

Introduction to Data Manipulation









Workshop Setup:

Wi-Fi

Network Name:

Password:

Resources

R (recommended: version 3.6.1 – current)

ORStudio (recommended: version 1.2.5001 – current)

New RStudio version (1.2.5001) on 19th Sept 2019!





What is tidyverse?

Tidyverse is a collection of packages that are designed for data science tasks, more specifically for data manipulation, transformation, exploration and visualisation.

These packages share a common design philosophy and contain functions that are consistent and uniform in coding style.

You can read more at https://www.tidyverse.org/





Topics

Workshop aim:

Learn how to do data manipulations using tidyverse packages.

- ► Topics:



Learn the "verbs" with
Improve your workflow with



Simple string manipulation using





Learn the "verbs" with



{dplyr} is one of the most commonly used R packages when dealing with data manipulations. It is very powerful in handling tabular data such as data frames and is easy to use through "verb" functions. You can use {dplyr} to:

- Select columns from your data
- Filter your data to keep the rows that meet some conditions
- Arrange your data in some order
- Mutate your data and create new columns
- Group and summarise your data

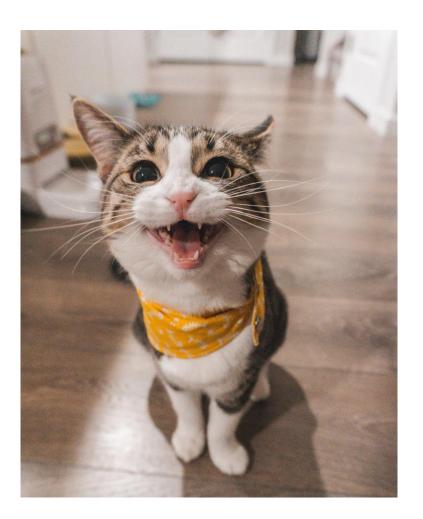
library(dplyr)

View(starwars)









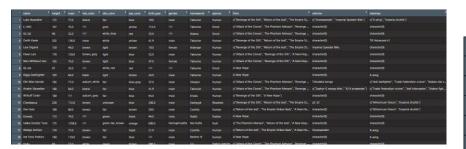
Use the starwars dataset to:

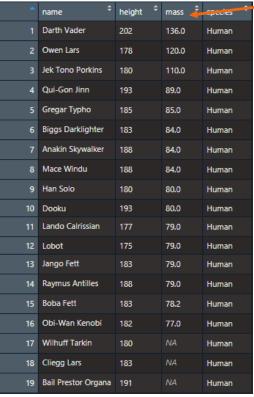
- 1. Select the columns: "name", "height", "mass", "species".
- 2. Filter the rows to keep only those characters that are taller than 175cm.
- 3. Filter the rows to keep only the human characters
- 4. Arrange the rows according to descending "mass" values.
- 5. Who is the character on the first row?





selected columns





arranged rows

Who is the character on the first row?







Live Coding Example 1 (/>

```
# Select columns
df <- select(starwars, name, height, mass, species)</pre>
# Filter rows by height condition
df <- filter(df, height >= 175)
# Filter rows by species condition
df <- filter(df, species == "Human")</pre>
# Arrange rows by descending mass
df <- arrange(df, desc(mass))</pre>
```



Mutate your data

Very often you will want to create new columns from your existing data. The function **mutate** in the {dplyr} package can be used to do exactly this task.

You can actually create multiple columns in a single function call.

```
library(dplyr)

# Create a column for height in meters

df <- mutate(starwars, height_m = height/100)</pre>
```





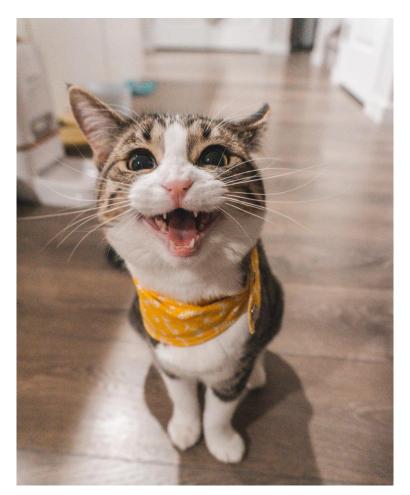
Group and summarise your data

Another very common task is to group your data by a column (or more than one column) and then create summarised values for the grouped data. The functions **group_by** and **summarise** in the {dplyr} package make it very easy to do these transformations.





Live Coding Example 2 (/>



Use the starwars dataset to:

- 1. Remove the columns "films", "vehicles", "starships" from the data. ••
- 2. Remove rows that have missing mass values.
- Calculate the Body Mass Index (BMI) for each character*.
- 4. Arrange the rows by descending BMI ... who do you think is at the top?
- 5. Find the median BMI value for each gender category.

*Tip: BMI = weight (kg) / height² (m)



Use minus sign "-" to remove columns

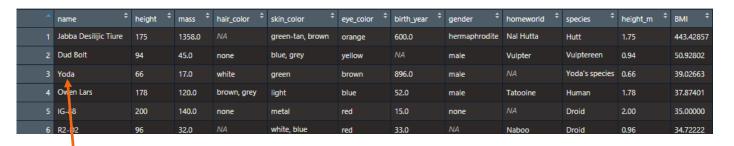




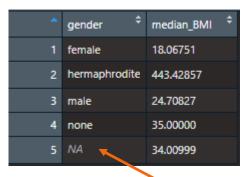
Live Coding Example 2 (//>

Who is the character with the highest BMI?









Investigate data quality





```
# Select columns
df <- select(starwars, -films, -vehicles, -starships)</pre>
# Filter rows that have missing mass
df <- filter(df, !is.na(mass))</pre>
# Create columns of: height in meters and the Body Mass Index (BMI) value
df \leftarrow mutate(df, height_m = height/100, BMI = mass / (height_m)^2)
# Arrange rows according to descending "BMI" values
df <- arrange(df, desc(BMI))</pre>
# Calculate the median BMI value for each gender
df <- summarise(group_by(df, gender), median_BMI = median(BMI))</pre>
```





Summary of {dplyr} "verb" functions

Function	Description
select	Select columns by name
filter	Filter rows that meet a condition
arrange	Arrange rows to some order
mutate	Mutate data to create new columns
group_by	Group data by columns
summarise	Summarise data to values



Improve your workflow with



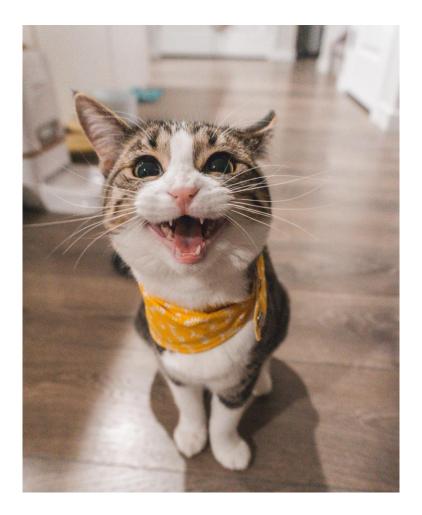
{magrittr} is one of a few packages that have changed the way we write R code. It has significantly improved the readability and workflow of code by introducing the "pipe" operator. It acts as a "then" operation where we can pass data from one function to another function very easily.

Fun fact: The package name is inspired by the famous artist René Magritte. One of his work, a pipe, has the text "this is not a pipe" as a caption ... this is where the {magrittr} package gets its image.

library(magrittr)







Repeat example 1 using the pipe operator from the {magrittr} package.

1. Select the columns: "name", "height",
 "mass", "species" THEN filter the rows to
 keep only those characters that are taller
 than 175cm THEN filter the rows to keep
 only the human characters THEN
 arrange the rows according to
 descending "mass" values.





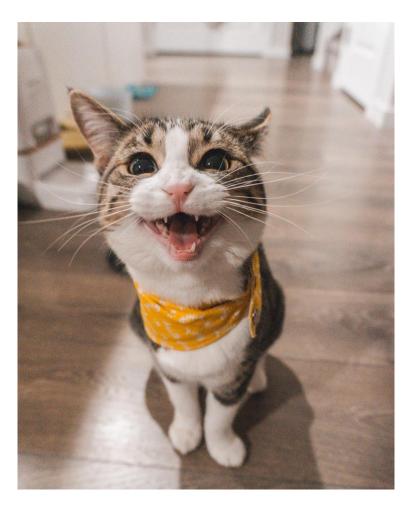
```
library(dplyr)
library(magrittr)
# Pipe each data manipulation operation to the next one
df <- starwars %>%
  select(name, height, mass, species) %>%
  filter(height >= 175) %>%
  filter(species == "Human") %>%
  arrange(desc(mass))
```

Try CTRL+SHIFT+M (Windows) and see what happens





Live Coding Example 4 (//>



Repeat example 2 using the pipe operator from the {magrittr} package.

Remove the columns "films", "vehicles", "starships" from the data THEN remove rows that have missing mass values
 THEN calculate the Body Mass Index (BMI) for each character THEN arrange the rows by descending BMI THEN find the median BMI value for each gender category.





```
library(dplyr)
library(magrittr)
# Pipe each data manipulation operation to the next one
df <- starwars %>%
  select(-films, -vehicles, -starships) %>%
  filter(!is.na(mass)) %>%
  mutate(height_m = height/100,
         BMI = mass / (height_m)^2) %>%
  arrange(desc(BMI)) %>%
  group_by(gender) %>%
  summarise(median_BMI = median(BMI))
```



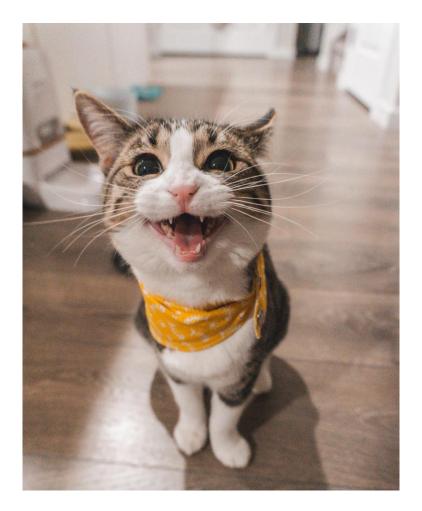
Simple string manipulation using



{stringr} is a package in the tidyverse collection that helps us do data manipulations involving strings. This is another common task, especially in data cleaning. Here are some examples:







Use the starwars dataset to:

- Transform the character names to upper case.
- 3. Create an indicator for the rows where characters have green skin.

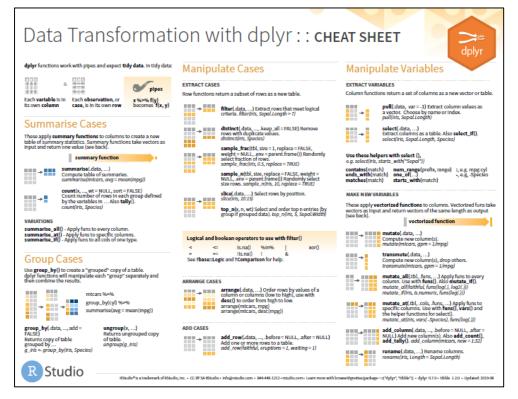


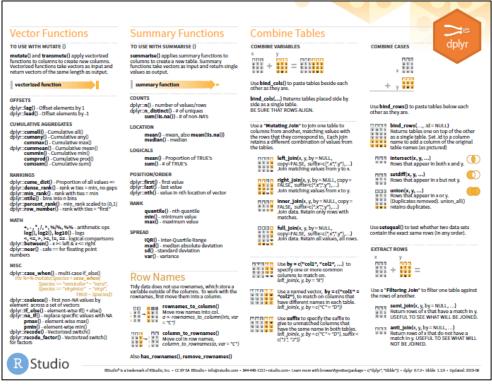






Other resources – {dplyr} cheat sheet





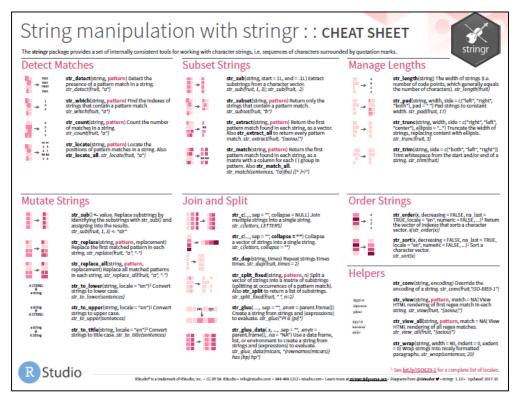


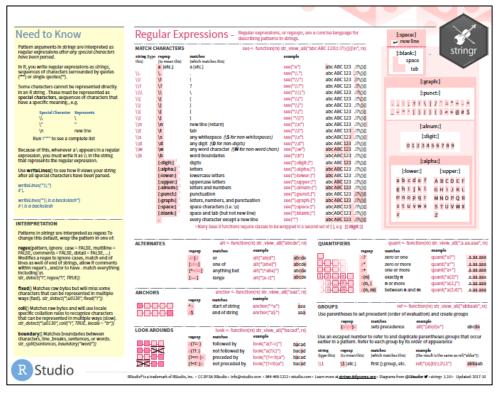
Get the cheat sheet at: https://rstudio.com/resources/cheatsheets/





Other resources – {stringr} cheat sheet







Get the cheat sheet at: https://rstudio.com/resources/cheatsheets/



Thank you to our sponsors and partners!



