

Rcpp 速查导引

Romain François

Dirk Eddelbuettel

Rcpp 版本 0.9.4 构建于 2011 年 4 月 12 日

重要说明

```
// 如果你遇到编译错误, 请检查是否安装了合适版本
// 的g++. 查看 `Rcpp-FAQ' 获取更多信息.

// 下表许多示例中隐含了如下语句:
using namespace Rcpp;
// inline包会自动添加上句. 故可直接使用:
Rcpp::NumericVector xx(10);
```

建立简单向量

```
SEXP x; std::vector<double> y(10);

// 你从SEXP走来
NumericVector xx(x);

// 指定向量长度 (以0填充)
NumericVector xx(10);
// 为所有元素指定一个默认值
NumericVector xx(10, 2.0);

// 以取值范围构造
NumericVector xx( y.begin(), y.end() );

// 使用 create
NumericVector xx = NumericVector::create
(
    1.0, 2.0, 3.0, 4.0 );
NumericVector yy = NumericVector::create
(
    Named["foo"] = 1.0,
    _["bar"]     = 2.0 ); // Named的简写
```

访问和设定单个元素

```
// 提取单个元素的值
double x0 = xx[0];
double x1 = xx(1);

double y0 = yy["foo"];
double y1 = yy["bar"];

// 设定单个元素的值
xx[0] = 2.1;
xx(1) = 4.2;

yy["foo"] = 3.0;

// 扩展向量大小
yy["foobar"] = 10.0;
```

使用矩阵

```
// 从SEXP初始化,
// 维数将被自动处理
SEXP x;
NumericMatrix xx(x);

// 4行5列的矩阵 (以0填充)
NumericMatrix xx(4, 5);

// 填充指定值
int xsize = xx.nrow() * xx.ncol();
for (int i = 0; i < xsize; i++) {
    xx[i] = 7;
}
// 同上, 使用STL中的fill进行填充
std::fill(xx.begin(), xx.end(), 8);

// 将数值赋给单个元素
// (第1行, 第2列)
xx(0,1) = 4;

// 引用第2列
// 将改变传递到xx (Row同理)
NumericMatrix::Column zzcol = xx( _, 1);
zzcol = zzcol * 2;

// 复制第2列到新对象
NumericVector zz1 = xx( _, 1);
// 复制子矩阵(左上角3x3)到新对象
NumericMatrix zz2 = xx( Range(0,2),
    Range(0,2));
```

使用 inline 包

```
## 注意 - 以下为R代码. inline 包允许快速测试.
require(inline)
testfun = cxxfunction(
    signature(x="numeric", i="
        integer"),
    body = '
        NumericVector xx(x);
        int ii = as<int>(i);
        xx = xx * ii;
        return( xx );
    ', plugin="Rcpp")
testfun(1:5, 3)
```

和 R 的接口

```
## 在R中，生成一个包的框架。  
## 详情参见"Writing R Extensions"手册。  
  
Rcpp.package.skeleton("myPackage")  
  
## 添加R代码到包中 R/ 目录。调用C++函数。  
## 在R中做类型检查。  
  
myfunR = function(Rx, Ry) {  
  ret = .Call("myCfun", Rx, Ry,  
             package="myPackage")  
  return(ret)  
}  
  
// 添加C++代码到包中 src/ 目录。  
using namespace Rcpp;  
// 使用 RcppExport 定义外部函数  
RcppExport SEXP myCfun(SEXP x, SEXP y) {  
  // 若R/C++的类型匹配，使用指向x的指针。  
  // 指针速度更快，但对xx的修改会传递给R  
  // (xx -> x == Rx).  
  NumericVector xx(x);  
  // clone速度更慢且会消耗额外的内存。  
  // 但安全稳妥，类似于R的做法。  
  NumericVector yy(clone(y));  
  xx[0] = yy[0] = -1.5;  
  int zz = xx[0];  
  // 使用wrap()来返回非SEXP对象，例如：  
  // return(wrap(zz));  
  // 建立并返回一个列表  
  List ret;  
  ret["x"] = xx;  
  ret["y"] = yy;  
  return(ret);  
}  
  
## 在shell中，于包目录下执行  
R CMD check myPackage ## 此句可选  
R CMD INSTALL myPackage  
  
## 在R中：  
require(myPackage)  
aa = 1.5; bb = 1.5; cc = myfunR(aa, bb)  
aa == bb ## FALSE, C++修改了aa  
aa = 1:2; bb = 1:2; cc = myfunR(aa, bb)  
identical(aa, bb)  
## TRUE, R/C++类型不匹配
```

环境

```
Environment stats("package:stats");  
Environment env( 2 ); // 依位置访问  
  
// 特殊环境  
Environment::Rcpp_namespace();  
Environment::base_env();  
Environment::base_namespace();  
Environment::global_env();  
Environment::empty_env();  
  
Function rnorm = stats["rnorm"];  
glob["x"] = "foo";  
glob["y"] = 3;  
std::string x = glob["x"];  
  
glob.assign( "foo" , 3 );  
int foo = glob.get( "foo" );  
int foo = glob.find( "foo" );  
CharacterVector names = glob.ls()  
bool b = glob.exists( "foo" );  
glob.remove( "foo" );  
  
glob.lockBinding("foo");  
glob.unlockBinding("foo");  
bool b = glob.bindingIsLocked("foo");  
bool b = glob.bindingIsActive("foo");  
  
Environment e = stats.parent();  
Environment e = glob.new_child();
```

STL 接口

```
std::accumulate( xx.begin(), xx.end(),  
                std::plus<double>(), 0.0 );  
int n = xx.size();
```

函数

```
Function rnorm("rnorm");  
rnorm(100, _["mean"] = 10.2, _["sd"] =  
      3.2 );
```

Rcpp sugar

```
// 详见文档 `Rcpp syntactic sugar`  
NumericVector x = NumericVector::create(  
  -2.0, -1.0, 0.0, 1.0, 2.0 );  
IntegerVector y = IntegerVector::create(  
  -2, -1, 0, 1, 2 );  
  
NumericVector xx = abs( x );  
IntegerVector yy = abs( y );  
  
bool b = all( x < 3.0 ).is_true() ;  
bool b = any( y > 2 ).is_true();  
  
NumericVector xx = ceil( x );  
NumericVector xx = ceiling( x );  
NumericVector yy = floor( y );  
NumericVector yy = floor( y );  
  
NumericVector xx = exp( x );  
NumericVector yy = exp( y );  
  
NumericVector xx = head( x, 2 );  
IntegerVector yy = head( y, 2 );  
  
IntegerVector xx = seq_len( 10 );  
IntegerVector yy = seq_along( y );  
  
NumericVector xx = rep( x, 3 );  
NumericVector xx = rep_len( x, 10 );  
NumericVector xx = rep_each( x, 3 );  
  
IntegerVector yy = rev( y );
```

有关随机数的函数

```
// 设置随机数种子  
RNGScope scope;  
  
// 详情参见6.7.1节 -- `Writing R Extensions`  
// 手册中关于分布函数的部分. 有些情况下 (如  
// rnorm), 和分布有关的参数可以略去不写;  
// 当不确定时, 请写全所有参数. 推荐在这些  
// 参数中使用双精度型变量而不是整型变量.  
// 除非特别指定, 默认的 log = FALSE.  
  
// 等价于R中的调用  
NumericVector xx = runif(20);  
NumericVector xx1 = rnorm(20);  
NumericVector xx1 = rnorm(20, 0);  
NumericVector xx1 = rnorm(20, 0, 1);  
  
// 分位点向量示例  
NumericVector quants(5);  
for (int i = 0; i < 5; i++) {  
  quants[i] = (i-2);  
}  
  
// 在R中, dnorm(-2:2)  
NumericVector yy = dnorm(quants) ;  
NumericVector yy = dnorm(quants, 0.0,  
  1.0) ;  
  
// 在R中, dnorm(-2:2, mean=2, log=TRUE)  
NumericVector yy = dnorm(quants, 2.0,  
  true) ;  
  
// 注意 - 不指定 mean 时不能单独指定 sd  
// 在R中, dnorm(-2:2, mean=0, sd=2, log=  
// TRUE)  
NumericVector yy = dnorm(quants, 0.0,  
  2.0, true) ;  
  
// 要获取原始的R API, 使用 Rf_*  
double zz = Rf_rnorm(0, 2);
```