

Integrando o R com C++: Uma breve introdução ao Rcpp

Autores: Erick Amorim e Rumenick Pereira da Silva¹

¹rumenickbf@hotmail.com

Seminário Internacional de Estatística com R
Universidade Federal Fluminense

21 de Maio de 2019

SUMÁRIO

Introdução

- Breve motivação
- Alguns Tipos de Objetos
- Comandos Importantes
- Primeiras Funções
- Exemplos Envolvendo Listas
- Exemplos Envolvendo Composição de Funções

RcppArmadillo

- Tipos de objetos
- Exemplos Básicos

Introdução

O que é preciso fazer para que os nossos códigos rodem mais rápido?

Introdução

Os principais pontos que vem em nossa mente são:

- ▶ Reduzir os laços (**for**),
- ▶ diminuir o tamanho da estrutura de dados,
- ▶ utilizar programação paralela,
- ▶ entre outros.

Com o R podemos aumentar a velocidade de nossos códigos sem alterar quase nada a sua estrutura. Isso se torna possível graças ao pacote **Rcpp**.

Introdução

Os principais pontos que vem em nossa mente são:

- ▶ Reduzir os laços (**for**),
- ▶ diminuir o tamanho da estrutura de dados,
- ▶ utilizar programação paralela,
- ▶ entre outros.

Com o R podemos aumentar a velocidade de nossos códigos sem alterar quase nada a sua estrutura. Isso se torna possível graças ao pacote **Rcpp**.

Introdução

- ▶ Em 2002 o pacote **RQuantLib** foi iniciado por Eddelbuettel.
- ▶ Em 2005 o **Rcpp** apareceu pela 1^a vez, com a contribuição de Dominick Samperi ao pacote **RQuantLib**
- ▶ No início 2006 o **Rcpp** se tornou um pacote no CRAN.
- ▶ Atualmente o **Rcpp** continua sob desenvolvimento ativo e extensões estão sendo adicionadas.

Introdução

O principal objetivo deste pacote é fazer a integração do R com C++. A construção de funções baseadas em C++ é o fruto dessa integração.

O **Rcpp** permite que o usuário retorne resultados obtidos na linguagem C++ para o R mostrando seu potencial em termos de velocidade (tempo) e facilidade de uso.

O programador pode manter sua estrutura de dados no R sem ter que se preocupar com a transferência para o C++.

Introdução

O principal objetivo deste pacote é fazer a integração do R com C++. A construção de funções baseadas em C++ é o fruto dessa integração.

O **Rcpp** permite que o usuário retorne resultados obtidos na linguagem C++ para o R mostrando seu potencial em termos de velocidade (tempo) e facilidade de uso.

O programador pode manter sua estrutura de dados no R sem ter que se preocupar com a transferência para o C++.

Introdução

O principal objetivo deste pacote é fazer a integração do R com C++. A construção de funções baseadas em C++ é o fruto dessa integração.

O **Rcpp** permite que o usuário retorne resultados obtidos na linguagem C++ para o R mostrando seu potencial em termos de velocidade (tempo) e facilidade de uso.

O programador pode manter sua estrutura de dados no R sem ter que se preocupar com a transferência para o C++.

Introdução

Neste Minicurso apresentaremos alguns conceitos básicos sobre o pacote **Rcpp**, que funciona como uma extensão do R com funções do C++.

Um pouco do conhecimento de C++ é muito útil embora não seja estrito.

Ambiente Rcpp

O que simplesmente acontece é que vamos definir uma simples função em C++ usando níveis de classes (tipos de objetos) como **NumericVertor**.

Em seguida o **Rcpp**, ocultando alguns detalhes, irá compilar, ligar e chamar essa função para ser avaliada no R.

O pacote basicamente mapeia os objetos do R para objetos do C++ na maioria das vezes de forma automática.

Uma preocupação de interesse é que algumas funções criadas no R não são tão “leves”. Isso torna o recurso de chamar essas funções pouco atrativo. Veja por exemplo o tempo de processamento de algumas funções simples:

Ambiente Rcpp

O que simplesmente acontece é que vamos definir uma simples função em C++ usando níveis de classes (tipos de objetos) como `NumericVector`.

Em seguida o **Rcpp**, ocultando alguns detalhes, irá compilar, ligar e chamar essa função para ser avaliada no R.

O pacote basicamente mapeia os objetos do R para objetos do C++ na maioria das vezes de forma automática.

Uma preocupação de interesse é que algumas funções criadas no R não são tão “leves”. Isso torna o recurso de chamar essas funções pouco atrativo. Veja por exemplo o tempo de processamento de algumas funções simples:

Ambiente Rcpp

O que simplesmente acontece é que vamos definir uma simples função em C++ usando níveis de classes (tipos de objetos) como `NumericVector`.

Em seguida o **Rcpp**, ocultando alguns detalhes, irá compilar, ligar e chamar essa função para ser avaliada no R.

O pacote basicamente mapeia os objetos do R para objetos do C++ na maioria das vezes de forma automática.

Uma preocupação de interesse é que algumas funções criadas no R não são tão “leves”. Isso torna o recurso de chamar essas funções pouco atrativo. Veja por exemplo o tempo de processamento de algumas funções simples:

Ambiente Rcpp

O que simplesmente acontece é que vamos definir uma simples função em C++ usando níveis de classes (tipos de objetos) como `NumericVector`.

Em seguida o **Rcpp**, ocultando alguns detalhes, irá compilar, ligar e chamar essa função para ser avaliada no R.

O pacote basicamente mapeia os objetos do R para objetos do C++ na maioria das vezes de forma automática.

Uma preocupação de interesse é que algumas funções criadas no R não são tão “leves”. Isso torna o recurso de chamar essas funções pouco atrativo. Veja por exemplo o tempo de processamento de algumas funções simples:

Primeiro Exemplo

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays the file `functions.cpp` containing R code for microbenchmarking. The code includes functions for `f_if`, `f_pm`, `f_colchett`, and `f_if_Rcpp`.
- Console:** Shows the command `microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchett(x,-1,1),f_if_Rcpp(x,-1,1),times=1000)` being run.
- Environment View:** Shows the global environment with variables `wd` and `x`. It also lists several functions defined in the current session.
- Functions View:** Lists the functions defined in the current session, including `cppfun`, `cpplist`, `f_colchett`, `f_if`, `f_if_Rcpp`, and `f_nm`.
- Session View:** Shows the user library and system library with their respective versions.

```
11 library("microbenchmark")
12 #teste da funcao microbenchmark
13 x=rnif(100)
14 microbenchmark(sqrt(x),x^0.5,x^(1/2),(x)^(1/2))
15
16 #primeiro exemplo
17 #funcao1
18 f_if=function(x,a,b){
19   ifelse(x<=a,a,ifelse(x>=b,b,x))
20 }
21 #funcao2
22 f_pm=function(x,a,b){
23   pmax(pmin(x,b),a)
24 }
25 #funcao3
26 f_colchett=function(x,a,b){
27   x[x<=a]=a
28   x[x>=a]=b
29   x
30 }
31 x=rnif(1000,-1,1)
32 microbenchmark(f_if(x,-1,1),f_pm(x,-1,1),f_colchett(x,-1,1),f_if_Rcpp(x,-1,1),times=1000)
33
34 <--
```

Primeiro Exemplo (Continuação)

The screenshot shows the RStudio interface with the following components:

- File Explorer:** Shows the file `functions.cpp` is open.
- Code Editor:** Displays the C++ code for microbenchmarking R functions. The code defines several functions using Rcpp:

```
10
11 library("microbenchmark")
12 #teste da função microbenchmark
13 x=runif(100)
14 microbenchmark(sqrt(x),x^0.5,x^(1/2),(x)^(1/2))
15
16 #primeiro exemplo
17 #funcão
18- f_if<-function(x,a,b){
19   ifelse(x<=a,a,ifelse(x>=b,b,x))
20 }
21 #funcão
22- f_pm<-function(x,a,b){
23   pmax(pmin(x,b),a)
24 }
25 #funcão
26- f_colchet<-function(x,a,b){
27   x[x<=a]=a
28   x[x>=a]=b
29   x
30 }
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
34 <--
```

- Console:** Shows the results of the microbenchmark command:

```
> 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) 142.263 157.257 199.43212 159.816 161.6460 2490.864 1000
f_pm(x, -1, 1) 44.983 47.908 55.18296 48.641 50.4685 2411.870 1000
f_colchet(x, -1, 1) 27.795 29.989 39.29023 30.720 31.4520 2156.237 1000
f_if_Rcpp(x, -1, 1) 6.949 9.144 12.42980 9.875 10.6060 2057.129 1000
>
```

- Environment:** Shows the global environment with variables `wd` and `x`.
- Functions:** Shows the definitions of the functions defined in the code.
- Packages:** Shows the installed packages: `microbench`, `Rcpp`, and `RcppArmadillo`.
- System Library:** Shows the available system libraries.

Primeiro Exemplo (Continuação)

The screenshot shows the RStudio interface with the following components:

- File Explorer:** Shows files like "MiniCursoSER.R" and "functions.cpp".
- Code Editor:** Displays the R code for the "functions.cpp" file, which includes various Rcpp functions for numerical operations.
- Console:** Shows the output of the microbenchmark command. The results table is highlighted with a red box.
- Environment:** Shows the global environment with variables "wd" and "x" defined.
- Functions:** Shows a list of available functions.
- Packages:** Shows the installed packages: microbenchmark, Rcpp, and RcppArmadillo.
- System Library:** Shows the system library with packages like boot, class, cluster, codetools, compiler, datasets, and foreign.

Console Output (highlighted area):

```
> 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) 142.263 157.257 199.43212 159.816 161.6460 2490.864 1000
 f_pm(x, -1, 1) 44.983 47.908 55.18296 48.641 50.4685 2411.870 1000
 f_colchet(x, -1, 1) 27.795 29.989 39.29023 30.720 31.4520 2156.237 1000
 f_if_Rcpp(x, -1, 1)  6.949  9.144 12.42980  9.875 10.6060 2057.129 1000
>
```

Segundo Exemplo

$$F_0 = 0; \quad F_1 = 1; \quad F_n = F_{n-1} + F_{n-2}$$

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** MiniCursoSER.R and functions.cpp. The functions.cpp file contains the following Rcpp code:

```
35 #segundo exemplo
36 ## serie fibonacci no R
37 f_if_Rcpp<-function(n){
38   if(n==0){ return (0)}
39   if(n==1){ return (1)}
40   return(f_if_Rcpp(n-1) + f_if_Rcpp(n-2))
41 }
42
43 microbenchmark(fibR(27),fibRcpp(27),times=10,unit = "ms")
44
45 <--
```
- Console:** Shows the execution of the microbenchmark command and its output. The output table compares the performance of fibR(27) and fibRcpp(27).

	expr	min	lq	mean	median	uq	max	neval
fibR(27)	401.72814	406.096196	411.373746	408.620704	411.126927	443.168030	10	
fibRcpp(27)	3.38832	3.392343	3.471081	3.410628	3.431474	3.927014	10	
- Environment Tab:** Shows the Global Environment with functions f_if_Rcpp, f_pm, fibR, and fibRcpp.
- Files Tab:** Shows the packages installed: microbenchmark, Rcpp, RcppArmadillo, and boot.
- Plots Tab:** Not visible in the screenshot.
- Packages Tab:** Not visible in the screenshot.
- Help Tab:** Not visible in the screenshot.
- Viewer Tab:** Not visible in the screenshot.

Segundo Exemplo

$$F_0 = 0; \quad F_1 = 1; \quad F_n = F_{n-1} + F_{n-2}$$

The screenshot shows an RStudio interface with the following details:

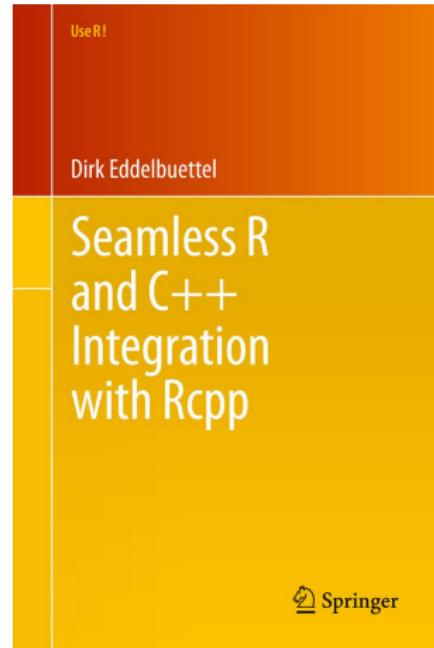
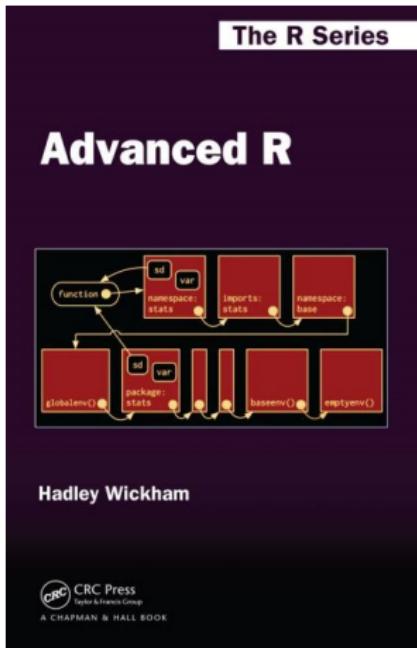
- File:** MiniCursoSER.R
- Code Editor:** functions.cpp
- Code Content:**

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

```
> microbenchmark(fibR(27),fibRcpp(27),times=10,unit = "ms")
unit: milliseconds
      expr      min       lq     mean      median       uq      max neval
    fibR(27) 401.72814 406.096196 411.373746 408.620704 411.126927 443.168030    10
  fibRcpp(27)  3.38832  3.392343  3.471081  3.410628  3.431474  3.927014    10
> |
```

A red box highlights the mean column of the microbenchmark output table.
- Environment View:** Shows the Global Environment, Files, Plots, Packages, Help, and Viewer tabs. The Global Environment pane lists functions: f_if_Rcpp, f_pm, fibR, and fibRcpp.
- Session View:** Shows the User Library, System Library, and other session details.

Livros Para Consultas



Como usamos ??

R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/.RData]

> |

Environment History

Global Environment

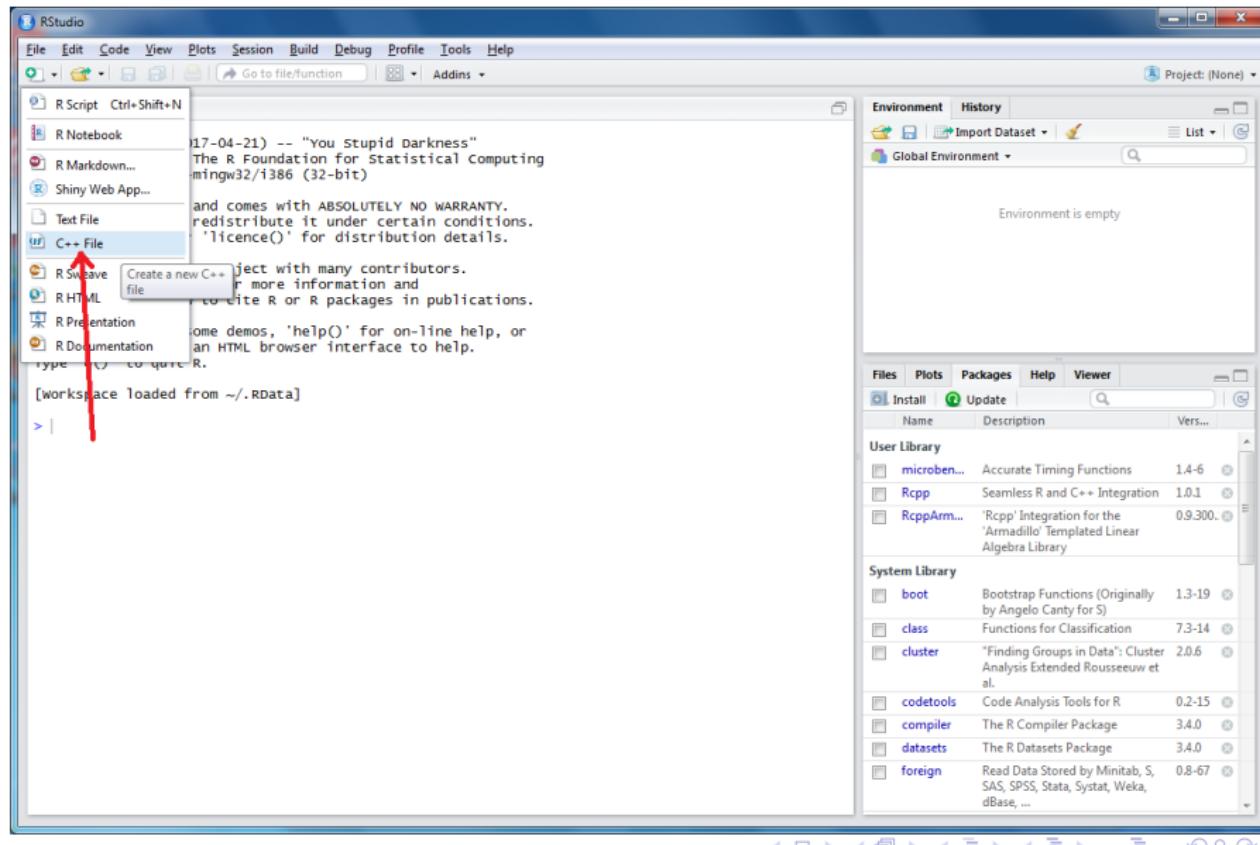
Environment is empty

Files Plots Packages Help Viewer

Install Update

Name	Description	Vers...
microben...	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration	1.0.1
RcppArm...	'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library	0.9.300..
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-19
class	Functions for Classification	7.3-14
cluster	"Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.	2.0.6
codetools	Code Analysis Tools for R	0.2-15
compiler	The R Compiler Package	3.4.0
datasets	The R Datasets Package	3.4.0
foreign	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...	0.8-67

Primeiros passos



Apresentação

The screenshot shows the RStudio IDE interface. The left pane displays a code editor for an R script named 'Untitled1'. The code is a simple example of exporting a C++ function to R using Rcpp. The right pane shows the R console output, which includes the R startup message about being free software, information about contributors, and help instructions. The bottom right pane shows the Global Environment and the Packages tab of the Help menu.

```
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
20 // You can include R code blocks in C++ files processed with sourceCpp
21 // (useful for testing and development). The R code will be automatically
22 // run after the compilation.
23 //
24
1:1 (Top Level) : C/C++
```

Console ~/ ↵
R IS FREE SOFTWARE AND COMES WITH ABSOLUTELY NO WARRANTY.
YOU ARE WELCOME TO REDISTRIBUTE IT UNDER CERTAIN CONDITIONS.
TYPE 'LICENSE()' OR 'LICENCE()' FOR DISTRIBUTION DETAILS.

R IS A COLLABORATIVE PROJECT WITH MANY CONTRIBUTORS.
TYPE 'CONTRIBUTORS()' FOR MORE INFORMATION AND
'CITATION()' ON HOW TO CITE R OR R PACKAGES IN PUBLICATIONS.

TYPE 'DEMO()' FOR SOME DEMOS, 'HELP()' FOR ON-LINE HELP, OR
'HELP.START()' FOR AN HTML BROWSER INTERFACE TO HELP.
TYPE 'Q()' TO QUIT R.

[WORKSPACE LOADED FROM ~/.RDATA]

>

Environment History
Import Dataset List
Global Environment
Environment is empty

Files Plots Packages Help Viewer
Install Update
Name Description Vers...
User Library
microben... Accurate Timing Functions 1.4-6
Rcpp Seamless R and C++ Integration 1.0.1
RcppArm... 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library 0.9.300...
System Library
boot Bootstrap Functions (Originally by Angelo Canty for S) 1.3-19
class Functions for Classification 7.3-14
cluster "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. 2.0.6
codetools Code Analysis Tools for R 0.2-15
compiler The R Compiler Package 3.4.0
datasets The R Datasets Package 3.4.0
foreign Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...

Apresentação

The screenshot shows the RStudio interface with the following details:

- Code Editor:** The main window displays an Rcpp script named "Untitled1". The code includes imports for Rcpp, defines a function `timestwo` that returns its input multiplied by 2, and provides documentation for sourceCpp usage.
- Environment Pane:** The right-hand pane shows the Global Environment, which is currently empty.
- Tools Pane:** The bottom-right pane lists installed packages: microbenchmark, Rcpp, and RcppArmadillo.
- Console:** The bottom-left pane shows the R startup message and the workspace loading confirmation.

```
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
20 // You can include R code blocks in C++ files processed with sourceCpp
21 // (useful for testing and development). The R code will be automatically
22 // run after the compilation.
23 //
24
25
26 // (Top Level) :
```

Console ~ /

R IS FREE SOFTWARE AND COMES WITH ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/.RData]

>

Apresentação

The screenshot shows the RStudio IDE interface. The left pane displays a C++ code editor for a file named 'Untitled1.R' containing code for the Rcpp package. The right pane shows the R console output.

```
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 }
```

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[workspace loaded from ~/.RData]

>

} Sites para consultas

Environment is empty

Files Plots Packages

Install Update

User Library

- ... Functor 0 to Inline C, C++, Fortran Functor

Script dos exemplos

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Source on Save.
- Code Editor:** Displays two files: MiniCursoSER.R and functions.cpp. The functions.cpp file contains the following C++ code:

```
4 // [[Rcpp::export]]
5 NumericVector f_if_Rcpp(NumericVector x, double a, double b){
6     int n = x.length(); NumericVector out(n);
7     for(int i=0; i<n; i++){
8         double xi=x[i];
9         if(xi<a){out[i]=a;} else if (xi>b){out[i]=b;} else {out[i]=xi;}
10    }
11    return(out);
12 }
13
14
15
16 // [[Rcpp::export]]
17 int fibRcpp(const int x) {
18     if(x<2){return x;}
19     else {return (fibRcpp(x-1) + fibRcpp(x-2)) ;}
20 }
```

- Console:** Shows the command >.
- Environment View:** Shows the Global Environment with variable x defined as num [1:1000] -0.8673 0... and a list of Functions:

 - cppfun function (x, f)
 - cpplist function (x, y, z)
 - f_colchet function (x, a, b)
 - f_if function (x, a, b)
 - f_if_Rcpp function (x, a, b)
 - f_if_

- Files View:** Shows the User Library and System Library:

 - User Library:** microb... Accurate Timing Functions (1.4-6), Rcpp Seamless R and C++ Integration (1.0.1), RcppAr... 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library (0.9.30).
 - System Library:** boot Bootstrap Functions (Originally by Angelo Canty for S) (1.3-19), class Functions for Classification (7.3-14), cluster "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. (2.0.6).

Tipos de objetos (classes) e Recursos

Tipos de objetos nativos do C++ que vamos usar:

- ▶ **double**: representa valores reais como 1.5 ou $-\frac{5}{7}$;
- ▶ **bool**: valores lógico de TRUE ou FALSE;
- ▶ **int**: valores inteiros como 3 ou -1.

Tipos de objetos (classes) e Recursos

O **Rcpp** também fornece classes de objetos como esses tipos.

- ▶ **IntegerVector** e **IntegerMatrix** : Usados para vetores e matrizes (elementos `int`);
- ▶ **NumericVector** e **NumericMatrix** : para vetores e matrizes (elementos `double`);
- ▶ **LogicalVector** e **LogicalMatrix**: para vetores e matrizes do tipo lógico (elementos `bool`);

Tipos de objetos (classes) e Recursos

- ▶ `CharacterVector`: para vetores do tipo caracter;
- ▶ `List`: utilizado para listas com quaisquer tipos de objetos;
- ▶ `Function`: usado para recorrer a funções;
- ▶ Os operadores `[]` e `()` acessam os elementos de vetores e matrizes (no **RcppArmadillo** eles são acessados via `()`);
- ▶ Os comandos `.length()` ou `.size()`: fornecem o comprimento de vetores ou matrizes;
- ▶ `.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.

Básico da linguagem C++

O C++ é similar ao R em muitas formas. Mas algumas diferenças são bastante importantes.

- ▶ As variáveis em C++ são **estaticamente tipadas** (checadas na hora de compilar).
- ▶ Uma vez declarada a variável não muda sua classe. Isso se aplica a valores retornados por funções.
- ▶ As expressões em C++ devem terminar com ; .
- ▶ O indexador começa em 0.

Na linguagem R (ou Python) você não precisa declarar previamente o tipo de variável ao criá-la (**Tipagem dinâmica**).

Comandos importantes

Dois recursos podem ser utilizados para carregar funções criadas em C++ para o R.

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

Estes comandos permitem criar, compilar e ligar um arquivo em C++ criando uma função no R para acessá-los.

Na maioria das vezes nossos códigos (funções) em C++ estarão em arquivos com extensão .cpp.

Vamos começar devagar!!

Primeiras Funções

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Code Editor:** Two files are open: "MiniCursoSER.R" and "functions.cpp". The "functions.cpp" file contains the following R code:

```
47 rm(list=ls())
48 #install.packages("Rcpp")
49 library("Rcpp")
50 wd = getwd()
51 sourceCpp(paste(wd,"/functions.cpp",sep=""))
52
53 gvn1()
54 gvn2()
55 gv11()
56 gv12()
57 gmn()
58 gm1()
```
- Tooltips:** A tooltip is displayed over the "Run" button in the toolbar, stating "Run the current line or selection (Ctrl+Enter)".
- Environment:** Shows "Environment" and "History" tabs. The environment is empty.
- Console:** The console window is empty, showing the prompt ">".
- Packages:** The "Packages" tab is selected in the bottom right panel. It lists the "User Library" with the following packages:

Name	Description	Version
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.1! (0)
micro..	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration via Rcpp	1.0.1 (0)

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays the file `functions.cpp` containing R code to source C++ functions.
- Console:** Shows the output of running the code in the editor, which includes the execution of Rcpp library and the source of `functions.cpp`.
- Environment View:** Shows the global environment with various functions defined, such as `f_if_RC_`, `fibRcpp`, `gml`, etc.
- File Browser:** Shows the user library with installed packages like `inline`, `micro...`, and `Rcpp`.

A red arrow points from the environment view towards the code editor, highlighting the connection between the source code and the resulting environment objects.

```
47 rm(list=ls())
48 #install.packages("Rcpp")
49 library("Rcpp")
50 wd = getwd()
51 sourceCpp(paste(wd,"/functions.cpp",sep=""))
52
53 gvn1()
54 gvn2()
55 gvl1()
56 gvl2()
57 gmn()
58 gml()
59
60
61
62
```

```
> library("Rcpp")
> wd = getwd()
> sourceCpp(paste(wd,"/functions.cpp",sep=""))
>
```

Environment

Name	Description
wd	"C:/Users/eric675360.."
values	
f_if_RC_	function (x, a, b)
fibRcpp	function (x)
gml	function ()
gmn	function ()
gvl1	function ()
gvl2	function ()
gvn1	function ()
gvn2	function ()

Files Plots Packages Help Viewer

User Library

Name	Description	Version
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.1!
micro...	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration via Rcpp	1.0.1

Primeiras Funções

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays the file `functions.cpp` containing R code to source C++ functions.
- Environment Pane:** Shows the global environment with the following objects:
 - `wd` pointing to the path `"C:/Users/eric675360..."`
 - A list of functions:
 - `f_if_Rc_` function (x, a, b)
 - `fibRcpp` function (x)
 - `gml` function ()
 - `gmn` function ()
 - `gv11` function ()
 - `gv12` function ()
 - `gvn1` function ()
 - `gvn2` function ()
- Global Environment:** Shows the global environment with the same objects listed above.
- Console:** Displays the command history:

```
> library("Rcpp")
> wd = getwd()
> sourceCpp(paste(wd,"/functions.cpp",sep=""))
> |
```
- Tools:** Shows the User Library with installed packages:
 - `inline`: Functions to Inline C, C++, Fortran Function Calls from R
 - `micro..`: Accurate Timing Functions
 - `Rcpp`: Seamless R and C++

Red arrows highlight the connection between the function definitions in the code editor and their corresponding objects in the environment panes.

```
47 rm(list=ls())
48 #install.packages("Rcpp")
49 library("Rcpp")
50 wd = getwd()
51 sourceCpp(paste(wd,"/functions.cpp",sep=""))

52
53 gvn1()
54 gvn2()
55 gv11()
56 gv12()
57 gmn()
58 gml()

59
60
61
62
```

```
> library("Rcpp")
> wd = getwd()
> sourceCpp(paste(wd,"/functions.cpp",sep=""))
> |
```

Name	Description	Version
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.1! (0)
micro..	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++	1.0.1 (0)

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Two tabs are open: "MiniCursoSER.R" and "functions.cpp". The "functions.cpp" tab contains the following C++ code:

```
21 // [[Rcpp::export()]]
22 Rcpp::NumericVector gvn1(){
23     /* cria um vetor de tamanho 4 com
24      entradas sendo o valor zero */
25     Rcpp::NumericVector x(4);
26     return(x);
27 }
28
29
30 // [[Rcpp::export()]]
31 Rcpp::NumericVector gvn2(){
32     /* cria um vetor de tamanho 4 com
33      entradas sendo o valor 7.3 */
34     Rcpp::NumericVector x(4, 7.3);
35     return(x);
36 }
```

- Console:** Shows the R session output:

```
> library("Rcpp")
> wd = getwd()
> sourceCpp(paste(wd,"/functions.cpp",sep=""))
> |
```
- Environment:** Shows the current working directory: "wd" pointing to "C:/Users/eric675360..".
- Functions:** A list of functions defined in the current environment, including f_if_Rc_, fibRcpp, gml, gmn, gvn1, gvn2, and gvn2.
- User Library:** A list of installed packages: inline (0.3.1!), micro.. (1.4-6), and Rcpp (1.0.1).

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Shows two files: MiniCursoSER.R and functions.cpp. The functions.cpp file contains the following C++ code:

```
21 // [[Rcpp::export()]]
22 Rcpp::NumericVector gvn1(){
23     /* cria um vetor de tamanho 4 com
24      entradas sendo o valor zero */
25     Rcpp::NumericVector x(4);
26     return(x);
27 }
28
29
30 // [[Rcpp::export()]]
31 Rcpp::NumericVector gvn2(){
32     /* cria um vetor de tamanho 4 com
33      entradas sendo o valor 7.3 */
34     Rcpp::NumericVector x(4, 7.3);
35     return(x);
36 }
```

- Environment View:** Shows the current working directory (wd) as "C:/Users/eric675360..".
- Functions View:** Lists the available functions: f_if_Rc_, fibRcpp, gml, gmn, gvn1, gvn1, and gvn2.
- Console View:** Displays the R session history:

```
> library("Rcpp")
> wd = getwd()
> sourceCpp(paste(wd,"/functions.cpp",sep=""))
> gvn1()
[1] 0 0 0 0
> gvn2()
[1] 7.3 7.3 7.3 7.3
>
```

- User Library View:** Shows installed packages: inline (0.3.1!), microbenchmark (1.4-6), and Rcpp (1.0.1).

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Shows two files: MiniCursoSER.R and functions.cpp. The functions.cpp file contains the following Rcpp code:

```
38 // [[Rcpp::export()]]
40 Rcpp::LogicalVector gvl1(){
41  /* cria um vetor de tamanho 4 com
42   entradas sendo o valor FALSE */
43  Rcpp::LogicalVector x(4);
44  return(x);
45 }
46
47 // [[Rcpp::export()]]
48 Rcpp::LogicalVector gvl2(){
49  /* cria um vetor de tamanho 4 com
50   entradas sendo o valor TRUE */
51  Rcpp::LogicalVector x(4, true);
52  return(x);
53 }
```

- Environment Tab:** Shows the current working directory (wd) is "C:/Users/eric675360..".
- Values Tab:** Lists variables: f_if_Rc_, fibRcpp, gml, gmn, gvl1, gvl2, gvn1, gvn2.
- Functions Tab:** Lists functions: f_if_Rc_, fibRcpp, gml, gmn, gvl1, gvl2, gvn1, gvn2.
- Console Tab:** An empty command line interface.
- Files Tab:** Shows packages installed: inline, micro., Rcpp.
- Plots Tab:** No plots displayed.
- Packages Tab:** Shows the User Library with packages: inline (0.3.1!), micro. (1.4-6), Rcpp (1.0.1).

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Shows two files: MiniCursoSER.R and functions.cpp. The functions.cpp file contains the following Rcpp code:

```
38 // [[Rcpp::export()]]
39 Rcpp::LogicalVector gvl1(){
40   /* cria um vetor de tamanho 4 com
41      entradas sendo o valor FALSE */
42   Rcpp::LogicalVector x(4);
43   return(x);
44 }
45
46
47 // [[Rcpp::export()]]
48 Rcpp::LogicalVector gvl2(){
49   /* cria um vetor de tamanho 4 com
50      entradas sendo o valor TRUE */
51   Rcpp::LogicalVector x(4, true);
52   return(x);
53 }
```
- Console:** Displays the output of the R session:

```
> gvl1()
[1] FALSE FALSE FALSE FALSE
> gvl2()
[1] TRUE TRUE TRUE TRUE
>
```
- Environment Viewer:** Shows the current environment with the following objects:

values	Functions
wd "C:/Users/eric675360.."	f_if_Rc... function (x, a, b)
	fibRcpp function (x)
	gml function ()
	gmn function ()
	gvl1 function ()
	gvl2 function ()
	gvn1 function ()
	gvn2 function ()
- Tools:** Files, Plots, Packages, Help, Viewer.
- User Library:** Shows installed packages:

Name	Description	Version
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.1!
micro..	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration via Rcpp	1.0.1

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Shows two files: MiniCursoSER.R and functions.cpp. The functions.cpp file contains the following C++ code:

```
55 // [[Rcpp::export()]]
56 Rcpp::NumericMatrix gmn(){
57     // cria uma matriz 4 por 6 com
58     // entradas sendo o valor zero
59     Rcpp::NumericMatrix x(4, 6);
60     return(x);
61 }
62
63
64 // [[Rcpp::export()]]
65 Rcpp::LogicalMatrix gml(){
66     // cria uma matriz 4 por 6 com
67     // entradas sendo o valor FALSE
68     Rcpp::LogicalMatrix x(4, 6);
69     return(x);
70 }
```

- Environment Tab:** Shows the current working directory (wd) as "C:/Users/eric675360..".
- Values Tab:** Lists variables: f_if_Rc_, fibRcpp, gml, gmn, gvn1, gvn2.
- Functions Tab:** Lists functions: f_if_Rc_ (function), fibRcpp (function), gml (function), gmn (function), gvn1 (function), gvn2 (function).
- Console Tab:** An empty command line interface.
- Tools Tab:** Shows the User Library with packages: inline (0.3.1!), micro.. (1.4-6), and Rcpp (1.0.1).

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Two tabs are open: "MiniCursoSER.R" and "functions.cpp". The "functions.cpp" tab contains the following C++ code:

```
55 // [[Rcpp::export()]]
56 Rcpp::NumericMatrix gmn(){
57     // cria uma matriz 4 por 6 com
58     // entradas sendo o valor zero
59     Rcpp::NumericMatrix x(4, 6);
60     return(x);
61 }
62
63
64 // [[Rcpp::export()]]
65 Rcpp::LogicalMatrix gml(){
66     // cria uma matriz 4 por 6 com
67     // entradas sendo o valor FALSE
68     Rcpp::LogicalMatrix x(4, 6);
69     return(x);
70 }
```

- Console:** Displays the results of running the functions:

```
> gmn()
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    0    0    0    0    0    0
[2,]    0    0    0    0    0    0
[3,]    0    0    0    0    0    0
[4,]    0    0    0    0    0    0
> gml()
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,] FALSE FALSE FALSE FALSE FALSE FALSE
[2,] FALSE FALSE FALSE FALSE FALSE FALSE
[3,] FALSE FALSE FALSE FALSE FALSE FALSE
[4,] FALSE FALSE FALSE FALSE FALSE FALSE
> |
```
- Environment Browser:** Shows the current workspace with the following objects:

values	Functions
wd "C:/Users/eric675360.."	f_if_Rc... function (x, a, b)
	fibRcpp function (x)
	gml function ()
	gmn function ()
	gv11 function ()
	gv12 function ()
	gvn1 function ()
	gvn2 function ()

- Tools:** Files, Plots, Packages, Help, Viewer.
- Install:** inline (0.3.1!), C++, Fortran Function Calls from R, micro.. (1.4-6), Rcpp (1.0.1).

Primeiras Funções

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Editor:** Shows the file `functions.cpp` containing R and C++ code. The code includes functions for vectors and matrices using the Rcpp package.
- Environment:** Global Environment pane shows "Environment is empty".
- Tools:** Shows the User Library and System Library sections. The User Library contains checked entries for `micro...`, `Rcpp`, and `Rcpp...`. The System Library contains an entry for `boot`.
- Console:** The console window is empty, showing the prompt `> |`.

```
1 rm(list=ls())
2 #install.packages("Rcpp")
3 library("Rcpp")
4 library("microbenchmark")
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 funcao1(NumericVector x){ return(x);}
12        ')
13
14 funcao1(c(1,2,3,4))
15
16 #exemplo que entra-se com uma matriz e sai uma matriz
17 sourceCpp(code='#include <Rcpp.h>
18         using namespace Rcpp;
19
20        //[[Rcpp::export]]
21        NumericMatrix funcao2(NumericMatrix x){ return(x);}
22        ')
23
24 funcao2(matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3))|
```

Primeiras Funções

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Code Editor:** Shows a file named "functions.cpp" containing Rcpp code. A red arrow points to the "Run" button in the toolbar above the editor.
- Run Dialog:** A tooltip for the "Run" button says "Run the current line or selection (Ctrl+Enter)".
- Console:** Displays the R session history, including the execution of the Rcpp code.
- Environment:** Shows the Global Environment and Functions pane.
- Functions:** Lists the function "funcao1" defined in the code.
- Packages:** Shows the User Library and System Library sections with installed packages like "microbenchmark", "Rcpp", and "boot".

```
functions.cpp
```

```
1 #include <Rcpp.h>
2 //using Rcpp
3 //example that enters with a character vector and outputs a vector
4 library("microbenchmark")
5 ##using Rcpp outside the environment .cpp
6 #example that enters with a vector and outputs a vector
7 sourceCpp(code='#include <Rcpp.h>
8         using namespace Rcpp;
9
10        //[[Rcpp::export]]
11        NumericVector funcao1(NumericVector x){ return(x);}
12    ')
13
14 funcao1(c(1,2,3,4))
15
16 #example that enters with a matrix and outputs a matrix
17 sourceCpp(code='#include <Rcpp.h>
18         using namespace Rcpp;
19
20        //[[Rcpp::export]]
21        NumericMatrix funcao2(NumericMatrix x){ return(x);}
22    ')
23
24 funcao2(matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3))
```

```
Console
```

```
> install.packages('Rcpp')
> library("Rcpp")
> library("microbenchmark")
> ##using Rcpp outside the environment .cpp
> #example that enters with a vector and outputs a vector
> sourceCpp(code='#include <Rcpp.h>
+         using namespace Rcpp;
+
+        //[[Rcpp::export]]
+        NumericVector funcao1(NumericVector x){ return(x);}
+
> funcao1(c(1,2,3,4))
[1] 1 2 3 4
```

Project: (None)

Environment History

Import Dataset List

Global Environment

Functions

funcao1 function (x)

Files Plots Packages Help Viewer

User Library

Name	Description	Version
micro...	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration	1.0.1
Rcpp...	'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library	0.9.3

System Library

Name	Description	Version
boot	Bootstrap Functions (Originally by Annelie	1.3-19

Primeiras Funções

A screenshot of the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The left pane shows an R script file named 'functions.cpp' with the following code:

```
1 //install.packages("Rcpp")
2 library("Rcpp")
3 library("microbenchmark")
4 ##usando o Rcpp fora do ambiente .cpp
5 #exemplo que entra-se com um vetor e sai um vetor
6 sourceCpp(code="#include <Rcpp.h>
7           using namespace Rcpp;
8
9           //[[Rcpp::export]]
10          NumericVector funcao1(NumericVector x){ return(x); }
11          ')
12
13 funcao1(c(1,2,3,4))
14
15 #exemplo que entra-se com uma matriz e sai uma matriz
16 sourceCpp(code="#include <Rcpp.h>
17           using namespace Rcpp;
18
19           //[[Rcpp::export]]
20           NumericMatrix funcao2(NumericMatrix x){ return(x); }
21           ')
22
23 funcao2(matrix(c(1,2,3,4,5,6,7,8,9),nrow = 3))
24
25
26
```

The right pane displays the 'Functions' panel with two entries: 'funcao1' and 'funcao2'. A red arrow points from the 'funcao1' entry towards the bottom of the screen. The bottom right corner of the RStudio window shows the system tray with icons for battery, signal, and volume.

The bottom-left pane is the 'Console' window, which shows the R command history and the output of the 'funcao2' function:

```
> #exemplo que entra-se com uma matriz e sai uma matriz
> sourceCpp(code="#include <Rcpp.h>
+           using namespace Rcpp;
+
+           //[[Rcpp::export]]
+           NumericMatrix funcao2(NumericMatrix x){ return(x); }
+           ')
>
> funcao2(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
```

Recursos Matemáticos e laços

Operadores matemáticos tais como: +, -, *, /, `pow()`, `sqrt()`; podem ser utilizados.

Os operadores lógicos: ==, !=, <=, >=, >, | (ou), & (e). Também podem ser usados da mesma forma como no R.

As estruturas de controles como laços (`for`) e condicionais como `if`, serão bastante usadas e vistas nos exemplos adiante.

Primeiras Funções

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Source on Save, Run, Source.
- Project:** Project: (None)
- Code Editor:** Shows two files: `rsoSER.R` and `primeirasfuncoes.cpp`. The `primeirasfuncoes.cpp` file contains C++ code for summing elements of a vector and defining Rcpp functions.
- Environment Tab:** Shows the Global Environment with two functions: `funcao1` and `funcao2`.
- Functions Tab:** Shows the definition of `funcao1` and `funcao2`.
- Files Tab:** Shows the User Library with the following packages:
 - inline**: Functions to Inline C, C++, Fortran Function Calls from R (version 0.3.15)
 - micro...**: Accurate Timing Functions (version 1.4-6)
 - Rcpp**: Seamless R and C++ Integration (version 1.0.1)
 - RcppA...**: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library (version 0.9.30)
- Console:** Displays the command `> |`.

Primeiras Funções

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Source on Save, Run, Source.
- Project:** Project: (None)
- Code Editor:** Shows two files: rsoSER.R and primeirasfuncoes.cpp. The primeirasfuncoes.cpp file contains C++ code for summing elements of a vector using Rcpp.
- Environment Tab:** Shows the Global Environment with functions: funcao1, funcao2, sumRcpp, and sumRcpp2.
- Files Tab:** Shows the User Library with packages: inline, microbenchmark, Rcpp, and RcppArmadillo.
- Console:** Displays the R command history and the resulting output for the Rcpp function.

```
25 #exemplo que entra-se com um vetor e sai a soma dos elementos
26 sourceCpp(code='#include <Rcpp.h>
27         using namespace Rcpp;
28
29         //[[Rcpp::export]]
30         double sumRcpp(NumericVector x){
31             int n=x.size();
32             double total=0;
33             for(int i=0; i<n; i++){
34                 total+=x[i];
35             }
36             return(total);
37         }
38
39
40 ##outra forma de criar funções no C++ e carrega-las para o R
41 code=" double sumRcpp2(NumericVector x){
42         int n=x.size();
43         double total=0;
44         for(int i=0; i<n; i++){
45             total+=x[i];
46         }
47         return(total);
48     }
49
50
51 sumRcpp2(c(6,2,4,3,5))
52 sumRcpp(c(6,2,4,3,5))
53 <--"
```

```
+     double total=0;
+     for(int i=0; i<n; i++){
+         total+=x[i];
+     }
+     return(total);
+
> cppFunction(code)
> |
```

Primeiras Funções

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Contains R and C++ code for summing elements of a vector.
- Console:** Shows the execution of the code and the resulting output.
- Environment:** Shows the global environment with functions defined.
- Files:** Shows the project structure.

Code in RStudio:

```
26 #exemplo que entra-se com um vetor e sai a soma dos elementos
27 sourceCpp(code="#include <Rcpp.h>
28         using namespace Rcpp;
29
30         //[[Rcpp::export]]
31         double sumRcpp(NumericVector x){
32             int n=x.size();
33             double total=0;
34             for(int i=0; i<n; i++){
35                 total+=x[i];
36             }
37             return(total);
38         }
39
40 ##outra forma de criar funções no C++ e carrega-las para o R
41 code=' double sumRcpp2(NumericVector x){
42     int n=x.size();
43     double total=0;
44     for(int i=0; i<n; i++){
45         total+=x[i];
46     }
47     return(total);}'
48
49 cppFunction(code)
50
51 sumRcpp2(c(6,2,4,3,5))
52 sumRcpp(c(6,2,4,3,5))
53
54 }
```

Console Output:

```
+         ,return(total);}
+
> cppFunction(code)
> sumRcpp2(c(6,2,4,3,5))
[1] 20
> sumRcpp(c(6,2,4,3,5))
[1] 20
>
```

Environment pane:

Values	Functions
code " double sumRcpp2(Num...	funcao1 function (x)
	funcao2 function (x)
	sumRcpp function (x)
	sumRcpp2 function (x)

Files pane:

Files	Plots	Packages	Help	Viewer
Install	Update			
Name	Description	V...		

User Library pane:

User Library	Functions to Inline C, C++, Fortran Function Calls from R	0.3.15
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.15
micro...	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration	1.0.1
RcppA...	'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library	0.9.30

A red bracket on the right side of the console output is labeled "Resultados!".

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Print, and Run, along with Go to file/function, Source on Save, Run, and Source buttons.
- Code Editor:** Shows an R script (rsoSER.R) and a C++ source file (primeirasfuncoes.cpp). The R script includes code to define a C++ function and call it from R.

```
54 #funcao sign criada em .cpp
55 code2=' int signcpp(int x){
56     if(x > 0){
57         return(1);
58     } else if (x==0){
59         return(0);
60     } else {
61         return(-1);
62     }
63 }'
64
65 cppFunction(code2)
66 signcpp(-57)
67 signcpp(0)
68 signcpp(9)
69 |
70 < [Top Level]
```

- Environment Panel:** Shows the Global Environment, Values, and Functions sections. The Functions section lists four functions: funcao1, funcao2, sumRcpp, and sumRcpp2.

Name	Description	V...
funcao1	function (x)	
funcao2	function (x)	
sumRcpp	function (x)	
sumRcpp2	function (x)	

- Console:** Displays the command prompt (>) and the R environment.
- Tools Panel:** Includes sections for Files, Plots, Packages, Help, and Viewer, along with an Install and Update button.
- User Library:** Lists installed packages: inline (0.3.15), micro... (1.4-6), and Rcpp (1.0.1).

Primeiras Funções

The screenshot shows the RStudio interface with several windows open:

- Code Editor:** Shows the file `primeirasfuncoes.cpp` containing the following C++ code:

```
54 #funcão sign criada em .cpp
55 code2=' int signCpp(int x){
56         if(x > 0){
57             return(1);
58         } else if (x==0){
59             return(0);
60         } else {
61             return(-1);
62         }
63     }'
64
65 cppFunction(code2)
66 signCpp(-57)
67 signCpp(0)
68 signCpp(9)
69
70
```
- Console:** Shows the R command `cppFunction(code2)` being run, which outputs the generated R code:

```
+         } else if (x==0){
+             return(0);
+         } else {
+             return(-1);
+         }
>
```
- Environment:** Shows the global environment with the following objects:

Values
code " double sumRcpp2(Num..."
code2 " int signCpp(int x){..."

Functions

Functions
funcao1 function (x)
funcao2 function (x)
signCpp function (x)
sumRcpp function (x)
sumRcpp2 function (x)
- Packages:** Shows the user library with the following packages:

Name	Description	V...
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.15
micro...	Accurate Timing Functions	1.4- 6
Rcpp	Seamless R and C++	1.0.1

Primeiras Funções

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays the file `primeirasfuncoes.cpp` containing C++ code for a `sign` function.
- Console:** Shows the R session history with the execution of the C++ code via the `cppFunction` command and the resulting output for different input values.
- Environment:** Shows the global environment with various functions defined.
- Help:** Shows the user library with installed packages like `inline`, `micro...`, and `Rcpp`.

A red arrow points from the "Run the current line or selection (Ctrl+Enter)" tooltip in the top right towards the Environment pane. A red bracket on the left side of the console output groups the last three lines of the session history under the heading "Resultados!".

```
53
54 #funcao sign criada em .cpp
55 code2=' int signCpp(int x){
56     if(x > 0){
57         return(1);
58     } else if (x==0){
59         return(0);
60     } else {
61         return(-1);
62     }
63 }'
64
65 cppFunction(code2)
66 signCpp(-57)
67 signCpp(0)
68 signCpp(9)
69
```

```
>
> cppFunction(code2)
> signCpp(-57)
[1] -1
> signCpp(0)
[1] 0
> signCpp(9)
[1] 1
>
```

Resultados!

Run the current line or selection (Ctrl+Enter)

Global Environment

values

code	" double sumRcpp2(Num...
code2	" int signCpp(int x){...

Functions

funcao1	function (x)
funcao2	function (x)
signCpp	function (x)
sumRcpp	function (x)
sumRcpp2	function (x)

Files Plots Packages Help Viewer

User Library

Name	Description	V...
inline	Functions to Inline C, C++, Fortran Function Calls from R	0.3.15
micro...	Accurate Timing Functions	1.4- 6
Rcpp	Seamless R and C++	1.0.1

Primeiras Funções

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** MiniCursoSER.R, MinicursoSERprimeirasfuncoes.R, primeirasfuncoes.cpp.
- Code Editor:** Shows the `primeirasfuncoes.cpp` file containing C++ code for calculating Euclidean distance.
- Environment Tab:** Displays the Global Environment, showing variables like `code`, `code2`, `wd`, and `x`.
- Values Tab:** Shows variable values: `code` and `code2` are the C++ code snippets; `wd` is the working directory; `x` is a vector of integers [1:100] with values: 3 1 6 3 2 6 5 ...
- Functions Tab:** Shows function definitions: `cppfun`, `cpplist`, `f`, `funcao1`, `funcao2`, `Rcppfunc`, `rfun`, `rlist`, `signcpp`, and `signr`.
- Console:** Shows the command `>` and the prompt for the next input.
- Bottom Navigation:** Includes tabs for Files, Plots, Packages, Help, and Viewer, along with standard window controls.

```
#include <Rcpp.h>
using namespace Rcpp;

// exemplo da distancia euclidiana entra um vetor e sai um escalar
[[Rcpp::export]]
double distEuclid(NumericVector x, NumericVector y){
  int n = x.size();
  double out=0;
  for(int i = 0; i < n; i++){
    out += pow(x[i]-y[i], 2);
  }
  return( sqrt(out) );
}
```

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays R and C++ code. The R code includes `sourceCpp` calls to a file named `primeirasfuncoes.cpp`. The C++ code defines several functions: `distEuclid`, `dist`, `funcao1`, `funcao2`, `Rcppfunc`, `rfun`, `rlist`, `signCpp`, and `sumP`.
- Console:** Shows the command `>` indicating the R prompt.
- Environment:** Shows the global environment with variables like `code`, `code2`, `wd`, and `x` defined.
- Files:** Shows the user library containing the `inline` package.
- Plots:** Shows the history of plots generated in the session.
- Packages:** Shows the installed packages.
- Help:** Shows help documentation for various functions.
- Viewer:** Shows the results of the current session.

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Source on Save, Run, Source.
- Code Editor:** Shows the content of `primeirasfuncoes.cpp`. A tooltip indicates: "Run the current line or selection (Ctrl+Enter)".
- Console:** Displays the R session output:

```
> #exemplo distancia euclidiana, entra vetores sai um escalar
> wd = getwd()
> sourceCpp(paste(wd,"/primeirasfuncoes.cpp",sep=""))
>
> distEuclid(c(1,3,5),c(3,4,5))      #funcao no arquivo .cpp
[1] 2.236068
> dist(matrix(c(1,3,4,3,5,5),nrow=2) ) #funcao do proprio R
1
2 2.236068
>
```
- Environment:** Shows variables and their values.
- Functions:** Shows the definitions of the functions used in the code.
- Project:** Shows the current project is "None".
- Bottom Navigation:** Files, Plots, Packages, Help, Viewer, Install, Update.

Primeiras Funções

Alguns pontos em que o desempenho do R deixa a desejar são os laços não vetorizáveis (uma iteração depende da anterior). Veja a diferença entre um laço em C++ e outro em R a seguir.

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Standard RStudio icons for file operations, search, and run.
- Project Bar:** Project: (None).
- Code Editor:** Displays two files:
 - MiniCursoSER.R (Top Level):

```
58 #funcao Soma criada no R
59 sumR=function(x){
60   total=0
61   for(i in 1:length(x)){
62     total=total+x[i]
63   }
64 }
65 x=rpois(1000,6)
66 microbenchmark(sumR(x),sumRcpp(x),sum(x),times=1000) #veja que
```
 - functions.cpp (Bottom Level):

```
#include <Rcpp.h>
using namespace Rcpp;
```
- Environment Tab:** Shows the Global Environment with values and functions defined.
- Console Tab:** An empty console window.
- Help Tab:** Shows the User Library with installed packages:

Name	Description	Version
micro...	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration	1.0.1
Rcpp...	'Rcpp' Integration for the 'Armadillo'	0.9.3i

Primeiras Funções

The screenshot shows the RStudio interface with the following components:

- Code Editor:** Displays R and C++ code. The R code defines a function `sumR` that calculates the sum of elements in a vector `x` using a for loop. The C++ code in `functions.cpp` includes the same logic.
- Console:** Shows the execution of the R code and a microbenchmark comparison between `sumR`, `sumRcpp`, and `sum`. The output indicates that `sumRcpp` is significantly faster than the other functions.
- Environment:** Shows the global environment with objects like `code`, `x`, and several `sum` functions from different packages.
- Files:** Shows the project structure with files like `MiniCursoSER.R` and `MinicursoSERprimeirasfuncoes.R`.
- PLOTS:** Shows the results of the microbenchmarking as a table:

	expr	min	lq	mean	median	uq	max	neval
sumR(x)	64.366	65.829	71.998755	66.195	66.561	4228.360	1000	
sumRcpp(x)	5.486	6.218	7.178903	6.949	7.680	57.051	1000	
sum(x)	1.097	1.463	1.637989	1.464	1.829	13.532	1000	

Primeiras Funções

O objetivo não é sair do R!!

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays an R script named "functions.cpp" containing C++ code for a sum function.
- Console:** Shows the R code being run and its output. The output table is highlighted with a red box.
- Environment:** Shows the global environment with various functions defined.
- Tools:** Shows the user library with packages like microbenchmark, Rcpp, and RcppArmadillo installed.

Code in "functions.cpp":

```
#funcao Soma criada no R
sumR=function(x){
  total=0
  for(i in 1:length(x)){
    total=total+x[i]
  }
}
x=rpois(1000,6)
microbenchmark(sumR(x),sumRcpp(x),sum(x),times=1000) #veja que o tempo da funcao sum é menor
```

Output in Console:

```
Unit: microseconds
expr      min       q1      mean   median       uq      max neval
sumR(x) 64.366 65.829 71.998755 66.195 66.561 4228.360 1000
sumRcpp(x) 5.486 6.218 7.178903 6.949 7.680  57.051 1000
sum(x)   1.097 1.463 1.637989 1.464 1.829 13.532 1000
```

A tooltip "Run the current line or selection (Ctrl+Enter)" is visible over the code editor.

Exemplos com listas

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Code Editor:** Three tabs are visible: MiniCursoSER.R, MinicursoSERprimeirasfuncoes.R, and primeirasfuncoes.cpp. The primeirasfuncoes.cpp tab contains the following C++ code:

```
1 #include <Rcpp.h>
2 using namespace Rcpp;
3
4 /* exemplos com listas */
5
6 //[[[Rcpp::export]]]
7 List cpplist(int x, CharacterVector y, NumericVector z){
8     return List::create(Rcpp::Named("x", rnorm(x,0,1)),
9                         Rcpp::Named("y", y),
10                        Rcpp::Named("z", pow(z,2) ) );
11 }
12
13 }
```

- Environment Pane:** Shows the **values** and **Functions** sections. The **values** section displays the code and its output:

code	x
#include <Rcpp.h>\...	int [1:1000] 5 5 3 ...

- Functions Section:** Lists the available functions:

 - funcao1 function (x)
 - funcao2 function (x)
 - sumR function (x)
 - sumRcpp function (x)
 - sumRcpp2 function (x)

- Console:** Shows the prompt >
- Packages Pane:** Shows the User Library with the following packages listed:

Name	Description	Version
micro...	Accurate Timing Functions	1.4-6
Rcpp	Seamless R and C++ Integration	1.0.1
Rcpp...	'Rcpp' Integration for the 'Armadillo'	0.9.30

Exemplos com listas

The screenshot shows the RStudio interface with several windows open:

- Code Editor:** Shows R and C++ code. The R code defines a function `rlist` that generates a list of random numbers. The C++ code includes a header file and defines a function `cpplist` that creates a list of strings.
- Environment:** Shows the global environment with variables `values` and `Functions`. `values` contains a list with elements `x` (a vector of integers) and `code` (the C++ code). `Functions` lists five functions: `funcao1`, `funcao2`, `sumR`, `sumRcpp`, and `sumRcpp2`.
- Console:** Displays the prompt `>` and the results of the R code execution.
- Packages:** Shows the user library with packages `micro...`, `Rcpp`, and `Rcpp...` installed.

```
69 ##exemplos com listas
70 sourceCpp(paste(wd,"/primeirasfuncoes.cpp",sep=""))
71 set.seed(9)
72 #exemplo no R
73 rlist=function(x,y,z){
74   return( list(x=rnorm(x), y=y, z=z^2) )
75 }
76
77 #lista no R
78 rlist(5,c("Adriana","Patricia","Renato","Silvio"),seq(-4,4))
79 #lista no Rcpp
80 cpplist(4,c("Adriana","Patricia","Renato","Silvio"),seq(-4,4))
81
```

70:53 (Top Level) R Script

Console >

Files Plots Packages Help Viewer

Install Update

Name	Description	V...
micro...	Accurate Timing Functions	1.4- 6
Rcpp	Seamless R and C++ Integration	1.0.1
Rcpp...	'Rcpp' Integration for the 'Armadillo'	0.9.3

Exemplos com listas

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None).
- Code Editor:** Shows R script code. A red arrow points to the "Run" button in the toolbar above the editor.
- Environment Tab:** Shows the Global Environment with objects like values and functions.
- Console Tab:** Shows the command line history:

```
> wd = getwd()
> sourceCpp(paste(wd,"/primeirasfuncoes.cpp",sep=""))
```
- Files Tab:** Shows the User Library with packages: micro... (Accurate Timing Functions), Rcpp (Seamless R and C++ Integration), and Rcpp... ('Rcpp' Integration for the 'Armadillo').

```
##exemplos com listas
wd = getwd()
sourceCpp(paste(wd,"/primeirasfuncoes.cpp",sep=""))
set.seed(9)
#exemplo no R
rlist=function(x,y,z){
  return( list(x=rnorm(x), y=y, z=z^2) )
}
#lista no R
rlist(5,c("Adriana","Patricia","Renato","Silvio"),seq(-4,4))
#lista no Rcpp
cpplist(4,c("Adriana","Patricia","Renato","Silvio"),seq(-4,4))
```

Exemplos com listas

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays R and C++ code. The R code defines a function `rlist` that generates a list of three elements: `x`, `y`, and `z`. The C++ code uses `Rcpp` to create a similar list.
- Console:** Shows the output of running the R script. It includes:
 - Variable `x` containing numerical values.
 - Variable `y` containing character strings: "Adriana", "Patricia", "Renato", and "silvio".
 - Variable `z` containing numerical values: 16, 9, 4, 1, 0, 1, 4, 9, 16.
- Environment:** Shows the current environment variables and their values.
- Functions:** Shows the available functions and their descriptions.
- Packages:** Shows the installed packages and their versions.
- User Library:** Shows the user library with checked boxes for `micro...`, `Rcpp`, and `Rcpp...`.

Exemplos de Funções que chamam funções

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Source on Save, Source.
- Code Editor:** Shows C++ code for two functions: `cppfun` and `Rcppfunc`.

```
12
13 /* exemplos de funções que chamam funções */
14
15 //[[Rcpp::export]]
16 NumericVector cppfun(NumericVector x, Function f){
17     NumericVector out(x.length());
18     for(int i=0; i<100; i++){
19         out=f(x);
20     }
21     return out;
22 }
23
24 /* chamando uma função do R no C++ */
25
26 //[[Rcpp::export]]
27 NumericVector Rcppfunc(NumericVector x, Function f){
28     NumericVector out=f(x);
29     return out;
30 }
31
```
- Environment Tab:** Shows the global environment with variables `values` and `Functions`.
 - values:** code, wd, x.
 - Functions:** cppfun, cpplist, funcao1, funcao2, Rcppfunc, rlist, sumR, sumRcpp, sumRcpp2.
- Console Tab:** Shows the command prompt >.
- Tools Tab:** Shows the User Library with installed packages:
 - inline: Functions to Inline C, C++, Fortran, Function Calls from R
 - micro...: Accurate Timing Functions
 - Rcpp: Seamless R and C++ Integration
 - Rcpp...: 'Rcpp' Integration for the 'Armadillo'

Exemplos de Funções que chamam funções

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays R and C++ code. The R code includes examples of inline C++ functions using the `inline` package and `microbenchmark` for performance testing.
- Environment:** Shows the global environment with variables like `code`, `wd`, and `x`, and functions like `cppfun`, `cpplist`, etc.
- Files:** Shows the user library with packages like `inline`, `micro...`, `Rcpp`, and `Rcpp...`.
- Console:** Shows the command prompt with the character `>`.

```
83 ##exemplos de funcoes que chamam funcoes
84 rfun=function(x,f){
85   for(i in 1:100){ out=f(x)}
86   return(out)
87 }
88 }
89
90 #exemplos com funcoes no R
91 rfun(c(1,4,9),sqrt)
92 rfun(c(1,4,9),function(x){1/x})
93 #exemplos com funcoes no Rcpp
94 cppfun(c(1,4,9),sqrt)
95 cppfun(c(1,4,9),function(x){1/x})
96
97 x=1:7
98 f=sqrt
99 microbenchmark(rfun(x,f),cppfun(x,f)) #veja aqui que o tempo no
100
101
102
```

Exemplos de Funções que chamam funções

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays an R script with code demonstrating inline C++ functions. The code includes a function `rfun` that loops through 1:100 and applies a function `f` to each element. It also shows examples of R functions (`sqrt`) and Rcpp functions (`cppfun`) being used.
- Console:** Shows the output of running the script, comparing the execution times of the R function `sqrt` and the Rcpp function `cppfun`.
- Environment:** Shows the global environment with variables like `code`, `wd`, and `x`, and functions like `cppfun`, `cpplist`, and `microbenchmark`.
- Files:** Shows the user library with packages like `inline`, `microbenchmark`, `Rcpp`, and `RcppArmadillo`.

```
##exemplos de funções que chamam funções
rfun=function(x,f){
  for(i in 1:100){ out=f(x)}
  return(out)
}

##exemplos com funções no R
rfun(c(1,4,9),sqrt)
rfun(c(1,4,9),function(x){1/x})

##exemplos com funções no Rcpp
cppfun(c(1,4,9),sqrt)
cppfun(c(1,4,9),function(x){1/x})

x=1:7
f=sqrt
microbenchmark(rfun(x,f),cppfun(x,f)) #veja aqui que o tempo no R é menor
```

```
+ return(out)
+
>
> #exemplos com funções 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 funções 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

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for New, Open, Save, Print, Go to file/function, Run, Source, and Addins.
- Code Editor:** Shows three tabs: MiniCursoSER.R, MinicursoSERprimeirasfuncoes.R, and primeirasfuncoes.cpp. The primeirasfuncoes.cpp tab is active, displaying the following code:

```
100
101
102 #exemplo: chamando uma função do R no C++
103
104 x=rnorm(1000)
105 fivenum(x) # retorna o mínimo,1 quartil, mediana, 3 quartil e m
106 sum(x)
107 Rcppfunc(x,fivenum)
108 Rcppfunc(x,sum)
109
110
111
```
- Console:** Shows the prompt > |
- Environment Tab:** Shows the Global Environment with variables and functions defined.
- Files Tab:** Shows the User Library with the **infl... Functions to Inline C, C++, Fortran Function Calls from R** package selected.

Exemplos de Funções que chamam funções

The screenshot shows the RStudio interface with several panes:

- Code Editor:** Displays R code in "primeirasfuncoes.R".

```
100
101
102 #exemplo: chamando uma função do R no C++
103
104 x=rnorm(1000)
105 fivenum(x) # retorna o mínimo,1 quartil, mediana, 3 quartil e maximo
106 sum(x)
107 Rcppfunc(x,fivenum)
108 Rcppfunc(x,sum)
109
110
111
```
- Console:** Shows the execution of the R code and its output.

```
> x=rnorm(1000)
> fivenum(x) # retorna o mínimo,1 quartil, mediana, 3 quartil e maximo
[1] -3.55764876 -0.59026997  0.05691132  0.66687964  3.14310294
> sum(x)
[1] 41.88714
> Rcppfunc(x,fivenum)
[1] -3.55764876 -0.59026997  0.05691132  0.66687964  3.14310294
> Rcppfunc(x,sum)
[1] 41.88714
>
```
- Environment:** Shows the global environment with variables like "values" and "Functions".
- Global Environment:** Shows the global environment with variables like "values" and "Functions".
- Help:** Shows the help menu with options like "Install", "Update", "Description", and "...".
- User Library:** Shows the user library with the "inline" package selected.

Exemplos de Funções que chamam funções

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for New, Open, Save, Print, Go to file/function, and Addins.
- Project Bar:** Project: (None)
- Code Editor:** Shows two files: MiniCursoSER.R and MinicursoSERprimeirasfuncoes.R. The latter contains the following C++ code:

```
26 //[[Rcpp::export]]
27 NumericVector Rcppfunc(NumericVector x, Function f){
28     NumericVector out=f(x);
29     return out;
30 }
31
32 /**
33  * R
34  * x=rpois(100,6)
35  * Rcppfunc(x,fivenum)
36  * Rcppfunc(x,sum)
37 */
38
39 (Top Level) ▾
```
- Environment Tab:** Shows the Global Environment with values and functions defined.
- Console Tab:** Shows the R prompt > |
- Help Tab:** Shows the User Library with the inline package selected.

Exemplos de Funções que chamam funções

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Addins dropdown.
- Code Editor:** Shows two tabs: "primeirasfuncoes.R" and "primeirasfuncoes.cpp". The "primeirasfuncoes.cpp" tab contains the following C++ code:

```
26 //[[Rcpp::export]]
27 NumericVector Rcppfunc(NumericVector x, Function f){
28     NumericVector out=f(x);
29     return out;
30 }
31
32 /*** R
33 x=rpois(100,6)
34 Rcppfunc(x,fivenum)
35 Rcppfunc(x,sum)
36 */
37
38:18 [C/C++ Chunk]
```
- Console:** Displays R session output:

```
> x=rpois(100,6)
> Rcppfunc(x,fivenum)
[1] 1.0 4.0 6.0 7.5 14.0
> Rcppfunc(x,sum)
[1] 588
> |
```
- Environment:** Shows the Global Environment with variables "values" and "Functions".
- Global Environment:** Shows the Global Environment with variables "values" and "Functions".
- Help:** Files, Plots, Packages, Help, View.
- User Library:** Shows the "inline" package selected, with version 0.3.0.

Exercícios



Armadillo

C++ library for linear algebra & scientific computing

Álgebra Linear

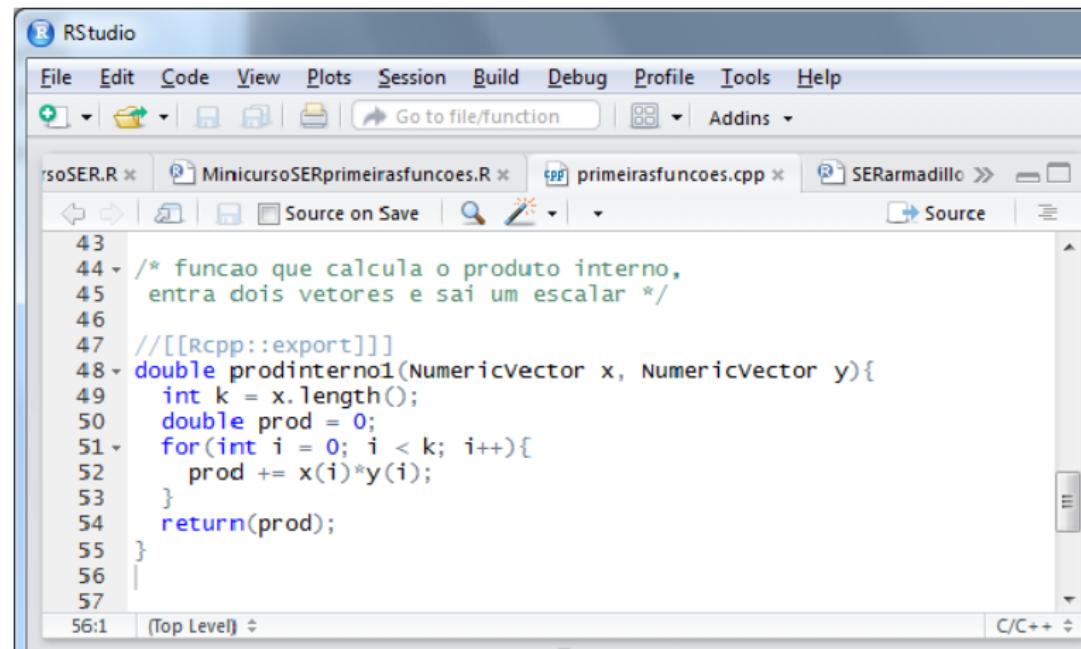
Estudos envolvendo Matrizes tem uma extrema importância para nós.

Estruturas de dados dos tipos de classes do **Rcpp** não nos fornecem ferramentas necessárias para que possamos evitar fazer implementações básicas ou elementares de algoritmos.

A implementação desses algoritmos feita manualmente pode ser bastante tedioso e propenso a erros.

Álgebra Linear

Nesta classe do **Rcpp** usando vetores, precisa-se de 1 *loop*. Para matrizes precisamos de 2 *loops*.



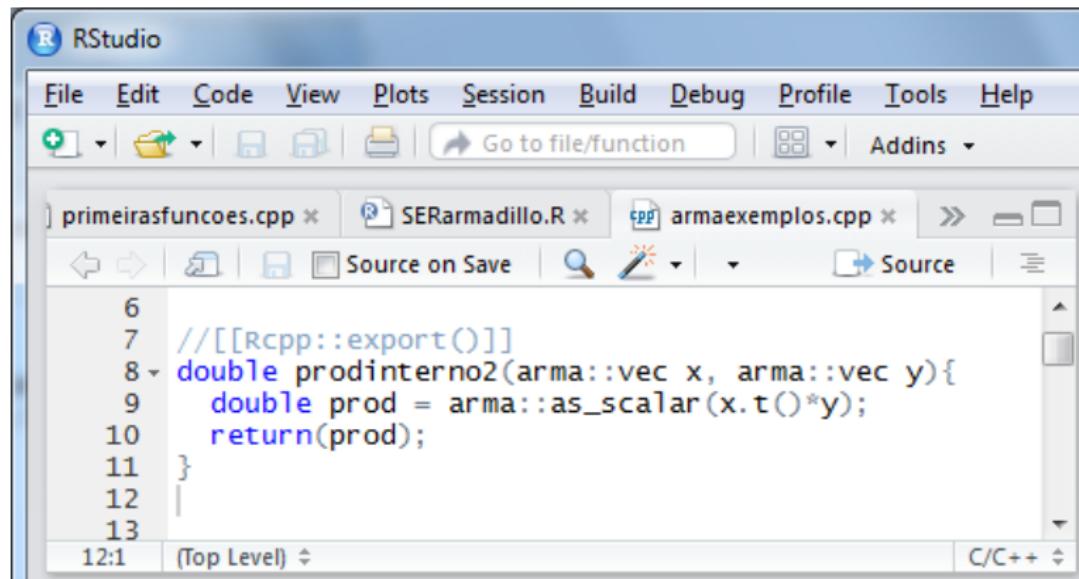
The screenshot shows the RStudio interface with two code files open. On the left is an R script file named 'soSER.R' containing the following code:

```
43
44 /* funcao que calcula o produto interno,
45 entra dois vetores e sai um escalar */
46
47 //[["Rcpp::export"]]
48 double prodinterno1(NumericVector x, NumericVector y){
49     int k = x.length();
50     double prod = 0;
51     for(int i = 0; i < k; i++){
52         prod += x(i)*y(i);
53     }
54     return(prod);
55 }
56
57
```

On the right is a C++ file named 'primeirasfuncoes.cpp' containing the same code. The tabs at the top show 'soSER.R', 'MinicursoSERprimeirasfuncoes.R', 'primeirasfuncoes.cpp', and 'SERarmadillo'. The status bar at the bottom indicates 'C/C++'.

Álgebra Linear no RcppArmadillo

O Armadillo permite que o código se pareça com a notação matemática padrão que estamos habituados.



The screenshot shows the RStudio interface with the following details:

- Title Bar:** RStudio
- Menu Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help
- Toolbar:** Includes icons for New, Open, Save, Print, Go to file/function, and Addins.
- Project Explorer:** Shows three files: primeirasfuncoes.cpp (active), SERarmadillo.R, and armaexemplos.cpp.
- Code Editor:** Displays the following C++ code:

```
6
7 //[[Rcpp::export()]]
8 double prodinterno2(arma::vec x, arma::vec y){
9     double prod = arma::as_scalar(x.t()*y);
10    return(prod);
11 }
12
13
```

The code editor status bar indicates line 12:1 and Top Level. The bottom right corner shows C/C++ navigation icons.

RcppArmadillo

O Armadillo [Sanderson (2010)] é uma moderna biblioteca do C++ com foco em álgebra linear e operações relacionadas, que visam atingir um bom equilíbrio entre velocidade e facilidade de uso.

Pode-se fazer uso de subconjuntos de funções trigonométricas, números complexos, decomposições de matrizes, funções estatísticas, etc.

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

RcppArmadillo

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
```

O `#include<Rcpp.h>` fica implícido.

Para melhor entendimento vamos a alguns exemplos básicos.

Exemplos básicos da estrutura de dados

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Addins.
- Project Pane:** Project: (None).
- Code Editor:** Shows R script code. A tooltip is displayed over line 6: "Run the current line or selection (Ctrl+Enter)".

```
1 rm(list=ls())
2 #install.packages("RcppArmadillo")
3 rm(..., list = character(), pos = -1, envir = as.environment(pos), inherits
4 library("Rcpp")
5 wd = getwd()
6 sourceCpp(paste(wd, "/armaexemplos.cpp", sep=""))
```
- Console:** Shows the command: > sourceCpp(paste(wd, "/armaexemplos.cpp", sep=""))
- Environment pane:** Shows the global environment and values. The 'values' section includes:
 - wd: "C:/us..."
- Right-hand sidebar:** Files, Plots, Package, Install, Update, User Library.

Exemplos básicos da estrutura de dados

A entrada é uma matriz.

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with a "Go to file/function" search bar.
- Project Explorer:** Shows three files: primeirasfuncoes.cpp, SERarmadillo.R, and armaexemplos.cpp.
- Code Editor:** Displays the contents of the primeirasfuncoes.cpp file, which includes C++ code for interfacing with Rcpp and the Armadillo library.
- Environment Tab:** Shows the current working directory (wd) as "C:/Users/erico...".
- Global Environment:** Lists global variables and functions.
- Values Tab:** Lists variable values.
- Functions Tab:** Lists user-defined functions.
- Console:** A text input field starting with '>'.
- File Menu:** Files, Plots, Packages, Help.
- Install and Update Buttons:** Buttons for managing packages.
- User Library:** Shows an installed package named "inl..." with version 0.3.

Exemplos básicos da estrutura de dados

A saída é uma matriz.

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** primeirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, armaexemplos.cpp.
- Code Editor:** Shows C++ code for a function `exemplo1` that returns a diagonal matrix.
- Console:** Displays the output of running the function `exemplo1(diag(3))`, which prints a 3x3 identity matrix.
- Environment:** Shows the global environment with a variable `values` set to "C:/US...".
- Functions:** Shows several user-defined functions named `ex... func...`.
- Files, Plots, Package:** Standard RStudio navigation tabs.

```
13 //[[Rcpp::export()]]
14 arma::mat exemplo1(arma::mat x){
15     return(x);
16 }
17
18
19
```

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

Exemplos básicos da estrutura de dados

A entrada é um vetor.

The screenshot shows the RStudio interface with the following components:

- Top Bar:** RStudio, File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Source on Save buttons.
- Project Bar:** Shows files like eirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, and armaexemplos.cpp, along with a Source button.
- Environment Tab:** Shows Global Environment, values (with wd set to "C:/..."), and Functions (listing several functions).
- Console Tab:** Shows the command > |.
- Code Editor:** Displays the following C++ code:

```
20 //[[Rcpp::export()]]  
21 arma::vec exemplo2(arma::vec x){  
22     return(x);  
23 }  
24
```
- Right Sidebar:** Includes tabs for Files, Plots, and Package, and buttons for Install and Update.

Exemplos básicos da estrutura de dados

A saída é uma matriz.

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with a "Go to file/function" search bar.
- Project Bar:** Shows files like "primeirasfuncoes.R", "primeirasfuncoes.cpp", "SERarmadillo.R", and "armaexemplos.cpp".
- Code Editor:** Displays the following C++ code:

```
20 // [[Rcpp::export()]]
21 arma::vec exemplo2(arma::vec x){
22     return(x);
23 }
24
```
- Console:** Shows the command `> exemplo2(c(2,4,6,8))` and its output:

```
[1] 2
[2] 4
[3] 6
[4] 8
> |
```
- Environment Tab:** Shows the global environment with variables like `values` and `Functions`.
- Files Tab:** Shows options for installing and updating packages.
- User Library:** Shows the current user library.

Exemplos básicos da estrutura de dados

Passando da classe Rcpp::NumericMatrix para arma::mat

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Source on Save.
- Project Bar:** Shows files primeirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, and armaexemplos.cpp.
- Code Editor:** Displays C++ code:

```
23
26 //[[Rcpp::export()]]
27 arma::mat exemplo3(NumericMatrix x){
28     arma::mat y=as<arma::mat>(x);
29     return (y);
30 }
```
- Console:** Shows the command >.
- Environment Tab:** Shows Global Environment, values (with wd set to "C:/..."), Functions (with entries ex..., func...), Files, Plots, and Package.
- Plots Tab:** Shows Install and Update buttons.
- User Library Tab:** Shows Functi... (with ii).

Exemplos básicos da estrutura de dados

Passando da classe Rcpp::NumericMatrix para arma::mat

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Run, along with a "Go to file/function" search bar.
- Project Bar:** Shows files like "primeirasfuncoes.R", "primeirasfuncoes.cpp", "SERarmadillo.R", and "armaexemplos.cpp".
- Code Editor:** Displays the following C++ code in "armaexemplos.cpp":

```
23
26 // [[Rcpp::export()]]
27 arma::mat exemplo3(NumericMatrix x){
28     arma::mat y=as<arma::mat>(x);
29     return (y);
30 }
```

- Console:** Shows the R command `exemplo3(diag(4))` being run and its output, which is a 4x4 identity matrix:

```
Console ~/
> exemplo3(diag(4)) #transforma de NumericMatrix para arma
[1,] [2,] [3,] [4,]
[1,] 1 0 0 0
[2,] 0 1 0 0
[3,] 0 0 1 0
[4,] 0 0 0 1
>
```

- Environment Tab:** Shows the global environment with variables like `values` and `Functions`.
- Files Tab:** Shows options for Install and Update.
- User Library:** Shows the current user library.

Exemplos básicos da estrutura de dados

Passando da classe arma::mat para Rcpp::NumericMatrix

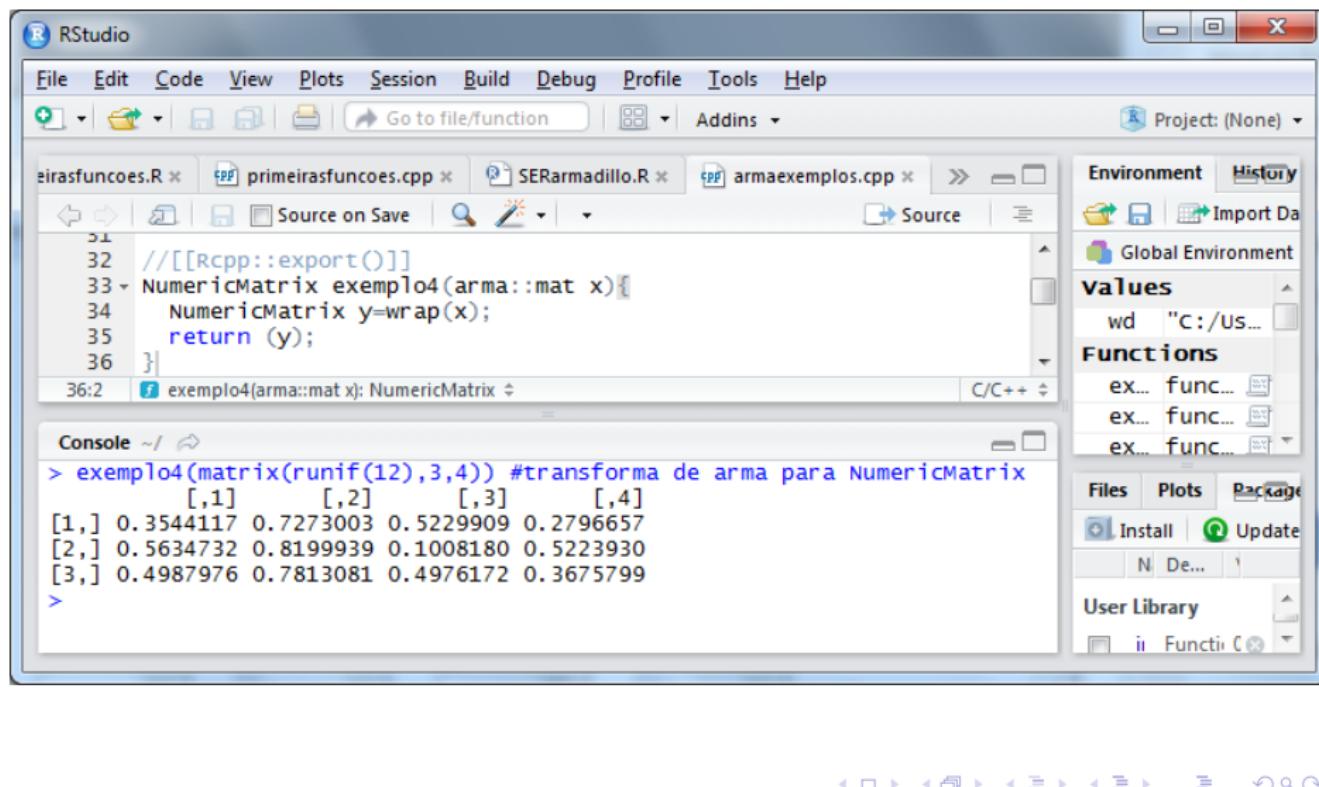
The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Source on Save.
- Project Bar:** Shows files eirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, and armaexemplos.cpp.
- Code Editor:** Displays the following C++ code:

```
31 // [[Rcpp::export()]]
32 NumericMatrix exemplo4(arma::mat x){
33     NumericMatrix y=wrap(x);
34     return (y);
35 }
```
- Console:** Shows the command >.
- Environment Tab:** Shows the Global Environment, values (wd: "C:/..."), and Functions (multiple entries for ex... func...).
- Files Tab:** Shows Install and Update buttons.
- User Library:** Shows the User Library pane.

Exemplos básicos da estrutura de dados

Passando da classe arma::mat para Rcpp::NumericMatrix



RStudio interface showing code editing, project management, and a console output.

Code Editor: Shows a C++ file named `armaexemplos.cpp` with the following content:

```
31 // [[Rcpp::export()]]
32 NumericMatrix exemplo4(arma::mat x){
33     NumericMatrix y=wrap(x);
34     return (y);
35 }
36 
```

The code defines a function `exemplo4` that takes an `arma::mat` and returns a `NumericMatrix`.

Console Output: Shows the execution of the function and its output:

```
> exemplo4(matrix(runif(12),3,4)) #transforma de arma para NumericMatrix
[1,] 0.3544117 0.7273003 0.5229909 0.2796657
[2,] 0.5634732 0.8199939 0.1008180 0.5223930
[3,] 0.4987976 0.7813081 0.4976172 0.3675799
>
```

Project: (None)

Environment: Shows the global environment with variables `wd` set to "C:/Users/...".

Functions: Shows three functions: `ex...`, `func...`, `ex...`, `func...`, and `ex...`, `func...`.

Files, Plots, Package: Buttons for managing files, plots, and packages.

User Library: Buttons for installing and updating packages.

Exemplos básicos da estrutura de dados

`printf` e dimensão de dados (**linhas** e **colunas**)

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Shows files: eirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, armaexemplos.cpp.
- Code Editor:** Displays the following C++ code in primeirasfuncoes.cpp:

```
38 // [[Rcpp::export()]]
39 arma::mat exemplo5(arma::mat x){
40     int r=x.n_rows; // na classe rcpp havia .nrow();
41     int c=x.n_cols;
42     printf(" Linhas:%d \n colunas:%d \n",r,c);
43     return(x);
44 }
```
- Console:** Shows the command `>` followed by a blank line.
- Environment:** Shows a list of global variables starting with "ex... func...".
- Plots:** Shows a small preview of a plot.
- Files:** Shows a list of files in the User Library.
- Plots:** Shows a small preview of a plot.
- Package:** Shows options to Install and Update.
- User Library:** Shows a list of functions and their descriptions.

Exemplos básicos da estrutura de dados

`printf` e dimensão de dados (linhas e colunas)

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None)
- Code Editor:** Shows C++ code in primeirasfuncoes.cpp:

```
38 // [[Rcpp::export()]]
39 arma::mat exemplo5(arma::mat x){
40     int r=x.n_rows; // na classe rcpp havia .nrow();
41     int c=x.n_cols;
42     printf(" Linhas:%d \n colunas:%d \n",r,c);
43     return(x);
44 }
```
- Console:** Shows the R command and its output:

```
> exemplo5(matrix(runif(8),4,2)) #print e dimensao
Linhas:4
colunas:2
 [,1]      [,2]
[1,] 0.2036337 0.4225002
[2,] 0.1665127 0.2957195
[3,] 0.6982719 0.1156541
[4,] 0.1406851 0.4027528
> |
```
- Environment:** Shows a list of global variables.
- Plots:** Shows a small thumbnail of a plot.
- Files:** Shows a list of files.
- Plots:** Shows a list of plots.
- Package:** Shows a list of packages.
- User Library:** Shows a list of functions and their descriptions.

Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.

The screenshot shows the RStudio interface with two files open:

- primeirasfuncoes.R**: An R script containing C++ code via Rcpp. It defines a function `exemplo6` that creates three matrices (`x1`, `x2`, `x3`) and returns them in a list.
- primeirasfuncoes.cpp**: A C++ file that includes the Rcpp header and provides the implementation for the `exemplo6` function.

The RStudio environment pane shows several global functions named `exe... funct...`. The User Library pane shows a single entry: `Function 0` with sub-options: `to`, `Inline`, `C_`, `C++,`, `Fortran`, `Function`, and `Call`.

```
43
46 // [[Rcpp::export()]]
47 List exemplo6(int n, int r, double v) {
48     arma::mat x1;
49     x1.print();
50     x1.reshape(n, r);
51     x1.fill(v);
52     arma::mat x2(n, r);
53     x2 = diagmat(x1);
54     arma::vec x3 = vectorise(x2);
55
56     List retorno;
57     retorno["x1"] = x1;  retorno["x2"] = x2;  retorno["x3"] = x3;
58     return(retorno) ;
59 }
```

Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.

The screenshot shows the RStudio interface with several files open:

- `primeirasfuncoes.R`: Contains R code for generating random numbers and creating a list.
- `primeirasfuncoes.cpp`: Contains C++ code for generating random numbers and creating a list.
- `SERarmadillo.R`: Contains R code for working with Armadillo matrices.
- `armaexemplos.cpp`: Contains C++ code for working with Armadillo matrices.

The **Console** pane shows the execution of the R code:

```
> exemplo6(2,3,3/4) #lista
[matrix size: 0x0]
$x1
 [,1] [,2] [,3]
[1,] 0.75 0.75 0.75
[2,] 0.75 0.75 0.75

$x2
 [,1] [,2] [,3]
[1,] 0.75 0.00 0
[2,] 0.00 0.75 0

$x3
 [,1]
[1,] 0.75
[2,] 0.00
[3,] 0.00
[4,] 0.75
[5,] 0.00
[6,] 0.00
```

The **Environment** pane shows the global environment with many unnamed functions named `exe.. funct...`. The **Files** pane shows the user library with various function types: `to`, `Inline`, `C`, `C++`, `Fortran`, `Function`, and `Call`.

Exemplos básicos da estrutura de dados

Agora entrando com uma matriz quadrada.

The screenshot shows the RStudio interface with the following details:

- Console:** Displays the R code and its output.

```
> exemplo6(4,4,5/7) #lista
[matrix size: 0x0]
$x1
 [,1]      [,2]      [,3]      [,4]
[1,] 0.7142857 0.7142857 0.7142857 0.7142857
[2,] 0.7142857 0.7142857 0.7142857 0.7142857
[3,] 0.7142857 0.7142857 0.7142857 0.7142857
[4,] 0.7142857 0.7142857 0.7142857 0.7142857

$x2
 [,1]      [,2]      [,3]      [,4]
[1,] 0.7142857 0.0000000 0.0000000 0.0000000
[2,] 0.0000000 0.7142857 0.0000000 0.0000000
[3,] 0.0000000 0.0000000 0.7142857 0.0000000
[4,] 0.0000000 0.0000000 0.0000000 0.7142857

$x3
 [,1]
[1,] 0.7142857
[2,] 0.0000000
[3,] 0.0000000
[4,] 0.0000000
[5,] 0.0000000
[6,] 0.7142857
[7,] 0.0000000
[8,] 0.0000000
[9,] 0.0000000
[10,] 0.0000000
[11,] 0.7142857
[12,] 0.0000000
[13,] 0.0000000
[14,] 0.0000000
[15,] 0.0000000
[16,] 0.7142857
```
- Environment:** Shows the global environment with the variable `values` containing the workspace path and a list of function objects.
- Files:** Shows the user library with packages: `Functor 0`, `to`, `Inline`, `C,`, `C++,`, `Fortran`, `Functor`, and `Calls`.

Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Standard file operations like Open, Save, Print, and a Go to file/function button.
- Project:** Project: (None)
- Code Editor:** Shows two files: irsoSERprimeirasfuncoes.R and primeirasfuncoes.cpp. The R file contains a function exemplo7 that generates matrices x, y, and z. The C++ file contains a function exemplo8 that takes a matrix x and returns a double value.
- Environment Tab:** Shows the Global Environment, Values, and Functions sections. The Functions section lists multiple entries for exemplo8.
- Console Tab:** Shows the command > | indicating the current input position.
- Files Tab:** Shows the User Library with entries for Functior 0, to, Inline, C, C++, Fortran, and Functior.

Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.

The screenshot shows the RStudio interface with the following components:

- File Editor:** Displays the file `irsoSERprimeirasfuncoes.R` containing C++ code for generating random numbers and creating lists.
- Console:** Shows the output of running the function `exemplo7` with arguments `(3,4,4/5)`. It prints three matrices `X`, `Y`, and `Z` to the console.
- Environment:** Shows the global environment with various functions named `exe.. funct...` and variables `wd`.
- Files:** Shows the user library with files for `Functor` in various languages: C, C++, Fortran, and C#.

```
60
61
62 //[[Rcpp::export()]]
63 List exemplo7(int n, int m, double v){
64     arma::mat x(n,m); x.fill(v);
65     arma::colvec y( rnorm(n,0,1) );
66     arma::rowvec z(rgamma(m,2,5));
67     List retorno;
68     retorno["X"] = x; retorno["Y"] = y; retorno["Z"] = z;
69     return retorno;
70 }
71
75:2 | exemplo8(arma::mat x, int i, int j): double |
```

```
> exemplo7(3,4,4/5) #lista
$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,] -0.2149450
[2,] 0.1803565
[3,] -0.3095933

$Z
 [,1]      [,2]      [,3]      [,4]
[1,] 6.268311 15.05453 15.08566 10.06794
```

Exemplos básicos da estrutura de dados

Exemplo exibindo a inicialização dos valores.

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Editor:** Shows two files: `primeirasfuncoes.R` and `primeirasfuncoes.cpp`. The `primeirasfuncoes.cpp` file contains the following C++ code:71 // [[Rcpp::export()]]
72 arma::mat exemplo8(int n, int r){
73 arma::mat x(n, r);
74 x.print("");
75 x.ones();
76 return(x);
77 }
78
81:13 # exemplo9(arma::mat x, int i, int j): double {
- Environment:** Global Environment pane showing multiple entries labeled "exe... funct...".
- Console:** An empty command-line interface.
- Tools:** Files, Plots, Packages, Install, Update, Na Desc., User Library (with a checked checkbox).
- Bottom:** A set of standard window control buttons (minimize, maximize, close) and a navigation toolbar.

Exemplos básicos da estrutura de dados

Veja que os valores não inicializam com zeros ou uns.

The screenshot shows the RStudio interface with several files open:

- `irsoSERprimeirasfuncoes.R`: Contains R code for a function `exemplo8`.
- `primeirasfuncoes.cpp`: Contains C++ code for the same function.
- `SERarmadillo.R`: Contains R code for a function `armaexemplos`.
- `armaexemplos.cpp`: Contains C++ code for the function `armaexemplos`.

The **Console** pane shows the output of the R command `> exemplo8(3,4)`:

```
1.7553e-292 1.4697e+265 1.4693e+265 9.7793e-210
6.0866e-297 1.0870e-311 1.2955e+215 3.3491e-308
6.1161e-297 1.8349e-296 1.4697e+265 1.0388e-209
 [,1] [,2] [,3] [,4]
[1,] 1 1 1 1
[2,] 1 1 1 1
[3,] 1 1 1 1
> |
```

The **Environment** pane shows a list of global variables, all of which are functions (indicated by the `funct...` suffix). The **Files** pane shows the current project structure.

Para inicialização com zeros use `.zeros` ou `.fill` para quaisquer valores.

Exemplos básicos da estrutura de dados

Localização de elementos ou subconjuntos em matrizes.

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Addins dropdown.
- Project:** Project: (None) is selected.
- Code Editor:** Shows a C++ file named "primeirasfuncoes.cpp" with the following code:

```
79
80 //[[Rcpp::export()]]
81 double exemplo9(arma::mat x, int i, int j){
82     return(x(i,j));
83 }
84
85 //[[Rcpp::export()]]
86 arma::mat exemplo10(arma::mat x, int i){
87     return(x.row(i));
88 }
89
90 //[[Rcpp::export()]]
91 arma::mat exemplo11(arma::mat x, int j){
92     return(x.col(j));
93 }
94
```

- Environment Tab:** Shows the Global Environment with Data, Values, and Functions sections. The Functions section lists 12 entries named "exe... funct...".
- Files Tab:** Shows Install and Update buttons, and a User Library section with a checked checkbox for "Function 0".
- Plots Tab:** Shows a "Na Desc..." message.
- Console Tab:** Shows the command "> |" entered.

Exemplos básicos da estrutura de dados

Localização de elementos ou subconjuntos em matrizes.

The screenshot shows the RStudio interface with several windows open:

- Code Editor:** Shows R code for matrix operations. The code includes three Rcpp functions: exemplo9, exemplo10, and exemplo11, which export C++ functions to R. The Rcpp code uses the `arma::mat` library to manipulate matrices.
- Console:** Displays the output of the R code. It shows the creation of matrix `z`, and the execution of `exemplo9(z,1,2)` and `exemplo11(z,2)` to extract elements and rows respectively.
- Environment:** Shows the global environment with variables `z` and `wd`.
- Data View:** Shows the contents of matrix `z` as a 2x3 numeric matrix.
- Files:** Shows the project files: irsoSERprimeirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, and armaexemplos.cpp.
- Plots:** Shows a small histogram.
- Packages:** Shows available packages: Na Desc... (None).
- User Library:** Shows a function named "Functor 0" with code snippets for C, C++, Fortran, and R.

```
File Edit Code View Plots Session Build Debug Profile Tools Help
+ Go to file/function Addins Project: (None)
Source on Save Source
79
80 //[[Rcpp::export()]]
81 double exemplo9(arma::mat x, int i, int j){
82     return(x(i,j));
83 }
84
85 //[[Rcpp::export()]]
86 arma::mat exemplo10(arma::mat x, int i){
87     return(x.row(i));
88 }
89
90 //[[Rcpp::export()]]
91 arma::mat exemplo11(arma::mat x, int j){
92     return(x.col(j));
93 }
94
94:1 (Top Level) ▾
C/C++ ▾
> z
[,1]      [,2]      [,3]
[1,] -0.08458607 -0.4634828  0.7360404
[2,]  0.84040013 -0.5508350 -0.1078814
> exemplo9(z,1,2) #elemento (i+1,j+1) da matriz z
[1] -0.1078814
> exemplo10(z,1)  #linha (i+1) da matriz
[,1]      [,2]      [,3]
[1,]  0.8404001 -0.550835 -0.1078814
> exemplo11(z,2)  #coluna (i+1) da matriz
[,1]
[1,]  0.7360404
[2,] -0.1078814
> |
```

Exemplos básicos da estrutura de dados

Operações com matrizes

The screenshot shows the RStudio interface with several tabs open in the top-left pane: 'irsoSERprimeirasfuncoes.R', 'primeirasfuncoes.cpp', 'SERarmadillo.R', and 'armaexemplos.cpp'. The 'primeirasfuncoes.cpp' tab is active, displaying the following C++ code:

```
94 // [[Rcpp::export()]]
95 arma::mat exemplo12(arma::mat x){
96     return (x+x);
97 }
98 }

99 // [[Rcpp::export()]]
100 arma::mat exemplo13(arma::mat x){
101    return (x-x);
102 }
103 }

104 // [[Rcpp::export()]]
105 arma::mat exemplo14(arma::mat x){
106    return (x%>x);
107 }
108 }

109 // [[Rcpp::export()]]
110 arma::mat exemplo15(arma::mat x){
111    return (exp(x));
112 }
113 }
```

The right-hand pane shows the 'Data' view with a 'Values' section containing a single entry: 'wd "C:/User..."'.

The bottom-left pane is the 'Console' window, which is currently empty.

Exemplos básicos da estrutura de dados

Operações com matrizes

The screenshot shows the RStudio interface with several files open:

- `irsoSERprimeirasfuncoes.R`: Contains R code for matrix operations.
- `primeirasfuncoes.cpp`: Contains C++ code for matrix operations.
- `SERarmadillo.R`: Contains R code for matrix operations using the Armadillo library.
- `armaexemplos.cpp`: Contains C++ code for matrix operations using the Armadillo library.

The `primeirasfuncoes.cpp` file contains the following code:

```
94 // [[Rcpp::export()]]
95 arma::mat exemplo12(arma::mat x){
96     return (x+x);
97 }
98 }
99
100 // [[Rcpp::export()]]
101 arma::mat exemplo13(arma::mat x){
102     return (x-x);
103 }
104
105 // [[Rcpp::export()]]
106 arma::mat exemplo14(arma::mat x){
107     return (x%>x);
108 }
109
110 // [[Rcpp::export()]]
111 arma::mat exemplo15(arma::mat x){
112     return (exp(x));
113 }
```

The `Console` pane shows the results of running the R code:

```
> exemplo12(z) #realiza a soma z+z
      [,1]      [,2]      [,3]
[1,] -0.1691721 -0.9269655  1.4720809
[2,]  1.6808003 -1.1016700 -0.2157628
> exemplo13(z) #realiza a diferenca z-z
      [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
> exemplo14(z) #multiplica elemento por elemento de z
      [,1]      [,2]      [,3]
[1,] 0.007154803 0.2148163  0.5417555
[2,] 0.706272371 0.3034192  0.0116384
> exemplo15(z) #calcula exponential de z
      [,1]      [,2]      [,3]
[1,] 0.9188926  0.6290889  2.0876529
[2,] 2.3172940  0.5764683  0.8977341
> |
```

The `Data` pane shows the variable `z` defined in the workspace.

Exemplos básicos da estrutura de dados

The screenshot shows the RStudio interface with several files open in the left pane:

- irsoSERprimeirasfuncoes.R
- primeirasfuncoes.cpp
- SERArmadillo.R
- armaexemplos.cpp

The code in primeirasfuncoes.cpp contains the following C++ code:

```
115 //[[Rcpp::export()]]  
116 arma::mat exemplo16(arma::mat x){  
117     return (x.t());  
118 }  
119  
120 //[[Rcpp::export()]]  
121 arma::mat exemplo17(arma::mat x){  
122     return (x.t() * x);  
123 }  
124  
125 //[[Rcpp::export()]]  
126 arma::mat exemplo18(arma::mat x){  
127     return ( (x.t() * x).i() );  
128 }  
129  
130 exemplo15(arma::mat x); arma::mat
```

The right pane displays the RStudio environment, showing the following sections:

- Environment**: Shows a list of objects: z, num, ..., Values.
- Data**: Shows a list of objects: wd, "C:/User...".
- Functions**: Shows a list of functions: exe.. funct..., repeated multiple times.
- Files**, **Plots**, **Packages**: Standard RStudio tabs.
- User Library**: A dropdown menu with options: Function, to, Inline, C, C++, Fortran, Function.

The bottom right corner of the interface includes standard window control buttons (minimize, maximize, close) and a search bar.

Exemplos básicos da estrutura de dados

The screenshot shows the RStudio interface with several tabs open:

- `irsoSERprimeirasfuncoes.R`
- `primeirasfuncoes.cpp`
- `SERarmadillo.R`
- `armaexemplos.cpp`

The `primeirasfuncoes.cpp` tab contains the following C++ code:

```
115 //[[Rcpp::export()]]
116 arma::mat exemplo16(arma::mat x){
117     return (x.t());
118 }
119
120 //[[Rcpp::export()]]
121 arma::mat exemplo17(arma::mat x){
122     return (x.t()*x);
123 }
124
125 //[[Rcpp::export()]]
126 arma::mat exemplo18(arma::mat x){
127     return ( (x.t()*x).i() );
128 }
129
130 exemplo15(arma::mat x); arma::mat
```

The `Console` pane shows the results of running these functions in R:

```
> exemplo16(z) #calcula a transposta de z
      [,1]      [,2]
[1,] -0.08458607  0.8404001
[2,] -0.46348277 -0.5508350
[3,]  0.73604043 -0.1078814
> exemplo17(z) #multiplicacao
      [,1]      [,2]      [,3]
[1,]  0.7134272 -0.4237176 -0.1529223
[2,] -0.4237176  0.5182355 -0.2817172
[3,] -0.1529223 -0.2817172  0.5533939
> exemplo18(z) #matriz inversa da multiplicacao no exemplo 17
      [,1]      [,2]      [,3]
[1,] 5.655642e+15 7.568083e+15 5.415551e+15
[2,] 7.568083e+15 1.012721e+16 7.246806e+15
[3,] 5.415551e+15 7.246806e+15 5.185652e+15
```

The `Environment` pane shows the following variables:

- `z` (num [3, 2])
- `wd` ("C:/User...")

The `Data` pane shows the variable `z` with its values:

Value	1	2	3
[1,1]	-0.08458607	0.8404001	
[2,1]	-0.46348277	-0.5508350	
[3,1]	0.73604043	-0.1078814	

The `Functions` pane lists many entries starting with `exe_ funct...`.

The `Files`, `Plots`, and `Packages` panes are also visible.

Exemplos básicos da estrutura de dados

Geração de matriz aleatória e mensagens de erro.

The screenshot shows the RStudio interface with several files open in the left pane:

- MinicursoSERprimeirasfuncoes.R
- primeirasfuncoes.cpp
- SERarmadillo.R
- armaexemplos.cpp

The code editor displays the following C++ code:

```
129 // [[Rcpp::export()]]
130 arma::mat exemplo19(int i){
131     arma::mat A(2,3);
132     arma::mat B(i,1);
133     A.randu(); A.print();
134     B.randu(); B.print();
135     return (A*B);
136 }
137
138
139
140 }
```

The right pane shows the Project browser with files like "Import Data", "Global Environment", "Data", "Values", "Functions", and "User Library". The "Functions" section lists many entries starting with "exe... funct...".

The bottom-left pane shows the Console window with the command prompt > |.

Exemplos básicos da estrutura de dados

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Addins dropdown.
- Project Bar:** MinicursoSERprimeirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, armaexemplos.cpp.
- Code Editor:** Shows C++ code for matrix multiplication. The code defines a function `exemplo19` that takes an integer `i` and returns the product of two matrices, A and B. Matrix A is 2x3 and matrix B is i x 1.
- Console:** Displays R command-line output. It shows the result of `exemplo19(3)` which has dimensions 2x3 and 2x1, resulting in a 2x1 vector. It also shows the result of `exemplo19(2)` which has dimensions 2x3 and 2x1, resulting in a 2x1 vector.
- Environment:** Shows the global environment with variables `z` and `values` and functions `exe...`.
- Data:** Shows data frames `z` and `values` with their respective dimensions.
- Functions:** Shows a list of 16 function definitions named `exe...` followed by a function name.
- Files:** Shows packages installed: Functor 0, to, Inline, C, C++, F.
- Plots:** Shows no plots.
- Packages:** Shows no packages.
- User Library:** Shows the user library with the checked item "Functor 0".

```
129
130 // [[Rcpp::export()]]
131 arma::mat exemplo19(int i){
132     arma::mat A(2,3);
133     arma::mat B(i,1);
134     A.randu(); A.print();
135     B.randu(); B.print();
136     return (A*B);
137 }
138
139
140
140:1 (Top Level) ◊
```

```
> exemplo19(3) # nao haverá mensagem de erro (dimensão ok!)
  0.3702   0.0673   0.3385
  0.0377   0.5139   0.3105
  0.1258
  0.7283
  0.0046
 [,1]
[1,] 0.0971648
[2,] 0.3804400
> exemplo19(2) # aqui haverá mensagens de erro (dimensão não ok)
  0.3578   0.5667   0.7145
  0.4936   0.2118   0.5584
  0.4381
  0.1786

error: matrix multiplication: incompatible matrix dimensions: 2x3 and 2x1
Error in exemplo19(2) :
  matrix multiplication: incompatible matrix dimensions: 2x3 and 2x1
```

Exemplos básicos da estrutura de dados

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, and Print, along with Go to file/function and Addins dropdown.
- Project Bar:** MinicursoSERprimeirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, armaexemplos.cpp.
- Code Editor:** Shows C++ code for matrix multiplication. The function `exemplo19` takes an integer `i` and returns the product of two matrices `A` and `B` of size `i` by 3. The code uses the Armadillo library.
- Console:** Displays R command-line output. It shows the result of `exemplo19(3)` which is a 3x3 matrix, and the result of `exemplo19(2)` which is a 2x3 matrix. Both results are highlighted with a green box.
- Environment:** Shows the global environment with variables `z` and `values` and functions `exe...` listed.
- Data:** Shows the contents of `z` as a numeric vector and the current working directory as "C:/User...".
- Functions:** A list of 19 functions named `exe...` followed by a short description.
- Plots:** No plots are present.
- Packages:** No packages are present.
- Install:** No packages are available for installation.
- User Library:** Shows a checked checkbox for "Functor 0" and a list of supported languages: to, Inline, C, C++, F, and R.

```
129
130 // [[Rcpp::export()]]
131 arma::mat exemplo19(int i){
132     arma::mat A(2,3);
133     arma::mat B(i,1);
134     A.randu(); A.print();
135     B.randu(); B.print();
136     return (A*B);
137 }
138
139
140
140:1 (Top Level) ◊
```

```
Console ~/ ↵
> exemplo19(3) # nao haverá mensagem de erro (dimensão ok!)
 0.3702  0.0673  0.3385
 0.0377  0.5139  0.3105
 0.1258
 0.7283
 0.0046
 [,1]
[1,] 0.0971648
[2,] 0.3804400
> exemplo19(2) # aqui haverá mensagens de erro (dimensão não ok)
 0.3578  0.5667  0.7145
 0.4936  0.2118  0.5584
 0.4381
 0.1786

error: matrix multiplication: incompatible matrix dimensions: 2x3 and 2x1
Error in exemplo19(2) :
  matrix multiplication: incompatible matrix dimensions: 2x3 and 2x1
```

Exemplo de um ajuste linear

The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Source on Save, Source.
- Project:** Project: (None).
- Code Editor:** The main pane displays the following C++ code (functions.cpp):

```
140
141 // [[Rcpp::export]]
142 List ajustaML(arma::vec y, arma::mat x){
143     int n = x.n_rows;
144     int k = x.n_cols;
145     arma::colvec beta = arma::solve(x, y); // estima os betas
146
147     // arma::colvec beta = (x.t()*x).i() * x.t()*y ;
148
149     arma::colvec erro = y - x*beta;          // calcula os resíduos
150     double sig2 = arma::as_scalar(erro.t()*erro / (n-k) );
151
152     arma::colvec sdbeta = sqrt( sig2*diagvec( (x.t()*x).i() ) );
153
154     return( List::create(Named("coeficientes", beta),
155                         Named("desvio padrao beta", sdbeta)
156                     ) );
157 }
158
159
160
161 }
```

- Environment Tab:** Shows the current working directory (wd) as "C:/Users/eric675...".
- Functions Tab:** Lists numerous functions named "exempl_ function" followed by various parameters (x, i, j, etc.).
- Console Tab:** Displays the command >.
- Files Tab:** Shows packages installed: inline Functions to Inline 0.3.0.
- Plots Tab:** No plots are present.
- Packages Tab:** No packages are present.
- Help Tab:** No help is present.
- Viewer Tab:** No viewer output is present.

Exemplo de um ajuste linear

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Print, and Run, along with Go to file/function, Addins, and Project dropdown.
- Code Editor:** Displays the following R script:

```
34  
35  
36 # ajuste de um modelo de regressão linear  
37  
38 n=1000  
39 x1=rnorm(n,0,sqrt(2))  
40 x2=rt(n,7)  
41 b0=0.1  
42 b1=0.8  
43 b2=-0.65  
44 sigma2=1.5  
45 erro=rnorm(n,0,sqrt(sigma2))  
46 y=b0+b1*x1+b2*x2+erro  
47 y=as.matrix(y)  
48 x=as.matrix( cbind(rep(1,n),x1,x2) )  
49  
50 ajustaML(y,x)  
51  
52  
53  
54  
55
```
- Console:** Shows the command >.
- Environment Tab:** Shows the current working directory as "C:/Users/eric675..." and lists several user-defined functions under "Functions".
- Files Tab:** Shows the User Library with an entry for "inline Functions to Inline 0.3.0".

Exemplo de um ajuste linear

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Run, Source.
- Project:** (None)
- Code Editor:** The script file contains R code for generating data and fitting a linear model. The code is as follows:

```
34
35
36 # ajuste de um modelo de regressão linear
37
38 n=1000
39 x1=rnorm(n,0,sqrt(2))
40 x2=rt(n,7)
41 b0=0.1
42 b1=0.8
43 b2=-0.65
44 sigma2=1.5
45 erro=rnorm(n,0,sqrt(sigma2))
46 y=b0+b1*x1+b2*x2+erro
47 y=as.matrix(y)
48 x=as.matrix( cbind(rep(1,n),x1,x2) )
49
50 ajustaML(y,x)
51
52
53
54
55
```

- Environment Tab:** Shows variables and their values:

Variable	Type	Value
x	num	[1:1000, 1:...]
y	num	[1:1000, 1:...]
b0	0.1	
b1	0.8	
b2	-0.65	
erro	num	[1:1000] 1.33...
n	1000	
sigma2	1.5	
wd	"C:/Users/eric675...	
x1	num	[1:1000] 1.43...
x2	num	[1:1000] -0.3...

- Functions Tab:** Shows defined functions:

 - ajustaML
 - exempl...
 - exempl...

- Console:** Displays the results of the R code execution:

```
$coeficientes
[,1]
[1,] 0.05317435
[2,] 0.78792382
[3,] -0.70038942

`desvio padrao beta`
[,1]
[1,] 0.03914280
[2,] 0.02783910
[3,] 0.03362227
```

Exemplos básicos da estrutura de dados

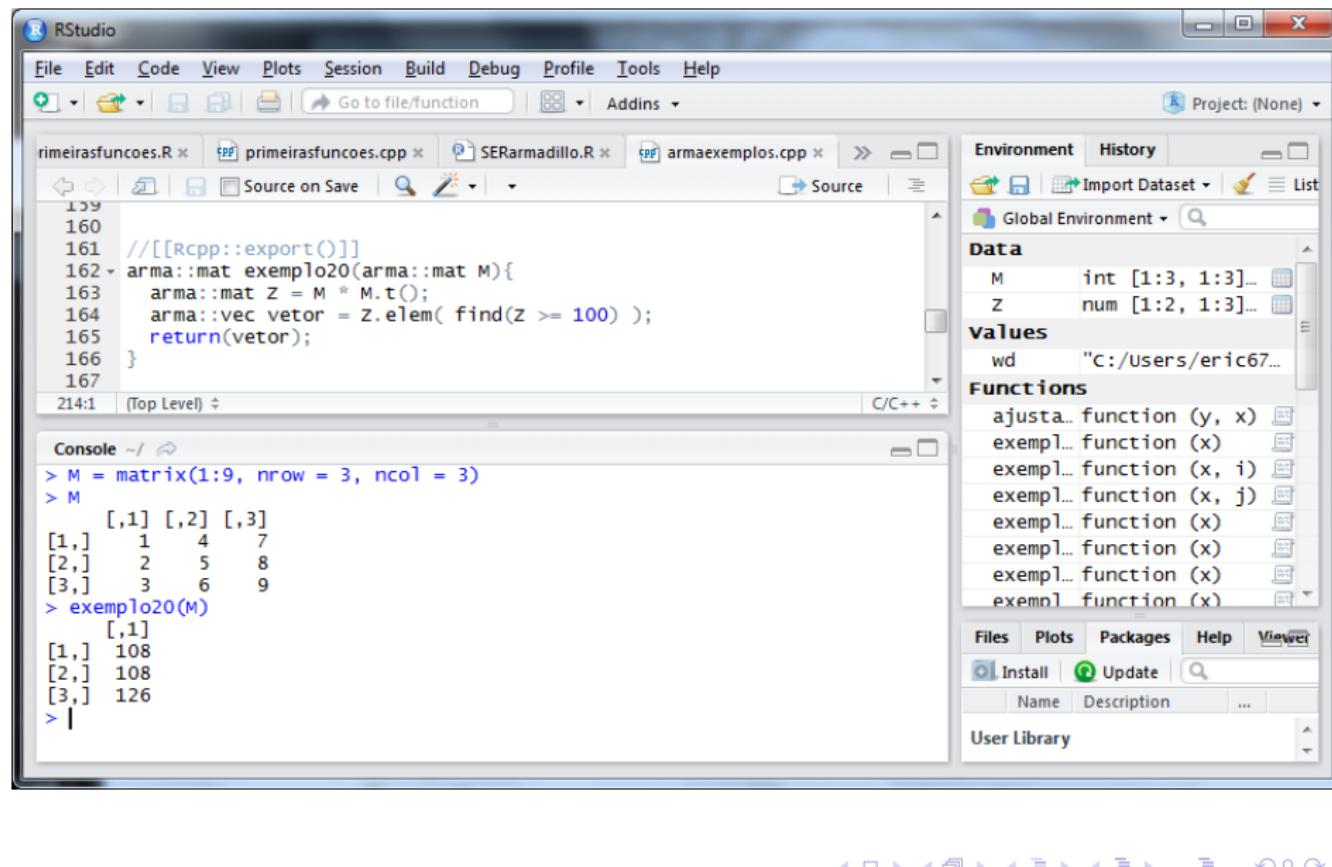
The screenshot shows the RStudio interface with the following components:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Go to file/function, Source on Save, Source.
- Project:** Project: (None)
- Code Editor:** Shows a C++ file named `primeirasfuncoes.cpp` containing the following code:

```
159
160
161 //[[Rcpp::export()]]
162 arma::mat exemplo20(arma::mat M){
163     arma::mat Z = M * M.t();
164     arma::vec vetor = Z.elem( find(Z >= 100) );
165     return(vetor);
166 }
167
```
- Console:** Shows the following R session:

```
> M = matrix(1:9, nrow = 3, ncol = 3)
> M
 [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>
```
- Environment:** Shows the Global Environment pane with variables `M` and `Z`, and the Working Directory set to `C:/Users/eric67...`.
- Data:** Shows the Data pane with variables `M` and `Z`.
- Values:** Shows the Values pane with functions `ajusta...`, `exempl...`, etc.
- Functions:** Shows the Functions pane with functions `ajusta...`, `exempl...`, etc.
- Bottom Navigation:** Files, Plots, Packages, Help, Viewer, Install, Update.

Exemplos básicos da estrutura de dados



Exemplos usando DataFrame

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Files:** primeirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, armaexemplos.cpp.
- Code Editor:** The primeirasfuncoes.cpp file contains C++ code for generating a DataFrame. The code uses Rcpp to generate four vectors (v1, v2, v3, v4) from various distributions (rpois, rgamma, rbinom, runif) and then creates a DataFrame df with these vectors as named columns.
- Environment:** Shows variables M (int [1:3, 1:3]), Z (num [1:2, 1:3]), and wd ("C:/Users/eric67...").
- Functions:** A list of functions starting with "exempl_".
- Console:** The command > | is shown at the bottom.
- Bottom Panels:** Files, Plots, Packages, Help, Viewer, Install, Update, User Library (inline).

```
168
169 //[[Rcpp::export()]]
170 DataFrame exemplo21( ){
171     arma::vec v1(rpois(15,5));
172     arma::vec v2(rgamma(15,2.4));
173     arma::vec v3(rbinom(15,7,0.6));
174     arma::vec v4(runif(15,-2,3));
175     DataFrame df = DataFrame::create( Named("Poisson",v1),
176                                         Named("Gamma", v2),
177                                         Named("Binomial", v3),
178                                         Named("Uniforme", v4)
179                                         );
180     return(df);
181 }
181:2 exemplo21():DataFrame <--
```

Exemplos usando DataFrame

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Print, and a search bar labeled "Go to file/function".
- Project Bar:** Shows "Project: (None)".
- Code Editor:** Displays Rcpp code for generating a DataFrame. The code includes imports for Rcpp and RcppArmadillo, and defines a function `exemplo21` that generates four vectors `v1` through `v4` and creates a DataFrame `df` from them.
- Console:** Shows the output of running `exemplo21()`, which prints a table with columns Poisson, Gamma, Binomial, and Uniforme, containing 15 rows of numerical values.
- Environment:** Shows variables `M` (int [1:3, 1:3]), `Z` (num [1:2, 1:3]), and `wd` (C:/Users/eric67...).
- Values:** Shows the generated DataFrame `df` with 15 rows and 4 columns.
- Functions:** Shows a list of functions defined in the workspace, including `ajusta..`, `exempl..`, and various `exempl..` functions for different parameters.
- Files, Plots, Packages, Help, Viewer:** Standard RStudio navigation tabs.
- User Library:** Shows an entry for `inline` with version 0.3.0.

Exemplos usando DataFrame

The screenshot shows the RStudio interface with the following components:

- File menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Toolbar:** Includes icons for file operations like Open, Save, Print, and a search bar labeled "Go to file/function".
- Code Editor:** Displays R code in the "primeirasfuncoes.R" file. The code defines a function `exemplo22` that generates a DataFrame with four columns: Poisson, Gamma, Binomial, and Uniforme. It uses Rcpp's `Rcpp::export` macro and the `arma` package for random number generation.
- Environment Tab:** Shows the global environment with objects M (int [1:3, 1:3]), Z (num [1:2, 1:3]), and wd (a character string pointing to a local directory).
- Data Tab:** Shows the values of objects M and Z.
- Functions Tab:** Shows a list of functions starting with `ajusta..` and ending with `exempl..`.
- Console:** An empty command-line interface where users can enter R commands.
- File menu:** Files, Plots, Packages, Help, Viewer.
- User Library:** Shows the package "inline" version 0.3.1.

Exemplos usando DataFrame

The screenshot shows the RStudio interface with the following components:

- File Bar:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project Bar:** Project: (None).
- Code Editor:** Shows R code for `exemplo22` function. The code generates a DataFrame with four columns (v1-v4) and a List `GU` containing two distributions (Gamma and Uniforme). The code uses Rcpp annotations like `[[Rcpp::export]]` and `[[Rcpp::local]].`

```
182
183 //[[Rcpp::export()]]
184 DataFrame exemplo22( ){
185     arma::vec v1(rpois(15,5));
186     arma::vec v2(rgamma(15,2,4));
187     arma::vec v3(rbinom(15,7,0.6));
188     arma::vec v4(runif(15,-2,3));
189     DataFrame df = DataFrame::create(Named("Poisson",v1),
190                                     Named("Gamma", v2),
191                                     Named("Binomial", v3),
192                                     Named("Uniforme", v4));
193
194 List GU = List::create(Named("dist.gama", df["Gamma"]),
195                         Named("dist.uniforme", df[3]));
196
197 return(GU);
198 }
```
- Environment Tab:** Shows variables `M` (int [1:3, 1:3]), `Z` (num [1:2, 1:3]), and `wd` (C:/Users/eric67...).
- Functions Tab:** Shows multiple entries for `ajusta..` and `exempl..` functions.
- Console Tab:** Displays the output of `exemplo22()` showing generated data for `dist.gama` and `dist.uniforme`.

	dist.gama	dist.uniforme
1	5.460536	2.92699471
2	7.068040	-1.17930534
3	5.404386	0.31154202
4	3.138351	1.51024870
5	20.808120	-0.84342501
6	12.137089	2.12540812
7	5.068426	1.24702312
8	15.604342	1.17699694
9	3.718003	0.70810986
10	2.391596	-0.84198939
11	14.744977	-0.69616346
12	2.109996	1.80642666
13	9.045662	0.33424372
14	27.914605	-0.13270910
15	1.872699	-0.07581043

Exemplos usando DataFrame

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Files:** eirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, armaexemplos.cpp.
- Code Editor:** Shows C++ code for creating a DataFrame. The code is as follows:

```
199
200
201 //[[Rcpp::export()]]
202 DataFrame exemplo23( ){
203     NumericVector v = {2,4,6};
204     DataFrame df = DataFrame::create( Named("vetor1",v),
205                                     Named("vetor2", clone(v) ) );
206     v = 2*v;
207     return(df);
208 }
215:2 exemplo24(CharacterVector u): DataFrame
```

- Environment:** Shows variables M and Z defined as integers and numerics respectively.
- Data:** Shows wd set to "C:/users/eric6..".
- Functions:** Shows several user-defined functions: ajusta_function, exempl..._function, exempl..._function, exempl..._function, exempl..._function, and exempl..._function.
- Console:** Shows the prompt > |
- Bottom Bar:** Includes standard window controls and a search bar.

Exemplos usando DataFrame

The screenshot shows the RStudio interface with several files open in the left pane:

- eirasfuncoes.R
- primeirasfuncoes.cpp
- SERarmadillo.R
- armaexemplos.cpp

The code editor contains C++ code for creating a DataFrame:

```
199
200
201 //[[Rcpp::export()]]
202 DataFrame exemplo23(){
203     NumericVector v = {2,4,6};
204     DataFrame df = DataFrame::create( Named("vetor1",v),
205                                         Named("vetor2", clone(v) ) );
206     v = 2*v;
207     return(df);
208 }
```

The code editor also shows a tooltip for a function call:

```
215:2 exemplo24(CharacterVector u): DataFrame
```

The right pane displays the Environment and Global Environment panes, showing variables M and Z defined as integers and numerics respectively, and the Global Environment pane listing functions like ajusta_function, exempl..._function, etc.

The Console pane shows the output of running the code:

```
> exemplo23()
vetor1  vetor2
1      4      2
2      8      4
3     12      6
>
```

Exemplos usando DataFrame

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Files:** eirasfuncoes.R, primeirasfuncoes.cpp, SERarmadillo.R, armaexemplos.cpp.
- Code Editor:** Displays C++ code for generating a DataFrame. The code uses Rcpp and the Armadillo library to create vectors w, x, and z, and then combines them into a DataFrame named exemplo24.
- Environment:** Shows the global environment with variables M and Z defined.
- Data:** Shows data frames M and Z.
- Values:** Shows the working directory wd set to "C:/users/eric6...".
- Functions:** Shows a list of functions exempl..._function(x) repeated multiple times.
- Console:** Shows the command > being typed.
- Bottom Navigation:** Includes tabs for Files, Plots, Packages, Help, and Viewer, along with Install and Update buttons.

Exemplos usando DataFrame

The screenshot shows the RStudio interface with the following components:

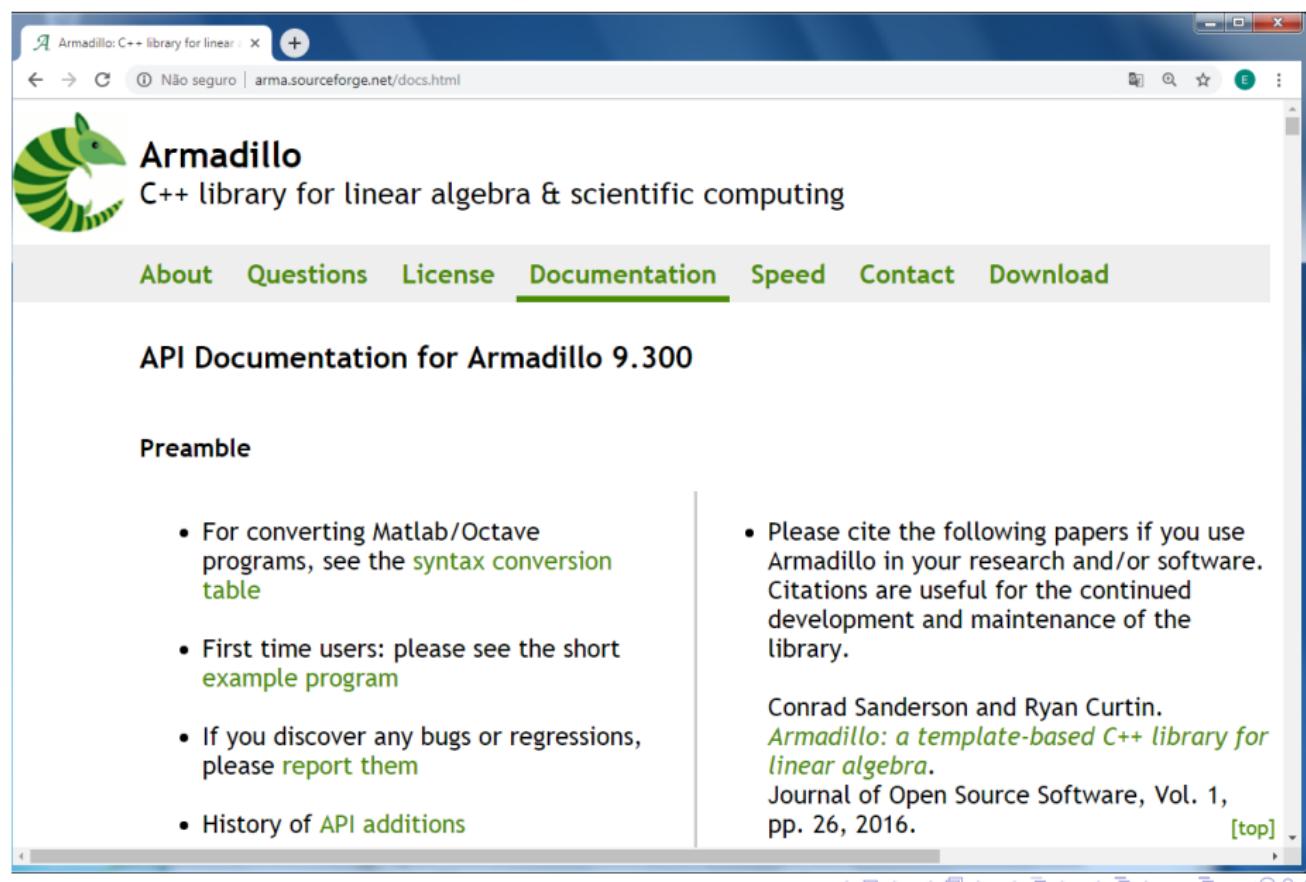
- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Code Editor:** Shows a C++ file named "armaexemplos.cpp" with Rcpp code for generating a DataFrame. The code defines a function `exemplo24` that creates a numeric vector `w` of length 5, generates a vector `x` of size `n` (from rbinom) times `w1`, and returns a DataFrame with three columns: "Times", "vetor criado", and "vetor gerado".
- Console:** Displays the output of running the function with the argument `c("Paysandu", "ceará", "Goiás", "Atlético-MG", "Chapecoense")`. The output shows the generated DataFrame with columns "Times", "vetor criado", "vetor gerado", and "conta".
- Environment:** Shows the global environment with variables `M` (int [1:3, 1:3...]), `Z` (num [1:2, 1:3...]), and `wd` ("C:/users/eric6...").
- Data:** Shows the `DataFrame` object created in the console.
- Functions:** Shows a list of functions starting with `ajusta_` and `exempl_`.
- Files:** Shows tabs for Files, Plots, Packages, Help, and Viewer.
- User Library:** Shows inline functions and a search bar.

```
210 //[[Rcpp::export()]]
211 DataFrame exemplo24(characterVector u ){
212   int n = u.length();
213   NumericVector w = NumericVector::create(1,2,3,4,5);
214   arma::vec w1 = as<arma::vec>(w);
215   arma::vec x(rbinom(n, 10, 0.7));
216   double z = arma::as_scalar( x.t() * w1 );
217   return(DataFrame::create( Named("Times",u),
218                           Named("vetor criado",w1),
219                           Named("vetor gerado", x),
220                           Named("conta",z) ) );
221
222 }
223
224 }
```

```
> exemplo24(c("Paysandu", "ceará", "Goiás", "Atlético-MG", "Chapecoense"))
      Times vetor.criado vetor.gerado conta
1    Paysandu           1           5     102
2     Ceará            2           5     102
3     Goiás            3           5     102
4 Atlético-MG          4           8     102
5 Chapecoense          5           8     102
>
```

Exercícios

Sites para aprendizagem



A screenshot of a web browser displaying the official documentation for the Armadillo C++ library. The page features a green armadillo logo, the text "Armadillo C++ library for linear algebra & scientific computing", and a navigation menu with links for About, Questions, License, Documentation (which is underlined), Speed, Contact, and Download. Below the menu, the title "API Documentation for Armadillo 9.300" is prominently displayed. The main content area contains a section titled "Preamble" followed by two columns of bulleted lists. The left column includes items about syntax conversion, example programs, reporting bugs, and API additions. The right column discusses citation requirements and provides author information: Conrad Sanderson and Ryan Curtin, along with a reference to their paper "Armadillo: a template-based C++ library for linear algebra" published in the Journal of Open Source Software, Vol. 1, pp. 26, 2016. A "[top]" link is also present.

Armadillo
C++ library for linear algebra & scientific computing

About Questions License Documentation Speed Contact Download

API Documentation for Armadillo 9.300

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](#)

- Please cite the following papers if you use Armadillo in your research and/or software. Citations are useful for the continued development and maintenance of the library.

Conrad Sanderson and Ryan Curtin.
Armadillo: a template-based C++ library for linear algebra.
Journal of Open Source Software, Vol. 1,
pp. 26, 2016.

[top]

Sites para aprendizagem

The screenshot shows a web browser window displaying the Rcpp Gallery at gallery.rcpp.org. The page title is "Rcpp Gallery" and the subtitle is "Articles and code examples for the Rcpp package". A search bar is located at the top right. Below the header, there is a tag cloud of terms related to Rcpp, including basics, dataframe, matrix, stl, armadillo, rmath, modeling, sugar, gsl, recursion, function, benchmark, rng, sparse, vector, environment, string, c++11, eigen, xts, boost, factor, bigmemory, interrupt, openmp, python, parallel, simulation, serialization, simd, finance, cube, c++14, c++17, machine_learning, mlpack, macros, data.table, nanotime, integer64, modules, gpu. The main content area is titled "Featured Articles" and lists several articles:

- Using Rcpp with C++11, C++14 and C++17** — Dirk Eddelbuettel
- Rcpp supports any C++ variant: from C++98 to C++11, C++14 and now C++17
- Parsing Dates and Times** — Dirk Eddelbuettel
- We demonstrate a new utility function built around the Boost Date_Time parsing functions.
- Sampling Importance Resampling (SIR) and social revolution.** — Jonathan Olmsted
- We use SIR to characterize the posterior distribution of parameters associated with the probability of social revolution.
- Implementing an EM Algorithm for Probit Regressions** — Jonathan Olmsted
- We illustrate the development process of creating code to estimate the parameters of a Probit regression model using the EM algorithm sequentially and in parallel.
- Parallel Distance Matrix Calculation with RcppParallel** — JJ Allaire and Jim Bullard
- Demonstrates computing an n x n distance matrix from an n x p data matrix.
- Summing a Vector in Parallel with RcppParallel** — JJ Allaire
- Demonstrates computing the sum of a vector in parallel using the RcppParallel package.
- Transforming a Matrix in Parallel using RcppParallel** — JJ Allaire
- Demonstrates transforming a matrix in parallel using the RcppParallel package.
- Call Python from R through Rcpp** — Wush Wu
- Integrate Python into R via Rcpp and Boost.Python
- Faster Multivariate Normal densities with RcppArmadillo and OpenMP** — Nino Hardt, Dicko Ahmadou
- Fast implementation of Multivariate Normal density using RcppArmadillo and OpenMP.
- Quick conversion of a list of lists into a data frame** — John Merrill

Obrigado!

Referências I

- Eddelbuettel, D., François, R. (2011). Rcpp:Seamless R and C++ Integration, Journal of Statistical Software. URL <http://www.jstatsoft.org/v40/i08>
- 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., Eddelbuettel, 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