R & GIS: Geospatial Plotting

Florian Endel * **Peter Filzmoser** **

* *FFG IFEDH project, Student at Vienna University of Technology (e-mail: florian@endel.at).*
** *Department of Statistics and Probability Theory, Vienna University of Technology (e-mail: P.Filzmoser@tuwien.ac.at)*

Abstract: Examples of spatial data within the R environment and the combination of R with data sets, spatial tools, libraries and other software products, which are common in real life environments, are provided in this paper.
Beginning with the setup of a new project using Git and an online repository, a Document build by Sweave - a combination of R and LaTeX - is explained. Once the fundamental setup is working rudimentarily, the import of data and (geo-) spatial information from different sources is shown. Finnaly some examples of spatial plots using the *sp* package in R are included.
Additionally some tools like "integrated development environments" (IDE), which may be supportive during the daily work and help newcomers learning the presented techniques, are mentioned and specific programs are recommended.
This paper itself is build using the described technologies and therefore illustrates all applied methods.

Keywords: R, Sweave, LaTeX, Spatial Data, Git

1. INTRODUCTION

Building a sophisticated report including spatial data, plots and usefule information (mostly) takes some effort.

Vast online resources and big data sets may overstrain even modern desktop computers, especially if common and established methods and routines need to scale up for new information. Working in groups or spatial distributed teams, changing requirements and specification, copyright issues, reproduction of foreign (or even own) results and updating reports using new data sets are common problems during the lifetime of a project. Additionally the amount of time, manpower and funding may be sometimes quite short.

The adaption of technologies presented in this paper may help overcome some of these challenges. Although there are great resources beginning with basic tutorials up to sophisticated books for every delineated software product, useful example of the whole workflow are still quite rare. Therefore the practical utilization of one single technique may be well documented, but the implementation and combination of several software tools during ongoing projects may still be challanging.

Additionally the awareness of great open source [1] software products, which are definitely ready to be used on a regular basis, especially the power of "The R Project for Statistical Computing", should be extended.

This paper itself is build using the described technologies and therefore illustrates all applied methods. To render the understanding and reproduction of the document considerable simple and provide an example of the methods used, all sourcecode is available online.

2. STRUCTURE

The structure of this report is tailored towards a possible workflow of a (simple) research project.

Beginning with the setup of a (online) repository to track changes and manage contributions of different team members, the structure of a Sweave document is explained. Using different data sets, partly provided within R packages, partly imported from other sources, the creation of geospatial plots is illustrated.

It should be obvious, that only a basic introduction and different examples can be provided in this document. Further details about this report can be looked up in the online repository. Additonally some useful online ressources and other references are included.

3. SETUP OF THE PROJECT

Before exploring the data and writing the report can be started, git has to be setup. On the official homepage of the git project, the following short definition is provided [2]

> Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
>
> Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access

⋆ Forschungsförderungsgesellschaft (FFG) COIN Project http://www.ffg.at/coin-programmlinie-kooperation-und-netzwerke

[1] Open Source Initiative: Definition of Open Source http://www.opensource.org/osd.html

[2] http://git-scm.com/ [2012-01-28]

or a central server. Branching and merging are fast and easy to do.

After installing the git software (binaries are available on the mentioned website and in the repositories of almost every linux distribution), a folder for the new project is created on the local file system.

If there is absolutely no possiblility to install the official git compilation, a pure Java implementation of git called JGit [3] can be used. JGit does not need any installation or extended administrative rights, be a current "Java Runtime Environment" (JRE) has to be installed. Besides the EGit plugin of the Eclipse "Integrated Development Environment" (IDE) there are hardly any additional (graphical) tools which are able to utilize JGit's functionality. Therefore users are bound to Eclipse or the command line interface.

Git also provides a graphical frontend called "Git Gui". Many tasks ranging from basic management of local and remote repositories up to complex merging can be achieved with it. Especially for novice git users Git Gui may provide a more even learning curve than the command line interface (CLI).

Additionally there are some other graphical user interfaces (GUI) which simplify the usage of git. Eclipse IDE (`http://www.eclipse.org/`) includes the EGit (`http://eclipse.org/egit/`) plugin, a graphical git frontend based on the mentioned JGit implementation.

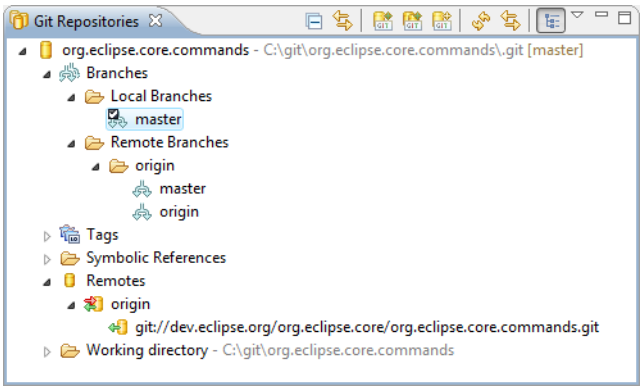

Fig. 1. git repositories presented by EGit / JGit in Eclipse IDE
screenshot provided by `http://eclipse.org/egit/` [2012-01-29]

Besides different, mostly platform dependent tools, SmartGit represents a current and very functional GUI, which is available on all popular operating system. Although SmartGit is sold commercially, a free version for non-commercial projects offered.

3.1 git init & clone

Now the local git repository can be set up by invoking the command line command *git init*. The *git clone* command can be used for creating an exact copy of an available remote repository.

[3] jgit homepage: `http://eclipse.org/jgit/`

How to clone (copy) the source of this paper, including the whole changelog, to a local folder and thereby create a local repository as an exact replica of the online version, is shown in `line 1` of *Listing 1*. After the download has finished, the document can be compiled using the *pdflatex* command.

Note that the example in *Listing 1* shows how to compile the LaTeXfile generated by R and Sweave (both are described in chapter 4). It is necessary to run *pdflatex* two times to produce all references and listings correctly.

Although the implementation of the sources takes place on different linux distributions, the compilation should flawlessly work on other operating systems (OS). While the tools themself are used in the same way on all popular systems, some details - like changing the directory using a terminal and setting up the search path to the binaries - may be different. If the creation of the PDF-file does not work on a specific setup, a prefabricated version is also located in the git repository.

Listing 1. Clone git repository of this paper and compile the included LaTeXfile

```
1 git clone git://github.com/FlorianEndel/
    Mathmod2012.git
2 cd Mathmod2012
3 pdflatex
    Mathmod2012_R_Geospatial_Plotting.tex
4 pdflatex
    Mathmod2012_R_Geospatial_Plotting.tex
```

After creating a new local git repository or cloning remote sources, everyting is ready for editing the contents. Git does not anticipate specific software tools like special editors or file browsers. Any file in the repository can be edited using any favorite program. Solely the folder (including all contents) *.git* has to be ignored. Otherwise the local repository may be damaged.

3.2 git commit, pull, push

All changes can be permantently added to git's history (the changelog) using *git commit* followed by a summary of all changes made (line 1 in listing 2). A commited version of the whole project can be restored later on and users are able to survey all changes over time. This may be quite useful if something breaks during the implementation of a new feature or adjustment of existing code. Also try and error approaches may be applied, because resetting a previous state of the whole project is quite easy and fast. If the status of the whole project at a specific point in time is necessary, for example to compare published and updated versions or include comments from a review process, activating a previous commit using *git checkout [commit−id]* may be feasible.

After commiting changes to the *local* repository, it is possible to merge the local version with the remote repository (if any exists and is configured) and provide all alterations to other team members or a public audiance. Before someone pushes the local version to a (central) git server, all changes made by other people have to be downloaded and included into the own *branch* of the projects source.

Downloading and merging can easily be carried out using *git fetch* and *git merge*. The command *git pull* is a combination of the two previous steps.

If there are no conflicting changes of the source code between the remote and the local verions, git is normally able to merge everything on its own flawlessly. Sooner or later every team faces the problem of conflicting and failing merges. The solution for such issues is not straigth forward. Because the official documentation of git (`http://git-scm.com/documentation`) and a quick `google.com` search may be most helpful, there is no explanation provided within this paper.

Especially during manual merging of conflicting source files, a grphical interface may be very helpful.

Just as the (automatic or manual) marging is finished, it is highly recommended to compile the Sweave and LaTeXfile again to be sure that no new errors were introduced. When the merging and testing procedure is successfully finished, the local changes can be loaded to another remote repository using *git push*.

As uploading (pushing) needs the right to write on a remote location, more configuration is normalley necessary. As there are serveral possibilities to authenticate to a remote location, including different additional tools, signation and encryption, no general explanation can be provided.

The source of this paper is hosted by the free (git) provider `github.com`. As shown in Listing 1, it is quite easy to copy (clone) a project from their webservers. Uploading (pushing) needs some extra configuration which is explained in the excelent tutorial provided on `help.github.com` (pushing: `help.github.com/remotes/#pushing`).

Listing 2. pull and push of a local git repository without merge conflicts

```
git commit -m 'Summary of changes made'
git pull
pdflatex
    Mathmod2012_R_Geospatial_Plotting.tex
pdflatex
    Mathmod2012_R_Geospatial_Plotting.tex
# everything alright?
git push
```

3.3 git: further features

There are much more useful features of git waiting to be explored and used. Yet again official documentation on `http://git-scm.com/documentation` and practical tutorials, for example from `help.github.com`, are recommended.

In the following listing, the most important 'advanced' features are summarized (in alphabetical order):

add *git add* is used to add newly created files (within the folder managed by git) to the current repository.

branch Different branches of the same codebase can be edited independent from another. In fact, every local clone of a remote repository represents a (remote tracking) branch. Branches may be combined using *merge*.

checkout Switching between different branches (and even create new branches) can be estabiched using *git checkout*. Even previousely commited versions can be activated.

log Using the command *git log* all prvious commit messages of the currently activated branch are shown.

rebase Rebase allows to easily change a series of commits, reordering, editing, or squashing commits together into a single commit. It is highly recommended that manipulating the history is performed with great care, because this is one of the rare possibilities to actually loose changes or (under special circumstances) even destroy the repository.

reset The *git reset* command is one of the most complicated functions of git. Be aware that using *reset* is one of the rare possibilities to actually loose changes! Reseting files or complete (commited) versions of a branch may be quite useful. A deeper understandig of git's different states - unstaged, staged, commited - is necessary to effectively use *reset*. A short and informative introduction to the 'normal workflow' of git can be found here: `http://learn.github.com/p/normal.html`.

status *git status* shows the status of the currently activated branch. Most notably new and changed files which may be involved in the next commit are listed.

tag Tags are special signs of commited versions. For example a specific revision of the repository of this paper (listing 1) is marked (tagged) as published.

Fig. 2. example for a git changelog and history of commits visualized with Smartgit
screenshot provided by `http://www.syntevo.com/smartgit/` [2012-01-29]

4. LaTeX, R & SWEAVE

Once the git repository is set up, the Sweave document has be arranged. In the following paragraphs, the mentioned technologies are briefly introduced.

Some central properties are shared by all of them. Availability under a open source license, platform independence, multilingualism, lots of documentation and a strong and proofen community willing to help are the most important characteristics of these

4.1 R

RR Development Core Team (2011) is known as an upcoming statistical environment.

4.2 Sweave

4.3 simple Sweave document

5. ABOUT DATA

getting Data from files, DB and online resources

6. GEOSPATIAL PLOTTING

examples of Geospatial Plots

7. CONCLUSION

REFERENCES

Bivand, R.S., Pebesma, E.J., and Gomez-Rubio, V. (2008). *Applied spatial data analysis with R*. Springer, NY. URL `http://www.asdar-book.org/`.

Group, T. (2011). *PostgreSQL 9.0 Official Documentation - Volume I. The SQL Language*. Fultus Corporation.

Kemper, A. and Eickler, A. (2006). *Datenbanksysteme*. Oldenbourg. URL `http://books.google.at/books?id=YezXpIacjkgC`.

Leisch, F. (2002). Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz (eds.), *Compstat 2002 — Proceedings in Computational Statistics*, 575–580. Physica Verlag, Heidelberg. URL `http://www.stat.uni-muenchen.de/~leisch/Sweave`. ISBN 3-7908-1517-9.

Pebesma, E.J. and Bivand, R.S. (2005). Classes and methods for spatial data in R. *R News*, 5(2), 9–13. URL `http://CRAN.R-project.org/doc/Rnews/`.

R Development Core Team (2011). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

Stein et al. (2011). East-west gradient in cardio-vascular mortality in austria: how much can we explain by following the pattern of risk factors? *International Journal of Health Geographics*, 10, 59.

Wickham, H. (2009). *ggplot2: elegant graphics for data analysis*. Springer New York. URL `http://had.co.nz/ggplot2/book`.