Data Cleaning

Computational Statistics

October 17, 2023

Importing and tidying data

Save in your computer the attached CSV files. For each problem, complete the following tasks:

(1) read the CSV file into R, and assign the imported dataset to a variable;
(2) display the imported dataset;
(3) investigate if the data type of each column is appropriate, and correct it when necessary;
(4) explain whether the dataset is tidy or not; and
(5) if not, tidy up the data using functions from `tidyr` and (if needed) `dplyr`.

Hint: Read the original paper on Tidy Data (https://www.jstatsoft.org/article/view/v059i10/v59i10.pdf) to learn more about the concept and principles of tidy data.

1a. Import and tidy data stored in `preg.csv`.

Solution:

```r
library(readr) #Load the library needed to import the csv files
preg<-read_csv("preg.csv") #Assigning imported data to a variable
```

```
## Rows: 3 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): name
## dbl (2): treatmenta, treatmentb
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```r
cat("
    (2) Printing the data will result in the following,  ")
```

```
##
##     (2) Printing the data will result in the following,
preg
```

```
## # A tibble: 3 x 3
##   name          treatmenta treatmentb
##   <chr>              <dbl>      <dbl>
## 1 John Smith            NA         18
## 2 Jane Doe               4          1
## 3 Mary Johnson           6          7
```

```
cat("
    (3) The datatype of each column in the given data is as follows, ")
```

```
##
##     (3) The datatype of each column in the given data is as follows,
```

```
str(preg)
```

```
## spc_tbl_ [3 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ name      : chr [1:3] "John Smith" "Jane Doe" "Mary Johnson"
##  $ treatmenta: num [1:3] NA 4 6
##  $ treatmentb: num [1:3] 18 1 7
##  - attr(*, "spec")=
##   .. cols(
##   ..   name = col_character(),
##   ..   treatmenta = col_double(),
##   ..   treatmentb = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The columns are of the correct datatypes name="character", treatmenta="double" and treatmentb="double"

(4) The given dataset "preg" is not tidy due to the following reasons

(a) Every column is not a varibale - The columns *Treatementa* and *Treatmentb* are not separate variables but values of the same varibale *Treatment*.

(5) The data is not tidy as each column represents values and not variables.To tidy the data we first check for missing values and remove the missing values.

```r
#To check for missing values in the data
missing_1<-any(is.na(preg))
if(missing_1)
{
  cat("There are missing values in the data.")
} else{cat("There are no missing values in the data. ")}
```

```
## There are missing values in the data.
```

In this experiment, the missing value represents an observation that should have been made, but was not, so it is important to keep it.

```r
library(tidyr)
preg<- preg |>
          pivot_longer(c("treatmenta", "treatmentb"),names_to="treatment", values_to="values")
cat("The tibble after tidying the data is as below, ")
```

```
## The tibble after tidying the data is as below,
```

```r
preg
```

```
## # A tibble: 6 x 3
##   name         treatment  values
##   <chr>        <chr>       <dbl>
## 1 John Smith   treatmenta     NA
## 2 John Smith   treatmentb     18
## 3 Jane Doe     treatmenta      4
## 4 Jane Doe     treatmentb      1
## 5 Mary Johnson treatmenta      6
## 6 Mary Johnson treatmentb      7
```

1b. Import and tidy data stored in `pew.csv`.

Solution:

```r
pew<-read_csv("pew.csv") #Assigning imported data to a variable
```

```
## Rows: 18 Columns: 11
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): religion
## dbl (10): <$10k, $10-20k, $20-30k, $30-40k, $40-50k, $50-75k, $75-100k, $100...
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
cat("
    (2) Printing the tibble will result in the following,  ")
```

```
##
##     (2) Printing the tibble will result in the following,
```

```
pew
```

```
## # A tibble: 18 x 11
##    religion ‘<$10k‘ ‘$10-20k‘ ‘$20-30k‘ ‘$30-40k‘ ‘$40-50k‘ ‘$50-75k‘ ‘$75-100k‘
##    <chr>      <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>      <dbl>
##  1 Agnostic      27        34        60        81        76       137        122
##  2 Atheist       12        27        37        52        35        70         73
##  3 Buddhist      27        21        30        34        33        58         62
##  4 Catholic     418       617       732       670       638      1116        949
##  5 Don’t k~      15        14        15        11        10        35         21
##  6 Evangel~     575       869      1064       982       881      1486        949
##  7 Hindu          1         9         7         9        11        34         47
##  8 Histori~     228       244       236       238       197       223        131
##  9 Jehovah~      20        27        24        24        21        30         15
## 10 Jewish        19        19        25        25        30        95         69
## 11 Mainlin~     289       495       619       655       651      1107        939
## 12 Mormon        29        40        48        51        56       112         85
## 13 Muslim         6         7         9        10         9        23         16
## 14 Orthodox      13        17        23        32        32        47         38
## 15 Other C~       9         7        11        13        13        14         18
## 16 Other F~      20        33        40        46        49        63         46
## 17 Other W~       5         2         3         4         2         7          3
## 18 Unaffil~     217       299       374       365       341       528        407
## # i 3 more variables: ‘$100-150k‘ <dbl>, ‘>150k‘ <dbl>,
## #   ‘Don’t know/refused‘ <dbl>
```

```
cat("
    (3) The datatype of each column in the given data is as follows, ")
```

```
##
##     (3) The datatype of each column in the given data is as follows,
```

```
str(pew)
```

```
## spc_tbl_ [18 x 11] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ religion          : chr [1:18] "Agnostic" "Atheist" "Buddhist" "Catholic" ...
##  $ <$10k             : num [1:18] 27 12 27 418 15 575 1 228 20 19 ...
##  $ $10-20k           : num [1:18] 34 27 21 617 14 869 9 244 27 19 ...
##  $ $20-30k           : num [1:18] 60 37 30 732 15 ...
##  $ $30-40k           : num [1:18] 81 52 34 670 11 982 9 238 24 25 ...
##  $ $40-50k           : num [1:18] 76 35 33 638 10 881 11 197 21 30 ...
##  $ $50-75k           : num [1:18] 137 70 58 1116 35 ...
##  $ $75-100k          : num [1:18] 122 73 62 949 21 949 47 131 15 69 ...
##  $ $100-150k         : num [1:18] 109 59 39 792 17 723 48 81 11 87 ...
##  $ >150k             : num [1:18] 84 74 53 633 18 414 54 78 6 151 ...
##  $ Don’t know/refused: num [1:18] 96 76 54 1489 116 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   religion = col_character(),
##   ..   ‘<$10k‘ = col_double(),
##   ..   ‘$10-20k‘ = col_double(),
##   ..   ‘$20-30k‘ = col_double(),
##   ..   ‘$30-40k‘ = col_double(),
##   ..   ‘$40-50k‘ = col_double(),
##   ..   ‘$50-75k‘ = col_double(),
##   ..   ‘$75-100k‘ = col_double(),
##   ..   ‘$100-150k‘ = col_double(),
##   ..   ‘>150k‘ = col_double(),
##   ..   ‘Don’t know/refused‘ = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

The columns are of the correct datatypes religion=“character”, $=“numeric”.

(4) The given dataset “pew” is not tidy due to the following reasons

(a) Every column is not a varibale - The columns 2 to 11 are all representing the same variable, assuming ‘Income in Thousands of Dollars’ which are not separate variables but values of the same varibale.

(5) The data is not tidy as each column represents values and not variables.To tidy the data we first check for missing values and remove the missing values.

```
#To check for missing values in the data
missing_2<-any(is.na(pew))
if(missing_2)
{
  cat("There are missing values in the data. We first need to eliminate these values. ")
} else{cat("There are no missing values in the data. ")}
```

```
## There are no missing values in the data.
```

```
library(tidyr)
pew<- pew |>
          pivot_longer(c("<$10k","$10-20k","$20-30k","$30-40k","$40-50k","$50-75k","$75-100k","$
cat("The tibble after tidying the data is as below, ")
```

```
## The tibble after tidying the data is as below,
```

```
pew
```

```
## # A tibble: 180 x 3
##    religion ‘Income in $‘      ‘Count of People‘
##    <chr>    <chr>                          <dbl>
##  1 Agnostic <$10k                             27
##  2 Agnostic $10-20k                           34
##  3 Agnostic $20-30k                           60
##  4 Agnostic $30-40k                           81
##  5 Agnostic $40-50k                           76
##  6 Agnostic $50-75k                          137
##  7 Agnostic $75-100k                         122
##  8 Agnostic $100-150k                        109
##  9 Agnostic >150k                             84
## 10 Agnostic Don’t know/refused                96
## # i 170 more rows
```

1c. Import and tidy data stored in `tb.csv`.

Solution: This dataset comes from the World Health Organization, and records the counts of confirmed tuberculosis cases by country, year, and demographic group. The demographic groups are broken down by sex (m, f) and age (0–14, 15–25, 25–34, 35–44, 45–54, 55–64, unknown).

```
library(readr) #Load the library needed to import the csv files
tb_1<-read_csv("tb.csv") #Assigning imported dataset to a variable
```

```
## Rows: 5769 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): iso2
## dbl (21): year, m04, m514, m014, m1524, m2534, m3544, m4554, m5564, m65, mu,...
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
cat("
    (2) Printing the dataset will result in the following,  ")
```

```
##
##     (2) Printing the dataset will result in the following,
```

```
tb_1
```

```
## # A tibble: 5,769 x 22
##    iso2   year   m04  m514  m014 m1524 m2534 m3544 m4554 m5564   m65    mu   f04
##    <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 AD     1989    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
##  2 AD     1990    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
##  3 AD     1991    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
##  4 AD     1992    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
##  5 AD     1993    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
##  6 AD     1994    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA    NA
##  7 AD     1996    NA    NA     0     0     0     4     1     0     0    NA    NA
##  8 AD     1997    NA    NA     0     0     1     2     2     1     6    NA    NA
##  9 AD     1998    NA    NA     0     0     0     1     0     0     0    NA    NA
## 10 AD     1999    NA    NA     0     0     0     1     1     0     0    NA    NA
## # i 5,759 more rows
## # i 9 more variables: f514 <dbl>, f014 <dbl>, f1524 <dbl>, f2534 <dbl>,
## #   f3544 <dbl>, f4554 <dbl>, f5564 <dbl>, f65 <dbl>, fu <dbl>
```

```
cat("
    (3) The datatype of each column in the given dataset is as follows, ")
```

```
##
##     (3) The datatype of each column in the given dataset is as follows,
```

```
str(tb_1) |> print(show_col_types=FALSE)
```

```
## spc_tbl_ [5,769 x 22] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ iso2 : chr [1:5769] "AD" "AD" "AD" "AD" ...
##  $ year : num [1:5769] 1989 1990 1991 1992 1993 ...
##  $ m04  : num [1:5769] NA NA NA NA NA NA NA NA NA NA ...
##  $ m514 : num [1:5769] NA NA NA NA NA NA NA NA NA NA ...
##  $ m014 : num [1:5769] NA NA NA NA NA NA 0 0 0 0 ...
##  $ m1524: num [1:5769] NA NA NA NA NA NA 0 0 0 0 ...
##  $ m2534: num [1:5769] NA NA NA NA NA NA 0 1 0 0 ...
##  $ m3544: num [1:5769] NA NA NA NA NA NA 4 2 1 1 ...
##  $ m4554: num [1:5769] NA NA NA NA NA NA 1 2 0 1 ...
##  $ m5564: num [1:5769] NA NA NA NA NA NA 0 1 0 0 ...
##  $ m65  : num [1:5769] NA NA NA NA NA NA 0 6 0 0 ...
##  $ mu   : num [1:5769] NA NA NA NA NA NA NA NA NA NA ...
##  $ f04  : num [1:5769] NA NA NA NA NA NA NA NA NA NA ...
##  $ f514 : num [1:5769] NA NA NA NA NA NA NA NA NA NA ...
##  $ f014 : num [1:5769] NA NA NA NA NA NA 0 0 NA 0 ...
##  $ f1524: num [1:5769] NA NA NA NA NA NA 1 1 NA 0 ...
##  $ f2534: num [1:5769] NA NA NA NA NA NA 1 2 NA 0 ...
##  $ f3544: num [1:5769] NA NA NA NA NA NA 0 3 NA 1 ...
##  $ f4554: num [1:5769] NA NA NA NA NA NA 0 0 NA 0 ...
##  $ f5564: num [1:5769] NA NA NA NA NA NA 1 0 NA 0 ...
##  $ f65  : num [1:5769] NA NA NA NA NA NA 0 1 NA 0 ...
##  $ fu   : num [1:5769] NA NA NA NA NA NA NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   iso2 = col_character(),
##   ..   year = col_double(),
##   ..   m04 = col_double(),
##   ..   m514 = col_double(),
##   ..   m014 = col_double(),
##   ..   m1524 = col_double(),
##   ..   m2534 = col_double(),
##   ..   m3544 = col_double(),
##   ..   m4554 = col_double(),
##   ..   m5564 = col_double(),
##   ..   m65 = col_double(),
##   ..   mu = col_double(),
##   ..   f04 = col_double(),
##   ..   f514 = col_double(),
##   ..   f014 = col_double(),
##   ..   f1524 = col_double(),
##   ..   f2534 = col_double(),
##   ..   f3544 = col_double(),
##   ..   f4554 = col_double(),
##   ..   f5564 = col_double(),
##   ..   f65 = col_double(),
##   ..   fu = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
## NULL
```

The columns are of the correct datatypes iso2="character", other columns="numeric"

(4) The given dataset "tb" is not tidy due to the following reasons

(a) The data contains a lot of missing values.

(b) Every column is not a variable - Every column header denotes a combination of gender type and the age.

(5)The data is not tidy as each column represents values and not variables.To tidy the data we first check for missing values and remove the missing values.

```
#To check for missing values in the data
missing_3<-any(is.na(tb_1))
if(missing_3)
{
  cat("There are missing values in the data. We first need to eliminate these values. ")
} else{cat("There are no missing values in the data. ")}
```

```
## There are missing values in the data. We first need to eliminate these values.
```

```
tb_2<- tb_1 |>
  pivot_longer(cols=starts_with(c('m','f')),
               names_to= "gender_age",
               values_to="cases",
               values_drop_na = TRUE)
cat("The tibble after initial tidying of the data is as below, ")
```

```
## The tibble after initial tidying of the data is as below,
```

```
tb_2
```

```
## # A tibble: 35,750 x 4
##    iso2   year gender_age cases
##    <chr> <dbl> <chr>      <dbl>
##  1 AD     1996 m014           0
##  2 AD     1996 m1524          0
##  3 AD     1996 m2534          0
##  4 AD     1996 m3544          4
##  5 AD     1996 m4554          1
##  6 AD     1996 m5564          0
##  7 AD     1996 m65            0
##  8 AD     1996 f014           0
##  9 AD     1996 f1524          1
## 10 AD     1996 f2534          1
## # i 35,740 more rows
```

We will now have to separate the column ‘gender_age’ into two columns ‘Gender’ and ‘Age’ using the separate() function and defining the character positions.

```
library(stringr)
tb_3<- tb_2 |>
          separate(gender_age, c("Gender", "Age"), 1)

for (i in 1:nrow(tb_2))
{
  n <- nchar(tb_3$Age[i])
  if (n >= 3) {
  tb_3$Age[i]<-str_replace(tb_3$Age[i],"(.*)(.)(.)$", "\\1-\\2\\3")
  }
  if(n<3)
    {tb_3$Age[i]<-paste(tb_3$Age[i],"+")}
}
  tb_3
```

```
## # A tibble: 35,750 x 5
##    iso2   year Gender Age   cases
##    <chr> <dbl> <chr>  <chr> <dbl>
##  1 AD     1996 m      0-14      0
##  2 AD     1996 m      15-24     0
##  3 AD     1996 m      25-34     0
##  4 AD     1996 m      35-44     4
##  5 AD     1996 m      45-54     1
##  6 AD     1996 m      55-64     0
##  7 AD     1996 m      65 +      0
##  8 AD     1996 f      0-14      0
##  9 AD     1996 f      15-24     1
## 10 AD     1996 f      25-34     1
## # i 35,740 more rows
```

1d. Import and tidy data stored in `weather.csv`.

Solution: Importing the data present in `weather.csv`and tidying it. This dataset shows daily weather data from the Global Historical Climatology Network for one weather station (MX17004) in Mexico for five months in 2010.

```
library(readr) #Load the library needed to import the csv files
weather<-read_csv("weather.csv") #Assigning imported dataset to a variable
```

```
## Rows: 22 Columns: 35
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): id, element
## dbl (25): year, month, d1, d2, d3, d4, d5, d6, d7, d8, d10, d11, d13, d14, d...
## lgl  (8): d9, d12, d18, d19, d20, d21, d22, d24
```

```
##
## i Use ‘spec()‘ to retrieve the full column specification for this data.
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
```

```
cat("
    (2) Printing the dataset will result in the following,  ")
```

```
##
##     (2) Printing the dataset will result in the following,
```

```
weather
```

```
## # A tibble: 22 x 35
##    id       year month element    d1    d2    d3    d4    d5    d6    d7    d8
##    <chr>   <dbl> <dbl> <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 MX17004  2010     1 tmax       NA  NA    NA       NA  NA       NA    NA    NA
##  2 MX17004  2010     1 tmin       NA  NA    NA       NA  NA       NA    NA    NA
##  3 MX17004  2010     2 tmax       NA  27.3  24.1     NA  NA       NA    NA    NA
##  4 MX17004  2010     2 tmin       NA  14.4  14.4     NA  NA       NA    NA    NA
##  5 MX17004  2010     3 tmax       NA  NA    NA       NA  32.1     NA    NA    NA
##  6 MX17004  2010     3 tmin       NA  NA    NA       NA  14.2     NA    NA    NA
##  7 MX17004  2010     4 tmax       NA  NA    NA       NA  NA       NA    NA    NA
##  8 MX17004  2010     4 tmin       NA  NA    NA       NA  NA       NA    NA    NA
##  9 MX17004  2010     5 tmax       NA  NA    NA       NA  NA       NA    NA    NA
## 10 MX17004  2010     5 tmin       NA  NA    NA       NA  NA       NA    NA    NA
## # i 12 more rows
## # i 23 more variables: d9 <lgl>, d10 <dbl>, d11 <dbl>, d12 <lgl>, d13 <dbl>,
## #   d14 <dbl>, d15 <dbl>, d16 <dbl>, d17 <dbl>, d18 <lgl>, d19 <lgl>,
## #   d20 <lgl>, d21 <lgl>, d22 <lgl>, d23 <dbl>, d24 <lgl>, d25 <dbl>,
## #   d26 <dbl>, d27 <dbl>, d28 <dbl>, d29 <dbl>, d30 <dbl>, d31 <dbl>
```

```
cat("
    (3) The datatype of each column in the given dataset is as follows, ")
```

```
##
##     (3) The datatype of each column in the given dataset is as follows,
```

```
str(weather)
```

```
## spc_tbl_ [22 x 35] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ id     : chr [1:22] "MX17004" "MX17004" "MX17004" "MX17004" ...
##  $ year   : num [1:22] 2010 2010 2010 2010 2010 2010 2010 2010 2010 2010 ...
##  $ month  : num [1:22] 1 1 2 2 3 3 4 4 5 5 ...
##  $ element: chr [1:22] "tmax" "tmin" "tmax" "tmin" ...
##  $ d1     : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d2     : num [1:22] NA NA 27.3 14.4 NA NA NA NA NA NA ...
##  $ d3     : num [1:22] NA NA 24.1 14.4 NA NA NA NA NA NA ...
##  $ d4     : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d5     : num [1:22] NA NA NA NA 32.1 14.2 NA NA NA NA ...
##  $ d6     : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d7     : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d8     : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d9     : logi [1:22] NA NA NA NA NA NA ...
##  $ d10    : num [1:22] NA NA NA NA 34.5 16.8 NA NA NA NA ...
##  $ d11    : num [1:22] NA NA 29.7 13.4 NA NA NA NA NA NA ...
##  $ d12    : logi [1:22] NA NA NA NA NA NA ...
##  $ d13    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d14    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d15    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d16    : num [1:22] NA NA NA NA 31.1 17.6 NA NA NA NA ...
##  $ d17    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d18    : logi [1:22] NA NA NA NA NA NA ...
##  $ d19    : logi [1:22] NA NA NA NA NA NA ...
##  $ d20    : logi [1:22] NA NA NA NA NA NA ...
##  $ d21    : logi [1:22] NA NA NA NA NA NA ...
##  $ d22    : logi [1:22] NA NA NA NA NA NA ...
##  $ d23    : num [1:22] NA NA 29.9 10.7 NA NA NA NA NA NA ...
##  $ d24    : logi [1:22] NA NA NA NA NA NA ...
##  $ d25    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d26    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d27    : num [1:22] NA NA NA NA NA NA 36.3 16.7 33.2 18.2 ...
##  $ d28    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d29    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  $ d30    : num [1:22] 27.8 14.5 NA NA NA NA NA NA NA NA ...
##  $ d31    : num [1:22] NA NA NA NA NA NA NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   id = col_character(),
##   ..   year = col_double(),
##   ..   month = col_double(),
##   ..   element = col_character(),
##   ..   d1 = col_double(),
##   ..   d2 = col_double(),
##   ..   d3 = col_double(),
##   ..   d4 = col_double(),
##   ..   d5 = col_double(),
##   ..   d6 = col_double(),
##   ..   d7 = col_double(),
##   ..   d8 = col_double(),
##   ..   d9 = col_logical(),
##   ..   d10 = col_double(),
##   ..   d11 = col_double(),
##   ..   d12 = col_logical(),
##   ..   d13 = col_double(),
##   ..   d14 = col_double(),
##   ..   d15 = col_double(),
##   ..   d16 = col_double(),
##   ..   d17 = col_double(),
##   ..   d18 = col_logical(),
##   ..   d19 = col_logical(),
##   ..   d20 = col_logical(),
##   ..   d21 = col_logical(),
##   ..   d22 = col_logical(),
##   ..   d23 = col_double(),
##   ..   d24 = col_logical(),
##   ..   d25 = col_double(),
##   ..   d26 = col_double(),
##   ..   d27 = col_double(),
##   ..   d28 = col_double(),
##   ..   d29 = col_double(),
##   ..   d30 = col_double(),
##   ..   d31 = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

All the columns are of the correct datatypes. The columns d9, d12, d18, d19, d20, d21, d22, d24 are of the Logical datatype as they constitute no data except for null values.

(4) The given dataset "weather" is not tidy due to the following reasons

(a) Every column is not a variable - The columns such as d1, d2, d3,. . . .,d31 are not separate variables but values of the same variable 'date of the month'.
(b) The data contains a lot of missing values(NA) which consume space that can be avoided.

(5)The data is not tidy as each column represents values and not variables.To tidy the data we first check for missing values and remove the missing values.

```
#To check for missing values in the data
missing_4<-any(is.na(weather))
if(missing_4)
{
  cat("There are missing values in the data. We first need to eliminate these values. ")
} else{cat("There are no missing values in the data. ")}
```

```
## There are missing values in the data. We first need to eliminate these values.
```

```
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
weather_1<- weather |>
          pivot_longer(!c("id","year","month","element"),names_to="date", values_to="Values", va
weather_1$month<-as.character(weather_1$month) #converting month to character
weather_1$year<-as.character(weather_1$year) #converting year to character
weather_1$date<-substr(weather_1$date,2,3) #substringing the date to pick only the numbers
weather_2<- weather_1  |>
  mutate(Date=paste(year,"-",month,"-",date)) #Adding a new column Date
weather_3<-weather_2 |>pivot_wider(names_from = element, values_from = Values)
weather_4<-select(weather_3,-c(2,3,4))
cat("The tibble after tidying the data is as below, ")
```

```
## The tibble after tidying the data is as below,
```

```
weather_4
```

```
## # A tibble: 33 x 4
##    id      Date            tmax  tmin
##    <chr>   <chr>          <dbl> <dbl>
##  1 MX17004 2010 - 1 - 30   27.8  14.5
##  2 MX17004 2010 - 2 - 2    27.3  14.4
##  3 MX17004 2010 - 2 - 3    24.1  14.4
##  4 MX17004 2010 - 2 - 11   29.7  13.4
##  5 MX17004 2010 - 2 - 23   29.9  10.7
##  6 MX17004 2010 - 3 - 5    32.1  14.2
##  7 MX17004 2010 - 3 - 10   34.5  16.8
##  8 MX17004 2010 - 3 - 16   31.1  17.6
##  9 MX17004 2010 - 4 - 27   36.3  16.7
## 10 MX17004 2010 - 5 - 27   33.2  18.2
## # i 23 more rows
```

1e. Import and tidy data stored in `billboard.csv`.

Solution:

```
library(readr) #Load the library needed to import the csv files
billboard<-read_csv("billboard.csv") #Assigning imported dataset to a variable
```

```
## Rows: 317 Columns: 81
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr   (2): artist, track
## dbl  (66): year, wk1, wk2, wk3, wk4, wk5, wk6, wk7, wk8, wk9, wk10, wk11, wk...
## lgl  (11): wk66, wk67, wk68, wk69, wk70, wk71, wk72, wk73, wk74, wk75, wk76
## date  (1): date.entered
## time  (1): time
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
cat("
    (2) Printing the dataset will result in the following,  ")
```

```
##
##     (2) Printing the dataset will result in the following,
```

```
billboard
```

```
## # A tibble: 317 x 81
##     year artist     track time  date.entered   wk1   wk2   wk3   wk4   wk5   wk6
##    <dbl> <chr>      <chr> <tim> <date>       <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1  2000 2 Pac      Baby~ 04:22 2000-02-26      87    82    72    77    87    94
##  2  2000 2Ge+her    The ~ 03:15 2000-09-02      91    87    92    NA    NA    NA
##  3  2000 3 Doors D~ Kryp~ 03:53 2000-04-08      81    70    68    67    66    57
##  4  2000 3 Doors D~ Loser 04:24 2000-10-21      76    76    72    69    67    65
##  5  2000 504 Boyz   Wobb~ 03:35 2000-04-15      57    34    25    17    17    31
##  6  2000 98^0       Give~ 03:24 2000-08-19      51    39    34    26    26    19
##  7  2000 A*Teens    Danc~ 03:44 2000-07-08      97    97    96    95   100    NA
##  8  2000 Aaliyah    I Do~ 04:15 2000-01-29      84    62    51    41    38    35
##  9  2000 Aaliyah    Try ~ 04:03 2000-03-18      59    53    38    28    21    18
## 10  2000 Adams, Yo~ Open~ 05:30 2000-08-26      76    76    74    69    68    67
## # i 307 more rows
## # i 70 more variables: wk7 <dbl>, wk8 <dbl>, wk9 <dbl>, wk10 <dbl>, wk11 <dbl>,
## #   wk12 <dbl>, wk13 <dbl>, wk14 <dbl>, wk15 <dbl>, wk16 <dbl>, wk17 <dbl>,
## #   wk18 <dbl>, wk19 <dbl>, wk20 <dbl>, wk21 <dbl>, wk22 <dbl>, wk23 <dbl>,
## #   wk24 <dbl>, wk25 <dbl>, wk26 <dbl>, wk27 <dbl>, wk28 <dbl>, wk29 <dbl>,
## #   wk30 <dbl>, wk31 <dbl>, wk32 <dbl>, wk33 <dbl>, wk34 <dbl>, wk35 <dbl>,
## #   wk36 <dbl>, wk37 <dbl>, wk38 <dbl>, wk39 <dbl>, wk40 <dbl>, wk41 <dbl>, ...
```

```
cat("
    (3) The datatype of each column in the given dataset is as follows, \n")
```

```
##
##     (3) The datatype of each column in the given dataset is as follows,
```

```
str(billboard)
```

```
## spc_tbl_ [317 x 81] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ year        : num [1:317] 2000 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
##  $ artist      : chr [1:317] "2 Pac" "2Ge+her" "3 Doors Down" "3 Doors Down" ...
##  $ track       : chr [1:317] "Baby Don’t Cry (Keep..." "The Hardest Part Of ..." "Kryptonite"
##  $ time        : ’hms’ num [1:317] 04:22:00 03:15:00 03:53:00 04:24:00 ...
##   ..- attr(*, "units")= chr "secs"
##  $ date.entered: Date[1:317], format: "2000-02-26" "2000-09-02" ...
##  $ wk1         : num [1:317] 87 91 81 76 57 51 97 84 59 76 ...
##  $ wk2         : num [1:317] 82 87 70 76 34 39 97 62 53 76 ...
##  $ wk3         : num [1:317] 72 92 68 72 25 34 96 51 38 74 ...
##  $ wk4         : num [1:317] 77 NA 67 69 17 26 95 41 28 69 ...
##  $ wk5         : num [1:317] 87 NA 66 67 17 26 100 38 21 68 ...
##  $ wk6         : num [1:317] 94 NA 57 65 31 19 NA 35 18 67 ...
##  $ wk7         : num [1:317] 99 NA 54 55 36 2 NA 35 16 61 ...
##  $ wk8         : num [1:317] NA NA 53 59 49 2 NA 38 14 58 ...
##  $ wk9         : num [1:317] NA NA 51 62 53 3 NA 38 12 57 ...
##  $ wk10        : num [1:317] NA NA 51 61 57 6 NA 36 10 59 ...
##  $ wk11        : num [1:317] NA NA 51 61 64 7 NA 37 9 66 ...
##  $ wk12        : num [1:317] NA NA 51 59 70 22 NA 37 8 68 ...
##  $ wk13        : num [1:317] NA NA 47 61 75 29 NA 38 6 61 ...
##  $ wk14        : num [1:317] NA NA 44 66 76 36 NA 49 1 67 ...
##  $ wk15        : num [1:317] NA NA 38 72 78 47 NA 61 2 59 ...
##  $ wk16        : num [1:317] NA NA 28 76 85 67 NA 63 2 63 ...
##  $ wk17        : num [1:317] NA NA 22 75 92 66 NA 62 2 67 ...
##  $ wk18        : num [1:317] NA NA 18 67 96 84 NA 67 2 71 ...
##  $ wk19        : num [1:317] NA NA 18 73 NA 93 NA 83 3 79 ...
##  $ wk20        : num [1:317] NA NA 14 70 NA 94 NA 86 4 89 ...
##  $ wk21        : num [1:317] NA NA 12 NA NA NA NA NA 5 NA ...
##  $ wk22        : num [1:317] NA NA 7 NA NA NA NA NA 5 NA ...
##  $ wk23        : num [1:317] NA NA 6 NA NA NA NA NA 6 NA ...
##  $ wk24        : num [1:317] NA NA 6 NA NA NA NA NA 9 NA ...
##  $ wk25        : num [1:317] NA NA 6 NA NA NA NA NA 13 NA ...
##  $ wk26        : num [1:317] NA NA 5 NA NA NA NA NA 14 NA ...
##  $ wk27        : num [1:317] NA NA 5 NA NA NA NA NA 16 NA ...
##  $ wk28        : num [1:317] NA NA 4 NA NA NA NA NA 23 NA ...
##  $ wk29        : num [1:317] NA NA 4 NA NA NA NA NA 22 NA ...
##  $ wk30        : num [1:317] NA NA 4 NA NA NA NA NA 33 NA ...
##  $ wk31        : num [1:317] NA NA 4 NA NA NA NA NA 36 NA ...
##  $ wk32        : num [1:317] NA NA 3 NA NA NA NA NA 43 NA ...
##  $ wk33        : num [1:317] NA NA 3 NA NA NA NA NA NA NA ...
##  $ wk34        : num [1:317] NA NA 3 NA NA NA NA NA NA NA ...
##  $ wk35        : num [1:317] NA NA 4 NA NA NA NA NA NA NA ...
##  $ wk36        : num [1:317] NA NA 5 NA NA NA NA NA NA NA ...
##  $ wk37        : num [1:317] NA NA 5 NA NA NA NA NA NA NA ...
##  $ wk38        : num [1:317] NA NA 9 NA NA NA NA NA NA NA ...
##  $ wk39        : num [1:317] NA NA 9 NA NA NA NA NA NA NA ...
##  $ wk40        : num [1:317] NA NA 15 NA NA NA NA NA NA NA ...
##  $ wk41        : num [1:317] NA NA 14 NA NA NA NA NA NA NA ...
##  $ wk42        : num [1:317] NA NA 13 NA NA NA NA NA NA NA ...
##  $ wk43        : num [1:317] NA NA 14 NA NA NA NA NA NA NA ...
##  $ wk44        : num [1:317] NA NA 16 NA NA NA NA NA NA NA ...
##  $ wk45        : num [1:317] NA NA 17 NA NA NA NA NA NA NA ...
##  $ wk46        : num [1:317] NA NA 21 NA NA NA NA NA NA NA ...
##  $ wk47        : num [1:317] NA NA 22 NA NA NA NA NA NA NA ...
##  $ wk48        : num [1:317] NA NA 24 NA NA NA NA NA NA NA ...
##  $ wk49        : num [1:317] NA NA 28 NA NA NA NA NA NA NA ...
##  $ wk50        : num [1:317] NA NA 33 NA NA NA NA NA NA NA ...
##  $ wk51        : num [1:317] NA NA 42 NA NA NA NA NA NA NA ...
##  $ wk52        : num [1:317] NA NA 42 NA NA NA NA NA NA NA ...
##  $ wk53        : num [1:317] NA NA 49 NA NA NA NA NA NA NA ...
##  $ wk54        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk55        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk56        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk57        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk58        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk59        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk60        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk61        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk62        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk63        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk64        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk65        : num [1:317] NA NA NA NA NA NA NA NA NA NA ...
##  $ wk66        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk67        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk68        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk69        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk70        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk71        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk72        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk73        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk74        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk75        : logi [1:317] NA NA NA NA NA NA ...
##  $ wk76        : logi [1:317] NA NA NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   year = col_double(),
##   ..   artist = col_character(),
##   ..   track = col_character(),
##   ..   time = col_time(format = ""),
##   ..   date.entered = col_date(format = ""),
##   ..   wk1 = col_double(),
##   ..   wk2 = col_double(),
##   ..   wk3 = col_double(),
##   ..   wk4 = col_double(),
##   ..   wk5 = col_double(),
##   ..   wk6 = col_double(),
##   ..   wk7 = col_double(),
##   ..   wk8 = col_double(),
##   ..   wk9 = col_double(),
##   ..   wk10 = col_double(),
##   ..   wk11 = col_double(),
##   ..   wk12 = col_double(),
##   ..   wk13 = col_double(),
##   ..   wk14 = col_double(),
##   ..   wk15 = col_double(),
##   ..   wk16 = col_double(),
##   ..   wk17 = col_double(),
##   ..   wk18 = col_double(),
##   ..   wk19 = col_double(),
##   ..   wk20 = col_double(),
##   ..   wk21 = col_double(),
##   ..   wk22 = col_double(),
##   ..   wk23 = col_double(),
##   ..   wk24 = col_double(),
##   ..   wk25 = col_double(),
##   ..   wk26 = col_double(),
##   ..   wk27 = col_double(),
##   ..   wk28 = col_double(),
##   ..   wk29 = col_double(),
##   ..   wk30 = col_double(),
##   ..   wk31 = col_double(),
##   ..   wk32 = col_double(),
##   ..   wk33 = col_double(),
##   ..   wk34 = col_double(),
##   ..   wk35 = col_double(),
##   ..   wk36 = col_double(),
##   ..   wk37 = col_double(),
##   ..   wk38 = col_double(),
##   ..   wk39 = col_double(),
##   ..   wk40 = col_double(),
##   ..   wk41 = col_double(),
##   ..   wk42 = col_double(),
##   ..   wk43 = col_double(),
##   ..   wk44 = col_double(),
##   ..   wk45 = col_double(),
##   ..   wk46 = col_double(),
##   ..   wk47 = col_double(),
##   ..   wk48 = col_double(),
##   ..   wk49 = col_double(),
##   ..   wk50 = col_double(),
##   ..   wk51 = col_double(),
##   ..   wk52 = col_double(),
##   ..   wk53 = col_double(),
##   ..   wk54 = col_double(),
##   ..   wk55 = col_double(),
##   ..   wk56 = col_double(),
##   ..   wk57 = col_double(),
##   ..   wk58 = col_double(),
##   ..   wk59 = col_double(),
##   ..   wk60 = col_double(),
##   ..   wk61 = col_double(),
##   ..   wk62 = col_double(),
##   ..   wk63 = col_double(),
##   ..   wk64 = col_double(),
##   ..   wk65 = col_double(),
##   ..   wk66 = col_logical(),
##   ..   wk67 = col_logical(),
##   ..   wk68 = col_logical(),
##   ..   wk69 = col_logical(),
##   ..   wk70 = col_logical(),
##   ..   wk71 = col_logical(),
##   ..   wk72 = col_logical(),
##   ..   wk73 = col_logical(),
##   ..   wk74 = col_logical(),
##   ..   wk75 = col_logical(),
##   ..   wk76 = col_logical()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

All the columns are of the correct datatypes. The columns wk66 to wk76 are of the Logical datatype as they constitute no data except for null values.

(4) The given dataset "billboard" is not tidy due to the following reasons

(a) Every column is not a variable - The columns such as wk1-wk76 are not separate variables but values of the same variable 'week'.
(b) The data contains a lot of missing values(NA) which consume space that can be avoided.

(5)The data is not tidy as each column represents values and not variables.To tidy the data we first check for missing values and remove the missing values.

```
#To check for missing values in the data
missing_5<-any(is.na(billboard))
if(missing_5)
{
  cat("There are missing values in the data. We first need to eliminate these values. ")
} else{cat("There are no missing values in the data. ")}
```

```
## There are missing values in the data. We first need to eliminate these values.
```

Approach: As this data has lots of data repetition, we split the data into two tables `artist_table` and `rank_table`. - To do this, we start by transforming the data using pivot_longer() function - Next, we select certain columns from the table and create an artist_table with unique details of the artist. - After that, we select and rename the columns 'artist', 'values', 'date' to create another table `new_rank_table` which contains additional information on the billboard ranks. - Both the tables can be linked back using the 'artist' column as the identifier.

```
library(tidyr)
library(dplyr)
billboard_1<- billboard |>
          pivot_longer(cols=starts_with(c('wk')),names_to="week", values_to="values",values_drop
cat("The billbaord tibble after tidying the data is as below, ")
```

```
## The billbaord tibble after tidying the data is as below,
```

```
n=n_distinct(billboard_1$artist)
billboard_1
```

```
## # A tibble: 5,307 x 7
##     year artist  track                   time   date.entered week  values
##    <dbl> <chr>   <chr>                   <time> <date>       <chr>  <dbl>
##  1  2000 2 Pac   Baby Don't Cry (Keep... 04:22  2000-02-26   wk1       87
##  2  2000 2 Pac   Baby Don’t Cry (Keep... 04:22  2000-02-26   wk2       82
##  3  2000 2 Pac   Baby Don’t Cry (Keep... 04:22  2000-02-26   wk3       72
##  4  2000 2 Pac   Baby Don’t Cry (Keep... 04:22  2000-02-26   wk4       77
##  5  2000 2 Pac   Baby Don’t Cry (Keep... 04:22  2000-02-26   wk5       87
##  6  2000 2 Pac   Baby Don’t Cry (Keep... 04:22  2000-02-26   wk6       94
##  7  2000 2 Pac   Baby Don’t Cry (Keep... 04:22  2000-02-26   wk7       99
##  8  2000 2Ge+her The Hardest Part Of ... 03:15  2000-09-02   wk1       91
##  9  2000 2Ge+her The Hardest Part Of ... 03:15  2000-09-02   wk2       87
## 10  2000 2Ge+her The Hardest Part Of ... 03:15  2000-09-02   wk3       92
## # i 5,297 more rows
```

```
artist_table<-billboard_1 |>select(artist, track, time)
artist_table<-distinct(artist_table)
#artist_table_1<- artist_table |>
        #mutate(id=seq(1:n_distinct(artist_table_1)))
#Printing the final artist table
cat("The artist tibble is as below, ")
```

```
## The artist tibble is as below,
```

```
artist_table
```

```
## # A tibble: 317 x 3
##    artist         track                   time
##    <chr>          <chr>                   <time>
##  1 2 Pac          Baby Don’t Cry (Keep... 04:22
##  2 2Ge+her        The Hardest Part Of ... 03:15
##  3 3 Doors Down   Kryptonite              03:53
##  4 3 Doors Down   Loser                   04:24
##  5 504 Boyz       Wobble Wobble           03:35
##  6 98^0           Give Me Just One Nig... 03:24
##  7 A*Teens        Dancing Queen           03:44
##  8 Aaliyah        I Don’t Wanna           04:15
##  9 Aaliyah        Try Again               04:03
## 10 Adams, Yolanda Open My Heart           05:30
## # i 307 more rows
```

```
#Printing the final rank table
new_rank_table<-billboard_1 |> select(artist,"date"=date.entered, "Rank"=values)
cat("The rank tibble is as below, ")
```

```
## The rank tibble is as below,
new_rank_table
```

```
## # A tibble: 5,307 x 3
##    artist  date        Rank
##    <chr>   <date>     <dbl>
##  1 2 Pac   2000-02-26    87
##  2 2 Pac   2000-02-26    82
##  3 2 Pac   2000-02-26    72
##  4 2 Pac   2000-02-26    77
##  5 2 Pac   2000-02-26    87
##  6 2 Pac   2000-02-26    94
##  7 2 Pac   2000-02-26    99
##  8 2Ge+her 2000-09-02    91
##  9 2Ge+her 2000-09-02    87
## 10 2Ge+her 2000-09-02    92
## # i 5,297 more rows
```

String processing

One common string processing task is to extract numbers from strings and convert them into appropriate data type for data transformation and visualization.

The code below obtains from the Wikipedia page (https://en.wikipedia.org/wiki/List_of_cities_by_murder_rate) raw data of the highest murder rates of all cities with a population of at least 300,000 people:

```
library(rvest)  # Part of the tidyverse
url <- "https://en.wikipedia.org/wiki/List_of_cities_by_murder_rate"
murders_city <- read_html(url) |>
    html_nodes("table") |>
    html_table() |>
    {\(x) x[[2]]}() |>
    setNames(c("rank", "city", "country", "homicide", "population", "rate"))
```

Before any analysis it is always a good idea to read the data description and check how the data are represented in R:

```
str(murders_city)
```

```
## tibble [50 x 6] (S3: tbl_df/tbl/data.frame)
##  $ rank      : int [1:50] 1 2 3 4 5 6 7 8 9 10 ...
##  $ city      : chr [1:50] "Colima" "Zamora" "Ciudad Obregón" "Zacatecas" ...
##  $ country   : chr [1:50] "Mexico" "Mexico" "Mexico" "Mexico" ...
##  $ homicide  : chr [1:50] "601" "552" "454" "490" ...
##  $ population: chr [1:50] "330,329" "310,575" "328,430" "363,996" ...
##  $ rate      : num [1:50] 182 178 138 135 105 ...
```

The values of `homicide` and `population` are stored as character. This is not ideal, and we would like to convert the values to integer, e.g.,

```
murders_city$homicide[1:3]
```

```
## [1] "601" "552" "454"
```

```
as.integer(murders_city$homicide[1:3])
```

```
## [1] 601 552 454
```

Not quite as we expected! The comma `,` is making trouble.

```
as.integer(murders_city$homicide[1])  # NA
as.integer(murders_city$homicide[3])  # works!
```

2a. Convert the values in `homicide` and `population` to integer. In order to do that, you need to first remove all the commas.

Solution: We first need to remove the commas in the values of `homicide` and `population`. To do this, we utilize the functions in `stringr` package.

```
library(stringr)
#Removing , in homicide
murders_city2<-murders_city
murders_city2$homicide<- str_replace(murders_city$homicide, ",", "")

#Removing , in population
murders_city2$population<- str_replace(murders_city$population, ",", "")
murders_city2
```

```
## # A tibble: 50 x 6
##     rank city            country       homicide population  rate
##    <int> <chr>           <chr>         <chr>    <chr>      <dbl>
##  1     1 Colima          Mexico        601      330329      182.
##  2     2 Zamora          Mexico        552      310575      178.
##  3     3 Ciudad Obregón Mexico        454      328430      138.
##  4     4 Zacatecas       Mexico        490      363996      135.
##  5     5 Tijuana         Mexico        2177     2070,875    105.
## 6      6 Celaya         Mexico        740      742662      99.6
## 7      7 Uruapan        Mexico        282      360338      78.3
## 8      8 New Orleans    United States 266      376971      70.6
## 9      9 Ciudad Juárez  Mexico        1034     1527,482    67.7
## 10    10 Acapulco       Mexico        513      782661      65.6
## # i 40 more rows
```

We now convert the values to integers using the `as.integer()` function.

```
murders_city3<-murders_city2
murders_city3$homicide<- as.integer(murders_city3$homicide)
murders_city3$population<- as.integer(murders_city3$population)
```

```
## Warning: NAs introduced by coercion
```

We have finally converted the values in `homicide` and `population` to integers.

```
cat("The type of homicide variable is: ",typeof(murders_city3$homicide ))
```

```
## The type of homicide variable is:  integer
```

```
cat("\nThe type of population variable is: ",typeof(murders_city3$population ))
```

```
##
## The type of population variable is:  integer
```

```
murders_city3
```

```
## # A tibble: 50 x 6
##     rank city           country       homicide population  rate
##    <int> <chr>          <chr>            <int>      <int> <dbl>
## 1      1 Colima         Mexico             601     330329 182.
## 2      2 Zamora         Mexico             552     310575 178.
## 3      3 Ciudad Obregón Mexico             454     328430 138.
## 4      4 Zacatecas      Mexico             490     363996 135.
## 5      5 Tijuana        Mexico            2177         NA 105.
## 6      6 Celaya         Mexico             740     742662  99.6
## 7      7 Uruapan        Mexico             282     360338  78.3
## 8      8 New Orleans    United States      266     376971  70.6
## 9      9 Ciudad Juárez  Mexico            1034         NA  67.7
## 10    10 Acapulco       Mexico             513     782661  65.6
## # i 40 more rows
```

2b. Use `homicide` and `population` to create a new variable `rate2` in `murders_city` for the number of homicides per 100,000. Confirm if your calculated `rate2` is equal to `rate`.

Solution: From the dataset we can infer that the number of homicides in a city are computed for the entire population of the city. We now need to calculate the homicides per 100,000 of the population.

```
#new variable rate2
rate2<- (murders_city3$homicide / murders_city3$population) * 100000
rate2<-round(rate2,2)
count_1<-sum(rate2!=murders_city$rate)
if(all(rate2==murders_city$rate))
  {
    cat("All the rates are equal")
  }else
  {
    cat("The calculated rate2 is not equal to the original rate for ", count_1  ,"cities")
    cat("\nThe data for these cities is, ")

  count_1
  print(murders_city|> filter(!(rate2==murders_city$rate)))
  }
```

```
## The calculated rate2 is not equal to the original rate for  NA cities
## The data for these cities is, # A tibble: 1 x 6
##    rank city         country  homicide population  rate
##   <int> <chr>        <chr>    <chr>    <chr>      <dbl>
## 1    43 Buenaventura Colombia 157      315,743     35.2
```

Working with factors

We will investigate the Gapminder data in the following problems. A great example of data exploration and visualization with this dataset can be found at https://www.youtube.com/watch?v=BPt8ElTQMIg

```
library(gapminder)
str(gapminder)
```

```
## tibble [1,704 x 6] (S3: tbl_df/tbl/data.frame)
##  $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
##  $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867
##  $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

The variable `continent` is represented as a factor with 5 levels:

```
levels(gapminder$continent)
```

```
## [1] "Africa"   "Americas" "Asia"     "Europe"   "Oceania"
```

The code below summarizes the number of observations for each continent, ordered alphabetically.

```
gapminder |>
    count(continent)
```

```
## # A tibble: 5 x 2
##   continent     n
##   <fct>     <int>
## 1 Africa      624
## 2 Americas    300
## 3 Asia        396
## 4 Europe      360
## 5 Oceania      24
```

3a. Modify the code so that in the summary the continent is ordered by frequency (low to high)

Solution: To order the above result by frequency we use the arrange() function in the `dplyr` package.

```
library(dplyr)
gapminder_2<- gapminder |>
    count(continent) |>arrange(n)
print(gapminder_2)
```

```
## # A tibble: 5 x 2
##   continent     n
##   <fct>     <int>
## 1 Oceania      24
## 2 Americas    300
## 3 Europe      360
## 4 Asia        396
## 5 Africa      624
```

3b. Take a subset of the data for "Australia", "New Zealand", "United Kingdom", and "United States" after Year 2000. Recode the factor levels of country to "Oz", "NZ", "UK", and "US".

Solution:

```
gap_2<-gapminder |> filter(year>2000 & country %in% c("Australia", "New Zealand","United Kingdom
gap_3<-gap_2|>
      mutate(country=factor(country, levels=c("Australia", "New Zealand","United Kingdom","Unite
```

The data after factoring is as follows,

```
gap_3
```

```
## # A tibble: 8 x 6
##   country continent  year lifeExp       pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>     <int>     <dbl>
## 1 Oz      Oceania    2002    80.4  19546792    30688.
## 2 Oz      Oceania    2007    81.2  20434176    34435.
## 3 NZ      Oceania    2002    79.1   3908037    23190.
## 4 NZ      Oceania    2007    80.2   4115771    25185.
## 5 UK      Europe     2002    78.5  59912431    29479.
## 6 UK      Europe     2007    79.4  60776238    33203.
## 7 US      Americas   2002    77.3 287675526    39097.
## 8 US      Americas   2007    78.2 301139947    42952.
```
