Вступ до R. Частина 1

#####Структура воркшопу \* Вправа на інтерпритацію коду \* Зберегти і завантажити дані \* Дослідження даних \* Візуалізація: на колінах і для звітів \* Вступ до Tidyverse \* Практична

###Вправа на інтерпритацію коду

Завдання: \* поясніть код, коментуючи кожну строку чи блок коду (користуйтеся документацією/Інтернетом за потреби); \* запустіть код; \* підсумуйте своїми словами, що робить весь блок коду; \* поміняйте місцями строки 69 і 71, запустіть код ще раз. Поясніть різницю у вихідних даних.

Опціонально: \* запропонуйте, як можна покращити код та/або записати його у меньшій кількості строк.

#@function that prints how many percent of code already executed  
  
#  
percent\_df <- function(num\_of\_itterations){  
 # We pass a specific num\_of\_itterations and divide by ten, and after we round it to zero digits after dot  
 per<-round(num\_of\_itterations/10,0)  
 # seq(seq(10, 100, by=10) create vector from with values from 10 to 100 with step = 10  
 # Sapply will apply custom func to every elems of created vector to create strings (10%, 60% and ... )in percent column  
 perdf<-data.frame(percent = sapply(seq(10, 100, by=10),function(x){paste(x,"%",sep="")}))  
 # create new column value with NAs  
 perdf$value<-NA  
 # Iterate 10 times as in seq() func and with this func we return perdf with new values in "value" column where the new value = per\*i, where if i = 1, we will create value for first value(row) in perdf for "value" column  
 for(i in 1:10){perdf[i,"value"]<-per\*i}  
 #  
 return(perdf)  
}  
#  
percent\_print <- function(now\_per,itterations\_passed,perdf){  
 #  
 if (itterations\_passed==(perdf[now\_per,"value"]+1) & now\_per<11){  
 print(paste(Sys.time(),perdf[now\_per,"percent"]))  
 now\_per<-now\_per+1  
 }  
 #  
 return(now\_per)  
}  
#@example of functions use  
#  
num\_of\_itterations<-76  
#  
perdf<-percent\_df(num\_of\_itterations)  
perdf  
#  
now\_per<-1  
#  
for (i in 1:num\_of\_itterations){  
 #  
 if (i==1){start\_time1<- Sys.time()}  
 #  
 Sys.sleep(1)  
 #  
 if (i==1){  
 end\_time1 <- Sys.time()  
 diff1<-end\_time1 - start\_time1  
 alltime<-round(diff1\*num\_of\_itterations,2)  
 print(paste("Estimated time of code execution is ",as.character(alltime),sep=""))  
 }  
 #  
 now\_per<-percent\_print(now\_per,i,perdf)  
}  
#Підсумуйте значення коду  
#...  
#Місце для пояснення різниці у вихідних даних:  
#...

референс: не опублікований код лекторки.

###Зберігти і завантажити дані

Давайте підемо далі і пограємо із зразком реальних даних.

#Loads specified data sets, or list the available data sets.

Нам знадобляться декілька датасетів для різноманітності прикладів.

#Load several data sets for later use.

###Зберегти і завантажити данні

Зазвичай дані, які потрібно переглянути й проаналізувати, зберігаються в текстовому файлі з розширеннями .txt, .csv, .tsv, .xlsx тощо. Розширення можуть вказувати на роздільник між стовпцями у файлі (наприклад, кома є роздільником у .csv і tab у .tsv). Існують функції, призначені для читання файлів в R. Але спочатку збережемо дата фрейм **mtcars** у файли з різним розширенням. > Зауважте, файли .xlsx читаються окремими пакетами, напр.”readxl”, який ми не розглядатимемо на семінарі.

#write.table prints its required argument x to a file.  
# row.names = FALSE does not allow to include row names in text file.  
# sep="\t" indicates tab separator.  
  
data()  
data("mtcars") # we added df mtcars to our environment and we may call it  
data("pressure")  
data(BOD)  
  
write.table(mtcars,"mtcars.tsv",row.names = FALSE, sep = "\t" )

# write.csv() by default sets separator as comma (,) and dot (.) as decimal point.  
write.csv(mtcars,"mtcars.csv", row.names = F)

Тепер файли *mtcars.tsv* і *mtcars.csv* мають з’явитися у вашому робочому каталозі. Ми можемо прочитати файл і зберегти його вміст у змінній.

#read file. header=TRUE indicates that first row becomes column names.  
mtcars\_fromFile = read.table("mtcars.tsv",sep = "\t", header = TRUE)

###Дослідження даних

Рекомендується досліджувати набір даних перед аналізом. Це дає вам розуміти дані та перевірити, чи щось з ними не так. Як приклад набору даних ми будемо використовувати *mtcars*. Запустіть наступний код і подивіться на праве нижнє вікно, щоб дізнатися більше про набір даних.

#data set help  
head(mtcars\_fromFile)  
?mtcars

Перевірте розмір набору даних.

# output: number of rows, number of columns  
dim(mtcars\_fromFile)

Для першого погляду на дані достатньо побачити назви стовпців і перші/останні рядки.

# second argument specifies number of rows you want to display (default is 5 rows).  
tail(mtcars\_fromFile,4)

#display last rows.

Крім того, компактно відобразіть внутрішню структуру об’єкта R.

# output: column names: data class: first 10 values.  
head(mtcars,10)  
str(mtcars)

А якщо ви хочете досліджувати дані вручну, відображайте їх у повному розмірі.

# the window will open next to R script.  
View(mtcars)  
view()

Ми хотіли б знати, чи якісь значення відсутні в даних або дублюються. Обидва випадки можуть спричинити оманливі результати після аналізу.

# check if there's any NA or NaN values  
# arr.ind - logical; should array indices be returned when x is an array?  
which(is.na(mtcars),arr.ind = TRUE)  
which(is.na(mtcars),arr.ind = FALSE)

# check if there blank values (no character)  
mtcars[mtcars == ""]  
mtcars == ""

#check if there duplicated rows  
# duplicated() determines which elements of a vector or data frame are duplicates.  
sum(duplicated(mtcars))

Якщо ви очікуєте, що будь-який стовпець матиме категоричні значення, перевірте, чи всі категорії присутні й названі правильно, а також кількість рядків, що містять кожну з категорій.

# unique() removes duplicated elements/rows.  
unique(mtcars$cyl)

Кількість значень кожної категорії можна легко підрахувати…

# table() builds a contingency table of the counts.  
table(mtcars$cyl)

Також можемо сортувати набір даних за стовпцями.

#order() returns a rearranged data set with into ascending or descending order.  
head(mtcars[order(mtcars$cyl),10])  
order(mtcars$cyl)  
mtcars[order(mtcars$cyl),]

Ви можете зробити швидку статистичну оцінку ваших даних. Можна перевірити кожен параметр окремо (середнє, мінімальне, максимальне тощо) або всі разом.

# by parameter  
min(mtcars$mpg)  
max(mtcars$mpg)  
mean(mtcars$mpg)

# all parameters  
summary(mtcars)

Ще один ефективний спосіб швидко зрозуміти дані – візуалізувати їх. Однак візуалізація також відіграє важливу роль у відображенні результатів аналізу у звітах чи статтях.

###Візуалізація

R base дозволяє робити прості візуалізації. Але найпопулярнішим пакетом для візуалізації біологічних даних є *ggplot2*.

*ggplot2* допомагає створювати складні графіки. Він надає гнучкіший програмний інтерфейс для контролю даних, які відображаються, того, як вони відображаються та загальних візуальних властивостей.

Тут ми швидко розглянемо попередньо завантажені дані за базовими графіками R і *ggplot2*, щоб побачити різницю та приклади синтаксису.

Існують такі типи простих графіків, як *діаграма розсіювання, лінійна діаграма, стовпчаста діаграма, гістограма, коробчаста діаграма, крива, хітмап*.

Однією з базових функцій є plot.

# Scatter plot shows relationship between two sets of numbers.   
plot(mtcars$wt,mtcars$mpg)  
  
tibble::view(mtcars)

*Висновок: ймовірно, залежність між зміними mpg i wt існує. Подальші висновки потребують додаткових обчислень.*

Лінійна діаграма зручна, коли необхідно показати тенденцію зміни у від х або порівняти декілька зміних.

# Line plot are useful when comparing multiple variables.  
# type="l" specifies that line should be plotted  
plot(pressure$temperature,pressure$pressure, type="l") +   
 points(pressure$temperature, pressure$pressure, col = "red")  
  
  
# add points to see all values.  
  
dev.off  
plot(pressure$temperature,pressure$pressure)

*Висновок: тиск зростає експоненційно із зростанням температури у данному ноборі даних. (Для впевненішого висновку необхідно побудувати математичну модель)*

Щоб побудувати кілька графіків разом (subplots), використовуйте наступний синтаксис.

# par() can be used to set or query graphical parameters.  
# mfrow - vector of the form c(number of rows, columns).  
  
par(mfrow = c(1,2))   
  
# hist() plots a histogram, where second argument breaks=20 means to devide the x range into 20 blocks.  
  
# hist() automatically adds plot title and axis labels.  
hist(ToothGrowth$len, 20)  
dev.off()  
#dev.off shuts down the specified (by default the current) graphic. If not written, new plots will overdraw the old. Here we commented previous line to display graphics.

*Висновок: довжина зубів у кожній групі відрізняється (суттєво чи ні - справа статистичних тестів). Найчастіше зустрічаються зуби в діапозоні 25-30 мм.*

Щоб зберегти *базові* графіки як файл, додайте **extention\_name\_function(“directory/name\_of\_plot.extention”)**

#also png, pdf etc.  
jpeg("./rplot.jpg")  
  
data <- as.matrix(mtcars)  
# heatmap displays numbers as degree of color transition throughout a color pallet.  
heatmap(data, col = heat.colors(10)) +   
 legend(x = "bottomright", legend = c("min","ave","max"), fill = heat.colors(3) )  
# a dendrogram is automatically add (displays resembling samples).  
  
# a good practice is to add a key to color ranges.   
  
# rplot.jpg should appear in your working directory.

Синтаксис *ggplot2* відрізняється від базових графіків. Якщо базові графіки зазвичай визиваються як **plot\_function(data, style\_arguments)**, то ggplots є складнішими: **ggplot(data = , mapping = aes()) + ()** \* естетична (aes) функція бере змінні, які потрібно побудувати, і визначає, як їх представити на графіку, наприклад, у вигляді позицій x/y і таких характеристик, як розмір, форма, колір тощо.

На початку воркшопу ми вже встановили та завантажили пакет *ggplot2*.

#load the package  
library(ggplot2)  
# blank plot  
ggplot(mtcars, aes(x = wt, y = mpg)) +   
 geom\_point()

Незважаючи на те, що x і y вказані, графік є пустим. Це тому, що ggplot не припускає, що ви мали на увазі точкову діаграму або лінійну діаграму тощо, яку потрібно намалювати (без «geoms»).

* додати ‘geoms’ – графічне представлення даних на графіку (точки, лінії, смуги тощо). Може бути кілька «geoms» у поєднанні з «+» таким чином, щоб відобразити кілька графіків в одному.
  + *geom\_point()* для діаграм розсіювання
  + *geom\_line()* для ліній тренду
  + *geom\_col()* для стовпчастої діаграми
  + *geom\_histogram()* для гістограм
  + *geom\_boxplot()* для бокс плот
  + *stat\_function()* для кривої

#Run whole chank.  
#scatter plot  
ggplot(mtcars, aes(x = wt, y = mpg)) +  
 geom\_point(aes(color = mpg)) + scale\_colour\_gradient(low = "yellow", high = "red", na.value = "black")  
  
  
#scatter plot  
ggplot(mtcars, aes(x = wt, y = mpg,color = mpg)) +  
 geom\_point() + scale\_colour\_gradient(low = "yellow", high = "red", na.value = "black")  
  
  
#scale\_colour\_gradient() gives color scale to chosen variable

#box plot  
boxplot(ToothGrowth$supp,ToothGrowth$len)  
ToothGrowth$len  
ToothGrowth$supp  
ggplot(ToothGrowth, aes(x = supp, y = len)) + geom\_boxplot()  
  
ggplot(ToothGrowth, aes(x = interaction(supp,len), y = len)) + geom\_boxplot()

#curve  
  
myfun <- function(xvar) {1/(1+exp(-xvar+10))}  
ggplot(data.frame(x = c(0,20)), aes(x=x )) + stat\_function(fun = myfun(x), geom = "point")

**Гарно, чи не так!?** > Є можливість змінювати назви осей, назву, розмір графіка, колір, обертати. Але ми пропустимо ці нюанси. Щоб дізнатися більше, відвідайте <https://r-graph-gallery.com/>.

Хоча ви можете використовувати par(), щоб зберегти ggplot як базовий графік, існує спеціальна функція з пакету *ggplot2*.

#bar plot  
ggplot(BOD, aes(x = Time, y = demand)) + geom\_col()  
# Save the plot to a pdf  
ggsave("myplot.pdf")

###Tidyverse

Tidyverse - це набір пакетів для написання “охайного” коду. Всесвіт пакетів tidyverse - це набір пакетів, спеціально орієнтованих на data science.

Більшість строк коду у цьому курсі, під час аналізу даних РНК-секвенування, написані у стилі Tidyverse.

У чому принципова відмінність tidyverse i base R? 1. Інший синтаксис

output <- processing(input) VS input |> processing() -> output

1. Спрощені команди
2. Відмінні назви функцій з однаковим призначенням
3. Запис декількох команд в одну строку

Чому ми почали з base R, якщо майже весь курс на tidyverse? “Вступ до R” має на меті підготувати вас не тільки до транскриптомного аналізу, але і озброїти знаннями для вашого розвитку у різноманітних напрямках програмування.

#install Tidyverse  
library(tidyverse)  
  
#if problems arise, load dplyr and readr  
library(readr)  
library(dplyr)

Зверніть увагу: якщо у вас виникли проблеми з інсталяцією *tidyverse*, блоки коду, які вимагають функцій поза завантаженими на початку семінару, позначені *#!*.

Всесвіт пакетів tidyverse - це набір пакетів, спеціально орієнтованих на data science.

#!  
# Display list of tidyverse packages.  
# Note, ggplot2 is also part of tidyverse  
tidyverse\_packages()

Оператор pipe **%>%** (з пакета *dplyr*) передає вихідні дані функції, застосовані до першого аргументу наступній функції. Такий спосіб об’єднання функцій дозволяє об’єднати декілька кроків одночасно, виконувати послідовні завдання.

#Pass vector c(1,2,3,4,5) to the function mean()  
a <- 1:5 %>% mean()   
a

Пакет *readr* спрощує читання або запис кількох форматів файлів за допомогою функцій, які починаються з read\_\* або write\_*. У порівнянні з R Base функції* readr\* є швидшими.

# mtcars.csv file we created above (see "Save/write and load/read data" section).  
my\_file <- read\_tsv("mtcars.tsv")

# Analogically to base R, read\_tsv reads .tsv files.  
write\_csv(my\_file, "my\_file.csv")

# Write data frame to file.

Імпортовані таблиці належать до класу **tibble**, сучасної версії data.frame з пакету *tibble*. Він ніколи не змінює тип вхідних даних (тобто більше не треба прописувати stringsAsFactors = FALSE).

matrix2 <- matrix(c(1,2,3,4), ncol = 2)  
as\_tibble(matrix2)

І можна використовувати колонки з списками.

vignette("tibble")

Дізнайтеся більше про tibble за допомогою vignette(“tibble”).

Пакет *dplyr* надає нам граматику обробки даних, набір корисних дієслів для вирішення типових проблем. Нижче основні функції.

#Select and rename.  
mtcars\_sel\_renm <- mtcars %>% select(mpg) %>% rename(col1=mpg)  
mtcars\_sel\_renm

#Filter and sort.  
# filter(data, condition1, condition2,...)  
# arrange(col) ascending, arrange(-col) descending  
mtcars\_fill\_sort <- filter(mtcars, cyl == 4, disp > 100) %>% arrange(qsec)  
mtcars\_fill\_sort

# Group and summarize.  
#group\_by() takes an existing tbl and converts it into a grouped tbl where operations are performed "by group"  
#summarise() creates a new data frame. It will contain one column for each grouping variable and one column for each of the summary statistics that you have specified.  
by\_cyl <- mtcars %>% group\_by(cyl)

# Mutate  
#mutate() creates new columns that are functions of existing variables. It can also modify (if the name is the same as an existing column) and delete columns (by setting their value to NULL).

###Практика: активний пошук

Знайдіть в інтернеті як виконати наступні операції через пакети tidyverse. Проведіть операцї на даних iris.

#load iris from data()

#Combine tables

#Reshaping data to the 'long' format (where class-parameter-number)

#Extract a subset of rows without duplicate values

#Find rows that appear in both data frames

#Find rows that appear in first data set and not in second

#Make a new column without retaining original

###Опціонально: додаткові посилання

1. <https://style.tidyverse.org/index.html>
2. <https://github.com/rstudio/cheatsheets/blob/main/data-visualization.pdf>

###Вивчені функції seq() generate regular numerical sequences

Sys.time() returns the system’s idea of the current date with and without time

Sys.sleep() suspend execution of R expressions for a specified time interval

data() loads specified data sets, or list the available data sets

write.table(), write.csv() prints its required argument x (after converting it to a data frame if it is not one nor a matrix) to a file

read.table() reads a file in table format and creates a data frame from it

dim() retrieve or set the dimension of an object

head(), tail() returns the first or last parts of a vector, matrix, table, data frame or function

str() compactly display the internal structure of an R object

View() invoke a spreadsheet-style data viewer on a matrix-like R object

duplicated() determines which elements of a vector or data frame are duplicates of elements, returns a logical vector

unique() returns a vector, data frame or array like x but with duplicate elements/rows removed

table() returns number of each element

order() returns a permutation which rearranges its first argument into ascending or descending order

plot()

points()

hist()

par() can be used to set or query graphical parameters

dev.off() shuts down the specified (by default the current) graph

jpeg() saves a graph in jpeg format

heatmap()

tidyverse\_packages() list all packages in the tidyverse

read\_csv(), read\_tsv() reading the most common types of flat file data

write\_csv() unlike write.csv(), these functions do not include row names as a column in the written file

tibble() constructs a data frame. It is used like base::data.frame(), but with a couple notable differences

select() select (and optionally rename) variables in a data frame

rename() changes the names of individual variables using new\_name = old\_name syntax

mutate() creates new columns that are functions of existing variables

filter() unction is used to subset a data frame, retaining all rows that satisfy your conditions

arrange()

group\_by()