

## \*Topic **package**

- RC-package, 2

## \*Topic **peer review**

- RC.feedback.computation, 7
- RC.feedback.submitted, 8

## \*Topic **print**

- RC.citation, 6
- RC.credits, 6
- RC.disclaimer, 7
- RC.powered.by, 20
- RC.print.html, 21
- RC.print.tree, 22
- RC.version, 29

## \*Topic **utilities**

- RC.hello, 9
- RC.ls, 11
- RC.ls.course, 12
- RC.ls.module, 13
- RC.ls.user, 14
- RC.pk, 19
- RC.prepare.input, 21
- RC.report.user, 23
- RC.traffic, 27
- RC.traffic.user, 28
- RC.wrapper.function, 30

igraph, 10

RC (RC-package), 2

RC-package, 2

RC.blog, 3

RC.browse, 5  
RC.citation, 6  
RC.credits, 6  
RC.demo, 6  
RC.disclaimer, 7  
RC.end.plot, 7  
RC.feedback.computation, 7  
RC.feedback.submitted, 8  
RC.hello, 9  
RC.igraph, 10  
RC.igraph.union, 10  
RC.load, 11  
RC.ls, 11, 13–15  
RC.ls.course, 12  
RC.ls.module, 13  
RC.ls.user, 14  
RC.meta.data, 15  
RC.mystep, 16  
RC.network, 17, 18  
RC.network.list, 17, 18  
RC.parent, 19  
RC.pk, 19  
RC.plot.network, 20  
RC.powered.by, 20  
RC.prepare.input, 21  
RC.print.html, 21  
RC.print.tree, 22  
RC.report.user, 23  
RC.reproduce, 24  
RC.reuse, 25, 30  
RC.save.image, 25  
RC.send.feedback, 26  
RC.start.plot, 26  
RC.traffic, 19, 27  
RC.traffic.user, 19, 27, 28  
RC.tree, 28  
RC.utag, 29  
RC.version, 29  
RC.wrapper.function, 30  
RCpicarr, 30  
RCpicnr, 30