Bridging a gap between mathematics and data in teaching data science and statistics with the R package caracas for computer algebra

Mikkel Meyer Andersen and Søren Højsgaard

1 Introduction

The capability of R [R Core Team, 2021] to handle symbolic mathematics is enhanced by two add-on packages: The caracas package [Andersen and Højsgaard, 2021] and the Ryacas package [Andersen and Højsgaard, 2019]. In this paper we will illustrate the use of the caracas package in connection with teaching mathematics and statistics, where symbolic mathematics is helpful, strongly aided by the packages' ability to enter in a reproducible framework (provided by, e.g. Rmarkdown [Allaire et al., 2021, Xie et al., 2018, 2020]). 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. The caracas package is available from CRAN [R Core Team, 2021]. 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 as follows:[1] Sec. 2 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 (Sec. A). Several vignettes illustrating caracas are provided and they are also available online, see https://r-cas.github.io/caracas/. Sec. 3 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. Sec. 4 contains suggestions about hand-on activities for students. Lastly, Sec. 5 contains a discussion of the paper.

2 Mathematics and documents containing mathematics

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

2.1 Calculus

First we define a caracas symbol x (see Sec. A) and subsequently p to be a polynomial 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
```

[1] FINISH LATER

```
##    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)
```

which 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: 0x55a187c00258>
```

```
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 (output omitted)

```
R> sol <- solve_sys(grad, 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
```

2.2 Linear algebra

Create a symbolic matrix and its inverse:

```
R> M <- matrix_sym(nrow = 2, ncol = 2, entry = "m")
R> Minv <- inv(M)
```

Default printing of `M` is

```
R> M
```

```
## c: [m11  m12]
##    [       ]
##    [m21  m22]
```

Matrix products are computed using the `%*%` operator:

```
R> simplify(M %*% Minv)
```

```
## c: [1  0]
##    [    ]
##    [0  1]
```

We need to verify the matrix and vector dimensions:

```
R> v <- vector_sym(2, "v")
R> v ## Not the transpose, v is a column vector
```

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

```
R> dim(v)
```

```
## [1] 2 1
```

```
R> M %*% v
```

```
## c: [m11*v1 + m12*v2  m21*v1 + m22*v2]^T
```

2.3 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 `$$p(x) = `r tex(p)`$$`

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

but typing `$$M = `r tex(M)`$$` produces the result

$$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

The inverse of M contains the determinant of M as denominator in each entry. This can be exploited as

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

```
## c:          1          [m22   -m12]
##    -----------------*[           ]
##    m11*m22 - m12*m21 [-m21  m11 ]
```

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

$$M^{-1} = \frac{1}{m_{11}m_{22} - m_{12}m_{21}} \begin{bmatrix} m_{22} & -m_{12} \\ -m_{21} & m_{11} \end{bmatrix}$$

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 makes output easier and more robust:

```
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_()`.

3 Statistics examples

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

3.1 Linear models – one way analysis of variance

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 one-way analysis of variance (ANOVA) with three groups and two replicates per group.

```
R> n_grp <- 3 # Number of groups
R> n_rpg <- 2 # Number of replicates per group
R> dat <- expand.grid(rep=1:n_rpg, grp=paste0("g", 1:n_grp))
R> X <- as_sym(model.matrix(~ grp, data = dat))
R> y <- vector_sym(nrow(X), "y")
R> b <- vector_sym(n_grp, "b")
R> mu <- X %*% b
```

For the specific model we have random variables $y_1, \ldots y_n$ where $n = 6$. All y_is are assumed independent and $y_i \sim N(\mu_i, v)$. The mean vector $\mu = (\mu_1, \ldots, \mu_n)$ has the form given below (dots represent zero).

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix}, \quad X = \begin{bmatrix} 1 & . & . \\ 1 & . & . \\ 1 & 1 & . \\ 1 & 1 & . \\ 1 & . & 1 \\ 1 & . & 1 \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}, \quad \mu = Xb = \begin{bmatrix} b_1 \\ b_1 \\ b_1 + b_2 \\ b_1 + b_2 \\ b_1 + b_3 \\ b_1 + b_3 \end{bmatrix}$$

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
R> y_hat <- X %*% b_hat
```

$$X^\top y = \begin{bmatrix} y_1+y_2+y_3+y_4+y_5+y_6 \\ y_3+y_4 \\ y_5+y_6 \end{bmatrix}, \quad \hat{b} = \frac{1}{2}\begin{bmatrix} y_1+y_2 \\ -y_1-y_2+y_3+y_4 \\ -y_1-y_2+y_5+y_6 \end{bmatrix}, \quad \hat{y} = \frac{1}{2}\begin{bmatrix} y_1+y_2 \\ y_1+y_2 \\ y_3+y_4 \\ y_3+y_4 \\ y_5+y_6 \\ y_5+y_6 \end{bmatrix},$$

Hence $X^\top y$ consists of the sum of all observations, the sum of observations in group 2 and the sum of observations in group 3. Similarly, $\hat{b}$ consists of the average in group 1, the average in group 2 minus the average in group 1 and the average in group 3 minus the average in group 1. Fitted values are simply group averages. The orthogonal projection matrix onto the column space of X is:

```
R> P <- X %*% XtXinv %*% t(X)
```

3.2 Logistic regression

In the following we go through details of a logistic regression model, see e.g. [McCullagh and Nelder] 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 $x_i = (x_{i1}, \ldots, x_{iq})$ and a q-vector of regression coefficients $b = (b_1, \ldots, b_q)$ as follows: Define $s_i = x_i \cdot b$ to be the linear predictor. The probability p_i can be related to s_i in different ways, but the most commonly employed is as $\text{logit}(p_i) = \log(p_i/(1-p_i)) = s_i$.

As an example, consider the `budworm` data from the `doBy` package. 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 Fig. 1. On the y-axis we have the empirical logits, i.e. $\log(\text{ndead} + 0.5/(\text{ntotal} - \text{ndead} + 0.5))$.

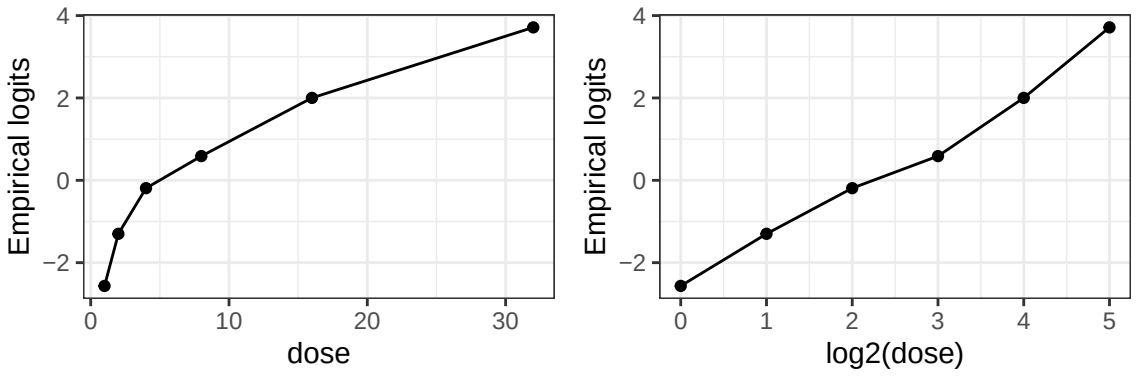

Figure 1: Insecticide mortality of the moth tobacco budworm.

3.2.1 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 $\frac{d}{db} s_i = x_i$.

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. The log-likelihood and its derivative are:

```
R> def_sym(y, n, p, x, s, b)
R> logL_ <- y * log(p) + (n - y) * log(1 - p)
R> gp_ <- der(logL_, p)
R> gp_
```

```
## c:   n - y   y
##    - ----- + -
##      1 - p   p
```

The underscore in `logL_` 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`. With $s = \log(p/(1-p))$ we can find p as:

```
R> sol_ <- solve_sys(log(p / (1 - p)), s, p)
R> p_ <- sol_[[1]]$p
R> p_
```

```
## c:   exp(s)
##    ----------
##    exp(s) + 1
```

The log-likelihood and its derivative as functions of s become:

```
R> logL2_ <- subs(logL_, p, p_)
R> logL2_
```

```
## c:      /  exp(s)  \              /      exp(s)  \
##    y*log|----------| + (n - y)*log|1 - ----------|
##         \exp(s) + 1/              \    exp(s) + 1/
```

```
R> gs_ <- der(logL2_, s) |> simplify()
R> gs_
```

```
## c: -n*exp(s) + y*exp(s) + y
##    ------------------------
##             exp(s) + 1
```

Lastly we connect s to the regression coefficients b and compute the score function, S, and the Hessian, H:

```
R> s_ <- sum(x * b)
R> logL3_ <- subs(logL2_, s, s_)
```

```
R> S_ <- score(logL3_, b) |> simplify()
R> H_ <- hessian(logL3_, b) |> simplify()
R> S_
```

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

```
R> H_
```

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

Since x and b are vectors, the term `x*b` 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, only the score function is shown in the following):

```
R> b <- vector_sym(2, "b")
R> x <- vector_sym(2, "x")
R> s_ <- sum(x * b)
R> logL3_ <- subs(logL2_, s, s_)
```

Again, we compute the score function S by differentiation with respect to the regression parameters. This gives a vector valued funtion of the regression parameters and data.

```
R> S_ <- score(logL3_, b) |> simplify()
```

Next, insert data, e.g. $x_1 = 1$, $x_2 = 2$, $y = 9$, $n = 20$ to obtain a function of the regression parameters only:

```
R> nms <- c("x1", "x2", "y", "n")
R> vls <- c(1, 2, 9, 20)
R> S. <- subs(S_, nms, vls)
```

Note how the expression depending on other symbols, `S_`, is named `S.` to indiciate that data has been inserted:

$$\texttt{S\_} = \begin{bmatrix} \frac{x_1\left(-ne^{b_1x_1+b_2x_2}+ye^{b_1x_1+b_2x_2}+y\right)}{e^{b_1x_1+b_2x_2}+1} \\ \frac{x_2\left(-ne^{b_1x_1+b_2x_2}+ye^{b_1x_1+b_2x_2}+y\right)}{e^{b_1x_1+b_2x_2}+1} \end{bmatrix}, \quad S. = \begin{bmatrix} \frac{9-11e^{b_1+2b_2}}{e^{b_1+2b_2}+1} \\ \frac{2\left(9-11e^{b_1+2b_2}\right)}{e^{b_1+2b_2}+1} \end{bmatrix} \tag{1}$$

An alternative is to create `R` functions and subsequently set default values

```
R> S.. <- as_func(S_, order = c("b1", "b2", "x1", "x2", "y", "n"))
R> S... <- function(b1, b2) {
+   S..(b1, b2, x1 = vls[1], x2 = vls[2], y = vls[3], n = vls[4])
+ }
R> S...(1, 1)
```

```
##       [,1]
## [1,] -10.1
## [2,] -20.1
```

3.2.2 The total likelihood numerically

The score and Hessian for a full data set is the sum of such terms and it is a straight forward `R` task to construct these sums. In either case the result is a function of the regression coefficients which can be used in connection with numerical optimization. This could be a Newton-Rapson algorithm (which would also require the Hessian as function) or a coordinate descent or similar method.

```
R> Sv <- Vectorize(S.., vectorize.args = c("x1", "x2", "y", "n"))
R> Sv_wrap <- function(b1, b2) {
+   Sv(b1, b2, x1 = rep(1, nrow(bud)), x2 = log2(bud$dose), y = bud$ndead, n = bud$ntotal)
+ }
R> Sv_wrap(1, 1)
```

```
##       [,1]  [,2]  [,3]   [,4]  [,5]   [,6]
## [1,] -13.6 -13.6 -10.1  -6.64 -1.87 0.0495
## [2,]   0.0 -13.6 -20.1 -19.92 -7.46 0.2473
```

```
R> apply(Sv_wrap(1, 1), 1, sum)
```

```
## [1] -45.7 -60.9
```

```
R> apply(Sv_wrap(2, 2), 1, sum)
```

```
## [1] -52.2 -66.5
```

3.2.3 The total likelihood symbolically

An alternative to the approach above is to specify the full likelihood directly:

```
R> N <- 6 ## Number of rows in dataset
R> q <- 2 ## Number of explanatory variables
R> X <- matrix_sym(N, q, "x")
R> y <- vector_sym(N, "y")
R> n <- vector_sym(N, "n")
R> p <- vector_sym(N, "p")
R> s <- vector_sym(N, "s")
R> b <- vector_sym(q, "b")
```

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \\ x_{41} & x_{42} \\ x_{51} & x_{52} \\ x_{61} & x_{62} \end{bmatrix}, \quad n = \begin{bmatrix} n_1 \\ n_2 \\ n_3 \\ n_4 \\ n_5 \\ n_6 \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix}$$

The symbolic computations are as follows:

```
R> ## log-likelihood:
R> logL_  <- sum(y * log(p) + (n-y) * log(1-p))
R> ## connecting p and s:
R> p_ <- 1 / (1 + exp(-s))
R> ## log-likelihood as function of linear predictor:
R> logL2_ <- subs(logL_, p, p_)
R> ## linear predictor as function of regression coefficients:
R> s_  <- X %*% b
R> ## log-Likelihood as function of regression coefficients:
R> logL3_ <- subs(logL2_, s, s_)
R> ## Score and Hessian:
R> S_ <- score(logL3_, b)
R> H_ <- hessian(logL3_, b)
```

Above we have analysed the logistic regression model with the logit link. If we instead used the complementary log log link, $\eta = \log(-\log(1-p))$ and find the inverse

```
R> sol_ <- solve_sys(log(-log(1-p)), s, p)
R> sol_
```

which can be used in the above by specifying

```
R> p_ <- 1 - exp(-exp(s))
```

[FIXME: SH: Above]

3.3 Auto regressive models

3.3.1 An $AR(1)$ model

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 $x_1, x_2, \ldots, x_n$ following a stationary zero mean AR(1) process

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

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 $\mathbf{Var}(x_1) = v/(1-a^2)$. Hence we can write $x_1 = \frac{1}{\sqrt{1-a^2}} e_1$.

For simplicity of exposition, we set $n = 4$. All terms $e_1, \ldots, e_4$ are independent and $N(0, v)$ distributed. Let $e = (e_1, \ldots, e_4)$ and $x = (x_1, \ldots x_4)$. Hence $e \sim N(0, vI)$. Isolating error terms 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 covariances matrix) is $K = v^{-1}L^{\top}L$.

```
R> n <- 4
R> def_sym(a)
R> x <- vector_sym(n, "x")
R> e <- vector_sym(n, "e")
R> L <- diff_mat(n, "-a")
R> L[1, 1] <- sqrt(1-a^2)
```

```
R> def_sym(v)
R> Linv <- inv(L)
R> K <- crossprod_(L) / v
R> V <- tcrossprod_(Linv) * v
```

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

$$K = \frac{1}{v} \begin{bmatrix} 1 & -a & . & . \\ -a & a^2+1 & -a & . \\ . & -a & a^2+1 & -a \\ . & . & -a & 1 \end{bmatrix} \tag{4}$$

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

The zeros in the concentration matrix K implies a conditional independence restrstriction: 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 et al., 2012], chap. 4 for details.[2]

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 multiplicative constants)

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

where we note that $\mathbf{tr}(KW)$ is the sum of the elementwise products of K and W since both matrices are symmetrical.

```
R> logL <- log(det(K)) - sum(K * (x %*% t(x))) %>% simplify()
```

[2] BETTER REFERENCE; Handbook of graphical models perhaps?

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

3.3.2 Bridging the gap - towards 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> f_wrap <- as_func(logL., vec_arg = TRUE)
R> eps <- 0.01
R> par <- optim(c(a=0, v=1), f_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 variables to only take values in $(-1, 1)$.

3.4 Maximum likelihood under constraints - independence model for two-way contingency table

In this section we illustrate constrained optimization using Lagrange multipliers. Consider a 2×2 contingency table with cell counts n_{ij} and cell probabilities p_{ij} for $i = 1, 2$ and $j = 1, 2$.

```
##    c
## r   1   2
##   1 n11 n12
##   2 n21 n22
```

Under multinomial sampling, the log likelihood is

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

Under the assumption of independence between rows and columns, the cell probabilities have the form, (see e.g. [REFERENCE], chap. XXX)

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

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

$$Lag(p) = -l(p) + \lambda g(p) \quad \text{under the constraint that} \tag{6}$$

$$g(p) = \sum_{ij} p_{ij} - 1 = 0. \tag{7}$$

where λ is a Lagrange multiplier.

```
R> n_ <- c("n11", "n21", "n12", "n22")
R> n  <- as_sym(n_)
R> def_sym(u, r2, s2, lam)
R> p <- as_sym(c("u", "u*r2", "u*s2", "u*r2*s2"))
R> logL  <- sum(n * log(p))
R> Lag  <- -logL + lam * (sum(p) - 1)
R> vars <- list(u, r2, s2, lam)
R> gLag <- der(Lag, vars)
R> sol <- solve_sys(to_vector(gLag), vars)
R> print(sol, method = "ascii")
```

```
## Solution 1:
##   u   =  (n11 + n12)*(n11 + n21)/(n11 + n12 + n21 + n22)^2
##   r2  =  (n21 + n22)/(n11 + n12)
##   s2  =  (n12 + n22)/(n11 + n21)
##   lam =  n11 + n12 + n21 + n22
```

```
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)
```

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

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()
```

3.5 A compound symmetry covariance structure

Consider random variables $x_1, \ldots, x_n$ 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 $(x_1, \ldots, x_n)$ is therefore

$$V = vR = v \begin{bmatrix} 1 & r & r \\ r & 1 & r \\ r & r & 1 \end{bmatrix}. \tag{8}$$

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 $\mathbf{Var}(\bar{x}) = \frac{1}{n^2} 1^\top V 1$. The answer lies in studying the limiting behaviour of this expression when $n \to \infty$ and `caracas` can not handle this directly.

What can be done in `caracas` is the following: 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 must by hand find 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
```

Now we can study the limiting behaviour 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 illustrate to supplement the symbolic computations above with numerical evaluations:

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

4 Possible topics and smaller projects for students

1. Related to Sec. 3.1: Verify that the residuals $r = (I - P)y$ are not all independent and that the correlation between is small and becomes smaller as the number of subjects per group increase. Verify that $PX = X$ and thus $(I - P)X = 0$. Verify also that the rank of P equals the number of groups, which is 3. A model matrix also spanning L is `X_2 = model.matrix(~ -1 + f)`. Investigate how the quantities above look for this choice of model matrix. Construct a balanced two way analysis of variance (two-way anova), first only with main effects and then with an interaction and compare the estimates.

2. Related to Sec. 3.2.1: Implement Newton-Rapson to solve the likelihood equations and compare your solution to the output from `glm()`. Related to Sec. 3.2.3: Used the above symbolic computations and substitute data in directly. Compare to `S.` and `H.` from Sec. 3.2.1.

3. Related to Sec. 3.4: A simple task is to consider a multinomial distribution with three categories, counts y_i and cell probabilities p_i, $i = 1, 2, 3$ where $\sum_i p_i = 1$. For this model, find the maximum likelihood estimate for p_i. Above, identifiability of the parameters was handled by not including r_1 and s_1 in the specification of p_{ij}. An alternative is to impose the restrictions $r_1 = 1$ and $s_1 = 1$, and this can also be handled via Lagrange multipliers.

4. Related to Sec. 3.3: Find (approximate) standard error and confidence interval for the parameter a. Modify the model in (2) by setting $x_1 = ax_n + e_1$ and see what happens to the pattern of zeros in

the concentration matrix. Extend the $AR(1)$ model to and $AR(2)$ model and investigate this model along the same lines as above.

5. Related to Sec. 3.5: It is illustrative to study such behaviours for other covariance functions. For example (1) $\mathbf{Cov}(x_i, x_j) = vr^{|i-j|}$ and (2) $\mathbf{Cov}(x_i, x_j) = vr$ if $|i-j| = 1$ and $\mathbf{Cov}(x_i, x_j) = 0$ if $|i-j| > 1$.

5 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.

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.

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

6 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.

Contents

1 Introduction **1**

2 Mathematics and documents containing mathematics **1**
2.1 Calculus . . . 1
2.2 Linear algebra . . . 2
2.3 Preparing mathematical documents . . . 3

3 Statistics examples **4**
3.1 Linear models – one way analysis of variance . . . 4
3.2 Logistic regression . . . 5
3.2.1 Each component of the likelihood . . . 6
3.2.2 The total likelihood numerically . . . 7
3.2.3 The total likelihood symbolically . . . 8
3.3 Auto regressive models . . . 8
3.3.1 An $AR(1)$ model . . . 8
3.3.2 Bridging the gap - towards numerical evaluation . . . 10
3.4 Maximum likelihood under constraints - independence model for two-way contingency table 10
3.5 A compound symmetry covariance structure . . . 11

4 Possible topics and smaller projects for students **12**

5 Discussion and future work **13**

6 Acknowledgements **13**

A A short primer **14**

A A short primer

This section provides a brief introduction to caracasto make this paper self contained. Readers are recommended to study the online documentation at https://r-cas.github.io/caracas/. The caracas package provides an interface from R to the Python package SymPy [Meurer et al., 2017]. This means that SymPy is "running under the hood" of R via the reticulate package [Ushey et al., 2020]. The SymPy package is mature and robust with many users and developers.

A caracas symbol is a list with a pyobj slot and the class caracas_symbol. The pyobj is an an object in Python (often a SymPy object). As such, a symbol (in R) provides a handle to a Python object. In the design of caracas we have tried to make this distinction something the user should not be concerned with, but it is worthwhile being aware of the distinction.

One way of defining a symbol is with def_sym() which declares the symbol in R and in Python. A symbol can also be defined in terms of other symbols: Define symbols s1 and s2 and define symbol s3 in terms of s1 and s2:

```
R> def_sym(s1, s2) ## Note: 's1' and 's2' exist in both R and Python
R> s1$pyobj
```

```
## s1
```

```
R> s3_ <- s1 * s2   ## Note: 's3' is a symbol in R; no corresponding object in Python
R> s3_$pyobj
```

```
## s1*s2
```

The underscore in s3_ indicates that this expression is defined in terms of other symbols. This convention is used through out the paper. Next express s1 and s2 in terms of symbols u and v (which are created on the fly):

```
R> s4_ <- subs(s3_, c("s1", "s2"), c("u+v", "u-v"))
R> s4_
```

```
## c: (u - v)*(u + v)
```

Another way of creating a caracas symbol is from an R object:

```
R> v <- c("v1", "v2")
R> as_sym(v) ## A 2 x 1 matrix
```

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

A caracas expression can be coerced to an R expression and subsequently evaluated numerically. Alternatively can be coerced to an R function:

```
R> s4_expr <- as_expr(s4_)
R> s4_expr
```

```
## expression((u - v) * (u + v))
```

```
R> s4_fun <- as_func(s4_)
R> s4_fun
```

```
## function(u, v)
## {
## (u - v) * (u + v)
## }
## <environment: 0x55a18763d818>
```

A numerical evaluation is obtained as (output omitted):

```
R> eval(s4_expr, list(u=1, v=2))
R> s4_fun(u=1, v=2)
```

References

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. *rmarkdown: Dynamic Documents for R*, 2021. URL https://github.com/rstudio/rmarkdown. R package version 2.7.

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

Mikkel Meyer Andersen and Søren Højsgaard. caracas: Computer algebra in r. *Journal of Open Source Software*, 6(63):3438, 2021. doi: 10.21105/joss.03438. URL https://doi.org/10.21105/joss.03438.

Søren Højsgaard, David Edwards, and Steffen Lauritzen. *Graphical Models with R*. Springer, New York, 2012. doi: 10.1007/978-1-4614-2299-0. ISBN 978-1-4614-2298-3.

Friedrich Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In Wolfgang Härdle and Bernd Rönz, editors, *Compstat*, pages 575–580, Heidelberg, 2002. Physica-Verlag HD.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. London.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi: 10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL http://www.R-project.org/. ISBN 3-900051-07-0.

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

United Nations General Assembly. Sustainable development goals, 2015. https://sdgs.un.org/.

Kevin Ushey, JJ Allaire, and Yuan Tang. *reticulate: Interface to 'Python'*, 2020. URL https://CRAN.R-project.org/package=reticulate. R package version 1.18.

Yihui Xie, J.J. Allaire, and Garrett Grolemund. *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida, 2018. URL https://bookdown.org/yihui/rmarkdown. ISBN 9781138359338.

Yihui Xie, Christophe Dervieux, and Emily Riederer. *R Markdown Cookbook*. Chapman and Hall/CRC, Boca Raton, Florida, 2020. URL https://bookdown.org/yihui/rmarkdown-cookbook. ISBN 9780367563837.