|  |
| --- |
| R version 3.2.5 (2016-04-14) -- "Very, Very Secure Dishes"  Copyright (C) 2016 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.  > if(!require("swirl"))install.packages("swirl")  Loading required package: swirl  Warning in install.packages :  'lib = "C:/Program Files/R/R-3.2.5/library"' is not writable  also installing the dependencies ‘stringi’, ‘magrittr’, ‘crayon’, ‘praise’, ‘R6’, ‘jsonlite’, ‘mime’, ‘curl’, ‘openssl’, ‘bitops’, ‘stringr’, ‘testthat’, ‘httr’, ‘yaml’, ‘RCurl’, ‘digest’  There are binary versions available but the source versions are later:  binary source needs\_compilation  R6 2.2.0 2.2.1 FALSE  jsonlite 1.4 1.5 TRUE  curl 2.5 2.6 TRUE  Binaries will be installed  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/stringi\_1.1.5.zip'  Content type 'application/zip' length 14220135 bytes (13.6 MB)  downloaded 13.6 MB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/magrittr\_1.5.zip'  Content type 'application/zip' length 149888 bytes (146 KB)  downloaded 146 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/crayon\_1.3.2.zip'  Content type 'application/zip' length 703837 bytes (687 KB)  downloaded 687 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/praise\_1.0.0.zip'  Content type 'application/zip' length 17696 bytes (17 KB)  downloaded 17 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/jsonlite\_1.4.zip'  Content type 'application/zip' length 1120239 bytes (1.1 MB)  downloaded 1.1 MB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/mime\_0.5.zip'  Content type 'application/zip' length 37449 bytes (36 KB)  downloaded 36 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/curl\_2.5.zip'  Content type 'application/zip' length 3090095 bytes (2.9 MB)  downloaded 2.9 MB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/openssl\_0.9.6.zip'  Content type 'application/zip' length 3558342 bytes (3.4 MB)  downloaded 3.4 MB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/bitops\_1.0-6.zip'  Content type 'application/zip' length 36012 bytes (35 KB)  downloaded 35 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/stringr\_1.2.0.zip'  Content type 'application/zip' length 148796 bytes (145 KB)  downloaded 145 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/testthat\_1.0.2.zip'  Content type 'application/zip' length 1019329 bytes (995 KB)  downloaded 995 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/httr\_1.2.1.zip'  Content type 'application/zip' length 281509 bytes (274 KB)  downloaded 274 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/yaml\_2.1.14.zip'  Content type 'application/zip' length 180051 bytes (175 KB)  downloaded 175 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/RCurl\_1.95-4.8.zip'  Content type 'application/zip' length 2860476 bytes (2.7 MB)  downloaded 2.7 MB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/digest\_0.6.12.zip'  Content type 'application/zip' length 172453 bytes (168 KB)  downloaded 168 KB  trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.2/swirl\_2.4.3.zip'  Content type 'application/zip' length 236175 bytes (230 KB)  downloaded 230 KB  package ‘stringi’ successfully unpacked and MD5 sums checked  package ‘magrittr’ successfully unpacked and MD5 sums checked  package ‘crayon’ successfully unpacked and MD5 sums checked  package ‘praise’ successfully unpacked and MD5 sums checked  package ‘jsonlite’ successfully unpacked and MD5 sums checked  package ‘mime’ successfully unpacked and MD5 sums checked  package ‘curl’ successfully unpacked and MD5 sums checked  package ‘openssl’ successfully unpacked and MD5 sums checked  package ‘bitops’ successfully unpacked and MD5 sums checked  package ‘stringr’ successfully unpacked and MD5 sums checked  package ‘testthat’ successfully unpacked and MD5 sums checked  package ‘httr’ successfully unpacked and MD5 sums checked  package ‘yaml’ successfully unpacked and MD5 sums checked  package ‘RCurl’ successfully unpacked and MD5 sums checked  package ‘digest’ successfully unpacked and MD5 sums checked  package ‘swirl’ successfully unpacked and MD5 sums checked  The downloaded binary packages are in  C:\Users\10276658\AppData\Local\Temp\RtmpW2juST\downloaded\_packages  installing the source package ‘R6’  trying URL 'https://cran.rstudio.com/src/contrib/R6\_2.2.1.tar.gz'  Content type 'application/x-gzip' length 325641 bytes (318 KB)  downloaded 318 KB  \* installing \*source\* package 'R6' ...  \*\* package 'R6' successfully unpacked and MD5 sums checked  \*\* R  \*\* inst  \*\* preparing package for lazy loading  \*\* help  \*\*\* installing help indices  \*\* building package indices  \*\* installing vignettes  \*\* testing if installed package can be loaded  \*\*\* arch - i386  \*\*\* arch - x64  \* DONE (R6)  The downloaded source packages are in  ‘C:\Users\10276658\AppData\Local\Temp\RtmpW2juST\downloaded\_packages’  Warning message:  In library(package, lib.loc = lib.loc, character.only = TRUE, logical.return = TRUE, :  there is no package called ‘swirl’  > library(swirl)  | Hi! Type swirl() when you are ready to begin.  > install\_course\_github('darrenredmond','R\_ProgrammingDR')  |==================================================================================================================| 100%  > swirl()  | Welcome to swirl! Please sign in. If you've been here before, use the same name as you did then. If you are new, call  | yourself something unique.  What shall I call you? emtee  | Thanks, emtee. Let's cover a couple of quick housekeeping items before we begin our first lesson. First of all, you  | should know that when you see '...', that means you should press Enter when you are done reading and ready to continue.  ... <-- That's your cue to press Enter to continue  | Also, when you see 'ANSWER:', the R prompt (>), or when you are asked to select from a list, that means it's your turn to  | enter a response, then press Enter to continue.  Select 1, 2, or 3 and press Enter  1: Continue.  2: Proceed.  3: Let's get going!  Selection: 1  | You can exit swirl and return to the R prompt (>) at any time by pressing the Esc key. If you are already at the prompt,  | type bye() to exit and save your progress. When you exit properly, you'll see a short message letting you know you've  | done so.  | When you are at the R prompt (>):  | -- Typing skip() allows you to skip the current question.  | -- Typing play() lets you experiment with R on your own; swirl will ignore what you do...  | -- UNTIL you type nxt() which will regain swirl's attention.  | -- Typing bye() causes swirl to exit. Your progress will be saved.  | -- Typing main() returns you to swirl's main menu.  | -- Typing info() displays these options again.  | Let's get started!  ...  | Please choose a course, or type 0 to exit swirl.  1: R ProgrammingDR  2: Take me to the swirl course repository!  Selection: 1  | Please choose a lesson, or type 0 to return to course menu.  1: Simple 2: Basic Building Blocks 3: Workspace and Files 4: Sequences of Numbers  5: Vectors 6: Missing Values 7: Subsetting Vectors 8: Matrices and Data Frames  9: Logic 10: Functions 11: lapply and sapply 12: vapply and tapply  13: Looking at Data 14: Simulation 15: Dates and Times 16: Base Graphics  Selection: 1  | | 0%  | In this lesson we will simply ensure that you can connect to the DBS Data Projects results services and validate that you  | can submit. Please answer Yes to the next question.  ...  |==================================================================================================================| 100%  | Would you like to receive credit for completing this course on dbsdataprojects.com?  1: No  2: Yes  Selection: yes  Enter an item from the menu, or 0 to exit  Selection: 1  Loading required package: digest  | That's correct!  | You've reached the end of this lesson! Returning to the main menu...  | Please choose a course, or type 0 to exit swirl.  1: R ProgrammingDR  2: Take me to the swirl course repository!  Selection: 1  | Please choose a lesson, or type 0 to return to course menu.  1: Simple 2: Basic Building Blocks 3: Workspace and Files 4: Sequences of Numbers  5: Vectors 6: Missing Values 7: Subsetting Vectors 8: Matrices and Data Frames  9: Logic 10: Functions 11: lapply and sapply 12: vapply and tapply  13: Looking at Data 14: Simulation 15: Dates and Times 16: Base Graphics  Selection: 1  | | 0%  | In this lesson we will simply ensure that you can connect to the DBS Data Projects results services and validate that you  | can submit. Please answer Yes to the next question.  ...  |==================================================================================================================| 100%  | Would you like to receive credit for completing this course on dbsdataprojects.com?  1: Yes  2: No  Selection: 1  What is your email address? 10276658@mydbs.ie  What is your student number? 10276658  Grade submission failed.  Press ESC if you want to exit this lesson and you  want to try to submit your grade at a later time.  | Not quite, but you're learning! Try again.  |  1: Yes  2: No  Selection: 2  | That's the answer I was looking for.  | You've reached the end of this lesson! Returning to the main menu...  | Please choose a course, or type 0 to exit swirl.  1: R ProgrammingDR  2: Take me to the swirl course repository!  Selection: 1  | Please choose a lesson, or type 0 to return to course menu.  1: Simple 2: Basic Building Blocks 3: Workspace and Files 4: Sequences of Numbers  5: Vectors 6: Missing Values 7: Subsetting Vectors 8: Matrices and Data Frames  9: Logic 10: Functions 11: lapply and sapply 12: vapply and tapply  13: Looking at Data 14: Simulation 15: Dates and Times 16: Base Graphics  Selection: 2  | | 0%  | In this lesson, we will explore some basic building blocks of the R programming language.  ...  |=== | 3%  | If at any point you'd like more information on a particular topic related to R, you can type help.start() at the prompt,  | which will open a menu of resources (either within RStudio or your default web browser, depending on your setup).  | Alternatively, a simple web search often yields the answer you're looking for.  ...  |====== | 5%  | In its simplest form, R can be used as an interactive calculator. Type 5 + 7 and press Enter.  >  > 5+7  [1] 12  | You are amazing!  |========= | 8%  | R simply prints the result of 12 by default. However, R is a programming language and often the reason we use a  | programming language as opposed to a calculator is to automate some process or avoid unnecessary repetition.  ...  |============ | 11%  | In this case, we may want to use our result from above in a second calculation. Instead of retyping 5 + 7 every time we  | need it, we can just create a new variable that stores the result.  ...  |=============== | 13%  | The way you assign a value to a variable in R is by using the assignment operator, which is just a 'less than' symbol  | followed by a 'minus' sign. It looks like this: <-  ...  |================== | 16%  | Think of the assignment operator as an arrow. You are assigning the value on the right side of the arrow to the variable  | name on the left side of the arrow.  ...  |===================== | 18%  | To assign the result of 5 + 7 to a new variable called x, you type x <- 5 + 7. This can be read as 'x gets 5 plus 7'.  | Give it a try now.  > x <- 5+7  | Nice work!  |======================== | 21%  | You'll notice that R did not print the result of 12 this time. When you use the assignment operator, R assumes that you  | don't want to see the result immediately, but rather that you intend to use the result for something else later on.  ...  |=========================== | 24%  | To view the contents of the variable x, just type x and press Enter. Try it now.  > x  [1] 12  | Keep working like that and you'll get there!  |============================== | 26%  | Now, store the result of x - 3 in a new variable called y.  > y<-x-3  | Perseverance, that's the answer.  |================================= | 29%  | What is the value of y? Type y to find out.  > y  [1] 9  | You are really on a roll!  |==================================== | 32%  | Now, let's create a small collection of numbers called a vector. Any object that contains data is called a data structure  | and numeric vectors are the simplest type of data structure in R. In fact, even a single number is considered a vector of  | length one.  ...  |======================================= | 34%  | The easiest way to create a vector is with the c() function, which stands for 'concatenate' or 'combine'. To create a  | vector containing the numbers 1.1, 9, and 3.14, type c(1.1, 9, 3.14). Try it now and store the result in a variable  | called z.  > z<-c(1.1,9,3.14)  | Your dedication is inspiring!  |========================================== | 37%  | Anytime you have questions about a particular function, you can access R's built-in help files via the `?` command. For  | example, if you want more information on the c() function, type ?c without the parentheses that normally follow a  | function name. Give it a try.  > ?c  | You're the best!  |============================================= | 39%  | Type z to view its contents. Notice that there are no commas separating the values in the output.  > z  [1] 1.10 9.00 3.14  | Keep up the great work!  |================================================ | 42%  | You can combine vectors to make a new vector. Create a new vector that contains z, 555, then z again in that order. Don't  | assign this vector to a new variable, so that we can just see the result immediately.  > c(z,555,z)  [1] 1.10 9.00 3.14 555.00 1.10 9.00 3.14  | Great job!  |=================================================== | 45%  | Numeric vectors can be used in arithmetic expressions. Type the following to see what happens: z \* 2 + 100.  > z\*2+100  [1] 102.20 118.00 106.28  | You got it right!  |====================================================== | 47%  | First, R multiplied each of the three elements in z by 2. Then it added 100 to each element to get the result you see  | above.  ...  |========================================================= | 50%  | Other common arithmetic operators are `+`, `-`, `/`, and `^` (where x^2 means 'x squared'). To take the square root, use  | the sqrt() function and to take the absolute value, use the abs() function.  ...  |============================================================ | 53%  | Take the square root of z - 1 and assign it to a new variable called my\_sqrt.  > my\_sqrt<-sqrt(z-1)  | Perseverance, that's the answer.  |=============================================================== | 55%  | Before we view the contents of the my\_sqrt variable, what do you think it contains?  1: a vector of length 0 (i.e. an empty vector)  2: a vector of length 3  3: a single number (i.e a vector of length 1)  Selection: 2  | Perseverance, that's the answer.  |================================================================== | 58%  | Print the contents of my\_sqrt.  > my\_sqrt  [1] 0.3162278 2.8284271 1.4628739  | Keep up the great work!  |===================================================================== | 61%  | As you may have guessed, R first subtracted 1 from each element of z, then took the square root of each element. This  | leaves you with a vector of the same length as the original vector z.  ...  |======================================================================== | 63%  | Now, create a new variable called my\_div that gets the value of z divided by my\_sqrt.  > my\_div<-(z/my\_sqrt)  | That's not the answer I was looking for, but try again. Or, type info() for more options.  | Enter my\_div <- z / my\_sqrt. The spaces on either side of the `/` sign are not required, but can often improve  | readability by making code appear less cluttered. In the end, it's personal preference.  > my\_div <- z/my\_sqrt  | You are doing so well!  |=========================================================================== | 66%  | Which statement do you think is true?  1: my\_div is undefined  2: my\_div is a single number (i.e a vector of length 1)  3: The first element of my\_div is equal to the first element of z divided by the first element of my\_sqrt, and so on...  Selection: 3  | That's correct!  |============================================================================== | 68%  | Go ahead and print the contents of my\_div.  > my\_div  [1] 3.478505 3.181981 2.146460  | You are amazing!  |================================================================================= | 71%  | When given two vectors of the same length, R simply performs the specified arithmetic operation (`+`, `-`, `\*`, etc.)  | element-by-element. If the vectors are of different lengths, R 'recycles' the shorter vector until it is the same length  | as the longer vector.  ...  |==================================================================================== | 74%  | When we did z \* 2 + 100 in our earlier example, z was a vector of length 3, but technically 2 and 100 are each vectors of  | length 1.  ...  |======================================================================================= | 76%  | Behind the scenes, R is 'recycling' the 2 to make a vector of 2s and the 100 to make a vector of 100s. In other words,  | when you ask R to compute z \* 2 + 100, what it really computes is this: z \* c(2, 2, 2) + c(100, 100, 100).  ...  |========================================================================================== | 79%  | To see another example of how this vector 'recycling' works, try adding c(1, 2, 3, 4) and c(0, 10). Don't worry about  | saving the result in a new variable.  > c(1,2,3,4) + c(0,10)  [1] 1 12 3 14  | That's a job well done!  |============================================================================================= | 82%  | If the length of the shorter vector does not divide evenly into the length of the longer vector, R will still apply the  | 'recycling' method, but will throw a warning to let you know something fishy might be going on.  ...  |================================================================================================ | 84%  | Try c(1, 2, 3, 4) + c(0, 10, 100) for an example.  > c(1,2,3,4) + c(0,10,100)  [1] 1 12 103 4  Warning message:  In c(1, 2, 3, 4) + c(0, 10, 100) :  longer object length is not a multiple of shorter object length  | You nailed it! Good job!  |=================================================================================================== | 87%  | Before concluding this lesson, I'd like to show you a couple of time-saving tricks.  ...  |====================================================================================================== | 89%  | Earlier in the lesson, you computed z \* 2 + 100. Let's pretend that you made a mistake and that you meant to add 1000  | instead of 100. You could either re-type the expression, or...  ...  |========================================================================================================= | 92%  | In many programming environments, the up arrow will cycle through previous commands. Try hitting the up arrow on your  | keyboard until you get to this command (z \* 2 + 100), then change 100 to 1000 and hit Enter. If the up arrow doesn't work  | for you, just type the corrected command.  > z\*2+1000  [1] 1002.20 1018.00 1006.28  | You are quite good my friend!  |============================================================================================================ | 95%  | Finally, let's pretend you'd like to view the contents of a variable that you created earlier, but you can't seem to  | remember if you named it my\_div or myDiv. You could try both and see what works, or...  ...  |=============================================================================================================== | 97%  | You can type the first two letters of the variable name, then hit the Tab key (possibly more than once). Most programming  | environments will provide a list of variables that you've created that begin with 'my'. This is called auto-completion  | and can be quite handy when you have many variables in your workspace. Give it a try. (If auto-completion doesn't work  | for you, just type my\_div and press Enter.)  > my\_div  [1] 3.478505 3.181981 2.146460  | You are really on a roll!  |==================================================================================================================| 100%  | Would you like to receive credit for completing this course on dbsdataprojects.com?  1: No  2: Yes |
|  |
| |  | | --- | | Selection: | |