Lab1

Maria Bassem_20011141

2024-02-14

This report documents the R programming activities done in Lab1. The tasks encompass a variety of topics, ranging from basic variable declarations to more complex data preprocessing and analysis using the tidyverse library. The exercises aim to reinforce fundamental R programming skills and introduce concepts relevant to bioinformatics data analysis.

Part 0: Warmup

1. Declare and initialize variables of the following types: numeric, integer, character and complex. Display the variables

```
# numeric
num_var <- 1
print(num_var)
```

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

```
# integer
int_var <- 2L
print(int_var)
```

```
## [1] 2
```

```
# character
char_var <- "character"
print(char_var)
```

```
## [1] "character"
```

```
# complex
comp_var <- 9i + 4
print(comp_var)
```

```
## [1] 4+9i
```

2. Show the data types of these variables.

```
print(class(num_var))
```

```
## [1] "numeric"
```

```
print(class(int_var))
```

```
## [1] "integer"
```

```
print(class(char_var))
```

```
## [1] "character"
```

```
print(class(comp_var))
```

```
## [1] "complex"
```

3. Implement a while loop to perform a countdown from 10 to 0.

```
i <- 10
while(i >= 0){
  print(i)
  i <- i - 1
}
```

```
## [1] 10
## [1] 9
## [1] 8
## [1] 7
## [1] 6
## [1] 5
## [1] 4
## [1] 3
## [1] 2
## [1] 1
## [1] 0
```

4. Create an if-else statement to check if a number is even or odd.

```
x <- 3
if(x %% 2 == 0) { # mod is %% in R
  print("even")
} else {
  print("odd")
}
```

```
## [1] "odd"
```

5. Write a for loop to iterate over a vector of 10 elements and print each element

```
vector <- c(1:10)
for(i in vector){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
```

6. Create a 4D array fill it with random numbers and show its content

```
arr <- array(c(1:16), dim = c(4,2,4,2))
print(arr)
```

```
## , , 1, 1
##
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
##
## , , 2, 1
##
##      [,1] [,2]
## [1,]    9   13
## [2,]   10   14
## [3,]   11   15
## [4,]   12   16
##
## , , 3, 1
##
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
##
## , , 4, 1
##
##      [,1] [,2]
## [1,]    9   13
## [2,]   10   14
## [3,]   11   15
## [4,]   12   16
##
## , , 1, 2
##
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
##
## , , 2, 2
##
##      [,1] [,2]
## [1,]    9   13
## [2,]   10   14
## [3,]   11   15
## [4,]   12   16
##
## , , 3, 2
##
##      [,1] [,2]
## [1,]    1    5
## [2,]    2    6
## [3,]    3    7
## [4,]    4    8
##
```

```
## , , 4, 2
##
##      [,1] [,2]
## [1,]    9   13
## [2,]   10   14
## [3,]   11   15
## [4,]   12   16
```

7. Load the iris dataset into a frame. Find the number of rows, number of columns and the names of the columns.

Find the number of rows where the petal length is greater than 1.5 and the species is Setosa

```
data(iris) # Used data() function to load the iris
nrow(iris)
```

```
## [1] 150
```

```
ncol(iris)
```

```
## [1] 5
```

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

```
nrow(subset(iris, Petal.Length > 1.5 & Species == "setosa"))
```

```
## [1] 13
```

Part 1: Dependency and Dataset

Task 1.1: Dependency

1. Install package tidyverse

2. Load the tidyverse library

```
# install.packages('tidyverse')
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.4.4     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

Task 1.2: Dataset Loading

1. Download the BrainCancer dataset from this link. The dataset is in the form of csv. You need to find a way to read it from your filesystem into R Studio (hint: it is a function)

```
# Use read.csv() function to read the data from a CSV file.
brain_data <- read.csv("Brain_GSE50161.csv", header = TRUE, sep = ",")
```

2. Find the number of rows, number of columns and the names of the columns

```
nrow(brain_data)
```

```
## [1] 130
```

```
ncol(brain_data)
```

```
## [1] 54677
```

```
names(brain_data)[1:20]
```

```
##  [1] "samples"       "type"          "X1007_s_at"    "X1053_at"
##  [5] "X117_at"       "X121_at"       "X1255_g_at"    "X1294_at"
##  [9] "X1316_at"      "X1320_at"      "X1405_i_at"    "X1431_at"
## [13] "X1438_at"      "X1487_at"      "X1494_f_at"    "X1552256_a_at"
## [17] "X1552257_a_at" "X1552258_at"   "X1552261_at"   "X1552263_at"
```

Data Preprocessing

Task 2.1: Determining the Working Set

1. Create a dataframe with the columns (samples, type) and the first 3 Genes and the last 4 Genes. (i.e the table should have 9 columns)

```
working_set <- select(brain_data, samples, type, names(brain_data) [3:5], names(brain_data) [5467
4:54677])
names(working_set)
```

```
## [1] "samples"         "type"            "X1007_s_at"      "X1053_at"
## [5] "X117_at"         "AFFX.ThrX.M_at"  "AFFX.TrpnX.3_at" "AFFX.TrpnX.5_at"
## [9] "AFFX.TrpnX.M_at"
```

2. Create a table from the categorical values of column “type” from your dataframe. What is the most occurring type of cancer? You are encouraged to use ?table() or refer to the references to find the rdocs.

```
# create a frequency table (type_table) of the categorical values in the "type" column of the work
ing_set dataframe.
type_table <- table(working_set$type)
type_table
```

```
##
##            ependymoma          glioblastoma       medulloblastoma
##                    46                    34                    22
##                normal pilocytic_astrocytoma
##                    13                    15
```

```
# max(type_table) # number of occurrences of the max type of cancer
index <- which.max(type_table) # index of the max type of cancer
most_occurring_type <- names(type_table[index])
cat("The most occurring type of cancer is:", most_occurring_type, "\n")
```

```
## The most occurring type of cancer is: ependymoma
```

Task 2.2: Data Cleaning and Filtering

1. From your dataframe, how many values are NA?

```
sum(is.na(working_set))
```

```
## [1] 0
```

2. Filter rows (samples) where the expression in the first gene (X1007_s_at) is greater than 12.0. Print number of rows before and after filtration

```
cat("before:", nrow(working_set), "\n")# before
```

```
## before: 130
```

```
filtered_data <- filter(working_set, X1007_s_at > 12)
cat("after:", nrow(filtered_data), "\n")# after
```

```
## after: 91
```

Task 2.3: Data Analysis

1. For each gene in your dataframe, calculate a summary of the mean and the standard deviation across all samples.

```
# across --> within summarise to apply summary functions to multiple columns.

gene_summary <- summarise(working_set, across(-(1:2), list(Mean = mean, StdDev = sd), na.rm = TRU
E))
```

```
## Warning: There was 1 warning in `summarise()`.
## ℹ In argument: `across(-(1:2), list(Mean = mean, StdDev = sd), na.rm = TRUE)`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```
# Print the summary
print(gene_summary)
```

```
##   X1007_s_at_Mean X1007_s_at_StdDev X1053_at_Mean X1053_at_StdDev X117_at_Mean
## 1        12.27639         0.7901601      8.769583       0.6733962     7.722634
##   X117_at_StdDev AFFX.ThrX.M_at_Mean AFFX.ThrX.M_at_StdDev AFFX.TrpnX.3_at_Mean
## 1       1.037339            3.916795             0.1659169             3.701539
##   AFFX.TrpnX.3_at_StdDev AFFX.TrpnX.5_at_Mean AFFX.TrpnX.5_at_StdDev
## 1               0.180251             4.627912              0.1606634
##   AFFX.TrpnX.M_at_Mean AFFX.TrpnX.M_at_StdDev
## 1             4.633377              0.1923529
```

2. For each gene in your dataframe, calculate a summary of the mean and the standard deviation for each type of cancer.

```
by_type <- group_by(working_set, type)
cancer_summary <- summarise(by_type, across(-(1:2), list(Mean = mean, StdDev = sd), na.rm = TRUE))
# Print the summary
print(cancer_summary)
```

```
## # A tibble: 5 × 13
##   type                X1053_at_Mean X1053_at_StdDev X117_at_Mean X117_at_StdDev
##   <chr>                       <dbl>           <dbl>        <dbl>          <dbl>
## 1 ependymoma                   8.57           0.523         7.96          1.13
## 2 glioblastoma                 9.25           0.621         8.21          0.972
## 3 medulloblastoma              9.10           0.520         6.94          0.533
## 4 normal                       8.04           0.578         7.07          0.905
## 5 pilocytic_astrocyto…         8.44           0.481         7.60          0.565
## # ℹ 8 more variables: AFFX.ThrX.M_at_Mean <dbl>, AFFX.ThrX.M_at_StdDev <dbl>,
## #   AFFX.TrpnX.3_at_Mean <dbl>, AFFX.TrpnX.3_at_StdDev <dbl>,
## #   AFFX.TrpnX.5_at_Mean <dbl>, AFFX.TrpnX.5_at_StdDev <dbl>,
## #   AFFX.TrpnX.M_at_Mean <dbl>, AFFX.TrpnX.M_at_StdDev <dbl>
```

3. Write your summaries into a .csv file

```
# Write the cancer_summary into a CSV file
write.csv(cancer_summary, "cancer_summary.csv")

# Write the gene_summary into a CSV file
write.csv(gene_summary, "gene_summary.csv")
```

4. Create a dataframe that has the (sample, type, and X1007_s_at)columns. Add a column that calculates the mean gene expression in the first gene (X1007_s_at) for each cancer type across all samples. (hint: Use piping)

```
working_set2 <- select(working_set, samples, type, X1007_s_at)
working_set2 <- working_set2 %>%
  group_by(type) %>%
  mutate(mean_fun = mean(X1007_s_at))
working_set2
```

```
## # A tibble: 130 × 4
## # Groups:   type [5]
##    samples type       X1007_s_at mean_fun
##      <int> <chr>           <dbl>    <dbl>
##  1     834 ependymoma       12.5     12.8
##  2     835 ependymoma       13.1     12.8
##  3     836 ependymoma       13.1     12.8
##  4     837 ependymoma       12.5     12.8
##  5     838 ependymoma       12.7     12.8
##  6     839 ependymoma       12.5     12.8
##  7     840 ependymoma       13.7     12.8
##  8     841 ependymoma       12.6     12.8
##  9     842 ependymoma       12.5     12.8
## 10     843 ependymoma       12.3     12.8
## # ℹ 120 more rows
```