Exercise:11

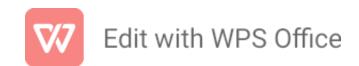
Introduction Dirty data problems:

Dirty data can be a result of various factors, including human error, data entry mistakes, software bugs, hardware malfunctions, or problems during data migration and integration.

1)Missing values:

Missing values are represented in R by the NA symbol. NA is a special value whose properties are different from other values. NA is one of the very few reserved words in R $\,$

```
x<- c(NA, 3, 4, NA, NA, NA)
is.na(x)
Output:
[1] TRUE FALSE FALSE TRUE TRUE TRUE
x <- c(1, 2, NA, 3, NA, 4)
d \leftarrow is.na(x)
x[! d]
Output:
[1] 1 2 3 4
x <- c(1, 2, 0 / 0, 3, NA, 4, 0 / 0)
x[! is.na(x)]
Output:
[1] 1 2 NaN 3 NA 4 NaN
[1] 1 2 3 4
# Create a data frame with 5 rows and 3 columns
data <- data.frame(
A = c(1, 2, NA, 4, 5),
 B = c(NA, 2, 3, NA, 5),
 C = c(1, 2, 3, NA, NA)
# View the resulting data frame
data
Output:
 ABC
1 1 NA 1
2 2 2 2
3 NA 3 3
4 4 NA NA
5 5 5 NA
Find all the missing values in the data
# Finding missing values in data.
sum(is.na(data))
Output:
[1] 5
Find all the missing values in the columns
# Finding missing values column wise
colSums(is.na(data))
```



Output: ABC 122

2) Data Manipulation:

In order to manipulate the data, R provides a library called dplyr which consists of many built-in methods to manipulate the data. So to use the data manipulation function, first need to import the dplyr package using library(dplyr) line of code. Below is the list of a few data manipulation functions present in dplyr package.

Function NameDescription

filter() Produces a subset of a Data Frame.

distinct() Removes duplicate rows in a Data Frame

Reorder the rows of a Data Frame arrange()

select()Produces data in required columns of a Data Frame

rename() Renames the variable names

mutate() Creates new variables without dropping old ones.

transmute() Creates new variables by dropping the old. summarize() Gives summarized data like Average, Sum, etc.

filter() method

The filter() function is used to produce the subset of the data that satisfies the condition specified in the filter() method. In the condition, we can use conditional operators, logical operators, NA values, range operators etc. to filter out data. Syntax of filter() function is given belowfilter(dataframeName, condition)

Example:

In the below code we used filter() function to fetch the data of players who scored more than 100 runs from the "stats" data frame. # import dplyr package

library(dplyr)

```
# create a data frame
stats <- data.frame(player=c('A', 'B', 'C', 'D'),
         runs=c(100, 200, 408, 19),
         wickets=c(17, 20, NA, 5))
```

fetch players who scored more

than 100 runs

filter(stats, runs>100)

Output

player runs wickets

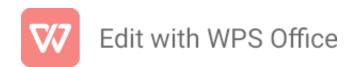
B 200 20 1

C 408 NA

distinct() method

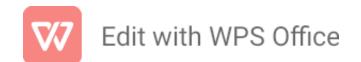
The distinct() method removes duplicate rows from data frame or based on the specified columns. The syntax of distinct() method is given belowdistinct(dataframeName, col1, col2,..., .keep_all=TRUE)

Here in this example, we used distinct() method to remove the duplicate rows from the data frame and also remove duplicates based on a specified column.



```
# import dplyr package
library(dplyr)
# create a data frame
stats <- data.frame(player=c('A', 'B', 'C', 'D', 'A', 'A'),
        runs=c(100, 200, 408, 19, 56, 100),
        wickets=c(17, 20, NA, 5, 2, 17))
# removes duplicate rows
distinct(stats)
#remove duplicates based on a column
distinct(stats, player, .keep_all = TRUE)
Output
player runs wickets
1
   A 100
    B 200
              20
3
    C 408
             NA
   D 19
4
              5
    A 56
player runs wickets
1
   A 100
             17
2
   B 200
              20
    C 408
3
             NA
    D 19
              5
arrange() method
In R, the arrange() method is used to order the rows based on a specified
column. The syntax of arrange() method is specified below-
arrange(dataframeName, columnName)
Example:
In the below code we ordered the data based on the runs from low to high
using arrange() function.
# import dplyr package
library(dplyr)
# create a data frame
stats <- data.frame(player=c('A', 'B', 'C', 'D'),
        runs=c(100, 200, 408, 19),
        wickets=c(17, 20, NA, 5))
# ordered data based on runs
arrange(stats, runs)
Output
player runs wickets
   D 19
1
              5
    A 100
             17
    B 200
              20
    C 408
              NA
select() method
The select() method is used to extract the required columns as a table by
```

The select() method is used to extract the required columns as a table by specifying the required column names in select() method. The syntax of select() method is mentioned below-



```
select(dataframeName, col1,col2,...)
Example:
Here in the below code we fetched the player, wickets column data only
using select() method.
# import dplyr package
library(dplyr)
# create a data frame
stats <- data.frame(player=c('A', 'B', 'C', 'D'),
        runs=c(100, 200, 408, 19),
        wickets=c(17, 20, NA, 5))
# fetch required column data
select(stats, player,wickets)
Output
player wickets
1
    Α
        17
2
    В
        20
3
    C
        NA
    D
rename() method
The rename() function is used to change the column names. This can be
done by the below syntax-
rename(dataframeName, newName=oldName)
Example:
In this example, we change the column name "runs" to "runs_scored" in
stats data frame.
# import dplyr package
library(dplyr)
# create a data frame
stats <- data.frame(player=c('A', 'B', 'C', 'D'),
        runs=c(100, 200, 408, 19),
        wickets=c(17, 20, NA, 5))
# renaming the column
rename(stats, runs_scored=runs)
Output
player runs_scored wickets
1
   Α
          100
                17
2
          200
                 20
  В
3
    С
          408
                NA
           19
    D
                 5
mutate() & transmute() methods
These methods are used to create new variables. The mutate() function
creates new variables without dropping the old ones but transmute()
function drops the old variables and creates new variables. The syntax of
both methods is mentioned below-
mutate(dataframeName, newVariable=formula)
```



In this example, we created a new column avg using mutate() and

transmute(dataframeName, newVariable=formula)

Example:

```
transmute() methods
# import dplyr package
library(dplyr)
# create a data frame
stats <- data.frame(player=c('A', 'B', 'C', 'D'),
        runs=c(100, 200, 408, 19),
        wickets=c(17, 20, 7, 5))
# add new column avg
mutate(stats, avg=runs/4)
# drop all and create a new column
transmute(stats, avg=runs/4)
Output
player runs wickets avg
   A 100 17 25.00
2
  B 200
            20 50.00
   C 408 7 102.00
  D 19 5 4.75
  avg
1 25.00
2 50.00
3 102.00
4 4.75
Here mutate() functions adds a new column for the existing data frame
without dropping the old ones where as transmute() function created a
new variable but dropped all the old columns.
summarize() method
Using the summarize method we can summarize the data in the data
frame by using aggregate functions like sum(), mean(), etc. The syntax of
summarize() method is specified below-
summarize(dataframeName, aggregate_function(columnName))
Example:
In the below code we presented the summarized data present in the runs
column using summarize() method
# import dplyr package
library(dplyr)
# create a data frame
stats <- data.frame(player=c('A', 'B', 'C', 'D'),
        runs=c(100, 200, 408, 19),
        wickets=c(17, 20, 7, 5))
# summarize method
summarize(stats, sum(runs), mean(runs))
Output
sum(runs) mean(runs)
    727 181.75
3) Duplicates:
```



A dataset can have duplicate values and to keep it redundancy-free and

```
accurate, duplicate rows need to be identified and removed.
# Create a sample vector with duplicate elements
vector_data <- c(1, 2, 3, 4, 4, 5)
# Identify duplicate elements
duplicated(vector_data)
# count of duplicated data
sum(duplicated(vector_data))
Output:
[1] FALSE FALSE FALSE TRUE FALSE
[1] 1
Removing Duplicate Data in a vector
We can remove duplicate data from vectors by using unique() functions
so it will give only unique values.
# Create a sample vector with duplicate elements
vector_data <- c(1, 2, 3, 4, 4, 5)
# Remove duplicate elements
unique(vector_data)
output:
Output:[1] 1 2 3 4 5
Identifying Duplicate Data in a data frame:
For identification, we will use the duplicated() function which returns the
count of duplicate rows.
Syntax:
duplicated(dataframe)
PROGRAM:
# Creating a sample data frame of students
# and their marks in respective subjects.
student_result=data.frame(name=c("Ram","Geeta","John","Paul",
                 "Cassie", "Geeta", "Paul"),
             maths=c(7,8,8,9,10,8,9),
             science=c(5,7,6,8,9,7,8),
             history=c(7,7,7,7,7,7,7)
# Printing data
student_result
duplicated(student_result)
sum(duplicated(student_result))
OUTPUT:
 name maths science history
1 Ram 7 5
                  7
2 Geeta 8 7 7
3 John 8 6 7
4 Paul 9 8 7
5 Cassie 10 9 7
6 Geeta 8 7 7
7 Paul 9
> duplicated(student_result)
[1] FALSE FALSE FALSE FALSE TRUE TRUE
> sum(duplicated(student_result))
[1] 2
```



4)OUTLIERS:

Outlier detection is a statistical approach used to find outliers in datasets. Measurement errors, incorrect data entry, or really anomalous data values are just a few of the causes of outliers.

data <- rnorm(500)

data[1:10] <- c(46,9,15,-90,42,50,-82,74,61,-32) data <- data[!data %in% boxplot.stats(data)\$out]

boxplot(data)

output:

5) SPELLING ERRORS:

Spell checking is the process of checking for and identifying spelling errors in a piece of text. It works by comparing each word in the text against a dictionary of correctly spelled words, and marking any words that are not found in the dictionary as potentially misspelled.

PROGRAM:

install.packages("hunspell")

library(hunspell)

words <- c("analyze", "langauge", "data")

correct <- hunspell_check(words)</pre>

print(correct)

OUTPUT:

[1] TRUE FALSE TRUE

