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

Autores: Erick Amorim e Rumenick Pereira da Silva[1]

[1]rumenickbf@hotmail.com

Seminário Internacional de Estatística com R
Universidade Federal Fluminese

21 de Maio de 2019

SUMÁRIO

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 1ª 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**, ocutando 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 `NumericVertor`.

Em seguida o **Rcpp**, ocutando 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 `NumericVertor`.

Em seguida o **Rcpp**, ocutando 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**

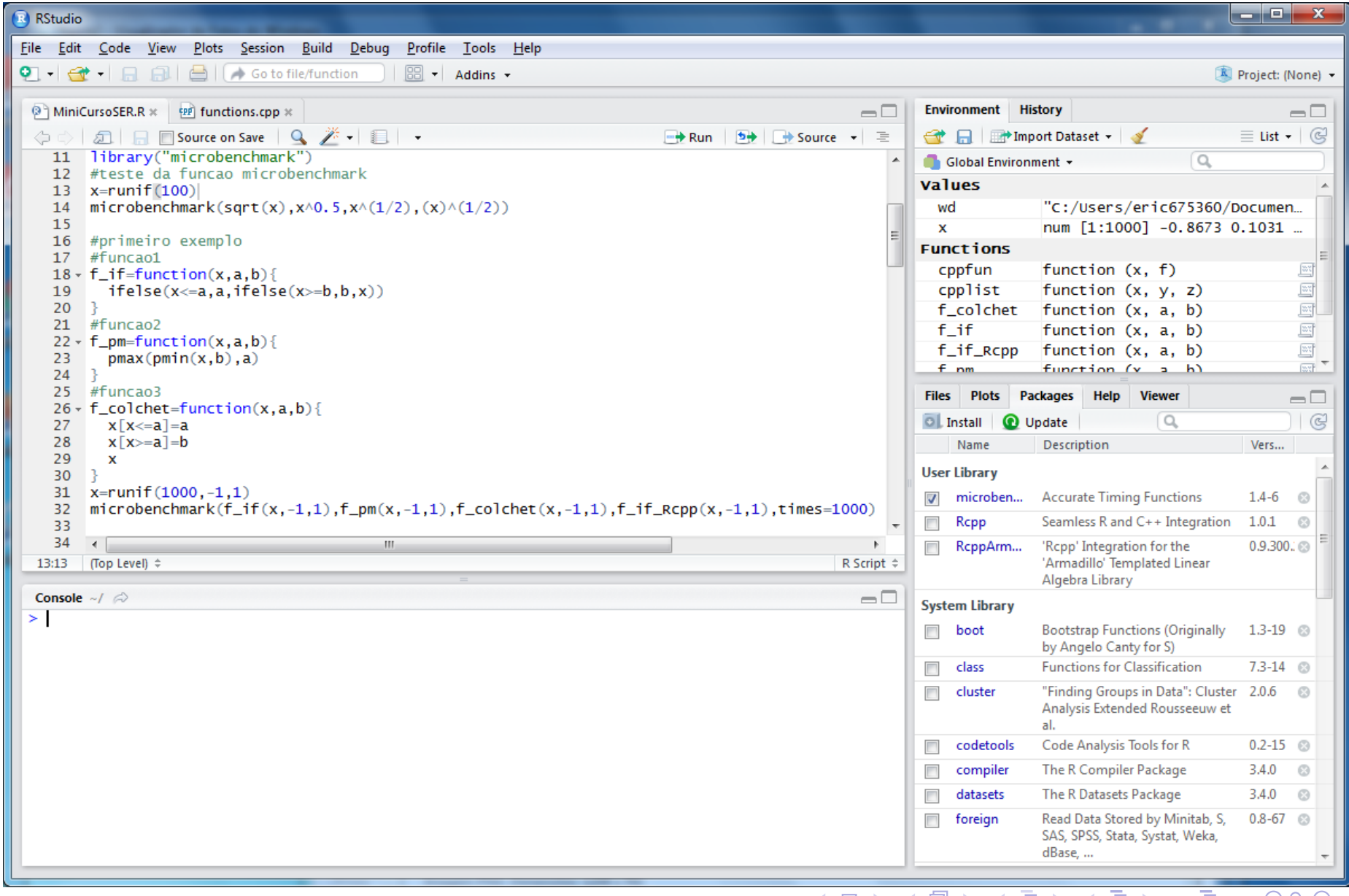
O que simplesmente acontece é que vamos definir uma simples função em `C++` usando níveis de classes (tipos de objetos) como `NumericVertor`.

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

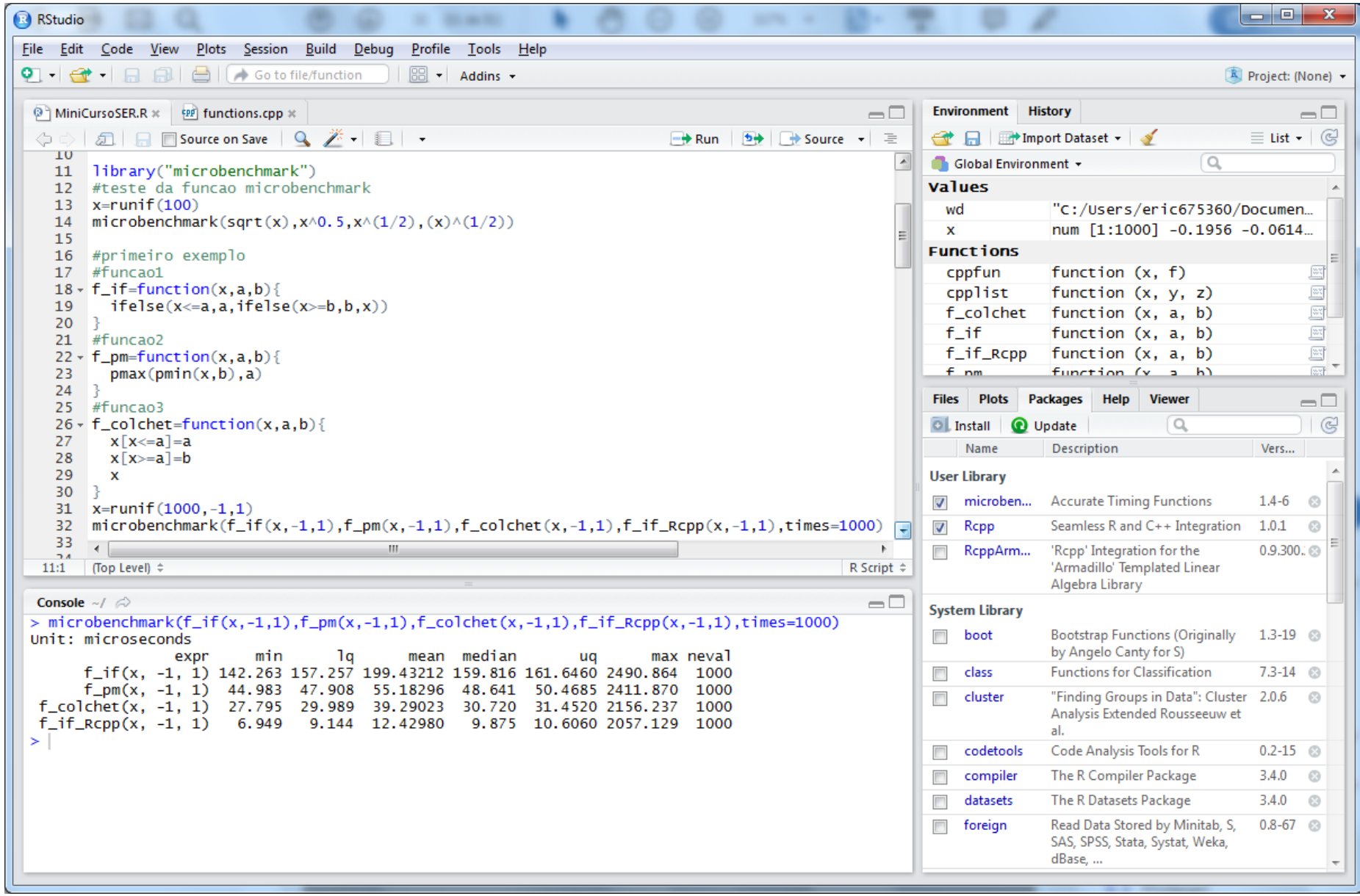
O pacote basicamente mapeia os objetos do `R` para objetos do `C++` na maioria das vezes de forma automática.

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:

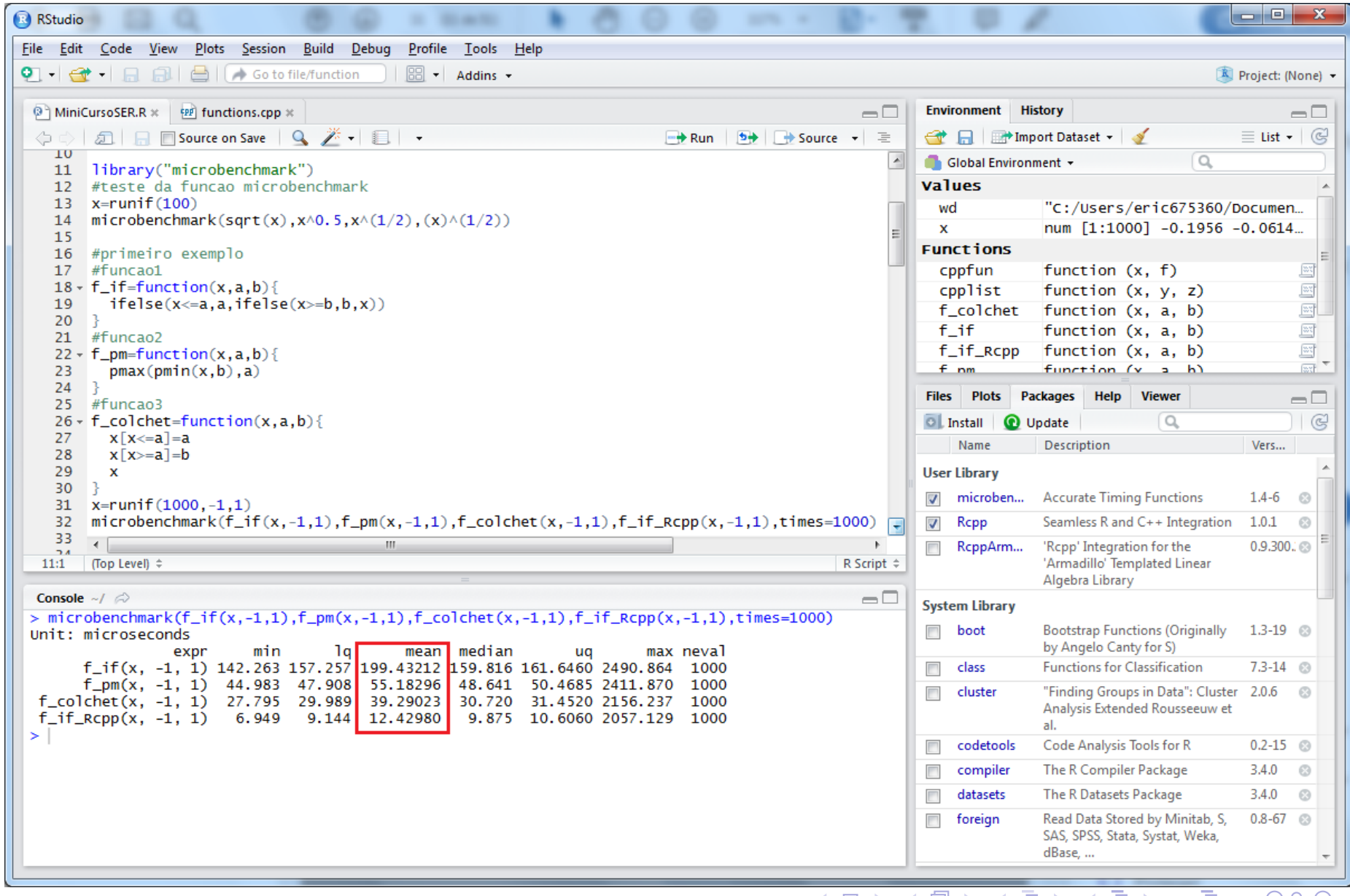
Primeiro Exemplo



Primeiro Exemplo (Continuação)

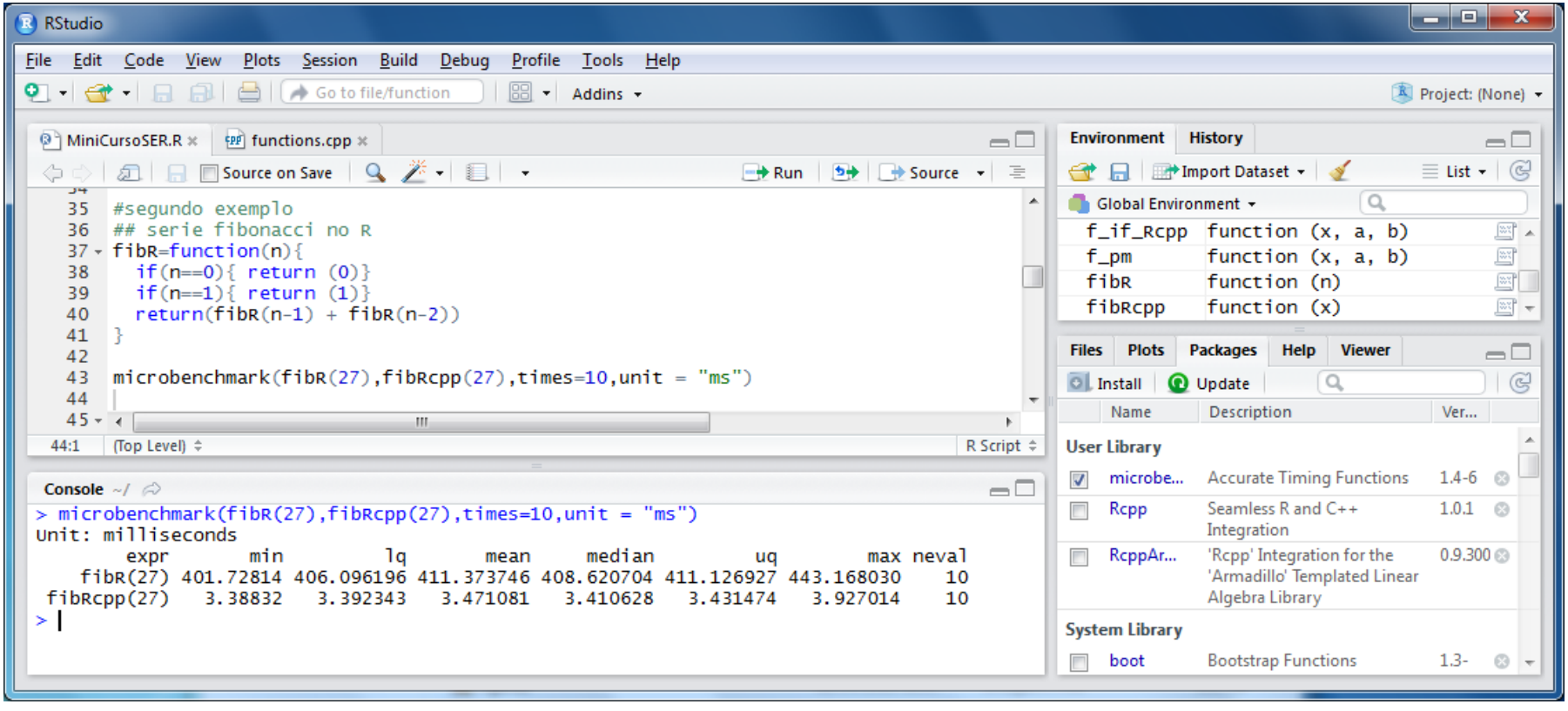


Primeiro Exemplo (Continuação)



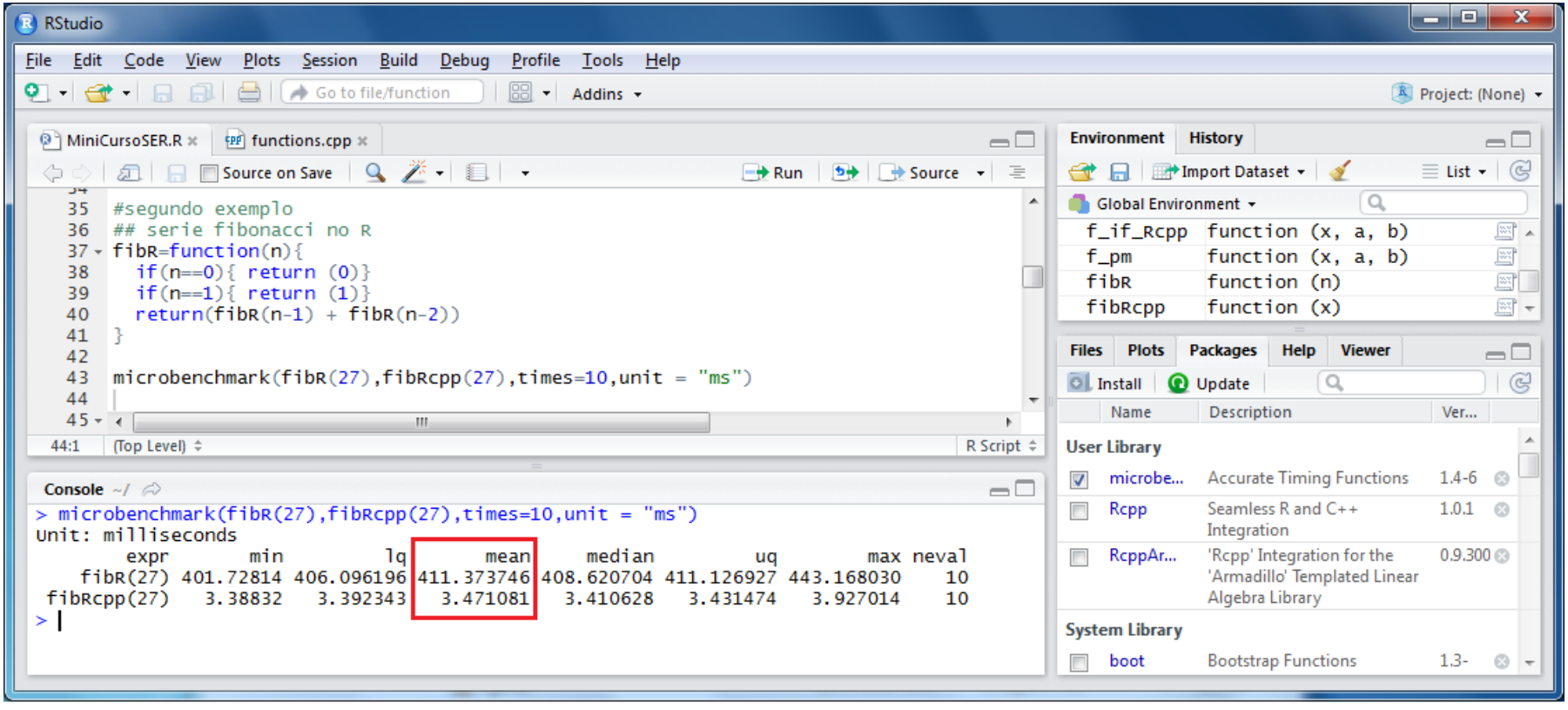
Segundo Exemplo

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

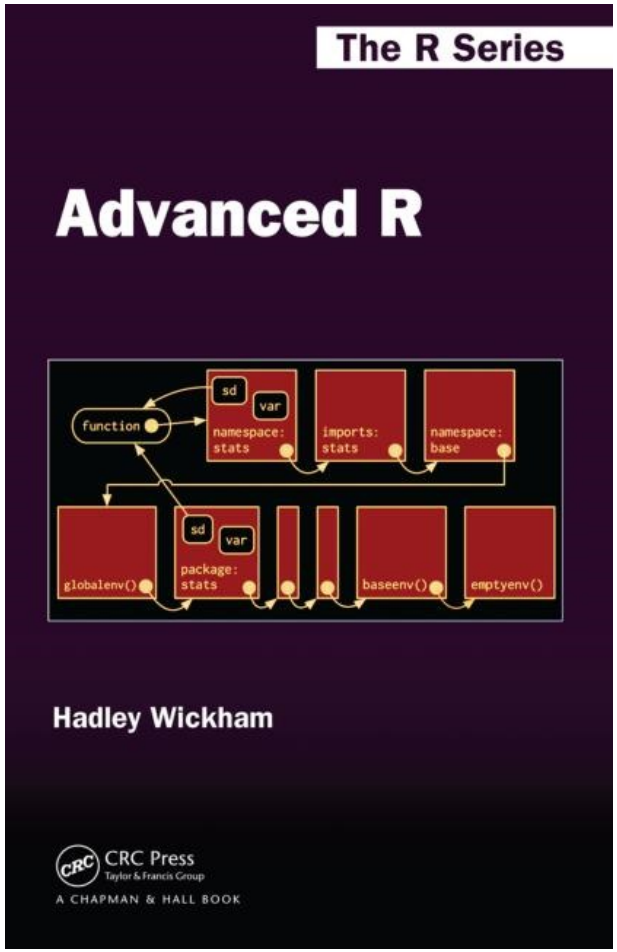


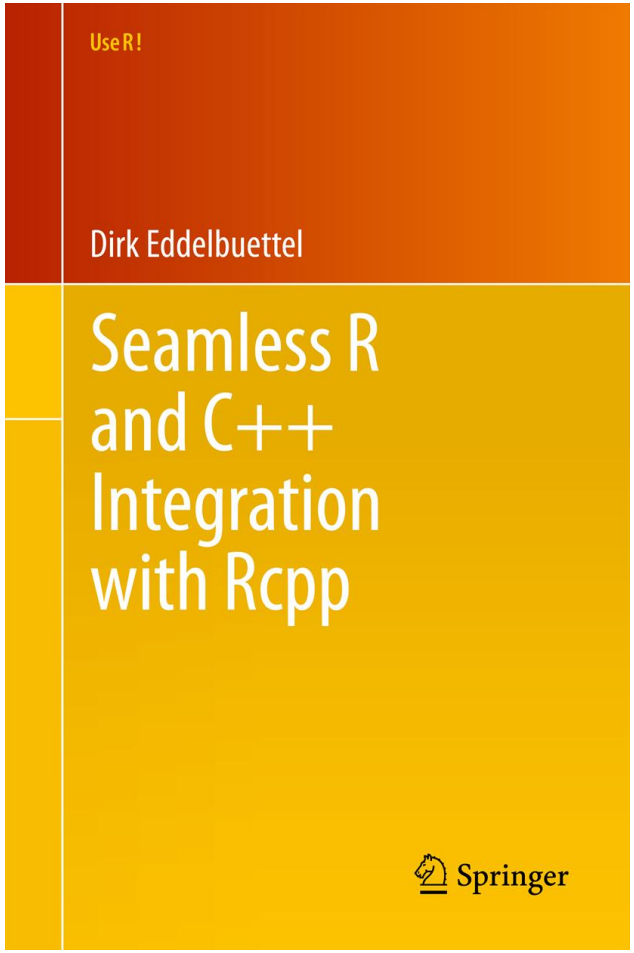
Segundo Exemplo

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

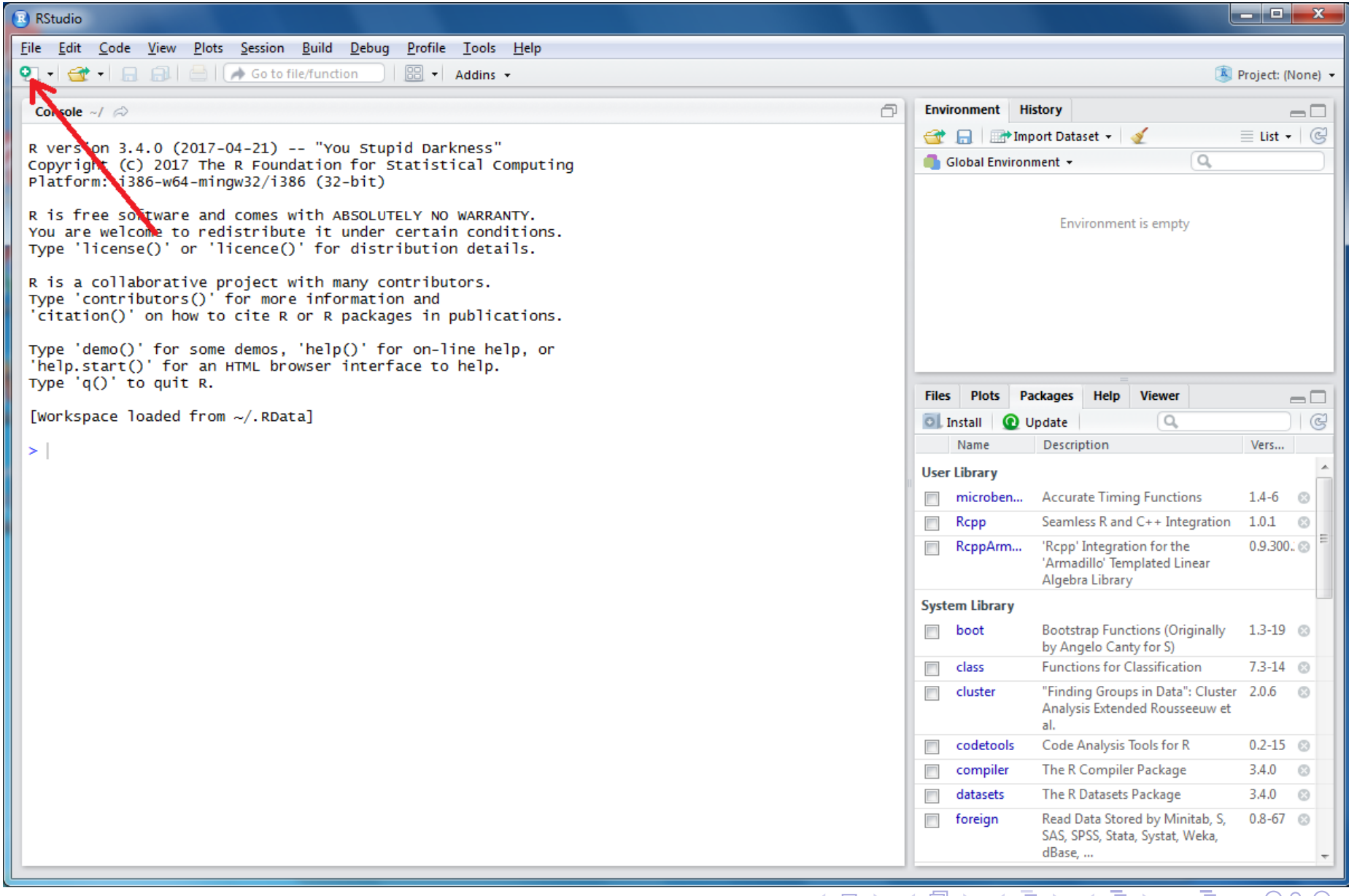


Livros Para Consultas

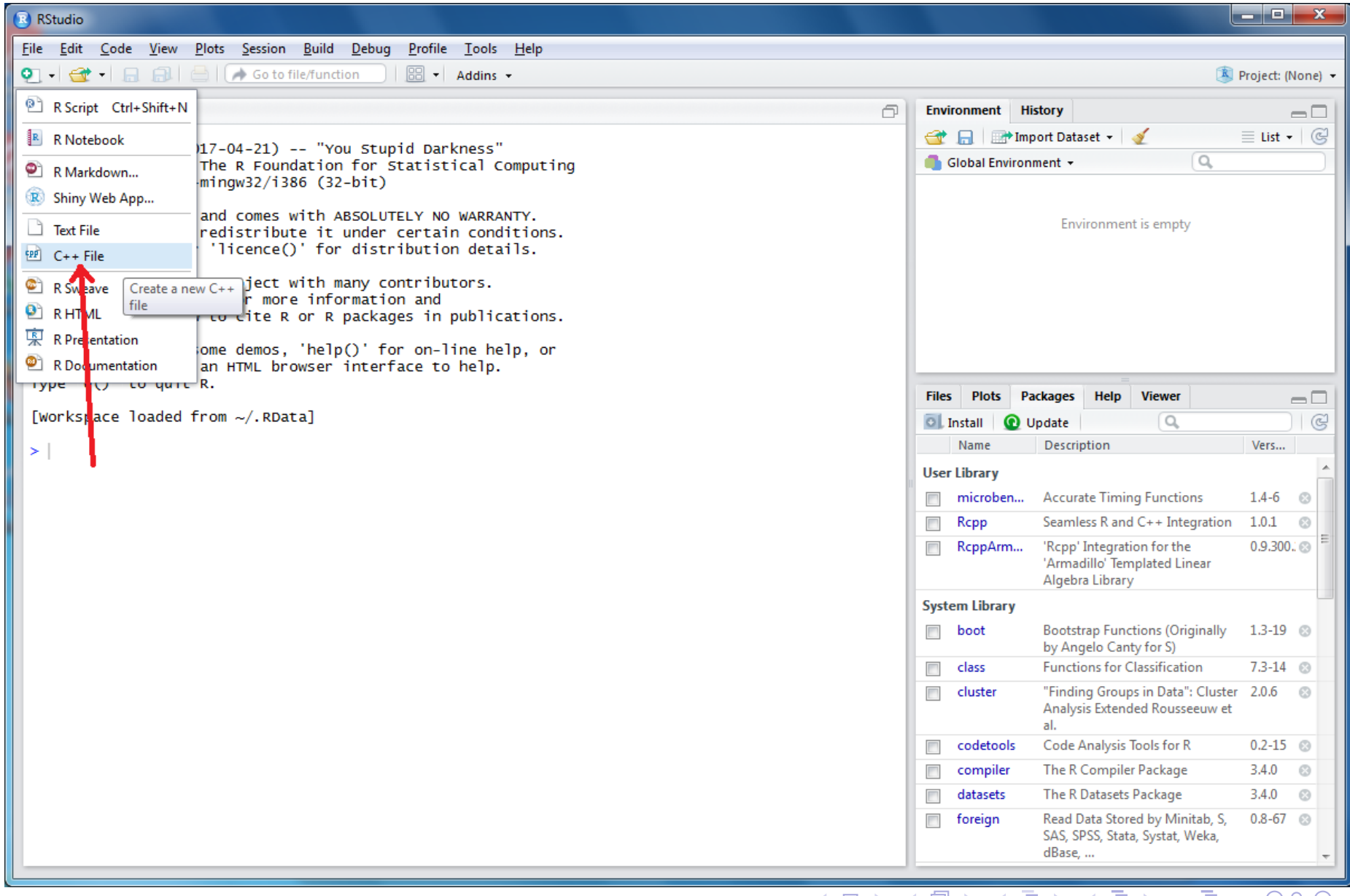




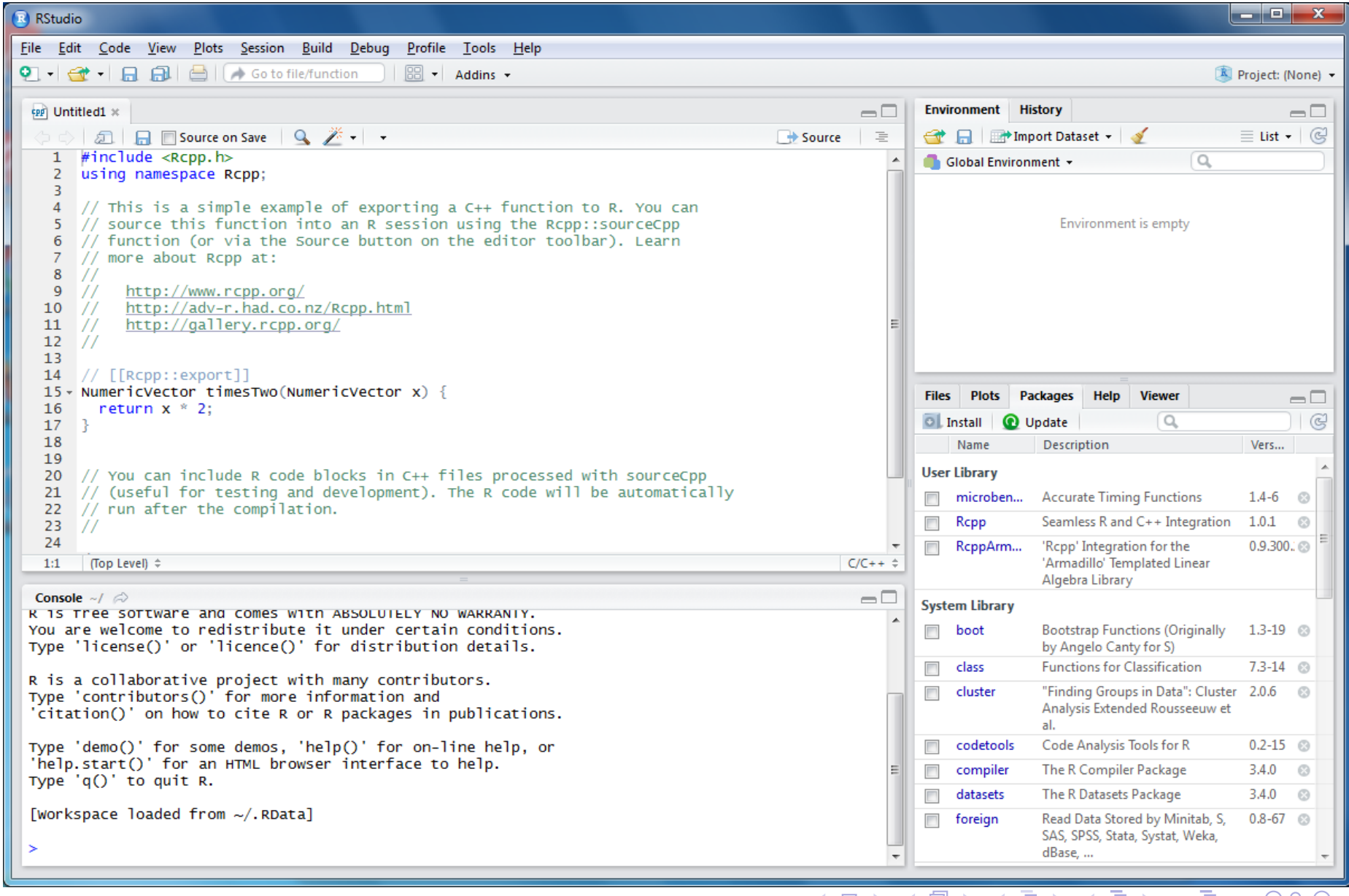
Como usamos ??


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console ~/
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
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, ... 0.8-67

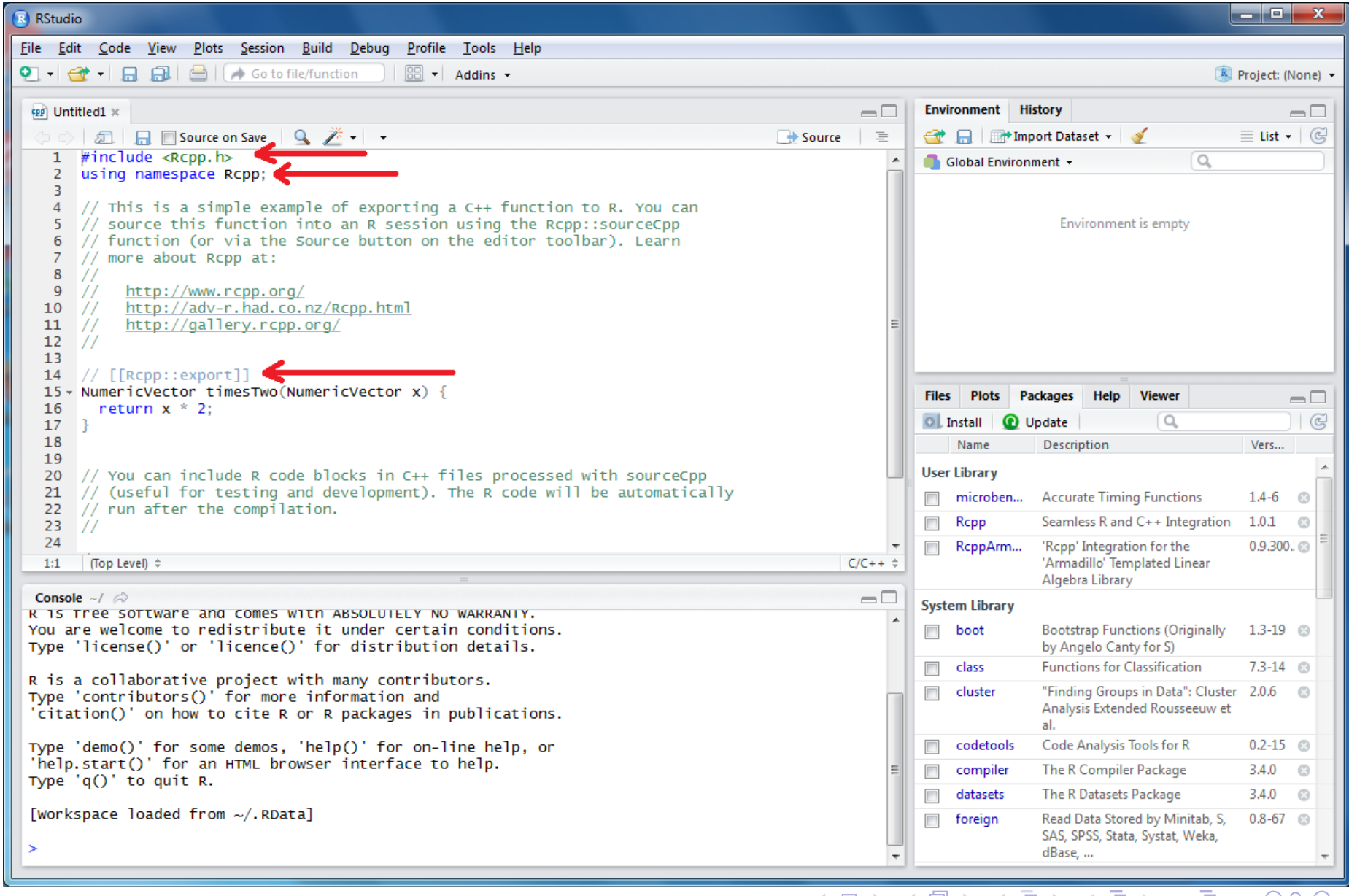
Primeiros passos



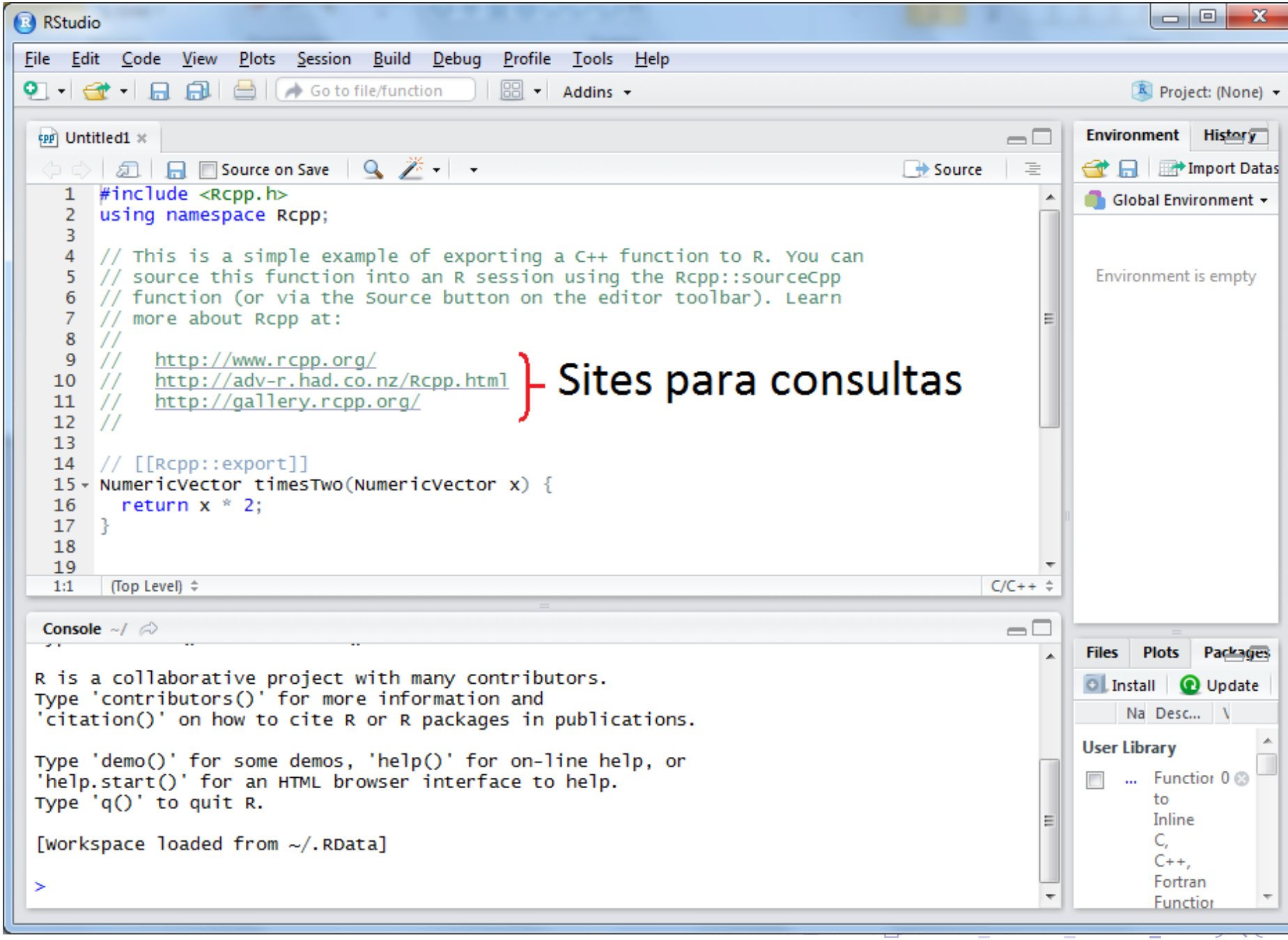
Apresentação



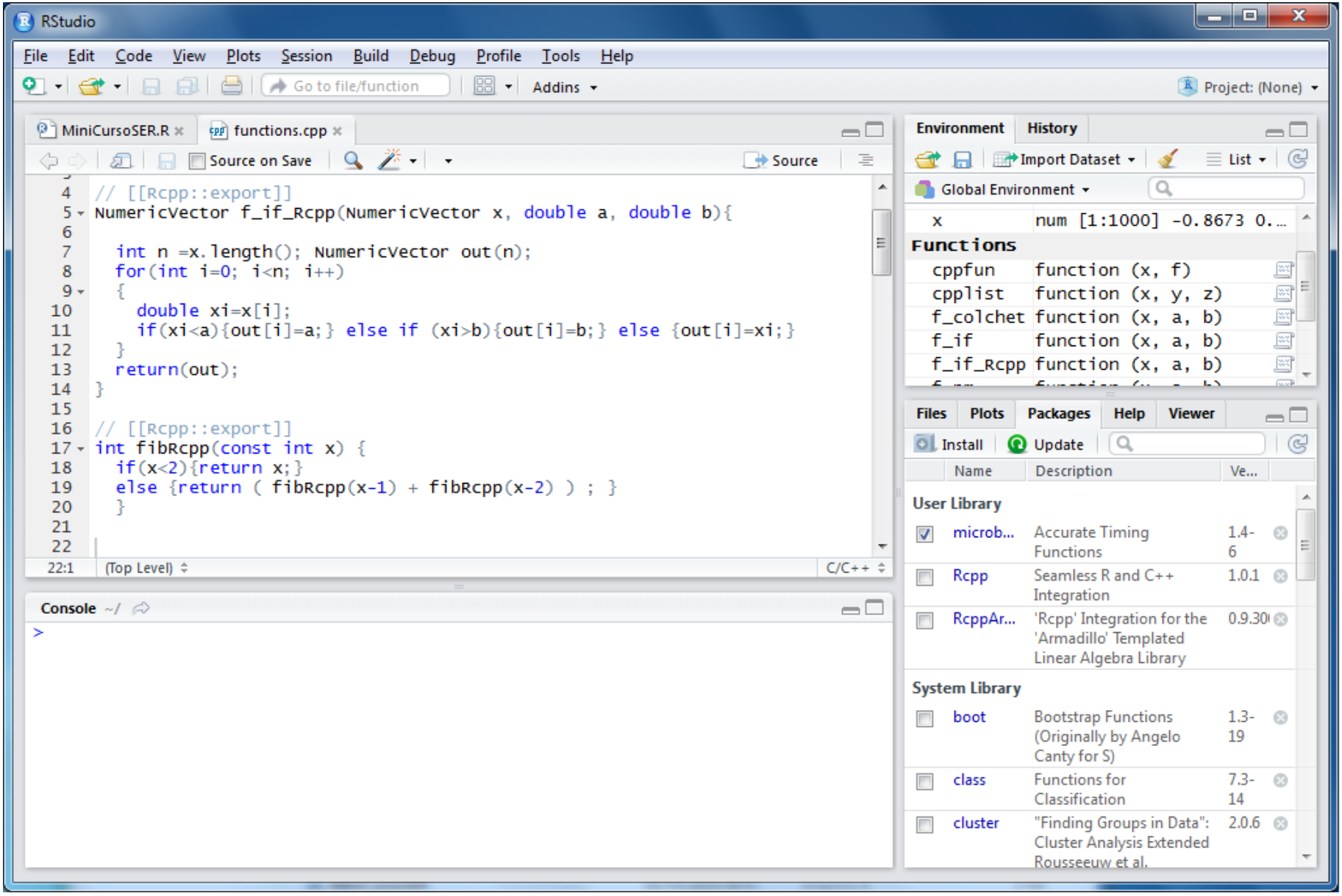
Apresentação



Apresentação



Script dos exemplos



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 matizes (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`.

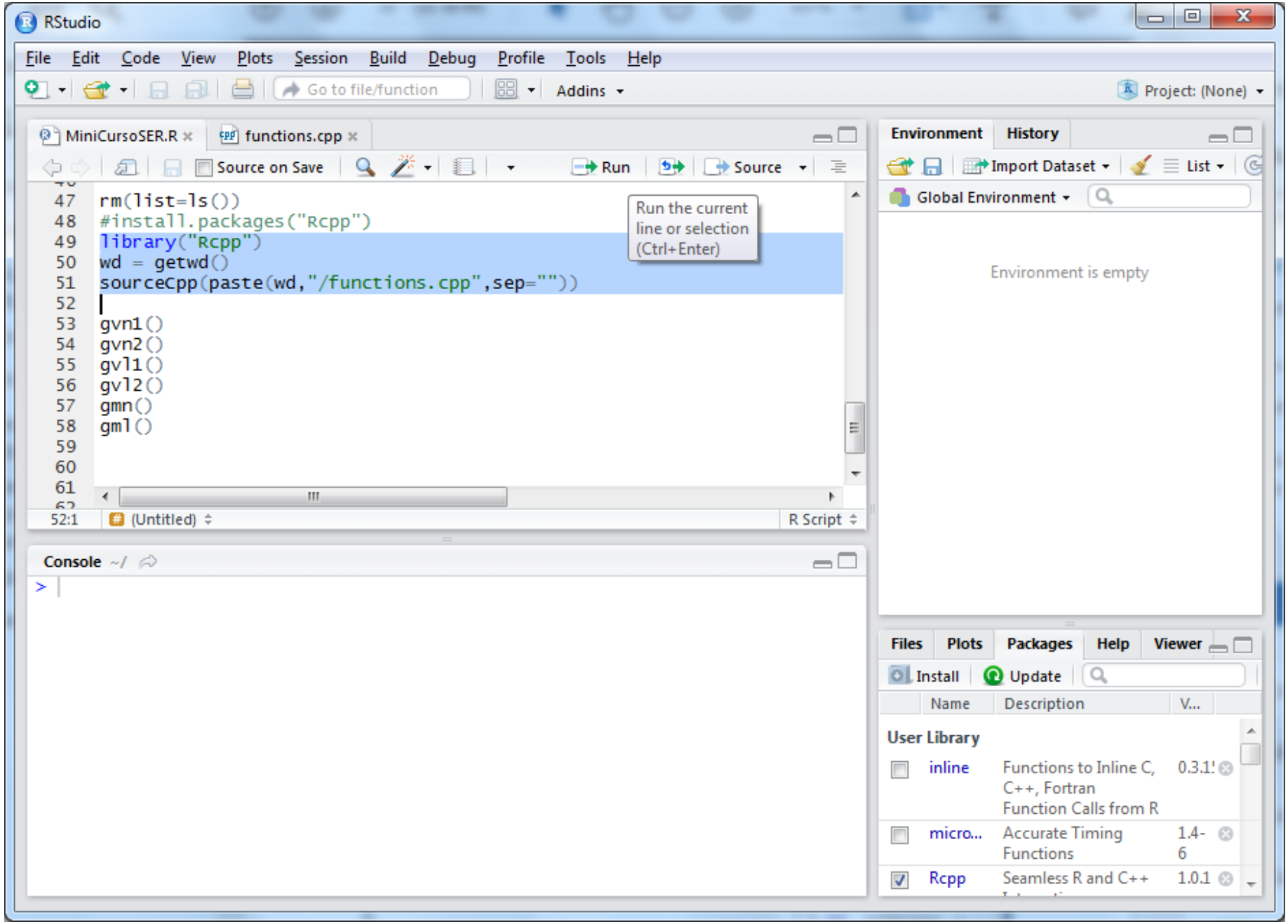
- A função `sourceCpp()`;
- A função `cppFunction()`;

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

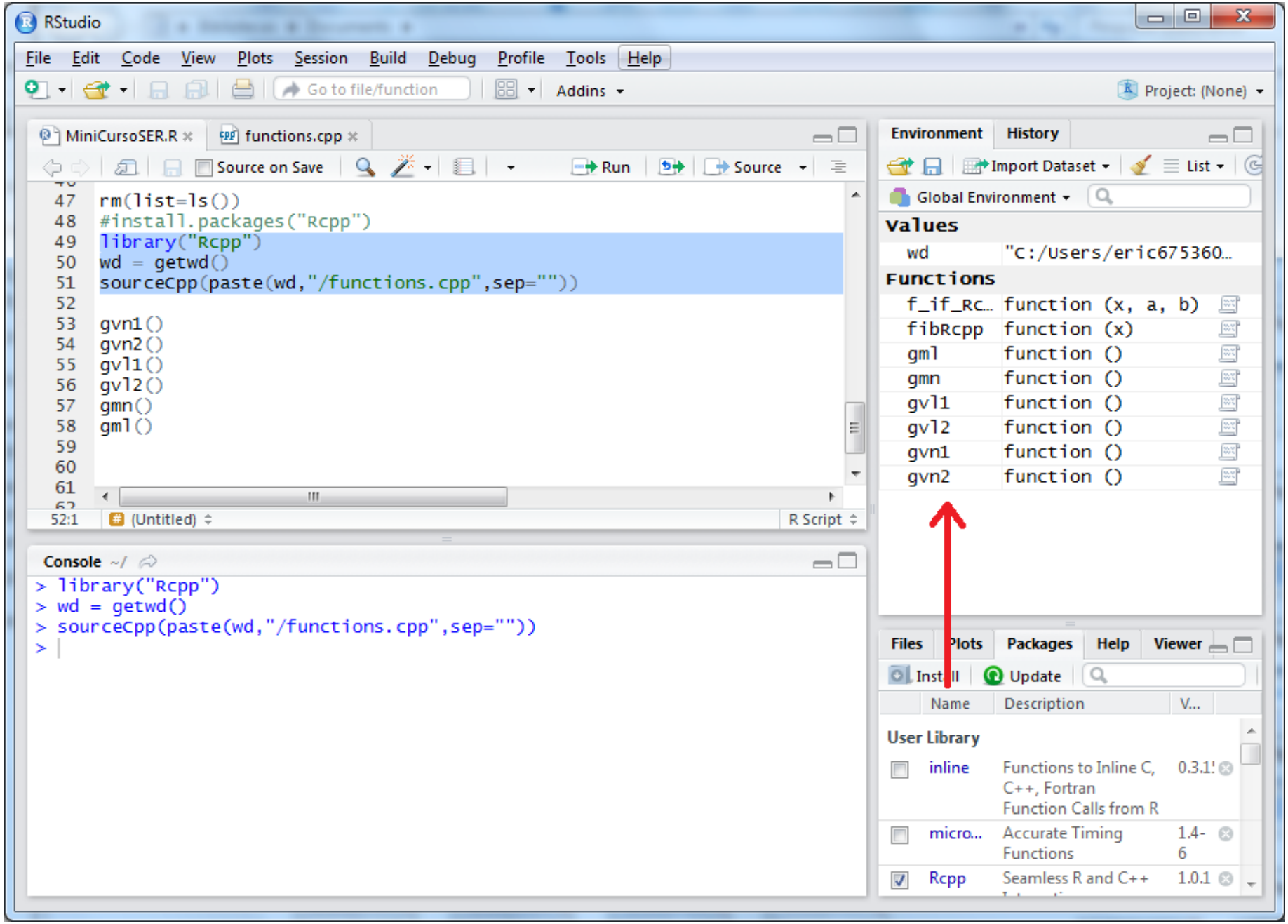
Na maioria das vezes nossos códigos (funções) em `C++` estarão em arquivos com extensão `.cpp`.

Vamos começar devagar!!

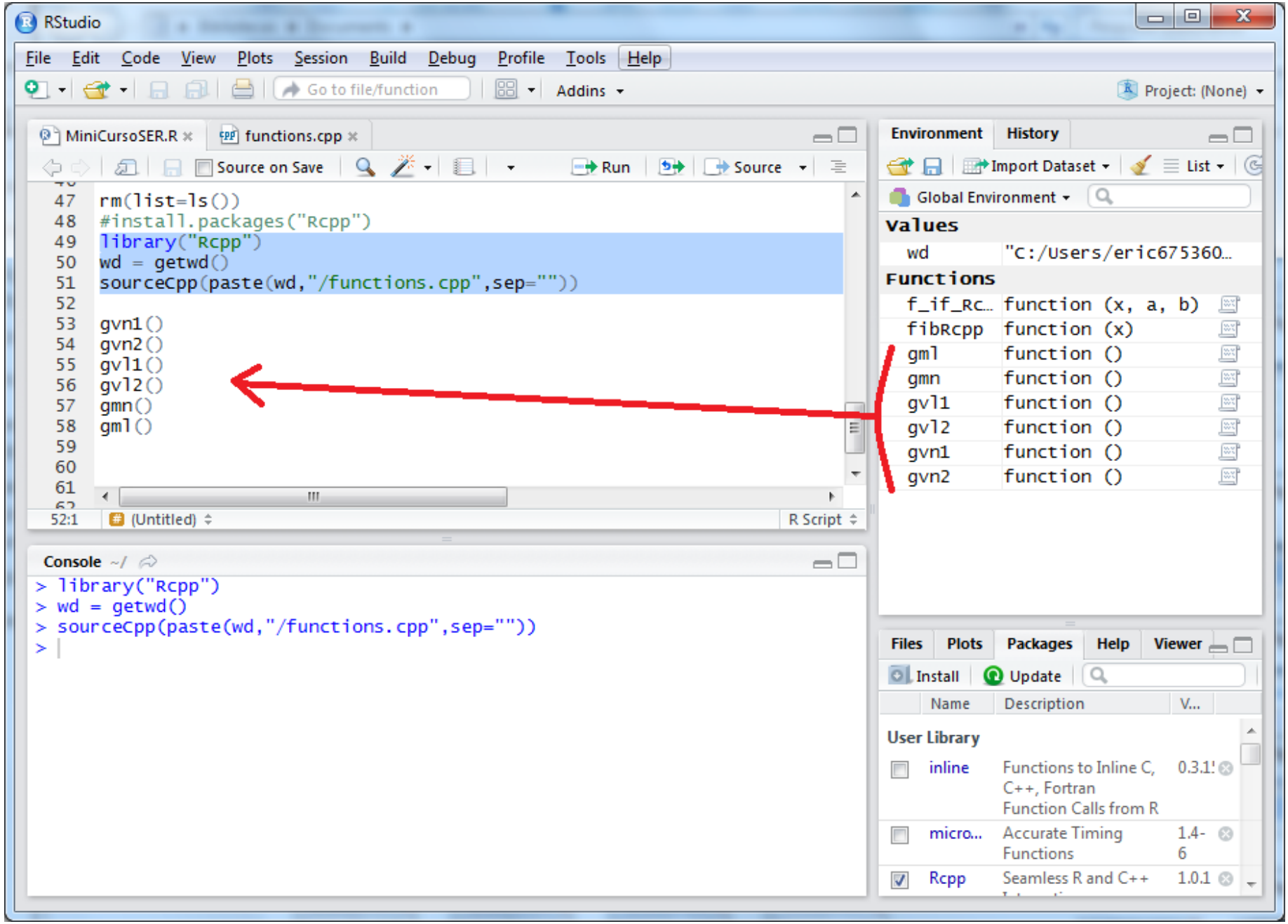
Primeiras Funções



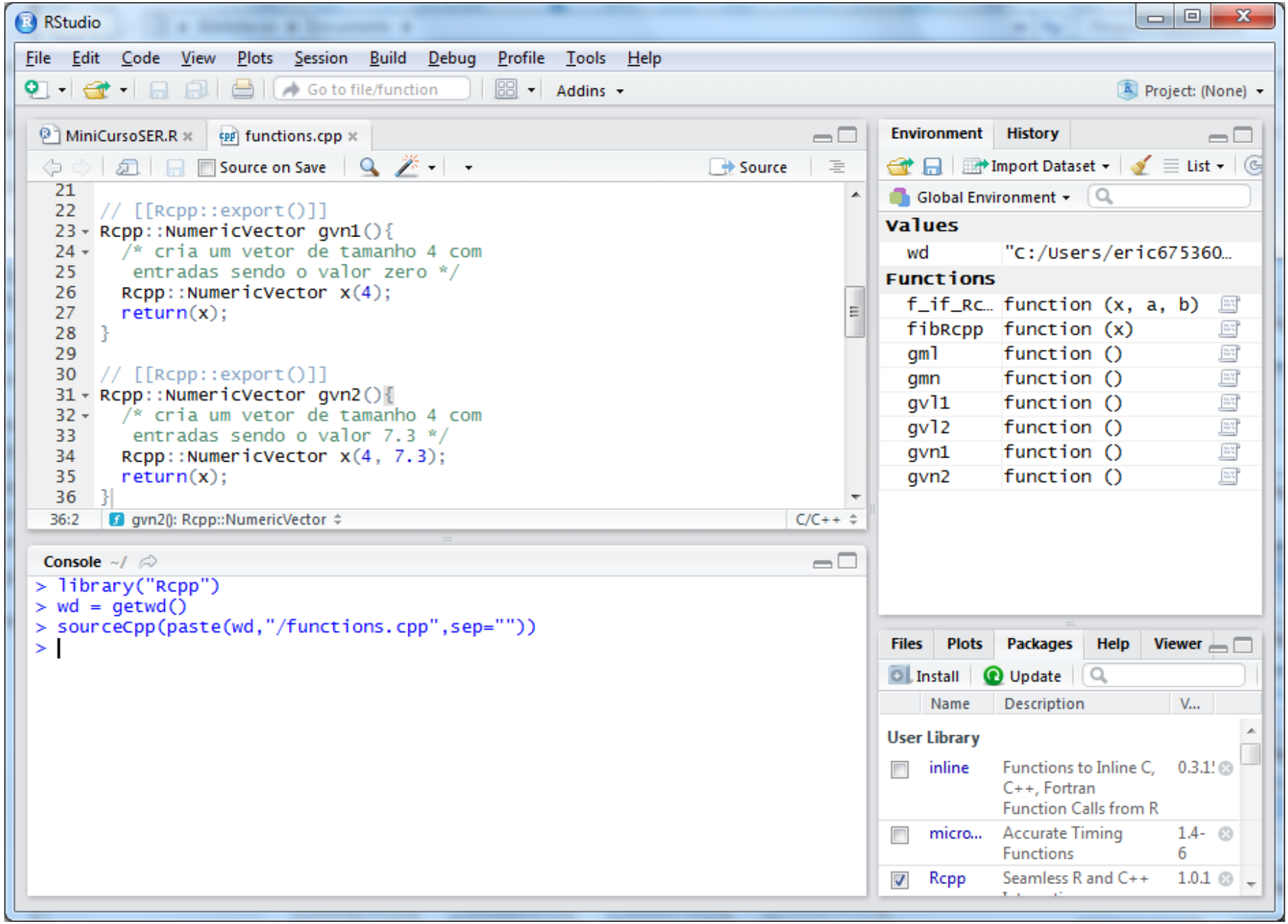
Primeiras Funções



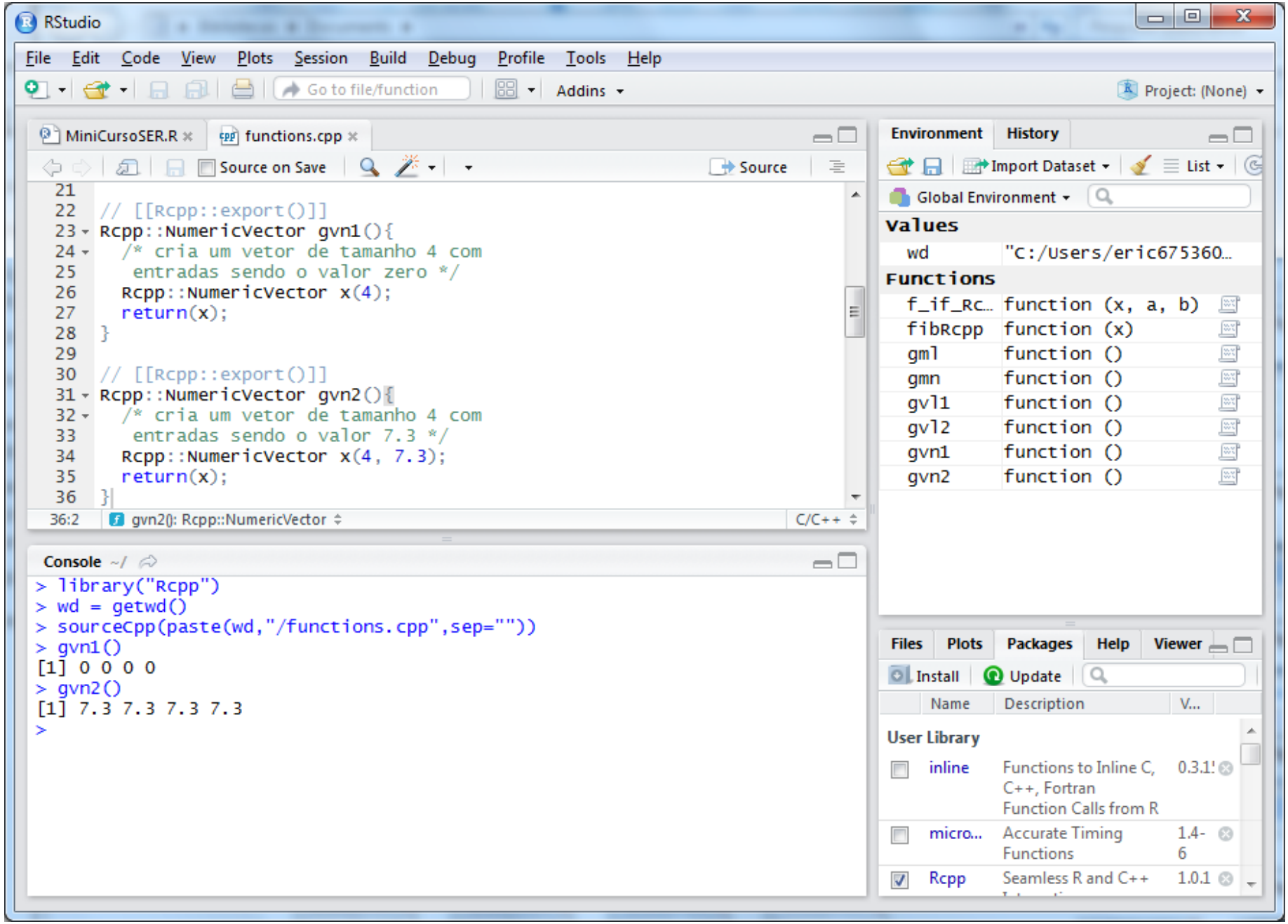
Primeiras Funções



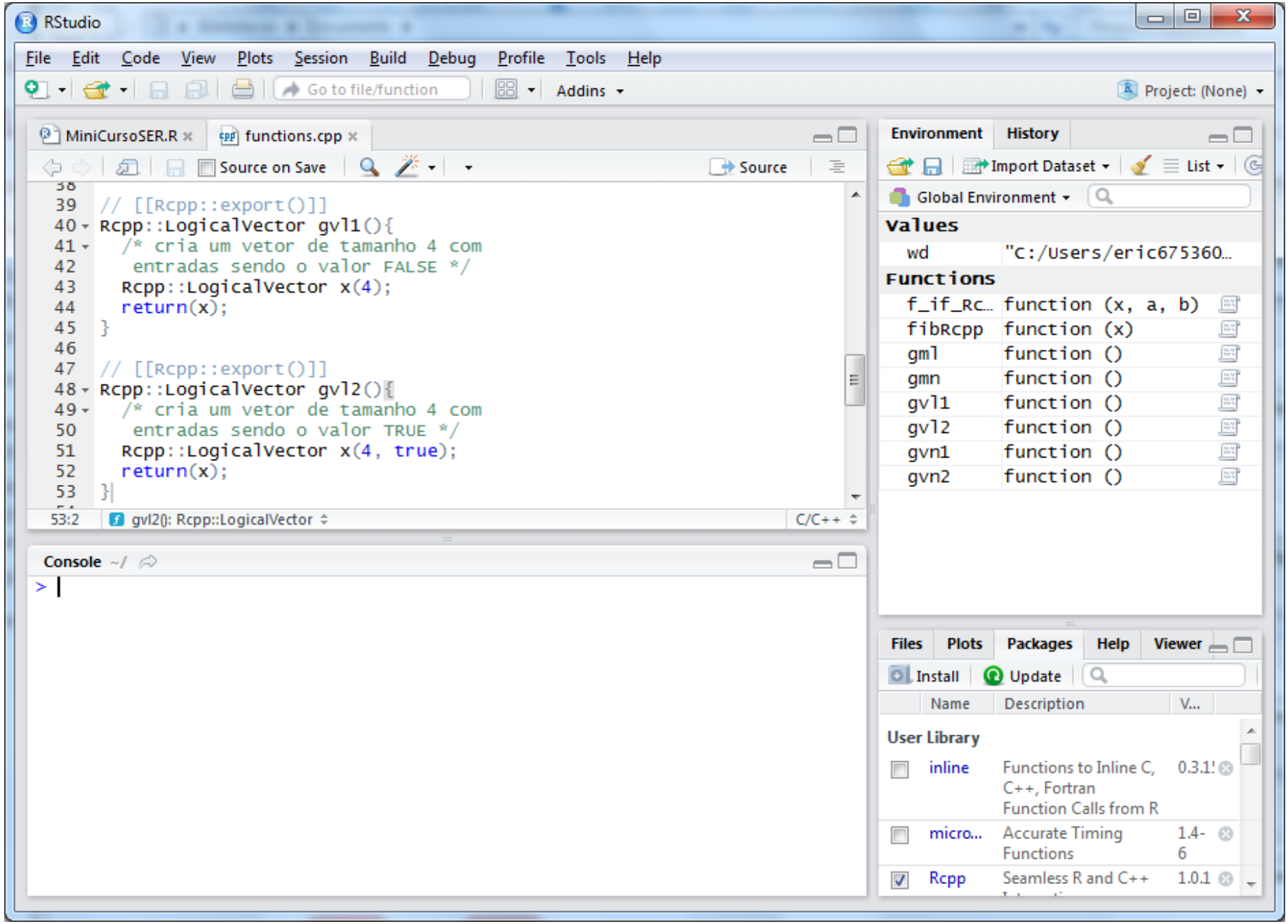
Primeiras Funções



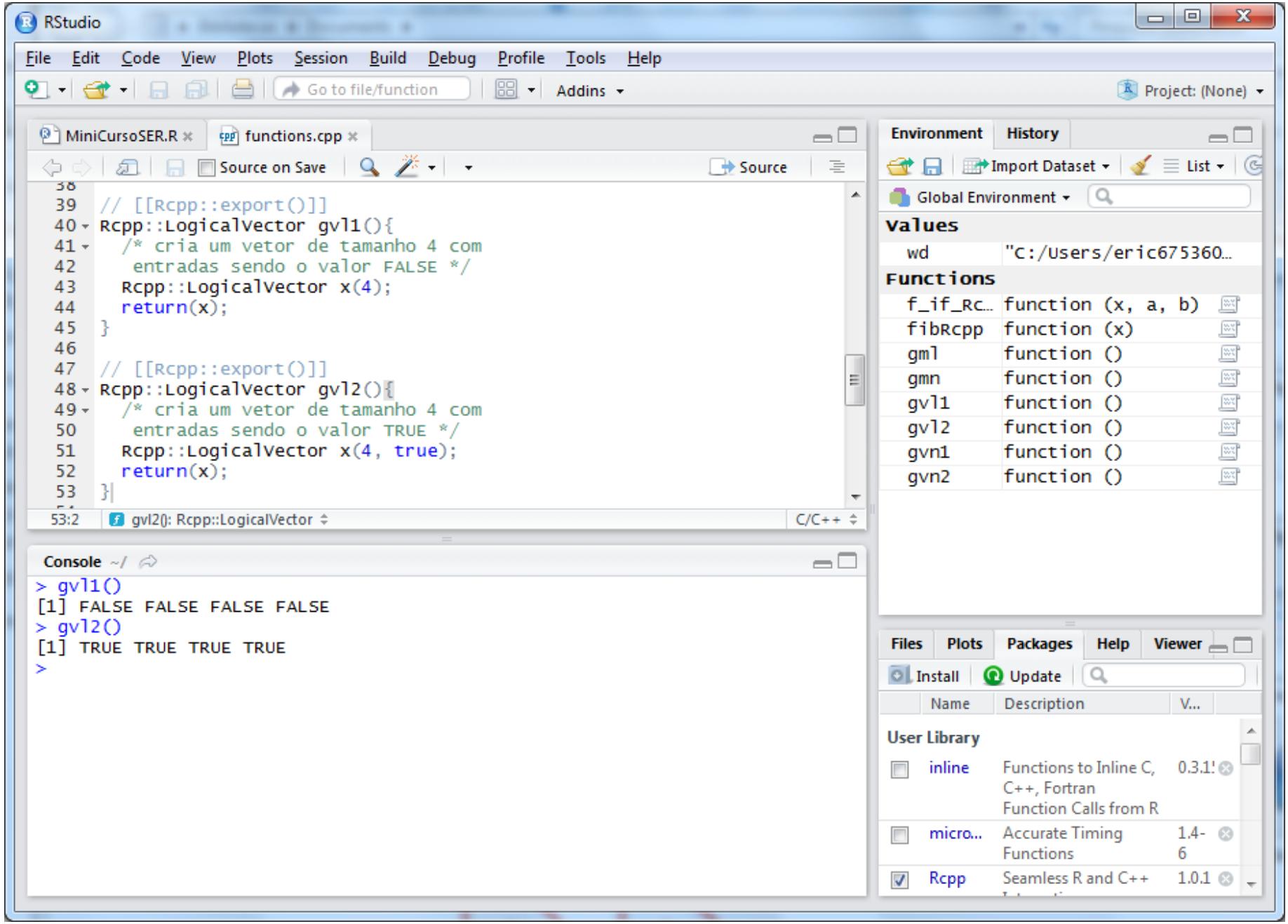
Primeiras Funções



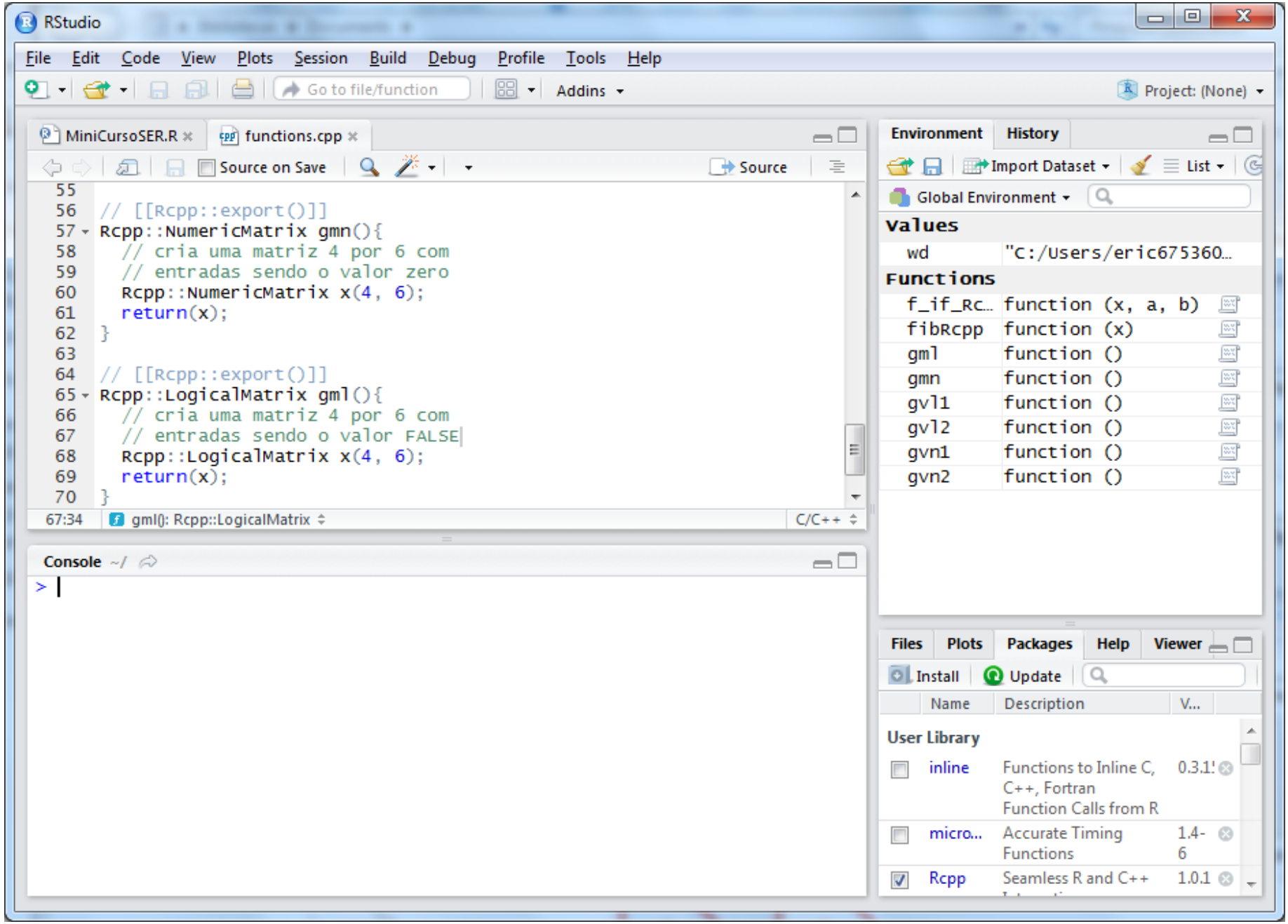
Primeiras Funções



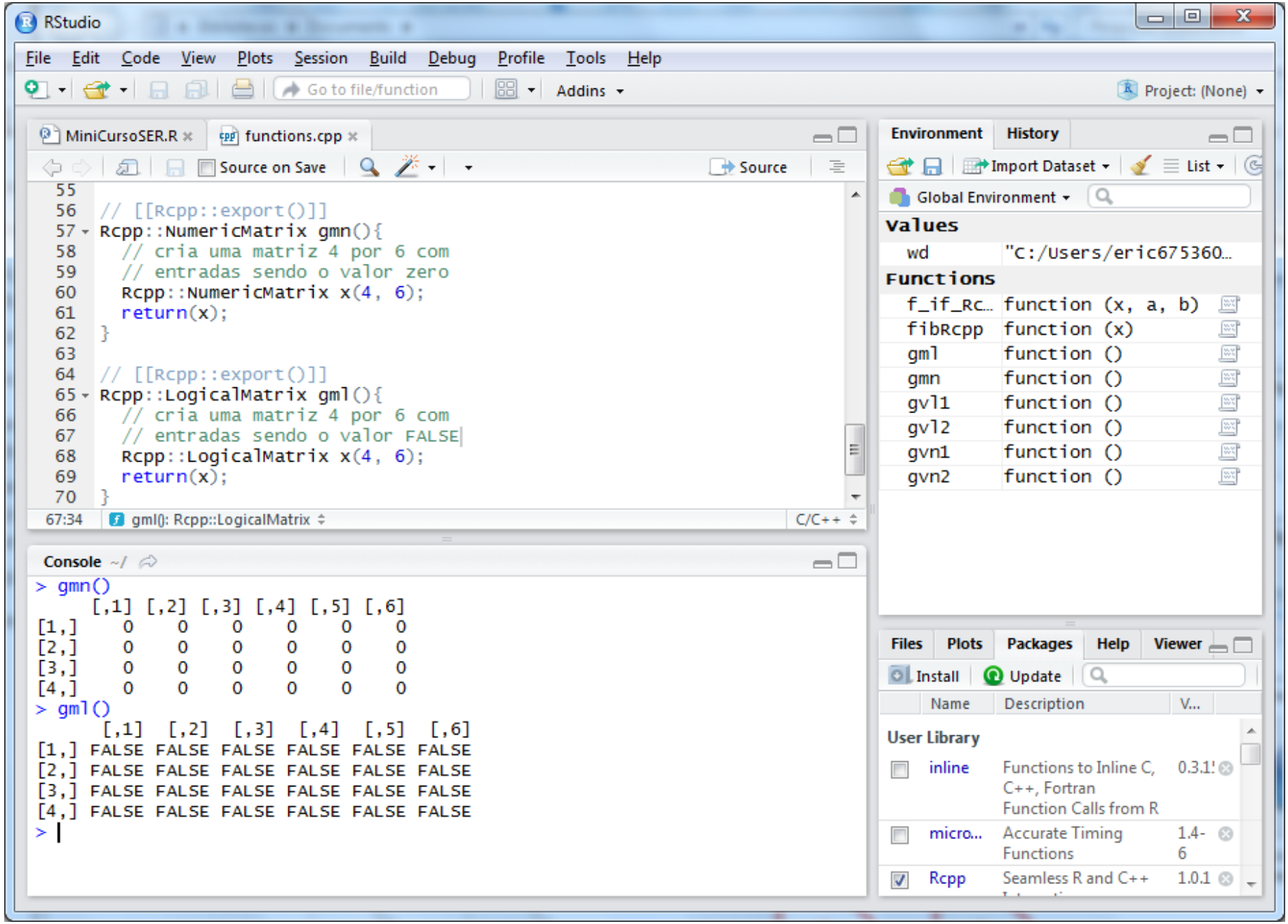
Primeiras Funções



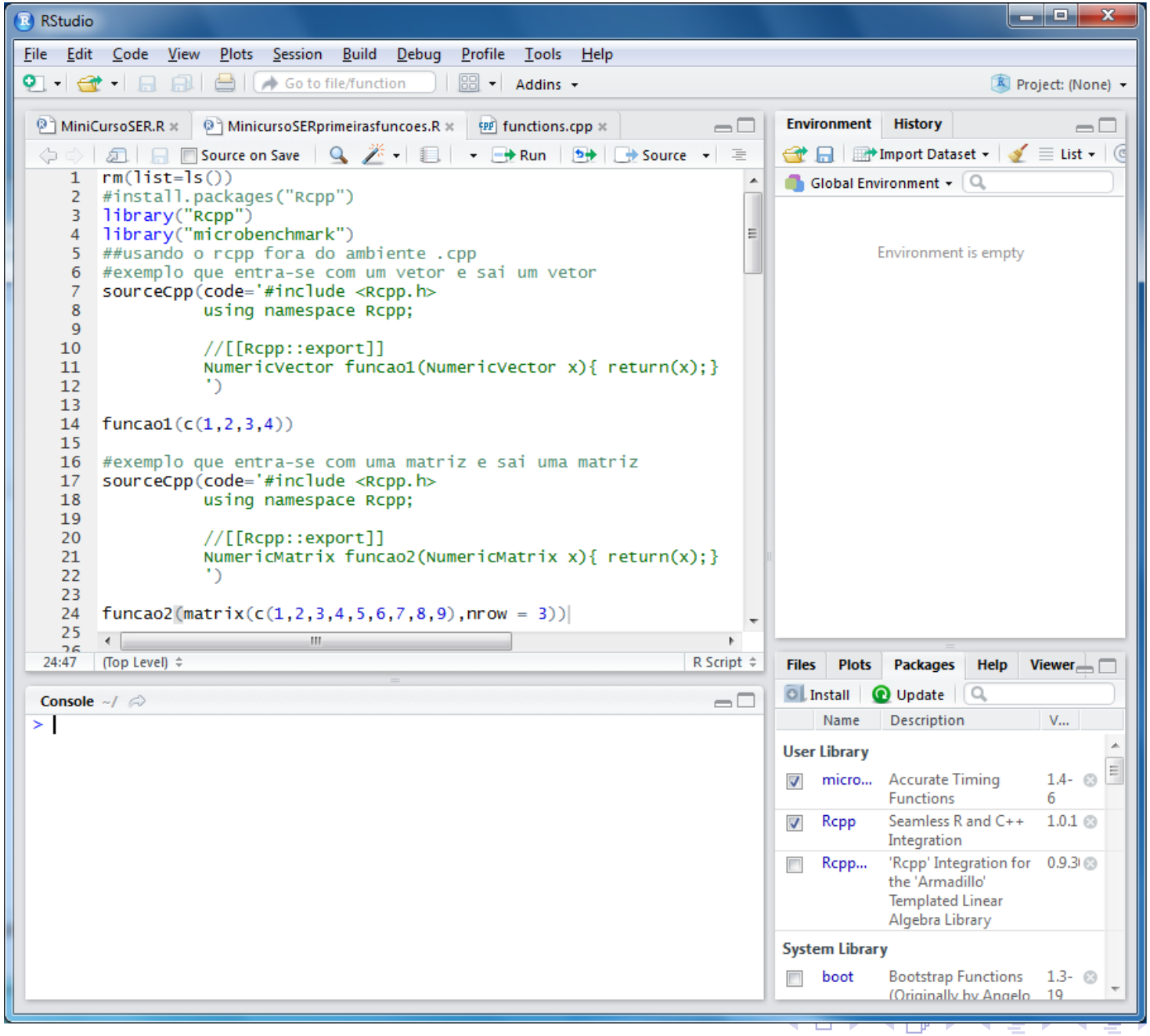
Primeiras Funções



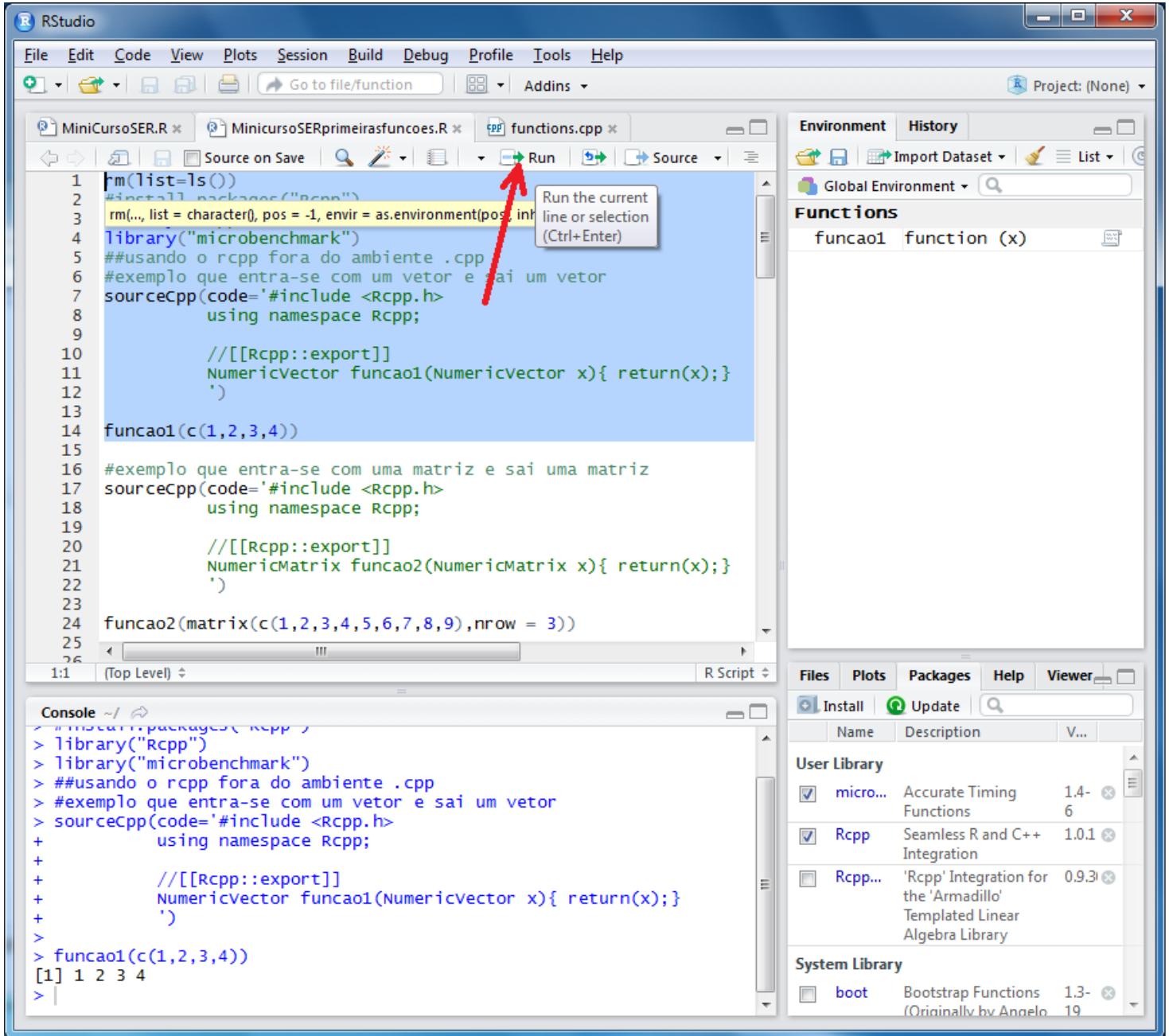
Primeiras Funções



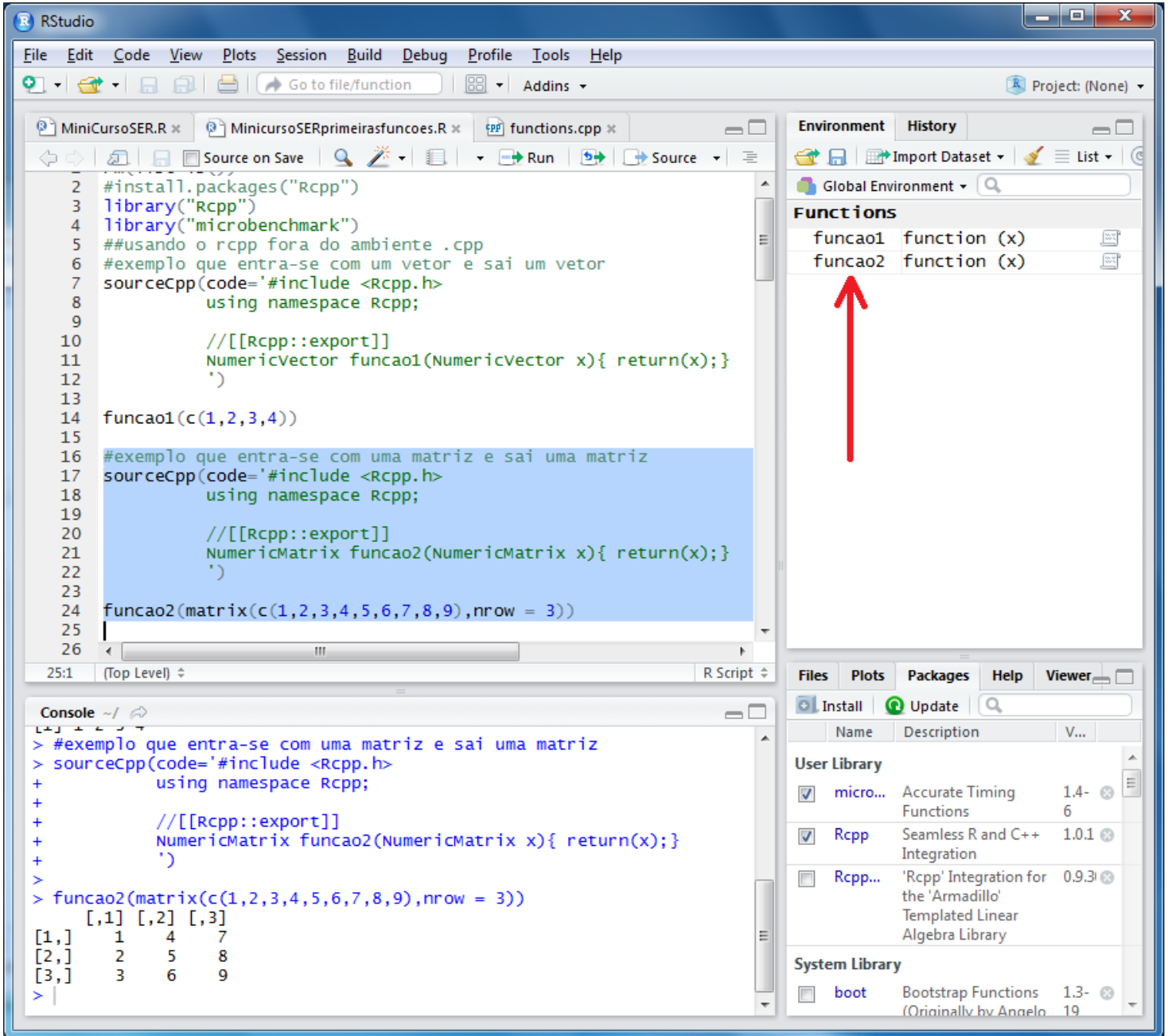
Primeiras Funções



Primeiras Funções



Primeiras Funções



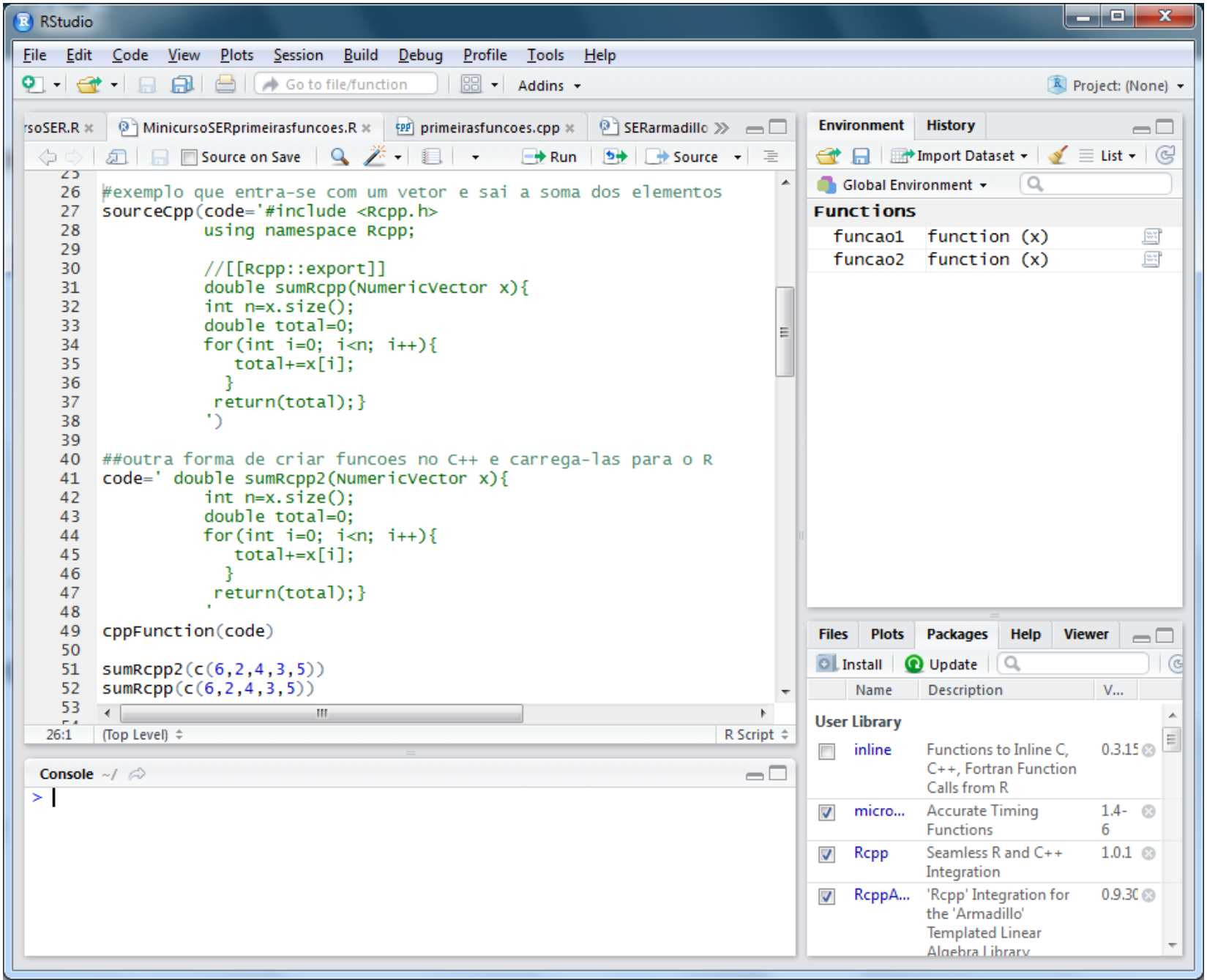
Recursos Matemáticos e laços

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

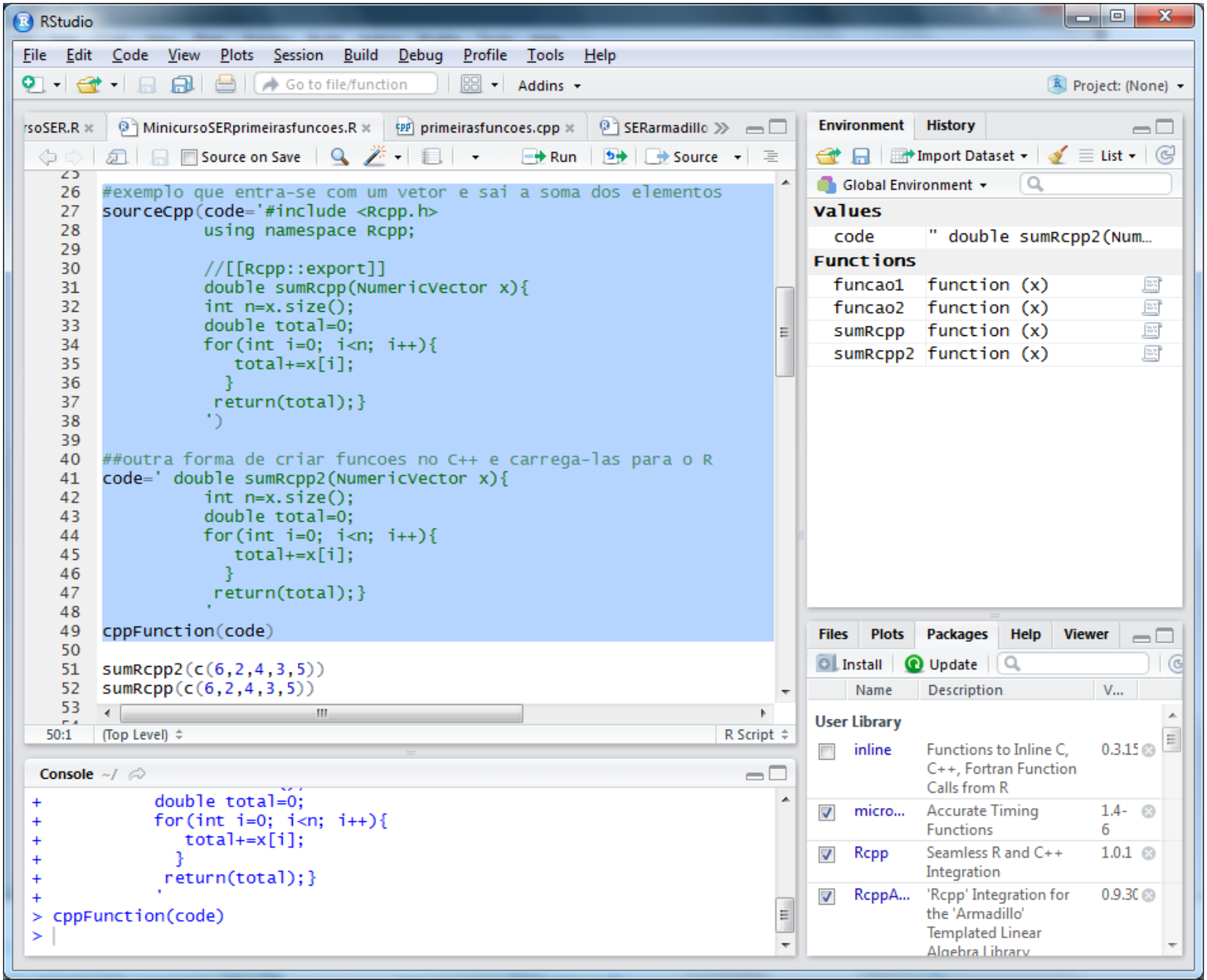
Os operadores lógicos: `==`, `!=`, `<=`, `>=`, `>`, `|` (ou), `&` (e). Também podem ser usados da mesma forma como no `R`.

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

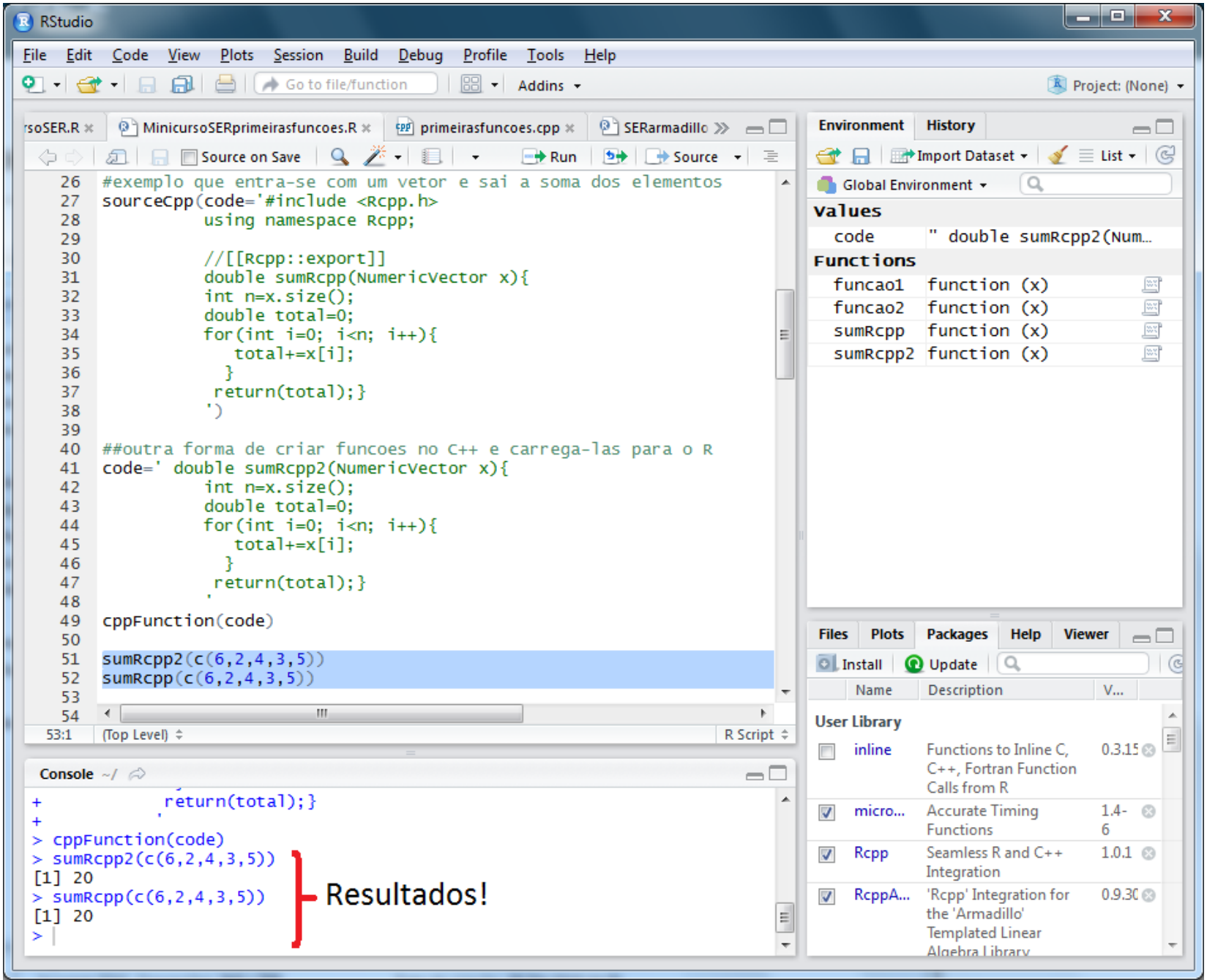
Primeiras Funções



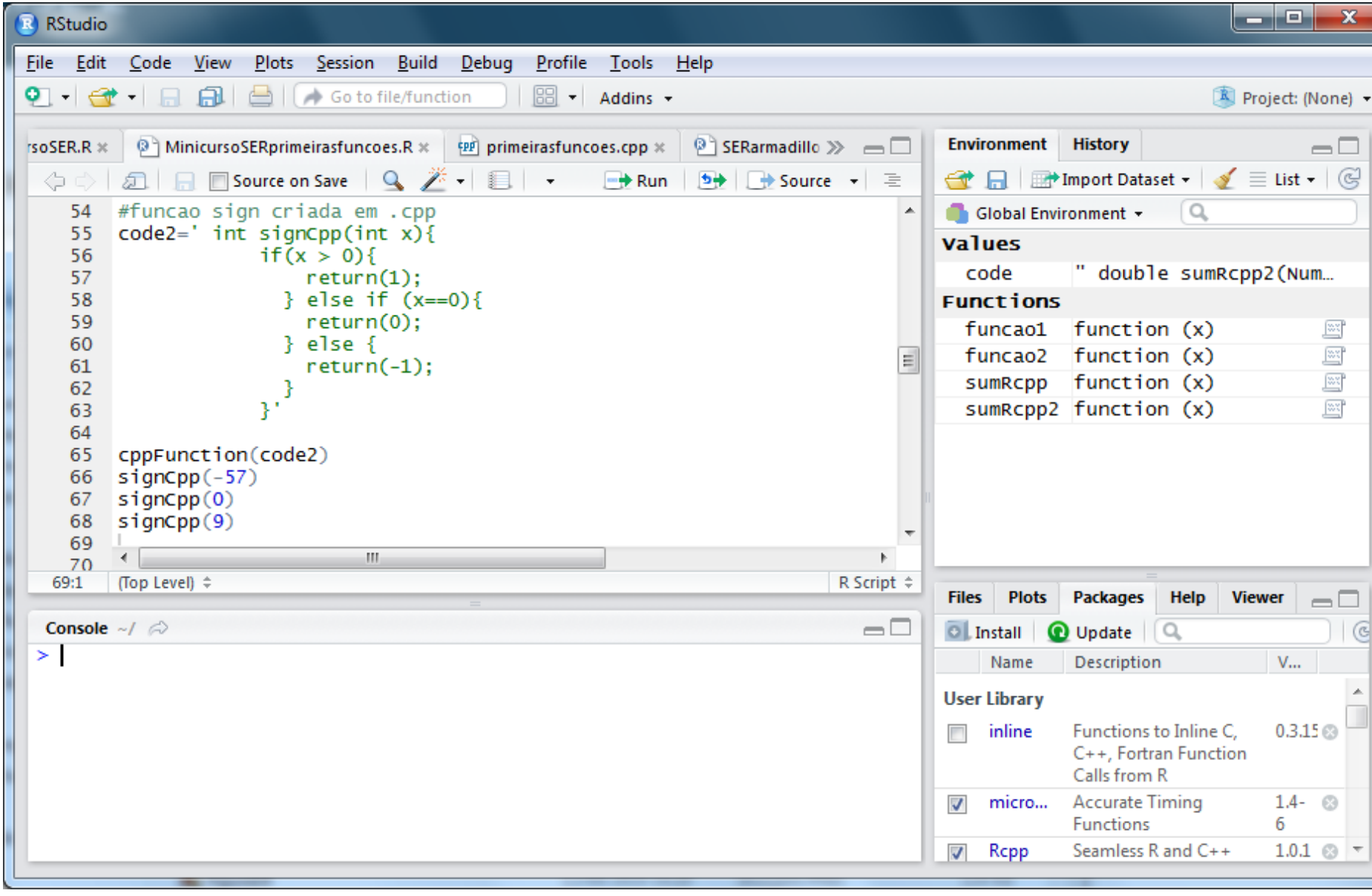
Primeiras Funções



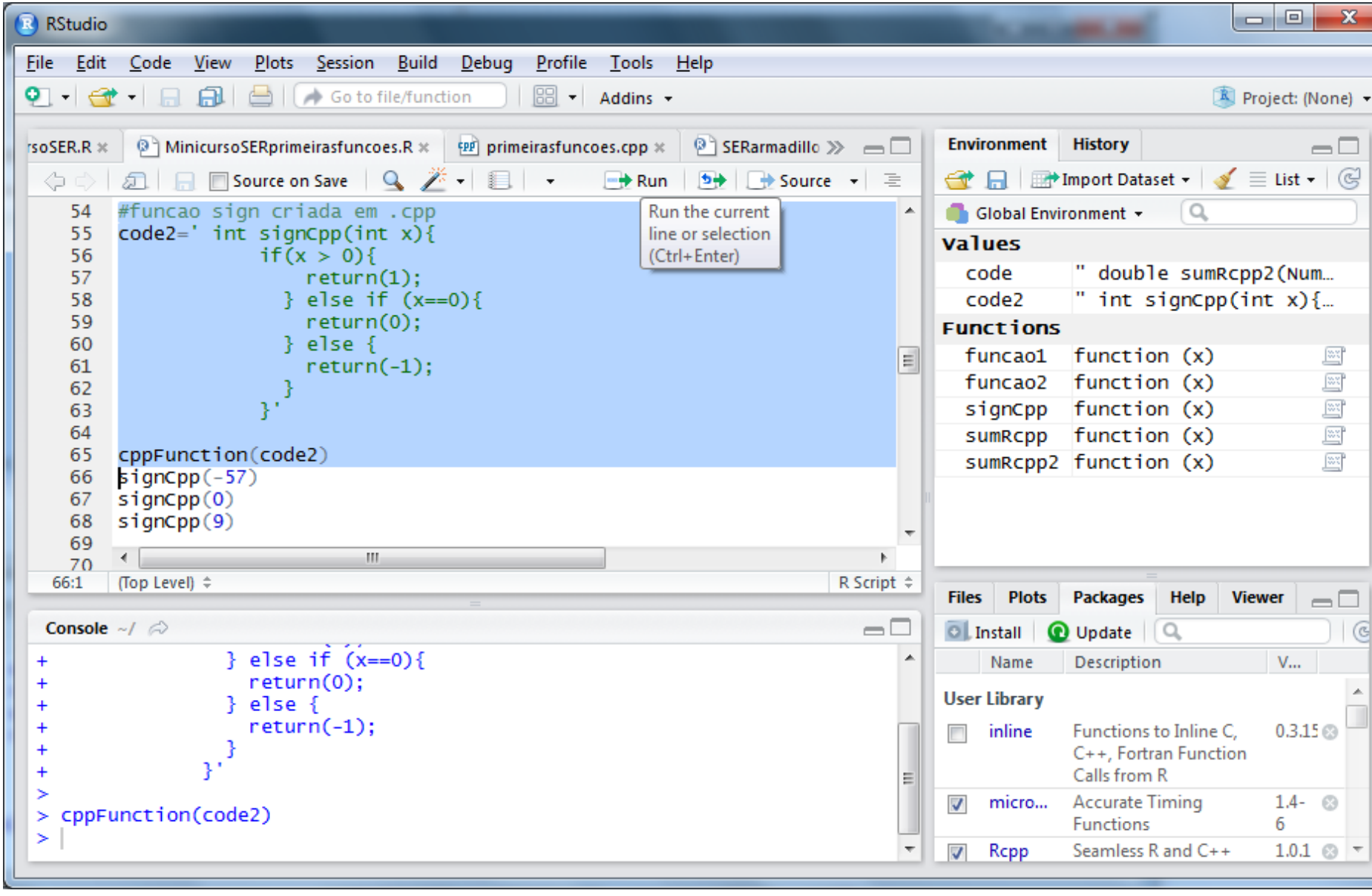
Primeiras Funções



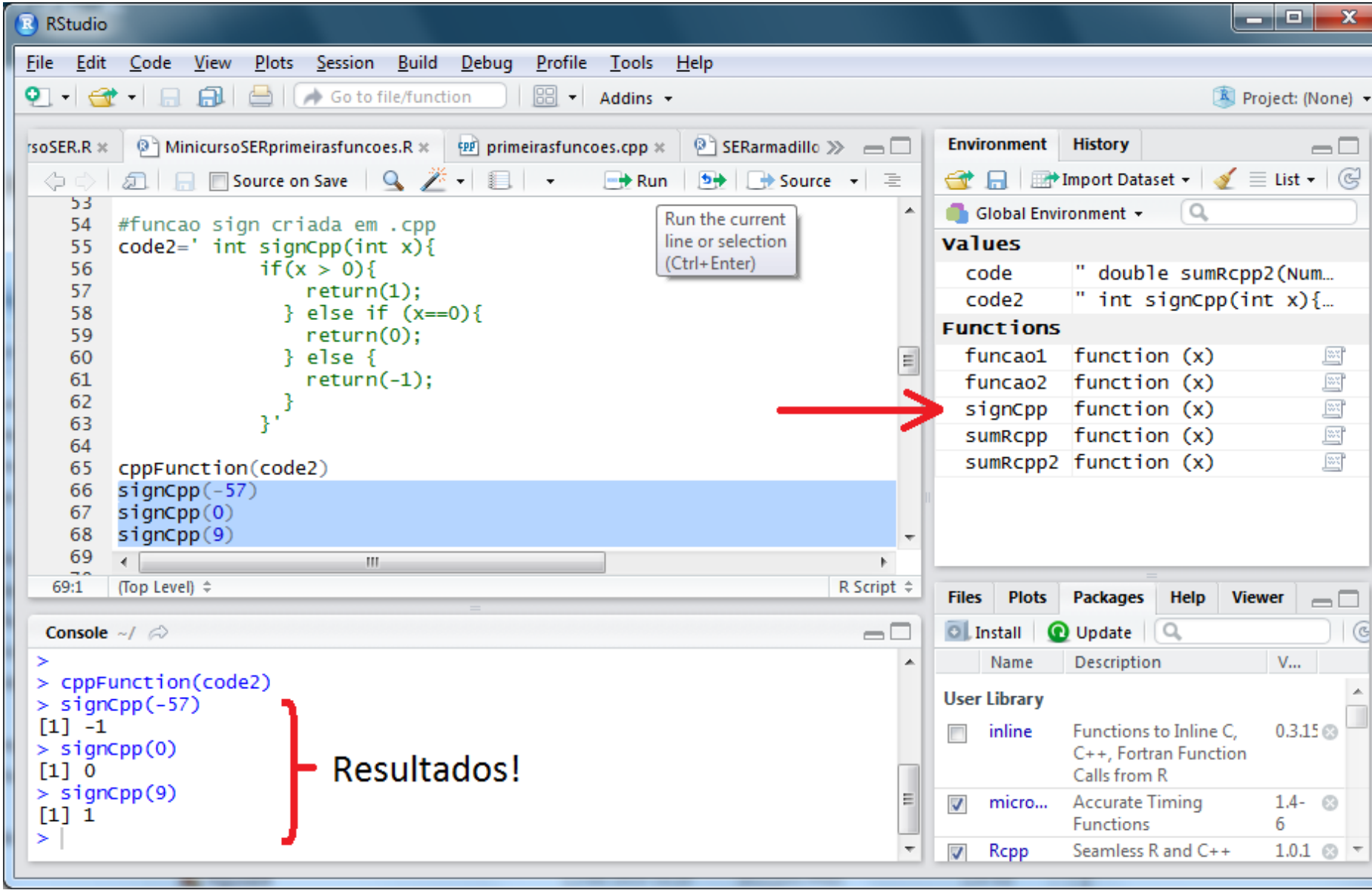
Primeiras Funções



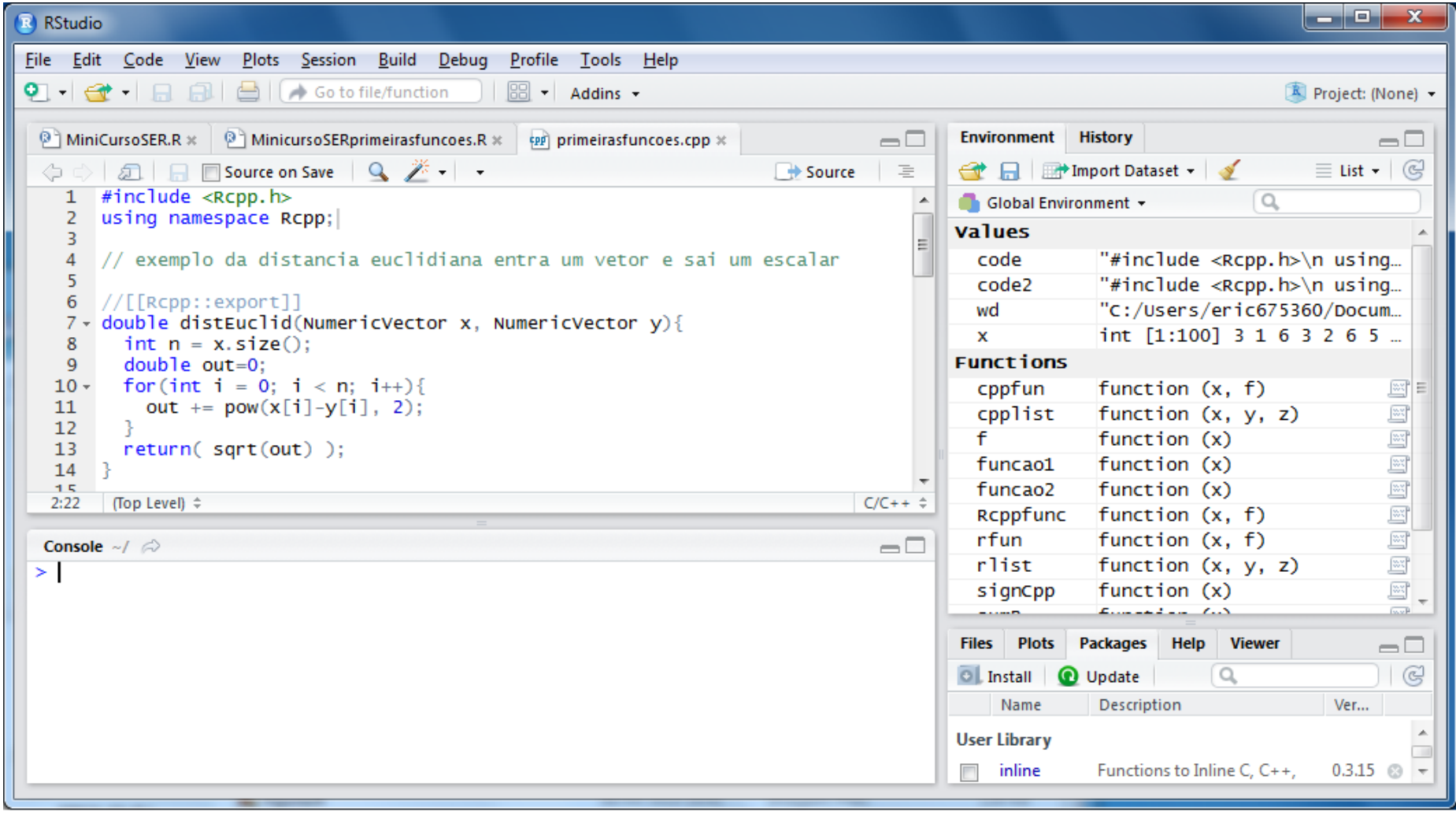
Primeiras Funções



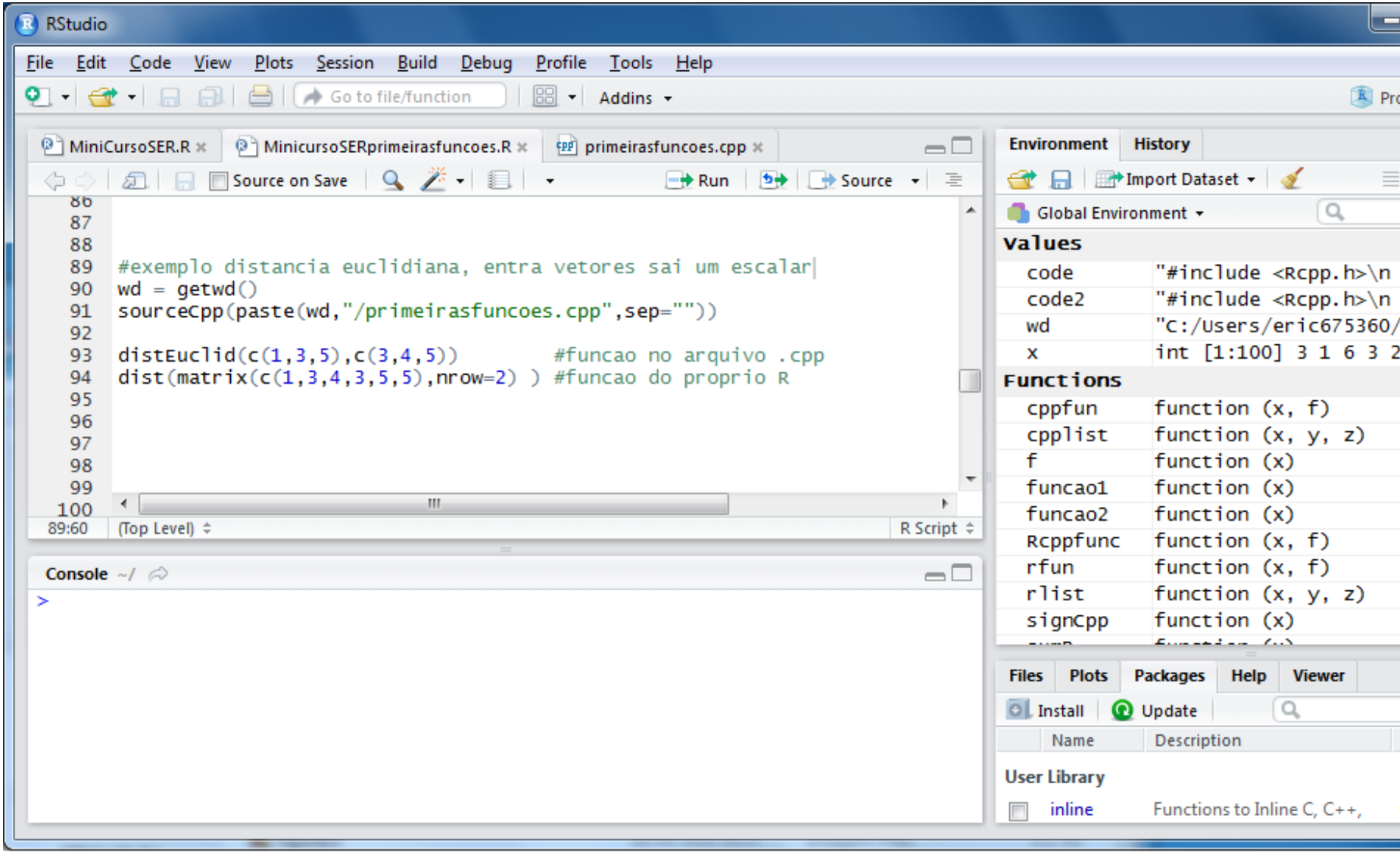
Primeiras Funções



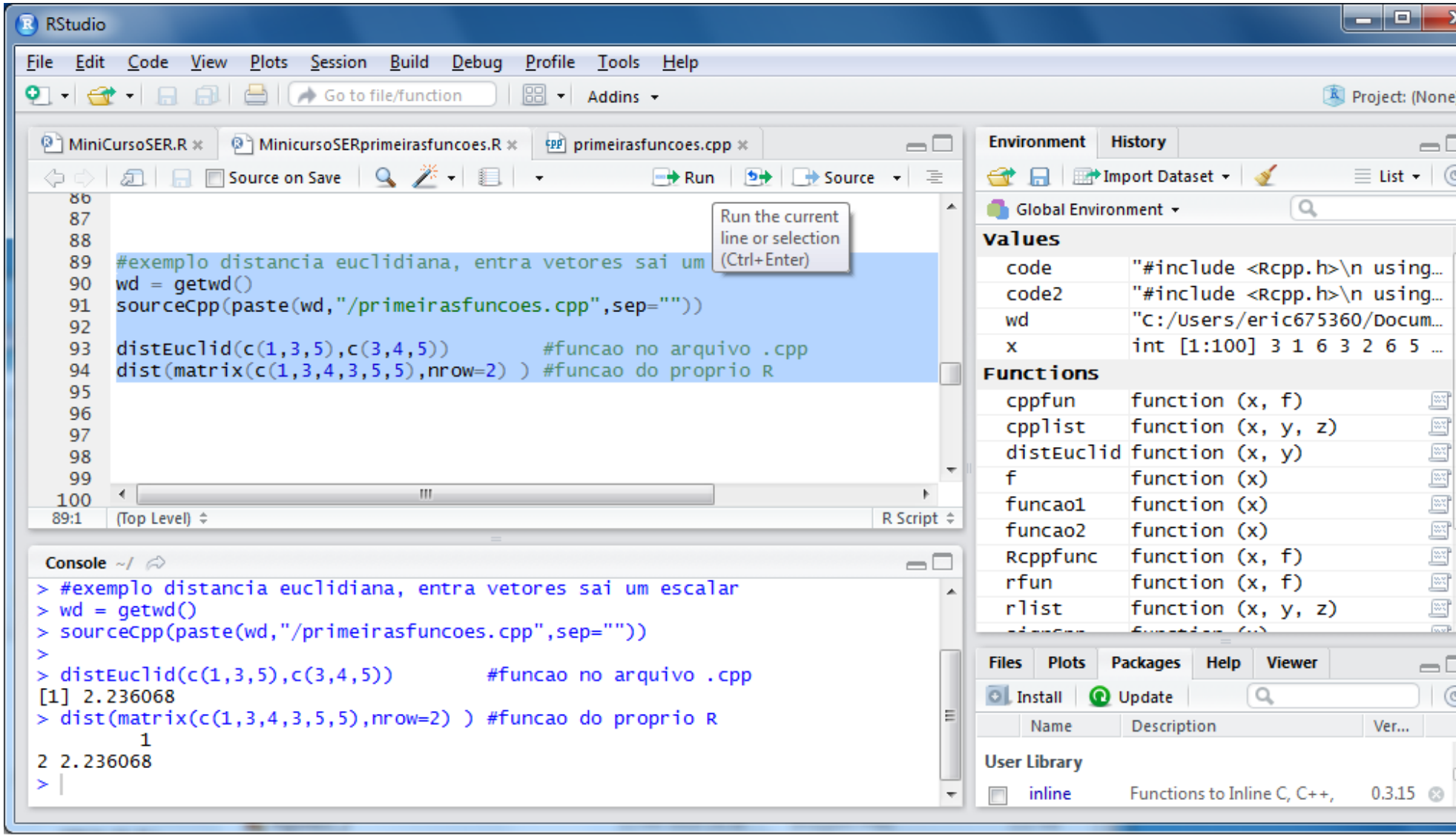
Primeiras Funções



Primeiras Funções

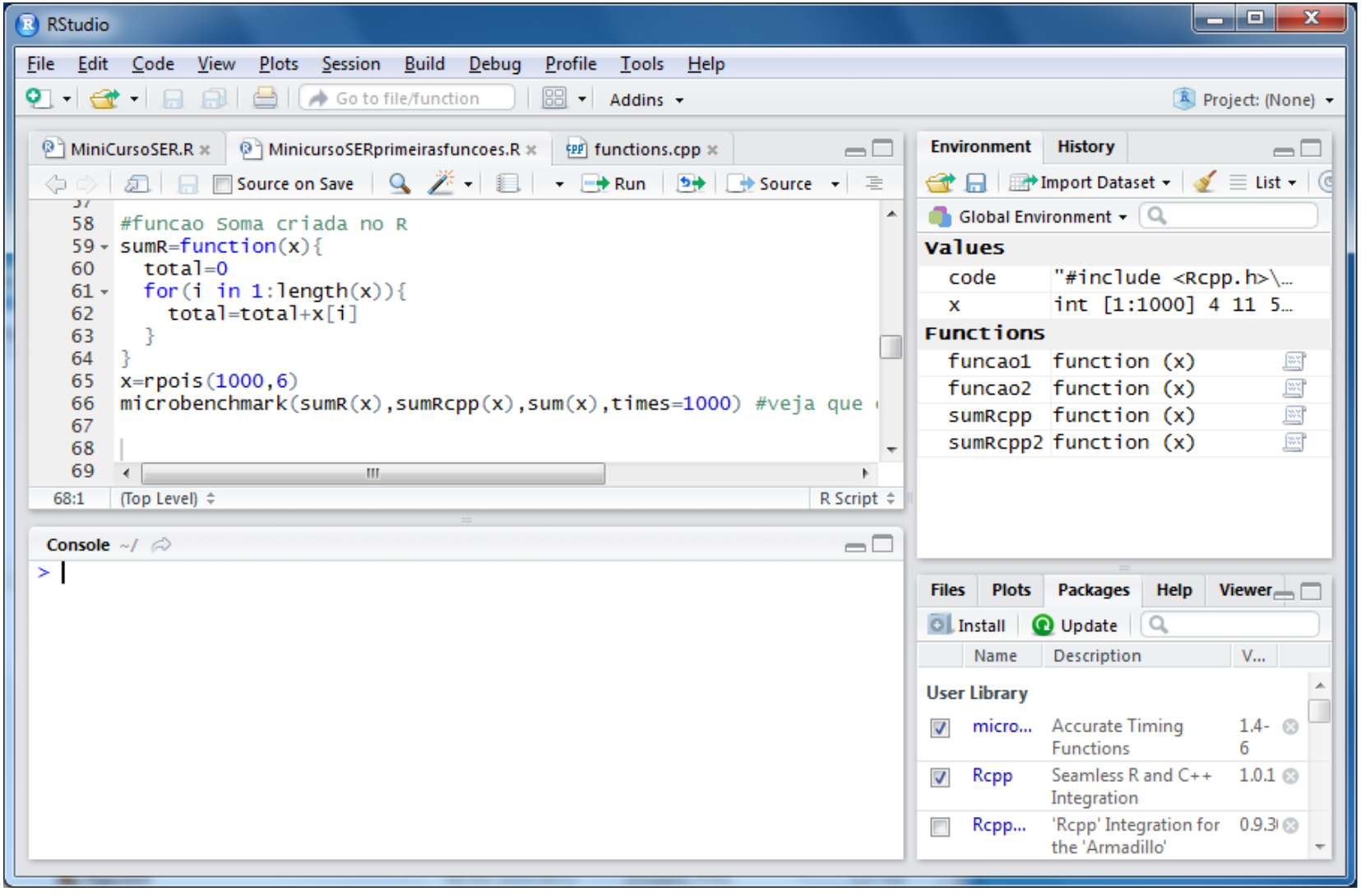


Primeiras Funções

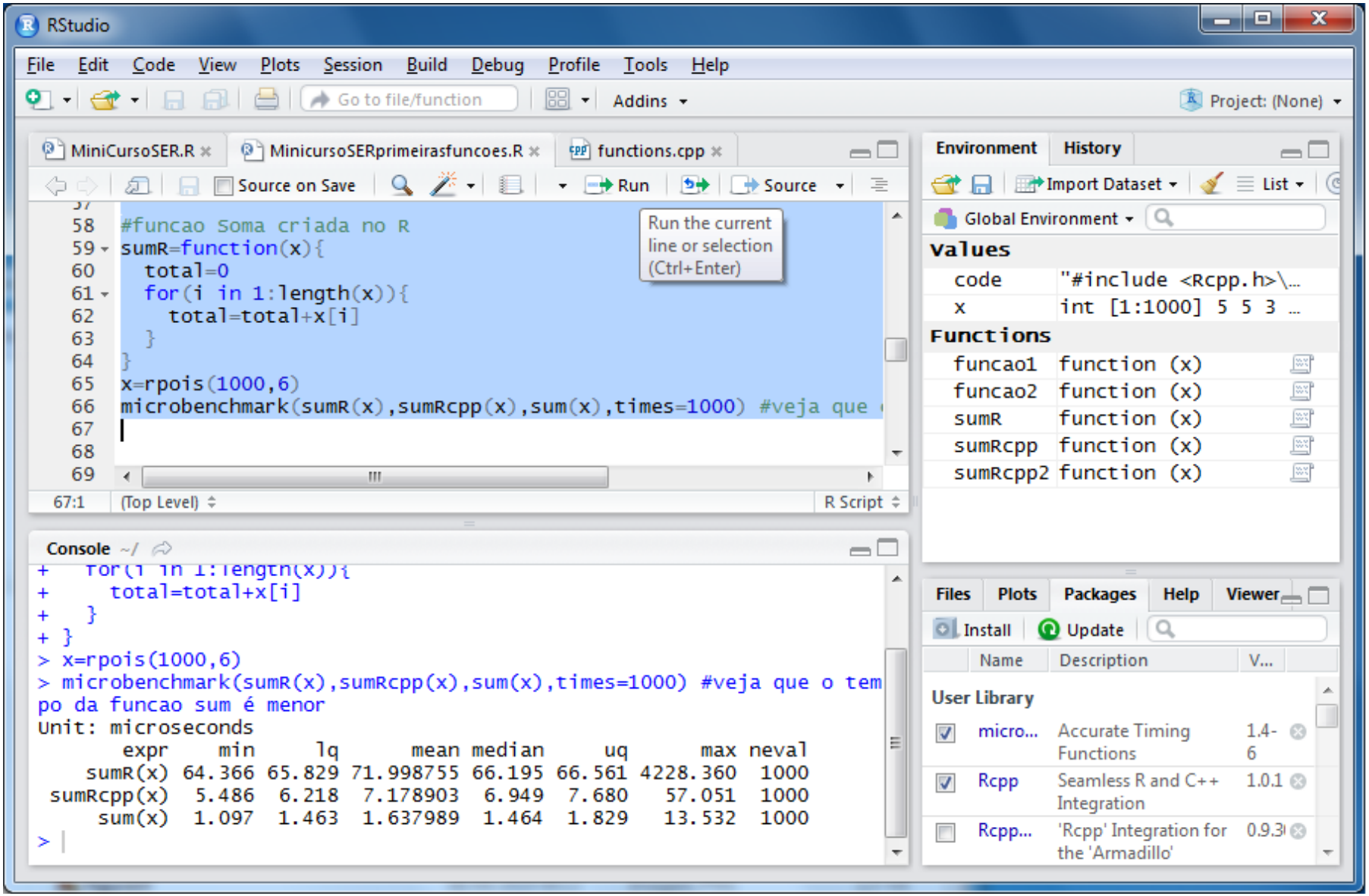


Primeiras Funções

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.

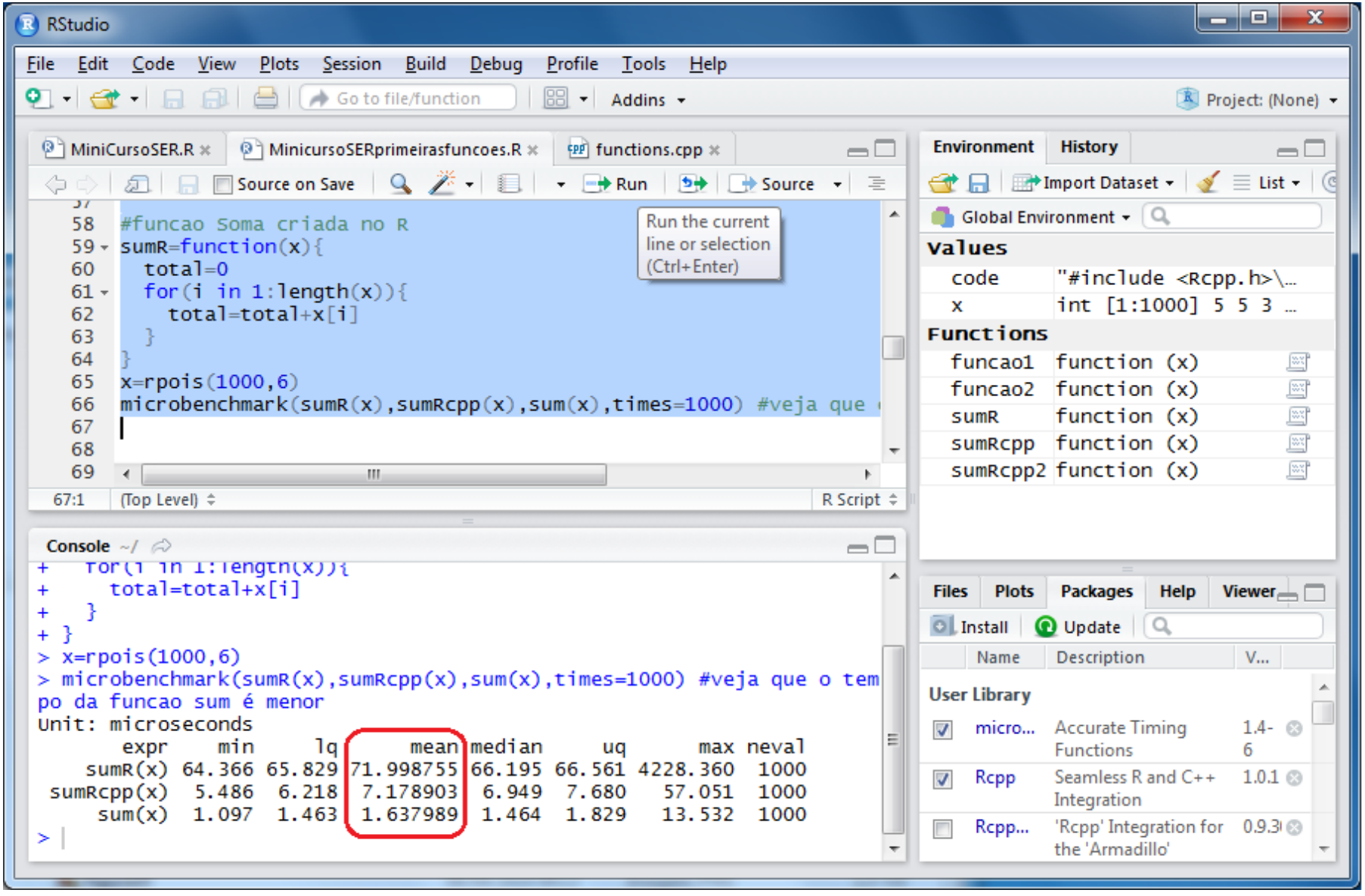


Primeiras Funções

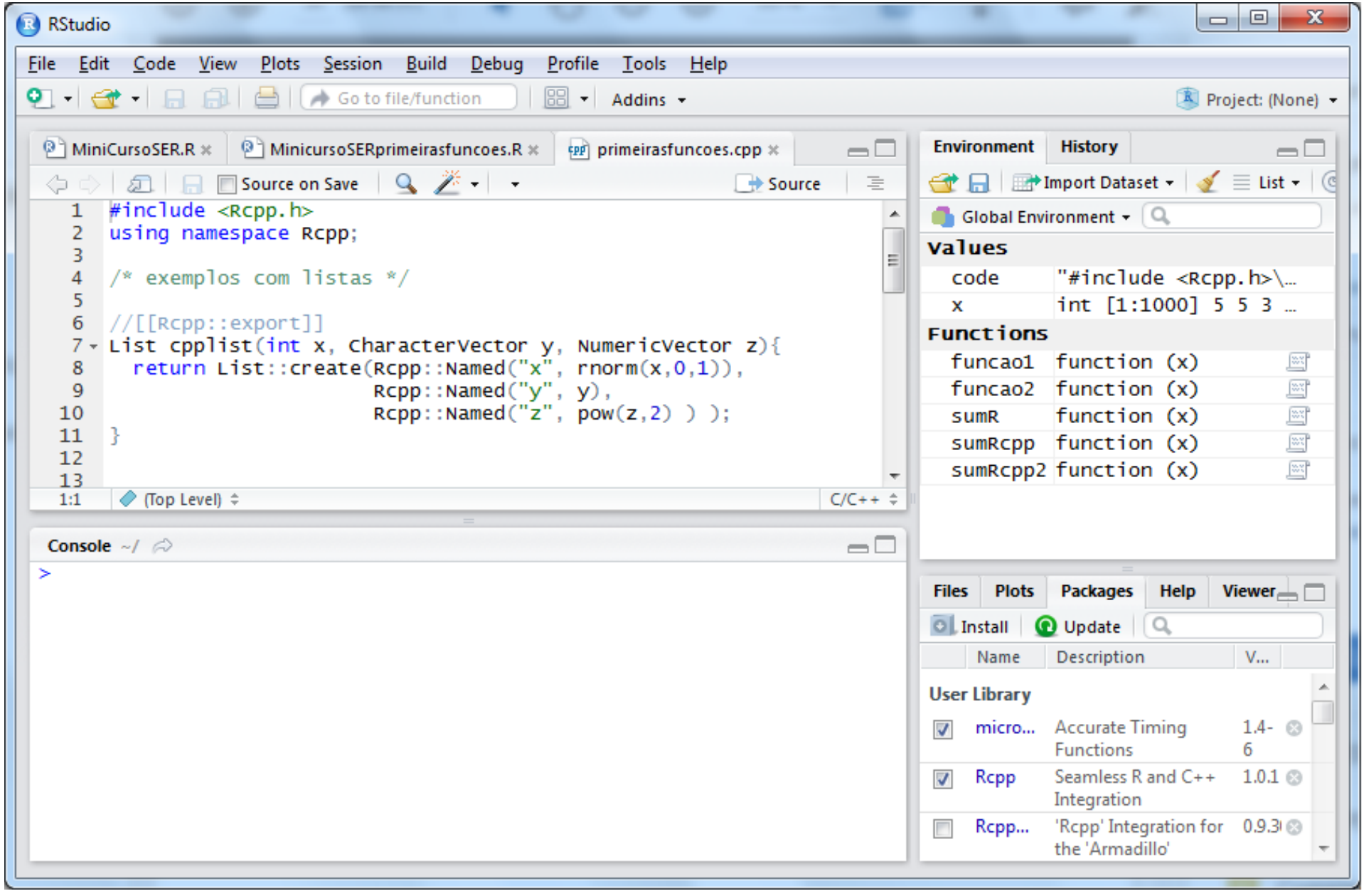


Primeiras Funções

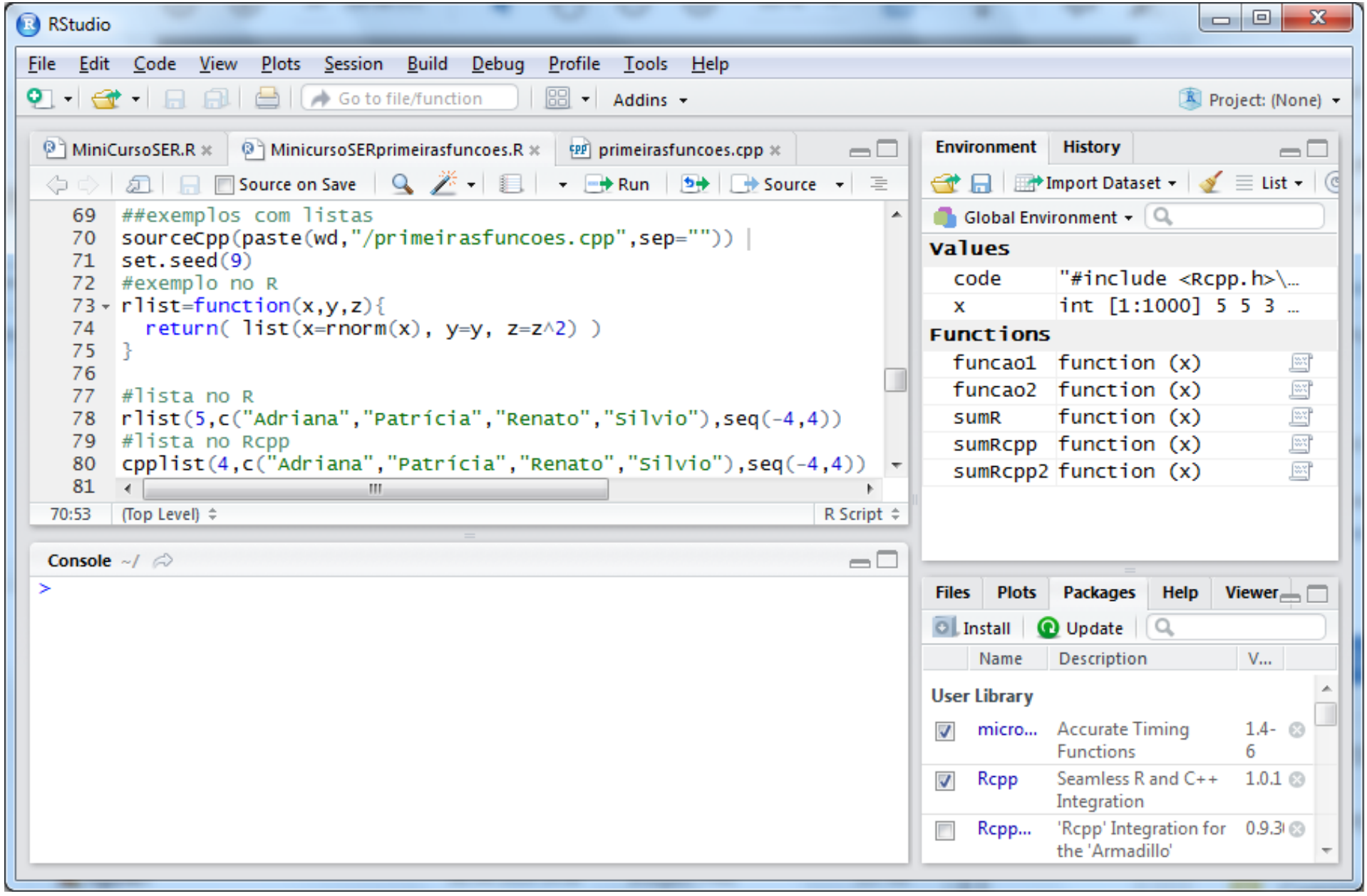
O objetivo não é sair do R!!



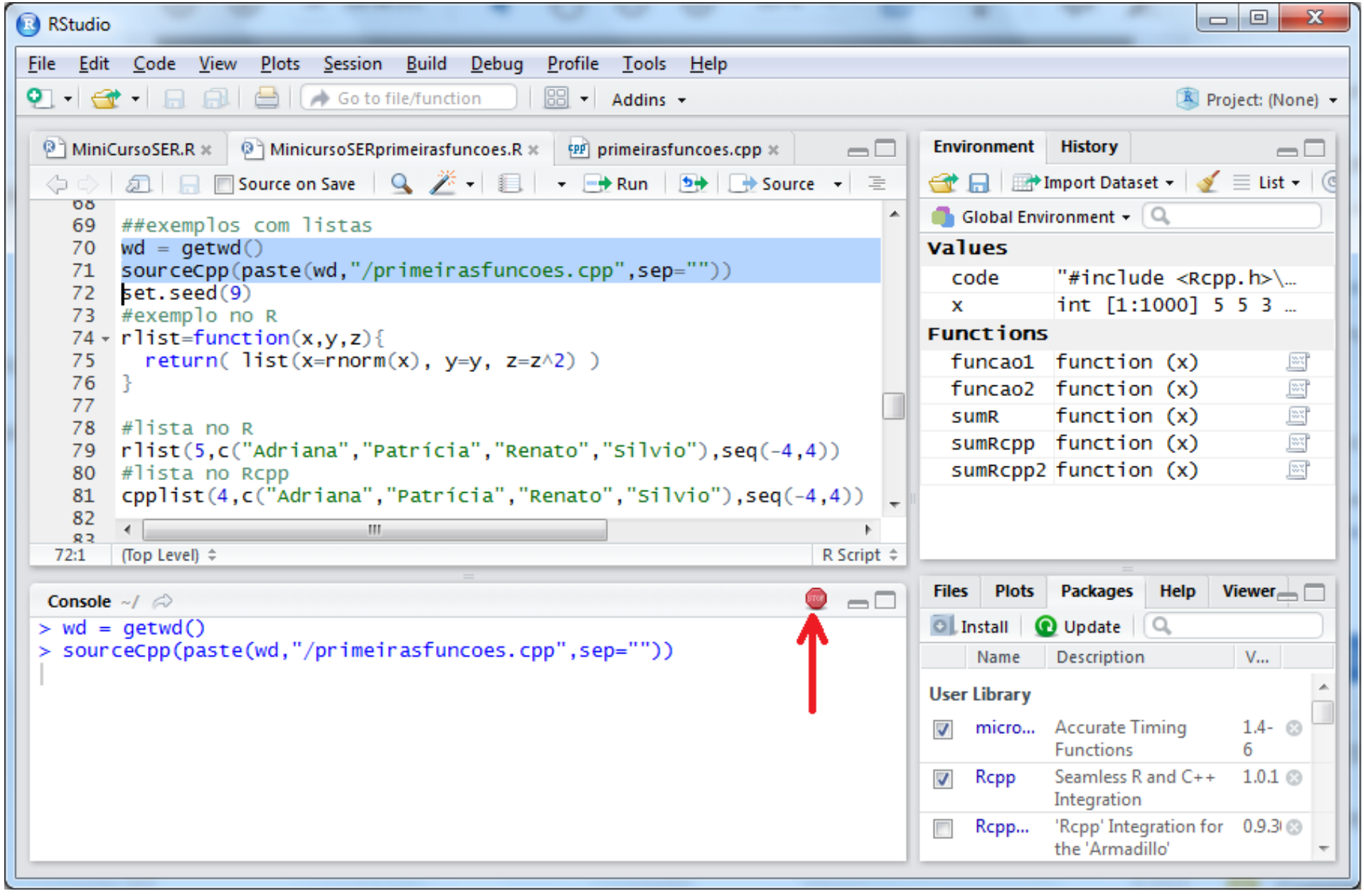
Exemplos com listas



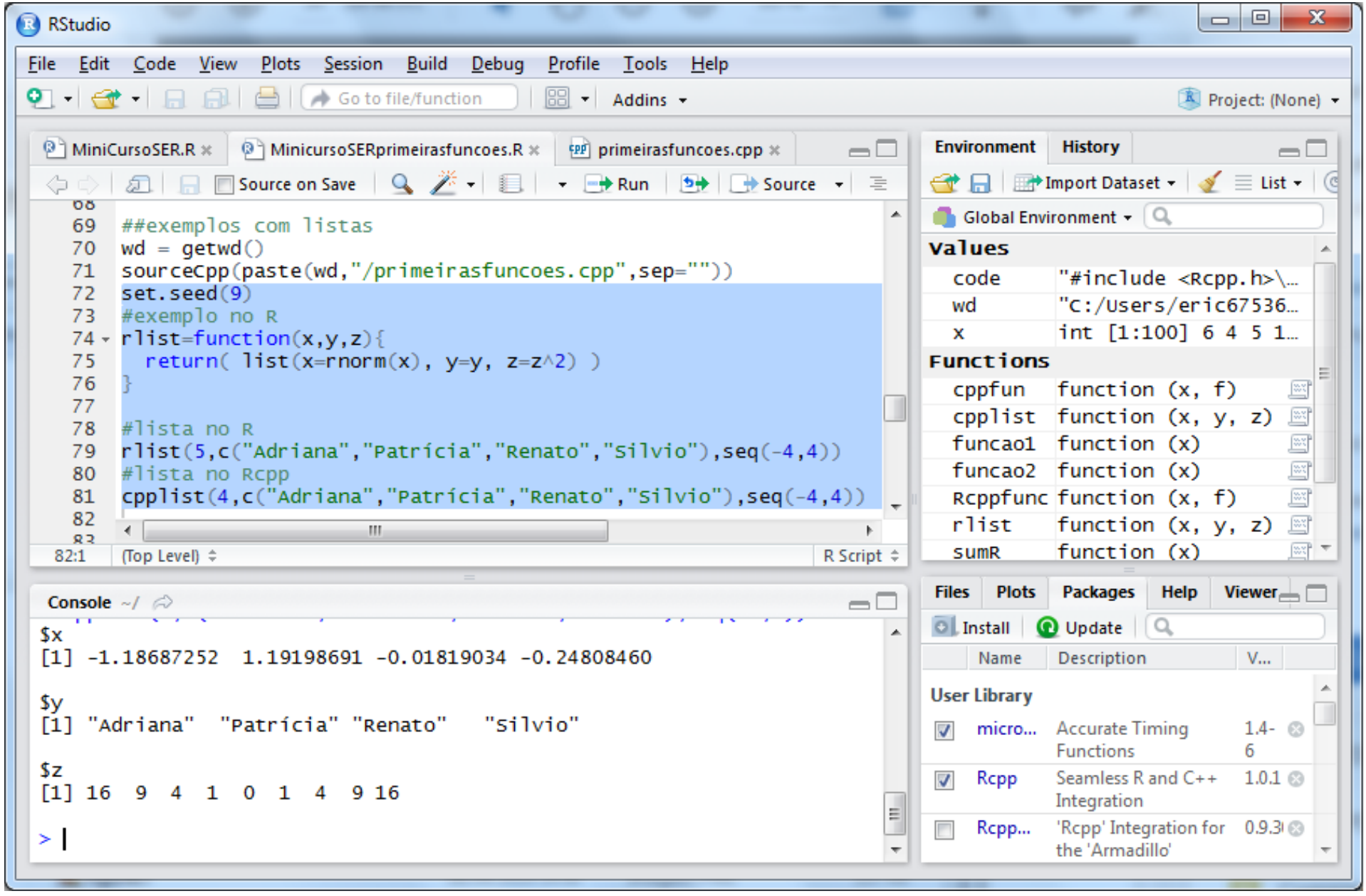
Exemplos com listas



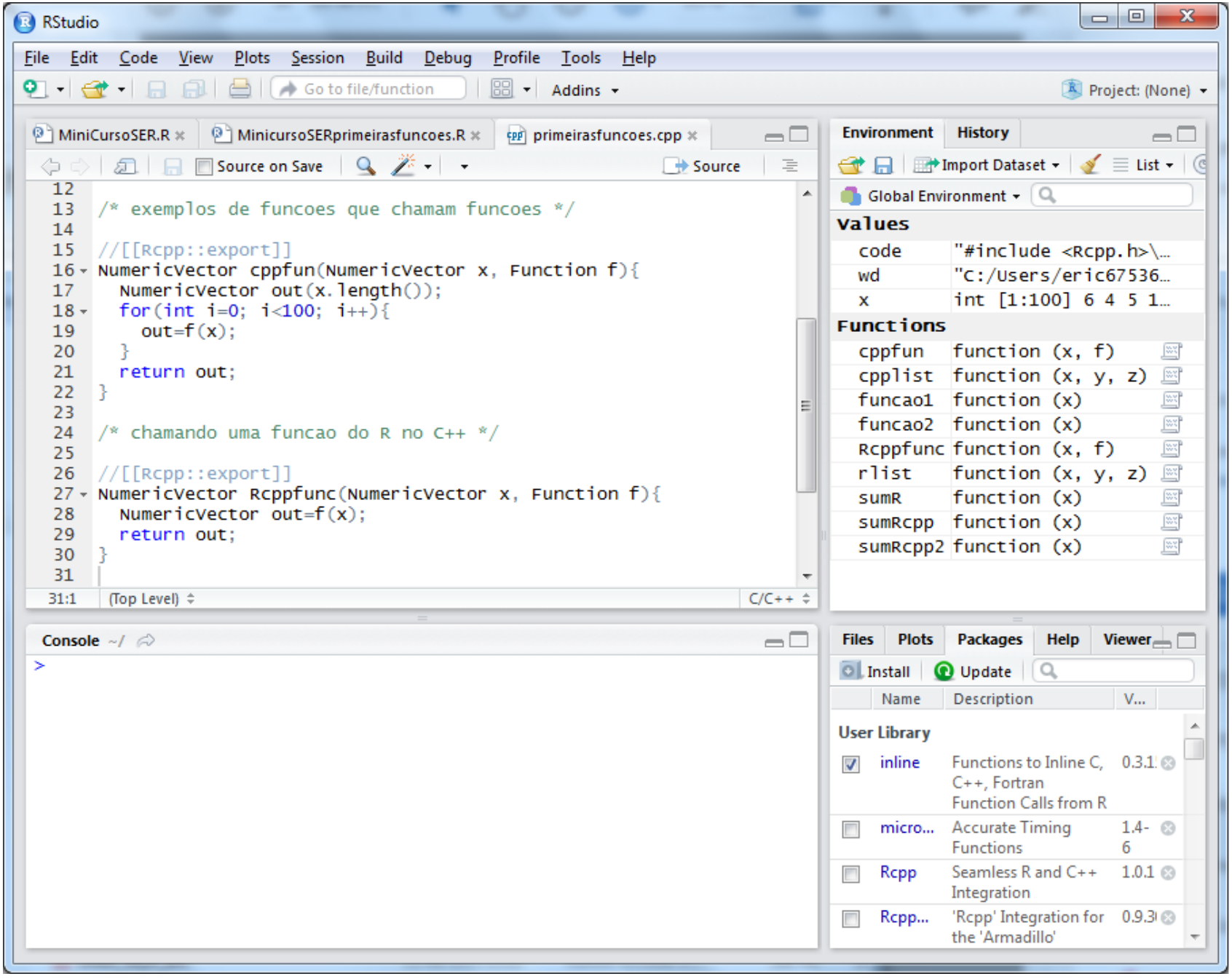
Exemplos com listas



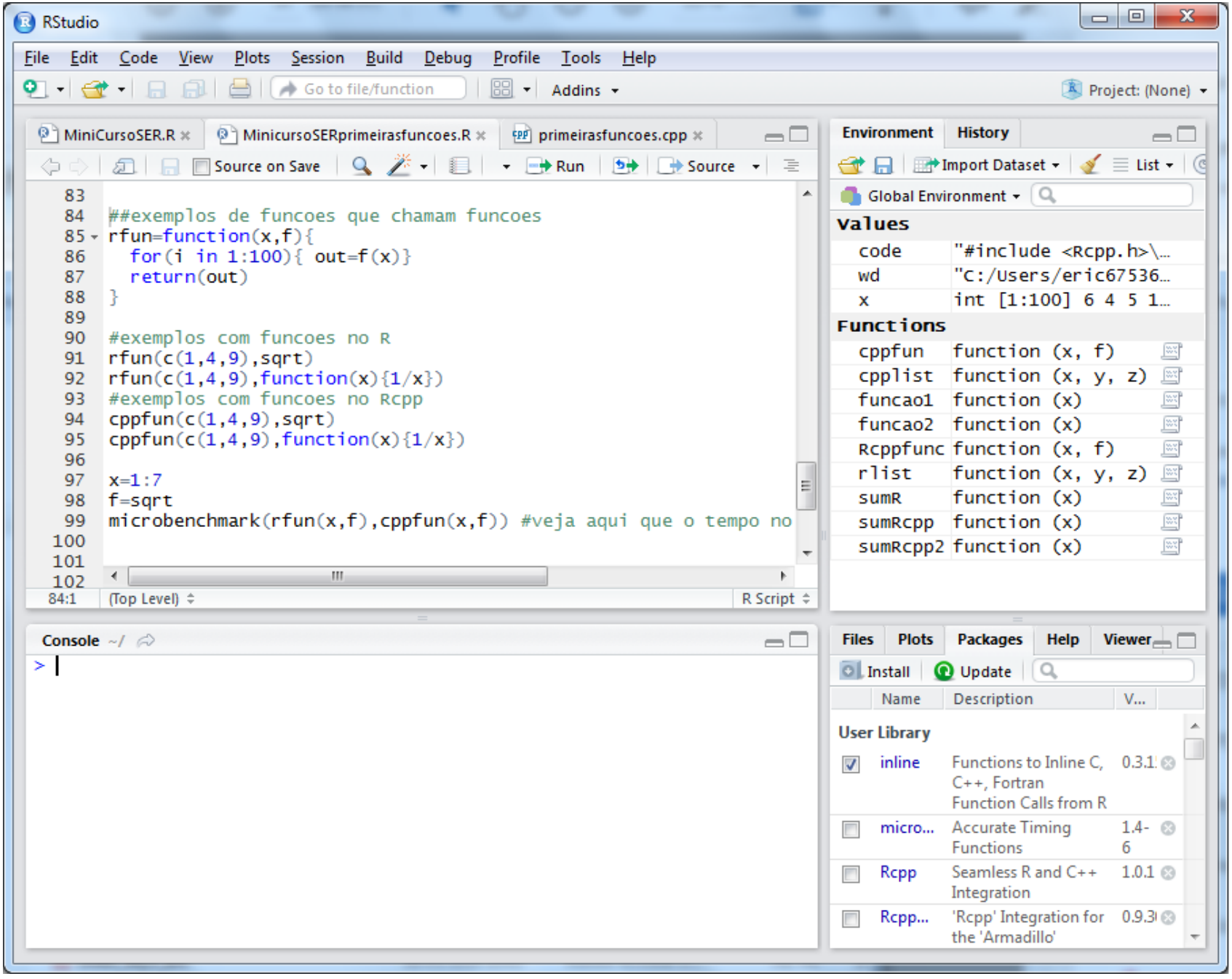
Exemplos com listas



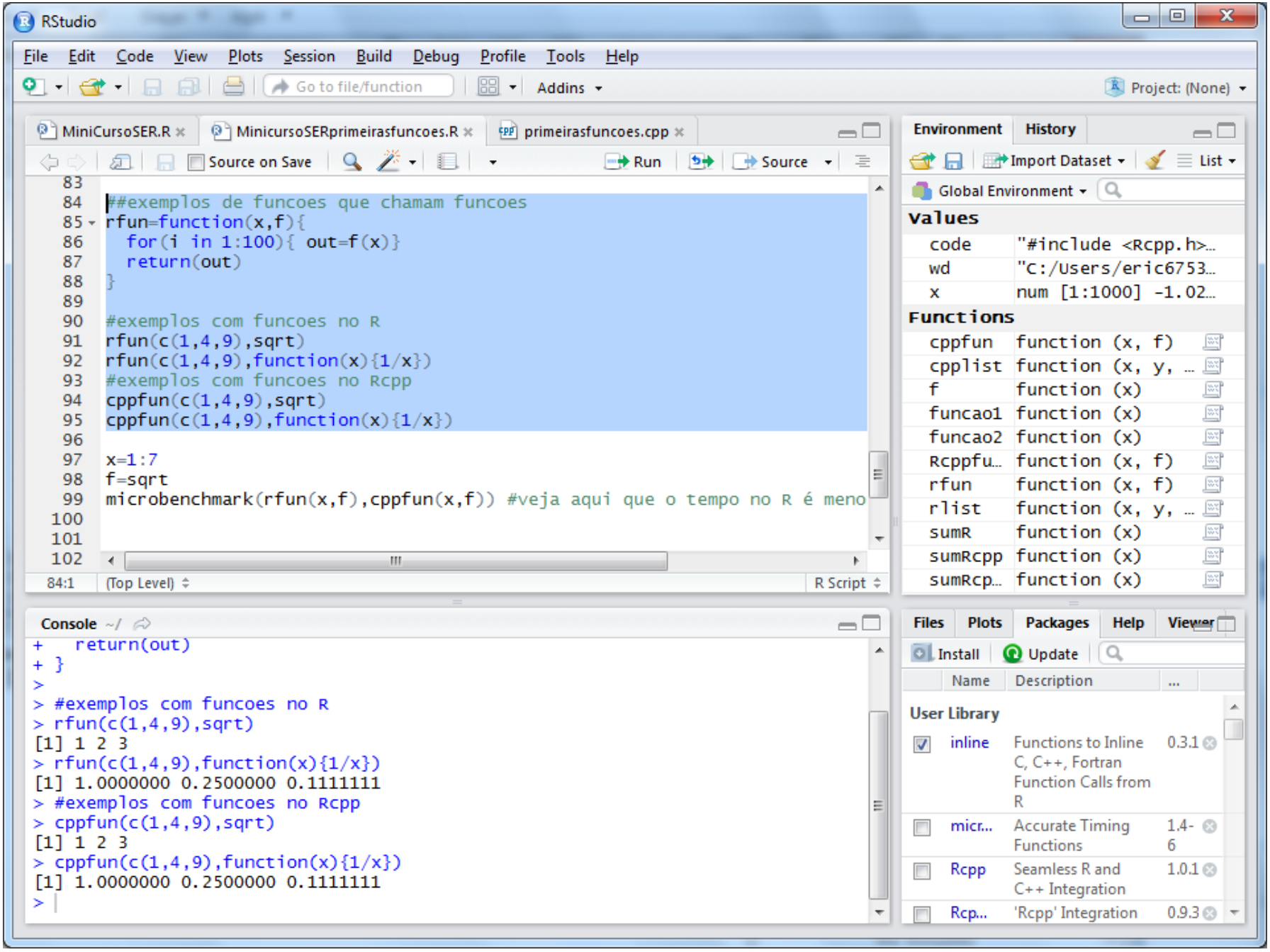
Exemplos de Funções que chamam funções



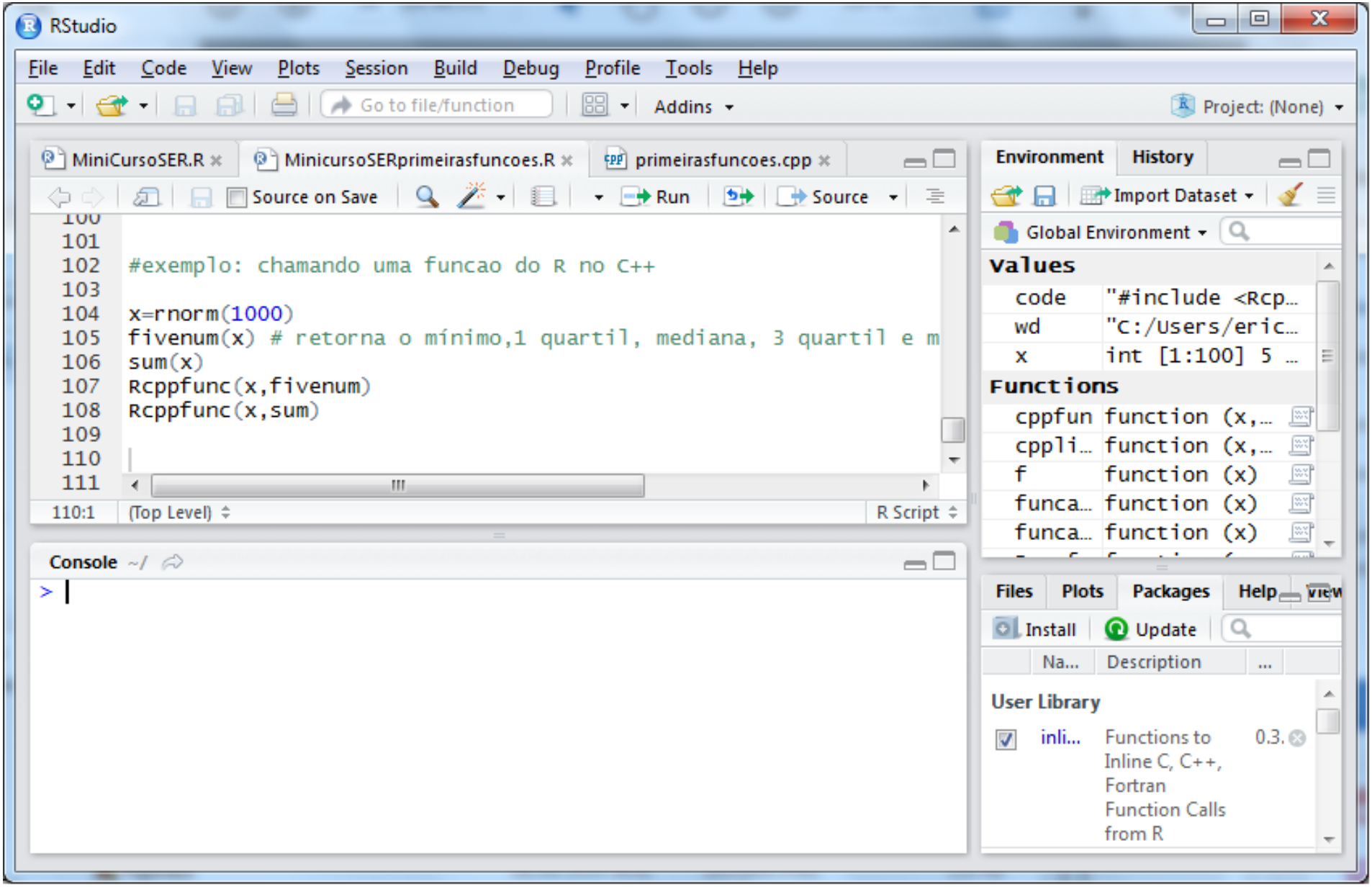
Exemplos de Funções que chamam funções



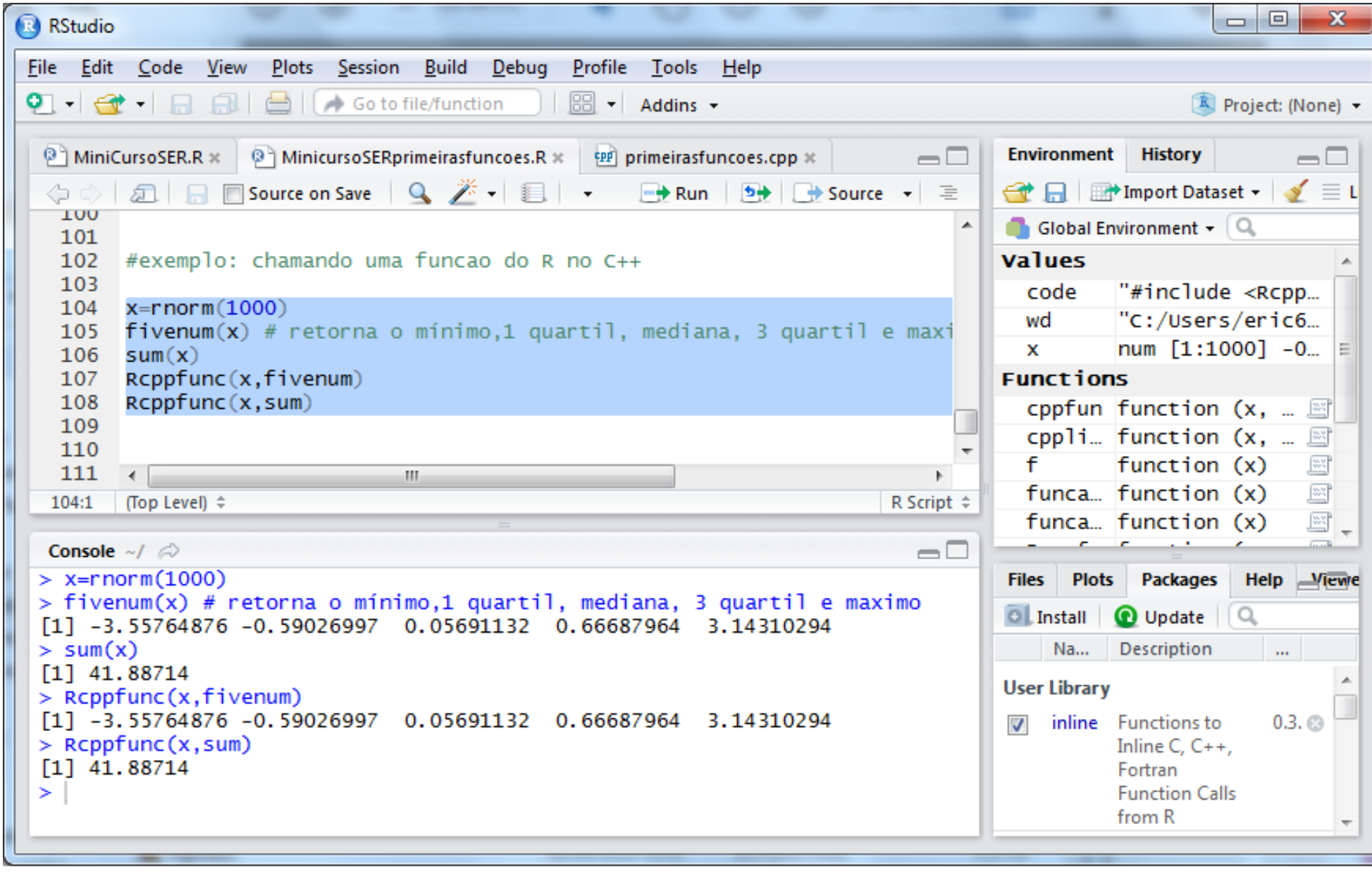
Exemplos de Funções que chamam funções



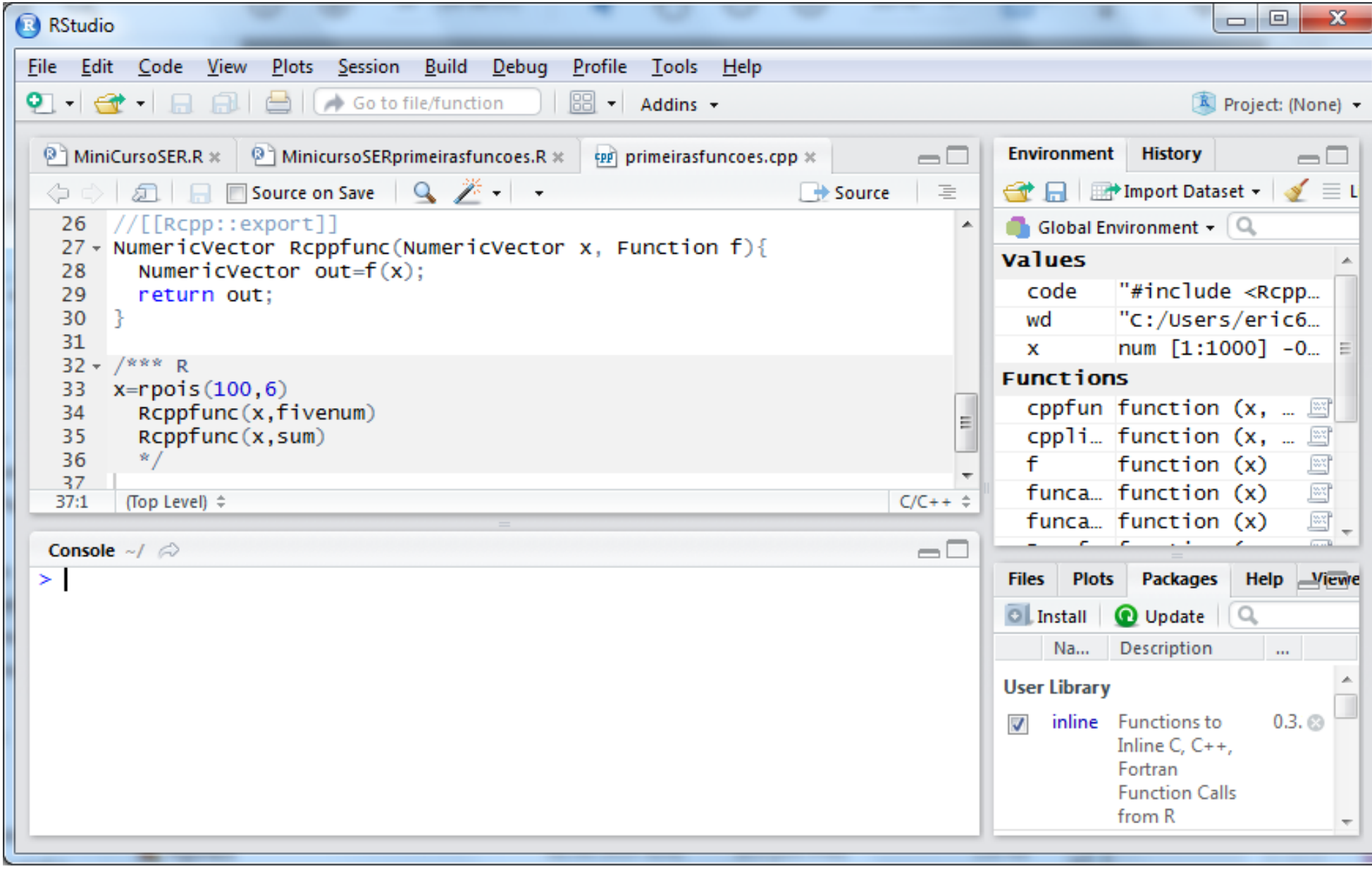
Exemplos de Funções que chamam funções



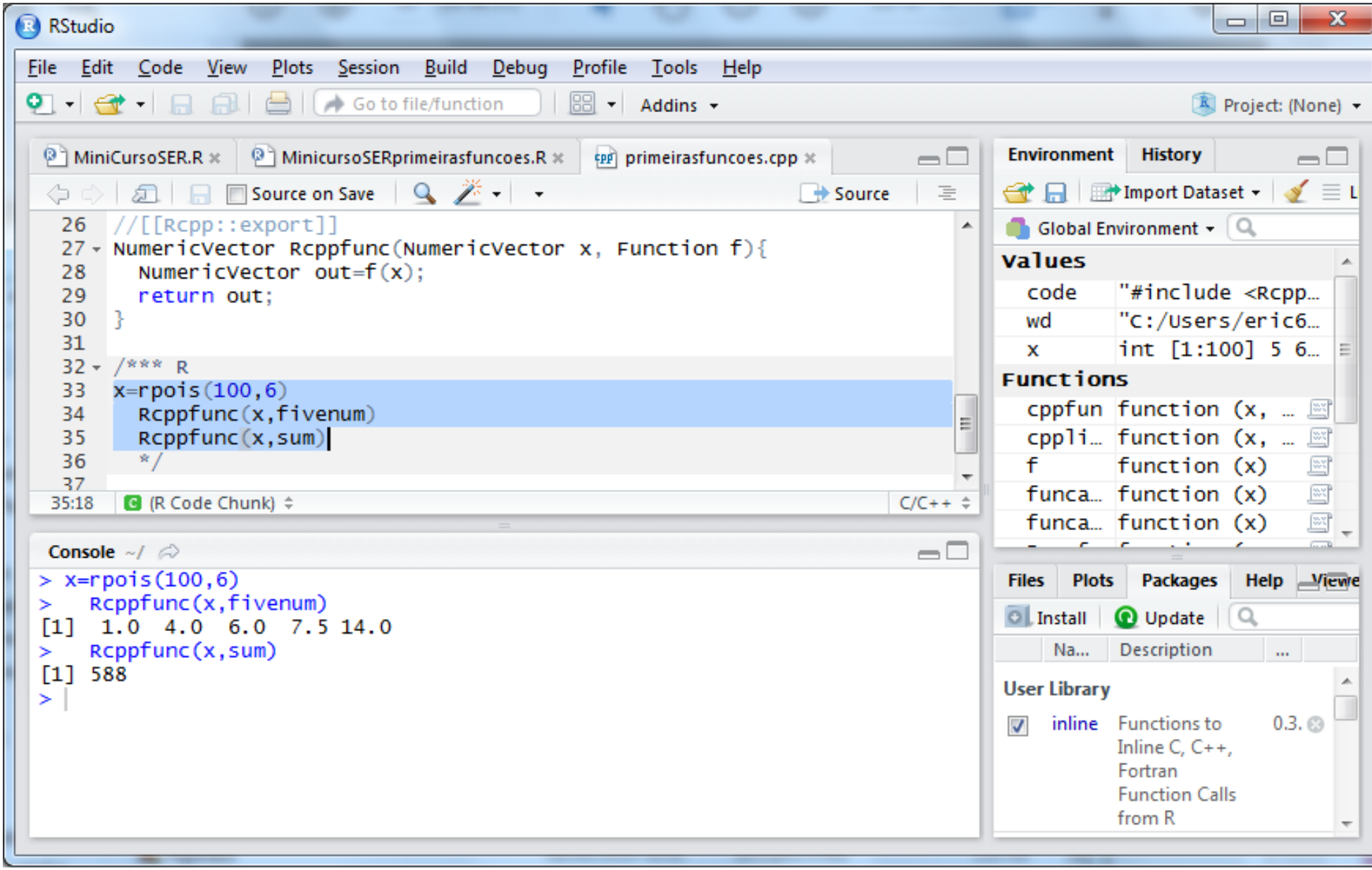
Exemplos de Funções que chamam funções



Exemplos de Funções que chamam funções



Exemplos de Funções que chamam funções



Exercícios

Armadillo

C++ library for linear algebra & scientific computing

Álgebra Linear

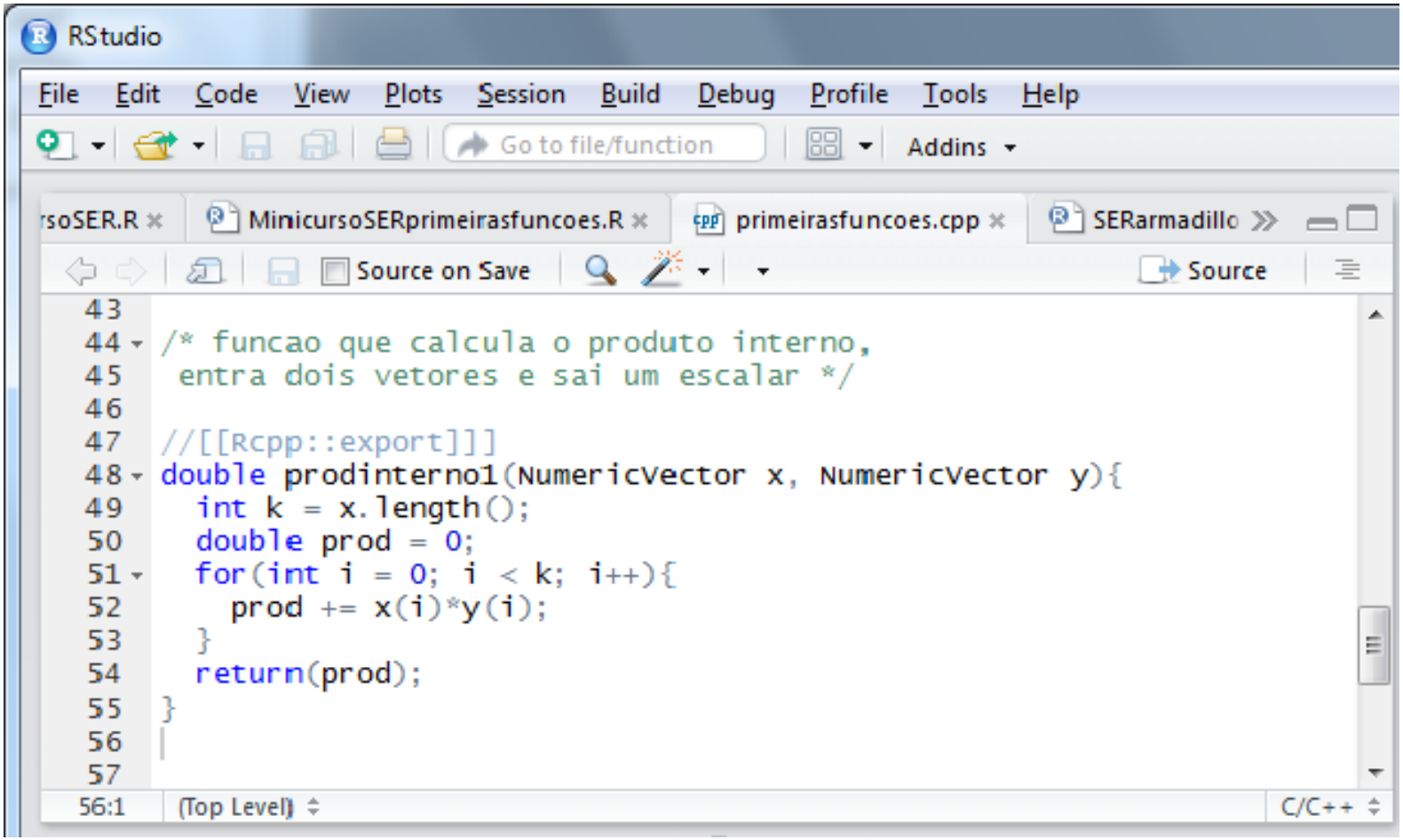
Estudos envolvendo Matrizes tem uma extrema importância para nós.

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.

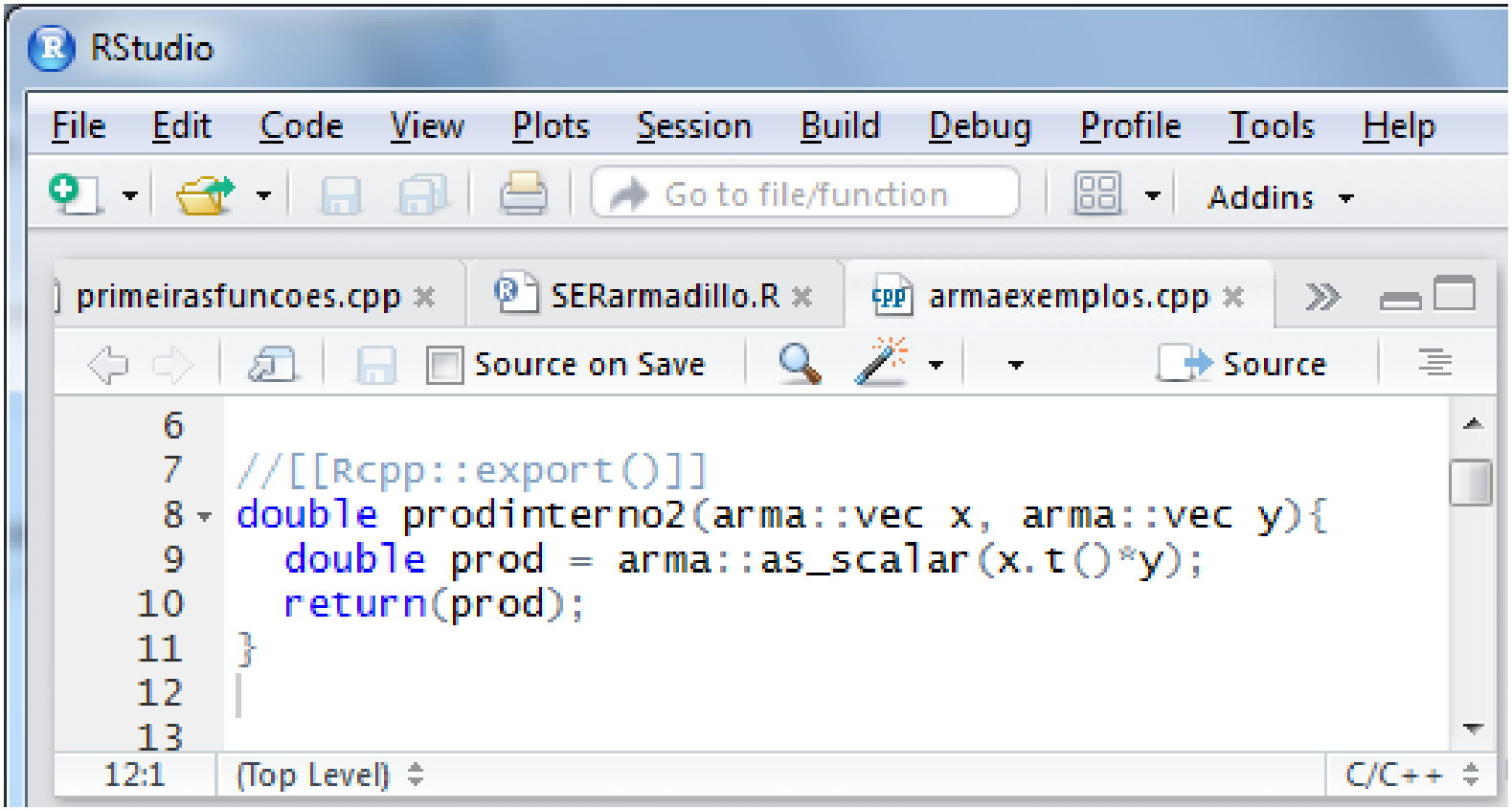
Álgebra Linear

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



Álgebra Linear no RcppArmadillo

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



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.

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

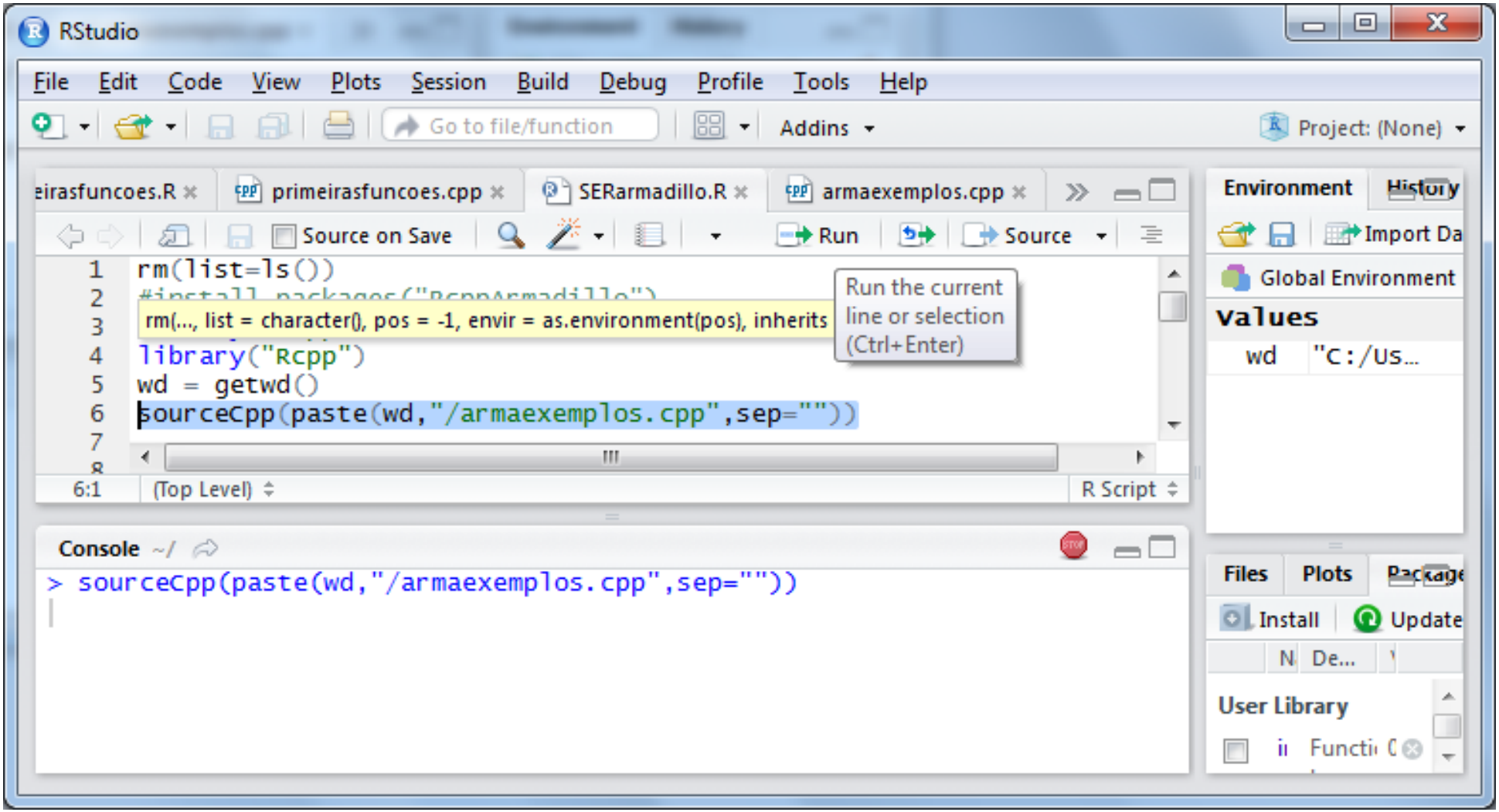
O que temos agora

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

O `#include<Rcpp.h>` fica implícido.

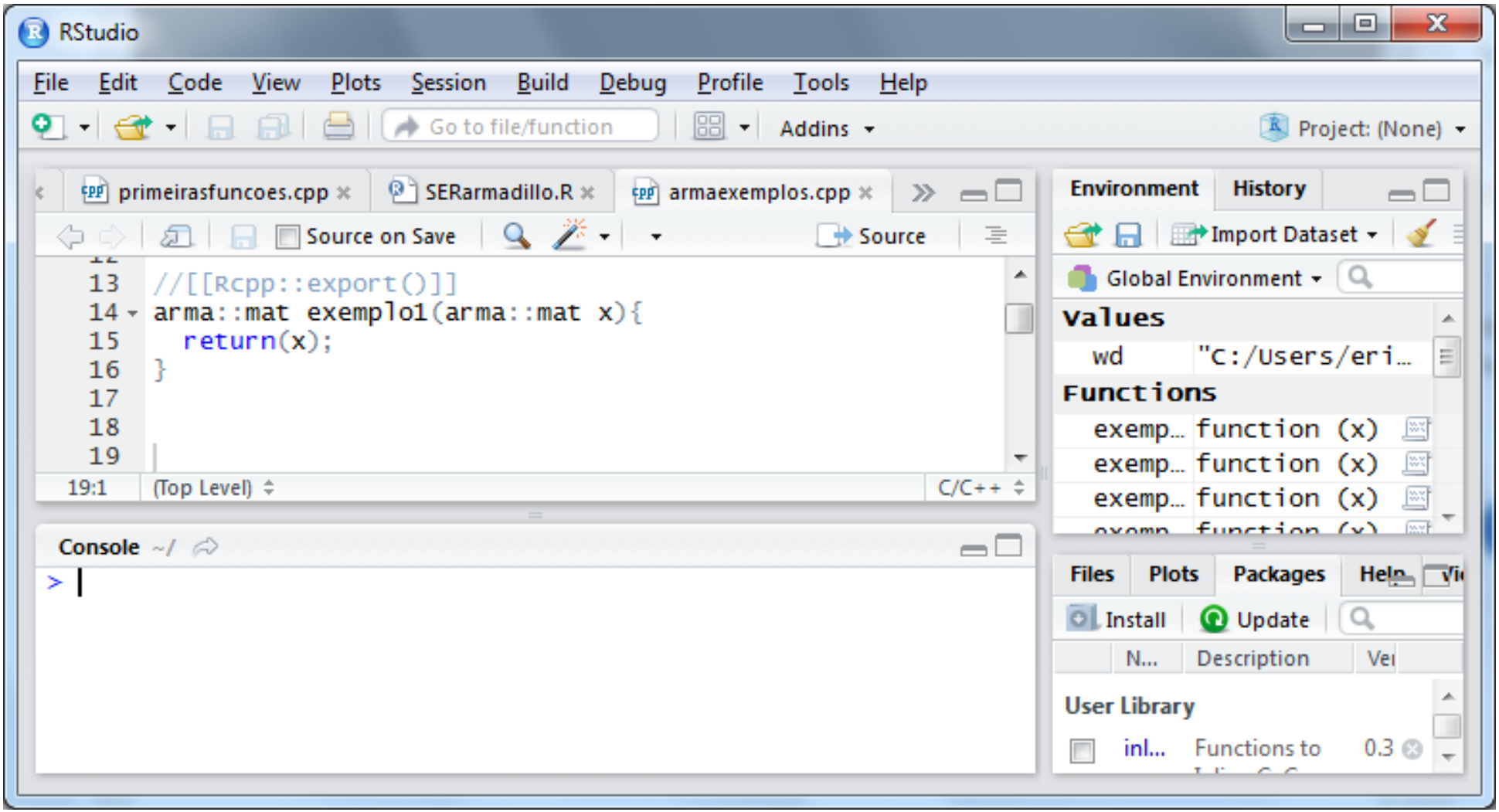
Para melhor entendimento vamos a alguns exemplos básicos.

Exemplos básicos da estrutura de dados



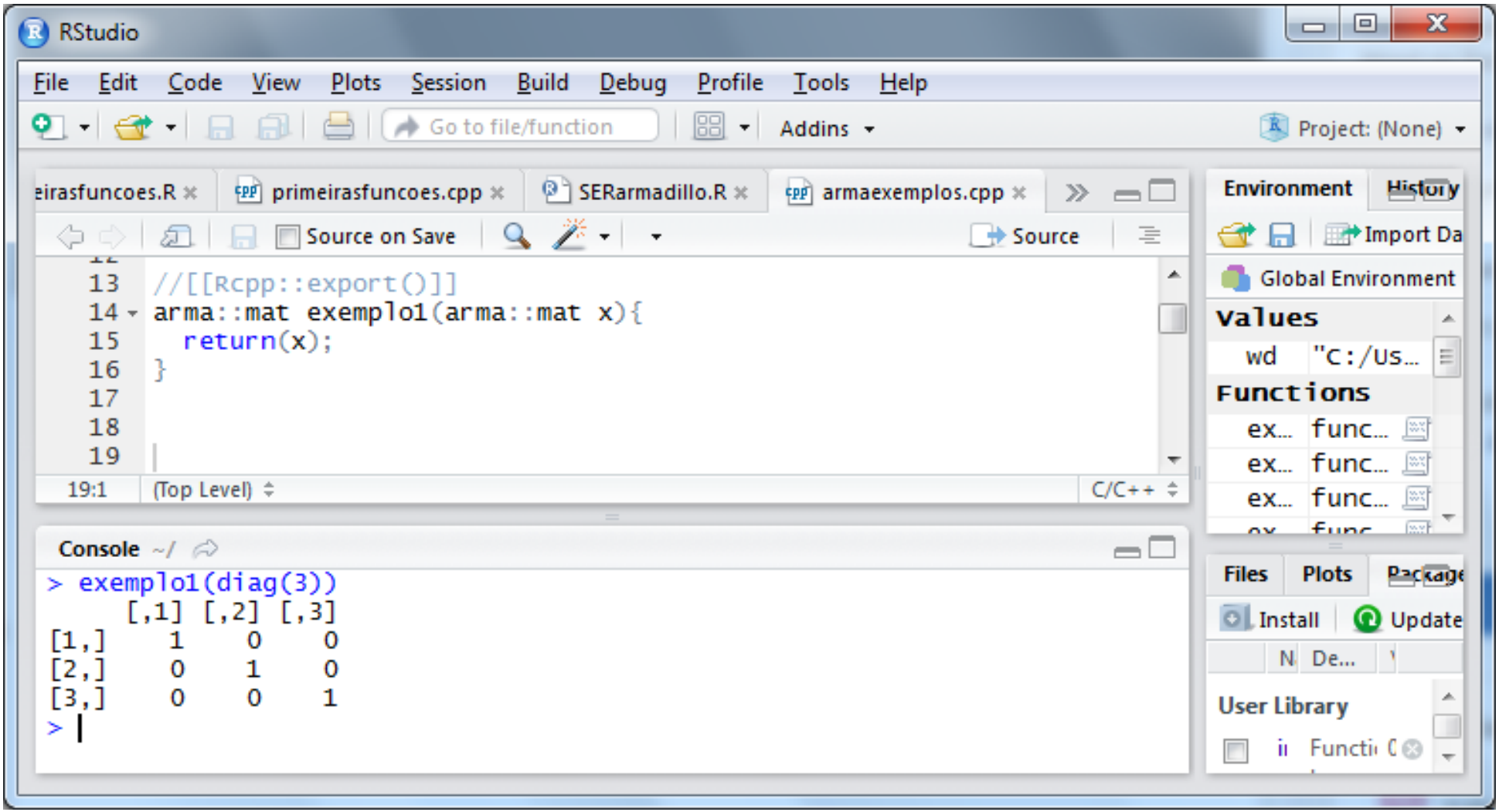
Exemplos básicos da estrutura de dados

A entrada é uma matriz.



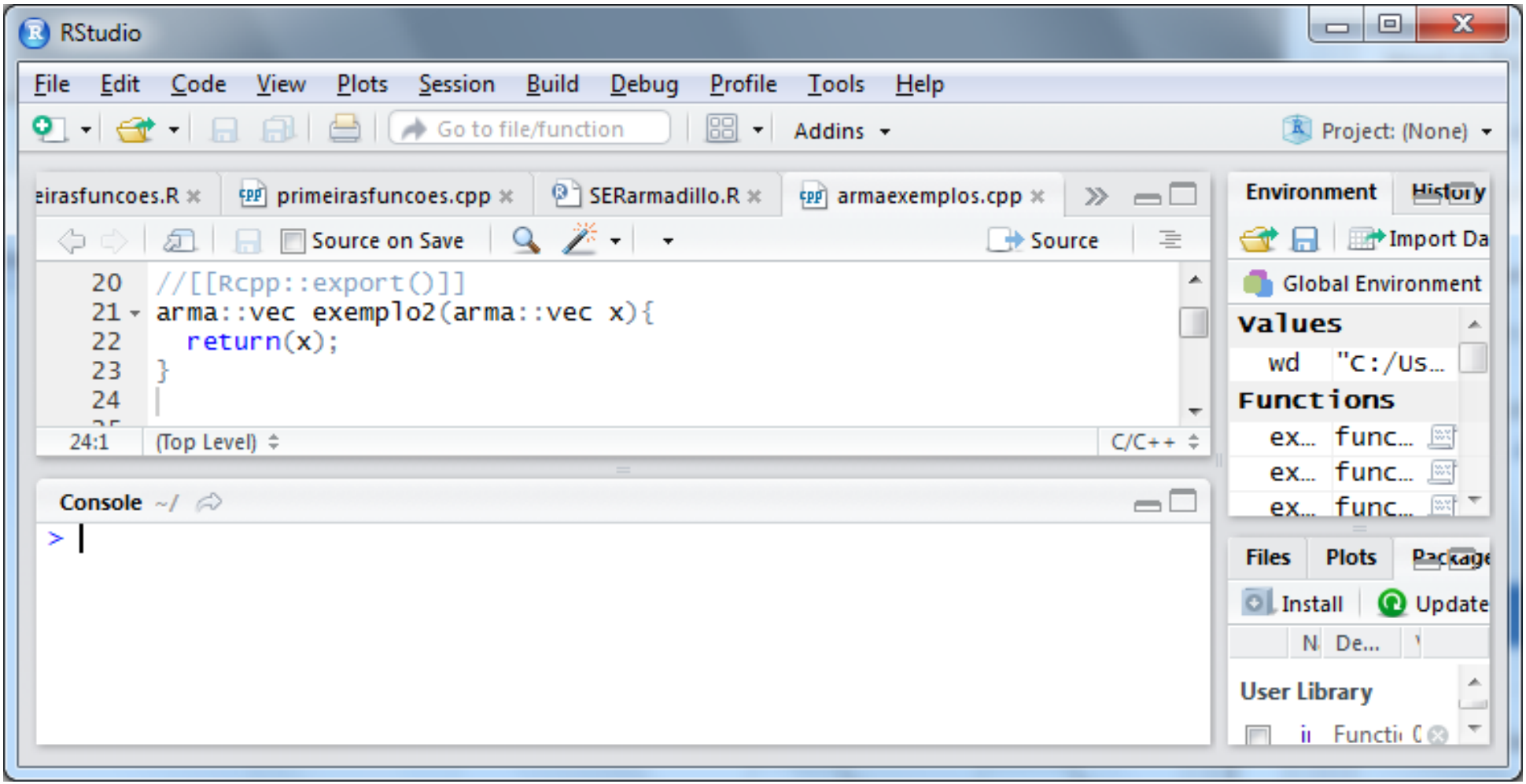
Exemplos básicos da estrutura de dados

A saída é uma matriz.



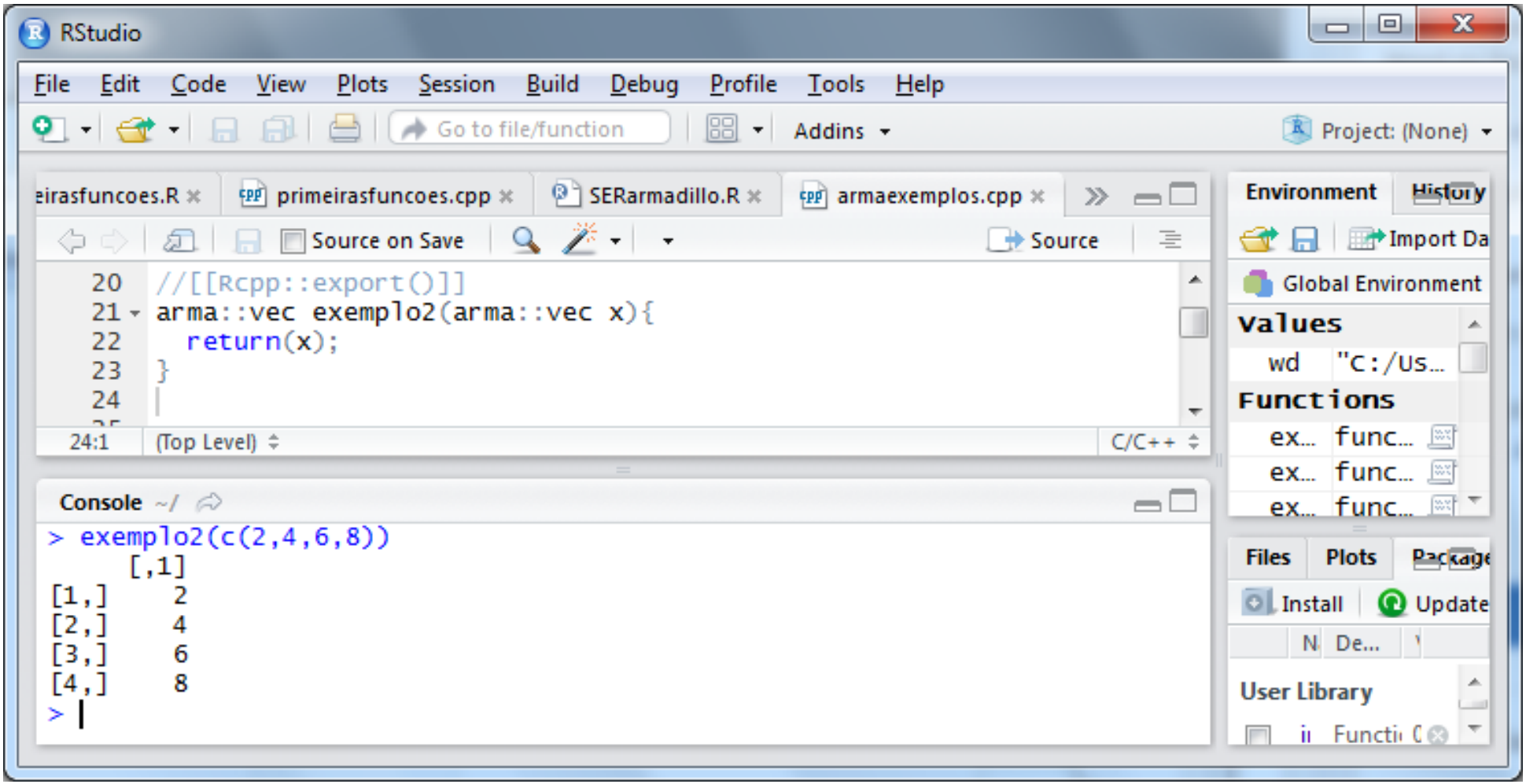
Exemplos básicos da estrutura de dados

A entrada é um vetor.



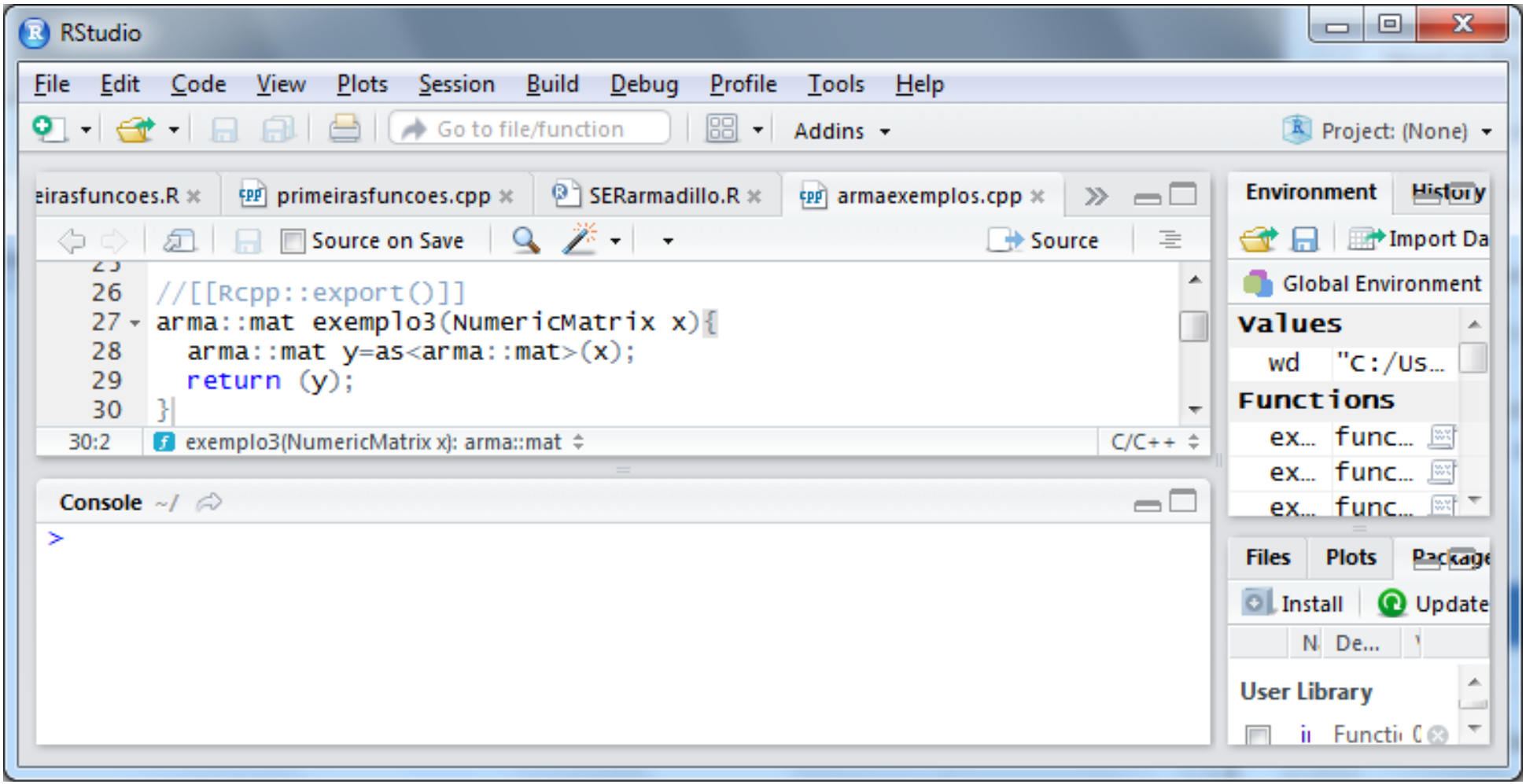
Exemplos básicos da estrutura de dados

A saída é uma matriz.



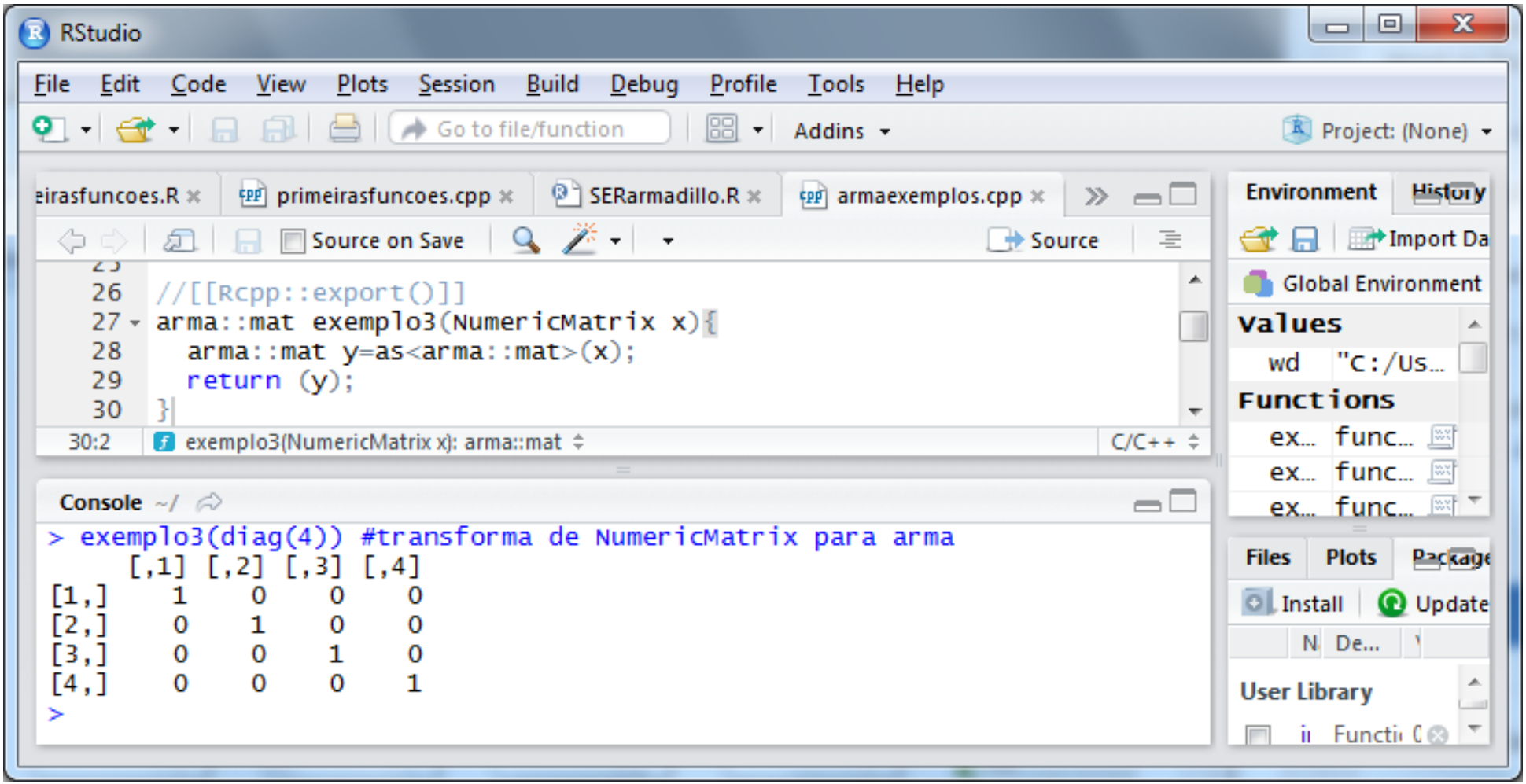
Exemplos básicos da estrutura de dados

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



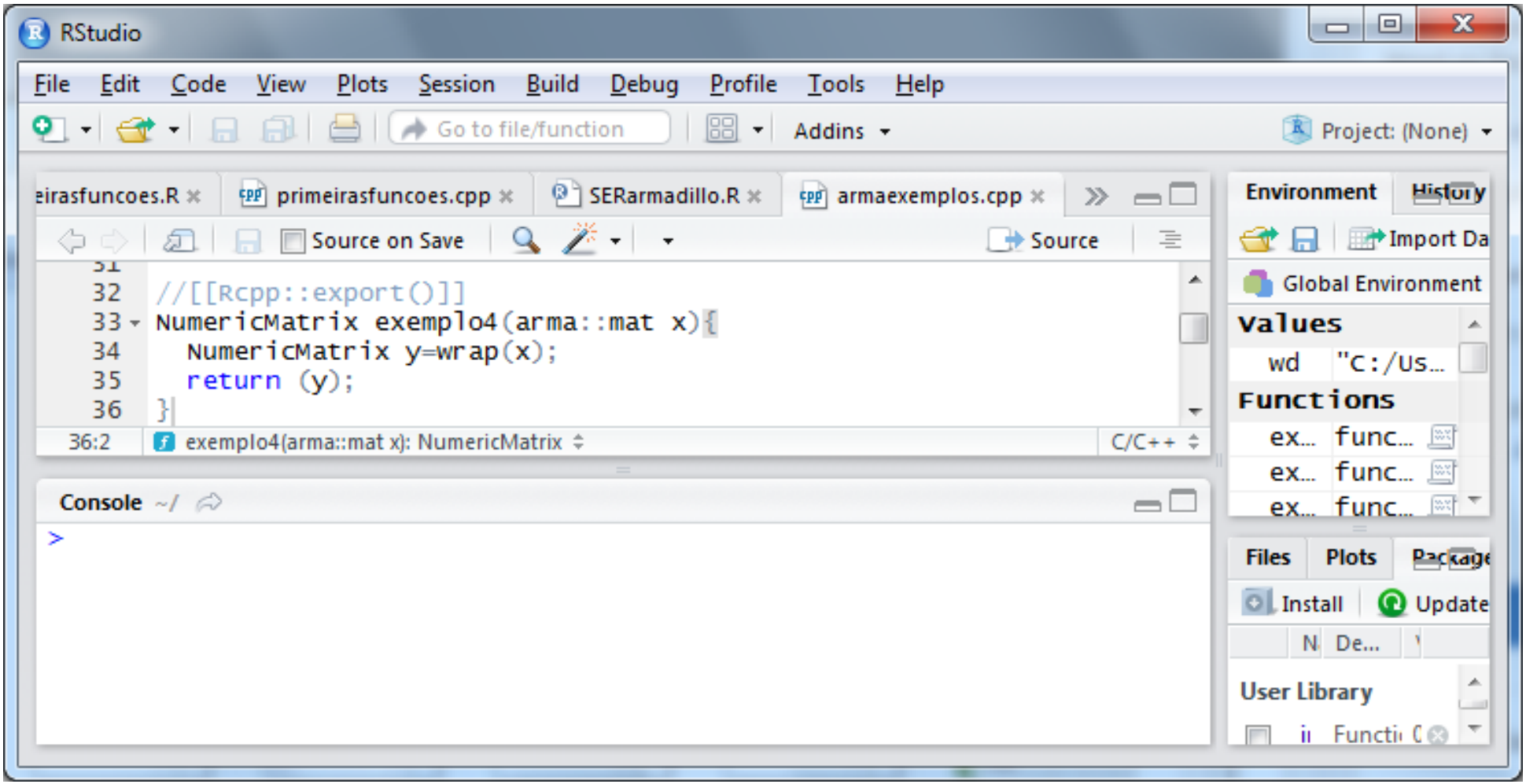
Exemplos básicos da estrutura de dados

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



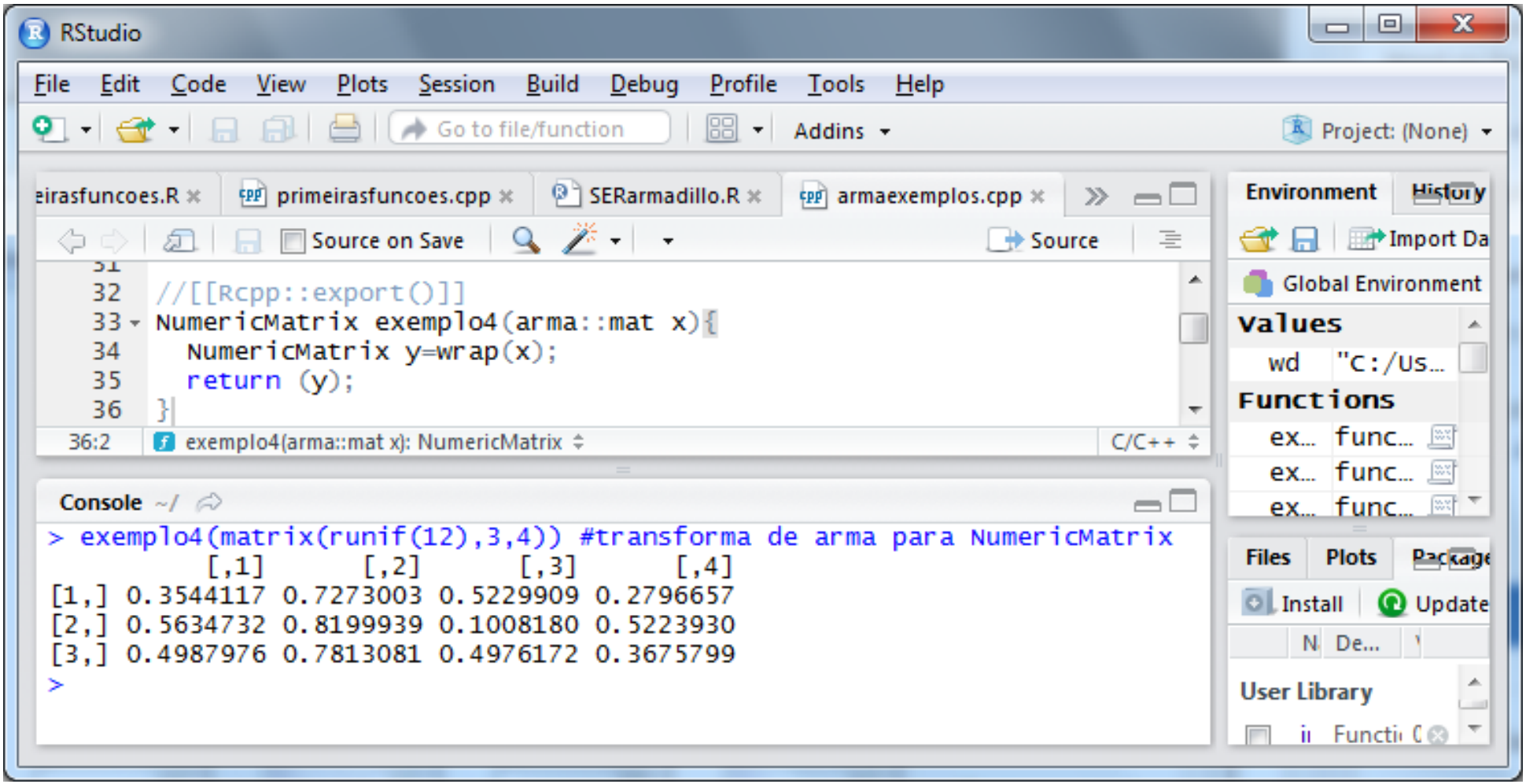
Exemplos básicos da estrutura de dados

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



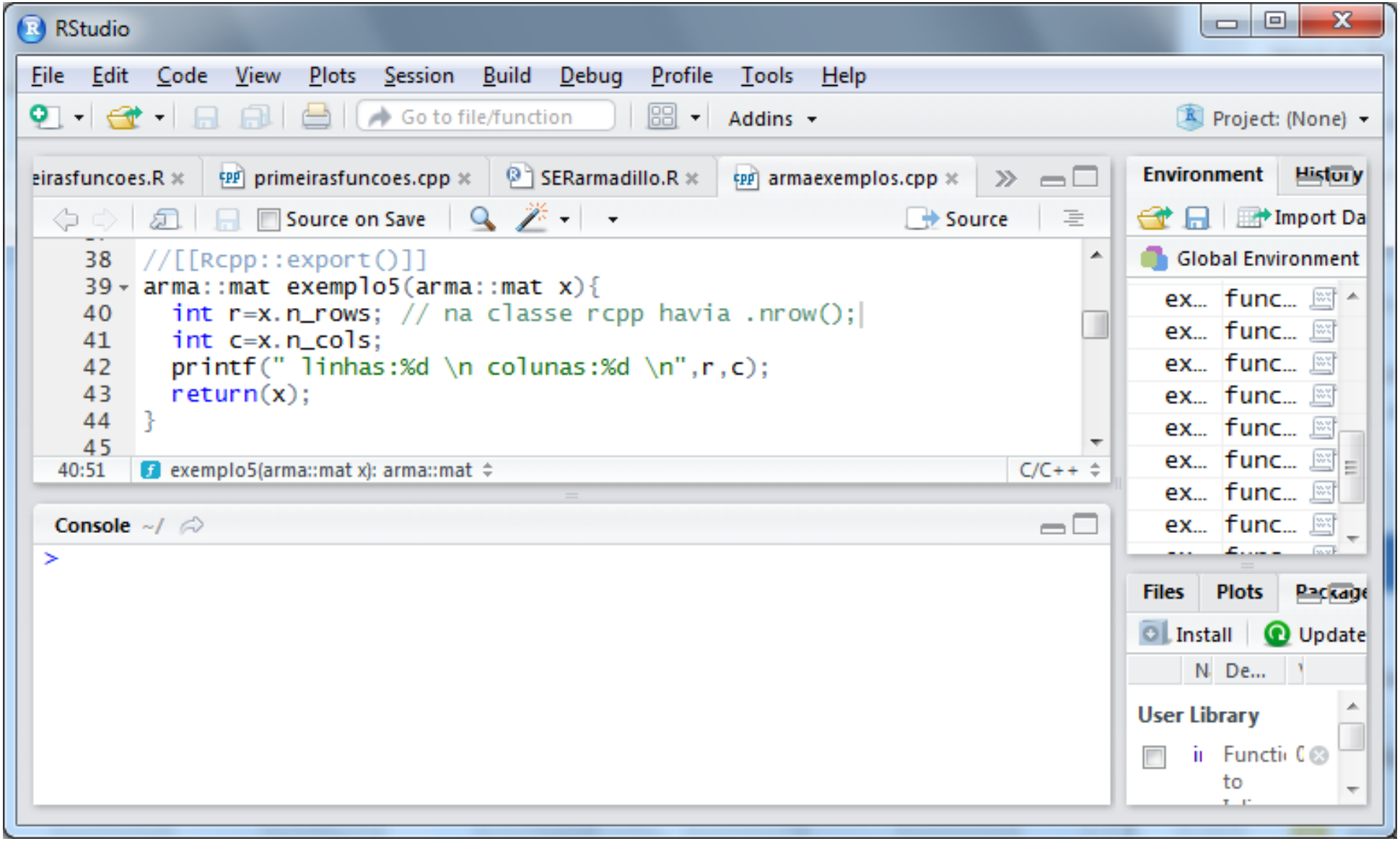
Exemplos básicos da estrutura de dados

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



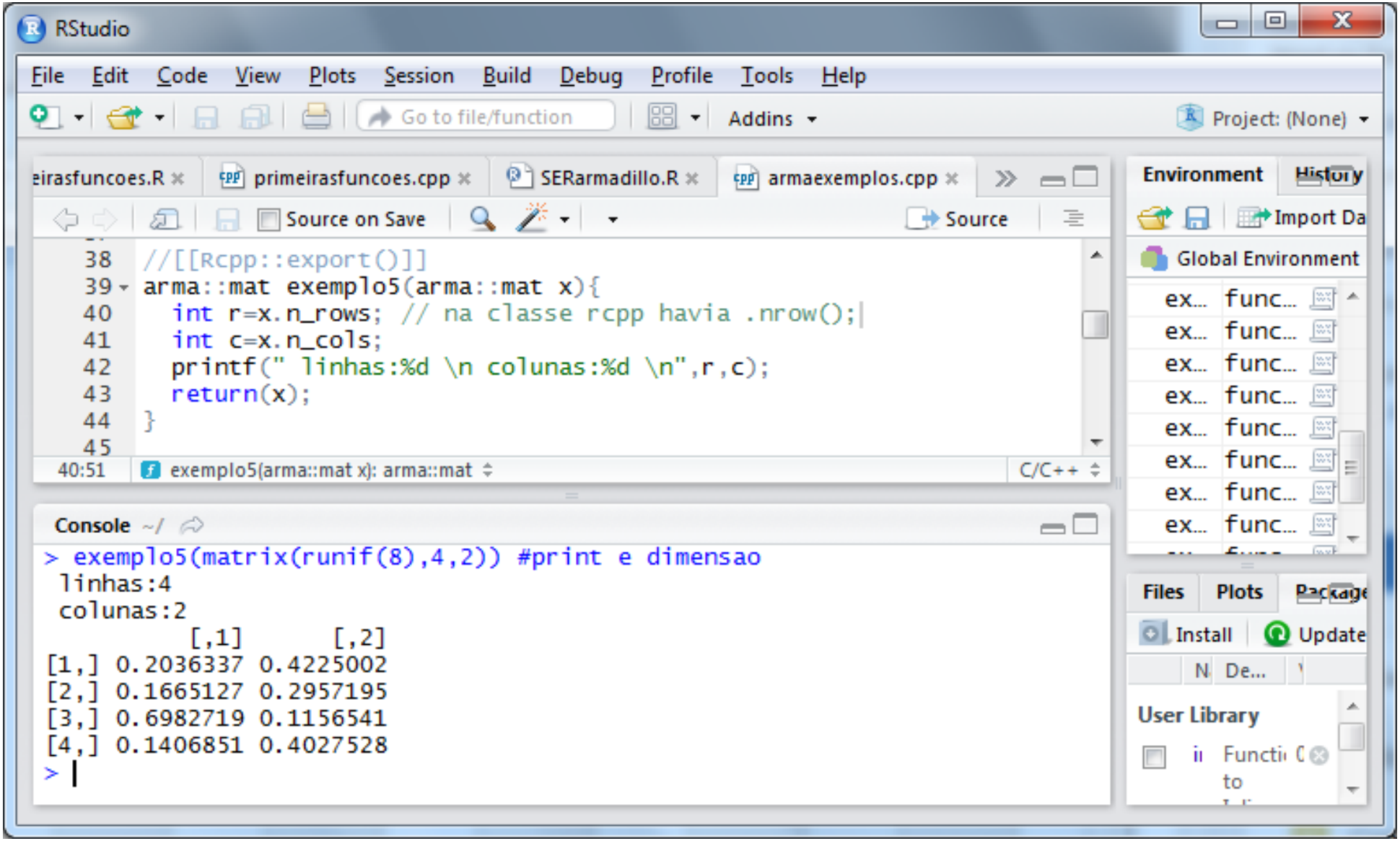
Exemplos básicos da estrutura de dados

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



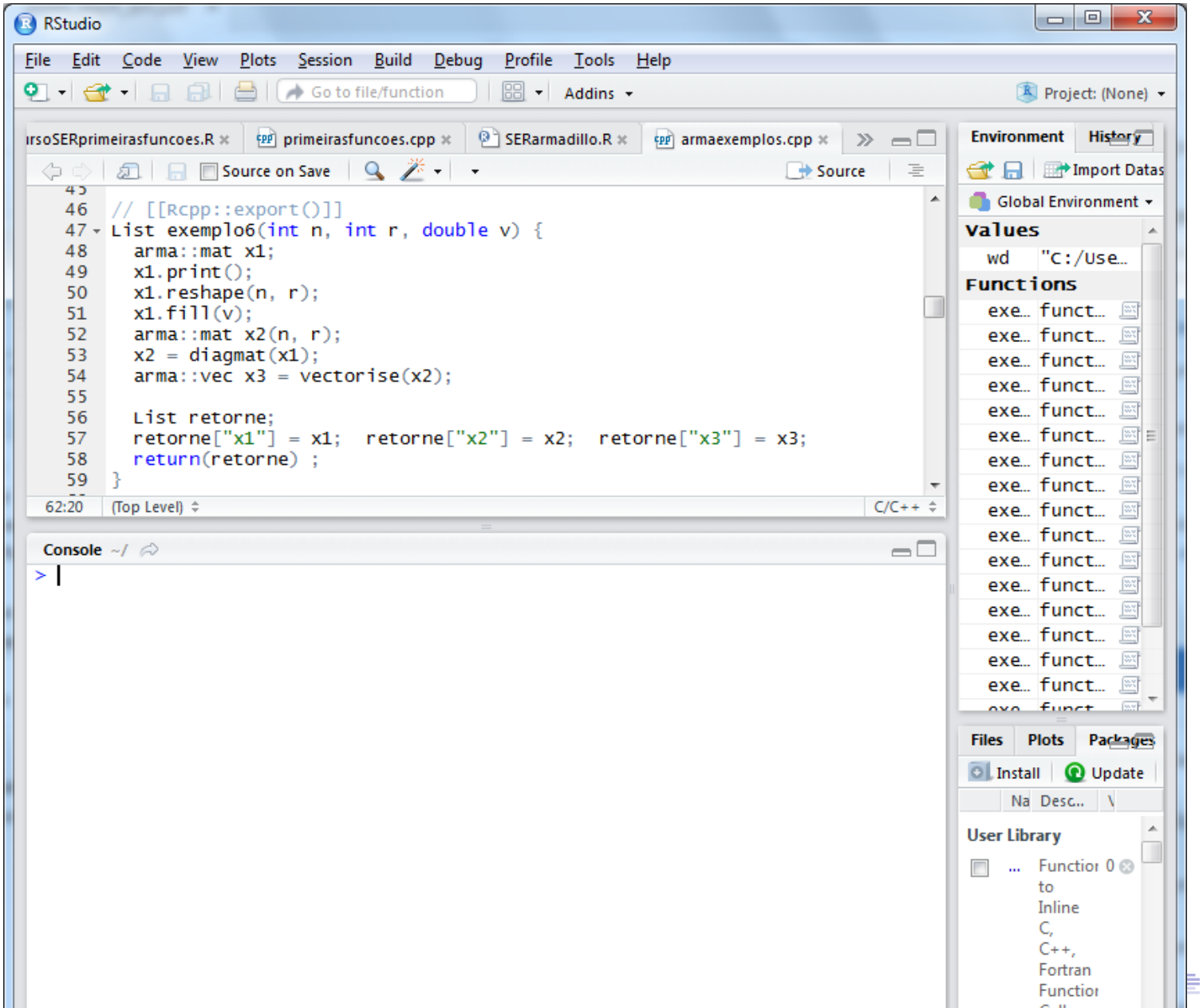
Exemplos básicos da estrutura de dados

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



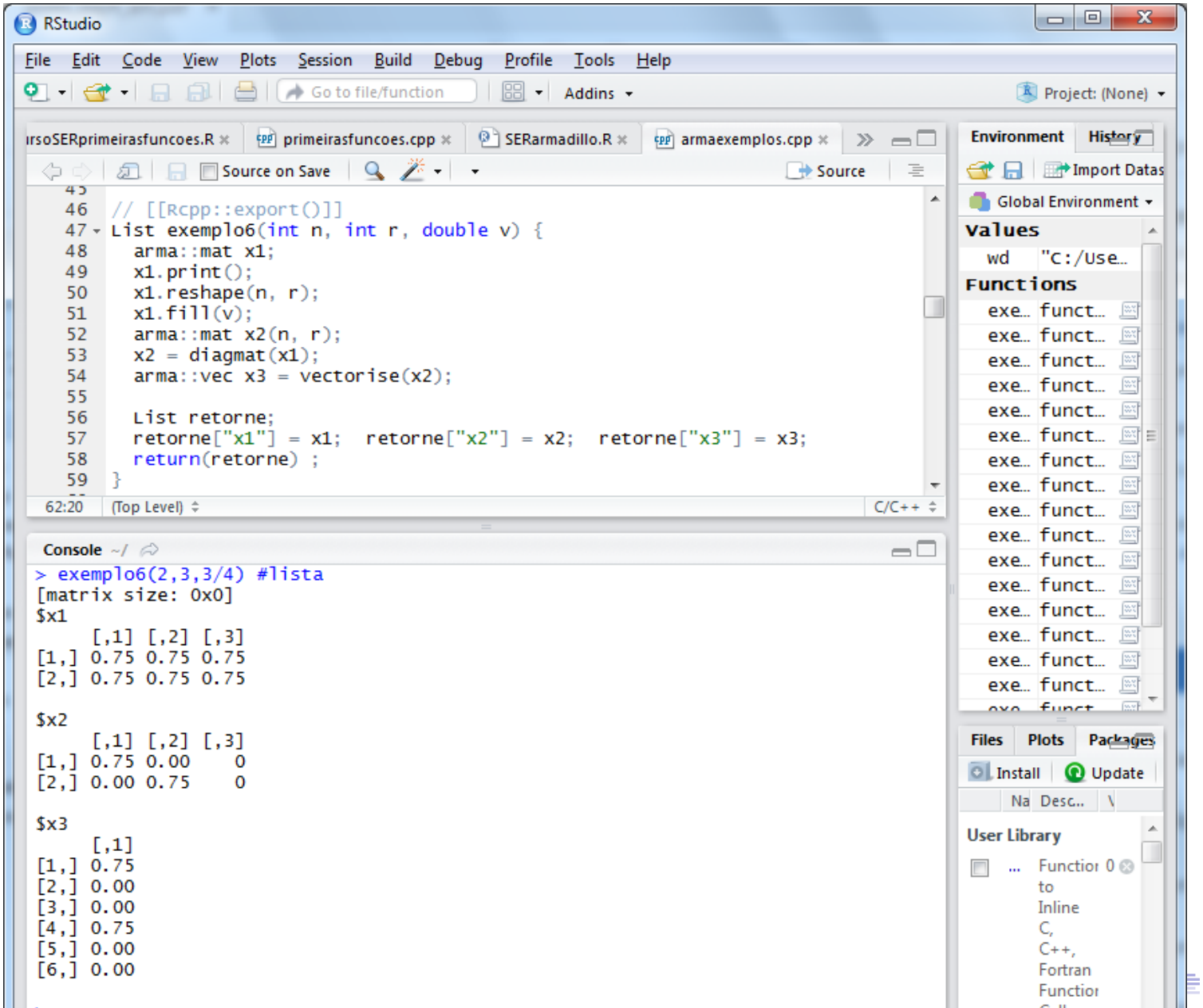
Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.



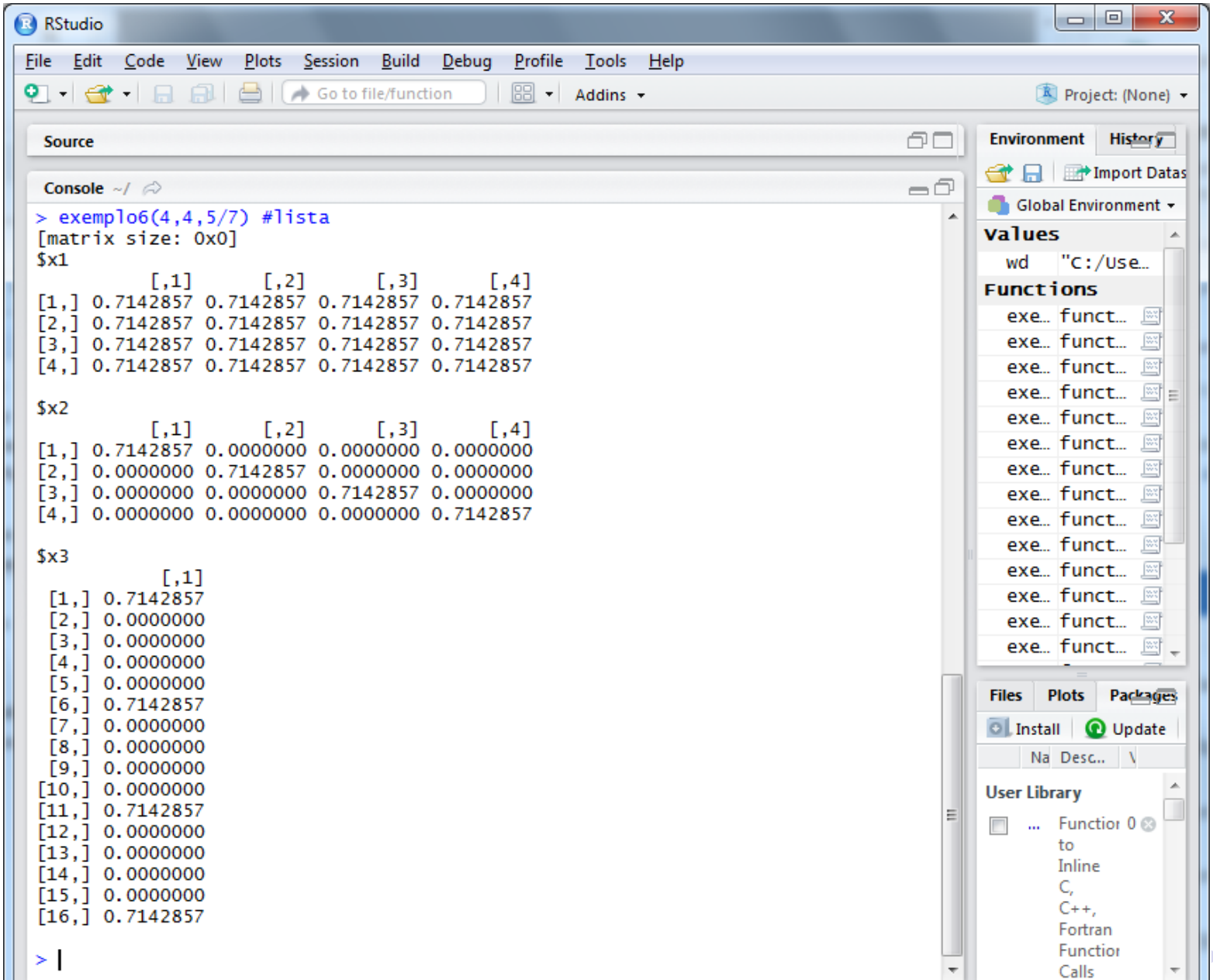
Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.



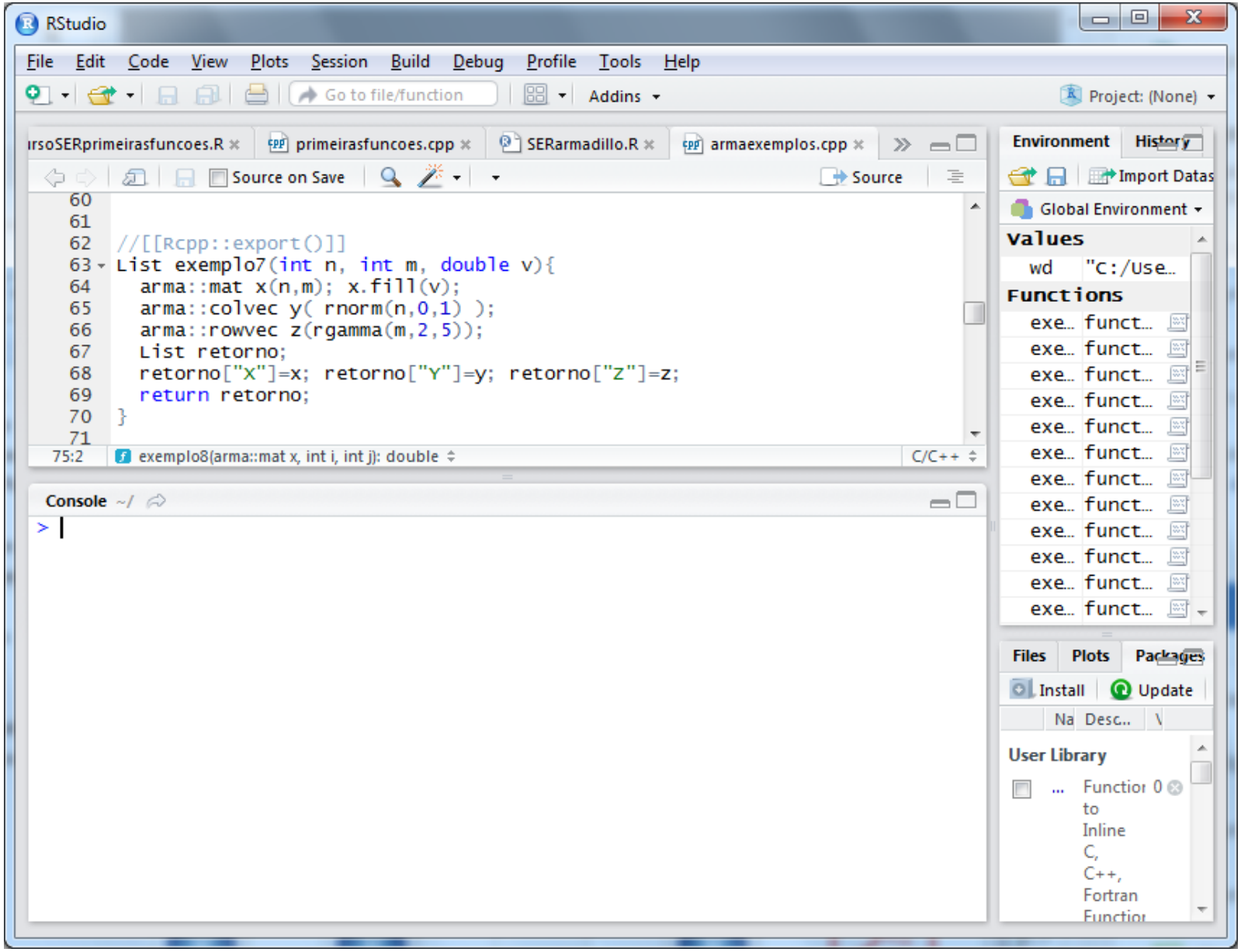
Exemplos básicos da estrutura de dados

Agora entrando com uma matriz quadrada.



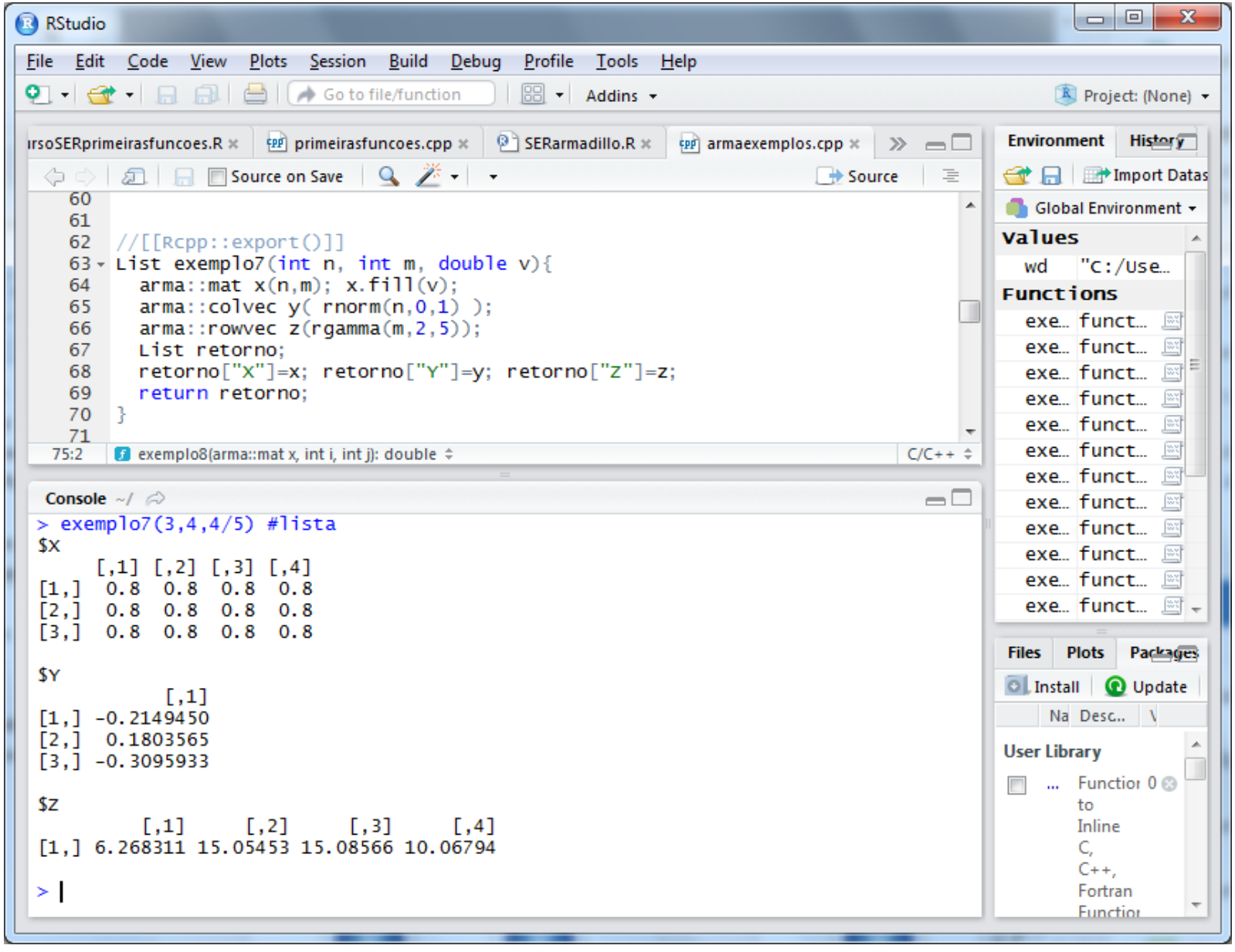
Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.



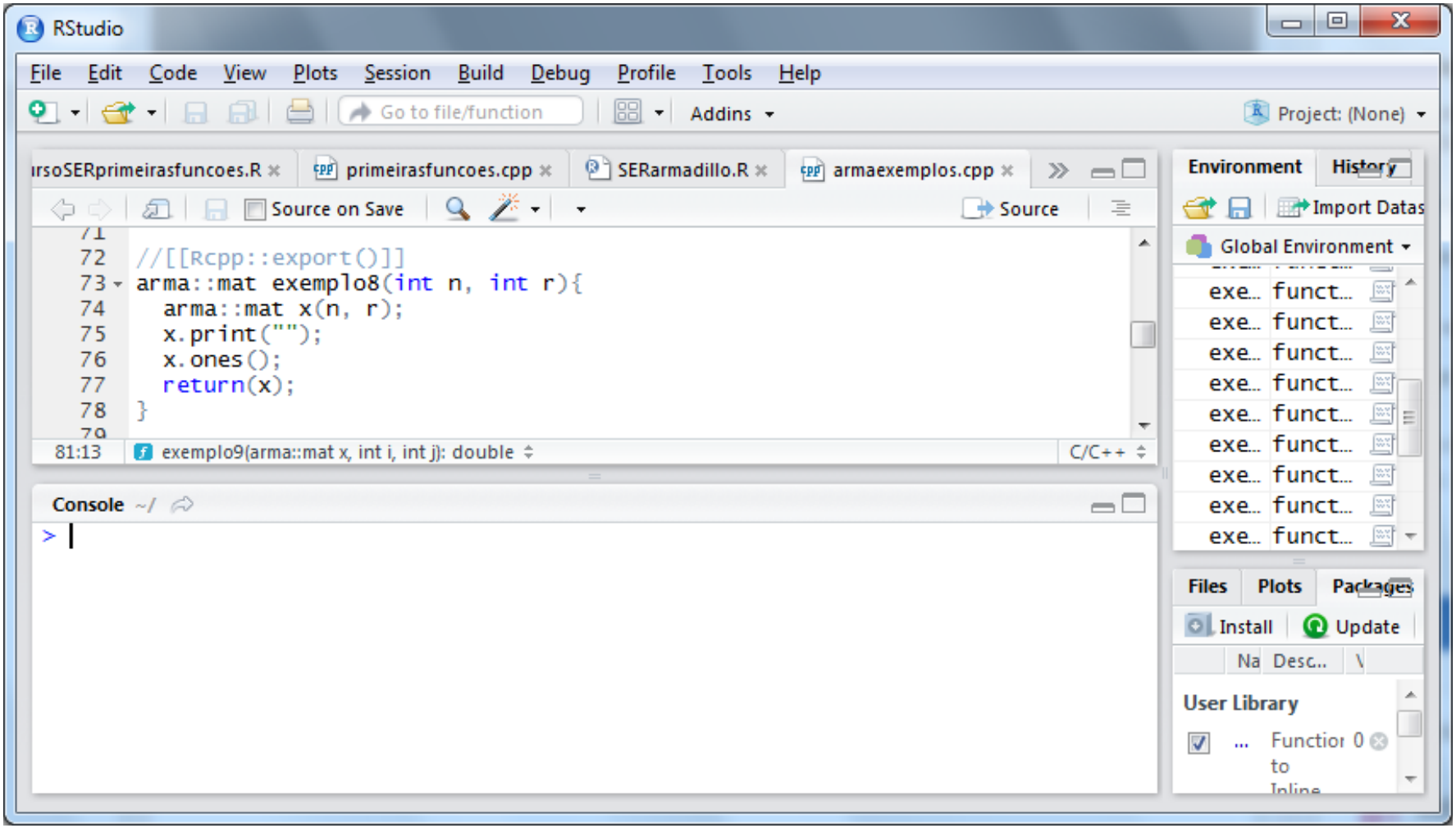
Exemplos básicos da estrutura de dados

Geração de números aleatórios e listas.



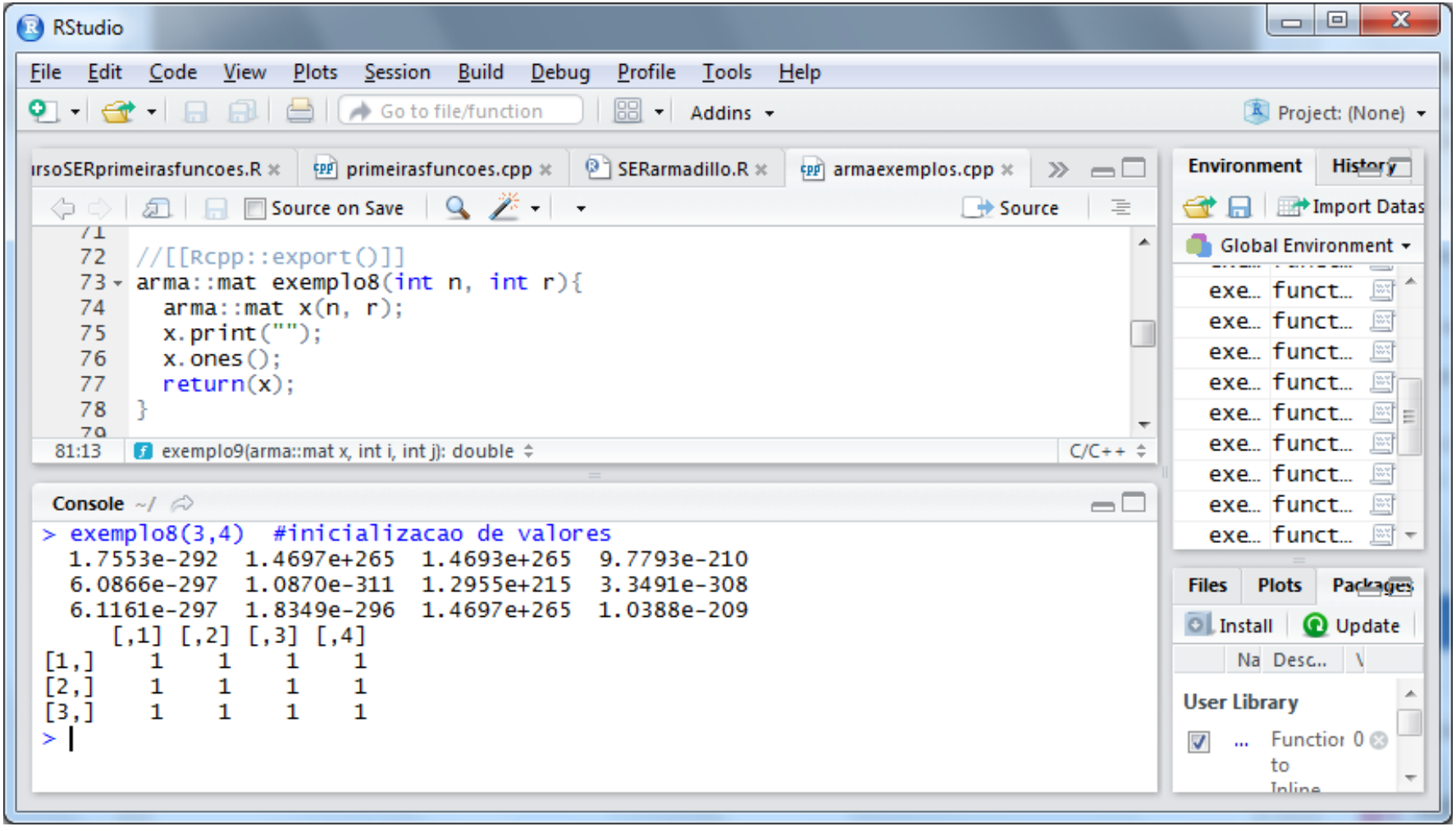
Exemplos básicos da estrutura de dados

Exemplo exibindo a inicialização dos valores.



Exemplos básicos da estrutura de dados

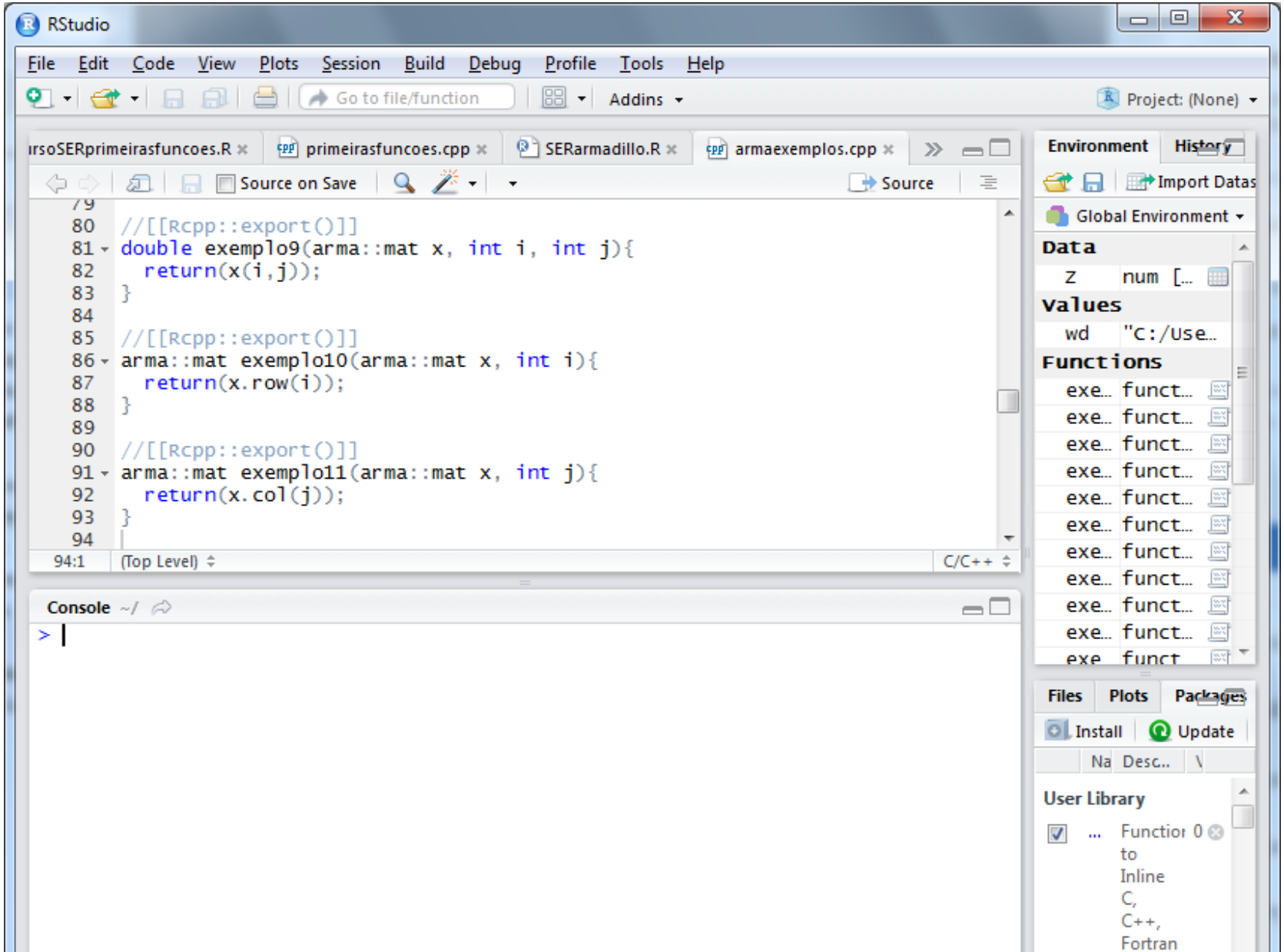
Veja que os valores não inicializam com zeros ou uns.



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

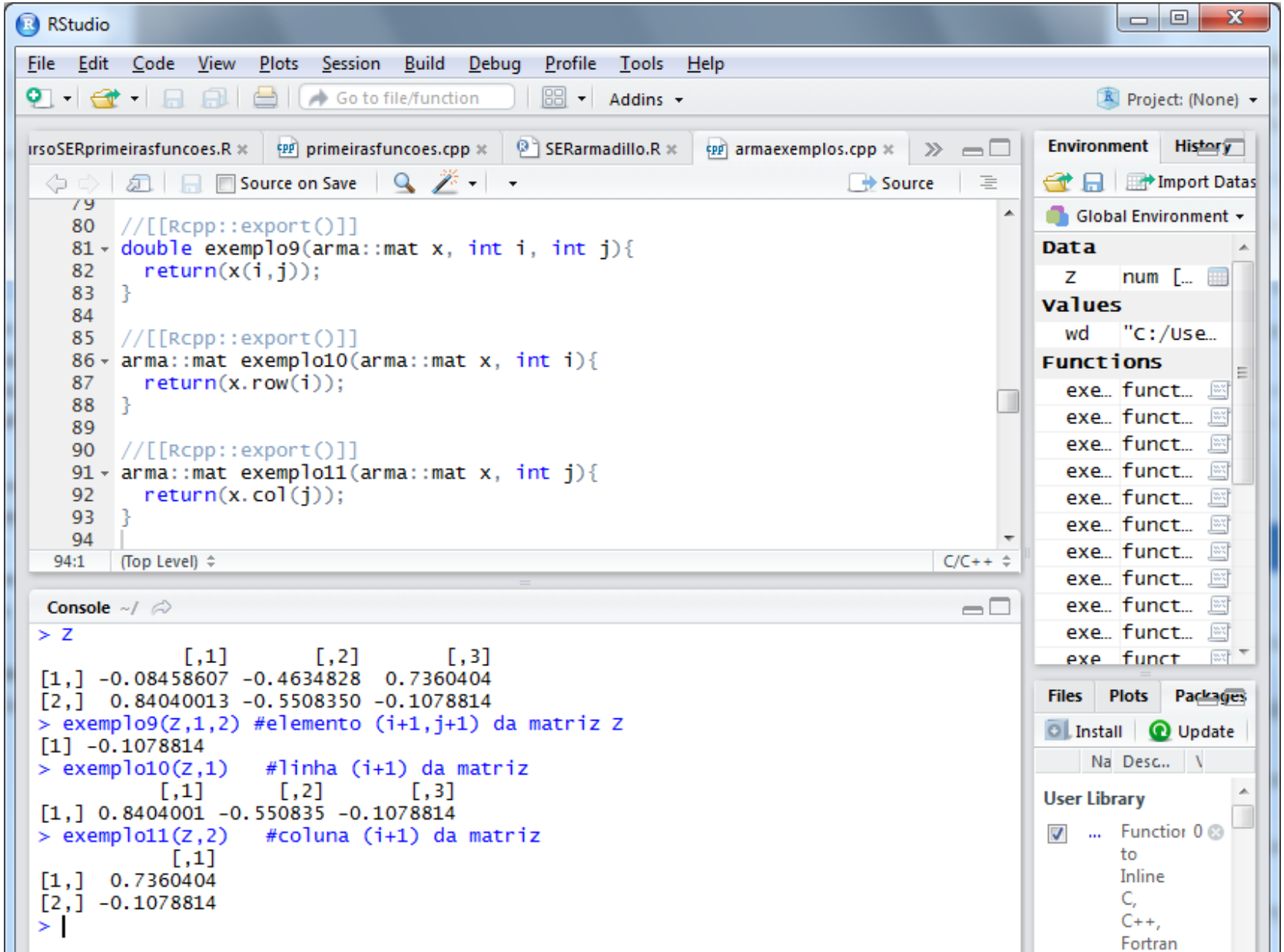
Exemplos básicos da estrutura de dados

Localização de elementos ou subconjuntos em matrizes.



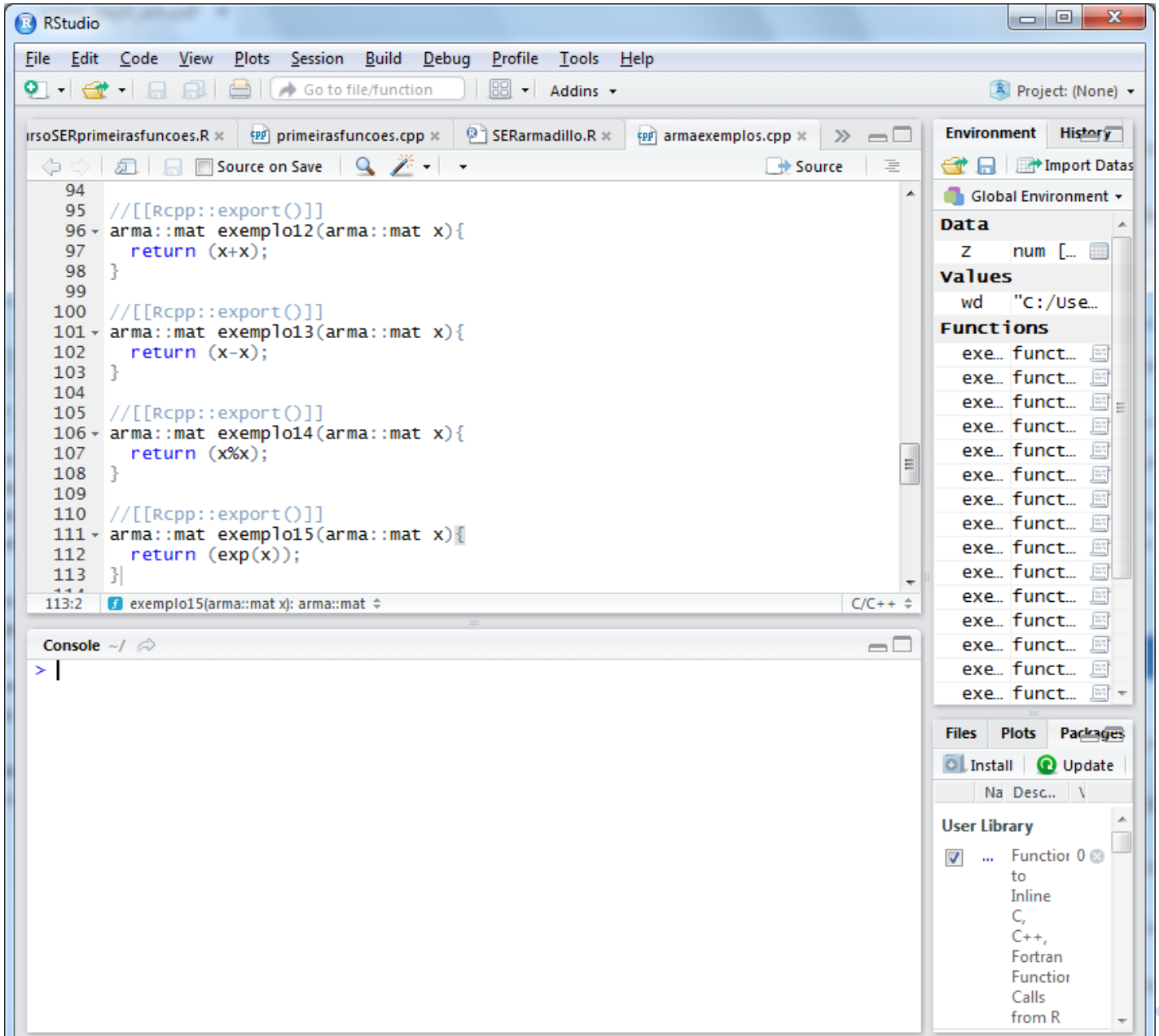
Exemplos básicos da estrutura de dados

Localização de elementos ou subconjuntos em matrizes.



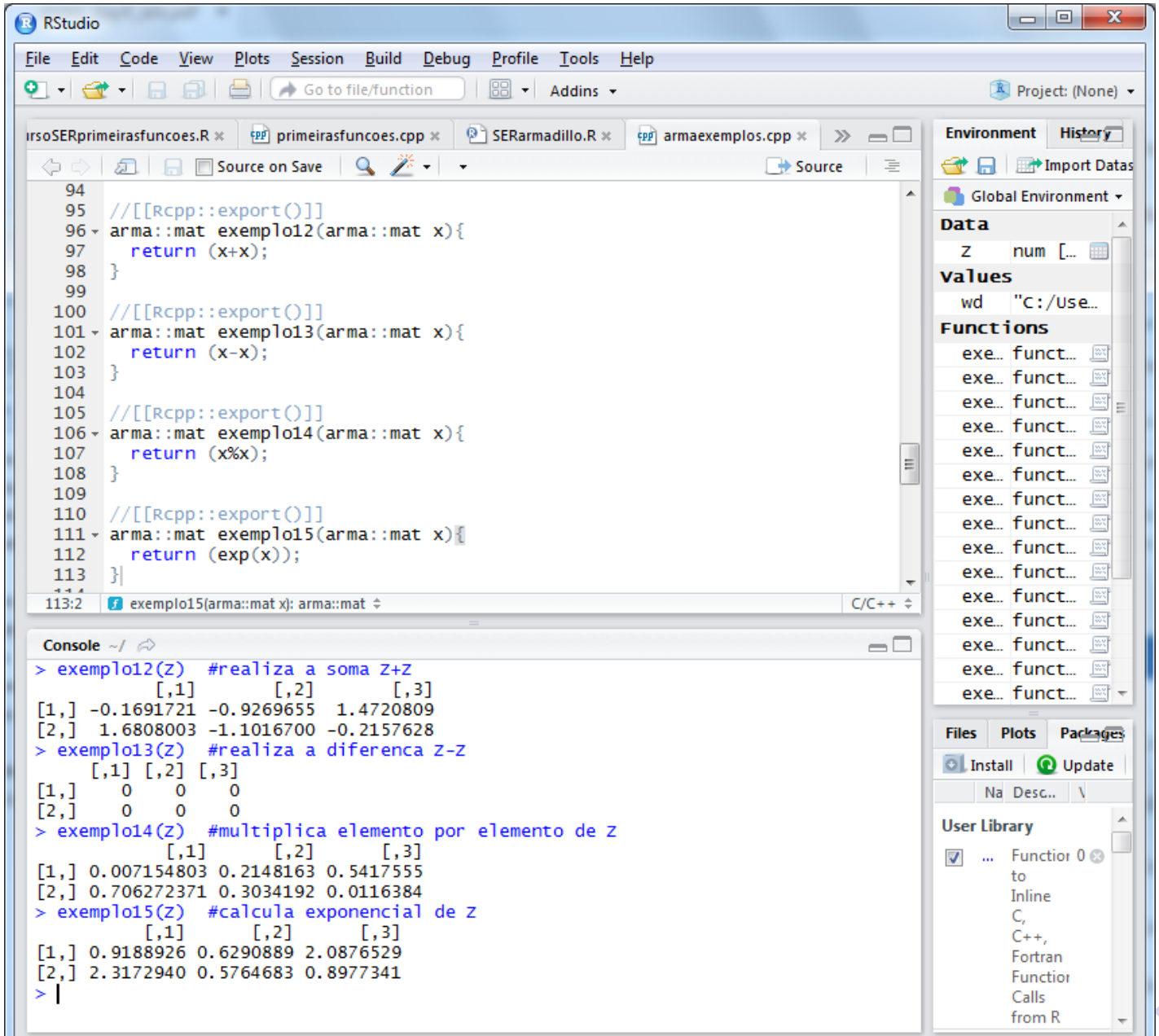
Exemplos básicos da estrutura de dados

Operações com matrizes

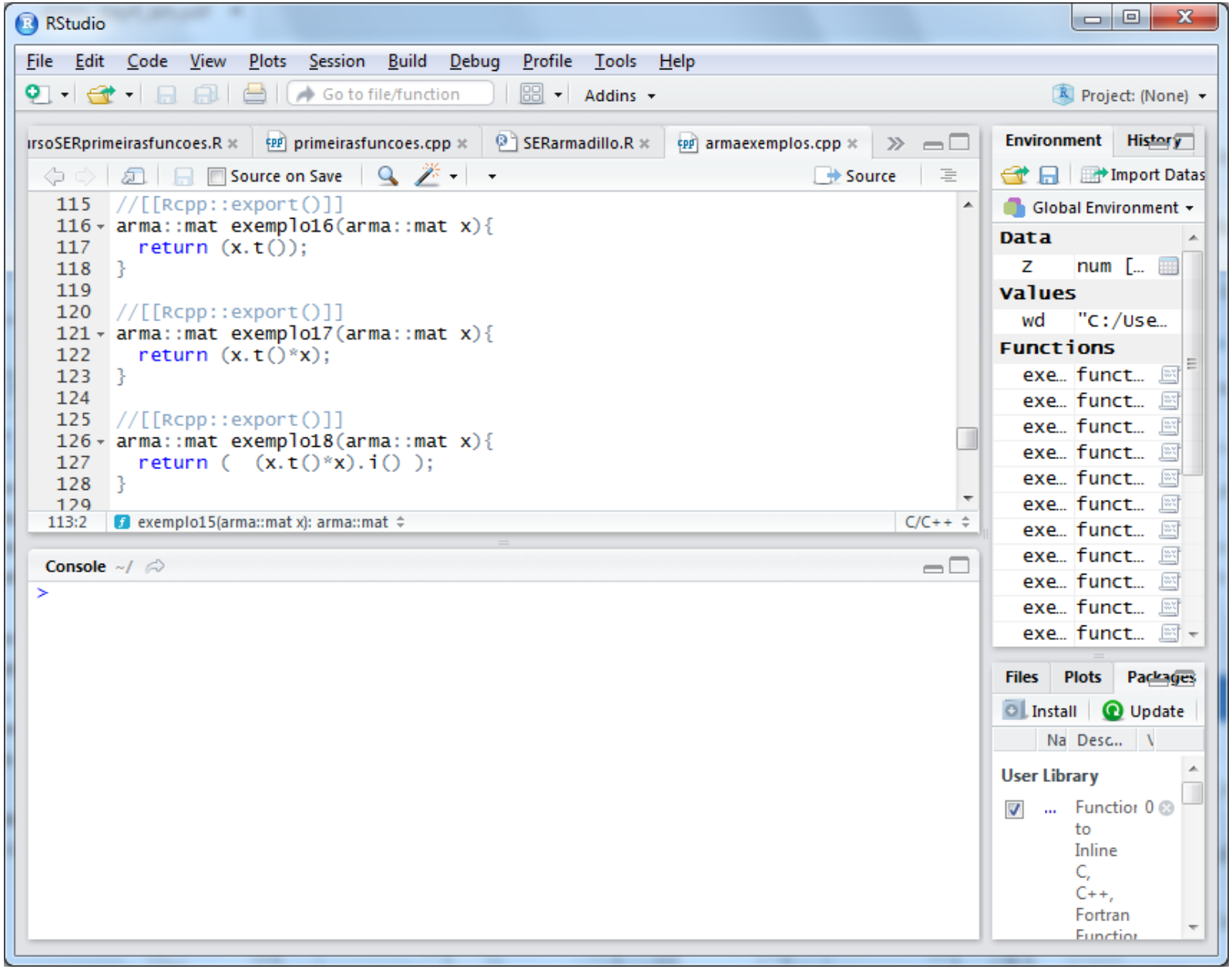


Exemplos básicos da estrutura de dados

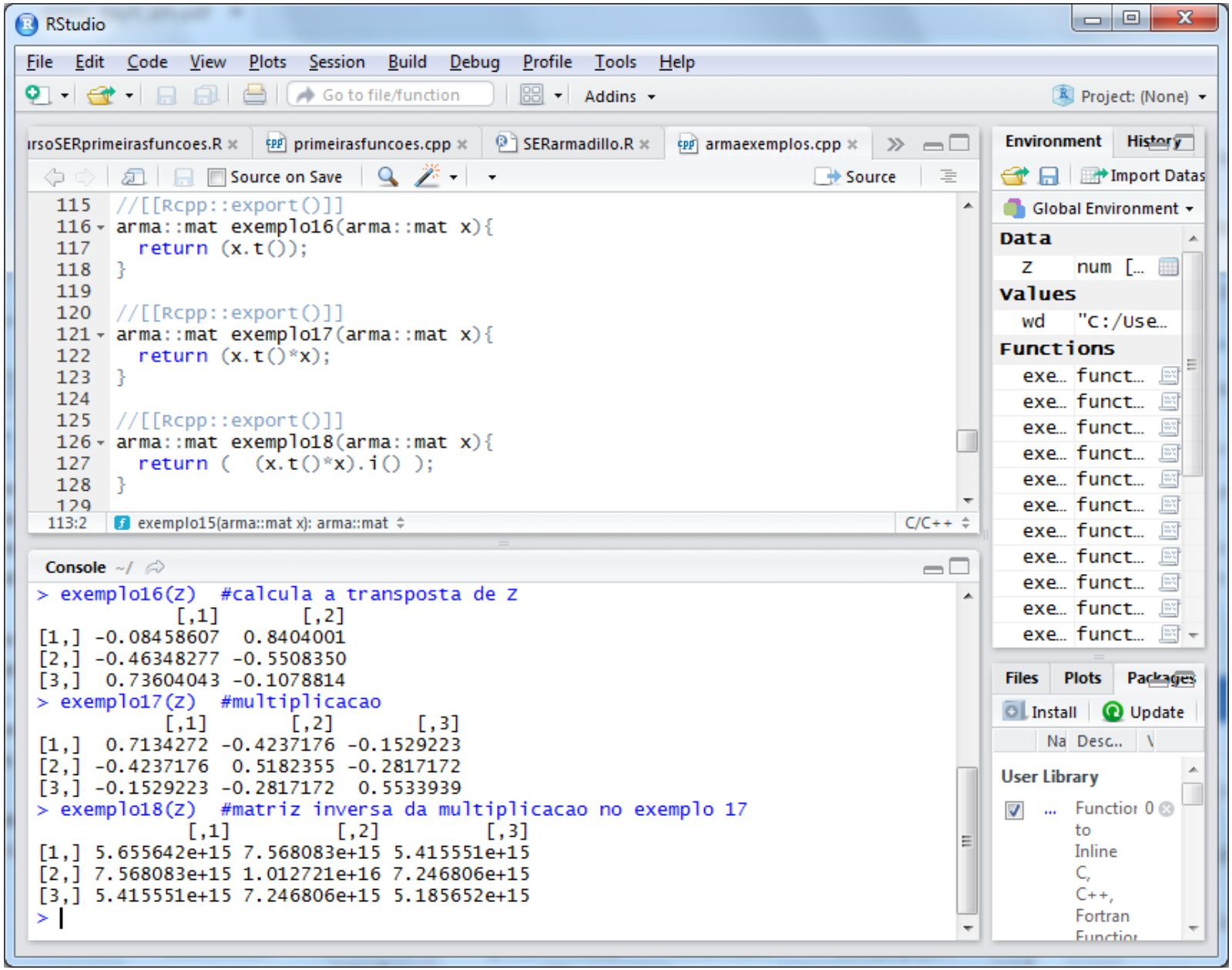
Operações com matrizes



Exemplos básicos da estrutura de dados

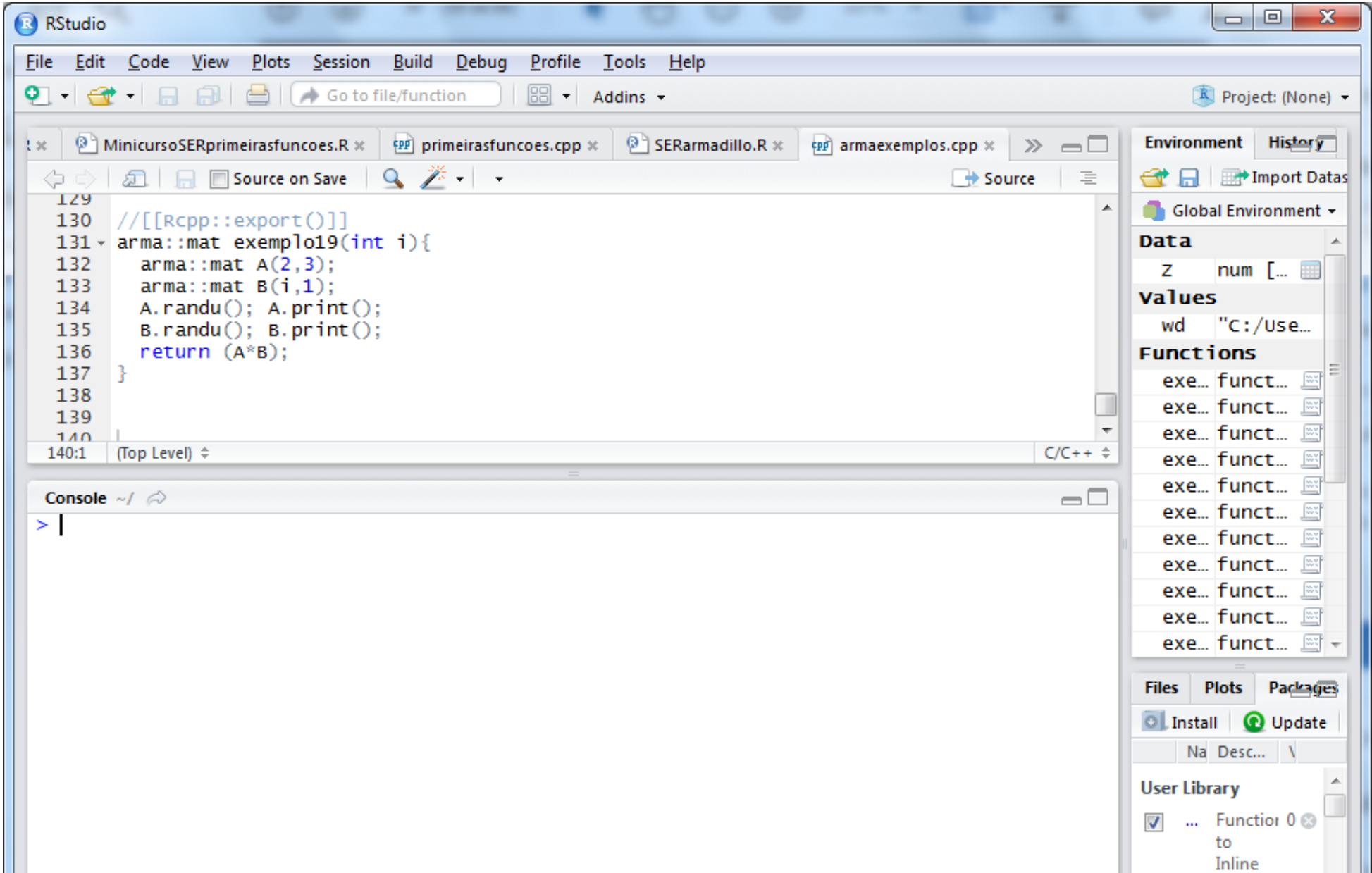


Exemplos básicos da estrutura de dados

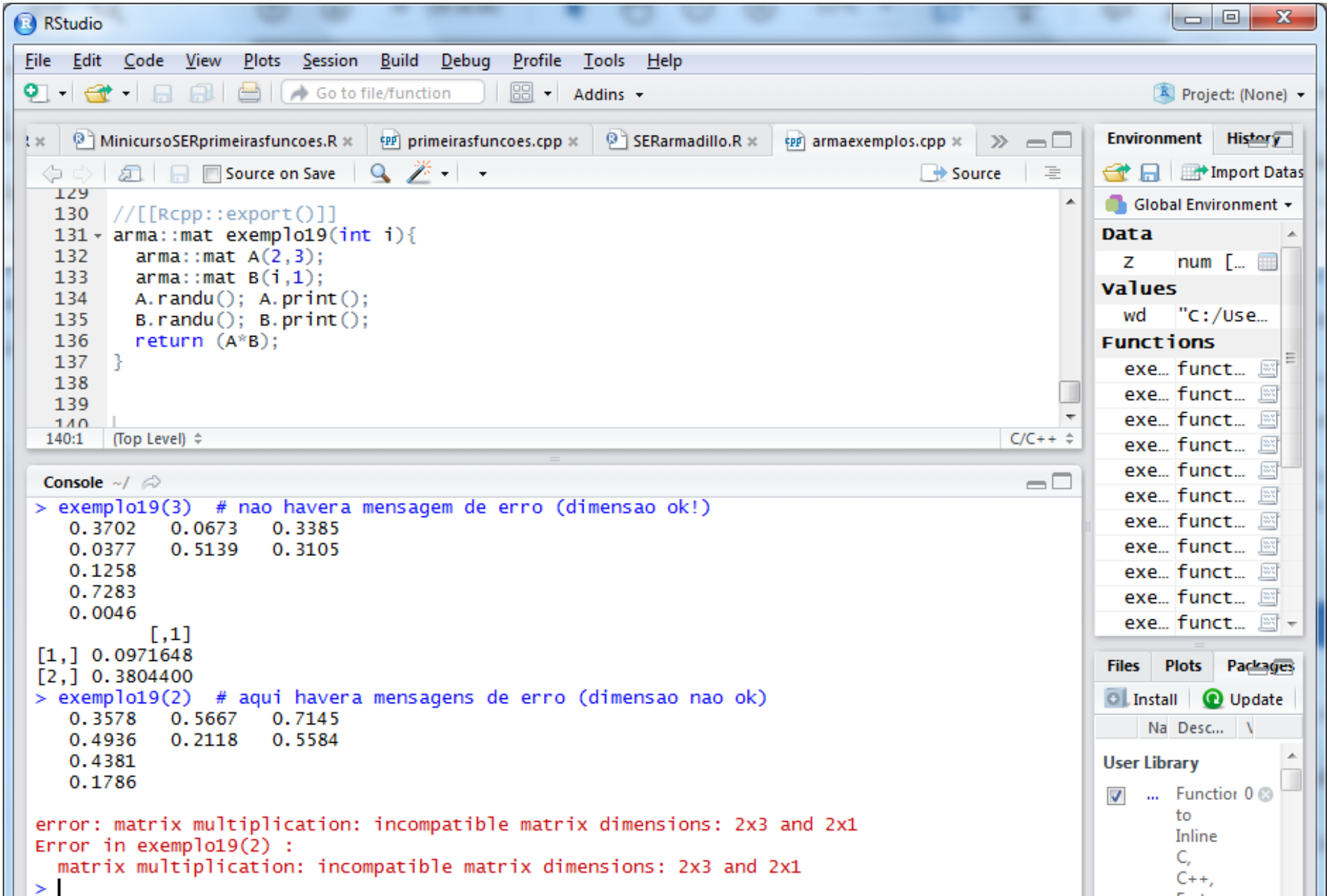


Exemplos básicos da estrutura de dados

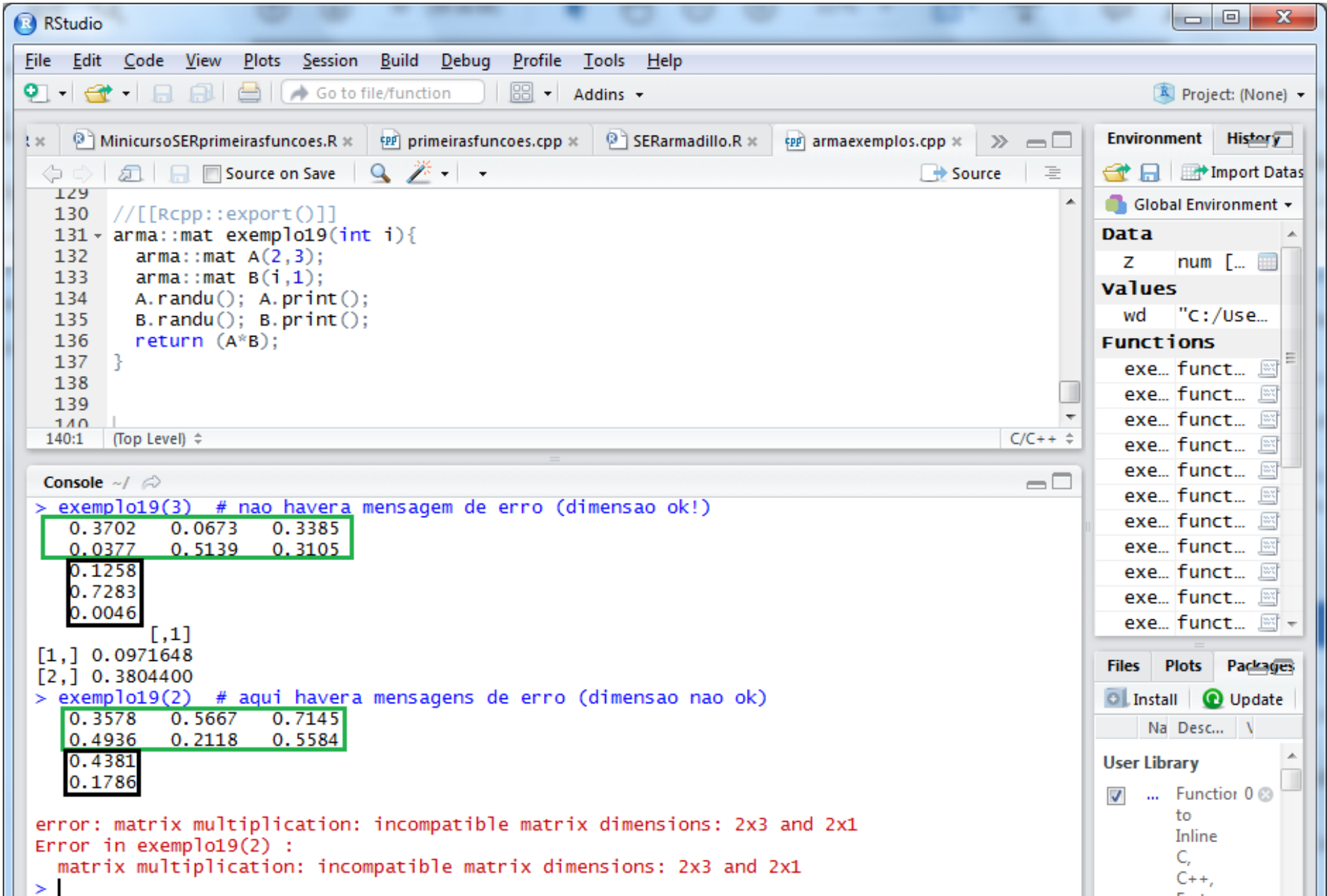
Geração de matriz aleatória e mensagens de erro.



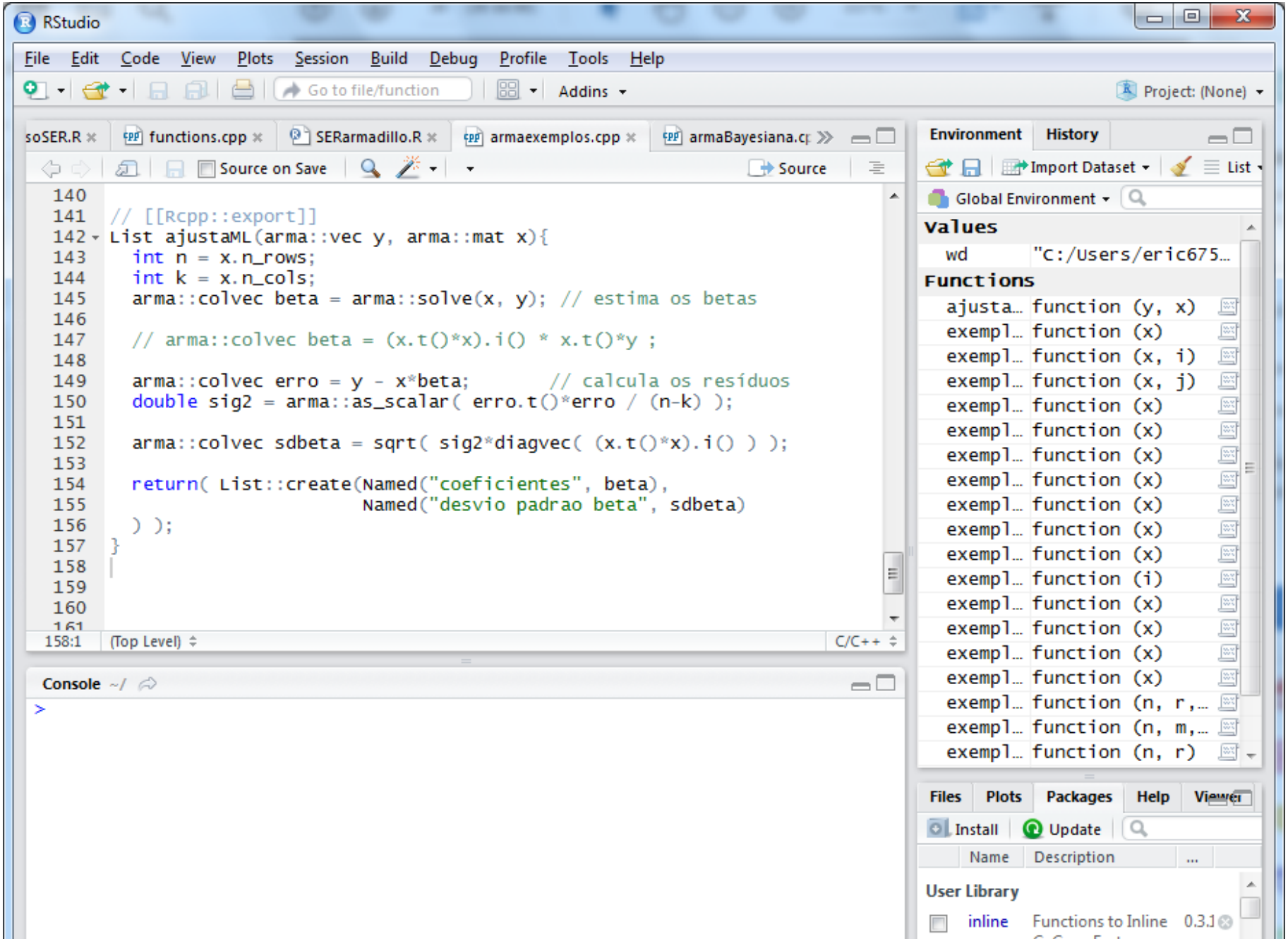
Exemplos básicos da estrutura de dados



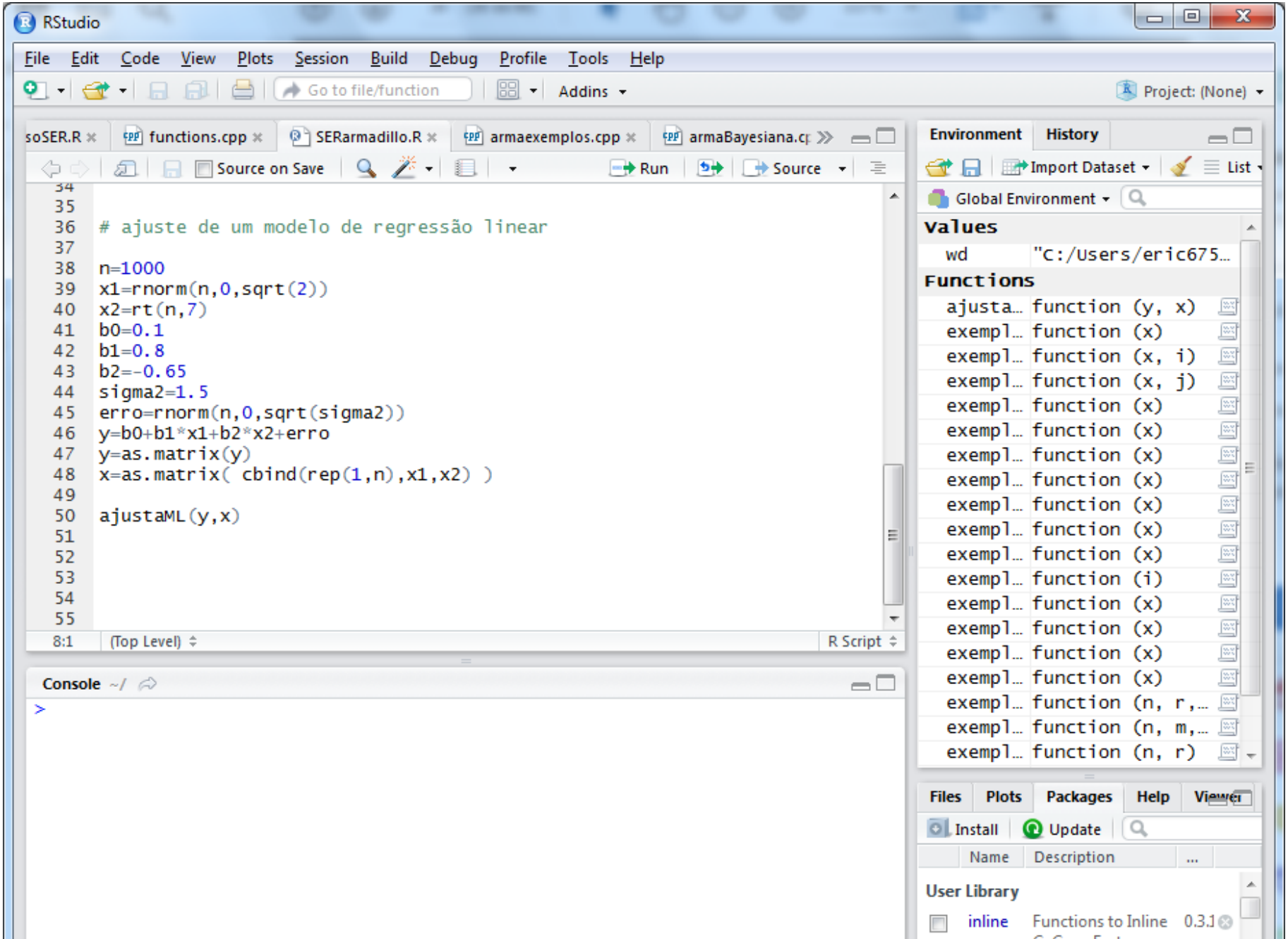
Exemplos básicos da estrutura de dados



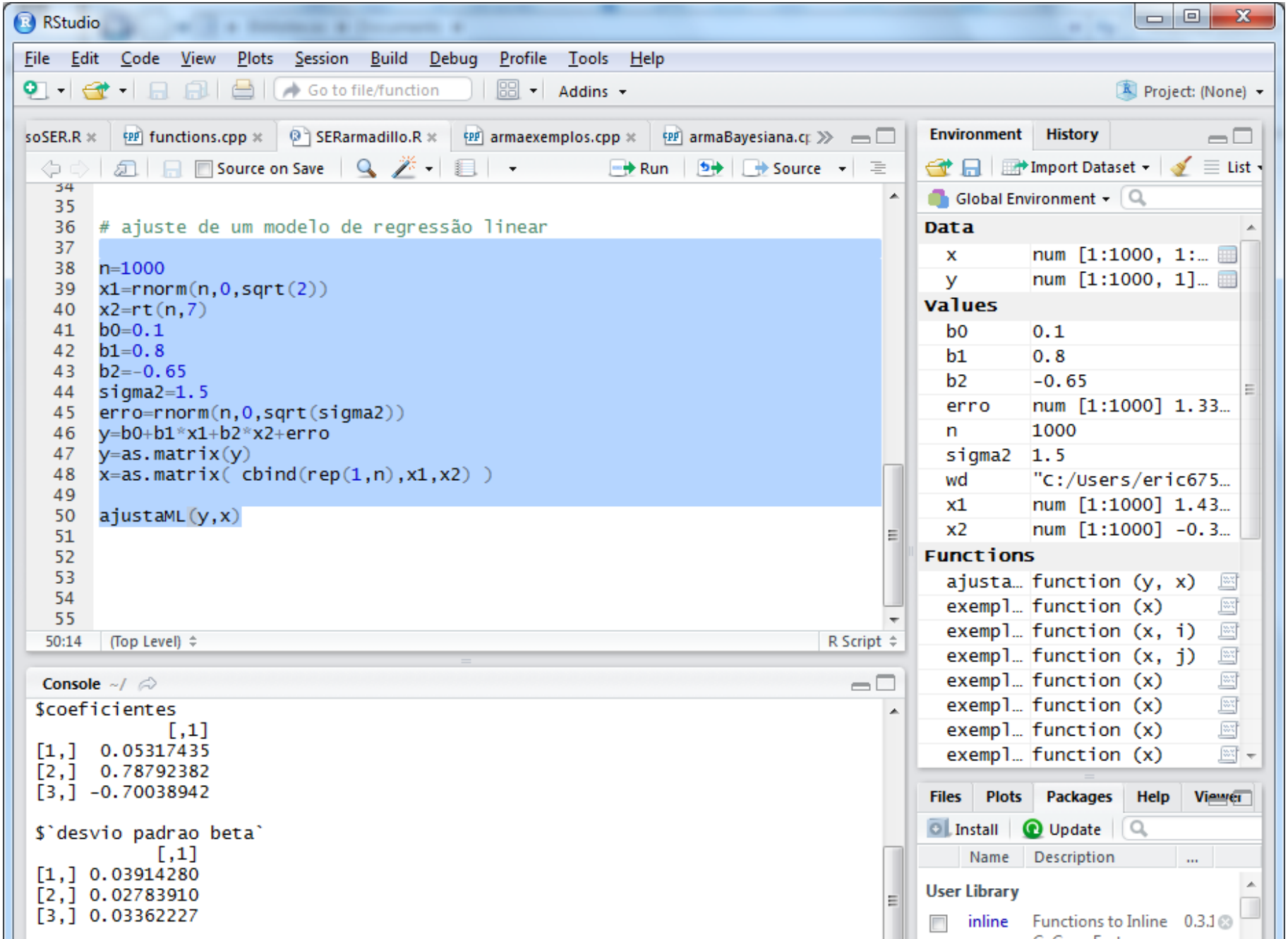
Exemplo de um ajuste linear



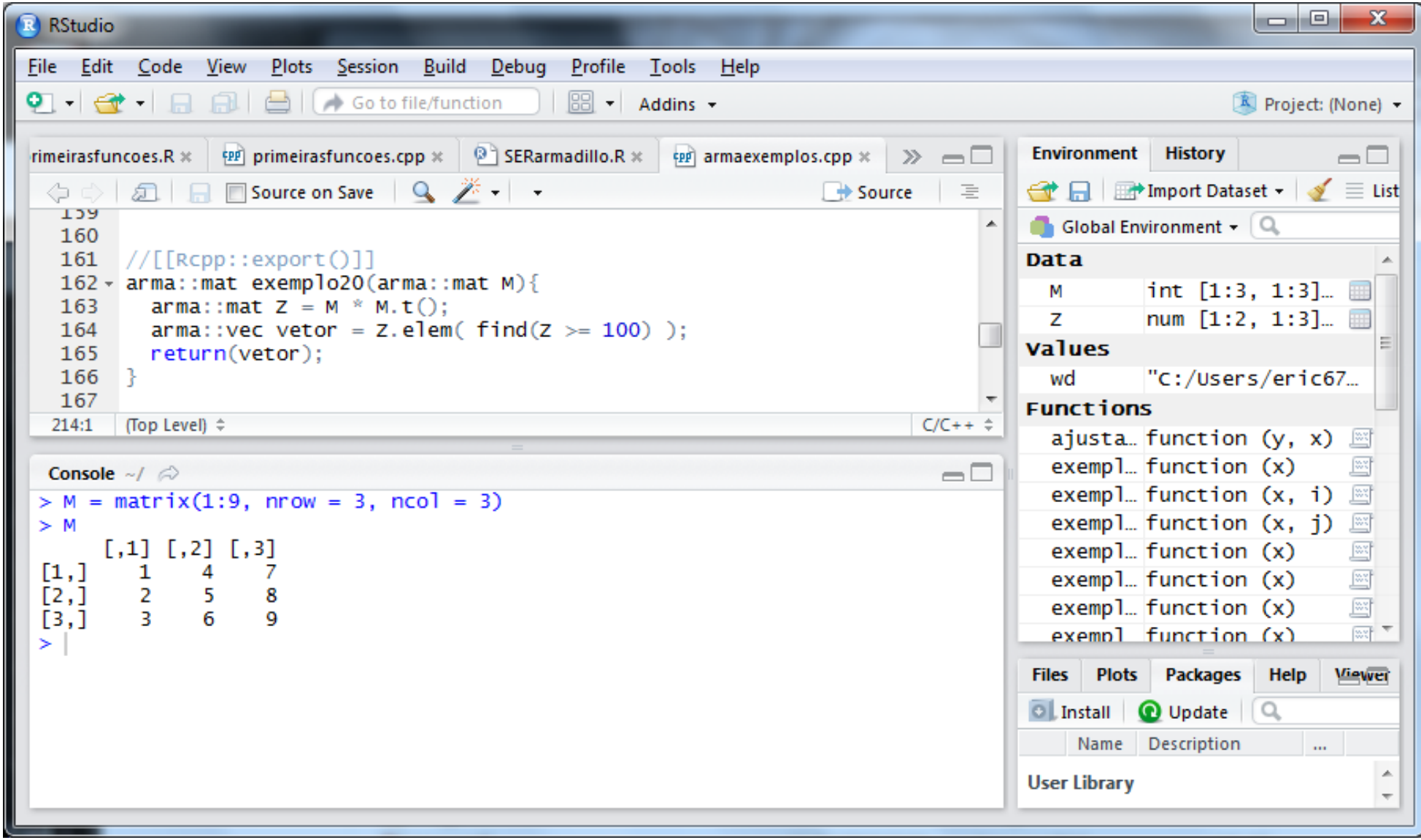
Exemplo de um ajuste linear



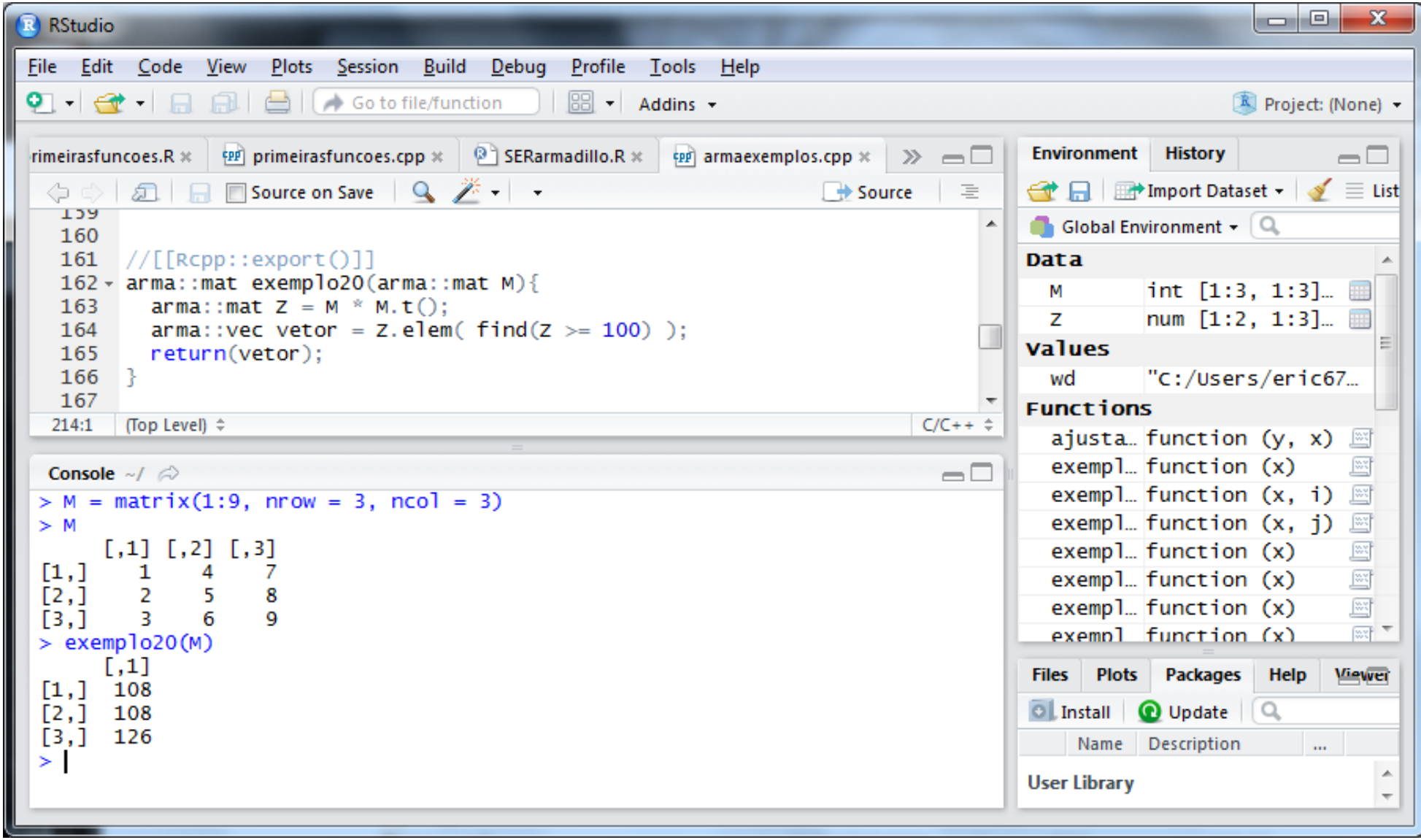
Exemplo de um ajuste linear



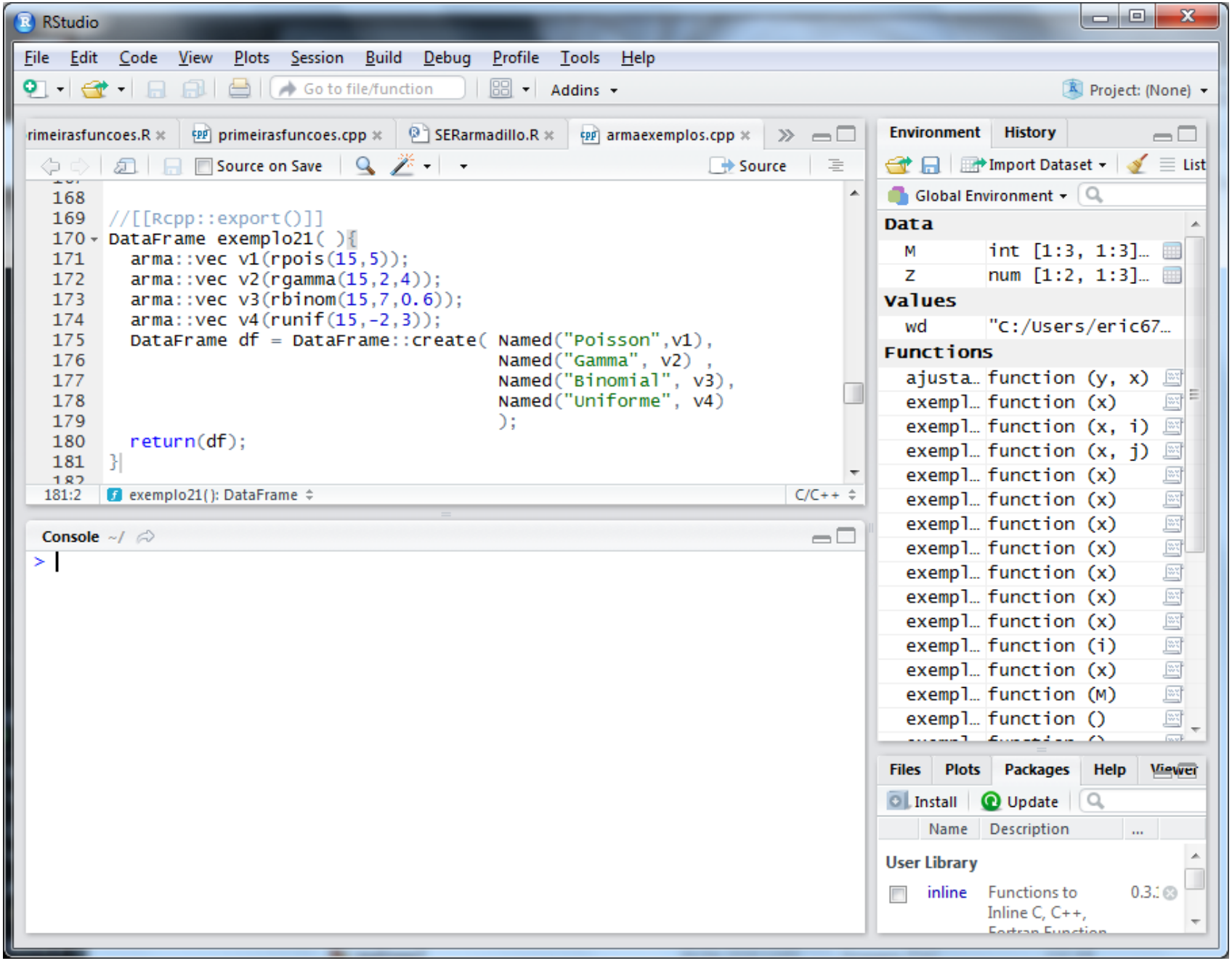
Exemplos usando DataFrame



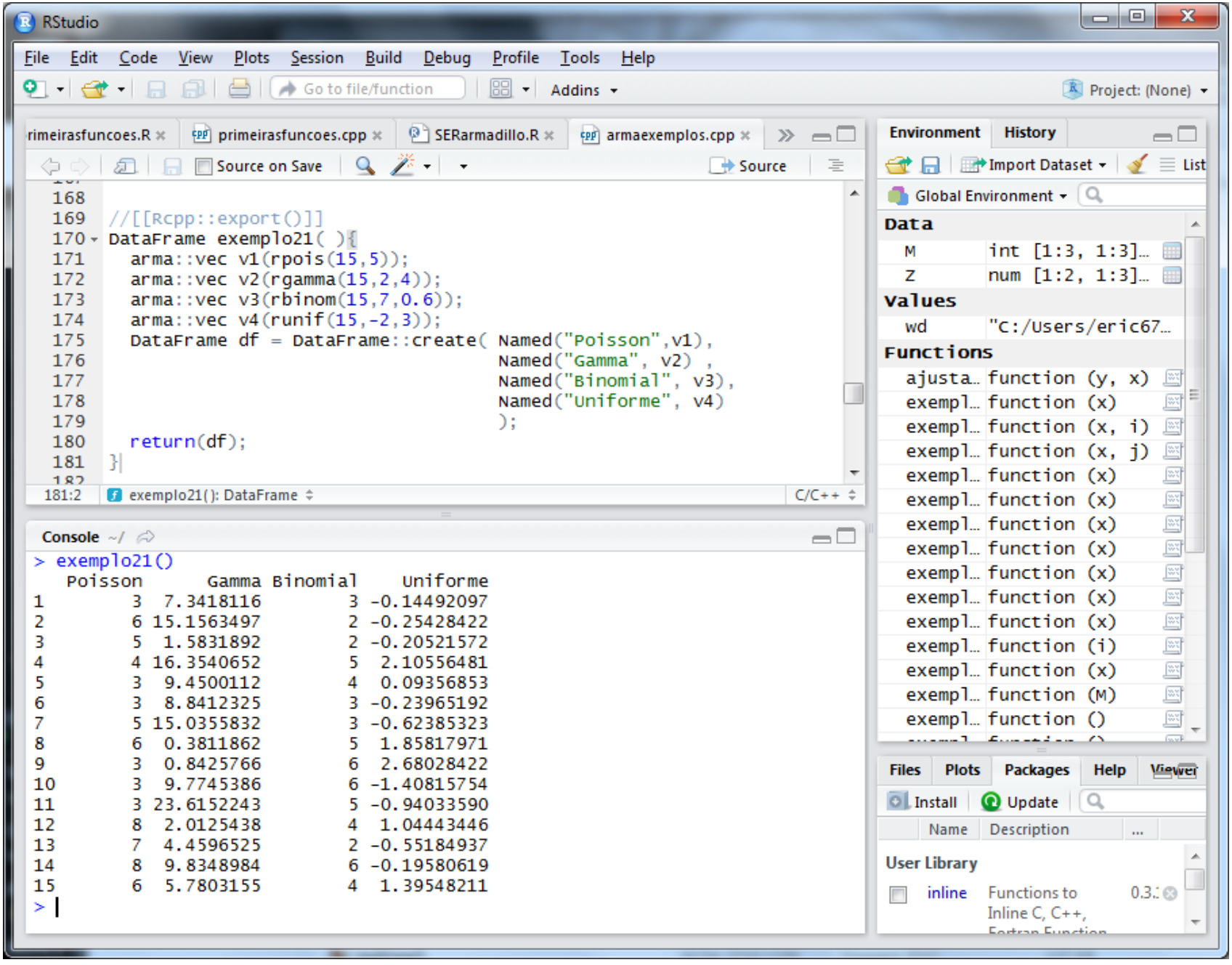
Exemplos usando DataFrame



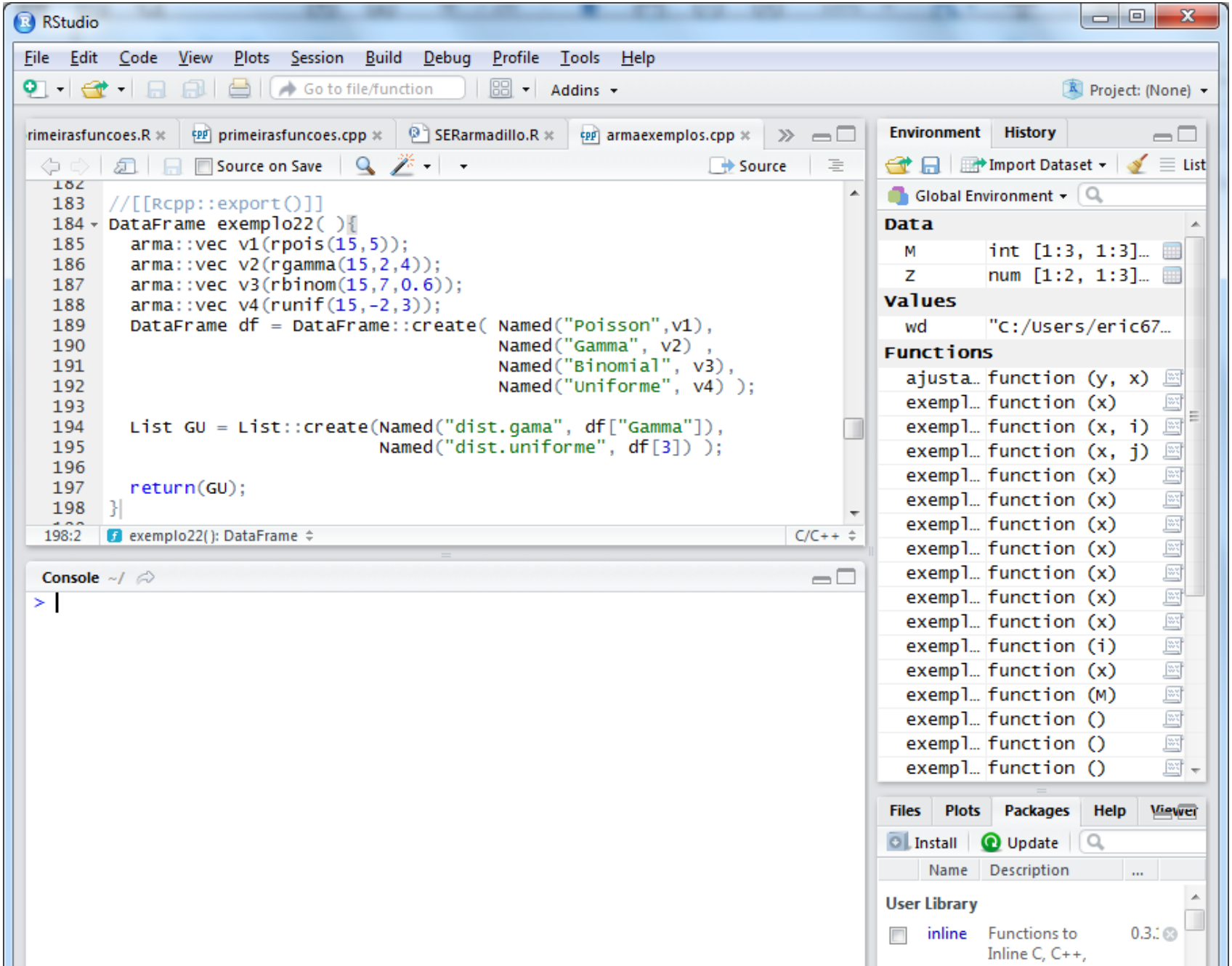
Exemplos usando DataFrame



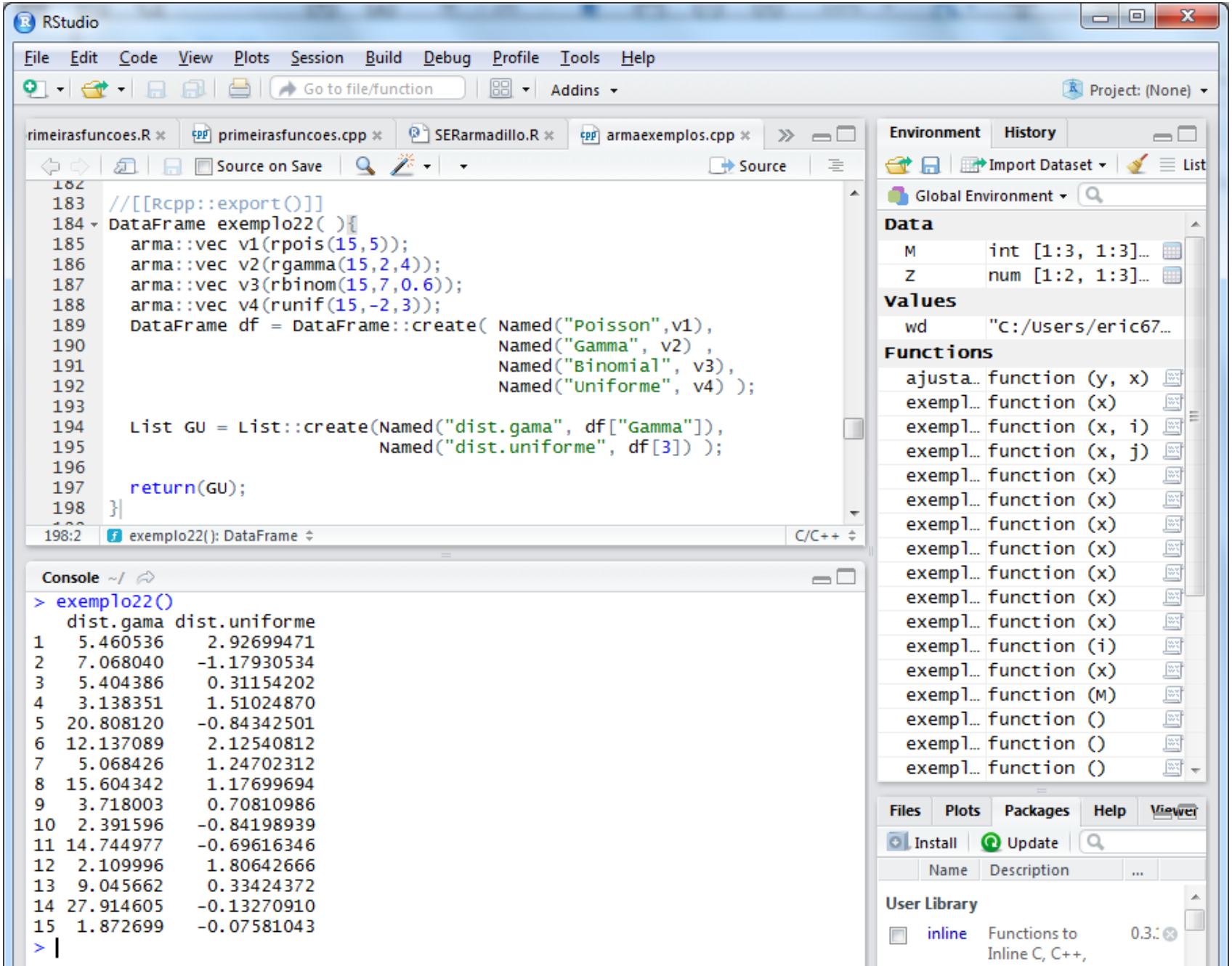
Exemplos usando DataFrame



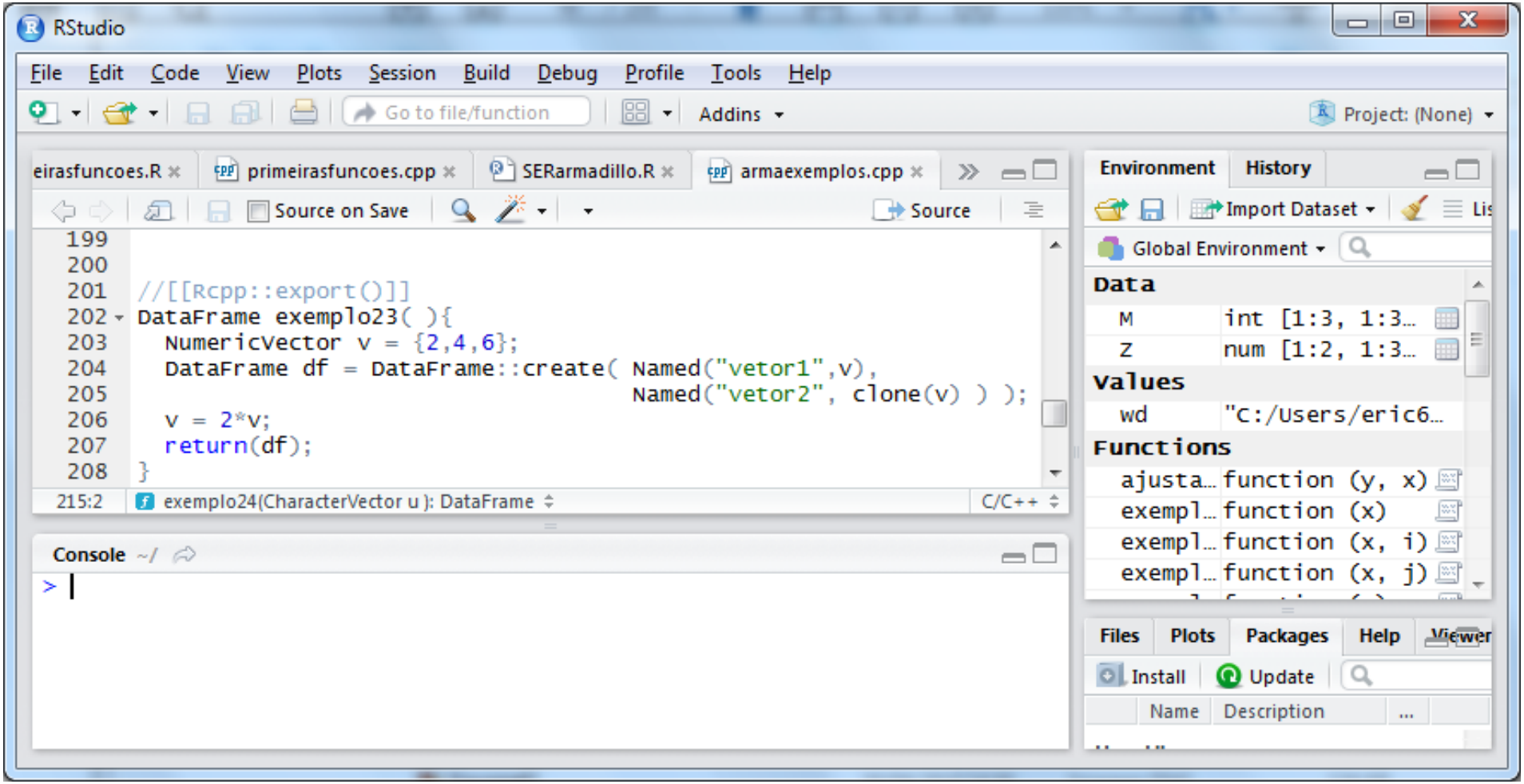
Exemplos usando DataFrame



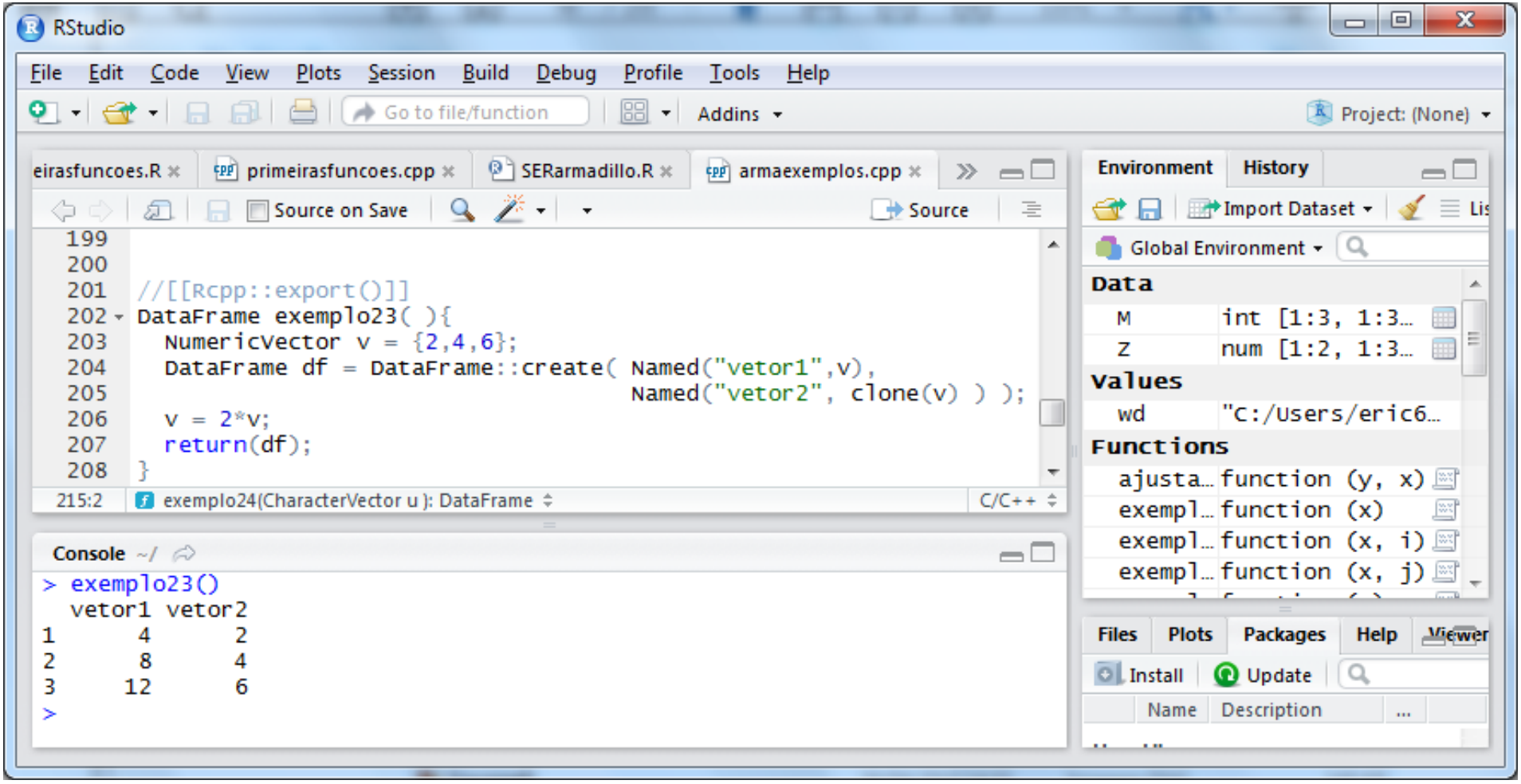
Exemplos usando DataFrame



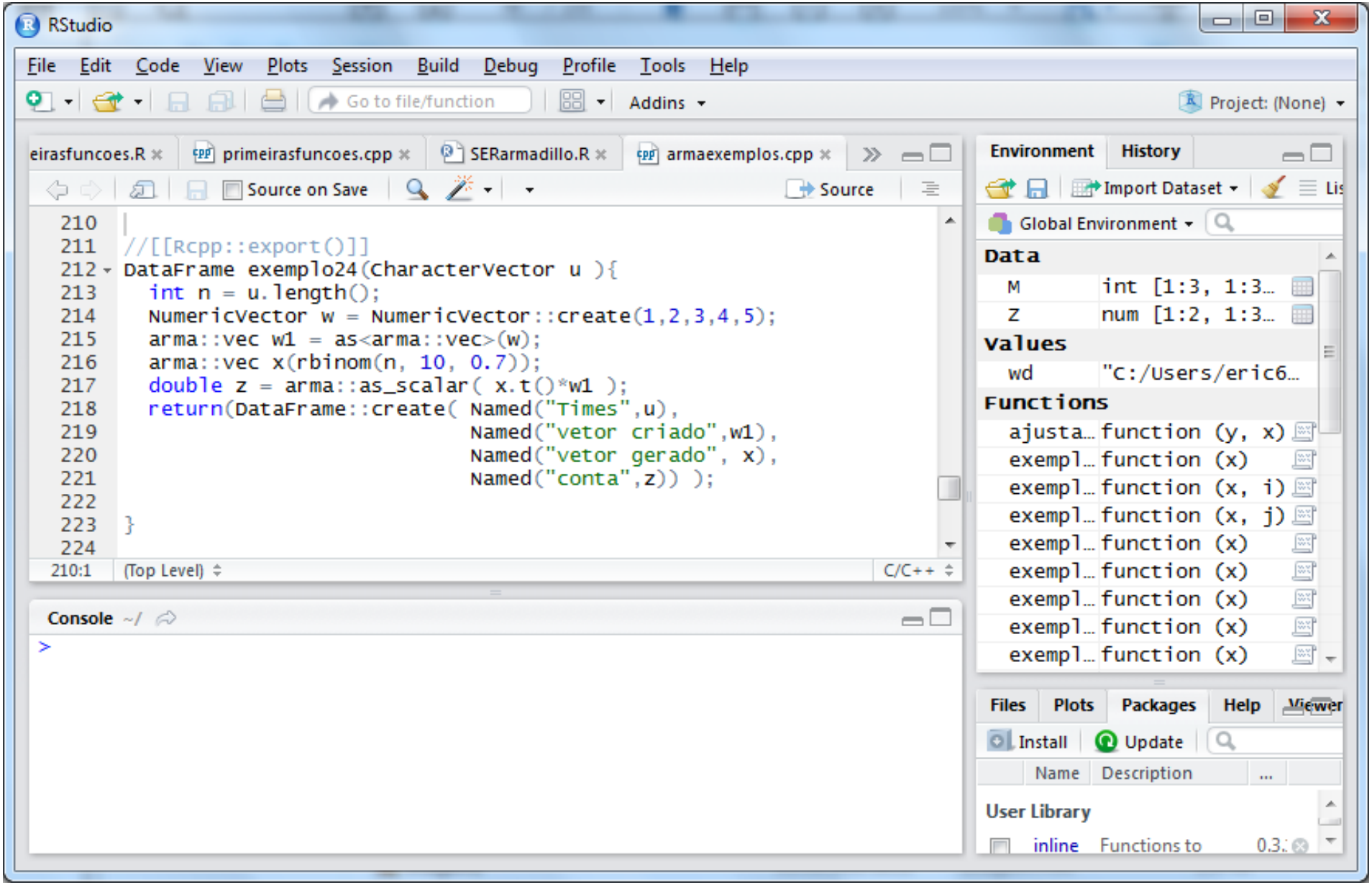
Exemplos usando DataFrame



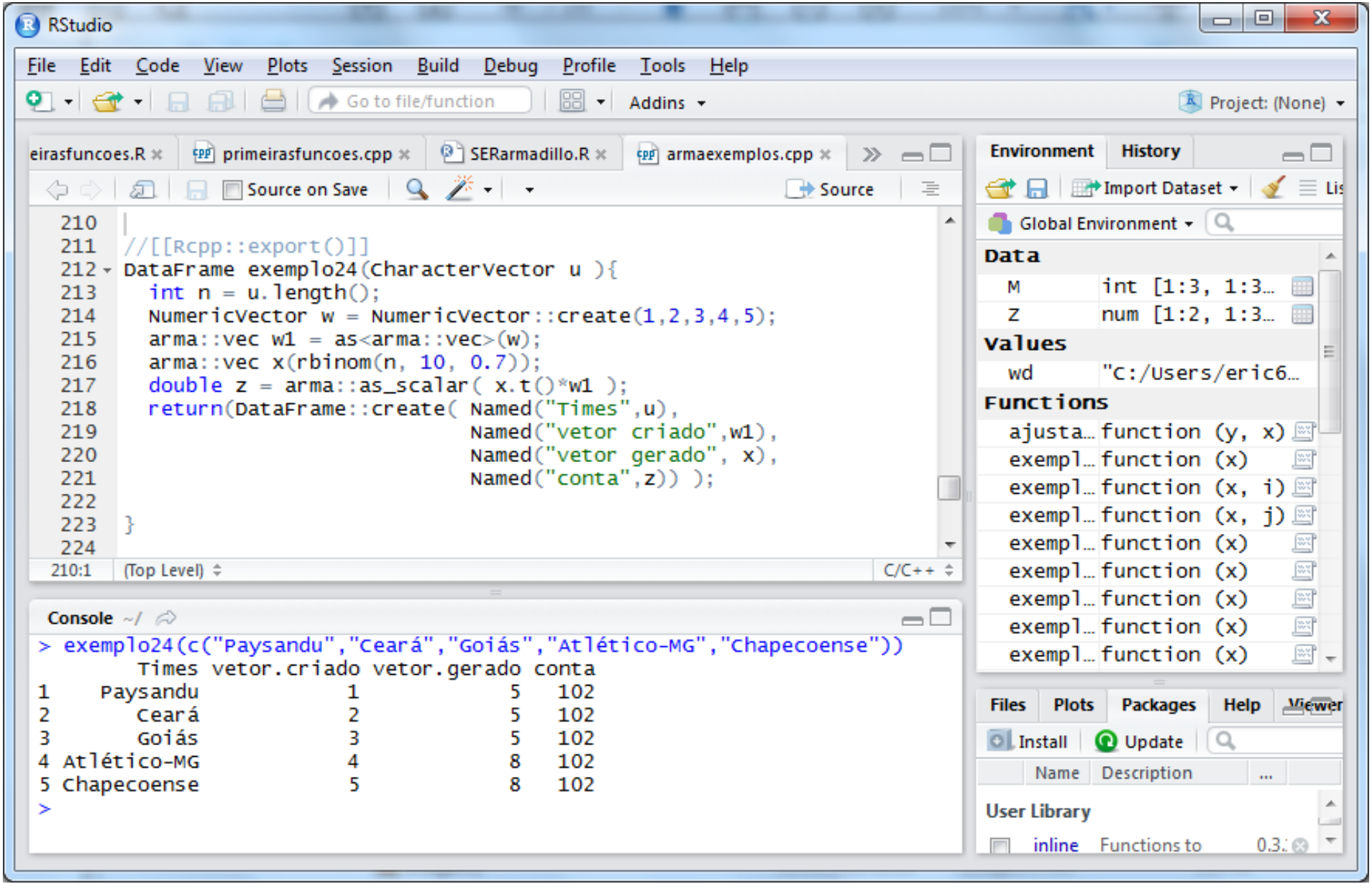
Exemplos usando DataFrame



Exemplos usando DataFrame

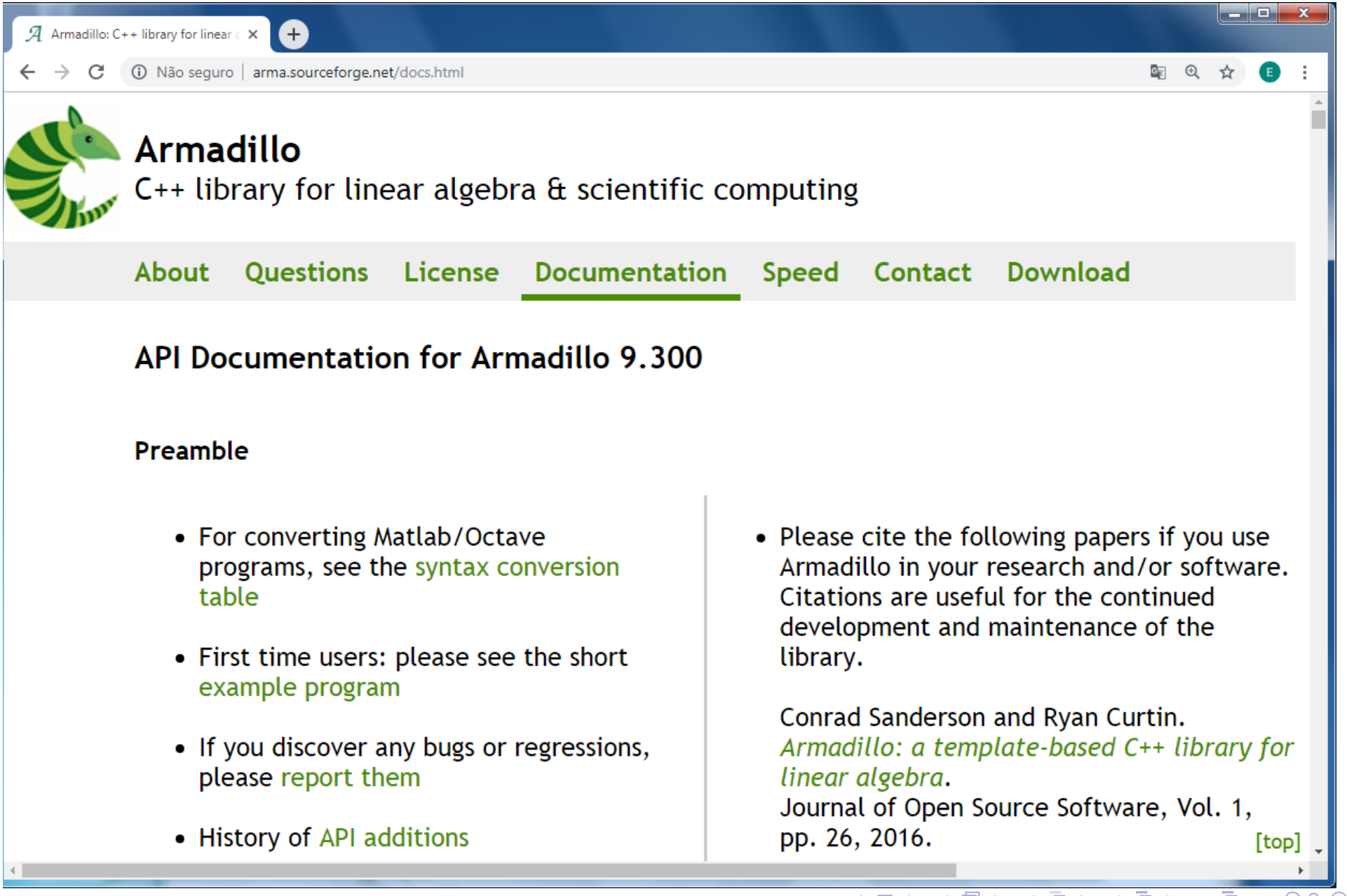


Exemplos usando DataFrame

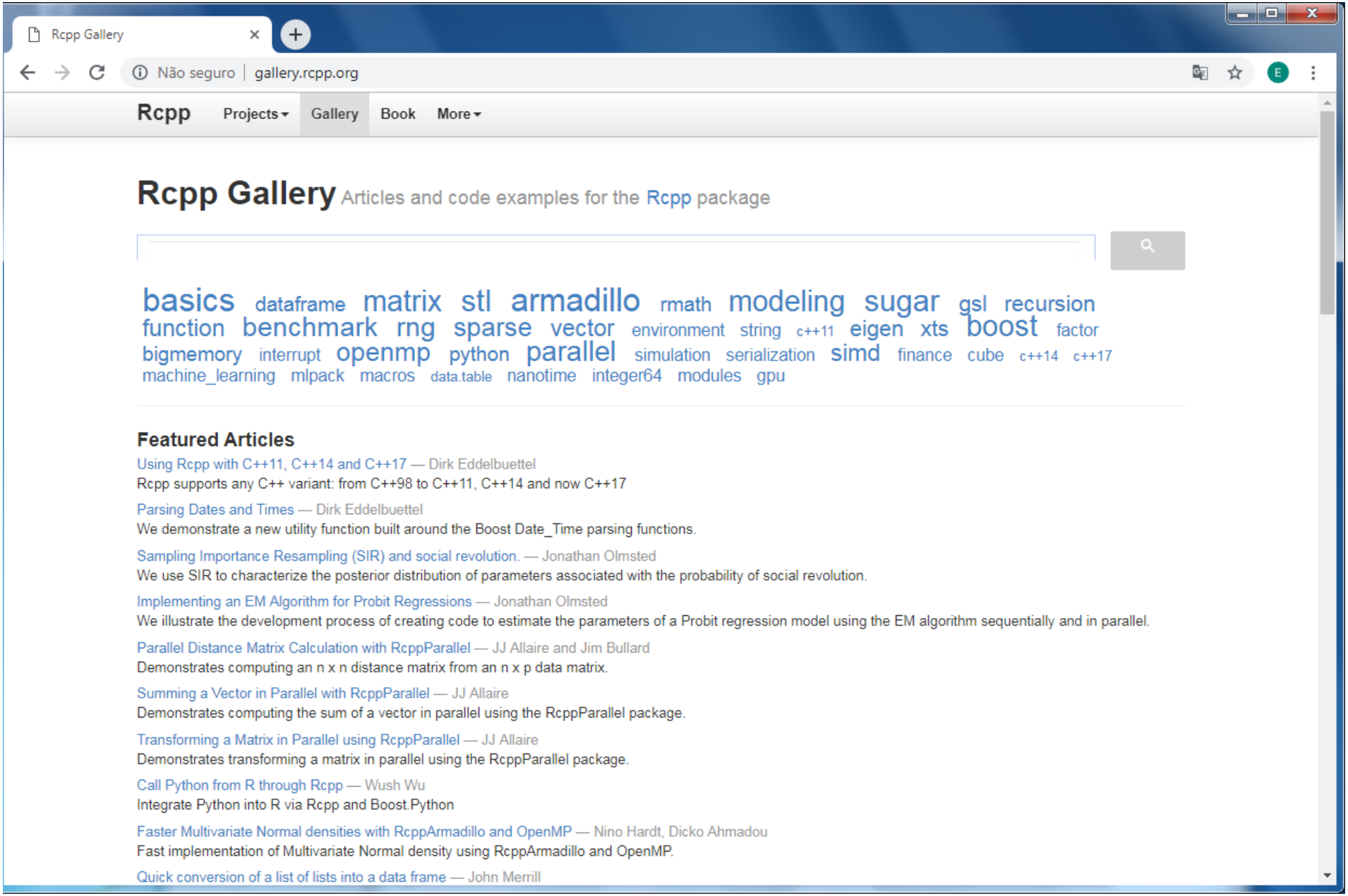


Exercícios

Sites para aprendizagem



Sites para aprendizagem



Obrigado!

Referências I

Eddulbuettel, D., François, R. (2011). Rcpp:Seamelless 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., Eddulbuettel, D., François, R. (2012). Inline C, C++, Fortran function calls for R. URL http://CRAN.R-Project.org/ package=inline, R package version 0.3.10