Some Features of R You Might Not Yet Know

Dr. Wolfgang Rolke

SIDIM 2015

Content

- My general setup
- Startup Options
- Two of my favorite packages:
  - Rcpp
  - Rshiny

My Setup

- I currently work with 2 desktop computers, one laptop and a tablet
- I want to keep all of them synchronized and do this as automatically as possible. That is if I make a change to any of my R projects I want it changed accordingly on all my computers.
- I use Dropbox for this but there are alternatives that would work just as well (OneDrive, Google Drive, SugarSync, and Box.com)

My Setup, cont.

- So on all my computers I have a folder called R, located in the Dropbox folder. This folder contains all my *.Rdata files.
- Each of them contains a routine called sv with

```
sv <- function () {
save.image(paste(getwd(),"/Rfoo.RData",sep=""))
NULL
}
```

and when I do sv() the project gets saved on the local machine, then on Dropbox and finally on all the other computers!

Startup Options

- .Rprofile and .First
- When starting R both of these are executed
- .Rprofile is a text file outside R, usually located in the same folder as the project files. It is a text file with R code. It contains startup options that I want to execute whenever I start R
- .First is a file inside an R project. This file contains things that should be done in this project specifically

.First of ESMA 6665 (Statistical Computing)

- .First <-

```
function() {
        library(multcomp)
        library(leaps)
        library(gam)
        library(lattice)
        library(class)
        library(rpart)
        library(Rcpp)
        library(survival,pos=2)
        library(MASS,pos=2)
        NULL
}
```

Next...

- .First is specific to each .Rdata, but there are some things I always want done at startup, no matter what .Rdata I open
- I make changes to this on occasion, but I want the changes to apply everywhere
- Solution: do these things in .Rprofile
- Like .First this gets executed at startup
- Mine is quite large, so I go through it slowly:

My .Rprofile

```
#Part 1: Set Options ----------------------------#
options(editor=paste(getwd(),"/notepad2.exe",sep=""))
options(help_type="html")
options(show.signif.stars=FALSE)
options(stringsAsFactors=FALSE)
options("repos" = c(CRAN = "http://cran.rstudio.com/"))
```

My .Rprofile

```
#Part 2: Load Libraries ----------------------------#
library("ggplot2")
library(“mvtnorm")
library(“random”) #True random numbers (not pseudo)
                  # from http://random.org.
```

My own routines

- I have a number of small routines I use a lot
- I want them available at all times
- But I don’t want them to appear when I do ls() so that does not get to messy
- Solution: put them in a new environment
- You can see all the environments with search()
- Usually shows attached libraries and dataframes
- But you can make your own!

My .Rprofile

```
#Part 3: Setup of Commonly Used Routines ---#
MyEnv <- new.env()
.MyEnv$h <- utils::head
.MyEnv$ht <- function(d) rbind(head(d,10),tail(d,10))
.MyEnv$hh <- function(d)
        if(class(d)=="matrix"|class(d)=="data.frame") d[1:5,1:5]
.MyEnv$sc <- function() source("clipboard")
.MyEnv$dp <- function(x)  dump(x,"clipboard")
.MyEnv$ip <- function(x) {
        install.packages(x,lib="C:\\R\\library")
        library(x,character.only =T) }
.MyEnv$mcat <- function (a) {cat(paste(paste(a,collapse=" "),"\n"))}
MyEnv$trw <- function(n,low=0,high=1e5)
c(randomNumbers(n,low,high))
```

My .Rprofile

#Part 3: Setup of Commonly Used Routines ---#

gout: function for sending graphs to different output devices

```
.MyEnv$gout <-
function(f,foldername,graphname,format="png") {
    f()
    if(nchar(foldername)==4) #graph for one of my courses
          file=paste("C:/Users/Owner/Dropbox/teaching/",foldername,
                              "/web/graphs/",graphname,".",format,sep="")
    else
          file=paste(foldername,graphname,".",format,sep="")
    mcat(file)
    if(format=="png") png(file)
    if(format=="pdf") pdf(file)
    if(format=="ps") postscript(file, horizontal = F, pointsize = 17)
    if(format=="eps") {
        setEPS()
        postscript(file, horizontal = F, pointsize = 17)
    }
    f()
    dev.off()
}
```

My .Rprofile
My other Routines

- mh: draws a histogram scaled as a density, if desired with density curve

- eplot: sets up a scatterplot without symbols, axes, labels, title etc.

- dpf: “dump function to html” – just like dump but with spaces recognized by html, for copying R routines to a webpage and keeping the indentation alive

My .Rprofile
My other Routines

- r_tbls: takes a matrix or dataframe and constructs text output that can be used as a table in either html or latex
- Example:

```
> Grades
      Exam1 Exam2 Grade
Anna     78    80     C
Peter    80    76     B
Paul     67    70     C
Mary     90    95     A
> r_tbls(Grades)
```

- Change to HTML editor and paste it in with CTRL-v:

```
<TABLE BORDER>
<TR> <TD> </TD> <TH> Exam1 </TH> <TH> Exam2 </TH>
<TH> Grade </TH> </TR>
<TR ALIGN=right> <TH> Anna </TH> <TD> 78 </TD> <TD>
80 </TD> <TD> C </TD> </TR>
<TR ALIGN=right> <TH> Peter </TH> <TD> 80 </TD> <TD>
76 </TD> <TD> B </TD> </TR>
<TR ALIGN=right> <TH> Paul </TH> <TD> 67 </TD> <TD>
70 </TD> <TD> C </TD> </TR>
<TR ALIGN=right> <TH> Mary </TH> <TD> 90 </TD> <TD>
95 </TD> <TD> A </TD> </TR>
</Table>
```

Or

> r_tbls(Grades,”l”)

Change to Latex editor and hit CTRL-v:

```
\begin{table}[!htbp] \centering
\begin{tabular}{cccc}
\hline
&   Exam1   &    Exam2   &   Grade \\
 Anna & 78 & 80 & C \\
 Peter & 80 & 76 & B \\
 Paul & 67 & 70 & C \\
 Mary & 90 & 95 & A \\
\hline
\end{tabular}
\end{table}
```

My .Rprofile

If you liked any of these routines, or want to make your own .Rprofile, mine is available from my webpage at

http://academic.uprm.edu/wrolke/.Rprofile

Rcpp

- R rocks!
- but it does it slowly
- sometimes simulation can take a long time
- and rewriting part of the code in C++ can speed it up dramatically

- This used to be painful

- But no more!

Example: Symmetric Random Walk

- Question: what is mean number of steps needed to reach ±100?
- Let's do a simulation:

```
> randomWalkR
function (M=1e3)
{
    tm <- proc.time()
    N=rep(0,M)
    for(i in 1:M) {
        x <- 0
        repeat {
            N[i] <- N[i]+1
            x <- x+sample(c(-1,1),1)
            if(abs(x)==100) break
        }
    }
    print(proc.time()-tm)
    mean(N)
}
> randomWalkR()
   user  system elapsed
  92.18    0.01   92.34
[1] 10143.41
```

- Rewrite inner loop as C++ function
- Notice: this is better than C++:
  - Vectorized
  - R functions can be used!
  - -Rsugar(?)

```cpp
#include <Rcpp.h>

using namespace Rcpp;

// [[Rcpp::export]]
int rwC() {

int k=0;
int z=0;
NumericVector u;

do {
    k++;
    u=runif(1);
    if(u[0]<0.5) z--;
    else z++;
}  while (abs(z)<100);
return k;
}
```

And Now:

```
> require(Rcpp)
> sourceCpp(paste(getwd(),"/randomWalk.cpp",sep=""))
> randomWalkC
function (M=1e3)
{
    tm <- proc.time()
    N=rep(0,M)
    for(i in 1:M) N[i] <- rwC()
    print(proc.time()-tm)
    mean(N)
}
> randomWalkC()
   user  system elapsed
   1.38    0.00    1.37
[1] 9885.228
```

Rshiny

- **A web application framework for R**
- **Turn your analyses into interactive web applications**
- **No CSS, or JavaScript knowledge required**
- **To create new apps you need to know some R, some HTML and a little bit of shiny**
- **To use an app you need to know nothing! (Good for our students..)**

Example: Illustration of Correlation

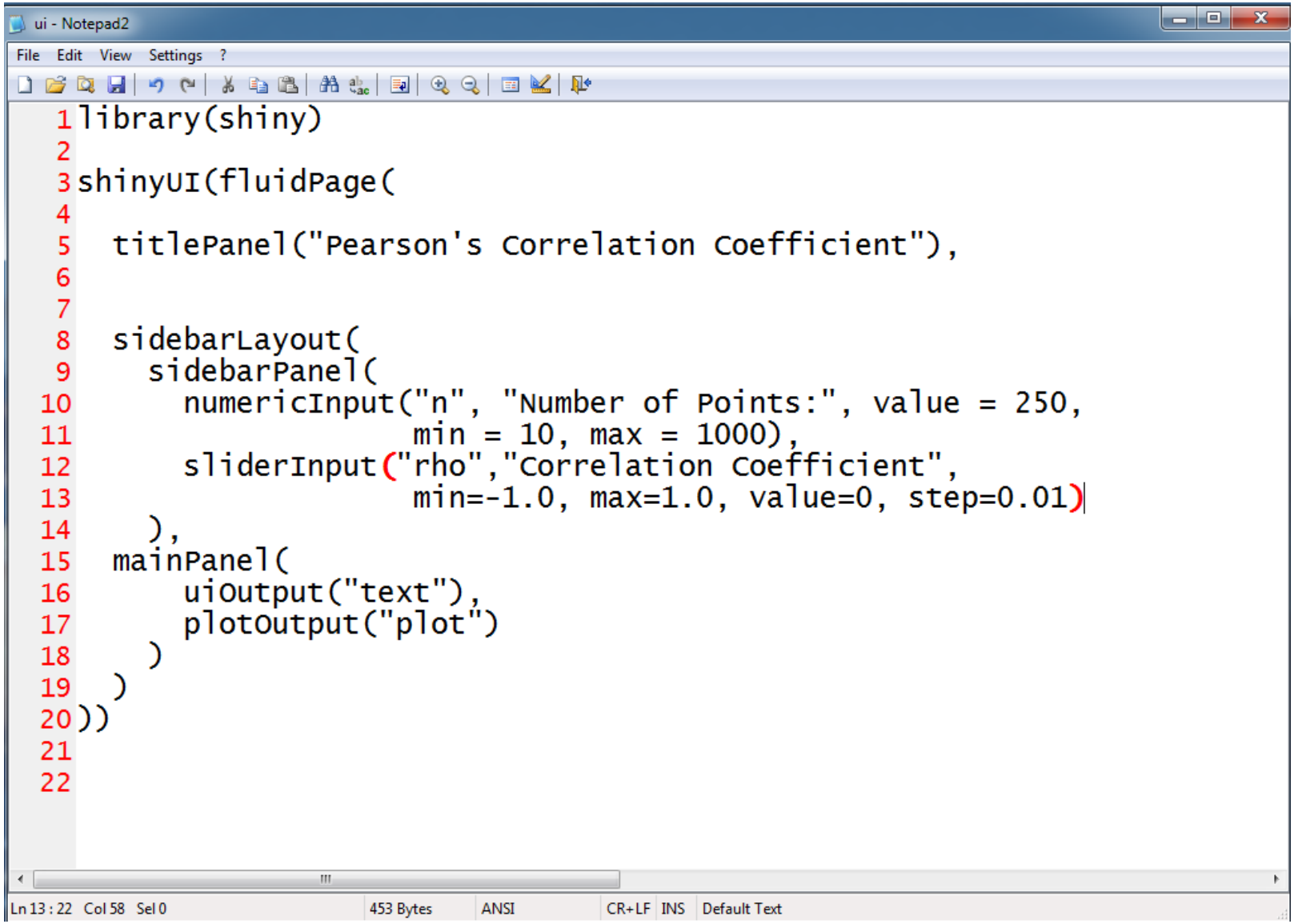
A demo meant to show students the meaning of the correlation coefficient, how that looks in a scatterplot, and illustrate the relationship/difference between the population correlation coefficient ρ and the sample correlation coefficient r.

So we want to generate n observations from a bivariate normal with correlation ρ, draw the scatterplot and calculate r.

But it would be nice if we could do this so the sample size n and the correlation ρ can be changed on the fly!

Result: https://wolfgangrolke.shinyapps.io/correlation/

All of this magic with two files: ui.R



server.R

server - Notepad2

File Edit View Settings ?

```
1  library(shiny)
2
3  shinyServer(function(input, output) {
4
5    data <- reactive({
6      a <- input$rho/sqrt(1-input$rho^2)
7      x <- rnorm(input$n)
8      y <- a*x+rnorm(input$n)
9      cbind(x,y)
10   })
11
12   output$plot <- renderPlot({
13     plot(data(),pch=20,xlab="x",ylab="y")
14   })
15
16   output$text <- renderText({
17         pc <- round(c(input$rho,cor(data())[1,2]),3)
18         l1 <- paste("<h4>Parameter Correlation Coefficient: ",HTML("&rho;"),
19                     " = ",pc[1],"<h4>")
20         l2 <- paste("<h4>Sample Correlation Coefficient r = ",
21                       pc[2],"</h4>")
22         c(l1,l2)
23        })
24 })
25
26
```

Ln 21 : 26 Col 38 Sel 0 | 645 Bytes | ANSI | CR+LF | INS | Default Text

How to run/distribute a shiny app:

- Get package Rshiny, runApp()
- Zip folder with ui.R and server.R, send with email or put on webpage, anyone with R can run it using runUrl
- Deploy app at shinyapps.io (some restrictions)

Two other examples

https://wolfgangrolke.shinyapps.io/boxplot/

https://wolfgangrolke.shinyapps.io/taylor/

- For my apps go to

http://academic.uprm.edu/wrolke/myapps.htm

Do you have any favorite R tips and tricks?

I would love to hear about them!

The End!