Part III - Manipulating data with dplyr

- How to import and export data in R
- How to inspect imported data
- Manage files & folders
- Tips on Warning/errors
- R documentation and manuals
- aggregating and analyzing data with dplyr

dplyr library

What is **dplyr**?

-dplyr is a R package that simplifies data manipulation through additional and intuitive functions

- Different functions are available for different tasks

- dplyr can work in “PIPE” simplifying complex workflows

most useful functions:

select()
filter()
arrange()
mutate()
group_by()
summarize()
left_join()

Let’s learn first how to import data in R!

data import

Import your data in R! By the graphic interface or … by the command line (far more quick and reproducible!)

- several functions: read.csv() read.delim() **read.table()**

```
> your_dataframe <- read.table(x = "~/.txt", sep = "\t", dec = ".", header = T)
```

  - pay attention to your decimal separator (should be both "." or ",") and your field separator (must be different!)

```
> str(your_dataframe)
```

  - Check your imported table: numbers should be imported as numbers, texts as character

```
> View(your_dataframe)
```

  - visually inspect and scroll your table!

data import

Exercise! Import the Flowers dataset (Flowers.txt) and inspect it

```
# download the flowers dataset from the course repository

>download.file("https://raw.githubusercontent.com/mchialva/PhDToolbox2023/main/Datasets/flowers.tsv", "flowers.tsv")

# import dataset

> flowers<-read.table("flowers.txt")

# View table
> View(flowers)

# Check data structure
> str(Flowers)
```

dataframe content inspection

Other ways to check your imported object

```
# display only the first rows

> head(your_dataframe)

# display only the final rows

> tail(your_dataframe)

# display dimensions of your table (number of rows and columns)

> dim(your_dataframe)     # returns a vector of length 2 with rows
and columns number

> nrow(your_dataframe)         # returns the number of rows
> ncol(your_dataframe)         # returns the number of columns
```

dataframe content inspection

display descriptive statistics for each column in the dataframe

> summary(your_dataframe)

Exercise!

1) How many columns and rows are in the Flowers dataset?
2) Select species with blue flowers (COL) and stem size (SSIZE) higher than 10 and save in the new object “Flowers_blue10”.
3) How many species?

Basic dataframe handling

We already explained cbind(), rbind() and how to remove entire columns using NULL, but we can also create new columns:

```
>df$colname_4 <- 1 # create a new column full of 1s

>df$colname_5 <- ifelse(df$colname_1 == "A", 50, 120) # create a new column with dependent values

>table(df$colname_1) # very cool function to see frequencies of a factor
```

Exercise:

Create a new column in df called colname_4. We need that where in colname_3 values are higher or equal to 8 in colnames_4 there is the character “small”, in all the other case there is “bigger”.

Calculate frequencies of factors in colnames_4.

data export

- write a dataframe into a delimited text file using **write.table()** function

```
> write.table(x = df, file = "~/.txt", sep = "\t", row.name = F)
```

Exercise!

Export Flowers_blue10 object into a tsv table

data export

- write a dataframe into a delimited text file using **write.table()** function

```
> write.table(x = df, file = "~/.txt", sep = "\t", row.name = F)
```

Exercise!

Export Flowers_blue10 object into a tsv table

```
> write.table(x = Flowers_blue10, file = "Flowers_blue10.tsv", sep =
"\t", row.name = F)
```

dplyr: subset columns

Subset Variables (Columns)

select(dataframe, columns to keep)

library(dplyr)

> **select**(flowers, SSIZE, AREA, CIT) # select non contiguous columns

> **select**(flowers, -IUCN) # remove column

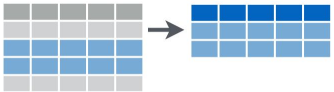
> **select**(flowers, starts_with("Q") # select columns which start with "Q"

> **select**(flowers, IUCN:NIT) # select contiguous columns

> **select**(flowers, SSIZE, everything()) # reorder columns

dplyr: subset rows

Subset Observations (Rows)



filter(dataframe, subsetting_conditions)

library(dplyr)

> **filter**(flowers, SSIZE>10) # select observations with stem size higher than 10

> **filter**(flowers, CIT==20) # select observations with 20 citations

> **filter**(flowers, SSIZE<5 | (FSIZE>50 & FLEN<2)) # multiple logical criteria

dplyr: work in pipe

Task: I want to filter and select columns at the same time

three ways to do this:

- create intermediate object
- nested functions **select**(filter(flowers, SSIZE>10), SP, starts_with("Q"))
- PIPES: Take the output of a function and send it to the next one using the **%>%** operator

> my_selection<-flowers **%>%** **filter**(SSIZE>10) **%>%** **select**(SP, starts_with("Q")

TIP: to visualize on the fly the operation you are performing on your data frame, append the View() function to your PIPE!

dplyr: work in pipe

Notes:

- pipes are made available through the package **magrittr** (installed automatically with **dplyr**)
- input data needs to be given only at the very beginning of the PIPE or in the first function
- no intermediate objects are created
- to make permanent your operations you still need to write your pipe into an object!
- there are no limits in the number of functions you can pipe

dplyr: work in pipe

Exercise!

- Filter the flowers dataset by SSIZE>10 and QRANGE from 0 to 600
- select the following columns (SP, NIT and all columns after (and including) QMIN)
- count how many columns and rows are in the subsetted dataframe
- do it with PIPES!

dplyr: work in pipe

solution

```
> flowers %>%
                filter(SSIZE>10 & QRANGE %in% seq(0:600)) %>%
                select(SP, NIT, QMIN, everything()) %>%
                dim()
[1]  8 22
```

dplyr: mutate

df %>% **mutate**(new_column_name = content)

Add a column containing the flower-to-stem ratio(FSR=FSIZE/SSIZE)

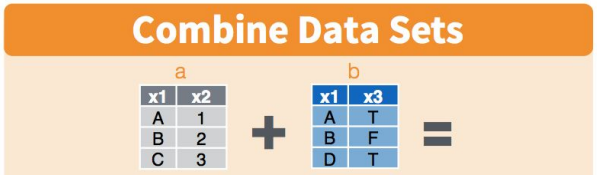
> flowers %>% **mutate**(FSR=FSIZE/SSIZE) %>% View()

add column using conditional rules with ifelse() or **if_else**(TEST, if TRUE, if FALSE)

add the column QRANGE1000 assigning to category “A” all the species with value higher than 1000 and the other to “B”

> flowers %>% **mutate**(QRANGE1000=**if_else**(QRANGE>1000, “A”, “B”)) %>% View()

dplyr: left_join



df **%>%** **left_join**(df2, index)

We need to join the flowers dataset with flowers_details.tsv table which contains additional attributes

>**download.file**("https://raw.githubusercontent.com/mchialva/PhDToolbox2023/main/Datasets/flowers_details.tsv", "flowers_details.tsv")

import table

> flowers_details<-**read.table**(“Datasets/flowers_details.tsv", header=T)

join (or merge) and save into a new object

> flowers_all<- flowers **%>%** **left_join**(flowers_details, by=”SP”)

dplyr: summarize()

df **%>%** **summarize**(new_column_name = content, ...)

Summarize min, mean and maximum SSIZE values

> flowers_all **%>%** **summarize**(min_SSIZE=**min**(SSIZE),
mean_SSIZE=**mean**(SSIZE),
max_SSIZE=**max**(SSIZE)) **%>%** View()

min_SSIZE	mean_SSIZE	max_SSIZE
1	1.5	29.40487	400

dplyr: summarize()

Task: I need to summarize values by categories.

E.g. which is the minimum and maximum flower size (FSIZE) for each plant family in the dataset?

group_by() groups by single or multiple factors

df **%>%** **group_by**() **%>%** **summarize**(new_column_name = content, ...)

Summarize min, mean and maximum SSIZE values

> flowers_all **%>%** **group_by**(FAM) **%>%** **summarize**(min_FSIZE=**min**(FSIZE),

max_FSIZE=**max**(FSIZE))

dplyr: group_by()

If called alone, the function **group_by**() does not produce any evident output (it is silently grouping your observation)

- you can summarize the number of observations in each of the grouped category by piping the function **tally()**

> flowers_all **%>%** **group_by**(FAM) **%>%** **tally**()

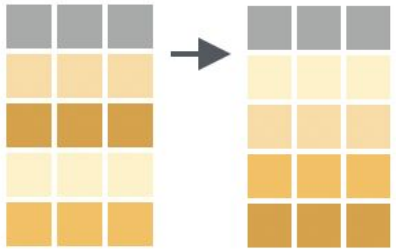
This command lists how many observations are for each Family in the dataset.

dplyr: group_by()

Tips: Some of dplyr functions coerce your output into a tibble object (*tbl* class object) which easily fits in your terminal when there are too many observations.

- you may need to convert your output into a data frame: pipe **as.data.frame**() function
- you may need to visualize more rows: pipe **print**(n=rows_to_show)

dplyr: arrange



df %>% **arrange**(columns to order by)

Order the data frame *flowers_all* by increasing FSIZE value

> flowers_all %>% **arrange**(FSIZE)

Order the dataframe flowers_all by decreasing FSIZE value

> flowers_all %>% **arrange**(-FSIZE)

if applied to character vectors the function orders in alphabetic order

> flowers_all %>% **arrange**(COL)

dplyr: exercise 1

Exercise!

- Take as input the *flowers_all* dataset
- Summarize (minimum, mean and maximum values) flower dimensions (FLEN) by color (COL), habitat (HBT) and biology (FBIO) excluding Asteraceae and Crassulaceae families
- sort observations by decreasing mean value
- How many observations are there?

TIP: If you get stuck in the pipe run your code function by function and look at the output. This will guide you on how to modify or tune the next function.

dplyr: complex exercise 1

Solution:

```
> flowers_all %>% filter(!FAM %in% c("Asteraceae", "Crassulaceae")) %>%
                  group_by(COL,HBT) %>%
                  summarize(      min=min(FLEN),
                                  mean=mean(FLEN),
                                  max=max(FLEN)       ) %>%
                  %>% arrange(-mean) %>%
                  print(n=100)
```