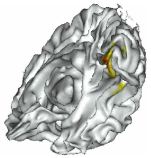
An introduction to the interface between C/C++ and R, and to the writing of R packages under Windows XP Pro

Pierre Lafaye de Micheaux

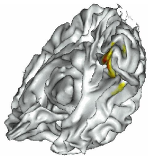
Short course UCL Institut de Statistique

3-4 april 2008



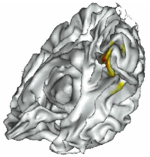
Why a package

- Give to the statistical community (or to your students) a set of R functions all packed in a ready and easy to use tool.
- You have made some theoretical research and you want end users to be able to use it on real data.



Why to use C/C++ code

- Improve speed of execution of your own R code.
- Use R graphical capabilities on numerical results given by your C compiled code.
- Use an already existing C statistical routine code in R.

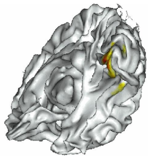


Softwares already installed

Disk size requested for all the softwares to be installed : 2Go

The following softwares have already been installed on your computer :

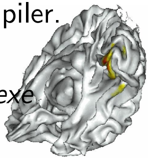
- A recent version of **R** (not a beta one)
- An R code editor : **Tinn-R**
- LaTeχ for windows : **MikTeX**
- A Postscript interpreter : **Ghostscript**
- A Postscript reader : **Ghostview**
- A PDF reader : **Acrobat Reader**
- A LaTeχ editor : **TeXnicCenter**



Softwares to install

We will now install together the following ones :

- the **Rtools** : download and install the Rtools file related to your R version. It contains a minimal subset of GNU Linux tools, and also minGW and Perl. `http://www.murdoch-sutherland.com/Rtools/`
- **Insight** : download the .exe file. `http://sourceforge.net/projects/mingw/`
- **Code : :Blocks** : take the binary with the MinGW compiler. `http://www.codeblocks.org/downloads/5`
- **Microsoft HTML compiler** : search the file *htmlhelp.exe* from `http://www.microsoft.com/downloads`

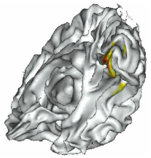


The PATH variable

You should modify the PATH variable to add this at the beginning :

```
C:\insight\bin;C:\Program Files\CodeBlocks\bin;
C:\Program Files\R\R-2.6.1\bin;
```

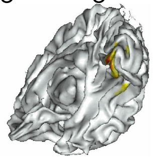
(do not delete what was already written).



The *combn* function

Generate all combinations of the elements of `1:5` of size 3.

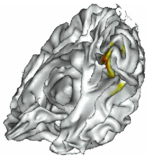
```
> combn(1:5,3)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    1    1    1    1    1    2    2    2     3
[2,]    2    2    2    3    3    4    3    3    4     4
[3,]    3    4    5    4    5    5    4    5    5     5
```



A slow function

Takes more than 40 seconds !

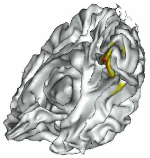
```
> system.time(x<-combn(1:200,3))
user       system     elapsed
44.496     0.650      48.954
```



A C++ combn function

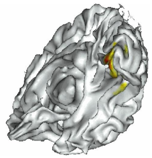
Let's us program our own combn function in C++. Create on Desktop a file called `main.cpp` containing this code (2 slides) :

```
//Fonction main
#include <iostream>
using namespace std;
#include <math.h>
extern "C" {
   int main()
  {
    void moncombn(int *combmat, int *n, int *m);
    int *n, *m, *combmat, j;
```



A C++ combn function - continued

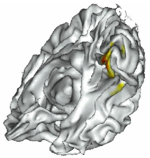
```
     double Cnm;
     n = new int[1];
     m = new int[1];
     *(n+0)=5;
    *(m+0)=3;
     Cnm=10;
     combmat = new int[(int)Cnm**(m+0)];
     moncombn(combmat,n,m);
     for (j = 1; j <= Cnm**(m+0); j++)  {
        cout << *(combmat+j-1) << " ";}  }
 } // extern C
```



A C++ combn function - continued

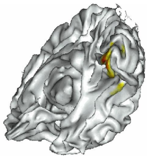
Create a file called `moncombn.cpp` containing this code (3 slides) :

```
extern "C" {
void moncombn(int *combmat, int *n, int *m)
{
  int i, j, e, h, nmmp1, mp1;
  int *a;
  a=new int[*(m+0)];
  for (i=1;i<=*(m+0);i=i+1) *(a+i-1)=i;
  e=0;
  h=*(m+0);
  for (i=1;i<=*(m+0);i=i+1) *(combmat+i-1)=i;
  i=2;
  nmmp1=*(n+0) - *(m+0) + 1;
```



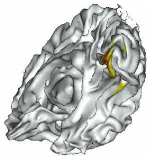
A C++ combn function - continued

```
  mp1=*(m+0) + 1;
while(*(a+0) != nmmp1) {
if(e < *(n+0) - h) {
  h=1;
  e=*(a+*(m+0)-1);
  *(a+*(m+0) - h)=e + 1;
  for (j=1;j<=*(m+0);j=j+1) {
    *(combmat+(i-1)**(m+0)+j-1)=*(a+j-1);}
i=i+1;
}
else {
  h=h + 1;
  e=*(a+mp1 - h-1);
```



A C++ combn function - continued

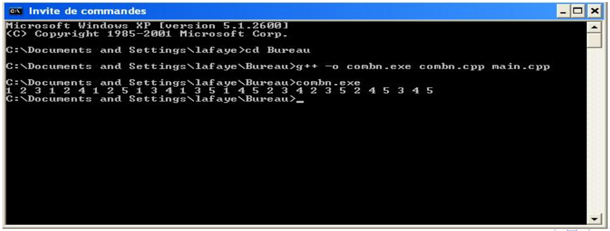
```
  for (j=1;j<=h;j=j+1) *(a+*(m+0) - h + j-1)=e + j;
for (j=1;j<=*(m+0);j=j+1) {
    *(combmat+(i-1)**(m+0)+j-1)=*(a+j-1);}
i=i + 1;
  }
}
//On libere de la memoire
delete[] a;
}
} // extern C
```



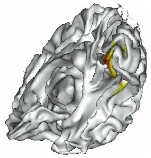
A C++ combn function - continued

Compile and run your code

```
cd Bureau
g++ -o moncombn.exe moncombn.cpp main.cpp
moncombn.exe
exit
```



```
Microsoft Windows XP [version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\lafaye>cd Bureau

C:\Documents and Settings\lafaye\Bureau>g++ -o combn.exe combn.cpp main.cpp

C:\Documents and Settings\lafaye\Bureau>combn.exe
1 2 3 1 2 4 1 2 5 1 3 4 1 3 5 1 4 5 2 3 4 2 3 5 2 4 5 3 4 5
C:\Documents and Settings\lafaye\Bureau>_
```



A C++ combn function - continued

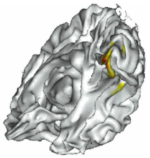
Modify your code to compute `moncombn(1:200,3)` :

```
*(n+0)=200;
Cnm=1313400;
//    for (j = 1; j <= Cnm**(m+0); j++) {
//       cout << *(combmat+j-1) << " ";}
```

Then compile and run your code

```
cd Bureau
g++ -o moncombn.exe moncombn.cpp main.cpp
moncombn.exe
exit
```

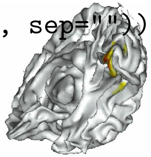
Note that it is very fast ! (compared to the 48 seconds in R).



Calling our C++ code from R

Create the file `moncombn.R` :

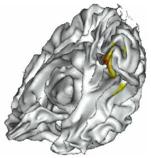
```
moncombn <- function(n,m) {
combmat<-matrix(0,nrow=m,ncol=choose(n,m))
dyn.load(paste("moncombn", .Platform$dynlib.ext, sep=""))
out <- .C("moncombn",res=as.integer(combmat),as.integer(n)
         ,as.integer(m))
combmat<-matrix(out$res,nrow=m,byrow=F)
dyn.unload(paste("moncombn",.Platform$dynlib.ext, sep=""))
return(combmat)
}
```



Calling our C++ code from R - continued

Create the DLL `moncombn.dll` :

```
cd Bureau
g++ -c moncombn.cpp -o moncombn.o
g++ -shared -o moncombn.dll moncombn.o
exit
```

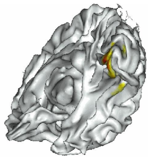


Calling our C++ code from R - continued

Launch R and issue the following commands :

```
setwd("C:/Documents and Settings/lafaye/Bureau")
# change lafaye to your name
source("moncombn.R")
combn(5,3)
moncombn(5,3)
system.time(x<-combn(200,3))
system.time(x<-moncombn(200,3))
```

CONGRATULATIONS !



Exercice

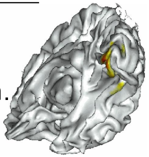
Code the following function `ar1sim` in R and in C :
Arguments :

- a vector $\boldsymbol{\epsilon} = (\epsilon_1, \ldots, \epsilon_n)$, for example *iid* $N(0,1)$ r.v.
- a real $\phi \in (-1,1)$, for example $\phi = 0.75$

Value : the vector $\mathbf{x}$ such that $\forall i = 2, \ldots, n$ $x_i = \phi x_{i-1} + \epsilon_i$

Compare R and C speed by means of a plot of $(n, time_n)$ values for $n = 1000, 2000, \ldots, 100000$.

Note : R function `arima.sim` is almost as fast as our ar1sim. This is because it is coded in C.



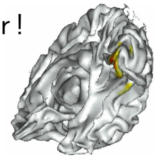
Adding an error in R code

Modify your `moncombn.R` file to add an error

```
fix(moncombn)
combmat<matrix(out$res,nrow=m,byrow=F)
```

Save it by issuing CTRL+S.

Note that we changed <- to <. This is a hard-to-detect error !

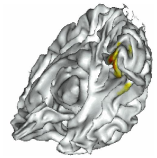


Finding an error in R code

Now launch your code :

```
> moncombn(5,3)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    0    0    0    0    0    0    0     0
[2,]    0    0    0    0    0    0    0    0    0     0
[3,]    0    0    0    0    0    0    0    0    0     0
```

You see there is a problem, but how to find it ?



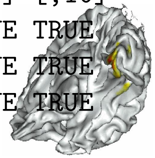
Debugging our R code

```
install.packages("debug")
require(debug)
mtrace(moncombn)
moncombn(5,3)
```

Press [ENTER] key until you reach

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[2,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
[3,] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

This is very strange ! So we found the error !



Detecting an error in C/C++ code

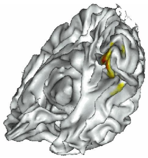
Repair the previous error in R code : `fix(moncombn)`.

We have seen that `mtrace` does not step into the following call :

```
.C("moncombn",res=as.integer(combmat),as.integer(n),
                 as.integer(m))
```

Now, add an error in your C/C++ code then recompile it (under Dos) using the debugging compiler flag : -g.

```
g++ -c moncombn.cpp -o moncombn.o -g
g++ -shared -o moncombn.dll moncombn.o
```



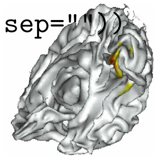
Debugging your R/C++ code

Under Dos, type : `insight Rgui.exe`

then click the Run button :

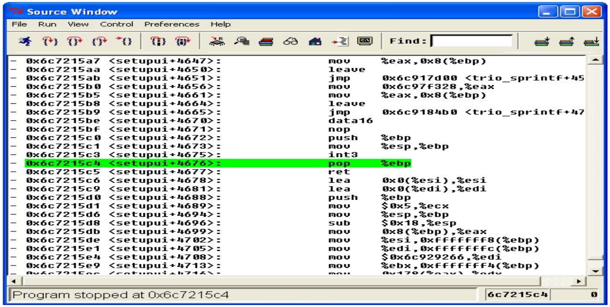
Issue the following commands in R console :

```
setwd("C:/Documents and Settings/lafaye/Bureau")
source("moncombn.R")
dyn.load(paste("moncombn",.Platform$dynlib.ext, sep=""))
```

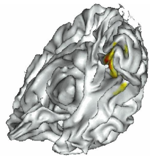


Debugging your R/C++ code - continued

Go to the R menu Misc then Interrompre vers le debuggeur.
We are now in the Insight window.



```
– 0x6c7215a7 <setupui+4647>:          mov     %eax,0x8(%ebp)
– 0x6c7215aa <setupui+4650>:          leave
– 0x6c7215ab <setupui+4651>:          jmp     0x6c917d00 <trio_sprintf+45
– 0x6c7215b0 <setupui+4656>:          mov     0x6c97f328,%eax
– 0x6c7215b5 <setupui+4661>:          mov     %eax,0x8(%ebp)
– 0x6c7215b8 <setupui+4664>:          leave
– 0x6c7215b9 <setupui+4665>:          jmp     0x6c9184b0 <trio_sprintf+47
– 0x6c7215be <setupui+4670>:          data16
– 0x6c7215bf <setupui+4671>:          nop
– 0x6c7215c0 <setupui+4672>:          push    %ebp
– 0x6c7215c1 <setupui+4673>:          mov     %esp,%ebp
– 0x6c7215c3 <setupui+4675>:          int3
– 0x6c7215c4 <setupui+4676>:          pop     %ebp
– 0x6c7215c5 <setupui+4677>:          ret
– 0x6c7215c6 <setupui+4678>:          lea     0x0(%esi),%esi
– 0x6c7215c9 <setupui+4681>:          lea     0x0(%edi),%edi
– 0x6c7215d0 <setupui+4688>:          push    %ebp
– 0x6c7215d1 <setupui+4689>:          mov     $0x5,%ecx
– 0x6c7215d6 <setupui+4694>:          mov     %esp,%ebp
– 0x6c7215d8 <setupui+4696>:          sub     $0x18,%esp
– 0x6c7215db <setupui+4699>:          mov     0x8(%ebp),%eax
– 0x6c7215de <setupui+4702>:          mov     %esi,0xfffffff8(%ebp)
– 0x6c7215e1 <setupui+4705>:          mov     %edi,0xfffffffc(%ebp)
– 0x6c7215e4 <setupui+4708>:          mov     $0x6c929266,%edi
– 0x6c7215e9 <setupui+4713>:          mov     %ebx,0xfffffff4(%ebp)

Program stopped at 0x6c7215c4                      6c7215c4      0
```



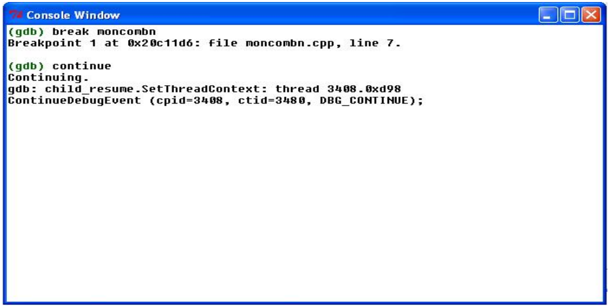
Debugging your R/C++ code - continued

Under Insight, go to menu View - Console [CTRL+N]

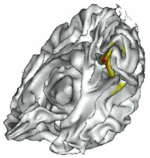
Add a breakpoint : `break moncombn`

and type : `continue`

in the GDB console. Then press [ENTER].



```
(gdb) break moncombn
Breakpoint 1 at 0x20c11d6: file moncombn.cpp, line 7.

(gdb) continue
Continuing.
gdb: child_resume.SetThreadContext: thread 3408.0xd98
ContinueDebugEvent (cpid=3408, ctid=3480, DBG_CONTINUE);
```

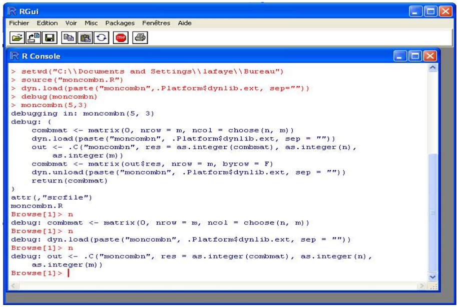


Debugging your R/C++ code - continued

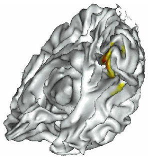
We are back in R ! Now type :

```
debug(moncombn)
moncombn(5,3)
```

Use `n` to step into your R code until you reach the C++ call.

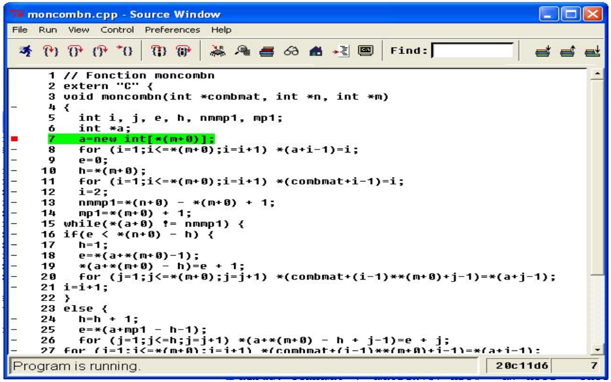


```
> setwd("C:\\Documents and Settings\\lafaye\\Bureau")
> source("moncombn.R")
> dyn.load(paste("moncombn",.Platform$dynlib.ext, sep=""))
> debug(moncombn)
> moncombn(5,3)
debugging in: moncombn(5, 3)
debug: {
    combmat <- matrix(0, nrow = m, ncol = choose(n, m))
    dyn.load(paste("moncombn", .Platform$dynlib.ext, sep = ""))
    out <- .C("moncombn", res = as.integer(combmat), as.integer(n),
        as.integer(m))
    combmat <- matrix(out$res, nrow = m, byrow = F)
    dyn.unload(paste("moncombn", .Platform$dynlib.ext, sep = ""))
    return(combmat)
}
attr(,"srcfile")
moncombn.R
Browse[1]> n
debug: combmat <- matrix(0, nrow = m, ncol = choose(n, m))
Browse[1]> n
debug: dyn.load(paste("moncombn", .Platform$dynlib.ext, sep = ""))
Browse[1]> n
debug: out <- .C("moncombn", res = as.integer(combmat), as.integer(n),
    as.integer(m))
Browse[1]>
```

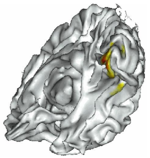


Debugging your R/C++ code - continued

We are back into Insight !

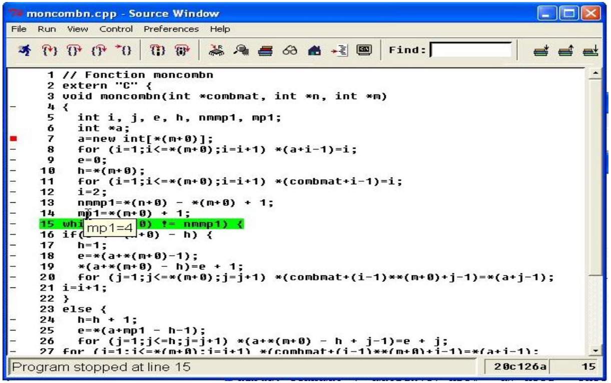

moncombn.cpp - Source Window
File Run View Control Preferences Help
Find:
Program is running.
20c11d6
7

```
// Fonction moncombn
extern "C" {
void moncombn(int *combmat, int *n, int *m)
{
  int i, j, e, h, nmmp1, mp1;
  int *a;
  a=new int[*(m+0)];
  for (i=1;i<=*(m+0);i=i+1) *(a+i-1)=i;
  e=0;
  h=*(m+0);
  for (i=1;i<=*(m+0);i=i+1) *(combmat+i-1)=i;
  i=2;
  nmmp1=*(n+0) - *(m+0) + 1;
  mp1=*(m+0) + 1;
while(*(a+0) != nmmp1) {
if(e < *(n+0) - h) {
  h=1;
  e=*(a+*(m+0)-1);
  *(a+*(m+0) - h)=e + 1;
  for (j=1;j<=*(m+0);j=j+1) *(combmat+(i-1)**(m+0)+j-1)=*(a+j-1);
i=i+1;
}
else {
  h=h + 1;
  e=*(a+mp1 - h-1);
  for (j=1;j<=h;j=j+1) *(a+*(m+0) - h + j-1)=e + j;
for (i=1;i<=*(m+0);i=i+1) *(combmat+(i-1)**(m+0)+j-1)=*(a+j-1);
```



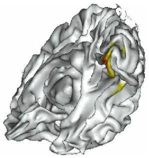
Debugging your R/C++ code - continued

Use to step through your C++ code and look at variables.



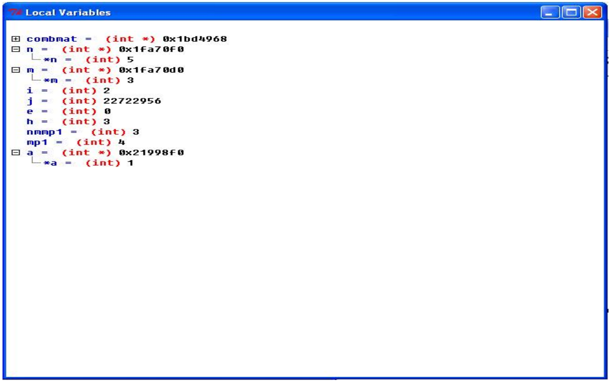
```
1 // Fonction moncombn
 2 extern "C" {
 3 void moncombn(int *combmat, int *n, int *m)
 4 {
 5   int i, j, e, h, nmmp1, mp1;
 6   int *a;
 7   a=new int[*(m+0)];
 8   for (i=1;i<=*(m+0);i=i+1) *(a+i-1)=i;
 9   e=0;
10   h=*(m+0);
11   for (i=1;i<=*(m+0);i=i+1) *(combmat+i-1)=i;
12   i=2;
13   nmmp1=*(n+0) - *(m+0) + 1;
14   mp1=*(m+0) + 1;
15 whi mp1=4 0) != nmmp1) {
16 if(     +0) - h) {
17   h=1;
18   e=*(a+*(m+0)-1);
19   *(a+*(m+0) - h)=e + 1;
20   for (j=1;j<=*(m+0);j=j+1) *(combmat+(i-1)*(m+0)+j-1)=*(a+j-1);
21 i=i+1;
22 }
23 else {
24   h=h + 1;
25   e=*(a+mp1 - h-1);
26   for (j=1;j<=h;j=j+1) *(a+*(m+0) - h + j-1)=e + j;
27 for (i=1;i<=*(m+0);i=i+1) *(combmat+(i-1)*(m+0)+i-1)=*(a+i-1);
```

```
Program stopped at line 15        20c126a        15
```

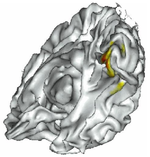


Debugging your R/C++ code - continued

You can also use menu View - Local Variable [CTRL+L] to look at the values of all the variables.



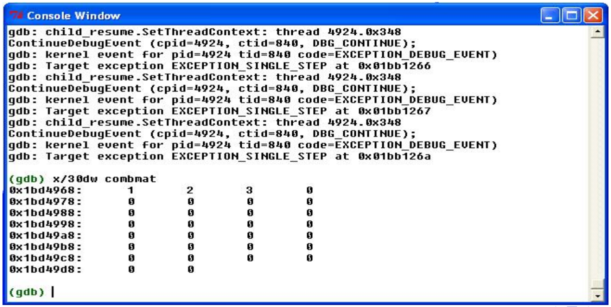
```
⊞ combmat =  (int *) 0x1bd4968
⊟ n =  (int *) 0x1fa70f0
   └*n =  (int) 5
⊟ m =  (int *) 0x1fa70d0
   └*m =  (int) 3
  i =  (int) 2
  j =  (int) 22722956
  e =  (int) 0
  h =  (int) 3
  nmmp1 =  (int) 3
  mp1 =  (int) 4
⊟ a =  (int *) 0x21998f0
   └*a =  (int) 1
```



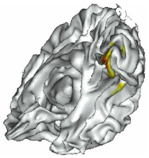
Debugging your R/C++ code - continued

If you want to look at a C++ array of values (matrix or vector in R), go to GDB console and issue for example :

```
x/30dw combmat
```



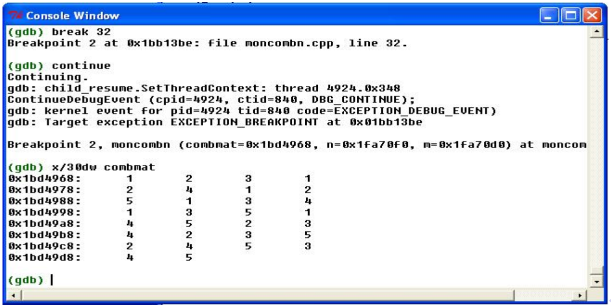
```
gdb: child_resume.SetThreadContext: thread 4924.0x348
ContinueDebugEvent (cpid=4924, ctid=840, DBG_CONTINUE);
gdb: kernel event for pid=4924 tid=840 code=EXCEPTION_DEBUG_EVENT)
gdb: Target exception EXCEPTION_SINGLE_STEP at 0x01bb1266
gdb: child_resume.SetThreadContext: thread 4924.0x348
ContinueDebugEvent (cpid=4924, ctid=840, DBG_CONTINUE);
gdb: kernel event for pid=4924 tid=840 code=EXCEPTION_DEBUG_EVENT)
gdb: Target exception EXCEPTION_SINGLE_STEP at 0x01bb1267
gdb: child_resume.SetThreadContext: thread 4924.0x348
ContinueDebugEvent (cpid=4924, ctid=840, DBG_CONTINUE);
gdb: kernel event for pid=4924 tid=840 code=EXCEPTION_DEBUG_EVENT)
gdb: Target exception EXCEPTION_SINGLE_STEP at 0x01bb126a

(gdb) x/30dw combmat
0x1bd4968:      1       2       3       0
0x1bd4978:      0       0       0       0
0x1bd4988:      0       0       0       0
0x1bd4998:      0       0       0       0
0x1bd49a8:      0       0       0       0
0x1bd49b8:      0       0       0       0
0x1bd49c8:      0       0       0       0
0x1bd49d8:      0       0

(gdb) |
```



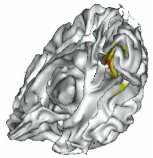
Debugging your R/C++ code - continued

Now type in the GDB console :

```
break 32
continue
x/30dw combmat
```



```
(gdb) break 32
Breakpoint 2 at 0x1bb13be: file moncombn.cpp, line 32.

(gdb) continue
Continuing.
gdb: child_resume.SetThreadContext: thread 4924.0x348
ContinueDebugEvent (cpid=4924, ctid=840, DBG_CONTINUE);
gdb: kernel event for pid=4924 tid=840 code=EXCEPTION_DEBUG_EVENT)
gdb: Target exception EXCEPTION_BREAKPOINT at 0x01bb13be

Breakpoint 2, moncombn (combmat=0x1bd4968, n=0x1fa70f0, m=0x1fa70d0) at moncom

(gdb) x/30dw combmat
0x1bd4968:      1       2       3       1
0x1bd4978:      2       4       1       2
0x1bd4988:      5       1       3       4
0x1bd4998:      1       3       5       1
0x1bd49a8:      4       5       2       3
0x1bd49b8:      4       2       3       5
0x1bd49c8:      2       4       5       3
0x1bd49d8:      4       5

(gdb) |
```

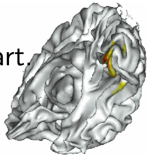


The goal

Create a little package called `bootLin` that will perform the Bootstrap of the residuals of a multiple linear regression. This will enable us to check graphically the normality of coefficients estimators when sample size is great enough.

We will propose both an R function (`bootLinR1`) and a C++ function (`bootLinCpp`) doing exactly the same thing.

We will evaluate the speed performance of C++ versus R part.



The linear model

The linear regression equation is

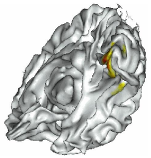
$$\mathbf{Y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where the random variables ϵ_i are *i.i.d.* with mean 0, but are not supposed to be Gaussian.
The least square estimators of $\boldsymbol{\beta}$ are given by

$$\hat{\boldsymbol{\beta}} = (X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}\mathbf{Y}$$

The residuals are given by :

$$\hat{\boldsymbol{\epsilon}} = \mathbf{Y} - X\hat{\boldsymbol{\beta}}.$$



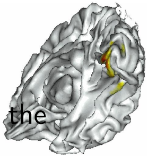
The Bootstrap algorithm

For $b = 1, \ldots, B$;

- Draw with replacement among $\hat{\epsilon}_1, \ldots, \hat{\epsilon}_n$ a sample with size n : $\tilde{\epsilon}_1^{(b)}, \ldots, \tilde{\epsilon}_n^{(b)}$;
- Center these values to obtain : $\epsilon_1^{*(b)} = \tilde{\epsilon}_1^{(b)} - \overline{\tilde{\epsilon}^{(b)}}, \ldots, \epsilon_1^{*(b)} = \tilde{\epsilon}_n^{(b)} - \overline{\tilde{\epsilon}^{(b)}}$;
- Compute $\mathbf{Y}^{*(b)} = X\hat{\boldsymbol{\beta}} + \boldsymbol{\epsilon}^{*(b)}$;
- Compute the least squares estimators $\boldsymbol{\beta}^{*(b)}$ of $\boldsymbol{\beta}$ for the regression :

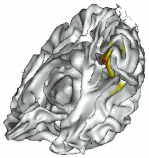
$$\mathbf{Y}^{*(b)} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}^{*(b)}.$$

Then we will draw, for each $i \in \{1, \ldots, p\}$, the histogram of the $\{\beta_i^{*(b)},\ b = 1, \ldots, B\}$.



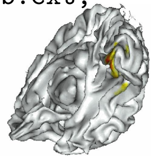
Main bootLin.R file

```
bootLin <- function(y,X,B,method=c("R1","R2","Cpp")) {
betahat <- solve(t(X)%*%X)%*%t(X)%*%y
residus <- y-X%*%as.matrix(betahat)
n <- nrow(X)
p <- length(betahat)-1
res <- matrix(0,nrow=B,ncol=p+1)
if (method[1] == "R1") {
 res <- bootLinR1(residus,B,X,betahat)}
if (method[1] == "R2") {
     require(bootstrap)
     res <- t(bootstrap(x=residus,nboot=B,
      theta=bootLinR2,X,betahat)$thetastar)}
```



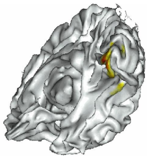
Main bootLin.R file - continued

```
if (method[1] == "Cpp") {
  dyn.load(paste("bootLinCpp", .Platform$dynlib.ext,
                                   sep=""))
  out <- .C("bootLinCpp",betahatstar=as.double(res),
     as.double(residus),as.integer(B),as.double(X),
     as.double(betahat),as.integer(n),as.integer(p))
  res <- matrix(out$betahatstar,nrow=B,ncol=p+1)
  dyn.unload(paste("bootLinCpp", .Platform$dynlib.ext,
                                   sep="")) }
 par(mfrow=c(ceiling(sqrt(p)),ceiling(sqrt(p))))
 apply(res,FUN=hist,MARGIN=2)
return(res)
}
```



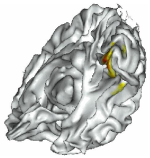
bootLinR1.R file

```
bootLinR1 <- function(residus,B,X,betahat){
n <- nrow(X)
p <- length(betahat)-1
betahatstar<-matrix(0,nrow=B,ncol=p+1)
for (b in 1:B) {
 restilde<-sample(residus,n,replace=TRUE)
 epsstar<-restilde-mean(restilde)
 ystar<-X%*%as.matrix(betahat)+epsstar
 betahatstar[b,]<-solve(t(X)%*%X)%*%t(X)%*%ystar
}
return(betahatstar)
}
```



bootLinR2.R file

```
bootLinR2 <- function(restilde,X,betahat){
epsstar<-restilde-mean(restilde)
ystar<-X%*%as.matrix(betahat)+epsstar
betahatstar<-solve(t(X)%*%X)%*%t(X)%*%ystar
return(betahatstar)
}
```

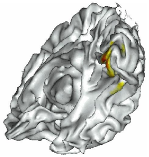


A C++ library

Download `http://www.robertnz.net/ftp/newmat10.zip` and unzip it in `C:/newmat`. Then type in DOS :

```
cd \
cd newmat
g++ -O2 -c *.cpp
ar cr newmat.a *.o
ranlib newmat.a
cp newmat.a newmat.dll
```

The static libraries `newmat.a` and `newmat.dll` are created.

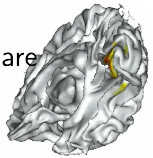


A C++ library - continued

If you want to debug, do also :

```
cd \
cd newmat
g++ -O2 -c *.cpp -g
ar cr newmatdebug.a *.o
ranlib newmatdebug.a
cp newmatdebug.a newmatdebug.dll
```

The static libraries `newmatdebug.a` and `newmatdebug.dll` are created.

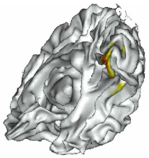


A C++ library - continued

Now download `http://www.robertnz.net/ftp/newran02.zip` and unzip it into `C:/newran`. Then type in DOS :

```
cd \
cd newran
g++ -O2 -c *.cpp -Wno-deprecated
ar cr newran.a *.o
ranlib newran.a
cp newran.a newran.dll
```

The static libraries `newran.a` and `newran.dll` are created.

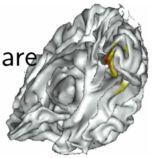


A C++ library - continued

If you want to debug, do also :

```
cd \
cd newran
g++ -O2 -c *.cpp -Wno-deprecated -g
ar cr newrandebug.a *.o
ranlib newrandebug.a
cp newrandebug.a newrandebug.dll
```

The static libraries `newrandebug.a` and `newrandebug.dll` are created.



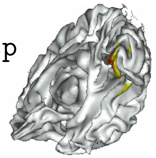
Other C/C++ libraries

Free ones :

`http://www.gnu.org/software/gsl/`
`http://www.math.uiowa.edu/~dstewart/meschach/`
`http://www.robertnz.net/ol_doc.htm` (newmat et newran)
`http://www.nrbook.com/a/bookcpdf.php`
R source code and also R.h

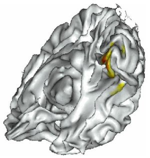
Not free ones :

`http://www.nag.co.uk/numeric/CL/CLdescription.asp`
`http://www.vni.com/products/imsl/c/imslc.php`



The bootLinCpp.cpp file

```
#define WANT_STREAM
#define WANT_MATH
#include "newmatap.h"
#include "newmatio.h"
#include "include.h"
#include "newran.h"
#ifdef use_namespace
using namespace NEWMAT;
#endif
#ifdef use_namespace
using namespace NEWRAN;
#endif
```

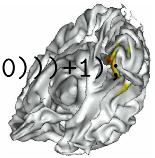


The bootLinCpp.cpp file - continued

```
extern "C" {
void bootLinCpp(double *betahatstar,double *residus,
     int *B,double *X,double *betahat,int *n,int *p) {
int i,j,b;
double temp;
Matrix betahatstarmat(*(B+0),*(p+0)+1), Xmat(*(n+0),
                                          *(p+0)+1);
ColumnVector residusvec(*(n+0)), restildevec(*(n+0)),
                                     epsstar(*(n+0));
ColumnVector betahatvec(*(p+0)+1);
ColumnVector invec(*(n+0)), outn(*(n+0)), ystar(*(n+0)),
             outp(*(p+0)+1), outB(*(B+0)), out(*(n+0));
Random::Set(0.46875);
```

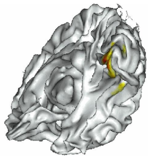
The bootLinCpp.cpp file - continued

```
Uniform Unif;
for (i=1;i<=*(n+0);i=i+1)  residusvec(i) = *(residus+i-1);
for (i=1;i<=*(p+0)+1;i=i+1) betahatvec(i)= *(betahat+i-1);
for (j=1;j<=*(p+0)+1;j=j+1) {
for(i=1;i<=*(n+0);i=i+1)Xmat(i,j)=*(X+(*(n+0))*(j-1)+i-1);
                            }
for (b=1;b<=*(B+0);b=b+1) {
 for (i=1;i<=*(n+0);i=i+1) out(i) = Unif.Next();
 for (i=1;i<=*(n+0);i=i+1) {
  restildevec(i) = residusvec((int)(out(i)*(*(n+0)))+1);
 }
 temp = restildevec.Sum()/(*(n+0));
 epsstar = restildevec - temp;
```



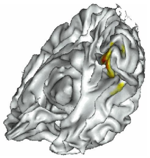
The bootLinCpp.cpp file - continued

```
 ystar = Xmat*betahatvec+epsstar;
 outp = ((Xmat.t()*Xmat).i())*Xmat.t()*ystar;
 betahatstarmat.Row(b) = outp.AsRow();
                        }
for (j=1;j<=*(p+0)+1;j=j+1) {
  for (b=1;b<=*(B+0);b=b+1) {
   *(betahatstar+(*(B+0))*(j-1)+b-1) =
             betahatstarmat(b,j);}
                          }
```



The bootLinCpp.cpp file - continued

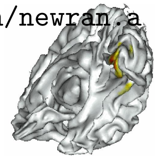
```
residusvec.Release();
restildevec.Release();
epsstar.Release();
betahatvec.Release();
invec.Release();
outn.Release();
ystar.Release();
outp.Release();
outB.Release();
out.Release();
betahatstarmat.Release();
Xmat.Release();
} }
```



Compile bootLinCpp.cpp file

Compile it under DOS :

```
cd %HOMEPATH%/Bureau
g++ -O2 -c bootLinCpp.cpp -o bootLinCpp.o
                    -I"C:/newmat" -I"C:/newran"
g++ -shared -o bootLinCpp.dll bootLinCpp.o
                    -I"C:/newmat" -I"C:/newran"
                    C:/newmat/newmat.a C:/newran/newran.a
```

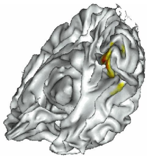


Debugging bootLinCpp.cpp file

```
cd Bureau
g++ -O2 -c bootLinCpp.cpp -o bootLinCpp.o
           -I"C:/newmat" -I"C:/newran" -g
g++ -shared -o bootLinCpp.dll bootLinCpp.o
          -I"C:/newmat" -I"C:/newran"
          C:/newmat/newmatdebug.a C:/newran/newrandebug.a
```

And for an example of an array values under GDB :

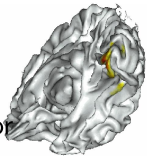
```
x/3fg residus
x/3fg residusvec->store
```



Testing our program

```
source("bootLin.R")
source("bootLinR1.R")
source("bootLinR2.R")
B <- 10000 ; n <- 100 ; p <- 10
truebeta<-floor(runif(p+1)*10)+1
eps<-rchisq(n,df=4)
X<-cbind(rep(1,n),matrix(rnorm(n*p),nrow=n,ncol=p))
y<-X%*%as.matrix(truebeta)+eps
system.time(res<-bootLin(y,X,B,method="R1"))
system.time(res<-bootLin(y,X,B,method="Cpp"))
```

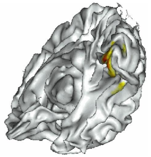
Not so much speed improvement, but could be far greater for double Bootstrap !



A first modification

Comment out the `dyn.load` and `dyn.unload` lines in the `bootLin` R function.

Add the last argument
`PACKAGE="bootLin"`
to the `.C("bootLinCpp",.....)` call.



Create the directory tree

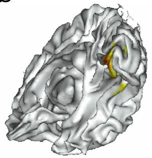
We should create a directory called `BootLin` containing two files :

```
DESCRIPTION
INDEX
```

and four sub-directories :

```
inst   << files CITATION and HISTORY
man    << containing the *.Rd documentation files
R      << containing the *.R programs
src    << containing the *.h and *.cpp files
```

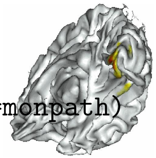
See next slide for how to proceed.



Create the directory tree - continued

Launch R and type :

```
rm(list=ls())
monpath <- paste(Sys.getenv("HOMEPATH"),"\\Bureau",sep="")
# Replace Bureau by Desktop
# if you have an english votre OS
setwd(monpath)
source("bootLin.R")
source("bootLinR1.R")
source("bootLinR2.R")
package.skeleton(name="bootLin",
 list=c("bootLin","bootLinR1","bootLinR2"),path=monpath)
q("no")
```



Create the directory tree - continued

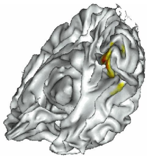
Go to `bootLin` directory and create the two sub-directories :

```
inst
src
```

Then put a copy of the file `bootLinCpp.cpp` in the sub-directory `src`
and delete the file `bootLin-package.Rd` in the sub-directory `man`.

To remove the CR/LF combinations, under Dos, type in :

```
cd %HOMEPATH%/Bureau/bootLin/src
dos2unix bootLinCpp.cpp
```



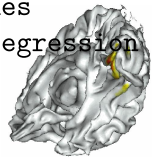
Creating supplementary files

Edit and modify the DESCRIPTION file, for example to add the following line

```
Depends: R (>= 2.6.0), bootstrap
```

Delete the file Read-and-delete-me.
Create a file called INDEX which should contain

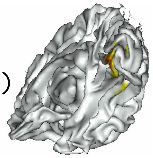
```
bootLin the main function to call the 3 other ones
bootLinR1 my code in R to perform Bootstrap in regression
bootLinR2 another code in R
```



Creating supplementary files - continued

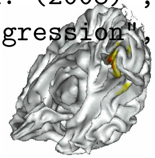
The CITATION file in inst directory will contain

```
citHeader("To cite the bootLin package in
                           publications use:")


citEntry(entry="Article",
        title = "A fake Bootstrap package for
                                Linear Regression",
        author = personList(as.person("P.
                      Lafaye de Micheaux"),
                      as.person("A. Other one")
),
```



Creating supplementary files - continued

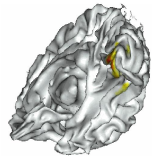
```
    publisher = "The American Statistician",
            year          = 2008,
            note          = "Software: R Package,
                                 bootLin, version 0.1",
     url   = "http://www.biostatisticien.eu/bootLin/",

     textVersion =
     paste("Lafaye De Micheaux, P. and Other one, A. (2008)",
            "A fake Bootstrap package for Linear Regression",
       "The American Statistician",
      "(Software: R Package, bootLin, version 0.1)"
     )
  )
```



Creating supplementary files - continued

The `HISTORY` file in `inst` directory will contain :

```
2008-04-04
----------

Version 0.1 : my first version
```



Creating supplementary files - continued

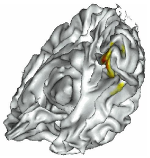
In sub-directory `src`, create ASCII file `Makevars.win` with :

```
PKG_LIBS=-LC:/newmat -lnewmat -LC:/newran -lnewran
CPPFLAGS=-IC:/newmat -IC:/newran
PKG_CPPFLAGS=-IC:/newmat -IC:/newran
```

In sub-directory `R`, create ASCII file `zzz.R` with :

```
.First.lib <- function(lib, pkg){
    library.dynam("bootLin", pkg, lib)
}
```

In sub-directory `man`, edit all the *.Rd files.



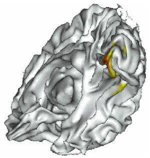
Check and compile

BE CAREFULL ! ! : Be sure to delete `make.exe` and `mkdir.exe` in directory `C:\Program Files\CodeBlocks\bin`

Under DOS :

```
cd %HOMEPATH%/Bureau
cp -R bootLin bootLin-save
R CMD check bootLin
R CMD build --binary --use-zip bootLin
```

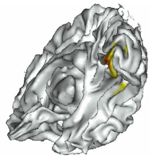
The file package `bootLin_1.0.zip` is created !



Installing the package

```
R CMD INSTALL bootLin
```

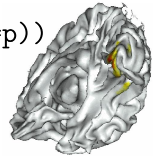
or use the R menu Packages - Installer le(s) package(s) depuis des fichiers zip ...



Using the package

Then launch R and :

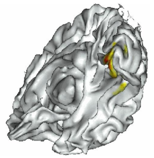
```
require(bootLin)
help(package="bootLin")
B <- 10000
n <- 100
p <- 10
truebeta<-floor(runif(p+1)*10)+1
eps<-rchisq(n,df=4)
X<-cbind(rep(1,n),matrix(rnorm(n*p),nrow=n,ncol=p))
y<-X%*%as.matrix(truebeta)+eps
method <- "Cpp"
system.time(res<-bootLin(y,X,B,method))
```



Deleting the package

Under DOS :

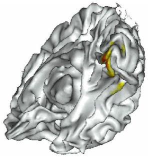
```
R CMD REMOVE bootLin
```



Submitting to CRAN

Be carefull ! Do not submit to CRAN as is because it will not compile due to the fact that the `newmat` and `newran` libraries are not present on CRAN.

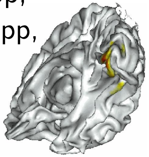
A **solution** is presented on the next two slides.



Submitting to CRAN - continued

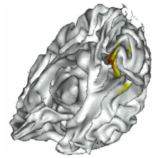
Add the following files from `newran02.zip` and `newmat10.zip` to the `src` directory of bootLin-save :

Newmat	boolean.h, controlw.h, include.h, myexcept.h, newmat.h, newmatap.h, newmatio.h, newmatnl.h, newmatrc.h, newmatrm.h, precisio.h, solution.h, bandmat.cpp, cholesky.cpp, evalue.cpp, fft.cpp, hholder.cpp, jacobi.cpp, myexcept.cpp, newfft.cpp, newmat1.cpp, newmat2.cpp, newmat3.cpp, newmat4.cpp, newmat5.cpp, newmat6.cpp, newmat7.cpp, newmat8.cpp, newmat9.cpp, newmatex.cpp, newmatnl.cpp, newmatrm.cpp, solution.cpp, sort.cpp, submat.cpp, svd.cpp.
Newran	extreal.h, newran.h, extreal.cpp, newran.cpp.



Submitting to CRAN - continued

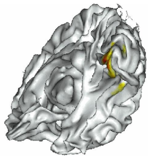
Remove the file `Makevars.win` from `src` directory of bootLin-save.



Submitting to CRAN - continued

Check again to be sure, and build as .tar.gz file :

```
cd %HOMEPATH%/Bureau
cp -R bootLin-save bootLin
R CMD check bootLin
R CMD build bootLin
```



Upload your package to CRAN

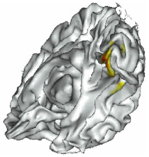
Upload the '.tar.gz' file, using 'anonymous' as log-in name and your e-mail address as password, to ftp ://cran.R-project.org/incoming/ (note : use ftp and not sftp to connect to this server).

Send a message to cran@r-project.org about it :

Dear CRAN maintainers,

I am the new maintainer of package bootLin.
I uploaded a new version today.

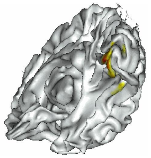
Best regards,
Pierre Lafaye de Micheaux



To go further : R.h and Rmath.h API

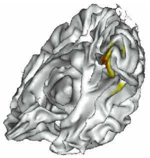
You can use many standards R functions in your C program : `http://cran.r-project.org/doc/manuals/R-exts.html#Numerical-analysis-subroutines`

Here is a simple example in file `monrnom.cpp` using R `rnorm` function.



To go further : R.h and Rmath.h API - continued

```
#include <iostream>
using namespace std;
#include <R.h>
#include "Rmath.h"
extern "C" {
   void monrnorm(double *res, int *n, double *mu,
       double *sigma) {
    int j;
    GetRNGstate();
    for (j = 1; j <= *(n+0); j++) {*(res+j-1) =
          rnorm(*(mu+0),*(sigma+0));}
    PutRNGstate(); }
}
```



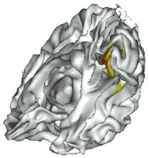
To go further : R.h and Rmath.h API - continued

Under Dos :

```
g++ -c monrnom.cpp -o monrnorm.o
                -I"C:\Program Files\R\R-2.6.1\include"
g++ -shared -o monrnorm.dll monrnorm.o
               -I"C:\Program Files\R\R-2.6.1\include"
               -L"C:\Program Files\R\R-2.6.1\bin" -lR
```

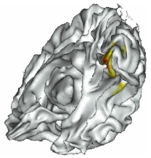
Launch R, change current directory to Desktop and type :

```
dyn.load("monrnorm.dll")
n<-23
res<-rep(0,n)
.C("monrnorm",as.double(res),as.integer(n),
              as.double(1.0),as.double(5.0))
```



To go further

See `SEXP` types (Simple EXPression) in
`http://cran.r-project.org/doc/manuals/R-exts.html`



Thank you for your attention