FDS::**CHEATSHEETS**[“project 2”]

DBI RSQLite dplyr tidyr ggplot2

EPFL EXTENSION SCHOOL

Hints

Part 1

- load the libraries
- Import the data following the workflow →
- Use janitor::**clean_names()** to clean the variable names

Part 2

- sum all the numerical variables in each tibble (you can use **sum()** over multiple variables)
- You can also already combine the datasets together and put them in long format with tidy::pivot_longer() to use the grouping options
- Use dplyr::**distinct()** to get unique values or combination of values

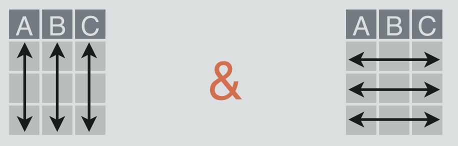
Part 3

- make sure you are working with long format tidy data
- Use ggplot2::geom_col() to create bars
- Use ggplot2::facet_wrap() to create mini plots
- Add ggplot2::labs() to annotate your chart

Part 4

- Group your data using dplyr::group_by()
- Use slice_max() to get the top value for each group

Tidy data is a way to organize tabular data in a consistent data structure across packages.
A table is tidy if:



Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own row

Check the **R help** for any function by running ?**function()** in the console: e.g., **?collect()**

Data import

Workflow:

1. Connect to the Database
2. List the tables that are included in the database
3. Establish a connection to a chosen table
4. Collect the table locally, as a tibble
5. Repeat for all wanted tables
6. Disconnect

Functions:

- DBI::**dbConnect()**
    - **dbConnect(**RSQLite::**SQLite(), “**my_database.db”)
- DBI::**dbListTables()**
- dplyr::**tbl()**
- dplyr::**collect()**
- DBI::**dbDisconnect()**

Reshaping data

pivot_longer(data, cols, names_to = "name", values_to = "value", values_drop_na = FALSE**)**

"Lengthen" data by collapsing several columns into two. Column names move to a new names_to column and values to a new values_to column.

pivot_longer(table4a, cols = 2:3, names_to ="year", values_to = "cases")

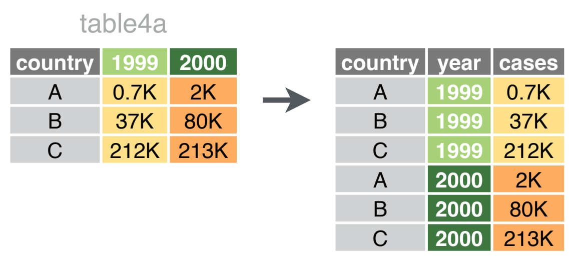


table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

Data manipulation

dplyr::**filter(**.data, …, .preserve = FALSE**)** Extract rows that meet logical criteria.
filter(mtcars, mpg > 20)

dplyr::**distinct(**.data, …, .keep_all = FALSE**)** Remove rows with duplicate values.
distinct(mtcars, gear)

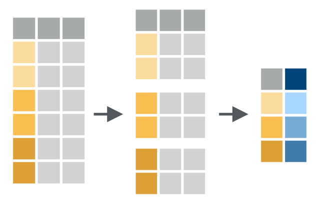
dplyr::**slice_max()** Select rows with the highest values.
slice_max(mtcars, mpg, n=1)

dplyr::**mutate()** to create new columns or transform existing ones

summarise() applies summary functions to columns to create a new table.

COUNTING GROUP SIZES

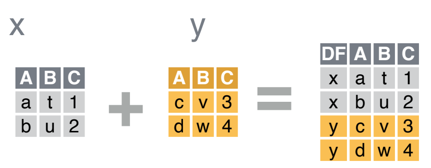
Use **group_by(**.data, …, .add = FALSE, .drop = TRUE**)** to create a "grouped" copy of a table grouped by columns in ... dplyr functions will manipulate each "group" separately and combine the results.



```
mtcars  %>%
   group_by(cyl) %>%
   summarise(n =n ())
```

Combine tables

bind_rows(…, .id = NULL**)**
Returns tables one on top of the other as a single table. Set .id to a column name to add a column of the original table names (as pictured).



x

A	B	C
a	t	1
b	u	2

+ y

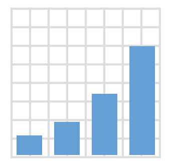
A	B	C
c	v	3
d	w	4

=

DF	A	B	C
x	a	t	1
x	b	u	2
y	c	v	3
y	d	w	4

Plot the results

Plot one discrete, one continuous variable
f <- ggplot(mpg, aes(class, hwy))



**ggplot(mpg, aes(x = class, y = hwy)) +
geom_col()**