Homework 4

Sasha Uspenskaya

2024-02-11

```
library(tidyverse)
```

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── 
tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force
all conflicts to become errors
```

```
library(dplyr)
```

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_int(iris, ~nrow(data.frame(.)))
```

```
## Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
##          150          150          150          150          150
```

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.

```
map(iris, n_distinct)
```

```
## $Sepal.Length
## [1] 35
##
## $Sepal.Width
## [1] 23
##
## $Petal.Length
## [1] 43
##
## $Petal.Width
## [1] 22
##
## $Species
## [1] 3
```

2.)

```
z <- matrix( nrow = 3, ncol = 5)
for (m in 1:3) {
  for (n in 1:5) {
    z[m, n] <- -1*(m -n)
  }
}
print(z)
```

```
##      [,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.

```
tibble(
  x = rnorm(10, mean = -10),
  y = rnorm(10, mean = 0),
  z = rnorm(10, mean = 10),
  w = rnorm(10, mean = 100)
  )
```

```
## # A tibble: 10 × 4
##         x      y     z     w
##     <dbl>  <dbl> <dbl> <dbl>
##  1 -11.0  -0.163 10.5  101.
```

```
##  2 -12.7  -1.47  11.5  101.
##  3  -8.86 -0.934  9.30 102.
##  4 -10.6   0.981  9.78  99.3
##  5 -11.7   0.783  9.08  99.4
##  6  -9.04  0.500  9.50 101.
##  7  -9.80 -0.143 10.1   99.2
##  8 -10.0  -0.605  9.64 100.
##  9 -11.9  -0.565  9.11  98.4
## 10 -11.1   0.233  8.82  98.7
```

```
tibble(
  x = rnorm(10, mean = -10),
  y = rnorm(10, mean = 0),
  z = rnorm(10, mean = 10),
  w = rnorm(10, mean = 100)
)
```

```
## # A tibble: 10 × 4
##         x      y     z     w
##     <dbl>  <dbl> <dbl> <dbl>
##  1 -10.7  -1.15   9.04 100.
##  2 -10.1   0.265 11.1  102.
##  3  -8.09  0.558  8.27 101.
##  4 -10.7   0.956 11.1   98.7
##  5 -11.9   1.58   9.73  98.1
##  6 -10.3   1.70   8.74  99.1
##  7  -9.21  0.218  9.27 101.
##  8  -9.36  0.972 10.8  100.
##  9 -11.6  -1.90   8.89 101.
## 10  -9.81  0.549  9.88 100.
```

4)

```
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)
```

a) 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.

```
pmap_dbl(list(X, mean, sd), function(o, m, s) (o-m)/s)
```

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

b) 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.

```
pmap_dbl(list(X, mean, sd), function(o, m, s) (o-m)/(s/sqrt(length(X))))
```

```
## [1] -6.0 -3.0 -1.5  3.0  4.5  9.0 -9.0  3.0  3.0
```

5)

```
V = c(10,15,17,22,32,38,42)
```

a) 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

```
keep(V, ~.x < 20)
```

```
## [1] 10 15 17
```

b) 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,~.x<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.

The safely() function in purrr allows you to safely apply a function to each element of a list or vector, catching any errors and returning NULL instead of failing. In this example, we are mapping the function ~.x + 15 over the vector U, which contains both numeric and character elements. The + operator can only be applied to numeric values, so when it hits the character elements “mary” and “james”, it will produce an error. The safely() function

catches those errors and replaces the character elements with NULL in the output, rather than failing and stopping execution.