Example with rSymPy

M Portela

1/9/2021

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

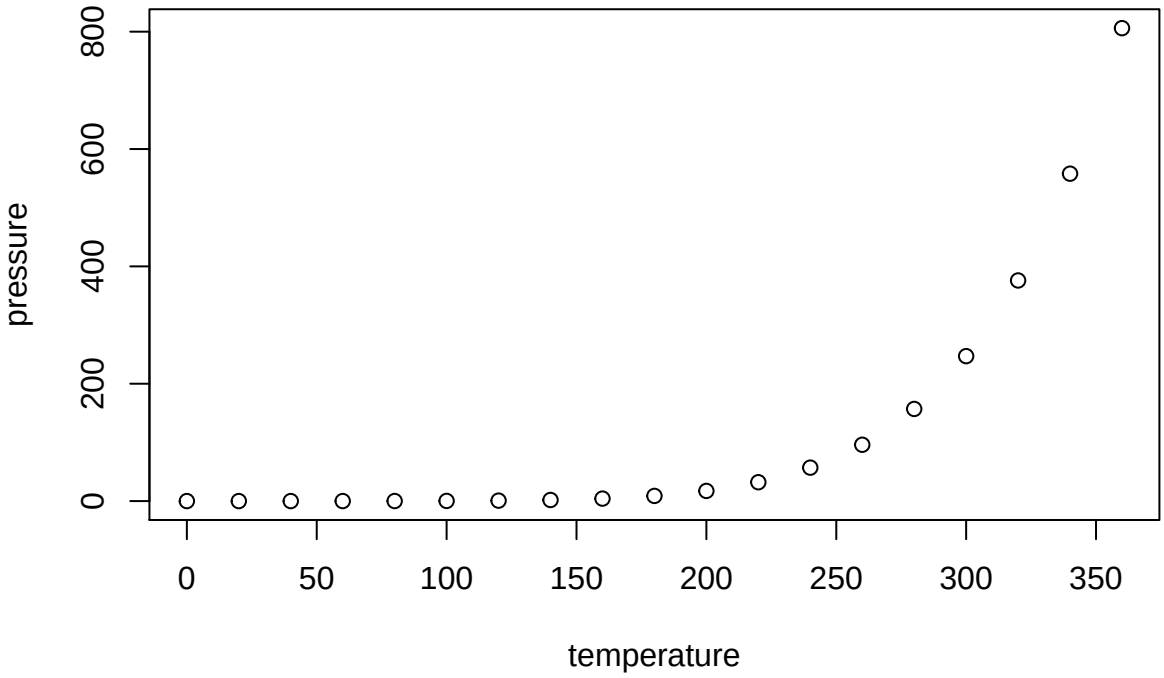
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

```
# https://rdrr.io/cran/rSymPy/

library(rSymPy)
```

```
## Loading required package: rJython
```

```
## Loading required package: rJava
```

```
## Loading required package: rjson
```

```
# create a SymPy variable called x
sympy("var('x')")
```

```
## [1] "x"
```

```
sympy("y = x*x")
```

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

```
sympy("y")
```

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

```
sympy("limit(1/x, x, oo)")
```

```
## [1] "0"
```

```
# the next line fails under jython even without R
# and seems to corrupt the rest of the session
# sympy("(1/cos(x)).series(x, 0, 10)")

sympy("diff(sin(2*x), x, 1)")
```

```
## [1] "2*cos(2*x)"
```

```
sympy("diff(sin(2*x), x, 2)")
```

```
## [1] "-4*sin(2*x)"
```

```
sympy("integrate(exp(-x), (x, 0, oo))")
```

```
## [1] "1"
```

```
sympy("xr = Symbol('xr', real=True)")
```

```
## [1] "xr"
```

```
sympy("exp(I*xr).expand(complex=True)")
```

```
## [1] "I*sin(xr) + cos(xr)"
```

```
# Matrices are stored row by row (unlike R matrices)
cat(sympy("A = Matrix([[1,x], [y,1]])"), "\n")
```

```
## [   1, x]
## [x**2, 1]
```

```
cat(sympy("A**2"), "\n")
```

```
## [1 + x**3,      2*x]
## [  2*x**2, 1 + x**3]
```

```
## End(Not run)
```

```
theta <- Var("theta")

sin.series <- function(n) sympy(paste0(  "sin(theta).series(theta, 0, ", n, ")"))
cos.series <- function(n) sympy(paste0(  "cos(theta).series(theta, 0, ", n, ")"))
exp.series <- function(n) sympy(paste0("exp(I*theta).series(theta, 0, ", n, ")"))

sympy("var('x')")
```

```
## [1] "x"
```

```
sympy("y = sin(2*x)")
```

```
## [1] "sin(2*x)"
```

```
sympy("y")
```

```
## [1] "sin(2*x)"
```

```
bb <- sympy("y")
aa <- sympy("diff(y, x, 1)")
```

The first order derivative of $sin(2 * x)$ is

$$2 * cos(2 * x) \tag{1}$$

or fsfsfs $2 * cos(2 * x)$

```
# https://kevinkotze.github.io/mm-tut1-symbolic/

hh <- Var("c_t")
hh <- Var("n_t")
hh <- Var("sigma_p")
hh <- Var("gamma_p")
hh <- Var("lambda_t")
hh <- Var("p_t")
hh <- Var("w_t")
hh <- Var("r_t")
hh <- Var("k_t")
hh <- Var("k_1t")
hh <- Var("delta_p")
hh <- Var("pi_t")

sympy("Eq((c_t**(1 - sigma_p))/(1 - sigma_p) - (n_t**(1 + gamma_p))/(1 + gamma_p) - lambda_t*(p_t*c_t +
```

```
## [1] "-lambda_t*(-pi_t + c_t*p_t + k_1t*p_t - k_t*r_t - n_t*w_t - k_t*p_t*(1 - delta_p)) + c_t**(1 - s
```

$-lambda_t * (-pi_t + c_t * p_t + k_1 t * p_t - k_t * r_t - n_t * w_t - k_t * p_t * (1 - delta_p)) + c_t * *(1 - sigma_p)/(1 - sigma_p) - n_t * *(1 + gamma_p)/(1 + gamma_p) == 0$

```
library(Deriv)
library(mosaic)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
##
## Attaching package: 'mosaic'
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
## The following object is masked from 'package:Matrix':
##
##     mean
## The following object is masked from 'package:ggplot2':
##
##     stat
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
library(mosaicCalc)
```

```
## Loading required package: mosaicCore
##
## Attaching package: 'mosaicCore'
## The following objects are masked from 'package:dplyr':
##
##     count, tally
##
## Attaching package: 'mosaicCalc'
## The following object is masked from 'package:stats':
##
##     D
```

```
library(rSymPy)

myf1=function(x){1-exp(-lambda*x)}
print(Deriv(myf1))
```

```
## function (x)
## lambda * exp(-(lambda * x))
```