Programming R and RShiny® for Pensions
Basics and Concepts applied to Business

Dominique Beckers

PENSIONMATTERS.BE

Start programming today step by step

Publisher : Pensionmatters.be

ISBN: 978-94-640-5190-2

Basics and Concepts of R and RShiny®
Programming for Pensions in a Business Context

INTRO

Objectives and starter

You always wanted to start programming or to take up again your programming capabilities left behind some while ago ?
Well, here you can realise this *today* !
R Statistical programming language ® offers a genuine possibility for doing so since this free available tool combines powerful computing with an easy to learn and use framework.
R is an open source computer language embedded in a platform. Unlike maybe you used before many other languages such as COBOL, Fortran, or even Compiled Basic, R simply interprets code line by line and execute these commands in order to get powerful results from data ordering, filtering, calculating and obtain graphical representation and even in a very smooth and evolved way but remaining rather easy, in sharing through a dynamic web application.

Moreover, R is growing in popularity because many statisticians and actuaries make their written code accessible though third party libraries called packages, which allows access to a wide variety of state-of-the-art procedures, ready from the shelf. The programming environment is available out there and ready for use.

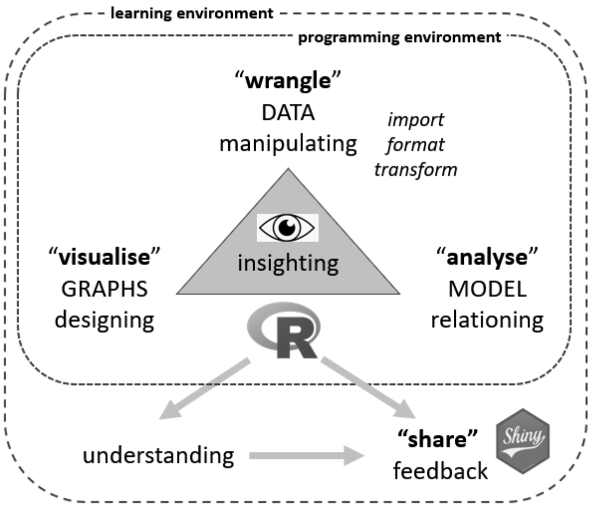


Figure 1. positioning (R) RStudio®, RShiny ® as programming environment

This book helps you to get started with practical calculations in the Pension Business using R. R Programming code is explained step by step allowing to use the (mathematical) basics of pension calculation going from (legal) life tables and interest rate structures, over the price of a unit pension, towards the capital value of a current pension at any time.

The latter two concepts, the price today of a future life contingent capital and the value of a series of future payments, are the two strong corner stones for pension calculations, which will be built up with R programming code. Although the mathematical formulas on themselves are of minor importance next to the way the programming results are obtained and despite the fact they will be shown explicitly while explaining the coding steps, a basic understanding of these numerical statements is somewhere necessary. If you want some additional background information on these concepts, this book aligns smoothly with "Actuarial Mathematics for Pensions, Basics and Concepts applied to Business", from the same author (ISBN978-90-465-9740-8 - Ed. Wolters Kluwer).

After being able to generate the basic corner stones for pension computing, a funding method is introduced as further building block for Pension calculation. The results of a classic business funding method for a given pension are calculated by programming the actuarial equivalence principle and the extension towards the practical yearly calculation mechanism of the pension scheme. This item will include the computing of mathematical reserves.

The next further step is using these basic concepts including the calculation of life expectancy in order to build a more advanced application for retirement planning, including projecting asset classes and inflation adjusted withdrawals. At this level, some advanced calculations and programming techniques such as Monte Carlo simulation in R, are used.

Finally this same application of retirement planning in the programming environment R will be made available for sharing in the web through the package RShiny. This will be shown step by step for easy reproducibility in your other, new R applications as that may be the desired outcome for reading and working on the content of this book.

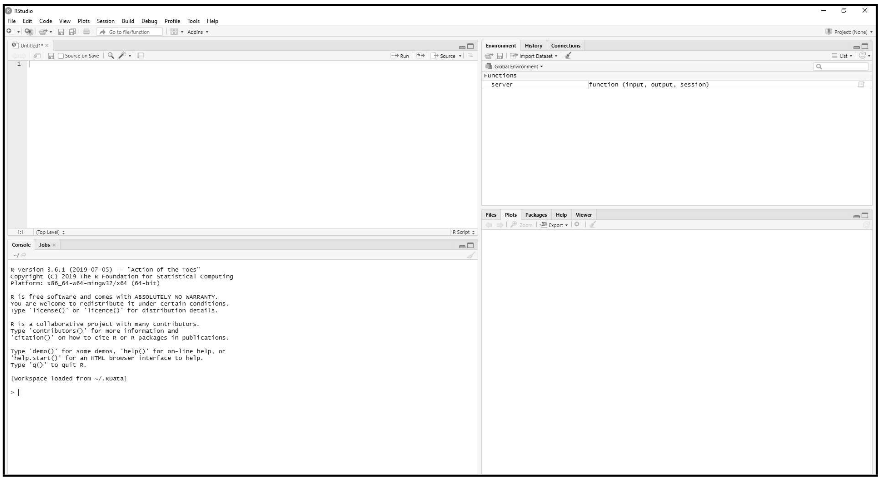
Getting started now

Obtaining R

The first thing to do is setting up R on your computer. See the Appendix for some useful and helping comments on how to set up the R framework.
R is a free software which runs on a wide variety of platforms, Windows, UNIX and MacOS. To download R, reference is made to CRAN, "The Comprehensive R Archive Network", which is made available through one of many mirrors.
Installing R for Windows (for a first use) lead to the version R 3.6.1 for Windows at the time of publication of this book. This version can run on Windows versions (from XP onwards) including 64-bit versions of Windows.
Although basically R can be launched from this installation (and which remains necessary), it is more convenient to use an integrated development environment IDE which makes the programming work a lot more easy and comprehensive.

RStudio®

Such an IDE is RStudio®. See the Appendix for some useful and helping comments on how to set up the RStudio® IDE.
On launching RStudio®, the following working environment opens as shown in figure 2 :



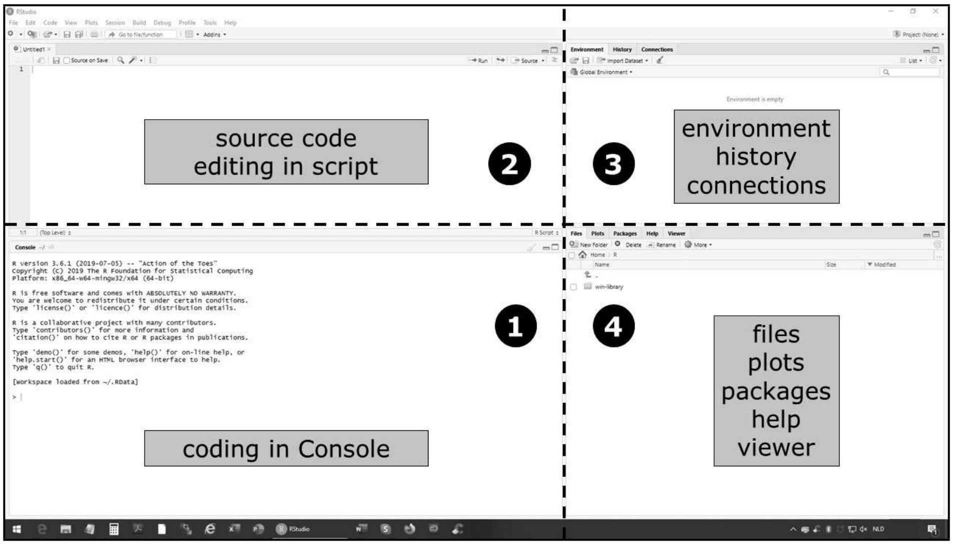
```
Functions
  server                function (input, output, session)
```

```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-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]

> |
```

Figure 2. RStudio® IDE integrated development environment

On the top of the screen appears a classic menu structure, File, Edit, Code, View, Plots… . Take the time to scroll over each of the submenus just to capture the idea of actions, but that is sufficient at this stage. Necessary steps will be explained further in the book.
Do not worry neither when the screen seems to be missing an area with “Untitled1” as shown in figure 2. Navigating to “File” in the menu structure, and choosing “New File” and further “R Script” will open this area.

This integrated environment subdivides the screen window now into 4 parts or “panes”, each dedicated to a specific display or interaction : *bottom left* (1) the console area, *top left* (2) the source code area which allows editing in script mode, *top right* (3) the area for keeping track of the environment in which the work is going on as well as the history and connections, and finally *bottom right* (4) the combined area for files management, plots, packages, help and viewer, as shown in figure 3.



```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-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]

>
```

Figure 3. RStudio® IDE integrated development environment – view on PANES

Each of these panes can be sized by dragging the division lines, or be re-arranged with the buttons in the top right corner of the pane, but mostly this standard setting will do the job.

R Console and some first instructions

First thing to notice is the Console (pane *bottom left*) which is the area in which instructions or commands in R language can be coded for execution. As written earlier, R is functionally an interpreter, so lines will be interpreted and executed line by line.

Any language requires to respect a certain syntax, i.e. a valid arrangement or order of elements to execute, a certain grammar, i.e. the structure of these elements and finally the vocabulary, i.e. the inventory of elements which are recognised.

The purpose of this book is not to give an exhaustive overview for each element in each of these three requirements, but merely to start using them in comprehensive, specific applications related to pensions, in order to gain the ability for continuing to program (simple) applications. In addition, the ambition of gaining an exhaustive R-language comprehension is hardly achievable since the R Community is producing constantly new evolutions in any directions only bounded by creativity and imagination.

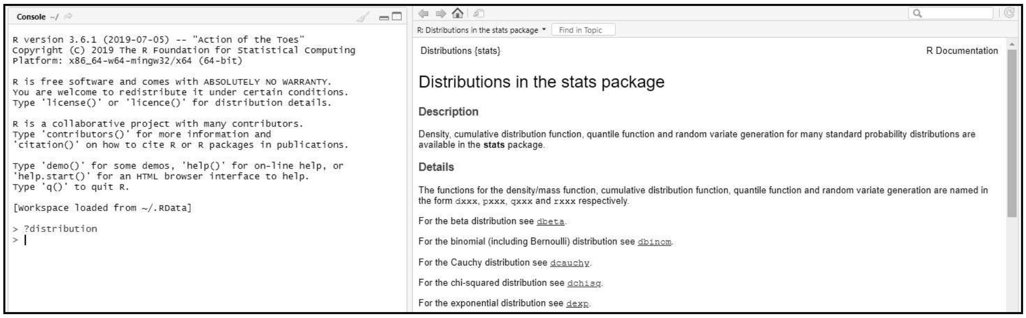
But we have to start somewhere!

One of the beautiful dimensions of R and R Studio® is the availability of the "Help" function.   Suppose you want to know at any stage some more information on how to use (statistical) distributions in R ?

Type in the Console at the prompt ">" help(distribution) or ?distribution :

```
>?distribution
```

R will show the available information or a list of possible candidates for further exploration in the View Pane (area bottom right) as shown in figure 4:



```
R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-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]

> ?distribution
> |
```

Distributions {stats} R Documentation

Distributions in the stats package

Description

Density, cumulative distribution function, quantile function and random variate generation for many standard probability distributions are available in the **stats** package.

Details

The functions for the density/mass function, cumulative distribution function, quantile function and random variate generation are named in the form `dxxx`, `pxxx`, `qxxx` and `rxxx` respectively.

For the beta distribution see `dbeta`.

For the binomial (including Bernoulli) distribution see `dbinom`.

For the Cauchy distribution see `dcauchy`.

For the chi-squared distribution see `dchisq`.

For the exponential distribution see `dexp`.

Figure 4. RStudio® IDE integrated development environment – view on HELP

Remark that the Pane bottom right has switched from the TAB “File” to the TAB “Help”. The same search could be performed starting from the “search function” at the top right of this pane. This function searches all of the available R documentation in R Studio®/R. Alternatively, Help can also be obtained from function key F1.

Suppose you want to know how to generate random numbers from a uniform distribution. R has many built-in functions and yes, also for generating random numbers.

Type in the console in a next line at the prompt “>” ?uniform

```
>?uniform
```

The View Pane right under will explain that these functions provide information about the uniform distribution on the interval from “min” to “max”, “unif” is the kernel of the functions, preceded by “d” for the density (dunif), “p” for the distribution function (punif), “q” for the quantile function (qunif) and preceded by “r” generates uniform random numbers (runif).

In the same way, these functions can be found for the normal distribution or most of the other types of distributions which may be needed. Each of these functions can be plotted, as is explained further in the book, but for now let us see the random number generation by: runif(n, min = 0, max = 1).

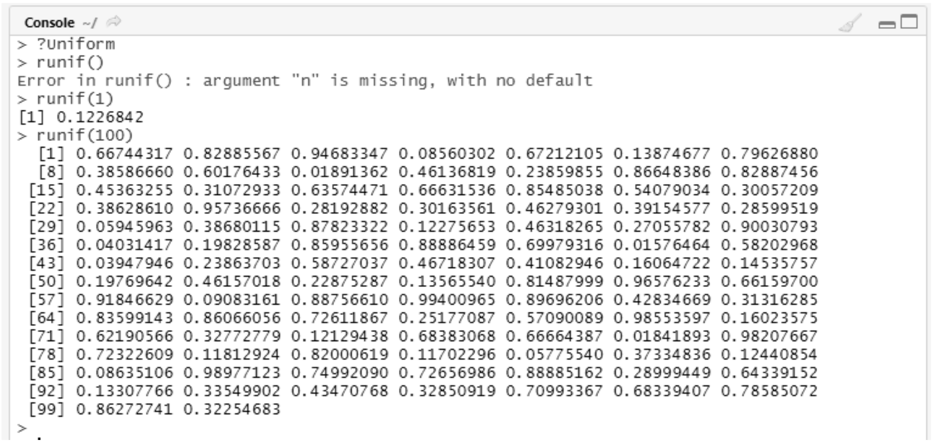
Remark also that further information on the arguments between the brackets is given : “n” is the number of observations, whereas “min” and “max” are the lower and upper limits of the generation which both must be finite.
In Details is explained that when “min” or “max” are not specified they assume the default values of 0 and 1 respectively.
Typing in Console on the next line prompt :

```
>runif()
```

will lead to an error in runif() : argument "n" is missing, with no default, explaining that (each) function requires a certain set of compulsory arguments (such as “n” in this case) next to a series of optional arguments (such as “min” and “max” in this case).
Typing in Console on the next line prompt :

```
>runif(1)
```

will lead to a first random number drawn from a uniform distribution between 0 and 1
The result of altering the number of random numbers to be drawn to 100 is shown in figure 5:



```
> ?Uniform
> runif()
Error in runif() : argument "n" is missing, with no default
> runif(1)
[1] 0.1226842
> runif(100)
  [1] 0.66744317 0.82885567 0.94683347 0.08560302 0.67212105 0.13874677 0.79626880
  [8] 0.38586660 0.60176433 0.01891362 0.46136819 0.23859855 0.86648386 0.82887456
 [15] 0.45363255 0.31072933 0.63574471 0.66631536 0.85485038 0.54079034 0.30057209
 [22] 0.38628610 0.95736666 0.28192882 0.30163561 0.46279301 0.39154577 0.28599519
 [29] 0.05945963 0.38680115 0.87823322 0.12275653 0.46318265 0.27055782 0.90030793
 [36] 0.04031417 0.19828587 0.85955656 0.88886459 0.69979316 0.01576464 0.58202968
 [43] 0.03947946 0.23863703 0.58727037 0.46718307 0.41082946 0.16064722 0.14535757
 [50] 0.19769642 0.46157018 0.22875287 0.13565540 0.81487999 0.96576233 0.66159700
 [57] 0.91846629 0.09083161 0.88756610 0.99400965 0.89696206 0.42834669 0.31316285
 [64] 0.83599143 0.86066056 0.72611867 0.25177087 0.57090089 0.98553597 0.16023575
 [71] 0.62190566 0.32772779 0.12129438 0.68383068 0.66664387 0.01841893 0.98207667
 [78] 0.72322609 0.11812924 0.82000619 0.11702296 0.05775540 0.37334836 0.12440854
 [85] 0.08635106 0.98977123 0.74992090 0.72656986 0.88885162 0.28999449 0.64339152
 [92] 0.13307766 0.33549902 0.43470768 0.32850919 0.70993367 0.68339407 0.78585072
 [99] 0.86272741 0.32254683
>
```

Figure 5. result of runif(100) shown in the Console

Remark also that each time an instruction in the Console is executed, this instruction is “lost” for further execution, and needs to be re-entered for executing again. The arrow-keys “up” (previous) and “down” (next) can be used to toggle in the last instructions given in the Console, whereas CTRL+arrow key up provokes a popup menu with all the last instructions.

CTRL+c will end a command which is too long running in the Console.

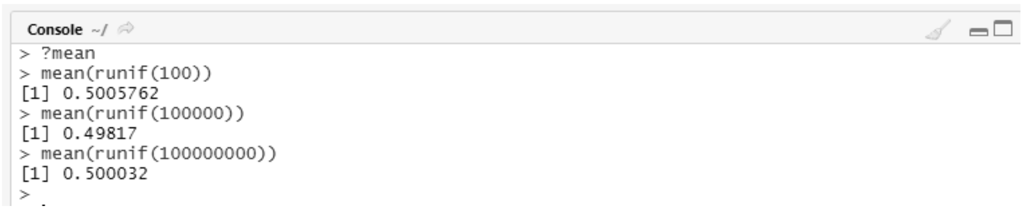
CTRL+l will clear the screen in the Console.

Remark also the number in squared brackets before the enumeration of random numbers created. This number in brackets at the beginning of each row gives the place in the enumeration of the next following number. So in the figure above, the random number “0.86066056” is the 65th number from 100.

All relevant and more statistical functions are present and can be used in the instructions, e.g. the function “mean” calculates the arithmetic mean value of a series of values (for an explanation, use the ? or help function in the Console),

```
>mean(runif(100)
```

And increasing the value of the number of random numbers generated, leads to a reduction of the standard deviation :



```
> ?mean
> mean(runif(100))
[1] 0.5005762
> mean(runif(100000))
[1] 0.49817
> mean(runif(100000000))
[1] 0.500032
>
```

Figure 6. result of runif(x) with increasing x as shown in the Console

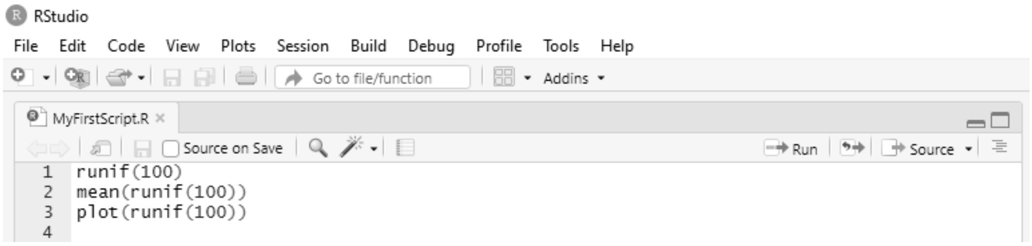
R script Files

Re-entering instructions in the Console each time these instructions need to be executed, is not an efficient way for programming. Lines of code need to be reproducible and editable in an easy way without encoding again. This is realised in the script file in the top left Pane (2) (see figure 3). A script file is a plain text document containing lines of code.

Type following multiple lines of code in the script-file “Untitled1*” :

```
runif(100)
mean(runif(100))
plot(runif(100))
```

Navigate in the menu right on top of the screen to “File” and “Save as” the file in one of the available directories as “MyFirstScript”,or later on once your file needs to be resaved, on the highlighted disc-icon underneath the name.



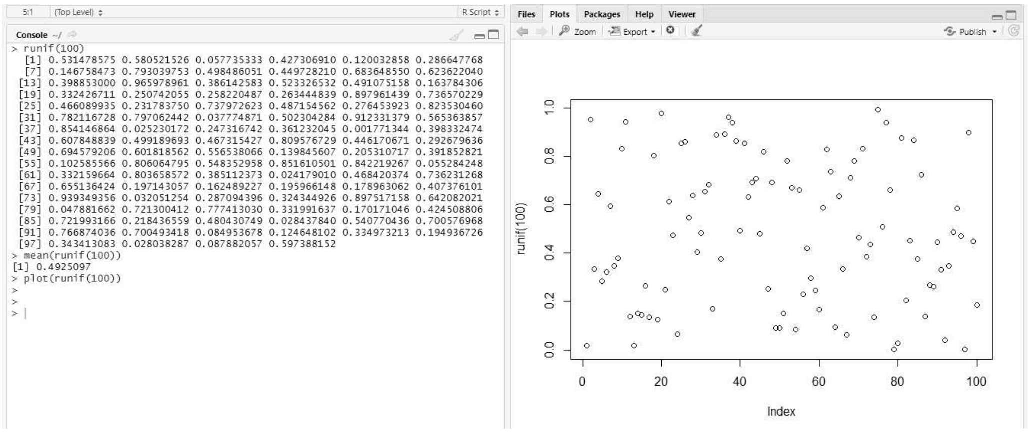
```
runif(100)
mean(runif(100))
plot(runif(100))
```

Figure 7. coding lines as shown in the Script Pane top left

The script will be saved with an extension .R

Running this script is realised by selecting the lines in the Script Pane by “shift”+arrow up or “shift”+arrow down and clicking the button “run” to the right underneath the name of the file (with the green arrow pointing to the right). Alternatively, instead of clicking the button, CTRL+<enter> runs the script as well.

The script is executed, the executed script is represented in the console (in blue) and a graphical output is presented in the Plot area in the Pane bottom right, as shown in figure 8.



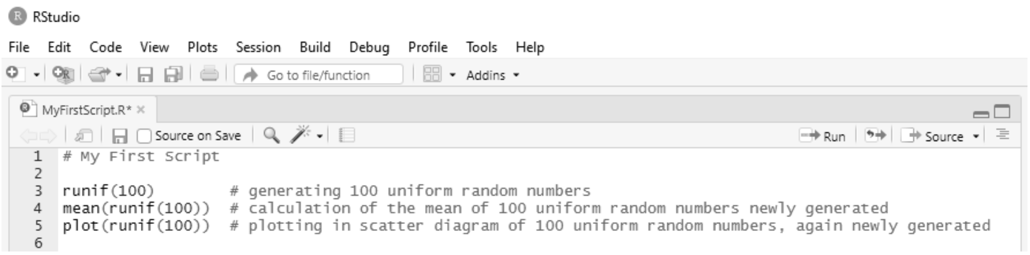
```
> runif(100)
  [1] 0.531478575 0.580521526 0.057735333 0.427306910 0.120032858 0.286647768
  [7] 0.146758473 0.793039753 0.498486051 0.449728210 0.683648550 0.623622040
 [13] 0.398853000 0.965978961 0.386142583 0.523326532 0.491075158 0.163784306
 [19] 0.332426711 0.250742055 0.258220487 0.263444839 0.897961439 0.736570229
 [25] 0.466089935 0.231783750 0.737972623 0.487154562 0.276453923 0.823530460
 [31] 0.782116728 0.797062442 0.037774871 0.502304284 0.912331379 0.565363857
 [37] 0.854146864 0.025230172 0.247316742 0.361232045 0.001771344 0.398332474
 [43] 0.607848839 0.499189693 0.467315427 0.809576729 0.446170671 0.292679636
 [49] 0.694579206 0.601818562 0.556538066 0.139845607 0.205310717 0.391852821
 [55] 0.102585566 0.806064795 0.548352958 0.851610501 0.842219267 0.055284248
 [61] 0.332159664 0.803658572 0.385112373 0.024179010 0.468420374 0.736231268
 [67] 0.655136424 0.197143057 0.162489227 0.195966148 0.178963062 0.407376101
 [73] 0.939349356 0.032051254 0.287094396 0.324344926 0.897517158 0.642082021
 [79] 0.047881662 0.721300412 0.777413030 0.331991637 0.170171046 0.424508806
 [85] 0.721993166 0.218436559 0.480430749 0.028437840 0.540770436 0.700576968
 [91] 0.766874036 0.700493418 0.084953678 0.124648102 0.334973213 0.194936726
 [97] 0.343413083 0.028038287 0.087882057 0.597388152
> mean(runif(100))
[1] 0.4925097
> plot(runif(100))
>
>
> |
```

Figure 8. result of the script as shown in the Console and Plot area

Later on in this book, plots will be discussed in more detail, titles, axis and a legend added together with the use of many more options.

Any errors in the script which may occur, will also be shown in the Console (in red) which facilitates de-bugging. Code in the script can be corrected, saved and executed over again.

It is good practice to provide some information in the form of comments to your code, which may be done by the #-sign : everything in the code line behind the #-sign (in green) will be ignored when executing the code, as shown in Figure 9



```
# My First Script

runif(100)        # generating 100 uniform random numbers
mean(runif(100))  # calculation of the mean of 100 uniform random numbers newly generated
plot(runif(100))  # plotting in scatter diagram of 100 uniform random numbers, again newly generated

```

Figure 9. illustration of commenting the script in the Script Pane

Many standards and style prescriptions for programming exists. When working on simple standalone own projects, standards and styling is often not that important, but may become relevant when the project becomes more important (or increases in the number of coded lines) or when more programmers get involved or have to take over the coding.
See the Appendix for some useful and helping comments for a good start on programming.

Files, Plots, Packages, Help and Viewer

In the View Pane (3) right under (see figure 3) figures also, next to the TABS "Plots" and "Help", also the TABS "File", "Packages" and "Viewer".

The TAB "File", as shown in Figure 10, allows standard file and directory organisation according to the desired specifications.
An interesting feature is selecting a directory and setting this directory to the working directory for your project. This is done in the menu structure just underneath the Pane menu, by selecting "More" and in the drop down list "Set As Working Directory". R Studio® will now work with this directory for file-handling.

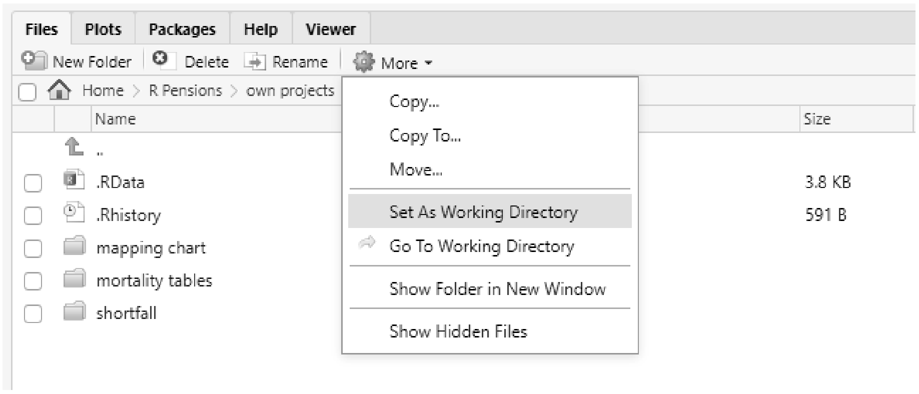


Figure 10. The View Pane in TAB Files and selecting the working directory

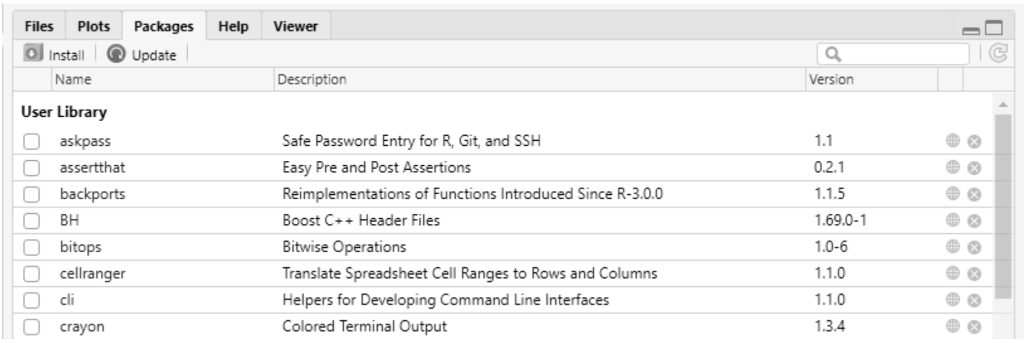
Alternatively, setting the working directory can also be done in the Console with following instruction, where R Pensions is the name of the directory :

```
>setwd("~/R Pensions")
```

When needed to know which directory is the current working directory, the following instruction can be used in the Console and in response the name of the directory will be shown.

```
>getwd()
```

Next in the menu, the TAB “Packages” can be found as shown in Figure 11, and which opens a complete, fantastic world of re-usable code for the programmer.



	Name	Description	Version
	User Library		
☐	askpass	Safe Password Entry for R, Git, and SSH	1.1
☐	assertthat	Easy Pre and Post Assertions	0.2.1
☐	backports	Reimplementations of Functions Introduced Since R-3.0.0	1.1.5
☐	BH	Boost C++ Header Files	1.69.0-1
☐	bitops	Bitwise Operations	1.0-6
☐	cellranger	Translate Spreadsheet Cell Ranges to Rows and Columns	1.1.0
☐	cli	Helpers for Developing Command Line Interfaces	1.1.0
☐	crayon	Colored Terminal Output	1.3.4

Figure 11. The View Pane in TAB Packages opens a new world of re-usable code

The beauty of using the open source R, is that many actuaries and data scientists share their code for use by others through “packages”. A package is an assembly of code, data and documentation for earlier solved problems or applications. R and R Studio® allow for loading those packages into the workspace of the current project.

An overwhelming variety of packages exists already and the probability that your current project is not tackled somewhere else in some way or another, is becoming smaller when time goes on.

Packages can be loaded in several ways. Most common place for getting packages is CRAN (The Comprehensive R Archive Network, see the appendix of setting up R from CRAN for more information) or GitHub. In order to install packages from CRAN, an instruction in the code (Console or Script) is sufficient :