Stats102A, Summer 2023 - Homework 2

Luke Villanueva - 206039397

07/19/23

1. Teacher's Gradebook

a.

```
FUNCTION gen_gradebook()

  SET seed to UID

  FUNCTION gen_score
    RETURN random double from 0-1 * 100, change to integer
  END FUNCTION

  FUNCTION gen_UID
    SET rnum = random double from 0-3
    SET num = character of rnum, substring from index 0-9
    SET indx = index of "." in num
    SET res = substring of num from 1 to indx + substring of num from indx + 1 to length of num
    RETURN res
  END FUNCTION

  SET nStudents = 100

  SET gradebook = data frame, row = nStudents, col = 11

  SET column names of gradebook = "UID","Homework_1",
                          "Homework_2","Homework_3",
                          "Homework_4","Homework_5",
                          "Quiz_1","Quiz_2","Quiz_3",
                          "Quiz_4","Quiz_5"

  SET uid = empty vector

  FOR i in vector 1:nStudents
    SET uid = uid + gen_UID()
  END FOR
  SET gradebook[,"UID"] = uid

  FOR col in columns of gradebook WITHOUT UID
    SET grades = empty vector
    FOR i in 1:nStudents
      SET grades = grades + gen_score()
    END FOR
    SET gradebook[,col] = grades
  END FOR

  RETURN gradebook

END FUNCTION
```

```r
gradebook <- gen_gradebook()
```

b.

```
FUNCTION na.replace(vec, percent = 10)

  FUNCTION get_random_index(vec)
    RETURN random number from 1-length of vec INCLUSIVE
  END FUNCTION

  SET res = vec

  IF percent is not number
    STOP
  END IF
  IF any NA’s in vec
    WARNING
  END IF
  IF percent % 1 != 0
    STOP
  END IF
  IF percent < 0
    STOP
  END IF
  IF percent == 0
    STOP
  END IF
  IF percent > 100
    STOP
  END IF

  SET percent = percent / 100

  SET needed_nas = roundup(percent * length(vec))

  SET n_na = length(vector of indices of na’s in vec)

  SET na_left = needed_nas - n_na

  WHILE na_left > 0

    SET indx = NA

    WHILE indx is NA
```

```
      SET indx = get_random_index(vec)
    END WHILE

    SET res[indx] = NA
    SET na_left = na_left - 1

  END WHILE

  RETURN res

END FUNCTION
```

```r
# randomly replace data in vector with NA
# percent is inputted as an integer
# function will AT LEAST make the NA percent, but it can over do it
na.replace <- function(vec,percent = 10)
{

  # function to pick random index in vec
  get_random_index <- function(vec)
  {
    floor(runif(1,min = 1, max = length(vec)+1))
  }

  # result vector
  res <- vec

  # default of percent is 10

  # input validation for percent
  if(!is.numeric(percent))
  {
    stop("inputted percent is not numeric")
  }
  if(length(percent) != 1)
  {
    stop("inputted percent is a vector, and should be one value")
  }

  # input validation for vec
  # send warning if there are na's in vec already, telling user
  # that those na's will count too
  if(any(is.na(vec)))
  {warning("there exists NA's in inputted vector.
           These NA's will count towards the percentage of NA's")}

  # check if percent is inputted as a decimal
  if(percent %% 1 != 0)
  {stop("inputted percent is decimal, but should be integer")}

  # check if percent is negative
  if(percent < 0)
  {stop("inputted percent should be nonnegative")}
  # check if percent is 0
  if(percent == 0)
  {warning("inputted percent is 0, should be positive; function will still run")}

  # check if percent is bigger than 100 percent
  if(percent > 100)
  {stop("inputted percent is greater than 100, should be from 1-100")}

  # get how many to change to reach at least 10 percent
  percent <- percent / 100
  # round up to get AT LEAST percent
  needed_nas <- ceiling(percent * length(vec))

  # count how many NAs, subtract current NA's from
  n_na <- length(which(is.na(vec)))


  # get percentage of NA's left to do
  na_left <- needed_nas - n_na

  # while there are na left to make
  while(na_left > 0)
  {
    # get random indx
    indx <- NA
    # check if not already NA
    while(is.na(indx))
    {indx <- get_random_index(vec)}

    # once we get here, indx is not NA, so change to NA
    res[indx] <- NA

    # update na_left counter
    na_left <- na_left - 1
  }

  # no more needed na's, so return vec
  return(res)

}
```

```
gradebook$Homework_4 <- na.replace(gradebook$Homework_4,10)
gradebook$Quiz_4 <- na.replace(gradebook$Quiz_4,10)

# show there are 10 na's in specific columns
sum(is.na(gradebook$Homework_4))
```

```
## [1] 10
```

```
sum(is.na(gradebook$Quiz_4))
```

```
## [1] 10
```

c. Messy Impute - messy_impute()

```
FUNCTION messy_impute()

  INPUT df, center = "mean", margin, ...

  IF df is not the right class
    STOP
  END IF

  IF df cols != "UID","Homework_1",
                     "Homework_2","Homework_3",
                     "Homework_4","Homework_5",
                     "Quiz_1","Quiz_2","Quiz_3",
                     "Quiz_4","Quiz_5"
    STOP
  END IF

  IF df does not have valid dimensions
    STOP
  END IF

  IF center is not character OR chars in center > 1 OR center != "mean" or "median"
    STOP
  END IF

  IF margin is not numeric OR length margin > 1
    STOP
  END IF


  FOR col in columns of df[1:length(df)]
    SET df[,col] = numeric of df[,col]
  END FOR

  SET fun = NULL
  IF center == "mean"
    SET fun = mean()
  END IF
  ELSE
    SET fun == median()
  END ELSE


  IF margin == 1

    FOR row in rows of df
      IF any of df[row,] is NA
        SET impute_val = round(fun(double(df[row,-1]), remove NA’s, ...), digitRound = 2)

        FOR i in indices of df[row,] == NA
          SET df[row,i] == impute_val
        END FOR
    END FOR

  END IF

  ELSE

    FOR col in cols of df
      IF any of df[,col] is NA
        SET impute_val = round(fun(double(df[,col]), remove NA's, ...), digitRound = 2)

        FOR i in indices of df[,col] == NA
          SET df[i,col] == impute_val
        END FOR
    END FOR

  END ELSE

  FOR col in cols of df
    SET df[i,col] = formatted to character rounded to 2 digits
  END FOR

  RETURN df

END FUNCTION
```

d.

```
# selecting and printing two students with missing HW 4 and missing Q4

print("Students missing Homework_4:")
```

```
## [1] "Students missing Homework_4:"
```

```
print(gradebook %>% filter(is.na(Homework_4)) %>% slice(1:2))
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00       <NA>      91.00  49.00
## 2 119352283      93.00      49.00      11.00       <NA>       9.00  18.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
```

```
print("Students missing Quiz_4:")
```

```
## [1] "Students missing Quiz_4:"
```

```
print(gradebook %>% filter(is.na(Quiz_4)) %>% slice(1:2))
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 2 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1  16.00  97.00   <NA>  19.00
## 2  86.00   6.00   <NA>  30.00
```

```
# get UID of example students
na_uids <- gradebook %>% mutate(index = rownames(gradebook)) %>% filter(is.na(Homework_4)) %>% slice(1:

na_uids <- rbind(na_uids, gradebook %>% mutate(index = rownames(gradebook)) %>% filter(is.na(Quiz_4)) %

# get indices of example students
indx <- as.numeric(na_uids %>% pull(index))

# print example students
gradebook %>% mutate(index = as.numeric(rownames(gradebook))) %>% slice(indx) %>% select(-index)
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00       <NA>      91.00  49.00
## 2 119352283      93.00      49.00      11.00       <NA>       9.00  18.00
## 3 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 4 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
## 3  16.00  97.00   <NA>  19.00
## 4  86.00   6.00   <NA>  30.00
```

```
# messy_impute cases
messy_impute(gradebook,"mean", 1) %>% mutate(index = as.numeric(rownames(gradebook))) %>% slice(indx) %
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00      52.78      91.00  49.00
## 2 119352283      93.00      49.00      11.00      39.00       9.00  18.00
## 3 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 4 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
## 3  16.00  97.00  42.11  19.00
## 4  86.00   6.00  27.11  30.00
```

```
messy_impute(gradebook,"mean", 2) %>% mutate(index = as.numeric(rownames(gradebook))) %>% slice(indx) %
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00      47.47      91.00  49.00
## 2 119352283      93.00      49.00      11.00      47.47       9.00  18.00
## 3 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 4 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
## 3  16.00  97.00  51.43  19.00
## 4  86.00   6.00  51.43  30.00
```

```
messy_impute(gradebook,"median", 1) %>% mutate(index = as.numeric(rownames(gradebook))) %>% slice(indx)
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00      49.00      91.00  49.00
## 2 119352283      93.00      49.00      11.00      18.00       9.00  18.00
## 3 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 4 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
## 3  16.00  97.00  34.00  19.00
## 4  86.00   6.00  13.00  30.00
```

```
messy_impute(gradebook,"median", 2) %>% mutate(index = as.numeric(rownames(gradebook))) %>% slice(indx)
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00      45.00      91.00  49.00
## 2 119352283      93.00      49.00      11.00      45.00       9.00  18.00
## 3 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 4 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
## 3  16.00  97.00  48.00  19.00
## 4  86.00   6.00  48.00  30.00
```

```
messy_impute(gradebook,"mean", 1, trim = 0.25) %>% mutate(index = as.numeric(rownames(gradebook))) %>% s
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00      54.20      91.00  49.00
## 2 119352283      93.00      49.00      11.00      30.80       9.00  18.00
## 3 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 4 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
## 3  16.00  97.00  35.40  19.00
## 4  86.00   6.00  20.40  30.00
```

```
messy_impute(gradebook,"mean", 2, trim= 0.25) %>% mutate(index = as.numeric(rownames(gradebook))) %>% sl
```

```
##         UID Homework_1 Homework_2 Homework_3 Homework_4 Homework_5 Quiz_1
## 1 171873817      80.00      31.00      99.00      46.11      91.00  49.00
## 2 119352283      93.00      49.00      11.00      46.11       9.00  18.00
## 3 288267538      70.00      41.00      19.00      24.00      34.00  59.00
## 4 282353196       7.00       3.00      51.00       2.00      46.00  13.00
##   Quiz_2 Quiz_3 Quiz_4 Quiz_5
## 1   1.00  84.00  27.00  13.00
## 2  60.00  94.00  16.00   1.00
## 3  16.00  97.00  49.52  19.00
## 4  86.00   6.00  49.52  30.00
```

e. gradebook_tidy - tidify()

```
# assumes there is gradebook name
tidify_gradebook <- function()
{

  # INPUT VALIDATION
  if(!exists("gradebook"))
  {stop("there should exists a \"gradebook\" object that holds UID, Homeworks, Quizzes")}

  # work with homework, mutate order of homework
  hw <- gradebook %>% select(c("UID","Homework_1","Homework_2","Homework_3","Homework_4","Homework_5"))

  # work with quizzes, mutate order
  qz <- gradebook %>% select(c("UID","Quiz_1","Quiz_2","Quiz_3","Quiz_4","Quiz_5")) %>% pivot_longer(c(

  res <- hw %>% left_join(qz, by = c("UID","order_num")) %>% select(c("UID", "order_num","HW_Score", "Q

  return(res)

}

gradebook_tidy <- tidify_gradebook()
```

f. tidy_impute()

```
FUNCTION tidy_impute()

  INPUT df, center = "mean", margin, ...

  IF df is not the right class
    STOP
  END IF

  IF df cols != "UID","Assgn_Number", "Homework", "Quiz"
    STOP
  END IF

  IF df does not have valid dimensions
    STOP
  END IF

  IF center is not character OR chars in center > 1 OR center != "mean" or "median"
    STOP
  END IF
  IF margin is not numeric OR length margin > 1
    STOP
  END IF


  FOR col in columns of df[3:4]
    SET df[,col] = numeric of df[,col]
  END FOR

  SET fun = NULL
  IF center == "mean"
    SET fun = mean()
  END IF
  ELSE
    SET fun == median()
  END ELSE


  IF margin == 1

    FOR row in rows of df
      IF any of df[row,] is NA
        SET impute_val = round(fun(double(df[row,3:4]), remove NA's, ...), digitRound = 2)

        FOR i in indices of df[row,] == NA
          SET df[row,i] == impute_val
        END FOR

    END FOR

  END IF

  ELSE

    FOR col in cols of df[3:4]
      IF any of df[,col] is NA
        SET impute_val = round(fun(double(df[,col]), remove NA's, ...), digitRound = 2)

        FOR i in indices of df[,col] == NA
          SET df[i,col] == impute_val
        END FOR
    END FOR

  END ELSE

  FOR col in cols of df [3:4]
    SET df[i,col] = formatted to character rounded to 2 digits
  END FOR

  RETURN df

END FUNCTION
```

g. tidy_impute Demo

```r
# get UID of example students
na_uids <- gradebook_tidy %>% mutate(index = rownames(gradebook_tidy)) %>% filter(is.na(Homework)) %>% 

na_uids <- rbind(na_uids, gradebook_tidy %>% mutate(index = rownames(gradebook_tidy)) %>% filter(is.na(

# get indices of example students
indx <- as.numeric(na_uids %>% pull(index))

# print example students
gradebook_tidy %>% mutate(index = as.numeric(rownames(gradebook_tidy))) %>% slice(indx) %>% select(-inde
```

```
## # A tibble: 4 x 4
##   UID       Assgn_Num Homework Quiz
##   <chr>         <dbl> <chr>    <chr>
## 1 171873817         4  <NA>    27.00
## 2 119352283         4  <NA>    16.00
## 3 288267538         4 "24.00"  <NA>
## 4 282353196         4 " 2.00"  <NA>
```

```r
tidy_impute(gradebook_tidy,"mean", 1) %>% mutate(index = as.numeric(rownames(gradebook_tidy))) %>% slic
```

```
## Warning in tidy_impute(gradebook_tidy, "mean", 1): input margin says to impute via row, i.e. missing
##            will copy other value. The code will still run
```

```
## # A tibble: 4 x 4
##   UID       Assgn_Num Homework Quiz
##   <chr>         <dbl> <chr>    <chr>
## 1 171873817         4 "27.00"  "27.00"
## 2 119352283         4 "16.00"  "16.00"
## 3 288267538         4 "24.00"  "24.00"
## 4 282353196         4 " 2.00"  " 2.00"
```

```r
tidy_impute(gradebook_tidy,"mean", 2) %>% mutate(index = as.numeric(rownames(gradebook_tidy))) %>% slic
```

```
## # A tibble: 4 x 4
##   UID       Assgn_Num Homework Quiz
##   <chr>         <dbl> <chr>    <chr>
## 1 171873817         4 "47.39"  27.00
## 2 119352283         4 "47.39"  16.00
## 3 288267538         4 "24.00"  47.84
## 4 282353196         4 " 2.00"  47.84
```

```r
tidy_impute(gradebook_tidy, "median", 1) %>% mutate(index = as.numeric(rownames(gradebook_tidy))) %>% sl
```

```
## Warning in tidy_impute(gradebook_tidy, "median", 1): input margin says to impute via row, i.e. missi
##            will copy other value. The code will still run
## # A tibble: 4 x 4
##   UID       Assgn_Num Homework Quiz
##   <chr>         <dbl> <chr>    <chr>
## 1 171873817         4 "27.00"  "27.00"
## 2 119352283         4 "16.00"  "16.00"
## 3 288267538         4 "24.00"  "24.00"
## 4 282353196         4 " 2.00"  " 2.00"
```

```
tidy_impute(gradebook_tidy, "median", 2) %>% mutate(index = as.numeric(rownames(gradebook_tidy))) %>% sl
```

```
## # A tibble: 4 x 4
##   UID       Assgn_Num Homework Quiz
##   <chr>         <dbl> <chr>    <chr>
## 1 171873817         4 "46.50"  27.00
## 2 119352283         4 "46.50"  16.00
## 3 288267538         4 "24.00"  48.00
## 4 282353196         4 " 2.00"  48.00
```

```
tidy_impute(gradebook_tidy, "mean", 1, trim = 0.25) %>% mutate(index = as.numeric(rownames(gradebook_ti
```

```
## Warning in tidy_impute(gradebook_tidy, "mean", 1, trim = 0.25): input margin says to impute via row,
##              will copy other value. The code will still run
```

```
## # A tibble: 4 x 4
##   UID       Assgn_Num Homework Quiz
##   <chr>         <dbl> <chr>    <chr>
## 1 171873817         4 "27.00"  "27.00"
## 2 119352283         4 "16.00"  "16.00"
## 3 288267538         4 "24.00"  "24.00"
## 4 282353196         4 " 2.00"  " 2.00"
```

```
tidy_impute(gradebook_tidy, "mean", 2, trim = 0.25) %>% mutate(index = as.numeric(rownames(gradebook_ti
```

```
## # A tibble: 4 x 4
##   UID       Assgn_Num Homework Quiz
##   <chr>         <dbl> <chr>    <chr>
## 1 171873817         4 "46.49"  27.00
## 2 119352283         4 "46.49"  16.00
## 3 288267538         4 "24.00"  46.98
## 4 282353196         4 " 2.00"  46.98
```

2. Short Answers

a.

```
gdp <- read.csv("gdp-countries.csv")

# display 10 observations
sampGdp <- gdp[1:10,]
# first 10 cols
print(sampGdp[,1:10])
```

```
##    Aruba Africa.Eastern.and.Southern Afghanistan Africa.Western.and.Central
## 1     NA                 20082715854   537777811                11404280784
## 2     NA                 20509450945   548888896                11128050589
## 3     NA                 22350432991   546666678                11943353288
## 4     NA                 26758657008   751111191                12676515454
## 5     NA                 24464990260   800000044                13838577015
## 6     NA                 27878943636  1006666638                14862472886
## 7     NA                 30313844064  1399999967                15832846881
## 8     NA                 31375545887  1673333418                14426432397
## 9     NA                 34187176314  1373333367                14880350847
## 10    NA                 39248424433  1408888922                16882094303
##    Angola Albania Andorra  Arab.World United.Arab.Emirates   Argentina
## 1      NA      NA      NA          NA                   NA          NA
## 2      NA      NA      NA          NA                   NA          NA
## 3      NA      NA      NA          NA                   NA 24450604878
## 4      NA      NA      NA          NA                   NA 18272123664
## 5      NA      NA      NA          NA                   NA 25605249382
## 6      NA      NA      NA          NA                   NA 28344705967
## 7      NA      NA      NA          NA                   NA 28630474728
## 8      NA      NA      NA          NA                   NA 24256667553
## 9      NA      NA      NA 35087069740                   NA 26436857247
## 10     NA      NA      NA 38236378393                   NA 31256284544
```

```
web <- read.csv("Most Popular websites.csv")

sampWeb <- web[1:10,]
# first 10 cols
print(sampWeb[,1:10])
```

```
##       Website
## 1         AOL
## 2     Prodigy
## 3   Bloomberg
## 4       Apple
## 5         MTV
## 6        IMBD
## 7         BBC
## 8  Britannica
## 9       Yahoo
## 10        MSN
##
## 1                                                     https://cdn.iconscout.com/icon/free/png-256/aol-
## 2                                                                https://cdn.worldvectorlogo.com/logos
## 3                                     https://i.pinimg.com/originals/71/b5/70/71b5706762354172b7ba6ea
## 4  https://upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Apple_logo_black.svg/1200px-Apple_logo_
## 5                    https://upload.wikimedia.org/wikipedia/commons/thumb/7/76/MTV_Logo.svg/625px-MTV_
## 6       https://upload.wikimedia.org/wikipedia/commons/thumb/6/69/IMDB_Logo_2016.svg/575px-IMDB_Logo_
## 7                                      https://www.newbridge-health.org.uk/wp-content/uploads/2019/08/
## 8                                                          https://www.britannica.com/resources/image
## 9                                                 https://cdn.iconscout.com/icon/free/png-256/yahoo-
## 10                             https://www.logolynx.com/images/logolynx/f8/f881c400b07a00dcbb192271
##       X1993  X1993.1    X1994    X1995    X1996     X1997     X1998     X1999
## 1  25600000 25600000 27280000 34000000 90432000 152510000 293998843 501100000
## 2   4600000  4600000  5147000  7150000        0        NA        NA        NA
## 3    360000   360000   380000   455000        0        NA        NA        NA
## 4    147000   147000   185000        0       NA        NA        NA        NA
## 5     43500    43500    60000        0       NA        NA        NA        NA
## 6     36000    36000        0       NA       NA        NA        NA        NA
## 7        NA       NA    56100        0       NA        NA        NA        NA
## 8        NA       NA   323000  1996000        0        NA        NA        NA
## 9        NA       NA 11490000 30500000 34731000  64676000 122045849 168400000
## 10       NA       NA       NA  1432000 13847000  33229000  71697663 181300000
```

```
watch <- read.csv("swiss watch brands.csv")

sampWatch <- watch[1:10,]
# first 10 cols
print(sampWatch[,1:10])
```

```
##              Brand          Brand2 X2006 X2006.1 X2007 X2008 X2009 X2010 X2011
## 1            Rolex     Independent  2840    2840  3400  3750  3000  3400  4000
## 2  Cartier watches Richemont Group  1770    1770  1943  1630  1540  1800  2200
## 3            Omega    Swatch Group  1350    1350  1600  1480  1380  1750  1950
## 4   Patek Philippe     Independent   611     611   709   830   830   850  1050
## 5        TAG Heuer      LVMH Group   822     822   970   915   690   760   880
## 6           Swatch    Swatch Group   610     610   680   675   640   710   670
## 7           Tissot    Swatch Group   360     360   400   475   470   640   800
## 8  Audemars Piguet     Independent   372     372   490   506   460   520   600
## 9         Longines    Swatch Group   325     325   340   440   430   570   890
## 10 Chopard watches     Independent   409     409   455   487   413   442   497
##    X2012
## 1   4500
## 2   2380
## 3   2230
## 4   1150
## 5    980
## 6    720
## 7    960
## 8    640
## 9   1120
## 10   567
```

The dataset `gdp` displays each country's GDP over time. Only the country's name and GDP were explicitly collected. The row order is assumed to be the years increasing. The rows are GDP over time and the columns are country names.

LINK TO GDP

The dataset `web` shows the popularity of each listed website. Website name, where the image of the company can be located, and the years with the respective website popularity were collected. The rows are different websites and the columns are the website names, image location, and years.

LINK TO WEB

The dataset `watch` shows an undisclosed measured metric of different watch brands over time. The rows are different brands and the columns are the metric changing over time.

LINK TO WATCH

b.

Tidy gdp

```
# print first 10 rows
print("sampGdp First 10 Rows:")
```

```
## [1] "sampGdp First 10 Rows:"
```

```
print((sampGdp %>% mutate(time = as.numeric(rownames(sampGdp))) %>% pivot_longer(colnames(sampGdp), name
```

```
## # A tibble: 10 x 3
##     time Countries                           GDP
##    <dbl> <chr>                             <dbl>
##  1     1 Aruba                                NA
##  2     1 Africa.Eastern.and.Southern 20082715854
##  3     1 Afghanistan                   537777811.
##  4     1 Africa.Western.and.Central  10404280784
##  5     1 Angola                               NA
##  6     1 Albania                              NA
##  7     1 Andorra                              NA
##  8     1 Arab.World                           NA
##  9     1 United.Arab.Emirates                 NA
## 10     1 Argentina                            NA
```

Tidy web

```
# print first 10 rows
print((web %>% pivot_longer(colnames(web)[-c(1,2)], names_to = "Years", values_to = "Popularity") %>% mu
```

```
## # A tibble: 10 x 4
##    Website Image.URL                                                   Years Popul~1
##    <chr>   <chr>                                                       <chr>   <dbl>
##  1 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1993   2.56e7
##  2 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1993~  2.56e7
##  3 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1994   2.73e7
##  4 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1995   3.4 e7
##  5 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1996   9.04e7
##  6 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1997   1.53e8
##  7 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1998   2.94e8
##  8 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 1999   5.01e8
##  9 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 2000   7.71e8
## 10 AOL     https://cdn.iconscout.com/icon/free/png-256/aol-1-2827~ 2001   8.04e8
## # ... with abbreviated variable name 1: Popularity
```

Tidy watch

```r
# print first 10 rows
print((sampWatch %>% pivot_longer(colnames(sampWatch)[-c(1,2)], names_to = "Years", values_to = "Measur
```

```
## # A tibble: 10 x 4
##    Brand Brand2      Years  Measure
##    <chr> <chr>       <chr>    <int>
##  1 Rolex Independent 2006      2840
##  2 Rolex Independent 2006.1    2840
##  3 Rolex Independent 2007      3400
##  4 Rolex Independent 2008      3750
##  5 Rolex Independent 2009      3000
##  6 Rolex Independent 2010      3400
##  7 Rolex Independent 2011      4000
##  8 Rolex Independent 2012      4500
##  9 Rolex Independent 2013      4600
## 10 Rolex Independent 2014      4800
```