

INTRODUCTION TO R PROGRAMMING

Day 3: Data wrangling

October 26, 2023



Workshop breakdown

Session 1: Lecture	11:00 a.m. – noon
Break	
Session 2: Interactive	2:00 p.m. – 4:00 p.m.



Course website: <https://stanley-manne-childrens-research.github.io/introR/>



Code repository: <https://github.com/Stamley-Manne-Childrens-Research/introR>

Recap of R Basics (Day 1)

- ❖ R is a programming language for data analysis
- ❖ Values of different types can be stored as variables
- ❖ Variables may have different structures (vector, matrix, data frame)
- ❖ Data import in R is simple with and without code
- ❖ Functions can be used to select columns, create new columns, and filter data
- ❖ Simple visualizations can be created with basic plotting functions



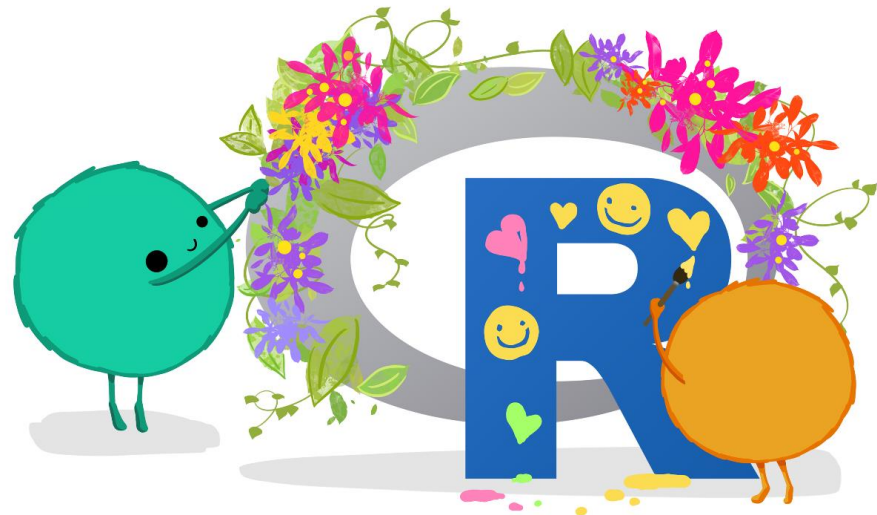


Day 2: Data wrangling



Learning objectives

- ❖ Review importing data
- ❖ Be aware of the tidyverse
- ❖ Use basic functions of **dplyr** to wrangle data sets.



Messy vs. clean data

color

merged cells

multiple column titles

	ReplicateA		ReplicateB	
Sample	Day1	Day2	Day1	Day2
Control	4	40	6	48
Treatment	4	7	3	5

“Tidy” data

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

each row an observation

Messy vs. clean data

	ReplicateA		ReplicateB	
Sample	Day1	Day2	Day1	Day2
Control	4	40	6	48
Treatment	4	7	3	5

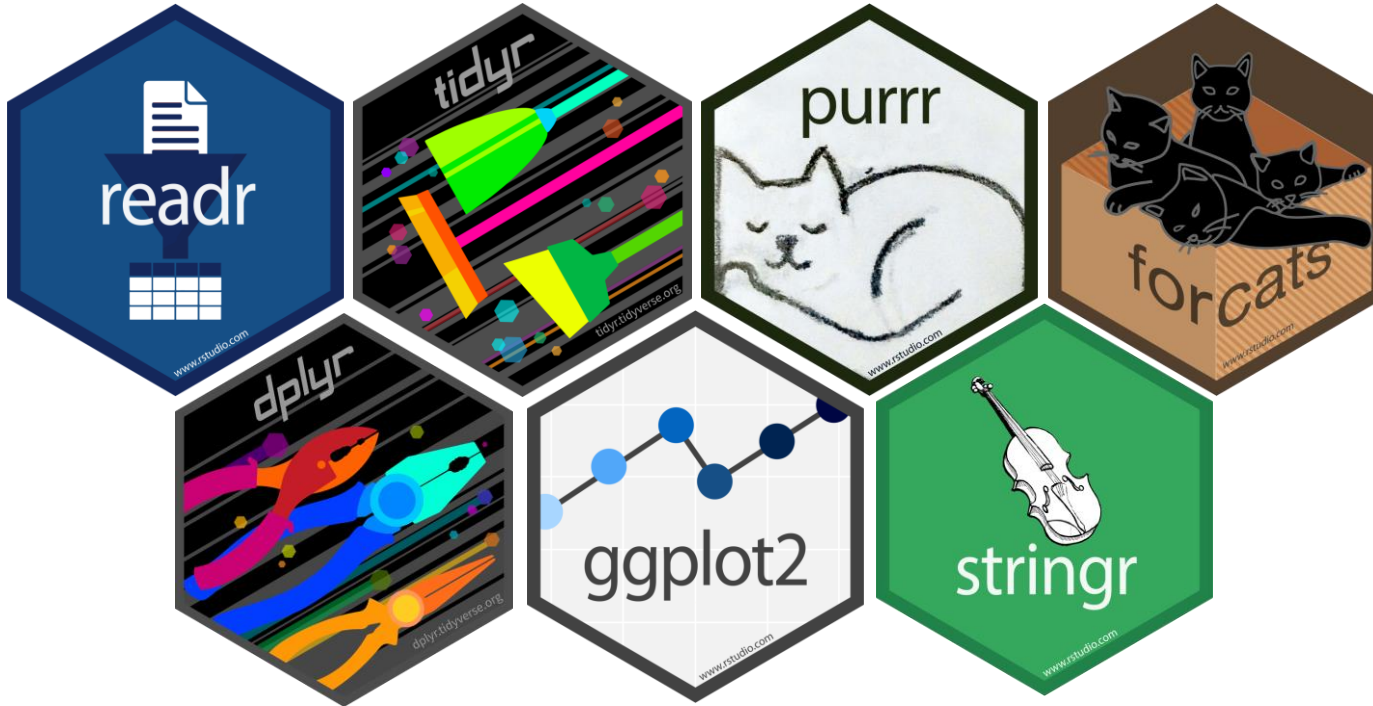
Sample	Value	Replicate	Day
Control	4	A	1
Control	40	A	2
Control	6	B	1
Control	48	B	2
Treatment	4	A	1
Treatment	7	A	2
Treatment	3	B	1
Treatment	5	B	2

What is the Tidyverse?

- ❖ **A collection of packages** designed to facilitate data science
- ❖ Intended to make data scientists more productive by guiding them through workflow
- ❖ Great for data manipulation, exploration, and visualization



Tidyverse packages

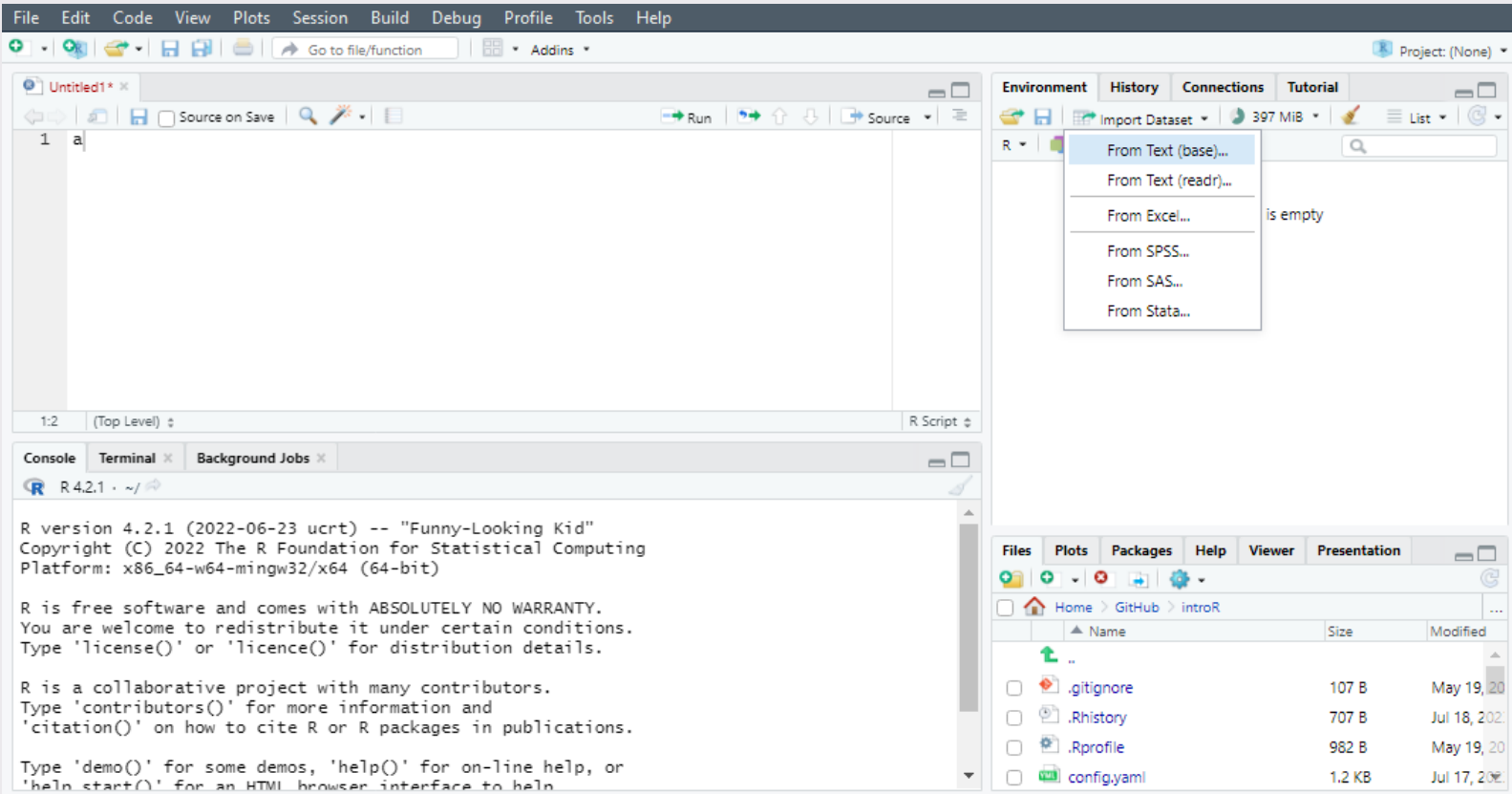


```
library(tidyverse)
```



readr for reading flat files

Importing data using the environment window



The screenshot displays the RStudio application window. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu bar is a toolbar with icons for file operations and a 'Go to file/function' search bar. The main editor window shows a script titled 'Untitled1*' with a single line of code: `1 a`. The status bar at the bottom of the editor indicates '1:2 (Top Level)' and 'R Script'.

The Environment window on the right side of the interface has a dropdown menu open, showing options for importing data: 'From Text (base)...', 'From Text (readr)...', 'From Excel...', 'From SPSS...', 'From SAS...', and 'From Stata...'. The Environment window also shows a memory usage of 397 MiB and a search bar.

The Console window at the bottom left displays the R startup message for version 4.2.1 (2022-06-23 ucrt) on a 64-bit platform. The message includes the R Foundation's copyright notice and instructions on how to use R, such as using `license()` for distribution details, `contributors()` for more information, and `demo()` for some demos.

The Files window at the bottom right shows the file explorer for the 'introR' directory. It lists several files: `..`, `.gitignore` (107 B, May 19, 2022), `.Rhistory` (707 B, Jul 18, 2022), `.Rprofile` (982 B, May 19, 2022), and `config.yaml` (1.2 KB, Jul 17, 2022).

Introduction: readr

- ❖ A collection of functions to import/export your data quickly and efficiently

Importing data using code

Code Preview:

```
library(readxl)  
chicago_temps <- read_excel("datasets/chicago_temps  
.xlsx")  
View(chicago_temps)
```

- ❖ readxl is an R package

Code Preview:

```
library(readr)  
dataset <- read_csv(NULL)  
View(dataset)
```

To import data:

```
readr::read_csv()
```

```
readr::read_tsv()
```

To export data:

```
readr::write_csv()
```

```
readr::write_tsv()
```



dplyr for wrangling data

Introduction: `dplyr`

Collection of functions as **verbs** to easily describe what you want to do with your data

- ❖ `select()` to keep columns based on names
- ❖ `filter()` to keep rows based on values
- ❖ `rename()` to give columns new names
- ❖ `mutate()` to add new (or change existing) columns
- ❖ `group_by()` to group rows by columns
- ❖ `summarize()` to condense multiple columns
- ❖ `xxx_join()` to combine datasets



Starwars dataset

variables

data(starwars)

	name	height	mass	hair_color	skin_color	eye_color	birth_year	sex	gender	homeworld	species	films	vehicles	starships
1	Luke Skywalker	172	77.0	blond	fair	blue	19.0	male	masculine	Tatooine	Human	c("The Empire Strikes Back", "Revenge of the Sith" [...])	c("Snowspeeder", "Imperial Speeder Bike")	c("X-wing", "Imperial shuttle")
2	C-3PO	167	75.0	NA	gold	yellow	112.0	none	masculine	Tatooine	Droid	c("The Empire Strikes Back", "Attack of the Clones [...])	character(0)	character(0)
3	R2-D2	96	32.0	NA	white, blue	red	33.0	none	masculine	Naboo	Droid	c("The Empire Strikes Back", "Attack of the Clones [...])	character(0)	character(0)
4	Darth Vader	202	136.0	none	white	yellow	41.9	male	masculine	Tatooine	Human	c("The Empire Strikes Back", "Revenge of the Sith" [...])	character(0)	TIE Advanced x1
5	Leia Organa	150	49.0	brown	light	brown	19.0	female	feminine	Alderaan	Human	c("The Empire Strikes Back", "Revenge of the Sith" [...])	Imperial Speeder Bike	character(0)
6	Owen Lars	178	120.0	brown, grey	light	blue	52.0	male	masculine	Tatooine	Human	c("Attack of the Clones", "Revenge of the Sith", " [...])	character(0)	character(0)
7	Beru Whitesun Lars	165	75.0	brown	light	blue	47.0	female	feminine	Tatooine	Human	c("Attack of the Clones", "Revenge of the Sith", " [...])	character(0)	character(0)
8	R5-D4	97	32.0	NA	white, red	red	NA	none	masculine	Tatooine	Droid	A New Hope	character(0)	character(0)
9	Biggs Darklighter	183	84.0	black	light	brown	24.0	male	masculine	Tatooine	Human	A New Hope	character(0)	X-wing
10	Obi-Wan Kenobi	182	77.0	auburn, white	fair	blue-gray	57.0	male	masculine	Stewjon	Human	c("The Empire Strikes Back", "Attack of the Clones [...])	Tribubble bongo	c("Jedi starfighter", "Trade Federation cruiser", " [...])
11	Anakin Skywalker	188	84.0	blond	fair	blue	41.9	male	masculine	Tatooine	Human	c("Attack of the Clones", "The Phantom Menace", "R [...])	c("Zephyr-G swoop bike", "XJ-6 airspeeder")	c("Trade Federation cruiser", "Jedi Interceptor", " [...])
12	Wilhuff Tarkin	180	NA	auburn, grey	fair	blue	64.0	male	masculine	Eriadu	Human	c("Revenge of the Sith", "A New Hope")	character(0)	character(0)
13	Chewbacca	228	112.0	brown	unknown	blue	200.0	male	masculine	Kashyyyk	Wookiee	c("The Empire Strikes Back", "Revenge of the Sith" [...])	AT-ST	c("Millennium Falcon", "Imperial shuttle")
14	Han Solo	180	80.0	brown	fair	brown	29.0	male	masculine	Corellia	Human	c("The Empire Strikes Back", "Return of the Jedi", " [...])	character(0)	c("Millennium Falcon", "Imperial shuttle")
15	Greedo	173	74.0	NA	green	black	44.0	male	masculine	Rodia	Rodian	A New Hope	character(0)	character(0)
16	Jabba Desilijic Tiure	175	1358.0	NA	green-tan, brown	orange	600.0	hermaphroditic	masculine	Nal Hutta	Hutt	c("The Phantom Menace", "Return of the Jedi", " A N [...])	character(0)	character(0)
17	Wedge Antilles	170	77.0	brown	fair	hazel	21.0	male	masculine	Corellia	Human	c("The Empire Strikes Back", "Return of the Jedi", " [...])	Snowspeeder	X-wing
18	Jek Tono Porkins	180	110.0	brown	fair	blue	NA	male	masculine	Bestine IV	Human	A New Hope	character(0)	X-wing
19	Yoda	66	17.0	white	green	brown	896.0	male	masculine	NA	Yoda's species	c("The Empire Strikes Back", "Attack of the Clones [...])	character(0)	character(0)
20	Palpatine	170	75.0	grey	pale	yellow	82.0	male	masculine	Naboo	Human	c("The Empire Strikes Back", "Attack of the Clones [...])	character(0)	character(0)
21	Boba Fett	183	78.2	black	fair	brown	31.5	male	masculine	Kamino	Human	c("The Empire Strikes Back", "Attack of the Clones [...])	character(0)	Slave 1
22	IG-88	200	140.0	none	metal	red	15.0	none	masculine	NA	Droid	The Empire Strikes Back	character(0)	character(0)
23	Bossk	190	113.0	none	green	red	53.0	male	masculine	Trandosha	Trandoshan	The Empire Strikes Back	character(0)	character(0)
24	Lando Calrissian	177	79.0	black	dark	brown	31.0	male	masculine	Socorro	Human	c("The Empire Strikes Back", "Return of the Jedi")	character(0)	Millennium Falcon
25	Lobot	175	79.0	none	light	blue	37.0	male	masculine	Bespin	Human	The Empire Strikes Back	character(0)	character(0)
26	Ackbar	180	83.0	none	brown mottle	orange	41.0	male	masculine	Mon Cala	Mon Calamari	c("Return of the Jedi", "The Force Awakens")	character(0)	character(0)
27	Mon Mothma	150	NA	auburn	fair	blue	48.0	female	feminine	Chandrilra	Human	Return of the Jedi	character(0)	character(0)
28	Aniel Crynyd	NA	NA	brown	fair	brown	NA	male	masculine	NA	Human	Return of the Jedi	character(0)	A-wing

observations

values


```
dplyr::select()
```

to keep columns based on names



Basic operations on data:

Selecting column(s)

A data frame named `2021_temp`:

	month	day	year	temp_F
1	January	1	2021	23
2	January	2	2021	14
3	January	3	2021	18
...				
363	December	29	2021	34
364	December	30	2021	36
365	December	31	2021	32

To single out the **month** column

```
2021_temp[, 1]
```

row

column

To single out the **month** column

```
2021_temp$month
```

name of column

To single out **month**, **day**, and **temp_F**

```
2021_temp[, c(month, day, temp_F)]
```

`dplyr::select()`

Objective: select **name** and **homeworld** columns

```
library(dplyr)
```

Load in data

```
data(starwars)
```

```
starwars[, c(1, 10)]
```

Select using base R

```
starwars[, c("name", "homeworld")]
```

```
dplyr::select(dataframe, columns_to_keep)
```

```
dplyr::select(starwars, name, homeworld)
```

Select using
`dplyr::select()`

```
starwars2 <- dplyr::select(starwars, -gender)
```

```
dplyr::filter()
```

to keep rows based on values



Basic operations on data: Subsetting data

A data frame named `2021_temp`:

	month	day	year	temp_F
1	January	1	2021	23
2	January	2	2021	14
3	January	3	2021	18
...				
363	December	29	2021	34
364	December	30	2021	36
365	December	31	2021	32

To select rows where **temp_F** < 20

```
2021_temp[2021_temp$temp_F < 20,]
```

row

column

	month	day	year	temp_F
1	January	2	2021	14
2	January	3	2021	18
3	January	4	2021	19
...				
53	December	17	2021	10
54	December	18	2021	12
55	December	22	2021	15

`dplyr::filter()`

Objective: Filter to keep rows with **height > 150**

```
starwars[starwars$height > 150,]
```

Filter using base R

```
dplyr::filter(dataframe, condition(s))
```

Filter using
`dplyr::filter()`

```
dplyr::filter(starwars, height > 150)
```

```
dplyr::filter(starwars, species == "Human")
```

```
dplyr:: filter(starwars, species == "Human", height > 150)
```

pipe (%>%)

for “piping” data from left to right



pipe (%>%)

Objective: Filter to include only **Humans** with **height > 150**

Without pipes

```
dplyr::filter(starwars, species == "Human", height > 150)
```

With pipes

```
starwars %>%  
  dplyr::filter(species == "Human") %>%  
  dplyr::filter(height > 150)
```

```
starwars %>%  
  dplyr::filter(species == "Human", height > 150) %>%  
  dplyr::select(name, species, height)
```



Combine `filter()` and `select()`

```
dplyr::filter(dataframe, condition(s))
```

```
dplyr::select(dataframe, columns_to_keep)
```

Challenge!

1. Keep height greater than 100
2. Keep species: human
3. Remove column: vehicles
4. Keep columns: name, homeworld, height, species



Combine `filter()` and `select()`

Challenge!

1. Keep height greater than 100
2. Keep species: human
3. Remove column: vehicles
4. Keep columns: name, homeworld, height, species

name	homeworld	height	species
Luke Skywalker	Tatooine	172	Human
Darth Vader	Tatooine	202	Human
Leia Organa	Alderaan	150	Human
Owen Lars	Tatooine	178	Human
Beru Whitesun lars	Tatooine	165	Human
Biggs Darklighter	Tatooine	183	Human

```
starwars %>%  
  dplyr::filter(height > 100) %>%  
  dplyr::filter(species == "Human") %>%  
  dplyr::select(-vehicles) %>%  
  dplyr::select(name, homeworld, height, species)
```

```
dplyr::rename()
```

to give columns new names



dplyr::rename()

```
dplyr::rename(dataframe, new_name = old_name)
```

Objective: Rename **name** column to **character**.

```
dplyr::rename(starwars, character = name)
```

```
starwars %>%  
  dplyr::rename(character = name)
```

	name	height	mass
1	Luke Skywalker	172	77.0
2	C-3PO	167	75.0
3	R2-D2	96	32.0
4	Darth Vader	202	136.0
5	Leia Organa	150	49.0
6	Owen Lars	178	120.0



	character	height	mass
1	Luke Skywalker	172	77.0
2	C-3PO	167	75.0
3	R2-D2	96	32.0
4	Darth Vader	202	136.0
5	Leia Organa	150	49.0
6	Owen Lars	178	120.0

`dplyr::mutate()`

to add new (or change existing) columns



dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

Objective: Calculate BMI of all starwars characters

$$\text{BMI} = \text{weight (kg)} / [\text{height (m)}]^2$$

cm

	name	height	mass	hair_color	skin_color	eye_color
1	Luke Skywalker	172	77.0	blond	fair	blue
2	C-3PO	167	75.0	NA	gold	yellow
3	R2-D2	96	32.0	NA	white, blue	red
4	Darth Vader	202	136.0	none	white	yellow

1. Convert height in cm to height in m

dplyr::mutate()

```
dplyr::mutate(dataframe, new_column = expression)
```

Objective: Calculate BMI of all starwars characters

$$\text{BMI} = \text{weight (kg)} / [\text{height (m)}]^2$$

1. Convert height in cm to height in m

2. Calculate BMI as new column

```
new_starwars <- starwars %>%  
  dplyr::mutate(height_m = height/100) %>%  
  dplyr::mutate(bmi = mass/height_m^2)
```

name	height_m	mass	bmi
Luke Skywalker	1.72	77.0	26.02758
C-3PO	1.67	75.0	26.89232
R2-D2	0.96	32.0	34.72222
Darth Vader	2.02	136.0	33.33007
Leia Organa	1.50	49.0	21.77778

Combine `dplyr` functions

```
dplyr::filter(dataframe, condition(s))
```

```
dplyr::rename(dataframe, new_name = old_name)
```

```
dplyr::select(dataframe, columns_to_keep)
```

```
dplyr::mutate(dataframe, new_column = expression)
```

Challenge!

1. Rename "height" column to "height_cm"
2. Calculate new column "height_m"
3. Keep height greater than 1 meter
4. Keep all columns: name -> species

Combine `dplyr` functions

Challenge!

1. Rename "height" column to "height_cm"
2. Calculate new column "height_m"
3. Keep height greater than 1 meter
4. Keep all columns: name -> species

```
starwars %>%  
  dplyr::rename(height_cm = height) %>%  
  dplyr::mutate(height_m = height_cm/100) %>%  
  dplyr::filter(height_m > 1) %>%  
  dplyr::select(name:species)
```

```
dplyr::group_by()
```

to group rows by columns



`dplyr::group_by()`

```
dplyr::group_by(dataframe, column_to_group_by)
```



Doesn't change how the data looks, changes how the data interacts with other dplyr verbs

Objective: Group starwars data frame by gender

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender)
```

```
> is.grouped_df(grouped_starwars)  
[1] TRUE  
> is.grouped_df(starwars)  
[1] FALSE
```

```
dplyr::summarize()  
dplyr::summarise()
```

to condense multiple columns



dplyr::summarize()

```
dplyr::summarize(dataframe, new_column = expression)
```



summarize() is similar to mutate() but only keeps grouped columns

Objective: Calculate average height per gender.

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender) %>%  
  dplyr::summarize(avg_height = mean(height))
```

	gender	avg_height
1	feminine	NA
2	masculine	NA
3	NA	NA

??

?mean

dplyr::summarize()

```
dplyr::summarize(dataframe, new_column = expression)
```

Arithmetic Mean

Description

Generic function for the (trimmed) arithmetic mean.

Usage

```
mean(x, ...)  
  
## Default S3 method:  
mean(x, trim = 0, na.rm = FALSE, ...)
```

Arguments

x An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.

trim the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of `trim` outside that range are taken as the nearest endpoint.

na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.

... further arguments passed to or from other methods.



`dplyr::summarize()`

```
dplyr::summarize(dataframe, new_column = expression)
```



`summarize()` is similar to `mutate()` but only keeps grouped columns

Objective: Calculate average height **by gender.**

```
grouped_starwars <- starwars %>%  
  dplyr::group_by(gender) %>%  
  dplyr::summarize(avg_height = mean(height, na.rm = TRUE))
```

	gender	avg_height
1	feminine	164.6875
2	masculine	176.5161
3	NA	181.3333

dplyr::summarize()



summarize() is similar to mutate() but only keeps grouped columns

Compare **mutate()** and **summarize()** to calculate average height by gender

dplyr::summarize()

gender	avg_height
feminine	164.6875
masculine	176.5161
NA	181.3333

dplyr::mutate()

name	gender	avg_height
Luke Skywalker	masculine	176.5161
C-3PO	masculine	176.5161
R2-D2	masculine	176.5161
Darth Vader	masculine	176.5161
Leia Organa	feminine	164.6875
Owen Lars	masculine	176.5161
Beru Whitesun lars	feminine	164.6875

...


```
dplyr::xxx_join()
```

to combine datasets



dplyr::xxx_join()

```
dplyr::xxx_join(dataframe1, dataframe2, by = column_in_common)
```

left_join()
right_join()
full_join()
inner_join()

"Mutating joins"

Add columns, matching rows based on keys

semi_join()
anti_join()

"Filtering joins"

Filters rows based on the presence or absence of matches

dplyr::xxx_join()

```
dplyr::xxx_join(dataframe1, dataframe2, by = column_in_common)
```

df1

	A	B	C
1	red	2	3
2	orange	4	6
3	yellow	8	9
4	green	0	0
5	indigo	3	3
6	blue	1	1
7	purple	5	5
8	white	8	2


df2

	A	D
1	red	3
2	orange	5
3	yellow	7
4	green	1
5	indigo	3
6	blue	6
7	pink	9

dplyr::left_join()

```
dplyr::left_join(dataframe1, dataframe2, by = column_in_common)
```

	A	B	C
1	red	2	3
2	orange	4	6
3	yellow	8	9
4	green	0	0
5	indigo	3	3
6	blue	1	1
7	purple	5	5
8	white	8	2



	A	D
1	red	3
2	orange	5
3	yellow	7
4	green	1
5	indigo	3
6	blue	6
7	pink	9

```
dplyr::left_join(df1, df2, by = "A")
```

`dplyr::left_join()`

df1

	A	B	C
1	red	2	3
2	orange	4	6
3	yellow	8	9
4	green	0	0
5	indigo	3	3
6	blue	1	1
7	purple	5	5
8	white	8	2

df2

	A	D
1	red	3
2	orange	5
3	yellow	7
4	green	1
5	indigo	3
6	blue	6
7	pink	9



`dplyr::left_join(df1, df2)`

	A	B	C	D
1	red	2	3	3
2	orange	4	6	5
3	yellow	8	9	7
4	green	0	0	1
5	indigo	3	3	3
6	blue	1	1	6
7	purple	5	5	NA
8	white	8	2	NA

- Return all rows from df1
- Return all columns from df1 and df2
- Rows in df1 with no match in df2 will be returned as NA

`dplyr::right_join()`

```
dplyr::right_join(dataframe1, dataframe2, by = column_in_common)
```

df1

	A	B	C
1	red	2	3
2	orange	4	6
3	yellow	8	9
4	green	0	0
5	indigo	3	3
6	blue	1	1
7	purple	5	5
8	white	8	2



df2

	A	D
1	red	3
2	orange	5
3	yellow	7
4	green	1
5	indigo	3
6	blue	6
7	pink	9

```
dplyr::right_join(df1, df2)
```

dplyr::right_join()

df1

	A	B	C
1	red	2	3
2	orange	4	6
3	yellow	8	9
4	green	0	0
5	indigo	3	3
6	blue	1	1
7	purple	5	5
8	white	8	2



df2

	A	D
1	red	3
2	orange	5
3	yellow	7
4	green	1
5	indigo	3
6	blue	6
7	pink	9

```
dplyr::right_join(df1, df2)
```

	A	B	C	D
1	red	2	3	3
2	orange	4	6	5
3	yellow	8	9	7
4	green	0	0	1
5	indigo	3	3	3
6	blue	1	1	6
7	pink	NA	NA	9

- Return all rows from df2
- Return all columns from df1 and df2
- Rows in df2 with no match in df1 will be returned as NA

dplyr::full_join()

```
dplyr::full_join(dataframe1, dataframe2, by = column_in_common)
```

df1

	A	B	C
1	red	2	3
2	orange	4	6
3	yellow	8	9
4	green	0	0
5	indigo	3	3
6	blue	1	1
7	purple	5	5
8	white	8	2

↔

df2

	A	D
1	red	3
2	orange	5
3	yellow	7
4	green	1
5	indigo	3
6	blue	6
7	pink	9

```
dplyr::full_join(df1, df2)
```


dplyr::right_join()

df1

	A	B	C
1	red	2	3
2	orange	4	6
3	yellow	8	9
4	green	0	0
5	indigo	3	3
6	blue	1	1
7	purple	5	5
8	white	8	2

df2

	A	D
1	red	3
2	orange	5
3	yellow	7
4	green	1
5	indigo	3
6	blue	6
7	pink	9



dplyr::right_join(df1, df2)

	A	B	C	D
1	red	2	3	3
2	orange	4	6	5
3	yellow	8	9	7
4	green	0	0	1
5	indigo	3	3	3
6	blue	1	1	6
7	purple	5	5	NA
8	white	8	2	NA
9	pink	NA	NA	9

- Return all rows from dataframe1 and data frame2
- Return all columns from dataframe1 and dataframe2
- Where there are not matching values, return NA

Recap

❖ Rows:

- `filter()` chooses rows based on column values.

❖ Columns:

- `select()` changes whether or not a column is included.
- `rename()` changes the name of columns.
- `mutate()` changes the values of columns and creates new columns.

❖ Groups of rows:

- `summarize()` collapses a group into a single row.

❖ Join data frames using `xxx_join()`

❖ Read/write data with `readr`

dplyr : go wrangling





Questions?

