install.packages("swirl")  
library(swirl)  
swirl()

|  |  |  |  |
| --- | --- | --- | --- |
| **Activity** | **Hours** | **Code Functions/Learned** | **Explain what you learned** |
| **1: Basic Building Blocks** | 30 mins | <- , c() , ? “help “, | Setting variables, calling variables, vectors, c() = “concatenate”, ? help function, operators(`+`, `-`, `/`, `^`,’sqrt()’, ‘abs()’ ), vector recycling, tab for filling the blank |
| **2: Workspace and Files** | 30 mins | getwd() ,dir() or list.files() , ls() , args(), dir.create(“dic”), setwd(“dic”), file.create("file") , file.exists() , file.info() ,  file.rename("mytest.R", "mytest2.R"), file.copy(“”), file.copy("mytest2.R", "mytest3.R")  file.path()  dir.create(file.path('testdir2', 'testdir3'), recursive = TRUE) to create a file inside a folde in one go | getwd() to get the current working directory,  list.files() or dir() to see all files  ls(ar) to show all object in workspace, args to see what arguments a function can take,  dir.create() to create a new directory ,  setwd() to set a a working directory , file.create("file") creates a file  file.exists() checks if a file exists  file.info() to see info about a file  file.rename() to rename a file ,  file.copy(“”),file.copy() to copy a file  file.path() to show relative path |
| **3: Sequences of Numbers** | 30 mins | : operator , seq(),seq(0, 10, by=0.5) array increasing by .5, length("my\_seq"), rep(), | 1:20 creates a set of numbers  Seq(1:20) does the same as above  seq(5, 10, length=30) list of 30 nums  length("my\_seq") = sees the length of the array  1:length(my\_seq) array = length, rep(0, times = 40)replicates the same varible 40 times  rep(c(0, 1, 2), times = 10) replicates 012 10 times  rep(c(0, 1, 2), each = 10) replicates each 0, 1, 2 10 times |
| **4: Vectors** | 30 mins | tf<- num\_vect<1, num\_vect<-c(0.5, 55, -10,6)  `<`,`>=`,`>`, `<=`, `==` `!=`  | for or & for and  paste() | An atomic vector contains exactly one data type  A list may contain multiple data types  paste(my\_char, collapse = " ") separates the strigns with single spaces  paste("Hello", "world!", sep = " ") separate with a space  paste(c(1:3),c("X", "Y", "Z"), sep = "")  joins them together and leaves no space |
| **5: Missing Values** | 30 mins | is.na(), sum(my\_na), my\_data == NA | z <- rep(NA, 1000) 1000 NAs  NaN, which stands for 'not a number'  Infinity = Inf – Inf =Nan |
| **6: Sub-setting Vectors** | 1 hour | y <- x[!is.na(x)]  x[0] indexing  x[c(-2, -10)] | x[-c(2, 10)] gives us all elements of x EXCEPT for the 2nd and 10 elements  vect <- c(foo = 11, bar = 2, norf = NA) array = key and value  identical(vect,vect2) checks if variables are the same | Logical vectors can contain the values TRUE, FALSE, and NA (for 'not available')  y <- x[!is.na(x)] contains all of the non-NA values from x  y[y > 0] only positive values are selected  x[!is.na(x) & x > 0] requests only values of x that are both non-missing AND greater than zero  x[c(3, 5, 7)] subset of 3rd, 5th, and 7th elements of x |
| **7: Matrices and Data Frames** | 30 mins | dim() function tells us the 'dimensions' of an object  length(),  attributes(),  class() to see the class variable  my\_matrix2 <- matrix(1:20, nrow=4, ncol=5) creates a table with 4 rows and 5 columns  identical() , cbind() , data.frame() | matrices can only contain a single class of data  data frames can consist of many different classes of data  dim(my\_vector) <- c(4, 5)  attributes(my\_vector)  creates a table with 4 rows and 5 columns  identical(my\_matrix,my\_matrix2) to see if two tables are the same  patients <- c("Bill", "Gina", "Kelly", "Sean") Stores strings in a variable  cbind(patients,my\_matrix) appends patients to the first column of the 4-5 matrix  my\_data <- data.frame(patients, my\_matrix) stores the character vector of names right alongside the matrix of numbers  cnames <- c("patient", "age", "weight", "bp", "rating", "test") strings that will be used to name the each column  colnames(my\_data) <- cnames the way to do the above |
| **8: Logic** | 30 min | == ,< ,> <= != AND(&, &&) OR (|, ||), isTRUE(), identical() xor() “exclusive or”  sample(10) creates a random example | boolean values, logical operators  TRUE && c(TRUE, FALSE, FALSE)  [1] TRUE takes only the first value  xor() If one argument evaluates to TRUE and one argument evaluates to FALSE, then this function will  | return TRUE  xor(4 >= 9, 8 != 8.0) equal to TRUE |
| **9: Functions** |  |  |  |
| **10: lapply and sapply** |  |  |  |
| **Activity** | **Hours** | **Code Functions/Learned** | **Explain what you learned** |
| **11: vapply and tapply** |  |  |  |
| **12: Looking at Data** |  |  |  |
| **13: Simulation** |  |  |  |
| **14: Dates and Times** |  |  |  |
| **15: Base Graphics** |  |  |  |

         
                      
               
           
    
                       
                         
        