# Lab 1

Bahaa Khaled Mohamed 21010383
Omar Aldawy Ibrahim Aldawy 21010864

2025-02-18

## Contents

**1 Declare Variables** **1**

**2 Data Type** **2**

**3 Countdown using while loop** **2**

**4 Function to check even or odd** **3**

**5 Create a vector** **3**

**6 Create a 4D array with random numbers** **4**

**7 Iris** **5**

**8 Dependency** **5**

**9 Read data-set** **6**

**10 Data pre-processing** **7**
10.1 Determining the Working Set . . . 7
10.2 Data Cleaning and Filtering . . . 9

**11 Data Analysis** **10**
11.1 Genes Analysis . . . 10
11.2 Genes Analysis By Type . . . 12
11.3 Save summaries to csv files . . . 13

## 1 Declare Variables

```
num_var <- 10


int_var <- 7L


char_var <- "Bioinformatics"
```

```r
complex_var <- 4 + 3i


print(num_var)
```

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

```r
print(int_var)
```

```
## [1] 7
```

```r
print(char_var)
```

```
## [1] "Bioinformatics"
```

```r
print(complex_var)
```

```
## [1] 4+3i
```

## 2 Data Type

- return the type of each variable

```r
typeof(num_var)
```

```
## [1] "double"
```

```r
typeof(int_var)
```

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

```r
typeof(char_var)
```

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

```r
typeof(complex_var)
```

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

## 3 Countdown using while loop

- countdown from 10 to 0 until the condition is false

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

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

## 4 Function to check even or odd

```r
check_even_odd <- function(num) {
  if (num %% 2 == 0) {
    print("Even")
  } else {
    print("Odd")
  }
}
```

## 5 Create a vector

- group a collection of elements together

```r
vec <- c(1,2,3,4,5,6,7,8,9,10)

for (element in vec) {
  print(element)
}
```

```
## [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 with random numbers

```
array_4d <- array(runif(16, min=0, max=10), dim = c(2,2,2,2))
print(array_4d)
```

```
## , , 1, 1
##
##          [,1]     [,2]
## [1,] 8.597593 9.875530
## [2,] 9.155143 4.486042
##
## , , 2, 1
##
##          [,1]     [,2]
## [1,] 2.036271 6.106684
## [2,] 4.551914 3.053248
##
## , , 1, 2
##
##          [,1]     [,2]
## [1,] 8.588567 7.383165
## [2,] 2.385416 6.557691
##
## , , 2, 2
##
##          [,1]      [,2]
## [1,] 9.395763 0.2293431
## [2,] 8.908006 3.8458555
```

## 7 Iris

- we use the flowers data set to perform some operations

```r
data(iris)


num_rows <- nrow(iris)
num_cols <- ncol(iris)


column_names <- colnames(iris)


filtered_rows <- subset(iris, Petal.Length > 1.5 & Species == "setosa")


print(paste("Number of rows:", num_rows))
```

```
## [1] "Number of rows: 150"
```

```r
print(paste("Number of columns:", num_cols))
```

```
## [1] "Number of columns: 5"
```

```r
print("Column names:")
```

```
## [1] "Column names:"
```

```r
print(column_names)
```

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

```r
print(paste("Rows where Petal.Length > 1.5 & Species == Setosa:", nrow(filtered_rows)))
```

```
## [1] "Rows where Petal.Length > 1.5 & Species == Setosa: 13"
```

## 8 Dependency

- we need some libraries to perform some operations optimally

```r
install.packages('tidyverse')
library(tidyverse)
library(dplyr)
```

## 9 Read data-set

- we read the data set from a csv file

```
dataset <- read.csv("BrainCancerMin.csv")


print(paste("-Number of rows =", nrow(dataset)))
```

```
## [1] "-Number of rows = 130"
```

```
print(paste("-Number of columns =", ncol(dataset)))
```

```
## [1] "-Number of columns = 150"
```

```
print("-Column names are")
```

```
## [1] "-Column names are"
```

```
print(colnames(dataset))
```

```
##   [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"
##  [21] "X1552264_a_at" "X1552266_at"   "X1552269_at"   "X1552271_at"
##  [25] "X1552272_a_at" "X1552274_at"   "X1552275_s_at" "X1552276_a_at"
##  [29] "X1552277_a_at" "X1552278_a_at" "X1552279_a_at" "X1552280_at"
##  [33] "X1552281_at"   "X1552283_s_at" "X1552286_at"   "X1552287_s_at"
##  [37] "X1552288_at"   "X1552289_a_at" "X1552291_at"   "X1552293_at"
##  [41] "X1552295_a_at" "X1552296_at"   "X1552299_at"   "X1552301_a_at"
##  [45] "X1552302_at"   "X1552303_a_at" "X1552304_at"   "X1552306_at"
##  [49] "X1552307_a_at" "X1552309_a_at" "X1552310_at"   "X1552311_a_at"
##  [53] "X1552312_a_at" "X1552314_a_at" "X1552315_at"   "X1552316_a_at"
##  [57] "X1552318_at"   "X1552319_a_at" "X1552320_a_at" "X1552321_a_at"
##  [61] "X1552322_at"   "X1552323_s_at" "X1552325_at"   "X1552326_a_at"
##  [65] "X1552327_at"   "X1552329_at"   "X1552330_at"   "X1552332_at"
##  [69] "X1552334_at"   "X1552335_at"   "X1552337_s_at" "X1552338_at"
##  [73] "X1552340_at"   "X1552343_s_at" "X1552344_s_at" "X1552347_at"
##  [77] "X1552348_at"   "X1552349_a_at" "X1552354_at"   "X1552355_s_at"
##  [81] "X1552359_at"   "X1552360_a_at" "X1552362_a_at" "X1552364_s_at"
##  [85] "X1552365_at"   "X1552367_a_at" "X1552368_at"   "X1552370_at"
##  [89] "X1552372_at"   "X1552373_s_at" "X1552375_at"   "X1552377_s_at"
##  [93] "X1552378_s_at" "X1552379_at"   "X1552381_at"   "X1552383_at"
##  [97] "X1552384_a_at" "X1552386_at"   "X1552388_at"   "X1552389_at"
## [101] "X1552390_a_at" "X1552391_at"   "X1552393_at"   "X1552394_a_at"
## [105] "X1552395_at"   "X1552396_at"   "X1552398_a_at" "X1552399_a_at"
## [109] "X1552400_a_at" "X1552401_a_at" "X1552402_at"   "X1552405_at"
## [113] "X1552408_at"   "X1552409_a_at" "X1552410_at"   "X1552411_at"
## [117] "X1552412_a_at" "X1552414_at"   "X1552415_a_at" "X1552417_a_at"
## [121] "X1552418_at"   "X1552419_s_at" "X1552421_a_at" "X1552422_at"
## [125] "X1552423_at"   "X1552424_at"   "X1552425_a_at" "X1552426_a_at"
## [129] "X1552427_at"   "X1552430_at"   "X1552432_at"   "X1552436_a_at"
## [133] "X1552438_a_at" "X1552439_s_at" "X1552440_at"   "X1552445_a_at"
## [137] "X1552448_a_at" "X1552449_a_at" "X1552450_a_at" "X1552452_at"
## [141] "X1552453_a_at" "X1552455_at"   "X1552456_a_at" "X1552457_a_at"
## [145] "X1552458_at"   "X1552459_a_at" "X1552461_at"   "X1552463_at"
## [149] "X1552466_x_at" "X1552467_at"
```

## 10 Data pre-processing

### 10.1 Determining the Working Set

- here we select the subset of the working data we want to work with
- we select the samples, type, and the first 3 and last 4 genes
- we also count the number of each type of cancer and plot the distribution
- we use ggplot2 to plot the distribution

```r
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
```

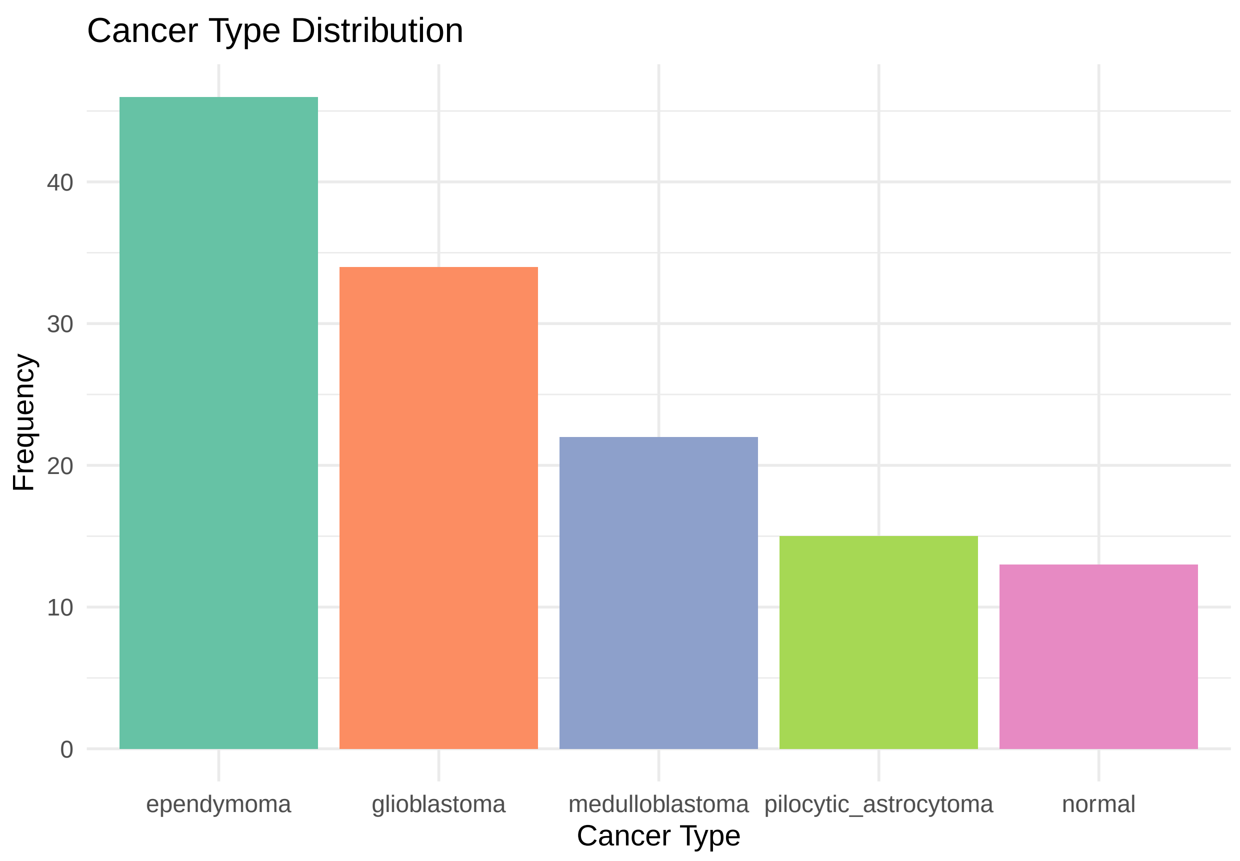
```r
subset_dataset <- dataset %>% select(samples, type, 3:5, 147:150)


type_count <- table(subset_dataset$type)
the_most_occurring_type_of_cancer <- names(which.max(type_count))
print(paste("The most occurring type of cancer is:", the_most_occurring_type_of_cancer))
```

```
## [1] "The most occurring type of cancer is: ependymoma"
```

```r
library(ggplot2)
cancer_dataframe <- as.data.frame(type_count)
colnames(cancer_dataframe) <- c("Type", "Count")


ggplot(cancer_dataframe, aes(x = reorder(Type, -Count), y = Count, fill = Type)) +
  geom_bar(stat = "identity") +
  scale_fill_brewer(palette = "Set2") +  # Use different colors for each type
  labs(title = "Cancer Type Distribution",
       x = "Cancer Type",
       y = "Frequency") +
  theme_minimal() +
  theme(legend.position = "none")  # Hide legend if unnecessary
```



## 10.2 Data Cleaning and Filtering

- we remove the rows with NA values
- we filter the data set to keep only the rows with gene X1007_s_at > 12
- we print the number of rows before and after filtering
- we also print the number of NA values in the data set

```r
print(paste("-The number of NA in dataset is", sum(is.na(dataset))))
```

```
## [1] "-The number of NA in dataset is 0"
```

```r
filtered_dataset <- dataset %>% filter(X1007_s_at > 12)
print(paste("-The number of rows before filtering is", nrow(dataset)))
```

```
## [1] "-The number of rows before filtering is 130"
```

```r
print(paste("-The number of rows after filtering is", nrow(filtered_dataset)))
```

```
## [1] "-The number of rows after filtering is 91"
```

# 11 Data Analysis

## 11.1 Genes Analysis

- we calculate the mean and standard deviation of each gene
- we print the results in a new data frame

```
genes <- dataset %>% select(!(1:2))

mean_summary <- summarise(genes, across(where(is.numeric),
                                         \(x) mean(x, na.rm = TRUE)))

sd_summary <- summarise(genes, across(where(is.numeric),
                                       \(x) sd(x, na.rm = TRUE)))



gene_summary <- bind_rows(mean_summary, sd_summary) %>%
  mutate(Summary = c("mean", "sd")) %>%
  select(Summary, everything())

print(gene_summary)
```

```
##   Summary X1007_s_at  X1053_at  X117_at   X121_at X1255_g_at  X1294_at
## 1    mean 12.2763929 8.7695830 7.722634 9.1602092  4.8420688 7.9683878
## 2      sd  0.7901601 0.6733962 1.037339 0.6153686  0.9220032 0.6302601
##    X1316_at  X1320_at X1405_i_at  X1431_at X1438_at X1487_at X1494_f_at
## 1 6.8001110 6.4724521  6.0689682 5.5483890 7.823669 8.445412  7.1270610
## 2 0.5374313 0.6598467  0.9041516 0.6277535 1.014098 0.417486  0.3803318
##   X1552256_a_at X1552257_a_at X1552258_at X1552261_at X1552263_at X1552264_a_at
## 1     9.4295216     9.1293691   6.0150955   5.6931106   6.8291438     8.4056834
## 2     0.7333103     0.4878505   0.3397897   0.3095495   0.6985547     0.7334797
##   X1552266_at X1552269_at X1552271_at X1552272_a_at X1552274_at X1552275_s_at
## 1    6.013895    6.199752   6.8580319     6.7745472   7.5486312     7.7630719
## 2    0.469087    1.511475   0.3121703     0.4125479   0.7349682     0.7290226
##   X1552276_a_at X1552277_a_at X1552278_a_at X1552279_a_at X1552280_at
## 1     6.6657325     9.2735055     6.7715511     8.3751234   5.1636937
## 2     0.3681259     0.6021066     0.6355755     0.4599907   0.5572503
##   X1552281_at X1552283_s_at X1552286_at X1552287_s_at X1552288_at X1552289_a_at
## 1   7.6015557      7.456754   7.0935824     8.1255486   5.3141125     6.1995777
## 2   0.5100853      1.126017   0.5472664     0.8751849   0.4687399     0.7462452
##   X1552291_at X1552293_at X1552295_a_at X1552296_at X1552299_at X1552301_a_at
## 1    7.896547   6.5121993     9.1568573    6.470438   7.0932490      6.856936
## 2    0.604485   0.4109929     0.5814464    1.296573   0.7483657      1.002242
##   X1552302_at X1552303_a_at X1552304_at X1552306_at X1552307_a_at X1552309_a_at
## 1   4.2045645     6.3340927   5.2574417   5.7376710     6.7151370     6.8155310
## 2   0.4190704     0.4207676   0.4550455   0.6329162     0.6909942     0.8520108
##   X1552310_at X1552311_a_at X1552312_a_at X1552314_a_at X1552315_at
## 1   9.0559179     7.8164358     7.3500769      5.547322   6.9608464
## 2   0.7548946     0.5183208     0.8865692      0.421423   0.4922071
##   X1552316_a_at X1552318_at X1552319_a_at X1552320_a_at X1552321_a_at
## 1     6.4724060   6.2639969     5.8361988      4.555108      5.968264
## 2     0.9718956   0.6261914     0.4337521      1.177537      1.536840
##   X1552322_at X1552323_s_at X1552325_at X1552326_a_at X1552327_at X1552329_at
## 1   4.2088241     6.6673290   4.2995910      7.753609   6.1408970    8.337783
## 2   0.3759827     0.6472075   0.8772059      1.577624   0.8473344    0.846414
##   X1552330_at X1552332_at X1552334_at X1552335_at X1552337_s_at X1552338_at
## 1    7.402024   7.8867326   6.4148783   7.2463515     6.2830837   5.5064811
## 2    0.631942   0.3676849   0.5830644   0.4418017     0.8668376   0.6022542
##   X1552340_at X1552343_s_at X1552344_s_at X1552347_at X1552348_at X1552349_a_at
## 1   5.8888185      7.107507      7.880360   8.7534131   6.9499892     5.6062154
## 2   0.3095162      0.718673      0.755687   0.6023439   0.8472756     0.3420977
##   X1552354_at X1552355_s_at X1552359_at X1552360_a_at X1552362_a_at
## 1   6.2160661     6.7875095   3.8758742     6.6291358     7.5254704
## 2   0.7976334     0.4143321   0.2441542     0.5002492     0.6500768
##   X1552364_s_at X1552365_at X1552367_a_at X1552368_at X1552370_at X1552372_at
## 1     7.0987786    7.660886      6.894104   4.7360488   7.6186211   3.5017308
## 2     0.6653559    2.001516      1.429586   0.5204502   0.6454441   0.1454611
##   X1552373_s_at X1552375_at X1552377_s_at X1552378_s_at X1552379_at X1552381_at
## 1     3.4681189   6.4861442     8.2288818      6.166455   3.2623323   6.4038040
## 2     0.2160424   0.5474507     0.4498286      1.092975   0.1879868   0.6719444
##   X1552383_at X1552384_a_at X1552386_at X1552388_at X1552389_at X1552390_a_at
## 1   7.8075200     5.9588600   5.1274161   6.7782373   3.9790220     3.8268622
## 2   0.4887146     0.2926548   0.7692635   0.4245398   0.4485788     0.4575297
##   X1552391_at X1552393_at X1552394_a_at X1552395_at X1552396_at X1552398_a_at
## 1   4.6920502   3.7737913     3.6950203    8.252928   5.9808732     4.8642857
## 2   0.4533146   0.2880898     0.2768093    0.481422   0.3175153     0.3778721
##   X1552399_a_at X1552400_a_at X1552401_a_at X1552402_at X1552405_at X1552408_at
## 1     5.6598208     6.3414804     4.5417425   5.4309594   6.0336303   4.6218301
## 2     0.5841043     0.7461352     0.3027757   0.2458358   0.4994683   0.3034554
##   X1552409_a_at X1552410_at X1552411_at X1552412_a_at X1552414_at X1552415_a_at
## 1     7.4423504   6.3472188   8.6383325     4.9799627   5.1722090     6.6342112
## 2     0.5292712   0.6549113   0.9701911     0.1992355   0.4645573     0.5376495
##   X1552417_a_at X1552418_at X1552419_s_at X1552421_a_at X1552422_at X1552423_at
## 1     6.5272752   5.8408605     7.1934457     4.7526449   7.7259748   6.9991053
## 2     0.9952833   0.4355428     0.6310376     0.3124227   0.5969705   0.7174029
##   X1552424_at X1552425_a_at X1552426_a_at X1552427_at X1552430_at X1552432_at
## 1   4.4513495     5.5395041    10.3905720   4.7247267   4.6009788   7.2527717
## 2   0.3537976     0.2798135     0.5563452   0.6512695   0.7949961   0.2751539
##   X1552436_a_at X1552438_a_at X1552439_s_at X1552440_at X1552445_a_at
## 1     5.6519386     5.7078774      7.970200   4.9650686      6.273696
## 2     0.6219971     0.4936738      2.003633   0.3047198      1.086908
##   X1552448_a_at X1552449_a_at X1552450_a_at X1552452_at X1552453_a_at
## 1      6.590378     5.0823332     6.6870138   4.9993240     4.8116805
## 2      1.390261     0.3554248     0.6455168   0.2988511     0.2511258
##   X1552455_at X1552456_a_at X1552457_a_at X1552458_at X1552459_a_at X1552461_at
## 1    6.193968     5.8783051     5.0564799   3.8663213     5.5107269   3.7559582
## 2    1.108910     0.3835441     0.5610003   0.2880709     0.2780204   0.2952949
##   X1552463_at X1552466_x_at X1552467_at
## 1    4.644858     3.7223114   7.1595733
## 2    0.485718     0.2936239   0.2819187
```

## 11.2 Genes Analysis By Type

- we calculate the mean and standard deviation of each gene by type
- we print the results in a new data frame

```r
library(dplyr)
library(tidyr)
grouped_summary <- dataset %>%
  group_by(type) %>%
  summarise(across(starts_with("X"),
                    list(mean = ~mean(.x, na.rm = TRUE),
                         sd = ~sd(.x, na.rm = TRUE)))) %>%
  pivot_longer(-type, names_to = c("Gene", "Measure"),
               names_pattern = "(.*)_(mean|sd)") %>%
  pivot_wider(names_from = Gene, values_from = value) %>%
  mutate(Measure = paste(Measure, type, sep = "_")) %>%
  select(-type)

colnames(grouped_summary)[1] <- "measure"

print(grouped_summary)
```

```
## # A tibble: 10 x 149
##    measure      X1007_s_at X1053_at X117_at X121_at X1255_g_at X1294_at X1316_at
##    <chr>             <dbl>    <dbl>   <dbl>   <dbl>      <dbl>    <dbl>    <dbl>
##  1 mean_ependy~     12.8      8.57    7.96    9.19       4.39     8.17     6.72
##  2 sd_ependymo~      0.355    0.523   1.13    0.599      0.573    0.572    0.525
##  3 mean_gliobl~     12.4      9.25    8.21    9.22       4.87     8.08     6.65
##  4 sd_glioblas~      0.484    0.621   0.972   0.607      0.830    0.647    0.481
##  5 mean_medull~     11.2      9.10    6.94    8.95       4.55     7.37     6.88
##  6 sd_medullob~      0.541    0.520   0.533   0.723      0.607    0.321    0.529
##  7 mean_normal      11.3      8.04    7.07    9.07       6.05     7.46     7.35
##  8 sd_normal         0.581    0.578   0.905   0.380      1.07     0.348    0.518
##  9 mean_pilocy~     12.9      8.44    7.60    9.33       5.53     8.43     6.79
## 10 sd_pilocyti~      0.288    0.481   0.565   0.665      0.990    0.405    0.456
## # i 141 more variables: X1320_at <dbl>, X1405_i_at <dbl>, X1431_at <dbl>,
## #   X1438_at <dbl>, X1487_at <dbl>, X1494_f_at <dbl>, X1552256_a_at <dbl>,
## #   X1552257_a_at <dbl>, X1552258_at <dbl>, X1552261_at <dbl>,
## #   X1552263_at <dbl>, X1552264_a_at <dbl>, X1552266_at <dbl>,
## #   X1552269_at <dbl>, X1552271_at <dbl>, X1552272_a_at <dbl>,
## #   X1552274_at <dbl>, X1552275_s_at <dbl>, X1552276_a_at <dbl>,
## #   X1552277_a_at <dbl>, X1552278_a_at <dbl>, X1552279_a_at <dbl>, ...
```

## 11.3 Save summaries to csv files

- here we create a function to save the summaries to csv files

```r
save_to_csv <- function(ds, path) {
  if(!endsWith(path, ".csv")){
    path <- paste0(path, ".csv")
  }

  write.csv(ds, path, row.names = TRUE)
}

save_to_csv(gene_summary, "gene_summary.csv")
save_to_csv(grouped_summary, "grouped_summary.csv")
```