The following code will be used throughout the complete tutorial

```
#Estimation of the derivative of Relative Risk
library(caracas)

#Parser for Sympy (you need sympy version 1.10)
#check sympy_version() to see you have the right one
sympy            <- get_sympy()
Parse            <- sympy$parsing$sympy_parser$parse_expr
RandomSymbol     <- sympy$stats$rv$RandomSymbol
Variance         <- sympy$stats$Variance
Covariance       <- sympy$stats$Covariance
Symbol           <- sympy$Symbol
Lambdify         <- sympy$utilities$lambdify
Simplify         <- sympy$simplify
Derivative       <- sympy$derive_by_array
```

Relative Risk

To estimate the relative risk we need to define it as a function of symbols

```
#In "Sympy" you need to declare variables for algebra as symbols
#and random variables as random symbols
p1     <- Symbol('p1', positive=T);
p2     <- Symbol('p2', positive=T);
N      <- Symbol('N', positive=T, integer=T) #Sample size
p1_hat <- RandomSymbol('p1_hat');
p2_hat <- RandomSymbol('p2_hat');

#List variables for parse
variable_list <- list('p1' = p1, 'p2' = p2, 'p1_hat' = p1_hat,
                      'p2_hat' = p2_hat, "N" = N)
```

Recall that in the case of estimating the Relative Risk: $RR = \frac{p_1}{p_2}$ we use the estimator:

$$\widehat{RR} = \frac{\hat{p}_1}{\hat{p}_2}.$$

As $\hat{p_1}$ and $\hat{p_2}$ are random variables thay have the following variance:

$$\text{Var}\big[\hat{p}_i\big] = \frac{p_i(1-p_i)}{N} \qquad \text{for } i = 1, 2.$$

which is specified as follows:

```
var_p1_hat <- Parse("p1*(1 - p1)/N", local_dict = variable_list)
var_p2_hat <- Parse("p2*(1 - p2)/N", local_dict = variable_list)
```

Finally, we create the log Relative Risk and calculate its derivative:

```
#We are working with the variance of log_RR
log_RR        <- Parse("log(p1/p2)", local_dict = variable_list)

#Obtain the gradient of log RR
g_RR          <- Derivative(log_RR,  list(p1, p2))

variable_list <- append(variable_list, list("g_RR" = g_RR))
```

```r
g_RR          <- Parse("Matrix(g_RR)", local_dict = list("g_RR" = g_RR))

#Direction vector (horizontal vector)
v <- Parse("Matrix([p1_hat - p1, p2_hat - p2])", local_dict = variable_list)

#Compute inner product
hadamard <- g_RR$dot(v)
print(hadamard)
```

```
## -(-p2 + p2_hat)/p2 + (-p1 + p1_hat)/p1
```

The variance of that gradient is given as follows:

```r
#Get the variance of gradient
var_log_rr <- Variance(hadamard)$expand() |> Simplify()
print(var_log_rr)
```

```
## Variance(p2_hat)/p2**2 - 2*Covariance(p1_hat, p2_hat)/(p1*p2) + Variance(p1_hat)/p1**2
```

Further simplifications are allowed that result in a cleaner expression:

```r
#Recall the covariance is 0 due to independence
var_log_rr <- var_log_rr$subs(Covariance(p1_hat, p2_hat), 0)

#Assign the variances of p1_hat and p2_hat
var_log_rr <- var_log_rr$subs(Variance(p1_hat), var_p1_hat)
var_log_rr <- var_log_rr$subs(Variance(p2_hat), var_p2_hat)

#This is the final expression for the variance
print(var_log_rr)
```

```
## (1 - p2)/(N*p2) + (1 - p1)/(N*p1)
```

Finally we transform the symbolic expression into an `R` function:

```r
#Transform the symbolic algebra into an R function
variable_list     <- append(variable_list, list("var_log_rr" = var_log_rr))
variance_function <- Parse("lambdify((p1,p2,N), var_log_rr)",
                           local_dict = variable_list)
```

The function can be evaluated for different values:

```r
#You can use the variance function to estimate the IF with data
variance_function(p1 = 0.7, p2 = 0.4, N = 100)
```

```
## [1] 0.01928571
```

```r
variance_function(p1 = 0.3, p2 = 0.5, N = 500)
```

```
## [1] 0.006666667
```

```r
variance_function(p1 = 0.1, p2 = 0.1, N = 2)
```

```
## [1] 9
```