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. Finally some examples of spatial plots using (among others) the *sp* package of 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 and plots 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 should help to overcome some of these challenges. Although there are great learning and information resources, beginning with basic up to sophisticated books for every delineated software product, useful examples 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 an ongoing project may still be challenging.

Additionally the awareness of great open source (Open Source Initiative: Definition of Open Source `http://www.opensource.org/osd.html`) 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 source code 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 (`https://github.com/FlorianEndel/Mathmod2012`). Additionally some useful online resources 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 [1]:

> Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

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

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

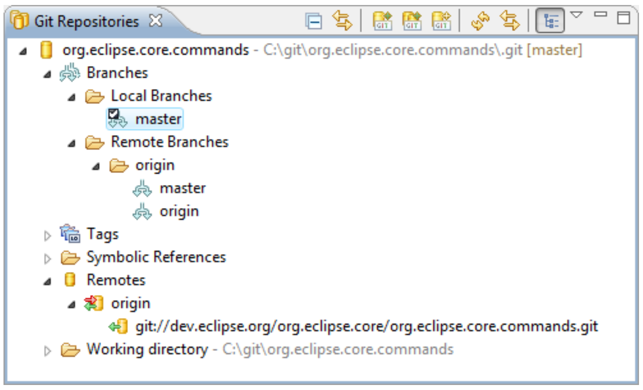
> Every Git clone is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server. Branching and merging are fast and easy to do.

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 possibility to install the official git compilation, a pure Java implementation of git called JGit (jgit homepage: `http://eclipse.org/jgit/`) can be used. JGit does not need any installation or extended administrative rights, but 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 front-end 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 front-end based on the mentioned JGit implementation.



```
Git Repositories
org.eclipse.core.commands - C:\git\org.eclipse.core.commands\.git [master]
  Branches
    Local Branches
      master
    Remote Branches
      origin
        master
        origin
  Tags
  Symbolic References
  Remotes
    origin
      git://dev.eclipse.org/org.eclipse.core/org.eclipse.core.commands.git
  Working directory - C:\git\org.eclipse.core.commands
```

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 is offered.

Additionally RStudio, a quite new development environment for R, supports git (and subversion) directly since January 2012.

3.1 git init & clone

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

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 provided *run.sh* script.

Note that the example in *Listing 1* shows how to compile the LaTeX-file generated by R and Sweave (both are described in chapter 4). Be aware that some settings need to be adapted to the local system. They are all stored in *./include/init.Rnw*. It is possible to maintain a local configuration and merge updates to the local source code by removing this file from the repository. It is also necessary to install all loaded R packages (*install.packages('name')*).

There is a script (*run.sh*) included which helps creating the document.

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 themselves 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 - is sometimes 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 LaTeX-file

```
1  git clone git://github.com/FlorianEndel/
       Mathmod2012.git
2  cd Mathmod2012
3
4  # start R & Sweave
5  R CMD Sweave Mathmod2012_R_Geospatial_Plotting.
       Rnw
6
7  # if there was no error...
8  pdflatex Mathmod2012_R_Geospatial_Plotting.tex
9
10 # Bibtex for References
11 bibtex Mathmod2012_R_Geospatial_Plotting
12
13 # 2x pdflatex to get all indices right
14 pdflatex Mathmod2012_R_Geospatial_Plotting.tex
15 pdflatex Mathmod2012_R_Geospatial_Plotting.tex
16
17 # shellscript executing this tasks
18 ./run.sh
```

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 can easily be damaged.

3.2 git commit, pull, push

All changes can be permanently added to git's history (the changelog) using `git commit` followed by a summary of the changes made (line 1 in listing 2). A committed 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 committing 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 audience. 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 versions, 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 straight 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 further explanation provided within this paper.

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

Just as the (automatic or manual) merging is finished, it is highly recommended to compile the Sweave and LaTeX-file 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 normally necessary. There are several possibilities to authenticate to a remote location, including different additional tools, signation and encryption. Therefore 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 web-servers. Uploading (pushing) needs some extra configuration which is explained in the excellent 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

```
1 git commit -m 'Summary of changes made'
2 git pull
3 ./run.sh
4 # everything alright?
5 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 code-base 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 established using `git checkout`. Even previously committed versions can be activated.

log Using the command `git log` all previous 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* may delete data irretrievably! Reseting files or complete (committed) versions of a branch, mixing different versions and revoking committed files are some areas of application. A deeper understanding of git's different states - unstaged, staged, committed - 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`.

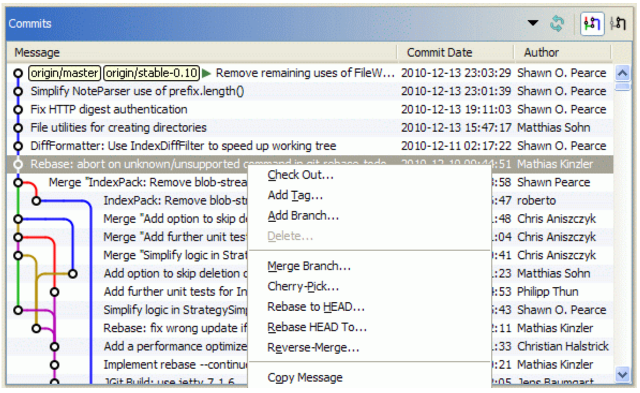
status `git status` shows the status of the currently activated branch. Most notably new and changed files which may be involved in the next commit are listed.

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

4. LaTeX, R & SWEAVE

Once the git repository is set up, the Sweave document has to be arranged. In the following paragraphs involved 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 products.



Message	Commit Date	Author
origin/master origin/stable-0.10 ▶ Remove remaining uses of FileW...	2010-12-13 23:03:29	Shawn O. Pearce
Simplify NoteParser use of prefix.length()	2010-12-13 23:01:39	Shawn O. Pearce
Fix HTTP digest authentication	2010-12-13 19:11:03	Shawn O. Pearce
File utilities for creating directories	2010-12-13 15:47:17	Matthias Sohn
DiffFormatter: Use IndexDiffFilter to speed up working tree	2010-12-11 02:17:22	Shawn O. Pearce
Rebase: abort on unknown/unsupported command in git rebase todo	2010-12-10 00:44:51	Mathias Kinzler
Merge "IndexPack: Remove blob-strea	:58	Shawn Pearce
IndexPack: Remove blob-st	:47	roberto
Merge "Add option to skip d	:48	Chris Aniszczyk
Merge "Add further unit tes	:04	Chris Aniszczyk
Merge "Simplify logic in Strat	:41	Chris Aniszczyk
Add option to skip deletion c	:23	Matthias Sohn
Add further unit tests for In	:53	Philipp Thun
Simplify logic in StrategySim	:43	Shawn O. Pearce
Rebase: fix wrong update if	:11	Mathias Kinzler
Add a performance optimize	:33	Christian Halstrick
Implement rebase --continu	:21	Mathias Kinzler
JGit Build: use jetty 7.1.6	:05	Jens Baumgart

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]

4.1 R

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

As an open source successor of S, it is known to be a powerful and flexible programming language and computing environment. Especially everyones ability to utilize and contribute to the very rich environment of additional packages hosted mainly on “The Comprehensive R Archive Network” are good reasons to give R a try. More advanced userRs (spelling “user” and “developer” with capital ’R’s is common in the R community) may use different extensions in addition to the variety of basic packages a normal installation offers. Most packages provide worthwhile introductions and there is even extensive literature for some of them.

R is mainly used through a command line interface. Although this - in comparison to current graphical user interfaces outdated - way to communicate with a computer program got a quite steep learning curve, practical advantages induce many users to learn and stick with it. Mainly the ease of combining playful exploration of datasets and automatic reproduction of any steps taken, using script- or batch-files, are advantageous.

Also the following examples, using special packages providing methods for spatial data, may be executed on a simple terminal, a graphical front-end to R, a remote server or inside a Sweave document. This flexibility enables the usage of the same R source code within a database [2], a local PC and even servers and cloud services.

Of course there are also some graphical user interfaces which may support the usage of R. The most popular and beginner friendly solutions are Eclipse with the StatEt extension and the lately published RStudio (`http://rstudio.org/`). Both are open source and platform independent solution, which offer many other advantages like project management, integration of version management (git), support for Sweave and a client/server separation.

While Eclipse provides considerably more additional plug-ins and possibilities, RStudio can be highly recommended to all useRs.

4.2 Sweave

Sweave is defined in the official manual (Leisch, 2002) as follows:

> Sweave provides a flexible framework for mixing text and S code for automatic document generation. A single source file contains both documentation text and S code, which are then woven into a final document

As explained above, R is an open source implementation of S.

In 2002, Sweave was presented in the former "R Journal" (`http://journal.r-project.org/`) called "R News" (official archive of R News: `http://cran.r-project.org/doc/Rnews/`). In the first part of this introduction, Sweave is summarized briefly (Leisch, 2002):

> The purpose of Sweave is to create dynamic reports, which can be updated automatically if data or analysis change. Instead of inserting a prefabricated graph or table into the report, the master document contains the R code necessary to obtain it. When run through R, all data analysis output (tables, graphs, ...) is created on the fly and inserted into a final LaTeX document. The report can be automatically updated if data or analysis change, which allows for truly reproducible research.

Normally Sweave files got the file extension `.Rnw`. After executing `R CMD Sweave` followed be the filename of the Sweave document, all R-code is extracted from the document and executed using R. The results are merged with the LaTeX code of the `.Rnw` file and saved to as `.tex` file with the same filename. Afterwards the common compilation using `latex`, `pdflatex`, `bibtex`, and relatives can be performed.

4.3 simple Sweave document

This subsection shows the basic structure of a Sweave document. As this paper is written using LaTeX and R, another working example can be found in the git repository, described in Listing 1 on page 2.

At the beginning (line 1 to 5) of Listing 3, a very simple LaTeX-Document is shown. On line 6 an R-environment is declared using the "noweb" syntax (Ramsey, 1994; Johnson and Johnson, 2000). Between
`«label1, echo=false, results=hide»=`
and the "at"-sign, R source code can be located. The header of the environment holds some (local) settings an the name, which is displayed during the compilation process. The *results* setting defines how output generated by R should be handled by the LaTeX processor and *echo* configures whether messages from R should be included in the output.

[2] for example *plr* (`http://www.joeconway.com/plr/` [2012-01-30]) for PostgreSQL (Group, T.P.G.D., 2011)

More details about configuration of the noweb-environment of Sweave can be found in the official manual (Leisch, 2011).

Some additional functionalities are shown in the example. The content of `\Sexpr{#R-Code}` is also executed by R and any output is included in the document. This may be quite useful if single values of an calculation should be inserted into continuous text.

`\SweaveInput{filename}` can be used to include another Sweave file. Although importing additional source code can be realized with R, partitioning a document to several Sweave files is advisable.

All single R environments which are executed in the same process share one global namespace. Also libraries, like `xtable` (loaded on line 7), which eases the creation of LaTeX-tables out of data stored in R, are available throughout the document.

Listing 3. basic structure of a Sweave document

```
 1 \documentclass[a4paper]{article}
 2 \begin{document}
 3
 4 <<label1, echo=false, results=hide>≥
 5 library(xtable)
 6 # R code following
 7 @
 8
 9 <<LaTex_Output, echo=hide, results=tex>≥
10 # R code generating \LaTeX
11 @
12
13 <<>≥ # empty header, standard settings
14 @
15
16 Normal Text and results from \Sexpr{print('R')}.
17
18 \SweaveInput{./includes/config}
19
20 \end{document}
```

5. GEOSPATIAL PLOTTING

Finally the most important preparations are ready and some geospatial plots generated by R can be shown. In this document, a slightly more complex setup for Sweave is used.

Most current packages which are dealing with spatial data are using the extensive *sp* package (Bivand et al., 2008). Additionally other extensions like *maptools* to handle shapefiles, *rgdal* as bridge to the geospatial abstraction library "GDAL" (`http://www.gdal.org/`) and *ggplot2* (Wickham, 2009) for advanced graphics are used.

The connection to databases (PostgreSQL), usage of multiple processors in parallel (with *snow* and *snowfall* (Tierney et al., 2011; Knaus, 2010)), advanced logging (*log4r* (White, 2011)) and caching (*cacheSweave*) are configured and partly used, but not explained in more detail.

5.1 visualizing friendships of facebook users

The first example is not created by the author. It illustrates not only a quite beautiful graphic made with the help of R, but also inspired some public interest and many speculations, whether such picture can be created using an open source technology. (In fact R was used for plotting, but some reworking took place afterwards.) The map shows the relation of facebook friends (Butler, 2010).

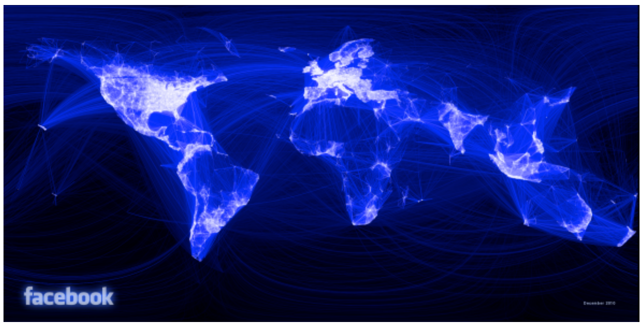


Fig. 3. visualizing friendships of facebook users

5.2 map connections with great circles

After the mentioned discussion about the friendship plot of facebook users, Nathan Yau of `flowingdata.com` showed how to use great circles (Riemannian circle) provided by the *geosphere* package. As example he chose to visualize flight data of US airports[3].

The presented plot is an adoption of this example. Open Data sources are downloaded from the Internet, combined and plotted within R and directly added to this document. Therefore, changing the displayed airline or plotting all airlines (in parallel using multiple processors simultaneously), can be easily achieved.

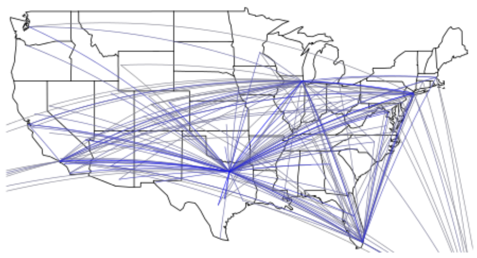


Fig. 4. flights of an US American airline in North America

5.3 Google Earth and Open Street Map

Using the *RgoogleMaps* package (Loecher, 2011), spatial data and pictures provided by Google can be downloaded. To get access to the "Maps Image APIs" of Google, a special key is necessary. Although limited access would be free of charge, the provided example uses data from the OpenStreetMap project (`http://www.openstreetmap.org/`).

The package *osmar 1.1-0* is specialized on the connection between R and OpenStreetMap, but was (at least at

[3] `http://flowingdata.com/2011/05/11/how-to-map-connections-with-great-circles/`

Debian Sid) quite hard to install and did not function correctly.

A map of Vienna and parts of the surrounding area, which was loaded directly from OpenStreetMap using R, is displayed in the following example. Also the manipulation of the map using coordinates is presented by marking Vienna University of Technology (TU Vienna).

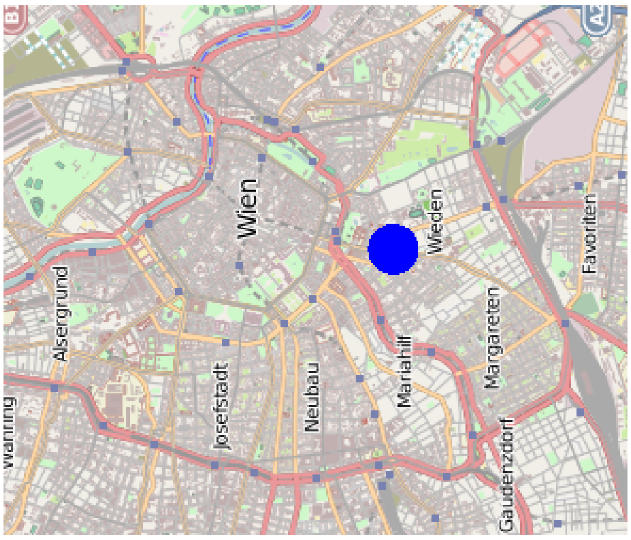


Fig. 5. manipulated OpenStreetMap of Vienna

5.4 gadm.org

The last example features a political map from the "GADM database of Global Administrative Areas" (`http://www.gadm.org/`). Maps in different granularity and formats are provided (for academic or non-profit reasons!) for free.

After loading the file with R, some corrections of the text encoding has to be performed. The data seems to be stored as "UTF-8", but wrongly imported and displayed in the "ISO-8859-2" format. After explicit conversion everything should run smoothly.

To show off the possibilities of this map, some federal states are removed. The coloring is created using a random heat-map. Only one region ("Tulln") is completely black. Additionally all borders are painted black and the names of each district is displayed over the particular centroid (without collisions).

6. RESULT

These simple examples may induce a vision of the vast possibilities of the presented techniques and especially their combination. Integration with established systems like Grass, utilization of Open Data and Open Government resources, cloud computing and many other current trends can all be handled sophisticatedly. The low price and freedom open source software provides is just another argument to try out these modern programs.

Summarizing, not only the spatial plots themselves, but also the whole workflow of a (distributed) project using Git, Sweave, R and LaTeX is the result and message of this paper.

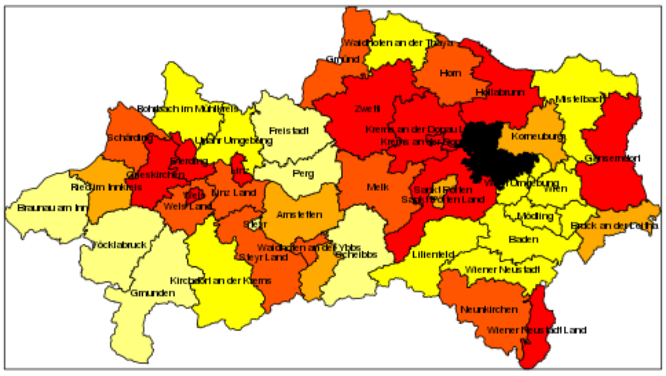


Fig. 6. lower Austria from gadm.org

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/`.

Butler, P. (2010). Visualizing friendships. Facebook. URL `https://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919`.

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

Johnson, A.L. and Johnson, B.C. (2000). Literate programming using noweb. *Linux Journal*, 42(42), 1–12. URL `ftp://ftp.ssc.com/pub/lj/listings/issue42/2188.tgz`.

Knaus, J. (2010). *snowfall: Easier cluster computing (based on snow).* URL `http://CRAN.R-project.org/package=snowfall`. R package version 1.84.

Leisch, F. (2011). *Sweave User Manual.*

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.

Loecher, M. (2011). *RgoogleMaps: Overlays on Google map tiles in R.* URL `http://CRAN.R-project.org/package=RgoogleMaps`. R package version 1.1.9.15.

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.

Ramsey, N. (1994). Literate programming simplified. *IEEE Software*, 11(5), 97–105. URL `http://doi.ieeecomputersociety.org/10.1109/52.311070`.

Tierney, L., Rossini, A.J., Li, N., and Sevcikova, H. (2011). *snow: Simple Network of Workstations.* URL `http://CRAN.R-project.org/package=snow`. R package version 0.3-8.

White, J.M. (2011). *log4r: A simple logging system for R, based on log4j.* URL `http://CRAN.R-project.org/package=log4r`. R package version 0.1-3.

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