News

The Newsletter of the R Project

Volume 7/2, October 2007

Editorial

by Torsten Hothorn

Welcome to the October 2007 issue of R News, the second issue for this year! Last week, R 2.6.0 was released with many new useful features: `dev2bitmap` allows semi-transparent colours to be used, `plotmath` offers access to all characters in the Adobe Symbol encoding, and three higher-order functions `Reduce`, `Filter`, and `Map` are now available. In addition, the startup time has been substantially reduced when starting without the **methods** package. All the details about changes in R 2.6.0 are given on pages 54 *ff*.

The Comprehensive R Archive Network continues to reflect a very active R user and developer community. Since April, 187 new packages have been uploaded, ranging from **ADaCGH**, providing facilities to deal with array CGH data, to **yest** for model selection and variance estimation in Gaussian independence models. Kurt Hornik presents this impressive list starting on page 61.

The first North American R user conference took place in Ames Iowa last August and many colleagues and friends from all over the world enjoyed a well-organized and interesting conference. Duncan Murdoch and Martin Maechler report on useR! 2007 on page 74. As I'm writing this, Uwe Ligges and his team at Dortmund university are working hard to organize next year's useR! 2008; Uwe's invitation to Dortmund Germany appears on the last page of this issue.

Three of the nine papers contained in this issue were presented at useR! 2006 in Vienna: Peter Dalgaard reports on new functions for multivariate analysis, Heather Turner and David Firth describe their new package **gnm** for fitting generalized nonlinear models, and Olivia Lau and colleagues use ecological inference models to infer individual-level behavior from aggregate data.

Further topics are the analysis of functional magnetic resonance imaging and environmental data, matching for observational studies, and machine learning methods for survival analysis as well as for categorical and continuous data. In addition, an application to create web interfaces for R scripts is described, and Fritz Leisch completes this issue with a review of Michael J. Crawley's *The R Book*.

Torsten Hothorn
Ludwig–Maximilians–Universität München, Germany
`Torsten.Hothorn@R-project.org`

Contents of this issue:

New Functions for Multivariate Analysis

Peter Dalgaard

R (and S-PLUS) used to have limited support for multivariate tests. We had the `manova` function, which extended the features of `aov` to multivariate responses, but like `aov`, this effectively assumed a balanced design, and was not capable of dealing with the within-subject transformations that are commonly used in repeated measurement modelling.

Although the methods encoded in procedures available in SAS and SPSS can seem somewhat old-fashioned, they do have some added value relative to analysis by mixed model methodology, and they have a strong tradition in several applied areas. It was thus worthwhile to extend R's capabilities to handle contrast tests, as well as Greenhouse-Geisser and Huynh-Feldt epsilons. The extensions also provide flexible ways of dealing with linear models with a multivariate response.

Theoretical setting

The general setup is given by

$$Y \sim N(\Xi B, I \otimes \Sigma)$$

Here, Y is $N \times p$ matrix and Σ is a $p \times p$ covariance matrix. The rows y_i of Y are independent with the same covariance Σ.

Ξ is a $N \times k$ design matrix (the reader will have to apologise that I am not using the traditional X, but that symbol is reserved for other purposes later on) and B is a $k \times p$ matrix of regression coefficients.

Thus, we have the same linear model for all p response coordinates, with separate parameters contained in the columns of B.

Standard test procedures

From classical univariate and multivariate theory, a number of standard tests are available. We shall focus on three of them:

1. Testing hypotheses of simplified mean value structure: This reduction is required to be the same for all coordinates, effectively replacing the design matrix Ξ with one spanning a subspace. Such tests take the form of generalized F tests, replacing the variance ratio by

   $$R = \mathrm{MS}_{\mathrm{res}}^{-1}\mathrm{MS}_{\mathrm{eff}}$$

   in which the MS terms are matrices which generalize the mean square terms from analysis of variance. Under the hypothesis, R should be distributed around the unit matrix (in the sense that the two MS matrices both have mean Σ; notice, however, that $\mathrm{MS}_{\mathrm{eff}}$ will be rank deficient whenever the degrees of freedom for the effect is less than p), but for a test statistic we need to reduce R to a scalar measure. Four such measures are cited in the literature, namely Wilks's Λ, the Pillai trace, the Hotelling-Lawley trace, and Roy's greatest root. These are all based on combinations of the eigenvalues of R. Details can be found in, e.g., Hand and Taylor (1987).

   Wilks's Λ is equivalent to the likelihood ratio test, but R and S-PLUS have traditionally favoured the Pillai trace based on the (rather vague) recommendations cited in Hand and Taylor (1987). Each test can be converted to an approximately F distributed statistic. If the tests are for a single-degree-of-freedom hypothesis, the matrix R has only one non-zero eigenvalue and all four tests are equivalent.

2. Testing whether Σ is proportional to a given matrix, say Σ_0 (which is usually the unit matrix I): This is known as Mauchly's test of sphericity. It is based on a comparison of the determinant and the trace of $U = \Sigma_0^{-1}S$ where S is the SSD (deviation sum-of-squares-and-products) matrix ($\mathrm{MS}_{\mathrm{res}}$ instead of S is equivalent). Specifically

   $$W = \det(U)/\mathrm{tr}(U/p)^p$$

   is close to 1 if U is close to a diagonal matrix of dimension p with a constant value along the diagonal. The test statistic $-f \log W$ is an asymptotic χ^2 on $p(p+1)/2 - 1$ degrees of freedom (where f is the degrees of freedom for the covariance matrix. An improved approximation is found in Anderson (1958).

3. Testing linear hypotheses as in point 1 above, but *assuming* that sphericity holds. In this case, we are assuming that the covariance is known up to a constant, and thus are effectively in a univariate setting of (weighted) least squares theory. The relevant F statistic will have (pf_1, pf_2) degrees of freedom if the coordinatewise tests have (f_1, f_2) degrees of freedom.

Within-subject transformations

It is often necessary to consider a transformation of responses: If, for instance, coordinates are repeated measures, we might wish to test for "no change over time" or "same changes over time in different groups" (profile analysis). This leads to analysis of

 ISSN 1609-3631

within-subjects contrasts, rather than of the observations themselves.

If we assume a covariance structure with random effects both between and within subject, the contrasts will cancel out the between-subject variation. They will effectively behave *as if* the between-subject term wasn't there and satisfy a sphericity condition.

Hence we consider the transformed response YT' (which has rows Ty_i). In many cases T is chosen to annihilate another matrix X, i.e. it satisfies $TX = 0$; by rotation invariance, different such choices of T will lead to the same inference as long as they have maximal rank. In such cases it may well be more convenient to specify X rather than T.

For profile analysis, X could be a p-vector of ones. In that case, the hypothesis of *compound symmetry* implies sphericity of $T\Sigma T'$ w.r.t. TT' (but sphericity does not imply compound symmetry), and the hypothesis $\mathsf{E}YT' = 0$ is equivalent to a flat (constant level) profile.

More elaborate within-subject designs exist. As an example, consider a a complete two-way layout, in which case we might want to look at the sets of (a) Contrasts between row means, (b) Contrasts between column means, and (c) Interaction contrasts,

These contrasts connect to the theory of balanced mixed-effects analysis of variance. A model with "all interactions with subject are considered random", i.e. random effects of subjects, as well as random interaction between subject and rows and between subjects and columns, implies sphericity of each of the above contrasts. The proportionality constants will be different linear combinations of the random effect variances.

A common pattern for such sets of contrasts is as follows: Define two subspaces of $\mathbb{R}^p$, defined by matrices X and M, such that $\mathrm{span}(X) \subset \mathrm{span}(M)$. The transformation is calculated as $T = P_M - P_X$ where P denotes orthogonal projection. Actually, T defined like this would be singular and therefore T is thinned by deletion of linearly dependent rows (it can fairly easily be seen that it does not matter exactly how this is done). Putting $M = I$ (the identity matrix) will make T satisfy $TX = 0$ which is consistent with the situation described earlier.

The choice of X and M is most easily done by viewing them as design matrices for linear models in p-dimensional space. Consider again a two-way intrasubject design. To make T a transformation which calculates contrasts between column means, choose M to describe a model with different means in each column and X to describe a single mean for all data. Preferably, but equivalently for a balanced design, let M describe an additive model in rows and columns and let X be a model in which there is only a row effect.

There is a hidden assumption involved in the choice of the *orthogonal* projection onto $\mathrm{span}(M)$. This calculates (for each subject) an ordinary least squares (OLS) fit and for some covariance structures, a weighted fit may be more appropriate. However, OLS is not actually biased, and the efficiency that we might gain is offset by the need to estimate a rather large number of parameters to find the optimal weights.

The epsilons

Assume that the conditions for a balanced analysis of variance are roughly valid. We may decide to compute, say, column means for each subject and test whether the contrasts are zero on average. There could be a loss of efficiency in the use of unweighted means, but the critical issue for an F test is whether the sphericity condition holds for the contrasts between column means.

If the sphericity condition is not valid, then the F test is in principle wrong. Instead, we could use one of the multivariate tests, but they will often have low power due to the estimation of a large number of parameters in the empirical covariance matrix.

For this reason, a methodology has evolved in which "near-sphericity" data are analyzed by F tests, but applying the so-called epsilon corrections to the degrees of freedom. The theory originates in Box (1954a) in which it is shown that F is approximately distributed as $F(\varepsilon f_1, \varepsilon f_2)$, where

$$\varepsilon = \frac{\sum \lambda_i^2/p}{(\sum \lambda_i/p)^2}$$

and the λ_i are the eigenvalues of the true covariance matrix (after contrast transformation, and with respect to a similarly transformed identity matrix). One may notice that $1/p \leq \varepsilon \leq 1$; the upper limit corresponds to sphericity (all eigenvalues equal) and the lower limit corresponds to the case where there is one dominating eigenvalue, so that data are effectively one-dimensional. Box (1954b) details the result as a function of the elements of the covariance matrix in the two-way layout.

The Box correction requires knowledge of the true covariance matrix. Greenhouse and Geisser (1959) suggest the use of $\varepsilon_{\mathrm{GG}}$, which is simply the empirical verson of ε, inserting the empirical covariance matrix for the true one. However, it is easily seen that this estimator is biased, at least when the true epsilon is close to 1, since $\varepsilon_{\mathrm{GG}} < 1$ almost surely when $p > 1$.

The Huynh-Feldt correction (Huynh and Feldt, 1976) is

$$\varepsilon_{\mathrm{HF}} = \frac{(f+1)p\varepsilon_{\mathrm{GG}} - 2}{p(f - p\varepsilon_{\mathrm{GG}})}$$

where f is the number of degrees of freedom for the empirical covariance matrix. (The original paper has N instead of $f + 1$ in the numerator for the split-plot design. A correction was published 15 years later

(Lecoutre, 1991), but another 15 years later, this error is still present in SAS and SPSS.)

Notice that ε_{HF} can be larger than one, in which case you should use the uncorrected F test. Also, ε_{HF} is obtained by bias-correcting the numerator and denominator of ε_{GG}, which is not guaranteed to be helpful, and simulation studies in (Huynh and Feldt, 1976) indicate that it actually makes the approximations worse when $\varepsilon_{GG} < 0.7$ or so.

Implementation in R

Basics

Quite a lot of the calculations could be copied from the existing `manova` and `summary.manova` code for balanced designs. Also, objects of class `mlm`, inheriting from `lm`, were already defined as the output from `lm(Y~....)` when `Y` is a matrix.

It was necessary to add the following new functions

- `SSD` creates an object of (S3) class `SSD` which is the sums of squares and products matrix augmented by degrees of freedom and information about the call. This is a generic function with a method for `mlm` objects.
- `estVar` calculates the estimated variance-covariance matrix. It has methods for `SSD` and `mlm` objects. In the former case, it just normalizes the SSD by the degrees of freedom, and in the latter case it calculates `SSD(object)` and then applies the `SSD` method.
- `anova.mlm` is used to compare two multivariate linear models or partition a single model in the usual cumulative way. The various multivariate tests can be specified using `test="Pillai"`, etc., just as in `summary.manova`. In addition, it is possible to specify `test="Spherical"` which gives the F test under assumption of sphericity.
- `mauchly.test` tests for sphericity of a covariance matrix with respect to another.
- `sphericity` calculates the ε_{GG} and ε_{HF} based on the SSD matrix and its degrees of freedom. This function is private to the `stats` namespace (at least currently, as of R version 2.6.0) and used to generate the headings and adjusted p-values in `anova.mlm`.

Representing transformations

As previously described, sphericity tests and tests of linear hypotheses may make sense only for a transformed response. Hence we need to be able to specify transformations in `anova.mlm` and `mauchly.test`. The code has several ways to deal with this:

- The transformation matrix can be given directly using the argument `T`.
- It is possible to specify the arguments `X` and `M` as the matrices described previously. The defaults are set so that the default transformation is the identity.
- It is also possible to specify `X` and/or `M` using model formulas. In that case, they usually need to refer to a data frame which describes the intra-subject design. This can be given in the `idata` argument. The default for `idata` is a data frame consisting of the single variable `index=1:p` which may be used to specify, e.g., a polynomial response for equispaced data.

Example

These data from the book by Maxwell and Delaney (1990) are also used by Baron and Li (2006). They show reaction times where ten subjects respond to stimuli in the absence and presence of ambient noise, and using stimuli tilted at three different angles.

```
> reacttime <- matrix(c(
+ 420, 420, 480, 480, 600, 780,
+ 420, 480, 480, 360, 480, 600,
+ 480, 480, 540, 660, 780, 780,
+ 420, 540, 540, 480, 780, 900,
+ 540, 660, 540, 480, 660, 720,
+ 360, 420, 360, 360, 480, 540,
+ 480, 480, 600, 540, 720, 840,
+ 480, 600, 660, 540, 720, 900,
+ 540, 600, 540, 480, 720, 780,
+ 480, 420, 540, 540, 660, 780),
+ ncol = 6, byrow = TRUE,
+ dimnames=list(subj=1:10,
+       cond=c("deg0NA", "deg4NA", "deg8NA",
+              "deg0NP", "deg4NP", "deg8NP")))
```

The same data are used by `example(estVar)` and `example(anova.mlm)`, so you can load the `reacttime` matrix just by running the examples. The following is mainly an expanded explanation of those examples.

First let us calculate the estimated covariance matrix:

```
> mlmfit <- lm(reacttime~1)
> estVar(mlmfit)
         cond
cond      deg0NA deg4NA deg8NA deg0NP deg4NP deg8NP
  deg0NA    3240   3400   2960   2640   3600   2840
  deg4NA    3400   7400   3600    800   4000   3400
  deg8NA    2960   3600   6240   4560   6400   7760
  deg0NP    2640    800   4560   7840   8000   7040
  deg4NP    3600   4000   6400   8000  12000  11200
  deg8NP    2840   3400   7760   7040  11200  13640
```

In this case there is no between-subjects structure, except for a common mean, so the result is equivalent to `var(reacttime)`.

Next we consider tests for whether the response depends on the design at all. We generate a contrast transformation which is orthogonal to an intercept-only within-subject model; this is equivalent to any full-rank set of within-subject contrasts. A test based on multivariate normal theory is performed as follows.

```
> mlmfit0 <- update(mlmfit, ~0)
> anova(mlmfit, mlmfit0, X=~1)
Analysis of Variance Table

Model 1: reacttime ~ 1
Model 2: reacttime ~ 1 - 1

Contrasts orthogonal to
~1

  Res.Df Df Gen.var.  Pillai approx F
1      9     1249.57
2     10  1  2013.16    0.95    17.38
  num Df den Df   Pr(>F)
1
2      5      5 0.003534 **
```

This gives the default Pillai test, but actually, you get the same result with the other multivariate tests since the degrees of freedom (per coordinate) only changes by one.

To perform the same test, but assuming sphericity of the covariance matrix, just add an argument to the anova call:

```
> anova(mlmfit, mlmfit0, X=~1, test="Spherical")
Analysis of Variance Table

Model 1: reacttime ~ 1
Model 2: reacttime ~ 1 - 1

Contrasts orthogonal to
~1

Greenhouse-Geisser epsilon: 0.4855
Huynh-Feldt epsilon:        0.6778

  Res.Df Df Gen.var.      F num Df
1      9      1249.6
2     10  1   2013.2 38.028      5
  den Df    Pr(>F)    G-G Pr    H-F Pr
1
2     45 4.471e-15 2.532e-08 7.393e-11
```

It can be noticed that the epsilon corrections in this case are rather strong, but that the corrected p-values nevertheless are considerably more significant with this approach, reflecting the low power of the multivariate test. This is commonly the case when the number of replications is low.

To test the hypothesis of sphericity, we employ Mauchly's criterion:

```
> mauchly.test(mlmfit, X=~1)

        Mauchly's test of sphericity
        Contrasts orthogonal to
        ~1

data:  SSD matrix from lm(formula = reacttime ~ 1)

W = 0.0311, p-value = 0.04765
```

Accordingly, the hypothesis of sphericity is rejected at the 0.05 significance level.

However, the analysis of contrasts completely disregards the two-way intrasubject design. To generate tests for overall effects of `deg` and `noise`, as well as interaction between the two, we do as follows: First we need to set up the `idata` data frame to describe the design.

```
> idata <- expand.grid(
+             deg=c("0", "4", "8"),
+             noise=c("A", "P"))
> idata
  deg noise
1   0     A
2   4     A
3   8     A
4   0     P
5   4     P
6   8     P
```

Then we can specify the tests using model formulas to specify the X and M matrices. To test for an effect of `deg` we let `M` specify an additive model in `deg` and `noise` and let `X` be the model with `noise` alone.

```
> anova(mlmfit, mlmfit0, M = ~ deg + noise,
+       X = ~ noise,
+       idata = idata, test="Spherical")
Analysis of Variance Table

Model 1: reacttime ~ 1
Model 2: reacttime ~ 1 - 1

Contrasts orthogonal to
~noise


Contrasts spanned by
~deg + noise

Greenhouse-Geisser epsilon: 0.9616
Huynh-Feldt epsilon:        1.2176

  Res.Df Df Gen.var.      F num Df
1      9      1007.0
2     10  1   2703.2 40.719      2
  den Df    Pr(>F)    G-G Pr    H-F Pr
1
2     18 2.087e-07 3.402e-07 2.087e-07
```

It might at this point be helpful to explain the roles of `X` and `M` a bit further. To do this, we need to access an internal function in the `stats` namespace, namely `proj.matrix`. This function calculates

the projection matrix for a given design matrix, i.e. $P_X = X(X'X)^{-1}X'$. In the above context, we have

```
M <- model.matrix(~deg+noise, data=idata)
P1 <- stats:::proj.matrix(M)
X <- model.matrix(~noise, data=idata)
P2 <- stats:::proj.matrix(X)
```

The two design matrices are (eliminating attributes)

```
> M
  (Intercept) deg4 deg8 noiseP
1           1    0    0      0
2           1    1    0      0
3           1    0    1      0
4           1    0    0      1
5           1    1    0      1
6           1    0    1      1
> X
  (Intercept) noiseP
1           1      0
2           1      0
3           1      0
4           1      1
5           1      1
6           1      1
```

For printing the projection matrices, it is useful to include the `fractions` function from `MASS`.

```
> library("MASS")
> fractions(P1)
  1    2    3    4    5    6
1  2/3  1/6  1/6  1/3 -1/6 -1/6
2  1/6  2/3  1/6 -1/6  1/3 -1/6
3  1/6  1/6  2/3 -1/6 -1/6  1/3
4  1/3 -1/6 -1/6  2/3  1/6  1/6
5 -1/6  1/3 -1/6  1/6  2/3  1/6
6 -1/6 -1/6  1/3  1/6  1/6  2/3
> fractions(P2)
  1   2   3   4   5   6
1 1/3 1/3 1/3   0   0   0
2 1/3 1/3 1/3   0   0   0
3 1/3 1/3 1/3   0   0   0
4   0   0   0 1/3 1/3 1/3
5   0   0   0 1/3 1/3 1/3
6   0   0   0 1/3 1/3 1/3
```

Here, `P2` is readily recognized as the operator that replaces each of the first three values by their average, and similarly the last three. The other one, `P1`, is a little harder, but if you look long enough, you will recognize the formula $\hat{x}_{ij} = \bar{x}_{i\cdot} + \bar{x}_{\cdot j} - \bar{x}_{\cdot\cdot}$ from two-way ANOVA.

The transformation T is the difference between `P1` and `P2`

```
> fractions(P1-P2)
  1    2    3    4    5    6
1  1/3 -1/6 -1/6  1/3 -1/6 -1/6
2 -1/6  1/3 -1/6 -1/6  1/3 -1/6
3 -1/6 -1/6  1/3 -1/6 -1/6  1/3
4  1/3 -1/6 -1/6  1/3 -1/6 -1/6
5 -1/6  1/3 -1/6 -1/6  1/3 -1/6
6 -1/6 -1/6  1/3 -1/6 -1/6  1/3
```

This is the representation of $\bar{x}_{i\cdot} - \bar{x}_{\cdot\cdot}$ where i and j index levels of `deg` and `noise`, respectively. The matrix has (row) rank 2; for the actual computations, only the first two rows are used.

The important aspect of this matrix is that it assigns equal weights to observations at different levels of `noise` and that it constructs a maximal set of within-`deg` differences. Any other such choice of transformation leads to equivalent results, since it will be a full-rank transformation of T. For instance:

```
> T1 <- rbind(c(1,-1,0,1,-1,0),c(1,0,-1,1,0,-1))
> anova(mlmfit, mlmfit0, T=T1,
+     idata = idata, test = "Spherical")
Analysis of Variance Table

Model 1: reacttime ~ 1
Model 2: reacttime ~ 1 - 1

Contrast matrix
 1 -1  0  1 -1  0
 1  0 -1  1  0 -1
Greenhouse-Geisser epsilon: 0.9616
Huynh-Feldt epsilon:        1.2176

  Res.Df Df Gen.var.      F num Df
1      9       12084
2     10  1    32438 40.719      2
1 den Df    Pr(>F)    G-G Pr    H-F Pr
2     18 2.087e-07 3.402e-07 2.087e-07
```

This differs from the previous analysis only in the `Gen.var.` column, which is not invariant to scaling and transformation of data.

Returning to the interpretation of the results, notice that the epsilons are much closer to one than before. This is consistent with a covariance structure induced by a mixed model containing a random interaction between subject and `deg`. If we perform the similar procedure for `noise` we get epsilons of exactly 1, because we are dealing with a single-df contrast.

```
> anova(mlmfit, mlmfit0, M = ~ deg + noise,
+       X = ~ deg,
+       idata = idata, test="Spherical")
Analysis of Variance Table

Model 1: reacttime ~ 1
Model 2: reacttime ~ 1 - 1

Contrasts orthogonal to
~deg


Contrasts spanned by
~deg + noise

Greenhouse-Geisser epsilon: 1
Huynh-Feldt epsilon:        1

  Res.Df Df Gen.var.      F num Df
1      9        1410
```

```
2     10  1     6030 33.766      1
  den Df    Pr(>F)    G-G Pr    H-F Pr
1
2      9 0.0002560 0.0002560 0.0002560
```

However, we really shouldn't be doing these tests on individual effects of `deg` and `noise` if there is interaction between the two, and as the following output shows, there is:

```
> anova(mlmfit, mlmfit0, X = ~ deg + noise,
+       idata = idata, test = "Spherical")
Analysis of Variance Table

Model 1: reacttime ~ 1
Model 2: reacttime ~ 1 - 1

Contrasts orthogonal to
~deg + noise

Greenhouse-Geisser epsilon: 0.904
Huynh-Feldt epsilon:        1.118

  Res.Df Df Gen.var.     F num Df
1      9      316.58
2     10  1   996.34 45.31      2
  den Df    Pr(>F)    G-G Pr    H-F Pr
1
2     18 9.424e-08 3.454e-07 9.424e-08
```

Finally, to test between-within interactions, rephrase them as effects of between-factors on within-contrasts. To illustrate this, we introduce a fake grouping `f` of the `reacttime` data into two groups. The interaction between `f` and the (implied) column factor is obtained by testing whether the contrasts orthogonal to `~1` depend on `f`.

```
> f <- factor(rep(1:2, 5))
> mlmfit2 <- update(mlmfit, ~f)
> anova(mlmfit2, X = ~1, test = "Spherical")
Analysis of Variance Table


Contrasts orthogonal to
~1

Greenhouse-Geisser epsilon: 0.4691
Huynh-Feldt epsilon:        0.6758

            Df       F num Df den Df
(Intercept)  1 34.9615      5     40
f            1  0.2743      5     40
Residuals    8
               Pr(>F)    G-G Pr    H-F Pr
(Intercept) 1.382e-13 2.207e-07 8.254e-10
f             0.92452   0.79608   0.86456
Residuals
```

More detailed tests of interactions between `f` and `deg`, `f` and `noise`, and the three-way interaction can be constructed using `M` and `X` settings as described previously.

Bibliography

T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Wiley, New York, 1958.

J. Baron and Y. Li. Notes on the use of R for psychology experiments and questionnaires, 2006. URL `http://www.psych.upenn.edu/~baron/rpsych/rpsych.html`.

G. E. P. Box. Some theorems on quadratic forms applied in the study of analysis of variance problems, i. effect of inequality of variance in the one-way classification. *Annals of Mathematical Statistics*, 25 (2):290–302, 1954a.

G. E. P. Box. Some theorems on quadratic forms applied in the study of analysis of variance problems, ii.effect of inequality of variance and of correlation between errors in the two-way classification. *Annals of Mathematical Statistics*, 25(3):484–498, 1954b.

S. W. Greenhouse and S. Geisser. On methods in the analysis of profile data. *Psychometrika*, 24:95–112, 1959.

D. J. Hand and C. C. Taylor. *Multivariate Analysis of Variance and Repeated Measures*. Chapman and Hall, 1987.

H. Huynh and L. S. Feldt. Estimation of the box correction for degrees of freedom from sample data in randomized block and split-plot designs. *Journal of Educational Statistics*, 1(1):69–82, 1976.

B. Lecoutre. A correction for the $\tilde{\epsilon}$ approximate test in repeated measures designs with two or more independent groups. *Journal of Educational Statistics*, 16(4):371–372, 1991.

S. E. Maxwell and H. D. Delaney. *Designing Experiments and Analyzing Data: A model comparison perspective*. Brooks/Cole, Pacific Grove, CA, 1990.

Peter Dalgaard
Deptartment of Biostatistics
University of Copenhagen
`Peter.Dalgaard@R-project.org`

gnm: A Package for Generalized Nonlinear Models

by Heather Turner and David Firth

In a generalized nonlinear model, the expectation of a response variable Y is related to a predictor η — a possibly nonlinear function of parameters — via a link function g:

$$g[E(Y)] = \eta(\beta)$$

and the variance of Y is equal to, or proportional to, a known function $v[E(Y)]$. This class of models may be thought of as extending the class of generalized linear models by allowing nonlinear terms in the predictor, or extending the class of nonlinear least squares models by allowing the variance of the response to depend on the mean.

The **gnm** package provides facilities for the specification, estimation and inspection of generalized nonlinear models. Models are specified via symbolic formulae, with nonlinear terms specified by functions of class `"nonlin"`. The fitting procedure uses an iterative weighted least squares algorithm, adapted to work with over-parameterized models. Identifiability constraints are not essential for model specification and estimation, and so the difficulty of automatically defining such constraints for the entire family of generalized nonlinear models is avoided. Constraints may be pre-specified, if desired; otherwise, functions are provided in the package for conducting inference on identifiable parameter combinations after a model in an over-parameterized representation has been fitted.

Specific models which the package may be used to fit include models with multiplicative interactions, such as row-column association models (Goodman, 1979), UNIDIFF (uniform difference) models for social mobility (Xie, 1992; Erikson and Goldthorpe, 1992), GAMMI (generalized additive main effects and multiplicative interaction) models (e.g. van Eeuwijk, 1995), and Lee-Carter models for trends in age-specific mortality (Lee and Carter, 1992); diagonal-reference models for dependence on a square or hyper-square classification (Sobel, 1981, 1985); Rasch-type logit or probit models for legislative voting (e.g. de Leeuw, 2006); and stereotype multinomial regression models for ordinal response (Anderson, 1984). A comprehensive manual is distributed with the package (see `vignette("gnmOverview", package = "gnm")`) and this manual may also be downloaded from `http://go.warwick.ac.uk/gnm`. Here we give an introduction to the key functions and provide some illustrative applications.

Key Functions

The primary function defined in package **gnm** is the model-fitting function of the same name. This function is patterned after `glm` (the function included in the standard **stats** package for fitting generalized linear models), taking similar arguments and returning an object of class `c("gnm", "glm", "lm")`.

A model formula is specified to `gnm` as the first argument. The conventional symbolic form is used to specify linear terms, whilst nonlinear terms are specified using functions of class `"nonlin"`. The `nonlin` functions currently exported in **gnm** are summarized in Figure 1. These functions enable the specification of basic mathematical functions of predictors (`Exp`, `Inv` and `Mult`) and some more specialized nonlinear terms (`MultHomog`, `Dref`). Often arguments of `nonlin` functions may themselves be parametric functions described in symbolic form. For example, an exponential decay model with additive errors

$$y = \alpha + \exp(\beta + \gamma x) + e \qquad (1)$$

is specified using the `Exp` function as

```
gnm(y ~ Exp(1 + x))
```

These "sub-formulae" (to which intercepts are *not* added by default) can include other `nonlin` functions, allowing more complex nonlinear terms to be built up. Users may also define custom `nonlin` functions, as we shall illustrate later.

`Dref`	to specify a diagonal reference term
`Exp`	to specify the exponential of a predictor
`Inv`	to specify the reciprocal of a predictor
`Mult`	to specify a product of predictors
`MultHomog`	to specify a multiplicative interaction with homogeneous effects

Figure 1: `"nonlin"` functions in the **gnm** package.

The remaining arguments to gnm are mainly control parameters and are of secondary importance. However, the `eliminate` argument — which implements a feature seen previously in the *GLIM 4* statistical modelling system — can be extremely useful when a model contains the additive effect of a (typically 'nuisance') factor with a large number of levels; in such circumstances the use of `eliminate` can substantially improve computational efficiency, as well as allowing more convenient model summaries with the 'nuisance' factor omitted.

Most of the methods and accessor functions implemented for `"glm"` or `"lm"` objects are also implemented for `"gnm"` objects, such as `print`, `summary`, `plot`, `coef`, and so on. Since `"gnm"` models may be

over-parameterized, care is needed when interpreting the output of some of these functions. In particular, for parameters that are not identified, an arbitrary parameterization will be returned and the standard error will be displayed as `NA`.

For estimable parameters, methods for `profile` and `confint` enable inference based on the profile likelihood. These methods are designed to handle the asymmetric behaviour of the log-likelihood function that is a well-known feature of many nonlinear models.

Estimates and standard errors of simple "sum-to-zero" contrasts can be obtained using `getContrasts`, if such contrasts are estimable. For more general linear combinations of parameters, the lower-level `se` function may be used instead.

Example

We illustrate here the use of some of the key functions in **gnm** through the analysis of a contingency table from Goodman (1979). The data are a cross-classification of a sample of British males according to each subject's occupational status and his father's occupational status. The contingency table is distributed with **gnm** as the data set `occupationalStatus`.

Goodman (1979) analysed these data using row-column association models, in which the interaction between the row and column factors is represented by components of a multiplicative interaction. The log-multiplicative form of the RC(1) model — the row-column association model with one component of the multiplicative interaction — is given by

$$\log \mu_{rc} = \alpha_r + \beta_c + \gamma_r \delta_c, \qquad (2)$$

where μ_{rc} is the cell mean for row r and column c. Before modelling the occupational status data, Goodman (1979) first deleted cells on the main diagonal. Equivalently we can separate out the diagonal effects as follows:

$$\log \mu_{rc} = \alpha_r + \beta_c + \theta_{rc} + \gamma_r \delta_c \qquad (3)$$

where

$$\theta_{rc} = \begin{cases} \phi_r & \text{if } r = c \\ 0 & \text{otherwise.} \end{cases}$$

In addition to the `"nonlin"` functions provided by **gnm**, the package includes a number of functions for specifying structured linear interactions. The diagonal effects in model 3 can be specified using the **gnm** function `Diag`. Thus we can fit model 3 as follows:

```
> set.seed(1)
> RC <- gnm(Freq ~ origin + destination +
+     Diag(origin, destination) +
+     Mult(origin, destination),
+     family = poisson,
+     data = occupationalStatus)
```

An abbreviated version of the output given by `summary(RC)` is shown in Figure 2. Default constraints (as specified by `options("contrasts")`) are applied to linear terms: in this case the first level of the `origin` and `destination` factors have been set to zero. However the "main effects" are not estimable, since they are aliased with the unconstrained multiplicative interaction. If the `gnm` call were re-evaluated from a different random seed, the parameterization for the main effects and multiplicative interaction would differ from that shown in Figure 2. On the other hand the diagonal effects, which are essentially contrasts with the off-diagonal elements, are identified here. This is immediately apparent from Figure 2, since standard errors are available for these parameters.

One could consider extending the model by adding a further component of the multiplicative interaction, giving an RC(2) model:

$$\log \mu_{rc} = \alpha_r + \beta_c + \theta_{rc} + \gamma_r \delta_c + \theta_r \phi_c. \qquad (4)$$

Most of the `"nonlin"` functions, including `Mult`, have an `inst` argument to allow the specification of multiple instances of a term, as here:

```
Freq ~ origin + destination +
    Diag(origin, destination) +
    Mult(origin, destination, inst = 1) +
    Mult(origin, destination, inst = 2)
```

Multiple instances of a term with an `inst` argument may be specified in shorthand using the `instances` function provided in the package:

```
Freq ~ origin + destination +
    Diag(origin, destination) +
    instances(Mult(origin, destination), 2)
```

The formula is then expanded by `gnm` before the model is fitted.

In the case of the occupational status data however, the RC(1) model is a good fit (a residual deviance of 29.149 on 28 degrees of freedom). So rather than extending the model, we shall see if it can be simplified by using homogeneous scores in the multiplicative interaction:

$$\log \mu_{rc} = \alpha_r + \beta_c + \theta_{rc} + \gamma_r \gamma_c \qquad (5)$$

This model can be obtained by updating `RC` using `update`:

```
> RChomog <- update(RC, . ~ .
+     - Mult(origin, destination)
+     + MultHomog(origin, destination))
```

We can compare the two models using `anova`, which shows that there is little gained by allowing heterogeneous row and column scores in the association of the fathers' occupational status and the sons' occupational status:

```
Call:
gnm(formula = Freq ~ origin + destination + Diag(origin, destination) +
    Mult(origin, destination), family = poisson, data = occupationalStatus)

Deviance Residuals: ...

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                   0.11881         NA      NA       NA
origin2                       0.49005         NA      NA       NA
...
origin8                       1.60214         NA      NA       NA
destination2                  0.95334         NA      NA       NA
...
destination8                  1.42813         NA      NA       NA
Diag(origin, destination)1    1.47923    0.45401   3.258  0.00112
...
Diag(origin, destination)8    0.40731    0.21930   1.857  0.06327
Mult(., destination).origin1 -1.74022         NA      NA       NA
...
Mult(., destination).origin8  1.65900         NA      NA       NA
Mult(origin, .).destination1 -1.32971         NA      NA       NA
...
Mult(origin, .).destination8  0.66730         NA      NA       NA

...
Residual deviance: 29.149 on 28 degrees of freedom
...
```

Figure 2: Abbreviated summary of model `RC`.

```
> anova(RChomog, RC)
Analysis of Deviance Table

Model 1: Freq ~ origin + destination +
    Diag(origin, destination) +
    MultHomog(origin, destination)
Model 2: Freq ~ origin + destination +
    Diag(origin, destination) +
    Mult(origin, destination)
  Resid. Df Resid. Dev Df Deviance
1        34     32.561
2        28     29.149  6    3.412
```

All of the parameters in model 5 can be made identifiable by constraining one level of the homogeneous multiplicative factor to zero. This can be achieved by using the `constrain` argument to `gnm`, but this would require re-fitting the model. As we are only really interested in the parameters of the multiplicative interaction, we investigate simple contrasts of these parameters using `getContrasts`. The **gnm** function `pickCoef` enables us to select the parameters of interest using a regular expression to match against the parameter names:

```
> contr <- getContrasts(RChomog,
+     pickCoef(RChomog, "MultHomog"))
```

A summary of `contr` is shown in Figure 3. By default, `getContrasts` sets the first level of the homogeneous multiplicative factor to zero. The quasi standard errors and quasi variances are independent of parameterization; see Firth (2003) and Firth and de Menezes (2004) for more detail. In this example the reported relative-error diagnostics indicate that the quasi-variance approximation is rather inaccurate — in contrast to the high accuracy found in many other situations by Firth and de Menezes (2004).

Users may run through the example covered in this section by calling `demo(gnm)`.

Custom `"nonlin"` functions

The small number of `"nonlin"` functions currently distributed with **gnm** allow a large class of generalized nonlinear models to be specified, as exemplified through the package documentation. Nevertheless, for some models a custom `"nonlin"` function may be necessary or desirable. As an illustration, we shall consider a logistic predictor of the form given by `SSlogis` (a selfStart model for use with the **stats** function `nls`),

$$\frac{Asym}{1 + \exp((xmid - x)/scal)}. \tag{6}$$

This predictor could be specified to `gnm` as

```
~ -1 + Mult(1, Inv(Const(1) +
    Exp(Mult(1 + offset(-x), Inv(1)))))
```

```
Model call:  gnm(formula = Freq ~ origin + destination + Diag(origin, destination)
    + MultHomog(origin, destination), family = poisson, data = occupationalStatus)
                                  estimate    SE quasiSE quasiVar
    MultHomog(origin, destination)1    0.000 0.000  0.1573  0.02473
    MultHomog(origin, destination)2    0.218 0.235  0.1190  0.01416
    MultHomog(origin, destination)3    0.816 0.167  0.0611  0.00374
    MultHomog(origin, destination)4    1.400 0.160  0.0518  0.00269
    MultHomog(origin, destination)5    1.418 0.172  0.0798  0.00637
    MultHomog(origin, destination)6    1.929 0.157  0.0360  0.00129
    MultHomog(origin, destination)7    2.345 0.173  0.0796  0.00634
    MultHomog(origin, destination)8    2.589 0.189  0.1095  0.01200
Worst relative errors in SEs of simple contrasts (%):  -19 4.4
Worst relative errors over *all* contrasts (%):  -17.2 51.8
```

Figure 3: Simple contrasts of homogeneous row and column scores in model `RChomog`.

where `Mult`, `Inv` and `Exp` are `"nonlin"` functions, `offset` is the usual **stats** function and `Const` is a **gnm** function specifying a constant offset. However, this specification is rather convoluted and it would be preferable to define a single `"nonlin"` function that could specify a logistic term.

Our custom `"nonlin"` function only needs a single argument, to specify the variable x, so the skeleton of our definition might be as follows:

```
Logistic <- function(x){
    }
class(Logistic) <- "nonlin"
```

Now we need to fill in the body of the function. The purpose of a `"nonlin"` function is to convert its arguments into a list of arguments for the internal function `nonlinTerms`. This function considers a nonlinear term as a mathematical function of *predictors*, the parameters of which need to be estimated, and possibly also *variables*, which have a coefficient of 1. In this terminology, *Asym*, *xmid* and *scal* in Equation 6 are parameters of intercept-only predictors, whilst x is a variable. Our `Logistic` function must specify the corresponding `predictors` and `variables` arguments of `nonlinTerms`. We can define the `predictors` argument as a named list of symbolic expressions

```
predictors = list(Asym = 1, xmid = 1,
    scal = 1)
```

and pass the user-specified variable `x` as the `variables` argument:

```
variables = list(substitute(x))
```

We must then define the `term` argument of `nonlinTerms`, which is a function that creates a deparsed mathematical expression of the nonlinear term from labels for the predictors and variables. These labels should be passed through arguments `predLabels` and `varLabels` respectively, so we can specify the `term` argument as

```
term = function(predLabels, varLabels){
    paste(predLabels[1], "/(1 + exp((",
    predLabels[2], "-", varLabels[1], ")/",
    predLabels[3], "))")
    }
```

Our `Logistic` function need not specify any further arguments of `nonlinTerms`. However, we do have an idea of useful starting values, so we can also specify the `start` argument. This should be a function which takes a named vector of the parameters in the term and returns a vector of starting values for these parameters. The following function will set the initial scale parameter to one:

```
start = function(theta){
    theta[3] <- 1
    theta
    }
```

Putting these components together, we have:

```
Logistic <- function(x){
    list(predictors =
        list(Asym = 1, xmid = 1, scal = 1),
        variables = list(substitute(x)),
        term =
        function(predLabels, varLabels){
            paste(predLabels[1],
            "/(1 + exp((", predLabels[2],
            "-", varLabels[1], ")/",
            predLabels[3], "))")
        },
        start = function(theta){
            theta[3] <- 1
            theta
        })
}
class(Logistic) <- "nonlin"
```

Having defined this function, we can reproduce the example from `?SSlogis` as shown below

```
> Chick.1 <-
+ ChickWeight[ChickWeight$Chick == 1, ]
```

```
> fm1gnm <- gnm(weight ~ Logistic(Time) - 1,
+    data = Chick.1, trace = FALSE)
> summary(fm1gnm)

Call:
gnm(formula = weight ~ Logistic(Time) - 1,
   data = Chick.1)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-4.154  -2.359   0.825   2.146   3.745

Coefficients:
     Estimate Std. Error t value Pr(>|t|)
Asym 937.0205   465.8569   2.011  0.07516
xmid  35.2228     8.3119   4.238  0.00218
scal  11.4052     0.9052  12.599 5.08e-07

(Dispersion parameter for gaussian family
   taken to be 8.51804)

Residual deviance: 76.662 on 9 degrees of
   freedom
AIC: 64.309

Number of iterations: 14
```

In general, whenever a nonlinear term can be represented as an expression that is differentiable using `deriv`, it should be possible to define a `"nonlin"` function to specify the term. Additional arguments of `nonlinTerms` allow for the specification of factors with homologous effects and control over the way in which parameters are automatically labelled.

Summary

The functions distributed in the **gnm** package enable a wide range of generalized nonlinear models to be estimated. Nonlinear terms that are not (easily) represented by the `"nonlin"` functions provided may be implemented as custom `"nonlin"` functions, under fairly weak conditions.

There are several features of **gnm** that we have not covered in this paper. Some facilitate model inspection, such as the `ofInterest` argument to `gnm` which allows the user to customize model summaries and simplify parameter selection. Other features are designed for particular models, such as the `residSVD` function for generating good starting values for a multiplicative interaction. Still others relate to particular types of data, such as the `expandCategorical` function for expressing categorical data as counts. These features complement the facilities we have already described, to produce a substantial and flexible package for working with generalized nonlinear models.

Acknowledgments

This work was supported by the Economic and Social Research Council (UK) through Professorial Fellowship RES-051-27-0055.

Bibliography

J. A. Anderson. Regression and ordered categorical variables. *J. R. Statist. Soc. B*, 46(1):1–30, 1984.

J. de Leeuw. Principal component analysis of binary data by iterated singular value decomposition. *Comp. Stat. Data Anal.*, 50(1):21–39, 2006.

R. Erikson and J. H. Goldthorpe. *The Constant Flux*. Oxford: Clarendon Press, 1992.

D. Firth. Overcoming the reference category problem in the presentation of statistical models. *Sociological Methodology*, 33:1–18, 2003.

D. Firth and R. X. de Menezes. Quasi-variances. *Biometrika*, 91:65–80, 2004.

L. A. Goodman. Simple models for the analysis of association in cross-classifications having ordered categories. *J. Amer. Statist. Assoc.*, 74:537–552, 1979.

R. D. Lee and L. Carter. Modelling and forecasting the time series of US mortality. *Journal of the American Statistical Association*, 87:659–671, 1992.

M. E. Sobel. Diagonal mobility models: A substantively motivated class of designs for the analysis of mobility effects. *Amer. Soc. Rev.*, 46:893–906, 1981.

M. E. Sobel. Social mobility and fertility revisited: Some new models for the analysis of the mobility effects hypothesis. *Amer. Soc. Rev.*, 50:699–712, 1985.

F. A. van Eeuwijk. Multiplicative interaction in generalized linear models. *Biometrics*, 51:1017–1032, 1995.

Y. Xie. The log-multiplicative layer effect model for comparing mobility tables. *American Sociological Review*, 57:380–395, 1992.

Heather Turner
University of Warwick, UK
`Heather.Turner@warwick.ac.uk`

David Firth
University of Warwick, UK
`David.Firth@warwick.ac.uk`

fmri: A Package for Analyzing fmri Data

by J. Polzehl and K. Tabelow

This article describes the usage of the R package **fmri** to analyze single time series BOLD fMRI (blood-oxygen-level dependent functional magnetic resonance imaging) data using structure adaptive smoothing procedures (Propagation- Separation approach) as described in (Tabelow et al., 2006). See (J. Polzehl and K. Tabelow, 2006) for an extended documentation.

Analysing fMRI data with the fmri package

The approach implemented in the **fmri** package is based on a linear model for the hemodynamic response and structural adaptive spatial smoothing of the resulting Statistical Parametric Map (SPM). The package requires R (version $\geq$ 2.2). 3D visualization needs the R package **tkrplot**.

The statistical modeling implemented with this software is described in (Tabelow et al., 2006).

> **NOTE!** This software comes with absolutely **no warranty**! It is **not** intended for clinical use, but for evaluation purposes only. Absence of bugs can not be guaranteed!

We first start with a basic script for using the **fmri** package in a typical fmri analysis. In the following sections we describe the consecutive steps of the analysis.

```
# read the data
data <- read.AFNI("afnifile")
# or read sequence of ANALYZE files
# analyze031file.hdr ... analyze137file.hdr
# data <- read.ANALYZE("analyze",
#              numbered = TRUE, "file", 31, 107)

# create expected BOLD signal and design matrix
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
x <- fmri.design(hrf)

# generate parametric map from linear model
spm <- fmri.lm(data, x)

# adaptive smoothing with maximum bandwith hmax
spmsmooth <- fmri.smooth(spm, hmax = 4)

# calculate p-values for smoothed parametric map
pvalue <- fmri.pvalue(spmsmooth)

# write slicewise results into a file or ...
plot(pvalue, maxpvalue = 0.01, device = "jpeg",
                           file = "result.jpeg")
# ... use interactive 3D visualization
plot(pvalue, maxpvalue = 0.01, type = "3d")
```

Reading the data

The **fmri** package can read ANALYZE (Mayo Foundation, 2001), AFNI- HEAD/BRIK (Cox, 1996), NIFTI and DICOM files. Use

```
data <- read.AFNI(<filename>)
data <- read.ANALYZE(<filename>)
```

to create the object `data` from the data in file 'filename'. Drop the extension in 'filename'. While AFNI data is generally given as a four dimensional datacube in one file, ANALYZE format data often comes in a series of numbered files. A sequence of images in ANALYZE format can be imported by

```
data <- read.ANALYZE(prefix = "", numbered = TRUE,
                     postfix = "", picstart = 1,
                     numbpic = 1)
```

Setting the argument `numbered` to TRUE allows to read a list of image files. The image files are assumed to be ordered by a number, contained in the filename. `picstart` is the number of the first file, and `numbpic` the number of files to be read. The increment between consecutive numbers is assumed to be 1. `prefix` specifies a string preceeding the number, whereas `postfix` identifies a string following the number in the filename. Note, that `read.ANALYZE()` requires the file names in the form: '<prefix>number<postfix>.hdr/img'.

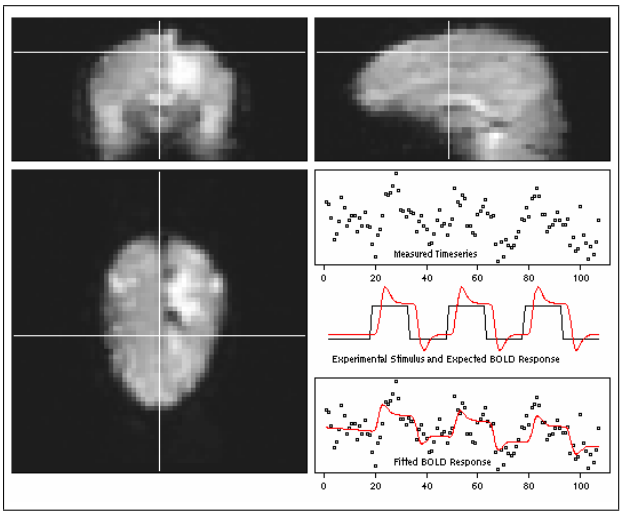

Figure 1: View of a typical fMRI dataset. Data dimension is 64x64x26 with a total of 107 scans. A high noise level is typical. The time series is described by a linear model containing the experimental stimulus (red) and quadratic drift. Data: courtesy of H. Voss, Weill Medical College of Cornell University.

Both functions return lists of class "fmridata" with components containing the datacube ('ttt'). See figure 1 for an illustration. The data cube is stored

in ('raw') format to save memory. The data can be extracted from the list using `extract.data()`. Additionally the list contains basic information like the size of the data cube ('dim') and the voxel size ('delta'), as well as the complete header information ('header'), which is itself a list with components depending to the data format read. A head mask is defined by simply using a 75% quantile of the data grey levels as cut- off. This is only be used to provide improved spatial correlation estimates for the head in `fmri.lm()`.

Expected BOLD response

In voxel affected by the experiment the observed signal is assumed to follow the expected Blood Oxygenation Level Dependent (BOLD) signal. This signal depends on both the experimental stimulus, described by a task indicator function and a hemodynamic response function $h(t)$. We define $h(t)$ as the difference of two gamma functions

$$h(t) = \left(\frac{t}{d_1}\right)^{a_1} \exp\left(-\frac{t-d_1}{b_1}\right) - c\left(\frac{t}{d_2}\right)^{a_2} \exp\left(-\frac{t-d_2}{b_2}\right)$$

with $a_1 = 6$, $a_2 = 12$, $b_1 = 0.9$, $b_2 = 0.9$, and $d_i = a_i b_i (i = 1, 2)$, $c = 0.35$ where t is the time in seconds, see (Glover, 1999). The expected BOLD response is given as a discrete convolution of this function with the task indicator function. Use

```
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
```

to create the expected BOLD response for a stimulus with 107 scans, onset times at the 18th, 48th, and 78th scan with a duration of the stimulus of 15 scans, and a time $TR = 2$s between two scans. In case of multiple stimuli the results of `fmri.stimulus()` for the different stimuli should be arranged as columns of a matrix `hrf` using `cbind()`.

The hemodynamic response function may have an unknown latency (Worsley and Taylor, 2005). This can be modeled creating an additional explanatory variable as the first numerical derivative of any experimental stimulus:

```
dhrf <- (c(0,diff(hrf)) + c(diff(hrf),0))/2
```

See the next section for how to include this into the linear model.

Construction of the SPM

We adopt the common view of a linear model for the time series $Y_i = (Y_{it})$ in each voxel i after reconstruction of the raw data and motion correction.

$$Y_i = X\beta_i + \varepsilon_i, \quad (1)$$

where X denotes the design matrix. The design matrix is created by

```
x <- fmri.design(hrf) .
```

This will include polynomial drift terms up to quadratic order. To deviate from this default, the order of the polynomial drift can be specified by a second argument. Use `cbind()` to combine several stimulus vectors into a matrix of stimuli before calling `fmri.design()`.

The first q columns of X contain values of the expected BOLD response for the different stimuli evaluated at scan acquisition times. The other $p - q$ columns are chosen to be orthogonal to the expected BOLD responses and to account for a slowly varying drift and possible other external effects. The error vector ε_i has zero expectation and is assumed to be correlated in time. In order to access the variability of the estimates of β_i correctly we have to take the correlation structure of the error vector ε_i into account. We assume an AR(1) model to be sufficient for commonly used MRI scanners. The autocorrelation coefficients ρ_i are estimated from the residual vector $r_i = (r_{i1}, \ldots, r_{iT})$ of the fitted model (1) as

$$\bar{\rho}_i = \sum_{t=2}^{T} r_{it} r_{i(t-1)} / \sum_{t=1}^{T} r_{it}^2.$$

This estimate of the correlation coefficient is biased due to fitting the linear model (1). We therefore apply the bias correction given by (Worsley et al., 2002) leading to an estimate $\tilde{\rho}_i$.

We then use prewhitening to transform model (1) into a linear model with approximately uncorrelated errors. The prewhitened linear model is obtained by multiplying the terms in (1) with some matrix A_i depending on $\tilde{\rho}_i$. The prewhitening procedure thus results in a new linear model

$$\tilde{Y}_i = \tilde{X}_i \beta_i + \tilde{\varepsilon}_i \quad (2)$$

with $\tilde{Y}_i = A_i Y_i$, $\tilde{X}_i = A_i X$, and $\tilde{\varepsilon}_i = A_i \varepsilon_i$. In the new model the errors $\tilde{\varepsilon}_i = (\tilde{\varepsilon}_{it})$ are approximately uncorrelated in time t, such that $\operatorname{var} \tilde{\varepsilon}_i = \sigma_i^2 I_T$. Finally least squares estimates $\tilde{\beta}_i$ are obtained from model (2) as

$$\tilde{\beta}_i = (\tilde{X}_i^T \tilde{X}_i)^{-1} \tilde{X}_i^T \tilde{Y}_i.$$

The error variance σ_i^2 is estimated from the residuals $\tilde{r}_i$ of the linear model (2) as $\tilde{\sigma}_i^2 = \sum_1^T \tilde{r}_{it}^2/(T-p)$ leading to estimated covariance matrices

$$\operatorname{var} \tilde{\beta}_i = \tilde{\sigma}_i^2 (\tilde{X}_i^T \tilde{X}_i)^{-1}.$$

Estimates $\tilde{\beta}_i$ and their estimated covariance matrices are, in the simplest case, obtained by

```
spm <- fmri.lm(data, x)
```

where `data` is the data object read by `read.AFNI()` or `read.ANALYZE()`, and `x` is the design matrix created with `fmri.design()`.

See figure 1 for an example of a typical fMRI dataset together with the result of the fit of the linear model to a time series.

To consider more than one stimulus and to estimate an effect

$$\tilde{\gamma} = c^T \tilde{\beta} \tag{3}$$

defined by a vector of contrasts c set the argument `contrast` of the function `fmri.lm()` correspondingly

```
hrf1 <- fmri.stimulus(214, c(18, 78, 138), 15, 2)
hrf2 <- fmri.stimulus(214, c(48, 108, 168), 15, 2)
x <- fmri.design(cbind(hrf1, hrf2))

# stimulus 1 only
spm1 <- fmri.lm(data, x, contrast = c(1,0))

# stimulus 2 only
spm2 <- fmri.lm(data, x, contrast = c(0,1))

# contrast between both
spm3 <- fmri.lm(data, x, contrast = c(1,-1))
```

If the argument `vvector` is set, the component "cbeta" of the object returned by `fmri.lm()` contains a vector with the parameters corresponding to the non- zero elements in `vvector` in each voxel. This may be used to include unknown latency of the hemodynamic response function in the analysis. First define the expected BOLD response for a given stimulus and its derivative and then combine them into the design matrix

```
hrf <- fmri.stimulus(107, c(18, 48, 78), 15, 2)
dhrf <- (c(0,diff(hrf)) + c(diff(hrf),0))/2
x <- fmri.design(cbind(hrf, dhrf))
spm <- fmri.lm(data, x, vvector = c(1,1)) .
```

The specification of `vvector` in the last statement results in a vector of length 2 containing the two parameter estimates for the expected BOLD response and its derivative in each voxel. Furthermore the ratio of the variance estimates for these parameters is calculated as a prerequisite for the smoothing procedure. See `fmri.smooth()` for details about smoothing this parametric map.

The function returns an object with class attributes "fmridata" and "fmrispm". This is again a list with components containing the estimated parameter contrast ('cbeta'), and its estimated variance ('var'), as well as estimated spatial correlations in all directions.

Structure Adaptive Smoothing (PS)

Smoothing of SPM's is applied in this context to improve the sensitivity of signal detection. This is expected to work since neural activations extends over regions of adjacent voxels. Averaging over such regions allows us to reduce the variance of the parameter estimates without compromising their mean. Introducing a spatial correlation structure also reduces the number of independent decisions made for signal detection and therefore eases the multiple test problem. Adaptive smoothing as implemented in this package also allows to improve the specificity of signal detection, see figure 2 for an illustration.

The parameter map is smoothed with

```
spmsmooth <- fmri.smooth(spm, hmax = hmax)
```

where `spm` is the result of the function `fmri.lm()`. `hmax` is the maximum bandwidth for the smoothing algorithm. For `lkern="Gaussian"` the bandwidth is given in units of FWHM, for any other localization kernel the unit is voxel. `hmax` should be chosen as the expected maximum size of the activation areas. As adaptive smoothing automatically adapts to different sizes and shapes of the activation areas, oversmoothing is not to be expected.

In (Tabelow et al., 2006) the use of a spatial adaptive smoothing procedure derived from the Propagation- Separation approach (Polzehl and Spokoiny, 2006) has been proposed in this context. The approach focuses, for each voxel i, on simultaneously identifying a region where the unknown parameter γ is approximately constant and to obtain an optimal estimate $\hat{\gamma}_i$ employing this structural information. This is achieved by an iterative procedure. Local smoothing is restricted to local vicinities of each voxel, that are characterized by a weighting scheme. Smoothing and characterization of local vicinities are alternated. Weights for a pair of voxels i and j are constructed as a product of kernel weights $K_{\mathrm{loc}}(\delta(i,j)/h)$, depending on the distance $\delta(i,j)$ between the two voxels and a bandwidth h, and a factor reflecting the difference of the estimates $\hat{\gamma}_i$ and $\hat{\gamma}_j$ obtained within the last iteration. The bandwidth h is increased with iterations up to a maximal bandwidth h_{max}.

The name Propagation- Separation is a synonym for the two main properties of this algorithm. In case of a completely homogeneous array $\tilde{\Gamma}$, that is $\mathbb{E}\,\tilde{\gamma} \equiv Const.$, the algorithm delivers essentially the same result as a nonadaptive kernel smoother employing the bandwidth h_{max}. In this case the procedure selects the best of a sequence of almost nonadaptive estimates, that is, it propagates to the one with maximum bandwidth. Separation means that as soon as within one iteration step significant differences of $\hat{\gamma}_i$ and $\hat{\gamma}_j$ are observed the corresponding weight is decreased to zero and the information from voxel j is no longer used to estimate γ_i. Voxels i and j belong to different regions of homogeneity and are therefore separated. As a consequence smoothing is restricted to regions with approximately constant values of γ, bias at the border of such regions is avoided and the spatial structure of activated regions

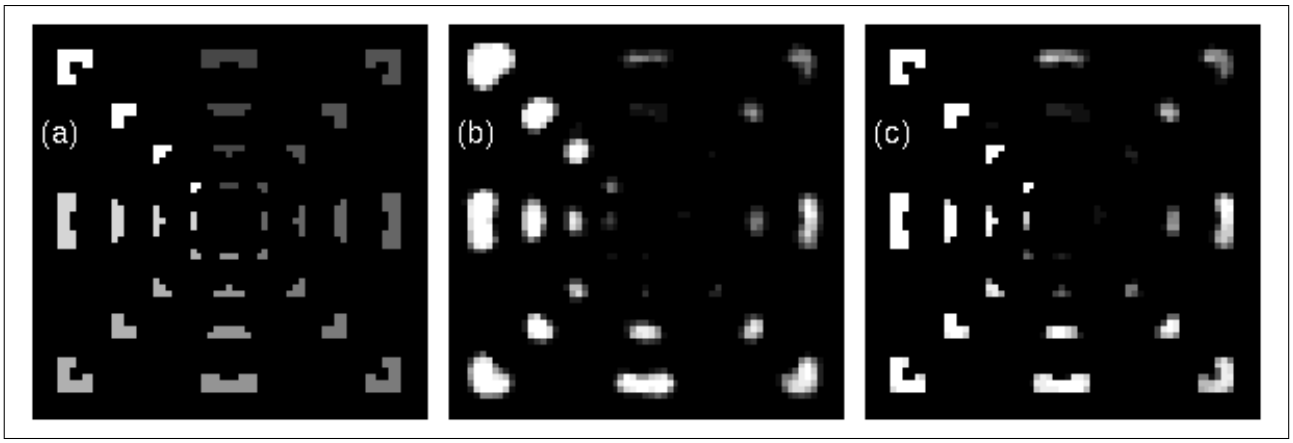

Figure 2: A numerical phantom for studying the performance of PS vs. Gaussian smoothing. (a) Signal locations within one slices. Eight different signal- to- noise ratios, increasing clockwise, are coded by gray values. The spatial extent of activations varies in the radial direction. The data cube contains 15 slices with activation alternated with 9 empty slices. (b) Smoothing with a conventional Gaussian filter. (c) Smoothing with PS. In both (b) and (c), the bandwidth is 3 voxels which corresponds to FWHM = 10 mm for typical voxel size. In (b) and (c) the proportion, over slices containing activations, of voxels that are detected in a given position is rendered by gray values. Black corresponds to the absence of a detection.

is preserved.

For a formal description of this algorithm, a discussion of its properties and theoretical results we refer to (Polzehl and Spokoiny, 2006) and (Tabelow et al., 2006). Numerical complexity, as well as smoothness within homogeneous regions is controlled by the maximum bandwidth $h_{\max}$.

If the argument object contains a parameter vector for each voxel (for example to include latency, see section 4) these will be smoothed according to their estimated variance ratio, see (Tabelow et al., 2006) for details on the smoothing procedure.

Figure 2 provides a comparison of signal detection employing a Gaussian filter and structural adaptive smoothing.

Signal detection

Smoothing leads to variance reduction and thus signal enhancement. It leaves us with three dimensional arrays $\hat{\Gamma}$, $\hat{S}$ containing the estimated effects $\hat{\gamma}_i = c^T\hat{\beta}_i$ and their estimated standard deviations $\hat{s}_i = (c^T \operatorname{var}\hat{\beta}_i c)^{1/2}$, obtained from time series of smoothed residuals. The voxelwise quotient $\hat{\theta}_i = \hat{\gamma}_i/\hat{s}_i$ of both arrays forms a statistical parametric map (SPM) $\hat{\Theta}$. The SPM as well as a map of p-values are generated by

```
pvalue <- fmri.pvalue(spmsmooth)
```

Under the hypothesis, that is, in absence of activation this SPM behaves approximately like a Gaussian Random Field, see (Tabelow et al., 2006). We therefore use the theory of Gaussian Random Fields to assign appropriate p-values as a prerequisite for signal detection. Such p-values can be defined (Worsley et al., 1996) as

$$p_i = \sum_{d=0}^{3} R_d(V(r_x, r_y, r_z))\rho_d(\hat{\theta}_i) \qquad (4)$$

where $R_d(V)$ is the *resel count* of the search volume V and $\rho_d(\hat{\theta}_i)$ is the *EC density* depending only on the parameter $\hat{\theta}_i$. r_x, r_y, r_z denotes the effective FWHM bandwidths that measure the smoothness (in resel space see (Worsley et al., 1996)) of the random field generated by a Gaussian filter that employs the bandwidth from the last iteration of the PS procedure (Tabelow et al., 2006). $R_d(V)$ and ρ_d are given in (Worsley et al., 1996). A signal will be detected in all voxels where the observed p-value is less or equal to a specified threshold.

Finally we provide a statistical analysis including unknown latency of the hemodynamic response function. If `spmsmooth` contains a vector (see `fmri.lm()` and `fmri.smooth()`), a χ^2 statistic is calculated from the first two parameters and used for p-value calculation. If `delta` is given, a cone statistics is used (Worsley and Taylor, 2005).

The parameter `mode` allows for different kinds of p-value calculation. "basic" corresponds to a global definition based on the amount of smoothness achieved by an equivalent Gaussian filter. The propagation condition ensures, that under the hypothesis $\hat{\Theta} = 0$ the adaptive smoothing perform like a non adaptive filter with the same kernel function. "local" corresponds to a more conservative setting, where the p-values are derived from the estimated local resel counts that has been achieved by the adaptive smoothing. "global" takes a global median of these resel counts for calculation.

Figure 3 provides the results of signal detection for an experimental fMRI data set.

Viewing results

Results can be displayed by a generic plot function

```
plot(object, anatomic,
     device="jpeg", file="result.jpeg")
```

`object` is an object of class "fmridata" (and "fmrispm" or "fmripvalue") as returned by `fmri.pvalue()`, `fmri.smooth()`, `fmri.lm()`, `read.ANALYZE()` or `read.AFNI()`. `anatomic` is an anatomic underlay of the same dimension as the functional data. The argument `type="3d"` provides an interactive display to produce 3 dimensional illustrations (requires R- package **tkrplot**) on the screen. The default for `type` is "slice", which can create output to the screen and image files.

We also provide generic functions `summary()` and `print()` for objects with class attribute "fmridata".

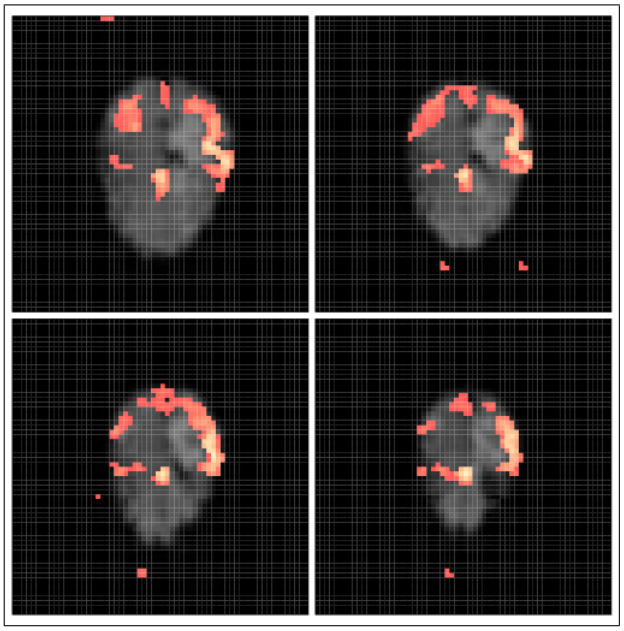

Figure 3: Signal detection in 4 consecutive slices using adaptive smoothing procedure. The shape and extent of the activation areas are conserved much better than with traditional Gaussian filtering. Data: courtesy of H. Voss, Weill Medical College of Cornell University.

Writing the results to files

Finally we provide functions to write data in standard medical image formats such as HEAD/BRIK, ANALYZE, and NIFTI.

```
write.AFNI("afnitest", data, c("signal"),
           note="random data", origin=c(0,0,0),
           delta=c(4,4,5), idcode="unique ID")
write.ANALYZE(data, list(pixdim=c(4,4,4,5)),
              file="analyzetest")
```

Some basic header information can be provided, in case of ANALYZE files as a list with several elements (see documentation for syntax). Any datacube created during the analysis can be written.

Bibliography

Biomedical Imaging Resource. *Analyze Program*. Mayo Foundation, 2001.

R. W. Cox. Afni: Software for analysis and visualization of functional magnetic resonance neuroimages. *Computers and Biomed. Res.*, 29:162- 173, 1996.

G. H. Glover. Deconvolution of impulse response in event- related BOLD fMRI. *NeuroImage*, 9:416- 429, 1999.

J. Polzehl and V. Spokoiny. Propagation- separation approach for local likelihood estimation. *Probab. Theory and Relat. Fields*, 135:335- 362, 2006.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007. ISBN 3- 900051- 07- 0.

K. Tabelow, J. Polzehl, H. U. Voss, and V. Spokoiny. Analyzing fMRI experiments with structural adaptive smoothing procedures. *NeuroImage*, 33:55- 62, 2006.

J. Polzehl and K. Tabelow. Analysing fMRI experiments with the fmri package in R. Version 1.0 - A users guide. *WIAS Technical Report No. 10*, 2006.

K. J. Worsley. Spatial smoothing of autocorrelations to control the degrees of freedom in fMRI analysis. *NeuroImage*, 26:635- 641, 2005.

K. J. Worsley, C. Liao, J. A. D. Aston, V. Petre, G. H. Duncan, F. Morales, and A. C. Evans. A general statistical analysis for fMRI data. *NeuroImage*, 15:1- 15, 2002.

K. J. Worsley, S. Marrett, P. Neelin, K. J. Friston, and A. C. Evans. A unified statistical approach for determing significant signals in images of cerebral activation. *Human Brain Mapping*, 4:58- 73, 1996.

K. J. Worsley and J. E. Taylor. Detecting fMRI activation allowing for unknown latency of the hemodynamic response. *Neuroimage*, 29:649- 654, 2006.

Jörg Polzehl & Karsten Tabelow
Weierstrass Institute for Applied Analysis and Stochastics, Berlin, Germany
polzehl@wias-berlin.de
tabelow@wias-berlin.de

Optmatch: Flexible, Optimal Matching for Observational Studies

Ben B. Hansen

Observational studies compare subjects who received a specified treatment to others who did not, without controlling assignment to treatment and comparison groups. When the groups differ at baseline in ways that are relevant to the outcome, the study has to adjust for the differences. An old and particularly direct method of making these adjustments is to match treated subjects to controls who are similar in terms of their pretreatment characteristics, then conduct an outcome analysis conditioning upon the matched sets. Adjustments of this type enjoy properties of robustness (Rubin, 1979) and transparency not shared with purely model-based adjustments, such as covariance adjustment without matching or stratification; and with the introduction of propensity scores to matching (Rosenbaum and Rubin, 1985), the approach was shown to be more broadly applicable than was previously thought. Arguably, the reach of techniques based on matching now exceeds that of purely model-based adjustment (Hansen, 2004).

To achieve these benefits, matched adjustment requires the analyst to articulate a distinction between desirable and undesirable potential matches, and then to match treated and control subjects in such a way as to favor the more desirable pairings. Propensity scoring fits under the first of these tasks, as do the construction of Mahalanobis matching metrics (Rosenbaum and Rubin, 1985), prognostic scoring (Hansen, 2006b), and the distance metric optimization of Diamond and Sekhon (2006). The second task, matching itself, is less statistical in nature, but doing it well can substantially improve the power and robustness of matched inference (Hansen and Klopfer, 2006; Hansen, 2004). The main purpose of **optmatch** is to relieve the analyst of responsibility for this important, if potentially tedious, undertaking, freeing attention for other aspects of the analysis. Given discrepancies between each treatment and control subject that might potentially be matched, **optmatch** places them into non-overlapping matched sets, in the process solving the discrete optimization problems needed to make sums of matched discrepancies as small as possible; after this, the analysis can proceed using permutation inference (Rosenbaum, 2002; Hothorn et al., 2006; Bowers and Hansen, 2006), conditional inference (Breslow and Day, 1980; Cox and Snell, 1989; Hansen, 2004; Lumley and Therneau, 2006), approximately conditional inference (Pierce and Peters, 1992; Brazzale, 2005; Brazzale et al., 2006), or multilevel models (Smith, 1997; Raudenbush and Bryk, 2002; Gelman and Hill, 2006).

Optimal matching of two groups

To illustrate the meaning of optimal matching, consider Cox and Snell's (1981, p.81) study of costs of nuclear power. Of 26 light water reactor plants constructed in the U.S. between 1967 and 1972, seven had been built on the site of existing plants. The problem is to estimate the cost benefit (or penalty) of building on an existing site as opposed to a new one. A matched analysis seeks to adjust for background characteristics determinative of cost, such as the date of construction and the capacity of the plant, by linking similar refurbished and new plants: plants of about the same capacity and constructed at about the same time, for example. To highlight the analogy with intervention studies, I refer to existing-site plants as "treatments" and new-site plants as "controls."

Consider the problem of arranging the plants in disjoint triples, each containing one treatment and two controls, placing each treatment and 14 of the 19 controls into some matched triple or another. A straightforward way to create such a match is to move down the list of treatments, pairing each to the two most similar controls that have not yet been matched; this is *nearest-available matching*. Figure 1 shows the 26 plants, their capacities and dates of construction, and a 1 : 2 matching constructed in this way. First A was matched to I and J, then B to L and N, and so forth. This example is discussed by Rosenbaum (2002, ch.10).

Existing site

	date	capacity
A	2.3	660
B	3.0	660
C	3.4	420
D	3.4	130
E	3.9	650
F	5.9	430
G	5.1	420

New site

	date	capacity
H	3.6	290
I	2.3	660
J	3.0	660
K	2.9	110
L	3.2	420
M	3.4	60
N	3.3	390
O	3.6	160
P	3.8	390
Q	3.4	130
R	3.9	650
S	3.9	450
T	3.4	380
U	4.5	440
V	4.2	690
W	3.8	510
X	4.7	390
Y	5.4	140
Z	6.1	730

"date" is date of construction, in years after 1965; "capacity" is net capacity of the power plant, in MWe above 400.

Figure 1: 1:2 matching by a nearest-available algorithm.

How might this process be improved? To complete step i, the nearest-available algorithm requires

a ranking of potential matches for treatment unit i, an ordering of available controls accounting for their differences with plant i in generating capacity and in year of construction. Typically controls j are ordered in terms of a numeric discrepancy, $d[i, j]$, from i; Figure 1's match follows Rosenbaum (2002, ch.10) in using the sum of rank differences on the two covariates (after restricting to a subset of the plants, `pt!=1`):

```
> data("nuclear", package="boot")
> attach(nuclear[nuclear$pt!=1,])
> drk <- rank(date)
> d <- outer(drk[pr==1], drk[pr!=1], "-")
> d <- abs(d)
> crk <- rank(cap)
> d <- d +
    abs(outer(crk[pr==1], crk[pr!=1], "-"))
```

(where `pr==1` indicates the treatment group). The d that results from these operations is shown (after rounding) in Figure 3. Having calculated this d, one can pose the task of matching as a discrete optimization problem: find the match $M = \{(i, j)\}$ minimizing $\sum_M d(i, j)$ among all sets of pairs (i, j) in which each treatment i appears twice and each control j appears at most once.

Optimal matching refers to algorithms guaranteed to find matches attaining this minimum, or falling within a specified tolerance of it, given a $n_t \times n_c$ discrepancy matrix M. Optimal matching's performance advantage over heuristic, non-optimal algorithms can be striking. For example, in the problem of Figure 1 optimal matching reduces nearest-available's sum of discrepancies by 23%. This optimal solution, found by **optmatch**'s `pairmatch` function, is shown in Figure 2.

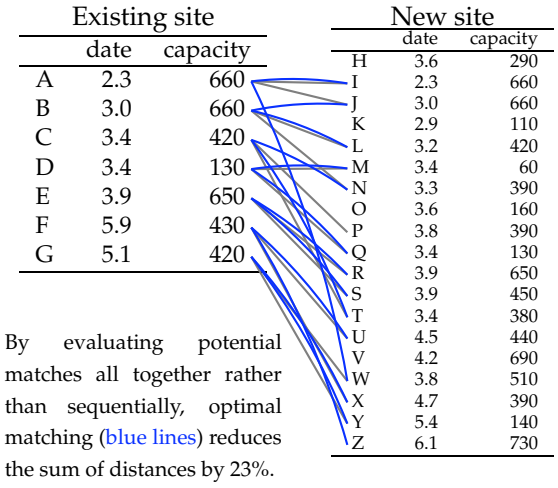

Figure 2: Optimal vs. greedy 1:2 matching.

An optimal match is optimal relative to given structural requirements, here that all treatments and a corresponding number of controls be arranged in 1 : 2 matched sets, and a given distance, here d. It is best-possible for the purposes of the analyst only insofar as the given distance and structural stipulations best represent the analyst's goals; in this sense "optimal" in "optimal matching" is analogous to the "maximum" of "maximum likelihood," which is never better than the chosen likelihood model it maximizes.

For example, in the problem just discussed the structural stipulation that the matched sets all be 1:2 triples may be poorly tailored to the goal of reducing baseline differences between the groups. It is appropriate if these differences stem entirely from a small minority of controls being too unlike any treatment subjects to bear comparison to them, since it does exclude 5/19 of potential controls from the final match; but differences on a larger number of controls, even small differences, require the techniques to be described under Generalizations of pair matching, below, which may also give similar or better bias reduction without excluding as many control observations. (See also Discussion, below.)

Growing your own discrepancy matrix

Figures 1 and 2 both illustrate *multivariate distance matching*, aligning units so as to minimize a sum of rank discrepancies. **Optmatch** is entirely flexible about the form of the distance on which matches are to be made. To propensity-score match nuclear plants, for example, one would prepare a propensity distance using

```
> pscr <- glm(pr ~ . -(pr+cost),
    family = binomial,
    data = nuclear)$linear.predictors
> PR <- nuclear$pr==1
> pdist <- outer(pscr[PR], pscr[PR], "-")

> pscr.v <- (var(pscr[PR])*(sum(PR)-1)+
    var(pscr[!PR])*(sum(!PR)-1))/
    (length(PR)-2)
> pdist <- abs(pdist)/sqrt(pscr.v)
```

or, more simply and reliably,

```
> pmodel <- glm(pr ~ . -(pr+cost),
    family = binomial, data = nuclear)
> pdist <- pscore.dist(pmodel)
```

Then `pdist` is passed to `pairmatch` or `fullmatch` as its first argument. Other discrepancies on which one might match include Mahalanobis distances (which can be produced using `mahal.dist`) and combinations of Mahalanobis and propensity-based distances (Rosenbaum and Rubin, 1985; Gu and Rosenbaum, 1993; Rubin and Thomas, 2000). Many special requirements, such as that matches be made only within given subclasses, or that specific matches be avoided, are also introduced through the discrepancy matrix.

First consider the case the matches are to be made within subclasses only. In the nuclear dataset, plants

with and without partial turnkey (pt) guarantees should be compared separately, since the meaning of the outcome variable, cost, changes with pt. Figure 2 shows only the pt!=1 plants, and its match is generated with the command pairmatch(d,controls=2), where d is the matrix in Figure 3. To match also partial turnkey plants to each other, one gathers into a list, dl, both d and a distance matrix dpt comparing new- and existing-site partial turnkey plants, then feeds dl to pairmatch as its first argument. For propensity or Mahalanobis matching, pscore.dist or mahal.dist would do this if given the formula pr~pt as their optional arguments 'structure.fmla'.

More generally, the helper function makedist (which is called by pscore.dist and mahal.dist) eases the application of a (user-defined) discrepancy matrix-producing function to each of a sequence of strata in order to generate a list of distance matrices. For separate summed-rank distances by pt subclass, one would write a function that extracts portions of relevant variables from a given data frame, looking to a treatment-group variable to decide what portions to take, as in

```
> capdatediffs <- function(trt, dat) {
    crk <- rank(dat[names(trt),"cap"])
    names(crk) <- names(trt)
    dmt <- outer(crk[trt], crk[!trt],"-")
    dmt <- abs(dmt)

    drk <- rank(dat[names(trt),"date"])
    dmt <- dmt +
    abs(outer(drk[trt], drk[!trt],"-"))
    dmt
    }
```

Then one would use makedist to apply the function separately within levels of pr:

```
> dl <- makedist(pr ~ pt, nuclear,
    capdatediffs)
```

The result of this is a list of two distance matrices, both submatrices of d created above, one comparing pt!=1 treatments and controls and a smaller one for pt==1 plants.

In larger problems, matching can be substantially faster if preceded by a division of the sample into subclasses; see Under the hood, below. The use of pscore.dist, mahal.dist, and makedist carry another advantage, that the lists of distances they generate carry metadata to prevent fullmatch or pairmatch from getting confused about the order of observations in the data frame from which the distances were generated.

Another common aim is to forbid unwanted matches. With **optmatch**, this is done by placing NA's, NaN's or Inf's at the relevant places in a distance matrix. Consider matching nuclear plants within *calipers* of three years on date of construction. Pairings of plants that would violate this requirement are indicated in red in Figure 3. To enforce the caliper, one could generate a matrix of discrepancies dy on year of construction, then replace the distance matrix of Figure 3, d, with d/(dy<=3); this new matrix has an Inf at each entry in Figure 3 currently shown in red, and otherwise is the same as in Figure 3.

Operations of these types, division and logical comparison, are compatible with subclassification prior to matching, despite the fact that the operations seem to require matrices while subclassification demands lists of matrices. Assuming one has defined a function datediffs as

```
> datediffs <- function(trt,data){
    sclr <- data[names(trt), 'date']
    names(sclr) <- names(trt)
    abs(outer(sclr[trt], sclr[!trt], '-'))
    }
```

then the command

```
> dly <- makedist(pr ~ pt, nuclear,
    datediffs)
```

tabulates absolute differences on date of construction, separately for pr==1 and pr!=1 strata. With **optmatch**, the expression dly<=3 returns a list of indicators of whether potential matches were built within three years of one another. Furthermore, dl/(dly<=3) is a list imposing the three-year caliper upon distances coded in *dl*. To pair match on propensity scores, but with a 3-year date-of-construction caliper, one would use pairmatch(dl/(dly<=3)).

Generalizations of pair matching

Matching with a varying number of controls

In Figures 1 and 2, both non-optimal and optimal matches insist on precisely two controls per treatment. If one's aim is to match as closely as possible, this is a limitation. To optimally match 14 of the 19 controls, as Figure 2 does, but without requiring that they always be matched two-to-one to treatments, one would use the command fullmatch, with options 'min.controls=1' and 'omit.fraction=5/19'. The flexibility this adds improves matching even more than the switch from greedy to optimal matching did; while optimal pair matching reduced greedy pair matching's net discrepancy from 82 to 63, optimal matching with a varying number of controls brings it to 44, just over half its original value.

If *mvnc* is the match created in this way, then the structure of *mvnc* is returned by

```
> stratumStructure(mvnc)
stratum treatment:control ratios
1:1 1:2 1:3 1:5
  4   1   1   1
```

Existing	New sites																		
	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	28	0	3	22	14	30	17	28	26	28	20	22	23	26	21	18	34	40	28
B	24	3	0	22	10	27	14	26	24	24	16	19	20	23	18	16	31	37	25
C	10	18	14	18	4	12	6	11	9	10	14	12	6	14	22	10	16	22	28
D	7	28	24	8	14	2	10	6	12	0	24	22	4	24	32	20	18	16	38
E	17	20	16	32	18	26	20	18	12	24	0	2	20	6	8	4	14	20	14
F	20	31	28	35	20	29	22	20	14	26	12	9	22	5	15	12	9	11	12
G	14	32	29	30	18	24	17	16	10	22	12	10	17	6	16	14	4	8	17

Figure 3: Rank discrepancies of new- and existing-site nuclear plants without partial turnkey guarantees. New- and existing-site plants which differ by more than 3 years in date of construction are indicated in red.

This means it consists of four matched pairs and a matched triple, quadruple, and sextuple, all containing precisely one treatment. Ming and Rosenbaum (2000) discuss matching with a varying number of controls, implementing it in their example with a somewhat different algorithm.

Full matching

A propensity score is the conditional probability, $p(\mathbf{x})$, of falling in the treatment group given covariates $\mathbf{x}$ (or an increasing transformation of it, such as its logit). Because subjects with large propensity scores more frequently fall in the treatment group, and subjects with low propensity scores are more frequently controls, propensity score matching is fundamentally at odds with matching treatments and controls in fixed ratios, such as $1:1$ or $1:k$. These ratios must be allowed to adapt, so that 1:1 matches can be made where $p(\mathbf{x})/(1-p(\mathbf{x})) \approx 1$ while $1:k$ matches are made where $p(\mathbf{x})/(1-p(\mathbf{x})) \approx 1/k$; otherwise either some subjects will have to go unmatched or some subjects are bound to be poorly matched on their propensity scores. Matching with multiple controls addresses part of this problem, but only *full matching* (Rosenbaum, 1991) addresses it in its entirety, by also permitting l:1 matches, for when $p(\mathbf{x})/(1-p(\mathbf{x})) \approx l \geq 2$.

In general, full matching is useful when there are some regions of covariate space in which controls outnumber treatments but others in which treatments outnumber controls. This pattern emerges most clearly when matching on a propensity score, but they influence the quality of matches even without propensity scores. The rank discrepancies of new- and existing-site `pt==1` plants, shown in Figure 4, show it; earlier dates of construction, and smaller capacities, are more common among controls (d and e) than treatments (b only), and this is reflected in Figure 4's sums of discrepancies on rank. As a consequence, full matching achieves a net rank discrepancy (3) that is half of the minimum possible (6) with techniques that don't permit both 1:2 and 2:1 matched sets.

Existing	New sites		
	d	e	f
a	6	6	[0]
b	[0]	[3]	6
c	6	6	[0]

Figure 4: Rank discrepancies of new- and existing-site nuclear plants with partial turnkey guarantees. Boxes indicate the optimal full matching of these plants.

Gu and Rosenbaum (1993) compare full and other forms of matching in an extensive simulation study, while Hansen (2004) and Hansen and Klopfer (2006) present applications. This literature emphasizes the importance of using *structural restrictions*, upper limits on K in $K:1$ matched sets and on L in $1:L$ matched sets, when full matching, in order to control the variance of matched estimation. With `fullmatch`, an upper limit K:1 on treatment:control ratios is conveyed using '`min.controls=1/K`', while a lower limit of $1:L$ on the treatment : control ratio would be given with '`max.controls=L`'. Hansen (2004) and Hansen and Klopfer (2006) give strategies to optimize these tuning parameters. In the context of a specific application, Hansen (2004) finds $(\text{min.controls}, \text{max.controls}) = (1/2, 2) \cdot (1-\hat{p})/\hat{p}$ to work best, where $\hat{p}$ represent the proportion treated in the stratum being matched. In an unrelated application, Stuart and Green (2006) find these values to work well; they may be a good starting point for general use.

A somewhat related technique is matching "with replacement," in which overlap between matched sets is permitted in the interests of achieving closer matches. Because of the overlap, methods appropriate to stratified data are not generally appropriate for samples matched with replacement. The technique forces one to resort to specialized techniques, such as those of Abadie and Imbens (2006). On the other hand, with-replacement matching would appear to offer the possibility of closer matches, since its pairing of one treatment unit in no way limits its pairing of the next treatment unit.

However, it is a surprising, and evidently

little-known, fact that with-replacement matching achieves no closer matches than full matching, a without-replacement matching technique. As discussed by Rosenbaum (1991) and (more explicitly) by Hansen and Klopfer (2006), given any criterion for a potential pairing of subjects to be acceptable, full matching matches *all* subjects with at least one suitable match in their comparison group, matching them *only* to acceptable counterparts. So one might insist, in particular, that each treatment unit be matched only to one of its nearest neighbors; by sharing controls among treated units where necessary, omitting controls who are not the nearest neighbor of some treatment, and matching to multiple controls where that can be done, full matching finds a way to honor this requirement. Since the matched sets produced by full matching never overlap, it has the advantage over with-replacement matching of combining with any method of estimation appropriate to finely stratified data.

Under the hood

Hansen and Klopfer (2006) describe the network-flows algorithm on which **optmatch** relies in some detail, establishing its optimality for full matching and matching with a fixed or varying number of controls. They also give upper bounds for the time complexity of the algorithm: roughly, $O(n^3 \log(nC))$, where n is the size of the sample and C is the quotient of largest discrepancy in the distance matrix and the matching tolerance. This is comparable to the time complexity of squaring a $n \times n$ matrix. More precisely, the algorithm requires $O(nn_t n_c \log(nC))$ floating-point operations, where n_t and n_c are the sizes of the treatment and control groups.

These bounds have two practical consequences for **optmatch**. First, computational costs grow steeply with the size of the discrepancy matrix. Just as squaring two $(n/2) \times (n/2)$ submatrices of an $n \times n$ matrix is about four times faster than squaring the full $n \times n$ matrix, matching is made much faster by subdividing large matching problems into smaller ones. For this reason `makedist` is written so as to facilitate subclassification prior to matching, the effect of which is to split larger matching problems into a sequence of smaller ones. Second, the C-factor contributes secondarily to computational cost; its contribution is reduced by increasing the value of the '`tol`' argument to `fullmatch` or `pairmatch`.

Discussion

When and how matching reduces systematic differences between groups

Matching can address bias in observational studies in either of two ways. In *matched sampling*, it is used to select a subset of control subjects most like treatments, with the remainder of subjects excluded from analysis; in *matched adjustment*, it is used to force treatment control comparisons to be based on individualized comparisons made within matched sets, which will have been so engineered that matched counterparts are more like one another than are treatments and controls on the whole. Matched sampling is typically followed by matched adjustment, but matched adjustment can be useful even when not preceded by matched sampling.

Because it excludes some five control subjects, the match depicted in Figures 1 and 2 might be used in a context of matched sampling, although it differs in important respects from typical matched samples. In archetypal cases, matched sampling is used when for cost reasons the number of controls to be followed up for outcome data has to be reduced anyway, not when outcome data is already available for the entire sample already; and in archetypal cases, the reservoir of potential controls is many times larger than the size of the desired control group. See, *e.g.*, Althauser and Rubin (1970) or Rosenbaum and Rubin (1985). The matches in Figures 1 and 2 are typical of matched sampling in matching a fixed number of controls to each treatment subject. When bias can be addressed by being very selective in the choice of controls, flexibility in the structure of matched sets becomes less important.

When there is no additional data to be collected, there may be little use for matched sampling per se, while matched adjustment may still be attractive. In these cases, it is important to recognize that matching, even optimal matching, does not in itself reduce systematic differences between treatment and control groups unless it is specifically given the flexibility to do so. Suppose, for instance, that adjustment for the variable `t2` is needed in the comparison of new- and existing site-plants. This variable, which represents the time between the issue of an operating permit and a construction permit, differs markedly in its distribution among "treatments" and "controls," as seen in Figure 5: treatments have systematically larger values of it, although the two distributions well overlap. When two groups compare in this way, no fixed-ratio matching of them can reduce their overall discrepancy. Some of the observations will have to be set aside — or, better yet, one could match the two groups in varying ratios, using matching with multiple controls or full matching. These techniques have surprising power to reconcile differences between treatments and controls while setting aside few or even no subjects because they lack suitable counterparts; the reader is referred to Hansen (2004, § 2) for general discussion and a case study.

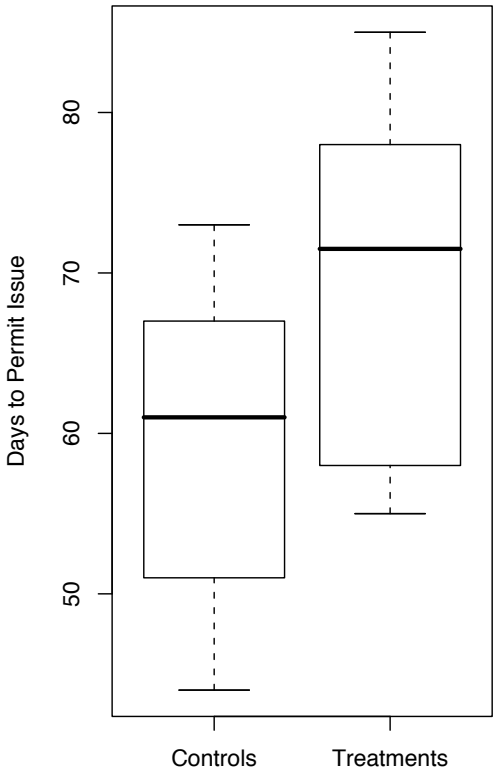

Figure 5: New and existing sites' differences on the variable `t2`. To reduce these differences, one has either to drop observations or to use flexible matching techniques.

In practice, one should look critically at an optimal match before moving ahead with it toward outcome analysis, refining its generating distance and structural requirements as appropriate, just as a careful analyst deploys various diagnostics in the process of developing and refining a likelihood-based analysis. Diagnostics for matching are discussed in various methodological papers, many of them recent (Rosenbaum and Rubin, 1985; Rubin, 2001; Lee, 2006; Hansen, 2006a; Sekhon, 2007).

optmatch output

Matched pairs are often analyzed by methods particular to that structure, for example the paired t-test. However, matching with multiple controls and full matching require methods that treat the matched sets as strata. With these uses in mind, matching functions in **optmatch** give factor objects as their output, creating unique identifiers for each matched set and tagging them as such in the factor. (Strictly speaking, the value of a call to `fullmatch` or `pairmatch` is of the class `c("optmatch", "factor")`, but it is safe to treat it as a factor.) If one in fact has produced a pair match, then one can recover the paired differences using the `split` command:

```
> pm <- pairmatch(dl)
> attach(nuclear)
> unlist(split(cost[PR],pm[PR])) -
    unlist(split(cost[!PR],pm[!PR]))
```

— the result of which is the vector of differences

```
  0.1    0.2    0.3  ...   1.2   1.3
-9.77 -10.09 184.75 ... -17.77 -4.52
```

For matched comparisons after full matching or matching with a varying number of controls, one uses such commands as

```
> fm <- fullmatch(dl)
> tapply(cost[PR], fm[PR], mean) -
    tapply(cost[!PR], fm[!PR], mean)
```

to return differences of treatment and control means by matched set. The sizes of the matched sets, in terms of treatment units, controls, or both, can be tabulated by

```
> tapply(PR, fm, sum)
> tapply(!PR, fm, sum)
> tapply(fm,fm,length)
```

respectively. Unmatched units are automatically dropped, and `split` and `tapply` return matched-set specific results in a common ordering (that of the levels of the match object, *e.g.* `pm` or `fm`.)

Summary

Optmatch offers a comprehensive implementation of matching of two groups, such as treatments and controls or cases and controls, including optimal pair matching, optimal matching with k controls, optimal matching with a varying number of controls, and full matching, with and without structural restrictions.

Bibliography

A. Abadie and G. W. Imbens. Large sample properties of matching estimators for average treatment effects. *Econometrica*, 74(1):235–267, 2006.

R. Althauser and D. Rubin. The computerized construction of a matched sample. *American Journal of Sociology*, 76(2):325–346, 1970.

J. Bowers and B. B. Hansen. Attributing effects to a cluster randomized get-out-the-vote campaign. Technical Report 448, Statistics Department, University of Michigan, October 2006. URL `http://www-personal.umich.edu/%7Ejwbowers/PAPERS/bowershansen2006-10TechReport.pdf`.

A. R. Brazzale. hoa: An R package bundle for higher order likelihood inference. *Rnews*, 5/1 May 2005: 20–27, 2005. URL `ftp://cran.r-project.org/doc/Rnews/Rnews_2005-1.pdf`. ISSN 609-3631.

A. R. Brazzale, A. C. Davison, and N. Reid. *Applied Asymptotics*. Cambridge University Press, 2006.

N. E. Breslow and N. E. Day. *Statistical Methods in Cancer Research (Vol. 1) — The Analysis of Case-control Studies*. Number 32 in IARC Scientific Publications. International Agency for Research on Cancer, 1980.

D. Cox and E. Snell. *Applied Statistics*. Chapman and Hall, 1981.

D. R. Cox and E. J. Snell. *Analysis of Binary Data*. Chapman & Hall Ltd, 1989.

A. Diamond and J. S. Sekhon. Genetic matching for estimating causal effects: A general multivariate matching method for achieving balance in observational studies. Technical report, Travers Department of Political Science, University of California, Berkeley, 2006. version 1.2.

A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, 2006.

X. Gu and P. R. Rosenbaum. Comparison of multivariate matching methods: Structures, distances, and algorithms. *Journal of Computational and Graphical Statistics*, 2(4):405–420, 1993.

B. B. Hansen. Appraising covariate balance after assignment to treatment by groups. Technical Report 436, University of Michigan, Statistics Department, 2006a.

B. B. Hansen. Full matching in an observational study of coaching for the SAT. *Journal of the American Statistical Association*, 99(467):609–618, September 2004.

B. B. Hansen. Bias reduction in observational studies via prognosis scores. Technical Report 441, University of Michigan, Statistics Department, April 2006b. URL `http://www.stat.lsa.umich.edu/%7Ebbh/rspaper2006-06.pdf`.

B. B. Hansen and S. O. Klopfer. Optimal full matching and related designs via network flows. *Journal of Computational and Graphical Statistics*, 15(3):609–627, 2006. URL `http://www.stat.lsa.umich.edu/%7Ebbh/hansenKlopfer2006.pdf`.

T. Hothorn, K. Hornik, M. van de Wiel, and A. Zeileis. A lego system for conditional inference. *The American Statistician*, 60(3):257–263, 2006.

W.-S. Lee. Propensity score matching and variations on the balancing test, 2006. URL `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=936782#`.

T. Lumley and T. Therneau. **survival***: Survival analysis, including penalised likelihood*, 2006. R package version 2.26.

K. Ming and P. R. Rosenbaum. Substantial gains in bias reduction from matching with a variable number of controls. *Biometrics*, 56:118–124, 2000.

D. Pierce and D. Peters. Practical use of higher order asymptotics for multiparameter exponential families. *Journal of the Royal Statistical Society. Series B (Methodological)*, 54(3):701–737, 1992.

S. W. Raudenbush and A. S. Bryk. *Hierarchical Linear Models: Applications and Data Analysis Methods*. Sage Publications Inc, 2002.

P. R. Rosenbaum. A characterization of optimal designs for observational studies. *Journal of the Royal Statistical Society*, 53:597– 610, 1991.

P. R. Rosenbaum. *Observational Studies*. Springer-Verlag, second edition, 2002.

P. R. Rosenbaum and D. B. Rubin. Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *The American Statistician*, 39:33–38, 1985.

D. B. Rubin. Using propensity scores to help design observational studies: application to the tobacco litigation. *Health Services and Outcomes Research Methodology*, 2(3):169–188, 2001.

D. B. Rubin. Using multivariate matched sampling and regression adjustment to control bias in observational studies. *Journal of the American Statistical Association*, 74:318–328, 1979.

D. B. Rubin and N. Thomas. Combining propensity score matching with additional adjustments for prognostic covariates. *Journal of the American Statistical Association*, 95(450):573–585, 2000.

J. S. Sekhon. Alternative balance metrics for bias reduction in matching methods for causal inference. Survey Research Center, University of California, Berkeley, 2007. URL `http://sekhon.berkeley.edu/papers/SekhonBalanceMetrics.pdf`.

H. Smith. Matching with multiple controls to estimate treatment effects in observational studies. *Sociological Methodology*, 27:325–353, 1997.

E. A. Stuart and K. M. Green. Using full matching to estimate causal effects in non-experimental studies: Examining the relationship between adolescent marijuana use and adult outcomes. Technical report, Johns Hopkins University, 2006.

Ben B. Hansen
Department of Statistics
University of Michigan
`bbh@umich.edu`

Random Survival Forests for R

Hemant Ishwaran and Udaya B. Kogalur

Introduction

In this article we introduce *Random Survival Forests*, an ensemble tree method for the analysis of right censored survival data. As is well known, constructing ensembles from base learners, such as trees, can significantly improve learning performance. Recently, Breiman showed that ensemble learning can be further improved by injecting randomization into the base learning process, a method called Random Forests (Breiman, 2001). Random Survival Forests is closely modeled after Breiman's approach. In Random Forests, randomization is introduced in two forms. First, a randomly drawn bootstrap sample of the data is used for growing the tree. Second, the tree learner is grown by splitting nodes on randomly selected predictors. While at first glance Random Forest might seem an unusual procedure, considerable empirical evidence has shown it to be highly effective. Extensive experimentation, for example, has shown it compares favorably to state of the art ensembles methods such as bagging (Breiman, 1996) and boosting (Schapire et al., 1998).

Random Survival Forests being closely patterned after Random Forests naturally inherits many of its good properties. Two features especially worth emphasizing are: (1) It is user-friendly in that only three, fairly robust, parameters need to be set (the number of randomly selected predictors, the number of trees grown in the forest, and the splitting rule to be used). (2) It is highly data adaptive and virtually model assumption free. This last property is especially helpful in survival analysis. Standard analyses often rely on restrictive assumptions such as proportional hazards. Also, with such methods there is always the concern whether associations between predictors and hazards have been modeled appropriately, and whether or not non-linear effects or higher order interactions for predictors should be included. In contrast, such problems are handled seamlessly and automatically within a Random Forests approach.

While R currently has a Random Forests package for classification and regression problems (the `randomForest()` package ported by Andy Liaw and Matthew Wiener), there is currently no version available for analyzing survival data[1]. The need for a Random Forests procedure separate from one that handles classification and regression problems is well motivated as survival data possesses unique features not handled within a CART (Classification and Regression Tree) paradigm. In particular, the notion of what constitutes a good node split for growing a tree, what prediction means, and how to measure prediction performance, pose unique problems in survival analysis.

Moreover, while a survival tree can in some instances be reformulated as a classification tree, thereby making it possible to use CART software for a Random Forests analysis, we believe such approaches are merely stop-gap measures that will be difficult for the average user to implement. For example, Ishwaran et al. (2004) show under a proportional hazards assumption that one can grow survival trees using the splitting rule of LeBlanc and Crowley (1992) using the `rpart()` algorithm (Therneau and Atkinson, 1997), hence making it possible to implement a relative risk forests analysis in R. However, this requires extensive coding on the users part, is limited to proportional hazard settings, and the splitting rule used is only approximate.

The algorithm

It is clear that a comprehensive method with accompanying software is needed. To fill this need we introduce `randomSurvivalForest`, an R software package for implementing Random Survival Forests. The algorithm used by `randomSurvivalForest` is broadly described as follows:

1. Draw `ntree` bootstrap samples from the original data.
2. Grow a tree for each bootstrapped data set. At each node of the tree randomly select `mtry` predictors (covariates) for splitting on. Split on a predictor using a survival splitting criterion. A node is split on that predictor which maximizes survival differences across daughter nodes.
3. Grow the tree to full size under the constraint that a terminal node should have no less than `nodesize` unique deaths.
4. Calculate an ensemble cumulative hazard estimate by combining information from the `ntree` trees. One estimate for each individual in the data is calculated.
5. Compute an out-of-bag (OOB) error rate for the ensemble derived using the first b trees, where $b = 1, \ldots,$ `ntree`.

Splitting rules

Node splits are a crucial ingredient to the algorithm. The `randomSurvivalForest` package pro-

[1] We are careful to distinguish Random Forests procedures following Breiman's methodology from other approaches. Readers, for example, should be aware of the R party() package, which implements a random forests style analysis using conditional tree base learners (Hothorn et al., 2006).

vides four different survival splitting rules for the user. These are: (i) a log-rank splitting rule, the default splitting rule, invoked by the option `splitrule="logrank"`; (ii) a conservation of events splitting rule, `splitrule="conserve"`; (iii) a logrank score rule, `splitrule="logrankscore"`; (iv) and a fast approximation to the logrank splitting rule, `splitrule="logrankapprox"`.

Notation

Assume we are at node h of a tree during its growth and that we seek to split h into two daughter nodes. We introduce some notation to help discuss how the the various splitting rules work to determine the best split. Assume that within h are n individuals. Denote their survival times and 0-1 censoring information by $(T_1, \delta_1), \ldots, (T_n, \delta_n)$. An individual l will be said to be right censored at time T_l if $\delta_l = 0$, otherwise the individual is said to have died at T_l if $\delta_l = 1$. In the case of death, T_l will be refered to as an event time, and the death as an event. An individual l who is right censored at T_l simply means the individual is known to have been alive at T_l, but the exact time of death is unknown.

A proposed split at node h on a given predictor x is always of the form $x \leq c$ and $x > c$. Such a split forms two daughter nodes (a left and right daughter) and two new sets of survival data. A good split maximizes survival differences across the two sets of data. Let $t_1 < t_2 < \cdots < t_N$ be the distinct death times in the parent node h, and let $d_{i,j}$ and $Y_{i,j}$ equal the number of deaths and individuals at risk at time t_i in the daughter nodes $j = 1, 2$. Note that $Y_{i,j}$ is the number of individuals in daughter j who are alive at time t_i, or who have an event (death) at time t_i. More precisely,

$$Y_{i,1} = \#\{T_l \geq t_i,\ x_l \leq c\}, \quad Y_{i,2} = \#\{T_l \geq t_i,\ x_l > c\},$$

where x_l is the value of x for individual $l = 1, \ldots, n$. Finally, define $Y_i = Y_{i,1} + Y_{i,2}$ and $d_i = d_{i,1} + d_{i,2}$. Let n_j be the total number of observations in daughter j. Thus, $n = n_1 + n_2$. Note that $n_1 = \#\{l : x_l \leq c\}$ and $n_2 = \#\{l : x_l > c\}$.

Log-rank splitting

The log-rank test for a split at the value c for predictor x is

$$L(x,c) = \frac{\sum_{i=1}^{N} \left(d_{i,1} - Y_{i,1} \frac{d_i}{Y_i} \right)}{\sqrt{\sum_{i=1}^{N} \frac{Y_{i,1}}{Y_i} \left(1 - \frac{Y_{i,1}}{Y_i} \right) \left(\frac{Y_i - d_i}{Y_i - 1} \right) d_i}}.$$

The value $|L(x,c)|$ is the measure of node separation. The larger the value for $|L(x,c)|$, the greater the difference between the two groups, and the better the split is. In particular, the best split at node h is determined by finding the predictor x^* and split value c^* such that $|L(x^*, c^*)| \geq |L(x,c)|$ for all x and c.

Conservation of events splitting

The log-rank test for splitting survival trees is a well established concept (Segal, 1988), having been shown to be robust in both proportional and non-proportional hazard settings (LeBlanc and Crowley, 1993). However, one criticism often heard is that it tends to favor continuous predictors and often suffers from an end-cut preference (favoring uneven splits). However, in our experience with Random Survival Forests we have not found this to be a serious deficiency. Nevertheless, to address this potential problem we introduce another important class of test statistics for splitting that are related to conservation of events; a concept introduced in Naftel et al. (1985) (our simulations have indicated these tests may be much less susceptible to the aforementioned problems).

Under fairly general conditions, conservation of events asserts that the sum of the estimated cumulative hazard function over the observed time points (deaths and censored values) must equal the total number of deaths. This applies to a wide collection of estimates including the the Nelson-Aalen estimator. The Nelson-Aalen cumulative hazard estimator for daughter j is

$$\hat{H}_j(t) = \sum_{t_{i,j} \leq t} \frac{d_{i,j}}{Y_{i,j}}$$

where $t_{i,j}$ are the ordered death times for daughter j (note: we define $0/0 = 0$).

Let $(T_{l,j}, \delta_{l,j})$, for $l = 1, \ldots, n_j$, denote all survival times and censoring indicator pairs for daughter j. Conservation of events asserts that

$$\sum_{l=1}^{n_j} \hat{H}_j(T_{l,j}) = \sum_{l=1}^{n_j} \delta_{l,j}. \tag{1}$$

In other words, the total number of deaths is conserved in each daughter.

The conservation of events splitting rule is motivated by (1). First, order the time points within each daughter node such that

$$T_{(1),j} \leq T_{(2),j} \leq \cdots \leq T_{(n_j),j}.$$

Let $\delta_{(l),j}$ be the censoring indicator function for the ordered value $T_{(l),j}$. Define

$$\mathscr{M}_{k,j} = \sum_{l=1}^{k} \hat{H}_j(T_{(l),j}) - \sum_{l=1}^{k} \delta_{(l),j}, \quad k = 1, \ldots, n_j.$$

One can think of $\mathscr{M}_{k,j}$ as "residuals" that measure accuracy of conservation of events. The proposed test statistic takes the sum of the absolute values of $\mathscr{M}_{k,j}$ for $k = 1, \ldots, n_j$ for each daughter j, and weights these values by the number of individuals at risk

within each group. Observe that $\mathscr{M}_{n_j,j} = 0$, but nothing can be said about $\mathscr{M}_{k,j}$ for $k < n_j$. Thus, by considering $\mathscr{M}_{k,j}$ for each k, the proposed test measures how evenly distributed conservation of events is over all deaths. The measure of conservation of events for the split on x at the value c is

$$\text{Conserve}(x,c) = \frac{1}{Y_{1,1}+Y_{1,2}} \sum_{j=1}^{2} Y_{1,j} \sum_{k=1}^{n_j-1} |\mathscr{M}_{k,j}|.$$

This value is small if the two groups are well separated. Because we want to maximize survival differences due to a split, we use the transformed value $1/(1+\text{Conserve}(x,c))$ as our measure of node separation.

The preceding expression for $\text{Conserve}(x,c)$ can be quite expensive to compute as it involves summing over all survival times within the daughter nodes. However, we can greatly reduce the amount of work by compressing the sums to involve only event times. With some work, one can show that $\text{Conserve}(x,c)$ is equivalent to:

$$\frac{1}{Y_{1,1}+Y_{1,2}} \sum_{j=1}^{2} Y_{1,j} \sum_{k=1}^{N-1} \left\{ N_{k,j} Y_{k+1,j} \sum_{l=1}^{k} \frac{d_{l,j}}{Y_{l,j}} \right\},$$

where $N_{i,j} = Y_{i,j} - Y_{i+1,j}$ equals the number of observations in daughter j with observed time falling within the interval $[t_i, t_{i+1})$ for $i = 1, \ldots, N$ where $t_{N+1} = \infty$.

Log-rank score splitting

Another useful splitting rule available within the `randomSurvivalForest` package is the log-rank score test of Hothorn and Lausen (2003). To describe this rule, assume the predictor x has been ordered so that $x_1 \le x_2 \le \cdots \le x_n$. Now, compute the "ranks" for each survival time T_l,

$$a_l = \delta_l - \sum_{k=1}^{\Gamma_l} \frac{\delta_k}{n - \Gamma_k + 1}$$

where $\Gamma_k = \#\{t : T_t \le T_k\}$. The log-rank score test is defined as

$$S(x,c) = \frac{\sum_{x_l \le c} a_l - n_1 \bar{a}}{\sqrt{n_1 \left(1 - \frac{n_1}{n}\right) s_a^2}}$$

where $\bar{a}$ and s_a^2 are the sample mean and sample variance of $\{a_l : l = 1, \ldots, n\}$. Log-rank score splitting defines the measure of node separation by $|S(x,c)|$. Maximizing this value over x and c yields the best split.

Approximate logrank splitting

An approximate log-rank test can be used in place of $L(x,c)$ to greatly reduce computations. To derive the approximation, first rewrite the numerator of $L(x,c)$ in a form that uses the Nelson-Aalen estimator for the parent node. The Nelson-Aalen estimator is

$$\hat{H}(t) = \sum_{t_i \le t} \frac{d_i}{Y_i}.$$

As shown in LeBlanc and Crowley (1993) one can write

$$\sum_{i=1}^{N} \left(d_{i,1} - Y_{i,1} \frac{d_i}{Y_i} \right) = D_1 - \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l),$$

where $D_j = \sum_{i=1}^{N} d_{i,j}$ for $j = 1, 2$. Because the Nelson-Aalen estimator is computed on the parent node, and not daughter nodes, this yields an efficient way to compute the numerator of $L(x,c)$.

Now to simplify the denominator, we approximate the variance of the numerator of $L(x,c)$ as in Section 7.7 of Cox and Oakes (1988) (this approximation was suggested to us by Michael LeBlanc in personal communication). Setting $D = \sum_{i=1}^{N} d_i$, we get the following approximation to the log-rank test $L(x,c)$:

$$\frac{D^{1/2} \left(D_1 - \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l) \right)}{\sqrt{\left\{ \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l) \right\} \left\{ D - \sum_{l=1}^{n} \text{I}\{x_l \le c\} \hat{H}(T_l) \right\}}}.$$

Ensemble estimation

The `randomSurvivalForest` package produces an ensemble estimate for the cumulative hazard function. This is our predictor and key deliverable. Error rate performance is calculated based on this value. The ensemble is derived as follows. First, for each tree grown from a bootstrap data set we estimate the cumulative hazard function for the tree. This is accomplished by grouping hazard estimates by terminal nodes. Consider a specific node h. Let $\{t_{l,h}\}$ be the distinct death times in h and let $d_{l,h}$ and $Y_{l,h}$ equal the number of deaths and individuals at risk at time $t_{l,h}$. The cumulative hazard estimate for node h is defined as

$$\hat{H}_h(t) = \sum_{t_{l,h} \le t} \frac{d_{l,h}}{Y_{l,h}}.$$

Each tree provides a sequence of such estimates, $\hat{H}_h(t)$. If there are M terminal nodes in the tree, then there are M such estimates. To compute $\hat{H}(t|\mathbf{x}_i)$ for an individual i with predictor $\mathbf{x}_i$, simply drop $\mathbf{x}_i$ down the tree. The terminal node for i yields the desired estimator. More precisely,

$$\hat{H}(t|\mathbf{x}_i) = \hat{H}_h(t), \text{ if } \mathbf{x}_i \in h. \tag{2}$$

Note this value is computed *for all individuals* i in the data.

The estimate (2) is based on one tree. To produce our ensemble we average (2) over all `ntree` trees. Let $\hat{H}_b(t|\mathbf{x})$ denote the cumulative hazard estimate (2) for tree $b = 1,\ldots,$`ntree`. Define $I_{i,b} = 1$ if i is an OOB point for b, otherwise set $I_{i,b} = 0$. The OOB ensemble cumulative hazard estimator for i is

$$\hat{H}^*_e(t|\mathbf{x}_i) = \frac{\sum_{b=1}^{\texttt{ntree}} I_{i,b}\hat{H}_b(t|\mathbf{x}_i)}{\sum_{b=1}^{\texttt{ntree}} I_{i,b}}.$$

Observe that the estimator is obtained by averaging over only those bootstrap samples in which i is excluded (i.e., those datasets in which i is an OOB value). The OOB estimator is in contrast to the ensemble cumulative hazard estimator that uses all samples:

$$\hat{H}_e(t|\mathbf{x}_i) = \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} \hat{H}_b(t|\mathbf{x}_i).$$

Concordance error rate

Given the OOB estimator $\hat{H}^*_e(t|\mathbf{x})$, it is a simple matter to compute the error rate. We measure error using Harrell's concordance index (Harrell et al., 1982). Unlike other measures of survival performance, Harrell's C-index does not depend on choosing a fixed time for evaluation of the model and specifically takes into account censoring of individuals (May et al., 2004). The method has quickly become quite popular in the literature as a means for assessing prediction performance in survival analysis settings. See Kattan et al. (1998) and references therein.

To compute the concordance index we must define what constitutes a worse predicted outcome. We take the following approach. Let $t^*_1,\ldots,t^*_N$ denote all unique event times in the data. Individual i is said to have a worse outcome than j if

$$\sum_{k=1}^{N} \hat{H}^*_e(t^*_k|\mathbf{x}_i) > \sum_{k=1}^{N} \hat{H}^*_e(t^*_k|\mathbf{x}_j).$$

The concordance error rate is computed as follows:

1. Form all possible pairs of observations over all the data.
2. Omit those pairs where the shorter event time is censored. Also, omit pairs i and j if $T_i = T_j$ unless $\delta_i = 1$ and $\delta_j = 0$ or $\delta_i = 0$ and $\delta_j = 1$. The last restriction only allows ties if one of the observations is a death and the other a censored observation. Let Permissible denote the total number of permissible pairs.
3. Count 1 for each permissible pair in which the shorter event time had the worse predicted outcome. Count 0.5 if the predicted outcomes are tied. Let Concordance denote the total sum over all permissible pairs.
4. Define the concordance index C as
$$C = \frac{\text{Concordance}}{\text{Permissible}}.$$
5. The error rate is Error $= 1 - C$. Note that $0 \leq$ Error ≤ 1 and that Error $= 0.5$ corresponds to a procedure doing no better than random guessing, whereas Error $= 0$ indicates perfect accuracy.

Usage in R

The user interface to `randomSurvivalforest` is similar in many aspects to `randomForest` and as the reader may have already noticed, many of the argument names are also the same. This was done deliberately in order to promote compatibility between the two packages. The primary R function call to the `randomSurvivalforest` package is `rsf()`. The online documentation describes `rsf()` in great detail and there is no reason to repeat this information here. Different R wrapper functions are provided with the `randomSurvivalforest` package to aid in interpreting the object produced by `rsf()`. The examples given below illustrate how some of these wrappers work, and also indicate how `rsf()` might be used in practice.

Lung-vet data

For our first example, we use the well known veteran's administration lung cancer data from Kalbfleisch and Prentice (Kalbfleisch and Prentice, 1980). This is an example data set available within the package. In total there are 6 predictors in the data. We first focus on analysis that includes only Karnofsky score as a predictor:

```
> library("randomSurvivalForest")
> data(veteran,package="randomSurvivalForest")
> ntree <- 1000
> v.out <- rsf(Survrsf(time,status) ~ karno,
        veteran, ntree=ntree, forest=T)
> print(v.out)

Call:
 rsf.default(formula = Survrsf(time, status)
      ~ karno, data = veteran, ntree = ntree)

                         Sample size: 137
                    Number of deaths: 128
                     Number of trees: 1000
          Minimum terminal node size: 3
       Average no. of terminal nodes: 8.437
No. of variables tried at each split: 1
              Total no. of variables: 1
                       Splitting rule: logrank
              Estimate of error rate: 36.28%
```

The error rate is significantly smaller than 0.5, the benchmark value associated with a procedure no better than flipping a coin. This is very strong evidence that Karnofsky score is predictive.

We can investigate the effect of Karnofsky score more closely by considering how the ensemble estimated mortality varies as a function of the predictor:

```
> plot.variable(v.out, partial=T)
```

Figure 1, produced by the above command, is a partial plot of Karnofsky score. The vertical axis represents expected number of deaths.

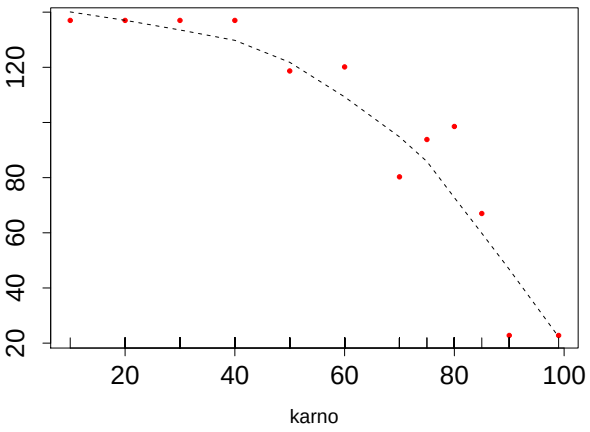

Figure 1: Partial plot of Karnofsky score. Vertical axis is mortality $\sum_{k=1}^{N} N_k \hat{H}_e(t_k^*|x)$ for a given Karnofsky value x and represents expected number of deaths.

Now we run an analysis with all six predictors under each of the four splitting rules. For each splitting rule we run 100 replications and record the mean and standard deviation of the concordance error rate (as before `ntree` equals 1000):

```
> splitrule <- c("logrank", "conserve",
          "logrankscore", "logrankapprox")
> nrep <- 100
> err.rate <- matrix(0, 4, nrep)
> names(err.rate) <- splitrule
> v.f <-
      as.formula("Survrsf(time,status) ~ .")
> for (j in 1:4) {
>  for (k in 1:nrep) {
>   err.rate[j,k] <- rsf(v.f,
      veteran,ntree=ntree,
      splitrule=splitrule[j])$err.rate[ntree]
>  }
> }
> err.rate <- rbind(
               mean=apply(err.rate, 1, mean),
               std=apply(err.rate, 1, sd))
> colnames(err.rate) <- splitrule
> print(round(err.rate,4))

     logrank conserve logrankscore logrankapx
mean  0.2982   0.3239       0.2951     0.3170
std   0.0027   0.0034       0.0027     0.0046
```

The analysis shows that `logrankscore` has the best predictive performance (`logrank` is a close second). Standard deviations in all cases are reasonably small. It is interesting to observe that the mean error rates are not substantially smaller than our previous analysis which used only Karnofsky score, thus indicating the predictor is highly influential. Our next example illustrates further techniques for studying the informativeness of a predictor.

Primary biliary cirrhosis (PBC) of the liver

Next we consider the PBC data set found in appendix D.1 of Fleming and Harrington (Fleming and Harrington, 1991). This is also an example data set available in the package. Similar to the previous analysis we analyzed the data by running a forest analysis for each of the four splitting rules, repeating the analysis 100 times independently (as before `ntree` was set to 1000). The R code is similar as before and suppressed:

```
     logrank conserve logrankscore logrankapx
mean  0.1703   0.1677       0.1719     0.1602
std   0.0014   0.0014       0.0015     0.0020
```

As can be seen, the error rates are between 16-17% with `logrankapprox` having the lowest value.

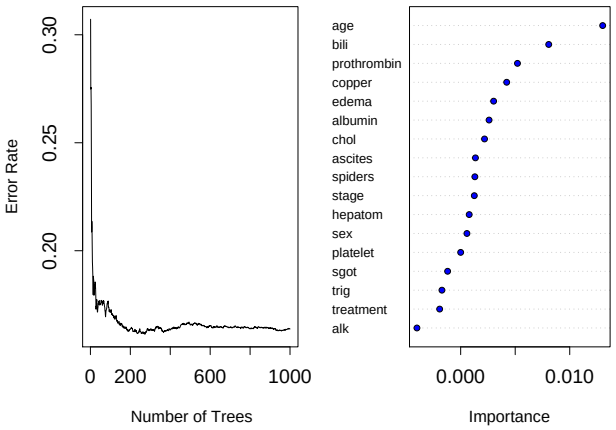

Figure 2: Error rate for PBC data as a function of trees (left-side) and out-of-bag importance values for predictors (right-side).

We now consider the informativeness of each predictor under the `logrankapprox` splitting rule:

```
> data("pbc",package="randomSurvivalForest")
> pbc.f <- as.formula("Survrsf(days,status)~.")
> pbc.out <- rsf(pbc.f, pbc, ntree=ntree,
    splitrule = "logrankapprox", forest=T)
> plot(pbc.out)
```

Figure 2 depicts the importance values for all 17 predictors. From the plot we see that "age" and "bili" are clearly predictive and have substantially larger importance values than all other predictors. The partial plots for the top six predictors are displayed in

Figure 3. The figure was produced using the command:

```
> plot.variable(pbc.out,3,partial=T,n.pred=6)
```

We now consider the incremental effect of each predictor using a nested analysis. We sort predictors by their importance values and consider the nested sequence of models starting with the top variable, followed by the model with the top 2 variables, then the model with the top three variables, and so on:

```
> imp <- pbc.out$importance
> pnames <- pbc.out$predictorNames
> pnames.order <- pnames[rev(order(imp))]
> n.pred <- length(pnames)
> pbc.err <- rep(0, n.pred)
> for (k in 1:n.pred){
>  rsf.f <- "Survrsf(days,status)~"
>  rsf.f <- as.formula(paste(rsf.f,
>    paste(pnames.order[1:k],collapse="+")))
>  pbc.err[k] <- rsf(rsf.f, pbc, ntree=ntree,
>   splitrule="logrankapprox")$err.rate[ntree]
> }
> pbc.imp.out <- as.data.frame(
>   cbind(round(rev(sort(imp)),4),
>         round(pbc.err,4),
>         round(-diff(c(0.5,pbc.err)),4)),
>         row.names=pnames.order)
>colnames(pbc.imp.out) <-
>               c("Imp","Err","Drop Err")
> print(pbc.imp.out)
```

	Imp	Err	Drop Err
age	0.0130	0.3961	0.1039
bili	0.0081	0.1996	0.1965
prothrombin	0.0052	0.1918	0.0078
copper	0.0042	0.1685	0.0233
edema	0.0030	0.1647	0.0038
albumin	0.0026	0.1569	0.0078
chol	0.0022	0.1606	-0.0037
ascites	0.0014	0.1570	0.0036
spiders	0.0013	0.1601	-0.0030
stage	0.0013	0.1557	0.0043
hepatom	0.0008	0.1570	-0.0013
sex	0.0006	0.1549	0.0021
platelet	0.0000	0.1565	-0.0016
sgot	-0.0012	0.1538	0.0027
trig	-0.0017	0.1545	-0.0007
treatment	-0.0019	0.1596	-0.0052
alk	-0.0040	0.1565	0.0032

The first column is the importance value of a predictor in the full model. The kth value in the second column is the error rate for the kth nested model, while the kth value in the third column is the difference between the error rate for the kth and $(k-1)$th nested model, where the error rate for the null model, $k = 0$, is 0.5. One can see not much is gained by using more than 6-7 predictors and that the top 3-4 predictors account for much of the predictive power.

Large scale problems

In terms of computationally challenging problems, we have applied `randomSurvivalForest` successfully to several large survival datasets. For example, we have considered data collected at the Cleveland Clinic involving over 20,000 records and well over 60 predictors. We have also analyzed a data set containing 1,000 records and with almost 250 predictors. Our success with these applications is consistent with that seen for Random Forests: namely, that the methodology has been shown to scale up very nicely, even in very large predictor spaces and with large sample sizes. In terms of computational speed, we have found that `logrankapprox` is almost always fastest. After that, `conserve` is second fastest. For very large datasets, discretizing continuous predictors and/or the observed survival times can greatly speed up computational times. Discretization does not have to be overly granular for substantial gains to be seen.

Acknowledgements

The authors are extremely grateful to Eugene H. Blackstone and Michael S. Lauer for their tremendous support, feedback, and generous time given to us throughout this project. Without their help this project would surely never have been completed. We also thank the referee of the paper and Paul Murrell for their constructive and helpful comments. This research was supported by the National Institutes of Health RO1 grant HL-072771.

Bibliography

L. Breiman. Random forests. *Machine Learning*, 45: 5–32, 2001.

L. Breiman. Bagging predictors. *Machine Learning*, 26:123–140, 1996.

D.R. Cox and D. Oakes. *Analysis of Survival Data*. Chapman and Hall, London, 1998.

T. Fleming and D. Harrington. *Counting Processes and Survival Analysis*. Wiley, New York, 1991.

F. Harrell, R. Califf, D. Pryor, K. Lee, and R. Rosati. Evaluating the yield of medical tests. *J. Amer. Med. Assoc.*, 247:2543–2546, 1982.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *J. Comp. Graph. Statist.*, 15:651–674, 2006.

T. Hothorn, and B. Lausen. On the exact distribution of maximally selected rank statistics. *Comput. Statist. Data Analysis*, 43:121–137, 2003.

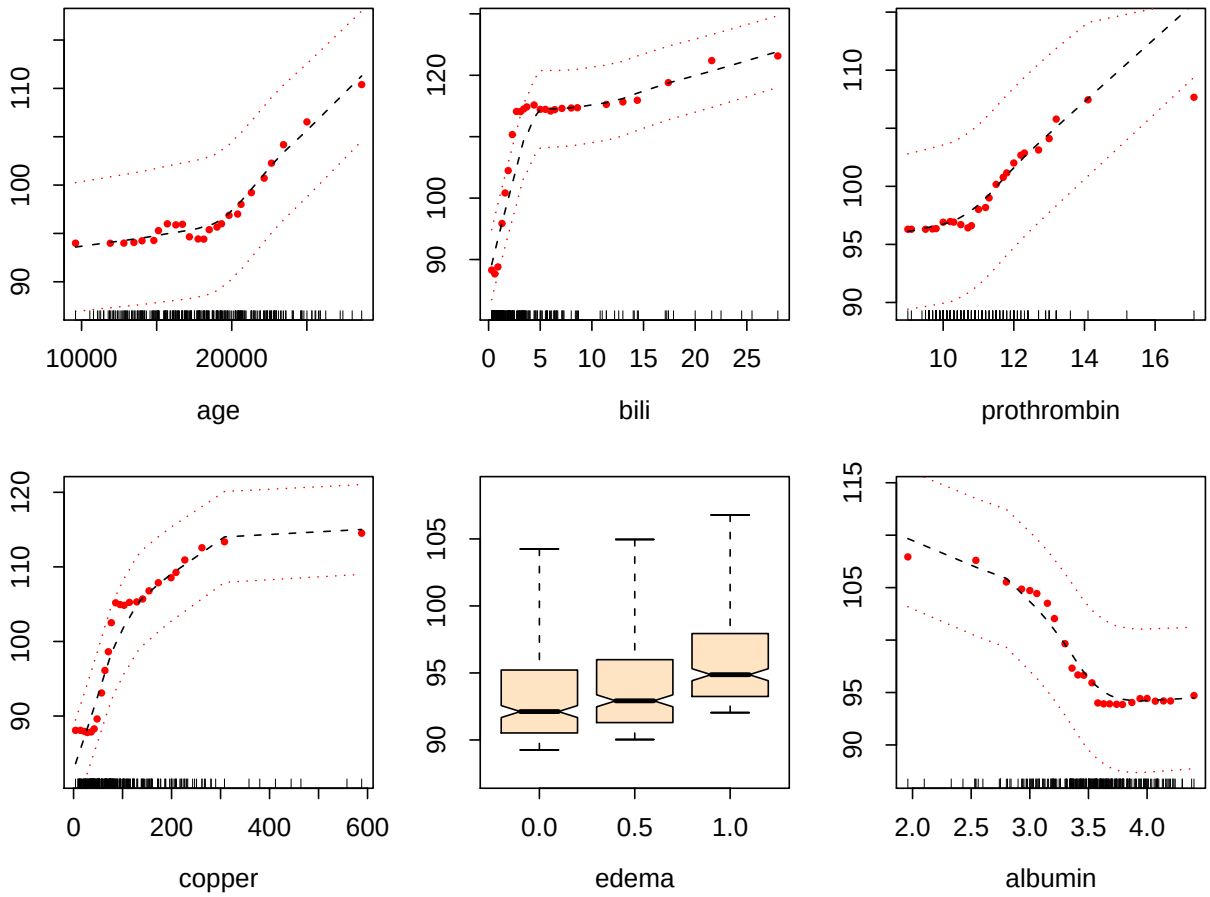

Figure 3: Partial plots for top six predictors from PBC data. Values on the vertical axis represent expected number of deaths for a given predictor, after adjusting for all other predictors. Dashed red lines for continuous predictors are ± 2 standard error bars.

H. Ishwaran, E. Blackstone, M. Lauer, and C. Pothier. Relative risk forests for exercise heart rate recovery as a predictor of mortality. *J. Amer. Stat. Assoc.*, 99: 591–600, 2004.

J. Kalbfleisch and R. Prentice. *The Statistical Analysis of Failure Time Data*. Wiley, New York, 1980.

M. Kattan, K. Hess, and J. Beck. Experiments to determine whether recursive partitioning (CART) or an artificial neural network overcomes theoretical limitations of Cox proportional hazards regression. *Computers and Biomedical Research*, 31:363–373, 1998.

M. LeBlanc and J. Crowley. Relative risk trees for censored survival data. *Biometrics*, 48:411–425, 1992.

M. LeBlanc and J. Crowley. Survival trees by goodness of split. *J. Amer. Stat. Assoc.*, 88:457–467, 1993.

M. May, P. Royston, M. Egger, A.C. Justice and J.A.C. Sterne. Development and validation of a prognostic model for survival time data: application to prognosis of HIV positive patients treated with antiretroviral therapy. *Statist. Medicine.*, 23: 2375–2398, 2004.

D. Naftel, E. Blackstone, and M. Turner. Conservation of events, 1985. Unpublished notes.

R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. *Ann. Statist.*, 26(5):1651–1686, 1998.

M.R. Segal. Regression trees for censored data. *Biometrics*, 44:35–47, 1988.

T. Therneau and E. Atkinson. An introduction to recursive partitioning using the rpart routine. Technical Report 61, 1997. Section of Biostatistics, Mayo Clinic, Rochester.

Hemant Ishwaran
Department of Quantitative Health Sciences
Cleveland Clinic, Cleveland, U.S.A.
`hemant.ishwaran@gmail.com`
Udaya B. Kogalur
Kogalur Shear Corporation
Clemmons, U.S.A.
`ubk@kogalur-shear.com`

Rwui: A Web Application to Create User Friendly Web Interfaces for R Scripts

by Richard Newton and Lorenz Wernisch

Summary

The web application Rwui is used to create web interfaces for running R scripts. All the code is generated automatically so that a fully functional web interface for an R script can be downloaded and up and running in a matter of minutes.

Rwui is aimed at R script writers who have scripts that they want people unversed in R to use. The script writer uses Rwui to create a web application that will run their R script. Rwui allows the script writer to do this without them having to do any web application programming, because Rwui generates all the code for them.

The script writer designs the web application to run their R script by entering information on a sequence of web pages. The script writer then downloads the application they have created and installs it on their own server. This is a simple matter of copying one file from the download onto the server. The script writer now has a web application on their server that runs their R script, that other people can use over the web.

Although of general applicability, Rwui was designed primarily with bioinformatics applications in mind; aimed at bioinformaticians who are developing a statistical analysis of experimental data for collaborators and who want to automate their analysis in a user friendly way. Rwui may also be of use for creating teaching applications.

Rwui may be found at http://rwui.cryst.bbk.ac.uk

Introduction

R is widely used in the field of bioinformatics. The Bioconductor project (Gentleman *et al.*, 2004) contains R packages specifically designed for this field. However many potential users of bioinformatics programs written in R, who come from a non-bioinformatics background, are unfamiliar with the language. One solution to this problem is for developers of R scripts to provide user-friendly web interfaces for their scripts.

Rwui (R Web User Interface) is a web application that the developer of an R script can use to create a web application for running their script. All the code for the web application is generated automatically. This is the key feature of Rwui and means that it only takes a few minutes for someone who is entirely unfamiliar with web application programming to design, download and install on their server a fully functional web interface for an R script.

A web interface for an R script means that the script can be used by anyone, even if they have no knowledge of R. Instead of using the R script directly, values for variables and data files for processing are first submitted by the user on a web form. The application then runs the R script on a server, out of sight of the user, and returns the results of the analysis to the user's web page. Because the web application runs on a server it can be accessed remotely and the user does not need to have R installed on their machine. And updates to the script need only be made to the copy on the server.

Although of general applicability, Rwui has been designed with bioinformatics applications in mind. To this end the web applications created by Rwui can include features often required in bioinformatics data analysis, such as a page for uploading replicate data files and their group identifiers. Rwui is typically aimed at bioinformaticians who have developed an R script for analysing experimental data and who want to make their method immediately accessible to collaborators who are unfamiliar with R. Rwui enables bioinformaticians to do this quickly and simply, without having to concern themselves with any aspects of web application programming.

The completed web applications run on Tomcat servers (http://tomcat.apache.org/). Tomcat is free and widely used server software, very easy to install on both Unix and Windows machines, and with an impeccable security record. There have been no known instances of the safety of data on Tomcat servers being compromised despite its widespread use. But if data cannot be sent over the internet then either the web applications created by Rwui can be installed on a local Tomcat server for internal use, situated behind a firewall, or Tomcat and the web applications created by Rwui can be installed on individual stand-alone machines, in which case the web applications are accessed in a browser on the machine via the 'localhost' URL. Instructions on installing and running Tomcat can be accessed from a link on the 'Help' page of Rwui.

A number of approaches to providing web interfaces to R scripts exist. A list with links can be found on the R website (http://cran.r-project.org/doc/FAQ/R-FAQ.html) in the section 'R Web Interfaces'. These fall into two main categories. Firstly there are projects which allow the user to submit R code to a remote server. With these methods, the end user must handle R code. In contrast, users of applications created by Rwui have no contact with R code.

Secondly there are projects which facilitate pro-

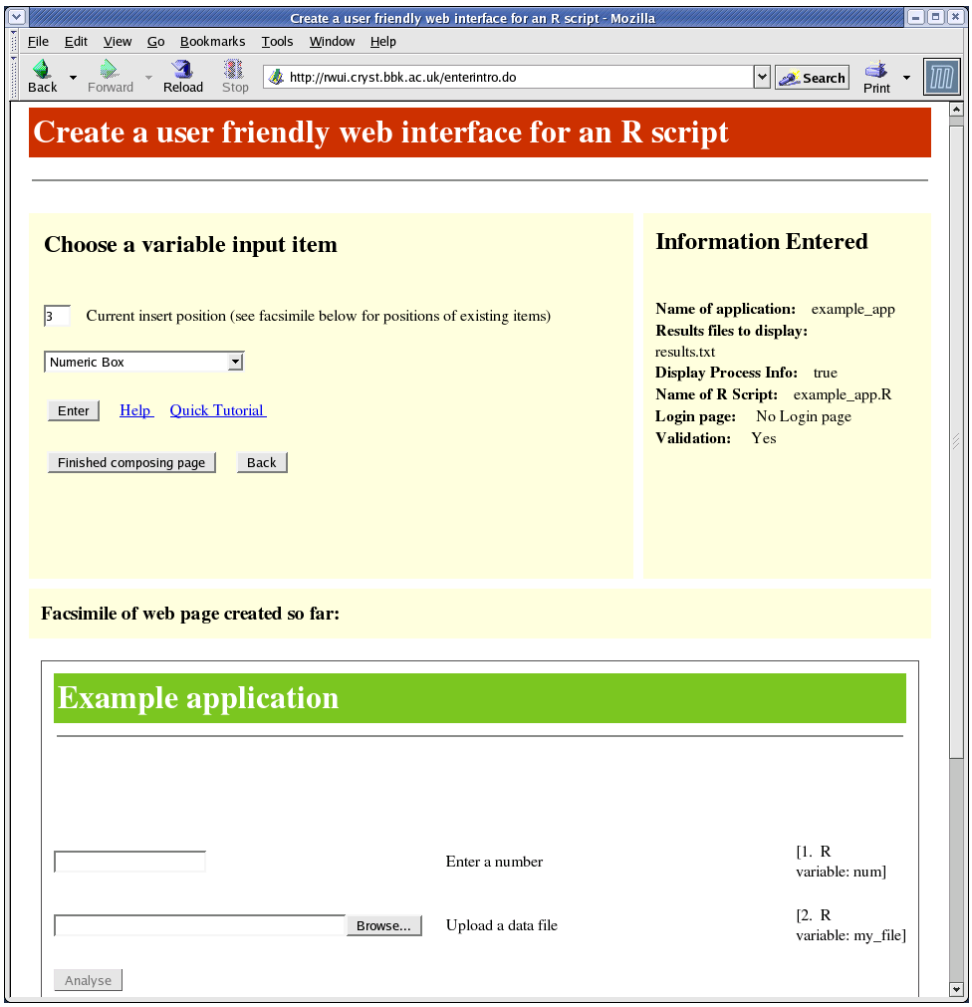

Figure 1: Screenshot showing the web page of Rwui where input items are added.

gramming web applications that run R code. With these methods, whoever is creating the application must write web application code. In contrast, when using Rwui no web application code needs to be written; a web interface for an R script is created simply by selecting options from a sequence of web pages.

Using the Application

The information that Rwui requires in order to create a web application for running an R script is entered on a sequence of forms. After entering a title and introductory text for the application, the input items that will appear on the application's web page are selected. Input items may be Numeric or Text entry boxes, Checkboxes, Drop-down lists, Radio Buttons, File Upload boxes and a Multiple/Replicate File Upload page. Each of the input variables of the R script, that is, those variables in the script that require a value supplied by the user, must have a corresponding input item on the application's web page. Figure 1 shows the web page of Rwui on which the input items that will appear in the application are added.

Section headings can also be added if required. Rwui displays a facsimile of the web page that has been created as items are added to the page. This can be seen in the lower part of the screenshot shown in Figure 1. Input items are given a number so that items can be deleted and new items inserted between existing items. After uploading the R script, Rwui generates the web application, which can be downloaded as a zip or tgz file.

Rwui creates Java based applications that use the Apache Struts framework (http://struts.apache.org/). Struts is open source and a popular framework for constructing well-organised, stable and extensible web applications.

The completed applications will run on a Tomcat server. All that needs to be done to use the downloaded web application is to place the application's 'web application archive' file, which is contained in the download, into a subdirectory of the Tomcat server. In addition, the file permissions of an included shell script must be changed to executable.

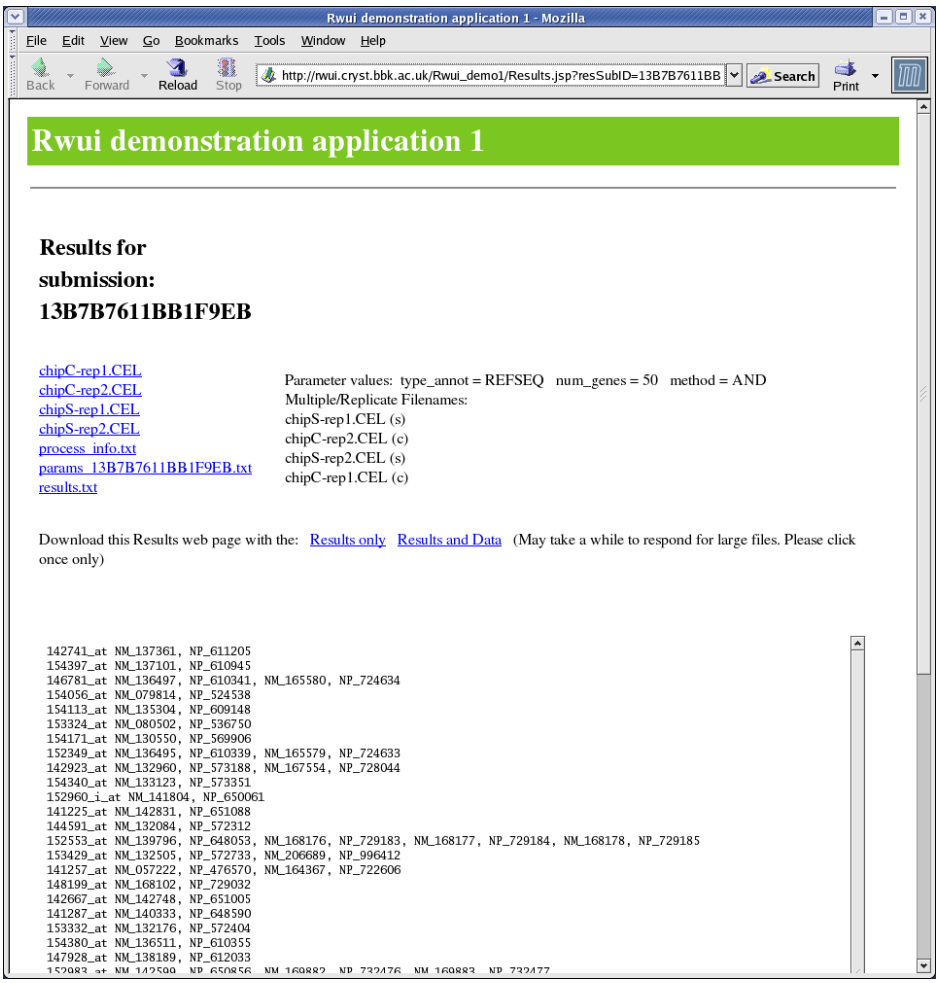

Figure 2: Screenshot showing an example of a results web page of an application created using Rwui.

An application description file, written in XML, is included in the download. This is useful if the application requires modification at a later date. The details of the application can be re-entered into Rwui by uploading the application description file. The application can then be edited and rebuilt within Rwui.

Further information on using Rwui can be found from links on the application's web pages. The 'Quick Tour' provides a ten minute introductory example of using Rwui. The 'Technical Report' gives a technical overview of the application. The 'Help' link accesses the manual for Rwui which contains detailed information for users.

System Requirements

In order to use the web applications created by Rwui a machine is required with Tomcat version 5.0 or later, Java version 1.5 and an R version compatible with the R script(s). Although a server running a Unix operating system is preferable, the applications will work without modification on a Tomcat server running Windows XP.

R script Requirements

An R script needs little or no modification in order to be run from a web application created by Rwui. There are three areas where the script may require attention.

The R script receives input from the user via R variables, which we term the input variables of the script. The values of input variables are entered by the user on the web page of the application. The input variables must be named according to the rules of both R and Java variable naming. These rules are given on the 'Help' page of Rwui. But those variables in the R script that are not input variables do not need to conform to the rules of Java variable naming.

Secondly, in order to make the results of the analysis available to the user, the R script must save the results to files. Thirdly, and optionally, the R script may also write progress information into a file which can be continously displayed for the user in a Javascript pop-up window.

Using applications created by Rwui

A demonstration application created by Rwui can be accessed from the 'Help' page of Rwui.

Applications created by Rwui can include a login page. Access can be controlled either by a single password, or by username/password pairs.

If an application created by Rwui includes a Multiple/Replicate File upload page, then the application consists of two web pages on which the user enters information. On the first web page the user uploads multiple files one at a time. Once completed, a button takes the user to a second web page where singleton data files and values for all other variables are entered. The 'Analyse' button on this page submits the values of variables, uploads any data files and runs the R script. If a Multiple/Replicate File upload page is not included, the application consists of this second web page only.

Before running the R script the application first checks the validity of the values that the user has entered and returns an error message to the page if any are invalid. During the analysis, progress information can be displayed for the user. To enable this the R script must append information to a text file at stages during its execution. This text file is displayed for the user in a JavaScript pop-up window which refreshes at fixed intervals.

On completion of the analysis, a link to a Results page appears at the bottom of the web page. The user can change data files and/or the values of any of the variables and re-analyse, and the new results will appear as a second link at the bottom of the page, and so on. Clicking on a link brings up the Results page for the corresponding analysis. Figure 2 shows an example of a results page.

The user can download individual results files by clicking on the name of the appropriate file on a Results page. Alternatively, each Results page also contains a link which will download all the results files from the page and the html of the page itself. In this way the user can view offline saved Results pages with their associated results files. Any uploaded data files are also linked to on the Results page, giving the user the opportunity to check that the correct data has been submitted and has been uploaded correctly.

The applications include a SessionListener which detects when a user's session is about to expire. The SessionListener then removes all the working directories created during the session from the server, to prevent it from becoming clogged with data. By default a session expires 30 minutes after it was last accessed by the user, but the delay can be changed in the server configuration.

All the code of a complete demonstration application created by Rwui can be downloaded from a link on the 'Help' page of Rwui.

Acknowledgment

RN was funded as part of a Wellcome Trust Functional Genomics programme grant.

Bibliography

Gentleman,R.C., Carey,V.J., Bates,D.M., Bolstad,B., Dettling,M., Dudoit,S., Ellis,B., Gautier,L., Ge,Y., Gentry,J., Hornik,K., Hothorn,T., Huber,W., Iacus,S., Irizarry,R., Leisch,F., Li,C., Maechler,M., Rossini,A.J., Sawitzki,G., Smith,C., Smyth,G., Tierney,L., Yang,J.Y.H., Zhang,J. (2004) Bioconductor: Open software development for computational biology and bioinformatics, *Genome Biology*, **5**, R80.

Richard Newton
School of Crystallography
Birkbeck College, University of London, UK
`r.newton@mail.cryst.bbk.ac.uk`
Lorenz Wernisch
MRC Biostatistics Unit, Cambridge, UK
`lorenz.wernisch@mrc-bsu.cam.ac.uk`

The np Package: Kernel Methods for Categorical and Continuous Data

by Tristen Hayfield and Jeffrey S. Racine

Readers of R News are certainly aware of a number of nonparametric kernel[1] methods that exist in R base (e.g., `density`) and in certain R packages (e.g., `locpoly` in the **KernSmooth** package). Such functionality allows R users to nonparametrically model a density or to conduct nonparametric local polynomial regression, to name but two applications of kernel methods. Nonparametric kernel approaches are appealing to applied researchers because they often reveal features in the data that might be missed by classical parametric methods. However, traditional nonparametric kernel methods presume that the underlying data is continuous, which is frequently not the case.

[1] A 'kernel' is simply a weighting function.

Practitioners often encounter a mix of categorical and continuous data types, but may still wish to proceed in a nonparametric direction. The traditional nonparametric approach is called a 'frequency' approach, whereby data is broken up into subsets ('cells') corresponding to the values assumed by the categorical variables, and only then do you apply say `density` or `locpoly` to the continuous data remaining in each cell. Nonparametric frequency approaches are widely acknowledged to be unsatisfactory. Recent theoretical developments offer practitioners a variety of kernel-based methods for categorical (i.e., unordered and ordered factors) data only or for a mix of continuous and categorical data; see Li and Racine (2007) and the references therein for an in-depth treatment of these methods, and also see the articles listed in the bibliography below.

The **np** package implements recently developed kernel methods that seamlessly handle the mix of continuous, unordered, and ordered factor data types often found in applied settings. The package also allows users to create their own routines using high-level function calls rather than writing their own C or Fortran code.[2] The design philosophy underlying **np** is simply to provide an intuitive, flexible, and extensible environment for applied kernel estimation.

Currently, a range of methods can be found in the **np** package including unconditional (Li and Racine, 2003; Ouyang et al., 2006) and conditional (Hall et al., 2004; Racine et al., 2004) density estimation and bandwidth selection, conditional mean and gradient estimation (local constant (Racine and Li, 2004; Hall et al., forthcoming) and local polynomial (Li and Racine, 2004)), conditional quantile and gradient estimation (Li and Racine, forthcoming), model specification tests (regression (Hsiao et al., forthcoming), quantile, significance (Racine et al., forthcoming)), semiparametric regression (partially linear, index models, average derivative estimation, varying/smooth coefficient models), etc.

Before proceeding, we caution the reader that data-driven bandwidth selection methods can be numerically intensive, which is the reason underlying the development of an MPI-aware[3] version of the **np** package that uses some of the functionality of the **Rmpi** package, which we have tentatively called the **npRmpi** package. The functionality of **np** and **npRmpi** will be identical; however, using **npRmpi** you could take advantage of a cluster computing environment or a multi-core/multi-cpu desktop machine thereby alleviating the computational burden associated with the nonparametric analysis of large datasets. We ought also to point out that data-driven (i.e., automatic) bandwidth selection procedures are not guaranteed always to produce good results. For this reason, we advise the reader to interrogate their bandwidth objects with the `summary` command which produces a table of the bandwidths for the continuous variables scaled by an appropriate constant ($\sigma_x n^{\alpha}$ where α depends on the kernel order and number of continuous variables, e.g. $\alpha = -1/5$ for one continuous variable and a second order kernel), which some readers may find helpful. Also, the admissible range for the bandwidths for the categorical variables is provided when `summary` is used, which some readers may also find helpful.

We have tried to make **np** flexible enough to be of use to a wide range of users. All options can be tweaked by users (kernel function, kernel order, bandwidth type, estimator type and so forth). One function, `npksum`, allows you to create your own estimators, tests, etc. The function `npksum` is simply a call to highly optimized C code, so you get the benefits of compiled code along with the power and flexibility of the R language. We hope that incorporating the `npksum` function renders the package suitable for teaching and research alike.

There are two ways in which you can interact with functions in **np**: i) using data frames, or ii) using a formula interface, where appropriate.

To some, it may be natural to use the data frame interface. The R `data.frame` function preserves a variable's type once it has been cast (unlike `cbind`, which we avoid for this reason). If you find this most natural for your project, you first create a data frame casting data according to their type (i.e., one of continuous (default), `factor`, `ordered`), as in

```
data.object <- data.frame(x1=factor(x1),
  x2, x3=ordered(x3))
```

where `x1` is, say, a binary factor, `x2` continuous, and `x3` an ordered factor. Then you could pass this data frame to the appropriate **np** function, say

```
npudensbw(dat=data.object)
```

To others, however, it may be natural to use the formula interface that is used for the regression example outlined below. For nonparametric regression functions such as `npregbw`, you would proceed just as you would using `lm`, e.g.,

```
npregbw(y ~ factor(x1) + x2)
```

except that you would of course not need to specify, e.g., polynomials in variables or interaction terms. Every function in **np** supports both interfaces, where appropriate.

Before proceeding, a few words on the formula interface are in order. We use the standard formula interface as it provided capabilities for handling missing observations and so forth. This interface, however, is simply a convenient device for telling a routine which variable is, say, the outcome

[2] The high-level functions found in the package in turn call compiled C code, allowing users to focus on the application rather than the implementation details.

[3] MPI is a library specification for message-passing, and has emerged as a dominant paradigm for portable parallel programming.

and which are, say, the covariates. That is, just because one writes `x1 + x2` in no way means or is meant to imply that the model will be linear and additive (why use fully nonparametric methods to estimate such models in the first place?). It simply means that there are, say, two covariates in the model, the first being `x1` and the second `x2`, we are passing them to a routine with the formula interface, and nothing more is presumed or implied.

Regardless of whether you use the data frame or formula interface, workflow for nonparametric estimation in **np** typically proceeds as follows:

1. compute appropriate bandwidths;
2. estimate an object and extract fitted or predicted values, standard errors, etc.;
3. optionally, plot the object.

In order to streamline the creation of a set of complicated graphics objects, `plot` (which calls `npplot`) is dynamic; i.e., you can specify, say, bootstrapped error bounds and the appropriate routines will be called in real time.

Efficient nonparametric regression in the presence of qualitative data

We begin with a motivating example that demonstrates the potential benefits arising from the use of kernel smoothing methods that smooth both the qualitative and quantitative variables in a particular manner; see the subsequent section for more detailed information regarding the method itself. For what follows, we consider an application taken from Wooldridge (2003, pg. 226) that involves multiple regression analysis with qualitative information.

We consider modeling an hourly wage equation for which the dependent variable is log(`wage`) (`lwage`) while the explanatory variables include three continuous variables, namely `educ` (years of education), `exper` (the number of years of potential experience), and `tenure` (the number of years with their current employer) along with two qualitative variables, `female` ("Female"/"Male") and `married` ("Married"/"Notmarried"). For this example there are $n = 526$ observations.

The classical parametric approach towards estimating such relationships requires that one first specify the functional form of the underlying relationship. We start by first modelling this relationship using a simple parametric linear model. By way of example, Wooldridge (2003, pg. 222) presents the following model:[4]

```
> data("wage1")
>
> model.ols <- lm(lwage ~ factor(female) +
+  factor(married) + educ + exper + tenure,
+  data=wage1)
>
> summary(model.ols)
....
```

This model is, however, restrictive in a number of ways. First, the analyst must specify the functional form (in this case linear) for the continuous variables (`educ`, `exper`, and `tenure`). Second, the analyst must specify how the qualitative variables (`female` and `married`) enter the model (in this case they affect the model's intercepts only). Third, the analyst must specify the nature of any interactions among all variables, quantitative and qualitative (in this case, there are none). Should any of these assumptions be incorrect, then the estimated model will be biased and inconsistent, potentially leading to faulty inference.

One might next test the null hypothesis that this parametric linear model is correctly specified using the consistent model specification test found in Hsiao et al. (forthcoming) that admits both categorical and continuous data (this example will likely take a few minutes on a desktop computer as it uses bootstrapping and cross-validated bandwidth selection):

```
> data("wage1")
> attach(wage1)
>
> model.ols <- lm(lwage ~ factor(female) +
+  factor(married) + educ + exper + tenure,
+  x=TRUE, y=TRUE)
>
> X <- data.frame(factor(female),
+  factor(married), educ, exper, tenure)
>
> output <- npcmstest(model=model.ols,
+  xdat=X, ydat=lwage, tol=.1, ftol=.1)
> summary(output)

Consistent Model Specification Test
....
IID Bootstrap (399 replications)
Test Statistic 'Jn': 5.594745e+00
P Value: 0.000000e+00
....
```

This naïve linear model is rejected by the data (the P-value for the null of correct specification is < 0.001), hence one might proceed instead to model this relationship using kernel methods.

As noted, the traditional nonparametric approach towards modeling relationships in the presence of qualitative variables requires that you first split your data into subsets containing only the continuous

[4] We would like to thank Jeffrey Wooldridge for allowing us to use his data. Also, we would like to point out that Wooldridge starts out with this naïve linear model, but quickly moves on to a more realistic model involving nonlinearities in the continuous variables and so forth.

variables of interest (lwage, exper, and tenure). For instance, we would have four such subsets, a) $n = 132$ observations for married females, b) $n = 120$ observations for single females, c) $n = 86$ observations for single males, and d) $n = 188$ observations for married males. One would then construct smooth nonparametric regression models for each of these subsets and proceed with the analysis in this fashion. However, this may lead to a loss in efficiency due to a reduction in the sample size leading to overly variable estimates of the underlying relationship.

Instead, however, we could construct smooth nonparametric regression models by i) using a smoothing function that is appropriate for the qualitative variables such as that proposed by Aitchison and Aitken (1976), and ii) modifying the nonparametric regression model as was done by Li and Racine (2004). One can then conduct sound nonparametric estimation based on the $n = 526$ observations rather than resorting to sample splitting. The rationale for this lies in the fundamental concept that doing so may introduce potential bias, but it will always reduce variability, thus leading to potential finite-sample efficiency gains. Our experience has been that the potential benefits arising from this approach more than offset the potential costs in finite-sample settings.

Next, we consider using the local-linear nonparametric method described in Li and Racine (2004). For the reader's convenience we supply precomputed cross-validated bandwidths which are automatically loaded when one reads the wage1 dataset. The commented out code that generates the cross-validated bandwidths is also provided should the reader wish to fully replicate the example (recall being cautioned about the computational burden associated with multivariate data-driven bandwidth methods).

```
> data("wage1")
>
> #bw.all <- npregbw(lwage ~ factor(female) +
> # factor(married) + educ + exper + tenure,
> # regtype="ll", bwmethod="cv.aic",
> # data=wage1)
>
> model.np <- npreg(bws=bw.all)
>
> summary(model.np)

Regression Data: 526 training points,
                 in 5 variable(s)
....
Kernel Regression Estimator: Local-Linear
Bandwidth Type:Fixed
Residual standard error: 1.371530e-01
R-squared: 5.148139e-01
....
```

Note that the bandwidth object is the only thing you need to pass to npreg as it encapsulates the kernel types, regression method, and so forth. You could, of course, use npreg with a formula and perhaps manually specify the bandwidths using a bandwidth vector if you so chose. We have tried to make each function as flexible as possible to meet the needs of a variety of users.

The goodness of fit of the nonparametric model ($R^2 = 51.5\%$) is better than that for the parametric model ($R^2 = 40.4\%$). In order to investigate whether this apparent improvement reflects overfitting or simply that the nonparametric model is in fact more faithful to the underlying data generating process, we shuffled the data and created two independent samples, one of size $n_1 = 400$ and one of size $n_2 = 126$. We fit the models on the n_1 training observations then evaluate the models on the n_2 independent hold-out observations using the predicted square error criterion, namely $n_2^{-1}\sum_{i=1}^{n_2}(y_i - \hat{y}_i)^2$, where the y_is are the lwage values for the hold-out observations and the $\hat{y}_i$s are the predicted values. Finally, we compare the parametric model, the nonparametric model that smooths both the qualitative and quantitative variables, and the traditional frequency nonparametric model that breaks the data into subsets and smooths the quantitative data only.

```
> data("wage1")
> set.seed(123)
>
> # Shuffle the data and create two datasets...
>
> ii <- sample(seq(nrow(wage1)),replace=FALSE)
>
> wage1.train <- wage1[ii[1:400],]
> wage1.eval <- wage1[ii[401:nrow(wage1)],]
>
> # Compute the parametric model for the
> # training data...
>
> model.ols <- lm(lwage ~ factor(female) +
+  factor(married) + educ + exper + tenure,
+  data=wage1.train)
>
> # Obtain the out-of-sample predictions...
>
> fit.ols <- predict(model.ols,
+  data=wage1.train, newdata=wage1.eval)
>
> # Compute the predicted square error...
>
> pse.ols <- mean((wage1.eval$lwage -
+                  fit.ols)^2)
>
> #bw.subset <- npregbw(
> # lwage~factor(female) + factor(married) +
> # educ + exper + tenure, regtype="ll",
> # bwmethod="cv.aic", data=wage1.train)
>
```

```
> model.np <- npreg(bws=bw.subset)
>
> # Obtain the out-of-sample predictions...
>
> fit.np <- predict(model.np,
+  data=wage1.train, newdata=wage1.eval)
>
> # Compute the predicted square error...
>
> pse.np <- mean((wage1.eval$lwage -
+                 fit.np)^2)
>
> # Do the same for the frequency estimator...
> # We do this by setting lambda=0 for the
> # qualitative variables...
>
> bw.freq <- bw.subset
> bw.freq$bw[1] <- 0
> bw.freq$bw[2] <- 0
>
> model.np.freq <- npreg(bws=bw.freq)
>
> # Obtain the out-of-sample predictions...
>
> fit.np.freq <- predict(model.np.freq,
+  data=wage1.train, newdata=wage1.eval)
>
> # Compute the predicted square error...
>
> pse.np.freq <- mean((wage1.eval$lwage -
+                      fit.np.freq)^2)
>
> # Compare performance for the
> # hold-out data...
>
> pse.ols
[1] 0.1469691
> pse.np
[1] 0.1344028
> pse.np.freq
[1] 0.1450111
```

The predicted square error on the hold-out data was 0.147 for the parametric linear model, 0.145 for the traditional nonparametric estimator that splits the data into subsets, and 0.134 for the nonparametric estimator that uses the full sample but smooths both the qualitative and quantitative data. We therefore conclude that the nonparametric model that smooths both the continuous and qualitative variables appears to provide a better description of the underlying data generating process than either the nonparametric model that uses sample splitting or the naïve linear model.

Note that for this example we have only four cells. If one used all qualitative variables included in the dataset (16 in total), one would have 65,536 cells, many of which would be empty, and most having far too few observations to provide meaningful nonparametric estimates. As the number of qualitative variables increases, the difference between the estimator that smooths both continuous and discrete variables in a particular manner and the traditional estimator that relies upon sample splitting will become even more pronounced.

We now briefly outline the essence of kernel smoothing of categorical data for the interested reader.

Categorical data kernel methods

To introduce the R user to the basic concept of kernel smoothing of categorical random variables, consider a kernel estimator of a probability function, i.e. $p(x^d) = P(X^d = x^d)$, where X^d is an unordered (i.e., discrete) categorical variable assuming values in, say, $\{0, 1, 2, \ldots, c-1\}$ where c is the number of possible outcomes of the discrete random variable X^d. Aitchison and Aitken (1976) proposed a kernel estimator of $p(x^d)$ given by

$$\hat{p}(x^d) = \frac{1}{n}\sum_{i=1}^{n} L(X_i^d = x^d),$$

where $L(\cdot)$ is a kernel function defined by, say,

$$L(X_i^d = x^d) = \begin{cases} 1-\lambda & \text{if } X_i^d = x^d \\ \lambda/(c-1) & \text{otherwise,} \end{cases}$$

and where λ is a 'smoothing' parameter that can be selected via a variety of data-driven methods; see also Ouyang et al. (2006) for theoretical underpinnings of least-squares cross-validation in this context.

The smoothing parameter λ is restricted to lie in $[0, (c-1)/c]$ for the kernel given above. Note that when $\lambda = 0$, then $L(X_i^d = x^d)$ becomes an indicator function, and hence $\hat{p}(x^d)$ would be the standard frequency probability estimator (i.e., the sample proportion of $X_i^d = x^d$). If, on the other hand, $\lambda = (c-1)/c$, its upper bound, then $L(X_i^d = x^d)$ becomes a constant function and $\hat{p}(x^d) = 1/c$ for all x^d, which is the discrete uniform distribution. Data-driven methods for selecting λ are based on minimizing expected square error loss, among other criteria.

A number of issues surrounding this estimator are noteworthy: i) this approach extends trivially to a general multivariate setting, and in fact is best suited to such settings; ii) to deal with ordered variables you simply use an 'ordered kernel'; iii) there is no "curse of dimensionality" associated with smoothing discrete probability functions – the estimators are $\sqrt{n}$ consistent, just like their parametric counterparts; and iv) you can "mix-and-match" data types using "generalized product kernels" as we describe in the next section.

Why would anyone want to smooth a probability function in the first place? For finite-sample efficiency reasons of course. That is, smoothing a probability function may introduce some finite-sample

bias, but, it *always* reduces variability. In settings where you are modeling a mix of categorical and continuous data or where one has sparse multivariate categorical data, the efficiency gains can be substantial indeed.

Mixed data kernel methods

Estimating a joint density function defined over mixed data (i.e., both categorical and continuous) follows naturally using generalized product kernels. For example, for one discrete variable x^d and one continuous variable x^c, our kernel estimator of the joint PDF would be

$$\hat{f}(x^d, x^c) = \frac{1}{nh_x} \sum_{i=1}^{n} L(X_i^d = x^d) W\left(\frac{X_i^c - x^c}{h_{x^c}}\right),$$

where $L(X_i^d = x^d)$ is a categorical data kernel function, while $W[(X_i^c - x^c)/h_{x^c}]$ is a continuous data kernel function (e.g., Epanechnikov, Gaussian, or uniform) and h_{x^c} is the bandwidth for the continuous variable; see Li and Racine (2003) for further details.

Once we can consistently estimate a joint density function defined over mixed data, we can then proceed to estimate a range of statistical objects of interest to practitioners. Some mainstays of applied data analysis include estimation of regression functions and their derivatives, conditional density functions and their quantiles, conditional mode functions (i.e., count data models, probability models), and so forth, each of which is currently implemented.

Nonparametric estimation of binary outcome and count data models

For what follows, we adopt the conditional probability estimator proposed in Hall et al. (2004) to estimate a nonparametric model of a binary outcome when there exist a number of categorical covariates.

For this example, we use the `birthwt` data taken from the **MASS** package, and compute a parametric Logit model and a nonparametric conditional mode model. We then compare their confusion matrices and assess their classification ability. The outcome is an indicator of low infant birthweight (0/1).

```
> library("MASS")
> attach(birthwt)
>
> # First, we model this with a simple
> # parametric Logit model...
>
> model.logit <- glm(low ~ factor(smoke) +
+  factor(race) + factor(ht) + factor(ui)+
+  ordered(ftv) + age + lwt, family=binomial)
>
> # Now we model this with the nonparametric
> # conditional density estimator and compute
> # the conditional mode...
>
> bw <- npcdensbw(factor(low) ~
+  factor(smoke) + factor(race) +
+  factor(ht) + factor(ui)+ ordered(ftv) +
+  age + lwt, tol=.1, ftol=.1)
> model.np <- npconmode(bws=bw)
>
> # Finally, we compare confusion matrices
> # from the logit and nonparametric models...
>
> # Parametric...
>
> cm <- table(low,
+  ifelse(fitted(model.logit) > 0.5, 1, 0))
> cm

low   0   1
  0 119  11
  1  34  25
>
> # Nonparametric...
>
> summary(model.np)

Conditional Mode data: 189 training points,
                       in 8 variable(s)
....
Bandwidth Type: Fixed

Confusion Matrix
      Predicted
Actual   0   1
     0 127   3
     1  27  32

....
```

For this example the nonparametric model is better able to predict low birthweight infants than its parametric counterpart, correctly predicting 159/189 birthweights compared with 144/189 for the parametric model.

Visualizing and summarizing nonparametric results

Summarizing nonparametric results and comparing them with parametric ones is perhaps one of the main challenges faced by the practitioner. We have attempted to provide a range of summary measures and plotting methods to facilitate these tasks.

The `plot` function (which calls `npplot`) can be used on most nonparametric objects. In order to further facilitate comparison of parametric with nonparametric results, we also compute a number of summary measures of goodness of fit, normalized bandwidths ('scale factors') and so forth that are ac-

cessed by the summary command, while objects generated by **np** also contain various alternative measures of goodness of fit and the like. We again consider the approach detailed in Li and Racine (2004) which conducts local linear kernel regression with a mix of discrete and continuous data types on a popular macroeconomic dataset. The original study assessed the impact of Organisation for Economic Co-operation and Development (OECD) membership on a country's growth rate when controlling for a range of factors such as population growth rates, stock of capital (physical and human) and so forth. For this example we shall use the formula interface rather than the data frame interface. For this example, we have already computed the cross-validated bandwidths which are loaded when one reads the oecdpanel dataset.

```
> data("oecdpanel")
> attach(oecdpanel)
>
> #bw <- npregbw(growth ~ factor(oecd) +
> # ordered(year) + initgdp + popgro+ inv +
> # humancap, bwmethod="cv.aic",
> # regtype="ll")
>
> summary(bw)

Regression Data (616 observations,
                  6 variable(s)):

Regression Type: Local-Linear

....

>
> model <- npreg(bw)
>
> summary(model)

Regression Data: 616 training points,
                 in 6 variable(s)
....
>
> plot(bw,
+      plot.errors.method="bootstrap",
+      plot.errors.boot.num=25)
>
```

We display partial regression plots in Figure 1.[5] We also plot bootstrapped variability bounds, where the bootstrapping is done via the **boot** package thereby facilitating a variety of bootstrap methods.

If you prefer gradients rather than levels, you can simply use the argument gradients=TRUE in plot to indicate that you wish to plot partial derivatives rather than the conditional mean function itself.[6]

Many of the accessor functions such as predict, fitted, and residuals work with most nonparametric objects, where appropriate. As well, gradients is a generic function which extracts gradients from objects. The R function coef can be used for extracting the parametric coefficients from semiparametric models such as those created with npplreg.

Creating mixed data kernel objects via npksum

Rather than being limited to only those kernel methods that exist in **np**, you could instead create your own mixed data kernel objects. npksum is a function that computes kernel sums on evaluation data, given a set of training data, data to be weighted (optional), and a bandwidth specification (any bandwidth object). npksum uses highly-optimized C code that strives to minimize its memory footprint, while there is low overhead involved when using repeated calls to this function. The convolution kernel option would allow you to create, say, the least squares cross-validation function for kernel density estimation. You can choose powers to which the kernels constituting the sum are raised (default=1), whether or not to divide by bandwidths, whether or not to generate leave-one-out sums, and so on. Three classes of kernel methods for the continuous data types are available: fixed, adaptive nearest-neighbor, and generalized nearest-neighbor. Adaptive nearest-neighbor bandwidths change with each sample realization in the set, x_i, when estimating the kernel sum at the point x. Generalized nearest-neighbor bandwidths change with the point at which the sum is computed, x. Fixed bandwidths are constant over the support of x. A variety of kernels may be specified by users. Kernels implemented for continuous data types include the second, fourth, sixth, and eighth order Gaussian and Epanechnikov kernels, and the uniform kernel. We have tried to anticipate the needs of a variety of users when designing this function. A number of semiparametric estimators and nonparametric tests in the **np** package in fact make use of npksum.

In the following example, we conduct local-constant (i.e., Nadaraya-Watson) kernel regression. We shall use cross-validated bandwidths drawn from npregbw for this example. Note that we extract the kernel sum from npksum via the '$ksum' return value in both the numerator and denominator of the resulting object. For this example, we use the data frame interface rather than the formula interface, and output from this example is plotted in Figure 2.

[5] A "partial regression plot" is simply a 2D plot of the outcome y versus one covariate x_j when all other covariates are held constant at their respective medians (this can be changed by users).

[6] plot(bw, gradients=TRUE, plot.errors.method="bootstrap", plot.errors.boot.num=25, common.scale=FALSE) will work for this example.

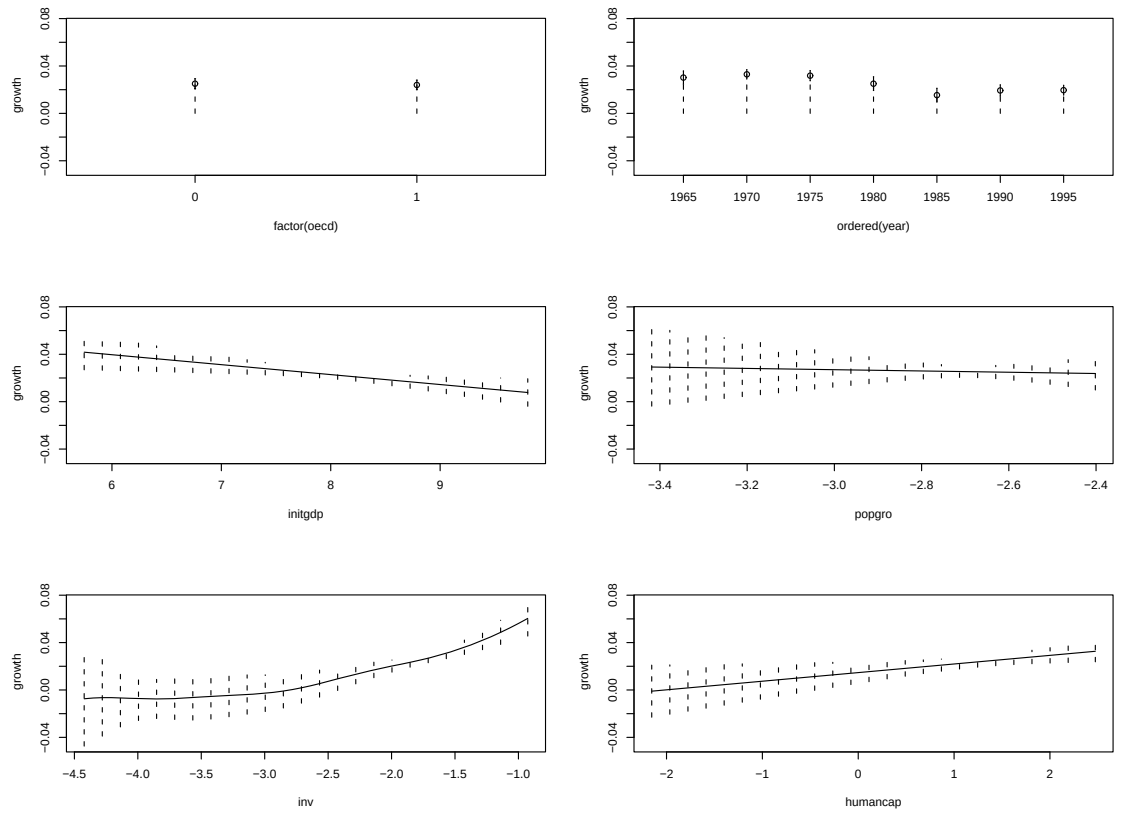

Figure 1: Partial regression plots with bootstrapped error bounds based on a cross-validated local linear kernel estimator.

```
> data("cps71")
> attach(cps71)
....
> fit.lc <- npksum(txdat=age,tydat=logwage,
+                  bws=1.892169)$ksum/
+         npksum(txdat=age,bws=1.892169)$ksum
>
```

Note that the arguments 'txdat' and 'tydat' refer to 'training' data for the regressors X and for the dependent variable y, respectively. One can also specify 'evaluation' data if you wish to evaluate the function on a set of points that differ from the training data. In such cases, you would also specify 'exdat' (and 'eydat' if so desired).

We direct the interested reader to see, by way of illustration, the example available via ?npksum that conducts leave-one-out cross-validation for a local constant regression estimator via calls to the R function nlm, and compares this to the npregbw function.

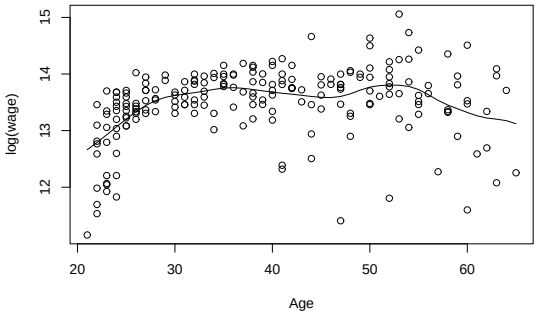

Figure 2: A local constant kernel estimator created with npksum using the Gaussian kernel (default) and a bandwidth of 1.89.

Acknowledgments

We would like to thank but not implicate Achim Zeileis and John Fox, whose many and varied suggestions led to a much improved version of the package and of this review.

Bibliography

J. Aitchison and C. G. G. Aitken. Multivariate binary discrimination by the kernel method. *Biometrika*, 63(3):413–420, 1976.

P. Hall, J. S. Racine, and Q. Li. Cross-validation and the estimation of conditional probability densities. *Journal of the American Statistical Association*, 99(468):1015–1026, 2004.

P. Hall, Q. Li, and J. S. Racine. Nonparametric estimation of regression functions in the presence of irrelevant regressors. *The Review of Economics and Statistics*, forthcoming.

C. Hsiao, Q. Li, and J. S. Racine. A consistent model specification test with mixed categorical and continuous data. *Journal of Econometrics*, 140(2):802–826, 2007.

Q. Li and J. Racine. *Nonparametric Econometrics: Theory and Practice*. Princeton University Press, 2007.

Q. Li and J. S. Racine. Nonparametric estimation of distributions with categorical and continuous data. *Journal of Multivariate Analysis*, 86:266–292, August 2003.

Q. Li and J. S. Racine. Cross-validated local linear nonparametric regression. *Statistica Sinica*, 14(2): 485–512, April 2004.

Q. Li and J. S. Racine. Nonparametric estimation of conditional CDF and quantile functions with mixed categorical and continuous data. *Journal of Business and Economic Statistics*, forthcoming.

D. Ouyang, Q. Li, and J. S. Racine. Cross-validation and the estimation of probability distributions with categorical data. *Journal of Nonparametric Statistics*, 18(1):69–100, 2006.

J. S. Racine and Q. Li. Nonparametric estimation of regression functions with both categorical and continuous data. *Journal of Econometrics*, 119(1):99–130, 2004.

J. S. Racine, Q. Li, and X. Zhu. Kernel estimation of multivariate conditional distributions. *Annals of Economics and Finance*, 5(2):211–235, 2004.

J. S. Racine, J. D. Hart, and Q. Li. Testing the significance of categorical predictor variables in nonparametric regression models. *Econometric Reviews*, 25:523–544, 2007.

J. M. Wooldridge. *Introductory Econometrics*. Thompson South-Western, 2003.

Tristen Hayfield and Jeffrey S. Racine
McMaster University
Hamilton, Ontario, Canada
hayfietj@mcmaster.ca
racinej@mcmaster.ca

eiPack: R × C Ecological Inference and Higher-Dimension Data Management

by Olivia Lau, Ryan T. Moore, and Michael Kellermann

Introduction

Ecological inference (EI) models allow researchers to infer individual-level behavior from aggregate data when individual-level data is unavailable. Table 1 shows a typical unit of ecological analysis: a contingency table with observed row and column marginals and unobserved interior cells.

	col_1	col_2	...	col_C	
row_1	N_{11i}	N_{12i}	...	N_{1Ci}	$N_{1\cdot i}$
row_2	N_{21i}	N_{22i}	...	N_{2Ci}	$N_{2\cdot i}$
...	...	...		...	...
row_R	N_{R1i}	N_{R2i}	...	N_{RCi}	$N_{R\cdot i}$
	$N_{\cdot 1i}$	$N_{\cdot 2i}$	...	$N_{\cdot Ci}$	N_i

Table 1: A typical $R \times C$ unit in ecological inference; red quantities are typically unobserved.

In ecological inference, challenges arise because information is lost when aggregating across individuals, a problem that cannot be solved by collecting more aggregate-level data. Thus, EI models are unusually sensitive to modeling assumptions. Testing these assumptions is difficult without access to individual-level data, and recent years have witnessed a lively discussion of the relative merits of various models (Wakefield, 2004).

Nevertheless, there are many applied problems in which ecological inferences are necessary, either because individual-level data is unavailable or because the aggregate-level data is considered more authoritative. The latter is true in the voting rights context in the United States, where federal courts often base decisions on evidence derived from one or more EI models (Cho and Yoon, 2001). While packages such as **MCMCpack** (Martin and Quinn, 2006) and **eco** (Imai and Lu, 2005), provide tools for 2×2 inference, this is insufficient in many applications. In **eiPack**,

we implement three existing methods for the general case in which the ecological units are $R \times C$ tables.

Methods and Data in eiPack

The methods currently implemented in **eiPack** are the method of bounds (Duncan and Davis, 1953), ecological regression (Goodman, 1953), and the Multinomial-Dirichlet model (Rosen et al., 2001).

The functions that implement these models share several attributes. The ecological tables are defined using a common formula of the form `cbind(col1, ..., colC) ~ cbind(row1, ...,rowR)`. The row and column marginals can be expressed as either proportions or counts. Auxiliary functions renormalize the results for some subset of columns taken from the original ecological table, and appropriate `print`, `summary`, and `plot` functions conveniently summarize the model output.

In the following section, we demonstrate the features of **eiPack** using the (included) `senc` dataset, which contains individual-level party affiliation data for Black, White, and Native American voters in 8 counties in southeastern North Carolina. These counties include 212 precincts, which form the ecological units in this dataset. Because the data are observed at the individual level, the interior cell counts are known, allowing us to benchmark the estimates generated by each method.

Method of Bounds

The method of bounds (Duncan and Davis, 1953) uses the observed row and column marginals to calculate upper and lower bounds for functions of the interior cells of each ecological unit. The method of bounds is not a statistical procedure in the traditional sense; the bounds implied by the row and column marginals are deterministic and there is no probabilistic model for the data-generating process.

As implemented in **eiPack**, the method of bounds allows the user to calculate for a specified column $k' \in k = \{1, \ldots, C\}$ the deterministic bounds on the proportion of individuals in each row who belong in that column. For each unit being considered, let j be the row of interest, k index columns, k' be the column of interest, k'' be the set of other columns considered, and $\tilde{k}$ be the set of columns excluded. For example, if we want the bounds on the proportion of Native American two-party registrants who are Democrats, j is Native American, k' is Democrat, k'' is Republican, and $\tilde{k}$ is No Party. The unit-level quantity of interest is

$$\frac{N_{jk'i}}{N_{jk'i} + \sum_{k \in k''} N_{jki}}$$

The lower and upper bounds on this quantity given by the observed marginals are, respectively,

$$\frac{\max(0, N_{ji} - \sum_{k \neq k'} N_{ki})}{\max(0, N_{ji} - \sum_{k \neq k'} N_{ki}) + \min(N_{ji}, \sum_{k \in k''} N_{ki})}$$

and

$$\frac{\min(N_{ji}, N_{k'i})}{\min(N_{ji}, N_{k'i}) + \max(0, N_{ji} - N_{k'i} - \sum_{k \in \tilde{k}} N_{ki})}$$

The intervals generated by the method of bounds can be analyzed in a variety of ways. Grofman (2000) suggests calculating the intersection of the unit-level bounds. If this intersection (calculated by **eiPack**) is non-empty, it represents the range of values that are consistent with the observed marginals in each of the ecological units.

Researchers and practitioners may also choose to restrict their attention to units in which one group dominates, since the bounds will typically be more informative in those units. **eiPack** allows users to set row thresholds to conduct this *extreme case analysis* (known as *homogeneous precinct analysis* in the voting context). For example, suppose the user is interested in the proportion of two-party White registrants registered as Democrats in precincts that are at least 90% White. **eiPack** calculates the desired bounds:

```
> out <- bounds(cbind(dem, rep, non) ~ cbind(black,
+   white, natam), data = senc, rows = "white",
+   column = "dem", excluded = "non",
+   threshold = 0.9, total = NULL)
```

These calculated bounds can then be represented graphically. Segments cover the range of possible values (the true value for each precinct is the red dot, not included in the standard bounds plot). In this example, the intersection of the precinct-level bounds is empty.

```
> plot(out, row = "white", column = "dem")
# add true values to plot
> idx <- as.numeric(rownames(out$bounds$white.dem))
> truth <- senc$whdem[idx]/(senc$white[idx]
+   - senc$non[idx])
> plot((1:length(idx)) / (length(idx) + 1), truth)
```

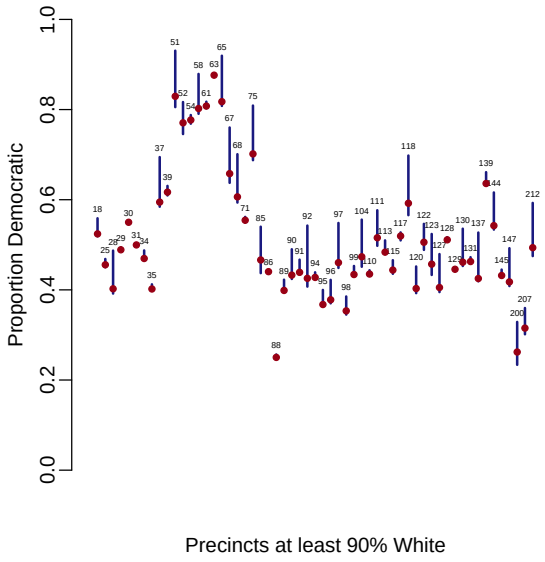

Figure 1: A plot of deterministic bounds.

Ecological Regression

In ecological regression (Goodman, 1953), observed row and column marginals are expressed as proportions and each column is regressed separately on the row proportions, thus performing C regressions. Regression coefficients then estimate the population internal cell proportions. For a given unit i, define

- X_{ri}, the proportion of individuals in row r,
- T_{ci}, the proportion of individuals in column c, and
- β_{rci}, the proportion of row r individuals in column c

The following identities hold:

$$T_{ci} = \sum_{r=1}^{R} \beta_{rci} X_{ri} \quad \text{and} \quad \sum_{c=1}^{C} \beta_{rci} = 1$$

Defining the population cell fractions β_{rc} such that $\sum_{c=1}^{C} \beta_{rc} = 1$ for every r, ecological regression assumes that $\beta_{rc} = \beta_{rci}$ for all i, and estimates the regression equations $T_{ci} = \beta_{rc} X_{ri} + \epsilon_{ci}$. Under the standard linear regression assumptions, including $E[\epsilon_{ci}] = 0$ and $Var[\epsilon_{ci}] = \sigma_c^2$ for all i, these regressions recover the population parameters β_{rc}. **eiPack** implements frequentist and Bayesian regression models (via `ei.reg` and `ei.reg.bayes`, respectively).

In the Bayesian implementation, we offer two options for the prior on β_{rc}. As a default, `truncate = FALSE` uses an uninformative flat prior that provides point estimates approaching the frequentist estimates (even when those estimates are outside the feasible range) as the number of draws $m \to \infty$. In cases where the cell estimates are near the boundaries, choosing `truncate = TRUE` imposes a uniform prior over the unit hypercube such that all cell fractions are restricted to the range $[0, 1]$.

Output from ecological regression can be summarized numerically just as in `lm`, or graphically using density plots. We also include functions to calculate estimates and standard errors of shares of a subset of columns in order to address questions such as, "What is the Democratic share of 2-party registration for each group?" For the Bayesian model, densities represent functions of the posterior draws of the β_{rc}; for the frequentist model, densities reflect functions of regression point estimates and standard errors calculated using the δ-method.

```
> out.reg <- ei.reg(cbind(dem, rep, non)
+    ~ cbind(black, white, natam), data = senc)
> lreg <- lambda.reg(out.reg,
    columns = c("dem", "rep"))
> density.plot(lreg)
```

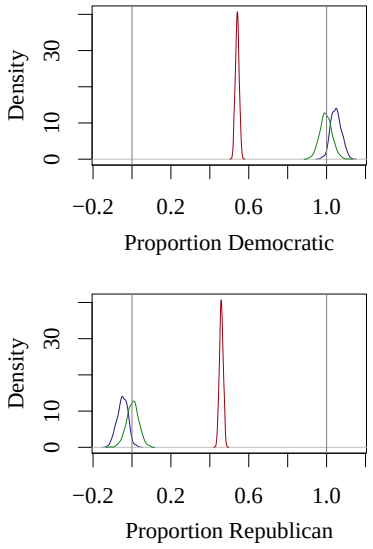

Figure 2: Density plots of ecological regression output.

Multinomial-Dirichlet (MD) model

In the Multinomial-Dirichlet model proposed by Rosen et al. (2001), the data is expressed as counts and a hierarchical Bayesian model is fit using a Metropolis-within-Gibbs algorithm implemented in `C`. Level 1 models the observed column marginals as multinomial (and independent across units); the choice of the multinomial corresponds to sampling with replacement from the population. Level 2 models the unobserved row cell fractions as Dirichlet (and independent across rows and units); Level 3 models the Dirichlet parameters as i.i.d. Gamma. More formally, without a covariate, the model is

$$
\begin{aligned}
(N_{\cdot 1i}, \ldots, N_{\cdot Ci}) &\overset{\perp\!\!\!\perp}{\sim} \text{Multinomial}(N_i, \sum_{r=1}^{R} \beta_{r1i} X_{ri}, \\
&\qquad \ldots, \sum_{r=1}^{R} \beta_{rCi} X_{ri}) \\
(\beta_{r1i}, \ldots, \beta_{rCi}) &\overset{\perp\!\!\!\perp}{\sim} \text{Dirichlet}(\alpha_{r1}, \ldots, \alpha_{rC}) \\
\alpha_{rc} &\overset{\text{i.i.d.}}{\sim} \text{Gamma}(\lambda_1, \lambda_2)
\end{aligned}
$$

With a unit-level covariate Z_i in the second level, the model becomes

$$
\begin{aligned}
(N_{\cdot 1i}, \ldots, N_{\cdot Ci}) &\overset{\perp\!\!\!\perp}{\sim} \text{Multinomial}(N_i, \sum_{r=1}^{R} \beta_{r1i} X_{ri}, \\
&\qquad \ldots, \sum_{r=1}^{R} \beta_{rCi} X_{ri}) \\
(\beta_{r1i}, \ldots, \beta_{rCi}) &\overset{\perp\!\!\!\perp}{\sim} \text{Dirichlet}(d_r e^{(\gamma_{rc} + \delta_{rc} Z_i)}, \ldots, \\
&\qquad d_r e^{(\gamma_{r(C-1)} + \delta_{r(C-1)} Z_i)}, d_r) \\
d_r &\overset{\text{i.i.d.}}{\sim} \text{Gamma}(\lambda_1, \lambda_2)
\end{aligned}
$$

In the model with a covariate, users have two options for the priors on γ_{rc} and δ_{rc} . They may assume an improper uniform prior, as was suggested by Rosen et al. (2001), or they may specify normal priors for each γ_{rc} and δ_{rc} as follows:

$$
\begin{aligned}
\gamma_{rc} &\sim \text{N}(\mu_{\gamma_{rc}}, \sigma^2_{\gamma_{rc}}) \\
\delta_{rc} &\sim \text{N}(\mu_{\delta_{rc}}, \sigma^2_{\delta_{rc}})
\end{aligned}
$$

As Wakefield (2004) notes, the weak identification that characterizes hierarchical models in the EI context is likely to make the results sensitive to the choice of prior. Users should experiment with different assumptions about the prior distribution of the upper-level parameters in order to gauge the robustness of their inferences.

The parameterization of the prior on each $(\beta_{r1i}, \ldots, \beta_{rCi})$ implies that the following log-odds ratio of expected fractions is linear with respect to the covariate Z_i:

$$
\log\left(\frac{E(\beta_{rci})}{E(\beta_{rCi})}\right) = \gamma_{rc} + \delta_{rc} Z_i
$$

Conducting an analysis using the MD model requires two steps. First, `tuneMD` calibrates the tuning parameters used for Metropolis-Hastings sampling:

```
> tune.nocov <- tuneMD(cbind(dem, rep, non)
+   ~ cbind(black, white, natam), data = senc,
+   ntunes = 10, totaldraws = 100000)
```

Second, `ei.MD.bayes` fits the model by calling `C` code to generate MCMC draws:

```
> out.nocov <- ei.MD.bayes(cbind(dem, rep, non)
+   ~ cbind(black, white, natam),
+   covariate = NULL, data = senc,
+   tune.list = tune.nocov)
```

The output of this function can be returned as `mcmc` objects or arrays; in the former case, the standard diagnostic tools in **coda** (Plummer et al., 2006) can be applied directly. The MD implementation includes `lambda` and `density.plot` functions, usage for which is analogous to ecological regression:

```
> lmd <- lambda.MD(out.nocov,
+   columns = c("dem", "rep"))
> density.plot(lmd)
```

If the precinct-level parameters are returned or saved, `cover.plot` plots the central credible intervals for each precinct. The segments represent the 95% central credible intervals and their medians for each unit (the true value for each precinct is the red dot, not included in the standard `cover.plot`).

```
> cover.plot(out.nocov, row = "white",
+   column = "dem")
# add true values to plot
> points(senc$white/senc$total,
+   senc$whdem/senc$white)
```

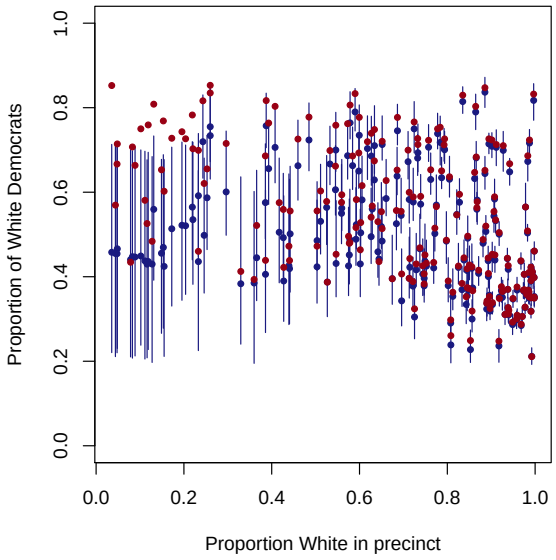

Figure 3: Coverage plot for MD model output.

Data Management

In the MD model, reasonable-sized problems produce unreasonable amounts of data. For example, a model for voting in Ohio includes 11000 precincts, 3 racial groups, and 4 parties. Implementing 1000 iterations yields about 130 million parameter draws. These draws occupy about 1GB of RAM, and this is almost certainly not enough iterations. We provide a few options to users in order to make this model tractable for large EI problems.

The unit-level parameters present the most significant data management problem. Rather than storing unit-level parameters in the workspace, users can save each chain as a `.tar.gz` file on

disk using the option `ei.MD.bayes(..., ret.beta = "s")`, or discard the unit-level draws entirely using `ei.MD.bayes(..., ret.beta = "d")`. To reconstruct the chains, users can select the row marginals, column marginals, and units of interest, without reconstructing the entire matrix of unit-level draws:

```
> read.betas(rows = c("black", "white"),
+   columns = "dem", units = 1:150,
+   dir = getwd())
```

If users are interested in some function of the unit-level parameters, the implementation of the MD model allows them to define a function in R that will be called from within the C sampling algorithm, in which case the unit-level parameters need not be saved for post-processing.

Acknowledgments

eiPack was developed with the support of the Institute for Quantitative Social Science at Harvard University. Thanks to John Fox, Gary King, Kevin Quinn, D. James Greiner, and an anonymous referee for suggestions and Matt Cox and Bob Kinney for technical advice. For further information, see `http://www.olivialau.org/software`.

Bibliography

W. T. Cho and A. H. Yoon. Strange bedfellows: Politics, courts and statistics: Statistical expert testimony in voting rights cases. *Cornell Journal of Law and Public Policy*, 10:237–264, 2001.

O. D. Duncan and B. Davis. An alternative to ecological correlation. *American Sociological Review*, 18: 665–666, 1953.

L. Goodman. Ecological regressions and the behavior of individuals. *American Sociological Review*, 18: 663–664, 1953.

B. Grofman. A primer on racial bloc voting analysis. In N. Persily, editor, *The Real Y2K Problem: Census 2000 Data and Redistricting Technology*. Brennan Center for Justice, New York, 2000.

K. Imai and Y. Lu. *eco: R Package for Fitting Bayesian Models of Ecological cvs c Inference in 2x2 Tables*, 2005. URL `http://imai.princeton.edu/research/eco.html`.

A. D. Martin and K. M. Quinn. Applied Bayesian inference in R using MCMCpack. *R News*, 6:2–7, 2006.

M. Plummer, N. Best, K. Cowles, and K. Vines. CODA: Convergence diagnostics and output analysis for MCMC. *R News*, 6:7–11, 2006.

O. Rosen, W. Jiang, G. King, and M. A. Tanner. Bayesian and frequentist inference for ecological inference: The $R \times C$ case. *Statistica Neerlandica*, 55(2):134–156, 2001.

J. Wakefield. Ecological inference for 2×2 tables (with discussion). *Journal of the Royal Statistical Society*, 167:385–445, 2004.

Olivia Lau, Ryan T. Moore, Michael Kellermann
Institute for Quantitative Social Science
Harvard University, Cambridge, MA
`olivia.lau@post.harvard.edu`
`ryantmoore@post.harvard.edu`
`kellerm@fas.harvard.edu`

The ade4 Package — II: Two-table and *K*-table Methods

by Stéphane Dray, Anne B. Dufour and Daniel Chessel

Introduction

The **ade4** package proposes a great variety of exploratory methods to analyse multivariate datasets. As suggested by the acronym **ade4** (**D**ata **A**nalysis functions to analyse **E**cological and **E**nvironmental data in the framework of **E**uclidean **E**xploratory methods), the package is devoted to ecologists but it could be useful in many other fields (e.g., Goecke, 2005). Methods available in the package are particular cases of the duality diagram (Escoufier, 1987; Holmes, 2006; Dray and Dufour, 2007) and the implementation of the functions follows the description of this unifying mathematical tool (class `dudi`). The main functions of the package for one-table analysis methods have been presented in Chessel et al. (2004). This new paper presents a short summary of two-table and *K*-table methods available in the package.

Ecological illustration

In order to illustrate the methods, we used the dataset `jv73` (Verneaux, 1973) which is available in the package. This dataset concerns 12 rivers. For

each river, a number of sites have been sampled. The number of sites per river is not constant. `jv73$poi` is a data.frame and contains presence/absence data for 19 fish species (columns) in 92 sites (rows). `jv73$fac.riv` is a factor indicating the river corresponding to each site. `jv73$morpho` contains the measurements of six environmental variables (altitude (m), distance between the site and the source (km), slope ($per\ thousand$), wetted cross section (m^2), average flow (m^3/s) and average speed (m/s)) for the same sites . Several ecological questions are related to these data:

1. Are the groups of fish species living together (i.e. species communities)?

2. Is there a relation between the composition of fish communities and the environmental variations?

3. Does the composition of fish communities vary (or not) among rivers?

4. Do the species-environment relationships vary (or not) among rivers?

Multivariate analyses help to answer these different questions: one-table methods for the first question, two-table methods for the second one and K-table methods for the last two.

Matching two tables

The main purpose of ecological data analysis is the matching of two data tables: a sites-by-environmental variables table and a sites-by-species table, to study the relationships between the composition of species communities and their environment. The **ade4** package contains the main variants of these methods (procrustean rotation, co-inertia analysis and principal component analyses with respect to instrumental variables).

The first approach is procrustean rotation (Gower, 1971), introduced in ecology by Digby and Kempton (1987, p. 116).

```
data(jv73)
pca1 <- dudi.pca(jv73$morpho, scannf = FALSE)
pca2 <- dudi.pca(jv73$poi, scale = FALSE,
    scannf = FALSE)
plot(procuste(pca1$tab, pca2$tab,
    nf = 2))
```

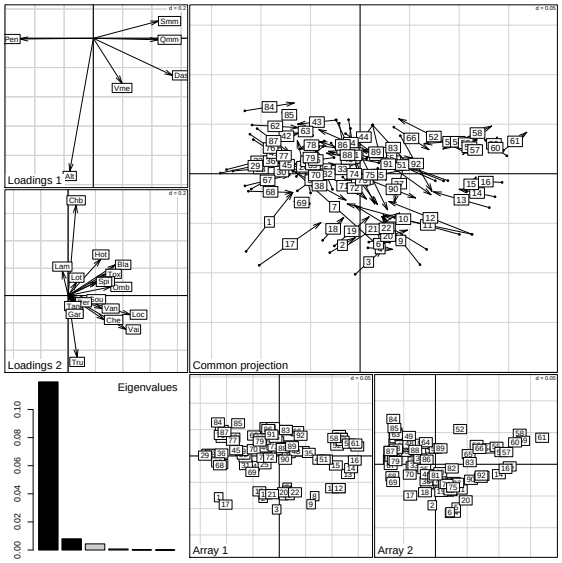

Figure 1: *Plot of a Procrustes analysis: loadings for environmental variables and species, eigenvalues screeplot, scores of sites for the two data sets, and projection of the two sets of sites after rotation (arrows link environment site score to the species site score) (Dray et al., 2003a).*

Two randomization procedures are available to test the association between two tables: PROTEST (Jackson, 1995) and RV (Heo and Gabriel, 1998).

```
plot(procuste.randtest(pca1$tab,
    pca2$tab), main = "PROTEST")

plot(RV.rtest(pca1$tab, pca2$tab),
    main = "RV")
```

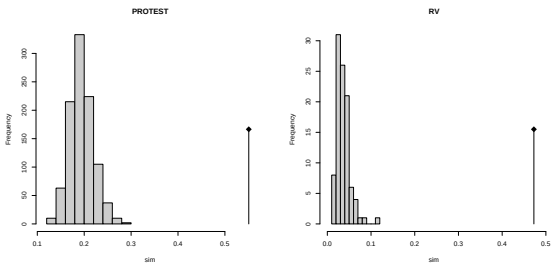

Figure 2: *Plots of PROTEST and RV tests: histograms of simulated values and observed value (vertical line).*

Co-inertia analysis (Dolédec and Chessel, 1994; Dray et al., 2003b) is a general approach that can be applied to any pair of duality diagrams having the same row weights. This method is symmetric and seeks for a common structure between two datasets. It extends psychometricians interbattery analysis (Tucker, 1958), canonical analysis on qualitative variables (Cazes, 1980), and ecological profiles analysis (Montaña and Greig-Smith, 1990; Mercier et al., 1992). Co-inertia analysis of the pair

of triplets $(\mathbf{X}_1, \mathbf{Q}_1, \mathbf{D})$ and $(\mathbf{X}_2, \mathbf{Q}_2, \mathbf{D})$ leads to the triplet $(\mathbf{X}_2^t \mathbf{D} \mathbf{X}_1, \mathbf{Q}_1, \mathbf{Q}_2)$. Note that the two triplets must have the same row weights. For a comprehensive definition of the statistical triplet of matrices **X**, **Q**, **D**, the reader could consult Chessel et al. (2004).

```
coa1 <- dudi.coa(jv73$poi, scannf = FALSE)
pca3 <- dudi.pca(jv73$morpho,
    row.w = coa1$lw, scannf = F)
plot(coinertia(coa1, pca3, scannf = FALSE))
```

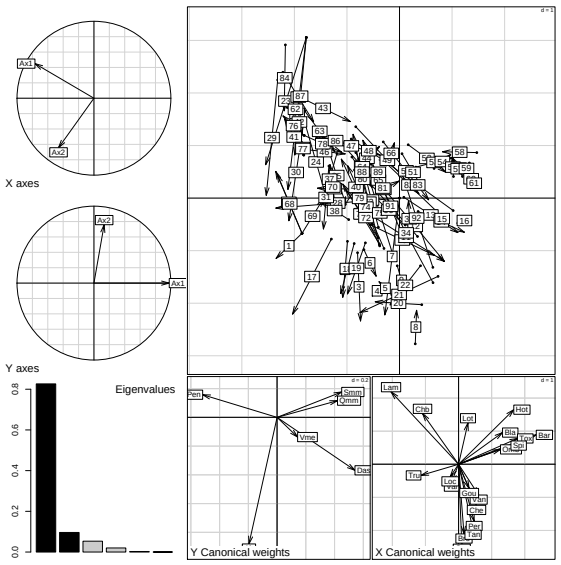

Figure 3: *Plot of a co-inertia analysis: projection of the principal axes of the two tables (species and environment) on co-inertia axes, eigenvalues screeplot, canonical weights of species and environmental variables, and joint display of the sites.*

For each coupling method, a generic `plot` function presents the various elements required to interpret the results. However, the quality of graphs could vary according to the data set. It is consequently impossible to manage relevant graphical outputs for all cases. That is why these generic `plot` use graphical functions of **ade4** which can be directly called by the user. A brief description of some of these functions is given in Table 1.

Another two-table matching strategy is principal component analyses with respect to instrumental variables (`pcaiv`, Rao, 1964). This approach consists in explaining a triplet $(\mathbf{X}_2, \mathbf{Q}_2, \mathbf{D})$ by a table of independent variables $\mathbf{X}_1$ and leads to triplet $(\mathbf{P}_{\mathbf{X}_1} \mathbf{X}_2, \mathbf{Q}_2, \mathbf{D})$ where $\mathbf{P}_{\mathbf{X}_1} = \mathbf{X}_1 (\mathbf{X}_1^t \mathbf{D} \mathbf{X}_1)^- \mathbf{X}_1^t \mathbf{D}$. This family of methods are constrained ordinations, among which redundancy analysis (van den Wollenberg, 1977) and canonical correspondence analysis (Ter Braak, 1986) are the most frequently used in ecology. Note that canonical correspondence analysis can also be performed using the `cca` wrapper function which takes two tables as arguments. The example given below is then exactly equivalent to `plot(cca(jv73$poi,jv73$morpho,scannf=FALSE))`. While the `cca` function of **ade4** is a particular case of `pcaiv`, the `cca` function of the package **vegan** is a more traditional implementation of the method which could be preferred by ecologists.

Function	Objective
`s.arrow`	cloud of points with vectors
`s.chull`	cloud of points with groups by convex hulls
`s.class`	cloud of points with groups by stars or ellipses
`s.corcircle`	correlation circle
`s.distri`	cloud of points with frequency distribution by stars and ellipses
`s.hist`	cloud of points with two marginal histograms
`s.image`	grid of gray-scale rectangles with contour lines
`s.kde2d`	cloud of points with kernel density estimation
`s.label`	cloud of points with labels
`s.logo`	cloud of points with pictures
`s.match`	matching two clouds of points with vectors
`s.traject`	cloud of points with trajectories
`s.value`	cloud of points with numerical variable

Table 1: *Objectives of some graphical functions.*

```
plot(pcaiv(coa1, jv73$morpho, scannf = FALSE))
```

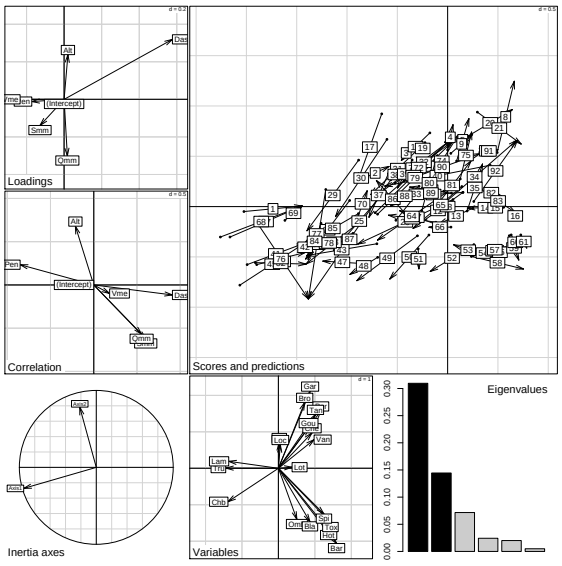

Figure 4: *Plot of a CCA seen as a particular case of PCAIV: environmental variables loadings and correlations with CCA axes, projection of principal axes on CCA axes, species scores, eigenvalues screeplot, and joint display of the rows of the two tables (position of the sites by averaging (points) and by regression (arrow tips)).*

Orthogonal analysis (`pcaivortho`) removes the effect of independent variables and corresponds to the triplet $(\mathbf{P}_{\perp \mathbf{X}_1}\mathbf{X}_2, \mathbf{Q}_2, \mathbf{D})$ where $\mathbf{P}_{\perp \mathbf{X}_1} = \mathbf{I} - \mathbf{P}_{\mathbf{X}_1}$. Between-class (`between`) and within-class (`within`) analyses (see Chessel et al., 2004, for details) are particular cases of PCAIV and orthogonal PCAIV when there is only one categorical variable (i.e. `factor`) in $\mathbf{X}_1$. Within-class analyses take into account a partition of individuals into groups and focus on structures which are common to all groups. It can be seen as a first step to *K*-table methods.

```
wit1 <- within(coa1, fac = jv73$fac.riv,
    scannf = FALSE)
plot(wit1)
```

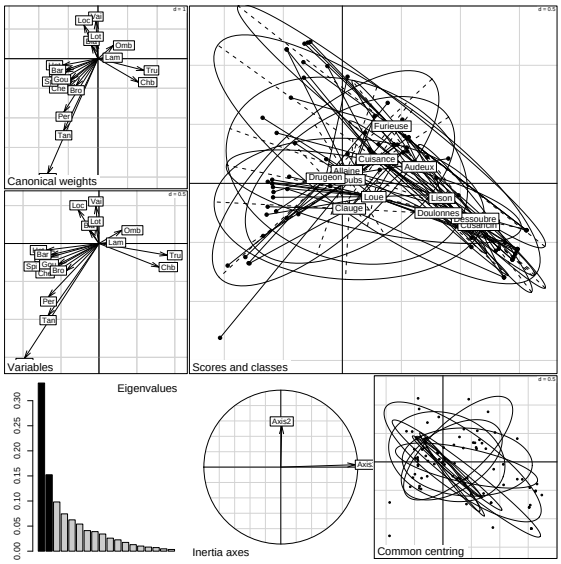

Figure 5: *Plot of a within-class analysis: species loadings, species scores, eigenvalues screeplot, projection of principal axes on within-class axes, sites scores (common centring), projections of sites and groups (i.e. rivers in this example) on within-class axes.*

The *K*-table class

Class `ktab` corresponds to collections of more than two duality diagrams, for which the internal structures are to be compared. Three formats of these collections can be considered:

- $(\mathbf{X}_1, \mathbf{Q}_1, \mathbf{D}), (\mathbf{X}_2, \mathbf{Q}_2, \mathbf{D}), \ldots, (\mathbf{X}_K, \mathbf{Q}_K, \mathbf{D})$
- $(\mathbf{X}_1, \mathbf{Q}, \mathbf{D}_1), (\mathbf{X}_2, \mathbf{Q}, \mathbf{D}_2), \ldots, (\mathbf{X}_K, \mathbf{Q}, \mathbf{D}_K)$ stored in the form of $(\mathbf{X}_1^t, \mathbf{D}_1, \mathbf{Q}), (\mathbf{X}_2^t, \mathbf{D}_2, \mathbf{Q}), \ldots, (\mathbf{X}_K^t, \mathbf{D}_K, \mathbf{Q})$
- $(\mathbf{X}_1, \mathbf{Q}, \mathbf{D}), (\mathbf{X}_2, \mathbf{Q}, \mathbf{D}), \ldots, (\mathbf{X}_K, \mathbf{Q}, \mathbf{D})$ which can also be stored in the form of $(\mathbf{X}_1^t, \mathbf{D}, \mathbf{Q}), (\mathbf{X}_2^t, \mathbf{D}, \mathbf{Q}), \ldots, (\mathbf{X}_K^t, \mathbf{D}, \mathbf{Q})$

Each statistical triplet corresponds to a separate analysis (e.g., principal component analysis, correspondence analysis ...). The common dimension of the K statistical triplets are the rows of tables which can represent individuals (samples, statistical units) or variables. Utilities for building and manipulating `ktab` objects are available. *K*-table can be constructed from a list of tables (`ktab.list.df`), a list of `dudi` objects (`ktab.list.dudi`), a within-class analysis (`ktab.within`) or by splitting a table (`ktab.data.frame`). Generic functions to transpose (`t.ktab`), combine (`c.ktab`) or extract elements (`[.ktab`) are also available. The `sepan` function can be used to compute automatically the K separate analyses.

```
kt1 <- ktab.within(wit1)
sep1 <- sepan(kt1)
kplot.sepan.coa(sep1, permute.row.col = TRUE)
```

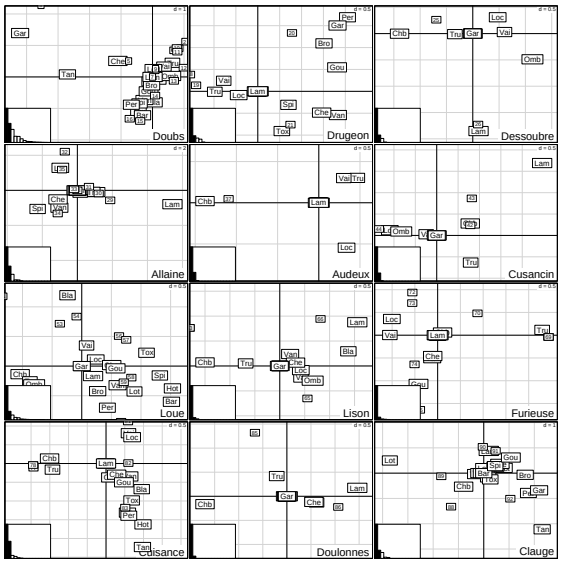

Figure 6: *Kplot of 12 separate correspondence analyses (same species, different sites).*

When the `ktab` object is built, various statistical methods can be used to analyse it. The `foucart` function can be used to analyse K tables of positive number having the same rows and the same columns and that can be analysed by a CA (Foucart, 1984; Pavoine et al., 2007). Partial triadic analysis (Tucker, 1966) is a first step toward three modes principal component analysis (Kroonenberg, 1989) and can be computed with the `pta` function. It must be used on K triplets having the same row and column weights. The `pta` function can be used to perform the STATICO method (Simier et al., 1999; Thioulouse et al., 2004). This makes it possible to analyse a pair of `ktab` objects which have been combined by the `ktab.match2ktabs` function.

Multiple factor analysis (`mfa`, Escofier and Pagès, 1994), multiple co-inertia analysis (`mcoa`, Chessel and

Hanafi, 1996) and the STATIS method (`statis`, Lavit et al., 1994) can be used to compare K triplets having the same row weights. The STATIS method can also be used to compare K triplets having the same column weights, which is a first step toward Common PCA (Flury, 1988).

```
sta1 <- statis(kt1, scannf = F)
plot(sta1)
```

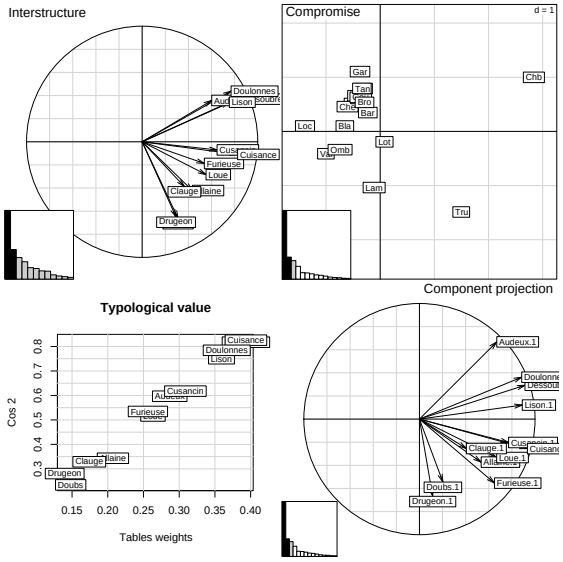

Figure 7: *Plot of STATIS analysis: interstructure, typological value of each table, compromise and projection of principal axes of separate analyses onto STATIS axes.*

The `kplot` generic function is associated to the `foucart`, `mcoa`, `mca`, `pta`, `sepan`, `sepan.coa` and `statis` methods, giving adapted collections of graphics.

```
kplot(sta1, traj = TRUE, arrow = FALSE,
    unique = TRUE, clab = 0)
```

Conclusion

The **ade4** package provides many methods to analyse multivariate ecological data sets. This diversity of tools is a methodological answer to the great variety of questions and data structures associated to biological questions. Specific methods dedicated to the analysis of biodiversity, spatial, genetic or phylogenetic data are also available in the package. The **adehabitat** brother-package contains tools to analyse habitat selection by animals while the **ade4TkGUI** package provides a graphical interface to **ade4**. More resources can be found on the **ade4** website (`http://pbil.univ-lyon1.fr/ADE-4/`).

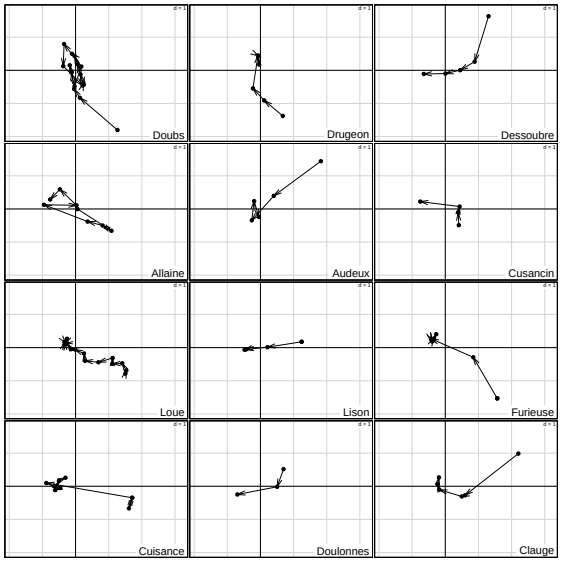

Figure 8: *Kplot of the projection of the sites of each table on the principal axes of the compromise of STATIS analysis.*

Bibliography

P. Cazes. L'analyse de certains tableaux rectangulaires décomposés en blocs : généralisation des propriétés rencontrées dans l'étude des correspondances multiples. I. Définitions et applications à l'analyse canonique des variables qualitatives. *Les Cahiers de l'Analyse des Données*, 5:145–161, 1980.

D. Chessel and M. Hanafi. Analyse de la co-inertie de K nuages de points. *Revue de Statistique Appliquée*, 44(2):35–60, 1996.

D. Chessel, A.-B. Dufour, and J. Thioulouse. The ade4 package-I- One-table methods. *R News*, 4:5–10, 2004.

P. G. N. Digby and R. A. . Kempton. *Multivariate Analysis of Ecological Communities*. Chapman and Hall, Population and Community Biology Series, London, 1987.

S. Dolédec and D. Chessel. Co-inertia analysis: an alternative method for studying species-environment relationships. *Freshwater Biology*, 31: 277–294, 1994.

S. Dray, D. Chessel, and J. Thioulouse. Procrustean co-inertia analysis for the linking of multivariate datasets. *Ecoscience*, 10:110–119, 2003a.

S. Dray, D. Chessel, and J. Thioulouse. Co-inertia analysis and the linking of ecological tables. *Ecology*, 84(11):3078–3089, 2003b.

S. Dray and A. Dufour. The ade4 package: implementing the duality diagram for ecologists. *Journal of Statistical Software*, 22(4):1–20, 2007.

B. Escofier and J. Pagès. Multiple factor analysis (AFMULT package). *Computational Statistics and Data Analysis*, 18:121–140, 1994.

Y. Escoufier. The duality diagram : a means of better practical applications. In P. Legendre and L. Legendre, editors, *Development in numerical ecology*, pages 139–156. NATO advanced Institute , Serie G .Springer Verlag, Berlin, 1987.

B. Flury. *Common Principal Components and Related Multivariate. models*. Wiley and Sons, New-York, 1988.

T. Foucart. *Analyse factorielle de tableaux multiples*. Masson, Paris, 1984.

R. Goecke. 3D lip tracking and co-inertia analysis for improved robustness of audio-video automatic speech recognition. In *Proceedings of the Auditory-Visual Speech Processing Workshop AVSP 2005*, pages 109–114, 2005.

J. Gower. Statistical methods of comparing different multivariate analyses of the same data. In F. Hodson, D. Kendall, and P. Tautu, editors, *Mathematics in the archaeological and historical sciences*, pages 138–149. University Press, Edinburgh, 1971.

M. Heo and K. Gabriel. A permutation test of association between configurations by means of the RV coefficient. *Communications in Statistics - Simulation and Computation*, 27:843–856, 1998.

S. Holmes. Multivariate analysis: The French way. In N. D. and S. T., editors, *Festschrift for David Freedman*. IMS, Beachwood, OH, 2006.

D. Jackson. PROTEST: a PROcustean randomization TEST of community environment concordance. *Ecosciences*, 2:297–303, 1995.

P. Kroonenberg. The analysis of multiple tables in factorial ecology. iii three-mode principal component analysis:"analyse triadique complète". *Acta OEcologica, OEcologia Generalis*, 10:245–256, 1989.

C. Lavit, Y. Escoufier, R. Sabatier, and P. Traissac. The ACT (STATIS method). *Computational Statistics and Data Analysis*, 18:97–119, 1994.

P. Mercier, D. Chessel, and S. Dolédec. Complete correspondence analysis of an ecological profile data table: a central ordination method. *Acta OEcologica*, 13:25–44, 1992.

C. Montaña and P. Greig-Smith. Correspondence analysis of species by environmental variable matrices. *Journal of Vegetation Science*, 1:453–460, 1990.

S. Pavoine, J. Blondel, M. Baguette, and D. Chessel. A new technique for ordering asymmetrical three-dimensional data sets in ecology. *Ecology*, 88:512–523, 2007.

C. Rao. The use and interpretation of principal component analysis in applied research. *Sankhya A*, 26: 329–359, 1964.

M. Simier, L. Blanc, F. Pellegrin, and D. Nandris. Approche simultanée de K couples de tableaux: application à l'étude des relations pathologie végétale-environment. *Revue de Statistique Appliquée*, 47:31–46, 1999.

C. Ter Braak. Canonical correspondence analysis : a new eigenvector technique for multivariate direct gradient analysis. *Ecology*, 67:1167–1179, 1986.

J. Thioulouse, M. Simier, and D. Chessel. Simultaneous analysis of a sequence of pairs of ecological tables with the STATICO method. *Ecology*, 85:272–283, 2004.

L. . Tucker. An inter-battery method of factor analysis. *Psychometrika*, 23:111–136, 1958.

L. Tucker. Some mathemetical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

A. van den Wollenberg. Redundancy analysis, an alternative for canonical analysis. *Psychometrika*, 42 (2):207–219, 1977.

J. Verneaux. *Cours d'eau de Franche-Comté (Massif du Jura). Recherches écologiques sur le réseau hydrographique du Doubs. Essai de biotypologie*. Thèse de doctorat, Université de Besançon, Besançon, 1973.

Stéphane Dray, Anne-Béatrice Dufour, Daniel Chessel
Laboratoire de Biométrie et
Biologie Evolutive (UMR 5558) ; CNRS
Université de Lyon ; université Lyon 1
43, Boulevard du 11 Novembre 1918
69622 Villeurbanne Cedex, France
dray@biomserv.univ-lyon1.fr
dufour@biomserv.univ-lyon1.fr
chessel@biomserv.univ-lyon1.fr

Review of "The R Book"

Michael J. Crawley, Wiley, 2007

by Friedrich Leisch

The back cover of this physically impressive 1000-page volume advertises it as *"...the first comprehemsive reference manual for the R language..."* which *"...introduces all the statistical models covered by R..."*. Considering (a) that the R Core Team considers its own language manual (R Development Core Team, 2007b) a draft, and only the source code the ultimate reference in more cases than we like, and (b) the multitude of models implemented by R packages on CRAN or Bioconductor, I thought I would be in for an interesting read.

The book has 27 chapters. Chapters 1–8 give an introduction to R, starting where to obtain and how to install the software, describing the language, data input and data manipulation, graphics, tables, mathematical calculations, and classical statistical tests. Chapters 9–20 on statistical modelling form the main part of the book with a detailed coverage of the linear regression model and its extensions like GLMs, GAMs, mixed effects and non-linear least squares. Chapters 20–27 show "other" topics like trees, time series, multivariate and spatial statistics, survival analysis, using R for simulation models and low-level graphics commands.

The preface states that the book is *"aimed at beginners and intermediate users"* and can be used *"as a text ... as well as a reference manual"*. I find that the book in its present form is not optimal for either purpose. The first section on "getting started" has a few minor problems, like using many functions without quoted character arguments. In some cases this is a matter of style (like `library(foo)` or `help(foo)`), but some instances simply do not work (`find(foo)`, `apropos(foo)`). Packages are often called libraries (which is a different thing), input lines can be longer than 128 characters (the current limit is 8192), and recommending MS Word as a source code editor is at least debatable even for Windows users. I personally find the R code throughout the book hard to read: it is typeset in a proportional font, uses no spaces around the assignment operator `<-`, no line indentation for nested code blocks, and path names sometimes contain erroneous spaces, especially after backslashes.

The chapter on "essentials of the R language" gives an introduction to the language and many non-statistical functions like string processing and regular expressions. What I found very confusing is the lack of clear structure. The book uses only one level of numbering (chapters), and this chapter is 100 pages long. E.g., on page 47 there are two headings: "The match function" and "Writing functions in R". Both seem to have the same font size and hence are on equal level. However, as is to be expected given the two topics, the section on the match function is 2 paragraphs long, how to write functions takes the next 20 pages, with many intermezzos and headings in two different sizes. The author also jumps around a lot, many concepts are discussed or introduced as a side note for a different theme, and it is often unclear where examples end. E.g., how formal and actual arguments are matched in a funcion call is the first paragraph in the section on "Saving data to disc". All of this will be confusing for beginners and makes it hard to use the book as a reference manual.

In the chapter on mathematics a dozen pages is used on introducing the OLS estimate (typo in several equations: $\hat{\beta} = X'X - 1X'y$), including a stepwise implementation of (the correct version of) this formula. Although the next page in the book starts with solving linear equations via `solve()`, it is not even mentioned that it is numerically not the best idea to compute regression coefficients using the formula above.

The quality of the book increases considerably in the chapters on statistical modelling. A minor drawback is that it sometimes gives the impression that linear models are the only ones available, even discriminant analysis is not considered a model, because response variables cannot be multi-level categorical according to the cookbook recipe on page 324. However, there is a nice general introduction to statistical modelling and model selection, and linear modelling is covered in depth with many examples.

Once the author leaves the territory of linear models (and their extensions), quality decreases again. The chapter on trees uses package **tree**, although even the author of **tree** recommends using package **rpart** (Venables and Ripley, 2002, p. 266). The chapter on multivariate statistics basically recommends not doing multivariate statistics at all, because one is too likely to shoot oneself into the foot.

In summary, the book fails to meet the high expectations that the title and cover texts raise. In this review I could list only a selection of problems I found, and of course there are good things too, like the detailed explanation on how to enter data into a spreadsheet to form a proper data frame. The book is definitely not a reference manual for the R system or R language, but a book on applied linear modelling with (many pages of) lower-quality additional material to give the impression of universal coverage. There are better introductionary books for beginners, and Venables and Ripley (2002) is still "the R book" when it comes to a reference text for applied statistics.

A (symptomatic) end note: The author gives detailed credit to the R core team and wider R com-

munity in the acknowledgements (thanks!). On page one he recommends the `citation()` function to users to give credit to developers (yes!), however he seems not to have used the function too often, because R Development Core Team (2007a,b) and many others are missing from the references, which cover only 4 of 1000 pages.

Bibliography

R Development Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2007a. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

R Development Core Team. *R Language Definition*. R Foundation for Statistical Computing, Vienna, Austria, 2007b. URL `http://www.R-project.org`. ISBN 3-900051-13-5.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S. Fourth Edition*. Springer, 2002. URL `http://www.stats.ox.ac.uk/pub/MASS4/`. ISBN 0-387-95457-0.

Friedrich Leisch
Ludwig-Maximilians-Universität München, Germany
`Friedrich.Leisch@R-project.org`

Changes in R 2.6.0

by the R Core Team

User-visible changes

- `integrate()`, `nlm()`, `nlminb()`, `optim()`, `optimize()` and `uniroot()` now have `...` much earlier in their argument list. This reduces the chances of unintentional partial matching but means that the later arguments must be named in full.
- The default type for `nchar()` is now `"chars"`. This is almost always what was intended, and differs from the previous default only for non-ASCII strings in a MBCS locale. There is a new argument `allowNA`, and the default behaviour is now to throw an error on an invalid multibyte string if `type = "chars"` or `type = "width"`.
- Connections will be closed if there is no R object referring to them. A warning is issued if this is done, either at garbage collection or if all the connection slots are in use.

New features

- `abs()`, `sign()`, `sqrt()`, `floor()`, `ceiling()`, `exp()` and the gamma, trig and hyperbolic trig functions now only accept one argument even when dispatching to a `Math` group method (which may accept more than one argument for other group members).
- `abbreviate()` gains a `method` argument with a new option `"both.sides"` which can make shorter abbreviations.
- `aggregate.data.frame()` no longer changes the group variables into factors, and leaves alone the levels of those which are factors. (Inter alia grants the wish of PR#9666.)
- The default `max.names` in `all.names()` and `all.vars()` is now `-1` which means unlimited. This fixes PR#9873.
- `as.vector()` and the default methods of `as.character()`, `as.complex()`, `as.double()`, `as.expression()`, `as.integer()`, `as.logical()` and `as.raw()` no longer duplicate in most cases where the object is unchanged. (Beware: some code has been written that invalidly assumes that they do duplicate, often when using `.C/.Fortran(DUP = FALSE)`.)
- `as.complex()`, `as.double()`, `as.integer()`, `as.logical()` and `as.raw()` are now primitive and internally generic for efficiency. They no longer dispatch on S3 methods for `as.vector()` (which was never documented). `as.real()` and `as.numeric()` remain as alternative names for `as.double()`.

  `expm1()`, `log()`, `log1p()`, `log2()`, `log10()`, `gamma()`, `lgamma()`, `digamma()` and `trigamma()` are now primitive. (Note that `logb()` is not.)

  The `Math2` and `Summary` groups (`round`, `signif`, `all`, `any`, `max`, `min`, `sum`, `prod`, `range`) are now primitive.

  See under Section "**methods** Package" below for some consequences for S4 methods.
- `apropos()` now sorts by name and not by position on the search path.

- `attr()` gains an `exact = TRUE` argument to disable partial matching.
- `bxp()` now allows `xlim` to be specified. (PR#9754)
- `C(f, SAS)` now works in the same way as `C(f, treatment)`, etc.
- `chol()` is now generic.
- `dev2bitmap()` has a new option to go via PDF and so allow semi-transparent colours to be used.
- `dev.interactive()` regards devices with the displaylist enabled as interactive, and packages can register the names of their devices as interactive via `deviceIsInteractive()`.
- `download.packages()` and `available.packages()` (and functions which use them) now support in `repos` or `contriburl` either '`file:`' plus a general path (including drives on a UNC path on Windows) or a '`file:///`' URL in the same way as `url()`.
- `dQuote()` and `sQuote()` are more flexible, with rendering controlled by the new option `useFancyQuotes`. This includes the ability to have TEX-style rendering and directional quotes (the so-called "smart quotes") on Windows. The default is to use directional quotes in UTF-8 locales (as before) and in the `Rgui` console on Windows (new).
- `duplicated()` and `unique()` and their methods in **base** gain an additional argument `fromLast`.
- `fifo()` no longer has a default `description` argument.

  `fifo("")` is now implemented, and works in the same way as `file("")`.
- `file.edit()` and `file.show()` now tilde-expand file paths on all interfaces (they used to on some and not others).
- The `find()` argument is now named `numeric` and not `numeric.`: the latter was needed to avoid warnings about name clashes many years ago, but partial matching was used.
- `stats:::.getXlevels()` confines attention to factors since some users expected R to treat `unclass(`*a_factor*`)` as a numeric vector.
- `grep()`, `strsplit()` and friends now warn if incompatible sets of options are used, instead of silently using the documented priority.
- `gsub()/sub()` with `perl = TRUE` now preserves attributes from the argument `x` on the result.
- `is.finite()` and `is.infinite()` are now S3 and S4 generic.
- `jpeg()`, `png()`, `bmp()` (Windows), `dev2bitmap()` and `bitmap()` have a new argument `units` to specify the units of `width` and `height`.
- `levels()` is now generic (`levels<-()` has been for a long time).
- Loading serialized raw objects with `load()` is now considerably faster.
- New primitive `nzchar()` as a faster alternative to `nchar(x) > 0` (and avoids having to convert to wide chars in a MBCS locale and hence consider validity).
- The way `old.packages()` and hence `update.packages()` handle packages with different versions in multiple package repositories has been changed. The first package encountered was selected, now the one with highest version number.
- `optim(method = "L-BFGS-B")` now accepts zero-length parameters, like the other methods. Also, `method = "SANN"` no longer attempts to optimize in this case.
- New options `showWarnCalls` and `showErrorCalls` to give a concise traceback on warnings and errors. `showErrorCalls = TRUE` is the default for non-interactive sessions. Option `showNCalls` controls how abbreviated the call sequence is.
- New options `warnPartialMatchDollar`, `warnPartialMatchArgs` and `warnPartialMatchAttr` to help detect the unintended use of partial matching in `$`, argument matching and `attr()` respectively.
- A device named as a character string in `options(device =)` is now looked for in the **grDevices** name space if it is not visible from the global environment.
- `pmatch(x, y, duplicates.ok = TRUE)` now uses hashing and so is much faster for large `x` and `y` when most matches are exact.
- `qr()` is now generic.
- It is now a warning to have an non-integer object for `.Random.seed`: this indicates a user had been playing with it, and it has always been documented that users should only save and restore it.
- New higher-order functions `Reduce()`, `Filter()` and `Map()`.

- `regexpr()` and `gregexpr()` gain an `ignore.case` argument for consistency with `grep()`. (This does change the positional matching of arguments, but no instances of positional matching beyond the second were found.)
- `relist()` utility, an S3 generic with several methods, providing an `inverse` for `unlist()`; thanks to a code proposal from Andrew Clausen.
- `require()` now returns invisibly.
- The interface to `reshape()` has been revised, allowing some simplified forms that did not work before, and somewhat improved error handling. A new argument `sep` has been introduced to replace simple usages of `split` (the old features are retained).
- `rmultinom()` uses a high-precision accumulator where available, and so is more likely to give the same result on different platforms (although it is still possible to get different results, and the result may differ from previous versions of R).
- `row()` and `col()` now work on matrix-like objects such as data frames, not just matrices.
- `Rprof()` allows smaller values of `interval` on machines that support it: for example modern Linux systems support `interval = 0.001`.
- `sample()` now requires its first argument `x` to be numeric (in the sense of `is.numeric()`) as well as of length 1 and ≥ 1 before it is regarded as shorthand for `1:x`.
- `sessionInfo()` now provides details about package name spaces that are loaded but not attached. The output of `sessionInfo()` has been improved to make it easier to read when it is inadvertently wrapped after being pasted into an email message.
- `setRepositories()` has a new argument `ind` to allow selections to be made programmatically.
- `showMethods()` has a "smart" default for `inherited` such that `showMethods(`*genfun*`, incl = TRUE)` becomes a useful short cut.
- `sprintf()` no longer has a output string length limit.
- `storage.mode<-()` is now primitive, and hence makes fewer copies of an object (none if the mode is unchanged). It is a little less general than `mode<-()`, which remains available. (See also the entry under Deprecated & defunct below.)
- `sweep()` gains an argument `check.margin = TRUE` which warns about mismatched dimensions.
- The mathematical annotation facility (`plotmath()`) now recognises a `symbol()` function which forces the font to be a symbol font. This allows access to all characters in the Adobe Symbol encoding within `plotmath` expressions.
- For OSes that cannot unset environment variables, `Sys.unsetenv()` sets the value to `""`, with a warning.
- New function `Sys.which()`, an interface to `which` on Unix-alikes and an emulation on Windows.
- On Unix-alikes, `system(, intern = TRUE)` reports on very long lines that may be truncated, giving the line number of the content being read.
- `termplot()` has a default for `ask` that uses `dev.interactive()`.

  It allows `ylim` to be set, or computed to cover all the plots to be made (the new default) or computed for each plot (the previous default).
- `uniroot(f, *)` is slightly faster for non-trivial `f()` because it computes `f(lower)` and `f(upper)` only once, and it has new optional arguments `f.lower` and `f.upper` by which the caller can pass these.
- `unlink()` is now internal, using common POSIX code on all platforms.
- `unsplit()` now works with lists of dataframes.
- The `vcov()` methods for classes `"gls"` and `"nlme"` have migrated to package **nlme**.
- `vignette()` has a new argument `all` to choose between showing vignettes in attached packages or in all installed packages.
- New function `within()`, which is like `with()`, except that it returns modified versions back of lists and data frames.
- `X11()`, `postscript()` (and hence `bitmap()`), `xfig()`, `jpeg()`, `png()` and the Windows devices `win.print()`, `win.metafile()` and `bmp()` now warn (once at first use) if semi-transparent colours are used (rather than silently treating them as fully transparent).
- New function `xspline()` to provide `base` graphics support of X-splines (cf. `grid.xspline()`).
- New function `xyTable()` does the 2D gridding "computations" used by `sunflowerplot()`.

- Rd conversion to HTML and CHM now makes use of classes, which are set in the stylesheets. Editing 'R.css' will change the styles used for `\env`, `\option`, `\pkg` etc. (CHM styles are set at compilation time.)

- The documented arguments of `%*%` have been changed to be `x` and `y`, to match S and the implicit S4 generic.

- If members of the Ops group (the arithmetic, logical and comparison operators) and `%*%` are called as functions, e.g., `` `>`(x, y) ``, positional matching is always used. (It used to be the case that positional matching was used for the default methods, but names would be matched for S3 and S4 methods and in the case of `!` the argument name differed between S3 and S4 methods.)

- Imports environments of name spaces are named (as `"imports:foo"`), and so are known e.g. to `environmentName()`.

- Package **stats4** uses lazy-loading not `SaveImage` (which is now deprecated).

- Installing help for a package now parses the '.Rd' file only once, rather than once for each type.

- PCRE has been updated to version 7.2.

- `bzip2` has been updated to version 1.0.4.

- `gettext` has been updated to version 0.16.1.

- There is now a global `CHARSXP` cache, `R_StringHash`. `CHARSXP`s are no longer duplicated and must not be modified in place. Developers should strive to only use `mkChar` (and `mkString`) for creating new `CHARSXP`s and avoid use of `allocString`. A new macro, `CallocCharBuf`, can be used to obtain a temporary char buffer for manipulating character data. This patch was written by Seth Falcon.

- The internal equivalents of `as.complex()`, `as.double()`, `as.integer()` and `as.logical()` used to handle `length - 1` arguments now accept character strings (rather than report that this is "unimplemented").

- Lazy-loading a package is now substantially more efficient (in memory saved and load time).

- Various performance improvements lead to a 45% reduction in the startup time without **methods** (and one-sixth with – **methods** now takes 75% of the startup time of a default session).

- The `[[` subsetting operator now has an argument `exact` that allows programmers to disable partial matching (which will in due course become the default). The default value is `exact = NA` which causes a warning to be issued when partial matching occurs. When `exact = TRUE`, no partial matching will be performed. When `exact = FALSE`, partial matching can occur and no warning will be issued. This patch was written by Seth Falcon.

- Many of the C-level warning/error messages (e.g., from subscripting) have been re-worked to give more detailed information on either the location or the cause of the problem.

- The S3 and S4 `Math` groups have been harmonized. Functions `log1p()`, `expm1()`, `log10()` and `log2()` are members of the S3 group, and `sign()`, `log1p()`, `expm1()`, `log2()`, `cummax()`, `cummin()`, `digamma()`, `trigamma()` and `trunk()` are members of the S4 group. `gammaCody()` is no longer in the S3 group. They are now all primitive.

- The initialization of the random-number stream makes use of the sub-second part of the current time where available.

  Initialization of the 1997 Knuth TAOCP generator is now done in R code, avoiding some C code whose licence status has been questioned.

- The reporting of syntax errors has been made more user-friendly.

methods Package

- Packages using **methods** have to have been installed in R 2.4.0 or later (when various internal representations were changed).

- Internally generic primitives no longer dispatch S4 methods on S3 objects.

- `load()` and restoring a workspace attempt to detect and warn on the loading of pre-2.4.0 S4 objects.

- Making functions primitive changes the semantics of S4 dispatch: these no longer dispatch on classes based on types but do dispatch whenever the function in the base name space is called.

  This applies to `as.complex()`, `as.integer()`, `as.logical()`, `as.numeric()`, `as.raw()`, `expm1()`, `log()`, `log1p()`, `log2()`, `log10()`, `gamma()`, `lgamma()`, `digamma()` and `trigamma()`, as well as the `Math2` and `Summary` groups.

Because all members of the group generics are now primitive, they are all S4 generic and setting an S4 group generic does at last apply to all members and not just those already made S4 generic.

`as.double()` and `as.real()` are identical to `as.numeric()`, and now remain so even if S4 methods are set on any of them. Since `as.numeric` is the traditional name used in S4, currently methods must be exported from a 'NAMESPACE' for `as.numeric` only.

- The S4 generic for `!` has been changed to have signature `(x)` (was `(e1)`) to match the documentation and the S3 generic. `setMethod()` will fix up methods defined for `(e1)`, with a warning.
- The `"structure"` S4 class now has methods that implement the concept of structures as described in the Blue Book—that element-by-element functions and operators leave structure intact unless they change the length. The informal behavior of R for vectors with attributes was inconsistent.
- The `implicitGeneric()` function and relatives have been added to specify how a function in a package should look when methods are defined for it. This will be used to ensure that generic versions of functions in R core are consistent. See `?implicitGeneric`.
- Error messages generated by some of the functions in the methods package provide the name of the generic to provide more contextual information.
- It is now possible to use `setGeneric(useAsDefault = FALSE)` to define a new generic with the name of a primitive function (but having no connection with the primitive).

Deprecated & defunct

- `$` on an atomic vector now gives a warning that it is "invalid". It remains deprecated, but may be removed in R $\geq$ 2.7.0.
- `storage.mode(x) <- "real"` and `storage.mode(x) <- "single"` are defunct: use instead `storage.mode(x) <- "double"` and `mode(x) <- "single"`.
- In package installation, 'SaveImage: yes' is deprecated in favour of 'LazyLoad: yes'.
- `seemsS4Object` (**methods** package) is deprecated in favour of `isS4()`.
- It is planned that `[[exact = TRUE]]` will become the default in R 2.7.0.

Utilities

- `checkS3methods()` (invoked by `R CMD check`) now checks the arguments of methods for primitive members of the S3 group generics.
- `R CMD check` now does a recursive copy on the 'tests' directory.
- `R CMD check` now warns on non-ASCII '.Rd' files without an `\encoding` field, rather than just on ones that are definitely not from an ISO-8859 encoding. This agrees with the long-standing stipulation in "Writing R Extensions", and catches some packages with UTF-8 man pages.
- `R CMD check` now warns on DESCRIPTION files with a non-portable `Encoding` field, or with non-ASCII data and no `Encoding` field.
- `R CMD check` now loads all the `Suggests` and `Enhances` dependencies to reduce warnings about non-visible objects, and also emulates standard functions (such as `shell()`) on alternative R platforms.
- `R CMD check` now (by default) attempts to latex the vignettes rather than just weave and tangle them: this will give a NOTE if there are latex errors.
- `R CMD check` computations no longer ignore Rd `\usage` entries for functions for extracting or replacing parts of an object, so S3 methods should use the appropriate `\method{}` markup.
- `R CMD check` now checks for CR (as well as CRLF) line endings in C/C++/Fortran source files, and for non-LF line endings in 'Makefile[.in]' and 'Makevars[.in]' in the package 'src' directory. `R CMD build` will correct non-LF line endings in source files and in the make files mentioned.
- `Rdconv` now warns about unmatched braces rather than silently omitting sections containing them. (Suggestion by Bill Dunlap, PR#9649)

  `Rdconv` now renders (rather than ignores) `\var{}` inside `\code{}` markup in LaTeXconversion.

  `R CMD Rdconv` gains a '`--encoding`' argument to set the default encoding for conversions.
- The list of CRAN mirrors now has a new (manually maintained) column `"OK"` which flags mirrors that seem to be OK, only those are used

by `chooseCRANmirror()`. The now exported function `getCRANmirrors()` can be used to get all known mirrors or only the ones that are OK.

- `R CMD SHLIB` gains arguments '`--clean`' and '`--preclean`' to clean up intermediate files after and before building.
- `R CMD config` now knows about FC and FCFLAGS (used for F9x compilation).
- `R CMD Rdconv` now does a better job of rendering quotes in titles in HTML, and `\sQuote` and `\dQuote` into text on Windows.

C-level facilities

- New utility function `alloc3DArray` similar to `allocMatrix`.
- The entry point `R_seemsS4Object` in 'Rinternals.h' has not been needed since R 2.4.0 and has been removed. Use `IS_S4_OBJECT` instead.
- Applications embedding R can use `R_getEmbeddingDllInfo()` to obtain `DllInfo` for registering symbols present in the application itself.
- The instructions for making and using standalone `libRmath` have been moved to the R Installation and Administration manual.
- `CHAR()` now returns `(const char *)` since `CHARSXP`s should no longer be modified in place. This change allows compilers to warn or error about improper modification. Thanks to Herve Pages for the suggestion.
- `acopy_string` is a (provisional) new helper function that copies character data and returns a pointer to memory allocated using `R_alloc`. This can be used to create a copy of a string stored in a `CHARSXP` before passing the data on to a function that modifies its arguments.
- `asLogical`, `asInteger`, `asReal` and `asComplex` now accept `STRSXP` and `CHARSXP` arguments, and `asChar` accepts `CHARSXP`.
- New `R_GE_str2col()` exported via 'R_ext/GraphicsEngine.h' for external device developers.
- `doKeybd` and `doMouseevent` are now exported in 'GraphicsDevice.h'.
- `R_alloc` now has first argument of type `size_t` to support 64-bit platforms (e.g., Win64) with a 32-bit `long` type.
- The type of the last two arguments of `getMatrixDimnames` (non-API but mentioned in 'R-exts.texi' and in 'Rinternals.h') has been changed to `const char **` (from `char **`).
- `R_FINITE` now always resolves to the function call `R_finite` in packages (rather than sometimes substituting isfinite). This avoids some issues where R headers are called from C++ code using features tested on the C compiler.
- The advice to include R headers from C++ inside `extern "C"` has been changed. It is nowadays better *not* to wrap the headers, as they include other headers which on some OSes should not be wrapped.
- 'Rinternals.h' no longer includes a substantial set of C headers. All but 'ctype.h' and 'errno.h' are included by 'R.h' which is supposed to be used before 'Rinternals.h'.
- Including C system headers can be avoided by defining `NO_C_HEADERS` before including R headers. This is intended to be used from C++ code, and you will need to include C++ equivalents such as `<cmath>` before the R headers.

Installation

- The `test-Lapack` test is now part of `make check`.
- The `stat` system call is now required, along with `opendir` (which had long been used but not tested for). (`make check` would have failed in earlier versions without these calls.)
- `evince` is now considered as a possible PDF viewer.
- `make install-strip` now also strips the DLLs in the standard packages.
- Perl 5.8.0 (released in July 2002) or later is now required. (R 2.4.0 and later have in fact required 5.6.1 or later.)
- The C function `finite` is no longer used: we expect a C99 compiler which will have `isfinite`. (If that is missing, we test separately for `NaN`, `Inf` and `-Inf`.)
- A script/executable `texi2dvi` is now required on Unix-alikes: it is part of the texinfo distribution.
- Files 'texinfo.tex' and 'txi-en.tex' are no longer supplied in doc/manual (as the latest versions have an incompatible licence). You will need to ensure that your texinfo and/or TeX installations supply them.

- `wcstod` is now required for MBCS support.
- There are some experimental provisions for building on Cygwin.

Package Installation

- The encoding declared in the 'DESCRIPTION' file is now used as the default encoding for '.Rd' files.
- A standard for specifying package license information in the 'DESCRIPTION' `License` field was introduced, see "Writing R Extensions". In addition, files 'LICENSE' or 'LICENCE' in a package top-level source directory are now installed (so putting copies into the 'inst' subdirectory is no longer necessary).
- `install.packages()` on a Unix-alike now updates 'doc/html/packages.html' only if packages are installed to '.Library' (by that exact name).
- `R CMD INSTALL` with option '`--clean`' now runs `R CMD SHLIB` with option '`--clean`' to do the clean up (unless there is a 'src/Makefile'), and this will remove `$(OBJECTS)` (which might have been redefined in 'Makevars').

  `R CMD INSTALL` with '`--preclean`' cleans up the sources after a previous installation (as if that had used '`--clean`') before attempting to install.

  `R CMD INSTALL` will now run `R CMD SHLIB` in the 'src' directory if 'src/Makevars' is present, even if there are no source files with known extensions.
- If there is a file 'src/Makefile', 'src/Makevars' is now ignored (it could be included by 'src/Makefile' if desired), and it is preceded by 'etc/Makeconf' rather than 'R_HOME/share/make/shlib.mk'. Thus the makefiles read are 'R_HOME/etc/Makeconf', 'src/Makefile' in the package and then any personal 'Makevars' files.
- `R CMD SHLIB` used to support the use of `OBJS` in 'Makevars', but this was changed to `OBJECTS` in 2001. The undocumented alternative of `OBJS` has finally been removed.
- `R CMD check` no longer issues a warning about no data sets being present if a lazyload db is found (as determined by the presence of 'Rdata.rdb', 'Rdata.rds', and 'Rdata.rdx' in the 'data' subdirectory).

Bug fixes

- `charmatch()` and `pmatch()` used to accept non-integer values for `nomatch` even though the return value was documented to be integer. Now `nomatch` is coerced to integer (rather than the result being coerced to the type of `nomatch`).
- `match.call()` no longer "works" outside a function unless `definition` is supplied. (Under some circumstances it used to "work", matching itself.)
- The formula methods of boxplot, cdplot, pairs and spineplot now attach **stats** so that `model.frame()` is visible where they evaluate it.
- Date-time objects are no longer regarded as numeric by `is.numeric()`.
- `methods("Math")` did not work if **methods** was not attached.
- `readChar()` read an extra empty item (or more than one) beyond the end of the source; in some conditions it would terminate early when reading an item of length 0.
- Added a promise evaluation stack so interrupted promise evaluations can be restarted.
- `R.version[1:10]` now nicely prints.
- In the **methods** package, prototypes are now inherited for the `.Data` "slot"; i.e., for classes that contain one of the basic data types.
- *data_frame*`[[i, j]]` now works if `i` is character.
- `write.dcf()` no longer writes `NA` fields (PR#9796), and works correctly on empty descriptions.
- `pbeta(x, log.p = TRUE)` now has improved accuracy in many cases, and so have functions depending on it such as `pt()`, `pf()` and `pbinom()`.
- `mle()` had problems with the L-BFGS-B in the no-parameter case and consequentially also when profiling 1-parameter models (fix thanks to Ben Bolker).
- Two bugs fixed in methods that in involve the `...` argument in the generic function: previously failed to catch methods that just dropped the `...`; and use of `callGeneric()` with no arguments failed in some circumstances when `...` was a formal argument.
- `sequence()` now behaves more reasonably, although not back-compatibly for zero or negative input.

- `nls()` now allows more peculiar but reasonable ways of being called, e.g., with `data = list(`*uneven_lengths*`)` or a model without variables.
- `match.arg()` was not behaving as documented when `several.ok = TRUE` (PR#9859), gave spurious warnings when `arg` had the wrong length and was incorrectly documented (exact matches are returned even when there is more than one partial match).
- The `data.frame` method for `split<-()` was broken.
- The test for `-D__NO_MATH_INLINES` was badly broken and returned true on all non-glibc platforms and false on all glibc ones (whether they were broken or not).
- LF was missing after the last prompt when '`--quiet`' was used without '`--slave`'. Use '`--slave`' when no final LF is desired.
- Fixed bug in initialisation code in **grid** package for determining the boundaries of shapes. Problem reported by Hadley Wickham; symptom was error message: '`Polygon edge not found`'.
- `str()` is no longer slow for large `POSIXct` objects. Its output is also slightly more compact for such objects; implementation via new optional argument `give.head`.
- `strsplit(*, fixed = TRUE)`, potentially `iconv()` and internal string formatting is now faster for large strings, thanks to report PR#9902 by John Brzustowski.
- `de.restore()` gave a spurious warning for matrices (Ben Bolker)
- `plot(fn, xlim = c(a, b))` would not set `from` and `to` properly when plotting a function. The argument lists to `curve()` and `plot.function()` have been modified slightly as part of the fix.
- `julian()` was documented to work with `POSIXt` origins, but did not work with `POSIXlt` ones. (PR#9908)
- Dataset `HairEyeColor` has been corrected to agree with Friendly (2000): the change involves the breakdown of the Brown hair / Brown eye cell by Sex, and only totals over Sex are given in the original source.
- Trailing spaces are now consistently stripped from `\alias{}` entries in '`.Rd`' files, and this is now documented. (PR#9915)
- `.find.packages()`, `packageDescription()` and `sessionInfo()` assumed that attached environments named `"package:foo"` were package environments, although misguided users could use such a name in `attach()`.
- `spline()` and `splinefun()` with `method = "periodic"` could return incorrect results when `length(x)` was 2 or 3.
- `getS3method()` could fail if the method name contained a regexp metacharacter such as `"+"`.
- `help(`*a_character_vector*`)` now uses the name and not the value of the vector unless it has length exactly one, so e.g. `help(letters)` now gives help on `letters`. (Related to PR#9927)
- Ranges in `chartr()` now work better in CJK locales, thanks to Ei-ji Nakama.

Changes on CRAN

by Kurt Hornik

New contributed packages

ADaCGH Analysis and plotting of array CGH data. Allows usage of Circular Binary Segmentation, wavelet-based smoothing, ACE method (CGH Explorer), HMM, BioHMM, GLAD, CGHseg, and Price's modification of Smith & Waterman's algorithm. Most computations are parallelized. Figures are imagemaps with links to IDClight (`http://idclight.bioinfo.cnio.es`). By Ramon Diaz-Uriarte and Oscar M. Rueda. Wavelet-based aCGH smoothing code from Li Hsu and Douglas Grove, imagemap code from Barry Rowlingson.

AIS Tools to look at the data ("Ad Inidicia Spectata"). By Micah Altman.

AcceptanceSampling Creation and evaluation of Acceptance Sampling Plans. Plans can be single, double or multiple sampling plans. By Andreas Kiermeier.

Amelia Amelia II: A Program for Missing Data. Amelia II "multiply imputes" missing data in a single cross-section (such as a survey), from a time series (like variables collected for each

year in a country), or from a time-series-cross-sectional data set (such as collected by years for each of several countries). Amelia II implements a bootstrapping-based algorithm that gives essentially the same answers as the standard IP or EMis approaches, is usually considerably faster than existing approaches and can handle many more variables. The program also generalizes existing approaches by allowing for trends in time series across observations within a cross-sectional unit, as well as priors that allow experts to incorporate beliefs they have about the values of missing cells in their data. The program works from the R command line or via a graphical user interface that does not require users to know R. By James Honaker, Gary King, and Matthew Blackwell.

BiodiversityR A GUI (via **Rcmdr**) and some utility functions (often based on the **vegan**) for statistical analysis of biodiversity and ecological communities, including species accumulation curves, diversity indices, Renyi profiles, GLMs for analysis of species abundance and presence-absence, distance matrices, Mantel tests, and cluster, constrained and unconstrained ordination analysis. By Roeland Kindt.

CORREP Multivariate correlation estimator and statistical inference procedures. By Dongxiao Zhu and Youjuan Li.

CPGchron Create radiocarbon-dated depth chronologies, following the work of Parnell and Haslett (2007, submitted to JRSSC). By Andrew Parnell.

Cairo Provides a Cairo graphics device that can be use to create high-quality vector (PDF, PostScript and SVG) and bitmap output (PNG, JPEG, TIFF), and high-quality rendering in displays (X11 and Win32). Since it uses the same back-end for all output, copying across formats is WYSIWYG. Files are created without the dependence on X11 or other external programs. This device supports alpha channel (semi-transparent drawing) and resulting images can contain transparent and semi-transparent regions. It is ideal for use in server environments (file output) and as a replacement for other devices that don't have Cairo's capabilities such as alpha support or anti-aliasing. Backends are modular such that any subset of backends is supported. By Simon Urbanek and Jeffrey Horner.

CarbonEL Carbon Event Loop: hooks a Carbon event loop handler into R. This is useful for enabling UI from a console R (such as using the Quartz device from Terminal or ESS). By Simon Urbanek.

DAAGbio Data sets and functions useful for the display of microarray and for demonstrations with microarray data. By John Maindonald.

Defaults Set, get, and import global function defaults. By Jeffrey A. Ryan.

Devore7 Data sets and sample analyses from Jay L. Devore (2008), "Probability and Statistics for Engineering and the Sciences (7th ed)", Thomson. Original by Jay L. Devore, modifications by Douglas Bates, modifications for the 7th edition by John Verzani.

G1DBN Perform Dynamic Bayesian Network inference using 1st order conditional dependencies. By Sophie Lebre.

GSA Gene Set Analysis. By Brad Efron and R. Tibshirani.

GSM Gamma Shape Mixture. Implements a Bayesian approach for estimation of a mixture of gamma distributions in which the mixing occurs over the shape parameter. This family provides a flexible and novel approach for modeling heavy-tailed distributions, is computationally efficient, and only requires to specify a prior distribution for a single parameter. By Sergio Venturini.

GeneF Implements several generalized F-statistics, including ones based on the flexible isotonic/monotonic regression or order restricted hypothesis testing. By Yinglei Lai.

HFWutils Functions for Excel connections, garbage collection, string matching, and passing by reference. By Felix Wittmann.

ICEinfer Incremental Cost-Effectiveness (ICE) Statistical Inference. Given two unbiased samples of patient level data on cost and effectiveness for a pair of treatments, make head-to-head treatment comparisons by (i) generating the bivariate bootstrap resampling distribution of ICE uncertainty for a specified value of the shadow price of health, λ, (ii) form the wedge-shaped ICE confidence region with specified confidence fraction within $[0.50, 0.99]$ that is equivariant with respect to changes in λ, (iii) color the bootstrap outcomes within the above confidence wedge with economic preferences from an ICE map with specified values of λ, beta and gamma parameters, (iv) display VAGR and ALICE acceptability curves, and (v) display indifference (iso-preference) curves from an ICE map with specified values of λ, β and γ or η parameters. By Bob Obenchain.

ICS ICS/ICA computation based on two scatter matrices. Implements Oja et al.'s method of two different scatter matrices to obtain an invariant coordinate system or independent components, depending on the underlying assumptions. By Klaus Nordhausen, Hannu Oja, and Dave Tyler.

ICSNP Tools for multivariate nonparametrics, as location tests based on marginal ranks, spatial median and spatial signs computation, Hotelling's T-test, estimates of shape. By Klaus Nordhausen, Seija Sirkia, Hannu Oja, and Dave Tyler.

LLAhclust Hierarchical clustering of variables or objects based on the likelihood linkage analysis method. The likelihood linkage analysis is a general agglomerative hierarchical clustering method developed in France by Lerman in a long series of research articles and books. Initially proposed in the framework of variable clustering, it has been progressively extended to allow the clustering of very general object descriptions. The approach mainly consists in replacing the value of the estimated similarity coefficient by the probability of finding a lower value under the hypothesis of "absence of link". Package **LLAhclust** contains routines for computing various types of probabilistic similarity coefficients between variables or object descriptions. Once the similarity values between variables/objects are computed, a hierarchical clustering can be performed using several probabilistic and non-probabilistic aggregation criteria, and indices measuring the quality of the partitions compatible with the resulting hierarchy can be computed. By Ivan Kojadinovic, Israël-César Lerman, and Philippe Peter.

LLN Learning with Latent Networks. A new framework in which graph-structured data are used to train a classifier in a latent space, and then classify new nodes. During the learning phase, a latent representation of the network is first learned and a supervised classifier is then built in the learned latent space. In order to classify new nodes, the positions of these nodes in the learned latent space are estimated using the existing links between the new nodes and the learning set nodes. It is then possible to apply the supervised classifier to assign each new node to one of the classes. By Charles Bouveyron & Hugh Chipman.

LearnBayes Functions helpful in learning the basic tenets of Bayesian statistical inference. Contains functions for summarizing basic one and two parameter posterior distributions and predictive distributions, MCMC algorithms for summarizing posterior distributions defined by the user, functions for regression models, hierarchical models, Bayesian tests, and illustrations of Gibbs sampling. By Jim Albert.

LogConcDEAD Computes the maximum likelihood estimator from an i.i.d. sample of data from a log-concave density in any number of dimensions. Plots are available for 1- and 2-d data. By Madeleine Cule, Robert Gramacy, and Richard Samworth.

MLDS Maximum Likelihood Difference Scaling. Difference scaling is a method for scaling perceived supra-threshold differences. The package contains functions that allow the user to design and run a difference scaling experiment, to fit the resulting data by maximum likelihood and test the internal validity of the estimated scale. By Kenneth Knoblauch and Laurence T. Maloney, based in part on C code written by Laurence T. Maloney and J. N. Yang.

MLEcens MLE for bivariate (interval) censored data. Contains functions to compute the nonparametric maximum likelihood estimator for the bivariate distribution of (X, Y), when realizations of (X, Y) cannot be observed directly. More precisely, the MLE is computed based on a set of rectangles ("observation rectangles") that are known to contain the unobservable realizations of (X, Y). The methods can also be used for univariate censored data, and for censored data with competing risks. The package contains the functionality of **bicreduc**, which will no longer be maintained. By Marloes Maathuis.

MiscPsycho Miscellaneous Psychometrics: statistical analyses useful for applied psychometricians. By Harold C. Doran.

ORMDR Odds ratio based multifactor-dimensionality reduction method for detecting gene-gene interactions. By Eun-Kyung Lee, Sung Gon Yi, Yoojin Jung, and Taesung Park.

PET Simulation and reconstruction of PET images. Implements different analytic/direct and iterative reconstruction methods of Peter Toft, and offers the possibility to simulate PET data. By Joern Schulz, Peter Toft, Jesper James Jensen, and Peter Philipsen.

PSAgraphics Propensity Score Analysis (PSA) Graphics. Includes functions which test balance within strata of categorical and quantitative covariates, give a representation of the estimated effect size by stratum, provide a graphic and loess based effect size estimate, and various balance functions that provide measures of the balance achieved via a PSA in a categorical

covariate. By James E. Helmreich and Robert M. Pruzek.

PerformanceAnalytics Econometric tools for performance and risk analysis. Aims to aid practitioners and researchers in utilizing the latest research in analysis of non-normal return streams. In general, the package is most tested on return (rather than price) data on a monthly scale, but most functions will work with daily or irregular return data as well. By Peter Carl and Brian G. Peterson.

QRMlib Code to examine Quantitative Risk Management concepts, accompanying the book "Quantitative Risk Management: Concepts, Techniques and Tools" by Alexander J. McNeil, Rüdiger Frey and Paul Embrechts. S-Plus original by Alexander McNeil; R port by Scott Ulman.

R.cache Fast and light-weight caching of objects. Methods for memoization, that is, caching arbitrary R objects in persistent memory. Objects can be loaded and saved stratified on a set of hashing objects. By Henrik Bengtsson.

R.huge Methods for accessing huge amounts of data. Provides a class representing a matrix where the actual data is stored in a binary format on the local file system. This way the size limit of the data is set by the file system and not the memory. Currently in an alpha/early-beta version. By Henrik Bengtsson.

RBGL Interface to boost C++ graph library. Demo of interface with full copy of all hpp defining boost. By Vince Carey, Li Long, and R. Gentleman.

RDieHarder An interface to the dieharder test suite of random number generators and tests that was developed by Robert G. Brown, extending earlier work by George Marsaglia and others. By Dirk Eddelbuettel.

RLRsim Exact (Restricted) Likelihood Ratio tests for mixed and additive models. Provides rapid simulation-based tests for the presence of variance components/nonparametric terms with a convenient interface for a variety of models. By Fabian Scheipl.

ROptEst Optimally robust estimation using S4 classes and methods. By Matthias Kohl.

ROptRegTS Optimally robust estimation for regression-type models using S4 classes and methods. By Matthias Kohl.

RSVGTipsDevice An R SVG graphics device with dynamic tips and hyperlinks using the w3.org XML standard for Scalable Vector Graphics. Supports tooltips with 1 to 3 lines and line styles. By Tony Plate, based on **RSvgDevice** by T Jake Luciani.

Rcapture Loglinear Models in Capture-Recapture Experiments. Estimation of abundance and other demographic parameters for closed populations, open populations and the robust design in capture-recapture experiments using loglinear models. By Sophie Baillargeon and Louis-Paul Rivest.

RcmdrPlugin.TeachingDemos Provides an **Rcmdr** "plug-in" based on the **TeachingDemos** package, and is primarily for illustrative purposes. By John Fox.

Reliability Functions for estimating parameters in software reliability models. Only infinite failure models are implemented so far. By Andreas Wittmann.

RiboSort Rapid classification of (TRFLP & ARISA) microbial community profiles, eliminating the laborious task of manually classifying community fingerprints in microbial studies. By automatically assigning detected fragments and their respective relative abundances to appropriate ribotypes, **RiboSort** saves time and greatly simplifies the preparation of DNA fingerprint data sets for statistical analysis. By Una Scallan & Ann-Kathrin Liliensiek.

Rmetrics Rmetrics —- Financial Engineering and Computational Finance. Environment for teaching "Financial Engineering and Computational Finance". By Diethelm Wuertz and many others.

RobLox Optimally robust influence curves in case of normal location with unknown scale. By Matthias Kohl.

RobRex Optimally robust influence curves in case of linear regression with unknown scale and standard normal distributed errors where the regressor is random. By Matthias Kohl.

Rsac Seismic analysis tools in R. Mostly functions to reproduce some of the Seismic Analysis Code (SAC, `http://www.llnl.gov/sac/`) commands in R. This includes reading standard binary '.SAC' files, plotting arrays of seismic recordings, filtering, integration, differentiation, instrument deconvolution, and rotation of horizontal components. By Eric M. Thompson.

Runuran Interface to the UNU.RAN library for Universal Non-Uniform RANdom variate generators. By Josef Leydold and Wolfgang Hörmann.

Ryacas An interface to the yacas computer algebra system. By Rob Goedman, Gabor Grothendieck, Søren Højsgaard, Ayal Pinkus.

SRPM Shared Reproducibility Package Management. A package development and management system for distributed reproducible research. By Roger D. Peng.

SimHap A comprehensive modeling framework for epidemiological outcomes and a multiple-imputation approach to haplotypic analysis of population-based data. Can perform single SNP and multi-locus (haplotype) association analyses for continuous Normal, binary, longitudinal and right-censored outcomes measured in population-based samples. Uses estimation maximization techniques for inferring haplotypic phase in individuals, and incorporates a multiple-imputation approach to deal with the uncertainty of imputed haplotypes in association testing. By Pamela A. McCaskie.

SpatialNP Multivariate nonparametric methods based on spatial signs and ranks. Contains test and estimates of location, tests of independence, tests of sphericity, several estimates of shape and regression all based on spatial signs, symmetrized signs, ranks and signed ranks. By Seija Sirkia, Jaakko Nevalainen, Klaus Nordhausen, and Hannu Oja.

TIMP Problem solving environment for fitting superposition models. Measurements often represent a superposition of the contributions of distinct sub-systems resolved with respect to many experimental variables (time, temperature, wavelength, pH, polarization, etc). **TIMP** allows parametric models for such superpositions to be fit and validated. The package has been extensively applied to modeling data arising in spectroscopy experiments. By Katharine M. Mullen and Ivo H. M. van Stokkum.

TwslmSpikeWeight Normalization of cDNA microarray data with the two-way semilinear model(TW-SLM). It incorporates information from control spots and data quality in the TW-SLM to improve normalization of cDNA microarray data. Huber's and Tukey's bisquare weight functions are available for robust estimation of the TW-SLM. By Deli Wang and Jian Huang.

Umacs Universal Markov chain sampler. By Jouni Kerman.

WilcoxCV Functions to perform fast variable selection based on the Wilcoxon rank sum test in the cross-validation or Monte-Carlo cross-validation settings, for use in microarray-based binary classification. By Anne-Laure Boulesteix.

YaleToolkit Tools for the graphical exploration of complex multivariate data developed at Yale University. By John W. Emerson and Walton Green.

adegenet Genetic data handling for multivariate analysis using **ade4**. By Thibaut Jombart.

ads Spatial point patterns analysis. Perform first- and second-order multi-scale analyses derived from Ripley's *K*-function, for univariate, multivariate and marked mapped data in rectangular, circular or irregular shaped sampling windows, with test of statistical significance based on Monte Carlo simulations. By R. Pelissier and F. Goreaud.

argosfilter Functions to filter animal satellite tracking data obtained from Argos. Especially indicated for telemetry studies of marine animals, where Argos locations are predominantly of low quality. By Carla Freitas.

arrayImpute Missing imputation for microarray data. By Eun-kyung Lee, Dankyu Yoon, and Taesung Park.

ars Adaptive Rejection Sampling. By Paulino Perez Rodriguez; original C++ code from Arnost Komarek.

arulesSequences Add-on for **arules** to handle and mine frequent sequences. Provides interfaces to the C++ implementation of cSPADE by Mohammed J. Zaki. By Christian Buchta and Michael Hahsler.

asuR Functions and data sets for a lecture in Advanced Statistics using R. Especially the functions `mancontr()` and `inspect()` may be of general interest. With the former, it is possible to specify your own contrasts and give them useful names. Function `inspect()` shows a wide range of inspection plots to validate model assumptions. By Thomas Fabbro.

bayescount Bayesian analysis of count distributions with JAGS. A set of functions to apply a zero-inflated gamma Poisson (equivalent to a zero-inflated negative binomial), zero-inflated Poisson, gamma Poisson (negative binomial) or Poisson model to a set of count data using JAGS (Just Another Gibbs Sampler). Returns information on the possible values for mean count, overdispersion and zero inflation present in count data such as faecal egg count data. By Matthew Denwood.

benchden Full implementation of the 28 distributions introduced as benchmarks for nonparametric density estimation by Berlinet and Devroye (1994). Includes densities, cdfs, quantile functions and generators for samples. By Thoralf Mildenberger, Henrike Weinert, and Sebastian Tiemeyer.

biOps Basic image operations and image processing. Includes arithmetic, logic, look up table and geometric operations. Some image processing functions, for edge detection (several algorithms including Roberts, Sobel, Kirsch, Marr-Hildreth, Canny) and operations by convolution masks (with predefined as well as user defined masks) are provided. Supported file formats are jpeg and tiff. By Matias Bordese and Walter Alini.

binMto Asymptotic simultaneous confidence intervals for comparison of many treatments with one control, for the difference of binomial proportions, allows for Dunnett-like-adjustment, Bonferroni or unadjusted intervals. Simulation of power of the above interval methods, approximate calculation of any-pair-power, and sample size iteration based on approximate any-pair power. Exact conditional maximum test for many-to-one comparisons to a control. By Frank Schaarschmidt.

bio.infer Compute biological inferences. Imports benthic count data, reformats this data, and computes environmental inferences from this data. By Lester L. Yuan.

blockTools Block, randomly assign, and diagnose potential problems between units in randomized experiments. Blocks units into experimental blocks, with one unit per treatment condition, by creating a measure of multivariate distance between all possible pairs of units. Maximum, minimum, or an allowable range of differences between units on one variable can be set. Randomly assign units to treatment conditions. Diagnose potential interference problems between units assigned to different treatment conditions. Write outputs to '.tex' and '.csv' files. By Ryan T. Moore.

bootStepAIC Model selection by bootstrapping the `stepAIC()` procedure. By Dimitris Rizopoulos.

brew A templating framework for mixing text and R code for report generation. Template syntax is similar to PHP, Ruby's erb module, Java Server Pages, and Python's psp module. By Jeffrey Horner.

ca Computation and visualization of simple, multiple and joint correspondence analysis. By Michael Greenacre and Oleg Nenadic.

catmap Case-control And Tdt Meta-Analysis Package. Conducts fixed-effects (inverse variance) and random-effects (DerSimonian and Laird, 1986) meta-analyses of case-control or family-based (TDT) genetic data; in addition, performs meta-analyses combining these two types of study designs. The fixed-effects model was first described by Kazeem and Farrell (2005); the random-effects model is described in Nicodemus (submitted for publication). By Kristin K. Nicodemus.

celsius Retrieve Affymetrix microarray measurements and metadata from Celsius web services, see `http://genome.ucla.edu/projects/celsius`. By Allen Day, Marc Carlson.

cghFLasso Spatial smoothing and hot spot detection using the fused lasso regression. By Robert Tibshirani and Pei Wang.

choplump Choplump tests: permutation tests for comparing two groups with some positive but many zero responses. By M. P. Fay.

clValid Statistical and biological validation of clustering results. By Guy Brock, Vasyl Pihur, Susmita Datta, and Somnath Datta.

classGraph Construct directed graphs of S4 class hierarchies and visualize them. Typically, these graphs are DAGs (directed acyclic graphs) in general, though often trees. By Martin Maechler partly based on code from Robert Gentleman.

clinfun Clinical Trial Design and Data Analysis Functions. Utilities to make your clinical collaborations easier if not fun. By E. S. Venkatraman.

clusterfly Explore clustering interactively using R and GGobi. Contains both general code for visualizing clustering results and specific visualizations for model-based, hierarchical and SOM clustering. By Hadley Wickham.

clv Cluster Validation Techniques. Contains most of the popular internal and external cluster validation methods ready to use for the most of the outputs produced by functions coming from package **cluster**. By Lukasz Nieweglowski.

codetools Code analysis tools for R. By Luke Tierney.

colorRamps Builds single and double gradient color maps. By Tim Keitt.

contrast Contrast methods, in the style of the **Design** package, for fit objects produced by the `lm`, `glm`, `gls`, and `geese` functions. By Steve Weston, Jed Wing and Max Kuhn.

coxphf Cox regression with Firth's penalized likelihood. R by Meinhard Ploner, Fortran by Georg Heinze.

crosshybDetector Identification of probes potentially affected by cross-hybridizations in microarray experiments. Includes functions for diagnostic plots. By Paolo Uva.

ddesolve Solver for Delay Differential Equations by interfacing numerical routines written by Simon N. Wood, with contributions by Benjamin J. Cairns. These numerical routines first appeared in Simon Wood's solv95 program. By Alex Couture-Beil, Jon T. Schnute, and Rowan Haigh.

desirability Desirability Function Optimization and Ranking. S3 classes for multivariate optimization using the desirability function by Derringer and Suich (1980). By Max Kuhn.

dplR Dendrochronology Program Library in R. Contains functions for performing some standard tree-ring analyses. By Andy Bunn.

dtt Discrete Trigonometric Transforms. Functions for 1D and 2D Discrete Cosine Transform (DCT), Discrete Sine Transform (DST) and Discrete Hartley Transform (DHT). By Lukasz Komsta.

earth Multivariate Adaptive Regression Spline Models. Build regression models using the techniques in Friedman's papers "Fast MARS" and "Multivariate Adaptive Regression Splines". (The term "MARS" is copyrighted and thus not used as the name of the package.). By Stephen Milborrow, derived from code in package **mda** by Trevor Hastie and Rob Tibshirani.

eigenmodel Semiparametric factor and regression models for symmetric relational data. Estimates the parameters of a model for symmetric relational data (e.g., the above-diagonal part of a square matrix) using a model-based eigenvalue decomposition and regression. Missing data is accommodated, and a posterior mean for missing data is calculated under the assumption that the data are missing at random. The marginal distribution of the relational data can be arbitrary, and is fit with an ordered probit specification. By Peter Hoff.

epibasix Elementary tools for the analysis of common epidemiological problems, ranging from sample size estimation, through 2×2 contingency table analysis and basic measures of agreement (kappa, sensitivity/specificity). Appropriate print and summary statements are also written to facilitate interpretation wherever possible. The target audience includes biostatisticians and epidemiologists who would like to apply standard epidemiological methods in a simple manner. By Michael A Rotondi.

experiment Various statistical methods for designing and analyzing randomized experiments. One main functionality is the implementation of randomized-block and matched-pair designs based on possibly multivariate pretreatment covariates. Also provides the tools to analyze various randomized experiments including cluster randomized experiments, randomized experiments with noncompliance, and randomized experiments with missing data. By Kosuke Imai.

fCopulae Rmetrics — Dependence Structures with Copulas. Environment for teaching "Financial Engineering and Computational Finance". By Diethelm Wuertz and many others.

forensic Statistical Methods in Forensic Genetics. Statistical evaluation of DNA mixtures, DNA profile match probability. By Miriam Marusiakova.

fractal Insightful Fractal Time Series Modeling and Analysis. Software to book in development entitled "Fractal Time Series Analysis in S-PLUS and R" by William Constantine and Donald B. Percival, Springer. By William Constantine and Donald Percival.

fso Fuzzy Set Ordination: a multivariate analysis used in ecology to relate the composition of samples to possible explanatory variables. While differing in theory and method, in practice, the use is similar to "constrained ordination". Contains plotting and summary functions as well as the analyses. By David W. Roberts.

gWidgetstcltk Toolkit implementation of the **gWidgets** API for the **tcltk** package. By John Verzani.

gamlss.mx A GAMLSS add on package for fitting mixture distributions. By Mikis Stasinopoulos and Bob Rigby.

gcl Computes a fuzzy rules classifier (Vinterbo, Kim & Ohno-Machado, 2005). By Staal A. Vinterbo.

geiger Analysis of evolutionary diversification. Features running macroevolutionary simulation and estimating parameters related to diversification from comparative phylogenetic data. By Luke Harmon, Jason Weir, Chad Brock, Rich Glor, and Wendell Challenger.

ggplot An implementation of the grammar of graphics in R. Combines the advantages of

both **base** and **lattice** graphics: conditioning and shared axes are handled automatically, and one can still build up a plot step by step from multiple data sources. Also implements a more sophisticated multidimensional conditioning system and a consistent interface to map data to aesthetic attributes. By Hadley Wickham.

ghyp Provides all about univariate and multivariate generalized hyperbolic distributions and its special cases (Hyperbolic, Normal Inverse Gaussian, Variance Gamma and skewed Student t distribution). Especially fitting procedures, computation of the density, quantile, probability, random variates, expected shortfall and some portfolio optimization and plotting routines. Also contains the generalized inverse Gaussian distribution. By Wolfgang Breymann and David Luethi.

glmmAK Generalized Linear Mixed Models. By Arnost Komarek.

granova Graphical Analysis of Variance. Provides distinctive graphics for display of ANOVA results. Functions were written to display data for any number of groups, regardless of their sizes (however, very large data sets or numbers of groups are likely to be problematic) using a specialized approach to construct data-based contrast vectors with respect to which ANOVA data are displayed. By Robert M. Pruzek and James E. Helmreich.

graph Implements some simple graph handling capabilities. By R. Gentleman, Elizabeth Whalen, W. Huber, and S. Falcon.

grasp Generalized Regression Analysis and Spatial Prediction for R. GRASP is a general method for making spatial predictions of response variables (RV) using point surveys of the RV and spatial coverages of predictor variables (PV). Originally, GRASP was developed to analyse, model and predict vegetation distribution over New Zealand. It has been used in all sorts of applications since then. By Anthony Lehmann, Fabien Fivaz, John Leathwick and Jake Overton, with contributions from many specialists from around the world.

hdeco Hierarchical DECOmposition of Entropy for Categorical Map Comparisons. A flexible and hierarchical framework for comparing categorical map composition and configuration (spatial pattern) along spatial, thematic, or external grouping variables. Comparisons are based on measures of mutual information between thematic classes (colors) and location (spatial partitioning). Results are returned in textual, tabular, and graphical forms. By Tarmo K. Remmel, Sandor Kabos, and Ferenc (Ferko) Csillag.

heatmap.plus Heatmaps with more sensible behavior. Allows the heatmap matrix to have non-identical x and y dimensions, and multiple tracks of annotation for `RowSideColors` and `ColSideColors`. By Allen Day.

hydrosanity Graphical user interface for exploring hydrological time series. Designed to work with catchment surface hydrology data (rainfall and streamflow); but could also be used with other time series data. Hydrological time series typically have many missing values, and varying accuracy of measurement (indicated by data quality codes). Furthermore, the spatial coverage of data varies over time. Much of the functionality of this package attempts to understand these complex data sets, detect errors and characterize sources of uncertainty. Emphasis is on graphical methods. The GUI is based on **rattle**. By Felix Andrews.

identity Calculate identity coefficients, based on Mark Abney's C code. By Na Li.

ifultools Insightful Research Tools. By William Constantine.

inetwork Network Analysis and Plotting. Implements a network partitioning algorithm to identify communities (or modules) in a network. A network plotting function then utilizes the identified community structure to position the vertices for plotting. Also contains functions to calculate the assortativity and transitivity of a vertex. By Sun-Chong Wang.

inline Functionality to dynamically define R functions and S4 methods with in-lined C, C++ or Fortran code supporting `.C` and `.Call` calling conventions. By Oleg Sklyar, Duncan Murdoch, and Mike Smith.

irtoys A simple common interface to the estimation of item parameters in IRT models for binary responses with three different programs (ICL, BILOG-MG, and `ltm`), and a variety of functions useful with IRT models. By Ivailo Partchev.

kin.cohort Analysis of kin-cohort studies. Provides estimates of age-specific cumulative risk of a disease for carriers and noncarriers of a mutation. The cohorts are retrospectively built from relatives of probands for whom the genotype is known. Currently the method of moments and marginal maximum likelihood are implemented. Confidence intervals are calculated from bootstrap samples. Most of the code is

a translation from previous MATLAB code by Chatterjee. By Victor Moreno, Nilanjan Chatterjee, and Bhramar Mukherjee.

kzs Kolmogorov-Zurbenko Spline. A collection of functions utilizing splines to smooth a noisy data set in order to estimate its underlying signal. By Derek Cyr and Igor Zurbenko.

lancet.iraqmortality Surveys of Iraq mortality published in The Lancet. The Lancet has published two surveys on Iraq mortality before and after the US-led invasion. The package facilitates access to the data and a guided tour of some of their more interesting aspects. By David Kane, with contributions from Arjun Navi Narayan and Jeff Enos.

ljr Fits and tests logistic joinpoint models. By Michal Czajkowski, Ryan Gill, and Greg Rempala.

luca Likelihood Under Covariate Assumptions (LUCA). Likelihood inference in case-control studies of a rare disease under independence or simple dependence of genetic and non-genetic covariates. By Ji-Hyung Shin, Brad McNeney, and Jinko Graham.

meifly Interactive model exploration using GGobi. By Hadley Wickham.

mixPHM Mixtures of proportional hazard models. Fits multiple variable mixtures of various parametric proportional hazard models using the EM algorithm. Proportionality restrictions can be imposed on the latent groups and/or on the variables. Several survival distributions can be specified. Missing values are allowed. Independence is assumed over the single variables. By Patrick Mair and Marcus Hudec.

mlmmm Computational strategies for multivariate linear mixed-effects models with missing values (Schafer and Yucel, 2002). By Recai Yucel.

modeest Provides estimators of the mode of univariate unimodal data or univariate unimodal distributions. Also allows to compute the Chernoff distribution. By Paul Poncet.

modehunt Multiscale Analysis for Density Functions. Given independent and identically distributed observations $X(1), \ldots, X(n)$ from a density f, this package provides five methods to perform a multiscale analysis about f as well as the necessary critical values. The first method, introduced in Duembgen and Walther (2006), provides simultaneous confidence statements for the existence and location of local increases (or decreases) of f, based on all intervals I(all) spanned by any two observations $X(j), X(k)$. The second method approximates the latter approach by using only a subset of I(all) and is therefore computationally much more efficient, but asymptotically equivalent. Omitting the additive correction term Gamma in either method offers another two approaches which are more powerful on small scales and less powerful on large scales, however, not asymptotically minimax optimal anymore. Finally, the block procedure is a compromise between adding Gamma or not, having intermediate power properties. The latter is again asymptotically equivalent to the first and was introduced in Rufibach and Walther (2007). By Kaspar Rufibach and Guenther Walther.

monomvn Estimation of multivariate normal data of arbitrary dimension where the pattern of missing data is monotone. Through the use of parsimonious/shrinkage regressions (plsr, pcr, lasso, ridge, etc.), where standard regressions fail, the package can handle an (almost) arbitrary amount of missing data. The current version supports maximum likelihood inference. Future versions will provide a means of sampling from a Bayesian posterior. By Robert B. Gramacy.

mota Mean Optimal Transformation Approach for detecting nonlinear functional relations. Originally designed for the identifiability analysis of nonlinear dynamical models. However, the underlying concept is very general and allows to detect groups of functionally related variables whenever there are multiple estimates for each variable. By Stefan Hengl.

nlstools Tools for assessing the quality of fit of a Gaussian nonlinear model. By Florent Baty and Marie-Laure Delignette-Muller, with contributions from Sandrine Charles, Jean-Pierre Flandrois.

oc OC Roll Call Analysis Software. Estimates Optimal Classification scores from roll call votes supplied though a `rollcall` object from package **pscl**. By Keith Poole, Jeffrey Lewis, James Lo and Royce Carroll.

oce Analysis of Oceanographic data. Supports CTD measurements, sea-level time series, coastline files, etc. Also includes functions for calculating seawater properties such as density, and derived properties such as buoyancy frequency. By Dan Kelley.

pairwiseCI Calculation of parametric and nonparametric confidence intervals for the difference or ratio of location parameters and for the difference, ratio and odds-ratio of binomial proportion for comparison of independent samples. CIs are *not* adjusted for multiplicity. A by

statement allows calculation of CI separately for the levels of one further factor. By Frank Schaarschmidt.

paran An implementation of Horn's technique for evaluating the components retained in a principle components analysis (PCA). Horn's method contrasts eigenvalues produced through a PCA on a number of random data sets of uncorrelated variables with the same number of variables and observations as the experimental or observational data set to produce eigenvalues for components that are adjusted for the sample error-induced inflation. Components with adjusted eigenvalues greater than one are retained. The package may also be used to conduct parallel analysis following Glorfeld's (1995) suggestions to reduce the likelihood of over-retention. By Alexis Dinno.

pcse Estimation of panel-corrected standard errors. Data may contain balanced or unbalanced panels. By Delia Bailey and Jonathan N. Katz.

penalized L1 (lasso) and L2 (ridge) penalized estimation in Generalized Linear Models and in the Cox Proportional Hazards model. By Jelle Goeman.

plRasch Fit log linear by linear association models. By Zhushan Li & Feng Hong.

plink Separate Calibration Linking Methods. Uses unidimensional item response theory methods to compute linking constants and conduct chain linking of tests for multiple groups under a nonequivalent groups common item design. Allows for mean/mean, mean/sigma, Haebara, and Stocking-Lord calibrations of single-format or mixed-format dichotomous (1PL, 2PL, and 3PL) and polytomous (graded response partial credit/generalized partial credit, nominal, and multiple-choice model) common items. By Jonathan Weeks.

plotAndPlayGTK A GUI for interactive plots using GTK+. When wrapped around plot calls, a window with the plot and a tool bar to interact with it pop up. By Felix Andrews, with contributions from Graham Williams.

pomp Inference methods for partially-observed Markov processes. By Aaron A. King, Ed Ionides, and Carles Breto.

poplab Population Lab: a tool for constructing a virtual electronic population of related individuals evolving over calendar time, by using vital statistics, such as mortality and fertility, and disease incidence rates. By Monica Leu, Kamila Czene, and Marie Reilly.

proftools Profile output processing tools. By Luke Tierney.

proj4 A simple interface to lat/long projection and datum transformation of the PROJ.4 cartographic projections library. Allows transformation of geographic coordinates from one projection and/or datum to another. By Simon Urbanek.

proxy Distance and similarity measures. An extensible framework for the efficient calculation of auto- and cross-proximities, along with implementations of the most popular ones. By David Meyer and Christian Buchta.

psych Routines for personality, psychometrics and experimental psychology. Functions are primarily for scale construction and reliability analysis, although others are basic descriptive stats. By William Revelle.

psyphy Functions that could be useful in analyzing data from pyschophysical experiments, including functions for calculating d' from several different experimental designs, links for mafc to be used with the binomial family in glm (and possibly other contexts) and self-Start functions for estimating gamma values for CRT screen calibrations. By Kenneth Knoblauch.

qualV Qualitative methods for the validation of models. By K.G. van den Boogaart, Stefanie Jachner and Thomas Petzoldt.

quantmod Specify, build, trade, and analyze quantitative financial trading strategies. By Jeffrey A. Ryan.

rateratio.test Exact rate ratio test. By Michael Fay.

regtest Functions for unary and binary regression tests. By Jens Oehlschlägel.

relations Data structures for k-ary relations with arbitrary domains, predicate functions, and fitters for consensus relations. By Kurt Hornik and David Meyer.

rgcvpack Interface to the GCVPACK Fortran package for thin plate spline fitting and prediction. By Xianhong Xie.

rgr Geological Survey of Canada (GSC) functions for exploratory data analysis with applied geochemical data, with special application to the estimation of background ranges to support both environmental studies and mineral exploration. By Robert G. Garrett.

rindex Indexing for R. Index structures allow quickly accessing elements from large collections. With btree optimized for disk databases

and ttree for RAM databases, uses hybrid static indexing which is quite optimal for R. By Jens Oehlschlägel.

rjson JSON for R. Converts R object into JSON objects and vice-versa. By Alex Couture-Beil.

rsbml R support for SBML. Links R to libsbml for SBML parsing and output, provides an S4 SBML DOM, converts SBML to R graph objects, and more. By Michael Lawrence.

sapa Insightful Spectral Analysis for Physical Applications. Software for the book "Spectral Analysis for Physical Applications" by Donald B. Percival and Andrew T. Walden, Cambridge University Press, 1993. By William Constantine and Donald Percival.

scagnostics Calculates Tukey's scagnostics which describe various measures of interest for pairs of variables, based on their appearance on a scatterplot. They are useful tool for weeding out interesting or unusual scatterplots from a scatterplot matrix, without having to look at ever individual plot. By Heike Hofmann,, Lee Wilkinson, Hadley Wickham, Duncan Temple Lang, and Anushka Anand.

schoolmath Functions and data sets for math used in school. A main focus is set to prime-calculation. By Joerg Schlarmann and Josef Wienand.

sdcMicro Statistical Disclosure Control methods for the generation of public- and scientific-use files. Data from statistical agencies and other institutions are mostly confidential. The package can be used for the generation of safe (micro)data, i.e., for the generation of public- and scientific-use files. By Matthias Templ.

seriation Infrastructure for seriation with an implementation of several seriation/sequencing techniques to reorder matrices, dissimilarity matrices, and dendrograms. Also contains some visualizations techniques based on seriation. By Michael Hahsler, Christian Buchta and Kurt Hornik.

signalextraction Real-Time Signal Extraction (Direct Filter Approach). The Direct Filter Approach (DFA) provides efficient estimates of signals at the current boundary of time series in real-time. For that purpose, one-sided ARMA-filters are computed by minimizing customized error criteria. The DFA can be used for estimating either the level or turning-points of a series, knowing that both criteria are incongruent. In the context of real-time turning-point detection, various risk-profiles can be operationalized, which account for the speed and/or the reliability of the one-sided filter. By Marc Wildi & Marcel Dettling.

simba Functions for similarity calculation of binary data (for instance presence/absence species data). Also contains wrapper functions for reshaping species lists into matrices and vice versa and some other functions for further processing of similarity data (Mantel-like permutation procedures) as well as some other useful stuff. By Gerald Jurasinski.

simco A package to import Structure files and deduce similarity coefficients from them. By Owen Jones.

snpXpert Tools to analysis SNP data. By Eun-kyung Lee, Dankyu Yoon, and Taesung Park.

spam SPArse Matrix: functions for sparse matrix algebra (used by **fields**). Differences with **SparseM** and **Matrix** are: (1) support for only one sparse matrix format, (2) based on transparent and simple structure(s) and (3) S3 and S4 compatible. By Reinhard Furrer.

spatgraphs Graphs, graph visualization and graph based summaries to be used with spatial point pattern analysis. By Tuomas Rajala.

splus2R Insightful package providing missing S-PLUS functionality in R. Currently there are many functions in S-PLUS that are missing in R. To facilitate the conversion of S-PLUS modules and libraries to R packages, this package helps to provide missing S-PLUS functionality in R. By William Constantine.

sqldf Manipulate R data frames using SQL. By G. Grothendieck.

sspline Smoothing splines on the sphere. By Xianhong Xie.

stochasticGEM Fitting Stochastic General Epidemic Models: Bayesian inference for partially observed stochastic epidemics. The general epidemic model is used for estimating the parameters governing the infectious and incubation period length, and the parameters governing susceptibility. In real-life epidemics the infection process is unobserved, and the data consists of the times individuals are detected, usually via appearance of symptoms. The package fits several variants of the general epidemic model, namely the stochastic SIR with Markovian and non-Markovian infectious periods. The estimation is based on Markov chain Monte Carlo algorithm. By Eugene Zwane.

surveyNG An experimental revision of the **survey** package for complex survey samples (featuring a database interface and sparse matrices). By Thomas Lumley.

termstrc Term Structure and Credit Spread Estimation. Offers several widely-used term structure estimation procedures, i.e., the parametric Nelson and Siegel approach, Svensson approach and cubic splines. By Robert Ferstl and Josef Hayden.

tframe Time Frame coding kernel. Functions for writing code that is independent of the way time is represented. By Paul Gilbert.

timsac TIMe Series Analysis and Control package. By The Institute of Statistical Mathematics.

trackObjs Track Objects. Stores objects in files on disk so that files are automatically rewritten when objects are changed, and so that objects are accessible but do not occupy memory until they are accessed. Also tracks times when objects are created and modified, and cache some basic characteristics of objects to allow for fast summaries of objects. By Tony Plate.

tradeCosts Post-trade analysis of transaction costs. By Aaron Schwartz and Luyi Zhao.

tripEstimation Metropolis sampler and supporting functions for estimating animal movement from archival tags and satellite fixes. By Michael Sumner and Simon Wotherspoon.

twslm A two-way semilinear model for normalization and analysis of cDNA microarray data. Huber's and Tukey's bisquare weight functions are available for robust estimation of the two-way semilinear models. By Deli Wang and Jian Huang.

vbmp Variational Bayesian multinomial probit regression with Gaussian process priors. By Nicola Lama and Mark Girolami.

vrtest Variance ratio tests for weak-form market efficiency. By Jae H. Kim.

waved The WaveD Transform in R. Makes available code necessary to reproduce figures and tables in recent papers on the WaveD method for wavelet deconvolution of noisy signals. By Marc Raimondo and Michael Stewart.

wikibooks Functions and datasets of the german WikiBook "GNU R" which introduces R to new users. By Joerg Schlarmann.

wmtsa Insightful Wavelet Methods for Time Series Analysis. Software to book "Wavelet Methods for Time Series Analysis" by Donald B. Percival and Andrew T. Walden, Cambridge University Press, 2000. By William Constantine and Donald Percival.

wordnet An interface to WordNet using the Jawbone Java API to WordNet. By Ingo Feinerer.

yest Model selection and variance estimation in Gaussian independence models. By Petr Simecek.

Other changes

- CRAN's Devel area is gone.
- Package **write.snns** was moved up from Devel.
- Package **anm** was resurrected from the Archive.
- Package **Rcmdr.HH** was renamed to **Rcmdr-Plugin.HH**.
- Package **msgcop** was renamed to **sbgcop**.
- Package **grasper** was moved to the Archive.
- Package **tapiR** was removed from CRAN.

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

R Foundation News

by Kurt Hornik

Donations and new members

Donations

AT&T Research (USA)
Jaimison Fargo (USA)
Stavros Panidis (Greece)
Saxo Bank (Denmark)
Julian Stander (United Kingdom)

New benefactors

Shell Statistics and Chemometrics, Chester, UK

New supporting institutions

AT&T Labs, New Jersey, USA

BC Cancer Agency, Vancouver, Canada

Black Mesa Capital, Santa Fe, USA

Department of Statistics, Unviersity of Warwick, Coventry, UK

New supporting members

Michael Bojanowski (Netherlands)
Caoimhin ua Buachalla (Ireland)
Jake Bowers (UK)
Sandrah P. Eckel (USA)
Charles Fleming (USA)
Hui Huang (USA)
Jens Oehlschlägel (Germany)
Marc Pelath (UK)
Tony Plate (USA)
Julian Stander (UK)

Kurt Hornik
Wirtschaftsuniversität Wien, Austria
Kurt.Hornik@R-project.org

News from the Bioconductor Project

by Hervé Pagès and Martin Morgan

Bioconductor 2.1 was released on October 8, 2007 and is designed to be compatible with R 2.6.0, released five days before Bioconductor. This release contains 23 new software packages, 97 new annotation (or metadata) packages, and many improvements and bug fixes to existing packages.

New packages

The 23 new software packages provide exciting analytic and interactive tools. Packages address Affymetrix array quality assessment (e.g., **arrayQualityMetrics**, **AffyExpress**), error assessment (e.g., **OutlierD**, **MCRestimate**), particular application domains (e.g., comparative genomic hybridization, **CGHcall**, **SMAP**; SNP arrays, **oligoClasses**, **RLMM**, **VanillaICE**; SAGE, **sagenhaft**; gene set enrichment, **GSEABase**; protein interaction, **Rintact**) and sophisticated modeling tools (e.g., **timecourse**, **vbmp**, **maigesPack**). **exploRase** uses the GTK toolkit to provide an integrated user interface for systems biology applications.

Several packages benefit from important infrastructure developments in R. Recent changes allow consolidation of C code common to several packages into a single location (**preprocessCore**), greatly simplifying code maintenance and improving reliability. Many new packages use the S4 class system. A number of these extend classes provided in **Biobase**, facilitating more seamless interoperability. The ability to access web resources, including convenient XML parsing, allow Bioconductor packages such as **GSEABase** to access important curated resources.

SQLite-based annotations

This release provides SQLite-based annotations in addition to the traditional environment-based ones. Annotations contain *maps* between information from microarray manufactures, standard naming conventions (e.g., Entrez gene identifiers) and resources such as the Gene Ontology consortium. Eighty-six SQLite-based annotation packages are currently available. The name of these packages end with `".db"` (e.g., **hgu95av2.db**). For an example of different metadata packages related to specific chips, view the annotations available for the hgu95av2 chip: `http://bioconductor.org/packages/2.1/hgu95av2.html`

New *genome wide* metadata packages provide a more complete set of maps, similar to those provided in the chip-based annotation packages. Genome wide annotations have an `"org."` prefix in their name, and are available as SQLite-based packages only. Five organisms are currently supported: human (**org.Hs.eg.db**), mouse (**org.Mm.eg.db**), rat (**org.Rn.eg.db**), fly (**org.Dm.eg.db**) and yeast (**org.Sc.sgd.db**). The ***LLMappings** packages will be deprecated in Bioconductor 2.2.

Environment-based (e.g., **hgu95av2**) and SQLite-based (e.g., **hgu95av2.db**) packages contain the same data. For the end user, moving from **hgu95av2** to **hgu95av2.db** is transparent because the objects (or *maps*) in **hgu95av2.db** can be manipulated as if they were environments (i.e., functions `ls`, `mget`, `get`, etc. . . still work on them). Using SQLite allows considerable flexibility in querying maps and in performing complex joins between tables, in addition to placing the burden of memory management and optimized query construction in sqlite. As with the implementation of operations like `ls` and `mget`, the intention is to recognize common use cases and to code these so that R users do not need to know the underlying SQL query.

Looking ahead

For the next release (BioC 2.2, April 2008) all our annotations will be SQLite-based and we will deprecate the environment-based versions.

We anticipate increasing emphasis on sequence-based technologies like Solexa (`http://www.illumina.com`) and 454 (`http://www.454.com`). The volume of data these technologies generate is very large (a three day Solexa run produces almost a terabyte of raw data, with 10's of gigabytes appropriate

for numerical analysis). This volume of data poses significant challenges, but graphical, statistical, and interactive abilities and web-based integration make R a strong candidate for making sense of, and making fundamental new contributions to, understanding and critically assessing this data.

The best Bioconductor packages are contributed by our users, based on their practical needs and sophisticated experience. We look forward to receiving your contribution over the next several months.

Hervé Pagès
Computational Biology Program
Fred Hutchinson Cancer Research Center
hpages@fhcrc.org

Martin Morgan
Computational Biology Program
Fred Hutchinson Cancer Research Center
mtmorgan@fhcrc.org

Past Events: useR! 2007

Duncan Murdoch and Martin Maechler

The first North American useR! meeting took place over three hot days at Iowa State University in Ames this past August.

The program started with John Chambers talking about his philosophy of programming: our mission is to enable effective and rapid exploration of data, and the prime directive is to provide trustworthy software and "tell no lies". This was followed by a varied and interesting program of presentations and posters, many of which are now online at http://useR2007.org. A panel on the use of R in clinical trials may be noteworthy because of the related publishing by the R Foundation of a document (http://www.r-project.org/certification.html) on "Regulatory Compliance and Validation" issues. In particular "21 CFR part 11" compliance is very important in the pharmaceutical industry.

The meeting marked the tenth anniversary of the formation of the R Core group, and an enormous blue R birthday cake was baked for the occasion (Figure 1). Six of the current R Core members were present for the cutting, and Thomas Lumley gave a short after dinner speech at the banquet.

Figure 1: R Core turns ten.

There were two programming competitions. The first requested submissions in advance, to produce a package useful for large data sets. This was won by the team of Daniel Adler, Oleg Nenadić, Walter Zucchini and Christian Gläser from Göttingen. They wrote the **ff** package to use paged memory-mapping of binary files to handle very large datasets. Daniel, Oleg and Christian attended the meeting and presented their package.

The second competition was a series of short R programming tasks to be completed within a time limit at the conference. The tasks included relabelling, working with ragged longitudinal data, and writing functions on functions. There was a tie for first place between Elaine McVey and Olivia Lau.

Three kinds of T-shirts with variations on the "R" theme were available: the official conference T-shirt (also worn by the local staff) sold out within a day, so the two publishers' free T-shirts gained even more attraction.

Local arrangements for the meeting were handled by Di Cook, Heike Hofmann, Michael Lawrence, Hadley Wickham, Denise Riker, Beth Hagenman and Karen Koppenhaver at Iowa State; the Program Committee also included Doug Bates, Dave Henderson, Olivia Lau, and Luke Tierney. Thanks are due to all of them, and to the team of Iowa State students who made numerous trips to the Des Moines airport carrying meeting participants back and forth, and who kept the equipment and facilities running smoothly–useR! 2007 was an excellent meeting.

Duncan Murdoch, University of Western Ontario
Duncan.Murdoch@R-project.org
Martin Maechler, ETH Zürich
Martin.Maechler@R-project.org

Forthcoming Events: useR! 2008

The international R user conference 'useR! 2008' will take place at the Universität Dortmund, Dortmund, Germany, August 12-14, 2008.

This world-meeting of the R user community will focus on

- R as the 'lingua franca' of data analysis and statistical computing,
- providing a platform for R users to discuss and exchange ideas how R can be used to do statistical computations, data analysis, visualization and exciting applications in various fields,
- giving an overview of the new features of the rapidly evolving R project.

The program comprises invited lectures, user-contributed sessions and pre-conference tutorials.

Invited Lectures

R has become the standard computing engine in more and more disciplines, both in academia and the business world. How R is used in different areas will be presented in invited lectures addressing hot topics. Speakers will include *Peter Bühlmann, John Fox, Andrew Gelman, Kurt Hornik, Gary King, Duncan Murdoch, Jean Thioulouse, Graham J. Williams*, and *Keith Worsley*.

User-contributed Sessions

The sessions will be a platform to bring together R users, contributers, package maintainers and developers in the S spirit that 'users are developers'. People from different fields will show us how they solve problems with R in fascinating applications. The scientific program is organized by members of the program committee, including *Micah Altman, Roger Bivand, Peter Dalgaard, Jan de Leeuw, Ramón Díaz-Uriarte, Spencer Graves, Leonhard Held, Torsten Hothorn, François Husson, Christian Kleiber, Friedrich Leisch, Andy Liaw, Martin Mächler, Kate Mullen, Ei-ji Nakama, Thomas Petzold, Martin Theus*, and *Heather Turner*. The program will cover topics of current interest such as

- Applied Statistics & Biostatistics
- Bayesian Statistics
- Bioinformatics
- Chemometrics and Computational Physics
- Data Mining
- Econometrics & Finance
- Environmetrics & Ecological Modeling
- High Performance Computing
- Machine Learning
- Marketing & Business Analytics
- Psychometrics
- Robust Statistics
- Sensometrics
- Spatial Statistics
- Statistics in the Social and Political Sciences
- Teaching
- Visualization & Graphics
- and many more.

Call for pre-conference Tutorials

Before the start of the official program, half-day tutorials will be offered on Monday, August 11.

We invite R users to submit proposals for three hour tutorials on special topics on R. The proposals should give a brief description of the tutorial, including goals, detailed outline, justification why the tutorial is important, background knowledge required and potential attendees. The proposals should be sent before 2007-10-31 to `useR-2008@R-project.org`.

Call for Papers

We invite all R users to submit abstracts on topics presenting innovations or exciting applications of R. A web page offering more information on 'useR! 2008' is available at:

`http://www.R-project.org/useR-2008/`

Abstract submission and registration will start in December 2007.

We hope to meet you in Dortmund!

The organizing committee:
Uwe Ligges, Achim Zeileis, Claus Weihs, Gerd Kopp, Friedrich Leisch, and Torsten Hothorn
`useR-2008@R-project.org`

Forthcoming Events: R Courses in Munich

by Friedrich Leisch

The department of statistics at Ludwig–Maximilians–Universität München, Germany, is offering a range of R courses to practitioners from industry, universities and all others interested in using R for data analysis, starting with R basics (November 8–9, by Torsten Hothorn and Friedrich Leisch) followed by R programming (December 12–13, by Torsten Hothorn and Friedrich Leisch), machine learning (January 23–24, by Torsten Hothorn, Friedrich Leisch and Carolin Strobl), econometrics (spring 2008, by Achim Zeileis), and generalized regression (spring 2008, by Thomas Kneib).

The regular courses are taught in German, more information is available from `http://www.statistik.lmu.de/R/`.

Friedrich Leisch
Ludwig-Maximilians-Universität München, Germany
`Friedrich.Leisch@R-project.org`