

makeR: An R Package for Managing Document Building and Versioning

by Jason M. Bryer

Abstract The idea of build automation is not new. GNU Make and Java Ant are well established and robust build automation systems but require the use and installation of additional software. The **makeR** package provides a simplified framework written entirely in R to manage Sweave, L^AT_EX and R scripted projects where multiple versions are created from a single source repository. For example, monthly reports where each version is identical, with perhaps the exception of easily extracted properties (e.g. date ranges for data extraction, title, etc.).

R (R Development Core Team, 2011), L^AT_EX (Mittelbach and Rowley, 1999), and Sweave (Leisch, 2002) have proven to be incredibly useful for conducting reproducible research. However, managing document versions within R is limited. The **makeR** package attempts to provide the same ease-of-use for document versioning that the **devtools** (Wickham, 2011) and **ProjectTemplate** (White, 2011) packages have provided for package development and data analysis, respectively. This package attempts to solve the problem where multiple versions of a document are required but the underlying analysis and typesetting code remains static or can be abstracted through the use of variables or properties. For example, many researchers conduct monthly, quarterly, or annual reports where the only difference from version-to-version, from an analysis and typesetting perspective, is the data input. Clearly R and L^AT_EX are an ideal solution to this problem. The **makeR** package provides a framework to automate the process of generating new documents from a single source repository.

Project framework

There are three attributes to a particular document version: major (which can be numeric or character), minor, and build. Each major version is explicitly defined by the user. For example, if the goal of the project is to generate monthly reports then it would be appropriate to name each version using the month and year. Within each major version are minor versions. The minor version is always numeric and is incremented automatically upon each release. Lastly, build is numerical and global to the project that is automatically incremented with each document build. This index provides a unique identifier for each document that is created separate from the major and

minor version identifiers.

From the perspective of the file system, the **makeR** package maintains a project file, 'PROJECT.xml' (this will be discussed in further detail below), and three directories, 'source', 'build', and 'release'. The 'source' directory should contain all source files (typically a '.Rnw' file) along with any support files. The entire contents of this directory, including any subdirectories, will be copied for each build. **makeR** will create a new subdirectory in 'builds' for each unique major and minor combination. Lastly, the 'release' directory will contain "released" documents (e.g. final PDFs). The `releaseVersion` function will rename build files to include the major and minor versions.

Consider, for example, a new project that contains a file 'Example.Rnw' in the 'source' directory. Each call of the `build` function will copy all files from the 'source' directory to the 'builds/1.0' directory, then call `Stangle`, `Sweave`, and `texi2pdf` to create 'builds/1.0/Example.pdf'. Calling `release` will copy 'builds/1.0/Example.pdf' to 'release/Example-1.0.pdf' and then increment the minor version so that subsequent calls to `build` will create the 'builds/1.1' directory.

Properties

Properties are what differentiate one version from another and can be defined at the project or version level. When **makeR** builds a document project level properties will be assigned before version level properties thereby giving version level properties priority over project level properties. The functions `addProperty` and `removeProperty` for the objects of class type `Project` or `Version` (e.g. `myproject$addProperty`, `myproject$Versions[1]$addProperty`) provide an interface for manipulating properties.

Package framework

Figure 2 represents the internal structure of a **makeR** project. All interaction with a project will occur through the `Project` class object. By convention, all attributes of classes begin with a capital letter and are represented in the boxes with dashed lined borders. Functions (or methods) begin with a lowercase letter and are represented in shaded boxes. Class objects and lists are represented by rounded squares. `Project`, `Version`, and `Build` are special classes for **makeR** whereas `Versions` and `Builds` are simply R lists. All manipulation of a project should be done

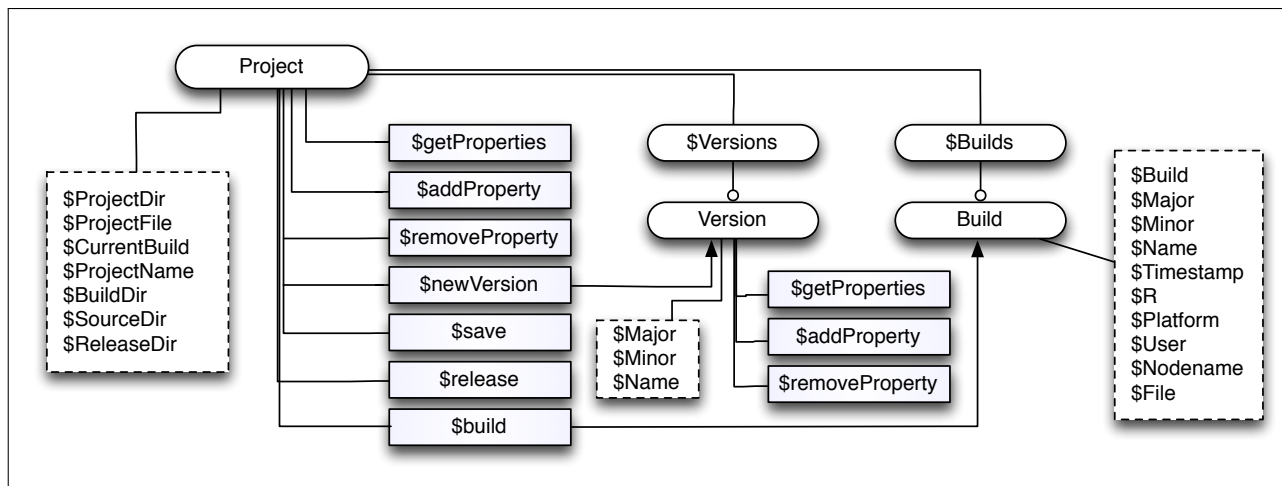


Figure 1: makeR structure

through the appropriate methods. Although the attributes (i.e. elements of a list or class) are accessible given R's programming framework, doing so could result in unexpected results. The following two sections list the available attributes and methods.

Project attributes

- **BuildDir** - the directory where builds will occur.
- **Builds** - list of completed builds. Each element in the list has a class type of **Build**.
- **CurrentBuild** - an integer of the last build.
- **ProjectDir** - the base directory where the project is located.
- **ProjectFile** - the name of the source file to be built.
- **ProjectName** - the name of the project.
- **ReleaseDir** - the directory where released files will be located.
- **SourceDir** - the directory containing the source files.
- **Versions** - a list of the project versions. Each element in the list has a class type of **Version**.

Project methods

- **build** Builds the project.
 - **version** - (optional) the version to build.
 - **saveEnv** - if TRUE, the build environment (.rda) will be saved in the build directory.
 - **builder** - the builder function.

- **rebuild** Rebuilds the project without first copying the files.
 - **version** - (optional) the version to rebuild.
 - **saveEnv** - if TRUE, the build environment (.rda) will be saved in the build directory.
 - **builder** - the builder function.
- **save** Saves the PROJECT.xml file.
- **newVersion** Creates a new versions of the project.
 - **name** - (optional) the version name.
 - **properties** - version specific properties.
- **release** Releases a version (i.e. copies the built file to the releases directory)
 - **version** - (optional) the version to release. If omitted the latest version will be released.
- **getProperties** Returns the project properties.
- **addProperty** Adds a project property.
 - **name** - The property name.
 - **value** - The property value.
- **removeProperty** Removes the given project property.
 - **name** - The property name.
- **getReleases** Returns a list of released files.
- **openRelease** Opens the given released file with the system's default application.
 - **file** - The released file to open.

The project file format

The ‘PROJECT.xml’ file provides an XML formatted file corresponding to a project allowing for a project to persist across R sessions. The file is created and edited using the **XML** (Lang, 2011) package. Although the **makeR** package provides functions to manage projects, using XML allows for other R programmers to interact with the **makeR** project framework. Figure 2 represents the contents of ‘PROJECT.xml’ after running `demo('stocks')`.

Example

The **R-Bloggers** site provides a wealth of information about R and the R community, aggregated from many different R bloggers. Like many continuously changing data sources, we wish to periodically report on recent activity on the R-Bloggers site. The base source file, ‘rbloggers.Rnw’, is included in the package (in the ‘inst/doc/robloggers’ directory) as well as being hosted on [Github](#). This example can also be run using `demo('rbloggers')`.

Create new project

```
myProject = Project(name="RBloggers",
  projectDir=~"/rbloggers",
  properties=list(email=email, passwd=passwd))
```

Create initial version

```
myProject$newVersion(name='2011-12',
  properties=list(startDate='2011-12-01',
    endDate='2011-12-31'))
```

Building the document

```
myProject$build()
```

Releasing the document

```
myProject$release()
```

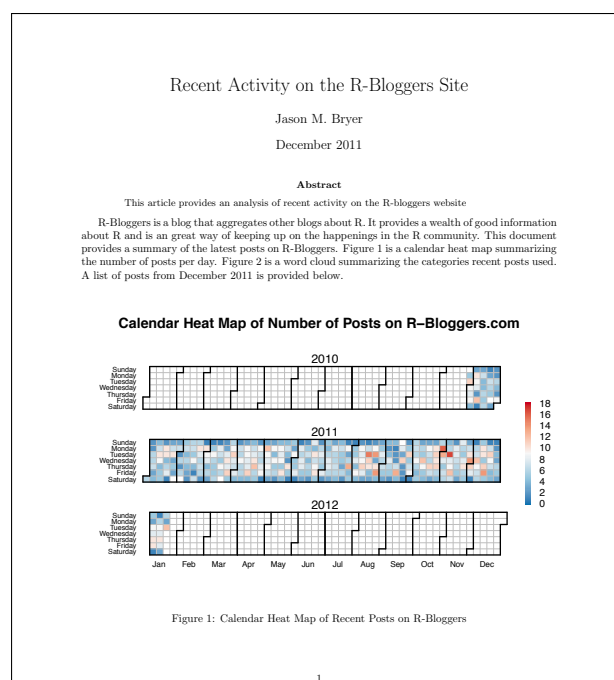


Figure 3: Output from the R-Bloggers example.

Custom builders

The **makeR** package includes builders for managing Sweave (.rnw) and \LaTeX (.tex) documents. A builder using the **cacheSweave** driver for Sweave is also included. However, the **makeR** package is designed to be extensible. By specifying the `builder` parameter on the `build` method, the user can define their own custom builder. The function requires two parameters, `project` and `theenv`. The former is simply the `Project` that is currently being built. The latter is an R environment with all the appropriate properties set. Any other parameters are passed from the `build` function to the builder. The following function builds projects where the input is an arbitrary R script file and the output are PNG image files.

```
builder.png <- function(project, theenv, ...) {
  sourceFile = ifelse(is.null(project$SourceFile),
    '.r$', project$SourceFile)
  wd = eval(getwd(), envir=theenv)
  files = list.files(path=wd, pattern=sourceFile,
    ignore.case=TRUE)
  for(i in seq_len(length(files))) {
    sys.source(files[i], envir=theenv)
  }
  return(list.files(path=wd, pattern=".png$",
    ignore.case=TRUE))
}
```

Package development

The latest stable version of **makeR** can be installed from your local CRAN server. Development versions

```

<?xml version="1.0"?>
<project name="stocks" buildDir="build" releaseDir="release" sourceDir="source">
  <property name="src" type="character">
    <value>yahoo</value>
  </property>
  <property name="stocks" type="character">
    <value>GOOG</value>
    <value>AAPL</value>
    <value>AMZN</value>
    <value>MSFT</value>
  </property>
  <versions>
    <version name="2011-12" major="1" minor="1">
      <property name="month" type="character">
        <value>2011-12</value>
      </property>
    </version>
  </versions>
  <builds>
    <build major="1" minor="0" build="1" name="2011-12" timestamp="Thu Jan 26 20:55:10 2012"
      R="R version 2.14.0 (2011-10-31)" platform="x86_64-apple-darwin9.8.0"
      nodename="Jason-Bryers-MacBook-Air-2.local" user="jbryer">
      <file>2011-12.png</file>
    </build>
  </builds>
</project>

```

Figure 2: ‘PROJECT.xml’ for the stocks demo.

are hosted on [Github](#). The latest development version can be installed using the **devtools** (Wickham, 2011) package:

```
install_github('makeR', 'jbryer')
```

Bibliography

- D. T. Lang. *XML: Tools for parsing and generating XML within R and S-Plus.*, 2011. URL <http://CRAN.R-project.org/package=XML>. R package version 3.4-3.
- F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *Compstat 2002 — Proceedings in Computational Statistics*, pages 575–580. Physika Verlag, Heidelberg, Germany, 2002. ISBN 3-7908-1517-9.
- F. Mittelbach and C. Rowley. *The L^AT_EX3 project*, 1999. URL <http://www.latex-project.org/guides/ltx3info.pdf>.
- 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.
- J. M. White. *ProjectTemplate: Automates the creation of new statistical analysis projects.*, 2011. URL <http://CRAN.R-project.org/package=ProjectTemplate>. R package version 0.3-5.
- H. Wickham. *devtools: Tools to make developing R code easier*, 2011. URL <http://CRAN.R-project.org/package=devtools>. R package version 0.4.

Jason M. Bryer
 Excelsior College
 7 Columbia Circle
 Albany, NY 12203
 USA
jason@bryer.org