

Introdução ao Rcpp: Construindo Funções

Erick Amorim

Departamento de Estatística
Universidade Federal de Minas Gerais

10 de Junho de 2016

SUMÁRIO

Rcpp

RcppArmadillo

Rcpp

O **Rcpp** apareceu pela 1ª vez em 2005 com a contribuição de Dominick Samperi ao pacote chamado de **RQuantLib** e tornou-se um pacote do CRAN no início de 2006.

- ▶ O **Rcpp** é um pacote do **R** que funciona como uma extensão do **R** com funções do **C++**
- ▶ O foco central do **Rcpp** sempre esteve em ajudar o programador em adicionar mais facilmente funções baseadas em **C++**.
- ▶ O pacote também permite que o programador retorne resultados obtidos no **C++** para o **R**.
- ▶ Um pouco do conhecimento de **C++** é muito útil embora não seja estrito.

Funções no Rcpp

- ▶ Uma preocupação de interesse é que algumas funções criadas no R não são tão “leves”;
- ▶ E isso torna o recurso de chamar essas funções pouco atra-
tivo. Veja por exemplo o tempo de processamento de algu-
mas funções simples:

Exemplo 1

```
15 library("microbenchmark")
16 #funcao1
17 f_if=function(x,a,b){
18   ifelse(x<=a,a,ifelse(x>=b,b,x))
19 }
20 #funcao2
21 f_pm=function(x,a,b){
22   pmax(pmin(x,b),a)
23 }
24 #funcao3
25 f_colchet=function(x,a,b){
26   x[x<=a]=a
27   x[x>=a]=b
28   x
29 }
```

Exemplo 1

```
31 x=runif(1000,-1,1)
32 microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchete(x,-1,1),times=1000)
33
```

Console C:/Users/3047/Desktop/ ↗

```
> microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchete(x,-1,1),times=1000)
```

Unit: microseconds

	expr	min	lq	mean	median	uq	max	neval
	f_if(x, -1, 1)	180.317	182.775	208.32391	183.6960	184.925	1443.461	1000
	f_pm(x, -1, 1)	64.816	66.967	72.94949	70.0385	71.267	1204.471	1000
	f_colchete(x, -1, 1)	27.953	29.183	34.75415	30.4120	31.026	1175.288	1000

```
> |
```

Exemplo 1

```
31 x=runif(1000,-1,1)
32 microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchet(x,-1,1),times=1000)
33
```

Console C:/Users/3047/Desktop/ ↗

```
> microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchet(x,-1,1),times=1000)
```

Unit: microseconds

	expr	min	lq	mean	median	uq	max	neval
	f_if(x, -1, 1)	180.317	182.775	208.32391	183.6960	184.925	1443.461	1000
	f_pm(x, -1, 1)	64.816	66.967	72.94949	70.0385	71.267	1204.471	1000
	f_colchet(x, -1, 1)	27.953	29.183	34.75415	30.4120	31.026	1175.288	1000

```
> |
```

Exemplo 1

```
31 x=runif(1000,-1,1)
32 microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchet(x,-1,1),f_if_Rcpp(x,-1,1),times=1000)
33
```

Console C:/Users/3047/Desktop/ ↗

```
> microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchet(x,-1,1),f_if_Rcpp(x,-1,1),times=1000)
```

Unit: microseconds

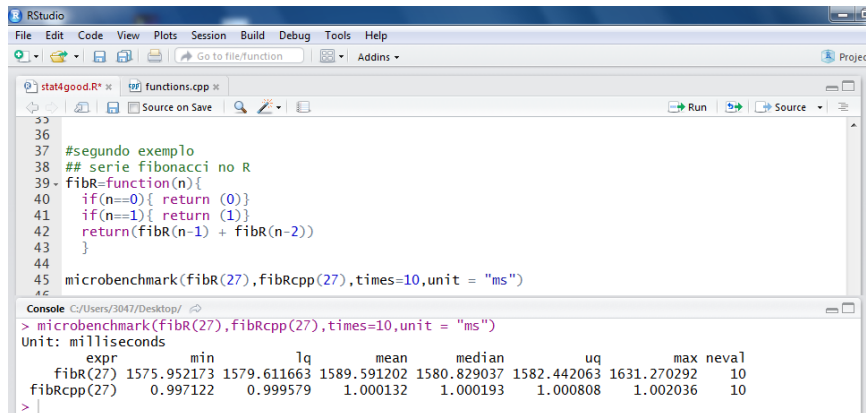
	expr	min	lq	mean	median	uq	max	neval
	f_if(x, -1, 1)	180.317	182.775	248.044721	184.003	185.233	37396.045	1000
	f_pm(x, -1, 1)	65.123	67.581	69.679802	70.345	71.267	92.155	1000
	f_colchet(x, -1, 1)	27.954	29.490	34.485040	30.412	30.719	1100.643	1000
	f_if_Rcpp(x, -1, 1)	5.222	6.451	8.488291	7.679	8.294	1057.944	1000

```
>
```


Exemplo 2

$$F_0 = 0; F_1 = 1;$$

$$F_n = F_{n-1} + F_{n-2}$$



The screenshot shows the RStudio interface with a C++ file named `functions.cpp` open. The code defines a Fibonacci function `fibR` and benchmarks it against a C++ implementation `fibRcpp` using `microbenchmark`. The console output shows the results of the benchmark in milliseconds.

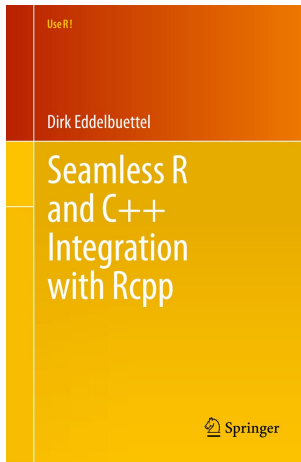
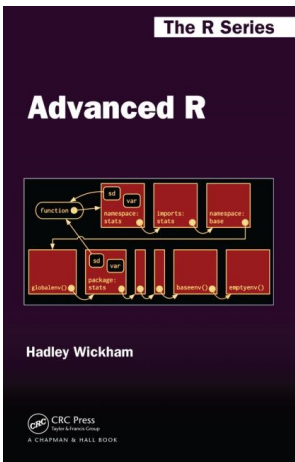
```
35
36
37 #segundo exemplo
38 ## serie fibonacci no R
39 fibR=function(n){
40   if(n==0){ return (0)}
41   if(n==1){ return (1)}
42   return(fibR(n-1) + fibR(n-2))
43 }
44
45 microbenchmark(fibR(27),fibRcpp(27),times=10,unit = "ms")
46
```

Console output:

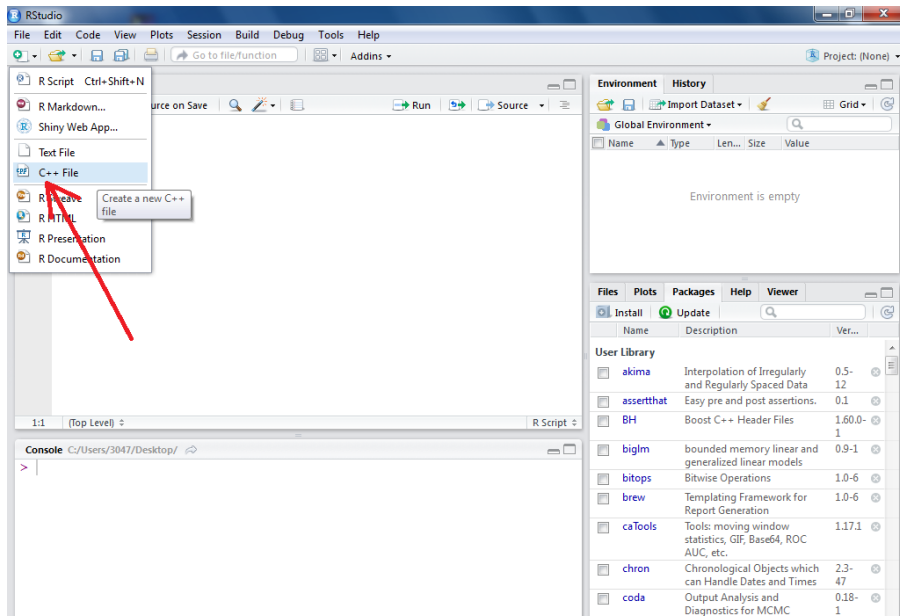
```
> microbenchmark(fibR(27),fibRcpp(27),times=10,unit = "ms")
Unit: milliseconds
```

expr	min	1q	mean	median	uq	max	neval
fibR(27)	1575.952173	1579.611663	1589.591202	1580.829037	1582.442063	1631.270292	10
fibRcpp(27)	0.997122	0.999579	1.000132	1.000193	1.000808	1.002036	10

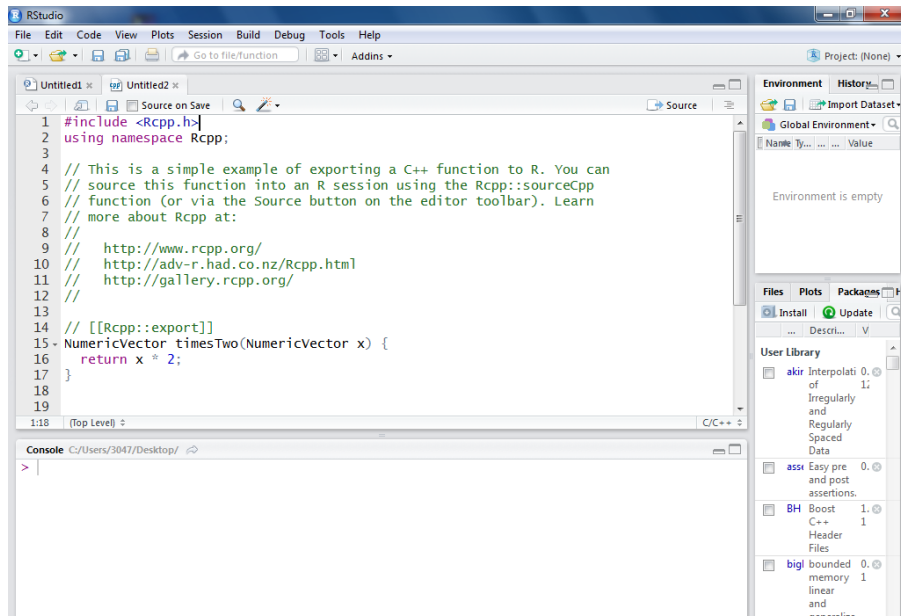
Livros Para Consultas



Como Usamos ??



Apresentação



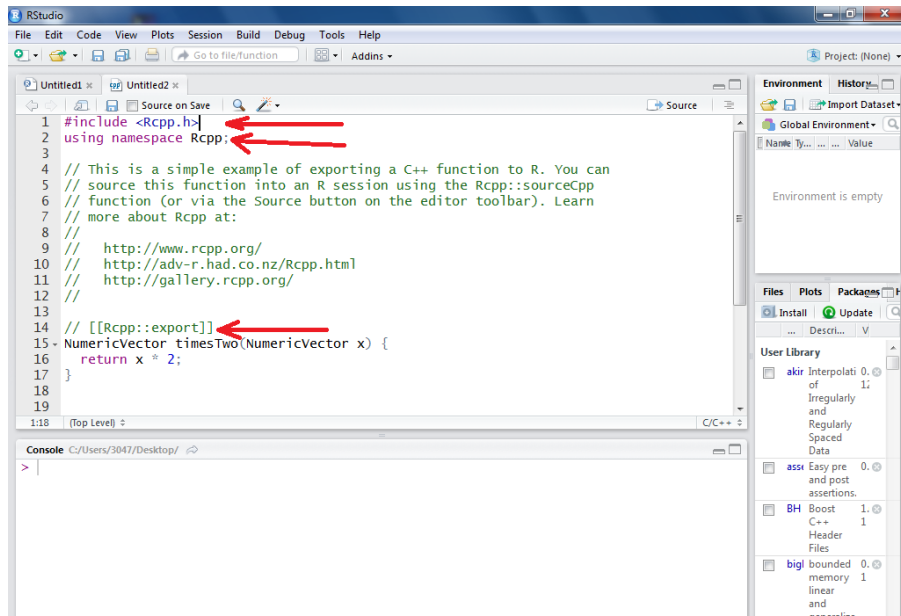
The screenshot displays the RStudio integrated development environment. The main editor window shows a C++ source file with the following code:

```
1 #include <Rcpp.h>
2 using namespace Rcpp;
3
4 // This is a simple example of exporting a C++ function to R. You can
5 // source this function into an R session using the Rcpp::sourceCpp
6 // function (or via the Source button on the editor toolbar). Learn
7 // more about Rcpp at:
8 //
9 // http://www.rcpp.org/
10 // http://adv-r.had.co.nz/Rcpp.html
11 // http://gallery.rcpp.org/
12 //
13
14 // [[Rcpp::export]]
15 NumericVector timesTwo(NumericVector x) {
16   return x * 2;
17 }
18
19
```

The console at the bottom shows the prompt `>` and the file path `C:/Users/3047/Desktop/`. The right-hand sidebar contains the Environment, History, and Packages panels. The Environment panel shows "Global Environment" and "Environment is empty". The Packages panel shows the "User Library" with the following installed packages:

Package Name	Version
akir	Interpolati 0.0
assx	Easy pre and post assertions. 0.0
BH	Boost C++ Header Files 1.0
bigl	bounded memory linear and 0.0

Apresentação



The screenshot displays the RStudio interface with a C++ source file open. The code defines a function `timesTwo` that takes a `NumericVector` and returns it multiplied by 2. The code is as follows:

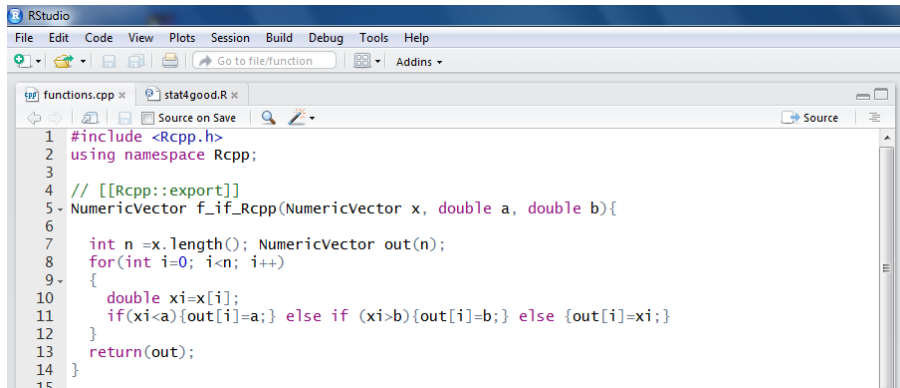
```
1 #include <Rcpp.h>
2 using namespace Rcpp;
3
4 // This is a simple example of exporting a C++ function to R. You can
5 // source this function into an R session using the Rcpp::sourceCpp
6 // function (or via the Source button on the editor toolbar). Learn
7 // more about Rcpp at:
8 //
9 //   http://www.rcpp.org/
10 //   http://adv-r.had.co.nz/Rcpp.html
11 //   http://gallery.rcpp.org/
12 //
13
14 // [[Rcpp::export]]
15 NumericVector timesTwo(NumericVector x) {
16   return x * 2;
17 }
18
19
```

Three red arrows highlight key components of the Rcpp setup:

- Arrow 1 points to `#include <Rcpp.h>` on line 1.
- Arrow 2 points to `using namespace Rcpp;` on line 2.
- Arrow 3 points to `// [[Rcpp::export]]` on line 14.

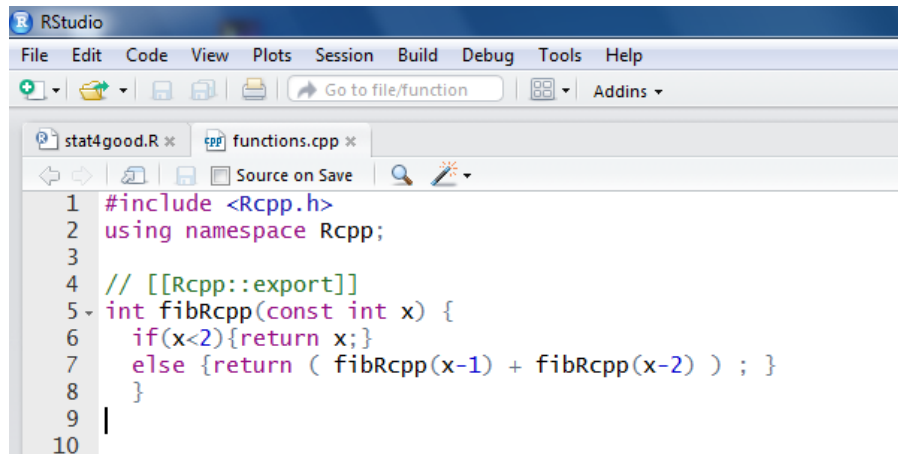
The right-hand pane shows the **Environment** tab, which is currently empty. The **Files** tab shows the project structure, and the **Packages** tab lists installed user libraries, including `akir`, `assx`, `BH`, and `bigl`.

Script do Primeiro Exemplo



```
1 #include <Rcpp.h>
2 using namespace Rcpp;
3
4 // [[Rcpp::export]]
5 NumericVector f_if_Rcpp(NumericVector x, double a, double b){
6
7     int n =x.length(); NumericVector out(n);
8     for(int i=0; i<n; i++)
9     {
10         double xi=x[i];
11         if(xi<a){out[i]=a;} else if (xi>b){out[i]=b;} else {out[i]=xi;}
12     }
13     return(out);
14 }
15
```

Script do segundo Exemplo



```
1 #include <Rcpp.h>
2 using namespace Rcpp;
3
4 // [[Rcpp::export]]
5 int fibRcpp(const int x) {
6     if(x<2){return x;}
7     else {return ( fibRcpp(x-1) + fibRcpp(x-2) ) ; }
8 }
9 |
10
```

Alguns Comandos e Recursos

- ▶ `IntegerVector`: para vetores do tipo inteiro;
- ▶ `NumericVector` ou `NumericMatrix` : para vetores/matrizes tipo numéricos;
- ▶ `LogicalVector`: para vetores do tipo lógico (TRUE/FALSE);
- ▶ `CharacterVector`: para vetores do tipo carácter;
- ▶ `List` : para listas;
- ▶ `Function` para funções;
- ▶ e outros.

Alguns Comandos e Recursos

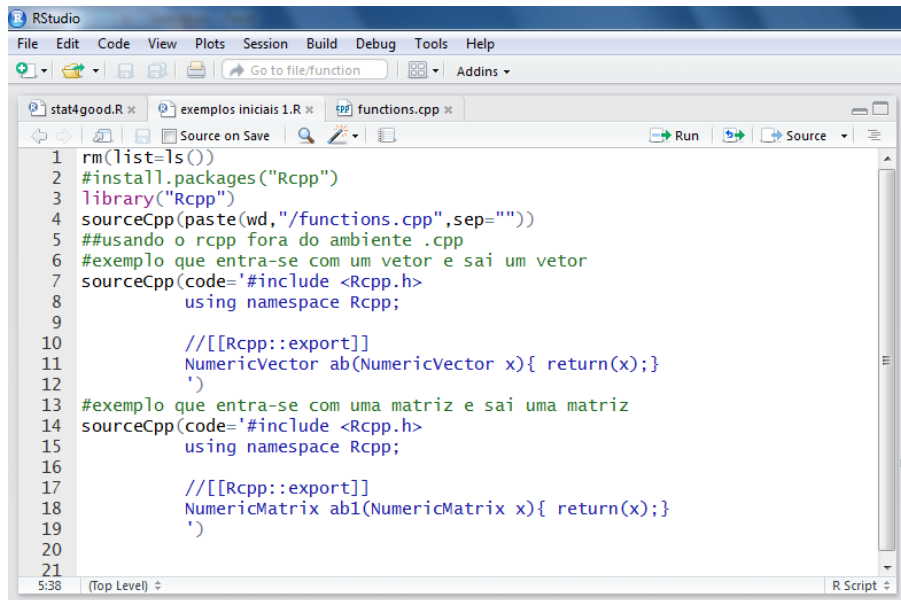
- ▶ operador `()` : acessa elementos via `()` (usado no RcppArmadillo para matrizes);
- ▶ operador `[]` : acessa elementos via `[]` (usado no Rcpp);
- ▶ `length()`: também usa-se `size()`;
- ▶ `fill(t)`: enche o vetor/matrix com `t`;
- ▶ `erase(i)`: remove o elemento da posição `i` no vetor;
- ▶ `insert(i,x)`: insere o elemento `x` na posição `i` do vetor;
- ▶ e outros.

Recursos para carregar funções criadas em C++ no R.

- ▶ função `sourceCpp()`
- ▶ função `cppFunction()`

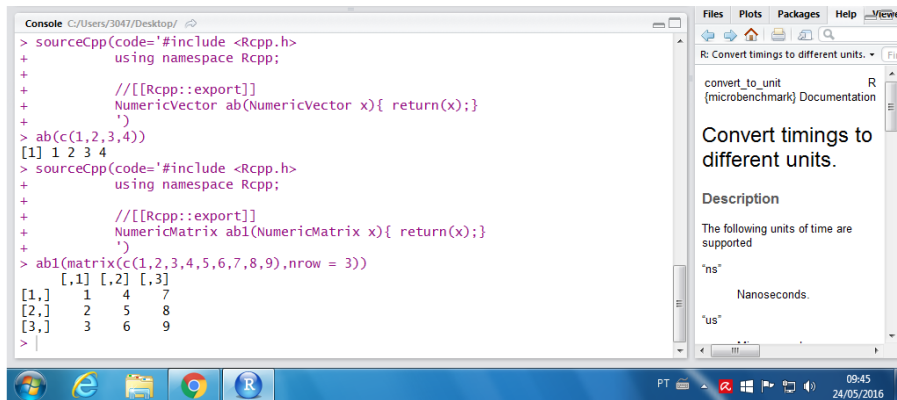
Para começar vamos devagar!!

Primeiras Funções



```
RStudio
File Edit Code View Plots Session Build Debug Tools Help
+ - - - - - Go to file/function - - - - - Addins -
stat4good.R x exemplos iniciais 1.R x functions.cpp x
Source on Save Run Source
1 rm(list=ls())
2 #install.packages("Rcpp")
3 library("Rcpp")
4 sourceCpp(paste(wd, "/functions.cpp", sep=""))
5 ##usando o rcpp fora do ambiente .cpp
6 #exemplo que entra-se com um vetor e sai um vetor
7 sourceCpp(code='#include <Rcpp.h>
8             using namespace Rcpp;
9
10             [[Rcpp::export]]
11             NumericVector ab(NumericVector x){ return(x);}
12             ')
13 #exemplo que entra-se com uma matriz e sai uma matriz
14 sourceCpp(code='#include <Rcpp.h>
15             using namespace Rcpp;
16
17             [[Rcpp::export]]
18             NumericMatrix ab1(NumericMatrix x){ return(x);}
19             ')
20
21
5:38 (Top Level) R Script
```

Primeiras Funções



The screenshot shows an R environment with two windows. The main window is the R console, displaying the execution of Rcpp code. The code defines two functions, `ab` and `ab1`, which use Rcpp to interface with C++ for numeric vector and matrix operations. The output of `ab` is a numeric vector, and the output of `ab1` is a 3x3 numeric matrix.

```
> sourceCpp(code='#include <Rcpp.h>
+           using namespace Rcpp;
+
+           //[[Rcpp::export]]
+           NumericVector ab(NumericVector x){ return(x);}
+           ')
> ab(c(1,2,3,4))
[1] 1 2 3 4
> sourceCpp(code='#include <Rcpp.h>
+           using namespace Rcpp;
+
+           //[[Rcpp::export]]
+           NumericMatrix ab1(NumericMatrix x){ return(x);}
+           ')
> ab1(matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3))
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> |
```

The right-hand window is the R help documentation for the `convert_to_unit` function. It shows the function name, its source package (microbenchmark), and a description of the units it supports: "ns" for Nanoseconds and "us" for microseconds.

Files **Plots** **Packages** **Help** **View**

R: Convert timings to different units. ▾

convert_to_unit R
{microbenchmark} Documentation

Convert timings to different units.

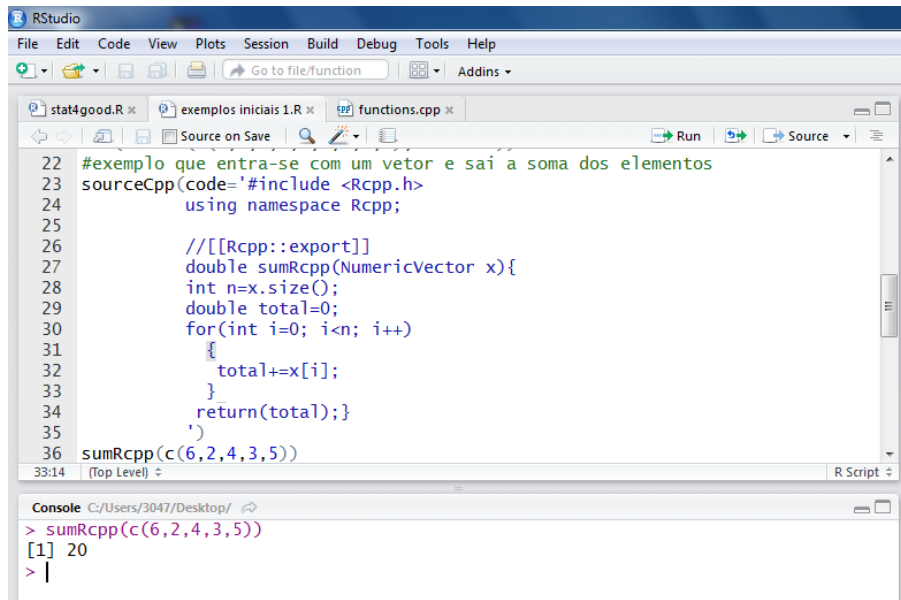
Description

The following units of time are supported

- "ns"
Nanoseconds.
- "us"

PT 09:45
24/05/2016

Primeiras Funções



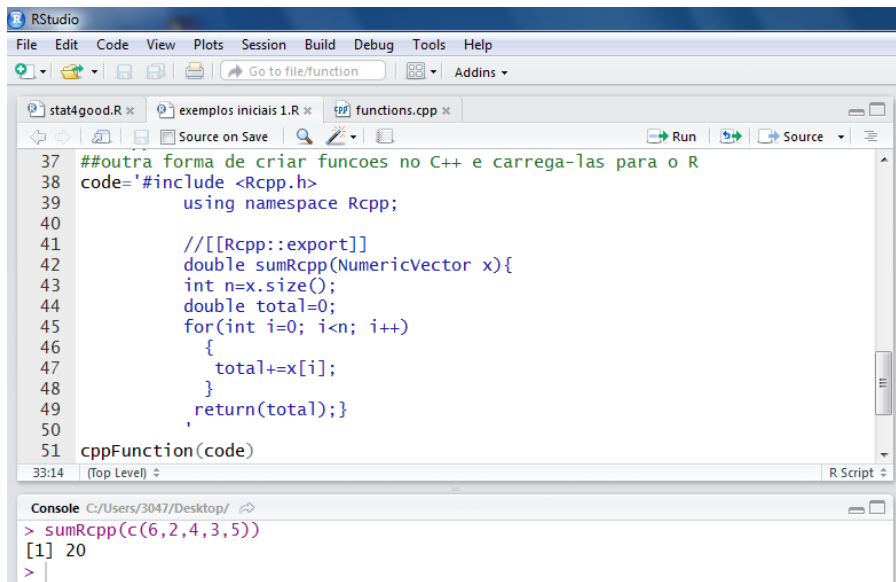
The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar. The editor window has three tabs: stat4good.R, exemplos iniciais 1.R, and functions.cpp. The functions.cpp tab is active, displaying a C++ function named sumRcpp. The function takes a NumericVector x and returns its sum. The console at the bottom shows the function being called with the vector c(6, 2, 4, 3, 5), resulting in the output 20.

```
22 #exemplo que entra-se com um vetor e sai a soma dos elementos
23 sourceCpp(code='#include <Rcpp.h>
24           using namespace Rcpp;
25
26           [[Rcpp::export]]
27           double sumRcpp(NumericVector x){
28             int n=x.size();
29             double total=0;
30             for(int i=0; i<n; i++)
31             {
32               total+=x[i];
33             }
34             return(total);}
35           ')
36 sumRcpp(c(6,2,4,3,5))
```

33:14 (Top Level) R Script

```
> sumRcpp(c(6,2,4,3,5))
[1] 20
> |
```

Primeiras Funções



The screenshot shows the RStudio interface. The source editor displays C++ code for a function that sums the elements of a numeric vector. The code includes the `<Rcpp.h>` header, uses the `Rcpp` namespace, and defines a `sumRcpp` function using `[[Rcpp::export]]`. The function iterates through the vector and returns the total. The console shows the function being called with the vector `c(6, 2, 4, 3, 5)`, resulting in the output `[1] 20`.

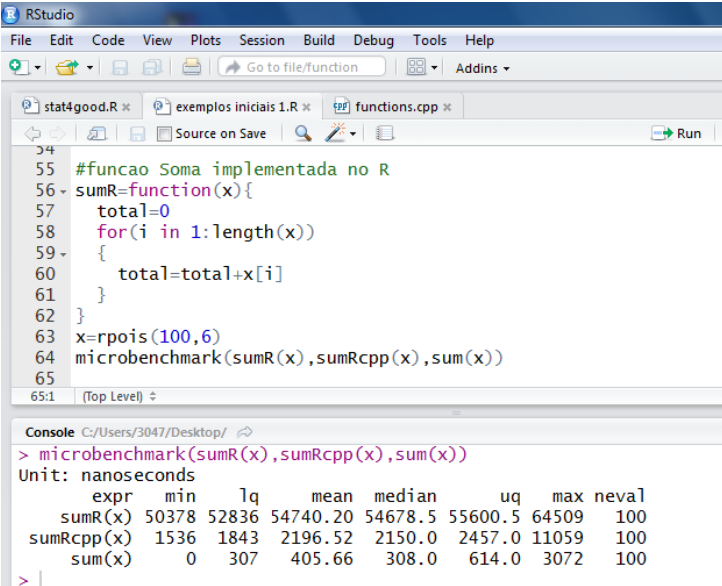
```
37  #outra forma de criar funcoes no C++ e carrega-las para o R
38  code='#include <Rcpp.h>
39      using namespace Rcpp;
40
41      [[Rcpp::export]]
42      double sumRcpp(NumericVector x){
43          int n=x.size();
44          double total=0;
45          for(int i=0; i<n; i++)
46              {
47                  total+=x[i];
48              }
49          return(total);}
50
51  cppFunction(code)
```

33:14 (Top Level) R Script

```
> sumRcpp(c(6,2,4,3,5))
[1] 20
> |
```

Primeiras Funções

O objetivo não é sair do R!!



The screenshot shows the RStudio interface. The editor window displays a C++ function `sumR` that calculates the sum of an array `x` using a `for` loop. The function is then benchmarked against `sumRcpp` and R's built-in `sum` function using `microbenchmark`. The console output shows the benchmark results in nanoseconds.

```
54  
55 #funcao Soma implementada no R  
56 sumR=function(x){  
57   total=0  
58   for(i in 1:length(x))  
59   {  
60     total=total+x[i]  
61   }  
62 }  
63 x=rpois(100,6)  
64 microbenchmark(sumR(x),sumRcpp(x),sum(x))  
65
```

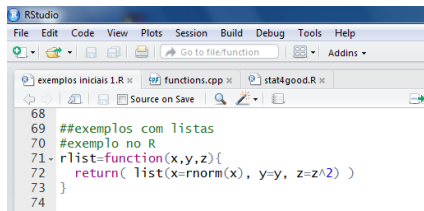
65:1 (Top Level) ⚡

Console C:/Users/3047/Desktop/ ⚡

```
> microbenchmark(sumR(x),sumRcpp(x),sum(x))  
Unit: nanoseconds  
      expr    min      lq      mean    median      uq     max neval  
sumR(x) 50378 52836 54740.20 54678.5 55600.5 64509   100  
sumRcpp(x) 1536  1843  2196.52  2150.0  2457.0 11059   100  
sum(x)      0    307   405.66   308.0   614.0  3072   100
```

> |

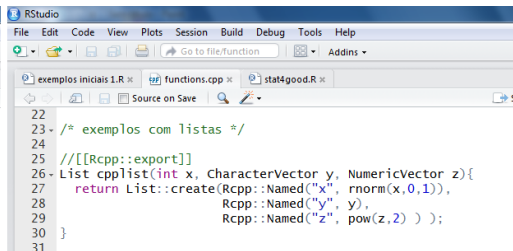
Exemplo com Listas



The screenshot shows the RStudio interface with the 'functions.cpp' file open. The code defines a function 'rlist' that takes three arguments: 'x', 'y', and 'z'. It uses 'rnorm(x)' to generate random numbers for 'x', and 'y' and 'z^2' for 'y' and 'z' respectively. The function returns a list containing these three values.

```
68
69 ##exemplos com listas
70 #exemplo no R
71 rlist=function(x,y,z){
72   return( list(x=rnorm(x), y=y, z=z^2) )
73 }
74
--
```

No Ambiente R



The screenshot shows the RStudio interface with the 'functions.cpp' file open. The code defines a C++ function 'cpplist' that takes three arguments: 'x' (int), 'y' (CharacterVector), and 'z' (NumericVector). It uses 'Rcpp::Named' to pass arguments to 'rnorm' and 'pow' functions. The function returns a List object containing the results of these functions.

```
22
23 /* exemplos com listas */
24
25 //[[Rcpp::export]]
26 List cpplist(int x, CharacterVector y, NumericVector z){
27   return List::create(Rcpp::Named("x", rnorm(x,0,1)),
28                       Rcpp::Named("y", y),
29                       Rcpp::Named("z", pow(z,2) ) );
30 }
31
```

No Ambiente .cpp

Exemplo com Listas

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains R code for creating lists. Line 77 sets the seed to 123. Lines 78-82 create a list `l` with 4 elements: "Nívea", "Larissa", "Juciane", and "Lívea".
- Console:** Shows the execution of the code. It displays the creation of the list `l` and the results of `seq(-4,4)` applied to the list elements.
- Environment:** Shows the global environment with the list `l` and the result of `seq(-4,4)`.
- Plots:** Empty.
- Packages:** Shows the `faithful` package loaded.
- Help:** Shows the documentation for the `faithful` dataset.

```
77 set.seed(123)
78 #lista no R
79 rlist(4,c("Nívea","Larissa","Juciane","Lívea"),seq(-4,4))
80 #lista no Rcpp
81 cpplist(4,c("Nívea","Larissa","Juciane","Lívea"),seq(-4,4))
82
```

```
> #lista no R
> rlist(4,c("Nívea","Larissa","Juciane","Lívea"),seq(-4,4))
$
[1] -0.56047565 -0.23017749  1.55870831  0.07050839

$y
[1] "Nívea"  "Larissa" "Juciane" "Lívea"

$z
[1] 16  9  4  1  0  1  4  9 16

> #lista no Rcpp
> cpplist(4,c("Nívea","Larissa","Juciane","Lívea"),seq(-4,4))
$
[1]  0.1292877  1.7150650  0.4609162 -1.2650612

$y
[1] "Nívea"  "Larissa" "Juciane" "Lívea"

$z
[1] 16  9  4  1  0  1  4  9 16

> |
```

Environment

Name	Type	Le...	Size	Value
cppli...	fun...	1	1...	functio...
f_if...	fun...	1	1...	functio...
fibRc...	fun...	1	84...	functio...
rlist	fun...	1	5...	functio...
wd	cha...	1	12...	"C:/Users...

Plots

Packages

Help

Viewer

R: Old Faithful Geyser Data

Find in Topic

Hardle, W. (1991) *Smoothing Techniques with Implementation in S*. New York: Springer.

Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Faithful geyser. *Applied Statistics* 39, 357–365.

See Also

geyser in package [MASS](#) for the Azzalini–Bowman version.

Examples

```
require(stats); require(graphics)
f.tit <- "faithful data: Eruptions"

ne60 <- round(e60 <- 60 * faithful$
all.equal(e60, ne60)
```


Exemplos de Funções que chamam funções

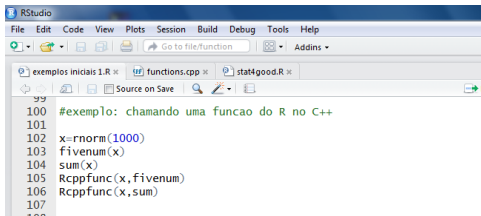
```
RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function Addins
exemplos iniciais 1.R x functions.cpp x stat4good.R x
Source on Save Run
82 ##exemplo de funcoes que chamam funcoes
83 rfun=function(x,f){
84   for(i in 1:100){ out=f(x)}
85   return(out)
86 }
87
88 #exemplos com funcoes no R
89 rfun(c(1,4,9),sqrt)
90 rfun(c(1,4,9),function(x){1/x})
91 #exemplos com funcoes no Rcpp
92 cppfun(c(1,4,9),sqrt)
93 cppfun(c(1,4,9),function(x){1/x})
94
95
96
97
98
99
101:1 (Top Level) ▾
```

```
RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function Addins
exemplos iniciais 1.R x functions.cpp x stat4good.R x
Source on Save Run
32
33 /* exemplos de funcoes que chamam funcoes */
34
35 //[[Rcpp::export]]
36 NumericVector cppfun(NumericVector x, Function f){
37   NumericVector out(x.length());
38   for(int i=0; i<100; i++)
39   {
40     out=f(x);
41   }
42   return out;
43 }
44
45
46
47
48
49
50
35:19 (Top Level) ▾
```

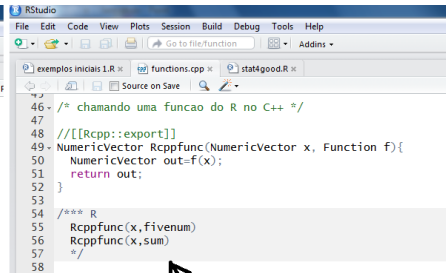
Console ~/erick/ ↻

```
> #exemplos com funcoes no R
> rfun(c(1,4,9),sqrt)
[1] 1 2 3
> rfun(c(1,4,9),function(x){1/x})
[1] 1.0000000 0.2500000 0.1111111
> #exemplos com funcoes no Rcpp
> cppfun(c(1,4,9),sqrt)
[1] 1 2 3
> cppfun(c(1,4,9),function(x){1/x})
[1] 1.0000000 0.2500000 0.1111111
>
```

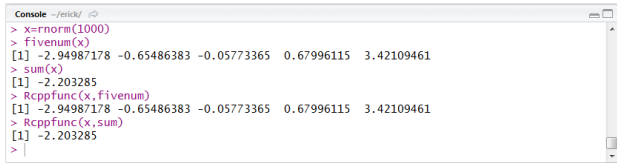
Exemplos de Funções que chamam funções



```
100 #exemplo: chamando uma funcao do R no C++
101
102 x=rnorm(1000)
103 fivenum(x)
104 sum(x)
105 Rcppfunc(x,fivenum)
106 Rcppfunc(x,sum)
107
```



```
46 /* chamando uma funcao do R no C++ */
47
48 //[[Rcpp::export]]
49 NumericVector Rcppfunc(NumericVector x, Function f){
50   NumericVector out=f(x);
51   return out;
52 }
53
54 /** R
55   Rcppfunc(x,fivenum)
56   Rcppfunc(x,sum)
57   */
58
```



```
> x=rnorm(1000)
> fivenum(x)
[1] -2.94987178 -0.65486383 -0.05773365 0.67996115 3.42109461
> sum(x)
[1] -2.203285
> Rcppfunc(x,fivenum)
[1] -2.94987178 -0.65486383 -0.05773365 0.67996115 3.42109461
> Rcppfunc(x,sum)
[1] -2.203285
>
```

Ambiente R dentro do arquivo .cpp !!

RcppArmadillo

Estudos envolvendo Matrizes tem uma extrema importância para nós.

O Armadillo [Sanderson (2010)] é uma moderna biblioteca do C++ com foco em álgebra linear e operações relacionadas.

- ▶ Visa atingir um bom equilíbrio entre velocidade e facilidade de uso.
- ▶ A sintaxe ser semelhante ao MATLAB.
- ▶ Pode-se fazer uso de subconjuntos de funções trigonométricas, números complexos, decomposições de matrizes, funções estatísticas, etc.

RcppArmadillo

O pacote **RcppArmadillo** [Francois et al (2012);. Eddelbuettel e Sanderson (2013)] integra ao R utilizando recursos fornecidos pelo pacote Rcpp.

As principais classes de interesse são `arma::mat` e `arma::vec` e os elementos são `double`.

O que se tinha.

```
1  
2 #include <Rcpp.h>  
3
```

O que temos agora

```
3 #include <RcppArmadillo.h>  
4  
5 // [[Rcpp::depends("RcppArmadillo")]]  
6  
7
```

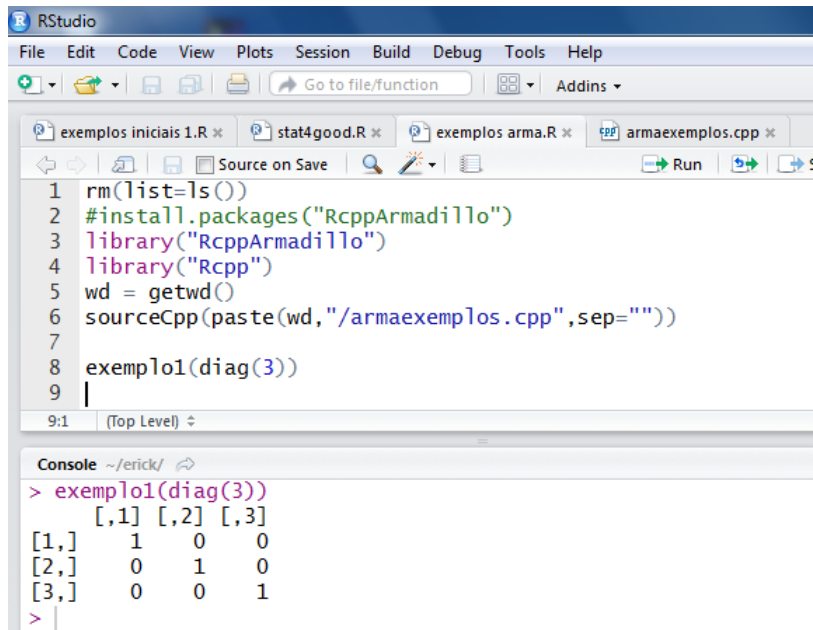
Para melhor entendimento vamos a alguns exemplos.

Exemplos Básicos

Entra matriz sai matriz.

```
1  #include<RcppArmadillo.h>
2  // [[Rcpp::depends("RcppArmadillo")]]
3
4  using namespace Rcpp;
5
6  //[[Rcpp::export()]]
7  arma::mat exemplo1(arma::mat x){
8      return(x);
9  }
10 |
```

Exemplos Básicos



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, and Help. Below the menu is a toolbar with icons for file operations and a search bar labeled 'Go to file/function'. The file explorer shows four open files: 'exemplos iniciais 1.R', 'stat4good.R', 'exemplos arma.R', and 'armaexemplos.cpp'. The editor window displays the following R code:

```
1 rm(list=ls())
2 #install.packages("RcppArmadillo")
3 library("RcppArmadillo")
4 library("Rcpp")
5 wd = getwd()
6 sourceCpp(paste(wd, "/armaexemplos.cpp", sep=""))
7
8 exemplo1(diag(3))
9 |
```

The status bar at the bottom of the editor indicates '9:1 (Top Level)'. Below the editor is the console window, which shows the output of the executed code:

```
> exemplo1(diag(3))
      [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> |
```

The console window title is 'Console ~/erick/'. In the bottom right corner of the RStudio window, there are icons for file explorer, search, and other utility functions.

Exemplos Básicos

Entra vetor.

```
10  
11 //[[Rcpp::export()]]  
12 arma::vec exemplo2(arma::vec x){  
13     return(x);  
14 }  
15
```

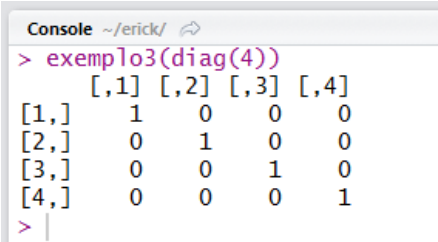
Sai matriz.

```
Console ~/erick/ ↵  
> exemplo2(c(2,4,6,8))  
      [,1]  
[1,]    2  
[2,]    4  
[3,]    6  
[4,]    8
```

Exemplos Básicos

Passando de `Rcpp::NumericMatrix` para `arma::mat`

```
16 //[[Rcpp::export()]]
17 arma::mat exemplo3(NumericMatrix x){
18     arma::mat y=as<arma::mat>(x);
19     return (y);
20 }
21
```



Console ~/erick/ ↗

```
> exemplo3(diag(4))
      [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
> |
```


Exemplos Básicos

Passando de `arma::mat` para `Rcpp::NumericMatrix`

```
22  //[[Rcpp::export()]]  
23  NumericMatrix exemplo4(arma::mat x){  
24    NumericMatrix y=wrap(x);  
25    return (y);  
26  }  
27
```

Console ~/erick/ ↻

```
> exemplo4(matrix(runif(12),3,4))  
      [,1]      [,2]      [,3]      [,4]  
[1,] 0.04503251 0.4826513 0.4484689 0.2144506  
[2,] 0.44901080 0.1361147 0.4828098 0.8853638  
[3,] 0.15733940 0.8979925 0.4416017 0.1232128  
> |
```

Exemplos Básicos

print e dimensão **linhas** e **colunas**

```
27  
28 //[[Rcpp::export()]]  
29 arma::mat exemplo5(arma::mat x){  
30     int r=x.n_rows;  
31     int c=x.n_cols;  
32     printf(" linhas:%d \n colunas:%d \n",r,c);  
33     return(x);  
34 }  
35
```

Console ~/erick/ ↗

```
> exemplo5(matrix(runif(8),4,2))  
linhas:4  
colunas:2  
      [,1]      [,2]  
[1,] 0.9237826 0.5335634  
[2,] 0.1045257 0.5427297  
[3,] 0.5188606 0.9127224  
[4,] 0.8129962 0.1377768  
> |
```

Exemplos Básicos

Geração de números aleatórios e lista

```
36
37 //[[Rcpp::export()]]
38 List exemplo6(int n, int m, double v){
39   arma::mat x(n,m); x.fill(v);
40   arma::colvec y( rnorm(n,0,1 ));
41   arma::rowvec z(rgamma(m,2,5));
42   List retorno;
43   retorno["X"]=x; retorno["Y"]=y; retorno["Z"]=z;
44   return retorno;
45 }
```

43:50 exemplo6(int n, int m, double v): List ↕

Console ~/erick/ ↗

> exemplo6(3,4,4/5)

\$X

	[,1]	[,2]	[,3]	[,4]
[1,]	0.8	0.8	0.8	0.8
[2,]	0.8	0.8	0.8	0.8
[3,]	0.8	0.8	0.8	0.8

\$Y

	[,1]
[1,]	0.4565580
[2,]	0.6843294
[3,]	-0.1900653

\$Z

	[,1]	[,2]	[,3]	[,4]
[1,]	9.310082	4.678181	6.418033	8.221046

Exemplos Básicos

localizar elementos ou subconjuntos na matriz

```
47 //[[Rcpp::export()]]
48 double exemplo7(arma::mat x, int i, int j){
49     return(x(i,j));
50 }
51
52 //[[Rcpp::export()]]
53 arma::mat exemplo8(arma::mat x, int i){
54     return(x.row(i));
55 }
56
57 //[[Rcpp::export()]]
58 arma::mat exemplo9(arma::mat x, int j){
59     return(x.col(j));
60 }
```

```
Console ~/erick/ ↵
> sourceCpp(paste(wd, "/armaexemplos.cpp", sep=""))
> exemplo7(Z,1,2)
[1] 0.1095678
> exemplo8(Z,1)
      [,1]      [,2]      [,3]
[1,] -0.1674742 -0.667986 0.1095678
> exemplo9(Z,2)
      [,1]
[1,] -0.01605317
[2,] 0.10956780
> |
```

Exemplos Básicos

Algumas operações como: soma, diferença, multiplicação elemento-elemento e exponenciação.

```
62 //[[Rcpp::export()]]
63 arma::mat exemplo10(arma::mat x){
64     return (x+x);
65 }
66
67 //[[Rcpp::export()]]
68 arma::mat exemplo11(arma::mat x){
69     return (x-x);
70 }
71
72 //[[Rcpp::export()]]
73 arma::mat exemplo12(arma::mat x){
74     return (x%x);
75 }
76
77 //[[Rcpp::export()]]
78 arma::mat exemplo13(arma::mat x){
79     return (exp(x));
80 }
```

Exemplos Básicos

Console ~/erick/ ↗

> exemplo10(Z) #realiza a soma Z+Z

	[,1]	[,2]	[,3]
[1,]	0.5522737	-0.6137135	-0.03210635
[2,]	-0.3349484	-1.3359721	0.21913561

> exemplo11(Z) #realiza a diferenca Z-Z

	[,1]	[,2]	[,3]
[1,]	0	0	0
[2,]	0	0	0

> exemplo12(Z) #multiplica elemento por elemento de Z

	[,1]	[,2]	[,3]
[1,]	0.07625155	0.09416107	0.0002577044
[2,]	0.02804760	0.44620533	0.0120051036

> exemplo13(Z) #calcula exponencial de Z

	[,1]	[,2]	[,3]
[1,]	1.3180282	0.7357560	0.984075
[2,]	0.8457984	0.5127402	1.115796

> |

Exemplos Básicos

Matriz transposta, multiplicação e Matriz inversa.

```
82  //[[Rcpp::export()]]
83  arma::mat exemplo14(arma::mat x){
84      return (x.t());
85  }
86
87  //[[Rcpp::export()]]
88  arma::mat exemplo15(arma::mat x){
89      return (x.t()*x);
90  }
91
92  //[[Rcpp::export()]]
93  arma::mat exemplo16(arma::mat x){
94      return ( (x.t()*x).i() );
95  }
```

Exemplos Básicos

Console ~/erick/ ↗

```
> Z=matrix(rnorm(6),2,3)
> exemplo14(Z) #calcula a transposta de Z
      [,1]      [,2]
[1,] 0.47480988 -1.1796706
[2,] 0.17695023 -1.0731056
[3,] 0.09337838  0.2924796
> exemplo15(Z) #multiplicacao
      [,1]      [,2]      [,3]
[1,]  1.6170672  1.3499288 -0.30069259
[2,]  1.3499288  1.1828669 -0.29733814
[3,] -0.3006926 -0.2973381  0.09426383
> exemplo16(Z) #matriz inversa da multiplicacao no exemplo 15
      [,1]      [,2]      [,3]
[1,]  2.829618e+15 -4.637126e+15 -5.600762e+15
[2,] -4.637126e+15  7.599237e+15  9.178426e+15
[3,] -5.600762e+15  9.178426e+15  1.108578e+16
> |
```


Exemplos Básicos

Geração de matriz aleatória e mensagens de erro.

```
97 //[[Rcpp::export()]]  
98 arma::mat exemplo17(int i){  
99     arma::mat A(2,3);  
100     arma::mat B(i,1);  
101     A.randu(); A.print();  
102     B.randu(); B.print();  
103     return (A*B);  
104 }
```

Exemplos Básicos

Multiplicação é válida para $i = 3$.

```
Console ~/erick/ ↗  
> exemplo17(3) # nao haverá mensagem de erro (dimensao ok!)  
0.7942 0.9417 0.9183  
0.2660 0.8246 0.0733  
0.8781  
0.3248  
0.3184  
[ ,1]  
[1,] 1.2956149  
[2,] 0.5247511  
> exemplo17(2) # aqui haverá mensagens de erro (dimensao nao ok)  
0.5506 0.6502 0.6176  
0.8066 0.7120 0.7330  
0.7345  
0.3708  
  
error: matrix multiplication: incompatible matrix dimensions: 2x3 and 2x1  
Error: matrix multiplication: incompatible matrix dimensions: 2x3 and 2x1  
> |
```

Acelerando o Passo...

Exemplos Intermediários!

Exemplos de Inferência Bayesiana

```
#include<RcppArmadillo.h>
// [[Rcpp::depends("RcppArmadillo")]]

using namespace Rcpp;

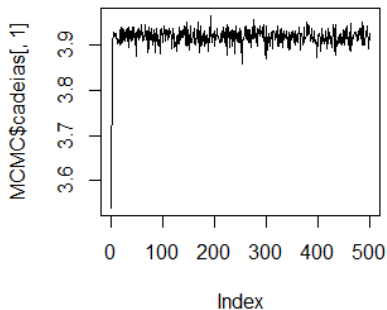
//[[Rcpp::export]]
List gibbsNG(arma::colvec x, int N){
  int n= x.size();
  arma::mat theta(N,2);
  double mi=0, phi=1;      /* chutes iniciais */
  double a=1, b=1;         /* valores para os hiperparâmetros da a
  double m=0, v=1;
  for(int i=0; i<N; i++)
  {
    double A = a + 0.5*n ;
    double B=0;
    for(int j=0; j<n; j++){ B += pow( (x(j,0)-mi) ,2) ; }
    B = b + 0.5*B;
    phi=R::rgamma(A,1/B); /* atualiza phi com base em A e B */
    double V = 1/( (n*phi) + (1/v) );
    double M = V*( (m/v) + sum(x)*phi );
    mi = R::rnorm(M,V);
    theta(i,0)=mi;
    theta(i,1)=phi;
  }
  List saida;
  saida["cadeias"]=theta;
  return saida;
}
```

Exemplos de Inferência Bayesiana

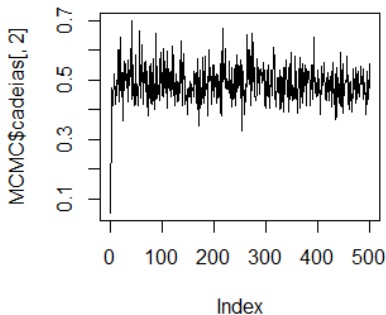
```
1 rm(list=ls())
2 #install.packages("RcppArmadillo")
3 library("RcppArmadillo")
4 library("Rcpp")
5 wd = getwd()
6 sourceCpp(paste(wd, "/armaBayesiana.cpp", sep=""))
7
8 #--exemplo gibbs normal-gama
9 n=150
10 mi=4
11 sigma2=2 #phi=1/sigma2 precisão
12 x=rnorm(n, mean = mi, sd=sqrt(sigma2))
13
14 MCMC=gibbsNG(x=x, N=500)
15 plot(MCMC$cadeias[,1], type="l")
16 plot(MCMC$cadeias[,2], type="l")
17 ##-----
```

Exemplos de Inferência Bayesiana

cadeia de μ .



cadeia de σ^2 .

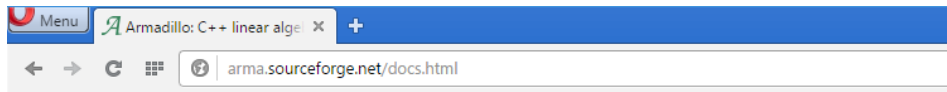


Exemplos de Inferência Bayesiana

```
armaBayesiana.cpp  exemplos.arma.R  armaexemplos.cpp  exemplosIntermediarios.R
Source on Save  Run  Source

18 #--exemplo gibbs regressão linear
19 rm(list=ls())
20 #install.packages("RcppArmadillo")
21 library("RcppArmadillo")
22 library("Rcpp")
23 wd = getwd()
24 sourceCpp(paste(wd, "/armaBayesiana.cpp", sep=""))
25 n=50
26 x1=rnorm(n,0,sqrt(0.4))
27 x2=rbinom(n,1,0.5)
28 B0=1
29 B1=0.5
30 B2=-0.5
31 sigma2=1.5
32 erro=rnorm(n,0,sqrt(sigma2))
33 y=B0+B1*x1+B2*x2+erro;
34 y=as.matrix(y)
35 dados=data.frame(rep(1,n),x1,x2)
36 X=as.matrix(dados)
37
38 MCMC2=GibbsRegress(y=y,x=X,N=2000,burnin = 1000)
39 plot(MCMC2$betas[,1],type="l")
40 abline(1,0,col="red",lwd=2)
41 plot(MCMC2$betas[,2],type="l")
42 abline(0.5,0,col="red",lwd=2)
43 plot(MCMC2$betas[,3],type="l")
44 abline(-0.5,0,col="red",lwd=2)
45 plot(MCMC2$sigma2[,1],type="l")
46 abline(1.5,0,col="red",lwd=2)
47
```

Exemplos de Inferência Bayesiana



Armadillo C++ linear algebra library

About

Support

Questions

Documentation

Speed

Contact

Download

API Documentation for Armadillo 7.100

Preamble

- For converting Matlab/Octave programs, see the [syntax conversion table](#)
- First time users: please see the short [example program](#)
- If you discover any bugs or regressions, please [report them](#)
- History of [API additions](#)

Obrigado!

Referências I

Eddulbuettel, D., François, R. (2011). Rcpp:Seamless R and C++ Integration, Journal of Statistical Software. URL <http://www.jstatsoft.org/v40/>

François, R., Eddelbuettel D., Bates D. (2012) RcppArmadillo: Rcpp integration for Armadillo templated linear algebra library. URL <http://CRAN.R-project.org/package=RcppArmadillo>, R package version 0.3.4.4

Sanderson, C. (2010). Armadillo: An open source C++ algebra library for fast prototyping and computationally intensive experiments. Tech. rep., NICTA, URL <http://arma.sf.net>

Sklyar, O., Murdoch, D., Smith, M., Eddulbuettel, D., François, R. (2012). Inline C, C++, Fortran function calls for R. URL <http://CRAN.R-project.org/package=inline>, R package version 0.3.10