# Ryacas

## A computer algebra system in R

Mikkel Meyer Andersen and Søren Højsgaard

2019

# Content

# Genesis of Ryacas

[Rd] Computer algebra in R - would that be an idea??

From: Søren Højsgaard Date: Tue 12 Jul 2005 - 11:19:48 GMT

> From time to time people request symbolic computations beyond what D() and deriv() etc can provide. A brief look at the internet shows that there are many more or less developed computer algebra packages freely available. Therefore, I wondered if it would be an idea to try to 'integrate' one of these packages in R, which I guess can be done in more or less elegant ways... I do not know any of the computer algebra people around the World, but perhaps some other people from the R-community do and would be able to/interested in establishing such a connection... Just and idea; comments appreciated :-)

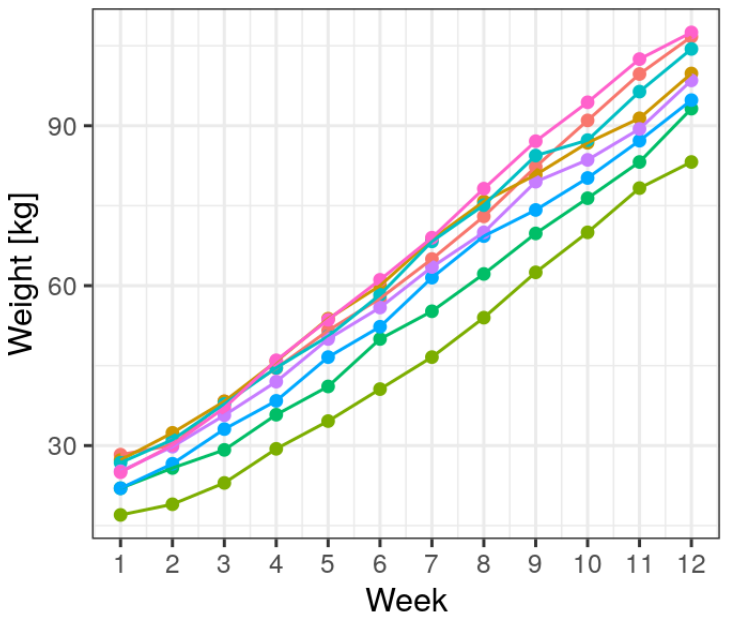
# Original motivation

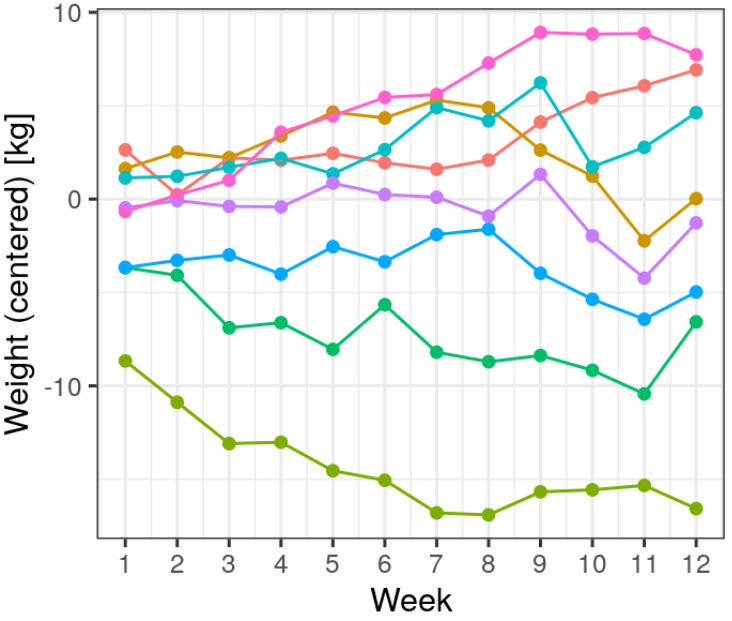
Modelling growth of slaugther pigs under different feed treatments.

One option is random regression model for $i$'th individual:

$$y_{i,t} = b_0 + b_1 t + B_0 + B_1 t + e_{i,t},$$

where $B_0$, $B_1$, $e_{i,t}$ are normal random effects (and $b_0$, $b_1$ are specific to treatment).



# Motivation

Random regression model:

$$y_{i,t} = B_0 + B_1 t + e_{i,t}$$

where $B_0$, $B_1$, $e_{i,t}$ are normal random effects.

Question: What happens to the correlation between measurements on the same unit, $i$, as the distance between measurements increase. That is, what it the limit of

$$cor(y_{i,t}, y_{i,t+k}) \to ?? \quad \text{as} \quad k \to \infty$$

# Background

- First version: `yacasR`; only on Windows; communication between R and yacas based client/server technology on local machine.
- Changed name to `Ryacas`; more contributors:
  - Rob Goedman
  - Gabor Grothendieck
  - **Søren Højsgaard**
  - Ayal Pinkus
  - Grzegorz Mazur
- Appeared on CRAN around 2009.

# Background

- Has lived a quiet life until recently (2017) when **Mikkel Meyer Andersen** became maintainer
    - `XML` replaced by `xml2`, then skip XML communication with `yacas`'s `RForm()`
- JOSS paper Andersen and Højsgaard (2019) Ryacas: A computer algebra system in R
- Synergy in open source projects
    - Bugs in `yacas` (`Solve()`, `()`)
    - New features in `yacas` (`SumFunc()`, `Solve()`, `RForm()`, `Variables()`)
- Other ressources: Wolfram Alpha

# Getting started

# Getting started

```
library(Ryacas)
```

- Stable version: https://CRAN.R-project.org/package=Ryacas
- Development version: https://github.com/mikldk/ryacas/
- Online documentation: http://mikldk.github.io/ryacas/

This talk was made using the development version available at https://github.com/mikldk/ryacas/:

```
devtools::install_github("mikldk/ryacas")
```

```
packageVersion("Ryacas")
```

```
## [1] '1.1.1.9005'
```

# yacas symbols

```
x <- ysym("x")
y <- ysym("y")
eq <- x^2 + 3*x + 4*y + y^4
eq
```

```
## y: x^2+3*x+4*y+y^4
```

# ...is just a class with a `yacas` command (for now)

```
str(eq)
```

```
## List of 3
##  $ yacas_cmd: chr "x^2+3*x+4*y+y^4"
##  $ is_mat   : logi FALSE
##  $ is_vec   : logi FALSE
```

```
class(eq)
```

```
## [1] "yac_symbol" "list"
```

```
as_r(eq)
```

```
## expression(x^2 + 3 * x + 4 * y + y^4)
```

```
as.character(eq)
```

```
## [1] "x^2+3*x+4*y+y^4"
```

# Generic functions

```
methods(class = "yac_symbol")
```

```
##  [1] [            [[           [<-          %*%          as_r
##  [6] as_y         as.character c            cbind        deriv
## [11] diag         diag<-       dim          Hessian      integrate
## [16] Jacobian     length       lim          lower.tri    Math
## [21] Ops          print        prod         rbind        simplify
## [26] solve        str          sum          t            tex
## [31] upper.tri    with_value   y_fn         y_rmvars     yac_assign
## [36] yac_expr     yac_silent   yac_str      yac
## see '?methods' for accessing help and source code
```

# Derivatives

```
deriv(eq, "x") # eq = x^2+3*x+4*y+y^4
```

```
## y: 2*x+3
```

```
H <- Hessian(eq, c("x", "y"))
H
```

```
## {{     2,      0},
##  {     0, 12*y^2}}
```

```
eval(as_r(H), list(x = 1, y = 1))
```

```
##      [,1] [,2]
## [1,]    2    0
## [2,]    0   12
```

```
tex(H)
```

```
## [1] "\\left( \\begin{array}{cc} 2 & 0 \\\\ 0 & 12 y ^{2} \\end{array} \\
```

# Solving equations

Syntax:

```
solve(lhs, rhs, vars)
```

A brief example:

```
sol <- solve(x^2, -1, "x")
sol
```

```
## {x==Complex(0, 1), x==Complex(0, -1)}
```

```
y_rmvars(sol)
```

```
## {Complex(0, 1), Complex(0, -1)}
```

```
as_r(y_rmvars(sol))
```

```
## [1] 0+1i 0-1i
```

Another example finding roots:

```
eq
```

```
## y: x^2+3*x+4*y+y^4
```

```
roots <- solve(eq, 0, "x") # shorthand rhs == 0: solve(eq, "x")
roots
```

```
## {x==(Sqrt(9-4*(4*y+y^4))-3)/2, x==(-(Sqrt(9-4*(4*y+y^4))+3))/2}
```

```
tex(roots)
```

```
## [1] "\\left( x = \\frac{\\sqrt{9 - 4 \\left( 4 y + y ^{4}\\right) } - 3}
```

```
### $$\begin{align}`r tex(roots)`\end{align}$$
```

$$\begin{align}\left( x = \frac{\sqrt{9 - 4 \left( 4 y + y ^{4}\right) } - 3}{2}, x = \frac{- \left( \sqrt{9 - 4 \left( 4 y + y ^{4}\right) } + 3\right)}{2} \right)\end{align}$$

# Solving a system of non-linear equations

```
x <- ysym("x")
y <- ysym("y")
lhs <- c(3*x*y - y, x)
rhs <- c(-5*x, y+4)
```

$$3xy - y = -5x$$
$$x = y + 4$$

```
sol <- solve(lhs, rhs, c("x", "y"))
sol
```

```
## {{      x==2,    y==(-2)},
##  {    x==2/3, y==(-10)/3}}
```

```
as_r(y_rmvars(sol))
```

```
##            [,1]      [,2]
## [1,] 2.0000000 -2.000000
## [2,] 0.6666667 -3.333333
```

# Integrals

Syntax:

```
integrate(expr, var)
integrate(expr, var, from, to)
```

```
x <- ysym("x")
res <- integrate(1/x, "x")
res
```

```
## y: Ln(x)
```

```
res <- integrate(sin(x)^2, "x")
res
```

```
## y: (x+Sin((-2)*x)/2)/2
```

```
as_r(res)
```

```
## expression((x + sin(-2 * x)/2)/2)
```

# Sums

Syntax:

```
sum(expr, var, from, to)
```

```
x <- ysym("x")
k <- ysym("k")
res <- sum(x^k, "k", 0, "n")
res
```

```
## y: (1-x^(n+1))/(1-x)
```

# Sums

```
k <- ysym("k")
res <- sum(1/k^2, "k", 1, Inf)
res
```

```
## y: Pi^2/6
```

```
as_r(res)
```

```
## [1] 1.644934
```

# Limits

Syntax:

```
lim(expr, var, to)
lim(expr, var, to, from_left = TRUE)
lim(expr, var, to, from_right = TRUE)
```

```
x <- ysym("x")
lim(sin(x)/x, "x", 0)
```

```
## y: 1
```

```
lim(1/x, "x", 0)
```

```
## y: Undefined
```

```
lim(1/x, "x", 0, from_left = TRUE)
```

```
## y: -Infinity
```

```
lim(1/x, "x", 0, from_right = TRUE)
```

```
## y: Infinity
```

# Linear algebra

```
A <- ysym(matrix(c(2, 1, 4, "x"), 2, 2))
A
```

```
## {{2, 4},
##  {1, x}}
```

```
t(A)
```

```
## {{2, 1},
##  {4, x}}
```

```
A[1, ]
```

```
## {2, 4}
```

```
# solve(A)
y_fn(A, "Determinant") # yacas function Determinant()
```

```
## y: 2*x-4
```

# Linear algebra

```
b <- ysym(c("x", 3))
y <- A %*% b
y
```

```
## {2*x+12, 4*x}
```

```
solve(A, y)
```

```
## {((2*x+12)*x-16*x)/(2*x-4), (8*x-(2*x+12))/(2*x-4)}
```

```
solve(A, y) %>% simplify()
```

```
## {x, 3}
```

```
b
```

```
## {x, 3}
```

## High-level interface (for everyday use)

- yacas symbols (ysym())
- R syntax (*, %*%, [, [<-, solve(), etc.)

## Low-level interface (for advanced stuff)

- yac_str(x): Run yacas command x and return string

```
yac_str("Determinant({{2, 4}, {1, x}})")
```

```
## [1] "2*x-4"
```

- yac_expr(): Run yacas command x and return R expression

```
yac_expr("Determinant({{2, 4}, {1, x}})")
```

```
## expression(2 * x - 4)
```

See the documentation for yacas at https://yacas.readthedocs.io/.

# Example: Multinomial likelihood

# Multinomial likelihood

The multinomial likelihood for three categories is

$$\text{lik}(p \mid y) \propto p_1^{y_1} p_2^{y_2} p_3^{y_3}.$$

Taking log we get

$$\begin{aligned} l(p) &= y_1 \log(p_1) + y_2 \log(p_2) + y_3 \log(p_3) \\ g(p) &= p_1 + p_2 + p_3 - 1. \end{aligned}$$

Maximise $l(p)$ for $g(p) = 0$ using Lagrange multiplier to get the unconstrainted optimisation problem

$$L = -l(p) + \lambda g(p)$$

```
p <- ysym(c("p1", "p2", "p3"))
y <- ysym(c("y1", "y2", "y3"))
lambda <- ysym("lambda")
l <- sum(y*log(p))
L <- -l + lambda*(sum(p) - 1)
L
```

```
## y: lambda*(p1+p2+p3-1)-(y1*Ln(p1)+y2*Ln(p2)+y3*Ln(p3))
```

# Multinomial likelihood

```
gL <- deriv(L, c("p1", "p2", "p3"))
gL
```

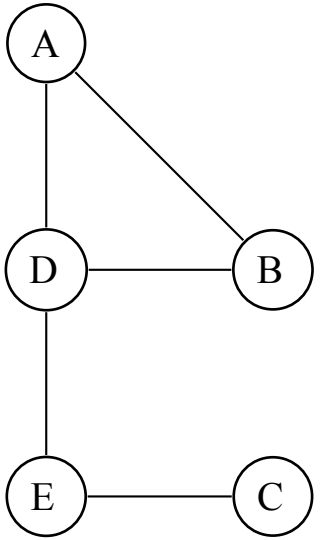
```
## {lambda-y1/p1, lambda-y2/p2, lambda-y3/p3}
```

- 3 equations by setting partial derivatives to 0
- 1 equation from the constraint

```
sol <- solve(c(gL, sum(p)),
             c(rep(0, 3), 1),
             c("p1", "p2", "p3", "lambda"))
sol
```

```
## {{p1==y1/lambda, p2==y2/lambda, p3==y3/lambda, lambda==y1+y2+y3}}
```

# Example: Graphical models




- Multivariate models with focus on conditional independence (CI) restrictions.
- Suppose $X = (X_A, X_B, \ldots, X_E) \sim N_5(0, \Sigma)$ .
- The key to interpretation is the concentration matrix $K = \Sigma^{-1}$
- $K_{ij} = 0$ iff $X_i$ and $X_j$ are independent given all other $X$s
- A missing edge reflects a CI-restriction.

- For example $X_D$ and $X_C$ are independent given all other $X$s
- A practical approach: $X_C$ and $X_D$ may be correlated. However
  - Regress $X_D$ on $X_A, X_B, X_E$ and extract residuals $r_D$.
  - Regress $X_C$ on $X_A, X_B, X_E$ and extract residuals $r_C$.
  - If the resuduals $r_D$ and $r_C$ are uncorrelated then we have the CI.

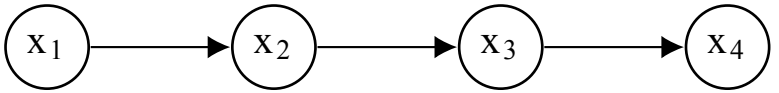
Many classical statistical models are built up on "modularization" based on CI-restrictions:

Examples:

- Autoregressive models
- State-space models / dynamic linear models
- Naive Bayesian models
- Probabilistic Principal Components

It is illustrative to see the connection between these model classes and CI-restrictions in the multivariate normal distribution.

# Autoregression, AR(1)




Consider $AR(1)$ process: $x_i = ax_{i-1} + e_i$ where $i = 2, 3, 4$ and where $x_1 = e_1$.

$$
\begin{aligned}
x_1 &= e_1 \\
x_2 &= ax_1 + e_2 \\
x_3 &= ax_2 + e_3 \\
x_4 &= ax_3 + e_4
\end{aligned}
$$

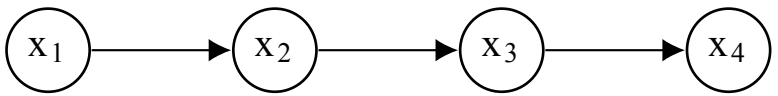
Let $e = (e_1, \ldots, e_4)$ and $x = (x_1, \ldots x_4)$. Isolating error terms gives that

$$
e = \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -a & 1 & 0 & 0 \\ 0 & -a & 1 & 0 \\ 0 & 0 & -a & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = Lx
$$

Define $L$:

```
# Or rbind(c(1, 0, 0, 0), ...)
L <- ysym("{{1,0,0,0},{-a,1,0,0},{0,-a,1,0},{0,0,-a,1}}")
L
```

```
## {{ 1,  0,  0,  0},
##  {-a,  1,  0,  0},
##  { 0, -a,  1,  0},
##  { 0,  0, -a,  1}}
```




If error terms $e_i$ are independent and $N(0, v^2)$ then $e = Lx$ gives

- $\mathbf{Var}(e) = v^2 I = L\mathbf{Var}(x)L'$
- Hence the covariance matrix of $x$ is $V = \mathbf{Var}(x) = v^2 L^{-1}(L^{-1})'$
- The concentration matrix is $K = V^{-1} = LL'/v^2$

We skip $v^2$ in the following (set $v^2 = 1$)

```
Li <- solve(L)
Li
```

```
## {{  1,   0,   0,   0},
##  {  a,   1,   0,   0},
##  {a^2,   a,   1,   0},
##  {a^3, a^2,   a,   1}}
```

```
V <- Li %*% t(Li)
K <- L %*% t(L)
```

$$K = \begin{pmatrix} 1 & -a & 0 & 0 \\ -a & a^2+1 & -a & 0 \\ 0 & -a & a^2+1 & -a \\ 0 & 0 & -a & a^2+1 \end{pmatrix}$$

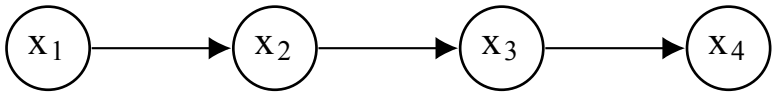
$$V = \begin{pmatrix} 1 & a & a^2 & a^3 \\ a & a^2+1 & a^3+a & a^4+a^2 \\ a^2 & a^3+a & a^4+a^2+1 & a^5+a^3+a \\ a^3 & a^4+a^2 & a^5+a^3+a & a^6+a^4+a^2+1 \end{pmatrix}$$

The pattern of 0s in $K$ is important:

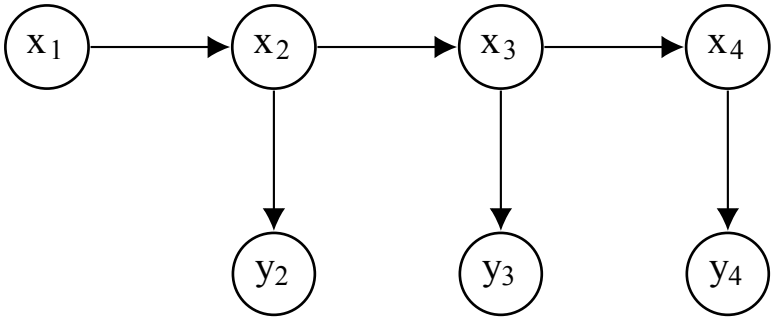
```
K
```

```
## {{    1,    -a,     0,     0},
##  {   -a, a^2+1,    -a,     0},
##  {    0,    -a, a^2+1,    -a},
##  {    0,     0,    -a, a^2+1}}
```

$K_{ij} = 0$ iff $x_i$ is independent of $x_j$ given all other $x$'s



# State space models




Same story, except that the setting is slightly more involved:

$$x_i = ax_{i-1} + e_i$$
$$y_i = bx_i + u_i$$

This gives a different $L$, but the method to find the concentration matrix is as before.

# Further down the road

# Further down the road

Symbolic math in `R` is limited to finding derivatives.

With `Ryacas` we have added a bit (still in `R` syntax thanks to `S3`)

- Integration
- Finding limits
- Simplifying expressions
- Solving (potentially systems of) non-linear equations
- Linear algebra
- Technical properties
  - No `SystemRequirements`
  - `R` expressions thanks to `RForm()` in `yacas`
  - `LaTeX` format thanks to `TeXForm()` in `yacas`

It may well be that in the future another engine than `yacas` should be invoked.

# Original motivation - revisited

Random regression model:

$$y_t = B_0 + B_1 t + e_t$$

where $B = (B_0, B_1) \sim N(0, \Sigma)$, $e_t \sim N(0, w^2)$ are independent normal random effects.

What is:

$$cor(y_t, y_{t+k}) \to ?? \quad \text{as} \quad k \to \infty$$

# Solution

From:

$$y_t = B_0 + B_1 t + e_t = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} + e_t$$

We get:

$$\mathbf{Var}(y_t) = \mathbf{Var}\left( \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \end{bmatrix} \right) + \mathbf{Var}\left(e_t\right) = \begin{bmatrix} 1 & t \end{bmatrix} \Sigma \begin{bmatrix} 1 & t \end{bmatrix}^\intercal + w^2.$$

Similarly:

$$\mathbf{Var}(y_{t+k}) = \begin{bmatrix} 1 & t+k \end{bmatrix} \Sigma \begin{bmatrix} 1 & t+k \end{bmatrix}^\intercal + w^2.$$

Finally:

$$\begin{aligned} \mathbf{Cov}(y_t, y_{t+k}) &= \begin{bmatrix} 1 & t \end{bmatrix} \Sigma \begin{bmatrix} 1 & t+k \end{bmatrix}^\intercal \\ \mathbf{Cor}(y_t, y_{t+k}) &= \frac{\mathbf{Cov}(y_t, y_{t+k})}{\sqrt{\mathbf{Var}(y_t)\mathbf{Var}(y_{t+k})}}. \end{aligned}$$

# Formulation in Ryacas

```
ysym_make(c("v1", "v2", "c12"))
```

```
## Warning in FUN(X[[i]], ...): v1 already exists, skipping
```

```
## Warning in FUN(X[[i]], ...): v2 already exists, skipping
```

```
## Warning in FUN(X[[i]], ...): c12 already exists, skipping
```

```
S <- rbind(c(v1, c12), c(c12, v2))
ysym_make(c("w" = "w", "xt" = "{{1, t}}", "xtk" = "{{1, t+k}}"))
```

```
## Warning in FUN(X[[i]], ...): w already exists, skipping
```

```
## Warning in FUN(X[[i]], ...): xt already exists, skipping
```

```
## Warning in FUN(X[[i]], ...): xtk already exists, skipping
```

```
v_t <- (xt %*% S %*% t(xt) + w^2)[1, 1]
v_t
```

```
## y: w^2+v1+t*c12+(c12+t*v2)*t
```

# Finding the limit

Now the progress stops

```
# lim Cor(y_t, y_{t+k}) for k -> Inf
lim(cv_t_tk / sqrt(v_t * v_tk), "k", Inf)
```

```
## Error in yac_core(x): Yacas returned this error: CommandLine(1) : Max eva
## Please use MaxEvalDepth to increase the stack size as needed.
```

```
k <- ysym("k")
lim((cv_t_tk / k) / sqrt((v_t * v_tk) / k^2), "k", Inf)
```

```
## Error in yac_core(x): Yacas returned this error: CommandLine(1) : Max eva
## Please use MaxEvalDepth to increase the stack size as needed.
```

(https://github.com/grzegorzmazur/yacas/issues/282)

# Another CAS

```
library(symr) # Based on SymPy; work in progress...
num <- paste0("(", as.character(cv_t_tk), ")/k")
den <- paste0("sqrt(((", as.character(v_t), ") * (",
              as.character(v_tk), ")/ k^2))")
num <- gsub("^", "**", num, fixed = TRUE) # Python's way
den <- gsub("^", "**", den, fixed = TRUE)
sympy <- symr::get_sympy()
res <- sympy$limit(paste0(num, "/", den), "k", "oo")
res
```

```
## (c12 + t*v2)/sqrt(v2*(c12*t + t*(c12 + t*v2) + v1 + w**2))
```

```
sympy$latex(res)
```

```
## [1] "\\frac{c_{12} + t v_{2}}{\\sqrt{v_{2} \\left(c_{12} t + t \\left(c_
```

$$cor(y_t, y_{t+k}) \rightarrow \frac{c_{12} + t v_2}{\sqrt{v_2 \left(c_{12} t + t \left(c_{12} + t v_2\right) + v_1 + w^2\right)}} \quad \text{as} \quad k \rightarrow \infty$$

More in 2020...

# References

- Yacas (**y**et **a**nother **c**omputer **a**lgebra **s**ystem): http://www.yacas.org/
  - Online docs: http://yacas.readthedocs.org/
- Stable version: https://CRAN.R-project.org/package=Ryacas
- Development version: https://github.com/mikldk/ryacas/
  - Online docs: http://mikldk.github.io/ryacas/