mathml: Translate R expressions to MathML and LaTeX/MathJax

by Matthias Gondan and Irene Alfarone

Abstract This R package translates R objects to suitable entities in MathML or LaTeX, thereby enabling pretty mathematical representation of R functions or algorithms in data analysis, scientific reports and interactive web content. R code text descriptions and mathematical content already coexist, side-by-side, in the RMarkdown document rendering language. The present package allows to use the same R objects for both data analysis and typesetting in documents or web content. By tightening the link between the statistical analysis and its verbal or symbolic representation, this new feature is a futher step towards reproducible science. User-defined hooks allow to extend the package by mapping new variable names and functions to new MathML and LaTeX entities. We provide a few working examples that illustrate the use of the package in a little scientific report and in an interactive web page.

Introduction

The R extension of the markdown language (Xie, Dervieux, and Riederer 2020) enables reproducible statistical reports with nice typesetting in HTML, Microsoft Word, and Latex. Since recently (R Core Team 2021, version 4.2), R's manual pages include support for mathematical expressions (Sarkar and Hornik 2022), which already is a big improvement. However, rules for the translation of R's built-in language elements to their mathematical representation are still lacking. So far, R expressions such as `pbinom(k, N, p)` are printed as they are; pretty mathematical formulae like $P_{\mathrm{Bi}}(X \le k; N, p)$ require explicit Latex commands, that is, `P_\mathrm{Bi}\left(X \le k; N, p\right)`. Except for minimalistic examples, these commands are tedious to type in and hard to read in their source code.

The present R package defines a set of rules for an *automatic* translation of R expressions to mathematical output in RMarkdown documents (Xie, Dervieux, and Riederer 2020) and ShinyApp webpages (Chang et al. 2022). The translation is done by an embedded Prolog interpreter that maps nested expressions recursively to MathML and LaTeX/MathJax, respectively. User-defined hooks enable to extend the set of rules, for example, to represent specific R elements by custom mathematical signs.

todo: mention that the *same* R expression is used for mathematical calculations and printing.

The paper is organized as follows. Section 2 presents the background of the package, including Prolog and the R package **rolog** and the two main classes of rules for translating R objects to mathematical expressions. Section 3 illustrates some important features of the **mathml** package, potential issues and workarounds using examples from the day-to-day perspective of the user. Sections 4 and 5 present two case studies for the use of the package in a scientific report and a ShinyApp web page. Section 6 concludes with a discussion and ideas for further development. A complete list of the R elements covered by the package is found in the Appendix.

Background

Prolog: the rolog package

The automatic translation of R expressions to mathematical output is achieved through a Prolog interpreter. Prolog is a classical logic programming language with many applications in expert systems, computer linguistics and traditional, that is, symbolic artificial intelligence. The main strength of Prolog is its concise representation of facts and rules for the representation of knowledge and grammar, as well as its efficient built-in search engine for closed world domains. R, as it is well-known, is a statistical programming language for data analysis and statistical modeling which is widely used in academia and industry. Besides the core library, a lot of packages have been developed for all kinds of statistical problems, including statistics-based artificial intelligence tools such as neural networks for machine learning and deep learning. Whereas Prolog is weak in statistical computation, but strong in symbolic manipulation, the converse may be said for the R language. The **rolog** package then bridges this gap by providing an interface to a SWI-Prolog distribution in R. The communication between the two systems is mainly in the form of queries from R to Prolog, but two predicates allow Prolog to ring back and evaluate terms in R. Here is reported a first trivial example. At the Prolog end, there is a handler for `pbinom/3` that translates the term into a pretty MathML syntax like P_bi(X <= k; N, pi).

```
library(rolog)

#> Found SWI-Prolog at SWI_HOME_DIR: /Library/Frameworks/R.framework/Versions/4.2/Resources/library/rswipl/swi

#> Welcome to SWI-Prolog (threaded, 64 bits, version 8.5.20)
#> SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
#> Please run ?- license. for legal details.
#>
#> For online help and background, visit https://www.swi-prolog.org
#> For built-in help, use ?- help(Topic). or ?- apropos(Word).

library(mathml)
term = quote(pbinom(k, N, p))

# Pretty print
mathout(term)
```

$$P_{\mathrm{Bi}}\left(X \leq k; N, p\right)$$

Beside the graphical rendition of the term, an important feature is that the package allows also for its calculation, for example:

```
k = 10
N = 22
p = 0.4
eval(term)
```

[1] 0.77195

Here we present a second less trivial example where Prolog needs to find out the name of the integration variable for `sin`. For that purpose, **rolog** provides a predicate `r_eval/2` that calls R from Prolog. Here, the predicate is used for the R function `formalArgs(args(sin))`, which returns the name of the function argument of `sin`, that is, `x`.

```
term = quote(integrate(sin, 0L, 2L*pi))
mathout(term)
```

$$\int_0^{2\pi} \sin x \, dx$$

```
eval(term)

#> 2.221501e-16 with absolute error < 4.4e-14
```

Note that, on the Prolog end, the handler for `integrate/3` is rather rigid; it accepts only these three arguments in that particular order, and without names, that is, `integrate(sin, lower=0L, upper=2L * pi)`, would not print the desired result.

Therefore, `pl/mathml.pl` includes two handlers that accept terms with named arguments, `integrate(f=Fn, lower=Lower, upper=Upper)`, as well as terms of the form `$(integrate(Fn, Lower, Upper), value)` that are needed for the evaluation below.

```
# A custom function
g <- function(u)
{
  sin(u)
}

# Mixture of (partially) named and positional arguments in unusual order
term = call("integrate", lower=-Inf, upper=Inf, g)
attributes(term)$name = quote(g)
mathout(canonical(term))
```

$$\int_{-\infty}^{\infty} g\left(u\right) du$$

```
# Even more complex
term <- call("*", 2L, call("$", call("integrate", low=-Inf, up=Inf, g), "value"))
attributes(term)$name = quote(g)
mathout(canonical(term))
```

$$2 \cdot \int_{-\infty}^{\infty} g\,(u)\,du$$

```
# It is a bit of a mystery that R knows the result of this integral.
eval(term)
```

```
[1] 0
```

The extra R function `canonical()` applies `match.call()` to non-primitive R calls, basically cleaning up the arguments and bringing them into the correct order.

Package capabilities

mathml is a package for pretty mathematical representation of R functions or algorithms in data analysis, scientific reports and interactive web content. The features currently supported are listed below, following the order proposed by (Murrell and Ihaka 2000).

Hooks

Before start listing all the possible operations with **mathml**, we mention here that **mathml** also allows the user to create hooks for specific functions. As an example:

```
# Improve display
hook(quote(mu_A), quote(mu["A"]))
hook(quote(mu_B), quote(mu["B"]))
hook(quote(sigma_A), quote(sigma["A"]))
hook(quote(sigma_B), quote(sigma["B"]))
hook(quote(mu_M), quote(mu["M"]))
hook(quote(zn), quote(z[tau]))
hook(quote(M), quote(overline(X)))
```

The hooks reported here will be the one used in Section *ref* to display the equations in the Case Study. On the right hand side of the parenthesis is reported the way that the variable, for instance

Arithmetic operators

The **mathml** package provides the graphical representation for addition, subtraction, multiplication and division. For instance:

```
catmath(x + y - z)
```

$$x + y - z$$

In addition, with **mathml** it is also possible to combine `minus` and `plus` and get a new operator. However, in R the operator $\pm$ is not present and this leads to some interesting questions. If we let R present the expression, this is what we get:

```
# Note wrong precedence of custom operator
catmath(mean(m) %+-% 1.96 * s/sqrt(N))
```

$$(\overline{m} \pm 1.96) \cdot s/\sqrt{N}$$

As can be seen, the new custom operator takes precedence over multiplication, as R automatically puts `mean(m) %+-% 1.96` in parentheses. Unfortunately, this is not consistent with the rules of mathematics. As the reader well knows, it is not possible in R to change the precedence of custom operators. This leaves two possible solutions. The first involves a mathematically unnecessary parenthesis around `(1.96 * s/sqrt(N))`. The second is more elegant and mathematically consistent: here `1.96 * s/sqrt(N)` is put in curly brackets and mathematical formality is respected. In both solutions R is able to produce correct output once the order of operations is made explicit.

```
# This is what we actually want
catmath(mean(m) %+-% (1.96 * s/sqrt(N)))
```

$$\overline{m} \pm \left(1.96 s/\sqrt{N}\right)$$

```
# Without parentheses
catmath(mean(m) %+-% {1.96 * s/sqrt(N)})
```

$$\overline{m} \pm 1.96 s/\sqrt{N}$$

The following can be inserted in a table

```
catmath(x*y)
```

$$xy$$

```
catmath(x %*% y)
```

$$x \times y$$

```
catmath(x %.% y)
```

$$x \cdot y$$

```
catmath(X %x% Y)
```

$$X \otimes Y$$

```
catmath(x/y)
```

$$x/y$$

```
catmath(x %/% y)
```

$$\lfloor x/y \rfloor$$

Subscripts and Superscripts

Secondly, **mathml** allows also for the display of subscripts, superscripts and their combination.

```
catmath(a["x"])
```

$$a_{\mathrm{x}}$$

```
catmath(a^x)
```

$$a^{x}$$

```
catmath(x["b"]^"a")
```

$$x_{\mathrm{b}}^{\mathrm{a}}$$

Fractions

As in (Murrell and Ihaka 2000) the `frac` and `over` functions aim to represent fractions. `choose` and `lchoose`, as in R **base**, display the binomial coefficient.

```
catmath(frac(x, y))
```

$$\frac{x}{y}$$

```
catmath(over(x, y))
```

$$\frac{x}{y}$$

```
catmath(choose(n, k))
```

$$\binom{n}{k}$$

```
catmath(lchoose(n, k))
```

$$\log \binom{n}{k}$$

Radicals

```
catmath(sqrt(x))
```

$$\sqrt{x}$$

Relations

Among the possible relational operators, following (Murrell and Ihaka 2000), we included:

```
catmath(x==y)
```

$$x = y$$

```
catmath(x < y)
```

$$x < y$$

```
catmath(x <= y)
```

$$x \leq y$$

```
catmath(x >= y)
```

$$x \geq y$$

```
catmath(!x)
```

$$\neg x$$

```
catmath(x != y)
```

$$x \neq y$$

```
catmath(X ~ Y)
```

$$X \sim Y$$

```
catmath(x %~~% y)
```

$$x \approx y$$

```
catmath(x %==% y)
```

$$x \equiv y$$

```
catmath(x %=~% y)
```

$$x \cong y$$

```
catmath(x %prop% y)
```

$$x \propto y$$

Set relations

Here is provided a series of useful set relations.

Still to be implemented

```
catmath(a %in% A)
```

$a \in A$

```
catmath(intersect(A, B))
```

$A \cap B$

```
catmath(union(A, B))
```

$A \cup B$

```
catmath(is.null(intersect(A, B)))
```

$A \cap B = \emptyset$

Arrows

Following once again (Murrell and Ihaka 2000), we provide here a list of useful arrows

```
# x %<->% y
catmath(x %<->% y)
```

$x \leftrightarrow y$

```
# x %->% y
catmath(x %->% y)
```

$x \rightarrow y$

```
# x %<-% y
catmath(x %<-% y)
```

$x \leftarrow y$

```
# x %up% y:
catmath(x %up% y)
```

$x \uparrow y$

```
# x %down% y:
catmath(x %down% y)
```

$x \downarrow y$

```
# x %<=>% y: solved it using iff
catmath(x %<=>% y)
```

$x \iff y$

```
# x %=>% y: solved using rArr
catmath(x %=>% y)
```

$x \Rightarrow y$

```
# x %<=% y: solved using lArr
catmath(x %<=% y)
```

$$x \Leftarrow y$$

```
# x %dblup% y:
catmath(x %dblup% y)
```

$$x \Uparrow y$$

```
# x %dbldown% y:
catmath(x %dbldown% y)
```

$$x \Downarrow y$$

Accents

```
catmath(hat(x))
```

$$\hat{x}$$

```
catmath(hat(xy))
```

$$x\hat{y}$$

```
catmath(mean(x))
```

$$\overline{x}$$

```
catmath(mean(xy))
```

$$\overline{xy}$$

Symbolic names

```
catmath(A-Omega)
```

$$A - \Omega$$

```
catmath(alpha-omega)
```

$$\alpha - \omega$$

```
catmath(pi-Pi)
```

$$\pi - \Pi$$

```
catmath(Inf)
```

$$\infty$$

Operators

A list of possible operators is available here.

```
catmath(sum(x[i], i==1L, n))
```

$$\sum(x_i, i = 1, n)$$

```
# fixme: braces not needed when right to *
catmath(a * sum(b))
```

$$a \cdot \sum b$$

```
# Multiplication (capital Pi) is missing
catmath(a * prod(b))
```

$$a \cdot \prod b$$

```
catmath(integrate(sin, 0L, 2L*pi))
```

$$\int_0^{2\pi} \sin x \, dx$$

```
catmath(A & B)
```

$$A \wedge B$$

```
catmath(A | B)
```

$$A \vee B$$

```
catmath(!A)
```

$$\neg A$$

```
catmath(xor(A, B))
```

$$A \veebar B$$

```
catmath(exp(2L*pi*i))
```

$$\exp(2\pi \cdot i)$$

```
catmath(expm1(2L*pi*i))
```

$$\exp(2\pi \cdot i) - 1$$

```
catmath(log(x))
```

$$\log x$$

```
catmath(log10(x))
```

$$\log_{10} x$$

```
catmath(log2(x))
```

$$\log_2 x$$

```
catmath(logb(x, e))
```

$$\log_e x$$

```
catmath(log1p(x))
```

$$1 + \log x$$

```
catmath(ceiling(pi))
```

$$\lceil \pi \rceil$$

```
catmath(floor(pi))
```

$\lfloor \pi \rfloor$

```
catmath(sin(pi/2L))
```

$\sin(\pi/2)$

```
catmath(cos(pi/2L))
```

$\cos(\pi/2)$

```
catmath(tan(pi/2L))
```

$\tan(\pi/2)$

Grouping

At first glance, grouping expressions may seem trivial enough. However, when you want to deal with, for instance, exponentiation or multiplication, things can get more complicated. Let us consider a trivial example. If we want to calculate and evaluate the expression `a * (b + c)`, we find that R automatically gives precedence to the addition even if we do not specify it in the text, as in the following.

```
first <- quote(a)
second <- quote(b + c)
mathout(call("*", first, second))
```

$a \cdot (b + c)$

```
# precedence
quote(a * (b + c))[[3]][[1]]
```

`(`

With exponentiation the situation can be less trivial. If we consider `x` to the power of `y + z`, following R notation we should add a parenthesis around the two terms. Otherwise R would consider only `y` as power of `x`.

```
catmath(x^y + z)
```

$x^y + z$

```
# mathout(call("+", call("^", as.symbol("x"), as.symbol("y")), as.symbol("z")))

catmath(x^(y + z))
```

$x^{(y+z)}$

Unfortunately, R automatically put `y + z` in parentheses even if, from a mathematical point of view, the brackets are not needed. To solve this issue two solutions are possible, the first one is cumbersome since it requires many specifications. The second, that can be considered a shortcut, uses a syntax similar to LateX and puts the power in curly brackets. In both cases the output produced is consistent with the mathematical notation.

```
# without parenthesis
mathout(canonical(call("^", as.symbol("x"), call("+", as.symbol("y"), as.symbol("z")))))
```

x^{y+z}

```
# shortcut
catmath(x^{y + z})
```

x^{y+z}

Absolute value

Is it also possible to calculate and represent the absolute value, as:

```
catmath(abs(x))
```

$$|x|$$

Matrix and Vectors

Since also Matrices and Vectors are important elements in R, we decided to implement them and render them graphically in the following ways.

```
vec <- 1:3
mathout(canonical(vec))
```

$$(1\,2\,3)$$

```
mathout(vec)
```

$$(1\,2\,3)$$

```
mat <- matrix(data = c(2L, 3L, 4L, 5L), nrow = 2, ncol = 2)
mathout(canonical(mat))
```

$$\begin{pmatrix} 2 & 4 \\ 3 & 5 \end{pmatrix}$$

Abbreviations

mathml allows also for the use of abbreviations in formulas. Let us consider here the t-statistic for independent samples with equal variance assumed. The procedure consists of three steps. The first one is to define the hooks to nicely print the variables. Second, we write the t-statistic specifying in the formula how to calculate the pooled variance with the function `denote(abbr, expr, info)`. Finally, since **mathml** also allows computation, we can directly compute the value of the t statistic with `eval()`.

```
hook(quote(x_1), quote(mean(x["1"])))
hook(quote(x_2), quote(mean(x["2"])))
hook(quote(n_1), quote(n["1"]))
hook(quote(n_2), quote(n["2"]))
hook(quote(s_1), quote(s["1"]))
hook(quote(s_2), quote(s["2"]))
hook(quote(s_p), quote(s["pool"]))


t <- quote(dfrac(x_1 - x_2, sqrt(denote(s_p^2L, frac((n_1 - 1L)*s_1^2L + (n_2 - 1L)*s_2^2L, n_1+n_2-2L), "the pool
```

```
mathout(t)
```

$$\frac{\overline{x_1} - \overline{x_2}}{\sqrt{s^2_{\text{pool}} \cdot \left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}, \text{ with } s^2_{\text{pool}} = \frac{(n_1-1)\cdot s_1^2 + (n_2-1)\cdot s_2^2}{n_1+n_2-2} \text{ denoting the pooled variance.}$$

```
# An example

x_1 = 20.5
x_2 = 35.7
n_1 = 150
n_2 = 160
s_1 = 3.4
s_2 = 4.2

eval(t)
```

```
[1] -34.88421
```

A case study

For a more clear representation of what the **mathml** package is supposed to do, we provide here a case study, based on the work by Schwarz (1994), in which the features of **mathml** are implemented.

The reader may recognize that Schwarz' (1994) study can be traced back to the strand of decision-making models, where decision-making is assumed to be a process of noisy accumulation of information over time (e.g., Ratcliff et al. 2016). The aim of Schwarz' (1994) study is to present a new explanation of redundancy gains when observers make speeded responses to stimuli of different sources, and the same information is presented on two channels. Schwarz (1994) describes a model that assumes the superposition of channel-specific diffusion processes that eventually reach an absorbing barrier to elicit the response. For a detailed description the reader may refer to the original article.

The model

Schwarz' (1994) model refers to two stimuli A and B, presented either alone or in combination (redundant stimuli), with the redundant stimuli being presented either simultaneously or with onset asynchrony τ. The channel activation is described as a two-dimensional Wiener process with drifts μ_A, μ_B, variances σ_A^2, σ_B^2, correlation ρ_{AB} and initial conditions $X_i(t=0)=0, i=\mathrm{A},\mathrm{B}$. The response is elicited when the process reaches the absorbing barrier $c>0$ for the first time. In combined stimuli ("redundant" stimuli), the overall diffusion process is $X_{AB}(t) = X_A(t) + X_B(t)$, which is again a Wiener process with drift $\mu_A + \mu_B$ and variance $\sigma_A^2 + \sigma_B^2 + 2\rho_{AB}\sigma_A\sigma_B$.

In single-target trials, the first passages at c are expected at $E[D_i] = \frac{c}{\mu_i}, i = \mathrm{A},\mathrm{B}$, whereas in synchronous redundant-target trials, we have $E[D_{AB}] = \frac{c}{\mu_A+\mu_B}$.

For asynchronous stimuli, Schwarz (1994) derived the expected first-passage time $E[D_{A(\tau)B}]$ as a function of the stimulus onset asyncrony $\tau > 0$ (Eq. 10),

```
f <- function(tau, c, mu_A, sigma_A, mu_B, sigma_B)
{ dfrac(c, mu_A) + (dfrac(1L, mu_A) - dfrac(1L, mu_A + mu_B)) *
    ((mu_A*tau - c) * pnorm(dfrac(c - mu_A*tau, sqrt(sigma_A^2L*tau)))
      - (mu_A*tau + c) * exp(dfrac(2L*mu_A*c, sigma_A^2L))
        * pnorm(dfrac(-c - mu_A*tau, sqrt(sigma_A^2L*tau))))
}

mathout(f)
```

$$\lambda\left(\tau\, c\, \mu_A\, \sigma_A\, \mu_B\, \sigma_B\right) = \frac{c}{\mu_A} + \left(\frac{1}{\mu_A} - \frac{1}{\mu_A+\mu_B}\right) \cdot \left[(\mu_A\tau - c)\cdot\Phi\left(\frac{c-\mu_A\tau}{\sqrt{\sigma_A^2\tau}}\right) - (\mu_A\tau + c)\cdot\exp\left(\frac{2\mu_A\cdot c}{\sigma_A^2}\right)\cdot\Phi\left(\frac{-c-\mu_A\tau}{\sqrt{\sigma_A^2\tau}}\right.\right.$$

The observable response time is assumed to be the sum of D, the time employed to reach the threshold for the decision, and M, denoting the of the other processes.

$E[T] = E[D+M] = E[D] + E[M] = E[D] + \mu_M$

Methods

Schwarz (1994) applied the model to Miller's (1986) data from a redundant signals task with 13 onset asynchronies $0, \pm 33, \pm 67, \pm 100, \pm 133, \pm 167, \pm\infty$ ms, where 0 refers to synchronous AB, ∞ refers to the single-target presentation, and negative τ denote those conditions in which B is presented before A. Each condition was replicated 400 times. The 13 mean RTs and standard deviations are given in Table 2 below.

Then, from Schwarz' (1994) Equations [2] [3] and [10], we calculated the expected first time passage for the respective τ and the parameters $\mu_A = 0.53, \mu_B = 1.32, \sigma_A = 4.43, \sigma_B = 18.3, c = 100, \mu_M = 161$

$$\lambda\left(\tau\, c\, \mu_A\, \sigma_A\, \mu_B\, \sigma_B\right) = rbind\left(\frac{c}{\mu_A}, \text{ if } \tau = \infty\, \frac{c}{\mu_B}, \text{ if } \tau = -\infty\, f\left(\tau, c, \mu_A, \sigma_A, \mu_B, \sigma_B\right), \text{ if } \tau > 0\, \frac{c}{\mu_A+\mu_B}, \text{ if } \tau = 0\, f\left(-\tau, c, \mu_B, \sigma_B\right.\right.$$

For instance, we can see that with $\tau = -167$ the expected first passage time is:

```
#> [1] 70.31943
```

To which we have to add $\mu_M = 161$ (which represents the amount of time that the movement of providing the answer takes) and we obtain:

```
#> [1] 231.3194
```

Similarly, we obtain the first-time passage for every τ:

$\lambda\left(\tau\, c\, \mu_{\mathrm{A}}\, \sigma_{\mathrm{A}}\, \mu_{\mathrm{B}}\, \sigma_{\mathrm{B}}\, \mu_{\mathrm{M}}\right) = G\left(\tau, c, \mu_{\mathrm{A}}, \sigma_{\mathrm{A}}, \mu_{\mathrm{B}}, \sigma_{\mathrm{B}}\right) + \mu_{\mathrm{M}}$

```
#>  [1] 236.7576 231.3194 230.0571 228.4017 226.0523 222.2336 215.0541 238.5977
#>  [9] 262.3817 282.9833 299.7299 312.9791 349.6792
```

Model fitting and Parameters Estimation

Then we proceeded with the model fitting, using the data provided by Miller (1986) study in Schwarz (1994). First, we calculated the predicted means RT as $E[RT] = E[T] + \mu_M$.

We briefly remind that in diffusion models there is across-trial variability, and the drift rates are assumed to be normally distributed (Ratcliff and McKoon 2008) with at the first passage of c $X(t) \sim N(\mu t, \sigma^2 t)$. Where T is expressed as $T = \arg\min_t X(t) \geq c$ with $T \sim IG(c, \mu, \sigma^2)$ with mean $E(T) = \frac{c}{\mu}$ and variance $Var(T) = \frac{c\sigma^2}{\mu^3}$. Furthermore, for the Central Limit Theorem $\bar{T} = \frac{\Sigma_i T_i}{N} \dot{\sim} N(\frac{c}{\mu}, \frac{c\sigma^2}{\mu^3})$

For these reasons, in order to obtain the Goodness of Fit measure we proceeded with a z-standardisation.

$\lambda\left(\overline{X}\, s\, N\, \tau\, c\, \mu_{\mathrm{A}}\, \sigma_{\mathrm{A}}\, \mu_{\mathrm{B}}\, \sigma_{\mathrm{B}}\, \mu_{\mathrm{M}}\right) = rbind\left(E = h\left(\tau, c, \mu_{\mathrm{A}}, \sigma_{\mathrm{A}}, \mu_{\mathrm{B}}, \sigma_{\mathrm{B}}, \mu_{\mathrm{M}}\right) \frac{\overline{X} - E}{s/\sqrt{N}}\right)$

```
#>  [1] -2.05627706  0.92433511 -0.02852697 -0.70084870  1.21730061 -0.88115730
#>  [7]  1.38996139 -0.42696106  0.47564623 -3.98888722 -1.08120914  1.77700915
#> [13] -0.36505332
```

Then, we calculated the Goodness of Fit as follows:

$X^2 = \Sigma_\tau z_\tau^2 \sim \chi^2(13 - 5\, dgf)$.

$\lambda\left(par\, \tau\, \overline{X}\, s\, N\right) = rbind\left(z_\tau = zf\left(\overline{X}, s, N, \tau, c = 100, \mu_{\mathrm{A}} = par_{\mathrm{mu_A}}, \sigma_{\mathrm{A}} = par_{\mathrm{sigma_A}}, \mu_{\mathrm{B}} = par_{\mathrm{mu_B}}, \sigma_{\mathrm{B}} = par_{\mathrm{sigma_B}}, \mu_{\mathrm{M}} = par\right.\right.$

[1] 30.54472 [1] 0.0001692742

$\lambda\left(\tau\, \overline{X}\, s\, N\right) = rbind\left(start = c\left(\mu_{\mathrm{A}} = 0.30, \sigma_{\mathrm{A}} = 2.00, \mu_{\mathrm{B}} = 1.00, \sigma_{\mathrm{B}} = 5.00, \mu_{\mathrm{M}} = 200.00\right) optim\left(start, fn = gof, \tau = \tau, \overline{X} =\right.\right.$

Results

The calculated Goodness of Fit value is $X^2 = 30.54$, $p = 0.0002$. Additionally, we proceeded with the parameters' estimation with the Least Squares Method.

The values of the estimated parameters reported in the table below are consistent with those reported by Schwarz (1994).

Table 1: Table 1: Estimated Model's Parameters

Parameters	Estimate	95% Confidence Interval
μ_{A}	0.53	0.51, 0.56
σ_{A}	4.4	3.8, 5.1
μ_{B}	1.32	1.2, 1.4
σ_{B}	18.3	11.6, 24.9
μ_{M}	161	157, 165

Furthermore, we can observe from the values reported in the Table below that, with few exceptions (e.i. $\tau = 100$) Miller's (1986) observed values are quite similar to the one predicted by Schwarz' (1994) model. Showing that, despite the calculated Goodness of Fit measure, the model proposed by Schwarz (1994) sufficiently fit the data.

Table 2: Table 2: Predicted Means RT and Observed Means RT and SDs from Miller's (1986) study in Schwarz (1994)

SOA	Predicted Mean RT	Observed Mean RT	Observed SD
$-\infty$	237	231	56
-167	231	234	58
-133	230	230	40
-100	228	227	40
-67	226	228	32
-33	222	221	28
0	215	217	28
33	239	238	28
67	262	263	26
100	283	277	30
133	300	298	32
167	313	316	34
∞	350	348	92

Note. Parameters: $\mu_A = 0.53, \mu_B = 1.32, \sigma_A = 4.4, \sigma_B = 18.3, c = 100, \mu_M = 161$

Mistakes

The case study presented above displays several possible ways of how **mathml** can be a useful tool in data representation and calculation. Furthermore, the careful reader might have also noticed that **mathml** allows also for a more smoothly detection of possible mistakes in writing.

An interesting use case of the package is in this sense the readability of self-written functions in R code. R code is notoriously hard to read, as are most programming languages (Green 1977). The poor legibility of the language is one of the main sources of mistakes. For illustration, we consider Equation 10 in Schwarz (1994), namely the same equation reported above. The first version has a wrong parenthesis, which is barely visible in the R code.

```
f <- function(tau)
{ dfrac(c, mu["A"]) + (dfrac(1L, mu["A"]) - dfrac(1L, mu["A"] + mu["B"]) *
    ((mu["A"]*tau - c) * pnorm(dfrac(c - mu["A"]*tau, sqrt(sigma["A"]^2L*tau)))
      - (mu["A"]*tau + c) * exp(dfrac(2L*mu["A"]*tau, sigma["A"]^2L))
        * pnorm(dfrac(-c - mu["A"]*tau, sqrt(sigma["A"]^2L*tau)))))
}

attributes(f)$name = quote(f)
mathout(f)
```

$$f(\tau) = \frac{c}{\mu_A} + \left\{ \frac{1}{\mu_A} - \frac{1}{\mu_A + \mu_B} \cdot \left[(\mu_A \tau - c) \cdot \Phi\left(\frac{c - \mu_A \tau}{\sqrt{\sigma_A^2 \tau}} \right) - (\mu_A \tau + c) \cdot \exp\left(\frac{2\mu_A \cdot \tau}{\sigma_A^2} \right) \cdot \Phi\left(\frac{-c - \mu_A \tau}{\sqrt{\sigma_A^2 \tau}} \right) \right] \right\}$$

But, if the function is rendered in MathML, the wrong parenthesis appears as a curly brace, and we immediately see the mistake. The correct version is shown below for comparison.

```
f <- function(tau)
{ dfrac(c, mu["A"]) + (dfrac(1L, mu["A"]) - dfrac(1L, mu["A"] + mu["B"])) *
    ((mu["A"]*tau - c) * pnorm(dfrac(c - mu["A"]*tau, sqrt(sigma["A"]^2L*tau)))
      - (mu["A"]*tau + c) * exp(dfrac(2L*mu["A"]*tau, sigma["A"]^2L))
        * pnorm(dfrac(-c - mu["A"]*tau, sqrt(sigma["A"]^2L*tau))))
}

attributes(f)$name = quote(f)
mathout(f)
```

$$f(\tau) = \frac{c}{\mu_A} + \left(\frac{1}{\mu_A} - \frac{1}{\mu_A + \mu_B} \right) \cdot \left[(\mu_A \tau - c) \cdot \Phi\left(\frac{c - \mu_A \tau}{\sqrt{\sigma_A^2 \tau}} \right) - (\mu_A \tau + c) \cdot \exp\left(\frac{2\mu_A \cdot \tau}{\sigma_A^2} \right) \cdot \Phi\left(\frac{-c - \mu_A \tau}{\sqrt{\sigma_A^2 \tau}} \right) \right]$$

This example may seem trivial. However, as the reader well knows, a missed parenthesis is often the cause of untraceable errors and contradictory results. Thus, rendering the expression in MathMl improves the readability of the text, allowing better identification of the sources of possible errors.

Errors

An additional feature that this package offers is related to its application in educational scenarios. The functions here reported are intended to identify and highlight possible errors that students make, as well as provide the correct answer for the task.

In this first example the function highlights that the student forgot to add the left-hand side of the expression and it does that by striking the missing part. In this case the letter A and the operator +. Similarly the function `omit_right` does the same for the omissions in the right-hand side of the expression. Finally the function `omit()` can be used, for instance, to highlight the omission in the elements of a list.

```
t <- quote(omit_left(a + b))
mathout(t)
```

$$\textit{cancel}\,(a+)\,b$$

```
t <- quote(omit_right(a + b))
mathout(t)
```

$$a\,\textit{cancel}\,(+\,b)$$

```
t <- list(quote(a), quote(b), quote(omit(c)))
mathout(t)
```

$$a\,b\,\textit{cancel}\,(c)$$

The functions `add_left()`, `add_right()`, `add()` reported below follow the same way of reasoning, but they account for the mistakes made in adding incorrect elements in the expression. They highlight the redundant element placing it in a box.

```
t <- quote(add_left(a + b))
mathout(t)
```

$$\textit{box}\,(a+)\,b$$

```
t <- quote(add_right(a + b))
mathout(t)
```

$$a\,\textit{box}\,(+\,b)$$

```
t <- list(quote(a), quote(b), quote(add(c)), quote(d))
mathout(t)
```

$$a\,b\,\textit{box}\,(c)\,d$$

Finally the function `instead( , )` place a curly bracket under the incorrect element and provides its correct version under the curly bracket. In order to provide the correction with this function one must fill it with its wrong and correct terms in this order: `instead(wrong, correct)`. An example of its use its provided below.

```
t <- quote(instead(a, b) + c)
mathout(t)
```

$$\textit{underbrace}\,(a, \text{insteadof}\,b) + c$$

It is also possible to use flags for context-dependent rendering. The highlight flag is the default one. It shows where the error is and what is its solution. The ignore flag allows the program to ignore the fact that there is an error and print the expression as if it were correct. Finally, the third flag, fix, tells you what is the term that you are supposed to fix without giving the solution. Below the code with its comment is provided.

```
# It highlights what has been omitted
t <- quote(omit_left(a + b))
mathout(t, flags=quote(error(highlight))) # default
```

$$cancel\,(a+)\,b$$

```
# It prints the correct version
mathout(t, flags=quote(error(ignore)))
```

$$a+b$$

```
# It prints the correct version
mathout(t, flags=quote(error(asis)))
```

$$b$$

```
# It highlights the correct version by framing it
mathout(t, flags=quote(error(fix)))
```

$$box\,(a+)\,b$$

```
# It frames the redundant term
t <- quote(add_right(a + b))
mathout(t, flags=quote(error(highlight))) # default
```

$$a\,box\,(+\,b)$$

```
# It prints the correct expression without the redundant term
mathout(t, flags=quote(error(ignore)))
```

$$a$$

```
# It strikes the redundant term
mathout(t, flags=quote(error(fix)))
```

$$a\,cancel\,(+\,b)$$

```
# It highlights that a is wrong and b is correct
t <- quote(instead(a, b) + c)
mathout(t, flags=quote(error(highlight))) # default
```

$$underbrace\,(a, \mathrm{instead of}\, b) + c$$

```
# It provides the correct solution
mathout(t, flags=quote(error(ignore)))
```

$$b+c$$

```
# It provides the correct solution framing the correct term
mathout(t, flags=quote(error(fix)))
```

$$box\,(b)+c$$

Conclusions

This package expands the possibilities of R by allowing calculations and graphical rendition at the same time. This makes it easier for the user to notice writing errors and correct them smoothly. Building on current features of R and existing packages for displaying formulas in R (e.g. Murrell and Ihaka (2000), Allaire et al. (2018)), the **mathml** package bridges the gap between computational needs, presentation of results, and their reproducibility. To do so, it uses the Prolog's grammar rules and translates the R expressions to MathML. Then, two possible solutions for typesetting are available to export R Markdown documents in different formats or ShinyApp webpages: MathML and MathJax.

The article and Appendix show examples of how the package can be used and what operations are currently possible. Additionally, **mathml** allows for user's custom hooks to provide a better definition of the needed functions.

In its current version **mathml** has some limitations. For example, it is still not possible to use the functions of **mathml** for writing intratextual formulas, while it is still more convenient to adopt the usual LateX notation. However, we do not consider this to be a serious limitation, since the most productive use of **mathml** still comes from more complicated calculations. Nevertheless, in the future versions of this package we will address this issue, to broaden its spectrum of possible applications.

mathml is available for R version (...) and later, and can be easily installed using the usual `install.packages("mathml")`. The source code of the package is found at (...) including installation instructions for Unix, Windows and macOS.

Acknowledgments

References

Allaire, JJ, Rich Iannone, Alison Presmanes Hill, and Yihui Xie. 2018. "Distill for r Markdown." https://rstudio.github.io/distill.

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2022. *Shiny: Web Application Framework for r*. https://CRAN.R-project.org/package=shiny.

Green, T. R. G. 1977. "Conditional Program Statements and Their Comprehensibility to Professional Programmers." *Journal of Occupational Psychology* 50: 93–109.

Miller, Jeff. 1986. "Timecourse of Coactivation in Bimodal Divided Attention." *Perception & Psychophysics* 40: 331–43.

Murrell, Paul, and Ross Ihaka. 2000. "An Approach to Providing Mathematical Annotation in Plots." *Journal of Computational and Graphical Statistics* 9: 582–99. https://www.jstor.con/stable/1390947.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ratcliff, Roger, and Gail McKoon. 2008. "The Diffusion Decision Model: Theory and Data for Two-Choice Decision Task." *Neural Computation* 20: 873–922.

Ratcliff, Roger, Philip L. Smith, Scott D. Brown, and Gail McKoon. 2016. "Diffusion Decision Model: Current Issues and History." *Trends in Cognitive Sciences* 20: 260–81.

Sarkar, Deepayan, and Kurt Hornik. 2022. *Enhancements to HTML Documentation*. https://blog.r-project.org/2022/04/08/enhancements-to-html-documentation/index.html.

Schwarz, Wolf. 1994. "Diffusion, Superposition, and the Redundant-Targets Effect." *Journal of Mathematical Psychology* 38: 504–20.

Xie, Y., C. Dervieux, and E. Riederer. 2020. *R Markdown Cookbook*. Cambridge: Chapman; Hall/CRC.

Matthias Gondan
University of Innsbruck
Department of Psychology
Innsbruck, Austria
https://www.uibk.ac.at/psychologie/mitarbeiter/gondan-rochon/index.html.en
ORCiD: 0000-0001-9974-0057
Matthias.Gondan-Rochon@uibk.ac.at

Irene Alfarone
University of Innsbruck
Department of Psychology
Innsbruck, Austria
https://www.uibk.ac.at/psychologie/mitarbeiter/alfarone/index.html.en
ORCiD: 0000-0002-8409-8900
Irene.Alfarone@uibk.ac.at