HW Week 4 MAP Function

Dr. Kimbria Jackson

2023-02-11

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──  
## ✔ ggplot2 3.3.6 ✔ purrr 0.3.4  
## ✔ tibble 3.1.8 ✔ dplyr 1.0.9  
## ✔ tidyr 1.2.0 ✔ stringr 1.4.1  
## ✔ readr 2.1.2 ✔ forcats 0.5.2  
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(purrr)  
library(dplyr)  
  
  
iris

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species  
## 1 5.1 3.5 1.4 0.2 setosa  
## 2 4.9 3.0 1.4 0.2 setosa  
## 3 4.7 3.2 1.3 0.2 setosa  
## 4 4.6 3.1 1.5 0.2 setosa  
## 5 5.0 3.6 1.4 0.2 setosa  
## 6 5.4 3.9 1.7 0.4 setosa  
## 7 4.6 3.4 1.4 0.3 setosa  
## 8 5.0 3.4 1.5 0.2 setosa  
## 9 4.4 2.9 1.4 0.2 setosa  
## 10 4.9 3.1 1.5 0.1 setosa  
## 11 5.4 3.7 1.5 0.2 setosa  
## 12 4.8 3.4 1.6 0.2 setosa  
## 13 4.8 3.0 1.4 0.1 setosa  
## 14 4.3 3.0 1.1 0.1 setosa  
## 15 5.8 4.0 1.2 0.2 setosa  
## 16 5.7 4.4 1.5 0.4 setosa  
## 17 5.4 3.9 1.3 0.4 setosa  
## 18 5.1 3.5 1.4 0.3 setosa  
## 19 5.7 3.8 1.7 0.3 setosa  
## 20 5.1 3.8 1.5 0.3 setosa  
## 21 5.4 3.4 1.7 0.2 setosa  
## 22 5.1 3.7 1.5 0.4 setosa  
## 23 4.6 3.6 1.0 0.2 setosa  
## 24 5.1 3.3 1.7 0.5 setosa  
## 25 4.8 3.4 1.9 0.2 setosa  
## 26 5.0 3.0 1.6 0.2 setosa  
## 27 5.0 3.4 1.6 0.4 setosa  
## 28 5.2 3.5 1.5 0.2 setosa  
## 29 5.2 3.4 1.4 0.2 setosa  
## 30 4.7 3.2 1.6 0.2 setosa  
## 31 4.8 3.1 1.6 0.2 setosa  
## 32 5.4 3.4 1.5 0.4 setosa  
## 33 5.2 4.1 1.5 0.1 setosa  
## 34 5.5 4.2 1.4 0.2 setosa  
## 35 4.9 3.1 1.5 0.2 setosa  
## 36 5.0 3.2 1.2 0.2 setosa  
## 37 5.5 3.5 1.3 0.2 setosa  
## 38 4.9 3.6 1.4 0.1 setosa  
## 39 4.4 3.0 1.3 0.2 setosa  
## 40 5.1 3.4 1.5 0.2 setosa  
## 41 5.0 3.5 1.3 0.3 setosa  
## 42 4.5 2.3 1.3 0.3 setosa  
## 43 4.4 3.2 1.3 0.2 setosa  
## 44 5.0 3.5 1.6 0.6 setosa  
## 45 5.1 3.8 1.9 0.4 setosa  
## 46 4.8 3.0 1.4 0.3 setosa  
## 47 5.1 3.8 1.6 0.2 setosa  
## 48 4.6 3.2 1.4 0.2 setosa  
## 49 5.3 3.7 1.5 0.2 setosa  
## 50 5.0 3.3 1.4 0.2 setosa  
## 51 7.0 3.2 4.7 1.4 versicolor  
## 52 6.4 3.2 4.5 1.5 versicolor  
## 53 6.9 3.1 4.9 1.5 versicolor  
## 54 5.5 2.3 4.0 1.3 versicolor  
## 55 6.5 2.8 4.6 1.5 versicolor  
## 56 5.7 2.8 4.5 1.3 versicolor  
## 57 6.3 3.3 4.7 1.6 versicolor  
## 58 4.9 2.4 3.3 1.0 versicolor  
## 59 6.6 2.9 4.6 1.3 versicolor  
## 60 5.2 2.7 3.9 1.4 versicolor  
## 61 5.0 2.0 3.5 1.0 versicolor  
## 62 5.9 3.0 4.2 1.5 versicolor  
## 63 6.0 2.2 4.0 1.0 versicolor  
## 64 6.1 2.9 4.7 1.4 versicolor  
## 65 5.6 2.9 3.6 1.3 versicolor  
## 66 6.7 3.1 4.4 1.4 versicolor  
## 67 5.6 3.0 4.5 1.5 versicolor  
## 68 5.8 2.7 4.1 1.0 versicolor  
## 69 6.2 2.2 4.5 1.5 versicolor  
## 70 5.6 2.5 3.9 1.1 versicolor  
## 71 5.9 3.2 4.8 1.8 versicolor  
## 72 6.1 2.8 4.0 1.3 versicolor  
## 73 6.3 2.5 4.9 1.5 versicolor  
## 74 6.1 2.8 4.7 1.2 versicolor  
## 75 6.4 2.9 4.3 1.3 versicolor  
## 76 6.6 3.0 4.4 1.4 versicolor  
## 77 6.8 2.8 4.8 1.4 versicolor  
## 78 6.7 3.0 5.0 1.7 versicolor  
## 79 6.0 2.9 4.5 1.5 versicolor  
## 80 5.7 2.6 3.5 1.0 versicolor  
## 81 5.5 2.4 3.8 1.1 versicolor  
## 82 5.5 2.4 3.7 1.0 versicolor  
## 83 5.8 2.7 3.9 1.2 versicolor  
## 84 6.0 2.7 5.1 1.6 versicolor  
## 85 5.4 3.0 4.5 1.5 versicolor  
## 86 6.0 3.4 4.5 1.6 versicolor  
## 87 6.7 3.1 4.7 1.5 versicolor  
## 88 6.3 2.3 4.4 1.3 versicolor  
## 89 5.6 3.0 4.1 1.3 versicolor  
## 90 5.5 2.5 4.0 1.3 versicolor  
## 91 5.5 2.6 4.4 1.2 versicolor  
## 92 6.1 3.0 4.6 1.4 versicolor  
## 93 5.8 2.6 4.0 1.2 versicolor  
## 94 5.0 2.3 3.3 1.0 versicolor  
## 95 5.6 2.7 4.2 1.3 versicolor  
## 96 5.7 3.0 4.2 1.2 versicolor  
## 97 5.7 2.9 4.2 1.3 versicolor  
## 98 6.2 2.9 4.3 1.3 versicolor  
## 99 5.1 2.5 3.0 1.1 versicolor  
## 100 5.7 2.8 4.1 1.3 versicolor  
## 101 6.3 3.3 6.0 2.5 virginica  
## 102 5.8 2.7 5.1 1.9 virginica  
## 103 7.1 3.0 5.9 2.1 virginica  
## 104 6.3 2.9 5.6 1.8 virginica  
## 105 6.5 3.0 5.8 2.2 virginica  
## 106 7.6 3.0 6.6 2.1 virginica  
## 107 4.9 2.5 4.5 1.7 virginica  
## 108 7.3 2.9 6.3 1.8 virginica  
## 109 6.7 2.5 5.8 1.8 virginica  
## 110 7.2 3.6 6.1 2.5 virginica  
## 111 6.5 3.2 5.1 2.0 virginica  
## 112 6.4 2.7 5.3 1.9 virginica  
## 113 6.8 3.0 5.5 2.1 virginica  
## 114 5.7 2.5 5.0 2.0 virginica  
## 115 5.8 2.8 5.1 2.4 virginica  
## 116 6.4 3.2 5.3 2.3 virginica  
## 117 6.5 3.0 5.5 1.8 virginica  
## 118 7.7 3.8 6.7 2.2 virginica  
## 119 7.7 2.6 6.9 2.3 virginica  
## 120 6.0 2.2 5.0 1.5 virginica  
## 121 6.9 3.2 5.7 2.3 virginica  
## 122 5.6 2.8 4.9 2.0 virginica  
## 123 7.7 2.8 6.7 2.0 virginica  
## 124 6.3 2.7 4.9 1.8 virginica  
## 125 6.7 3.3 5.7 2.1 virginica  
## 126 7.2 3.2 6.0 1.8 virginica  
## 127 6.2 2.8 4.8 1.8 virginica  
## 128 6.1 3.0 4.9 1.8 virginica  
## 129 6.4 2.8 5.6 2.1 virginica  
## 130 7.2 3.0 5.8 1.6 virginica  
## 131 7.4 2.8 6.1 1.9 virginica  
## 132 7.9 3.8 6.4 2.0 virginica  
## 133 6.4 2.8 5.6 2.2 virginica  
## 134 6.3 2.8 5.1 1.5 virginica  
## 135 6.1 2.6 5.6 1.4 virginica  
## 136 7.7 3.0 6.1 2.3 virginica  
## 137 6.3 3.4 5.6 2.4 virginica  
## 138 6.4 3.1 5.5 1.8 virginica  
## 139 6.0 3.0 4.8 1.8 virginica  
## 140 6.9 3.1 5.4 2.1 virginica  
## 141 6.7 3.1 5.6 2.4 virginica  
## 142 6.9 3.1 5.1 2.3 virginica  
## 143 5.8 2.7 5.1 1.9 virginica  
## 144 6.8 3.2 5.9 2.3 virginica  
## 145 6.7 3.3 5.7 2.5 virginica  
## 146 6.7 3.0 5.2 2.3 virginica  
## 147 6.3 2.5 5.0 1.9 virginica  
## 148 6.5 3.0 5.2 2.0 virginica  
## 149 6.2 3.4 5.4 2.3 virginica  
## 150 5.9 3.0 5.1 1.8 virginica

#1)   
#a) Use the R function nrow to confirm that the iris data frame has 150 rows.  
#Then use and show R code that features a map function to confirm that the iris   
#data frame has 150 rows.  
  
nrow(iris)

## [1] 150

#map(iris,nrow(x))  
  
#b) Each column of the iris data frame has a unique number of values or objects.   
#For example, the column Sepal.Length has 150 values but 35 of them are unique.  
#Use and show R code that features a map function to   
#find the number of unique values or objects for each column of the iris data   
#frame.  
view(iris)  
data("iris")  
iris\_uniq <- vector("double", ncol(iris))  
names(iris\_uniq) <- names(iris)  
for (i in names(iris)) {  
 iris\_uniq[i] <- n\_distinct(iris[[i]])  
}  
iris\_uniq

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species   
## 35 23 43 22 3

#2 Use and show R code that features a nested loop that will produce the 5 by 3 matrix shown below.  
matrix5 = matrix(nrow=3, ncol=5)   
for(c in 1:nrow(matrix5))   
{  
 for(j in 1:ncol(matrix5))   
 {  
 matrix5[c,j] = (c-j)\* -1   
 }  
}  
print(matrix5)

## [,1] [,2] [,3] [,4] [,5]  
## [1,] 0 1 2 3 4  
## [2,] -1 0 1 2 3  
## [3,] -2 -1 0 1 2

#3. Use and show R code that will produce a tibble that features 10 randomly   
#generated values that are normally distributed, with means of -10, 0, 10 and   
#100 respectfully. Run your code again, producing a second tibble, that   
#confirms random values, hence the second table will not have the same values.  
  
A<-rnorm(10,-10,1)  
A

## [1] -10.242436 -8.936545 -10.154046 -9.298199 -9.482216 -11.165145  
## [7] -8.998751 -10.050364 -11.490986 -9.690712

B<-rnorm(10,0,1)  
B

## [1] -0.07795516 0.24124194 1.82731489 0.55481122 0.56960523 -0.94394053  
## [7] -0.02135550 0.86641299 -0.05334714 -0.43161533

C<-rnorm(10, 10, 1)  
C

## [1] 10.839916 9.493025 10.345068 9.873744 8.756271 11.184764 9.377992  
## [8] 12.137082 9.825950 9.501452

D<-rnorm(10, 100, 1)  
D

## [1] 99.84456 99.33988 101.09359 99.02720 100.15691 98.03526 100.44100  
## [8] 99.64312 99.99383 101.91462

data.frame(A,B,C,D)

## A B C D  
## 1 -10.242436 -0.07795516 10.839916 99.84456  
## 2 -8.936545 0.24124194 9.493025 99.33988  
## 3 -10.154046 1.82731489 10.345068 101.09359  
## 4 -9.298199 0.55481122 9.873744 99.02720  
## 5 -9.482216 0.56960523 8.756271 100.15691  
## 6 -11.165145 -0.94394053 11.184764 98.03526  
## 7 -8.998751 -0.02135550 9.377992 100.44100  
## 8 -10.050364 0.86641299 12.137082 99.64312  
## 9 -11.490986 -0.05334714 9.825950 99.99383  
## 10 -9.690712 -0.43161533 9.501452 101.91462

data.frame(A,B,C,D)

## A B C D  
## 1 -10.242436 -0.07795516 10.839916 99.84456  
## 2 -8.936545 0.24124194 9.493025 99.33988  
## 3 -10.154046 1.82731489 10.345068 101.09359  
## 4 -9.298199 0.55481122 9.873744 99.02720  
## 5 -9.482216 0.56960523 8.756271 100.15691  
## 6 -11.165145 -0.94394053 11.184764 98.03526  
## 7 -8.998751 -0.02135550 9.377992 100.44100  
## 8 -10.050364 0.86641299 12.137082 99.64312  
## 9 -11.490986 -0.05334714 9.825950 99.99383  
## 10 -9.690712 -0.43161533 9.501452 101.91462

#4a) In statistics, a z score indicates the standard deviation distance between   
#the mean and a specific value of the data set. What formula is used to find a z   
#score? Use and show R coding that features a map function to   
#iteratively find z scores across the lists given above.  
  
X <- list(12, 14, 15, 18, 19, 22,10,18,18)  
Mean <- list(16, 16, 16, 16, 16,16,16,16,16)  
sd <- list(2, 2, 2, 2, 2,2,2,2,2)  
  
  
  
  
z\_score<-pmap\_dbl(list(X,Mean,sd),function(X, Mean, sd) ((X - Mean)/sd))  
z\_score

## [1] -2.0 -1.0 -0.5 1.0 1.5 3.0 -3.0 1.0 1.0

#4b) The test statistic for a population mean is given by the formula   
#((X - mean)/s/sqrt(n)) Use and show R coding that features a map function to  
#iteratively find test statistics for population means across the lists given  
#above  
  
pop\_mean<- pmap\_dbl(list(X,Mean,sd),function(X, Mean, sd) ((X - Mean)/sd))/sqrt(9)  
pop\_mean

## [1] -0.6666667 -0.3333333 -0.1666667 0.3333333 0.5000000 1.0000000 -1.0000000  
## [8] 0.3333333 0.3333333

#5a) Another purr package function is the keep( ) function. Research, explore,  
#and use the keep( ) function to extract all number from the vector V given   
#above that are less than 20  
  
V = c(10,15,17,22,32,38,42)  
  
keep(V, V <20)

## [1] 10 15 17

#5b) Another purr package function is the discard( ) function. Research,   
#explore, and use the discard( ) function to eliminate all numbers from the  
#vector V given above that are less than 20  
  
discard(V, V <20)

## [1] 22 32 38 42

#6) Another purr package function is the safely( ) function. Research, explore,   
#and apply the safely( ) function to the given vector below as illustrated.  
  
 U = list(10,15,"mary",22,32,"james",42)  
 map(U, safely(~ .x + 15))

## [[1]]  
## [[1]]$result  
## [1] 25  
##   
## [[1]]$error  
## NULL  
##   
##   
## [[2]]  
## [[2]]$result  
## [1] 30  
##   
## [[2]]$error  
## NULL  
##   
##   
## [[3]]  
## [[3]]$result  
## NULL  
##   
## [[3]]$error  
## <simpleError in .x + 15: non-numeric argument to binary operator>  
##   
##   
## [[4]]  
## [[4]]$result  
## [1] 37  
##   
## [[4]]$error  
## NULL  
##   
##   
## [[5]]  
## [[5]]$result  
## [1] 47  
##   
## [[5]]$error  
## NULL  
##   
##   
## [[6]]  
## [[6]]$result  
## NULL  
##   
## [[6]]$error  
## <simpleError in .x + 15: non-numeric argument to binary operator>  
##   
##   
## [[7]]  
## [[7]]$result  
## [1] 57  
##   
## [[7]]$error  
## NULL

#In four or five sentences, explain the specific output for this problem and how  
#the definition and the application of the safely( ) function is used.  
  
 #It appears that the output added 15 to the numbers in the list. However,   
 #there is a printout that an error message occurs when the list element is a  
 #string.

```