

R-Forge User's Manual, BETA

SVN Revision: 47, August 12, 2008

R-Forge Administration and Development Team

Copyright 2006–2008 R-Forge Administration and Development Team



The content in this manual is licensed under a Creative Commons Attribution-Share Alike 2.0 license.

The R-Forge Administration and Development Team has chosen to apply the Creative Commons Attribution License (CCAL) to this manual, i.e., that under the CCAL, the authors retain ownership of the copyright for this manual, but the authors allow anyone to download, reuse, reprint, modify, distribute, and/or copy the contents of this manual, so long as the original authors and source are credited. Furthermore, the authors permit others to distribute derivative works only under the same license or one compatible with the one that governs the authors' work. This broad license was developed to facilitate open access to, and free use of, original works of all types. Applying this standard license to this work will ensure the authors' right to make this work freely and openly available (see <http://creativecommons.org/licenses/by-sa/2.0/> for details).

The current members of the R-Forge Administration and Development Team are Kurt Hornik, Martin Kober, David Meyer, Stefan Theußl and Achim Zeileis. To contact the authors please write an email to R-Forge@R-project.org.

Contents

1	Introduction	1
2	Registration	2
2.1	Registering a New User	2
2.2	Joining a Project	2
2.3	Registering a New Project	2
3	Source Code Management	2
3.1	How to get SVN to work	2
3.1.1	Windows	2
3.1.2	Linux/Unix	3
3.1.3	Mac OS	4
4	R Package Development	4
4.1	Building and Checking	4
4.2	External SVNs	5
4.3	Submit Packages to CRAN	5
5	Additional Features	5
5.1	Project Homepage	5
5.2	SCM Options	5
5.3	Mailing Lists	5
6	Migration from an existing CVS/SVN repository	5
7	Acknowledgements	6

1 Introduction

For a decade the R Development Core Team is using collaborative development tools like Subversion (SVN, see Pilato et al., 2004) or Concurrent Versions System (CVS, see Cederqvist et al., 2006) provided by ETH Zürich. Many R package developers around the world use similar infrastructure built upon their own solutions as collaborative development is a key factor for a lot of successful open source projects. Therefore the **R-project** wants to provide infrastructure not only for the R Development Core Team but also for the entire R community.

R-Forge (<http://R-Forge.R-project.org>) provides a set of tools for source code management and various web-based features. It aims to provide a platform for collaborative development of R packages, R-related software or further projects. It is closely related to the most famous of such platforms—the world’s largest open source software (OSS) development website—namely <http://SourceForge.net>. What is different is the provided set of tools with a strong focus on the need of the R community. All in all, R-Forge is a place where all R developers and users can come together and exchange their knowledge. Joining this community service gives you access to all we have to offer.

Basically, R-Forge is based on GForge (Copeland et al., 2006) which is a framework integrating various tools like SVN for collaborative work on source code, mailing lists, bug tracking etc. into one platform. On R-Forge all work is organized in the same entity namely a “Project”. In these projects it is possible to host one or more packages. The main advantage is, that your package is going to be built and checked every day not only with the latest patched version of R but also with the release candidate just before a new version of R is to be released. It is not necessary anymore, that developers have to set up a local platform-independent build and check environment. Furthermore, packages passing the quality management system on R-Forge can be released to CRAN in a standardized way.

Additionally, R-Forge provides other tools especially for larger projects to coordinate the work between project members and to communicate with their user base.

- Project websites are a way for developers to present their work on a subdomain of R-Forge (E.g., <http://foo.R-forge.R-project.org>). It is also possible to offer a link to another website on the project summary page instead.
- Mailing lists: By default one is automatically created when setting up a project. Additional mailing lists can be set up easily as well.
- Projects can be categorized into different topics (e.g., biostatistics, finance, regression analysis, ...). This enables other people to quickly find what they are looking for. People can browse the categories in the so-called “Project Tree” tab. In the default setting the project tree lists alphabetically all projects including a short description similar to CRAN. It is clear that this tree cannot be complete, so people are welcome to make suggestions on how we could improve it.
- Forums can be set up separately by the project administrators.
- News can be submitted to the project summary page as well as to the front page. The latter needs approval by one of the R-Forge Administration and Development Team members. It is also possible to download them as RSS feeds.

In this manual we present all relevant steps to get started with R-Forge. If you need further help or have comments regarding R-Forge please send an email to R-Forge@R-project.org. For a more detailed documentation regarding the underlying GForge system see Copeland et al. (2006).

2 Registration

2.1 Registering a New User

To register a new user, click on the “New Account” link on the top right side of the browser window at <http://R-Forge.R-project.org>. Fill out the form (there are descriptions and hints for each field on this site) and click on “Submit” afterwards. You will receive an email with a URL to your specified email address. After clicking on this link your account has been verified. Now you’re able to login to the website.

Important note: The tab “My Page” is the most important page on R-Forge. Here you configure your account, you see your project memberships and see the items, which have been assigned to you (i.e. bugs, feature requests, etc.).

Now you can start your own project (see Section 2.3 for details) or become a member of an existing project (Section 2.2).

2.2 Joining a Project

If you like to join an existing project you achieve this by doing the following steps:

1. First you need the name of the project you want to become a member of. You can “search” for the project (top middle side of the browser window) or you click on one of the projects showing in the “Project Tree”. The latter is an alphabetically sorted list of all **R-Forge** projects.
2. Then you go to the project summary page (should be the default entry point). There is a window on the right side called “Developer Info”. To join this project you need the permission of the project admin. So click on “Request to join” to send the project admin an email.
3. If the project admin decides to add you as developer you will receive an email confirming your developer account. Now you have full SVN access (see section 3 for details).

2.3 Registering a New Project

Registering a new project is easy: Go to the R-Forge website, login and go to “My Page” section. You have a “Register Project” link in the menu at the top of your page. Now fill in the form and submit your project. After approval of the project by the R-Forge admins you will be notified via email and you will be able to start with your project on R-Forge.

3 Source Code Management

3.1 How to get SVN to work

R-Forge uses **Subversion** (SVN, <http://subversion.tigris.org>) for source code management. Therefore you need an SVN client (e.g. “Tortoise SVN” on windows machines, <http://tortoisesvn.tigris.org>). For security reasons we use secure shell (SSH) tunneling for developer accounts which means that all network traffic is encrypted. Therefore you have to setup your machine accordingly.

3.1.1 Windows

The software mentioned in this section can be found on <http://www.chiark.greenend.org.uk/~sgtatham/putty/>.

1. get and install the latest TortoiseSVN client from <http://tortoisesvn.net/downloads>.
2. it is sufficient to use password authentication to write to the project repository (if you decided to do so please go to step 5). But for convenience purposes (avoiding password queries) it is probably better to generate an SSH keypair and upload the **public** key to **R-Forge**:
 - Download and execute `puttygen.exe`.
 - leave `ssh-2 rsa` marked and click on “generate” (make some random movements).
 - save the private key using the corresponding button (you don’t need to set a password).
 - Mark the text in the text field “Public key for pasting into OpenSSH authorized_keys file”.
 - copy and paste it into your shell account information configuration: Go to <http://R-forge.R-project.org> and login. Go to “My Page” and then click on “Account maintenance”. At the bottom of this page you click on “edit keys” in the “Shell Account Information” window (**Important note:** you must be a project admin or member of a project to do this, otherwise there won’t be an option “Shell Account Information”, see section 2.2 for joining a project or section 2.3 for registering a project). The key you enter here is typically of the form:

```
ssh-rsa AAAA... foo@bar
```

The first field describes the type of key, the second field is the key itself, and the third field is a comment. **It is important that there are no newlines within a key.**

3. Now you have to wait until the next full hour. The keys are activated once an hour only.
4. Next you need an authentication agent like `pageant.exe`. Load your **private** key with `pageant.exe` (right click on the pageant tray icon and then “add key”).
5. Finally check out the repository using the URL given on the project website (tab “SCM”) under “developer account” with “Tortoise SVN” or the SVN client of your choice. (Note: if this last step fails for some reason, wait an hour and try again. The logins are created only once an hour only)

3.1.2 Linux/Unix

1. It is sufficient to use password authentication to write to the project repository (if you decided to do so please go to step 4). But for convenience purposes (avoiding password queries) it is probably better to upload an SSH keypair: Generate and save the keys using `ssh-keygen` on the command line or use your existing keypair.
2. Upload the **public** key to **R-Forge** using the webplatform: Go to “My Page” and then click on “Account maintenance”. At the bottom of this page you click on “edit keys” in the “Shell Account Information” window (**Important note:** you need to be a project admin or member of a project to do this, otherwise there won’t be an option “Shell Account Information”, see section 2.2 for joining a project or section 2.3 for registering a project). The key you enter here is typically of the form:

```
ssh-dsa AAAA... foo@bar
```

The first field describes the type of key, the second field is the key itself, and the third field is a comment. **It is important that there are no newlines within a key.**

3. Now you have to wait until the next full hour (the key gets activated).
4. Finally check out the repository using the URL given on the project website (tab “SCM”) under “developer account” using `svn checkout`. (Note: if this last step fails for some reason, wait an hour and try again. The logins are created once an hour only)

3.1.3 Mac OS

Mac OS users should refer to Unix section 3.1.2. They can alternatively use GUI applications to manage their SVN repositories and SSH keychains.

1. To manage the ssh keypair you can use SSHKeychain (<http://www.sshkeychain.org/>). You can generate your keychains in the 'Preferences...' dialog box of SSHKeychain, but you have to provide a passphrase of 5 characters minimum. If you do not want a passphrase, use 'ssh-keygen' at the terminal (a keychain without passphrase is required to use SCPlugin, see step 4. In SSHKeychain, select 'Agent' - 'Add single key...' and select the key you just generated.
2. Upload the public key to **R-Forge** as explained in section 3.1.2 Unix. The public key is in a file name [keyfile].pub in the same directory as your private key. If you used default directory and name to generate your key, the files are in a hidden directory named '.ssh' in your home path, and is thus not visible in the Finder. To make a copy of the public key on your desktop type the following commands in the terminal (replace the name of the file with yours):


```
cp /.ssh/id_rsa.pub /Desktop/id_rsa.pub
```

 Then, upload this file to **R-Forge**.
3. Wait until the next full hour so that the key gets activated.
4. If you want a convenient management of your SVN repository in the Finder, install SCPlugin. To install it, follow instructions at <http://scplugin.tigris.org/installation.html>. Once SCPlugin is installed, navigate in the Finder to the directory where you want to place the local copy of your SVN repository. Create a folder with the name of your project, right click on this folder and select *Subversion -> Checkout*. Indicate the URL given on the project website (tab "SCM") under "developer account". (Note: if this last step fails, wait an hour and try again. The logins are created once an hour only).

Another page to consult for SVN installation on Mac OS is <http://www.wikihow.com/Install-Subversion-on-Mac-OS-X>. Alternate SVN software are proposed there.

4 R Package Development

R-Forge aims to provide a platform for collaborative development of R packages, R-related software or further projects. Especially providing good infrastructure for the development of R packages is a main mission of the R-Forge Administration and Development Team. Packages committed to the SVN repositories are built and checked on multiple platforms, including Linux, Windows (32 bit) and Mac OS.

4.1 Building and Checking

To build your packages, simply place your package or, alternatively, multiple packages in the `pkg/` directory in your SVM repository (see Section 3.1 on how to achieve this) and check it in. Typically this directory contains either the R package with the usual `DESCRIPTION` file and `R/`, `man/`, `data/` directories (see R Development Core Team (2006) for more details) or contains two or more directories containing the actual package sources, i.e., you can build **more than one package** by putting the packages in subdirectories, e.g. `pkg/foo/`, `pkg/bar/`, etc.

Packages are automatically checked out every night (Central European Time), built and checked on several platforms (make sure that the `DESCRIPTION` files are well-formed and contain a proper package name, otherwise your package may not get checked out). All packages successfully checked out appear on the project home page in the R packages tab. Project members also have access to the **check and build logs** on all supported platforms.

Furthermore, these packages are made available in a CRAN-style repository—the R-Forge package repository. This allows packages to be installed simply by typing `install.packages("foo", repos = "http://R-Forge.R-project.org")` on the R command prompt.

4.2 External SVNs

Developers who wish to use their own external package repository can use the **external SVN** feature in the R packages Admin tab (follow the instructions on the site). Please note that only repositories with anonymous access and using the R-Forge repository structure (i.e., `pkg` is available in the repositories) are supported. It is possible to use external and internal SVN repositories side-by-side. Regardless of origin, all packages will show up on the R packages tab.

If you decided to migrate your repository to R-Forge please go to Section 6.

4.3 Submit Packages to CRAN

Packages can be **submitted easily to CRAN** through the web interface. Note that you have to be project Administrator which indicates maintainership of the package to achieve this.

5 Additional Features

5.1 Project Homepage

Each project on R-Forge has its own homepage with the URL `http://foo.r-forge.r-project.org/` where `foo` is the project's unix name. Use the pre-defined `www` directory in your SVN repository to create or modify your homepage. Note that it will be checked out daily, so please take into consideration that it will not be available right after you commit your changes or updates. Please also note that only html or xhtml will be allowed in the future.

5.2 SCM Options

Two options can be accessed through the SCM Admin tab:

1. Enable anonymous access Self-explanatory. If disabled, only project members can access the SVN repository and **packages will not be checked out for automatic building**. The project homepage will still be available though. Default is enabled.
2. Delivery of Commit Messages This allows you to direct commit messages to an email address of your choice, e.g. the project's commit mailing list (see below). Default is disabled.

5.3 Mailing Lists

R-Forge provides a mailing list service for all projects accessible through the Lists tab.

A list named `foo-commits` where `foo` is the project's unix name is already created with each project and can be used for distributing commit messages (see Section 5.2 on how to enable this feature). Project administrators can create and manage additional lists for their project.

Furthermore, **searching the lists** is provided by the Swish-e engine and can be accessed via the List tab. Please note that private lists are not included in the search index.

6 Migration from an existing CVS/SVN repository

Steps to include the complete history of your current repository in your R-Forge project:

1. Please register a project on R-Forge first.

2. If you use CVS please convert from CVS to SVN (see e.g., <http://www.xs4all.nl/~carlo17/svn/cvs2svn.html> on how to do this).
3. Then dump the whole SVN repository (using `svnadmin dump > foo.dump`).
4. Verify that the dump file can be loaded into a freshly created SVN repository (`svnadmin load newrep < foo.dump`).
5. Send it to us: If the dump file is smaller than 10 MB you can send it via email to `R-Forge@R-project.org` otherwise you have to provide it somewhere for download.
6. The last step probably includes to `svn move` your package to the `/pkg` directory.

7 Acknowledgements

Setting up this project would not have been possible without Douglas Bates and the University of Wisconsin as they provided us with a server for hosting this platform. Furthermore, the authors would like to thank the Zentrum für Informatikdienste of the Wirtschaftsuniversität Wien for their support and for providing us with additional hardware as well as a professional server infrastructure. Thanks to Philippe Grosjean for providing a Mac OS description (Section 3.1.3).

References

- Per Cederqvist et al. *Version Management with CVS*. Network Theory Limited, Bristol, 2006. Full book available online at <http://ximbiot.com/cvs/manual/>.
- Tom Copeland, Roland Mas, Ken McCullagh, Tim Perdue, Guillaume Smet, and Reinhard Spisser. *GForge Manual*, 2006. URL http://gforge.org/docman/view.php/1/34/gforge_manual.pdf.
- C. Michael Pilato, Ben Collins-Sussman, and Brian W. Fitzpatrick. *Version Control with Subversion*. O'Reilly, 2004. Full book available online at <http://svnbook.red-bean.com/>.
- R Development Core Team. *Writing R Extensions*. R Foundation for Statistical Computing, Vienna, Austria, 2006. URL <http://www.R-project.org>. ISBN 3-900051-11-9.