Assignment5

Samriddh Gupta

All the packages

library(tidyverse)

## -- Attaching packages ------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.4  
## v tidyr 1.0.2 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts ---------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidyverse)  
library(purrr)  
library(ggplot2)  
library(dplyr)  
options(java.parameters = "-Xmx2048m")  
library("xlsx")  
library("openxlsx")

##   
## Attaching package: 'openxlsx'

## The following objects are masked from 'package:xlsx':  
##   
## createWorkbook, loadWorkbook, read.xlsx, saveWorkbook, write.xlsx

require(data.table)

## Loading required package: data.table

##   
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':  
##   
## between, first, last

## The following object is masked from 'package:purrr':  
##   
## transpose

library(tidyr)

1. Write for loops to:
2. Compute the mean of every column in mtcars.

data<-mtcars  
  
output<-double()  
for (i in seq\_along(data)) {  
 output<-c(output,mean(data[[i]]))  
}  
output

## [1] 20.090625 6.187500 230.721875 146.687500 3.596563 3.217250  
## [7] 17.848750 0.437500 0.406250 3.687500 2.812500

1. Determine the type of each column in nycflights13::flights.

data<-nycflights13::flights  
output<-character()  
for (i in seq\_along(data)) {  
 output<-c(output,typeof(data[[i]]))  
}  
output

## [1] "integer" "integer" "integer" "integer" "integer" "double"   
## [7] "integer" "integer" "double" "character" "integer" "character"  
## [13] "character" "character" "double" "double" "double" "double"   
## [19] "double"

1. Compute the number of unique values in each column of iris.

data<-iris  
output<-character()  
for (i in seq\_along(data)) {  
 output<-c(output,length(unique(data[[i]])))  
}  
output

## [1] "35" "23" "43" "22" "3"

#iris %>% map\_int(function(x) length(unique(x)))

1. Generate 10 random normals for each of μ = −10, 0, 10, and 100.

means<-c(-10,0,10,100)  
out <- vector("list", length(means))  
for (i in seq\_along(means)) {  
 n <- 10  
 out[[i]] <- rnorm(n, means[[i]])  
}  
str(out)

## List of 4  
## $ : num [1:10] -10.71 -8.99 -10.52 -9.81 -8.42 ...  
## $ : num [1:10] 1.897 0.735 -0.261 -0.11 -0.535 ...  
## $ : num [1:10] 10.48 10.86 10.02 10.8 8.59 ...  
## $ : num [1:10] 99.3 101.1 98.3 101.3 98.9 ...

1. Eliminate the for loop in each of the following examples by taking advantage of an existing function that works with vectors: out <- "" for (x in letters) { out <- stringr::str\_c(out, x) }

x <- sample(100) sd <- 0 for (i in seq\_along(x)) { sd <- sd + (x[i] - mean(x)) ^ 2 } sd <- sqrt(sd / (length(x) - 1))

x <- runif(100) out <- vector(“numeric”, length(x)) out[1] <- x[1] for (i in 2:length(x)) { out[i] <- out[i - 1] + x[i] }

## First part  
  
alpha<-character()  
alpha<-letters  
  
  
x <- sample(100)  
sd <- 0  
  
sd<-sd(x)  
  
  
x <- runif(100)  
out <- vector("numeric", length(x))  
out[1] <- x[1]  
for (i in 2:length(x)) {  
 out[i] <- out[i - 1] + x[i]  
}

1. Imagine you have a directory full of CSV files that you want to read in. You have their paths in a vector, files <- dir(“data/”, pattern = “\.csv$”, full.names = TRUE), and now want to read each one with read\_csv(). Write the for loop that will load them into a single data frame. You may assume that all csv files contain the same variables and formats. Test your code by downloading the stock price .csv files from the data folder on blackboard and then print out rows 50,000 to 50,015.

all\_csv<-dir("data/",pattern = "\\.csv$",full.names = TRUE)  
all\_csv

## [1] "data/AFL2.csv" "data/CVX2.csv" "data/GE2.csv" "data/MMM2.csv"  
## [5] "data/PEP2.csv" "data/T2.csv" "data/VZ2.csv"

all\_dfs <- vector("list")  
for (i in seq\_along(all\_csv)) {  
 all\_dfs[[i]] <- read\_csv(all\_csv[[i]])  
}

## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_double(),  
## `Adj Close` = col\_double()  
## )  
## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_double(),  
## `Adj Close` = col\_double()  
## )  
## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_double(),  
## `Adj Close` = col\_double()  
## )  
## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_double(),  
## `Adj Close` = col\_double()  
## )  
## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_double(),  
## `Adj Close` = col\_double()  
## )  
## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_double(),  
## `Adj Close` = col\_double()  
## )  
## Parsed with column specification:  
## cols(  
## Date = col\_character(),  
## Open = col\_double(),  
## High = col\_double(),  
## Low = col\_double(),  
## Close = col\_double(),  
## Volume = col\_double(),  
## `Adj Close` = col\_double()  
## )

bind\_rows(all\_dfs)

## # A tibble: 70,163 x 7  
## Date Open High Low Close Volume `Adj Close`  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 7/23/2015 61.9 62.1 61.6 61.9 1531700 61.9  
## 2 7/22/2015 61.8 62.1 61.6 61.9 1601100 61.9  
## 3 7/21/2015 61.7 62.1 61.5 61.8 1818000 61.8  
## 4 7/20/2015 61.7 61.9 61.1 61.6 2247600 61.6  
## 5 7/17/2015 62.2 62.2 61.2 61.6 1616400 61.6  
## 6 7/16/2015 62.1 62.4 61.8 62.2 1585800 62.2  
## 7 7/15/2015 61.6 61.8 61.3 61.8 2677200 61.8  
## 8 7/14/2015 61.4 61.7 61.2 61.4 3030300 61.4  
## 9 7/13/2015 60.8 61.7 60.6 61.4 5609400 61.4  
## 10 7/10/2015 61.3 61.8 61.3 61.7 2617700 61.7  
## # ... with 70,153 more rows

all\_dfs[50000:50015]

## [[1]]  
## NULL  
##   
## [[2]]  
## NULL  
##   
## [[3]]  
## NULL  
##   
## [[4]]  
## NULL  
##   
## [[5]]  
## NULL  
##   
## [[6]]  
## NULL  
##   
## [[7]]  
## NULL  
##   
## [[8]]  
## NULL  
##   
## [[9]]  
## NULL  
##   
## [[10]]  
## NULL  
##   
## [[11]]  
## NULL  
##   
## [[12]]  
## NULL  
##   
## [[13]]  
## NULL  
##   
## [[14]]  
## NULL  
##   
## [[15]]  
## NULL  
##   
## [[16]]  
## NULL

1. Write a function that prints the mean of each numeric column in a data frame, along with its name. For example, show\_mean(iris) would print: show\_mean(iris) #> Sepal.Length: 5.84 #> Sepal.Width: 3.06 #> Petal.Length: 3.76 #> Petal.Width: 1.20

show\_means <- function(x) {  
   
 the\_class <- vector("logical", length(x))  
 for (i in seq\_along(x))   
 the\_class[[i]] <- is.numeric(x[[i]])  
   
 x <- x[the\_class]  
   
 for (i in seq\_along(x)) {  
 cat(paste0(names(x)[i], ": ", round(mean(x[[i]]), 2)), fill = TRUE)  
 }  
}  
show\_means(iris)

## Sepal.Length: 5.84  
## Sepal.Width: 3.06  
## Petal.Length: 3.76  
## Petal.Width: 1.2

1. Adapt col\_summary() so that it only applies to numeric columns . You may want to look at the purrr function keep. Test your function by computing the median on the flights dataframe.

col\_summary <- function(x) {  
   
 the\_numeric <- vector("logical", length(x))  
 for (i in seq\_along(x))   
 the\_numeric[[i]] <- is.numeric(x[[i]])  
   
 x <- x[the\_numeric]  
   
 the\_mean <- vector("numeric", length(x))  
 for (i in seq\_along(x))   
 cat(paste0(names(x)[i], ": ", round(mean(x[[i]]), 2)), fill = TRUE)  
}  
col\_summary(nycflights13::flights)

## year: 2013  
## month: 6.55  
## day: 15.71  
## dep\_time: NA  
## sched\_dep\_time: 1344.25  
## dep\_delay: NA  
## arr\_time: NA  
## sched\_arr\_time: 1536.38  
## arr\_delay: NA  
## flight: 1971.92  
## air\_time: NA  
## distance: 1039.91  
## hour: 13.18  
## minute: 26.23

1. Write code that uses one of the map functions to:
2. Compute the mean of every column in mtcars.

map\_dbl(mtcars, mean)

## mpg cyl disp hp drat wt qsec   
## 20.090625 6.187500 230.721875 146.687500 3.596563 3.217250 17.848750   
## vs am gear carb   
## 0.437500 0.406250 3.687500 2.812500

1. Determine the type of each column in nycflights13::flights.

map(nycflights13::flights, class)

## $year  
## [1] "integer"  
##   
## $month  
## [1] "integer"  
##   
## $day  
## [1] "integer"  
##   
## $dep\_time  
## [1] "integer"  
##   
## $sched\_dep\_time  
## [1] "integer"  
##   
## $dep\_delay  
## [1] "numeric"  
##   
## $arr\_time  
## [1] "integer"  
##   
## $sched\_arr\_time  
## [1] "integer"  
##   
## $arr\_delay  
## [1] "numeric"  
##   
## $carrier  
## [1] "character"  
##   
## $flight  
## [1] "integer"  
##   
## $tailnum  
## [1] "character"  
##   
## $origin  
## [1] "character"  
##   
## $dest  
## [1] "character"  
##   
## $air\_time  
## [1] "numeric"  
##   
## $distance  
## [1] "numeric"  
##   
## $hour  
## [1] "numeric"  
##   
## $minute  
## [1] "numeric"  
##   
## $time\_hour  
## [1] "POSIXct" "POSIXt"

1. Compute the number of unique values in each column of iris.

map(iris, ~ length(unique(.)))

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

1. Generate 10 random normals for each of μ = −10, 0, 10, and 100.

map(c(-10, 0, 10, 100), rnorm, n = 10)

## [[1]]  
## [1] -10.914617 -9.543983 -10.341860 -10.750048 -9.504284 -10.701598  
## [7] -12.059698 -8.490803 -9.283322 -10.006313  
##   
## [[2]]  
## [1] 0.31276303 2.92892791 0.78030951 0.32307234 -0.46127986 -1.32693117  
## [7] 0.07325098 0.70827969 -0.21460418 -0.02076439  
##   
## [[3]]  
## [1] 10.454527 10.033960 9.310741 9.671158 9.129981 10.537538 9.654231  
## [8] 9.184779 8.797381 9.517856  
##   
## [[4]]  
## [1] 99.12341 99.96673 99.74365 102.30394 101.58696 99.76512 98.96733  
## [8] 100.32564 99.08244 100.08438

1. How can you create a single vector that for each column in a data frame indicates whether or not it’s a factor? Test on the diamonds dataset.

data(diamonds)  
map\_lgl(diamonds, is.factor)

## carat cut color clarity depth table price x y z   
## FALSE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

1. Write code to generate 6 random numbers from a normal distribution with mean of 3 and standard deviation of 0.2, 10 random numbers from a uniform distribution from 2 to 5, and 12 random numbers from a poisson distribution with a lambda of 3.5. Set your seed to 613 at the beginning.

set.seed(613)  
out1 <- vector("list",3)  
for (i in seq\_along(length(out1))) {  
 out1[[i]]<-rnorm(6,3,0.2)  
 i=i+1  
 out1[[i]]<-runif(10,2,5)  
 i=i+1  
 out1[[i]]<-rpois(12,3.5)  
  
}  
str(out1)

## List of 3  
## $ : num [1:6] 3.41 2.75 2.91 3.03 2.82 ...  
## $ : num [1:10] 3.2 2.7 3.79 4.12 2.69 ...  
## $ : int [1:12] 5 9 6 3 6 1 3 5 7 1 ...

1. Combine the following datasets from Lahman: Master, Salaries, AwardsPlayers, Batting and BattingPost

library(Lahman)  
  
d1<-merge(Master,Salaries)  
d2<-merge(d1,AwardsPlayers)  
d3<-merge(d2,Batting)  
d4<-merge(d3,BattingPost)

Extra Credit (5 pts) Download the stock price .xlsx files from Blackboard to your data subfolder. Read in the “dividends” sheet from all .xlxs files in the data folder, add the company symbol from the filename as a new variable called Symbol and combine the data from the excel spreadsheets into one large tibble. Then tidy the data so the head looks like:

## Getting the files  
all\_xlsx<-dir("data/",pattern = "\\.xlsx$",full.names = TRUE)  
all\_xlsx

## [1] "data/AFL.xlsx" "data/CVX.xlsx" "data/GE.xlsx" "data/MMM.xlsx"  
## [5] "data/PEP.xlsx" "data/T.xlsx" "data/VZ.xlsx"

## adding the all files together  
all\_dfs <- vector("list")  
for (i in seq\_along(all\_xlsx)) {  
 all\_dfs[[i]] <- readxl::read\_xlsx(all\_xlsx[i],sheet = "dividends")  
}  
bind\_rows(all\_dfs)

## # A tibble: 509 x 5  
## Ex\_Div Pay\_date Dividends Declared   
## <dttm> <dttm> <dbl> <dttm>   
## 1 2017-11-14 00:00:00 2017-12-01 00:00:00 0.45 2017-10-25 00:00:00  
## 2 2017-08-21 00:00:00 2017-09-01 00:00:00 0.43 2017-07-27 00:00:00  
## 3 2017-05-22 00:00:00 2017-06-01 00:00:00 0.43 2017-04-27 00:00:00  
## 4 2017-02-15 00:00:00 2017-03-01 00:00:00 0.43 2017-01-31 00:00:00  
## 5 2016-11-14 00:00:00 2016-12-01 00:00:00 0.43 2016-10-27 00:00:00  
## 6 2016-08-22 00:00:00 2016-09-01 00:00:00 0.41 2016-07-28 00:00:00  
## 7 2016-05-16 00:00:00 2016-06-01 00:00:00 0.41 2016-04-26 00:00:00  
## 8 2016-02-11 00:00:00 2016-03-01 00:00:00 0.41 2016-02-01 00:00:00  
## 9 2015-11-16 00:00:00 2015-12-01 00:00:00 0.41 2015-10-27 00:00:00  
## 10 2015-08-17 00:00:00 2015-09-01 00:00:00 0.39 2015-07-28 00:00:00  
## # ... with 499 more rows, and 1 more variable: Record <dttm>

## getting the names of the files  
all\_filenames <- all\_xlsx %>%  
 basename() %>%  
 as.list()  
  
## combining the two tables  
all\_lists <- mapply(c, all\_dfs, all\_filenames, SIMPLIFY = FALSE)  
all\_result <- rbindlist(all\_lists, fill = T)  
names(all\_result)[6] <- "Symbol"  
#all\_result  
all\_result$Symbol=str\_remove\_all(all\_result$Symbol,".xlsx")  
all\_result$Dividends=round(all\_result$Dividends,digits = 2)  
all\_result %>%   
 pivot\_longer(c(Ex\_Div,Pay\_date,Declared,Record),names\_to = "Event", values\_to = "Date")

## # A tibble: 2,036 x 4  
## Dividends Symbol Event Date   
## <dbl> <chr> <chr> <dttm>   
## 1 0.45 AFL Ex\_Div 2017-11-14 00:00:00  
## 2 0.45 AFL Pay\_date 2017-12-01 00:00:00  
## 3 0.45 AFL Declared 2017-10-25 00:00:00  
## 4 0.45 AFL Record 2017-11-15 00:00:00  
## 5 0.43 AFL Ex\_Div 2017-08-21 00:00:00  
## 6 0.43 AFL Pay\_date 2017-09-01 00:00:00  
## 7 0.43 AFL Declared 2017-07-27 00:00:00  
## 8 0.43 AFL Record 2017-08-23 00:00:00  
## 9 0.43 AFL Ex\_Div 2017-05-22 00:00:00  
## 10 0.43 AFL Pay\_date 2017-06-01 00:00:00  
## # ... with 2,026 more rows