Intro\_to\_R\_I

2023-05-10

## Intro to R (Part I)

### Installing R under conda

Create new environment for R

**conda create –name r-lang**

Install R and a package r-essentials

**conda install -c r r-essentials**

Install R studio

**conda install -c r rstudio**

Installing R packages under conda

**conda install -c r package-name**

### Section 1. R Basics

# R as a calculator  
  
# Addition  
2 + 7

## [1] 9

# Subtraction  
10 - 0.5

## [1] 9.5

# Division  
6 / 3

## [1] 2

# Remainder after division  
5 %% 3

## [1] 2

# Multiplication  
3 \* 5

## [1] 15

# Raise to the power  
3^2

## [1] 9

# logarithms  
log2(10) # logarithm with base 2

## [1] 3.321928

log10(10) # logarithm with base 10

## [1] 1

# log(x, base)  
log(5, 3)

## [1] 1.464974

# Trigonometry  
x <- 10  
cos(x) # Cosine of x

## [1] -0.8390715

sin(x) # Sine of x

## [1] -0.5440211

tan(x) #Tangent of x

## [1] 0.6483608

acos(x) # arc-cosine of x

## [1] NaN

asin(x) # arc-sine of x

## [1] NaN

atan(x) #arc-tangent of x

## [1] 1.471128

# Other functions  
abs(-10) # absolute

## [1] 10

sqrt(10) # square root

## [1] 3.162278

### Section 2. Assign values to variables

# <- assign value to a variable  
# = - this will also work  
  
a <- 10  
a

## [1] 10

a = 10  
a

## [1] 10

# Use descriptive variable names for better readablity  
gene\_length <- 1000 # good variable names  
  
geneLength <- 1000 # another example of a good name  
  
gene1 <- "P53" # you can use numbers in gene names  
  
.gridRemove <- 1 # this is also fine  
  
# Do not use special characters in variable naming:  
# -, %, $, \*, (, etc.  
  
# You can directly address a variable or use print() function to display value  
gene1 <- "P53"  
gene1

## [1] "P53"

print(gene1)

## [1] "P53"

# ls() function will show objects created during the session  
ls()

## [1] "a" "gene\_length" "gene1" "geneLength" "x"

# It is possible to remove variable from the environment with rm()  
rm(gene1)

### Section 3. Data types

# Basic data types in R  
  
# Boolean: holds TRUE/FALSE values  
gene1 <- "P53"  
gene1\_type <- is.character(gene1)  
print(gene1\_type)

## [1] TRUE

is.numeric(gene1)

## [1] FALSE

# Numeric: any numeric data  
geneLength <- 1000  
is.numeric(geneLength)

## [1] TRUE

# Character: any characters  
rnaseq\_dir <- "/home/steve/rnaseq\_mar10\_2023"  
rnaseq\_dir

## [1] "/home/steve/rnaseq\_mar10\_2023"

is.character(rnaseq\_dir)

## [1] TRUE

# Quotes with convert anything into characters  
num\_genes <- "123"  
num\_genes

## [1] "123"

is.character(num\_genes)

## [1] TRUE

# Same with single quotes  
num\_genes <- '123'  
num\_genes

## [1] "123"

is.character(num\_genes)

## [1] TRUE

### Section 4.Vectors

# Create a numeric vector  
qual <- c(10, 15, 30, 30)  
qual

## [1] 10 15 30 30

# Create a character vector   
bases <- c('A', 'C', 'T', 'G')  
bases

## [1] "A" "C" "T" "G"

# A vector of boolean values  
logic <- c(TRUE, FALSE)  
logic

## [1] TRUE FALSE

# R is centered around vectors,  
# a variable that holds one value is just  
# a vector with length 1  
base <- c('A')  
base

## [1] "A"

length(base)

## [1] 1

# R vectors are subject to vector algebra  
2 \* c(1, 5, 10)

## [1] 2 10 20

c(2, 2, 2) + c(2, 2, 2)

## [1] 4 4 4

c(4, 2, 2) + c(2)

## [1] 6 4 4

# If two vectors are of unequal length, the shorter one will be recycled in order to match   
# the longer vector.  
c(10, 20, 30, 50, 60, 70) + c(1, 2, 3)

## [1] 11 22 33 51 62 73

# Named vector  
foldChanges <- c("gene1"=2.3, "gene2"=3.4, "gene3"=6.7, "gene4"=2)  
foldChanges

## gene1 gene2 gene3 gene4   
## 2.3 3.4 6.7 2.0

# Subsetting vectors  
  
# by index  
foldChanges[2]

## gene2   
## 3.4

# select multiple elements  
foldChanges[c(1,3)]

## gene1 gene3   
## 2.3 6.7

# Range of values  
foldChanges[c(2:4)]

## gene2 gene3 gene4   
## 3.4 6.7 2.0

# By name in named vectors  
foldChanges[c("gene1", "gene3")]

## gene1 gene3   
## 2.3 6.7

# Selection by logical vector  
foldChanges > 3

## gene1 gene2 gene3 gene4   
## FALSE TRUE TRUE FALSE

foldChanges[foldChanges > 3]

## gene2 gene3   
## 3.4 6.7

foldChanges != 2

## gene1 gene2 gene3 gene4   
## TRUE TRUE TRUE FALSE

foldChanges[foldChanges != 2]

## gene1 gene2 gene3   
## 2.3 3.4 6.7

# Vectors with missing values  
genes <- c('P53', 'IL1', NA, NA)  
genes

## [1] "P53" "IL1" NA NA

is.na(genes)

## [1] FALSE FALSE TRUE TRUE

# We can sum vectors  
sum(foldChanges)

## [1] 14.4

mean(foldChanges)

## [1] 3.6

median(foldChanges)

## [1] 2.85

# Many other statistical and methematical functions

### Section 5. Matrices

# Create a matrix from vectors  
# Numeric vectors  
col1 <- c(5, 6, 7)  
col2 <- c(2, 4, 5)  
col3 <- c(7, 3, 4)  
  
# Combine the vectors by column  
my\_data <- cbind(col1, col2, col3)  
my\_data

## col1 col2 col3  
## [1,] 5 2 7  
## [2,] 6 4 3  
## [3,] 7 5 4

# Add rownames  
rownames(my\_data) <- c("row1", "row2", "row3")  
my\_data

## col1 col2 col3  
## row1 5 2 7  
## row2 6 4 3  
## row3 7 5 4

# bind by rows  
my\_data <- rbind(c(1,2,3), c(1,2,3), c(1,2,3))  
my\_data

## [,1] [,2] [,3]  
## [1,] 1 2 3  
## [2,] 1 2 3  
## [3,] 1 2 3

colnames(my\_data) <- c("col1", "col2", "col3")  
my\_data

## col1 col2 col3  
## [1,] 1 2 3  
## [2,] 1 2 3  
## [3,] 1 2 3

# Another way to create a matrix  
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,  
 dimnames = NULL)

## [,1]  
## [1,] NA

# data: an optional data vector  
# nrow, ncol: the desired number of rows and columns, respectively.  
# byrow: logical value. If FALSE (the default) the matrix is filled by columns, otherwise the matrix is filled by rows.  
# dimnames: A list of two vectors giving the row and column names respectively.  
  
mat <- matrix(  
 data = c(1,2,3, 11,12,13),   
 nrow = 2, byrow = TRUE,  
 dimnames = list(c("row1", "row2"), c("C.1", "C.2", "C.3"))  
 )  
mat

## C.1 C.2 C.3  
## row1 1 2 3  
## row2 11 12 13

# Dimensions of the matrix   
ncol(mat)

## [1] 3

nrow(mat)

## [1] 2

dim(mat)

## [1] 2 3

# Subsetting matrices  
  
# Select second row  
mat[2,] # comma on the right

## C.1 C.2 C.3   
## 11 12 13

# Select a second column  
mat[,2] # comma on the left

## row1 row2   
## 2 12

mat[1:2,] # select rows from 1 to 2

## C.1 C.2 C.3  
## row1 1 2 3  
## row2 11 12 13

mat[,1:2] # select columns from 1 to 2

## C.1 C.2  
## row1 1 2  
## row2 11 12

mat[,c(1, 3)] # select columns 1 and 3

## C.1 C.3  
## row1 1 3  
## row2 11 13

mat[,"C.1"] # select by name

## row1 row2   
## 1 11

# R support matrix algebra  
mat \* 2

## C.1 C.2 C.3  
## row1 2 4 6  
## row2 22 24 26

mat + c(1, 2, 3)

## C.1 C.2 C.3  
## row1 2 5 5  
## row2 13 13 16

mat / 3

## C.1 C.2 C.3  
## row1 0.3333333 0.6666667 1.000000  
## row2 3.6666667 4.0000000 4.333333

# Get row and col sums  
rowSums(mat)

## row1 row2   
## 6 36

colSums(mat)

## C.1 C.2 C.3   
## 12 14 16

### Section 6. Factors

# Create a factor variable  
animals <- factor(c("frog", "frog", "cat", "cat"))  
animals

## [1] frog frog cat cat   
## Levels: cat frog

# Factors have categories or levels  
  
# Access levels  
levels(animals)

## [1] "cat" "frog"

# Check if factor  
is.factor(animals)

## [1] TRUE

# Summarize factor  
summary(animals)

## cat frog   
## 2 2

# Crosstabulation with factors  
habitat <- factor(c("water", "water", "land", "land"))  
table(animals, habitat)

## habitat  
## animals land water  
## cat 2 0  
## frog 0 2

### Section 7. Data frames

# Create a data frame  
gene\_data <- data.frame(  
 geneID = c("gene1", "gene2", "gene3"),  
 gene\_length = c(1000, 2000, 3000),  
 gc\_content = c(0.25, 0.50, 0.75)  
 )  
  
gene\_data

## geneID gene\_length gc\_content  
## 1 gene1 1000 0.25  
## 2 gene2 2000 0.50  
## 3 gene3 3000 0.75

# Check class  
is.data.frame(gene\_data)

## [1] TRUE

# Select rows  
gene\_data[1,]

## geneID gene\_length gc\_content  
## 1 gene1 1000 0.25

# Select columns  
gene\_data[2,]

## geneID gene\_length gc\_content  
## 2 gene2 2000 0.5

# Select column by name (same applies to rows)  
gene\_data[,'geneID']

## [1] "gene1" "gene2" "gene3"

# Select range of columns (sample applies to rows)  
gene\_data[,c(1:3)]

## geneID gene\_length gc\_content  
## 1 gene1 1000 0.25  
## 2 gene2 2000 0.50  
## 3 gene3 3000 0.75

# Exclude columns  
gene\_data[,-1]

## gene\_length gc\_content  
## 1 1000 0.25  
## 2 2000 0.50  
## 3 3000 0.75

# Select rows by condition  
gene\_data[gene\_data$gene\_length > 2000,]

## geneID gene\_length gc\_content  
## 3 gene3 3000 0.75

# Combine with column indexes  
gene\_data[gene\_data$gene\_length > 2000, c(1,2)]

## geneID gene\_length  
## 3 gene3 3000

# subset function  
subset(gene\_data, gene\_length > 2000)

## geneID gene\_length gc\_content  
## 3 gene3 3000 0.75

# Select columns with $  
gene\_data$geneID

## [1] "gene1" "gene2" "gene3"

gene\_data$gene\_length

## [1] 1000 2000 3000

# Add a column with dollar sign  
gene\_data$exprs <- c(0.5, 10, 200)  
gene\_data

## geneID gene\_length gc\_content exprs  
## 1 gene1 1000 0.25 0.5  
## 2 gene2 2000 0.50 10.0  
## 3 gene3 3000 0.75 200.0

# Add a column with cbind  
gene\_data <- cbind(gene\_data, c(10, 15, 20))  
names(gene\_data)[5] <- "num\_exons"  
gene\_data

## geneID gene\_length gc\_content exprs num\_exons  
## 1 gene1 1000 0.25 0.5 10  
## 2 gene2 2000 0.50 10.0 15  
## 3 gene3 3000 0.75 200.0 20

### Section 8. Lists

# Create a list  
gene\_data <- list(  
 geneID = c("gene1", "gene2", "gene3"),  
 gene\_length = c(1000, 2000, 3000),  
 gc\_content = c(0.25, 0.50, 0.75)  
 )  
  
gene\_data

## $geneID  
## [1] "gene1" "gene2" "gene3"  
##   
## $gene\_length  
## [1] 1000 2000 3000  
##   
## $gc\_content  
## [1] 0.25 0.50 0.75

# Get names of the elements of a list  
names(gene\_data)

## [1] "geneID" "gene\_length" "gc\_content"

# Number of elements in the list  
length(gene\_data)

## [1] 3

# Select elements of the list with $  
gene\_data$geneID

## [1] "gene1" "gene2" "gene3"

# Select by name  
gene\_data[["geneID"]]

## [1] "gene1" "gene2" "gene3"

# Select by index  
gene\_data[[1]]

## [1] "gene1" "gene2" "gene3"

# Add elements to the list  
gene\_data$exprs <- c(02, 5, 10)  
gene\_data

## $geneID  
## [1] "gene1" "gene2" "gene3"  
##   
## $gene\_length  
## [1] 1000 2000 3000  
##   
## $gc\_content  
## [1] 0.25 0.50 0.75  
##   
## $exprs  
## [1] 2 5 10

# Select values within components of a list  
gene\_data[[4]][2]

## [1] 5

gene\_data$exprs[2]

## [1] 5

gene\_data[["geneID"]][3]

## [1] "gene3"