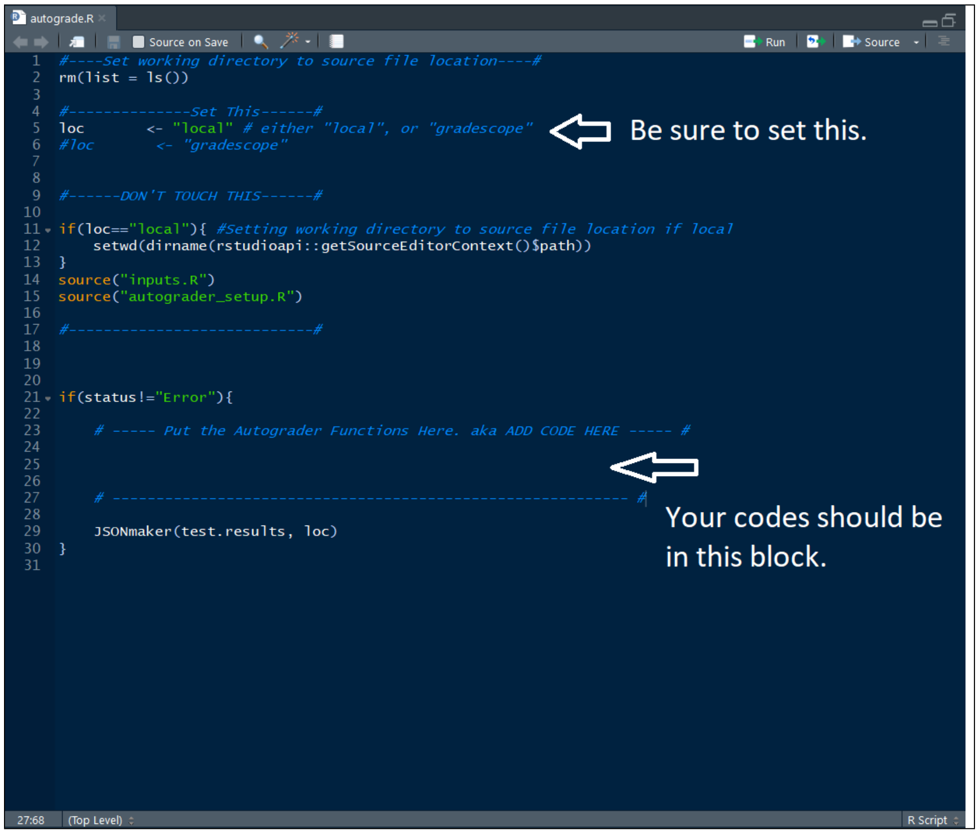
writing

Writing the Autograder

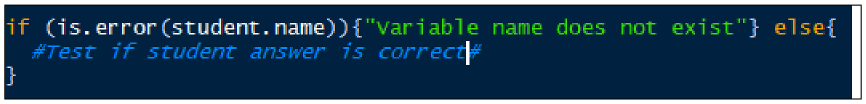
1: The bulk of your autograder codes should be in **autograde.R** in the respective homework folder. Make sure that the autograder codes are placed in the right location, and *loc* is set to correct variable for either `local` or `gradescope` grading.



```r
#----Set working directory to source file location----#
rm(list = ls())

#--------------Set This------#
loc       <- "local" # either "local", or "gradescope"
#loc       <- "gradescope"


#------DON'T TOUCH THIS------#

if(loc=="local"){ #Setting working directory to source file location if local
    setwd(dirname(rstudioapi::getSourceEditorContext()$path))
}
source("inputs.R")
source("autograder_setup.R")

#---------------------------#



if(status!="Error"){

    # ----- Put the Autograder Functions Here. aka ADD CODE HERE ----- #



    # ------------------------------------------------------------ #

    JSONmaker(test.results, loc)
}

```

2: Start your autograder by providing correct answers to each question. Make sure the answer variable's names are different from the ones specified in the prompts, and it is advised to comment your answers to later reference.

3: Compare your answers with the ones provided by students, and grade each questions accordingly. It is recommended to test if said variable exists before comparing. You can do so with **is.error()** function from **berryFunctions** library.



```r
if (is.error(student.name)){"Variable name does not exist"} else{
  #Test if student answer is correct#
}
```

4: How to grade XXX?

1 How do you grade a tibble? Three functions can be used to compare objects like dataframe and tibble. `all.equal()` is the default function of R. It can perform crude comparison between tibble objects, but minor changes like ordering or different grouping could affects its output. `compare()` from library *compare*. It has several built-in arguments that allowed you to specify if orders, data_types, or capitalizations matter in the comparison. But this function works poorly with tibbles especially when grouping, or factor levels are different. `all_equal()` from library *tidyverse*. It is similar to `compare()` in that it allowed you specify if order, etc. matter. It performs better with tibbles that have factors. But it would not recognise 1*n dataframes as valid input. Thus, it is recommended to use `all_equal (as.tibble(df1),as.tibble(df2))`. Troubleshooting: it often happens when minor class attributes could affects the autograder, like student answer has factor as a column while your answer converted it to characters. Or differences like double versus integer. Some easy fixes for these issues are `type_convert()` fomr library *readr*.

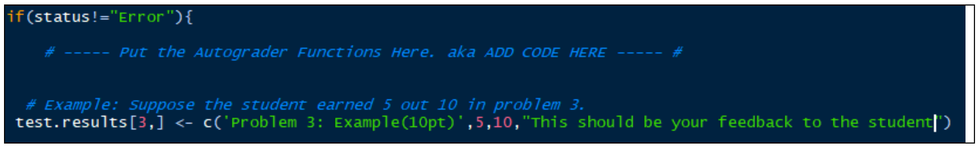
2 How do you grade a function? Sadly no function in R provided tools that could compare if two functions are fundamentally different. A good approximation to this could be to provide a variaty of inputs to feed into both functions and compare answers. It is adviced to prepare inputs that covers multiple data types, with either valid or invalid output for more accuracy. A great example can be found on Michael Guerzhoy's Github.

3 How do you grade a numeric value with potential round-offs? Though this arrises less than often, it could be the case when a numeric answer requries multiple steps of calculation, and some round-offs are involved. Many methods exist to fix this issue. One of them is to use `almost.equal()` from library *berryFunctions*. This function allowed you to do fuzzy comparison with a set scale.

4 How do you grade console outputs? Sometimes students are asked to print stuff. You can use the `sink()` function from base R to access their printed outputs. A simple way to do so is provided below. Notice that `student.output` is a character vector. Each element corresponds to a line of output from running the student's script. You can then use functions like `str_detect()` from the *stringr* package to check if a desired output does exist.

```r
sink("temp.txt") # The outputs of the following lines of code enclosed by sink() are sunk to a
                 # newly created "temp.txt" in the working directory. You can name this text file
                 # whatever you want.
source(rScript) # Run the student's script while recording the outputs.
sink() # Tell R we are done sinking and close "temp.txt".
student.output <- read_lines("temp.txt") # Read in the outputs of the student's script that is to
                                         # be graded.
file.remove("temp.txt") # Remove the temporary output file in the directory.
```

5: Save the grades in json file. This step is mostly automatic. In **autograder_setup.R**, functions should already initiate a dataframe **test.results**, with each row corresponding to one question and four columns: *description of the question*, *the score*, *max score*, and *notes*. To save the grades, you should make to fill each cell with corresponding inputs.



```r
if(status!="Error"){

   # ----- Put the Autograder Functions Here. aka ADD CODE HERE ----- #


 # Example: Suppose the student earned 5 out 10 in problem 3.
test.results[3,] <- c('Problem 3: Example(10pt)',5,10,"This should be your feedback to the student")
```

6: Pushing to Github by Git push.

7: My autograder broke, why? If this happens to you, don't fret. This happens **A LOT**. Try to see if you have done the following:

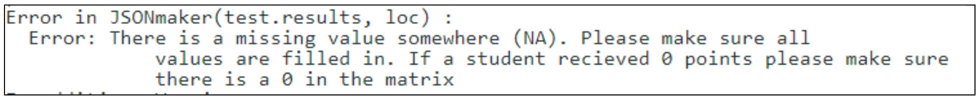
1 Pushed **autograde.R** before saving *loc* to *gradescope*.

2 Created the autograder in a branch other than **master** and did not merge the branches after pushing.

3 Used functions from packages that are installed but not declared, this can fixed by 'library(NameOfthePackage)'.

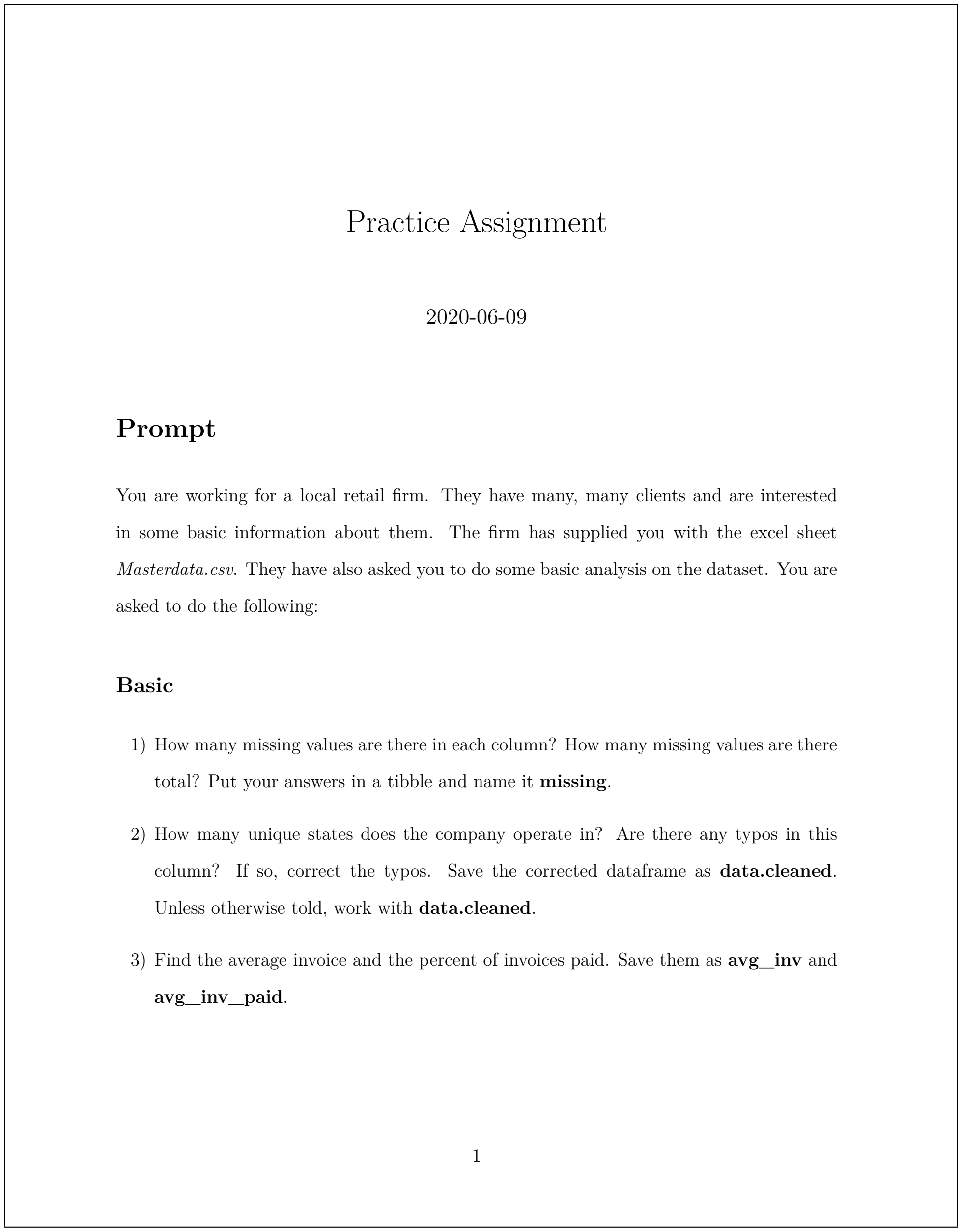
4 Used functions from packages that are not installed in the Virtual Machine, to fix this you should edit **setup.sh** in your **gradescope.zip**.

5 some parts of the autograder returned NA instead of correct grading, this is usually hinted by:



```
Error in JSONmaker(test.results, loc) :
  Error: There is a missing value somewhere (NA). Please make sure all
             values are filled in. If a student recieved 0 points please make sure
             there is a 0 in the matrix
```

Below is an example homework prompt with accompanying autograder:



Practice Assignment

2020-06-09

Prompt

You are working for a local retail firm. They have many, many clients and are interested in some basic information about them. The firm has supplied you with the excel sheet *Masterdata.csv*. They have also asked you to do some basic analysis on the dataset. You are asked to do the following:

Basic

1) How many missing values are there in each column? How many missing values are there total? Put your answers in a tibble and name it **missing**.
2) How many unique states does the company operate in? Are there any typos in this column? If so, correct the typos. Save the corrected dataframe as **data.cleaned**. Unless otherwise told, work with **data.cleaned**.
3) Find the average invoice and the percent of invoices paid. Save them as **avg_inv** and **avg_inv_paid**.

1

```r
#----Set working directory to source file location----#

# clear workspace
rm(list = ls())



#SET THIS!!!#
loc       <- "local" # either local, or gradescope
#------------#

#------DON'T TOUCH THIS------#

if(loc=="local"){ #Setting working directory to source file location if local
```

```r
    setwd(dirname(rstudioapi::getSourceEditorContext()$path))
}
source("inputs.R")
source("autograder_setup.R")

#----------------------------#



if (status != "Error") {
    # ----- Put the Autograder Functions Here ----- #

    # Create the answers. Notice they aren't named the same as the homework
    answer.1 <- tibble(Class=c(colnames(fakedata),"Total"),Num=c(colSums(is.na(fakedata)),sum(is.na(fak
    answer.2 <- fakedata
    answer.2$State[answer.2$State=='Arisona']<-'Arizona'
    answer.3 <- mean(answer.2$Invoice..USD..,na.rm =TRUE)
    answer.4 <- sum(answer.2[which(answer.2$Paid=='No'),3],na.rm = T)/sum(answer.2[,3],na.rm = T)

    # Grading <--> comparing answer key answers to student answers
    # Problem 1 score:
    p1.score <-
        if (is.error(missing) == TRUE) {
            #checking to make sure they saved the name right
            test.results[1, ] <- c(
                "Problem 1: Missing value (10 pt.)",
                0,
                10 ,
                "mssing not found. Please ensure the variable is properly named"
            )
        } else if (isTRUE (all_equal(missing , answer.1))) {
            test.results[1, ] <- c("Problem 1: Missing value (10 pt.)",
                                   10,
                                   10 ,
                                   "nice work")
        } else{
            test.results[1, ] <- c("Problem 1: Missing value (10 pt.)",
                                   0,
                                   10 ,
                                   "Try Again")
        }

    p2.score <-
        if (is.error(data.cleaned == TRUE)) {
            #checking to make sure they saved the name right
            test.results[2, ] <-
                c(
                    "Problem 2: Data cleaning (12 pt.)",
                    0,
                    12 ,
                    "data.cleaned not found. Please ensure the variable is properly named"
                )
        } else if (isTRUE(all_equal(data.cleaned,answer.2))) {
            test.results[2, ] <- c("Problem 2: Data cleaning (12 pt.)",
                                   12,
                                   12 ,
                                   "nice work")
        } else{
            test.results[2, ] <- c("Problem 2: Data cleaning (12 pt.)",
                                   0,
                                   12 ,
                                   "Try Again")
        }
    p3.score <-
        if (is.error(avg_invoice == TRUE) || is.error(avg_invoice_paid==TRUE)) {
            #checking to make sure they saved the name right
            if (is.error(avg_invoice==TRUE)){
                if (is.error(avg_invoice_paid==TRUE)){
                    test.results[3, ] <-
                        c(
                            "Problem 3: Average (8 pt.)",
                            0,
                            8 ,
                            "both avg_invoice and avg_invoice_paid not found. Please ensure the variabl
                        )
                } else {
                    if (avg_invoice_paid==answer.4){
                        test.results[3, ]<-c(
                            "Problem 3: Average (8 pt.)",
                            4,
                            8 ,
                            "avg_invoice not found. Please ensure the variable is properly named"
                        )
                    }  else{
                        test.results[3, ]<-c(
                            "Problem 3: Average (8 pt.)",
                            0,
                            8 ,
                            "avg_invoice not found. ang_invoice_paid incorrect. Please ensure the varia
                        )
                    }
                }
            }       else{
                if (avg_invoice==answer.3){
                    test.results[3, ]<- c(
                        "Problem 3: Average (8 pt.)",
                        4,
                        8 ,
                        "avg_invoice_paid not found. Please ensure the variable is properly named"
                    )
                }    else{
                    test.results[3, ]<- c(
                        "Problem 3: Average (8 pt.)",
                        0,
                        8 ,
                        "avg_invoice incorrect and avg_invoice_paid not found. Please ensure the variab
                )
            }
        }} else {
            if (avg_invoice == answer.3) {
                if (avg_invoice_paid==answer.4){
                    test.results[3, ] <- c("Problem 3: Average (8 pt.)",
                                           8,
                                           8 ,
                                           "nice work")} else {
                                               test.results[3, ] <- c("Problem 3: Average (8 pt.)",
                                                                      4,
                                                                      8 ,
                                                                      "check avg_invoice_paid")
                                           }
            }     else {
                if (avg_invoice_paid==answer.4){
                    test.results[3, ] <- c("Problem 3: Average (8 pt.)",
                                           4,
                                           8 ,
                                           "check avg_invoice")}  else{
                                               test.results[3, ] <-
                                                   c(
                                                       "Problem 3: Average (8 pt.)",
                                                       0,
                                                       8 ,
                                                       "Try again")        }

            }
        }
    p4.score<-
        if (is.error(data.cleaned.split==TRUE)){
            test.results[4,]<- c("Problem 4: Split Data (2 pt.)",
                                 0,
                                 2,
                                 'check data.cleaned.split')
        } else{
            if( isTRUE (all_eqqual(data.cleaned.split,answer.5))){
                test.results[4,]<- c("Problem 4: Split Data (2 pt.)",
                                     2,
                                     2,
                                     'Well done.')
            } else{
                test.results[4,]<- c("Problem 4: Split Data (2 pt.)",
                                     0,
                                     2,
                                     'Try again')
            }
        }


    # ---------------------------------------------- #

    JSONmaker(test.results, loc)
```

```
}
```