

The Journal

Volume 3/1, June 2011

A peer-reviewed, open-access publication of the R Foundation
for Statistical Computing

Contents

Editorial	3
---------------------	---

Contributed Research Articles

testthat: Get Started with Testing	5
Content-Based Social Network Analysis of Mailing Lists	11
Rmetrics - timeDate Package	19
The digitize Package: Extracting Numerical Data from Scatterplots	25
Differential Evolution with DEoptim	27
rworldmap: A New R package for Mapping Global Data	35
Cryptographic Boolean Functions with R	44
Raster Images in R Graphics	48
Probabilistic Weather Forecasting in R	55
Analyzing an Electronic Limit Order Book	64

Help Desk

Giving a useR! Talk	69
Tips for Presenting Your Work	72

Book Reviews

Forest Analytics with R (Use R)	75
---	----

News and Notes

Conference Review: Third Meeting of Polish R Users, The Influenza Challenge	76
Conference Review: Kickoff Workshop for Project MOSAIC	78
Changes in R	79
Changes on CRAN	89
News from the Bioconductor Project	101
R Foundation News	102

The  Journal is a peer-reviewed publication of the R Foundation for Statistical Computing. Communications regarding this publication should be addressed to the editors. All articles are copyrighted by the respective authors.

Prospective authors will find detailed and up-to-date submission instructions on the Journal's homepage.

Editor-in-Chief:

Heather Turner

Pharmatherapeutics Statistics

Pfizer Global R & D (UK)

Ramsgate Road (ipc 292)

Sandwich

Kent CT13 9NJ

UK

Editorial Board:

Peter Dalgaard, Martyn Plummer, and Hadley Wickham.

Editor Programmer's Niche:

Bill Venables

Editor Help Desk:

Uwe Ligges

Editor Book Reviews:

G. Jay Kerns

Department of Mathematics and Statistics

Youngstown State University

Youngstown, Ohio 44555-0002

USA

gkerns@ysu.edu

R Journal Homepage:

<http://journal.r-project.org/>

Email of editors and editorial board:

firstname.lastname@R-project.org

The R Journal is indexed/abstracted by EBSCO, DOAJ.

Editorial

by Heather Turner

This issue comes approximately one month after CRAN passed a new milestone: the number of packages in the repository had reached 3000. With such a large number of packages (and the number is still rising of course) it becomes increasingly difficult for package authors to get their work known and for R users to find packages that will be useful to them. The R Journal plays an important role in this context; it provides a platform for developers to introduce their packages in a user-friendly form and although the presented packages are not validated, publication in The R Journal gives the quality assurance that comes from thorough peer review.

Given that The R Journal only publishes around 20 contributed articles a year however, there is more that needs to be done to communicate developments on CRAN. The editorial board had some discussion recently over whether to keep the traditional "Changes on CRAN" section. Although there are other ways to keep up-to-date with CRAN news, notably CRANberries (<http://dirk.eddelbuettel.com/cranberries/>) and cran-tastic (<http://crantastic.org/>), some people still appreciate an occasional round-up that they can browse through. So this section stays for now, but we are open to suggested improvements.

Another important way to keep abreast of developments in the R community is via conferences and workshops. As program chair for user! 2011, I have been pulling together the program for this

conference at the same time as preparing this issue. At this year's user! there will be around 150 contributed talks and more than 30 contributed posters from across the R community. Looking forward to this conference, we invited Rob Hyndman and Di Cook to share their presentation tips in the Help Desk column of this issue. Of course, useRs present their work at many and varied conferences, and this is reflected in our News section where we have a report of an experimental format used for the Polish R Users' Conference and a report of workshops run by Project MOSAIC, a statistical education initiative in which R features prominently.

Although Vince Carey officially stepped down from the editorial board at the end of last year, this issue features several articles that he saw through to publication, so we thank him once again for all his work on the Journal. We welcome Hadley Wickham on board, an enthusiastic contributor to the R community, as evidenced not only by his own article in this issue, but by the social network analysis of R mailing lists presented by Angela Bohn and colleagues. These two articles are just the start of a fine collection of contributed articles and we also have a new book review to report.

I hope this issue goes some way to helping useRs navigate their way through the R world!

*Heather Turner
PharmaTherapeutics Statistics, Pfizer Ltd, UK
Heather.Turner@R-project.org*

testthat: Get Started with Testing

by Hadley Wickham

Abstract Software testing is important, but many of us don't do it because it is frustrating and boring. **testthat** is a new testing framework for R that is easy learn and use, and integrates with your existing workflow. This paper shows how, with illustrations from existing packages.

Introduction

Testing should be something that you do all the time, but it's normally painful and boring. **testthat** (Wickham, 2011) tries to make testing as painless as possible, so you do it as often as possible. To make that happen, **testthat**:

- Provides functions that make it easy to describe what you expect a function to do, including catching errors, warnings and messages.
- Easily integrates in your existing workflow, whether it's informal testing on the command line, building test suites, or using '`R CMD check`'.
- Can re-run tests automatically as you change your code or tests.
- Displays test progress visually, showing a pass, fail or error for every expectation. If you're using the terminal, it'll even colour the output.

testthat draws inspiration from the xUnit family of testing packages, as well from many of the innovative Ruby testing libraries like `rspec`¹, `testy`², `bacon`³ and `cucumber`⁴. I have used what I think works for R, and abandoned what doesn't, creating a testing environment that is philosophically centred in R.

Why test?

I wrote **testthat** because I discovered I was spending too much time recreating bugs that I had previously fixed. While I was writing the original code or fixing the bug, I'd perform many interactive tests to make sure the code worked, but I never had a system for retaining these tests and running them, again and again. I think this is a common development practice of R programmers: it's not that we don't test our code, it's that we don't store our tests so they can be re-run automatically.

In part, this is because existing R testing packages, such as **RUnit** (Burger et al., 2009) and **svUnit** (Grosjean, 2009), require a lot of up-front work to get started. One of the motivations of **testthat** is to make the initial effort as small as possible, so you can start off slowly and gradually ramp up the formality and rigour of your tests.

It will always require a little more work to turn your casual interactive tests into reproducible scripts: you can no longer visually inspect the output, so instead you have to write code that does the inspection for you. However, this is an investment in the future of your code that will pay off in:

- Decreased frustration. Whenever I'm working to a strict deadline I always seem to discover a bug in old code. Having to stop what I'm doing to fix the bug is a real pain. This happens less when I do more testing, and I can easily see which parts of my code I can be confident in by looking at how well they are tested.
- Better code structure. Code that's easy to test is usually better designed. I have found writing tests makes me extract out the complicated parts of my code into separate functions that work in isolation. These functions are easier to test, have less duplication, are easier to understand and are easier to re-combine in new ways.
- Less struggle to pick up development after a break. If you always finish a session of coding by creating a failing test (e.g. for the feature you want to implement next) it's easy to pick up where you left off: your tests let you know what to do next.
- Increased confidence when making changes. If you know that all major functionality has a test associated with it, you can confidently make big changes without worrying about accidentally breaking something. For me, this is particularly useful when I think of a simpler way to accomplish a task - often my simpler solution is only simpler because I've forgotten an important use case!

Test structure

testthat has a hierarchical structure made up of expectations, tests and contexts.

¹<http://rspec.info/>

²<http://github.com/ahoward/testy>

³<http://github.com/chneukirchen/bacon>

⁴<http://wiki.github.com/aslakhellesoy/cucumber/>

- An *expectation* describes what the result of a computation should be. Does it have the right value and right class? Does it produce error messages when you expect it to? There are 11 types of built-in expectations.
- A *test* groups together multiple expectations to test one function, or tightly related functionality across multiple functions. A test is created with the `test_that` function.
- A *context* groups together multiple tests that test related functionality.

These are described in detail below. Expectations give you the tools to convert your visual, interactive experiments into reproducible scripts; tests and contexts are just ways of organising your expectations so that when something goes wrong you can easily track down the source of the problem.

Expectations

An expectation is the finest level of testing; it makes a binary assertion about whether or not a value is as you expect. An expectation is easy to read, since it is nearly a sentence already: `expect_that(a, equals(b))` reads as “I expect that *a* will equal *b*”. If the expectation isn’t true, `testthat` will raise an error.

There are 11 built-in expectations:

- `equals()` uses `all.equal()` to check for equality with numerical tolerance:

```
# Passes
expect_that(10, equals(10))
# Also passes
expect_that(10, equals(10 + 1e-7))
# Fails
expect_that(10, equals(10 + 1e-6))
# Definitely fails!
expect_that(10, equals(11))
```

- `is_identical_to()` uses `identical()` to check for exact equality:

```
# Passes
expect_that(10, is_identical_to(10))
# Fails
expect_that(10, is_identical_to(10 + 1e-10))
```

- `is_equivalent_to()` is a more relaxed version of `equals()` that ignores attributes:

```
# Fails
expect_that(c("one" = 1, "two" = 2),
           equals(1:2))
# Passes
expect_that(c("one" = 1, "two" = 2),
           is_equivalent_to(1:2))
```

- `is_a()` checks that an object `inherit()`s from a specified class:

```
model <- lm(mpg ~ wt, data = mtcars)
# Passes
expect_that(model, is_a("lm"))
# Fails
expect_that(model, is_a("glm"))
```

- `matches()` matches a character vector against a regular expression. The optional `all` argument controls where all elements or just one element need to match. This code is powered by `str_detect()` from the `stringr` (Wickham, 2010) package:

```
string <- "Testing is fun!"
# Passes
expect_that(string, matches("Testing"))
# Fails, match is case-sensitive
expect_that(string, matches("testing"))
# Passes, match can be a regular expression
expect_that(string, matches("t.+ting"))
```

- `prints_text()` matches the printed output from an expression against a regular expression:

```
a <- list(1:10, letters)
# Passes
expect_that(str(a), prints_text("List of 2"))
# Passes
expect_that(str(a),
            prints_text(fixed("int [1:10]")))
```

- `shows_message()` checks that an expression shows a message:

```
# Passes
expect_that(library(mgcv),
            shows_message("This is mgcv"))
```

- `gives_warning()` expects that you get a warning:

```
# Passes
expect_that(log(-1), gives_warning())
expect_that(log(-1),
            gives_warning("NaNs produced"))
# Fails
expect_that(log(0), gives_warning())
```

- `throws_error()` verifies that the expression throws an error. You can also supply a regular expression which is applied to the text of the error:

```
# Fails
expect_that(1 / 2, throws_error())
# Passes
expect_that(1 / "a", throws_error())
# But better to be explicit
expect_that(1 / "a",
            throws_error("non-numeric argument"))
```

Full	Short cut
<code>expect_that(x, is_true())</code>	<code>expect_true(x)</code>
<code>expect_that(x, is_false())</code>	<code>expect_false(x)</code>
<code>expect_that(x, is_a(y))</code>	<code>expect_is(x, y)</code>
<code>expect_that(x, equals(y))</code>	<code>expect_equal(x, y)</code>
<code>expect_that(x, is_equivalent_to(y))</code>	<code>expect_equivalent(x, y)</code>
<code>expect_that(x, is_identical_to(y))</code>	<code>expect_identical(x, y)</code>
<code>expect_that(x, matches(y))</code>	<code>expect_matches(x, y)</code>
<code>expect_that(x, prints_text(y))</code>	<code>expect_output(x, y)</code>
<code>expect_that(x, shows_message(y))</code>	<code>expect_message(x, y)</code>
<code>expect_that(x, gives_warning(y))</code>	<code>expect_warning(x, y)</code>
<code>expect_that(x, throws_error(y))</code>	<code>expect_error(x, y)</code>

Table 1: Expectation shortcuts

- `is_true()` is a useful catchall if none of the other expectations do what you want - it checks that an expression is true. `is_false()` is the complement of `is_true()`.

If you don't like the readable, but verbose, `expect_that` style, you can use one of the shortcut functions described in Table 1.

You can also write your own expectations. An expectation should return a function that compares its input to the expected value and reports the result using `expectation()`. `expectation()` has two arguments: a boolean indicating the result of the test, and the message to display if the expectation fails. Your expectation function will be called by `expect_that` with a single argument: the actual value. The following code shows the simple `is_true` expectation. Most of the other expectations are equally simple, and if you want to write your own, I'd recommend reading the source code of `testthat` to see other examples.

```
is_true <- function() {
  function(x) {
    expectation(
      identical(x, TRUE),
      "isn't true"
    )
  }
}
```

Running a sequence of expectations is useful because it ensures that your code behaves as expected. You could even use an expectation within a function to check that the inputs are what you expect. However, they're not so useful when something goes wrong: all you know is that something is not as expected, you know nothing about where the problem is. Tests, described next, organise expectations into coherent blocks that describe the overall goal of that set of expectations.

Tests

Each test should test a single item of functionality and have an informative name. The idea is that when a test fails, you should know exactly where to look for the problem in your code. You create a new test with `test_that`, with parameters name and code block. The test name should complete the sentence "Test that ..." and the code block should be a collection of expectations. When there's a failure, it's the test name that will help you figure out what's gone wrong.

Figure 1 shows one test of the `floor_date` function from `lubridate` (Wickham and Grolemund, 2010). There are 7 expectations that check the results of rounding a date down to the nearest second, minute, hour, etc. Note how we've defined a couple of helper functions to make the test more concise so you can easily see what changes in each expectation.

Each test is run in its own environment so it is self-contained. The exceptions are actions which have effects outside the local environment. These include things that affect:

- The filesystem: creating and deleting files, changing the working directory, etc.
- The search path: package loading & detaching, `attach`.
- Global options, like `options()` and `par()`.

When you use these actions in tests, you'll need to clean up after yourself. Many other testing packages have set-up and teardown methods that are run automatically before and after each test. These are not so important with `testthat` because you can create objects outside of the tests and rely on R's copy-on-modify semantics to keep them unchanged between test runs. To clean up other actions you can use regular R functions.

You can run a set of tests just by `source()`ing a file, but as you write more and more tests, you'll probably want a little more infrastructure. The first

```

test_that("floor_date works for different units", {
  base <- as.POSIXct("2009-08-03 12:01:59.23", tz = "UTC")

  is_time <- function(x) equals(as.POSIXct(x, tz = "UTC"))
  floor_base <- function(unit) floor_date(base, unit)

  expect_that(floor_base("second"), is_time("2009-08-03 12:01:59"))
  expect_that(floor_base("minute"), is_time("2009-08-03 12:01:00"))
  expect_that(floor_base("hour"), is_time("2009-08-03 12:00:00"))
  expect_that(floor_base("day"), is_time("2009-08-03 00:00:00"))
  expect_that(floor_base("week"), is_time("2009-08-02 00:00:00"))
  expect_that(floor_base("month"), is_time("2009-08-01 00:00:00"))
  expect_that(floor_base("year"), is_time("2009-01-01 00:00:00"))
})

```

Figure 1: A test case from the **lubridate** package.

part of that infrastructure is contexts, described below, which give a convenient way to label each file, helping to locate failures when you have many tests.

Contexts

Contexts group tests together into blocks that test related functionality and are established with the code: `context("My context")`. Normally there is one context per file, but you can have more if you want, or you can use the same context in multiple files.

Figure 2 shows the context that tests the operation of the `str_length` function in **stringr**. The tests are very simple, but cover two situations where `nchar()` in base R gives surprising results.

Workflow

So far we've talked about running tests by `source()`ing in R files. This is useful to double-check that everything works, but it gives you little information about what went wrong. This section shows how to take your testing to the next level by setting up a more formal workflow. There are three basic techniques to use:

- Run all tests in a file or directory `test_file()` or `test_dir()`.
- Automatically run tests whenever something changes with `autotest`.
- Have R CMD check run your tests.

Testing files and directories

You can run all tests in a file with `test_file(path)`. Figure 3 shows the difference between `test_file` and `source` for the tests in Figure 2, as well as those same tests for `nchar`. You can see the advantage of `test_file` over `source`: instead of seeing the first failure, you see the performance of all tests.

Each expectation is displayed as either a green dot (indicating success) or a red number (indicating failure). That number indexes into a list of further details, printed after all tests have been run. What you can't see is that this display is dynamic: a new dot gets printed each time a test passes and it's rather satisfying to watch.

`test_dir` will run all of the test files in a directory, assuming that test files start with `test` (so it's possible to intermix regular code and tests in the same directory). This is handy if you're developing a small set of scripts rather than a complete package. The following shows the output from the **stringr** tests. You can see there are 12 contexts with between 2 and 25 expectations each. As you'd hope in a released package, all the tests pass.

```

> test_dir("inst/tests/")
String and pattern checks : .....
Detecting patterns : .....
Duplicating strings : .....
Extract patterns : ...
Joining strings : .....
String length : .....
Locations : .....
Matching groups : .....
Test padding : .....
Splitting strings : .....
Extracting substrings : .....
Trimming strings : .....

```

If you want a more minimal report, suitable for display on a dashboard, you can use a different reporter. `testthat` comes with three reporters: `stop`, `minimal` and `summary`. The `stop` reporter is the default and `stop()`s whenever a failure is encountered; the `summary` report is the default for `test_file` and `test_dir`. The `minimal` reporter prints '.' for success, 'E' for an error and 'F' for a failure. The following output shows (some of) the output from running the **stringr** test suite with the `minimal` reporter.

```

> test_dir("inst/tests/", "minimal")
.....
```

```

context("String length")

test_that("str_length is number of characters", {
  expect_that(str_length("a"), equals(1))
  expect_that(str_length("ab"), equals(2))
  expect_that(str_length("abc"), equals(3))
})

test_that("str_length of missing is missing", {
  expect_that(str_length(NA), equals(NA_integer_))
  expect_that(str_length(c(NA, 1)), equals(c(NA, 1)))
  expect_that(str_length("NA"), equals(2))
})

test_that("str_length of factor is length of level", {
  expect_that(str_length(factor("a")), equals(1))
  expect_that(str_length(factor("ab")), equals(2))
  expect_that(str_length(factor("abc")), equals(3))
})

```

Figure 2: A complete context from the **stringr** package that tests the `str_length` function for computing string length.

```

> source("test-str_length.r")
> test_file("test-str_length.r")
.....
> source("test-nchar.r")
Error: Test failure in 'nchar of missing is missing'
* nchar(NA) not equal to NA_integer_
'is.NA' value mismatch: 0 in current 1 in target
* nchar(c(NA, 1)) not equal to c(NA, 1)
'is.NA' value mismatch: 0 in current 1 in target
> test_file("test-nchar.r")
...12..34

1. Failure: nchar of missing is missing -----
nchar(NA) not equal to NA_integer_
'is.NA' value mismatch: 0 in current 1 in target

2. Failure: nchar of missing is missing -----
nchar(c(NA, 1)) not equal to c(NA, 1)
'is.NA' value mismatch: 0 in current 1 in target

3. Failure: nchar of factor is length of level -----
nchar(factor("ab")) not equal to 2
Mean relative difference: 0.5

4. Failure: nchar of factor is length of level -----
nchar(factor("abc")) not equal to 3
Mean relative difference: 0.6666667

```

Figure 3: Results from running the `str_length` context, as well as results from running a modified version that uses `nchar`. `nchar` gives the length of `NA` as 2, and converts factors to integers before calculating length. These tests ensures that `str_length` doesn't make the same mistakes.

Autotest

Tests are most useful when run frequently, and autotest takes that idea to the limit by re-running your tests whenever your code or tests change. autotest() has two arguments, `code_path` and `test_path`, which point to a directory of source code and tests respectively.

Once run, autotest() will continuously scan both directories for changes. If a test file is modified, it will test that file; if a code file is modified, it will reload that file and rerun all tests. To quit, you'll need to press `Ctrl + Break` on windows, `Escape` in the Mac GUI, or `Ctrl + C` if running from the command line.

This promotes a workflow where the *only* way you test your code is through tests. Instead of modify-save-source-check you just modify and save, then watch the automated test output for problems.

R CMD check

If you are developing a package, you can have your tests automatically run by ‘R CMD check’. I recommend storing your tests in `inst/tests/` (so users also have access to them), then including one file in `tests/` that runs all of the package tests. The `test_package(package_name)` function makes this easy. It:

- Expects your tests to be in the `inst/tests/` directory.
- Evaluates your tests in the package namespace (so you can test non exported functions).
- Throws an error at the end if there are any test failures. This means you'll see the full report of test failures and ‘R CMD check’ won't pass unless all tests pass.

This setup has the additional advantage that users can make sure your package works correctly in their run-time environment.

Future work

There are two additional features I'd like to incorporate in future versions:

- Code coverage. It's very useful to be able to tell exactly what parts of your code have been tested. I'm not yet sure how to achieve this in R, but it might be possible with a combination of `RProf` and `codetools` (Tierney, 2009).
- Graphical display for `auto_test`. I find that the more visually appealing I make testing, the more fun it becomes. Coloured dots are pretty primitive, so I'd also like to provide a GUI widget that displays test output.

Bibliography

- M. Burger, K. Juenemann, and T. Koenig. *RUnit: R Unit test framework*, 2009. URL <http://CRAN.R-project.org/package=RUnit>. R package version 0.4.22.
- P. Grosjean. *SciViews-R: A GUI API for R*. URL <http://www.sciviews.org/SciViews-R>. UMH, Mons, Belgium, 2009.
- L. Tierney. *codetools: Code Analysis Tools for R*, 2009. URL <http://CRAN.R-project.org/package=codetools>. R package version 0.2-2.
- H. Wickham. *stringr: Make it easier to work with strings.*, 2010. URL <http://CRAN.R-project.org/package=stringr>. R package version 0.4.
- H. Wickham. *testthat: Testthat code. Tools to make testing fun :)*, 2011. URL <http://CRAN.R-project.org/package=testthat>. R package version 0.5.
- H. Wickham and G. Grolemund. *lubridate: Make dealing with dates a little easier*, 2010. URL <http://www.jstatsoft.org/v40/i03/>. R package version 0.1.
- Hadley Wickham
Department of Statistics
Rice University
6100 Main St MS#138
Houston TX 77005-1827
USA
hadley@rice.edu*

Content-Based Social Network Analysis of Mailing Lists

by Angela Bohn, Ingo Feinerer, Kurt Hornik and Patrick Mair

Abstract Social Network Analysis (SNA) provides tools to examine relationships between people. Text Mining (TM) allows capturing the text they produce in Web 2.0 applications, for example, however it neglects their social structure. This paper applies an approach to combine the two methods named “content-based SNA”. Using the R mailing lists, R-help and R-devel, we show how this combination can be used to describe people’s interests and to find out if authors who have similar interests actually communicate. We find that the expected positive relationship between sharing interests and communicating gets stronger as the centrality scores of authors in the communication networks increase.

Introduction

Social Network Analysis (SNA) provides powerful methods to study the relationships between people expressed as binary or weighted adjacency matrices. It can be used to find influential or popular nodes, communities and informal hierarchies. However, it is limited in the sense that it cannot capture the context of their encounter. An actor might be a regarded adviser or trend setter concerning one topic and might not know anything about another. If all his or her relationships are expressed as purely numerical networks, a differentiation is not possible.

In open source software communities, a large part of the developers’ communication and thus collaboration happens electronically via e-mails and forums. In the R mailing lists, R-help and R-devel, all kinds of application and development related questions are discussed. From experts to newcomers, everyone shares the same platform, so a view on the entire network does not offer a detailed picture, leaving the question of how well the communication really works in the R community and which role the most central actors play in this context. As bad coordination can be reflected by redundant code or hardly compatible packages, it is important that developers and users working in the same field stay in contact.

In this paper, we use content-based SNA, an approach to combine SNA and Text Mining (TM) in order to find out to which extent sharing interests is associated with communicating in two R mailing lists. Content-based SNA consists of extracting overlapping topic related subnetworks from the entire communication network. The paper shows how the au-

thors’ interests can be found based on their centralities in these topic-based subnetworks. By comparing the networks showing who shares interests with their communication networks, we find that communicating is the more associated with sharing interests the more central the authors are in the communication networks. We argue that this finding is due to the motives of authors to use the mailing lists.

Related work

There are several approaches to combine SNA and TM. One has to distinguish between combinations that enrich TM through SNA and those that enrich SNA through TM. One of the most prominent applications on the TM side is used by the search engines Google and Yahoo. They rank search results found by TM procedures according to centrality measures based on hyperlink graphs (Brin and Page, 1998; Kleinberg, 1999). Another very important TM plus SNA application is the summarization of texts by calculating centrality measures in word networks (networks where nodes represent words that are connected if they appear in the same text unit). The results are often visualized as tag clouds (Erkan and Radev, 2004; Vanderwende et al., 2004).

However, approaches to enrich an SNA with TM are not as numerous. McCallum et al. (2007) introduced the author-recipient-topic model which consists of fitting a multinomial distribution over topics and authors/recipients of a message simultaneously. The results are combinations of concepts and pairs of authors and recipients that characterize the network. By studying the Enron e-mail corpus, they defined social roles based on such combinations. For example the relationships of an Enron lawyer to a few other people were characterized by the concept “legal contracts”. A similar approach was applied to authors of Wikipedia articles by Chang et al. (2009).

The tradition this paper stands in is called “content-based SNA”. In content-based SNA, the network is partitioned into several overlapping subgraphs of people who discuss the same topic. Examples in the literature comprise Velardi et al. (2008), who analyzed the evolution of content-based communication networks in a company by measuring the agent-aggregation around topics, and Viermetz (2008), who calculated local actor centrality in content-based networks to find opinion leaders. To the best of our knowledge, content-based SNA has not been applied to mailing lists or to any kind of open source related data.

Data preparation

The data were taken from the mailing lists R-help and R-devel from January 2008 to May 2009. The e-mails can be downloaded as compressed text files from the R website (<https://stat.ethz.ch/pipermail/r-help/> and <https://stat.ethz.ch/pipermail/r-devel/>). There is one text file for each month. The R code for the entire data preparation process can be downloaded from R-Forge <https://r-forge.r-project.org/projects/snatum/>. First, the e-mails for each month were written into a single file using `as.one.file()` from the **snatum** package (example of the R-devel mailing list):

```
> as.one.file(files,
+   filename = "allthreads.txt",
+   list = "rdevel")
```

Then, the meta data (author, date, subject, thread-ID and e-mail-ID) and the e-mail content were extracted from these texts and transformed into a matrix.

```
> forest <- makeforest(month = "allthreads")
> head(forest)
```

	emailID	threadID
[1,]	"1"	"1"
[2,]	"2"	"2"
[3,]	"3"	"2"
[4,]	"4"	"3"
[5,]	"5"	"2"
[6,]	"6"	"3"
	author	
[1,]	"ggrothendieck at gmail.com (Gabor..."	
[2,]	"huber at ebi.ac.uk (Wolfgang Hube..."	
[3,]	"ripley at stats.ox.ac.uk (Prof Br..."	
[4,]	"dfaden at gmail.com (David Faden)"	
[5,]	"ripley at stats.ox.ac.uk (Prof Br..."	
[6,]	"ripley at stats.ox.ac.uk (Prof Br..."	
	subjects	
[1,]	"[Rd] Wish List"	
[2,]	"[Rd] Error from wilcox.test"	
[3,]	"[Rd] Error from wilcox.test"	
[4,]	"[Rd] setting the seed in standalo..."	
[5,]	"[Rd] Error from wilcox.test"	
[6,]	"[Rd] setting the seed in standalo..."	
	content	
[1,]	"In: trunk/src/library/base/ma..."	
[2,]	"On 1/3/2008 9:03 AM, Gabor Grothe..."	
[3,]	"What it is trying is % env R_DEF..."	
[4,]	"On 1/4/08, Prof Brian Ripley <rip..."	
[5,]	"Full_Name: Hendrik Weisser Versio..."	
[6,]	"hendrik.weisser at gmx.net wrote:..."	

The function `makeforest()` from **snatum** is based on the **tm.plugin.mail** package (Feinerer et al., 2008) which uses the Message-IDs and In-Reply-To-IDs contained in the e-mail headers to assign a thread-ID to each e-mail (`threads()` from the **tm.plugin.mail** package). The e-mail-IDs sort the e-mails according to the sequence in which they were sent. Answers to e-mails that were sent before January 2008 had to be removed because in this case the `threads()` function

cannot identify complete threads. Furthermore, citations and signatures were omitted from the e-mail content (`removeCitation()` and `removeSignature()` from the **tm.plugin.mail** package). The data contains 43760 R-help e-mails and 5008 R-devel e-mails.

Data preparation for TM

First, in order to reduce spelling ambiguities resulting from differences in American versus British English, the e-mail subjects and content were transformed into British English using the `ae-to-be()` function from the **snatum** package.

Second, the `wn.replace()` function from the **snatum** package was used to find groups of synonyms that were used in the texts based on the WordNet Database (Fellbaum, 1998; Wallace, 2007; Feinerer and Hornik, 2010). All terms in these groups of synonyms are replaced by one single representative of the group. However, in order to account for the R-specific vocabulary, not all of the synonym terms should be replaced. For example, the word "car" should not be replaced by "auto" where it "typically" refers to the **car** package. The `wn.replace()` function allows for manual deselection of certain terms, if they should not be replaced by a synonym.

Third, terms are stemmed using the Snowball stemmer `stemDocument()` from the **tm** package.

Fourth, term frequencies of the resulting terms (`termFreq()` from the **tm** package (Feinerer et al., 2008)) were calculated (separately for subjects and content). The function `tolower()` from the **base** package was applied and stopwords were ignored. Words of length less than three were omitted as the vast majority of them did not have a meaning in terms of abbreviations and the like, but were code chunks. Terms that were used less than 10 times (for subjects) or less than 20 times (for content) were ignored as well.

The four steps can be done all at once by the `prepare.text()` function from the **snatum** package.

Data preparation for SNA

To obtain a social network from the mailing list data, first, an alias detection procedure was performed on the Author-column of `forest`. It matched the different user names and e-mail addresses belonging to one and the same person by using the Levenshtein Distance (`agrep()` (Levenshtein, 1966)) and a few transformations inspired by Bird et al. (2006) that can be found in `find.aliases()` from the **snatum** package. This way, the number of unique authors was reduced from 6393 to 5972 in R-help and from 1049 to 983 in R-devel.

Second, the new `forest` was transformed into an edge list (`createedges()` from **snatum**). Somebody answering an e-mail was connected to all the authors who wrote something before (chronologically)

in the same thread as we assume that the respondent is aware of all the previous e-mails. After this, the edgelist was transformed into a matrix representation of a network (`adjacency()` from **snatm**).

```
> network <- adjacency(createedges(forest))
> network[1:2, 1:2]
```

	Gabor G.	Brian R.
Gabor G.	130	29
Brian R.	19	250

The resulting social network consists of nodes representing e-mail authors and directed valued lines indicating who answered whom and the number of mails sent in the corresponding direction. For example, Gabor Grothendieck answered on 29 of Brian Ripley's e-mails. If the diagonal is not zero the corresponding authors appeared several times in the same thread. (However, the diagonal will be ignored in the subsequent analysis.) We will call these networks "communication networks".

Combining network data and textual data

This section describes the crucial step where network data and textual data are combined. Three data preparation steps are needed.

First step

For all the subject terms that were used 10 times or more and all the content terms that appeared 20 times or more (using the prepared text corpora resulting from the `prepare.text()` function) a communication network was extracted that shows only the connections between authors who used this particular term. In a loop each term was assigned to `subjectfilter`, such that only connections between authors who used a certain term were extracted from `forest`.

```
> network <- adjacency(createedges(
+           forest,
+           subjectfilter = "lattice"))
```

As an example, Figure 1 shows the communication network of all the R-help authors who used the term "lattice" in the e-mail subject.

Deepayan Sarkar, who is the author of the **lattice** package (Sarkar, 2008), is clearly the most central person in this network. This indicates that he answered nearly everyone having a question about **lattice**.

Second step

This step is based on the idea that someone's interest for a certain subject can be measured by his or her centrality or activity in the corresponding communication network. For example, we would conclude

from Figure 1 that Deepayan Sarkar is very interested in the word "lattice". In SNA, there are several measures to calculate centrality, for example degree, closeness, betweenness and others. We chose to use the degree (number of direct neighbors; `degree` from the **sna** package (Butts, 2008)) because it measures activity while the others measure the connectedness to the entire network. The degree was calculated for all the nodes of networks that have more than 20 members. If a network has only 20 members or less, the results of centrality measures and thus the assumption of somebody being interested in a certain topic are considered not to be meaningful. Then, we created a network consisting of two sets of nodes (2-mode network), one representing e-mail authors and the other representing terms.

```
> twomode <- adjacency(centrality.edgelist(
+           terms, apply.to = "subjects"))
```

Each term and each author who used the term was connected by a line having the normalized degree centrality rank as line weight. (For instance the person with the highest degree in a certain communication network is connected to the corresponding word with a line weight of 1.) Thus, the line weight can be interpreted as the extent to which somebody is interested in a particular term in a [0,1] interval. Figure 2 shows the 0.45% strongest lines of R-help authors and subject-terms (largest connected subnetwork only). For example, Jim Lemon is connected to terms like "bar", "scatter", "axi(s)", "barplot" and "diagram", so he seems to be interested in plots.

Third step

In the third step, this 2-mode network was reduced to a 1-mode network by omitting the term nodes and connecting the author nodes that were adjacent to the term (`shrink` from the **snatm** package).

```
> shrink(twomode, keep = people,
+       values = "min")
```

For example, in the 1-mode network John Fox and Gabor Grothendieck are connected because they are both connected to "mean" in the 2-mode network. The networks have line weights representing the minimum of the line weights two people were formally connected with through the term node. For example, in the 2-mode network John Fox was connected to the term "mean" with a line weight of 0.9957 and Gabor Grothendieck was also connected to "graph" but with a line weight of 1. Then the two authors are connected with a line weight of 0.9957 in the 1-mode network, meaning that the extent to which they share interests is 0.9957 of 1. We will call these networks "interest networks".

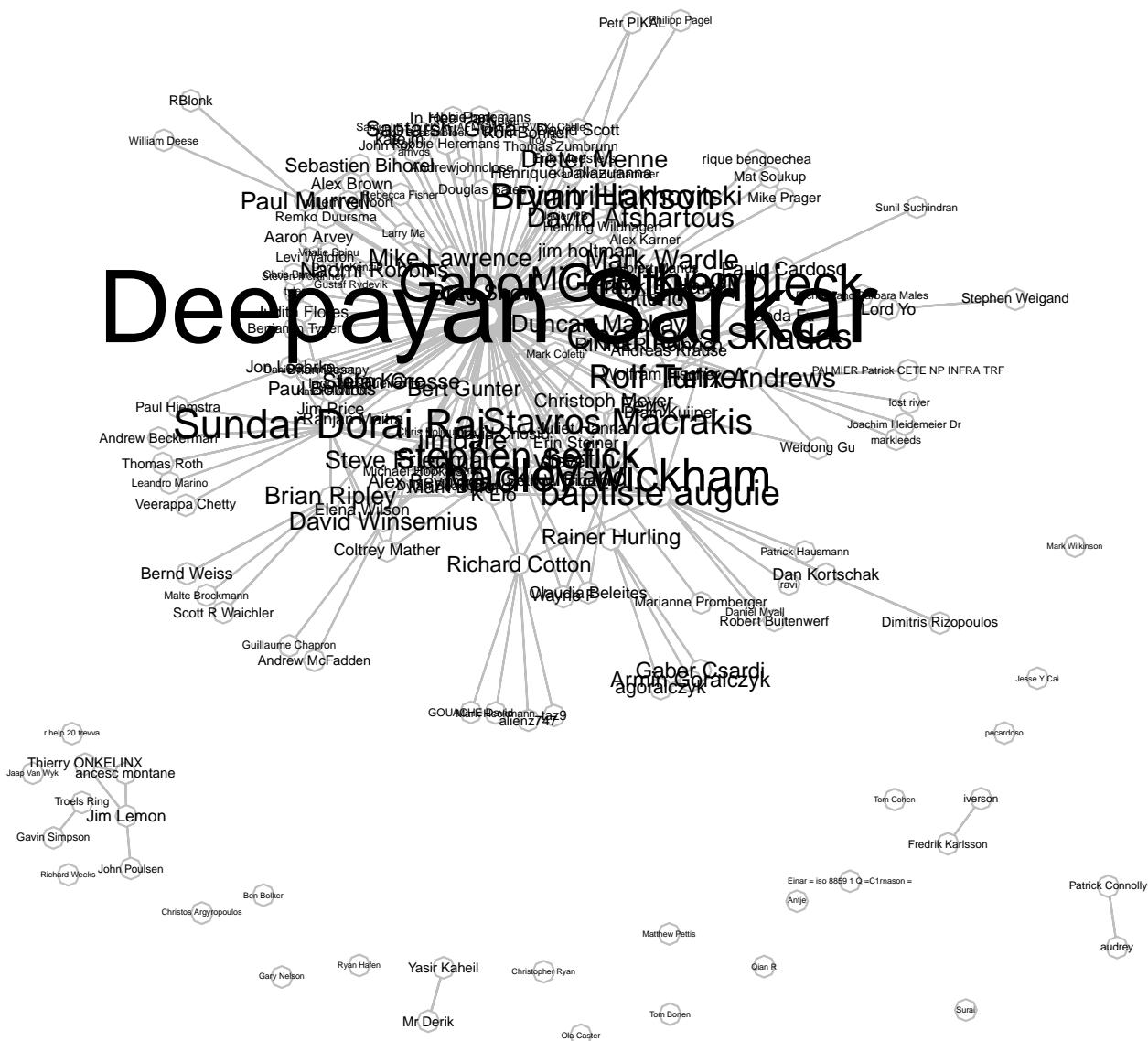


Figure 1: Communication network of all R-help authors who used “lattice” in the e-mail subject.

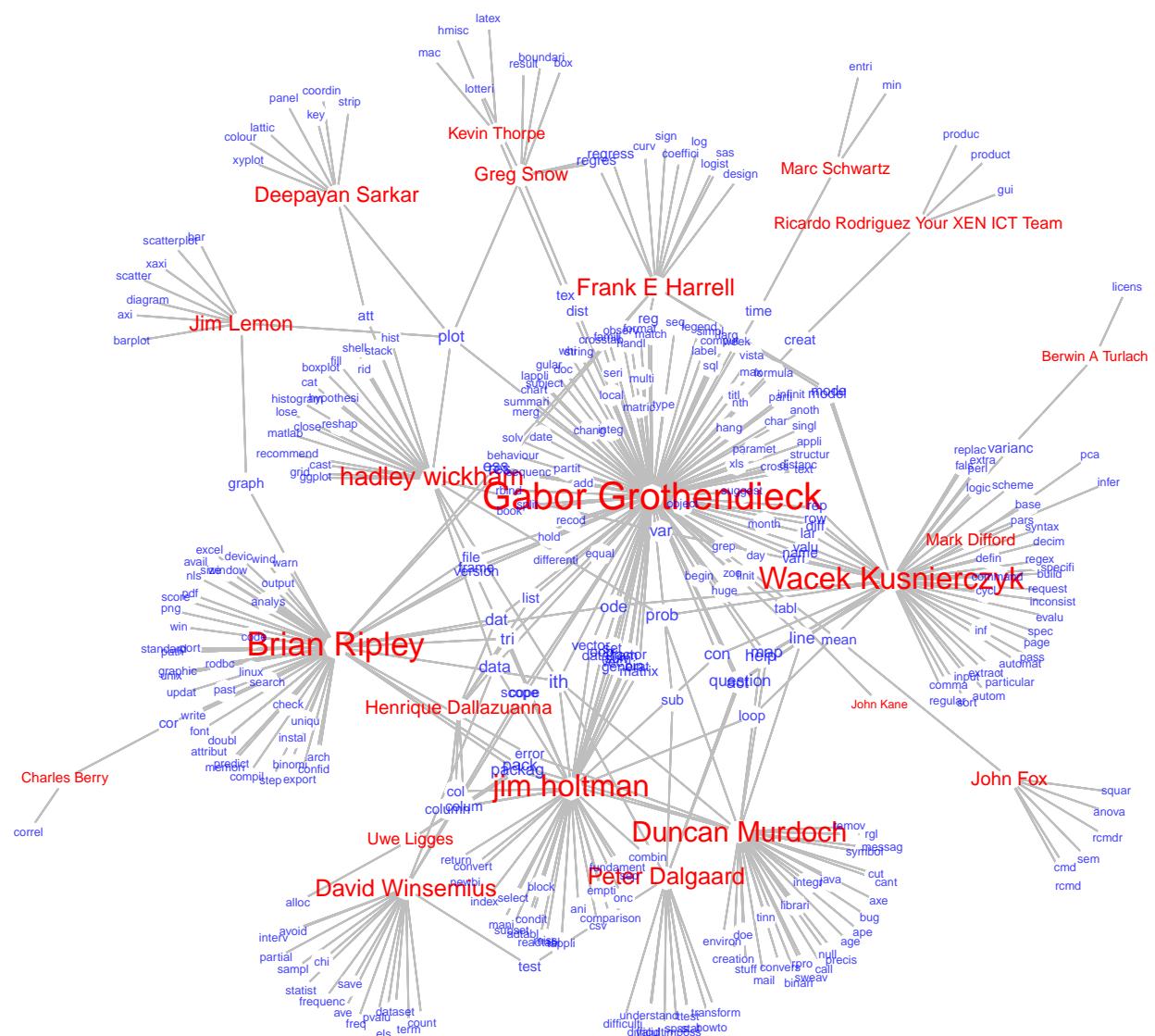


Figure 2: Author-term network showing the largest connected subnetwork of the 0.45% strongest connections between R-help authors and subject-terms.

Results: Comparison of communication networks and interest networks

At this point, the two kinds of networks, the communication networks and the interest networks can be compared. The communication networks have line weights representing the number of e-mails exchanged. The line weights of the interest networks indicate the extent to which two people have similar interests in a 0–1 interval. If we calculate the correlation between the line weights of both networks, we get the extent to which the fact of sharing interests is associated with communicating. We should expect that the more two people are interested in the same topic, the more e-mails they exchange.

There are two communication networks, one for R-help having 5972 nodes and one for R-devel having 983 nodes. Furthermore, there are four interest networks because the extent of shared interests was measured once by using the subjects and once by using the content for each of the mailing lists. The interest networks have fewer nodes because only the most frequently used terms were included into the interest analysis. Thus, people who only used less frequent terms do not appear in the interest networks. To compare the two kinds of networks the communication networks were reduced to those nodes who also appeared in the interest networks. Furthermore, the reduced communication network was permuted (permutation() from the **sna** package) such that the sequence of author names assigned to rownames(network) is the same in both networks. However, the correlation between sharing interests and communicating is only approximately 0 for all possible combinations of communication networks and interest networks (Table 1). (The diagonals of the communication networks were set to 0.)

Interest network	Communication network	
	R-help	R-devel
subjects	0.030	0.070
content	0.009	0.044

Table 1: Correlations between all combinations of communication networks and interest networks.

However, if the centrality of authors in the communication network is taken into account, we get a different picture. Figure 3 shows how the correlation between intensity of communication and extent of sharing interests changes (y-axis) as the normalized centrality (degree, betweenness and closeness) of nodes increases (x-axis). (Degree and closeness were calculated with the **sna** package (Butts, 2008), betweenness with

the **igraph** package (Csardi and Nepusz, 2006).)

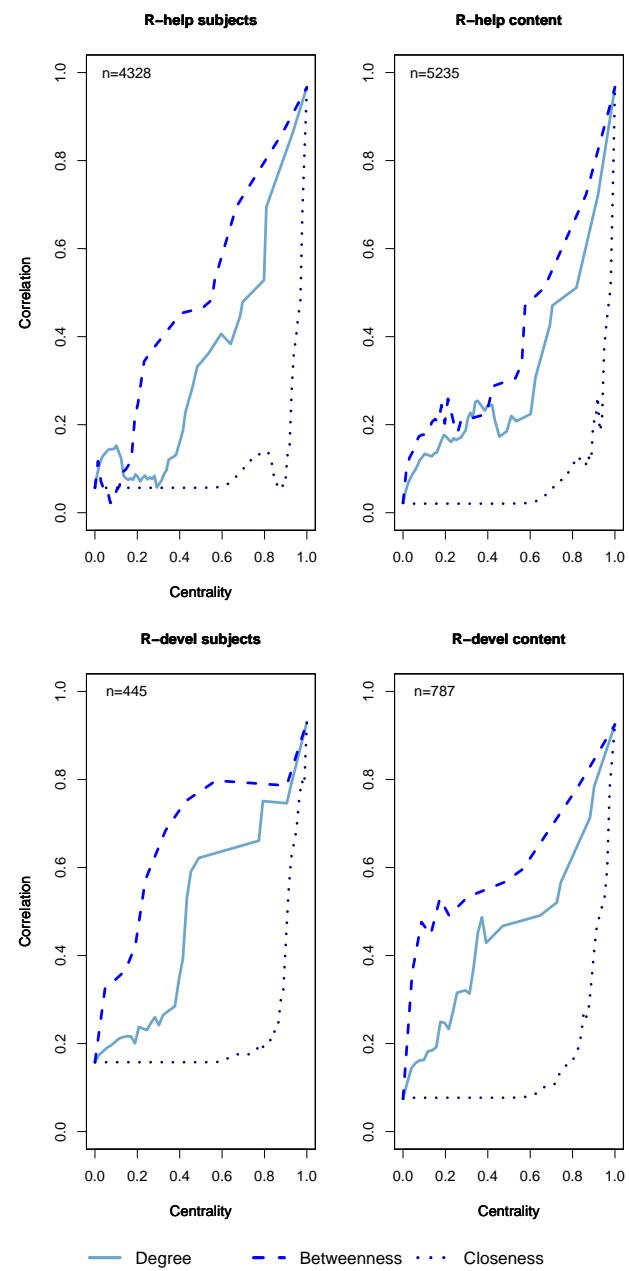


Figure 3: Correlation between intensity of communication and sharing interests (y-axis) and degree centrality of nodes (x-axis).

More precisely, it shows the correlations between communication networks and interest networks consisting of only those authors who have a higher or equally high centrality than indicated on the x-axis.

There is an almost linear increase in correlation in the R-help subjects network for the centrality measures degree and betweenness, whose distributions are highly right skewed. Closeness is approximately normally distributed. Thus, networks of people having an average closeness (around 0.5) are not correlated just as networks of authors with an average normalized degree (around 0). In the R-help

content network, a relation between the centrality of authors and the correlation between sharing interests and communicating begins only at a centrality of around 0.5 for right skewed measures and around 0.95 for closeness. This is a result of the discussion topics sometimes changing in the course of threads. In the R-devel networks, the relation is also positive but not as stable, which is due to the much smaller network sizes n of 445 (subjects) and 787 (content) compared to 4328 (subjects) and 5235 (content) in the R-help networks. The correlation in R-devel is generally higher than in R-help. In both, the R-help (for high centralities) and the R-devel networks, the lines are smoother when the subjects were used to find people's interests. This might be due to some additional chatting in the content while the subjects are more focused on the actual topics. However, the choice of subjects or content does not influence the general finding that sharing interests and communicating is the more associated the higher people's centrality scores are. This might be explained by impartiality or indifference of highly active authors towards the personality of their e-mail partners. They rather concentrate on the topics. Figure 4 supports this interpretation:

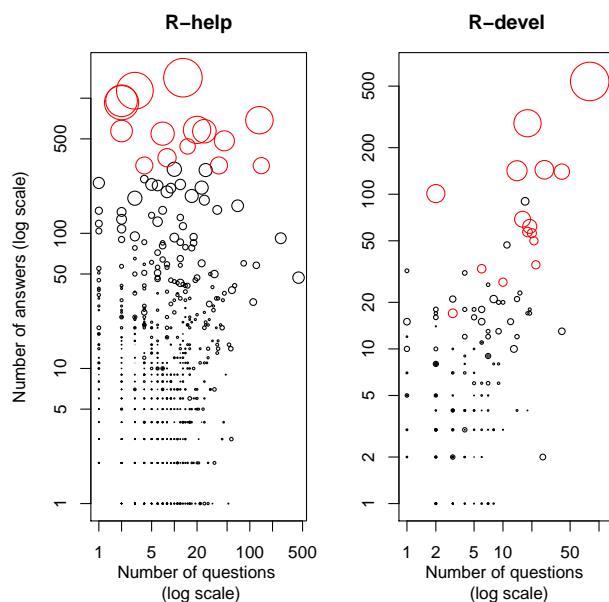


Figure 4: Scatterplot showing the number of questions (x-axis) and answers (y-axis) as well as the degree centrality (dot size) of each author.

Each point represents one author and shows how many questions (x-axis) and answers (y-axis) he or she wrote (`ans.quest()` from the **snatm** package). The larger a point the higher the author's degree centrality. The plots indicate that in R-help the 15 most active authors (red) write far more answers than questions. All but one author are developers of CRAN packages, which suggests that their motivation is to provide help concerning their packages or specific topics, no matter who asked the questions. In

contrary, less central R-help authors are either more choosy with the choice of e-mail partners or they are not active enough to answer everyone having similar interests. In R-devel the proportion of answers to questions of central authors (red) is not different from less central authors. Still, Figure 3 suggests that there is a great correspondency between communicating and sharing interests among the most active authors. The low correlation between intensity of communication and extent of sharing interests for peripheral nodes in both mailing lists can be due to their very low activity which hinders the definition of their interests.

Summary

The paper showed how content-based SNA can be used to find people's interests in mailing list networks. By comparing communication graphs and networks showing who has similar interests, a relationship between the correlation of these two and node centrality could be found. Accordingly, the expected relationship between sharing interests and communicating exists only for very active authors while less active authors do not answer everyone who has similar interests. Thus, the communication efficiency can be regarded to be high for very active mailing list authors while it is moderate for mid-active authors. Additionally, the paper suggests using only the subjects to find the relationship between communicating and sharing interests because the content contains more noise.

Code for figures 1 and 2

```
> # Figure 1
> gplot.snatm(network, vertex.col = "white",
+   vertex.border = "grey",
+   label = rownames(network),
+   boxed.labels = FALSE,
+   label.pos = 5,
+   label.cex =
+     ((sna::degree(network)+1)^0.45)*0.25,
+   gmode = "graph", edge.lwd = 0.1,
+   edge.col = "grey")
> # Figure 2
> # See how to get peoplelist in the snatm demo.
> people <- which(is.element(rownames(twomode)
+   ,unique(peoplelist)))
> labelcol <- rep(rgb(0,0,1,0.75),dim(twomode)[1])
> labelcol[people] <- "red"
> gplot.snatm(twomode, gmode = "graph",
+   vertex.col = "white", vertex.cex = 1,
+   label = rownames(twomode),
+   label.col = labelcol,
+   label.cex =
+     (sna::degree(twomode)^0.25)*0.35,
+   label.pos = 5, edge.lwd = 0.1,
+   boxed.labels = FALSE,
+   vertex.border = "white",edge.col = "grey")
```

Bibliography

- C. Bird, A. Gourley, P. Devanbu, M. Gertz, and A. Swaminathan. Mining email social networks. In *Proceedings of the 2006 International Workshop on Mining Software Repositories, Shanghai, China*, pages 137–143, 2006. URL <http://macbeth.cs.ucdavis.edu/msr06.pdf>.
- S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998. URL <http://wx.bsjh.tcc.edu.tw/~t2003013/wiki/images/8/8f/Anatomy.pdf>.
- C. T. Butts. Social network analysis with sna. *Journal of Statistical Software*, 24(6), 2008. URL <http://www.jstatsoft.org/v24/i06/>.
- J. Chang, J. B. Gruber, and D. M. Blei. Connections between the lines: Augmenting social networks with text. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178, New York, NY, USA, 2009. ACM. URL <http://www.cs.princeton.edu/~jbg/documents/kdd2009.pdf>.
- G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal, Complex Systems*:1695, 2006. URL <http://igraph.sf.net>.
- G. Erkan and D. R. Radev. Lexrank: Graph-based centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 2004. URL <http://tangra.si.umich.edu/~radev/lexrank/lexrank.pdf>.
- I. Feinerer and K. Hornik. wordnet: WordNet Interface. *R package version 0.1-6.*, 2010. URL <http://CRAN.R-project.org/package=wordnet>.
- I. Feinerer, K. Hornik, and D. Meyer. Text mining infrastructure in R. *Journal of Statistical Software*, 25(5):1–54, 2008. ISSN 1548-7660. URL <http://www.jstatsoft.org/v25/i05>.
- C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.
- J. M. Kleinberg. Authoritative sources in a hyper-linked environment. *Journal of the ACM*, 46(5):604–632, 1999. URL <http://www.cs.cornell.edu/home/kleinber/auth.pdf>.
- V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, Soviet Physics Doklady, 1966.
- A. McCallum, X. Wang, and A. Corrada-Emmanuel. Topic and role discovery in social networks with experiments on Enron and academic email. *Journal of Artificial Intelligence Research*, 30:249–272, 2007. URL <http://www.cs.umass.edu/~mccallum/papers/art-ijcai05.pdf>.
- D. Sarkar. *Lattice: Multivariate Data Visualization with R*. Springer, 2008.
- L. Vanderwende, M. Banko, and A. Menezes. Event-centric summary generation. In *In Working Notes of DUC 2004*, 2004. URL <http://www-nlpir.nist.gov/projects/duc/pubs/2004papers/microsoft.banko.pdf>.
- P. Velardi, R. Navigli, A. Cucchiarelli, and F. D’Antonio. A new content-based model for social network analysis. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing*, pages 18–25, Washington, DC, USA, 2008. IEEE Computer Society. URL <http://www.dsi.uniromal.it/~velardi/ICSCpubblicato.pdf>.
- M. Viermetz. *Partitioning Massive Graphs for Content Oriented Social Network Analysis*. PhD thesis, Heinrich-Heine-Universität Düsseldorf, June 2008. URL http://docserv.uni-duesseldorf.de/servlets/DerivateServlet/Derivate-8825/diss_viermetz_pdfalb.pdf.
- M. Wallace. *Jawbone Java WordNet API*, 2007. URL <http://mfwallace.googlepages.com/jawbone.html>.
- Angela Bohn, Kurt Hornik, Patrick Mair
Institute for Statistics and Mathematics
Department of Finance, Accounting and Statistics
Wirtschaftsuniversität Wien
Augasse 2–6
A-1090 Wien
Austria
angela.bohn@gmail.com
kurt.hornik@wu-wien.ac.at
patrick.mair@wu.ac.at*
- Ingo Feinerer
Database and Artificial Intelligence Group
Institute of Information Systems
Technische Universität Wien
Favoritenstrasse 9
A-1040 Wien
Austria
feinerer@logic.at*

Rmetrics - timeDate Package

by Yohan Chalabi, Martin Mächler, and Diethelm Würtz

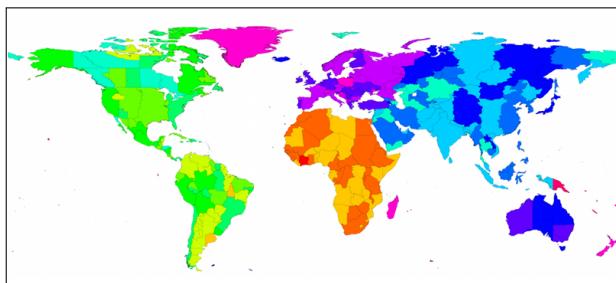


Figure 1: World map with major time zones.¹

Abstract The management of time and holidays can prove crucial in applications that rely on historical data. A typical example is the aggregation of a data set recorded in different time zones and under different daylight saving time rules. Besides the time zone conversion function, which is well supported by default classes in R, one might need functions to handle special days or holidays. In this respect, the package **timeDate** enhances default date-time classes in R and brings new functionalities to time zone management and the creation of holiday calendars.

Chronological data sets recorded in different time zones play an important role in industrial applications. For example, in financial applications, it is common to aggregate time series that have been recorded in financial centers with different daylight saving time (DST) rules and time zones.

R includes different classes to represent dates and time. The class that holds particular interest for us is the "POSIXct" one. It internally records timestamps as the number of seconds from "1970-01-01 UTC", where UTC stands for universal time coordinated. Moreover, it supports the DST and time zone functions by using the rules provided by the operating system (OS). However, at the time **timeDate** (Würtz et al. (2011))—formerly known as **fCalendar**—was first released, the implementation of the DST function was not consistent across OSs. Back then, the main purpose of the package was to have DST rules directly available to bring consistency over OSs. Today, DST support by OSs is not a problematic question as it used to be. As we will show later, the "timeDate" class is based on the "POSIXct" one. Both classes hence share common functionalities. However, the **timeDate** package has some additional functionalities, which we will emphasize in this note. For related date-time classes in

R, see Ripley & Hornik (2001) and Grothendieck & Petzoldt (2004).

Another problem commonly faced in managing time and dates is the midnight standard. Indeed, the problem can be handled in different ways depending on the standard in use. For example, the standard C library does not allow for the "midnight" standard in the format "24:00:00" (Bateman (2000)). However, the **timeDate** package supports this format.

Moreover, **timeDate** includes functions for calendar manipulations, business days, weekends, and public and ecclesiastical holidays. One can handle day count conventions and rolling business conventions. Such periods can pertain to, for example, the last working day of the month. The below examples illustrate this point.

In the remaining part of this note, we first present the structure of the "timeDate" class. Then, we explain the creation of "timeDate" objects. The use of financial centers with DST rules is described next. Thereafter, we explain the management of holidays. Finally, operations on "timeDate" objects such as mathematical operations, rounding, subsetting, and coercions are discussed. Throughout this note, we provide many examples to illustrate the functionalities of the package.

Structure of the "timeDate" class

The "timeDate" S4 class is composed of a "POSIXct" object that is always in Greenwich Mean Time (GMT) and of a financial center that keeps track of DST. We use the term *financial center* to denote geographical locations. This terminology stems from the main application of the Rmetrics packages. However, this denomination should not stop programmers from using the package in other fields. By default, the local financial center is set to GMT. The default setting can be changed via `setRmetricsOptions(myFinCenter =)`. The formal S4 class is defined as

```
> showClass("timeDate")
Class "timeDate" [package "timeDate"]
```

Slots:

Name:	Data	format	FinCenter
Class:	POSIXct	character	character

where the slot `Data` contains the timestamps in the `POSIXct` class, `format` is the format typically applied to `Data`, and `FinCenter` is the financial center.

Note: we use the abbreviation `GMT` equivalently to `UTC`, the universal time coordinated.

¹The original data set of the world map with time zones is available at <http://efele.net/maps/tz/world/>. Full and reduced `rda` versions were kindly contributed by Roger Bivand.

"timeDate" object creation

There are different ways to generate a "timeDate" object. It can be generated using either `timeDate()`, `timeSequence()`, or `timeCalendar()`.

The function `timeDate()` creates a "timeDate" object from scratch. It requires a character vector of timestamps and optional arguments to specify the format of this vector and the financial center. The financial center can be specified, as mentioned, via `setRmetricsOptions()` or (more cleanly) with the argument `FinCenter`. By default, it is set to GMT².

In the following, we illustrate the creation of "timeDate" objects as well as the method used to convert timestamps from different time zones.

We first create character vectors of two timestamps with the default financial center (GMT):

```
> Dates <- c("2009-09-28", "2010-01-15")
> Times <- c( "23:12:55", "10:34:02")
> charvec <- paste(Dates, Times)
> getRmetricsOption("myFinCenter")
myFinCenter
    "GMT"
> timeDate(charvec)
GMT
[1] [2009-09-28 23:12:55] [2010-01-15 10:34:02]
```

As a second step, we set the local financial center to Zurich and create a "timeDate" object.

```
> setRmetricsOptions(myFinCenter = "Zurich")
> timeDate(charvec)
Zurich
[1] [2009-09-28 23:12:55] [2010-01-15 10:34:02]
```

The third example shows how the timestamps can be conveniently converted into different time zones; `charvec` is recorded in Tokyo and subsequently converted to our local center, i.e., Zurich (see above):

```
> timeDate(charvec, zone = "Tokyo")
Zurich
[1] [2009-09-28 16:12:55] [2010-01-15 02:34:02]
```

or converted from Zurich to New York:

```
> timeDate(charvec, zone = "Zurich",
+           FinCenter = "NewYork")
NewYork
[1] [2009-09-28 17:12:55] [2010-01-15 04:34:02]
```

It is also possible to use the function `finCenter()` to view or modify the local center:

```
> td <- timeDate(charvec, zone = "Zurich",
+                  FinCenter = "NewYork")
> finCenter(td)
[1] "NewYork"
```

```
> finCenter(td) <- "Zurich"
> td
Zurich
[1] [2009-09-28 23:12:55] [2010-01-15 10:34:02]
```

If the format of `charvec` is not specified, `timeDate` uses an automated date-time format identifier called `whichFormat()` that supports common date-time formats.

```
> whichFormat(charvec)
[1] "%Y-%m-%d %H:%M:%S"
```

The function `timeSequence()` creates a "timeDate" object representing an equidistant sequence of points in time. You can specify the range of dates with the arguments `from` and `to`. If `from` is missing, `length.out` defines the length of the sequence. In the case of a monthly sequence, you can define specific rules. For example, you can generate the sequence with the last days of the month or with the last or `n`-th Friday of every month. This can be of particular interest in financial applications.

Let us first reset the financial center to an international environment:

```
> setRmetricsOptions(myFinCenter = "GMT")
> # 'timeDate' is now in the financial center "GMT"
> timeDate(charvec)
GMT
[1] [2009-09-28 23:12:55] [2010-01-15 10:34:02]
```

A sequence of days or months can be created as follows:

```
> # first three days in January 2010,
> timeSequence(from = "2010-01-01",
+               to = "2010-01-03", by = "day")
GMT
[1] [2010-01-01] [2010-01-02] [2010-01-03]
> # first 3 months in 2010:
> timeSequence(from = "2010-01-01",
+               to = "2010-03-31", by = "month")
GMT
[1] [2010-01-01] [2010-02-01] [2010-03-01]
```

The function `timeCalendar()` creates "timeDate" objects from calendar atoms. You can specify values or vectors of equal length denoting year, month, day, hour, minute, and seconds as integers. For example, the monthly calendar of the current year or a specific calendar in a given time zone can be created as follows:

```
> timeCalendar()
GMT
[1] [2011-01-01] [2011-02-01] [2011-03-01]
[4] [2011-04-01] [2011-05-01] [2011-06-01]
[7] [2011-07-01] [2011-08-01] [2011-09-01]
[10] [2011-10-01] [2011-11-01] [2011-12-01]
```

²GMT can be considered as a "virtual" financial center.

The following represents the first four days of January recorded in Tokyo at local time “16:00” and converted to the financial center Zurich:

```
> timeCalendar(2010, m=1, d=1:4, h=16,
+               zone = "Tokyo", FinCenter = "Zurich")
Zurich
[1] [2010-01-01 08:00:00] [2010-01-02 08:00:00]
[3] [2010-01-03 08:00:00] [2010-01-04 08:00:00]
```

Midnight standard

The “timeDate” printing format is designed in accordance with the ISO-8601 (1988) standard. It uses the 24-hour clock format. Dates are expressed in the “%Y-%m-%d” format while time-dates are stated in the “%Y-%m-%d %H:%M:%S” format. A special case in the 24-hour clock system is the representation of midnight. It can be equivalently represented by “00:00” and “24:00”. The former is usually used to denote the beginning of the day whereas the latter denotes the end. `timeDate` supports the midnight standard as described by the ISO-8601 (1988) standard as illustrated here:

```
> timeDate(ch <- "2010-01-31 24:00:00")
GMT
[1] [2010-02-01]
```

Note, following the revision of this paper, the R source has been amended in May 2011 to also allow “24:00” time specifications, such that a midnight standard will be part of standard R.

Financial centers – via daylight saving time rules

As mentioned earlier, the global financial center can be set with the function `setRmetricsOptions()` and accessed with the function `getRmetricsOption()`. Its default value is set to “GMT”:

```
> getRmetricsOption("myFinCenter")
myFinCenter
  "GMT"
> # change to Zurich:
> setRmetricsOptions(myFinCenter = "Zurich")
```

From now on, all dates and times are handled in accordance with the Central European time zone and the DST rule for Zurich. Note that setting the financial center to a continent/city that lies outside the time zone used by your OS does not change any of the time settings or environment variables of your computer.

There are almost 400 financial centers supported thanks to the Olson database. They can be accessed by the function `listFinCenter()`, and partial lists can be extracted through the use of regular expressions.

```
> # first few financial centers:
> head(listFinCenter())
[1] "Africa/Abidjan"      "Africa/Accra"
[3] "Africa/Addis_Ababa"  "Africa/Algiers"
[5] "Africa/Asmara"       "Africa/Bamako"
> # European centers starting with A or B:
> listFinCenter("Europe/[AB].*") # -> nine
[1] "Europe/Amsterdam"    "Europe/Andorra"
[3] "Europe/Athens"       "Europe/Belgrade"
[5] "Europe/Berlin"        "Europe/Bratislava"
[7] "Europe/Brussels"     "Europe/Bucharest"
[9] "Europe/Budapest"
```

Each financial center has an associated function that returns its DST rules in the form of a “`data.frame`”. These functions share the name of their financial center, e.g., `Zurich()`.

```
> Zurich() [64:67, ]
   Zurich offSet isdst TimeZone
64 2010-03-28 01:00:00  7200   1   CEST
65 2010-10-31 01:00:00  3600   0   CET
66 2011-03-27 01:00:00  7200   1   CEST
67 2011-10-30 01:00:00  3600   0   CET
      numeric
64 1269738000
65 1288486800
66 1301187600
67 1319936400
```

The returned “`data.frame`” shows when the clock was changed in Zurich, the offset in seconds with respect to GMT, a flag that tells us if DST is in effect or not, the time zone abbreviation, and the number of seconds since “1970-01-01” in GMT. The reader interested in the history of DST is referred to Bartky & Harrison (1979).

Note new centers can be easily added as long as their associated functions return a “`data.frame`” with the same structure as described above.

Holidays and calendars

Holidays are usually grouped by their origins. The first category, as the etymology suggests, is based on religious origin. For example, the ecclesiastical calendars of Christian churches are based on cycles of movable and immovable feasts. Christmas, December 25, is the principal immovable feast, whereas Easter is the principal movable feast. Most of the other dates are movable feasts that are determined with respect to Easter, Montes (1996). A second category of holidays is secular holidays, which denotes days that are celebrated internationally and in different cultures, such as Labor Day. Another category of holidays includes days that are relative to natural events. For example, the dates can be related to astronomical events such as cycles of the moon or the equinox. Moreover, there are also country-specific national holidays.

The calculation of holidays might prove tedious in some circumstances. Indeed, the estimation of the Easter date is a complex procedure with different algorithms involved in its computation. The algorithm implemented in the package is the one of Oudin (1940) as quoted in Seidelmann (1992). This approach is valid for any Gregorian calendar year. Further details about holiday calculation can be found in Tøndering (2008).

The dates of Easter for the next five years can be calculated with

```
> thisYear <- getRmetricsOption("currentYear")
> Easter(thisYear:(thisYear+5))
Zurich
[1] [2011-04-24] [2012-04-08] [2013-03-31]
[4] [2014-04-20] [2015-04-05] [2016-03-27]
```

The **timeDate** package includes functions for bank holidays in Canada, France, Germany, Great Britain, Italy, Japan³, Switzerland, and the US. These holidays can be grouped in calendars. At the moment, the package provides functions for the New York stock exchange, `holidayNYSE()`; for the North American Reliability Council, `holidayNERC()`⁴; for the Toronto stock exchange `holidayTSX()`, and for Zurich, `holidayZURICH()`. Other calendars can be easily implemented given that the package already provides many holidays functions. A list of all holidays is provided in the appendix.

Logical test

It is possible to construct tests for weekdays, weekends, business days, and holidays with the functions `isWeekday()`, `isWeekend()`, `isBizday()` and `isHoliday()`, respectively.

Let us take a sequence of dates around Easter:

```
> Easter(2010)
Zurich
[1] [2010-04-04]
> (ts <- timeSequence(Easter(2010, -2),
+                      Easter(2010, +3)))
Zurich
[1] [2010-04-02] [2010-04-03] [2010-04-04]
[4] [2010-04-05] [2010-04-06] [2010-04-07]
```

We can now extract the weekdays or business days according to a holiday calendar.

```
> (ts1 <- ts[isWeekday(ts)])
Zurich
[1] [2010-04-02] [2010-04-05] [2010-04-06]
[4] [2010-04-07]
> (ts2 <- ts[isBizday(ts, holidayZURICH(2010))])
Zurich
[1] [2010-04-06] [2010-04-07]
> dayOfWeek(ts2)
```

2010-04-06 2010-04-07

"Tue" "Wed"

Thank to the comments of one of the referees, we have added a new argument, `wday`, in the functions `isWeekend()`, `isWeekday()`, `isBizday()` and `isHoliday()` that can be used to specify which days should be considered as business days. This is important when using calendars in Islamic countries or in Israel. By default, `wday` specifies the weekdays as Monday to Friday.

Special dates

As mentioned earlier, holidays often refer to a specific date or event. It is therefore crucial to have functions to compute: the first day in a given month, the last day in a given month and year, a given day before or after a date, the n-th occurrences of a day in a specified year/month, or a given last day for a specified year/month. We have summarized these functions in a table in the appendix.

In the following, we demonstrate how to retrieve the last day in each quarter or the second Sunday of each month. Note that days are numbered from 0 to 6 where 0 corresponds to Sunday and 6 to Saturday.

```
> charvec <- c("2011-03-01", "2011-04-01")
> # Last day in quarter
> timeLastDayInQuarter(charvec)
Zurich
[1] [2011-03-31] [2011-06-30]
> # Second Sunday of each month:
> timeNthNdayInMonth(charvec, nday = 0, nth = 2)
Zurich
[1] [2011-03-13] [2011-04-10]
> # Closest Friday that occurred before:
> timeNdayOnOrBefore(charvec, nday = 5)
Zurich
[1] [2011-02-25] [2011-04-01]
```

Operations on "timeDate" objects

Just like the other date-time classes in R, the "timeDate" class supports common operations. It allows for mathematical operations such as addition, subtraction, and comparisons to be performed. Moreover, methods for the generic functions to concatenate, replicate, sort, re-sample, unify, revert, or lag are available as the well known calls `c()`, `rep()`, `sort()`, `sample()`, `unique()`, `rev()`, and `diff()`, respectively. We spare the reader superfluous examples of functions that are common to other date-time classes. In the rest of the section, we emphasize methods that are not available for other classes or are not strictly identical. The reader is referred to the ebook "Chronological Objects with Rmetrics" (Würtz, Chalabi & Ellis, 2010) for more examples.

³The Japanese holidays were contributed by Parlamic Franklin.
⁴`holidayNERC()` was contributed by Joe W. Byers.

Subsetting methods

The **timeDate** package has different functions to subset a `timeDate` object. The usual function `'['` extracts or replaces subsets of "`timeDate`" objects as expected. However, the package provides some additional functions. For example, the function `window()` extracts a sequence from a starting and ending point. The functions `start()` and `end()` extract the first and last timestamps, respectively.

Coercion methods

The package provides both S3 and S4 methods to coerce to and from "`timeDate`" objects. Below, we list the S4 coercion methods available. The equivalent S3 methods `as.*` are also provided, although the mixing of S4 classes and S3 methods is discouraged. Note that information can be lost in the coercion process if the destination class does not support the same functionality.

```
> showMethods("coerce", class = "timeDate")
Function: coerce (package methods)
from="ANY", to="timeDate"
from="Date", to="timeDate"
from="POSIXt", to="timeDate"
from="timeDate", to="Date"
from="timeDate", to="POSIXct"
from="timeDate", to="POSIXlt"
from="timeDate", to="character"
from="timeDate", to="data.frame"
from="timeDate", to="list"
from="timeDate", to="numeric"
```

We would like to point out a particular difference between the `as.numeric` methods of "`timeDate`" and "`POSIXct`" classes. Indeed, the `as.numeric-timeDate` method returns the time in minutes, in contrast to `as.numeric.POSIXct`, which returns the results in seconds. However, the `as.numeric-timeDate` method has an additional argument, `unit`, to select other units. These include seconds, hours, days, and weeks.

Concatenation method

The concatenation `c()` method for "`timeDate`" objects takes care of the different financial centers of the object to be concatenated. In such cases, all timestamps are transformed to the financial center of the first "`timeDate`" object. This feature is now also supported by R's "`POSIXct`" class. However, it was not available in previous versions of the class.

```
> ZH <- timeDate("2010-01-01 16:00", zone = "GMT",
+                  FinCenter = "Zurich")
> NY <- timeDate("2010-01-01 18:00", zone = "GMT",
+                  FinCenter = "NewYork")
> c(ZH, NY)
Zurich
[1] [2010-01-01 17:00:00] [2010-01-01 19:00:00]
```

```
> c(NY, ZH)
NewYork
[1] [2010-01-01 13:00:00] [2010-01-01 11:00:00]
```

Rounding and truncation methods

The rounding and truncation methods have similar functionalities to those of their counterparts in other date-time classes. However, the default `unit` argument is not the same.

Summary

The **timeDate** package offers functions for calendar manipulations for business days, weekends, and public and ecclesiastical holidays that are of interest in financial applications as well as in other fields. Moreover, the *Financial Center* concept facilitates the mixing of data collected in different time zones and the manipulation of data recorded in the same time zone but with different DST rules.

Acknowledgments

We thank the anonymous referees and the editor for their valuable comments as well as all R users and developers who have helped us to improve the **timeDate** package.

Bibliography

- R.I. Bartky and E. Harrison. Standard and Daylight Saving Time. *Scientific American*, 240:46–53, 1979.
- R. Bateman. Time Functionality in the Standard C Library, Novell AppNotes, September Issue, 73–85, 2000.
- N. Dershowitz and E.M. Reingold. Calendrical Calculations. *Software - Practice and Experience*, 20:899–928, 1990.
- N. Dershowitz and E.M. Reingold. *Calendrical Calculations*. Cambridge University Press, 1997.
- G. Grothendieck and T. Petzoldt. Date and Time Classes in R. *R News*, 4(1):29–32, 2004.
- ISO-8601. *Data Elements and Interchange Formats – Information Interchange, Representation of Dates and Time*. International Organization for Standardization, Reference Number ISO 8601, 1988.
- M.J. Montes. Butcher's Algorithm for Calculating the Date of Easter in the Gregorian Calendar, 1996.
- B.D. Ripley and K. Hornik. Date-Time Classes. *R-News*, 1(2):8–12, 2001.

P.K. Seidelmann (Editor). *Explanatory Supplement to the Astronomical Almanac*. University Science Books, Herndon, 1992.

C. Tøndering. Frequently Asked Questions about Calendars, 2008. URL <http://www.tondering.dk/claus/calendar.html>.

D. Würz and Y. Chalabi with contribution from M. Maechler, J.W. Byers, and others. *timeDate: Rmetrics - Chronological and Calendarical Objects*, 2011. URL <http://cran.r-project.org/web/packages/timeDate/>.

D. Würz, Y. Chalabi and A. Ellis. *Chronological Objects with Rmetrics*, 2010. URL <http://www.rmetrics.org/ebooks>.

*Yohan Chalabi
Finance Online GmbH, Zurich &
Institute of Theoretical Physics, ETH Zurich,
Switzerland
chalabi@phys.ethz.ch*

*Martin Mächler
Seminar für Statistik, ETH Zurich,
Switzerland
maechler@stat.math.ethz.ch*

*Diethelm Würz
Institute of Theoretical Physics, ETH Zurich,
Switzerland
wuertz@phys.ethz.ch*

Appendix

Session Info

```
> toLatex(sessionInfo())
```

- R version 2.13.0 (2011-04-13),
x86_64-apple-darwin10.7.0
- Locale: C ...
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: timeDate 2130.93

Tables

1. Special dates

timeFirstDayInMonth	First day in a given month
timeLastDayInMonth	Last day in a given month
timeFirstDayInQuarter	First day in a given quarter
timeLastDayInQuarter	Last day in a given quarter
timeNdayOnOrAfter	Day "on-or-after" n-days
timeNdayOnOrBefore	Day "on-or-before" n-days
timeNthNdayInMonth	N-th occurrence of a n-day in month
timeLastNdayInMonth	Last n-day in month

2. List of holidays

Advent1st	JPBunkaNoHi
Advent2nd	JPChildrensDay
Advent3rd	JPComingOfAgeDay
Advent4th	JPConstitutionDay
AllSaints	JPEmperorsBirthday
AllSouls	JPGantan
Annunciation	JPGreeneryDay
Ascension	JPHealthandSportsDay
AshWednesday	JPKeirouNoHi
AssumptionOfMary	JPKenkokuKinenNoHi
BirthOfVirginMary	JPKenpouKinenBi
BoxingDay	JPKinrouKanshaNoHi
CACanadaDay	JPKodomoNoHi
CACivicProvincialHoliday	JPKokuminNoKyujitu
CALabourDay	JPMarineDay
CATHanksgivingDay	JPMidoriNoHi
CAVictoriaDay	JPNatFoundationDay
CHAscension	JPNationHoliday
CHBerchtoldsDay	JPNationalCultureDay
CHConfederationDay	JPNNewYearsDay
CHKnabenschiessen	JPRespectForTheAgedDay
CHSechselaeuten	JPSeijinNoHi
CaRemembranceDay	JPShuubunNoHi
CelebrationOfHolyCross	JPTaiikuNoHi
ChristTheKing	JPTennouTanjyouBi
ChristmasDay	JPTurkeysgivingDay
ChristmasEve	JPUmiNoHi
CorpusChristi	LaborDay
DEAscension	MassOfArchangels
DEChristmasEve	NewYearsDay
DECorpusChristi	PalmSunday
DEGermanUnity	Pentecost
DENewYearsEve	PentecostMonday
Easter	PresentationOfLord
EasterMonday	Quinquagesima
EasterSunday	RogationSunday
Epiphany	Septuagesima
FRAAllSaints	SolemnityOfMary
FRArmisticeDay	TransfigurationOfLord
FRAscension	TrinitySunday
FRAssumptionVirginMary	USCPulaskisBirthday
FRBastilleDay	USChristmasDay
FRFeteDeLaVictoire1945	USColumbusDay
GBBankHoliday	USDecorationMemorialDay
GBMayDay	USElectionDay
GBMilleniumDay	USGoodFriday
GBSummerBankHoliday	USInaugurationDay
GoodFriday	USIndependenceDay
ITAAllSaints	USLaborDay
ITAAssumptionOfVirginMary	USLincolnsBirthday
ITEpiphany	USMLKingsBirthday
ITImmaculateConception	USMemorialDay
ITLiberationDay	USNewYearsDay
ITStAmrose	USPresidentsDay
JPAutumnalEquinox	USTurkeysgivingDay
JPBankHolidayDec31	USVeteransDay
JPBankHolidayJan2	USWashingtonsBirthday
JPBankHolidayJan3	

The digitize Package: Extracting Numerical Data from Scatterplots

by Timothée Poisot

Abstract I present the small R package **digitize**, designed to extract data from scatterplots with a simple method and suited to small datasets. I present an application of this method to the extraction of data from a graph whose source is not available.

The package **digitize**, that I present here, allows a user to load a graphical file of a scatterplot (with the help of the `read.jpeg` function of the **ReadImages** package) in the graphical window of R, and to use the `locator` function to calibrate and extract the data. Calibration is done by setting four reference points on the original graph axis, two for the x values and two for the y values. The use of four points for calibration is justified by the fact that it makes calibrations on the axis possible, as y data are not taken into account for calibration of the x axis, and vice versa.

This is useful when working on data that are not available in digital form, e.g. when integrating old papers in meta-analyses. Several commercial or free software packages allow a user to extract data from a plot in image format, among which we can cite PlotDigitizer (<http://plotdigitizer.sourceforge.net/>) or the commercial package GraphClick (<http://www.arizona-software.ch/graphclick/>). While these programs are powerful and quite ergonomic, for some lightweight use, one may want to load the graph directly into R, and as a result get the data directly in R format. This paper presents a rapid digitization of a scatterplot and subsequent statistical analysis of the data. As an example, we will use the data presented by Jacques Monod in a seminal microbiology paper (Monod, 1949).

The original paper presents the growth rate (in terms of divisions per hour) of the bacterium *Escherichia coli* in media of increasing glucose concentration. Such a hyperbolic relationship is best represented by the equation

$$R = R_K \frac{C}{C_1 + C},$$

where R is the growth rate at a given concentration of nutrients C , R_K is the maximal growth rate, C_1 is the concentration of nutrients at which $R = 0.5R_K$. In R, this function is written as

```
MonodGrowth <- function(params, M) {
  with(params, rK*(M/(M1+M)))
}
```

In order to characterize the growth parameters of a bacterial population, one can measure its growth

rate in different concentrations of nutrients. Monod (1949) proposed that, in the measured population, $R_K = 1.35$ and $C_1 = 22 \times 10^{-6}$. By using the **digitize** package to extract the data from this paper, we will be able to get our own estimates for these parameters.

Values of R_K and C_1 were estimated using a simple genetic algorithm, which minimizes the error function (sum of squared errors) defined by

```
MonodError <- function(params, M, y) {
  with(params,
    sum((MonodGrowth(params, M)-y)^2))
}
```

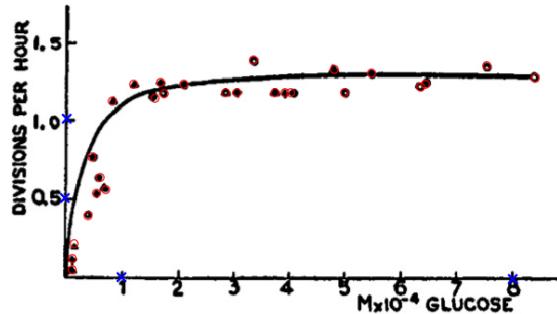


Figure 1: Original figure, as obtained after pointing and clicking the data. Calibration was made using the points $x_1 = 1$, $x_2 = 8$, $y_1 = 0.5$ and $y_2 = 1$. All the points that have been clicked are marked by a red point. The digitization of each series is stopped by right-clicking or pressing the Esc key.

The first step when using the **digitize** package is to specify four points on the graph that will be used to calibrate the axes. They must be in the following order : leftmost x , rightmost x , lower y , upper y . For the first two of them, the y value is not important (and vice versa). For this example, it is assumed that we set the first two points at $x_1 = 1$ and $x_2 = 8$, and the two last points at $y_1 = 0.5$ and $y_2 = 1$, simply by clicking in the graphical window at these positions (preferentially on the axes). It should be noted that it is not necessary to calibrate using the extremity of the axes.

Loading the figure and printing it in the current device for calibration is done by

```
cal <- ReadAndCal('monod.jpg')
```

Once the graph appears in the window, the user must input (by clicking on them) the four calibration points, marked as blue crosses. The calibration values will be stocked in the `cal` object, which is a list with x and y values. The next step is to read the data,

simply by pointing and clicking on the graph. This is done by calling the `DigitData` function, whose arguments are the type of lines/points drawn.

```
growth <- DigitData(col = 'red')
```

When all the points have been identified, the digitization is stopped in the same way that one stops the `locator` function, i.e. either by right-clicking or pressing the `Esc` key (see `?locator`). The outcome of this step is shown in figure 1. The next step for these data to be exploitable is to use the calibration information to have the correct coordinates of the points. We can do this by using the function `Calibrate(data, calpoints, x1, x2, y1, y2)`, and the correct coordinates of the points of calibration (x_1, x_2, y_1 and y_2 correspond, respectively, to the points x_1, x_2, y_1 and y_2).

```
data <- Calibrate(growth, cal, 1, 8, 0.5, 1)
```

The internals of the function `Calibrate` are quite simple. If we consider $X = (X_1, X_2)$ a vector containing the x coordinates of the calibration points for the x axis on the graphic window, and $x = (x_1, x_2)$ a vector with their true value, it is straightforward that $x = aX + b$. As such, performing a simple linear regression of x against X allows us to determine the coefficients to convert the data. The same procedure is repeated for the y axis. One advantage of this method of calibration is that you do not need to focus on the y value of the points used for x calibration, and reciprocally. It means that in order to accurately calibrate a graph, you only need to have two x coordinates, and two y coordinates. Eventually, it is very simple to calibrate the graphic by setting the calibration points directly on the tick mark of their axis.

The object returned by `Calibrate` is of class "data.frame", with columns "x" and "y" representing the x and y coordinates so that we can plot it directly. The following code produces Figure 2, assuming that `out` is a list containing the arguments of `MonodGrowth` obtained after optimization (`out$set`), and `paper` is the same list with the values of Monod (1949).

```
plot(data$x, data$y, pch=20, col='grey',
      xlab = 'Nutrients concentration',
      ylab = 'Divisions per hour')
points(xcal, MonodGrowth(out$set, xcal),
       type = 'l', lty = 1, lwd = 2)
points(xcal, MonodGrowth(paper, xcal),
       type = 'l', lty = 2)
legend('bottomright',
       legend = c('data', 'best fit', 'paper value'),
       pch = c(20, NA, NA),
       lty = c(NA, 1, 2),
       lwd = c(NA, 2, 1),
       col = c('grey', 'black', 'black'),
       bty = 'n')
```

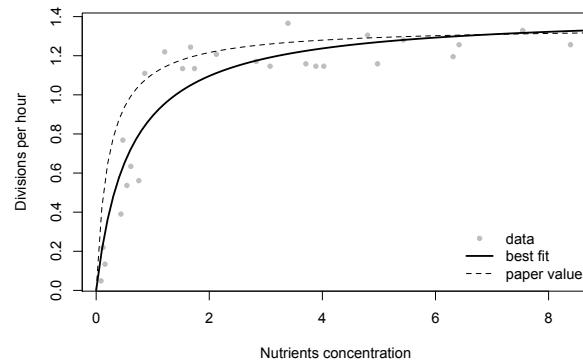


Figure 2: Result of the digitization. Original data are in grey, the proposed fit is in the dashed line, and the estimate realized from the raw data is in the bold line. Our estimates for the growth parameters are $R_K = 1.43$ and $C_1 = 62 \times 10^{-6}$.

While using the `MonodError` function with the proposed parameters yields a sum of squares of 1.32, our estimated parameters minimizes this value to 0.45, thus suggesting that the values of R_K and C_1 presented in the paper are not optimal given the data.

Conclusion

I presented an example of using the `digitize` package to reproduce data in an ancient paper that are not available in digital form. While the principle of this package is really simple, it allows for a quick extraction of data from a scatterplot (or any kind of planar display), which can be useful when data from old or proprietary figures are needed.

Acknowledgements

Thanks are due to two anonymous referees for their comments on an earlier version of this manuscript.

Bibliography

J. Monod. The growth of bacterial cultures. *Annual Reviews in Microbiology*, 3(1):371–394, 1949.

Timothée Poisot

Université Montpellier II

Institut des Sciences de l'Évolution, UMR 5554

Place Eugène Bataillon

34095 Montpellier CEDEX 05

France

tpoisot@um2.fr

Differential Evolution with DEoptim

An Application to Non-Convex Portfolio Optimization

by David Ardia, Kris Boudt, Peter Carl, Katharine M. Mullen and Brian G. Peterson

Abstract The R package **DEoptim** implements the Differential Evolution algorithm. This algorithm is an evolutionary technique similar to classic genetic algorithms that is useful for the solution of global optimization problems. In this note we provide an introduction to the package and demonstrate its utility for financial applications by solving a non-convex portfolio optimization problem.

Introduction

Differential Evolution (DE) is a search heuristic introduced by Storn and Price (1997). Its remarkable performance as a global optimization algorithm on continuous numerical minimization problems has been extensively explored (Price et al., 2006). DE has also become a powerful tool for solving optimization problems that arise in financial applications: for fitting sophisticated models (Gilli and Schumann, 2009), for performing model selection (Maringer and Meyer, 2008), and for optimizing portfolios under non-convex settings (Krisk and Paterlini, 2011). DE is available in R with the package **DEoptim**.

In what follows, we briefly sketch the DE algorithm and discuss the content of the package **DEoptim**. The utility of the package for financial applications is then explored by solving a non-convex portfolio optimization problem.

Differential Evolution

DE belongs to the class of genetic algorithms (GAs) which use biology-inspired operations of crossover, mutation, and selection on a population in order to minimize an objective function over the course of successive generations (Holland, 1975). As with other evolutionary algorithms, DE solves optimization problems by evolving a population of candidate solutions using alteration and selection operators. DE uses floating-point instead of bit-string encoding of population members, and arithmetic operations instead of logical operations in mutation, in contrast to classic GAs.

Let NP denote the number of parameter vectors (members) $\mathbf{x} \in \mathbb{R}^d$ in the population, where d denotes dimension. In order to create the initial generation, NP guesses for the optimal value of the parameter vector are made, either using random values be-

tween upper and lower bounds (defined by the user) or using values given by the user. Each generation involves creation of a new population from the current population members $\{\mathbf{x}_i | i = 1, \dots, NP\}$, where i indexes the vectors that make up the population. This is accomplished using *differential mutation* of the population members. An initial mutant parameter vector \mathbf{v}_i is created by choosing three members of the population, \mathbf{x}_{i_1} , \mathbf{x}_{i_2} and \mathbf{x}_{i_3} , at random. Then \mathbf{v}_i is generated as

$$\mathbf{v}_i \doteq \mathbf{x}_{i_1} + F \cdot (\mathbf{x}_{i_2} - \mathbf{x}_{i_3}),$$

where F is a positive scale factor, effective values for which are typically less than one. After the first mutation operation, mutation is continued until d mutations have been made, with a crossover probability $CR \in [0, 1]$. The crossover probability CR controls the fraction of the parameter values that are copied from the mutant. Mutation is applied in this way to each member of the population. If an element of the trial parameter vector is found to violate the bounds after mutation and crossover, it is reset in such a way that the bounds are respected (with the specific protocol depending on the implementation). Then, the objective function values associated with the children are determined. If a trial vector has equal or lower objective function value than the previous vector it replaces the previous vector in the population; otherwise the previous vector remains. Variations of this scheme have also been proposed; see Price et al. (2006).

Intuitively, the effect of the scheme is that the shape of the distribution of the population in the search space is converging with respect to size and direction towards areas with high fitness. The closer the population gets to the global optimum, the more the distribution will shrink and therefore reinforce the generation of smaller difference vectors.

For more details on the DE strategy, we refer the reader to Price et al. (2006) and Storn and Price (1997).

The package **DEoptim**

DEoptim (Ardia et al., 2011) was first published on CRAN in 2005. Since becoming publicly available, it has been used by several authors to solve optimization problems arising in diverse domains. We refer the reader to Mullen et al. (2011) for a detailed description of the package.

DEoptim consists of the core function **DEoptim** whose arguments are:

- `fn`: the function to be optimized (minimized).
- `lower, upper`: two vectors specifying scalar real lower and upper bounds on each parameter to

be optimized. The implementation searches between lower and upper for the global optimum of `fn`.

- `control`: a list of tuning parameters, among which: `NP` (default: $10 \cdot d$), the number of population members and `itermax` (default: 200), the maximum number of iterations (i.e., population generations) allowed. For details on the other `control` parameters, the reader is referred to the documentation manual (by typing `?DEoptim`). For convenience, the function `DEoptim.control()` returns a list with default elements of `control`.
- ...: allows the user to pass additional arguments to the function `fn`.

The output of the function `DEoptim` is a member of the S3 class `DEoptim`. Members of this class have a `plot` and a `summary` method that allow to analyze the optimizer's output.

Let us quickly illustrate the package's usage with the minimization of the Rastrigin function in \mathbb{R}^2 , which is a common test for global optimization:

```
> Rastrigin <- function(x) {
+   sum(x^2 - 10 * cos(2 * pi * x)) + 20
}
```

The global minimum is zero at point $x = (0,0)'$. A perspective plot of the function is shown in Figure 1.

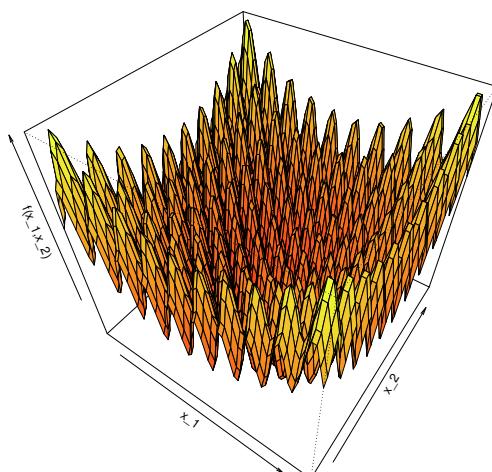


Figure 1: Perspective plot of the Rastrigin function.

The function `DEoptim` searches for a minimum of the objective function between lower and upper bounds. A call to `DEoptim` can be made as follows:

```
> set.seed(1234)
> DEoptim(fn = Rastrigin,
+   lower = c(-5, -5),
+   upper = c(5, 5),
+   control = list(storepopfrom = 1))
```

The above call specifies the objective function to minimize, `Rastrigin`, the lower and upper bounds on the parameters, and, via the `control` argument, that we want to store intermediate populations from the first generation onwards (`storepopfrom = 1`). Storing intermediate populations allows us to examine the progress of the optimization in detail. Upon initialization, the population is comprised of 50 random values drawn uniformly within the lower and upper bounds. The members of the population generated by the above call are plotted at the end of different generations in Figure 2. `DEoptim` consistently finds the minimum of the function (marked with an open white circle) within 200 generations using the default settings. We have observed that `DEoptim` solves the Rastrigin problem more efficiently than the simulated annealing method available in the R function `optim` (for all annealing schedules tried). Note that users interested in exact reproduction of results should set the seed of their random number generator before calling `DEoptim` (using `set.seed`). `DE` is a randomized algorithm, and the results may vary between runs.

Finally, note that `DEoptim` relies on repeated evaluation of the objective function in order to move the population toward a global minimum. Therefore, users interested in making `DEoptim` run as fast as possible should ensure that evaluation of the objective function is as efficient as possible. Using pure R code, this may often be accomplished using vectorization. Writing parts of the objective function in a lower-level language like C or Fortran may also increase speed.

Risk allocation portfolios

Mean-risk models were developed in early fifties for the portfolio selection problem. Initially, variance was used as a risk measure. Since then, many alternative risk measures have been proposed. The question of which risk measure is most appropriate is still the subject of much debate. Value-at-Risk (VaR) and Conditional Value-at-Risk (CVaR) are the most popular measures of downside risk. VaR is the negative value of the portfolio return such that lower returns will only occur with at most a preset probability level, which typically is between one and five percent. CVaR is the negative value of the mean of all return realizations that are below the VaR. There is a voluminous literature on portfolio optimization problems with VaR and CVaR risk measures, see, e.g., Fábián and Veszprémi (2008) and the references therein.

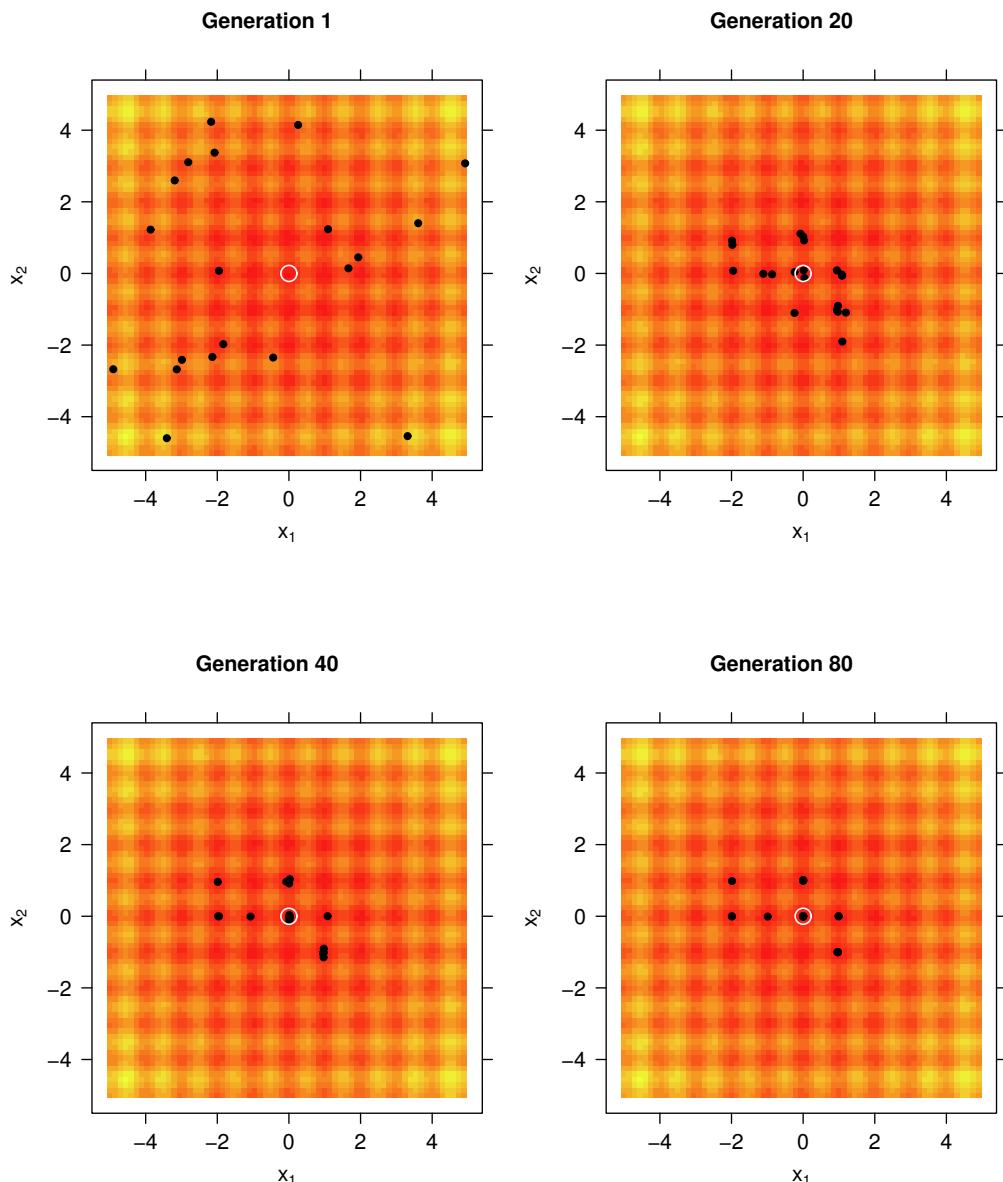


Figure 2: The population (market with black dots) associated with various generations of a call to `DEoptim` as it searches for the minimum of the Rastrigin function at point $x = (0,0)'$ (market with an open white circle). The minimum is consistently determined within 200 generations using the default settings of `DEoptim`.

Modern portfolio selection considers additional criteria such as to maximize upper-tail skewness and liquidity or minimize the number of securities in the portfolio. In the recent portfolio literature, it has been advocated by various authors to incorporate risk contributions in the portfolio allocation problem. Qian's (2005) *Risk Parity Portfolio* allocates portfolio variance equally across the portfolio components. Maillard et al. (2010) call this the *Equally-Weighted Risk Contribution (ERC) Portfolio*. They derive the theoretical properties of the ERC portfolio and show that its volatility is located between those of the minimum variance and equal-weight portfolio. Zhu et al. (2010) study optimal mean-variance portfolio selection under a direct constraint on the contributions to portfolio variance. Because the resulting optimization model is a non-convex quadratically constrained quadratic programming problem, they develop a branch-and-bound algorithm to solve it.

Boudt et al. (2010a) propose to use the contributions to portfolio CVaR as an input in the portfolio optimization problem to create portfolios whose percentage CVaR contributions are aligned with the desired level of CVaR risk diversification. Under the assumption of normality, the percentage CVaR contribution of asset i is given by the following explicit function of the vector of weights $w \doteq (w_1, \dots, w_d)'$, mean vector $\mu \doteq (\mu_1, \dots, \mu_d)'$ and covariance matrix Σ :

$$\frac{w_i \left[-\mu_i + \frac{(\Sigma w)_i}{\sqrt{w' \Sigma w}} \frac{\phi(z_\alpha)}{\alpha} \right]}{-w' \mu + \sqrt{w' \Sigma w} \frac{\phi(z_\alpha)}{\alpha}},$$

with z_α the α -quantile of the standard normal distribution and $\phi(\cdot)$ the standard normal density function. Throughout the paper we set $\alpha = 5\%$. As we show here, the package **DEoptim** is well suited to solve these problems. Note that it is also the evolutionary optimization strategy used in the package **PortfolioAnalytics** (Boudt et al., 2010b).

To illustrate this, consider as a stylized example a five-asset portfolio invested in the stocks with tickers GE, IBM, JPM, MSFT and WMT. We keep the dimension of the problem low in order to allow interested users to reproduce the results using a personal computer. More realistic portfolio applications can be obtained in a straightforward manner from the code below, by expanding the number of parameters optimized. Interested readers can also see the **DEoptim** package vignette for a 100-parameter portfolio optimization problem that is typical of those encountered in practice.

We first download ten years of monthly data using the function `getSymbols` of the package **quantmod** (Ryan, 2010). Then we compute the log-return series and the mean and covariance matrix estimators. For an overview of alternative estimators of the covariance matrix, such as outlier robust or shrink-

age estimators, and their implementation in portfolio allocation, we refer the reader to Würtz et al. (2009, Chapter 4). These estimators might yield better performance in the case of small samples, outliers or departures from normality.

```
> library("quantmod")
> tickers <- c("GE", "IBM", "JPM", "MSFT", "WMT")
> getSymbols(tickers,
+   from = "2000-12-01",
+   to   = "2010-12-31")
> P <- NULL
> for(ticker in tickers) {
+   tmp <- Cl(to.monthly(eval(parse(text = ticker))))
+   P <- cbind(P, tmp)
+ }
> colnames(P) <- tickers
> R <- diff(log(P))
> R <- R[-1,]
> mu <- colMeans(R)
> sigma <- cov(R)
```

We first compute the equal-weight portfolio. This is the portfolio with the highest weight diversification and often used as a benchmark. But is the risk exposure of this portfolio effectively well diversified across the different assets? This question can be answered by computing the percentage CVaR contributions with function `ES` in the package **PerformanceAnalytics** (Carl and Peterson, 2011). These percentage CVaR contributions indicate how much each asset contributes to the total portfolio CVaR.

```
> library("PerformanceAnalytics")
> pContribCVaR <- ES(weights = rep(0.2, 5),
+   method = "gaussian",
+   portfolio_method = "component",
+   mu = mu,
+   sigma = sigma)$pct_contrib_ES
> rbind(tickers, round(100 * pContribCVaR, 2))
      [,1]     [,2]     [,3]     [,4]     [,5]
tickers "GE"    "IBM"   "JPM"   "MSFT"  "WMT"
      "21.61" "18.6"  "25.1"  "25.39" "9.3"
```

We see that in the equal-weight portfolio, 25.39% of the portfolio CVaR risk is caused by the 20% investment in MSFT, while the 20% investment in WMT only causes 9.3% of total portfolio CVaR. The high risk contribution of MSFT is due to its high standard deviation and low average return (reported in percent):

```
> round(100 * mu, 2)
      GE    IBM    JPM    MSFT    WMT
-0.80  0.46 -0.06 -0.37  0.0
> round(100 * diag(sigma)^{(1/2)}, 2)
      GE    IBM    JPM    MSFT    WMT
  8.90  7.95  9.65 10.47  5.33
```

We now use the function `DEoptim` of the package **DEoptim** to find the portfolio weights for which the portfolio has the lowest CVaR and each investment can contribute at most 22.5% to total portfolio CVaR

risk. For this, we first define our objective function to minimize. The current implementation of **DEoptim** allows for constraints on the domain space. To include the risk budget constraints, we add them to the objective function through a penalty function. As such, we allow the search algorithm to consider infeasible solutions. A portfolio which is unacceptable for the investor must be penalized enough to be rejected by the minimization process and the larger the violation of the constraint, the larger the increase in the value of the objective function. A standard expression of these violations is $\alpha \times |\text{violation}|^p$. Crama and Schyns (2003) describe several ways to calibrate the scaling factors α and p . If these values are too small, then the penalties do not play their expected role and the final solution may be infeasible. On the other hand, if α and p are too large, then the CVaR term becomes negligible with respect to the penalty; thus, small variations of w can lead to large variations of the penalty term, which mask the effect on the portfolio CVaR. We set $\alpha = 10^3$ and $p = 1$, but recognize that better choices may be possible and depend on the problem at hand.

```
> obj <- function(w) {
+   if (sum(w) == 0) { w <- w + 1e-2 }
+   w <- w / sum(w)
+   CVaR <- ES(weights = w,
+     method = "gaussian",
+     portfolio_method = "component",
+     mu = mu,
+     sigma = sigma)
+   tmp1 <- CVaR$ES
+   tmp2 <- max(CVaR$pct_contrib_ES - 0.225, 0)
+   out <- tmp1 + 1e3 * tmp2
+ }
```

The penalty introduced in the objective function is non-differentiable and therefore standard gradient-based optimization routines cannot be used. In contrast, **DEoptim** is designed to consistently find a good approximation to the global minimum of the optimization problem:

```
> set.seed(1234)
> out <- DEoptim(fn = obj,
+   lower = rep(0, 5),
+   upper = rep(1, 5))
> out$optim$bestval
[1] 0.1143538
> wstar <- out$optim$bestmem
> wstar <- wstar / sum(wstar)
> rbind(tickers, round(100 * wstar, 2))
      par1    par2    par3    par4    par5
tickers "GE"    "IBM"   "JPM"   "MSFT"   "WMT"
      "17.38" "19.61" "14.85" "15.19" "32.98"
> 100 * (sum(wstar * mu) - mean(mu))
[1] 0.04150506
```

Note that the main differences with the equal-weight portfolio is the low weights given to JPM and MSFT and the high weight to WMT. As can be seen

from the last two lines, this *minimum risk* portfolio has a higher expected return than the equal weight portfolio. The following code illustrates that **DEoptim** yields superior results than the gradient-based optimization routines available in R.

```
> out <- optim(par = rep(0.2, 5),
+   fn = obj,
+   method = "L-BFGS-B",
+   lower = rep(0, 5),
+   upper = rep(1, 5))
> out$value
[1] 0.1255093
> out <- nlminb(start = rep(0.2, 5),
+   objective = obj,
+   lower = rep(0, 5),
+   upper = rep(1, 5))
> out$objective
[1] 0.1158250
```

Even on this relatively simple stylized example, the **optim** and **nlminb** routines converged to local minima.

Suppose now the investor is interested in the most risk diversified portfolio whose expected return is higher than the equal-weight portfolio. This amounts to minimizing the largest CVaR contribution subject to a return target and can be implemented as follows:

```
> obj <- function(w) {
+   if(sum(w) == 0) { w <- w + 1e-2 }
+   w <- w / sum(w)
+   contribCVaR <- ES(weights = w,
+     method = "gaussian",
+     portfolio_method = "component",
+     mu = mu,
+     sigma = sigma)$contribution
+   tmp1 <- max(contribCVaR)
+   tmp2 <- max(mean(mu) - sum(w * mu), 0)
+   out <- tmp1 + 1e3 * tmp2
+ }
> set.seed(1234)
> out <- DEoptim(fn = obj,
+   lower = rep(0, 5),
+   upper = rep(1, 5))
> wstar <- out$optim$bestmem
> wstar <- wstar / sum(wstar)
> rbind(tickers, round(100 * wstar, 2))
      par1    par2    par3    par4    par5
tickers "GE"    "IBM"   "JPM"   "MSFT"   "WMT"
      "17.38" "19.61" "14.85" "15.19" "32.98"
> 100 * (sum(wstar * mu) - mean(mu))
[1] 0.04150506
```

This portfolio invests more in the JPM stock and less in the GE (which has the lowest average return) compared to the portfolio with the upper 22.5% percentage CVaR constraint. We refer to Boudt et al. (2010a) for a more elaborate study on using CVaR allocations as an objective function or constraint in the portfolio optimization problem.

A classic risk/return (i.e., CVaR/mean) scatter chart showing the results for portfolios tested by

DEoptim is displayed in Figure 3. Gray elements depict the results for all tested portfolios (using hexagon binning which is a form of bivariate histogram, see Carr et al. (2011); higher density regions are darker). The yellow-red line shows the path of the best member of the population over time, with the darkest solution at the end being the *optimal* portfolio. We can notice how **DEoptim** does not spend much time computing solutions in the scatter space that are suboptimal, but concentrates the bulk of the calculation time in the vicinity of the final *best* portfolio. Note that we are not looking for the tangency portfolio; simple mean/CVaR optimization can be achieved with standard optimizers. We are looking here for the best balance between return and risk concentration.

We have utilized the mean return/CVaR concentration examples here as more realistic, but still stylized, examples of non-convex objectives and constraints in portfolio optimization; other non-convex objectives, such as drawdown minimization, are also common in real portfolios, and are likewise suitable to application of Differential Evolution. One of the key issues in practice with real portfolios is that a portfolio manager rarely has only a single objective or only a few simple objectives combined. For many combinations of objectives, there is no unique global optimum, and the constraints and objectives formed lead to a non-convex search space. It may take several hours on very fast machines to get the best answers, and the best answers may not be a true global optimum, they are just *as close as feasible* given potentially competing and contradictory objectives.

When the constraints and objectives are relatively simple, and may be reduced to quadratic, linear, or conical forms, a simpler optimization solver will produce answers more quickly. When the objectives are more layered, complex, and potentially contradictory, as those in real portfolios tend to be, **DEoptim** or other global optimization algorithms such as those integrated into **PortfolioAnalytics** provide a portfolio manager with a feasible option for optimizing their portfolio under real-world non-convex constraints and objectives.

The **PortfolioAnalytics** framework allows any arbitrary R function to be part of the objective set, and allows the user to set the relative weighting that they want on any specific objective, and use the appropriately tuned optimization solver algorithm to locate portfolios that most closely match those objectives.

Summary

In this note we have introduced DE and **DEoptim**. The package **DEoptim** provides a means of applying the DE algorithm in the R language and environment for statistical computing. DE and the package **DEoptim** have proven themselves to be powerful tools

for the solution of global optimization problems in a wide variety of fields. We have referred interested users to Price et al. (2006) and Mullen et al. (2011) for a more extensive introduction, and further pointers to the literature on DE. The utility of using **DEoptim** was further demonstrated with a simple example of a stylized non-convex portfolio risk contribution allocation, with users referred to **PortfolioAnalytics** for portfolio optimization using DE with real portfolios under non-convex constraints and objectives.

The latest version of the package includes the adaptive Differential Evolution framework by Zhang and Sanderson (2009). Future work will also be directed at parallelization of the implementation. The **DEoptim** project is hosted on R-forge at <https://r-forge.r-project.org/projects/deoptim/>.

Acknowledgements

The authors would like to thank Dirk Eddelbuettel, Juan Ospina, and Enrico Schumann for useful comments and Rainer Storn for his advocacy of DE and making his code publicly available. The authors are also grateful to two anonymous reviewers for helpful comments that have led to improvements of this note. Kris Boudt gratefully acknowledges financial support from the National Bank of Belgium.

Disclaimer

The views expressed in this note are the sole responsibility of the authors and do not necessarily reflect those of aeris CAPITAL AG, Guidance Capital Management, Cheiron Trading, or any of their affiliates. Any remaining errors or shortcomings are the authors' responsibility.

Bibliography

- D. Ardia, K. M. Mullen, B. G. Peterson and J. Ulrich. *DEoptim: Differential Evolution Optimization in R*, 2011. URL <http://CRAN.R-project.org/package=DEoptim>. R package version 2.1-0.
- K. Boudt, P. Carl, and B. G. Peterson. Portfolio optimization with conditional value-at-risk budgets. *Working paper*, 2010a.
- K. Boudt, P. Carl, and B. G. Peterson. *PortfolioAnalytics: Portfolio Analysis, including Numeric Methods for Optimization of Portfolios*, 2010b. URL <http://r-forge.r-project.org/projects/returnanalytics/>. R package version 0.6.
- P. Carl and B. G. Peterson. *PerformanceAnalytics: Econometric tools for performance and risk analysis*, 2011. URL <http://r-forge.r-project.org/>

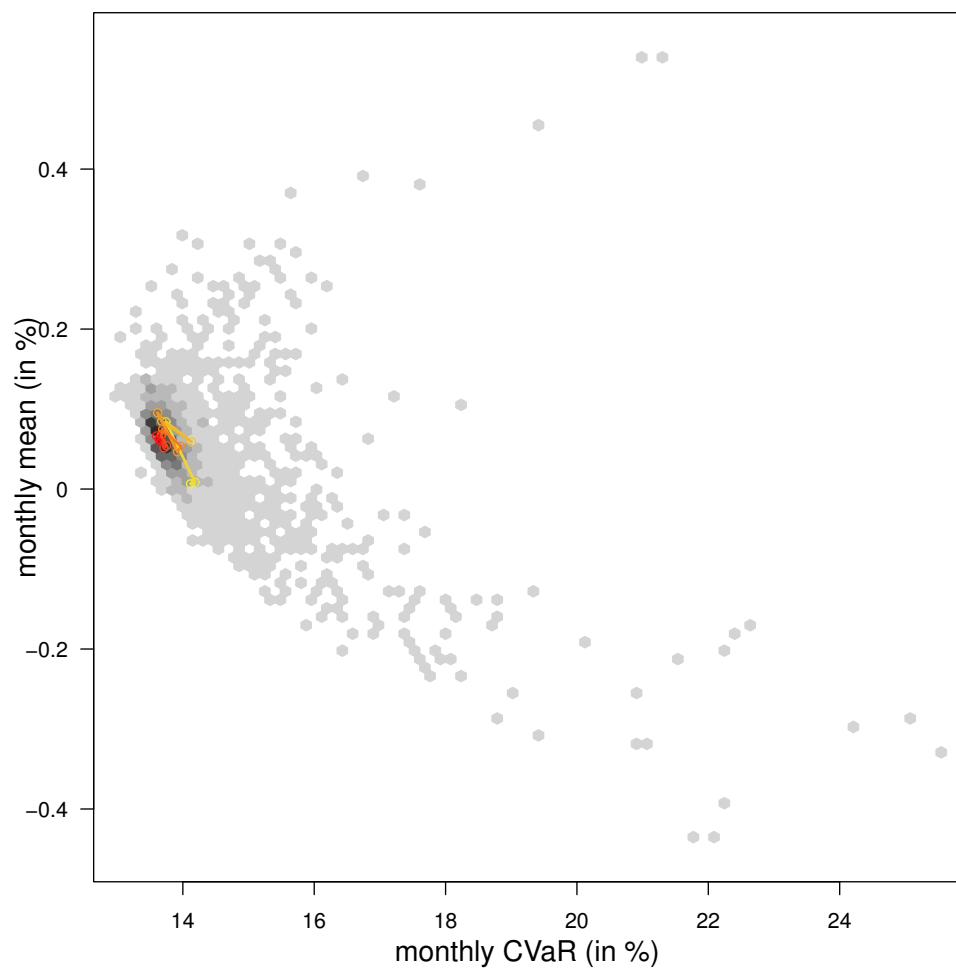


Figure 3: Risk/return scatter chart showing the results for portfolios tested by `DEoptim`.

- `projects/returnanalytics/`. R package version 1.0.3.3.
- D. Carr, N. Lewin-Koh and M. Maechler. *hexbin: Hexagonal binning routines*, 2011. URL <http://CRAN.R-project.org/package=hexbin>. R package version 1.26.0
- Y. Crama and M. Schyns. Simulated annealing for complex portfolio selection problems. *European Journal of Operations Research*, 150(3):546–571, 2003.
- C. Fábián and A. Veszprémi. Algorithms for handling CVaR constraints in dynamic stochastic programming models with applications to finance. *Journal of Risk*, 10(3):111–131, 2008.
- M. Gilli and E. Schumann. Heuristic optimisation in financial modelling. COMISEF wps-007 09/02/2009, 2009.
- J. H. Holland. *Adaptation in Natural Artificial Systems*. University of Michigan Press, 1975.
- T. Krink and S. Paterlini. Multiobjective optimization using Differential Evolution for real-world portfolio optimization. *Computational Management Science*, 8:157–179, 2011.
- S. Maillard, T. Roncalli, and J. Teiletche. On the properties of equally-weighted risk contributions portfolios. *Journal of Portfolio Management*, 36(4):60–70, 2010.
- D. G. Maringer and M. Meyer. Smooth transition autoregressive models: New approaches to the model selection problem. *Studies in Nonlinear Dynamics & Econometrics*, 12(1):1–19, 2008.
- K. M. Mullen, D. Ardia, D. L. Gil, D. Windover, and J. Cline. DEoptim: An R package for global optimization by Differential Evolution. *Journal of Statistical Software*, 40(6):1–26, 2011.
- K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, Berlin, Heidelberg, second edition, 2006. ISBN 3540209506.
- E. Qian. Risk parity portfolios: Efficient portfolios through true diversification of risk. *PanAgora Asset Management working paper*, 2005.
- J. A. Ryan. *quantmod: Quantitative Financial Modelling Framework*, 2010. URL <http://CRAN.R-project.org/package=quantmod>. R package version 0.3-16.
- O. Scaillet. Nonparametric estimation and sensitivity analysis of expected shortfall. *Mathematical Finance*, 14(1):74–86, 2002.
- R. Storn and K. Price. Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.
- D. Würtz, Y. Chalabi, W. Chen and A. Ellis *Portfolio Optimization with R/Rmetrics*. Rmetrics Association and Finance Online, 2009.
- J. Zhang and A. C. Sanderson. JADE: Adaptive Differential Evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.
- S. Zhu, D. Li, and X. Sun. Portfolio selection with marginal risk control. *Journal of Computational Finance*, 14(1):1–26, 2010.

David Ardia
aeris CAPITAL AG, Switzerland
da@aeris-capital.com

Kris Boudt
Lessius and K.U.Leuven, Belgium
kris.boudt@econ.kuleuven.be

Peter Carl
Guidance Capital Management, Chicago, IL
pcarl@gsb.uchicago.edu

Katharine M. Mullen
National Institute of Standards and Technology
Gaithersburg, MD
katharine.mullen@nist.gov

Brian G. Peterson
Cheiron Trading, Chicago, IL
brian@braverock.com

rworldmap: A New R package for Mapping Global Data

by Andy South

Abstract **rworldmap** is a relatively new package available on CRAN for the mapping and visualisation of global data. The vision is to make the display of global data easier, to facilitate understanding and communication. The initial focus is on data referenced by country or grid due to the frequency of use of such data in global assessments. Tools to link data referenced by country (either name or code) to a map, and then to display the map are provided as are functions to map global gridded data. Country and gridded functions accept the same arguments to specify the nature of categories and colour and how legends are formatted. This package builds on the functionality of existing packages, particularly **sp**, **maptools** and **fields**. Example code is provided to produce maps, to link with the packages **classInt**, **RColorBrewer** and **ncdf**, and to plot examples of publicly available country and gridded data.

Introduction

Global datasets are becoming increasingly common and are frequently seen on the web, in journal papers and in our newspapers (for example a 'carbon atlas' of global emissions available at http://image.guardian.co.uk/sys-files/Guardian/documents/2007/12/17/CARBON_ATLAS.pdf). At the same time there is a greater interest in global issues such as climate change, the global economy, and poverty (as for example outlined in the Millennium Development Goals, <http://www.un.org/millenniumgoals/bkgd.shtml>). Thirdly, there is an increasing availability of software tools for visualising data in new and interesting ways. Gapminder (<http://www.gapminder.org>) has pioneered making UN statistics more available and intelligible using innovative visualisation tools and Many Eyes (<http://www-958.ibm.com/>) provides a very impressive interface for sharing data and creating visualisations.

World maps have become so common that they have even attracted satire. The Onion's Atlas of the Planet Earth (The Onion, 2007), contains a 'Bono Awareness' world map representing 'the intensity with which artist Bono is aware of the plight and daily struggles of region', with a categorisation ranging from 'has heard of nation once' through 'moved enough by nations crisis to momentarily remove sunglasses' to 'cares about welfare of nation nearly as much as his own'.

There appears to be a gap in the market for free software tools that can be used across disciplinary boundaries to produce innovative, publication quality global visualisations. Within R there are great building blocks (particularly **sp**, **maptools** and **fields**) for spatial data but users previously had to go through a number of steps if they wanted to produce world maps of their own data. Experience has shown that difficulties with linking data and creating classifications, colour schemes and legends, currently constrains researchers' ability to view and display global data. We aim to reduce that constraint to allow researchers to spend more time on the more important issue of what they want to display. The vision for **rworldmap** is to produce a package to facilitate the visualisation and mapping of global data. Because the focus is on global data, the package can be more specialised than existing packages, making world mapping easier, partly because it doesn't have to deal with detailed local maps. Through **rworldmap** we aim to make it easy for R users to explore their global data and also to produce publication quality figures from their outputs.

rworldmap was partly inspired and largely funded by the UK Natural Environment Research Council (NERC) program Quantifying Uncertainty in Earth System Science (QUEST). This program brings together scientists from a wide range of disciplines including climate modellers, hydrologists and social scientists. It was apparent that while researchers have common tools for visualisation within disciplines, they tend to use different ones across disciplines and that this limits the sharing of data and methods necessary for truly interdisciplinary research. Within the project, climate and earth system modellers tended to use IDL, ecologists ArcGIS, hydrologists and social scientists Matlab and fisheries scientists R. With the exception of R, these software products cost thousands of pounds which acts as a considerable constraint on users being able to try out techniques used by collaborators. This high cost and learning curve of adopting new software tools hinders the sharing of data and methods between disciplines. To address this, part of the vision for **rworldmap** was to develop a tool that can be freely used and modified across a multi-disciplinary project, to facilitate the sharing of scripts, data and outputs. Such freely available software offers greater opportunity for collaboration with research institutes in developing countries that may not be able to afford expensive licenses.

rworldmap data inputs

rworldmap consists of tools to visualise global data and focuses on two types of data. Firstly, data that are referenced by country codes or names and secondly, data that are referenced on a grid.

Country data

There is a wealth of global country level data available on the internet including UN population data, and many global indices for, among others: Environmental Performance, Global Hunger and Multi-dimensional Poverty.

Data are commonly referenced by country names as these are the most easily recognised by users, but country names have the problem that vocabularies are not well conserved and many countries have a number of subtly different alternate names (e.g. Ivory Coast and Cote d'Ivoire, Laos and People's Democratic Republic of Lao). To address this problem there are ISO standard country codes of either 2 letters, 3 letters or numeric, and also 2 letter FIPS country codes, so there is still not one universally adopted standard. **rworldmap** supports all of these country codes and offers tools to help when data are referenced by names or other identifiers.

Gridded data

Global datasets are frequently spatially referenced on a grid, because such gridded or raster data formats offer advantages in efficiency of data storage and processing. Remotely sensed data and other values calculated from it are most frequently available in gridded formats. These can include terrestrial or marine data or both.

There are many gridded data formats, here I will concentrate on two: ESRI GridAscii and netCDF.

ESRI GridAscii files are an efficient way of storing and transferring gridded data. They are straightforward text files so can be opened by any text editor. They have a short header defining the structure of the file (e.g. number, size and position of rows and columns), followed by a single row specifying the value at each grid cell. Thus they use much less space than if the coordinates for each cell had to be specified.

Example start of gridAscii file for a half degree global grid:

```
ncols 720
nrows 360
xllcorner -180
yllcorner -90
cellsize 0.5
NODATA_value -999
-999 1 0 1 1 ... [all 259200 cell values]
```

NetCDF is a data storage file format commonly used by climate scientists and oceanographers. NetCDF files can be multi-dimensional, e.g. holding (x, y) data for multiple attributes over multiple months, years, days etc. The package **ncdf** is good for reading data from netCDF files.

rworldmap functionality

rworldmap has three core functions outlined below and others that are described later.

1. `joinCountryData2Map()` joins user country data referenced by country names or codes to a map to enable plotting
2. `mapCountryData()` plots a map of country data
3. `mapGriddedData()` plots a map of gridded data

Joining country data to a map

To join the data to a map use `joinCountryData2Map`. You will need to specify the name of column containing your country identifiers (`nameJoinColumn`) and the type of code used (`joinCode`) e.g. "ISO3" for ISO 3 letter codes or "UN" for numeric country codes.

```
data(countryExData)
SPDF <- joinCountryData2Map( countryExData
                           , joinCode = "ISO3"
                           , nameJoinColumn = "ISO3V10")
```

This code outputs, to the R console, a summary of how many countries are successfully joined. You can specify `verbose=TRUE` to get a full list of countries. The object returned (named `sPDF` in this case) is of type "`SpatialPolygonsDataFrame`" from the package `sp`. This object is required for the next step, displaying the map.

If you only have country names rather than codes in your data, use `joinCode="NAME"`; you can expect more mismatches due to the greater variation within country names mentioned previously. To address this you can use the `identifyCountries()` function described below, and change any country names in your data that do not exactly match those in the internal map.

Mapping country data

To plot anything other than the default map, `mapCountryData` requires an object of class "`SpatialPolygonsDataFrame`" and a specification of the name of the column containing the data to plot:

```
data(countryExData)
SPDF <- joinCountryData2Map( countryExData
                           , joinCode = "ISO3"
                           , nameJoinColumn = "ISO3V10")
mapDevice() #create world map shaped window
mapCountryData(sPDF
               , nameColumnToPlot='BIODIVERSITY')
```

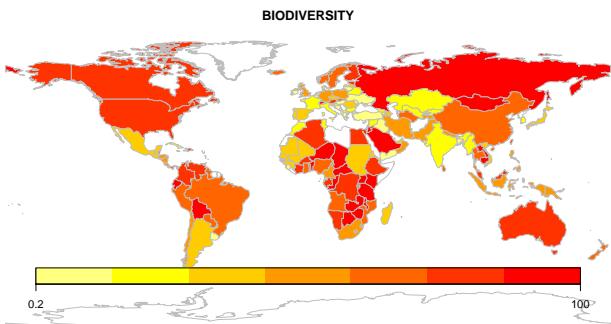


Figure 1: Output of `mapCountryData()`

Mapping gridded data

The `mapGriddedData` function can accept either

1. an object of type "SpatialGridDataFrame", as defined in the package `sp`
2. the name of an ESRI GridAscii file as a character string
3. a 2D R matrix or array (rows by columns)

`rworldmap` contains a "SpatialGridDataFrame" example that can be accessed and mapped as shown in the code and figure below.

```
data(gridExData)
mapDevice() #create world map shaped window
mapGriddedData(gridExData)
```

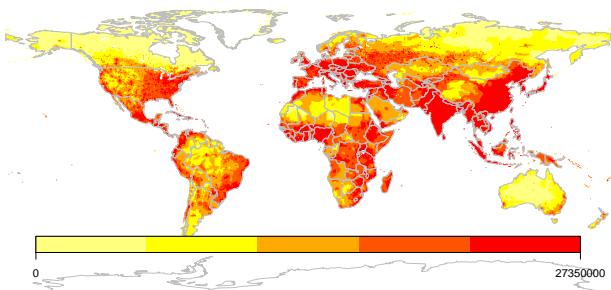


Figure 2: Output of `mapGriddedData()`

Modifying map appearance

`rworldmap` plotting functions are set up to work with few parameters specified (usually just those identifying the data) and in such cases, default values will be used. However, there is considerable flexibility to modify the appearance of plots by specifying values for arguments. Details of argument options are provided in the help files but here are a selection of the main arguments:

- `catMethod` determines how the data values are put into categories (then `colourPalette` determines the colours for those categories). Options for `catMethod` are: "pretty", "fixedWidth", "diverging", "logfixedWidth",

"quantiles", "categorical", or a numeric vector defining the breaks between categories. Works with the next argument (`numCats`), although `numCats` is not used for "categorical", where the data are not modified, or if the user specifies a numeric vector, where the number of categories will be a result.

- `numCats` specifies the favoured number of categories for the data. The number of categories used may be different if the number requested is not possible for the chosen `catMethod` (e.g. for "quantiles" if there are only 2 values in the data it is not possible to have more than 2 categories).
- `colourPalette` specifies a colour palette to use from:
 1. "palette" for the current palette
 2. a vector of valid colours, e.g. `c("red", "white", "blue")` or output from `RColorBrewer`
 3. a string defining one of the internal `rworldmap` palettes from: "heat", "diverging", "white2Black", "black2White", "topo", "rainbow", "terrain", "negpos8", "negpos9".
- `addLegend` set to TRUE for a default legend, if set to FALSE the function `addMapLegend` or `addMapLegendBoxes` can be used to create a more flexible legend.
- `mapRegion` a region to zoom in on, can be set to a country name from `getMap()$NAME` or one of "eurasia", "africa", "latin america", "uk", "oceania", "asia".

mapBubbles(), mapBars(), and mapPies()

Another option for displaying data is to use the `mapBubbles` function which allows flexible creation of bubble plots on global maps. You can specify data columns that will determine the sizing and colouring of the bubbles (using `nameZSize` and `nameZColour`). The function also accepts other `spatialDataFrame` objects or data frames containing columns specifying the x and y coordinates. If you wish to represent more attribute values per location there are also the newer `mapBars()` and `mapPies()` functions to produce bar and pie charts respectively (noting that pie charts may not be the best way of presenting data when there are more than a few categories).

```
mapDevice() #create world map shaped window
mapBubbles(dF=getMap()
           ,nameZSize="POP2005"
           ,nameZColour="REGION"
           ,colourPalette="rainbow"
           ,oceanCol="lightblue"
           ,landCol="wheat")
```

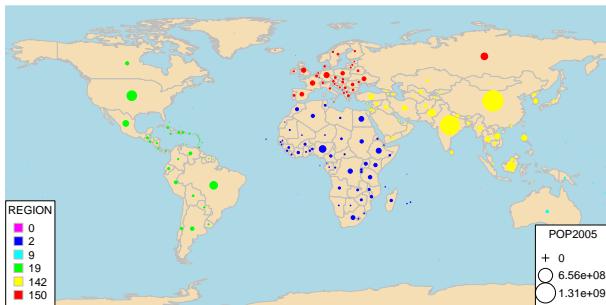


Figure 3: Output of mapBubbles()

Identifying countries

The interactive function `identifyCountries()` allows the user to click on the current map and, provided the cursor is sufficiently close to country centroid, will add the country name to the map. Optionally, the name of an attribute variable can also be passed to the function to cause the value of the attribute to be added to the label. e.g. 'Cape Verde 506807' will be printed on the map if the code below is entered and the mouse clicked over those islands (if you know where they are!).

```
identifyCountries(getMap()
  ,nameColumnToPlot="POP2005")
```

Aggregating data to produce different outputs

rworldmap offers options for aggregating half degree gridded data to countries, and in turn for aggregating country level data to regions. In both of these options a range of aggregation options are available including mean, minimum, maximum and variance.

`mapHalfDegreeGridToCountries()` takes a gridded input file, aggregates to a country level and plots the map. It accepts most of the same arguments as `mapCountryData()`.

Country level data can be aggregated to global regions specified by `regionType` in `country2Region()` which outputs as text, and `mapByRegion()` which produces a map plot. The regional classifications available include SRES (The Special Report on Emissions Scenarios of the Intergovernmental Panel on Climate Change (IPCC)), GEO3(Global Earth Observation), Stern and GBD (Global Burden of Disease).

```
data(countryExData)
country2Region(countryExData
  ,nameDataColumn="CLIMATE"
  ,joinCode="ISO3"
  ,nameJoinColumn="ISO3V10"
  ,regionType="Stern"
  ,FUN="mean")
```

Outputs this text:

	meanCLIMATEbyStern
Australasia	56.92000
Caribbean	65.20000
Central America	76.11250
Central Asia	56.18000
East Asia	69.18462
Europe	73.87619
North Africa	71.00000
North America	62.70000
South America	77.01818
South Asia	77.22000
South+E Africa	75.79474
West Africa	78.68421
West Asia	49.62000

```
data(countryExData)
mapDevice() #create world map shaped window
mapByRegion(countryExData
  ,nameDataColumn="CLIMATE"
  ,joinCode="ISO3"
  ,nameJoinColumn="ISO3V10"
  ,regionType="Stern"
  ,FUN="mean")
```

Produces this map:

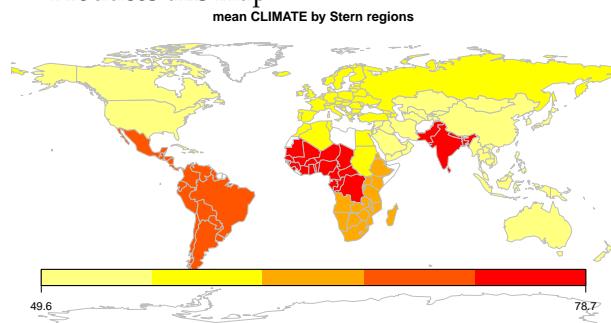


Figure 4: Output of mapByRegion()

The identity of which countries are in which regions are stored in the data frame `countryRegions`. This also identifies which countries are currently classified by the UN as Least Developed Countries (LDC), Small Island Developing states (SID) and Landlocked Developing Countries (LLDC). To map just the Least Developed Countries the code below could be used:

```
data(countryRegions)
sPDF <- joinCountryData2Map( countryRegions
  ,joinCode = "ISO3"
  ,nameJoinColumn = "ISO3")
mapDevice() #create world map shaped window
mapCountryData(sPDF[which(sPDF$LDC=='LDC'),]
  ,nameColumnToPlot="POP2005")
```

Using rworldmap with other packages classInt and RColorBrewer

While **rworldmap** sets many defaults internally there are also options to use other packages to have greater flexibility. In the example below **classInt** is used to create the classification and **RColorBrewer** to specify the colours.

```

library(classInt)
library(RColorBrewer)

#getting small example data and joining to a map
data(countryExData)
sPDF <- joinCountryData2Map(countryExData
  ,joinCode = "ISO3"
  ,nameJoinColumn = "ISO3V10"
  ,mapResolution = "coarse")

#getting class intervals
classInt <- classIntervals( sPDF[["EPI"]]
  ,n=5, style = "jenks")
catMethod = classInt[["brks"]]

#getting colours
colourPalette <- brewer.pal(5,'RdPu')

#plot map
mapDevice() #create world map shaped window
mapParams <- mapCountryData(sPDF
  ,nameColumnToPlot="EPI"
  ,addLegend=FALSE
  ,catMethod = catMethod
  ,colourPalette=colourPalette )

#adding legend
do.call(addMapLegend
  ,c(mapParams
    ,legendLabels="all"
    ,legendWidth=0.5
    ,legendIntervals="data"
    ,legendMar = 2))

```

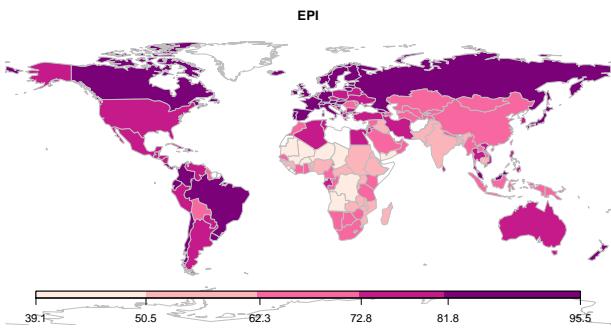


Figure 5: Output of `mapCountryData()` using inputs from `classInt` and `RColorBrewer`

Examples using data freely available on the web

Example country data from the web

The Happy Planet Index or HPI (<http://www.happyplanetindex.org>) combines country by country estimates of life expectancy, life satisfaction and ecological footprint to create an index of sustainable well-being that makes much more sense than GDP for assessing how countries are doing (Marks, 2010).

An Excel file containing the data ('`hpi-2-0-results.xls`') can be downloaded from the HPI

website at: <http://www.happyplanetindex.org/learn/download-report.html>.

I copied the country data from the sheet 'All Data' and pasted into a '.csv' file so that the header row was at the top of the file and the summary statistics at the bottom were left off. I then edited some of the column names to make them appropriate R variable names (e.g. changing 'Life Sat (0-10)' to 'LifeSat').

This data can then be read in easily using `read.csv()` and `joinCountryData2Map()`.

```

inFile <- 'hpi2_0edited2.csv'
dF <- read.csv(inFile,header=TRUE,as.is=TRUE)
sPDF <- joinCountryData2Map(dF
  ,joinCode='NAME'
  ,nameJoinColumn='country'
  ,verbose='TRUE')

```

Unfortunately, but in common with many global country level datasets, the countries are specified by name alone rather than by ISO country codes. The `verbose=TRUE` option in `joinCountryData2Map()` can be used to show the names of the 10 countries that don't join due to their names being slightly different in the data than `rworldmap`. The names of the countries that failed to join can be edited in the csv to be the same as those in `getMap() [['NAME']]`, and then they will join. A selection is shown below.

name in HPI	name in rworldmap
Iran	Iran (Islamic Republic of)
Korea	Korea, Republic of
Laos	Lao People's Democratic Republic
Moldova	Republic of Moldova
Syria	Syrian Arab Republic
Tanzania	United Republic of Tanzania
Vietnam	Viet Nam
United States of America	United States

The map of the HPI on the website is interesting in that the colours applied to countries are not determined by the value of the HPI for that country, but instead by the values of the three component indices for Life Expectancy, Life Satisfaction and Ecological Footprint. Therefore I needed to add some extra R code to be able to recreate the HPI map.

```

#categorise component indices
dF$LifeSatcolour <-
  ifelse(dF$LifeSat < 5.5,'red'
  ,ifelse(dF$LifeSat > 7.0,'green'
  ,'amber' ))

dF$LifeExpcolour <-
  ifelse(dF$LifeExp < 60,'red'
  ,ifelse(dF$LifeExp > 75,'green'
  ,'amber' ))

dF$HLYcolour <-
  ifelse(dF$HLY < 33,'red'
  ,ifelse(dF$HLY > 52.5,'green'
  ,'amber' ))

```

```

dF$Footprintcolour <-
  ifelse(dF$Footprint > 8.4, 'blood red'
    ,ifelse(dF$Footprint > 4.2, 'red'
      ,ifelse(dF$Footprint < 2.1, 'green'
        , 'amber' )))

#count red, amber , greens per country
numReds<-
  (as.numeric(dF$Footprintcolour=='red')
  +as.numeric(dF$LifeExpcolour=='red')
  +as.numeric(dF$LifeSatcolour=='red'))

numAmbers<-
  (as.numeric(dF$Footprintcolour=='amber')
  +as.numeric(dF$LifeExpcolour=='amber')
  +as.numeric(dF$LifeSatcolour=='amber'))

numGreens<-
  (as.numeric(dF$Footprintcolour=='green')
  +as.numeric(dF$LifeExpcolour=='green')
  +as.numeric(dF$LifeSatcolour=='green'))

#calculate HPI colour per country
dF$HPIcolour <-
  ifelse(dF$Footprintcolour=='blood red'
    | numReds>1,6
  ,ifelse(numReds==1,5
    ,ifelse(numAmbers==3,4
      ,ifelse(numGreens==1 & numAmbers==2,3
        ,ifelse(numGreens==2 & numAmbers==1,2
          ,ifelse(numGreens==3,1
            ,NA))))))

#join data to map
sPDF <- joinCountryData2Map(dF
  ,joinCode="NAME"
  ,nameJoinColumn="country")

#set colours
colourPalette <- c('palegreen'
  ,'yellow'
  ,'orange'
  ,'orangered'
  ,'darkred')

#plot map
mapDevice() #create world map shaped window
mapParams <- mapCountryData(sPDF
  ,nameColumnToPlot='HPIcolour'
  ,catMethod='categorical'
  ,colourPalette=colourPalette
  ,addLegend=FALSE
  ,mapTitle='Happy Planet Index')

#changing legendText
mapParams$legendText <-
  c('2 good, 1 middle'
  ,'1 good, 2 middle'
  ,'3 middle'
  ,'1 poor'
  ,'2 poor or footprint v.poor')

#add legend
do.call( addMapLegendBoxes
  , c(mapParams
    ,x='bottom'
    ,title="HPI colour"))

```

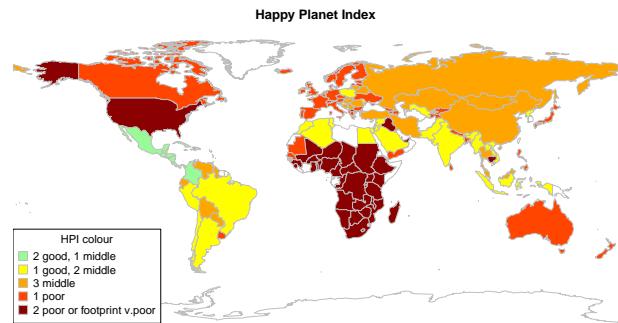


Figure 6: Happy Planet Index 2.0, using `rworldmap` to replicate map in happy planet report

Example gridded data from the web

'Koeppen Geiger' is a published classification dividing the world into 30 climatic zones. The GIS files for a Koeppen Geiger gridded climatic regions map are freely available from <http://koeppen-geiger.vu-wien.ac.at/>. The code below shows how to read in and plot an ascii file downloaded from that site.

```

inFile1 <- 'Koeppen-Geiger-ASCII.txt'
#read in data which is as lon,lat,catID
dF<-read.table(inFile1,header=TRUE,as.is=TRUE)
#convert to sp SpatialPointsDataFrame
coordinates(dF) = c("Lon", "Lat")
# promote to SpatialPixelsDataFrame
gridded(dF) <- TRUE
# promote to SpatialGridDataFrame
sGDF = as(dF, "SpatialGridDataFrame")
#plotting map
mapDevice() #create world map shaped window
mapParams <- mapGriddedData(sGDF
  ,catMethod='categorical'
  ,addLegend=FALSE)
#adding formatted legend
do.call(addMapLegendBoxes
  ,c(mapParams
    ,cex=0.8
    ,ncol=10
    ,x='bottom'
    ,title='Koeppen-Geiger Climate Zones'))

```

This produces a map that looks a lot different from the published map because it is using a different colour palette. The default "heat" colour palette is not the best for this categorical data and one of the palettes from **RColorBrewer** more suitable for categorical data could be used. However it would be good to retain the palette created by the authors of the data. The ascii file does not contain any colour information, however the authors also provide ESRI and Google Earth compatible files that do have colour information. It appeared to be impossible to extract the palette from the ESRI files, but by opening the '.kmz' file in Google Earth, saving to '.kml' and some fiddly text editing in R the colours

could be extracted as a single column then saved to a '.csv' file, the first few lines of which are:

```
colour
#960000
#ff0000
#ff9999
#ffcccc
...
```

Plotting the map in the original colours can then be achieved relatively simply by:

```
#reading in colour palette
#as.is=T stops conversion to factors
#which otherwise messes up colours
tst <- read.csv('paletteSaved.csv',as.is=TRUE)

#plotting map
mapDevice() #create world map shaped window
#tst$x passes the palette as a vector
mapParams <- mapGriddedData(sGDF
    ,catMethod='categorical'
    ,addLegend=FALSE
    ,colourPalette=tst$x)

#adding legend
do.call(addMapLegendBoxes
    ,c(mapParams
    ,cex=0.8
    ,ncol=3
    ,x='bottomleft'
    ,title='Koeppen-Geiger Climate Zones'))
```

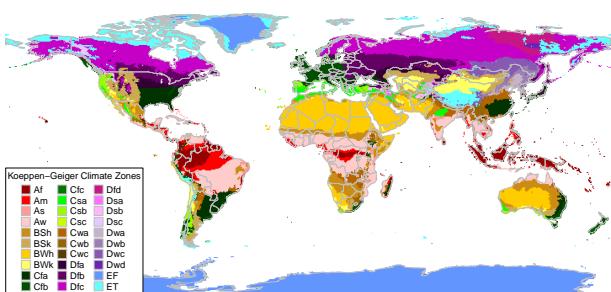


Figure 7: 'Koeppen Geiger' climatic zones, map produced using `mapGriddedData()` and freely available data

Using `rworldmap` to map netCDF data in tandem with `ncdf`

netCDF is a common file format used in meteorology and oceanography, it is both multi-dimensional and self documented. The package `ncdf` allows netCDF files to be opened, queried and data extracted directly to R. Data extracted from netCDF files in this way can then be converted to `sp` objects and plotted using `rworldmap`. In the example below the netCDF file (~ 333 KB) is first downloaded from the IPCC data distribution centre at: http://www.ipcc-data.org/cgi-bin/down1/ar4_nc/tas/HADCM3_SRA1B_1-tas-change_2046-2065.cyto180.nc.

This particular file was accessed from: http://www.ipcc-data.org/ar4/info/UKMO-HADCM3_SRA1B_tas.html

The 'lon180' option was chosen to get a file with longitude values from -180 to 180 as that requires less editing before plotting.

```
library(ncdf)
#the downloaded file
inFile <-
  'HADCM3_SRA1B_1_tas-change_2046-2065.cyto180.nc'
memory.limit(4000) #set memory limit to max
nc = open.ncdf(inFile, write=FALSE)
```

`print(nc)` prints to console a description of the file contents which includes the information shown below (edited slightly for brevity).

```
file ... has 4 dimensions:
time   Size: 12
latitude  Size: 73
longitude  Size: 96
bounds   Size: 2
-----
file ... has 4 variables
float climatological_bounds [bounds,time]
float latitude_bounds [bounds,latitude]
float longitude_bounds [bounds,longitude]
float
air_temperature_anomaly [longitude,latitude,time]
```

After the netCDF file has been opened in this way, selected variables can be read into R using `get.var.ncdf()`, converted to a `SpatialGridDataFrame` and plotted using the `rworldmap` function `mapGriddedData`. The code below first creates a grid from the parameters in netCDF file, and is written to be generic so that it should work on other similar netCDF files. Then it creates a standard classification and colour scheme for all of the plots based on having looked at the values in each months data first. Finally it loops through all months reads in the data for that month, creates a `spatialGridDataFrame` for each, plots it and saves it as a '.png'.

```
ncArray =
  get.var.ncdf(nc, 'air_temperature_anomaly')

# creating gridTopology from the netCDF metadata
offset = c(min(nc$dim$longitude$vals)
           ,min(nc$dim$latitude$vals))
cellsize = c( abs(diff(nc$dim$longitude$vals[1:2]))
            , abs(diff(nc$dim$latitude$vals[1:2])))
# add cellsize/2 to offset
# to convert from lower left referencing to centre
offset = offset + cellsize/2
cells.dim = c(nc$dim$longitude$len
              ,nc$dim$latitude$len )

gt <- GridTopology(cellcentre.offset = offset
                    , cellsize = cellsize
                    , cells.dim = cells.dim )
```

```

mapDevice()
#creating a vector to classify the data
catMethod=seq(from=-5,to=19,by=2)
#creating a colourPalette for all plots
#-ve blue, 0 white, +ve yellow to red
colourPalette=c('blue','lightblue','white',
                ,brewer.pal(9,'YlOrRd'))

#looping for each month
for( zDim in 1 : nc$dim$time$len ){
  #reading the values for this month
  ncMatrix <- ncArray[,zDim]
  #to get the image up the right way
  #this reverses the y values but not the x ones
  ncMatrix2 <-ncMatrix[ ,nc$dim$latitude$len:1 ]
  gridVals <-data.frame(att=as.vector(ncMatrix2))
  #creating a spatialGridDataFrame
  sGDF <-SpatialGridDataFrame(gt, data=gridVals)

  #plotting the map and getting params for legend
  mapParams <- mapGriddedData( sGDF
                                ,nameColumnToPlot='att'
                                ,catMethod=catMethod
                                ,colourPalette=colourPalette
                                ,addLegend=FALSE )

  #adding formatted legend
  do.call(addMapLegend
         ,c(mapParams
            ,legendLabels="all"
            ,legendWidth=0.5
            ,legendMar = 3))
  title(paste('month : ',zDim))#adding brief title

  outputPlotType = 'png'
  savePlot(paste("ipccAirAnomalyMonth",zDim,sep=''))
  ,type=outputPlotType)
} #end of month loop
close.ncdf(nc) #closing the ncdf file

```

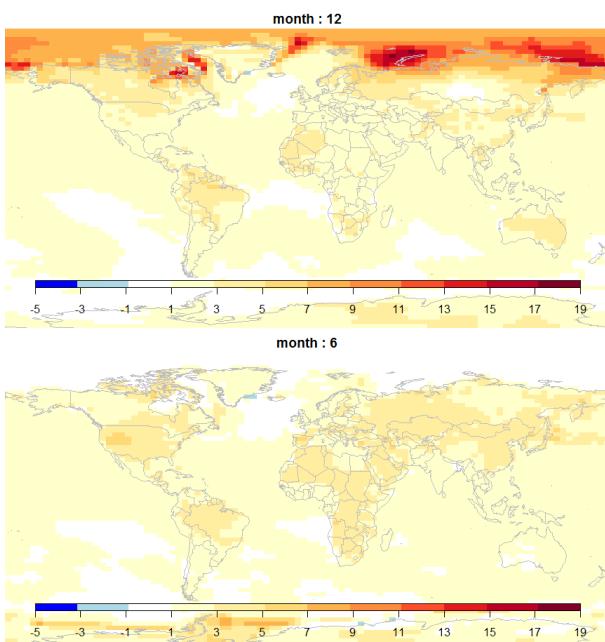


Figure 8: Two examples of IPCC temperature anomaly data (for December and June respectively) plotted using `mapGriddedData()`

Summary

rworldmap is a new package to facilitate the display of global data, referenced by grid, country or region. It is available on CRAN at <http://cran.r-project.org/web/packages/rworldmap>. **rworldmap** aims to provide tools to improve the communication and understanding of world datasets. If you have any comments or suggestions or would like to contribute code please get in touch. The source code and development versions are available from <http://code.google.com/p/rworld/>. I plan to extend this work by making datasets, including those used in this paper, more easily accessible, perhaps through a package called **rworldmapData**. We are also working on visualisations to communicate more information than by maps alone. For more help on rworldmap there is a vignette available at <http://cran.r-project.org/web/packages/rworldmap/vignettes/rworldmap.pdf> and an FAQ at <http://cran.r-project.org/web/packages/rworldmapvignettes/rworldmapFAQ.pdf>.

Acknowledgements

This work was funded by NERC through the QUEST Data Initiative (QESDI) and QUEST Global Systems Impacts (GSI) projects. Thanks to Pru Foster and Sarah Cornell at QUEST and Graham Pilling at Cefas without whose enthusiasm this would not have happened. Thanks to Martin Juckes for understanding project management, Barry Rowlingson and Roger Bivand for their contributions at a startup workshop and after, and to Nick Dulvy, Joe Scutt-Phillips and Elisabeth Simelton for inspiration on global vulnerability analyses and colour.

Bibliography

N. Marks GDP RIP (1933-2010). *Significance*, 7(1):27–30, 2010. Published Online: 23 Feb 2010 URL <http://www3.interscience.wiley.com/cgi-bin/fulltext/123300797/PDFSTART>.

The Onion *Our Dumb World: The Onion's Atlas of the Planet Earth*. Little, Brown, and Company, 2007. ISBN: 978 0 7528 9120 0. URL <http://www.theonion.com/>.

Andy South

Centre for Environment, Fisheries and Aquaculture Science (Cefas)

Pakefield Road, Lowestoft, NR33 OHT, UK

Current address: Graduate School of Education

University of Exeter

Exeter, EX1 2LU, UK

southandy@gmail.com

Appendix: Comparing using rworldmap to not using it

Here I show a brief comparison between using **rworldmap** and not. It is not my intention to set up a straw-man that demonstrates how superior my package is to others. Other packages have different objectives, by focusing on world maps I can afford to ignore certain details. It is only by building on the great work in other packages that I have been able to get this far. Those caveats aside here is the comparison using some data on alcohol consumption per adult by country, downloaded as an Excel file from the gapminder website at <http://www.gapminder.org/data/> and saved as a '.csv'. Reading in the data is common to both approaches:

```
inFile <- 'indicatoralcoholconsumption20100830.csv'
dF <- read.csv(inFile)
```

Using rworldmap

```
library(rworldmap)
sPDF <- joinCountryData2Map(dF,
  , joinCode = "NAME"
  , nameJoinColumn = "X"
  , nameCountryColumn = "X"
  , verbose = TRUE)

mapCountryData(sPDF, nameColumnToPlot='X2005')
```

Not using rworldmap

```
library(maptools)
library(fields)
## get map
data(wrld_simpl) #from package maptools
## joining
#first identify failures
matchPosnsInLookup <- match(
  as.character(dF$X)
  ,as.character(wrld_simpl$NAME))
failedCodes <- dF$X[is.na(matchPosnsInLookup)]
numFailedCodes <- length(failedCodes)

#printing info to console
cat(numFailedCodes
  , "countries failed to join to the map\n")
print(failedCodes)
#find match positions in the data
matchPosnsInData <- match(
  as.character(wrld_simpl$NAME)
  ,as.character(dF$X))
# join data to the map
wrld_simpl@data <- cbind(wrld_simpl@data
  , dF[matchPosnsInData,])

#sizing window to a good shape for the world
```

```
dev.new(width=9,height=4.5)
#so that maps extends to edge of window
oldpar <- par(mai=c(0,0,0.2,0))

#categorising the data
numCats <- 7
quantileProbs <- seq(0,1,1/numCats)
quantileBreaks <- quantile(wrld_simpl$X2005
  ,na.rm=T
  ,probs=quantileProbs)

wrld_simpl$toPlot <- cut( wrld_simpl$X2005
  , breaks=quantileBreaks
  , labels=F )

#plotting map
plot(wrld_simpl
  ,col=rev(heat.colors(numCats)) [wrld_simpl$toPlot])

#adding legend using the fields package
zlim <- range(quantileBreaks,na.rm=TRUE)
image.plot(legend.only=TRUE
  ,zlim=zlim
  ,col=rev(heat.colors(numCats))
  ,breaks=quantileBreaks
  ,horizontal=TRUE)

par(oldpar) #reset graphics settings
```

Slight differences in country naming, and an absence of country codes, causes 9 countries not to join in both approaches. `joinCountryData2Map()` outputs the identities of these countries. This means that in the maps it incorrectly looks like alcohol consumption is zero in both the UK and USA. To correct this the country names need to be renamed prior to joining, either in the '.csv' or in the dataframe. In the R dataframe the countries could be renamed by:

```
n1<-'United Kingdom of Great Britain and Northern Ireland'
n2<-'United Kingdom'
levels(dF$X) [which(levels(dF$X)==n1)] <- n2
```

The objective for **rworldmap** is to make world mapping easier, based on difficulties experienced by the author and project collaborators. Of course, there is still a learning curve associated with being able to use **rworldmap** itself. All of the operations shown in this paper can be accomplished by accomplished R programmers without the need for **rworldmap**, however in an inter-disciplinary project **rworldmap** made it easier for scientists who are new to R to start producing outputs, and encouraged them to extend their R learning further. There are repetitive data manipulation routines that are tricky to implement, **rworldmap** reduces the time spent on these so that more time can be spent on the important task of communicating what the data say.

Cryptographic Boolean Functions with R

by Frédéric Lafitte, Dirk Van Heule and Julien Van hamme

Abstract A new package called **boolfun** is available for R users. The package provides tools to handle Boolean functions, in particular for cryptographic purposes. This document guides the user through some (code) examples and gives a feel of what can be done with the package.

A Boolean function is a mapping $\{0,1\}^n \rightarrow \{0,1\}$. Those mappings are of much interest in the design of cryptographic algorithms such as secure pseudorandom number generators (for the design of stream ciphers among other applications), hash functions and block ciphers. The lack of open source software to assess cryptographic properties of Boolean functions and the increasing interest for statistical testing of properties related to random Boolean functions (Filiol, 2002; Saarinen, 2006; Englund et al., 2007; Aumasson et al., 2009) are the main motivations for the development of this package.

The number of Boolean functions with n variables is 2^{2^n} , i.e. 2 possible output values for each of the 2^n input assignments. In this search space of size 2^{2^n} , looking for a function which has specific properties is impractical. Already for $n \geq 6$, an exhaustive search would require too many computational resources. Furthermore, most properties are computed in $O(n2^n)$. This explains why the cryptographic community has been interested in evolutionary techniques (e.g. simulated annealing, genetic algorithms) to find an optimal function in this huge space. Another way to tackle the computational difficulties is to study algebraic constructions of Boolean functions. An example of such constructions is to start from two functions with m variables and to combine them to get a function with $n > m$ variables that provably preserves or enhances some properties of the initial functions. In both cases, the **boolfun** package can be used to experiment on Boolean functions and test cryptographic properties such as nonlinearity, algebraic immunity and resilience. This short article gives an overview of the package. More details can be found in the package vignettes.

First steps

In R, type `install.packages("boolfun")` in order to install the package and `library(boolfun)` to load it.

A Boolean function is instantiated with its truth table, a character or integer vector representing the output values of the function. For example, `f <- BooleanFunction("01101010")` defines a Boolean function with $n = 3$ input variables. Also, `g <- BooleanFunction(c(tt(f),tt(f)))` defines a Boolean

function with $n = 4$ by concatenating `f`'s truth table.

In order to represent the truth table as a vector of return values without ambiguity, a total order needs to be defined over the input assignments. The position of the element (x_1, \dots, x_n) in this ordering is simply the integer encoded in base 2 by the digits $x_n \dots x_1$. For example, the first function defined above is explicitly represented by the following truth table.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	0
1	1	1	1

Methods of the `BooleanFunction` object can be called in two ways, using functional notation as in `tt(f)` or using object oriented notation as in `f$tt()`. This is a feature of any object that inherits from `Object` defined in the **R.oo** package (Bengtsson, 2003).

An overview of all public methods is given in Figure 1 with a short description.

method	returned value
<code>n()</code>	number of input variables n
<code>tt()</code>	truth table (vector of integers)
<code>wh()</code>	walsh spectrum (vector of integers)
<code>anf()</code>	algebraic normal form (vector of integers)
<code>ANF()</code>	algebraic normal form as <code>Polynomial</code>
<code>deg()</code>	algebraic degree
<code>nl()</code>	nonlinearity
<code>ai()</code>	algebraic immunity
<code>ci()</code>	correlation immunity
<code>res()</code>	resiliency
<code>isBal()</code>	TRUE if function is balanced
<code>isCi(t)</code>	TRUE if <code>ci()</code> returns t
<code>isRes(t)</code>	TRUE if <code>res()</code> returns t

Figure 1: Public methods of `BooleanFunction`.

Also some generic functions such as `print()` or `equals()` are overloaded so that they support instances of `BooleanFunction`. Any additional information can be found in the document displayed by the R command `vignette("boolfun")`.

Note that an object `Polynomial` is also implemented in the **boolfun** package and is discussed in the following section. The other sections give some examples on how to use the package in order to visualize properties or to analyze random Boolean functions.

Multivariate polynomials over \mathbb{F}_2

Let \mathbb{F}_2 denote the finite field with two elements and let \oplus denote the addition in \mathbb{F}_2 (exclusive or). Formally, the algebraic normal form of a Boolean function f is an element $\text{anf}(f)$ of the quotient ring

$$\mathbb{F}_2[x_1, \dots, x_n] / \langle x_1^2 = x_1, \dots, x_n^2 = x_n \rangle$$

defined as follows

$$\text{anf}(f) = \bigoplus_{(a_1, \dots, a_n) \in \{0,1\}^n} h(a_1, \dots, a_n) \cdot x_1^{a_1} \cdots x_n^{a_n}$$

where the function h is based on the Möbius inversion principle

$$h(a_1, \dots, a_n) = \bigoplus_{(x_1, \dots, x_n) \in \{0,1\}^n} f(x_1, \dots, x_n)$$

Simply put, $\text{anf}(f)$ is a multivariate polynomial in (Boolean) variables x_1, \dots, x_n where coefficients and exponents are in $\{0,1\}$. Those polynomials are commonly called *Boolean polynomials* and represent uniquely the corresponding Boolean function.

The recent package **multipol** (Hankin, 2008) allows the user to handle multivariate polynomials but is not well suited to handle operations over polynomial Boolean rings. In **boolfun**, an S3 object **Polynomial** is defined and implements basic functionality to manipulate the algebraic normal form of Boolean functions.

```
> f <- BooleanFunction("01011010")
> p <- f$ANF()
> data.class(p)
[1] "Polynomial"
> q <- Polynomial("0101")
> p
[1] "x1 + x3"
> q
[1] "x1*x2 + x1"
> p * q
[1] "x1*x2*x3 + x1*x2 + x1*x3 + x1"
> p * q + q
[1] "x1*x2*x3 + x1*x3"
> deg(p * q + q)
[1] 3
```

In this example, `p` holds the algebraic normal form of `f` which is represented in **Polynomial** by a vector of length 2^n , i.e. the truth table of the Boolean function h defined above. The operator `[[]]` can be used to evaluate the polynomial for a particular assignment.

```
> r <- p * q + q
> x <- c(0, 1, 1) # x1=0, x2=1, x3=1
> if( length(x) != r$n() ) stop("this is an error")
> p[[x]]
[1] 1
> x[3] <- 0
> p[[x]]
[1] 0
```

The **Polynomial** object inherits from **Roo**'s **Object** and Figure 2 gives an overview of the implemented methods.

method	returned value
<code>n()</code>	number of input variables n
<code>deg()</code>	algebraic degree
<code>anf()</code>	vector of coefficients for 2^n monomials
<code>len()</code>	returns 2^n
<code>string()</code>	algebraic normal form as character string
<code>*</code>	multiplication returns a new Polynomial
<code>+</code>	addition returns a new Polynomial
<code>[[]]</code>	evaluates the polynomial

Figure 2: Public methods of **Polynomial**.

Addition and multiplication are executed using C code. The addition is straightforward given two vectors of coefficients as it consists in making a component-wise exclusive or of the input vectors. The multiplication is built upon the addition of two polynomials and the multiplication of two monomials. Hence addition is computed in $O(2^n)$ and multiplication in $O(n \cdot 2^{2n})$. Note that lower complexities can be achieved using graph based representations such as binary decision diagrams. For now, only vectors of length 2^n are used which are sufficient to manipulate sets of Boolean functions having up to about $n = 22$ variables (and $n = 13$ variables if algebraic immunity needs to be assessed).

Visualizing the tradeoff between cryptographic properties

Depending on its cryptographic application, a Boolean function needs to satisfy a set of properties in order to be used safely. Some of those properties cannot be satisfied simultaneously as there are trade-offs between them. In this section we focus on two properties, resilience and algebraic immunity, and show how R can be used to illustrate the trade-off that exists between them.

The algebraic immunity of a Boolean function f is defined to be the lowest degree of the nonzero functions g such that $f \cdot g = 0$ or $(f \oplus 1)g = 0$. This property is used to assess the resistance to some algebraic attacks.

Resilience is a combination of two properties: balancedness and correlation immunity. A function is balanced if its truth table contains as many zeros as ones and correlation immune of order t if the probability distribution of its output does not change when at most t input variables are fixed. Hence a function is resilient of order t if its output remains balanced even after fixing at most t input variables.

The following example illustrates the trade-off between algebraic immunity and resilience.

```

n <- 3
N <- 2^2^n # number of functions with n var

allRes <- vector("integer", N)
allAIs <- vector("integer", N)

for( i in 1:N ) { # forall Boolean function
  f <- BooleanFunction( toBin(i-1,2^n) )
  allRes[i] <- res(f) # get its resiliency
  allAIs[i] <- ai(f) # and algebraic immunity
}

xlabel <- "Truth tables as integers"

plot( x=1:N, y=allRes, type="b",
      xlab=xlabel, ylab="Resiliency" )
plot( x=1:N, y=allAIs, type="b",
      xlab=xlabel, ylab="Algebraic immunity" )

```

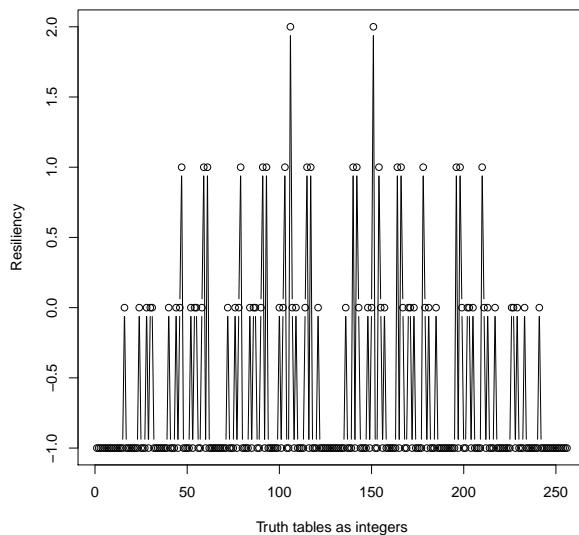


Figure 3: Resiliency of all functions with 3 variables.

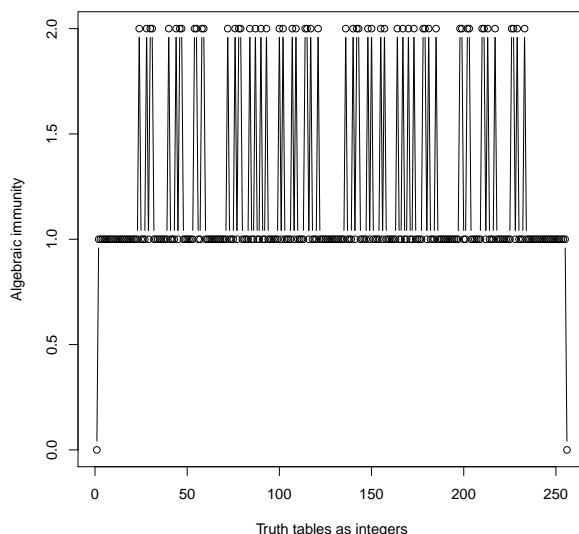


Figure 4: Algebraic immunity of all functions with 3 variables.

The example code plots the resilience and the algebraic immunity of all functions with 3 variables. The result is shown in Figures 3 and 4.

Observe that the search space of size 2^6 has a particular structure. The symmetry between the two halves of the search space is justified in that a function with constant term zero, i.e. with $f(0, \dots, 0) = 0$ will have the same properties as its complement $1 \oplus f$ (i.e. the same function with constant term 1). Note also that in those figures the functions achieving the highest resiliencies belong to the second quarter of the search space, which also seems more dense in algebraically immune functions. Very similar plots are observed when considering more variables.

Now let us consider the functions achieving a good tradeoff between algebraic immunity and resilience. The sum $ai(f) + res(f)$ is computed for each function f with 3 variables according to the following code and plotted in Figure 5.

```
plot( 1:N, allRes+allAIs, type="b",
      xlab="f", ylab="ai(f)+res(f) " )
```

Note that for all function f , the value $res(f) + ai(f)$ never reaches the value $\max(allRes) + \max(allAIs)$, hence the tradeoff. Also this figure suggests which part of the search space should be explored in order to find optimal tradeoffs.

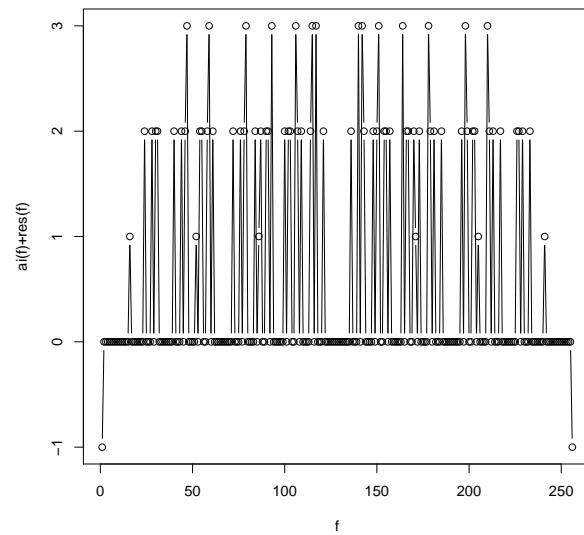


Figure 5: The sum of algebraic immunity with resiliency for all functions with 3 variables.

Properties of random Boolean functions

Figures 3 and 4 suggest that some properties are more likely to be observed in a random Boolean function. For example, there are more functions having maximal algebraic immunity than functions having

maximal resilience. The following example gives a better idea of what is expected in random Boolean functions.

```
n <- 8
data <- data.frame( matrix(nrow=0,ncol=4) )
names(data) <- c( "deg", "ai", "nl", "res" )
for( i in 1:1000 ) { # for 1000 random functions
  randomTT <- round(runif(2^n, 0, 1))
  randomBF <- BooleanFunction(randomTT)
  data[i,] <-c( deg(randomBF), ai(randomBF),
    nl(randomBF), res(randomBF) )
}
```

After the code is executed, `data` holds the values of four properties (columns) for 1000 random Boolean functions with $n = 8$ variables. The mean and standard deviation of each property is given below.

```
> mean(data)
   deg      ai      nl      res
 7.479  3.997 103.376 -0.939
> sd(data)
   deg      ai      nl      res
0.5057814 0.0547174 3.0248593 0.2476698
```

It is also very easy to apply statistical tests using R. For example, in (Englund et al., 2007; Aumasson et al., 2009) cryptographic pseudorandom generators are tested for non random behavior. Those tests consider the i^{th} output bit of the generator as a (Boolean) function f_i of n chosen input bits, the remaining input bits being fixed to some random value. Then the properties of the f_i s are compared to the expected properties of a random function. All those tests involve a χ^2 statistic in order to support a goodness of fit test. Such testing is easy with R using the function `qchisq` from the package `stats` as suggested by the following code.

```
data <- getTheBooleanFunctions()
chistat <- computeChiStat(data)

outcome <- "random"
if(chistat > qchisq(0.99, df=n))
  outcome <- "cipher"

print(outcome)
```

Summary

A free open source package to manipulate Boolean functions is available at cran.r-project.org. The package also provides tools to handle Boolean polynomials (multivariate polynomials over \mathbb{F}_2). `boolfun` has been developed to evaluate some cryptographic properties of Boolean functions and carry statistical analysis on them. An effort has been made to optimize execution speed rather than memory usage using C code. It is easy to extend this package

in order to add new functionality such as computing the autocorrelation spectrum, manipulating (rotation) symmetric Boolean functions, etc... as well as additional features for multivariate polynomials.

Bibliography

- J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir. Cube testers and key recovery attacks on reduced-round MD6 and Trivium. In *FSE*, pages 1–22, 2009.
- H. Bengtsson. The R.oo Package - Object-Oriented Programming with References Using Standard R Code. In *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria*, 2003.
- H. Englund, T. Johansson, and M. S. Turan. A framework for chosen iv statistical analysis of stream ciphers. In *INDOCRYPT*, pages 268–281, 2007.
- E. Filoli. A new statistical testing for symmetric ciphers and hash functions. In *ICICS*, pages 342–353, 2002.
- R. Hankin. Programmers' Niche: Multivariate polynomials in R. *R News*, 8(1):41–45, 2008.
- D. Knuth. A draft of section 7.1.1: Boolean basics. *The Art of Computer Programming, volume 4 pre-fascicle 0B*, 2006.
- W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989. ISSN 0933-2790.
- W. Meier, E. Pasalic, and C. Carlet. Algebraic attacks and decomposition of boolean functions. In *EUROCRYPT*, pages 474–491, 2004.
- M. Saarinen. Chosen-IV statistical attacks on estream ciphers. In *Proceeding of SECRYPT 2006*. Citeseer, 2006.

Frédéric Lafitte
Department of Mathematics
Royal Military Academy
Belgium
frederic.lafitte@rma.ac.be

Dirk Van Heule
Department of Mathematics
Royal Military Academy
Belgium
dirk.van.heule@rma.ac.be

Julien Van hamme
Department of Mathematics
Royal Military Academy
Belgium
julien.van.hamme@rma.ac.be

Raster Images in R Graphics

by Paul Murrell

Abstract The R graphics engine has new support for rendering raster images via the functions `rasterImage()` and `grid.raster()`. This leads to better scaling of raster images, faster rendering to screen, and smaller graphics files. Several examples of possible applications of these new features are described.

Prior to version 2.11.0, the core R graphics engine was entirely vector based. In other words, R was only capable of drawing mathematical shapes, such as lines, rectangles, and polygons (and text).

This works well for most examples of statistical graphics because plots are typically made up of data symbols (polygons) or bars (rectangles), with axes (lines and text) alongside (see Figure 1).

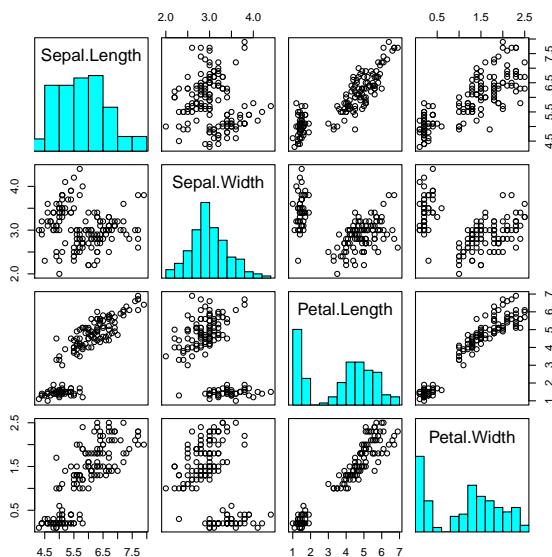


Figure 1: A typical statistical plot that is well-described by vector elements: polygons, rectangles, lines, and text.

However, some displays of data are inherently *raster*. In other words, what is drawn is simply an array of values, where each value is visualized as a square or rectangular region of colour (see Figure 2).

It is possible to draw such raster elements using vector primitives—a small rectangle can be drawn for each data value—but this approach leads to at least two problems: it can be very slow to draw lots of small rectangles when drawing to the screen; and it can lead to very large files if output is saved in a vector file format such as PDF (and viewing the resulting PDF file can be very slow).

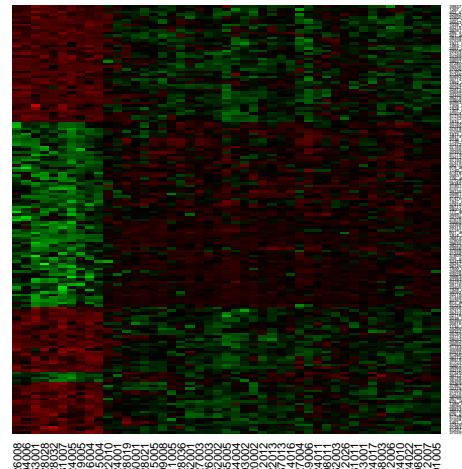


Figure 2: An example of an inherently raster graphical image: a heatmap used to represent microarray data. This image is based on Cock (2010); the data are from Chiaretti et al. (2004).

Another minor problem is that some PDF viewers have trouble reconciling their anti-aliasing algorithms with a large number of rectangles drawn side by side and may end up producing an ugly thin line between the rectangles.

To avoid these problems, from R version 2.11.0 on, the R graphics engine supports rendering raster elements as part of a statistical plot.

Raster graphics functions

The low-level R language interface to the new raster facility is provided by two new functions: `rasterImage()` in the **graphics** package and `grid.raster()` in the **grid** package.

For both functions, the first argument provides the raster image that is to be drawn. This argument should be a "raster" object, but both functions will accept any object for which there is an `as.raster()` method. This means that it is possible to simply specify a vector, matrix, or array to describe the raster image. For example, the following code produces a simple greyscale image (see Figure 3).

```
> library(grid)
> grid.raster(1:10/11)
```



Figure 3: A simple greyscale raster image generated from a numeric vector.

As the previous example demonstrates, a numeric vector is interpreted as a greyscale image, with 0 corresponding to black and 1 corresponding to white. Other possibilities are logical vectors, which are interpreted as black-and-white images, and character vectors, which are assumed to contain either colour names or RGB strings of the form "#RRGGBB".

The previous example also demonstrates that a vector is treated as a matrix of pixels with a single column. More usefully, the image to draw can be specified by an explicit matrix (numeric, logical, or character). For example, the following code shows a simple way to visualize the first 100 named colours in R.

```
> grid.raster(matrix(colors() [1:100], ncol=10),
+               interpolate=FALSE)
```

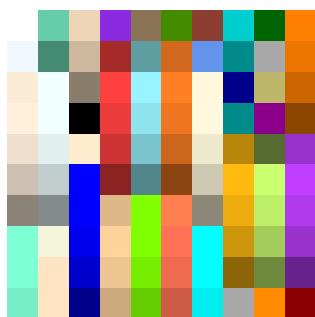


Figure 4: A raster image generated from a character matrix (of the first 100 values in `colors()`).

It is also possible to specify the raster image as a *numeric array*: either three-planes of red, green, and blue channels, or four-planes where the fourth plane provides an "alpha" (transparency) channel.

Greater control over the conversion to a "raster" object is possible by directly calling the `as.raster()` function, as shown below.

```
> grid.raster(as.raster(1:10, max=11))
```

Interpolation

The simple image matrix example above demonstrates another important argument in both of the

new raster functions—the `interpolate` argument.

In most cases, a raster image is not going to be rendered at its "natural" size (using exactly one device pixel for each pixel in the image), which means that the image has to be resized. The `interpolate` argument is used to control how that resizing occurs.

By default, the `interpolate` argument is `TRUE`, which means that what is actually drawn by R is a linear interpolation of the pixels in the original image. Setting `interpolate` to `FALSE` means that what gets drawn is essentially a sample from the pixels in the original image. The former case was used in Figure 3 and it produces a smoother result, while the latter case was used in Figure 4 and the result is more "blocky." Figure 5 shows the images from Figures 3 and 4 with their interpolation settings reversed.

```
> grid.raster(1:10/11, interpolate=FALSE)

> grid.raster(matrix(colors() [1:100], ncol=10))
```

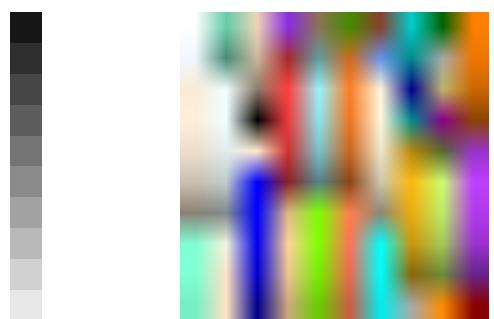


Figure 5: The raster images from Figures 3 and 4 with their interpolation settings reversed.

The ability to use linear interpolation provides another advantage over the old behaviour of drawing a rectangle per pixel. For example, Figure 6 shows a greyscale version of the R logo image drawn using both the old behaviour and with the new raster support. The latter version of the image is smoother thanks to linear interpolation.

```
> download.file("http://cran.r-project.org/Rlogo.jpg",
+                 "Rlogo.jpg")
> library(ReadImages)
> logo <- read.jpeg("Rlogo.jpg")

> par(mar=rep(0, 4))
> plot(logo)

> grid.raster(logo)
```



Figure 6: The R logo drawn using the old behaviour of drawing a small rectangle for each pixel (left) and using the new raster support with linear interpolation (right), which produces a smoother result.

In situations where the raster image represents actual data (e.g., microarray data), it is important to preserve each individual “pixel” of data. If an image is viewed at a reduced size, so that there are fewer screen pixels used to display the image than there are pixels in the image, then some pixels in the original image will be lost (though this is true whether the image is rendered as pixels or as small rectangles).

What has been described so far applies equally to the `rasterImage()` function and the `grid.raster()` function. The next few sections look at each of these functions separately to describe their individual features.

The `rasterImage()` function

The `rasterImage()` function is analogous to other low-level **graphics** functions, such as `lines()` and `rect()`; it is designed to *add* a raster image to the current plot.

The image is positioned by specifying the location of the bottom-left and top-right corners of the image in user coordinates (i.e., relative to the axis scales in the current plot).

To provide an example, the following code sets up a matrix of normalized values based on a mathematical function (taken from the first example on the `image()` help page).

```
> x <- y <- seq(-4*pi, 4*pi, len=27)
> r <- sqrt(outer(x^2, y^2, "+"))
> z <- cos(r^2)*exp(-r/6)
> image <- (z - min(z))/diff(range(z))
>
```

The following code draws a raster image from this matrix that occupies the entire plot region (see Figure 7). Notice that some work needs to be done to correctly align the raster cells with axis scales when the pixel coordinates in the image represent data values.

```
> step <- diff(x)[1]
> xrange <- range(x) + c(-step/2, step/2)
> yrange <- range(y) + c(-step/2, step/2)
```

```
> plot(x, y, ann=FALSE,
+       xlim=xrange, ylim=yrange,
+       xaxs="i", yaxs="i")
> rasterImage(image,
+              xrange[1], yrange[1],
+              xrange[2], yrange[2],
+              interpolate=FALSE)
```

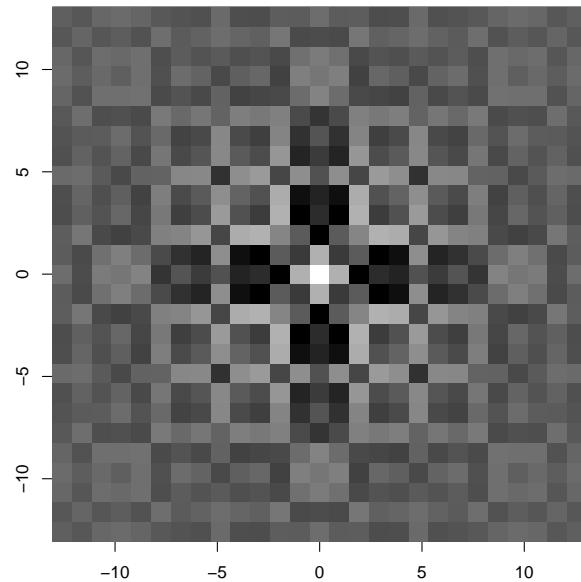


Figure 7: An image drawn from a matrix of normalized values using `rasterImage()`.

It is also possible to rotate the image (about the bottom-left corner) via the `angle` argument.

To avoid distorting an image, some calculations using functions like `xinch()`, `yinch()`, and `dim()` (to get the dimensions of the image) may be required. More sophisticated support for positioning and sizing the image is provided by `grid.raster()`.

The `grid.raster()` function

The `grid.raster()` function works like any other **grid** graphical primitive; it draws a raster image within the current **grid** viewport.

By default, the image is drawn as large as possible while still respecting its native aspect ratio (Figure 3 shows an example of this behaviour). Otherwise, the image is positioned according to the arguments `x` and `y` (justified by `just`, `hjust`, and `vjust`) and sized via `width` and `height`. If only one of `width` or `height` is given, then the aspect ratio of the image is preserved (and the image may extend beyond the current viewport).

Any of `x`, `y`, `width`, and `height` can be vectors, in which case multiple copies of the image are drawn. For example, the following code uses `grid.raster()` to draw the R logo within each bar of a **lattice** bar-chart.

```
> x <- c(0.00, 0.40, 0.86, 0.85, 0.69, 0.48,
+      0.54, 1.09, 1.11, 1.73, 2.05, 2.02)
> library(lattice)

> barchart(1:12 ~ x, origin=0, col="white",
+           panel=function(x, y, ...) {
+             panel.barchart(x, y, ...)
+             grid.raster(logo, x=0, width=x, y=y,
+                         default.units="native",
+                         just="left",
+                         height=unit(2/37,
+                                     "npc"))
+           })
+ }
```

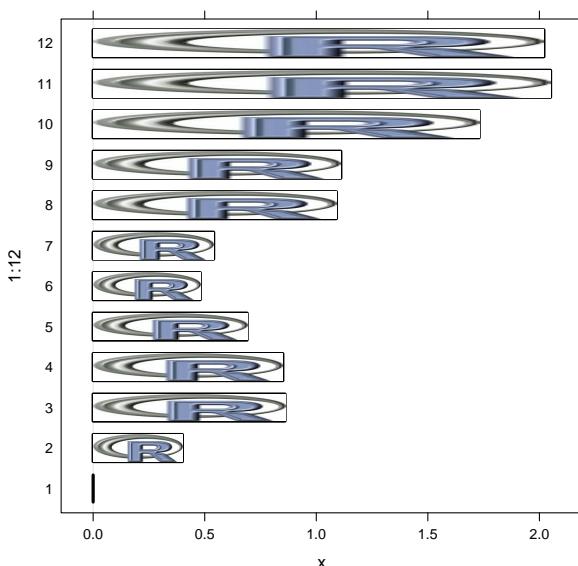


Figure 8: A **lattice** barchart of the change in the “level of interest in R” during 1996 (see `demo(graphics)`) with the R logo image used to annotate each bar (with apologies to Edward Tufte and all of his heirs and disciples).

In addition to `grid.raster()`, there is also a `rasterGrob()` function to create a raster image graphical object.

Support for raster image packages

A common source of raster images is likely to be external files, such as digital photos and medical scans. A number of R packages exist to read general raster formats, such as JPEG or TIFF files, plus there are many packages to support more domain-specific formats, such as NIFTI and ANALYZE.

Each of these packages creates its own sort of data structure to represent the image within R, so in order to render an image from an external file, the data structure must be converted to something that `rasterImage()` or `grid.raster()` can handle.

Ideally, the package will provide a method for the `as.raster()` function to convert the package-specific

image data structure into a "raster" object. In the absence of that, the simplest path is to convert the data structure into a matrix or array, for which there already exist `as.raster()` methods.

In the example that produced Figure 6, the R logo was loaded into R using the **ReadImages** package, which created an "imagematrix" object called `logo`. This object could be passed directly to either `rasterImage()` or `grid.raster()` because an "imagematrix" is also an array, so the predefined conversion for arrays did the job.

Profiling

This section briefly demonstrates that the claimed improvements in terms of file size and speed of rendering are actually true. The following code generates a simple random test case image (see Figure 9). The only important feature of this image is that it is a reasonably large image (in terms of number of pixels).

```
> z <- matrix(runif(500*500), ncol=500)
```

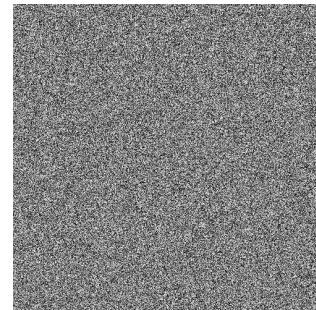


Figure 9: A simple random test image.

The following code demonstrates that a PDF version of this image is more than three times larger if drawn using a rectangle per pixel (via the `image()` function) compared to a file that is generated using `grid.raster()`.

```
> pdf("image.pdf")
> image(z, col=grey(0:99/100))
> dev.off()

> pdf("gridraster.pdf")
> grid.raster(z, interp=FALSE)
> dev.off()

> file.info("image.pdf", "gridraster.pdf") ["size"]
          size
image.pdf     14893004
gridraster.pdf 1511027
```

The next piece of code can be used to demonstrate that rendering speed is also much slower when drawing an image to screen as many small rectangles. The timings are from a run on a CentOS Linux

system with the Cairo-based X11 device. There are likely to be significant differences to these results if this code is run on other systems.

```
> system.time({
+   for (i in 1:10) {
+     image(z, col=grey(0:99/100))
+   }
+ })

  user  system elapsed
42.017  0.188 42.484

> system.time({
+   for (i in 1:10) {
+     grid.newpage()
+     grid.raster(z, interpolate=FALSE)
+   }
+ })

  user  system elapsed
2.013  0.081  2.372
```

This is not a completely fair comparison because there are different amounts of input checking and housekeeping occurring inside the `image()` function and the `grid.raster()` function, but more detailed profiling (with `Rprof()`) was used to determine that most of the time was being spent by `image()` doing the actual rendering.

Examples

This section demonstrates some possible applications of the new raster support.

The most obvious application is simply to use the new functions wherever images are currently being drawn as many small rectangles using a function like `image()`. For example, Granovskaia et al. (2010) used the new raster graphics support in R in the production of gene expression profiles (see Figure 10).

Having raster images as a graphical primitive also makes it easier to think about performing some graphical tricks that were not necessarily obvious before. An example is gradient fills, which are not explicitly supported by the R graphics engine. The following code shows a simple example where the bars of a barchart are filled with a greyscale gradient (see Figure 11).

```
> barchart(1:12 ~ x, origin=0, col="white",
+           panel=function(x, y, ...) {
+             panel.barchart(x, y, ...)
+             grid.raster(t(1:10/11), x=0,
+                         width=x, y=y,
+                         default.units="native",
+                         just="left",
+                         height=unit(2/37,
+                                     "npc"))
+           })
```

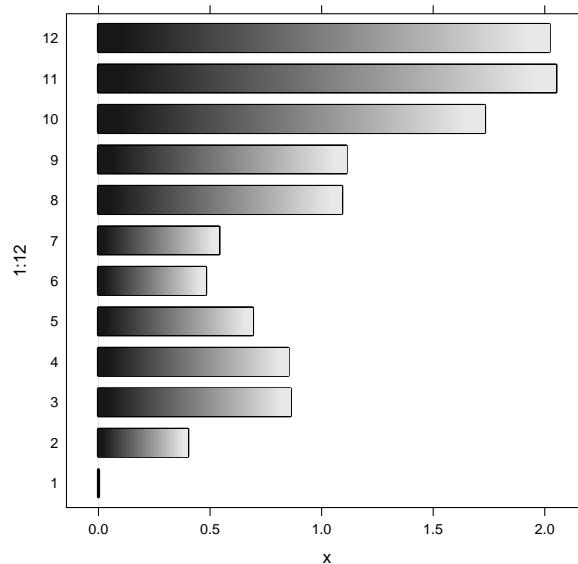


Figure 11: A `lattice` barchart of the change in the “level of interest in R” during 1996 (see `demo(graphics)`) with the a greyscale gradient used to fill each bar (once again, begging the forgiveness of Edward Tufte).

Another example is non-rectangular clipping operations via raster “masks” (R’s graphics engine only supports clipping to rectangular regions). The code below is used to produce a map of Spain that is filled with black (see Figure 12).

```
> library(maps)
> par(mar=rep(0, 4))
> map(region="Spain", col="black", fill=TRUE)
```

Having produced this image on screen, the function `grid.cap()` can be used to capture the current screen image as a raster object.

```
> mask <- grid.cap()
```

An alternative approach would be produce a PNG file and read that in, but `grid.cap()` is more convenient for interactive use.

The following code reads in a raster image of the Spanish flag from an external file (using the `png` package), converting the image to a “raster” object.

```
> library(png)
> espana <- readPNG("1000px-Flag_of_Spain.png")
> espanaRaster <- as.raster(espana)
```

We now have two raster images in R. The following code trims the flag image on its right edge and trims the map of Spain on its bottom edge so that the two images are the same size (demonstrating that “raster” objects can be subsetted like matrices).

```
> espanaRaster <- espanaRaster[, 1:dim(mask)[2]]
> mask <- mask[1:dim(espanaRaster)[1], ]
```

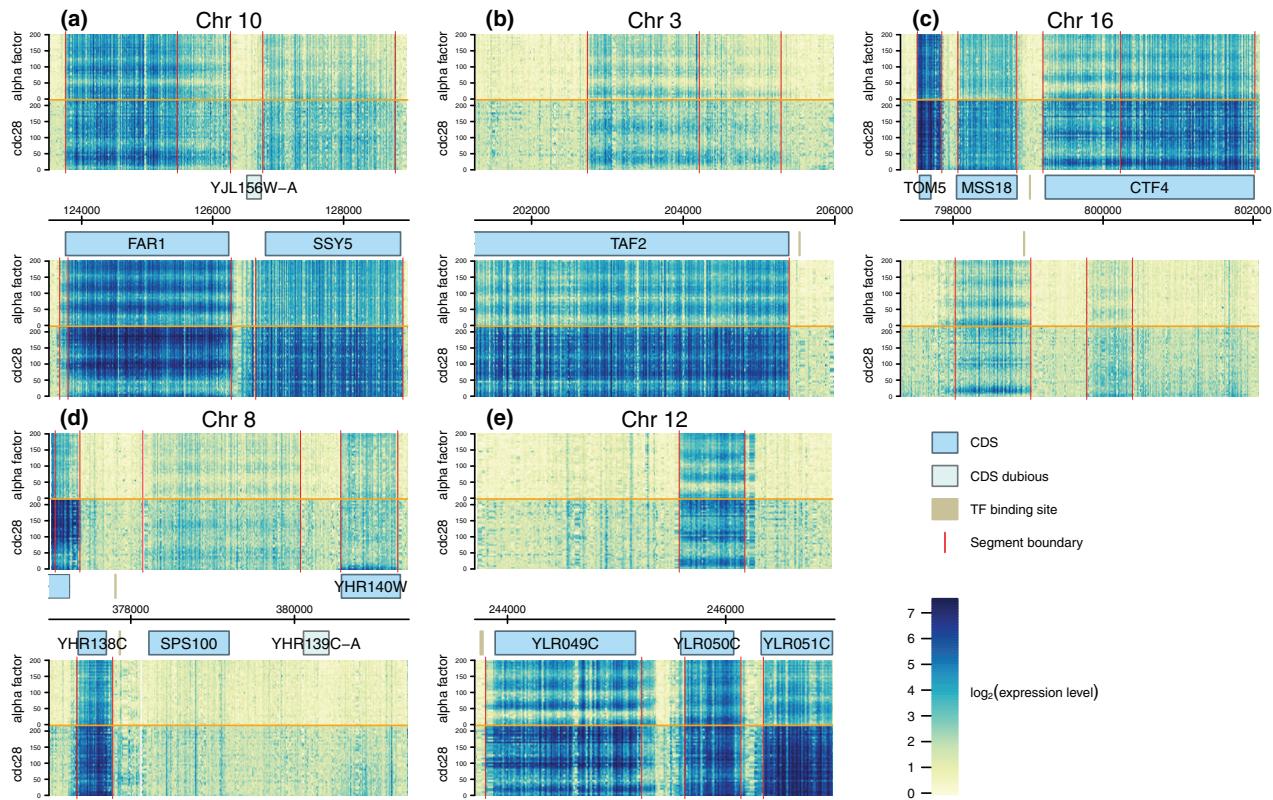


Figure 10: An example of the use of raster images in an R graphic (reproduced from Figure 3 of Granovskiaia et al., 2010, which was published as an open access article by Biomed Central).

Now, we use the map as a “mask” to set all pixels in the flag image to transparent wherever the map image is not black (demonstrating that assigning to subsets also works for “raster” objects).

```
> espanaRaster[mask != "black"] <- "transparent"
```

The flag image can now be used to fill a map of Spain, as shown by the following code (see Figure 12).

```
> par(mar=rep(0, 4))
> map(region="Spain")
> grid.raster(espanaRaster, y=1, just="top")
> map(region="Spain", add=TRUE)
```

Known issues

The raster image support has been implemented for the primary screen graphics devices that are distributed with R—Cairo (for Linux), Quartz (for Mac OS X), and Windows—plus the vector file format devices for PDF and PostScript. The screen device support also covers support for standard raster file formats (e.g., PNG) on each platform.

The X11 device has basic raster support, but rotated images can be problematic and there is no support for transparent pixels in the image. The Win-

dows device does not support images with a different alpha level per pixel.¹

There is no support for raster images for the XFig or PCTEX devices.

A web page on the R developer web site, <http://developer.r-project.org/Raster/raster-RFC.html>, will be maintained to show the ongoing state of raster support.

Summary

The R graphics engine now has native support for rendering raster images. This will improve rendering speed and memory efficiency for plots that contain large raster images. It also broadens the set of graphical effects that are possible (or convenient) with R.

Acknowledgements

Thanks to the editors and reviewers for helpful comments and suggestions that have significantly improved this article.

¹However, Brian Ripley has added support in the development version of R.



Figure 12: A raster image of the Spanish flag being used as a fill pattern for a map of Spain. On the left is a map of spain filled in black (produced by the `map()` function from the `maps` package). In the middle is a PNG image of Spanish flag (a public domain image from Wikimedia Commons, http://en.wikipedia.org/wiki/File:Flag_of_Spain.svg), and on the right is the result of clipping the Spanish flag image using the map as a mask.

Bibliography

R. Bivand, F. Leisch, and M. Maechler. *pixmap: Bitmap Images (“Pixel Maps”)*, 2009. URL <http://CRAN.R-project.org/package=pixmap>. R package version 0.4-10.

S. Chiaretti, X. Li, R. Gentleman, A. Vitale, M. Vignetti, F. Mandelli, J. Ritz, and R. Foa. Gene expression profile of adult T-cell acute lymphocytic leukemia identifies distinct subsets of patients with different response to therapy and survival. *Blood*, 103(7):2771–2778, 2004.

P. Cock. Using R to draw a heatmap from microarray data, 2010. URL http://www2.warwick.ac.uk/fac/sci/moac/students/peter_cock/r/heatmap.

O. S. code by Richard A. Becker and A. R. W. R. version by Ray Brownrigg Enhancements by Thomas P Minka <surname@stat.cmu.edu>. *maps: Draw Geographical Maps*, 2009. URL <http://CRAN.R-project.org/package=maps>. R package version 2.1-0.

M. V. Granovskaia, L. M. Jensen, M. E. Ritchie, J. Toedling, Y. Ning, P. Bork, W. Huber, and L. M. Steinmetz. High-resolution transcription atlas of the mitotic cell cycle in budding yeast. *Genome Biology*, 11(3):R24, 2010. URL <http://genomebiology.com/2010/11/3/R24>.

M. Loecher. *ReadImages: Image Reading Module for R*, 2009. URL <http://CRAN.R-project.org/package=ReadImages>. R package version 0.1.3.1.

D. Sarkar. *lattice: Lattice Graphics*, 2010. URL <http://CRAN.R-project.org/package=lattice>. R package version 0.18-3.

*Paul Murrell
Department of Statistics
The University of Auckland
Private Bag 92019, Auckland
New Zealand
paul@stat.auckland.ac.nz*

Probabilistic Weather Forecasting in R

by Chris Fraley, Adrian Raftery, Tilmann Gneiting,
McLean Sloughter and Veronica Berrocal

Abstract This article describes two R packages for probabilistic weather forecasting, **ensembleBMA**, which offers ensemble postprocessing via Bayesian model averaging (BMA), and **ProbForecastGOP**, which implements the geostatistical output perturbation (GOP) method. BMA forecasting models use mixture distributions, in which each component corresponds to an ensemble member, and the form of the component distribution depends on the weather parameter (temperature, quantitative precipitation or wind speed). The model parameters are estimated from training data. The GOP technique uses geostatistical methods to produce probabilistic forecasts of entire weather fields for temperature or pressure, based on a single numerical forecast on a spatial grid. Both packages include functions for evaluating predictive performance, in addition to model fitting and forecasting.

Introduction

Over the past two decades, weather forecasting has experienced a paradigm shift towards probabilistic forecasts, which take the form of probability distributions over future weather quantities and events. Probabilistic forecasts allow for optimal decision making for many purposes, including air traffic control, ship routing, agriculture, electricity generation and weather-risk finance.

Up to the early 1990s, most weather forecasting was deterministic, meaning that only one “best” forecast was produced by a numerical model. The recent advent of ensemble prediction systems marks a radical change. An ensemble forecast consists of multiple numerical forecasts, each computed in a different way. Statistical postprocessing is then used to convert the ensemble into calibrated and sharp probabilistic forecasts (Gneiting and Raftery, 2005).

The ensembleBMA package

The **ensembleBMA** package (Fraley et al., 2010) offers statistical postprocessing of forecast ensembles via Bayesian model averaging (BMA). It provides functions for model fitting and forecasting with ensemble data that may include missing and/or exchangeable members. The modeling functions estimate BMA parameters from training data via the EM algorithm. Currently available options are normal mixture models (appropriate for temperature or

pressure), mixtures of gamma distributions (appropriate for wind speed), and Bernoulli-gamma mixtures with a point mass at zero (appropriate for quantitative precipitation). Functions for verification that assess predictive performance are also available.

The BMA approach to the postprocessing of ensemble forecasts was introduced by Raftery et al. (2005) and has been developed in Berrocal et al. (2007), Sloughter et al. (2007), Wilson et al. (2007), Fraley et al. (2010) and Sloughter et al. (2010). Detail on verification procedures can be found in Gneiting and Raftery (2007) and Gneiting et al. (2007).

"ensembleData" objects

Ensemble forecasting data for weather typically includes some or all of the following information:

- ensemble member forecasts
- initial date
- valid date
- forecast hour (prediction horizon)
- location (latitude, longitude, elevation)
- station and network identification

The *initial date* is the day and time at which initial conditions are provided to the numerical weather prediction model, to run forward the partial differential equations that produce the members of the forecast ensemble. The *forecast hour* is the prediction horizon or time between initial and valid dates. The ensemble member forecasts then are valid for the hour and day that correspond to the forecast hour ahead of the initial date. In all the examples and illustrations in this article, the prediction horizon is 48 hours.

For use with the **ensembleBMA** package, data must be organized into an "ensembleData" object that minimally includes the ensemble member forecasts. For model fitting and verification, the corresponding weather observations are also needed. Several of the model fitting functions can produce forecasting models over a sequence of dates, provided that the "ensembleData" are for a single prediction horizon. Attributes such as station and network identification, and latitude and longitude, may be useful for plotting and/or analysis but are not currently used in any of the modeling functions. The "ensembleData" object facilitates preservation of the data as a unit for use in processing by the package functions.

Here we illustrate the creation of an "ensembleData" object called `srtfData` that corresponds to the `srtf` data set of surface temperature (Berrocal et al., 2007):

```

data(srft)
members <- c("CMCG", "ETA", "GASP", "GFS",
            "JMA", "NGPS", "TCWB", "UKMO")
srftData <-
  ensembleData(forecasts = srft[,members],
    dates = srft$date,
    observations = srft$obs,
    latitude = srft$lat,
    longitude = srft$lon,
    forecastHour = 48)

```

The `dates` specification in an `"ensembleData"` object refers to the valid dates for the forecasts.

Specifying exchangeable members. Forecast ensembles may contain members that can be considered exchangeable (arising from the same generating mechanism, such as random perturbations of a given set of initial conditions) and for which the BMA parameters, such as weights and bias correction coefficients, should be the same. In `ensembleBMA`, exchangeability is specified by supplying a vector that represents the grouping of the ensemble members. The non-exchangeable groups consist of singleton members, while exchangeable members belong to the same group. See Fraley et al. (2010) for a detailed discussion.

Specifying dates. Functions that rely on the `chron` package (James and Hornik, 2010) are provided for converting to and from Julian dates. These functions check for proper format ('YYYYMMDD' or 'YYYYMMDDHH').

BMA forecasting

BMA generates full predictive probability density functions (PDFs) for future weather quantities. Examples of BMA predictive PDFs for temperature and precipitation are shown in Figure 1.

Surface temperature example. As an example, we fit a BMA normal mixture model for forecasts of surface temperature valid January 31, 2004, using the `srft` training data. The `"ensembleData"` object `srftData` created in the previous section is used to fit the predictive model, with a rolling training period of 25 days, excluding the two most recent days because of the 48 hour prediction horizon.

One of several options is to use the function `ensembleBMA` with the valid date(s) of interest as input to obtain the associated BMA fit(s):

```

srftFit <-
  ensembleBMA(srftData, dates = "2004013100",
              model = "normal", trainingDays = 25)

```

¹The package implements the original BMA method of Raftery et al. (2005) and Sloughter et al. (2007), in which there is a single, constant bias correction term over the whole domain. Model biases are likely to differ by location, and there are newer methods that account for this (Gel, 2007; Mass et al., 2008; Kleiber et al., in press).

When no dates are specified, a model fit is produced for each date for which there are sufficient training data for the desired rolling training period.

The BMA predictive PDFs can be plotted as follows, with Figure 1 showing an example:

```
plot(srftFit, srftData, dates = "2004013100")
```

This steps through each location on the given dates, plotting the corresponding BMA PDFs.

Alternatively, the modeling process for a single date can be separated into two steps: first extracting the training data, and then fitting the model directly using the `fitBMA` function. See Fraley et al. (2007) for an example. A limitation of this approach is that date information is not automatically retained.

Forecasting is often done on grids that cover an area of interest, rather than at station locations. The dataset `srftGrid` provides ensemble forecasts of surface temperature initialized on January 29, 2004 and valid for January 31, 2004 at grid locations in the same region as that of the `srft` stations.

Quantiles of the BMA predictive PDFs at the grid locations can be obtained with the function `quantileForecast`:

```
srftGridForc <- quantileForecast(srftFit,
                                   srftGridData, quantiles = c(.1, .5, .9))
```

Here `srftGridData` is an `"ensembleData"` object created from `srftGrid`, and `srftFit` provides a forecasting model for the corresponding date.¹ The forecast probability of temperatures below freezing at the grid locations can be computed with the `cdf` function, which evaluates the BMA cumulative distribution function:

```
probFreeze <- cdf(srftFit, srftGridData,
                    date = "2004013100", value = 273.15)
```

In the `srft` and `srftGrid` datasets, temperature is recorded in degrees Kelvin (K), so freezing occurs at 273.15 K.

These results can be displayed as image plots using the `plotProbcast` function, as shown in Figure 2, in which darker shades represent higher probabilities. The plots are made by binning values onto a plotting grid, which is the default in `plotProbcast`. Loading the `fields` (Furrer et al., 2010) and `maps` (Brownrigg and Minka, 2011) packages enables display of the country and state outlines, as well as a legend.

Precipitation example. The `prcpFit` dataset consists of the fitted BMA parameters for 48 hour ahead forecasts of daily precipitation accumulation (in hundredths of an inch) over the U.S. Pacific Northwest from December 12, 2002 through March 31, 2005, as described by Sloughter et al. (2007). The fitted models are Bernoulli-gamma mixtures with a point

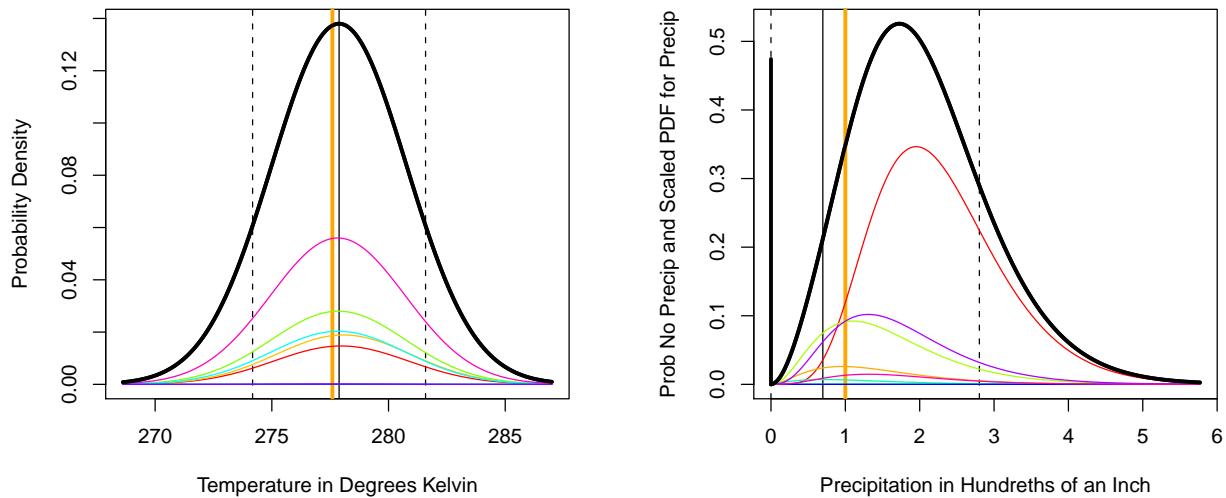


Figure 1: BMA predictive distributions for temperature (in degrees Kelvin) valid January 31, 2004 (left) and for precipitation (in hundredths of an inch) valid January 15, 2003 (right), at Port Angeles, Washington at 4PM local time, based on the eight-member University of Washington Mesoscale Ensemble (Grimit and Mass, 2002; Eckel and Mass, 2005). The thick black curve is the BMA PDF, while the colored curves are the weighted PDFs of the constituent ensemble members. The thin vertical black line is the median of the BMA PDF (occurs at or near the mode in the temperature plot), and the dashed vertical lines represent the 10th and 90th percentiles. The orange vertical line is at the verifying observation. In the precipitation plot (right), the thick vertical black line at zero shows the point mass probability of no precipitation (47%). The densities for positive precipitation amounts have been rescaled, so that the maximum of the thick black BMA PDF agrees with the probability of precipitation (53%).

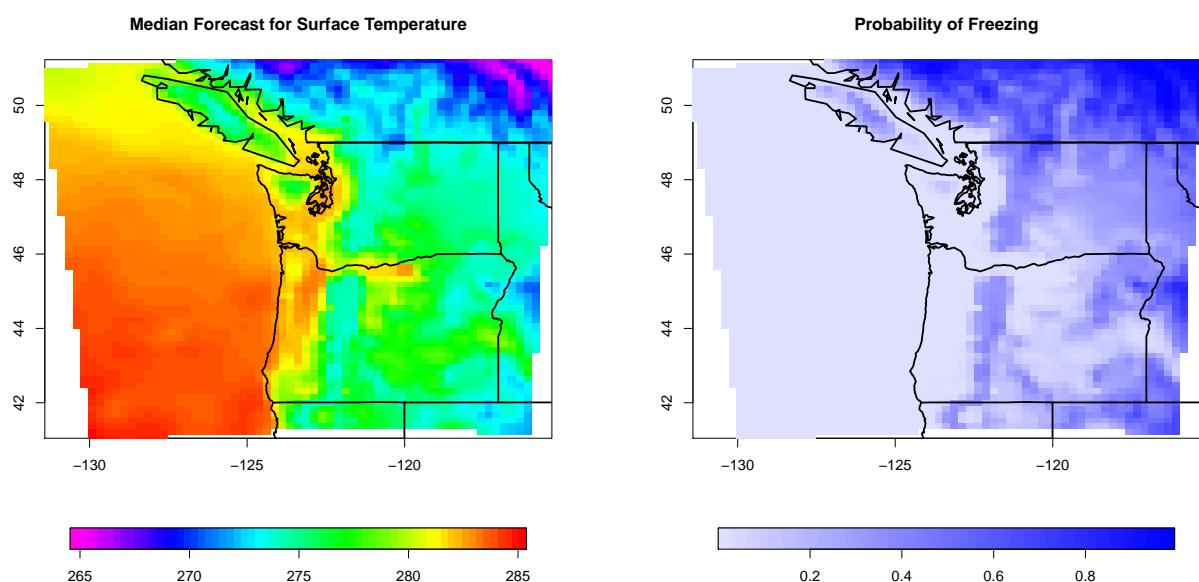


Figure 2: Image plots of the BMA median forecast for surface temperature and BMA probability of freezing over the Pacific Northwest, valid January 31, 2004.

mass at zero that apply to the cube root transformation of the ensemble forecasts and observed data. A rolling training period of 30 days is used. The dataset used to obtain `prcpFit` is not included in the package on account of its size. However, the corresponding "ensembleData" object can be constructed in the same way as illustrated for the surface temperature data, and the modeling process also is analogous, except that the "gamma0" model for quantitative precipitation is used in lieu of the "normal" model.

The `prcpGrid` dataset contains gridded ensemble forecasts of daily precipitation accumulation in the same region as that of `prcpFit`, initialized January 13, 2003 and valid January 15, 2003. The BMA median and upper bound (90th percentile) forecasts can be obtained and plotted as follows:

```
data(prcpFit)

prcpGridForc <- quantileForecast(
  prcpFit, prcpGridData, date = "20030115",
  q = c(0.5, 0.9))
```

Here `prcpGridData` is an "ensembleData" object created from the `prcpGrid` dataset. The 90th percentile plot is shown in Figure 3. The probability of precipitation and the probability of precipitation above .25 inches can be obtained as follows:

```
probPrecip <- 1 - cdf(prcpFit, prcpGridData,
  date = "20030115", values = c(0, 25))
```

The plot for the BMA forecast probability of precipitation accumulation exceeding .25 inches is also shown in Figure 3.

Verification

The `ensembleBMA` functions for verification can be used whenever observed weather conditions are available. Included are functions to compute verification rank and probability integral transform histograms, the mean absolute error, continuous ranked probability score, and Brier score.

Mean absolute error, continuous ranked probability score, and Brier score. In the previous section, we obtained a gridded BMA forecast of surface temperature valid January 31, 2004 from the `srft` data set. To obtain forecasts at station locations, we apply the function `quantileForecast` to the model fit `srftFit`:

```
srftForc <- quantileForecast(srftFit,
  srftData, quantiles = c(.1, .5, .9))
```

The BMA quantile forecasts can be plotted together with the observed weather conditions using the function `plotProbcast` as shown in Figure 4. Here the R core function `loess` was used to interpolate from the

station locations to a grid for surface plotting. It is also possible to request image or perspective plots, or contour plots.

The mean absolute error (MAE) and mean continuous ranked probability score (CRPS; e.g., Gneiting and Raftery, 2007) can be computed with the functions `CRPS` and `MAE`:

```
CRPS(srftFit, srftData)
# ensemble      BMA
# 1.945544 1.490496

MAE(srftFit, srftData)
# ensemble      BMA
# 2.164045 2.042603
```

The function `MAE` computes the mean absolute difference between the ensemble or BMA median forecast² and the observation. The BMA CRPS is obtained via Monte Carlo simulation and thus may vary slightly in replications. Here we compute these measures from forecasts valid on a single date; more typically, the CRPS and MAE would be computed from a range of dates and the corresponding predictive models.

Brier scores (see e.g., Jolliffe and Stephenson, 2003; Gneiting and Raftery, 2007) for probability forecasts of the binary event of exceeding an arbitrary precipitation threshold can be computed with the function `brierScore`.

Assessing calibration. Calibration refers to the statistical consistency between the predictive distributions and the observations (Gneiting et al., 2007). The *verification rank histogram* is used to assess calibration for an ensemble forecast, while the *probability integral transform* (PIT) histogram assesses calibration for predictive PDFs, such as the BMA forecast distributions.

The verification rank histogram plots the rank of each observation relative to the combined set of the ensemble members and the observation. Thus, it equals one plus the number of ensemble members that are smaller than the observation. The histogram allows for the visual assessment of the calibration of the ensemble forecast (Hamill, 2001). If the observation and the ensemble members are exchangeable, all ranks are equally likely, and so deviations from uniformity suggest departures from calibration. We illustrate this with the `srft` dataset, starting at January 30, 2004:

```
use <- ensembleValidDates(srftData) >=
  "2004013000"

srftVerifRank <- verifRank(
  ensembleForecasts(srftData[use,]),
  ensembleVerifObs(srftData[use,]))
```

²Raftery et al. (2005) employ the BMA predictive mean rather than the predictive median.

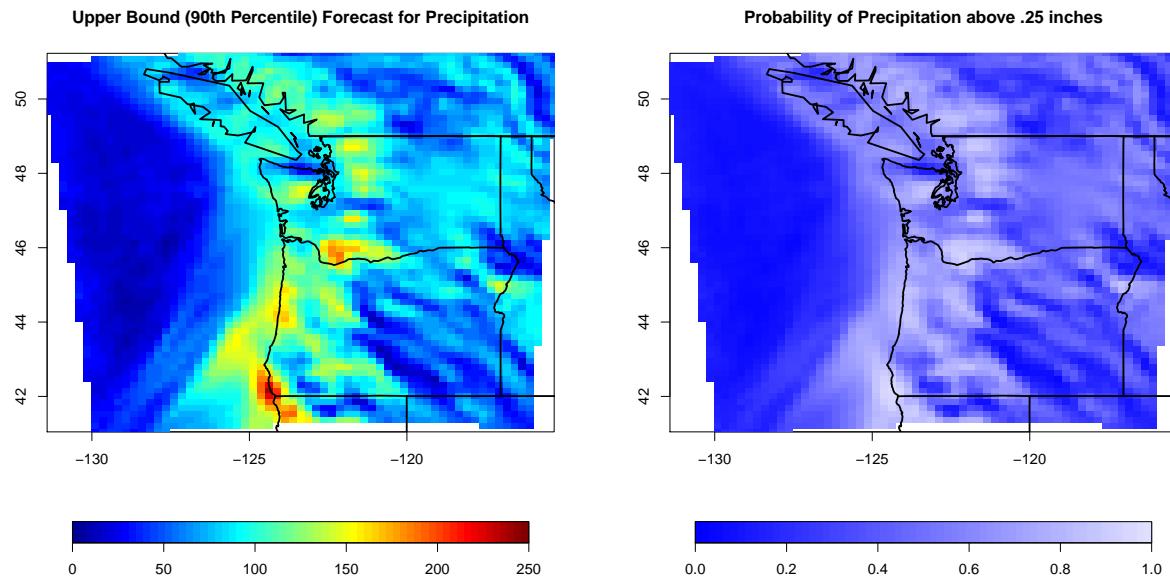


Figure 3: Image plots of the BMA upper bound (90th percentile) forecast of precipitation accumulation (in hundredths of an inch), and the BMA probability of precipitation exceeding .25 inches over the Pacific Northwest, valid January 15, 2003.

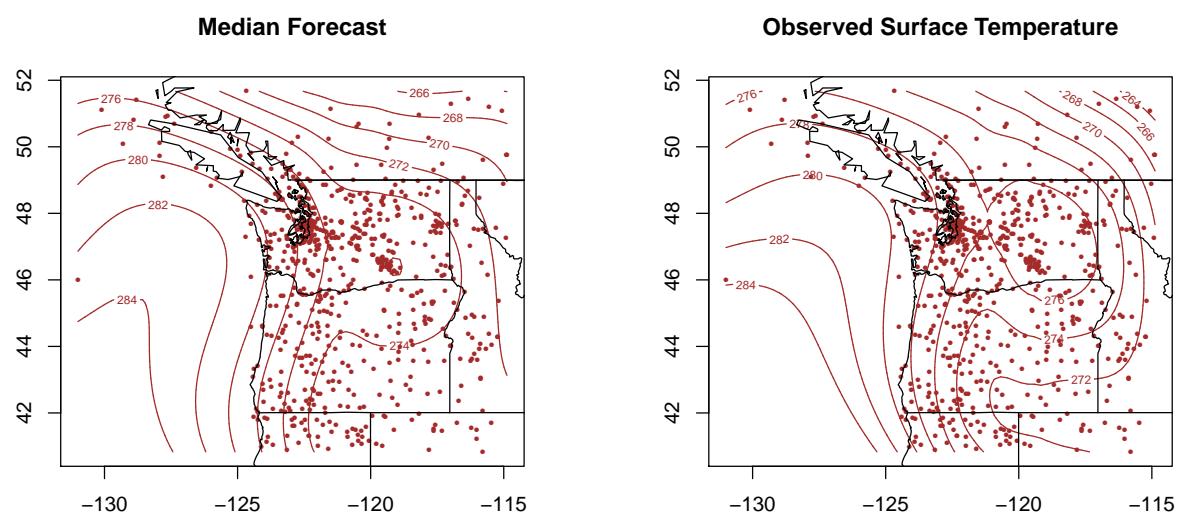


Figure 4: Contour plots of the BMA median forecast of surface temperature and verifying observations at station locations in the Pacific Northwest, valid January 31, 2004 (srft dataset). The plots use loess fits to the forecasts and observations at the station locations, which are interpolated to a plotting grid. The dots represent the 715 observation sites.

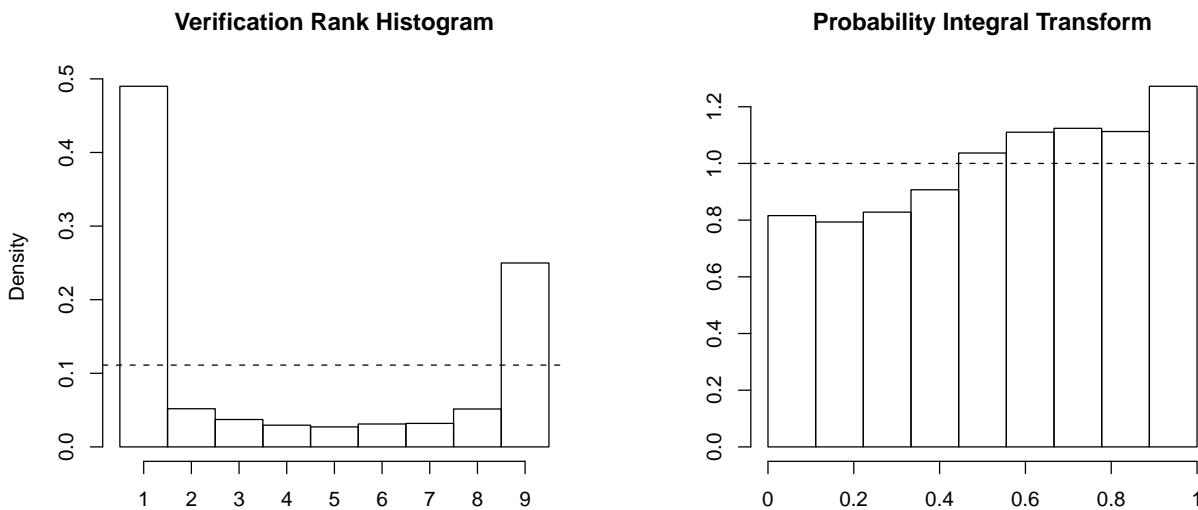


Figure 5: Verification rank histogram for ensemble forecasts, and PIT histogram for BMA forecast PDFs for surface temperature over the Pacific Northwest in the `srft` dataset valid from January 30, 2004 to February 28, 2004. More uniform histograms correspond to better calibration.

```
k <- ensembleSize(srftData)

hist(srftVerifRank, breaks = 0:(k+1),
     prob = TRUE, xaxt = "n", xlab = "",
     main = "Verification Rank Histogram")
axis(1, at = seq(.5, to = k+.5, by = 1),
     labels = 1:(k+1))
abline(h=1/(ensembleSize(srftData)+1), lty=2)
```

The resulting rank histogram composites ranks spatially and is shown in Figure 5. The U shape indicates a lack of calibration, in that the ensemble forecast is underdispersed.

The PIT is the value that the predictive cumulative distribution function attains at the observation, and is a continuous analog of the verification rank. The function `pit` computes it. The PIT histogram allows for the visual assessment of calibration and is interpreted in the same way as the verification rank histogram. We illustrate this on BMA forecasts of surface temperature obtained for the entire `srft` data set using a 25 day training period (forecasts begin on January 30, 2004 and end on February 28, 2004):

```
srftFitALL <- ensembleBMA(srftData,
                           trainingDays = 25)

srftPIT <- pit(srftFitALL, srftData)

hist(srftPIT, breaks = (0:(k+1))/(k+1),
     xlab="", xaxt="n", prob = TRUE,
     main = "Probability Integral Transform")
axis(1, at = seq(0, to = 1, by = .2),
     labels = (0:5)/5)
abline(h = 1, lty = 2)
```

The resulting PIT histogram is shown in Figure 5. It shows signs of negative bias, which is not surprising because it is based on only about a month of verifying data. We generally recommend computing the PIT histogram for longer periods, ideally at least a year, to avoid its being dominated by short-term and seasonal effects.

The ProbForecastGOP package

The **ProbForecastGOP** package (Berrocal et al., 2010) generates probabilistic forecasts of entire weather fields using the geostatistical output perturbation (GOP) method of Gel et al. (2004). The package contains functions for the GOP method, a wrapper function named `ProbForecastGOP`, and a plotting utility named `plotfields`. More detailed information can be found in the PDF document installed in the package directory at ‘`ProbForecastGOP/docs/vignette.pdf`’ or in the help files.

The GOP method uses geostatistical methods to produce probabilistic forecasts of entire weather fields for temperature and pressure, based on a single numerical weather forecast on a spatial grid. The method involves three steps:

- an initial step in which linear regression is used for bias correction, and the empirical variogram of the forecast errors is computed;
- an estimation step in which the weighted least squares method is used to fit a parametric model to the empirical variogram; and
- a forecasting step in which a statistical ensemble of weather field forecasts is generated,

by simulating realizations of the error field from the fitted geostatistical model, and adding them to the bias-corrected numerical forecast field.

Empirical variogram

In the first step of the GOP method, the empirical variogram of the forecast errors is found. Variograms are used in spatial statistics to characterize variability in spatial data. The empirical variogram plots one-half the mean squared difference between paired observations against the distance separating the corresponding sites. Distances are usually binned into intervals, whose midpoints are used to represent classes.

In **ProbForecastGOP**, four functions compute empirical variograms. Two of them, `avg.variog` and `avg.variog.dir`, composite forecast errors over temporal replications, while the other two, `Emp.variog` and `EmpDir.variog`, average forecast errors temporally, and only then compute variograms. Alternatively, one can use the wrapper function `ProbForecastGOP` with argument `out = "VARIOG"`.

Parameter estimation

The second step in the GOP method consists of fitting a parametric model to the empirical variogram of the forecast errors. This is done by the `Variog.fit` function using the weighted least squares approach. Alternatively, the wrapper function `ProbForecastGOP` with entry `out` set equal to "FIT" can be employed. Figure 6 illustrates the result for forecasts of surface temperature over the Pacific Northwest in January through June 2000.

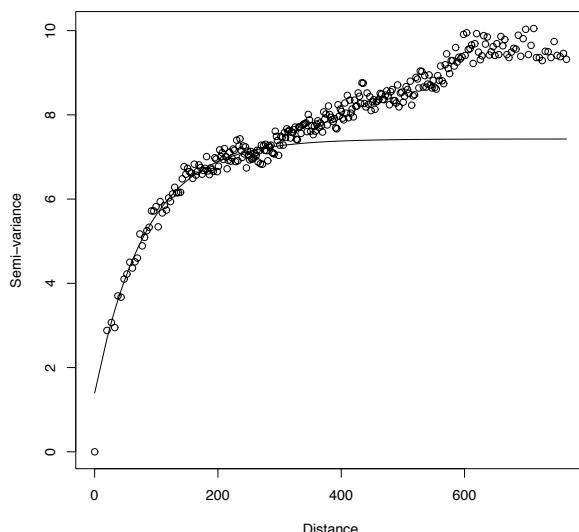


Figure 6: Empirical variogram of temperature forecast errors and fitted exponential variogram model.

The parametric models implemented in `Variog.fit` are the exponential, spherical, Gaussian, generalized Cauchy and Whittle-Matérn, with further detail available in the help files. The function `linesmodel` computes these parametric models.

Generating ensemble members

The final step in the GOP method involves generating multiple realizations of the forecast weather field. Each realization provides a member of a statistical ensemble of weather field forecasts, and is obtained by simulating a sample path of the fitted error field, and adding it to the bias-adjusted numerical forecast field.

This is achieved by the function `Field.sim`, or by the wrapper function `ProbForecastGOP` with the entry `out` set equal to "SIM". Both options depend on the `GaussRF` function in the **RandomFields** package that simulates Gaussian random fields (Schlather, 2011).

The output of the functions is both numerical and graphical, in that they return members of the forecast field ensemble, quantile forecasts at individual locations, and plots of the ensemble members and quantile forecasts. The plots are created using the `image.plot`, `US` and `world` functions in the **fields** package (Furrer et al., 2010). As an illustration, Figure 7 shows a member of a forecast field ensemble of surface temperature over the Pacific Northwest valid January 12, 2002.

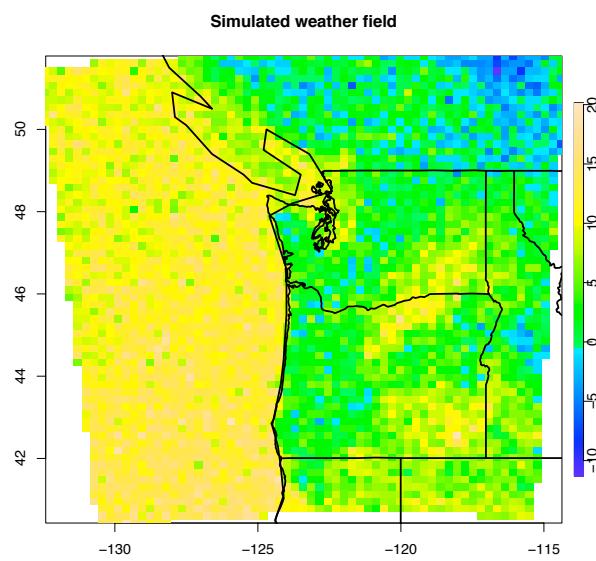


Figure 7: A member of a 48-hour ahead forecast field ensemble of surface temperature (in degrees Celsius) over the Pacific Northwest valid January 12, 2002 at 4PM local time.

Summary

We have described two packages, **ensembleBMA** and **ProbForecastGOP**, for probabilistic weather forecasting. Both packages provide functionality to fit forecasting models to training data.

In **ensembleBMA**, parametric mixture models, in which the components correspond to the members of an ensemble, are fit to a training set of ensemble forecasts, in a form that depends on the weather parameter. These models are then used to postprocess ensemble forecasts at station or grid locations.

In **ProbForecastGOP**, a parametric model is fit to the empirical variogram of the forecast errors from a single member, and forecasts are obtained by simulating realizations of the fitted error field and adding them to the bias-adjusted numerical forecast field. The resulting probabilistic forecast is for an entire weather field, and shows physically realistic spatial features.

Supplementing model fitting and forecasting, both packages provide functionality for verification, allowing the quality of the forecasts produced to be assessed.

Acknowledgements

This research was supported by the DoD Multidisciplinary University Research Initiative (MURI) program administered by the Office of Naval Research under Grant N00014-01-10745, by the National Science Foundation under grants ATM-0724721 and DMS-0706745, and by the Joint Ensemble Forecasting System (JEFS) under subcontract S06-47225 from the University Corporation for Atmospheric Research (UCAR). We are indebted to Cliff Mass and Jeff Baars of the University of Washington Department of Atmospheric Sciences for sharing insights, expertise and data, and thank Michael Scheuerer for comments. This paper also benefitted from the critiques of two anonymous reviewers and the associate editor.

Bibliography

- V. J. Berrocal, Y. Gel, A. E. Raftery, and T. Gneiting. *ProbForecastGOP: Probabilistic weather field forecast using the GOP method*, 2010. URL <http://CRAN.R-project.org/package=ProbForecastGOP>. R package version 1.3.2.
- V. J. Berrocal, A. E. Raftery, and T. Gneiting. Combining spatial statistical and ensemble information in probabilistic weather forecasts. *Monthly Weather Review*, 135:1386–1402, 2007.
- R. Brownrigg and T. P. Minka. *maps: Draw geographical maps*, 2011. URL <http://CRAN.R-project.org/>
- package=maps.** R package version 2.1-6. S original by Richard A. Becker and Allan R. Wilks, R port by Ray Brownrigg, enhancements by Thomas P. Minka.
- F. A. Eckel and C. F. Mass. Effective mesoscale, short-range ensemble forecasting. *Weather and Forecasting*, 20:328–350, 2005.
- C. Fraley, A. E. Raftery, T. Gneiting, and J. M. Sloughter. *ensembleBMA: Probabilistic forecasting using ensembles and Bayesian Model Averaging*, 2010. URL <http://CRAN.R-project.org/package=ensembleBMA>. R package version 4.5.
- C. Fraley, A. E. Raftery, T. Gneiting, and J. M. Sloughter. *ensembleBMA : An R package for probabilistic forecasting using ensembles and Bayesian model averaging*. Technical Report 516, University of Washington, Department of Statistics, 2007. (revised 2011).
- C. Fraley, A. E. Raftery, and T. Gneiting. Calibrating multi-model forecasting ensembles with exchangeable and missing members using Bayesian model averaging. *Monthly Weather Review*, 138: 190–202, 2010.
- R. Furrer, D. Nychka, and S. Sain. *fields: Tools for spatial data*, 2010. URL <http://CRAN.R-project.org/package=fields>. R package version 6.3.
- Y. Gel, A. E. Raftery, and T. Gneiting. Calibrated probabilistic mesoscale weather field forecasting: the Geostatistical Output Perturbation (GOP) method (with discussion). *Journal of the American Statistical Association*, 99:575–590, 2004.
- Y. R. Gel. Comparative analysis of the local observation-based (LOB) method and the non-parametric regression-based method for gridded bias correction in mesoscale weather forecasting. *Weather and Forecasting*, 22:1243–1256, 2007.
- T. Gneiting and A. E. Raftery. Weather forecasting with ensemble methods. *Science*, 310:248–249, 2005.
- T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102:359–378, 2007.
- T. Gneiting, F. Balabdaoui, and A. E. Raftery. Probabilistic forecasts, calibration, and sharpness. *Journal of the Royal Statistical Society, Series B*, 69:243–268, 2007.
- E. P. Grimit and C. F. Mass. Initial results for a mesoscale short-range ensemble forecasting system over the Pacific Northwest. *Weather and Forecasting*, 17:192–205, 2002.
- T. M. Hamill. Interpretation of rank histograms for verifying ensemble forecasts. *Monthly Weather Review*, 129:550–560, 2001.

- D. James and K. Hornik. *chron: Chronological objects which can handle dates and times*, 2010. URL <http://CRAN.R-project.org/package=chron>. R package version 2.3-39. S original by David James, R port by Kurt Hornik.
- I. T. Jolliffe and D. B. Stephenson, editors. *Forecast Verification: A Practitioner's Guide in Atmospheric Science*. Wiley, 2003.
- W. Kleiber, A. E. Raftery, J. Baars, T. Gneiting, C. Mass, and E. P. Grimit. Locally calibrated probabilistic temperature forecasting using geostatistical model averaging and local Bayesian model averaging. *Monthly Weather Review*, 2011 (in press).
- C. F. Mass, J. Baars, G. Wedam, E. P. Grimit, and R. Steed. Removal of systematic model bias on a model grid. *Weather and Forecasting*, 23:438–459, 2008.
- A. E. Raftery, T. Gneiting, F. Balabdaoui, and M. Polakowski. Using Bayesian model averaging to calibrate forecast ensembles. *Monthly Weather Review*, 133:1155–1174, 2005.
- M. Schlather. *RandomFields: Simulation and analysis tools for random fields*, 2011. URL <http://CRAN.R-project.org/package=RandomFields>. R package version 2.0.45.
- J. M. Sloughter, A. E. Raftery, T. Gneiting, and C. Fraley. Probabilistic quantitative precipitation forecasting using Bayesian model averaging. *Monthly Weather Review*, 135:3209–3220, 2007.
- J. M. Sloughter, T. Gneiting, and A. E. Raftery. Probabilistic wind speed forecasting using ensembles and Bayesian model averaging. *Journal of the American Statistical Association*, 105:25–35, 2010.
- L. J. Wilson, S. Beauregard, A. E. Raftery, and R. Verret. Calibrated surface temperature forecasts from the Canadian ensemble prediction system using Bayesian model averaging. *Monthly Weather Review*, 135:1364–1385, 2007.

Chris Fraley

Department of Statistics

University of Washington

Seattle, WA 98195-4322 USA

fraley@stat.washington.edu

Adrian Raftery

Department of Statistics

University of Washington

Seattle, WA 98195-4322 USA

raftery@u.washington.edu

Tilmann Gneiting

Institute of Applied Mathematics

Heidelberg University

69120 Heidelberg, Germany

t.gneiting@uni-heidelberg.de

McLean Sloughter

Department of Mathematics

Seattle University

Seattle, WA 98122 USA

sloughtj@seattleu.edu

Veronica Berrocal

School of Public Health

University of Michigan

1415 Washington Heights

Ann Arbor, MI 48109-2029 USA

berrocal@umich.edu

Analyzing an Electronic Limit Order Book

by David Kane, Andrew Liu, and Khanh Nguyen

Abstract The **orderbook** package provides facilities for exploring and visualizing the data associated with an order book: the electronic collection of the outstanding limit orders for a financial instrument. This article provides an overview of the **orderbook** package and examples of its use.

Introduction

The **orderbook** package provides facilities for exploring and visualizing the data associated with an order book: the electronic collection of the outstanding limit orders for a financial instrument, e.g. a stock. A *limit order* is an order to buy or sell a given quantity of stock at a specified limit price or better. The *size* is the number of shares to be bought or sold. An order remains in the order book until fully executed, i.e. until its size is zero as a result of trades. Partial executions occur as a result of trades for less than the entire size of the order.

Consider a simple order book containing five limit orders: sell 150 shares of IBM at \$11.11, sell 150 shares of IBM at \$11.08, buy 100 shares of IBM at \$11.05, buy 200 shares of IBM at \$11.05, and buy 200 shares of IBM at \$11.01.

	Price	Ask Size
	\$11.11	150
	\$11.08	100
300	\$11.05	
200	\$11.01	

Bid Size Price

Orders on the *bid* (*ask*) side represent orders to buy (sell). The price levels are \$11.11, \$11.08, \$11.05, and \$11.01. The *best bid* at \$11.05 (highest bid price) and the *best ask* at \$11.08 (lowest ask price) make up the *inside market*. The *spread* (\$0.03) is the difference between the best bid and best ask. The *midpoint* (\$11.065) is the average of the best bid and best ask.

There are four types of messages that traders can submit to an order book: *add*, *cancel*, *cancel/replace*, and *market order*. A trader can *add* a limit order in to the order book. She can also *cancel* an order and remove it from the order book. If a trader wants to reduce the size of her order, she can issue a *cancel/replace*, which cancels the order, then immediately replaces it with another order at the same price, but with a lower size. Every limit order is assigned a unique ID so that cancel and

cancel/replace orders can identify the corresponding limit order. A *market order* is an order to immediately buy or sell a quantity of stock at the best available prices. A trade occurs when a market order “hits” a limit order on the other side of the inside market.

All orders have timestamps indicating the time at which they were accepted into the order book. The timestamp determines the *time priority* of an order. Earlier orders are executed before later orders. For example, suppose that the order to buy 100 shares at \$11.05 was submitted before the order to buy 200 shares at \$11.05. Now suppose a market order selling 200 shares is submitted to the order book. The limit order for 100 shares will be executed because it is at the front of the queue at the best bid. Then, 100 shares of the order with 200 total shares will be executed, since it was second in the queue. 100 shares of the 200 share order remain in the order book at \$11.05.

A market order for more shares than the size at the inside market will execute at worse price levels until it is complete. For example, if a market order to buy 200 shares is submitted to the order book, the order at \$11.08 will be fully executed. Since there are no more shares available at that price level, 100 shares at the \$11.11 price level will be transacted to complete the market order. An order to sell 50 shares at \$11.11 will remain in the order book. Executing these two market orders (a sell of 200 shares and a buy of 200 shares) on our hypothetical order book results in a new state for the order book.

	Price	Ask Size
	\$11.11	50
100	\$11.05	
200	\$11.01	

Bid Size Price

Note that cancel/replace orders can lower the size of an order, but not increase it. Cancel/replace orders maintain the time priority of the original order, so if size increases were allowed, traders with orders at the highest time priority for a price level could perpetually increase the size of their order, preventing others from being able to transact stock using limit orders at that price level. See Johnson (2010) for more details on the order book.

Example

NVIDIA is a graphics processing unit and chipset developer with ticker symbol NVDA. Consider the order book for NVDA at a leading electronic exchange

on June 8, 2010. We create the `orderbook` object by specifying the location of our data file.

```
> library(orderbook)
> filename <- system.file("extdata",
+                         "sample.txt",
+                         package = "orderbook")
> ob <- orderbook(file = filename)
> ob <- read.orders(ob, 10000)
> ob
```

An object of class `orderbook` (default)

```
-----
```

Current orderbook time:	09:35:02
Message Index:	10,000
Bid Orders:	631
Ask Orders:	1,856
Total Orders:	2,487

We read in the first 10,000 messages then show the object. The current time is 9:35:02 AM. This is the time of the last message read. The message index indicates which row in the data file the object has read through. The display also shows that there are 631 bids and 1,856 asks outstanding, for a total of 2,487 orders. This indicates that many earlier orders have been removed through either cancels or trades.

```
> summary(ob)
```

Current time is 09:35:02

```
Ask price levels: 540
Bid price levels: 179
Total price levels: 719
```

```
-----
```

Ask orders:	1,856
Bid orders:	631
Total orders:	2,487

```
-----
```

Spread:	0.02
---------	------

```
-----
```

Mid point:	11.37
------------	-------

```
-----
```

Inside market

Best Bid:	11.36
Size:	2,700

Best Ask:	11.38
Size:	400

Using `summary` the total order information from `show` is repeated. Additionally, we see that there are 540 ask and 179 bid price levels, for a total of 719. This indicates that many orders have been submitted at the same price level. The spread is \$0.02, and the midpoint is \$11.37. The inside market is composed of 2,700 shares offered at the best bid of \$11.36 and 400 shares offered at the best ask of \$11.38.

```
> display(ob)
```

Current time is 09:35:02

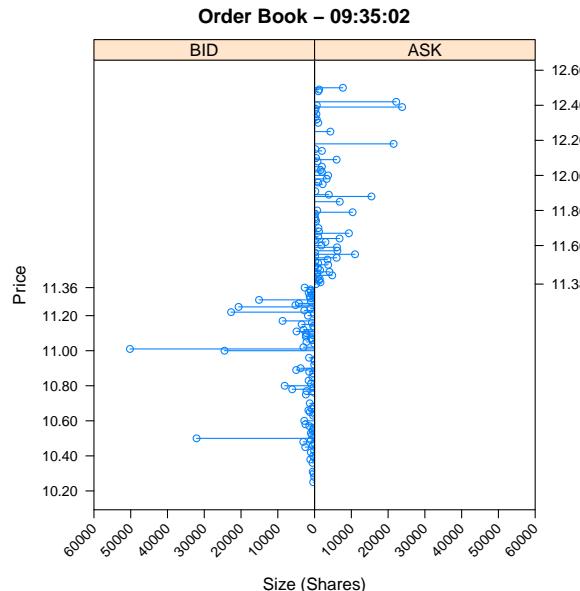
	Price	Ask Size
	11.42	900
	11.41	1,400
	11.40	1,205
	11.39	1,600
	11.38	400

	Price
2,700	11.36
1,100	11.35
1,100	11.34
1,600	11.33
700	11.32

Bid Size	Price
----------	-------

`display` shows the inside market, along with the four next best bid and ask price levels and the size at each price level.

```
> plot(ob)
```



`plot` is a graphical representation of `display`. Price levels are on the y-axis, and size is on the x-axis. The maximum and minimum price levels displayed by default are 10% above and below the midpoint. Note the large number of shares at \$11.01. It is helpful to know the number of orders which make up the large size at that price level. Using the "[" method we can view the order information at particular price levels.

```
> ob["11.01"]
  price size type      time      id
1 11.01 109   BID 34220988 4403084
2 11.01 50000  BID 34220988 4403085
3 11.01 100   BID 34220988 4403086
```

There is an order for 50,000 shares at the \$11.01 price level that accounts for almost all of the size. We can view a plot of the number of orders rather than the number of shares at each price level by specifying `type = 'o'` when using `plot`. In the previous plot the maximum and minimum price levels were 10% off from the midpoint, but for this plot we specify a range of only 3.3%.

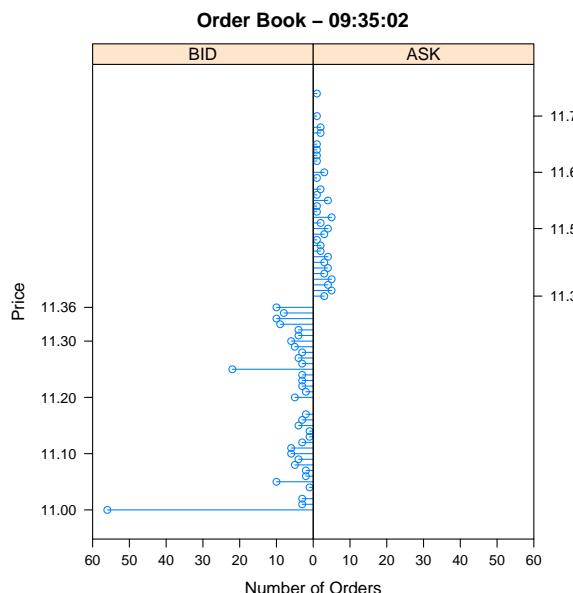
Note the large number of orders at \$11.00. The `"["` method returns a `data.frame`, so we can use `nrow` to return the number of orders at \$11.00.

```
> nrow(ob["11.00"])
```

```
[1] 56
```

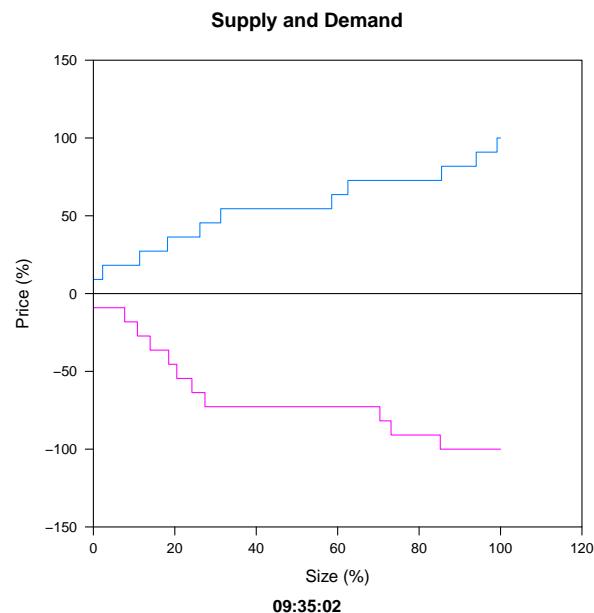
There are 56 orders at that price level, which confirms what we see in the plot.

```
> plot(ob, bounds = 0.033, type = 'o')
```



The `type` argument on `plot` allows for an "sd" option which shows supply and demand curves for the order book. The demand (supply) curve is downsloping (upsloping). This is because more people want to buy (sell) a stock when the price decreases (increases). The ask (bid) prices are normalized by the absolute value of the difference between the highest (lowest) plotted ask (bid) price level and the the midpoint. Following Cao et al. (2009), the sizes are normalized by the sum of the sizes across all plotted price levels for each side.

```
> plot(ob, bounds = 0.01, type = "sd")
```



`orderbook` has methods for creating new `orderbook` objects at specified clock times of interest. `read.time` returns an `orderbook` object that has read all messages before the specified time. For example, this returns the `orderbook` object at 9:30:00.

```
> ob <- read.time(ob, "9:30:00")
```

`read.orders` is used to move forwards or backwards in the order book by a specified number of messages. In this case, an `orderbook` object at 50 messages before the current message is returned.

```
> ob <- read.orders(ob, n = -50)
> ob
```

```
An object of class orderbook (default)
-----
Current orderbook time: 09:28:41
Message Index: 292
Bid Orders: 72
Ask Orders: 81
Total Orders: 153
```

Data

Data files should contain all messages for one stock on a single trading day. Most brokers and exchanges have their own format for transmitting raw message data to customers, so it would be unfeasible for us to write scripts to automatically process all data formats. Consequently, raw data for an `orderbook` object must be in the following form:

```
type,time,id,price,size,type,status
A,34226539,5920814,25.95,100,ASK,TRUE
A,34226788,5933949,25.91,100,BID,FALSE
R,34226900,5933949,50
C,34226904,5920814
T,34226904,755377,25.95,100,TRUE
```

where A, R, T, and C mean Add, Replace, Trade, and Cancel, respectively. The second column is the timestamp of the message in milliseconds after midnight, and the third column is the order ID. For a Replace the next column is the new size, while for Add and Trade a column for price comes before the size column. Add messages also have the type of order (BID/ASK) in the sixth column. The optional seventh (sixth) column is TRUE if the order (trade) belongs to the user, and FALSE otherwise. This allows the user to create plots that show the time priority of his own orders. If the column is omitted, the first line of the data file should be type, time, id, price, size, type and not include status.

In this example a user order to sell 100 shares at \$25.95 is added to the order book, followed by an order to buy 100 shares at \$25.91. The size of the order at \$25.91 is then replaced to 50 shares. Finally, the order at \$25.95 is cancelled, and a trade for 100 shares at \$25.95 occurs.

Analyzing trades

A user can create plots that show the time priority of his own orders if a status column is present in the data file.

```
> filename <- system.file("extdata",
+                           "tradersample.txt",
+                           package = "orderbook")
> ob <- orderbook(file = filename)
> ob <- read.time(ob, "9:30:05")
> ob <- next.trade(ob)
> ob
```

An object of class orderbook (trader)

```
-----
```

Current orderbook time:	09:30:05
Message Index:	6,062
Bid Orders:	164
Ask Orders:	252
Total Orders:	416

Note that this orderbook object is of type trader. The next.trade function sets the state of the order book to when the trade after the current time occurs. There is also a previous.trade function with the same functionality moving backwards

```
> view.trade(ob, tradenum = 584)
```

```
trade 584
row      6063
time    09:30:05
id       636783
price   25.94
size     1000
status   FALSE

> mid.point(ob)

price
25.935
```

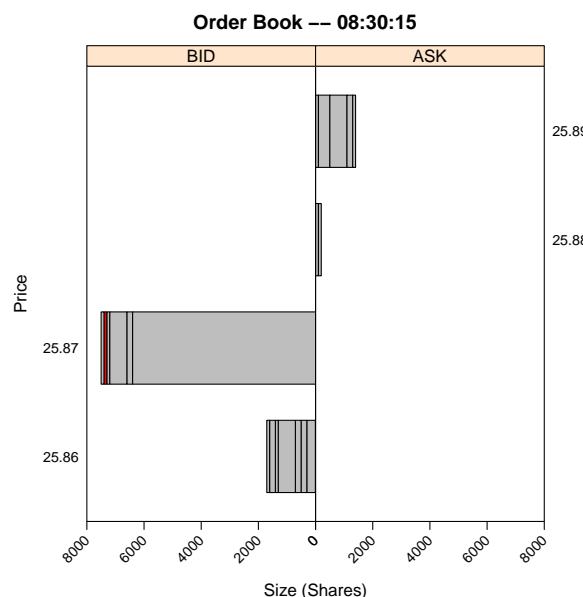
Since the trade price is higher than the midpoint price, we know that the trade occurred as a result of an ask order getting hit. Note that trade data is stored into the order book only after it has read through the corresponding trade message.

```
> midpoint.return(ob, tradenum = 584, time = 10)

midpoint.return
10 second          0.065
```

The *midpoint return* is the difference in cents between the execution price and the midpoint price after a specified period of time. For example, the above calculates the ten second midpoint return for the first trade. Since it was a sell order, the midpoint return will be positive if the stock price decreases, and negative if the stock price increases.

```
> ob <- read.time(ob, "9:30:15")
> plot(ob, type = "t", bounds = 0.02)
```



This plot shows two pennies above and below the best bid and best ask. We see that the midpoint has dropped to 25.875, confirming the midpoint return above. This graph shows two pennies above and below the best bid and ask. Orders at these price levels are shown in time priority, with the earliest submitted order being closest to the middle y-axis. Note the red order--this is an order marked TRUE by the user, indicating that it belonged to him.

Simulation

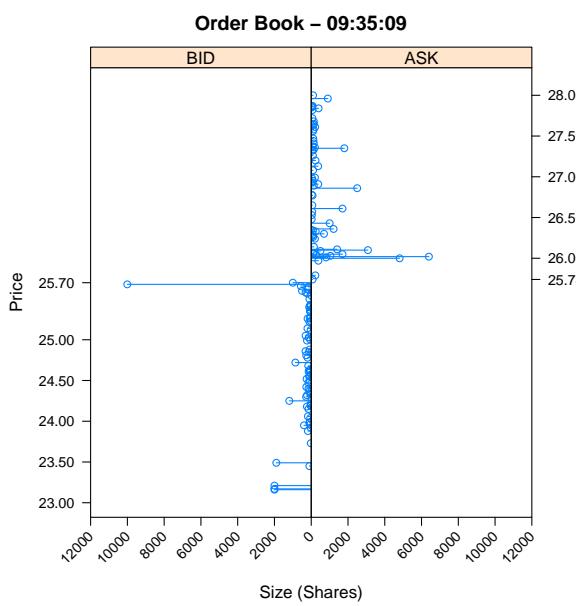
Simulating and modelling the intraday decisions of traders is a topic of active research in behavioral finance and economics. **orderbook** supports adding, replacing, and cancelling orders. Add orders require the price, size, and type (ASK/BID) of the limit order. Time and ID are optional, and will default to the

maximum time + 1 and the maximum ID + 1. Replace messages require the new size and ID. Cancel orders only require ID. In addition, market orders can be issued to the order book. Market orders require size and side (BUY/SELL).

```
> ob <- add.order(ob, 11.20, 300, "ASK")
> ob <- remove.order(ob, 1231883)
> ob <- replace.order(ob, 1231883, 150)
> ob <- market.order(ob, 200, "BUY")
```

Using these tools, the user can write functions to simulate the an order book. In the following example, we consulted [Gilles \(2006\)](#). We simulate 1,000 messages. The messages are chosen based on the following probabilities: 50% for a cancel message, 20% for a market order, and 30% for a limit order. In the event of a cancel message the order cancelled is randomly chosen. Market order have a 50-50 chance for a buy or sell order. The size of the market order always corresponds to the size of the individual order at the best ask or bid with the highest time priority. Limit orders have a 50-50 chance to be an ask or bid. There is a 35% chance for the price of a limit order to be within the spread. If the price is outside of the spread, a price is chosen using a power law distribution. Finally, the size follows a log-normal distribution. A plot of this example simulation is shown below.

```
> ob <- simulate(ob)
> plot(ob)
```



[Gilles \(2006\)](#) used simulations to test the impact of liquidity on price level stabilization. He concluded that most price changes are the result of uninformed traders (who supply liquidity), rather than informed traders (who demand liquidity).

Conclusion

The **orderbook** package is part of a collection of packages (see [Campbell et al. \(2007\)](#) and [Kane and Enos \(2006\)](#)) for working with financial market data. R provides all the necessary tools for managing institutional sized portfolios.

*David Kane
Kane Capital Management
Cambridge, MA
USA
dave@kanecap.com*

*Andrew Liu
Williams College
Williamstown, MA
USA
Andrew.T.Liu@williams.edu*

*Khanh Nguyen
University Of Massachusetts at Boston
Boston, MA
USA
knguyen@cs.umb.edu*

Bibliography

- K. Campbell, J. Enos, D. Gerlanc, and D. Kane. Backtests. *R News*, 7(1):36–41, April 2007.
- C. Cao, O. Hansch, and X. Wang. The information content of an open limit-order book. *The Journal of Futures Markets*, 29(1):16–41, 2009.
- D. Gilles. *Asynchronous Simulations of a Limit Order Book*. PhD thesis, University of Manchester, 2006.
- B. Johnson. *Algorithmic Trading & DMA: An introduction to direct access trading strategies*. 4Myeloma Press, London, 2010.
- D. Kane and J. Enos. Analysing equity portfolios in R. *R News*, 6(2):13–19, May 2006.

Giving a useR! Talk

by Rob J. Hyndman

Abstract Giving a UseR! talk at the the international R user conference is a balancing act in which you have to try to impart some new ideas, provide sufficient background and keep the audience interested, all in a very short period of time.

I've sat through more than my fair share of bad conference talks. Slides full of equations flashing past quickly, tables containing tiny figures that no-one can read, most of the audience lost on the third slide. Anyone who has attended even one conference will have seen these examples and more.

What is the aim of the talk?

The problems often stem from confusion about the purpose of the talk. Some speakers clearly think the aim of a talk is to impress the audience with their technical skills, even (or especially) if that means the audience does not understand what they are talking about. Others speakers appear to believe that talks are for those few members of the audience who are working in the same narrow research area — and so no attempt is made to provide an introduction to the topic. Still others see conference talks as an oral version of an academic paper, in all its painful detail.

Instead, I like to think of conference talks as advertisements for the associated paper or R package, or shared wisdom gained through valuable experience. The talk is not intended to cover everything you have done, or even to summarize what you have done. In giving a talk, I am hoping that (1) everyone in the audience will have a clear idea of what I have been working on; and (2) some of those in the audience will be motivated to read my paper, download the associated R package, or put into practice some of my advice.

These aims mean that I never bother with proofs or derivations — they are for the people who read the paper. Similarly, there is no point discussing the internals of R code or algorithm technicalities. Those who care will explore the details afterwards.

Instead, I tend to spend at least half the time going through a motivating example and reviewing the relevant background — most of the audience will need that context in order to understand what the talk is about. In fact, it is reasonable to assume that the audience knows about as much as you did at the start of your work in this area. That is, probably very little. So it is important to spend some time providing background information or you will lose the audience quickly. Do not assume the audience already

knows what you have spent long hours learning on your own.

Consider the context

For a useR! talk, there is the additional challenge of getting the right balance of code, technical and application detail. The appropriate mix depends on the session.

If you are giving a kaleidoscope talk (for a wide audience), you need to make a particular effort to make your talk accessible, keeping examples simple and minimising technical detail. Speakers in focus sessions should consider what other talks are in their session in order to determine how specialised the audience is likely to be (e.g., people at a high performance computing session are probably comfortable with technical details, but those at a session on ecology are likely to be more interested in the application detail.)

Looking at some related talks from previous years can be useful. Many talks from previous useR! conferences can be found on the conference websites (see <http://www.r-project.org/conferences.html> for links).

A suggested structure

I recommend the following structure for a conference talk:

- (a) Start with a motivating example demonstrating the problem you are trying to solve;
- (b) Explain existing approaches to the problem and their weaknesses;
- (c) Describe your main contributions;
- (d) Show how your ideas solve the problem/example you started with.

This structure will not necessarily work for every talk, but it is a good place to start. In particular, beginning with a motivating example is much better than setting up the problem algebraically.

For a 15 minute conference presentation, I divide the time approximately into 3/4/6/2 minute sections.

Using this structure, you will have barely started on your own contributions when you are half way through your allocated time. Resist the temptation to trim the first two sections. The audience needs time to absorb the purpose of your work and the context in which it is set.

Keeping to time

Do not deliver a 30-minute talk in 15 minutes. Nothing irritates an audience more than a rushed presentation. It is like trying to get a drink out of a fire hydrant. Your objective is to engage the audience and have them understand your message. Do not flood them with more than they can absorb.

Present only as much material as can reasonably fit into the allocated time. Generally that means no more than one slide per minute. I tend to use an average of about 0.8 slides per minute of talking. It is helpful to use some slides as time-markers and make sure you are at the relevant slide at the right time.

Never go over time. Keep an eye out for signals from the session chair indicating when you need to conclude. If necessary, be prepared to cut your talk short and finish with a quick summary.

Rehearsing is invaluable. Practise. Out loud. Standing up. Using a data projector. Get colleagues to listen to you, including some who are not knowledgeable on the topic of your talk; they will be able to point out places where you may not come across clearly. If several people in your research group are attending the same conference, get together beforehand and practice your talks to each other. Make such rehearsals as realistic as possible and time them. After the rehearsal, you may wish to delete some of your material to ensure the talk can be delivered within the allocated time.

Balance the amount of material you present with a reasonable pace of presentation. If you feel rushed when you practise, then you have too much material. Budget your time to take a minute or two less than your maximum allotment.

Preparing slides

High quality slides are an important part of a good presentation. I recommend using the beamer package with L^AT_EX. MS-Powerpoint is also popular, but it makes it much harder to format mathematical symbols and equations nicely.

Avoid distracting slide transitions and dazzling slide themes. You want the audience to focus on your content, not wonder how you implemented some gimmick. Save animation for aiding interpretation.

Use at least a 20-point font so everyone in the room can read your material. (The default font size in beamer is generally too small — use a 14pt font in beamer which is equivalent to 23pt on the screen.) Similarly, view your slides at 50% on screen to check that code and figures are legible. Unreadable material is worse than useless — it inspires a negative attitude by the audience to your work and, ultimately, to you. Many R users are near-sighted; do not make it any harder for them.

Limit the material on each slide, keeping the

number of words to a minimum. Do not include every detail of what you plan to say. Keep it simple. It is easy, but boring, to have bulleted lists summarizing your main points. Instead, use pictures and graphics as much as possible. Resort to text only when illustrations fail you.

If you must present tables, only show the essential information. No-one is going to read a slide full of tiny numbers, let alone absorb the information. Very often, a table can be replaced with a suitable graphic.

Give only the most necessary mathematical details. People not working in the research area can find equations difficult to follow as part of a rapidly delivered presentation. When you do need to use equations, define your notation.

Avoid showing too much R code. Trim it back to the bare minimum so the audience can focus on the essential details. Use different colours to clearly distinguish R code from output.

Slides are not there to remind you what to say — use a page of notes for that purpose. The slides are for the audience — make sure everything on your slides is there because it will help the audience understand what you are saying.

On your last slide, give your website or email address for people to contact you if they want to read the paper or download your R code or just ask a question.

It is useful to add slide numbers so the audience can refer back to specific slides in question time.

I spend a lot of time going over my slides looking for ways to improve them. After a presentation is essentially complete, I go through all the slides to see what I can remove — less text is better. I also look for places where I can simplify the presentation, where I can replace text with graphics, and where the titles can be improved. I often spend almost as much time refining the slides as in creating the first version.

Always preview your slides on the computer being used for the talk. You will look foolish if symbols and Greek letters that looked OK on your computer translate into something unreadable on the big screen. This is much more common if you use MS-Powerpoint as the fonts may not be embedded in the document and so equations lose important symbols.

Giving the presentation

By the time you give the talk, you will have spent enough time preparing your slides and practising your talk that you should feel confident of giving a great presentation.

At the conference, make sure you talk to the session chair beforehand so they are aware of who you are. Arrive at the meeting room 10 minutes before the session begins to take care of last-minute details.

Talk at a pace that everybody in the audience can

understand. Speak slowly, clearly, and loudly, especially if your English is heavily accented. Speak loudly enough to be easily heard by those sitting in the back row.

Engage the audience — speak to them, not to the projector screen or to your notes. It helps to move around, look at your audience and smile.

Never apologize for your slides. Make apologies unnecessary by producing great slides in the first place. Do not say, “I know you can’t see this, but ...” If the audience cannot read your slide, there is no point displaying it.

Do not apologize for incomplete results either. Researchers understand that all research is incom-

plete. Just present the results and let the audience judge. It is okay to say, “work is on-going”.

When finished, thank the audience for their attention. Stay for the entire session, for the courtesy and benefit of your audience and your co-speakers. Afterwards, be available for people to ask you questions.

Rob J Hyndman

Department of Econometrics & Business Statistics

Monash University

Australia

Rob.Hyndman@monash.edu

Tips for Presenting Your Work

by Dianne Cook

Abstract With the international R user conference, useR! 2011, approaching, many participants may be contemplating how to put their thoughts together for presentation. This paper provides some suggestions for giving presentations and making posters.

Some background

Just after completing a practice of the talk I planned to give about my PhD research in several upcoming academic job interviews, my advisor, Andreas Buja, sat me down, and completely re-drafted my talk! I had produced what I had seen many times presented by numerous speakers in weekly seminars at Rutgers University, slide after slide of details. Andreas explained that while this might be appropriate for a paper it is not the best approach for a talk. We laid out the key problem that my work addressed, and then put in a slide that simply said “*Stay tuned....*” The next few slides addressed the methodology, and the key answers to the problem came near the end of the talk.

The “*Stay tuned*” sticks with me.

Although, today this phrase might be something of a cliché, overused by the electronic media to keep your attention through another barrage of advertising, the effect is useful to re-create. What did this phrase do for the talk at the time? It allowed it to build interest in the topic early, and let the audience know they would be rewarded later. The middle of the talk also showed a VHS video of high-dimensional data being explored using various tours – computers that could display videos were not so portable in 1993! (That video is now available in Flash format at <http://stat-graphics.org/movies/grand-tour.html>.)

Key pointers

The web sites Schoeberl & Toon (2011) and van Marle (2007) have some useful tips on giving scientific presentations. Here are a few collected from my experiences.

- If your audience remembers and takes away one thing your talk is a success. What is this going to be for your talk?
- Make a plan, map out the start, with motivation, middle and finish. Sketch out the slides, and what you want to communicate with each slide.

- Avoid slide after slide of bullet points (the Pow-erpoint syndrome), or slide after slide of equations.
- Establish your credentials. For a traditional audience this might be a few equations, and the key parts to a proof. For the statistical computing and computational statistics audience show some code fragments – ones that may be particularly elegant, or illustrate the critical pieces. (Some colleagues and I have occasionally jested that watching programmer program might be more interesting than listening to some talks. This is only partially in jest because watching seasoned programmers tackling a computational problem can be very insightful.)
- Draw the audience in with *convention*, a usual format, style of slides, font, Familiarity is in-groupness, shows the audience that they are like you, that you are bringing them with you, and that you can be trusted.
- Use pictures, that is, plots (of data). BUT, BUT, please don't say “You can see from the picture that” without helping interpret with something like “because the points follow a curved pattern”. That is, explain the pattern in the graphical elements of the plot that allows one to make the analytical leap.
- Use pictures, or, cartoons, to help explain concepts.
- Use pictures, for visual stimulation. Not because they “are devices for showing the obvious to the ignorant” or to “prevent the dullards in the audience from falling asleep” (Tufte, 1990) but because graphics are beautiful. Plots evoke creativity.
- Tell a story.

Practicalities

Many statisticians use L^AT_EX, with Beamer style, to make the slides for their talk. This is convenient for equations, and makes elegant, beautifully type-set slides. It has a lot of flexibility to make colored text, different fonts, navigation strips, including figures and even animations or movies. My preference, however, is to use keynote on the Mac. It provides a lot more flexibility in the formatting, the ability to use wild fonts, such as my own handwriting, and seamless incorporation of movies and animations. To include equations I have a generic ‘tmp.tex’ file on

my computer with all the equations that I have ever used, and I cut and paste these out of the pdf from Preview. Keynote maintains the quality of the equations, and images, through resizing, unlike Powerpoint. It also, like TeX, keeps figures in separate files, actually the ‘slides.key’ might look like a file but is really a directory.

Just out of graduate school I would meticulously write down every word that I wanted to say in association with the talk. I would practice, and practice, and practice, then throw the notes away to actually give the talk. I still occasionally do this. Why? With a limited time frame, and under the glare of my colleagues, making language precise helps get the message across, which is respectful of the audience. Making the notes makes the language precise, but giving the talk without notes lends spontaneity.

Check the equipment. How does your font and the graphics appear from the back of the room? Does your computer work with the projector? Or does the provided computer have the software that you need for your presentation?

How do you choose colors? The **colorspace** (Ihaka, Murrell, Hornik & Zeileis, 2011) package in R provides a reasonable selection of color palettes for plots, more specifically applied to statistical graphics than Cynthia Brewer’s map work in the **RColorBrewer** package (Neuwirth, 2011). Several web sites (Etre, 2011; Dougherty & Wade, 2011) provide tools to help color blind proof your work.

Robbins (2006) is a basic guide to good graphics. The R package **ggplot2** (Wickham, 2009) has elegant and cognitively perceptive plot defaults.

Not a talk, a poster!

Slides from a 2007 Joint Statistical Meetings Introductory Overview Lecture (Cook, 2007) give guidelines for constructing a poster. Posters allow the presenter to engage the audience in small group individualized discussion. But in order to engage a small audience you need to attract the attention of passers-by. Designing your poster with a visual focal point that can be seen from several feet away will draw people to your work.

Some of the key recommendations in this lecture are:

- Plan the layout and flow of the poster.
- As with a talk, decide on the main message, and determine who is your audience.
- Choose your color scheme, keeping in mind color blindness limitations, readability, and avoid color combinations that have subliminal meanings, e.g. red, yellow and black of the German flag.

- Choose your text size and font. Titles should be about 100pt font, headings 50pt and text in the body at least 25pt. Avoid all capitals.
- Data plots make good focal points. A contextual image can help the visitors grasp the context of the data quickly, and provide people with something familiar to draw their attention. Good quality graphics are important, generated by R software for example.
- A movie, or audio recording, can help draw the attention of passers-by. These should not be substitutes for engaging the audience in discussion.

Remember that there are lots of bad examples of posters at Statistics meetings. The excuse of “this is how everyone else does their poster” is not a reasonable justification for perpetuating poor scholarship. Each generation is held to higher and higher standards as we develop our understanding about good practices. Excellent advice on producing posters can be found at the web site by Cape Higher Education Consortium (2011). Also the web site by Purrington (2011) has some useful discussion about designing scientific posters. The Data Expo competitions (ASA, 2011) run in conjunction with the Joint Statistical Meetings often have examples of good posters, and examples of previous useR! posters can be found via <http://www.r-project.org/conferences.html>.

Responsible audience?

Occasionally, well maybe, more than occasionally, I hear some members of our profession extolling the virtues of a talk – but it is clear they didn’t have a clue what the talk was about. There is a responsibility of the audience to **not** be impressed because they are snowballed by a speaker. The audience has a right to expect the speaker to make the work clear, and easier to understand, and to do some of the work of deciphering the material for them.

Last words

Be mindful, that giving a talk in front of peers is a privilege – not many people in this world have the opportunity to speak their mind and be listened to, particularly in a prominent setting such as useR!.

Many people enthuse about a TED talk (Rosling, 2006). I’ve recently been pointed to another by Chris Wild (2009) which is a marvellous statistical presentation.

Bibliography

- American Statistical Association. (2011) Data Expo Posters URL <http://stat-computing.org/dataexpo/>
- Beamer Developers. (2011) Beamer—A LaTeX class for producing presentations. URL <https://bitbucket.org/rivanvx/beamer/wiki/Home>.
- Cape Higher Education Consortium. (2011) Information Literacy URL <http://www.lib.uct.ac.za/infolit/poster.htm>.
- D. Cook. (2007) Improving Statistical Posters. URL <http://www.amstat.org/meetings/jsm/2008/pdfs/ImprovingStatisticalPosters.pdf>.
- B. Dougherty and A. Wade. (2011) URL <http://www.vischeck.com/vischeck/>.
- Etre Limited. (2011) URL <http://www.etre.com/tools/colourblindsight/>.
- R. Ihaka, P. Murrell, K. Hornik, A. Zeleis. (2011) colorspace: Color Space Manipulation. URL <http://cran.r-project.org>
- E. Neuwirth. (2011) RColorBrewer: ColorBrewer palettes. URL <http://cran.r-project.org>.
- C. Purrrington. (2011) Advice on Designing Scientific Posters. URL <http://www.swarthmore.edu/NatSci/cpurrin1/posteradvice.htm>.
- N. Robbins. (2006) Creating More Effective Graphs. URL <http://www.wiley.com>.
- H. Rosling. (2006) GapMinder URL http://www.ted.com/talks/lang/eng/hans_rosling_shows_the_best_stats_you_ve_ever_seen.html
- M. Schoeberl and B. Toon. (2011) Ten Secrets to Giving a Good Scientific Talk. URL http://www.cgd.ucar.edu/cms/agu/scientific_talk.html.
- E. Tufte. (1990) The Visual Display of Quantitative Information. Graphics Press. Cheshire, CT.
- A. J. van Marle. (2007) The Art of Scientific Presentations. URL https://www.cfa.harvard.edu/~scranmer/vanmarle_talks.html#Technical_preparation.
- H. Wickham. (2009) *ggplot2: Elegant graphics for data analysis*. useR. Springer.
- C. Wild. (2009) Early Statistical Inferences: The Eyes Have It. URL <http://www.stat.auckland.ac.nz/~wild/09.wild.USCOTS.html>.

Dianne Cook
Department of Statistics
Iowa State University
USA
dcook@iastate.edu

Book Reviews

Forest Analytics with R (Use R)

Andrew R. ROBINSON and Jeff D. HAMANN.
Berlin: Springer, 2011. ISBN 978-1-4419-7761-8. xv + 339 pp. €57.99 (paperback).

Forestry is a broad field touching economical management as well as landscape planning, survey design/analysis, spatial statistics, growth simulations and much more. Accordingly, also the topics related to statistical computing (and hence R) cover a lot of ground. The present book has luckily refrained from trying to cover all possible aspects, while at the same time still being surprisingly comprehensive. It aims at forest scientists, managers, researchers and students, who have little experience with R.

The book is organised in four parts. Part I, *Introduction and Data Management*, introduces R and typical forest data sets, of which several are provided in the companion R-package **FAwR**. Part II, *Sampling and Mapping*, illustrates the use of the **survey** package to estimate mean and variance of typical forest survey designs. It then continues to briefly sketch imputation and spatial interpolation techniques. Part III, *Allometry and Fitting Models*, covers regression, non-linear regression and mixed effect models. Part IV, *Simulation and Optimization* introduce the interaction of C-code with R through two forest growth models and uses a question from forest planing to illustrate the use of linear programming for optimisation.

The four parts differ greatly in their style, depth and quality. Part I can easily be skipped by the more experienced R-user, but offers a useful and gentle introduction to general R functionality with respect to the following three parts. To appreciate the sampling analyses of part II (including, for example, simple random and systematic sampling, cluster and two-stage sampling), a more detailed knowledge of the **survey** package (Lumley, 2010) and of sampling designs in general (e.g., Lohr, 2009) is required. I found the notation for variance estimation somewhat unsavoury, because it deviated from both these books, as well as the book dedicated to forest sampling (Gregoire and Valentine, 2008). Imputation and interpolation techniques receive only a superficial brush, e.g. focussing on (spatial!) nearest-neighbour imputation without mentioning regression-based imputation (Harrell, 2001) at all.

In stark contrast, regression, non-linear and mixed models are dealt with much more carefully and very competently. While the purpose of the book is not to explain so much why, but how, to carry out certain computations, this section is a showcase

of how to teach model interpretation. Even after a decade of stats teaching I found this part inspiring. The key ingredient, I think, is that the authors carry a single example through different analyses. They show improvements (or lack thereof) compared to previous models, explain very accurately the model output and they give intuitive and rule-of-thumb guidance on which “knobs to turn” during analysis.

The final part is again rather different in style. It provides walk-through examples without much explanation of why one would want to use this specific forest growth model (provided through the **rconifers** package: Hamann et al., 2010) or any implementational details. The optimisation section using linear programming is virtually incomprehensible without parallel reading on the subject. This fourth part feels like an incomplete draft shoved in for the sake of topic coverage.

Overall, Forest Analytics with R offers an entry to several statistical computation topics encountered by forestry students and practitioners. The sampling-section is sufficient for many standard designs. Only the regression-section is both in-depth and generic, while the simulation studies are too specific to offer themselves to an easy transfer to other problems.

Carsten F. Dormann
Helmholtz Centre for Environmental
Research-UFZ, Leipzig, Germany

Bibliography

- T. G. Gregoire and H. T. Valentine. *Sampling Strategies for Natural Resources and the Environment*. Chapman and Hall/CRC, New York, 2008. ISBN 9781584883708.
- J. Hamann, M. Ritchie, and the R Core team. *rconifers: Alternative Interface to the CONIFERS Forest Growth Model*, 2010. R package version 1.0.5.
- F. E. Harrell. *Regression Modeling Strategies - with Applications to Linear Models, Logistic Regression, and Survival Analysis*. Springer, New York, 2001. ISBN 0-387-95232-2.
- S. L. Lohr. *Sampling: Design and Analysis*. Duxbury Press, North Scituate, Mass., 2nd edition, 2009. ISBN 9780495105275.
- T. Lumley. *Complex Surveys: A Guide to Analysis Using R*. Wiley, Hoboken, NJ, 2010. ISBN 9780470284308.

Conference Review: Third Meeting of Polish R Users, The Influenza Challenge

by Przemysław Biecek

The third Polish R Users' Conference took place at the Mathematical Institute of Polish Academy of Sciences in Wrocław on 24-25 of September 2010. The meeting was traditionally named WZUR, an abbreviation which has the same meaning as "equation" in spoken Polish. This year the conference adopted a fresh format. I wish to share with you both the reasons behind changing the format and the afterthoughts of the conference participants.

This time the leading subject for the conference was the epidemiology of influenza. Almost 50 people from various parts of Poland and representing numerous professions (statisticians, mathematicians, medical doctors, students) gathered to find out more about influenza and methods for analyzing data related to this disease. A few months in advance of the conference, we had gathered weekly reports covering the last 11 years of influenza occurrences in Poland. Both the absolute numbers and the rates per 100,000 individuals of infections and mortalities were available for four age groups and for 16 different regions. All participants had access to the same data set and all of them had time to experiment on the dataset using an assortment of statistical methods. During the two-day conference various applications were presented of existing statistical procedures available in R as well as newly developed methods.

The dataset was a common basis for all talks. We requested all presenters to prepare talks composed of three parts. In the first part the statistical methods, the mathematical theory behind them along with some insights were presented. In the second part a set of R functions covering these methods was introduced. And finally in the third part a sample application of presented methods to the dataset described above was demonstrated. All participants were familiarized with the dataset, and this created a common platform which facilitated better communication among statisticians and physicians. The epidemiology of influenza was tackled from different angles. Results from variety of statistical procedures created a framework for discussion between people who are developing new statistical methods and for people who are applying them.

The conference was conducted in Polish, and most of the conference materials are in Polish but the equations and figures are international, and presentations are available on the web site <http://www.biecek.pl/WZUR>. Almost all talks were recorded and the video files are freely available from the above address.

List of presented talks:

1. Short-term changes in mortality during influenza epidemics, Daniel Rabczenko (PZH),
2. Calibration estimators in the study of influenza, Tomek Józefowski, Marcin Szymkowiak (UE Poznań),
3. Time series comparison using Dynamic Time Warping, Paweł Teisseyre (IPI PAN),
4. Selected generalized linear models. Alicja Szabelska (UE Poznań),
5. CUMSUM method in prediction of influenza epidemics. Konrad Furmańczyk, Marta Zalewska, Przemysław Biecek, Stanisław Jaworski (WUM+UW),
6. The ECAP study, the correspondence analysis in the asthma study. Konrad Furmańczyk, Stanisław Jaworski, Bolesław Samoliński, Marta Zalewska (WUM),
7. Seasonality of influenza. Marta Musiał, Łukasz Wawrowski, Maciej Beręsewicz, Anita Mackowiak (UE Poznań),
8. Logistic regression in the study of influenza. Piotr Śliwka (UKW).
9. The simecol package in Influenza study. Marta Markiewicz, Anna Sikora, Katarzyna Załęcka, Michał Balcerak, Piotr Kupczyk (PWr),
10. The artificial neural networks in forecasting the influenza epidemics. Anna Noga, Roksana Kowalska, Maciej Kawecki, Paweł Szczypior (PWr),
11. Intensity estimation with multiplicative intensity models. Anna Dudek Piotr Majerski (AGH),
12. Influenza epidemiology, Grażyna Dulny (WUM).

Additionally, during the second day of the conference the survey "Epidemiology of Allergic Disease in Poland" was presented (<http://ecap.pl/eng-www/material.html>). The project was conducted with the Department of Prevention of Environmental Hazards and Allergology at the Medical University of Warsaw by the initiative of the Minister of Health.

The organizing committee: Przemysław Biecek, Marta Zalewska, Konrad Furmańczyk, Stanisław Jaworski would like to thank:

The Scientific Committee Prof. Leon Bobrowski, Prof Antoni Dawidowicz, Prof. Jacek Koronacki, Dr Witold Kupść, Prof. Wojciech Niemiro, Dr Daniel Rabczenko, Prof. Bolesław Samoliński, Prof. Łukasz Stettner, and Prof. Wojciech Zieliński.

Our host The Polish Mathematical Society in Wrocław,

Our generous sponsors and supporters Netezza, Warsaw University, The Medical University of Warsaw, GlaxoSmithKline, Sanofi Pasteur, and The Independent Public Central Clinical Hospital.

We are looking forward to the next WZUR. This

time we will focus on the data set “Social Diagnosis 2000-2011” (see: <http://diagnoza.com/index-en.html>). The challenge is to understand changes in objective and subjective quality of life in Poland in last 11 years.

As always we warmly welcome new participants and sponsors!

On behalf of The Organizing Committee,

*Przemysław Biećek
Institute of Applied Mathematics
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw
Banacha2, 02-097 Warszawa,
Poland
p.biecek@mimuw.edu.pl*

Conference Review: Kickoff Workshop for Project MOSAIC

by Nicholas J. Horton, Randall Pruim and Danny Kaplan

Project MOSAIC (<http://www.mosaic-web.org>) is a community of educators working to develop new ways of introducing modeling, statistics, computation and calculus to students in colleges and universities. The Institute for Mathematics and its Applications at the University of Minnesota hosted the kickoff workshop for Project MOSAIC from June 30–July 2, 2010.

The MOSAIC community helps educators share ideas and resources that will improve undergraduate teaching, and to develop a curricular and assessment infrastructure to support the dissemination and evaluation of these ideas and resources. Other goals of this NSF-funded project (0920350) are to develop important skills in students of science, technology, engineering, and mathematics that will be needed for their professional careers, with a particular focus on the integration of capacities in four main areas:

Modeling The ability to create useful and informative representations of real-world situations.

Statistics The analysis of variability that draws on our ability to quantify uncertainty and to draw logical inferences from observations and experiments.

Computation The capacity to think algorithmically, to manage data on large scales, to visualize and interact with models, and to automate tasks for efficiency, accuracy, and reproducibility.

Calculus The traditional mathematical entry point for college and university students and a subject that still has the potential to provide important insights to today's students.

The workshop, which spanned three days, provided an opportunity for educators to present curricular innovations that have already been developed, to discuss the overall organization of curricula that will effectively unify the four areas outlined above, and to establish strategic plans for Project MOSAIC. More than 30 people from nearly as many institutions attended.

R was featured as a computational environment for many of the presentations. These included examples of the use of R for teaching calculus (Danny Kaplan, Macalester College), resampling in the introductory statistics course (Nathan Tintle, Hope College), computer languages to support MOSAIC

instruction (Randall Pruim, Calvin College), stock market simulation to better understand variability (Nicholas Horton, Smith College), statistical modeling for poets (Vittorio Addona, Macalester College), reproducible analysis (Nicholas Horton, Smith College) and the cost of information (Danny Kaplan, Macalester College)

Since the workshop, the project organizers have been scheduling *M-Casts*. These 20-minute webinars broadcast over the Internet on the 2nd, 4th, and 5th Friday of each month are designed to provide a quick and easy way for educators to share ideas, to get reactions from others, and to form collaborations. Recent R-related M-Casts include discussion of new differentiation and anti-differentiation operators in R for calculus, and using simulation in R to teach the logic of hypothesis tests. Future M-Casts are planned to encourage additional use of R as an environment for computation within mathematics, science, statistics and modeling courses.

To further promote and facilitate the use of R in these efforts, Project MOSAIC has developed the **mosaic** package, which is now available on CRAN. It includes a number of datasets and functions which work to clarify and simplify integration of computing and modeling in introductory and intermediate statistics courses. This package was used extensively as part of the May 2011 Project MOSAIC USCOTS (United States Conference on Teaching Statistics) pre-conference workshop on using R to teach statistics. A low-volume discussion list (r-users@mosaic-web.org) has been created. For more information or to subscribe, see the URL at <http://mailman-mail5.webfaction.com/listinfo/r-users>.

Nicholas J. Horton

*Department of Mathematics and Statistics, Smith College
Clark Science Center, 44 College Lane
Northampton, MA 01063-0001 USA
nhorton@smith.edu*

Randall Pruim

*Department of Mathematics and Statistics, Calvin College
Grand Rapids, MI 49546 USA
rpruim@calvin.edu*

Danny Kaplan

*Department of Mathematics and Computer Science,
Macalester College
1600 Grand Avenue, St. Paul, MN 55105 USA
kaplan@macalester.edu*

Changes in R

From version 2.12.2 to version 2.13.0

by the R Core Team

CHANGES IN R VERSION 2.13.0

SIGNIFICANT USER-VISIBLE CHANGES

- `replicate()` (by default) and `vapply()` (always) now return a higher-dimensional array instead of a matrix in the case where the inner function value is an array of dimension ≥ 2 .
- Printing and formatting of floating point numbers is now using the correct number of digits, where it previously rarely differed by a few digits. (See “scientific” entry below.) This affects many `.*.Rout.save`’ checks in packages.

NEW FEATURES

- `normalizePath()` has been moved to the **base** package (from **utils**): This is so it can be used by `library()` and friends.

It now does tilde expansion.

It gains new arguments `winslash` (to select the separator on Windows) and `mustWork` to control the action if a canonical path cannot be found.

- The previously barely documented limit of 256 bytes on a symbol name has been raised to 10,000 bytes (a sanity check). Long symbol names can sometimes occur when deparsing expressions (for example, in `model.frame`).
- `reformulate()` gains a `intercept` argument.
- `cmdscale(add = FALSE)` now uses the more common definition that there is a representation in $n-1$ or less dimensions, and only dimensions corresponding to positive eigenvalues are used. (Avoids confusion such as PR#14397.)
- Names used by `c()`, `unlist()`, `cbind()` and `rbind()` are marked with an encoding when this can be ascertained.
- R colours are now defined to refer to the sRGB color space.
The PDF, PostScript, and Quartz graphics devices record this fact. X11 (and Cairo) and Windows just assume that your screen conforms.
- `system.file()` gains a `mustWork` argument (suggestion of Bill Dunlap).
- `new.env(hash = TRUE)` is now the default.

- `list2env(envir = NULL)` defaults to hashing (with a suitably sized environment) for lists of more than 100 elements.
- `text()` gains a `formula` method.
- `IQR()` now has a `type` argument which is passed to `quantile()`.
- `as.vector()`, `as.double()`, etc., duplicate less when they leave the mode unchanged but remove attributes.
`as.vector(mode = "any")` no longer duplicates when it does not remove attributes. This helps memory usage in `matrix()` and `array()`. `matrix()` duplicates less if data is an atomic vector with attributes such as `names` (but no `class`).
`dim(x) <- NULL` duplicates less if `x` has neither dimensions nor `names` (since this operation removes `names` and `dimnames`).
- `setRepositories()` gains an `addURLs` argument.
- `chisq.test()` now also returns a `stdres` component, for standardized residuals (which have unit variance, unlike the Pearson residuals).
- `write.table()` and friends gain a `fileEncoding` argument, to simplify writing files for use on other OSes (e.g. a spreadsheet intended for Windows or Mac OS X Excel).
- Assignment expressions of the form `foo:::bar(x) <- y` and `foo:::bar(x) <- y` now work; the replacement functions used are `foo:::'bar<-'` and `foo:::'bar<-'`.
- `Sys.getenv()` gains a `names` argument so `Sys.getenv(x, names = FALSE)` can replace the common idiom of `as.vector(Sys.getenv())`. The default has been changed to not name a length-one result.
- Lazy loading of environments now preserves attributes and locked status. (The locked status of bindings and active bindings are still not preserved; this may be addressed in the future).
- `options("install.lock")` may be set to `FALSE` so that `install.packages()` defaults to ‘`--no-lock`’ installs, or (on Windows) to `TRUE` so that binary installs implement locking.
- `sort(partial = p)` for large `p` now tries Shell-sort if quicksort is not appropriate and so works for non-numeric atomic vectors.

- `sapply()` gets a new option `simplify = "array"` which returns a “higher rank” array instead of just a matrix when `FUN()` returns a `dim()` length of two or more.
`replicate()` has this option set by default, and `vapply()` now behaves that way internally.
 - `aperm()` becomes S3 generic and gets a table method which preserves the class.
 - `merge()` and `as.hclust()` methods for objects of class "dendrogram" are now provided.
 - `as.POSIXlt.factor()` now passes ... to the character method (suggestion of Joshua Ulrich).
 - The character method of `as.POSIXlt()` now tries to find a format that works for all non-NA inputs, not just the first one.
 - `str()` now has a method for class "Date" analogous to that for class "POSIXt".
 - New function `file.link()` to create hard links on those file systems (POSIX, NTFS but not FAT) that support them.
 - New `Summary()` group method for class "ordered" implements `min()`, `max()` and `range()` for ordered factors.
 - `mostattributes<-()` now consults the "dim" attribute and not the `dim()` function, making it more useful for objects (such as data frames) from classes with methods for `dim()`. It also uses `attr<-()` in preference to the generics `name<-()`, `dim<-()` and `dimnames<-()`. (Related to PR#14469.)
 - There is a new option "browserNLdisabled" to disable the use of an empty (e.g. via the 'Return' key) as a synonym for `c` in `browser()` or `n` under `debug()`. (Wish of PR#14472.)
 - `example()` gains optional new arguments `character.only` and `give.lines` enabling programmatic exploration.
 - `serialize()` and `unserialize()` are no longer described as ‘experimental’. The interface is now regarded as stable, although the serialization format may well change in future releases. (`serialize()` has a new argument `version` which would allow the current format to be written if that happens.)
- New functions `saveRDS()` and `readRDS()` are public versions of the ‘internal’ functions `.saveRDS()` and `.readRDS()` made available for general use. The dot-name versions remain available as several package authors have made use of them, despite the documentation. `saveRDS()` supports `compress = "xz"`.

- Many functions when called with a not-open connection will now ensure that the connection is left not-open in the event of error. These include `read.dcf()`, `dput()`, `dump()`, `load()`, `parse()`, `readBin()`, `readChar()`, `readLines()`, `save()`, `writeBin()`, `writeChar()`, `writeLines()`, `.readRDS()`, `.saveRDS()` and `tools:::parse_Rd()`, as well as functions calling these.
- Public functions `find.package()` and `path.package()` replace the internal dot-name versions.
- The default method for `terms()` now looks for a "terms" attribute if it does not find a "terms" component, and so works for model frames.
- `httpd()` handlers receive an additional argument containing the full request headers as a raw vector (this can be used to parse cookies, multi-part forms etc.). The recommended full signature for handlers is therefore `function(url, query, body, headers, ...)`.
- `file.edit()` gains a `fileEncoding` argument to specify the encoding of the file(s).
- The format of the HTML package listings has changed. If there is more than one library tree, a table of links to libraries is provided at the top and bottom of the page. Where a library contains more than 100 packages, an alphabetic index is given at the top of the section for that library. (As a consequence, package names are now sorted case-insensitively whatever the locale.)
- `isSeekable()` now returns `FALSE` on connections which have non-default encoding. Although documented to record if ‘in principle’ the connection supports seeking, it seems safer to report `FALSE` when it may not work.
- R CMD REMOVE and `remove.packages()` now remove file `R.css` when removing all remaining packages in a library tree. (Related to the wish of PR#14475: note that this file is no longer installed.)
- `unzip()` now has a `unzip` argument like `zip.file.extract()`. This allows an external `unzip` program to be used, which can be useful to access features supported by Info-ZIP’s `unzip` version 6 which is now becoming more widely available.
- There is a simple `zip()` function, as wrapper for an external `zip` command.
- `bzfile()` connections can now read from concatenated `bzip2` files (including files written with `bzfile(open = "a")`) and files created by

some other compressors (such as the example of PR#14479).

- The primitive function `c()` is now of type `BUILTIN`.
- `plot(<dendrogram>, ..., nodePar=*)` now obeys an optional `xpd` specification (allowing clipping to be turned off completely).
- `nls(algorithm="port")` now shares more code with `nlminb()`, and is more consistent with the other `nls()` algorithms in its return value.
- `xz` has been updated to 5.0.1 (very minor bugfix release).
- `image()` has gained a logical `useRaster` argument allowing it to use a bitmap raster for plotting a regular grid instead of polygons. This can be more efficient, but may not be supported by all devices. The default is `FALSE`.
- `list.files()/dir()` gains a new argument `include.dirs()` to include directories in the listing when `recursive = TRUE`.
- New function `list.dirs()` lists all directories, (even empty ones).
- `file.copy()` now (by default) copies read/write/execute permissions on files, moderated by the current setting of `Sys.umask()`.
- `Sys.umask()` now accepts `mode = NA` and returns the current `umask` value (visibly) without changing it.
- There is a `!` method for classes "octmode" and "hexmode": this allows `xor(a, b)` to work if both `a` and `b` are from one of those classes.
- `as.raster()` no longer fails for vectors or matrices containing `NAs`.
- New hook "before.new.plot" allows functions to be run just before advancing the frame in `plot.new`, which is potentially useful for custom figure layout implementations.
- Package `tools` has a new function `compactPDF()` to try to reduce the size of PDF files via `qpdf` or `gs`.
- `tar()` has a new argument `extra_flags`.
- `dotchart()` accepts more general objects `x` such as 1D tables which can be coerced by `as.numeric()` to a numeric vector, with a warning since that might not be appropriate.
- The previously internal function `create.post()` is now exported from `utils`, and the documentation for `bug.report()`

and `help.request()` now refer to that for `create.post()`.

It has a new method = "mailto" on Unix-alikes similar to that on Windows: it invokes a default mailer via `open` (Mac OS X) or `xdg-open` or the default browser (elsewhere).

The default for `ccaddress` is now `getOption("ccaddress")` which is by default unset: using the username as a mailing address nowadays rarely works as expected.

- The default for `options("mailer")` is now "mailto" on all platforms.
- `unlink()` now does tilde-expansion (like most other file functions).
- `file.rename()` now allows vector arguments (of the same length).
- The "glm" method for `logLik()` now returns an "nobs" attribute (which `stats4::BIC()` assumed it did).
The "nls" method for `logLik()` gave incorrect results for zero weights.
- There is a new generic function `nobs()` in package `stats`, to extract from model objects a suitable value for use in BIC calculations. An S4 generic derived from it is defined in package `stats4`.
- Code for S4 reference-class methods is now examined for possible errors in non-local assignments.
- `findClasses`, `getGeneric`, `findMethods` and `hasMethods` are revised to deal consistently with the `package=` argument and be consistent with soft namespace policy for finding objects.
- `tools::Rdiff()` now has the option to return not only the status but a character vector of observed differences (which are still by default sent to 'stdout').
- The startup environment variables `R_ENVIRON_USER`, `R_ENVIRON`, `R_PROFILE_USER` and `R_PROFILE` are now treated more consistently. In all cases an empty value is considered to be set and will stop the default being used, and for the last two tilde expansion is performed on the file name. (Note that setting an empty value is probably impossible on Windows.)
- Using `R -no-environ CMD`, `R -no-site-file CMD` or `R -no-init-file CMD` sets environment variables so these settings are passed on to child R processes, notably those run by `INSTALL`, `check` and `build`. `R -vanilla CMD` sets these three options (but not '--no-restore').

- `smooth.spline()` is somewhat faster. With `cv=NA` it allows some leverage computations to be skipped,
- The internal (C) function ‘`scientific()`’, at the heart of R’s `format.info(x)`, `format(x)`, `print(x)`, etc, for numeric `x`, has been re-written in order to provide slightly more correct results, fixing PR#14491, notably in border cases including when `digits >= 16`, thanks to substantial contributions (code and experiments) from Petr Savicky. This affects a noticeable amount of numeric output from R.
- A new function `grepRaw()` has been introduced for finding subsets of raw vectors. It supports both literal searches and regular expressions.
- Package `compiler` is now provided as a standard package. See `?compiler::compile` for information on how to use the compiler. This package implements a byte code compiler for R: by default the compiler is not used in this release. See the ‘R Installation and Administration Manual’ for how to compile the base and recommended packages.
- Providing an `exportPattern` directive in a NAMESPACE file now causes classes to be exported according to the same pattern, for example the default from `package.skeleton()` to specify all names starting with a letter. An explicit directive to `exportClassPattern` will still over-ride.
- There is an additional marked encoding “`bytes`” for character strings. This is intended to be used for non-ASCII strings which should be treated as a set of bytes, and never re-encoded as if they were in the encoding of the current locale: `useBytes = TRUE` is automatically selected in functions such as `writeBin()`, `writeLines()`, `grep()` and `strsplit()`. Only a few character operations are supported (such as `substr()`). Printing, `format()` and `cat()` will represent non-ASCII bytes in such strings by a ‘`\xab`’ escape.
- The new function `removeSource()` removes the internally stored source from a function.
- “`srcref`” attributes now include two additional line number values, recording the line numbers in the order they were parsed.
- New functions have been added for source reference access: `getSrcFilename()`, `getSrcDirectory()`, `getSrcLocation()` and `getSrcref()`.

- `Sys.chmod()` has an extra argument `use_umask` which defaults to true and restricts the file mode by the current setting of `umask`. This means that all the R functions which manipulate file/directory permissions by default respect `umask`, notably R CMD INSTALL.
 - `tempfile()` has an extra argument `fileext` to create a temporary filename with a specified extension. (Suggestion and initial implementation by Dirk Eddelbuettel.)
- There are improvements in the way `Sweave()` and `Stangle()` handle non-ASCII vignette sources, especially in a UTF-8 locale: see ‘Writing R Extensions’ which now has a subsection on this topic.
- `factanal()` now returns the rotation matrix if a rotation such as “`promax`” is used, and hence factor correlations are displayed. (Wish of PR#12754.)
 - The `gctorture2()` function provides a more refined interface to the GC torture process. Environment variables `R_GCTORTURE`, `R_GCTORTURE_WAIT`, and `R_GCTORTURE_INHIBIT_RELEASE` can also be used to control the GC torture process.
 - `file.copy(from, to)` no longer regards it as an error to supply a zero-length `from`: it now simply does nothing.
 - `rstandard.glm` gains a `type` argument which can be used to request standardized Pearson residuals.
 - A start on a Turkish translation, thanks to Murat Alkan.
 - `.libPaths()` calls `normalizePath(winslash = "/")` on the paths: this helps (usually) present them in a user-friendly form and should detect duplicate paths accessed via different symbolic links.

SWEAVE CHANGES

- `Sweave()` has options to produce PNG and JPEG figures, and to use a custom function to open a graphics device (see `?RweaveLatex`). (Based in part on the contribution of PR#14418.)
- The default for `Sweave()` is to produce only PDF figures (rather than both EPS and PDF).
- Environment variable `SWEAVE_OPTIONS` can be used to supply defaults for existing or new options to be applied after the Sweave driver setup has been run.

- The Sweave manual is now included as a vignette in the **utils** package.
- `Sweave()` handles `keep.source=TRUE` much better: it could duplicate some lines and omit comments. (Reported by John Maindonald and others.)

C-LEVEL FACILITIES

- Because they use a C99 interface which a C++ compiler is not required to support, `Rvprintf` and `REvprintf` are only defined by ‘`R_ext/Print.h`’ in C++ code if the macro `R_USE_C99_IN_CXX` is defined when it is included.
- `pythag` duplicated the C99 function `hypot`. It is no longer provided, but is used as a substitute for `hypot` in the very unlikely event that the latter is not available.
- `R_inspect(obj)` and `R_inspect3(obj, deep, pvec)` are (hidden) C-level entry points to the internal `inspect` function and can be used for C-level debugging (e.g., in conjunction with the `p` command in `gdb`).
- Compiling R with ‘`--enable-strict-barrier`’ now also enables additional checking for use of unprotected objects. In combination with `gctorture()` or `gctorture2()` and a C-level debugger this can be useful for tracking down memory protection issues.

UTILITIES

- R CMD `Rdiff` is now implemented in R on Unix-alikes (as it has been on Windows since R 2.12.0).
- R CMD `build` no longer does any cleaning in the supplied package directory: all the cleaning is done in the copy.

It has a new option ‘`--install-args`’ to pass arguments to R CMD `INSTALL` for ‘`--build`’ (but not when installing to rebuild vignettes).

There is new option, ‘`--resave-data`’, to call `tools:::resaveRdaFiles()` on the ‘`data`’ directory, to compress tabular files (‘`.tab`’, ‘`.csv`’ etc) and to convert ‘`.R`’ files to ‘`.rda`’ files. The default, ‘`--resave-data=gzip`’, is to do so in a way compatible even with years-old versions of R, but better compression is given by ‘`--resave-data=best`’, requiring R $\geq 2.10.0$.

It now adds a ‘`datafile`’ file for ‘`data`’ directories of more than 1Mb.

Patterns in ‘`.Rbuildignore`’ are now also matched against all directory names (including those of empty directories).

There is a new option, ‘`--compact-vignettes`’, to try reducing the size of PDF files in the ‘`inst/doc`’ directory. Currently this tries `qpdf`: other options may be used in future.

When re-building vignettes and a ‘`inst/doc/Makefile`’ file is found, `make clean` is run if the makefile has a `clean:` target.

After re-building vignettes the default clean-up operation will remove any directories (and not just files) created during the process: e.g. one package created a ‘`.R_cache`’ directory.

Empty directories are now removed unless the option ‘`--keep-empty-dirs`’ is given (and a few packages do deliberately include empty directories).

If there is a field `BuildVignettes` in the package ‘`DESCRIPTION`’ file with a false value, re-building the vignettes is skipped.

- R CMD `check` now also checks for filenames that are case-insensitive matches to Windows’ reserved file names with extensions, such as ‘`nul.Rd`’, as these have caused problems on some Windows systems.

It checks for inefficiently saved ‘`data/*.rda`’ and ‘`data/*.RData`’ files, and reports on those large than 100Kb. A more complete check (including of the type of compression, but potentially much slower) can be switched on by setting environment variable `_R_CHECK_COMPACT_DATA2_` to ‘`TRUE`’.

The types of files in the `data` directory are now checked, as packages are *still* misusing it for non-R data files.

It now extracts and runs the R code for each vignette in a separate directory and R process: this is done in the package’s declared encoding. Rather than call `tools:::checkVignettes()`, it calls `tool:::buildVignettes()` to see if the vignettes can be re-built as they would be by R CMD build. Option ‘`--use-valgrind`’ now applies only to these runs, and not when running code to rebuild the vignettes. This version does a much better job of suppressing output from successful vignette tests.

The ‘`00check.log`’ file is a more complete record of what is output to ‘`stdout`’: in particular contains more details of the tests.

It now check all syntactically valid Rd usage entries, and warns about assignments (unless these give the usage of replacement functions).

‘`.tar.xz`’ compressed tarballs are now allowed, if `tar` supports them (and setting environment variable `TAR` to ‘`internal`’ ensures so on all platforms).

- R CMD check now warns if it finds ‘inst/doc/makefile’, and R CMD build renames such a file to ‘inst/doc/Makefile’.

INSTALLATION

- Installing R no longer tries to find perl, and R CMD no longer tries to substitute a full path for awk nor perl – this was a legacy from the days when they were used by R itself. Because a couple of packages do use awk, it is set as the make (rather than environment) variable AWK.
- make check will now fail if there are differences from the reference output when testing package examples and if environment variable R_STRICT_PACKAGE_CHECK is set to a true value.
- The C99 double complex type is now required.

The C99 complex trigonometric functions (such as ‘csin’) are not currently required (FreeBSD lacks most of them): substitutes are used if they are missing.

- The C99 system call ‘va_copy’ is now required.
- If environment variable R_LD_LIBRARY_PATH is set during configuration (for example in ‘config.site’) it is used unchanged in file ‘etc/ldpaths’ rather than being appended to.
- configure looks for support for OpenMP and if found compiles R with appropriate flags and also makes them available for use in packages: see ‘Writing R Extensions’.

This is currently experimental, and is only used in R with a single thread for colSums() and colMeans(). Expect it to be more widely used in later versions of R.

This can be disabled by the ‘--disable-openmp’ flag.

PACKAGE INSTALLATION

- R CMD INSTALL -clean now removes copies of a ‘src’ directory which are created when multiple sub-architectures are in use. (Following a comment from Berwin Turlach.)
- File ‘R.css’ is now installed on a per-package basis (in the package’s ‘html’ directory) rather than in each library tree, and this is used for all the HTML pages in the package. This helps when installing packages with static HTML pages for use on a webserver. It will also allow future versions of R to use different stylesheets for the packages they install.

- A top-level file ‘.Rinstignore’ in the package sources can list (in the same way as ‘.Rbuildignore’) files under inst that should not be installed. (Why should there be any such files? Because all the files needed to re-build vignettes need to be under inst/doc, but they may not need to be installed.)
- R CMD INSTALL has a new option ‘--compact-docs’ to compact any PDFs under the ‘inst/doc’ directory. Currently this uses qpdf, which must be installed (see ‘Writing R Extensions’).
- There is a new option ‘--lock’ which can be used to cancel the effect of ‘--no-lock’ or ‘--pkglock’ earlier on the command line.
- Option ‘--pkglock’ can now be used with more than one package, and is now the default if only one package is specified.
- Argument lock of install.packages() can now be used for Mac binary installs as well as for Windows ones. The value “pkglock” is now accepted, as well as TRUE and FALSE (the default).
- There is a new option ‘--no-clean-on-error’ for R CMD INSTALL to retain a partially installed package for forensic analysis.
- Packages with names ending in ‘.’ are not portable since Windows does not work correctly with such directory names. This is now warned about in R CMD check, and will not be allowed in R 2.14.x.
- The vignette indices are more comprehensive (in the style of browseVignettes()).

DEPRECATED & DEFUNCT

- require(save = TRUE) is defunct, and use of the save argument is deprecated.
- R CMD check -no-latex is defunct: use ‘--no-manual’ instead.
- R CMD Sd2Rd is defunct.
- The gamma argument to hsv(), rainbow(), and rgb2hsv() is deprecated and no longer has any effect.
- The previous options for R CMD build -binary (‘--auto-zip’, ‘--use-zip-data’ and ‘--no-docs’) are deprecated (or defunct): use the new option ‘--install-args’ instead.
- When a character value is used for the EXPR argument in switch(), only a single unnamed alternative value is now allowed.
- The wrapper utils::link.html.help() is no longer available.

- Zip-ing data sets in packages (and hence R CMD INSTALL options '--use-zip-data' and '--auto-zip', as well as the 'ZipData: yes' field in a DESCRIPTION file) is defunct. Installed packages with zip-ed data sets can still be used, but a warning that they should be re-installed will be given.
- The 'experimental' alternative specification of a name space via .Export() etc is now defunct.
- The option '--unsafe' to R CMD INSTALL is deprecated: use the identical option '--no-lock' instead.
- The entry point `pythag` in 'Rmath.h' is deprecated in favour of the C99 function `hypot`. A wrapper for `hypot` is provided for R 2.13.x only.
- Direct access to the "source" attribute of functions is deprecated; use `deparse(fn, control="useSource")` to access it, and `removeSource(fn)` to remove it.
- R CMD build -binary is now formally deprecated: R CMD INSTALL -build has long been the preferred alternative.
- Single-character package names are deprecated (and R is already disallowed to avoid confusion in 'Depends:' fields).

BUG FIXES

- `drop.terms` and the [method for class "terms" no longer add back an intercept. (Reported by Niels Hansen.)
- `aggregate` preserves the class of a column (e.g. a date) under some circumstances where it discarded the class previously.
- `p.adjust()` now always returns a vector result, as documented. In previous versions it copied attributes (such as dimensions) from the p argument: now it only copies names.
- On PDF and PostScript devices, a line width of zero was recorded verbatim and this caused problems for some viewers (a very thin line combined with a non-solid line dash pattern could also cause a problem). On these devices, the line width is now limited at 0.01 and for very thin lines with complex dash patterns the device may force the line dash pattern to be solid. (Reported by Jari Oksanen.)
- The `str()` method for class "POSIXt" now gives sensible output for 0-length input.
- The one- and two-argument complex maths functions failed to warn if NAs were generated (as their numeric analogues do).

- Added `.requireCachedGenerics` to the `dont.mind` list for `library()` to avoid warnings about duplicates.
- `$<-data.frame` messed with the class attribute, breaking any S4 subclass. The S4 `data.frame` class now has its own `$<-` method, and turns dispatch on for this primitive.
- `Map()` did not look up a character argument f in the correct frame, thanks to lazy evaluation. (PR#14495)
- `file.copy()` did not tilde-expand from and to when to was a directory. (PR#14507)
- It was possible (but very rare) for the loading test in R CMD INSTALL to crash a child R process and so leave around a lock directory and a partially installed package. That test is now done in a separate process.
- `plot(<formula>, data=<matrix>, ...)` now works in more cases; similarly for `points()`, `lines()` and `text()`.
- `edit.default()` contained a manual dispatch for matrices (the "matrix" class didn't really exist when it was written). This caused an infinite recursion in the no-GUI case and has now been removed.
- `data.frame(check.rows = TRUE)` sometimes worked when it should have detected an error. (PR#14530)
- `scan(sep= , strip.white=TRUE)` sometimes stripped trailing spaces from within quoted strings. (The real bug in PR#14522.)
- The rank-correlation methods for `cor()` and `cov()` with `use = "complete.obs"` computed the ranks before removing missing values, whereas the documentation implied incomplete cases were removed first. (PR#14488)
They also failed for 1-row matrices.
- The perpendicular adjustment used in placing text and expressions in the margins of plots was not scaled by `par("mex")`. (Part of PR#14532.)
- Quartz Cocoa device now catches any Cocoa exceptions that occur during the creation of the device window to prevent crashes. It also imposes a limit of 144ft² on the area used by a window to catch user errors (unit misinterpretation) early.
- The browser (invoked by `debug()`, `browser()` or otherwise) would display attributes such as "wholeSrcref" that were intended for internal use only.

- R's internal filename completion now properly handles filenames with spaces in them even when the readline library is used. This resolves PR#14452 provided the internal filename completion is used (e.g., by setting `rc.settings(files = TRUE)`).
- Inside `uniroot(f, ...)`, `-Inf` function values are now replaced by a maximally **negative** value.
- `rowsum()` could silently over/underflow on integer inputs (reported by Bill Dunlap).
- `as.matrix()` did not handle "dist" objects with zero rows.

CHANGES IN R VERSION 2.12.2 patched

NEW FEATURES

- `max()` and `min()` work harder to ensure that `NA` has precedence over `NaN`, so e.g. `min(NaN, NA)` is `NA`. (This was not previously documented except for within a single numeric vector, where compiler optimizations often defeated the code.)

BUG FIXES

- A change to the C function '`R_tryEval`' had broken error messages in S4 method selection; the error message is now printed.
- PDF output with a non-RGB color model used RGB for the line stroke color. (PR#14511)
- `stats4::BIC()` assumed without checking that an object of class "logLik" has an "nobs" attribute: `glm()` fits did not and so `BIC()` failed for them.
- In some circumstances a one-sided `mantelhaen.test()` reported the p-value for the wrong tail. (PR#14514)
- Passing the invalid value `lty = NULL` to `axis()` sent an invalid value to the graphics device, and might cause the device to segfault.
- `Sweave()` with `concordance=TRUE` could lead to invalid PDF files; '`Sweave.sty`' has been updated to avoid this.
- Non-ASCII characters in the titles of help pages were not rendered properly in some locales, and could cause errors or warnings.
- `checkRd()` gave a spurious error if the `\href` macro was used.

CHANGES IN R VERSION 2.12.2

SIGNIFICANT USER-VISIBLE CHANGES

- Complex arithmetic (notably `z^n` for complex `z` and integer `n`) gave incorrect results since R 2.10.0 on platforms without C99 complex support. This and some lesser issues in trigonometric functions have been corrected.

Such platforms were rare (we know of Cygwin and FreeBSD). However, because of new compiler optimizations in the way complex arguments are handled, the same code was selected on `x86_64` Linux with `gcc 4.5.x` at the default `-O2` optimization (but not at `-O`).

- There is a workaround for crashes seen with several packages on systems using '`zlib 1.2.5`'; see the INSTALLATION section.

NEW FEATURES

- PCRE has been updated to 8.12 (two bug-fix releases since 8.10).
- `rep()`, `seq()`, `seq.int()` and `seq_len()` report more often when the first element is taken of an argument of incorrect length.
- The Cocoa back-end for the `quartz()` graphics device on Mac OS X provides a way to disable event loop processing temporarily (useful, e.g., for forked instances of R).
- `kernel()`'s default for `m` was not appropriate if `coef` was a set of coefficients. (Reported by Pierre Chausse.)
- `bug.report()` has been updated for the current R bug tracker, which does not accept emailed submissions.
- R CMD check now checks for the correct use of '`$(LAPACK_LIBS)`' (as well as '`$(BLAS_LIBS)`'), since several CRAN recent submissions have ignored 'Writing R Extensions'.

INSTALLATION

- The '`zlib`' sources in the distribution are now built with all symbols remapped: this is intended to avoid problems seen with packages such as `XML` and `rgobi` which link to '`zlib.so.1`' on systems using '`zlib 1.2.5`'.
- The default for `FFLAGS` and `FCFLAGS` with `gfortran` on `x86_64` Linux has been changed back to '`-g -O2`': however, setting '`-g -O`' may still be needed for `gfortran 4.3.x`.

PACKAGE INSTALLATION

- A ‘LazyDataCompression’ field in the ‘DESCRIPTION’ file will be used to set the value for the ‘--data-compress’ option of R CMD INSTALL.
- Files ‘R/sysdata.rda’ of more than 1Mb are now stored in the lazyload database using xz compression: this for example halves the installed size of package **Imap**.
- R CMD INSTALL now ensures that directories installed from ‘inst’ have search permission for everyone.

It no longer installs files ‘inst/doc/Rplots.ps’ and ‘inst/doc/Rplots.pdf’. These are almost certainly left-overs from Sweave runs, and are often large.

DEPRECATED & DEFUNCT

- The ‘experimental’ alternative specification of a name space via `.Export()` etc is now deprecated.
- `zip.file.extract()` is now deprecated.
- Zip-ing data sets in packages (and hence R CMD INSTALL `--use-zip-data` and the ‘ZipData: yes’ field in a DESCRIPTION file) is deprecated: using efficiently compressed ‘.rda’ images and lazy-loading of data has superseded it.

BUG FIXES

- `identical()` could in rare cases generate a warning about non-pairlist attributes on CHARXP. As these are used for internal purposes, the attribute check should be skipped. (Reported by Niels Richard Hansen).
- If the filename extension (usually ‘.Rnw’) was not included in a call to `Sweave()`, source references would not work properly and the `keep.source` option failed. (PR#14459)
- `format.data.frame()` now keeps zero character column names.
- `pretty(x)` no longer raises an error when `x` contains solely non-finite values. (PR#14468)
- The `plot.TukeyHSD()` function now uses a line width of 0.5 for its reference lines rather than ‘`lwd = 0`’ (which caused problems for some PDF and PostScript viewers).
- The `big.mark` argument to `prettyNum()`, `format()`, etc. was inserted reversed if it was more than one character long.

- R CMD check failed to check the filenames under ‘man’ for Windows’ reserved names.
- The “Date” and “POSIXt” methods for `seq()` could overshoot when `to` was supplied and `by` was specified in months or years.
- The internal method of `untar()` now restores hard links as file copies rather than symbolic links (which did not work for cross-directory links).
- `unzip()` did not handle zip files which contained filepaths with two or more leading directories which were not in the zipfile and did not already exist. (It is unclear if such zipfiles are valid and the third-party C code used did not support them, but PR#14462 created one.)
- `combn(n, m)` now behaves more regularly for the border case $m = 0$. (PR#14473)
- The rendering of numbers in plotmath expressions (e.g. `expression(10^2)`) used the current settings for conversion to strings rather than setting the defaults, and so could be affected by what has been done before. (PR#14477)
- The methods of `napredict()` and `naresid()` for `na.action = na.exclude` fits did not work correctly in the very rare event that every case had been omitted in the fit. (Reported by Simon Wood.)
- `weighted.residuals(drop0=TRUE)` returned a vector when the residuals were a matrix (e.g. those of class “`mlm`”). (Reported by Bill Dunlap.)
- Package HTML index files ‘`<pkg>/html/00Index.html`’ were generated with a stylesheet reference that was not correct for static browsing in libraries.
- `ccf(na.action = na.pass)` was not implemented.
- The parser accepted some incorrect numeric constants, e.g. `20x2`. (Reported by Olaf Mersmann.)
- `format(*, zero.print)` did not always replace the full zero parts.
- Fixes for subsetting or subassignment of “raster” objects when not both `i` and `j` are specified.
- R CMD INSTALL was not always respecting the ‘ZipData: yes’ field of a ‘DESCRIPTION’ file (although this is frequently incorrectly specified for packages with no data or which specify lazy-loading of data). R CMD INSTALL `--use-zip-data` was incorrectly implemented as ‘`--use-zipdata`’ since R 2.9.0.

- `source(file, echo=TRUE)` could fail if the file contained '#line' directives. It now recovers more gracefully, but may still display the wrong line if the directive gives incorrect information.
- `atan(1i)` returned `NaN+Infi` (rather than `0+Infi`) on platforms without C99 complex support.
- `library()` failed to cache S4 metadata (unlike `loadNamespace()`) causing failures in S4-using packages without a namespace (e.g. those using reference classes).
- The function `qlogis(lp, log.p=TRUE)` no longer prematurely overflows to `Inf` when `exp(lp)` is close to 1.
- Updating S4 methods for a group generic function requires resetting the methods tables for the members of the group (patch contributed by Martin Morgan).
- In some circumstances (including for package XML), R CMD INSTALL installed version-control directories from source packages.
- Added `PROTECT` calls to some constructed expressions used in C level eval calls.
- `utils:::create.post()` (used by `bug.report()` and `help.request()`) failed to quote arguments to the mailer, and so often failed.
- `bug.report()` was naive about how to extract maintainer email addresses from package descriptions, so would often try mailing to incorrect addresses.
- `debugger()` could fail to read the environment of a call to a function with a ... argument. (Reported by Charlie Roosen.)
- `prettyNum(c(1i, NA), drop0=TRUE)` or `str(NA_complex_)` now work correctly.

Changes on CRAN

2010-12-13 to 2011-05-31

by Kurt Hornik and Achim Zeileis

New packages in CRAN task views

Bayesian BLR, BMS, BaBooN, BayesDA, BayesX, Bmix, LaplacesDemon, SampleSizeMeans, SampleSizeProportions, abc, bayespack, bbe-mkr, bclust, bfp, bisoreg, bnlearn, catnet, cudaBayesreg, dclone, ebdbNet, factorQR, gausspred, mlogitBMA, mombf, pred-bayescor, predmixcor, rWMBAT, sbgcop, spikeSlabGAM, varSelectIP.

ChemPhys ChemoSpec, ChemometricsWithR, ChemometricsWithRData.

Cluster HDclassif, MOCCA, MetabolAnalyze, SDisc, clusterCons, dpmixsim, edci, fast-cluster, isa2, isopam, kernlab, kml, kml3d, lcmm, mixsmsn, movMF, optpart, pdfCluster, profdpm, sigclust, sparcl.

Distributions CompQuadForm, FAdist, GB2, GeneralizedHyperbolic, HI, PearsonDS*, RMTstat, SkewHyperbolic, distrEllipse, emg, gaussDiff, hyperdirichlet, loglognorm, mgpd, movMF, mvtnorm, poibin, poilog, poistweedie, reliaR*, skellam, spd, stabledist, trapezoid.

Econometrics apt, censReg, erer, ordinal.

Environmetrics diveMove, wq.

Finance RTAQ.

HighPerformanceComputing RInside, doSMP.

MachineLearning CORElearn, Cubist, ahaz, ncvreg, rminer.

Optimization RcppDE, alabama, cmaes, dclone, neldermead, pso, rneos, soma.

Phylogenetics PHYLOGR, RBrownie, TreePar, diversitree.

Psychometrics irtoys, mpt, qgraph, semPLS.

Spatial Geneland, adehabitatHR, adehabitatHS, adehabitatLT, adehabitatMA, gdistance, rgeos.

Survival Biograph, NADA, PermAlgo, bujar, compeir, concreg, crrSC, frailtyHL, global-boosttest, msSurv, plsRcox, powerSurvEpi, risksetROC, survAUC, survJamda, survival-BIV.

TimeSeries FitARMA, Rssa, egarch, fastVAR, lubridate, season.

(* = core package)

New contributed packages

ADM3 An interpretation of the ADM method — automated detection algorithm. Author: Tomas William Fitzgerald.

Agreement Statistical Tools for Measuring Agreement. Author: Yue Yu and Lawrence Lin.

AnnotLists Data extraction tool from annotations files. Author: Nicolas Cagnard.

BCA Business and Customer Analytics. Author: Dan Putler.

BaBooN Bayesian Bootstrap Predictive Mean Matching — Multiple and single imputation for discrete data. Author: Florian Meinfelder. In view: *Bayesian*.

Bessel Bessel Functions Computations and Approximations. Author: Martin Maechler.

BiGGR Creates an interface to BiGG database, provides a framework for simulation and produces flux graphs. Author: Anand K. Gavai.

BioMark Find biomarkers in two-class discrimination problems. Authors: Ron Wehrens and Pietro Franceschi.

Biograph Explore life histories. Author: Frans Willekens. In view: *Survival*.

CCM Correlation classification method (CCM). Authors: Garrett M. Dancik and Yuanbin Ru.

CDVine Statistical inference of C- and D-vine copulas. Authors: Ulf Schepsmeier and Eike Christian Brechmann.

CGene Causal Genetic Analysis. Author: Peter Lipman.

CITAN CITation ANalysis toolpack. Author: Marek Gagolewski.

CRTSize Sample Size Estimation Functions for Cluster Randomized Trials. Author: Michael A Rotondi.

ChemoSpec Exploratory Chemometrics for Spectroscopy. Authors: Bryan A. Hanson DePauw University, Greencastle Indiana USA. In view: *ChemPhys*.

ChemometricsWithR Chemometrics with R — Multivariate Data Analysis in the Natural Sciences and Life Sciences. Author: Ron Wehrens. In view: *ChemPhys*.

ChemometricsWithRData Data for package **ChemometricsWithR**. Author: Ron Wehrens. In view: *ChemPhys*.

CommonJavaJars Useful libraries for building a Java based GUI under R. Author: Kornelius Rohmeyer.

CompareTests Estimate diagnostic accuracy (sensitivity, specificity, etc) and agreement statistics when one test is conducted on only a subsample of specimens. Authors: Hormuzd A. Katki and David W. Edelstein.

Covpath Implements a pathwise algorithm for covariance selection. Author: Vijay Krishnamurthy.

Cubist Rule- and Instance-Based Regression Modeling. Authors: Max Kuhn, Steve Weston, and Chris Keefer. C code for Cubist by R. Quinlan. In view: *MachineLearning*.

DAGGER Consensus genetic maps. Author: Jeffrey Endelman.

DALY DALY Calculator — A GUI for stochastic DALY calculation in R. Authors: Brecht Devleesschauwer, Arie Havelaar, Juanita Haagsma, Nicolas Praet, and Niko Speybroeck.

DART Denoising Algorithm based on Relevance network Topology. Authors: Yan Jiao and Andrew E. Teschendorff.

DARTData Example data set for **DART** package. Author: Yan Jiao.

DBGSA Methods of distance-based gene set functional enrichment analysis. Authors: Li Jin and Huang Meilin.

DIME Differential Identification using Mixture Ensemble. Author: Cenny Taslim, with contributions from Dustin Potter, Abbasali Khalili, and Shili Lin.

DOBAD Analysis of Discretely Observed linear Birth-And-Death(-and-immigration) Markov Chains. Author: Charles Doss.

DeducerMMR A Deducer plugin for moderated multiple regressions and simple slopes analysis. Authors: Alberto Mirisola, Luciano Seta, Manuel Gentile, and Dario La Guardia.

ENmisc Neuwirth miscellaneous. Author: Erich Neuwirth.

ESPRESSO Power Analysis and Sample Size Calculation. Authors: Amadou Gaye and Paul Burton.

EuclideanMaps Displays Euclidean Maps. Author: Elisa Frutos Bernal.

FAdist Distributions that are sometimes used in hydrology. Author: François Aucoin. In view: *Distributions*.

FAMle Maximum Likelihood and Bayesian Estimation of Univariate Probability Distributions. Author: François Aucoin.

FRBData Download financial data from FRB's website. Author: Shinichi Takayanagi.

FisherEM Model-Based Clustering in the Fisher Discriminative Subspace. Authors: J. Loisel, T. Souvannarath, M. Tchebanenko, C. Bouveyron, and C. Brunet.

FuncMap Hive Plots of R Package Function Calls. Author: Bryan A. Hanson.

GA4Stratification A genetic algorithm approach to determine stratum boundaries and sample sizes of each stratum in stratified sampling. Authors: Sebnem Er, Timur Keskinturk, and Charlie Daly.

GANPA Gene Association Network-based Pathway Analysis. Authors: Zhaoyuan Fang, Weidong Tian, and Hongbin Ji.

GANPADATA The **GANPA** Datasets Package. Authors: Zhaoyuan Fang, Weidong Tian, and Hongbin Ji.

GB2 Generalized Beta Distribution of the Second Kind: properties, likelihood, estimation. Authors: Monique Graf and Desislava Nedyalkova. In view: *Distributions*.

GenSA R functions for Generalized Simulated Annealing. Authors: Sylvain Gubian, Yang Xiang, Brian Suomela, and Julia Hoeng.

GrapheR A multiplatform GUI for drawing customizable graphs in R. Author: Maxime Hervé.

GriegSmith Uses Grieg-Smith method on 2 dimensional spatial data. Author: Brian McGuire.

HLMdiag Diagnostic tools for two-level normal hierarchical linear models. Author: Adam Loy.

HPbayes Heligman Pollard mortality model parameter estimation using Bayesian Melding with Incremental Mixture Importance Sampling. Author: David J Sharow.

HSROC Joint meta-analysis of diagnostic test sensitivity and specificity with or without a gold standard reference test. Authors: Ian Schiller and Nandini Dendukuri.

- HiDimDA** High Dimensional Discriminant Analysis. Author: António Pedro Duarte Silva.
- ICC** Functions facilitating the estimation of the Intraclass Correlation Coefficient. Author: Matthew Wolak.
- IQMNMR** Identification and Quantification of Metabolites by Using 1D 1H NMR FID. Author: XuSong.
- ImageMetrics** Facilitates image analysis for social scientists. Author: Solomon Messing.
- IndependenceTests** Nonparametric tests of independence between random vectors. Authors: P Lafaye de Micheaux and M Bilodeau.
- InfDim** Infine-dimensional model (IDM) to analyse phenotypic variation in growth trajectories. Authors: Anna Kuparinen and Mats Björklund.
- Interpol** Interpolation of amino acid sequences. Author: Dominik Heider.
- IsotopeR** Estimates diet contributions from isotopic sources using JAGS. Authors: Jack Hopkins and Jake Ferguson.
- Johnson** Johnson Transformation. Author: Edgar Santos Fernandez.
- Julia** Fractal Image Data Generator. Author: Mehmet Suzen.
- KrigInv** Kriging-based Inversion for Deterministic and Noisy Computer Experiments. Authors: Victor Picheny and David Ginsbourger.
- KsPlot** Check the power of a statistical model. Author: Issei Kurahashi.
- LCAextend** Latent Class Analysis (LCA) with familial dependence in extended pedigrees. Authors: Arafat Tayeb, Alexandre Bureau, and Aurelie Labbe.
- LaplaceDemon** Software for Bayesian Inference. Author: Byron Hall. In view: *Bayesian*.
- LeLogicielR** Functions and datasets to accompany the book “Le logiciel R: Maîtriser le langage, Effectuer des analyses statistiques” (in French). Authors: Lafaye de Micheaux Pierre, Drouilhet Rémy, and Liquet Benoit.
- MALDIquant** Quantitative analysis of MALDI-TOF MS data. Author: Sebastian Gibb.
- MAPLES** Smoothed age profile estimation. Author: Roberto Impicciatore.
- MAT** Multidimensional Adaptive Testing (MAT). Author: Seung W. Choi.
- MDR** Detect gene-gene interactions using multifactor dimensionality reduction. Author: Stacey Winham.
- MEWMA** Multivariate Exponentially Weighted Moving Average (MEWMA) Control Chart. Author: Edgar Santos Fernandez.
- MHadaptive** General Markov Chain Monte Carlo for Bayesian Inference using adaptive Metropolis-Hastings sampling. Author: Corey Chivers.
- MISA** Bayesian Model Search and Multilevel Inference for SNP Association Studies. Author: Melanie Wilson.
- MLPAsats** MLPA analysis to detect gains and loses in genes. Authors: Alejandro Cáceres and Juan R González.
- MSG** Data and functions for the book “Modern Statistical Graphics”. Author: Yihui Xie.
- MUCflights** Munich Franz-Josef-Strauss Airport Pattern Analysis. Authors: The students of the “Advanced R Programming Course”: Basil Abou El-Komboz, Andreas Bender, Abdelilah El Hadad, Laura Göres, Roman Hornung, Max Hughes-Brandl, Christian Lindenlaub, Christina Riedel, Ariane Straub, Florian Wickler, under the supervision of Manuel Eugster and Torsten Hothorn.
- MVA** An Introduction to Applied Multivariate Analysis with R. Authors: Brian S. Everitt and Torsten Hothorn.
- MetABEL** Meta-analysis of genome-wide SNP association results. Authors: Maksim Struchalin and Yurii Aulchenko.
- MetaPCA** Meta-analysis in the Dimension Reduction of Genomic data. Author: Don Kang.
- MetaQC** Quantitative Quality Assessment for Inclusion/Exclusion Criteria of Genomic Meta-Analysis. Author: Don Kang.
- Meth27QC** Sample quality analysis and sample control analysis. Authors: Ling Teng, Chao Chen, and Chunyu Liu.
- MethComp** Functions for analysis of method comparison studies. Authors: Bendix Carstensen and Lyle Gurrin.
- Miney** Implementation of the Well-Known Game to Clear Bombs from a Given Field (Matrix). Author: Roland Rau.
- NBPSeq** The NBP Negative Binomial Model for Assessing Differential Gene Expression from RNA-Seq. Authors: Yanming Di and Daniel W. Schafer, with contributions from Jason S. Cumbie and Jeff H. Chang.

NightDay Night and Day Boundary Plot Function. Author: Max Hughes-Brandl.

Nippon Japanese utility functions and data. Author: Susumu Tanimura.

PBImisc A set of datasets used in my classes or in the book “Modele liniowe i mieszane w R, wraz z przykładami w analizie danych”. Author: Przemysław Biecek.

PL.popN Population Size Estimation. Author: Jakub Stoklosa.

PSCN Parent Specific DNA Copy Number Estimation. Authors: Hao Chen, Haipeng Xing, and Nancy R. Zhang.

Peak2Trough Estimation of the peak-to-trough ratio of a seasonal variation component. Author: Anette Luther Christensen.

PoissonSeq Significance analysis of sequencing data based on a Poisson log linear model. Author: Jun Li.

PottsUtils Utility Functions of the Potts Models. Author: Dai Feng.

PredictABEL Assessment of risk prediction models. Authors: Suman Kundu, Yurii S. Aulchenko, A. Cecile, and J.W. Janssens.

ProfileLikelihood Profile Likelihood for a Parameter in Commonly Used Statistical Models. Author: Leena Choi.

QTLRel Tools for mapping of quantitative traits of genetically related individuals and calculating identity coefficients from a pedigree. Author: Riyan Cheng.

R2SWF Convert R Graphics to Flash Animations. Authors: Yixuan Qiu and Yihui Xie.

RAD Fit RAD models to biological data. Authors: Piers K Dunstan and Scott D Foster.

RAtmosphere Standard Atmospheric profiles. Author: Gionata Biavati.

RISmed Search Pubmed, Make bibliographic content analyzable. Author: S. A. Kovalchik.

RMark R Code for MARK Analysis. Author: Jeff Laake.

RMediation Mediation Analysis. Authors: Davood Tofighi and David P. MacKinnon.

RNCEP Obtain, organize, and visualize NCEP weather data. Authors: Michael U. Kemp, with contributions from E. Emiel van Loon, Judy Shamoun-Baranes, and Willem Bouten.

ROAuth R interface to liboauth. Author: Jeff Gentry.

ROCwoGS Non-parametric estimation of ROC curves without Gold Standard Test. Author: Chong Wang.

RStackExchange R based StackExchange client. Author: Jeff Gentry.

RTAQ Tools for the analysis of trades and quotes in R. Authors: Kris Boudt and Jonathan Cornelissen. In view: *Finance*.

RVAideMemoire Fonctions diverses accompagnant l’ouvrage “Aide-mémoire de statistique appliquée à la biologie”. Author: Maxime Hervé.

RXMCDA Authors: Patrick Meyer and Sébastien Bigaret.

RcmdrPlugin.BCA Rcmdr Plug-In for Business and Customer Analytics. Author: Dan Putler.

RcmdrPlugin.depthTools R commander Depth Tools Plug-In. Authors: Sara López-Pintado and Aurora Torrente.

RcppBDT Rcpp bindings for the Boost Date Time library. Authors: Dirk Eddelbuettel and Romain François.

RcppClassic Deprecated ‘classic’ Rcpp API. Authors: Dirk Eddelbuettel and Romain François, with contributions by David Reiss, and based on code written during 2005 and 2006 by Dominick Samperi.

RcppDE Global optimization by differential evolution in C++. Authors: Dirk Eddelbuettel extending **DEoptim** (by David Ardia, Katharine Mullen, Brian Peterson, Joshua Ulrich) which itself is based on DE-Engine (by Rainer Storn). In view: *Optimization*.

RepeatedHighDim Detection of global group effect in microarrays data. Author: Klaus Jung.

RiDMC R interface to the idmclib library. Authors: Antonio, Fabio Di Narzo.

RnavGraph Using graphs as a navigational infrastructure. Authors: Adrian R. Waddell and R. Wayne Oldford.

RnavGraphImageData Some image data used in the **RnavGraph** package demos. Authors: Adrian R. Waddell and R. Wayne Oldford.

RobustRankAggreg Methods for robust rank aggregation. Authors: Raivo Kolde and Sven Laur.

Rook A web server interface for R. Author: Jeffrey Horner.

- Rssa** A collection of methods for singular spectrum analysis. Author: Anton Korobeynikov. In view: *TimeSeries*.
- Rz** GUI Tool for Data Management like SPSS or Stata. Author: Masahiro Hayashi.
- SPARQL** SPARQL client. Author: Willem Robert van Hage.
- SPECIES** Statistical package for species richness estimation. Author: Ji-Ping Wang.
- SSsimple** State space models. Author: Dave Zes.
- SVMmaj** SVMmaj algorithm. Authors: Hoksan Yip, Patrick J.F. Groenen, and Georgi Nalbandov.
- SYNCSA** Analysis of functional and phylogenetic patterns in metacommunities. Author: Vanderlei Júlio Debastiani.
- SamplingStrata** Optimal stratification of sampling frames for multipurpose sampling surveys. Author: Giulio Barcaroli.
- SemiParBIVProbit** Semiparametric Bivariate Probit Modelling. Authors: Giampiero Marra and Rosalba Radice.
- Sim.DiffProc** Simulation of Diffusion Processes. Authors: Boukhetala Kamal and Guidoum Arsalane.
- Sim.DiffProcGUI** Graphical User Interface for Simulation of Diffusion Processes. Authors: Boukhetala Kamal and Guidoum Arsalane.
- SimultAnR** Correspondence and Simultaneous Analysis. Authors: Amaya Zarraga and Beatriz Goitisolo.
- SixSigma** Six Sigma Tools for Quality and Process Improvement. Authors: Emilio López, Andrés Redchuk, Javier M.Moguerza.
- SpeciesMix** Fit Mixtures of Archetype Species. Authors: Piers K Dunstan, Scott D Foster and Ross Darnell.
- TERAplusB** Test for A+B Traditional Escalation Rule. Author: Eun-Kyung Lee.
- TSgetSymbol** Time Series Database Interface extension to connect with getSymbols. Author: Paul Gilbert.
- TSxls** Time Series Database Interface extension to connect to spreadsheets. Author: Paul Gilbert.
- TSzip** Time Series Database Interface extension to connect to zip files. Author: Paul Gilbert.
- TauP.R** Earthquake Traveltime Calculations for 1-D Earth Models. Authors: Jake Anderson and Jonathan Lees.
- TwoPhaseInd** Two-Phase Independence. Author: James Dai.
- TwoStepCLogit** Conditional logistic regression: A two-step estimation method. Authors: Radu V. Craiu, Thierry Duchesne, Daniel Fortin and Sophie Baillargeon.
- VBmix** Variational Bayesian mixture models. Author: Pierrick Bruneau.
- VennDiagram** Generate high-resolution Venn and Euler plots. Author: Hanbo Chen.
- XLConnect** Excel Connector for R. Author: Mirai Solutions GmbH.
- XLConnectJars** JAR dependencies for the **XLConnect** package. Author: Mirai Solutions GmbH.
- YjdnJlp** Text Analysis by Yahoo! Japan Developer Network. Author: Yohei Sato.
- abd** The Analysis of Biological Data. Authors: Kevin M. Middleton and Randall Pruijm.
- adehabitatHR** Home Range Estimation. Author: Clément Calenge, contributions from Scott Fortmann-Roe. In view: *Spatial*.
- adehabitatHS** Analysis of habitat selection by animals. Author: Clément Calenge, contributions from Mathieu Basille. In view: *Spatial*.
- adehabitatLT** Analysis of animal movements. Author: Clément Calenge, contributions from Stéphane Dray and Manuela Royer. In view: *Spatial*.
- adehabitatMA** Tools to Deal with Raster Maps. Author: Clément Calenge, contributions from Mathieu Basille. In view: *Spatial*.
- afmtools** Estimation, Diagnostic and Forecasting Functions for ARFIMA models. Authors: Javier Contreras-Reyes, Georg M. Goerg, and Wilfredo Palma.
- agridat** Agricultural datasets. Author: Kevin Wright.
- ahaz** Regularization for semiparametric additive hazards regression. Author: Anders Gorst-Rasmussen. In view: *MachineLearning*.
- allanvar** Allan Variance Analysis. Author: Javier Hidalgo Carrió.
- apt** Asymmetric Price Transmission. Author: Changyou Sun. In view: *Econometrics*.
- arulesViz** Visualizing Association Rules and Frequent Itemsets. Authors: Michael Hahsler and Sudheer Chelluboina.

- audit** Bounds for Accounting Populations. Author: Glen Meeden.
- baseline** Baseline Correction of Spectra. Authors: Kristian Hovde Liland and Bjørn-Helge Mevik.
- basicospace** Recover a Basic Space from Issue Scales. Authors: Keith Poole, Howard Rosenthal, Jeffrey Lewis, James Lo, and Royce Carroll.
- bayesLife** Bayesian Projection of Life Expectancy. Authors: Hana Ševčíková, Adrian Raftery; original WinBugs code written by Jennifer Chunn.
- bayesQR** Bayesian quantile regression. Authors: Dries F. Benoit, Rahim Al-Hamzawi, Keming Yu, and Dirk Van den Poel.
- bayespck** Numerical Integration for Bayesian Inference. Authors: Alan Genz (BAYESPACK Fortran source code) and Björn Bornkamp (R interface and minor adaptions in Fortran source code). In view: *Bayesian*.
- bayespref** Hierarchical Bayesian analysis of ecological count data. Authors: Zachariah Gompert and James A. Fordyce.
- bgmm** Gaussian Mixture Modeling algorithms, including the belief-based mixture modeling. Author: Przemysław Biecek and Ewa Szczurek.
- biGraph** Analysis Toolbox For Bipartite Graphs. Author: Ingo Vogt.
- bisoreg** Bayesian Isotonic Regression with Bernstein Polynomials. Author: S. McKay Curtis. In view: *Bayesian*.
- blender** Analyze biotic homogenization of landscapes. Author: David J. Harris.
- bmem** Mediation analysis with missing data using bootstrap. Authors: Zhiyong Zhang and Li-juan Wang.
- bsml** Basis Selection from Multiple Libraries. Authors: Junqing Wu, Jeff Sklar, Yuedong Wang and Wendy Meiring.
- bst** Gradient Boosting. Author: Zhu Wang.
- bujar** Buckley-James Regression for Survival Data with High-Dimensional Covariates. Author: Zhu Wang. In view: *Survival*.
- c3net** Inferring large-scale gene networks with C3NET. Authors: Gokmen Altay and Frank Emmert-Streib.
- caschrono** Séries temporelles avec R — Méthodes et cas. Author: Yves Aragon.
- cems** Conditional expectation manifolds. Author: Samuel Gerber.
- cg** Comparison of groups. Authors: Bill Pikounis and John Oleynick.
- changepoint** Contains functions that run various single and multiple changepoint methods. Authors: Rebecca Killick and Idris A. Eckley.
- citbcmst** Assigning tumor samples to CIT Breast Cancer Molecular Subtypes from gene expression data. Authors: Laetitia Marisa, Aurélien de Reyniès, Mickael Guedj.
- clime** Constrained L_1 -minimization for Inverse (covariance) Matrix Estimation. Authors: T. Tony Cai, Weidong Liu and Xi (Rossi) Luo.
- cncaGUI** Canonical Non-symmetrical Correspondence Analysis in R. Authors: Ana Belén Nieto Librero, Priscila Willems, Purificación Galindo Villardón.
- compeir** Event-specific incidence rates for competing risks data. Authors: Nadine Grambauer and Andreas Neudecker. In view: *Survival*.
- concreg** Concordance regression. Authors: R by Meinhard Ploner, Daniela Dunkler and Georg Heinze; Fortran by Georg Heinze. In view: *Survival*.
- copBasic** Basic Copula Functions. Author: William H. Asquith.
- copulaedas** Estimation of Distribution Algorithms Based on Copula Theory. Authors: Yasser González-Fernández and Marta Soto.
- cosso** COSSO, adaptive COSSO, variable selection for nonparametric smoothing spline models. Author: Hao Helen Zhang.
- crrSC** Competing risks regression for Stratified and Clustered data. Authors: Bingqing Zhou and Aurélien Latouche. In view: *Survival*.
- darts** Statistical Tools to Analyze Your Darts Game. Author: Ryan Tibshirani.
- dataview** Human readable data presentation. Author: Christofer Backlin.
- dbConnect** Provides a graphical user interface to connect with databases that use MySQL. Authors: Dason Kurkiewicz, Heike Hofmann, Ulrike Genschel.
- dbEmpLikeGOF** Goodness-of-fit and two sample comparison tests using sample entropy. Authors: Jeffrey C. Miecznikowski, Lori A. Shepherd, and Albert Vexler.
- deducorrect** Deductive correction of simple rounding, typing and sign errors. Authors: Mark van der Loo, Edwin de Jonge, and Sander Scholtus.

- depthTools** Depth Tools. Authors: Sara López-Pintado and Aurora Torrente.
- detrendeR** A Graphical User Interface (GUI) to process and visualize tree-ring data using R. Author: Filipe Campelo.
- devEMF** EMF graphics output device. Authors: Philip Johnson, with code from R-core.
- dfoptim** Derivative-free Optimization. Author: Ravi Varadhan.
- diff** Statistics: What's the Difference?. Authors: Andrew Gelman, Vincent Dorie, Val Chan, and Daniel Lee.
- diversitree** Comparative phylogenetic tests of diversification. Authors: Richard G. FitzJohn, with GeoSSE by Emma E. Goldberg. In view: *Phylogenetics*.
- dlmodeler** Generalized Dynamic Linear Modeler. Author: Cyrille Szymanski.
- dmt** Dependency Modeling Toolkit. Authors: Leo Lahti and Olli-Pekka Huovilainen.
- doSMP** Foreach parallel adaptor for the **revolIPC** package. Author: Revolution Analytics. In view: *HighPerformanceComputing*.
- eaf** Plots of the Empirical Attainment Function. Authors: Carlos Fonseca, Luís Paquete, Thomas Stützle, Manuel López-Ibáñez and Marco Chiarandini.
- editrules** Parsing and manipulating edit rules. Authors: Edwin de Jonge and Mark van der Loo.
- english** Translate integers into English. Authors: John Fox and Bill Venables.
- enrichv** Enrichment assessment of virtual screening approaches. Author: Hiroaki Yabuuchi.
- epoc** EPoC (Endogenous Perturbation analysis of Cancer). Authors: Rebecka Jornsten, Tobias Abenius, and Sven Nelander.
- erer** Empirical Research in Economics with R. Author: Changyou Sun. In view: *Econometrics*.
- ergm.userterms** User-specified terms for the statnet suite of packages. Authors: Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, Martina Morris and Pavel N. Krivitsky.
- esotericR** esotericR articles from lemnica.com. Author: Jeffrey A. Ryan.
- extfunnel** Additional Funnel Plot Augmentations. Authors: Dean Langan, Alexander Sutton, Julian PT Higgins, and Walter Gregory.
- eyetracking** Eyetracking Helper Functions. Author: Ryan M. Hope.
- factualR** Thin wrapper for the Factual.com server API. Author: Ethan McCallum.
- fastVAR** Fast implementations to estimate Vector Autoregressive models and Vector Autoregressive models with Exogenous Inputs. Author: Jeffrey Wong. In view: *TimeSeries*.
- fastcluster** Fast hierarchical clustering routines for R and Python. Author: Daniel Müllner. In view: *Cluster*.
- fastmatch** Fast `match()` function. Author: Simon Urbanek.
- fda.usc** Functional Data Analysis and Utilities for Statistical Computing. Authors: Febrero-Bande, M. and Oviedo de la Fuente, M.
- ffbase** Basic statistical functions for package **ff**. Author: Edwin de Jonge.
- flux** Flux rate calculation from dynamic closed chamber measurements. Authors: Gerald Jurasinski and Franziska Koebsch.
- fractaldim** Estimation of fractal dimensions. Authors: Hana Ševčíková, Tilmann Gneiting, and Don Percival.
- frailtyHL** Frailty Models via H-likelihood. Authors: Il Do HA, Maengseok Noh, and Youngjo Lee. In view: *Survival*.
- functional** Curry, Compose, and other higher-order functions. Author: Peter Danenberg.
- gaussquad** Collection of functions for Gaussian quadrature. Author: Frederick Novomestky.
- gdistance** Distances and routes on geographical grids. Author: Jacob van Etten. In view: *Spatial*.
- gdsfmt** CoreArray Generic Data Structurs (GDS) R Interface. Author: Xiuwen Zheng.
- geoPlot** Performs Address Comparison. Authors: Randall Shane.
- ggcolpairs** Combination of GGplot2 plots into a matrix based on data columns. Authors: Fernando Fuentes and George Ostrouchov.
- glmnetcr** Fit a penalized constrained continuation ratio model for predicting an ordinal response. Author: Kellie J. Archer.
- globalboosttest** Testing the additional predictive value of high-dimensional data. Authors: Anne-Laure Boulesteix and Torsten Hothorn. In view: *Survival*.

gooJSON Google JSON Data Interpreter for R. Author: Christopher Steven Marcum.

graphite GRAPH Interaction from pathway Topological Environment. Authors: Gabriele Sales, Enrica Calura, and Chiara Romualdi.

gtcorr Calculate efficiencies of group testing algorithms with correlated responses. Author: Sam Lendle.

gwerAM Controlling the genome-wide type I error rate in association mapping experiments. Authors: Benjamin Stich, Bettina Müller, and Hans-Peter Piepho.

hive Hadoop InteractiVE. Authors: Ingo Feinerer and Stefan Theussl.

homeR Functions useful for building physics. Author: Neurobat AG.

hypred Simulation of genomic data in applied genetics. Author: Frank Technow.

iSubpathwayMiner Graph-based reconstruction, analyses, and visualization of the KEGG pathways. Author: Chunquan Li.

iWebPlots Interactive web-based scatter plots. Authors: Eleni Chatzimichalia and Conrad Bessant.

ieeeeround Functions to set and get the IEEE rounding mode. Author: Gianluca Amato.

imputation Fill missing values in a data matrix using SVD, SVT, or kNN imputation. Author: Jeffrey Wong.

infoDecompuTE Information Decomposition of Two-phase Experiments. Authors: Kevin Chang and Katya Ruggiero.

insideRODE Functions with deSolve solver and C/FORTRAN interfaces to **nlme**, together with compiled codes. Authors: Yuzhuo Pan and Xiaoyu Yan.

irlba Fast partial SVD by implicitly-restarted Lanczos bidiagonalization. Authors: Jim Baglama and Lothar Reichel.

isocir Isotonic Inference for Circular Data. Author: The implementation in R was done by Sandra Barragan based on the SAS routines written by Miguel A. Fernández.

iterLap Approximate probability densities by iterated Laplace Approximations. Author: Björn Bornkamp.

kerdiest Nonparametric kernel estimation of the distribution function. Bandwidth selection and estimation of related functions. Authors: Alejandro Quintela del Río and Graciela Estévez Pérez.

knn knn-gcv. Authors: Ernest Liu, Daisy Sun and Aaron Swoboda.

kulife Data sets and functions from the Faculty of Life Sciences, University of Copenhagen. Authors: Claus Ekstrøm, Ib M. Skovgaard, and Torben Martinussen.

lassoshooting L_1 regularized regression (Lasso) solver using the Cyclic Coordinate Descent algorithm aka Lasso Shooting. Author: Tobias Abenius.

lcmr Bayesian estimation of latent class models with random effects. Authors: Liangliang Wang and Nandini Dendukuri.

ldlasso LD LASSO Regression for SNP Association Study. Author: Samuel G. Younkin.

lfe Linear Group Fixed Effects. Authors: Simen Gaure, The Ragnar Frisch Centre for Economic Research.

linkcomm Tools for Generating, Visualizing, and Analysing Link Communities in Networks. Author: Alex T. Kalinka.

lmSupport Support for Linear Models. Author: John Curtin.

lmmlasso Linear mixed-effects models with Lasso. Author: Jürg Schelldorfer.

mRm Conditional maximum likelihood estimation in mixed Rasch models. Author: David Preiningerstorfer.

margLikArrogance Marginal Likelihood Computation via Arrogance Sampling. Author: Benedict Escoto.

maxLinear Conditional Samplings for Max-Linear Models. Author: Yizao Wang.

mcga Machine Coded Genetic Algorithms for the Real Value Optimization Problems. Author: Mehmet Hakan Satman.

mederrRank Bayesian Methods for Identifying the Most Harmful Medication Errors. Authors: Sergio Venturini and Jessica Myers.

mefa4 Multivariate Data Handling with S4 Classes and Sparse Matrices. Author: Peter Solymos.

mgpd Functions for multivariate generalized Pareto distribution (MGPD of Type II). Author: Pal Rakonczai. In view: *Distributions*.

- mht** Multiple Hypotheses Testing For Variable Selection. Author: F. Rohart.
- miP** Multiple Imputation Plots. Author: Paul Brix.
- miRtest** Combined miRNA- and mRNA-testing. Authors: Stephan Artmann, Klaus Jung, and Tim Beissbarth.
- microbenchmark** Sub microsecond accurate timing functions. Author: Olaf Mersmann.
- minPtest** Gene region-level testing procedure for SNP data, using the min P test resampling approach. Author: Stefanie Hieke.
- missForest** Nonparametric Missing Value Imputation using Random Forest. Author: Daniel J. Stekhoven.
- mnspc** Multivariate Nonparametric Statistical Process Control. Authors: Martin Bezener and Peihua Qiu.
- mosaic** Project MOSAIC (mosaic-web.org) math and stats teaching utilities. Authors: Randall Pruim, Daniel Kaplan, and Nicholas Horton.
- moult** Models for Analysing Moult Data. Authors: Birgit Erni. Based on models developed by Underhill and Zucchini (1988,1990).
- movMF** Mixtures of von Mises Fisher Distributions. Authors: Kurt Hornik and Bettina Grün. In view: *Cluster, Distributions*.
- mpa** CoWords Method. Author: Daniel Hernando Rodríguez & Campo Elías Pardo.
- msSurv** Nonparametric Estimation for Multistate Models. Authors: Nicole Ferguson, Guy Brock, and Somnath Datta. In view: *Survival*.
- msir** Model-based Sliced Inverse Regression. Author: Luca Scrucca.
- msr** Morse-Smale approximation, regression and visualization. Authors: Samuel Gerber, Kristi Potter, and Oliver Ruebel.
- mtcreator** Creating MAGE-TAB files using mtcreator. Author: Fabian Grandke.
- multiPIM** Variable Importance Analysis with Population Intervention Models. Authors: Stephan Ritter, Alan Hubbard, and Nicholas Jewell.
- multibiplotGUI** Multibiplot Analysis in R. Authors: Ana Belén Nieto Librero, Nora Baccala, Purificación Vicente Galindo, and Purificación Galindo Villardon.
- multxpert** Common Multiple Testing Procedures and Gatekeeping Procedures. Authors: Alex Dmitrienko, Eric Nantz, and Gautier Paux, with contributions by Thomas Brechenmacher.
- mvmeta** Multivariate meta-analysis and meta-regression. Author: Antonio Gasparrini.
- mxkssd** Efficient mixed-level k -circulant supersaturated designs. Author: B N Mandal.
- neariso** Near-Isotonic Regression. Authors: Holger Hoefling, Ryan Tibshirani, and Robert Tibshirani.
- nullabor** Tools for graphical inference. Author: Hadley Wickham.
- numConversion** Test of accuracy of `as.character()` for a numeric input using the standard arithmetic. Author: Petr Savicky.
- nutshellDE** Beispieldaten zu "R in a Nutshell" (deutsche Ausgabe). Author: Joseph Adler (Autor des **nutshell**-Pakets) and Jörg Beyer (deutsche Übersetzung und Bearbeitung; ergänzender Code).
- nvis** Combination of visualization functions for nuclear data using **ggplot2** and **ggcolpairs**. Authors: Fernando Fuentes and George Ostrouchov.
- oncomodel** Maximum likelihood tree models for oncogenesis. Authors: Anja von Heydebreck, contributions from Christiane Heiss.
- orclus** ORCLUS subspace clustering. Author: Gero Szepannek.
- ordPens** Selection and/or Smoothing of Ordinal Predictors. Author: Jan Gertheiss.
- orddom** Ordinal Dominance Statistics. Author: Jens J. Rogmann.
- palaeoSig** Significance tests for palaeoenvironmental reconstructions. Author: Richard Telford.
- paramlink** Parametric linkage analysis. Author: Magnus Dehli Vigeland.
- parmigene** Parallel Mutual Information estimation for Gene Network reconstruction. Authors: Gabriele Sales and Chiara Romualdi.
- pbivnorm** Vectorized Bivariate Normal CDF. Authors: Fortran code by Alan Genz. R code by Brenton Kenkel, based on Adelchi Azzalini's **mnormt** package.
- pdfCluster** Cluster analysis via nonparametric density estimation. Authors: Adelchi Azzalini, Giovanna Menardi and Tiziana Rosolin. In view: *Cluster*.
- penalizedLDA** Penalized classification using Fisher's linear discriminant. Author: Daniela Witten.

- pencopula** Flexible Copula Density Estimation with Penalized Hierarchical B-Splines. Author: Christian Schellhase.
- pensim** Simulation of high-dimensional data and parallelized repeated penalized regression. Authors: L. Waldron, M. Pintilie, C. Huttenhower* and I. Jurisica* (*equal contribution).
- phyext** An extension of some of the classes in phylobase. Tree objects now support subnodes on branches. Author: J. Conrad Stack.
- pkDACLASS** A complete statistical analysis of MALDI-TOF mass spectrometry proteomics data. Authors: Juliet Ndakum, Mourad Atlas, and Susmita Datta.
- plotmo** Plot a model's response while varying the values of the predictors. Author: Stephen Milborrow.
- plsRcox** Partial least squares Regression for Cox models and related techniques. Authors: Frederic Bertrand, Myriam Maumy-Bertrand, and Nicolas Meyer. In view: *Survival*.
- poibin** The Poisson Binomial Distribution. Author: Yili Hong. In view: *Distributions*.
- pracma** Practical Numerical Math Functions. Author: Hans W Borchers.
- press** Protein REsidue-level Structural Statistics. Authors: Yuanyuan Huang, Stephen Bonett, and Zhijun Wu.
- pressData** Data sets for package **press**. Authors: Yuanyuan Huang, Stephen Bonett, and Zhijun Wu.
- pycno** Pycnophylactic Interpolation. Author: Chris Brunsdon.
- qat** Quality Assurance Toolkit. Author: Andre Duesterhus.
- qgraph** Visualize data as networks. Authors: Sacha Epskamp, Angelique O. J. Cramer, Lourens J. Waldorp, Verena D. Schmittmann, and Denny Borsboom. In view: *Psychometrics*.
- queueing** Basic Markovian queueing models. Author: Pedro Canadilla.
- rCUR** CUR decomposition package. Authors: Andras Bodor and Norbert Solymosi.
- rChoiceDialogs** rChoiceDialogs collection. Authors: Alex Lisovich and Roger Day.
- rdynllcall** Improved Foreign Function Interface (FFI) and Dynamic Bindings to C Libraries (e.g. OpenGL). Author: Daniel Adler.
- readBrukerFlexData** Reads mass spectrometry data in Bruker *flex format. Author: Sebastian Gibb.
- readMzXmlData** Reads mass spectrometry data in mzXML format. Authors: Jarek Tuszyński and Sebastian Gibb.
- rebmix** Random univariate and multivariate finite mixture generation. Author: Marko Nagode.
- reliaR** Package for some probability distributions. Authors: Vijay Kumar and Uwe Ligges. In view: *Distributions*.
- reshapeGUI** A GUI for the **reshape2** and **plyr** packages. Author: Jason Crowley.
- review** Manage Review Logs. Author: Tim Bergsma.
- revoIPC** Shared memory parallel framework for multicore machines. Author: Revolution Analytics.
- rgam** Robust Generalized Additive Model. Authors: Matías Salibián-Barrera, Davor Cubranic, and Azadeh Alimadad.
- rgeos** Interface to Geometry Engine — Open Source (GEOS). Authors: Roger Bivand and Colin Rundel. In view: *Spatial*.
- rich** Species richness estimation and comparison. Author: Jean-Pierre Rossi.
- rminer** Simpler use of data mining methods (e.g. NN and SVM) in classification and regression. Author: Paulo Cortez. In view: *MachineLearning*.
- rneos** XML-RPC Interface to NEOS. Author: Bernhard Pfaff. In view: *Optimization*.
- rocplus** ROC, Precision-Recall, Convex Hull and other plots. Author: Bob Wheeler.
- rpart.plot** Plot rpart models. An enhanced version of `plot.rpart()`. Author: Stephen Milborrow.
- rrBLUP** Genomic selection and association analysis. Author: Jeffrey Endelman.
- rrdf** Support for the Resource Description Framework. Author: Egon Willighagen.
- rrdflibs** Jena libraries for use with **rrdf**. Author: Egon Willighagen.
- rsdepth** Ray Shooting Depth (i.e., RS Depth) functions for bivariate analysis. Authors: Nabil Mustafa, Saurabh Ray, and Mudassir Shabbir.
- rtape** Manage and manipulate large collections of R objects stored as tape-like files. Author: Miron B. Kursa.
- rvmbinary** RVM Binary Classification. Author: Robert Lowe.

- rysgran** Grain size analysis, textural classifications and distribution of unconsolidated sediments. Authors: Maurício Garcia de Camargo, Eliandro Ronael Gilbert, Leonardo Sandrini-Neto.
- s3x** Enhanced S3-Based Programming. Author: Charlotte Maia.
- saves** Fast load variables. Author: Gergely Daróczsi.
- scaleCoef** Scale regression coefficients. Author: Sandrah P. Eckel.
- scriptests** Transcript-based unit tests that are easy to create and maintain. Author: Tony Plate.
- seg** A set of tools for residential segregation research. Authors: Seong-Yun Hong, David O'Sullivan.
- sfa** Stochastic Frontier Analysis. Authors: Ariane Straub, under the supervision of Torsten Hothorn.
- sideChannelAttack** Side Channel Attack. Authors: Liran Lerman, Gianluca Bontempi, and Olivier Markowitch.
- simboot** Simultaneous Confidence Intervals and Adjusted p -values for Diversity Indices. Author: Ralph Scherer.
- skills** Skills in Knowledge Space Theory. Author: Angela Pilhofer.
- smco** A simple Monte Carlo optimizer using adaptive coordinate sampling. Author: Juan David Velásquez.
- snort** Social Network-Analysis On Relational Tables. Authors: Eugene Dubossarsky and Mark Norrie.
- soma** General-purpose optimisation with the Self-Organising Migrating Algorithm. Author: Jon Clayden; based on the work of Ivan Zelinka. In view: *Optimization*.
- somplot** Visualisation of hexagonal Kohonen maps. Authors: Benjamin Schulz and Andreas Dominik.
- sos4R** An R client for the OGC Sensor Observation Service. Author: Daniel Nüst.
- spa** Implements The Sequential Predictions Algorithm. Author: Mark Culp.
- spacodiR** Spatial and Phylogenetic Analysis of Community Diversity. Authors: Jonathan Eastman, Timothy Paine, and Olivier Hardy.
- spd** Semi Parametric Distribution. Author: Alexios Ghalanos. In view: *Distributions*.
- speedR** A GUI based importing and filtering tool. Author: Ilhami Visne.
- speedRlibTF** speedR's table filter library. Author: Ilhami Visne.
- speedRLibs** Libraries (jars) for speedR. Author: Ilhami Visne.
- spi** Compute SPI index. Author: Josemir Neves.
- spikeSlabGAM** Bayesian variable selection and model choice for generalized additive mixed models. Author: Fabian Scheipl. In view: *Bayesian*.
- sporm** Semiparametric proportional odds rate model. Authors: Zhong Guan and Cheng Peng.
- sra** Selection Response Analysis. Author: Arnaud Le Rouzic.
- stabledist** Stable Distribution Functions. Authors: Diethelm Wuertz, Martin Maechler and Rmetrics core team members. In view: *Distributions*.
- stepp** Subpopulation Treatment Effect Pattern Plot (STEPP). Author: Wai-ki Yip, with contributions from Ann Lazar, David Zahrieh, Chip Cole, Ann Lazar, Marco Bonetti, and Richard Gelber.
- superMDS** Implements the supervised multidimensional scaling (superMDS) proposal of Witten and Tibshirani (2011). Author: Daniela M. Witten.
- survAUC** Estimators of prediction accuracy for time-to-event data. Authors: Sergej Potapov, Werner Adler and Matthias Schmid. In view: *Survival*.
- survivalBIV** Estimation of the Bivariate Distribution Function. Authors: Ana Moreira and Luis Meira-Machado. In view: *Survival*.
- svd** Interfaces to various state-of-art SVD and eigen-solvers. Author: Anton Korobeynikov.
- swamp** Analysis and visualization of high-dimensional data in respect to sample annotations. Author: Martin Lauss.
- tabplot** Tableplot, a visualization of large statistical datasets. Authors: Martijn Tennekes and Edwin de Jonge.
- texmex** Threshold exceedences and multivariate extremes. Authors: Harry Southworth and Janet E. Heffernan.
- textir** Inverse Regression for Text Analysis. Author: Matt Taddy.
- timeordered** Time-ordered and time-aggregated network analyses. Author: Benjamin Blonder.

tm.plugin.dc Text Mining Distributed Corpus Plug-in. Authors: Ingo Feinerer and Stefan Theussl.

tmle Targeted Maximum Likelihood Estimation of additive treatment effect. Author: Susan Gruber, in collaboration with Mark van der Laan.

tonymisc Functions for Econometrics Output. Author: J. Anthony Cookson.

track Track Objects. Author: Tony Plate.

trapezoid The Trapezoidal Distribution. Author: Jeremy Thoms Hetzel. In view: *Distributions*.

treecm Estimating and plotting the centre of mass of a tree. Author: Marco Bascietto.

triggr Fast, simple RPC controller for R. Author: Miron B. Kursa.

two.stage.boot Two-stage cluster sample bootstrap algorithm. Author: Patrick Zimmerman.

udunits2 udunits-2 bindings for R. Author: James Hiebert.

unknownR You didn't know you didn't know? Author: Matthew Dowle.

varSelectIP Objective Bayes Model Selection. Authors: Gopal, V. and Novelo, L. L. and Casella, G. In view: *Bayesian*.

varcompci Computation of Confidence Intervals for Variance Components of Mixed Models in R. Authors: Civit, S., Vilardell, M., Hess, A., Matthew, Z., Ge, Y., Caballe, A.

vines Multivariate Dependence Modeling with Vines. Authors: Yasser González-Fernández and Marta Soto.

visualizationTools A few functions to visualize statistical circumstances. Authors: Thomas Roth and Etienne Stockhausen.

voronoi Methods and applications related to Voronoi tessellations. Authors: Christopher D. Barr, Travis A. Gerke, and David M. Diez.

vwr Useful functions for visual word recognition research. Author: Emmanuel Keuleers.

wavemulcor Wavelet routine for multiple correlation. Author: Javier Fernandez-Macho (UPV/EHU).

websockets HTML 5 WebSocket Interface for R. Author: B. W. Lewis.

weights Weighting and Weighted Statistics. Author: Josh Pasek.

xtermStyle Basic text formatting using xterm escape sequences. Author: Christofer Backlin.

zipcode U.S. ZIP Code database. Author: Jeffrey Breen.

Other changes

- The following packages which were previously double-hosted on CRAN and Bioconductor were moved to the Archive, and are now solely available from Bioconductor: **LM-Gene**, **gene2pathway**, **genefu**, **graph**, **minet**, **multtest**, **survcomp**, **vbmp**.
- The following packages were moved to the Archive: **EMC**, **EMCC**, **EffectiveDose**, **Funct-SNP**, **G1DBN**, **ISA**, **PermuteNGS**, **RSeq-qMeth**, **SMC**, **cxxPack**, **distributions**, **elec**, **formula.tools**, **hierfstat**, **muscor**, **pARccs**, **ripa**, **ris**, **sigma2tools**, **tossm**, **tradeCosts**.
- The following packages were resurrected from the Archive: **NADA**, **glmc**, **ifa**, **ipp.toolbox**, **locfdr**, **nparcomp**, **ramps**, **risksetROC**, **rv**.

Kurt Hornik

WU Wirtschaftsuniversität Wien, Austria

Kurt.Hornik@R-project.org

Achim Zeileis

Universität Innsbruck, Austria

Achim.Zeileis@R-project.org

News from the Bioconductor Project

*Bioconductor Team
Program in Computational Biology
Fred Hutchinson Cancer Research Center*

We are pleased to announce Bioconductor 2.8, released on 14 April 2011. Bioconductor 2.8 is compatible with R 2.13.0, and consists of 467 software packages and more than 500 up-to-date annotation packages. There are 49 new software packages, and enhancements to many others. Explore Bioconductor at <http://bioconductor.org>, and install packages with

```
> source("http://bioconductor.org/biocLite.R")
> biocLite() # install standard packages...
> biocLite("IRanges") # ...or only IRanges
```

A Bioconductor machine instance for use with Amazon's cloud service is available; see <http://bioconductor.org/help/bioconductor-cloud-ami>.

New and revised packages

This release includes new packages for diverse areas of high-throughput analysis. Highlights include:

Next-generation sequencing: **ArrayExpressHTS** (RNA-seq), **mosaics** (ChIP-seq), **Rsubread** (alignment), **qrqc**, **seqbias**, **TEQC** (quality assessment), **clst**, **clstutils** (taxonomic placement).

SNPs and small variants: **inveRsion**, **chopsticks**, **snpStats**.

Flow cytometry and proteomics: **flowPhyto**, **flowPlots**, **IPPD**, **MSnbase**, **procoil**.

Microarray: The **a4** collection of packages, **ExiMiR**, **pvac**, **snm**, **TurboNorm**.

Copy number variation: **cn.farms**, **genoset**, **Vega**.

Advanced statistical implementations: **anota**, **Clonality**, **clusterProfiler**, **gaia**, **genefu**, **GSVA**, **ibh**, **joda**, **lol**, **mgsa**, **MLP**, **phenoDist**, **phenoTest**, **RNAinteract**, **survcomp**, **TDARACNE**.

Annotation and visualization: **AnnotationFuncs**, **NCIgraph**, **ENVISIONQuery**, **mcaGUI**.

Our large collection of microarray- and organism-specific annotation packages have, as usual, been updated to include current information. Developments in the **GenomicFeatures** package allow users to retrieve and easily save annotation tracks found at the UCSC genome browser. This provides easy access to diverse information about, e.g., binding factor locations or nucleosome positions, and complements

known gene annotations already available as *TranscriptDb* objects.

Further information on new and existing packages can be found on the Bioconductor web site; 'BiocViews' identify coherent groups of packages, with links to package descriptions, vignettes, reference manuals, and use statistics.

Other activities

The Bioconductor community meets on July 27-29 at our annual conference (<https://secure.bioconductor.org/BioC2011>) in Seattle for a combination of scientific talks and hands-on tutorials; a developer meeting in the late fall will highlight contributions from the European developer community. The active Bioconductor mailing lists (<http://bioconductor.org/help/mailing-list/>) connect users with each other, to domain experts, and to maintainers eager to ensure that their packages satisfy the needs of leading edge approaches. Bioconductor package maintainers and the Bioconductor team invest considerable effort in producing high-quality software. The Bioconductor team continues to ensure quality software through technical and scientific reviews of new packages, and daily builds of released packages on Linux, Windows, and MacOS platforms.

Looking forward

Contributions from the Bioconductor community shape each release; we are seeing a tremendous number of new contributed packages addressing sequence, advanced microarray, and other diverse areas of high-throughput genomic analysis.

Sequence analysis continues to pose significant statistical and computational challenges, although remarkable progress is being made both in the Bioconductor community and more generally. The next release will see contributions enabling common data management (e.g., efficiently querying very large sequence alignments), representation (e.g., of gapped alignments and small variants), and work flow (e.g., RNA-seq read counts and estimation) tasks.

Important developments in the annotation domain include efforts to enhance reproducibility by providing transcript annotation packages, and the ability to easily generate packages for non-model organisms. New efforts are also under way to better use existing and new annotation resources to help users with common tasks in next-generation sequence work flows, for instance immediately identifying whether single nucleotide and other small variants disrupt protein production.

R Foundation News

by Kurt Hornik

Donations and new members

Donations

Joseph J. Allaire, RStudio Inc., USA

J. Brian Loria, USA

Jeffrey R. Stevens, Germany

New benefactors

Joseph J. Allaire, RStudio Inc., USA

New supporting members

Masoud Charkhabi, Canada

James Davis, USA

Patrick Hausmann, Germany

Sebastian Köhler, Germany

Dieter Menne, Germany

Fausto Molinari, Sweden

Stefan Wyder, Switzerland

Kurt Hornik

WU Wirtschaftsuniversität Wien, Austria

Kurt.Hornik@R-project.org