Data Wrangling (Data Preprocessing)

Code

Practical assessment 1

Hardik Ramparia

19/08/2024

Setup

Hide

```
# Load the necessary packages required to reproduce the report.
library(kableExtra)
library(magrittr)
```

Student names, numbers and percentage of contributions

Group information

Student name	Student number	Percentage of contribution
Hardik Ramparia	S4105620	51
Shagun Dixit	S4087905	49

Data Description

Provide explanations here. You may use bulleted lists like this:

- The data used for this report is of the recorded rainfall in different regions in Australia.
- The features of the data are Station number, Station name, Region, Rainfall, Latitude, Longitude, Elevation.
- The station number denotes a unique value given to each station for identification purposes.
- The Station name records the names of the stations across Australia.
- The Region column denotes the states across Australia that the rainfall is recorded in.
- The Rainfall column records the values in mm for the total rainfall recorded in a weekly time frame.
- The station number denotes a unique value given to each station for identification purposes.
- The Latitude and Longitude columns denote the location of the station in Australia.
- The Elevation column records the elevation of the station above sea level in meter.

Read/Import Data

Hide

```
# loading the readr package
library(readr)
# creating a tibble by reading the data frame
data = read_csv("rainfall.2024-08-13.weekly.aus.csv")
```

```
Rows: 2045 Columns: 7
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (2): Station name, Region
dbl (5): Station number, Rainfall (mm), Latitude (°), Longitude (°), Elevation (m)

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.
```

Hide

```
# converting the tibble to a data frame
data <- as.data.frame(data)
# showing the first few rows of data
head(data)
```

	Station number	Station name	Reg…	Rainfall (mm)	Latitud
	<dbl>	<chr>	<chr>	<dbl>	<
1	58212	EVANS HEAD RAAF BOMBING RANGE AWS	NSW	234.2	-2
2	58162	NASHUA (WILSONS RIVER)	NSW	223.0	-2
3	58198	BALLINA AIRPORT AWS	NSW	218.8	-2
4	58012	YAMBA PILOT STATION	NSW	206.4	-2
5	58206	CORNDALE (COOPERS CREEK)	NSW	186.0	-2
6	58023	MCLEANS RIDGES (LASCOTT DRIVE)	NSW	182.2	-2

6 rows | 1-6 of 7 columns

Explanations:

- We use the library() function to load the required package.
- After reading the package, we use the readr function to load the .csv file.
- Since the .csv file is stored locally in the same directory as the .rmd file, we can directly load the file by specifying its name.
- When the data is read using read_csv(), it gets read as a tibble. Hence to convert it to a data frame, we use as.data.frame().
- We use the head() function to view the first few rows of the data frame.

Inspect and Understand

Hide

```
# printing the structure of the data
str(data)
```

```
'data.frame':   2045 obs. of  7 variables:
 $ Station number: num  58212 58162 58198 58012 58206 ...
 $ Station name  : chr  "EVANS HEAD RAAF BOMBING RANGE AWS" "NASHUA (WILSONS RIVER)" "BALLINA
AIRPORT AWS" "YAMBA PILOT STATION" ...
 $ Region        : chr  "NSW" "NSW" "NSW" "NSW" ...
 $ Rainfall (mm) : num  234 223 219 206 186 ...
 $ Latitude (°)  : num  -29.2 -28.7 -28.8 -29.4 -28.7 ...
 $ Longitude (°) : num  153 153 154 153 153 ...
 $ Elevation (m) : num  63 30 1 27 25 120 75 15 9 3 ...
```

Hide

```
# printing the column names of the data
names(data)
```

```
[1] "Station number" "Station name"   "Region"         "Rainfall (mm)"  "Latitude (°)"
[6] "Longitude (°)"  "Elevation (m)"
```

Hide

```
# renaming the column names
names(data)[names(data) == "Latitude (°)"] <- "Latitude"
names(data)[names(data) == "Longitude (°)"] <- "Longitude"
# checking the data types of the features
sapply(data, class)
```

```
Station number   Station name         Region  Rainfall (mm)       Latitude      Longitude
     "numeric"    "character"    "character"      "numeric"      "numeric"      "numeric"
 Elevation (m)
     "numeric"
```

Explanations:

- To view the structure of the data frame, we use the str() function.
- To view the column names of the data frame, we use the names() function.
- Renaming the columns can be done by subsetting the required name from the names() function and then renaming it to the new name.
- We can check the data types of the variables of the data frame by using the sapply() function. Since all the data types are correct, we do not need to change them.
- Since we do not have any factor variables, there is no need to check for levels.

Subsetting

Hide

```
# creating a subset of the data
data_subset <- data[1:10, ]
data_subset
```

	Station number	Station name	Reg…	Rainfall (mm)	Latitude
	<dbl>	<chr>	<chr>	<dbl>	<dbl>
1	58212	EVANS HEAD RAAF BOMBING RANGE AWS	NSW	234.2	-29.18
2	58162	NASHUA (WILSONS RIVER)	NSW	223.0	-28.73
3	58198	BALLINA AIRPORT AWS	NSW	218.8	-28.84
4	58012	YAMBA PILOT STATION	NSW	206.4	-29.43
5	58206	CORNDALE (COOPERS CREEK)	NSW	186.0	-28.72
6	58023	MCLEANS RIDGES (LASCOTT DRIVE)	NSW	182.2	-28.79
7	58070	ROSEBANK (REPENTANCE CREEK)	NSW	161.2	-28.64
8	58040	MULLUMBIMBY (FAIRVIEW FARM)	NSW	160.8	-28.55
9	200283	WILLIS ISLAND	QLD	135.0	-16.29
10	58007	BYRON BAY (JACARANDA DRIVE)	NSW	130.6	-28.64

1-10 of 10 rows | 1-7 of 7 columns

Hide

```
# creating a matrix of the subset
data_subset_matrix <- as.matrix(data_subset)
data_subset_matrix
```

```
   Station number Station name                          Region Rainfall (mm) Latitude Longitude
1   " 58212"      "EVANS HEAD RAAF BOMBING RANGE AWS" "NSW"  "234.2"       "-29.18" "153.40"
2   " 58162"      "NASHUA (WILSONS RIVER)"            "NSW"  "223.0"       "-28.73" "153.46"
3   " 58198"      "BALLINA AIRPORT AWS"               "NSW"  "218.8"       "-28.84" "153.56"
4   " 58012"      "YAMBA PILOT STATION"               "NSW"  "206.4"       "-29.43" "153.36"
5   " 58206"      "CORNDALE (COOPERS CREEK)"          "NSW"  "186.0"       "-28.72" "153.36"
6   " 58023"      "MCLEANS RIDGES (LASCOTT DRIVE)"    "NSW"  "182.2"       "-28.79" "153.40"
7   " 58070"      "ROSEBANK (REPENTANCE CREEK)"       "NSW"  "161.2"       "-28.64" "153.41"
8   " 58040"      "MULLUMBIMBY (FAIRVIEW FARM)"       "NSW"  "160.8"       "-28.55" "153.49"
9   "200283"      "WILLIS ISLAND"                     "QLD"  "135.0"       "-16.29" "149.97"
10  " 58007"      "BYRON BAY (JACARANDA DRIVE)"       "NSW"  "130.6"       "-28.64" "153.59"
   Elevation (m)
1   " 63"
2   " 30"
3   "  1"
4   " 27"
5   " 25"
6   "120"
7   " 75"
8   " 15"
9   "  9"
10  "  3"
```

Hide

```
# printing the structure of the data
str(data_subset_matrix)
```

```
 chr [1:10, 1:7] " 58212" " 58162" " 58198" " 58012" " 58206" " 58023" " 58070" " 58040" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:10] "1" "2" "3" "4" ...
  ..$ : chr [1:7] "Station number" "Station name" "Region" "Rainfall (mm)" ...
```

Explanations:

- Subsetting the data can be done by using the [] characters after the data frame. We use the ':' character to mention the rows and columns needed.
- To convert the data frame into a matrix, we use the as.matrix() function.
- When converting a data frame to a matrix, all variable types will be coerced to the same data type (typically characters if the data frame contains both numeric and character data).
- Hence when we check the structure of the matrix, we see that it is of a character type.

Create a new Data Frame

Hide

```
# declaring the integer and ordinal variables
integer_variable <- c(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
ordinal_variable <- c("Low", "Medium", "High", "High", "High", "Medium", "Low",
                      "Medium", "Low", "Low")
# assigning the variable as a factor and ordering the values
ordinal_variable <- factor(ordinal_variable, levels=c("Low", "Medium", "High"), ordered=TRUE)
# creating the data frame with the variables
data <- data.frame(Integer_Values = integer_variable, Ordinal_Values = ordinal_variable)
data
```

Integer_Values <dbl>	Ordinal_Values <ord>
10	Low
20	Medium
30	High
40	High
50	High
60	Medium
70	Low
80	Medium
90	Low
100	Low

1-10 of 10 rows

Hide

```
# checking the structure of the data
str(data)
```

```
'data.frame':   10 obs. of  2 variables:
 $ Integer_Values: num  10 20 30 40 50 60 70 80 90 100
 $ Ordinal_Values: Ord.factor w/ 3 levels "Low"<"Medium"<..: 1 2 3 3 3 2 1 2 1 1
```

Hide

```
# checking the levels of the data
levels(data$Ordinal_Values)
```

```
[1] "Low"    "Medium" "High"
```

Hide

```
# creating a new variable
new_vector <- c(11, 21, 31, 41, 51, 61, 71, 81, 91, 101)
# adding the new feature to the data frame
new_data <- cbind(data, New_Values=new_vector)
new_data
```

Integer_Values <dbl>	Ordinal_Values <ord>	New_Values <dbl>
10	Low	11
20	Medium	21
30	High	31
40	High	41
50	High	51
60	Medium	61
70	Low	71
80	Medium	81
90	Low	91
100	Low	101

1-10 of 10 rows

Explanations:

- We can create a vector using c(). We create two such vectors, one for the integer variable and one for the ordinal variable.
- When a string variable is created, it is of a categorical type. We can convert the variable into a factor by using the factor() function.
- When converting the variable into a factor, we mention the levels and if they are in order as a part of the parameters of the function. This helps in ordering the variable.
- Once our variables are ready, we use the data.frame() function to convert the variables into a data frame.

- To check the structure of the data frame, we use the str() function.
- To check the levels of the ordinal variable, we use the levels() function and pass the variable of the data frame.
- Using the cbind() function, we can bind the previous data frame and the new numeric vector into a new data frame. We can mention the name of the column and give it the variable of the new numeric vector.