Computer algebra in R bridges a gap between mathematics and data in the teaching of statistics and data science

by Mikkel Meyer Andersen and Søren Højsgaard

Abstract The capability of R to do symbolic mathematics is enhanced by the caracas package. This package uses the Python computer algebra library SymPy as a back-end but caracas is tightly integrated in the R environment, thereby enabling the R user with symbolic mathematics within R. We demonstrate how mathematics and statistics can benefit from bridging computer algebra and data via R. This is done thought a number of examples and we propose some topics for small student projects. The caracas package integrates well with e.g. Rmarkdown, and as such creation of scientific reports and teaching is supported.

Introduction

The **caracas** package (Andersen and Højsgaard 2021) and the **Ryacas** package (Andersen and Højsgaard 2019) enhance the capability of R (R Core Team 2023) to handle symbolic mathematics. In this paper we will illustrate the use of the caracas package in connection with teaching mathematics and statistics. Focus is on 1) treating statistical models symbolically, 2) on bridging the gap between symbolic mathematics and numerical computations and 3) on preparing teaching material in a reproducible framework (provided by, e.g. **Rmarkdown** (Allaire et al. 2021; Xie, Allaire, and Grolemund 2018; Xie, Dervieux, and Riederer 2020)). The caracas package is available from CRAN (R Core Team 2023). The open-source development version of caracas is available at https://github.com/r-cas/caracas.

Neither caracas nor Ryacas are as powerful as some of the larger commercial computer algebra systems (CAS). The virtue of caracas and Ryacas lie elsewhere: (1) Mathematical tools like equation solving, summation, limits, symbolic linear algebra, outputting in tex format etc. are directly available from within R. (2) The packages enable working with the same language and in the same environment as the user does for statistical analyses. (3) Symbolic mathematics can easily be combined with data which is helpful in e.g. numerical optimization. (4) The packages are open-source and therefore support e.g. education - also for people with limited economical means and thus contributing to United Nations sustainable development goals, cfr. (United Nations General Assembly 2015).

The paper is organized in the following sections: The section Mathematics and documents containing mathematics briefly introduces the caracas package and its syntax, including how caracas can be used in connection with preparing texts, e.g. teaching material. More details are provided in the appendix. Several vignettes illustrating caracas are provided and they are also available online, see https://r-cas.github.io/caracas/. The section Statistics examples is the main section of the paper and here we present a sample of statistical models where we believe that a symbolic treatment is a valuable supplement to a numerical in connection with teaching. The section Possible topics to study contains suggestions about hand-on activities for students. Lastly, the section Discussion and future work contains a discussion of the paper.

Mathematics and documents containing mathematics

We start by introducing the caracas syntax on familiar topics within calculus and linear algebra.

Calculus

First we define a caracas symbol x (see the appendix) and subsequently a caracas polynomial p in x (p becomes a symbol because x is):

```
R> library(caracas)
R> def_sym(x) ## Declares 'x' as a symbol
R> p <- 1 - x^2 + x^3 + x^4/4 - 3 * x^5 / 5 + x^6 / 6
R> p

#> [c]:  6      5    4
#>      x    3*x    x     3    2
#>      -- - ---- + -- + x  - x  + 1
#>      6     5     4
```

The gradient of `p` is:

```
R> grad <- der(p, x) ## 'der' is shorthand for derivative
R> grad
```

```
#> [c]:  5      4    3      2
#>      x  - 3*x  + x  + 3*x  - 2*x
```

Stationary points of p can be found by finding roots of the gradient. In this simple case we can factor the gradient:

```
R> factor_(grad)
```

```
#> [c]:                  2
#>      x*(x - 2)*(x - 1) *(x + 1)
```

The factorizations shows that stationary points are -1, 0, 1 and 2. To investigate if extreme points are local minima, local maxima or saddle points, we compute the Hessian and evaluate the Hessian in the stationary points:

```
R> hess <- der2(p, x)
R> hess
```

```
#> [c]:    4       3      2
#>      5*x  - 12*x  + 3*x  + 6*x - 2
```

```
R> hess_ <- as_func(hess)
R> hess_
```

```
#> function (x)
#> {
#>     5 * x^4 - 12 * x^3 + 3 * x^2 + 6 * x - 2
#> }
#> <environment: 0x55e67ee87128>
```

```
R> stationary_points <- c(-1, 0, 1, 2)
R> hess_(stationary_points)
```

```
#> [1] 12 -2  0  6
```

The sign of the Hessian in these points gives that $x = -1$ and $x = 12$ are local minima, $x = 0$ is a local maximum and $x = 1$ is a saddle point. In general we can find the stationary symbolically and evaluate the Hessian as follows (output omitted):

```
R> sol <- solve_sys(lhs = grad, vars = x) ## finds roots by default
R> subs(hess, sol[[1]]) ## the first solution
R> lapply(sol, function(s) subs(hess, s)) ## iterate over all solutions
```

Linear algebra

Next, we create a symbolic matrix and find its inverse:

```
R> M <- as_sym(toeplitz(c("a", "b", 0)))
R> Minv <- inv(M) %>% simplify()
```

Default printing of `M` is (`Minv` is shown below in next section):

```
R> M

#> [c]: [a  b  0]
#>      [       ]
#>      [b  a  b]
#>      [       ]
#>      [0  b  a]
```

A vector is a one-column matrix, but it is printed as its transpose to save space:

```
R> v <- vector_sym(3, "v")
R> v

#> [c]: [v1  v2  v3]^T
```

Matrix products are computed using the %*% operator:

```
R> M %*% v

#> [c]: [a*v1 + b*v2  a*v2 + b*v1 + b*v3  a*v3 + b*v2]^T
```

Preparing mathematical documents

The packages Sweave (Leisch 2002) and Rmarkdown (Allaire et al. 2021) provide integration of LaTeX and other text formatting systems into R helping to produce text document with R content. In a similar vein, caracas provides an integration of computer algebra into R and in addition, caracas also facilitates creation of documents with mathematical content without e.g. typing tedious LaTeX instructions.

A LaTeX rendering of the caracas symbol p is obtained by typing `$$p(x) = `r tex(p)`$$` which results in the following when the document is compiled:

$$p(x) = \frac{x^6}{6} - \frac{3x^5}{5} + \frac{x^4}{4} + x^3 - x^2 + 1$$

Typing `$$M^{-1} = `r tex(Minv)`$$` produces the result:

$$M^{-1} = \begin{bmatrix} \frac{a^2-b^2}{a(a^2-2b^2)} & -\frac{b}{a^2-2b^2} & \frac{b^2}{a(a^2-2b^2)} \\ -\frac{b}{a^2-2b^2} & \frac{a}{a^2-2b^2} & -\frac{b}{a^2-2b^2} \\ \frac{b^2}{a(a^2-2b^2)} & -\frac{b}{a^2-2b^2} & \frac{a^2-b^2}{a(a^2-2b^2)} \end{bmatrix}.$$

The determinant of M is $det(M) = a^3 - 2 * a * b^2$ and this can be factored out of the matrix by dividing each entry with the determinant and multiplying the new matrix by the determinant which simplifies the appearence of the matrix:

```
R> Minv_fact <- as_factor_list(1 / factor_(det(M)), simplify(Minv * det(M)))
```

Typing `$$M^{-1} = `r tex(Minv_fact)`$$` produces this:

$$M^{-1} = \frac{1}{a\left(a^2 - 2b^2\right)} \begin{bmatrix} a^2 - b^2 & -ab & b^2 \\ -ab & a^2 & -ab \\ b^2 & -ab & a^2 - b^2 \end{bmatrix}.$$

Finally we illustrate creation of additional mathematical expressions:

```
R> def_sym(x, n)
R> y <- (1 + x/n)^n
R> lim(y, n, Inf)

#> [c]: exp(x)
```

Typing `$$y = `r tex(y)`$$` etc. gives

$$y = \left(1 + \frac{x}{n}\right)^n, \lim_{n->\infty} y = exp(x).$$

We can also prepare unevaluated expressions using the `doit` argument. That helps making reproducible documents where the changes in code appears automatically in the generated formulas. This is done as follows:

```
R> l <- lim(y, n, Inf, doit = FALSE)
R> l
```

```
#> [c]:             n
#>          /    x\
#>       lim |1 + -|
#>      n->oo\    n/
```

```
R> doit(l)
```

```
#> [c]: exp(x)
```

Typing `$$`r tex(l)` = `r tex(doit(l))`$$` gives

$$\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n = e^x.$$

Several functions have the `doit` argument, e.g. `lim()`, `int()` and `sum_()`.

Statistics examples

In this section we examine larger statistical examples and demonstrate how caracas can help improve understanding of the models.

Linear models

A matrix algebra approach to e.g. linear models is very clear and concise. On the other hand, it can also be argued that matrix algebra obscures what is being computed. Numerical examples are useful for some aspects of the computations but not for other. In this respect symbolic computations can be enlightening.

Consider a two-way analysis of variance (ANOVA) with one observation per group, see Table 1.

Table 1: Two-by-two layout of data.

y_{11}	y_{21}
y_{12}	y_{22}

```
R> nr <- 2
R> nc <- 2
R> y <- matrix_sym(nr, nc, "y")
R> dim(y) <- c(nr*nc, 1)
R> y
```

```
#> [c]: [y11  y21  y12  y22]^T
```

```
R> dat <- expand.grid(r=factor(1:nr), s=factor(1:nc))
R> X <- model.matrix(~r+s, data=dat) |> as_sym()
R> b <- vector_sym(ncol(X), "b")
R> mu <- X %*% b
```

For the specific model we have random variables $y = (y_{ij})$. All y_{ij}s are assumed independent and $y_{ij} \sim N(\mu_{ij}, v)$. The corrensponding mean vector μ has the form given below:

$$y = \begin{bmatrix} y_{11} \\ y_{21} \\ y_{12} \\ y_{22} \end{bmatrix}, \quad X = \begin{bmatrix} 1 & . & . \\ 1 & 1 & . \\ 1 & . & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad \mu = Xb = \begin{bmatrix} b_1 \\ b_1 + b_2 \\ b_1 + b_3 \\ b_1 + b_2 + b_3 \end{bmatrix}.$$

Above and elsewhere, dots represent zero. The least squares estimate of b is the vector $\hat{b}$ that minimizes $||y - Xb||^2$ which leads to the normal equations $(X^\top X)b = X^\top y$ to be solved. If X has full rank, the unique solution to the normal equations is $\hat{b} = (X^\top X)^{-1} X^\top y$. Hence the estimated mean vector is $\hat{\mu} = X\hat{b} = X(X^\top X)^{-1} X^\top y$. Symbolic computations are not needed for quantities involving only the model matrix X, but when it comes to computations involving y, a symbolic treatment of y is useful:

```
R> XtX <- t(X) %*% X
R> XtXinv <- inv(XtX)
R> Xty <- t(X) %*% y
R> b_hat <- XtXinv %*% Xty
```

$$X^\top y = \begin{bmatrix} y_{11} + y_{12} + y_{21} + y_{22} \\ y_{21} + y_{22} \\ y_{12} + y_{22} \end{bmatrix}, \quad \hat{b} = \frac{1}{4} \begin{bmatrix} 3y_{11} + y_{12} + y_{21} - y_{22} \\ -2y_{11} - 2y_{12} + 2y_{21} + 2y_{22} \\ -2y_{11} + 2y_{12} - 2y_{21} + 2y_{22} \end{bmatrix} \tag{1}$$

Hence $X^\top y$ (a sufficient reduction of data if the variance is known) consists of the sum of all observations, the sum of observations in the second row and the sum of observations in the second column. For $\hat{b}$, the second component is, apart from a scaling, the sum of the second row minus the sum of the first row. Likewise, the third component is the sum of the second column minus the sum of the first column. It is hard to give an interpretation of the first component of $\hat{b}$.

Logistic regression

In the following we go through details of a logistic regression model, see e.g. (McCullagh and Nelder 1989) for a classical description of logistic regression: Observables are binomially distributed, $y_i \sim \text{bin}(p_i, n_i)$. The probability p_i is connected to a a q-vector of covariates $x_i = (x_{i1}, \dots, x_{iq})$ and a q-vector of regression coefficients $b = (b_1, \dots, b_q)$ as follows: The term $s_i = x_i \cdot b$ is denoted the *linear predictor*. The probability p_i can be linked to s_i in different ways, but the most commonly employed is via the *logit link function* which is $\text{logit}(p_i) = \log(p_i/(1 - p_i))$ so here $\text{logit}(p_i) = s_i$.

As an example, consider the `budworm` data from the **doBy** package (Højsgaard and Halekoh 2023). The data shows the number of killed moth tobacco budworm *Heliothis virescens*. Batches of 20 moths of each sex were exposed for three days to the pyrethroid and the number in each batch that were dead or knocked down was recorded:

```
R> data(budworm, package = "doBy")
R> bud <- subset(budworm, sex == "male")
R> bud
```

```
#>    sex dose ndead ntotal
#> 1 male    1     1     20
#> 2 male    2     4     20
#> 3 male    4     9     20
#> 4 male    8    13     20
#> 5 male   16    18     20
#> 6 male   32    20     20
```

Below we focus only on male budworms and the mortality is illustrated in Figure 1 (produced with **ggplot2** (Wickham 2016)). On the y-axis we have the empirical logits, i.e. $\log((\text{ndead} + 0.5)/(\text{ntotal} - \text{ndead} + 0.5))$. The figure suggests that logit grows linearly with log dose.

Each component of the likelihood

The log-likelihood is $\log L = \sum_i y_i \log(p_i) + (n_i - y_i)\log(1 - p_i) = \sum_i \log L_i$, say. With $\log(p_i/(1 - p_i)) = s_i$ we have $p_i = 1/(1 + \exp(-s_i))$ and $\frac{d}{ds_i} p_i = \frac{\exp(-s_i)}{(1+\exp(-s_i))^2}$. With $s_i = x_i \cdot b$, we have $\frac{d}{db} s_i = x_i$.

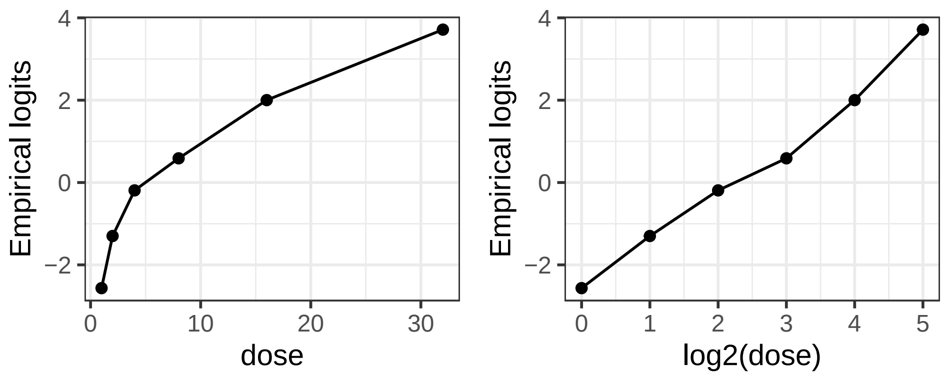

Figure 1: Insecticide mortality of the moth tobacco budworm.

Consider the contribution to the total log-likelihood from the ith observation which is $l_i = y_i \log(p_i) + (n_i - y_i)\log(1 - p_i)$. Since we are focusing on one observation only, we shall ignore the subscript i in this section. First notice that with $s = \log(p/(1-p))$ we can find p as:

```
R> def_sym(s, p)
R> sol_ <- solve_sys(lhs = log(p / (1 - p)), rhs = s, vars = p)
R> sol_[[1]]$p

#> [c]:   exp(s)
#>      ----------
#>      exp(s) + 1
```

Next, find the likelihood as a function of p, as a function of s and as a function of b. The underscore in `logLb_` and elsewhere indicates that this expression is defined in terms of other symbols (this is in contrast to the free variables, e.g. `y`, `p`, and `n`.):

```
R> def_sym(y, n, p, x, s, b)
R> logLp_ <- y * log(p) + (n - y) * log(1 - p)
R> p_ <- exp(s) / (exp(s) + 1)
R> logLs_ <- subs(logLp_, p, p_)
R> s_ <- sum(x * b)
R> logLb_ <- subs(logLs_, s, s_)
R> logLb_

#> [c]:      /  exp(b*x)  \             /      exp(b*x)  \
#>      y*log|------------| + (n - y)*log|1 - ------------|
#>           \exp(b*x) + 1/             \    exp(b*x) + 1/
```

The log-likelihood can be maximized using e.g. Newton-Rapson and in this connection we need the score function, S, and the Hessian, H:

```
R> Sb_ <- score(logLb_, b) |> simplify()
R> Hb_ <- hessian(logLb_, b) |> simplify()
R> Sb_

#> [c]: [x*(y - (n - y)*exp(b*x))]
#>      [------------------------]
#>      [      exp(b*x) + 1      ]

R> Hb_

#> [c]: [          2              ]
#>      [      -n*x *exp(b*x)       ]
#>      [---------------------------]
#>      [exp(2*b*x) + 2*exp(b*x) + 1]
```

Since x and b are vectors, the term `b*x` above should be read as the inner product $x \cdot b$ (or as $x^\top b$ in matrix notation). Also, since x is a vector, the term `x^2` above should be read as the outer product $x \otimes x$ (or as $xx^\top$ in matrix notation). More insight in the structure is obtained by letting b and x be 2-vectors (to save space, the Hessian matrix is omitted in the following):

```
R> b <- vector_sym(2, "b")
R> x <- vector_sym(2, "x")
R> s_ <- sum(x * b)
R> logLb_ <- subs(logLs_, s, s_)
R> Sb_ <- score(logLb_, b) |> simplify()
```

$$\texttt{logLb\_} = y \log\left(\frac{e^{b_1x_1+b_2x_2}}{e^{b_1x_1+b_2x_2}+1}\right) + (n-y)\log\left(1-\frac{e^{b_1x_1+b_2x_2}}{e^{b_1x_1+b_2x_2}+1}\right), \quad (2)$$

$$\texttt{Sb\_} = \begin{bmatrix} \frac{x_1\left(-ne^{b_1x_1+b_2x_2}+ye^{b_1x_1+b_2x_2}+y\right)}{e^{b_1x_1+b_2x_2}+1} \\ \frac{x_2\left(-ne^{b_1x_1+b_2x_2}+ye^{b_1x_1+b_2x_2}+y\right)}{e^{b_1x_1+b_2x_2}+1} \end{bmatrix}. \quad (3)$$

Next, insert data, e.g. $x_1 = 1$, $x_2 = 2$, $y = 9$, $n = 20$ to obtain a function of the regression parameters only. Note how the expression depending on other symbols, `S_`, is named `S.` to indicate that data has been inserted:

```
R> nms <- c("x1", "x2", "y", "n")
R> vls <- c(1, 2, 9, 20)
R> logLb. <- subs(logLb_, nms, vls)
R> Sb. <- subs(Sb_, nms, vls)
```

The total score for the entire dataset can be obtained as follows:

```
R> Sb_list <- lapply(seq_len(nrow(bud)), function(r){
+   vls <- c(1, log2(bud$dose[r]), bud$ndead[r], bud$ntotal[r])
+   subs(Sb_, nms, vls)
+ })
R> Sb_total <- Reduce(`+`, Sb_list)
```

This score can be used as part of an iterative algorithm for solving the score equations. If one wants to use Newton-Rapson, the total Hessian matrix must also be created following lines similar to those above. It is straight forward implement a Newton-Rapson algorithm based on these quantities, one must only note the distinction between the two expressions below (and it is the latter one would use in an iterative algorithm):

```
R> subs(Sb_total, b, c(1, 2))
R> subs(Sb_total, b, c(1, 2)) |> as_expr()
```

An alternative is to construct the total log-likelihood for the entire dataset as a `caracas` object, convert this object to an `R` function and maximize this function using one of `R`'s optimization methods:

```
R> logLb_list <- lapply(seq_len(nrow(bud)), function(r){
+   vls <- c(1, log2(bud$dose[r]), bud$ndead[r], bud$ntotal[r])
+   subs(logLb_, nms, vls)
+ })
R> logLb_total <- Reduce(`+`, logLb_list)
R> logLb_total_func <- as_func(logLb_total, vec_arg = TRUE)
```

The total likelihood symbolically

We conclude this section by illustrating that the log-likelihood for the entire dataset can be constructed in a few steps (output is omitted to save space):

```
R> X. <- as_sym(cbind(1, log2(bud$dose)))
R> n. <- as_sym(bud$ntotal)
```

```
R> y. <- as_sym(bud$ndead)
R> N <- nrow(X.)
R> q <- ncol(X.)
R> X <- matrix_sym(N, q, "x")
R> n <- vector_sym(N, "n")
R> y <- vector_sym(N, "y")
R> p <- vector_sym(N, "p")
R> s <- vector_sym(N, "s")
R> b <- vector_sym(q, "b")
```

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \\ x_{51} & x_{52} \\ x_{61} & x_{62} \end{bmatrix}, \quad X. = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix}, \quad n. = \begin{bmatrix} 20 \\ 20 \\ 20 \\ 20 \\ 20 \\ 20 \end{bmatrix}, \quad n = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \end{bmatrix}, \quad y. = \begin{bmatrix} 1 \\ 4 \\ 9 \\ 13 \\ 18 \\ 20 \end{bmatrix}.$$

The symbolic computations are as follows:

```
R> ## log-likelihood as function of p
R> logLp  <- sum(y * log(p) + (n-y) * log(1-p))
R> ## log-likelihood as function of s
R> p_ <- exp(s) / (exp(s) + 1)
R> logLs <- subs(logLp, p, p_)
R> ## linear predictor as function of regression coefficients:
R> s_  <- X %*% b
R> ## log-Likelihood as function of regression coefficients:
R> logLb <- subs(logLs, s, s_)
```

Next, numerical values can be inserted:

```
R> logLb <- subs(logLb, cbind(n, y, X), cbind(n., y., X.))
```

An alternative would have been to define logLp above in terms of n. and y. and similarly define s_ in terms of X. If doing so, the last step where numerical values are inserted could have been avoided. From here, one may proceed by computing the score function and the Hessian matrix and solve the score equation, using e.g. Newton-Rapson. Alternatively, one might create an R function based on the log-likelihood, and maximize this function using one of R's optimization methods (see the example in the previous section):

```
R> logLb_func <- as_func(logLb, vec_arg = TRUE)
R> optim(c(0, 0), logLb_func, control = list(fnscale = -1), hessian = TRUE)
```

Maximum likelihood under constraints

In this section we illustrate constrained optimization using Lagrange multipliers. This is demonstrated for the independence model for a two-way contingency table. Consider a 2×2 contingency table with cell counts y_{ij} and cell probabilities p_{ij} for $i = 1, 2$ and $j = 1, 2$, where i refers to row and j to column as illustrated in Table 1.

Under multinomial sampling, the log likelihood is

$$l = \log L = \sum_{ij} y_{ij} \log(p_{ij}).$$

Under the assumption of independence between rows and columns, the cell probabilities have the form, (see e.g. (Højsgaard, Edwards, and Lauritzen 2012), p. 32)

$$p_{ij} = u \cdot r_i \cdot s_j.$$

To make the parameters (u, r_i, s_j) identifiable, constraints must be imposed. One possibility is to require that $r_1 = s_1 = 1$. The task is then to estimate u, r_2, s_2 by maximizing the log likelihood under the constraint that $\sum_{ij} p_{ij} = 1$. This can be achieved using a Lagrange multiplier where we instead solve the unconstrained optimization problem $\max_p Lag(p)$ where

$$Lag(p) = -l(p) + \lambda g(p) \quad \text{under the constraint that} \tag{4}$$
$$g(p) = \sum_{ij} p_{ij} - 1 = 0, \tag{5}$$

where λ is a Lagrange multiplier. In `SymPy`, `lambda` is a reserved symbol. Hence the underscore as postfix below:

```
R> y_ <- c("y_11", "y_21", "y_12", "y_22")
R> y  <- as_sym(y_)
R> def_sym(u, r2, s2, lambda_)
R> p <- as_sym(c("u", "u*r2", "u*s2", "u*r2*s2"))
R> logL  <- sum(y * log(p))
R> Lag  <- -logL + lambda_ * (sum(p) - 1)
R> vars <- list(u, r2, s2, lambda_)
R> gLag <- der(Lag, vars)
R> sol <- solve_sys(gLag, vars)
R> print(sol, method = "ascii")

#> Solution 1:
#>   u       =  (y_11 + y_12)*(y_11 + y_21)/(y_11 + y_12 + y_21 + y_22)^2
#>   r2      =  (y_21 + y_22)/(y_11 + y_12)
#>   s2      =  (y_12 + y_22)/(y_11 + y_21)
#>   lambda_ =  y_11 + y_12 + y_21 + y_22

R> sol <- sol[[1]]
```

There is only one critical point. Fitted cell probabilities $\hat{p}_{ij}$ are:

```
R> p11 <- sol$u
R> p21 <- sol$u * sol$r2
R> p12 <- sol$u * sol$s2
R> p22 <- sol$u * sol$r2 * sol$s2
R> p.hat <- matrix_(c(p11, p21, p12, p22), nrow = 2)
```

$$\hat{p} = \frac{1}{(y_{11} + y_{12} + y_{21} + y_{22})^2} \begin{bmatrix} (y_{11} + y_{12})(y_{11} + y_{21}) & (y_{11} + y_{12})(y_{12} + y_{22}) \\ (y_{11} + y_{21})(y_{21} + y_{22}) & (y_{12} + y_{22})(y_{21} + y_{22}) \end{bmatrix}$$

To verify that the maximum likelihood estimate has been found, we compute the Hessian matrix which is negative definite (the Hessian matrix is diagonal so the eigenvalues are the diagonal entries and these are all negative), output omitted:

```
R> H <- hessian(logL, list(u, r2, s2)) |> simplify()
```

An $AR(1)$ model

Symbolic computations

In this section we study the auto regressive model of order 1 (an AR(1) model), see e.g. (Shumway and Stoffer 2016), p. 75 ff. for details: Consider random variables $x_1, x_2, \dots, x_n$ following a stationary zero mean AR(1) process:

$$x_i = a x_{i-1} + e_i; \quad i = 2, \dots, n, \tag{6}$$

where $e_i \sim N(0, v)$ and all e_is are independent. Note that v denotes the variance. The marginal distribution of x_1 is also assumed normal, and for the process to be stationary we must have that the variance $\mathbf{Var}(x_1) = v/(1 - a^2)$. Hence we can write $x_1 = \frac{1}{\sqrt{1-a^2}} e_1$.

For simplicity of exposition, we set $n = 4$. All terms $e_1, \dots, e_4$ are independent and $N(0, v)$ distributed. Let $e = (e_1, \dots, e_4)$ and $x = (x_1, \dots x_4)$. Hence $e \sim N(0, vI)$. Isolating error terms in (6) gives

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \begin{bmatrix} \sqrt{1-a^2} & . & . & . \\ -a & 1 & . & . \\ . & -a & 1 & . \\ . & . & -a & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = Lx.$$

Since $\mathbf{Var}(e) = vI$ we have $\mathbf{Var}(e) = vI = L\mathbf{Var}(x)L'$ so the covariance matrix of x is $V = \mathbf{Var}(x) = vL^-(L^-)^\top$ while the concentration matrix (the inverse covariance matrix) is $K = v^{-1}L^\top L$:

```
R> # FIXME: Added ->
R> n <- 4
R> L <- diff_mat(n, "-a")
R> def_sym(a)
R> L[1, 1] <- sqrt(1-a^2)
R> # <-
R> def_sym(v)
R> Linv <- inv(L)
R> K <- crossprod_(L) / v
R> V <- tcrossprod_(Linv) * v
```

$$L^{-1} = \begin{bmatrix} \frac{1}{\sqrt{1-a^2}} & . & . & . \\ \frac{a}{\sqrt{1-a^2}} & 1 & . & . \\ \frac{a^2}{\sqrt{1-a^2}} & a & 1 & . \\ \frac{a^3}{\sqrt{1-a^2}} & a^2 & a & 1 \end{bmatrix}, \tag{7}$$

$$K = \frac{1}{v}\begin{bmatrix} 1 & -a & . & . \\ -a & a^2+1 & -a & . \\ . & -a & a^2+1 & -a \\ . & . & -a & 1 \end{bmatrix}, \tag{8}$$

$$V = v\begin{bmatrix} \frac{1}{1-a^2} & \frac{a}{1-a^2} & \frac{a^2}{1-a^2} & \frac{a^3}{1-a^2} \\ \frac{a}{1-a^2} & \frac{a^2}{1-a^2}+1 & \frac{a^3}{1-a^2}+a & \frac{a^4}{1-a^2}+a^2 \\ \frac{a^2}{1-a^2} & \frac{a^3}{1-a^2}+a & \frac{a^4}{1-a^2}+a^2+1 & \frac{a^5}{1-a^2}+a^3+a \\ \frac{a^3}{1-a^2} & \frac{a^4}{1-a^2}+a^2 & \frac{a^5}{1-a^2}+a^3+a & \frac{a^6}{1-a^2}+a^4+a^2+1 \end{bmatrix}. \tag{9}$$

The zeros in the concentration matrix K implies a conditional independence restriction: If the ijth element of a concentration matrix is zero then x_i and x_j are conditionally independent given all other variables, see e.g. (Højsgaard, Edwards, and Lauritzen 2012), p. 84 for details.

Next, we take the step from symbolic computations to numerical evaluations. The joint distribution of x is multivariate normal distribution, $x \sim N(0, K^{-1})$. Let $W = xx^\top$ denote the matrix of (cross) products. The log-likelihood is therefore (ignoring additive constants)

$$\log L = \frac{n}{2}(\log \mathbf{det}(K) - x^\top Kx) = \frac{n}{2}(\log \mathbf{det}(K) - \mathbf{tr}(KW)),$$

where we note that $\mathbf{tr}(KW)$ is the sum of the elementwise products of K and W since both matrices are symmetric. Ignoring the constant $\frac{n}{2}$, this can be written symbolically to obtain the expression in this particular case:

```
R> x <- vector_sym(n, "x") # FIXME: ADDED
R> logL <- log(det(K)) - sum(K * (x %*% t(x))) %>% simplify()
```

$$\log L = \log\left(-\frac{a^2}{v^4} + \frac{1}{v^4}\right) - \frac{-2ax_1x_2 - 2ax_2x_3 - 2ax_3x_4 + x_1^2 + x_2^2\left(a^2+1\right) + x_3^2\left(a^2+1\right) + x_4^2}{v}.$$

Numerical evaluation

Next we illustrate how bridge the gap from symbolic computations to numerical computations based on a dataset: For a specific data vector we get:

```
R> xt <- c(0.1, -0.9, 0.4, .0)
R> logL. <- subs(logL, x, xt)
```

$$\log L = \log\left(-\frac{a^2}{v^4} + \frac{1}{v^4}\right) - \frac{0.97a^2 + 0.9a + 0.98}{v}.$$

We can use R for numerical maximization of the likelihood and constraints on the parameter values can be imposed e.g. in the `optim()` function:

```
R> logL_wrap <- as_func(logL., vec_arg = TRUE)
R> eps <- 0.01
R> par <- optim(c(a=0, v=1), logL_wrap,
+              lower=c(-(1-eps), eps), upper=c((1-eps), 10),
+              method="L-BFGS-B", control=list(fnscale=-1))$par
R> par

#>      a      v
#> -0.376  0.195
```

The same model can be fitted e.g. using R's `arima()` function as follows (output omitted):

```
R> arima(xt, order = c(1, 0, 0), include.mean = FALSE, method = "ML")
```

It is less trivial to do the optimization in caracas by solving the score equations. There are some possibilities for putting assumptions on variables in caracas (see the "Reference" vignette), but it is not possible to restrict the parameter a to only take values in $(-1, 1)$.

Variance of the average of correlated data

```
R> R <- as_sym(toeplitz(rep(paste0("r^", 0:n))))
```

$$V = vR = v\begin{bmatrix}1 & r & r^2 & r^3\\ r & 1 & r & r^2\\ r^2 & r & 1 & r\\ r^3 & r^2 & r & 1\end{bmatrix}. \tag{10}$$

Consider random variables $x_1, \dots, x_n$ where $\mathbf{Var}(x_i) = v$ and $\mathbf{Cov}(x_i, x_j) = vr$ for $i \neq j$, where $0 \leq |r| \leq 1$. For $n = 3$, the covariance matrix of $(x_1, \dots, x_n)$ is therefore

$$V = vR = v\begin{bmatrix}1 & r & r\\ r & 1 & r\\ r & r & 1\end{bmatrix}. \tag{11}$$

Let $\bar{x} = \sum_i x_i/n$ denote the average. Suppose interest is in the variance of the average, $\mathbf{Var}(\bar{x})$, when n goes to infinity. One approach is as follow: Let 1 denote an n-vector of 1's and let V be an $n \times n$ matrix with v on the diagonal and vr outside the diagonal. Then $\mathbf{Var}(\bar{x}) = \frac{1}{n^2}1^\top V1$. The answer lies in studying the limiting behaviour of this expression when $n \to \infty$ and caracas can not handle this directly, so some work must be done by hand: The variance of a sum $x. = \sum_i x_i$ is $\mathbf{Var}(x.) = \sum_i \mathbf{Var}(x_i) + 2\sum_{ij:i<j}\mathbf{Cov}(x_i, x_j)$. For the specific model, one finds that:

$$\mathbf{Var}(x.) = nv + 2vrn(n-1)/2 = nv(1 + r(n-1)), \quad \mathbf{Var}(\bar{x}) = v(1 + (n-1)r)/n.$$

```
R> def_sym(v, r, n)
R> var_sum <- n * v * ( 1 + r * (n - 1))
R> var_avg <- var_sum / n^2
R> var_avg %>% simplify()

#> [c]: v*(r*(n - 1) + 1)
#>      -----------------
#>              n
```

From hereof, we can study the limiting behavior of the variance $\mathbf{Var}(\bar{x})$ in different situations:

```
R> l_1 <- lim(var_avg, n, Inf)         ## when sample size n goes to infinity
R> l_2 <- lim(var_avg, r, 0, dir='+')  ## when correlation r goes to zero
R> l_3 <- lim(var_avg, r, 1, dir='-')  ## when correlation r goes to one
```

For a given correlation r it is instructive to investigate how many independent variables k the n correlated variables correspond to (in the sense of the same variance of the average), because the k can be seen as a measure of the amount of information in data. Moreover, one might study how k behaves as function of n when $n \to \infty$. That is we must (1) solve $v(1 + (n-1)r)/n = v/k$ for k and (2) find $\lim_{n\to\infty} k$:

```
R> def_sym(k)
R> k <- solve_sys(var_avg - v / k, k)[[1]]$k
R> l_k <- lim(k, n, Inf)
```

The findings above are:

$$l_1 = rv, \quad l_2 = \frac{v}{n}, \quad l_3 = v, \quad k = \frac{n}{nr - r + 1}, \quad l_k = \frac{1}{r}.$$

With respect to k, it is illustrative to supplement the symbolic computations above with numerical evaluations, which shows that even a moderate correlation reduces the effective sample size substantially:

```
R> dat <- expand.grid(r=c(.1, .2, .5), n=c(10, 50))
R> k_fun <- as_func(k)
R> dat$k <- k_fun(r=dat$r, n=dat$n)
R> dat$ri <- 1/dat$r
R> dat
```

```
#>     r  n    k ri
#> 1 0.1 10 5.26 10
#> 2 0.2 10 3.57  5
#> 3 0.5 10 1.82  2
#> 4 0.1 50 8.47 10
#> 5 0.2 50 4.63  5
#> 6 0.5 50 1.96  2
```

Possible topics to study

Discussion and future work

We have presented the caracas package and argued that the package extends the functionality of R significantly with respect to symbolic mathematics. One practical virtue of caracas is that the package integrates nicely with Rmarkdown, Allaire et al. (2021), (e.g. with the tex() functionality) and thus supports creating of scientific documents and teaching material. As for the usability in practice we await feedback from users.

Another related package we mentioned in the introduction is Ryacas. This package has existed for many years and is still of relevance. Ryacas probably has fewer features than caracas. On the other hand, Ryacas does not require Python (it is compiled), is faster for some computations (like matrix inversion). Finally, the Yacas language (A. Z. Pinkus and Winitzki 2002; A. Pinkus, Winnitzky, and Mazur 2016) is extendable (see e.g. the vignette "User-defined yacas rules" in the Ryacas package).

One possible future development could be an R package which is designed without a view towards the underlying engine (SymPy or Yacas) and which then draws more freely from SymPy and Yacas.

Lastly, with respect to freely available resources in a CAS context, we would like to draw attention to WolframAlpha, see https://www.wolframalpha.com/, which provides an online service for answering (mathematical) queries.

Acknowledgements

We would like to thank the R Consortium for financial support for creating the caracas package, users for pin pointing points that can be improved in caracas and Ege Rubak (Aalborg University, Denmark) and Malte Bødkergaard Nielsen (Aalborg University, Denmark) for comments on this manuscript.

References

Allaire, JJ, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. 2021. *Rmarkdown: Dynamic Documents for r*. https://github.com/rstudio/rmarkdown.

Andersen, Mikkel Meyer, and Søren Højsgaard. 2019. "Ryacas: A computer algebra system in R." *Journal of Open Source Software* 4 (42). https://doi.org/10.21105/joss.01763.

———. 2021. "Caracas: Computer Algebra in r." *Journal of Open Source Software* 6 (63): 3438. https://doi.org/10.21105/joss.03438.

Højsgaard, Søren, David Edwards, and Steffen Lauritzen. 2012. *Graphical Models with R*. New York: Springer. https://doi.org/10.1007/978-1-4614-2299-0.

Højsgaard, Søren, and Ulrich Halekoh. 2023. *doBy: Groupwise Statistics, LSmeans, Linear Estimates, Utilities*. https://github.com/hojsgaard/doby.

Leisch, Friedrich. 2002. "Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis." In *Compstat*, edited by Wolfgang Härdle and Bernd Rönz, 575–80. Heidelberg: Physica-Verlag HD.

McCullagh, P., and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. London: Chapman Hall.

Pinkus, Ayal Z., and Serge Winitzki. 2002. "YACAS: A Do-It-Yourself Symbolic Algebra Environment." In *Proceedings of the Joint International Conferences on Artificial Intelligence, Automated Reasoning, and Symbolic Computation*, 332–36. AISC '02/Calculemus '02. London, UK, UK: Springer-Verlag. https://doi.org/10.1007/3-540-45470-5_29.

Pinkus, Ayal, Serge Winnitzky, and Grzegorz Mazur. 2016. "Yacas - yet Another Computer Algebra System." https://yacas.readthedocs.io/en/latest/.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. http://www.R-project.org/.

Shumway, Robert H., and David S. Stoffer. 2016. *Time Series Analysis and Its Applications*. Fourth Edition. Springer.

United Nations General Assembly. 2015. "Sustainable Development Goals."

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Xie, Yihui, J. J. Allaire, and Garrett Grolemund. 2018. *R Markdown: The Definitive Guide*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown.

Xie, Yihui, Christophe Dervieux, and Emily Riederer. 2020. *R Markdown Cookbook*. Boca Raton, Florida: Chapman; Hall/CRC. https://bookdown.org/yihui/rmarkdown-cookbook.

Mikkel Meyer Andersen
Department of Mathematical Sciences, Aalborg University, Denmark
Skjernvej 4A
9220 Aalborg Ø, Denmark
ORCiD: 0000-0002-0234-0266
mikl@math.aau.dk

Søren Højsgaard
Department of Mathematical Sciences, Aalborg University, Denmark
Skjernvej 4A
9220 Aalborg Ø, Denmark
ORCiD: 0000-0002-3269-9552
sorenh@math.aau.dk