

# R intensive

Session 1

May 14<sup>th</sup>, 2024

- A **HUGE** thank you to Sarah Lewis, who is the ICTR program manager and an incredible resource in the planning of this workshop.
- Also, a big thanks to Stephanie Puff who worked with you all when you were registering.

# Welcome!

Question: How many people have experience with coding or analysis in some software package?

- EXCEL
- SPSS
- STATA
- SAS
- PYTHON
- MATLAB
- R

# The goals of this workshop

- This is an introductory workshop!
- We want you to walk away today feeling comfortable:
  1. navigating the four panels of Rstudio
  2. importing data into R
  3. performing some data manipulations and cleaning using dplyr
  4. summarizing your data
  5. plotting your data

**Like any language, you need to practice a lot to get fluent!**

# Why R?

- Cleaning, analyzing, plotting all in one place
- **Reproducibility** – you can save your code and know exactly what you did. You can also rerun code if you get new data
- It's freely available

# A few things to know first

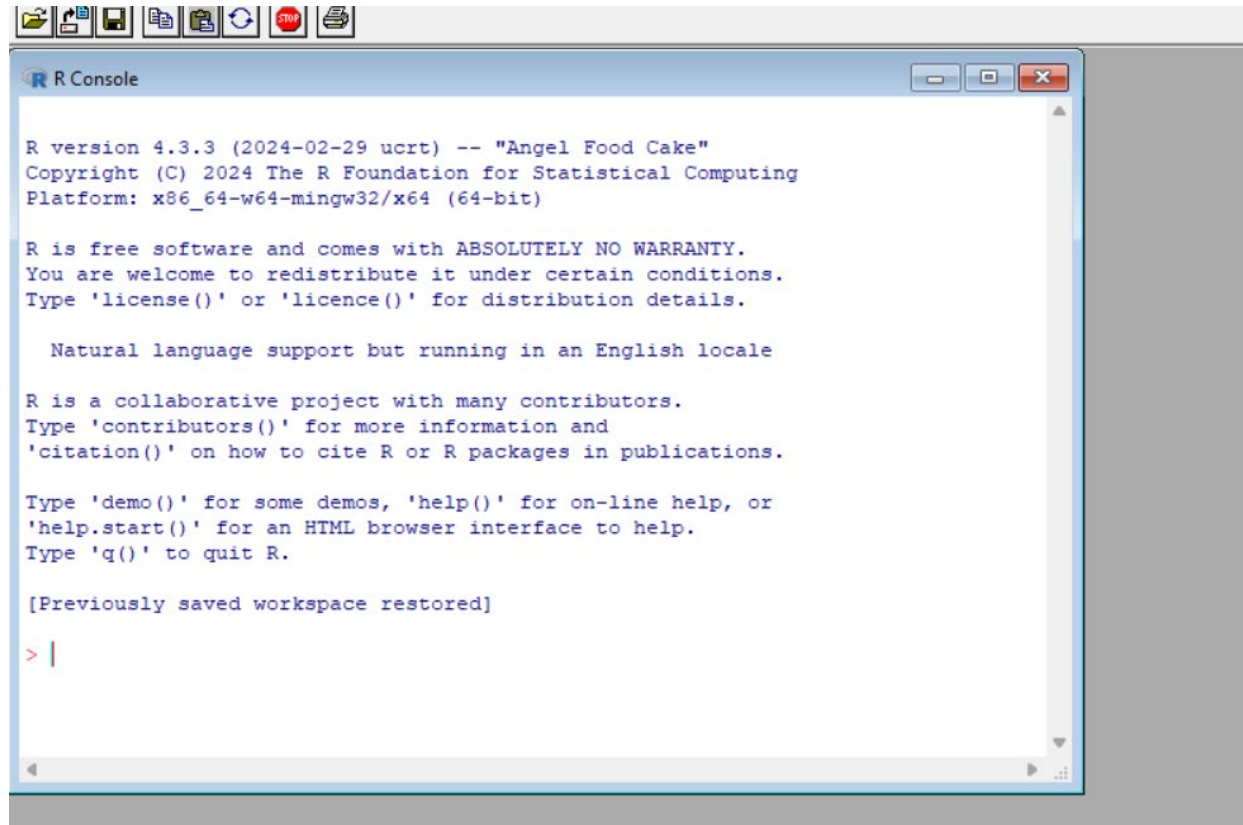
- R has many contributed **packages** (>10,000) to **extend R** for basic and advanced analysis

CRAN: The Comprehensive R Archive Network

- Typically, a **package** will include code, documentation for the package, the functions inside, and data sets.
- An example of a package is the **gam** package. This package contains multiple functions for fitting generalized additive models. Another example is the **shiny** package, which make interactive, web apps with R.
- **Packages** are stored in **libraries**

# The R console

The R console serves as the primary interface for executing R code.



```
R version 4.3.3 (2024-02-29 ucrt) -- "Angel Food Cake"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

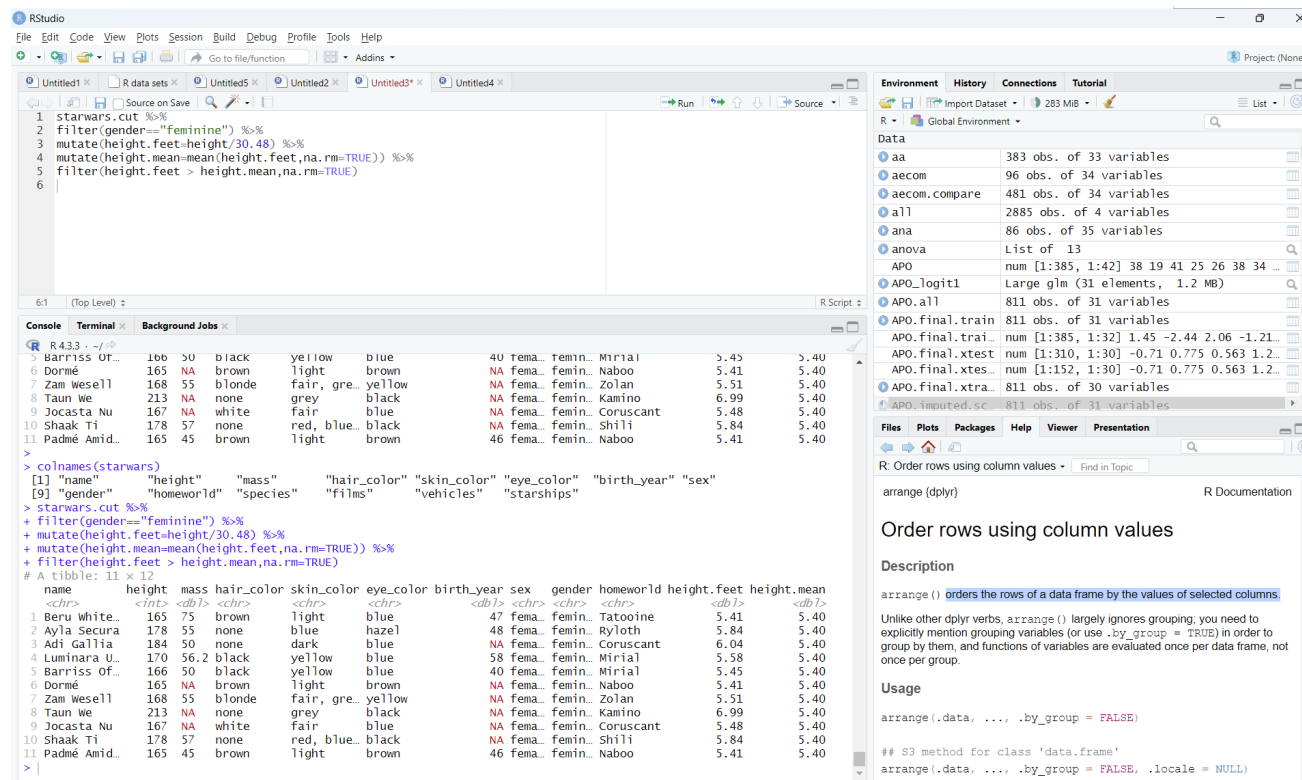
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> |
```

# R Studio

- **R Studio** provides a user-friendly interface with various tools and features such as debugging and fill-ins.





# The 4 panels of R Studio

The screenshot shows the R Studio interface with a blue circle highlighting the Source and Console panels. The Source panel (top left) contains R code for filtering and mutating a dataset. The Console panel (bottom left) shows the output of the code, including a tibble of Star Wars characters. The Environment panel (top right) lists loaded objects. The Plots panel (bottom right) shows the documentation for the `arrange()` function.

**Source panel – write code you can save and reuse**

```
1 starwars.cut %>%
2 filter(gender=="feminine") %>%
3 mutate(height.feet=height/30.48) %>%
4 mutate(height.mean=mean(height.feet,na.rm=TRUE)) %>%
5 filter(height.feet > height.mean,na.rm=TRUE)
6
```

**R console – work interactively**

```
> colnames(starwars)
[1] "name"      "height"    "mass"      "hair_color" "skin_color" "eye_color"  "birth_year" "sex"
[9] "gender"    "homeworld" "species"   "films"      "vehicles"   "starships"

> starwars.cut %>%
+ filter(gender=="feminine") %>%
+ mutate(height.feet=height/30.48) %>%
+ mutate(height.mean=mean(height.feet,na.rm=TRUE)) %>%
+ filter(height.feet > height.mean,na.rm=TRUE)
# A tibble: 11 x 12
  name      height mass hair_color skin_color eye_color birth_year sex  gender homeworld height.feet height.mean
<chr>    <dbl> <dbl> <chr>    <chr>    <chr>    <dbl> <chr> <chr>    <chr>    <dbl>    <dbl>
1 Beru White 165 75 brown light blue 47 fema... femin... Tatooine 5.41 5.40
2 Ayla Secura 178 55 none blue hazel 48 fema... femin... Ryloth 5.84 5.40
3 Adi Gallia 184 50 none dark blue 58 fema... femin... Coruscant 6.04 5.40
4 Luminara U... 170 56.2 black yellow blue 40 fema... femin... Mirial 5.58 5.40
5 Barriss Of... 166 50 black yellow blue 40 fema... femin... Mirial 5.45 5.40
6 Dormé 165 NA brown light brown NA fema... femin... Naboo 5.41 5.40
7 Zam Wesell 168 55 blonde fair, gre... yellow 5.51 5.40
8 Taun We 213 NA none grey black NA fema... femin... Kamino 6.99 5.40
9 Jocasta Nu 167 NA white fair blue NA fema... femin... Coruscant 5.48 5.40
10 Shaak Ti 178 57 none red, blue... black NA fema... femin... Shili 5.84 5.40
11 Padmé Amid... 165 45 brown light brown 46 fema... femin... Naboo 5.41 5.40
```

**Environment – variables**

Object	Size
aa	383 obs. of 33 variables
aecom	96 obs. of 34 variables
aecom.compare	481 obs. of 34 variables
all	2885 obs. of 4 variables
ana	86 obs. of 35 variables
anova	List of 13
APO	num [1:385, 1:42] 38 19 41 25 26 38 34 ...
APO_logit1	Large glm (31 elements, 1.2 MB)
APO.all	811 obs. of 31 variables
APO.final.train	811 obs. of 31 variables
APO.final.train	num [1:385, 1:32] 1.45 -2.44 2.06 -1.21...
APO.final.xtest	num [1:310, 1:30] -0.71 0.775 0.563 1.2...
APO.final.xtes...	num [1:152, 1:30] -0.71 0.775 0.563 1.2...
APO.final.xtra...	811 obs. of 30 variables
APO.imputed.sc	811 obs. of 31 variables

**Plots, help, packages**

R: Order rows using column values - Find in Topic

arrange(dplyr)

R Documentation

Order rows using column values

Description

arrange() orders the rows of a data frame by the values of selected columns.

Unlike other dplyr verbs, arrange() largely ignores grouping; you need to explicitly mention grouping variables (or use .by\_group = TRUE) in order to group by them, and functions of variables are evaluated once per data frame, not once per group.

Usage

```
arrange(.data, ..., .by_group = FALSE)
```

```
## S3 method for class 'data.frame'
arrange(.data, ..., .by_group = FALSE, .locale = NULL)
```

R Studio tour – live demonstration (tour 1)

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ - Import Save Print Run Go to file/function Addins Project: (None)

r code for session 1 markdown.Rmd R session 1 tour 1.R

```
1 #This is a comment
2 #####
3
4 # 1. First we are going to see how to run code in R
5 # typing up here does not do anything unless you run it!
6 #####
7
8 print('hello world')
9
10 print('Run the whole thing')
11
12 # 2. Now we are going to create a couple of new variables
13
14 ## <- is the assignment operator. You can also use an equal sign (=) but considered better form to use <-
15
16
17 x <- c(1:10)
18
19 print(x)
20
21 # we can see that x is now a variable in the global environment (see upper right panel)
22
23 y <- x*x
24
```

39:1 (Untitled) R Script

Console Terminal Render Background Jobs

R 4.3.3 · ~/R intensive/

```
>
> ## <- is the assignment operator. You can also use an equal sign (=) but considered .... [TRUNCATED]
>
> print(x)
[1] 1 2 3 4 5 6 7 8 9 10
>
> # we can see that x is now a variable in the global environment (see upper right panel)
>
> y <- x*x
>
> # variable named y consisting of an integer vector is now also listed in the global environment
>
> # we can also work interactively in the console .... [TRUNCATED]
>
> # see that the plot is now in the lower right panel
>
> plot(y)
>
> # we can scroll between the two plots we created
>
> |
```

Environment History Connections Tutorial

Import Dataset 850 MiB List

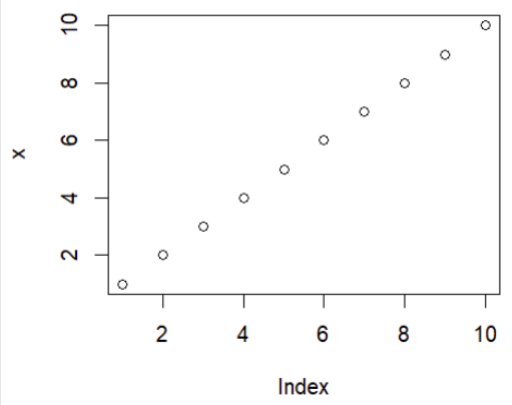
R Global Environment

Values

x	int [1:10]	1 2 3 4 5 6 7 8 9 10
y	int [1:10]	1 4 9 16 25 36 49 64...
z		385L

Files Plots Packages Help Viewer Presentation

Zoom Export



# Some useful shortcuts

- To clear your console → **CNTL+L**
- To clear your environment -> use the **broom** function
- To run a selected line from the source panel -> **CNTL+ENTER**
- To run the whole script from the source panel -> **CNTL+ALT+R**
- To repeat a command in the console – use the **^** arrow

# Getting data into R

- In the first tour, I created vectors x and y ***within R***, which is not the usual workflow
- Usually, you will ***import an Excel spreadsheet*** where you have collected and organized your data. We will show you how to do this.
- Today we will primarily work with data using a standard *wide* format – unique observations in rows and variables in columns
- A *long* format is used when you have multiple observations for an individual over time or space

# Wide vs long data formats

Example of wide format:

ID	x1	x2	x3	x4
1	3.3	1	2	17
2	2.1	1	3	22
3	7.0	0	1	10
4	4.1	1	1	42

Example of long format:

ID	time	x1	x2	x3	x4
1	1	3.3	1	2	17
1	2	0.5	1	1	14
2	1	2.1	1	3	22
2	2	2.8	0	5	16
3	1	7.0	0	1	10
4	1	4.1	1	1	42
4	2	4.1	0	0	57
5	1	3.8	1	1	12

# Data frames/tibbles

- When you import your data set into R, you typically store your data in a **data frame** within R
  - A **dataFrame** in R is a tabular (i.e., 2-dimensional, rectangular) data structure used to store values of any data type
  - A **tibble** is a newer version of the data frame, used in tidyverse

# Installing and Loading R packages (libraries)

- Over 10,000 libraries to help you manage, analyze, and output data
- These libraries contain **R packages**, which are collections of R functions, data and compiled code
- R libraries have to be loaded every time you open an R session



# tidyverse

- **Base R** refers to all the functionality that comes built into the R programming language. It is what you get when you open up the R console for the first time.
- The **tidyverse** is a collection of packages that **add onto R** to allow you to manipulate your data intuitively. It emphasizes readability. It does not replace base R (which is a large set of core packages).
- You can work ***entirely*** in base R and never work in Tidyverse
- In this session, we focus on learning the basics of some main packages in ***tidyverse (dplyr and ggplot2)***

# Installing and loading the tidyverse core

```
install.packages('tidyverse')    #Note: you did this prior to the workshop  
library('tidyverse')  
library('dplyr')                 #Note: dplyr automatically gets installed with tidyverse
```

Within ***tidyverse***, there is package called ***dplyr***. Think of the **d** as standing for **data** and the **plyr** standing for **plyers** – the goal of this package is to manipulate data frames in useful ways.

We will be focusing on the ***dplyr*** package in R this morning. This afternoon, we will learn about another package called ***ggplot2***

# The main verbs of *dplyr*

I just used a simple dplyr function called `bind_cols()` to combine vectors x and y, but this is only one of many. Below are the **main verbs** we will focus on this morning

- **Operation on columns (variables)**
  - select
  - mutate
  - rename
  - relocate
  - Pull
- **Operations on rows (observations)**
  - arrange
  - filter
  - slice\_min, slice\_max
- **Grouping and summarizing (observations)**
  - group\_by
  - summarize
- **Joining two data frames**
  - inner\_join, left\_join, right\_join, full\_join

# Data transformation with dplyr :: CHEATSHEET



dplyr functions work with pipes and expect **tidy data**. In tidy data:



Each **variable** is in its own **column**

&



Each **observation**, or **case**, is in its own **row**

**pipes**

**x** |> **f(y)**  
becomes **f(x, y)**

## Summarize Cases

Apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

**summary function**



**summarize(.data, ...)**

Compute table of summaries.  
mtcars |> summarize(avg = mean(mpg))



**count(.data, ..., wt = NULL, sort = FALSE, name = NULL)** Count number of rows in each group defined by the variables in ... Also **tally()**, **add\_count()**, **add\_tally()**.  
mtcars |> count(cyl)

## Group Cases

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.



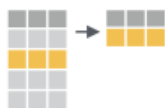
mtcars |>  
group\_by(cyl) |>  
summarize(avg = mean(mpg))

Use **rowwise(.data, ...)** to group data into individual rows. dplyr functions will compute results for each row. Also apply functions to list-columns. See tidyr cheat sheet for list-column workflow.

## Manipulate Cases

### EXTRACT CASES

Row functions return a subset of rows as a new table.



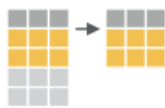
**filter(.data, ..., .preserve = FALSE)** Extract rows that meet logical criteria.  
mtcars |> filter(mpg > 20)



**distinct(.data, ..., .keep\_all = FALSE)** Remove rows with duplicate values.  
mtcars |> distinct(gear)



**slice(.data, ..., .preserve = FALSE)** Select rows by position.  
mtcars |> slice(10:15)



**slice\_sample(.data, ..., n, prop, weight\_by = NULL, replace = FALSE)** Randomly select rows. Use n to select a number of rows and prop to select a fraction of rows.  
mtcars |> slice\_sample(n = 5, replace = TRUE)

**slice\_min(.data, order\_by, ..., n, prop, with\_ties = TRUE)** and **slice\_max()** Select rows with the lowest and highest values.  
mtcars |> slice\_min(mpg, prop = 0.25)

**slice\_head(.data, ..., n, prop)** and **slice\_tail()** Select the first or last rows.  
mtcars |> slice\_head(n = 5)

### Logical and boolean operators to use with filter()

==	<	<=	is.na()	%in%		xor()
!=	>	>=	!is.na()	!	&	

See **?base::Logic** and **?Comparison** for help.

### ARRANGE CASES



**arrange(.data, ..., .by\_group = FALSE)** Order rows by values of a column or columns (low to high). Use **desc()** to order from high to low.

## Manipulate Variables

### EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.



**pull(.data, var = -1, name = NULL, ...)** Extract column values as a vector, by name or index.  
mtcars |> pull(wt)



**select(.data, ...)** Extract columns as a table.  
mtcars |> select(mpg, wt)



**relocate(.data, ..., .before = NULL, .after = NULL)** Move columns to new position.  
mtcars |> relocate(mpg, cyl, .after = last\_col())

### Use these helpers with select() and across()

e.g. mtcars |> select(mpg:cyl)

<b>contains(match)</b>	<b>num_range(prefix, range)</b>	! e.g., mpg:cyl
<b>ends_with(match)</b>	<b>all_of(x)/any_of(x, ..., vars)</b>	! e.g., !gear
<b>starts_with(match)</b>	<b>matches(match)</b>	<b>everything()</b>

### MANIPULATE MULTIPLE VARIABLES AT ONCE

df <- tibble(x\_1 = c(1, 2), x\_2 = c(3, 4), y = c(4, 5))



**across(.cols, .funs, ..., .names = NULL)** Summarize or mutate multiple columns in the same way.  
df |> summarize(across(everything(), mean))



**c\_across(.cols)** Compute across columns in row-wise data.  
df |>  
rowwise() |>  
mutate(x\_total = sum(c\_across(1:2)))

### MAKE NEW VARIABLES

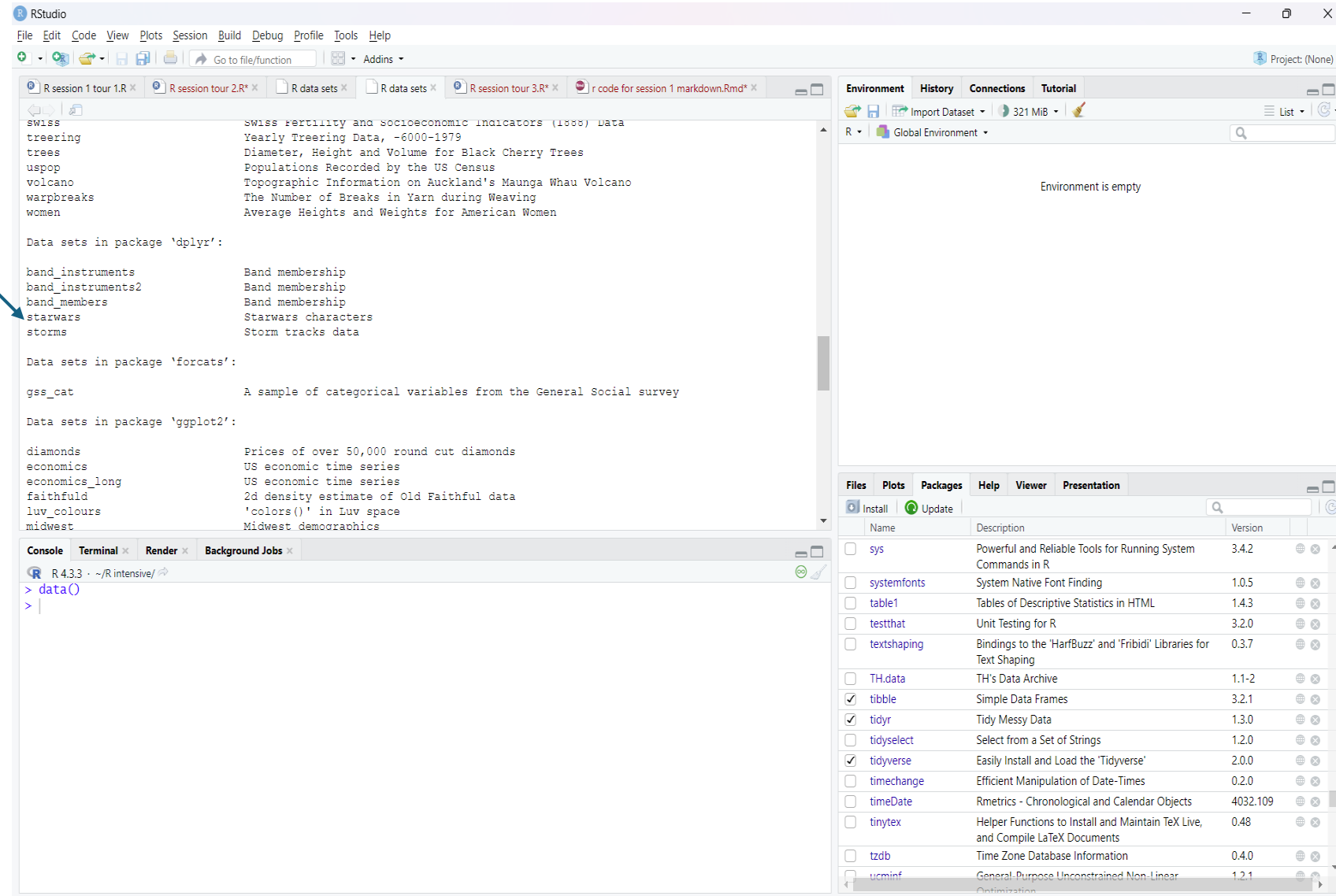
Apply **vectorized functions** to columns. Vectorized functions take vectors as input and return vectors of the same length as output (see back).

**vectorized function**

# Example data set

We will use the **starwars data set** available in the dplyr package

This data set consists of 87 observations and 14 variables.



The screenshot shows the RStudio interface with the following components:

- Source Editor:** Displays the 'Data sets in package 'dplyr'' list. The 'starwars' data set is highlighted in blue. The list includes:
  - swiss: Swiss fertility and socioeconomic indicators (1988) Data
  - treering: Yearly Treering Data, ~6000-1979
  - trees: Diameter, Height and Volume for Black Cherry Trees
  - uspop: Populations Recorded by the US Census
  - volcano: Topographic Information on Auckland's Maunga Whau Volcano
  - warpbreaks: The Number of Breaks in Yarn during Weaving
  - women: Average Heights and Weights for American Women
  - band\_instruments: Band membership
  - band\_instruments2: Band membership
  - band\_members: Band membership
  - starwars: Starwars characters
  - storms: Storm tracks data
  - gss\_cat: A sample of categorical variables from the General Social survey
  - diamonds: Prices of over 50,000 round cut diamonds
  - economics: US economic time series
  - economics\_long: US economic time series
  - faithful: 2d density estimate of Old Faithful data
  - luv\_colours: 'colors()' in Luv space
  - midwest: Midwest demographics
- Console:** Shows the R prompt and the command `> data()`.
- Environment:** Shows the 'Global Environment' with the message 'Environment is empty'.
- Packages:** A table of installed and available packages.

Name	Description	Version
<input type="checkbox"/> sys	Powerful and Reliable Tools for Running System Commands in R	3.4.2
<input type="checkbox"/> systemfonts	System Native Font Finding	1.0.5
<input type="checkbox"/> table1	Tables of Descriptive Statistics in HTML	1.4.3
<input type="checkbox"/> testthat	Unit Testing for R	3.2.0
<input type="checkbox"/> textshaping	Bindings to the 'HarfBuzz' and 'Fribidi' Libraries for Text Shaping	0.3.7
<input type="checkbox"/> TH.data	TH's Data Archive	1.1-2
<input checked="" type="checkbox"/> tibble	Simple Data Frames	3.2.1
<input checked="" type="checkbox"/> tidyr	Tidy Messy Data	1.3.0
<input type="checkbox"/> tidyselect	Select from a Set of Strings	1.2.0
<input checked="" type="checkbox"/> tidyverse	Easily Install and Load the 'Tidyverse'	2.0.0
<input type="checkbox"/> timechange	Efficient Manipulation of Date-Times	0.2.0
<input type="checkbox"/> timeDate	Rmetrics - Chronological and Calendar Objects	4032.109
<input type="checkbox"/> tinytex	Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents	0.48
<input type="checkbox"/> tzdb	Time Zone Database Information	0.4.0
<input type="checkbox"/> ucminf	General Purpose Unconstrained Non-Linear Optimization	1.2.1

# Example data set

> glimpse(starwars)

```
Rows: 87
Columns: 14
$ name      <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Organa", "Owen Lars", "Beru Whitesun lars", "R5-D4", "Biggs Da...
$ height    <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 180, 228, 180, 173, 175, 170, 180, 66, 170, 183, 200, 190, 177, 17...
$ mass      <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 84.0, 77.0, 84.0, NA, 112.0, 80.0, 74.0, 1358.0, 77.0, 110.0, 17.0, 7...
$ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", NA, "black", "auburn, white", "blond", "auburn, grey", "brown...
$ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "light", "light", "white, red", "light", "fair", "fair", "fair", "unkn...
$ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue", "red", "brown", "blue-gray", "blue", "blue", "blue", "brown...
$ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0, 57.0, 41.9, 64.0, 200.0, 29.0, 44.0, 600.0, 21.0, NA, 896.0, 82.0,...
$ sex        <chr> "male", "none", "none", "male", "female", "male", "female", "none", "male", "male", "male", "male", "male", "male", "ma...
$ gender      <chr> "masculine", "masculine", "masculine", "masculine", "feminine", "masculine", "feminine", "masculine", "masculine", "mas...
$ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "Tatooine", "Tatooine", "Tatooine", "Tatooine", "Stewjon", "Ta...
$ species     <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Human", "Droid", "Human", "Human", "Human", "Human", "Wookiee", ...
$ films      <list> <"The Empire Strikes Back", "Revenge of the Sith", "Return of the Jedi", "A New Hope", "The Force Awakens">, <"The Emp...
$ vehicles   <list> <"Snowspeeder", "Imperial Speeder Bike">, <,>, <,>, <,"Imperial Speeder Bike", <,>, <,>, <,"Tribubble bongo", <"Ze...
$ starships  <list> <"X-wing", "Imperial shuttle">, <,>, <,"TIE Advanced x1", <,>, <,>, <,"X-wing", <"Jedi starfighter", "Trade Feder...
```

*87 rows x 14 columns*

# Column operations

# *dplyr::select*

- A dplyr function that selects (and optionally renames) variables in a data frame

Example:

```
> starwars.cut <- select(starwars, c(1:6, 8:10))    #dplyr
```

```
> starwars.cut <- starwars[, c(1:6, 8:10)]         #base R
```



## *dplyr::select*

*Note: these are the same, though assignment operator (<-) preferred*

```
>starwars.cut <- select(starwars,c(1:6,8:10))
```

```
>starwars.cut = select(starwars,c(1:6,8:10))
```

*Note: these are NOT the same*

```
>starwars.cut <- select(starwars,c(1:6,8:10))
```

```
>starwars_cut = select(starwars,c(1:6,8:10))
```

## *dplyr::select*

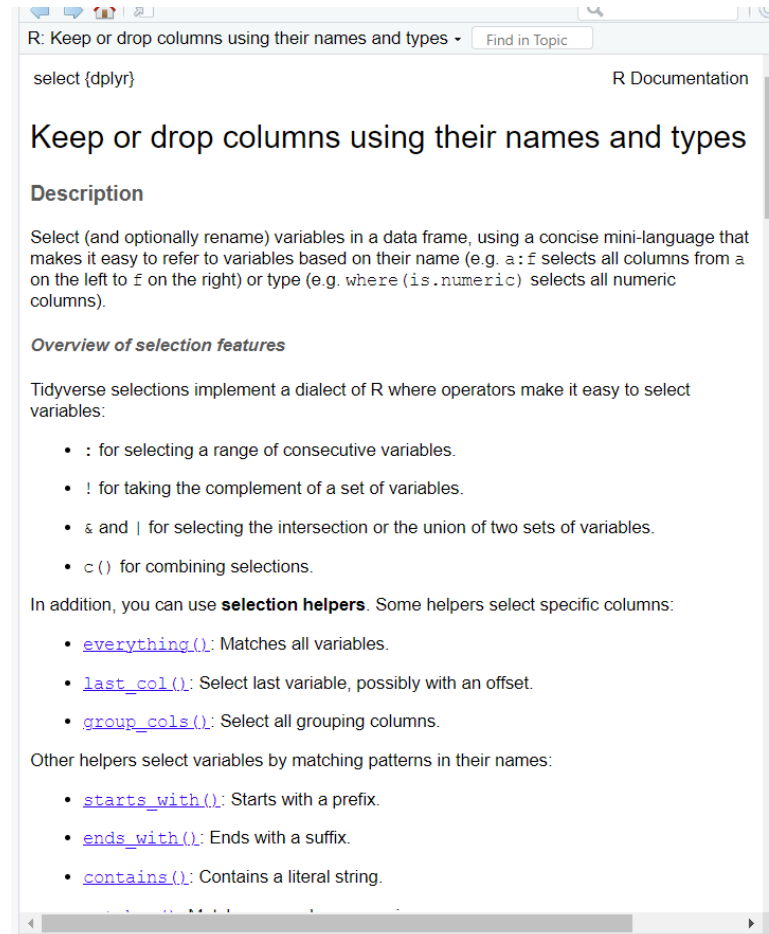
```
starwars.cut <- select(starwars,name:homeworld)
```

```
starwars.cut < select(starwars,name:eye_color,sex:homeworld)
```

```
starwars.cut <- select(starwars,-birth_year,-species:-starships)
```

# Can't exactly remember how the function works, key words, or options?

> ?select

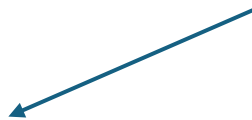


# Practice and learn:

Create a small data set and play around with different options

```
Console Terminal x Render x Background Jobs x
R 4.3.3 · ~/R intensive/ ↗
> example.data<-data.frame(cbind(c('A','B','C'),c(1,2,3),c(0,NA,NA)))
> example.data
  X1 X2 X3
1  A  1  0
2  B  2 <NA>
3  C  3 <NA>
> select(example.data,1)
  X1
1  A
2  B
3  C
> select(example.data,1:2)
  X1 X2
1  A  1
2  B  2
3  C  3
> select(example.data,contains("3"))
  X3
1  0
2 <NA>
3 <NA>
>
```

Looking for a variable name (column name) containing “3”,  
not a value in the data set



# Example data set

```
>starwars.cut
```

```
## # A tibble: 6 x 9
##   name      height  mass hair_color skin_color eye_color sex  gender homeworld
##   <chr>      <int> <dbl> <chr>      <chr>      <chr>  <chr> <chr>  <chr>
## 1 Luke Skyw~    172    77 blond      fair        blue   male  mascu~ Tatooine
## 2 C-3PO        167    75 <NA>       gold        yellow none  mascu~ Tatooine
## 3 R2-D2         96    32 <NA>       white, bl~ red     none  mascu~ Naboo
## 4 Darth Vad~    202   136 none       white       yellow male  mascu~ Tatooine
## 5 Leia Orga~    150    49 brown      light      brown  fema~ femin~ Alderaan
## 6 Owen Lars    178   120 brown, gr~ light      blue   male  mascu~ Tatooine
```

*A tibble 6x 9 – a tibble is a tidyverse data frame.*

*All dplyr verbs take a tibble as input*

# *dplyr::mutate*

- Used to replace/update the values of columns

Suppose we want to compute height in feet instead of meters

```
>mutate(starwars.cut, height.feet = height/30.48)
```

```
# A tibble: 87 × 10
  name      height mass hair_color skin_color eye_color sex gender homeworld height.feet
  <chr>    <int> <dbl> <chr>    <chr>    <chr>    <chr> <chr> <chr>    <dbl>
1 Luke Sk...    172    77 blond    fair     blue    male  mascu... Tatooine    5.64
2 C-3PO      167    75 NA       gold     yellow  none  mascu... Tatooine    5.48
3 R2-D2       96    32 NA       white, bl... red     none  mascu... Naboo      3.15
4 Darth V...   202   136 none     white    yellow  male  mascu... Tatooine    6.63
5 Leia Or...   150    49 brown    light    brown   fema... femin... Alderaan    4.92
6 Owen La...   178   120 brown, gr... light    blue    male  mascu... Tatooine    5.84
7 Beru Wh...   165    75 brown    light    blue    fema... femin... Tatooine    5.41
8 R5-D4       97    32 NA       white, red  red     none  mascu... Tatooine    3.18
9 Biggs D...   183    84 black    light    brown   male  mascu... Tatooine    6.00
10 Obi-Wan...   182    77 auburn, w... fair     blue-gray male  mascu... Stewjon    5.97
# i 77 more rows
# i Use `print(n = ...)` to see more rows
```

# Combining verbs with pipes (%>%)

```
>mutate(starwars.cut, height.feet = height/30.48)
```

is the same as:

```
>starwars.cut %>% mutate(height.feet = height/30.48)
```

*“take the starwars.cut data set **then** add a new variable with height in feet”*

Why use pipes? It can make code easy to follow and can avoid repetitive typing of (for example) the data frame name in every function

Shortcut to type %>%: **CNTL+SHIFT+M**

# Combining verbs with pipes (%>%)

If we instead want to create a new data frame with height.feet included:

```
> starwars.cut1 = starwars %>% mutate(height.feet = height/30.48)
```

We can now look on the top right (environment) to see this new data frame



# dplyr::rename, rename\_with

- Changes the name of a column

```
>rename(starwars.cut1,eye.color=eye_color)
```

```
>starwars.upper <- rename_with(starwars.cut1,toupper)
```

# dplyr::pull

Extracts a variable (column) as a vector

```
>pull(starwars.cut,height)
```

```
>pull(starwars.cut,2)
```

```
> pull(starwars.cut,height)
```

```
[1] 172 167  96 202 150 178 165  97 183 182 188 180 228 180 173 175 170 180  66 170 183 200 190 177 175 180 150  NA  88 160 193 191 170 196  
[35] 224 206 183 137 112 183 163 175 180 178  94 122 163 188 198 196 171 184 188 264 188 196 185 157 183 183 170 166 165 193 191 183 168 198  
[69] 229 213 167  79  96 193 191 178 216 234 188 178 206  NA  NA  NA  NA  NA 165
```

```
> pull(starwars.cut,2)
```

```
[1] 172 167  96 202 150 178 165  97 183 182 188 180 228 180 173 175 170 180  66 170 183 200 190 177 175 180 150  NA  88 160 193 191 170 196  
[35] 224 206 183 137 112 183 163 175 180 178  94 122 163 188 198 196 171 184 188 264 188 196 185 157 183 183 170 166 165 193 191 183 168 198  
[69] 229 213 167  79  96 193 191 178 216 234 188 178 206  NA  NA  NA  NA  NA 165
```

# Row operations

# dplyr::arrange

- arrange orders the rows of a data frame by the values of selected columns

Note: We can find more information using

`??dplyr::arrange` (or `??arrange`) to get relevant help pages

# Example using *arrange*

```
> starwars.cut %>% arrange(desc(height))  
> arrange(starwars.cut, desc(height))
```

The 87x9 tibble `starwars.cut` is to be sorted by height



```
# A tibble: 87 × 9  
  name          height mass hair_color skin_color eye_color sex gender homeworld  
  <chr>         <int> <dbl> <chr>      <chr>      <chr>  <chr> <chr> <chr>  
1 Yarael Poof    264    NA none      white      yellow male mascul... Quermia  
2 Tarfful        234   136 brown     brown      blue   male mascul... Kashyyyk  
3 Lama Su        229    88 none      grey       black  male mascul... Kamino  
4 Chewbacca      228   112 brown     unknown    blue   male mascul... Kashyyyk  
5 Roos Tarpals   224    82 none      grey       orange male mascul... Naboo  
6 Grievous       216   159 none      brown, white green, yellow male mascul... Kalee  
7 Taun We        213    NA none      grey       black  female femini... Kamino  
8 Rugor Nass     206    NA none      green      orange male mascul... Naboo  
9 Tion Medon     206    80 none      grey       black  male mascul... Utapau  
10 Darth Vader   202   136 none      white      yellow male mascul... Tatooine  
# i 77 more rows  
# i Use `print(n = ...)` to see more rows
```

# Example using *arrange*

- We can also sort by more than one variable

```
> starwars.cut %>% arrange(gender, desc(height))
```

which sorts by gender and then ascending height

```
> starwars.cut %>% group_by(gender) %>%  
  arrange(desc(height), .by_group=TRUE)
```

accomplishes the same thing

# dplyr::filter

- used to subset a data frame, retaining all rows that satisfy your conditions
- Useful functions:
  - == > <
  - & | ! xor()
  - is.na()
  - between()
  - near()

# Examples using *filter*

```
> starwars.cut %>% filter(homeworld=="Naboo")
```

```
# A tibble: 11 × 9
```

	name	height	mass	hair_color	skin_color	eye_color	sex	gender	homeworld
	<chr>	<int>	<dbl>	<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
1	R2-D2	96	32	NA	white, blue	red	none	masculine	Naboo
2	Palpatine	170	75	grey	pale	yellow	male	masculine	Naboo
3	Jar Jar Binks	196	66	none	orange	orange	male	masculine	Naboo
4	Roos Tarpals	224	82	none	grey	orange	male	masculine	Naboo
5	Rugor Nass	206	NA	none	green	orange	male	masculine	Naboo
6	Ric Olié	183	NA	brown	fair	blue	NA	NA	Naboo
7	Quarsh Panaka	183	NA	black	dark	brown	NA	NA	Naboo
8	Gregar Typho	185	85	black	dark	brown	male	masculine	Naboo
9	Cordé	157	NA	brown	light	brown	female	feminine	Naboo
10	Dormé	165	NA	brown	light	brown	female	feminine	Naboo
11	Padmé Amidala	165	45	brown	light	brown	female	feminine	Naboo



# Examples using *filter*

```
> starwars.cut %>%  
  filter(homeworld=="Naboo") %>%  
  filter(hair_color != 'grey')
```

== logical comparison to check equality, assignment uses =

Note we can use a double ("Naboo") or single ('grey') quote interchangeably in R

```
# A tibble: 9 × 9  
  name      height mass hair_color skin_color eye_color sex  gender homeworld  
  <chr>    <int> <dbl> <chr>    <chr>    <chr>  <chr> <chr>    <chr>  
1 Jar Jar Binks    196    66 none     orange   orange male  masculine Naboo  
2 Roos Tarpals    224    82 none     grey     orange   male  masculine Naboo  
3 Rugor Nass      206    NA none     green    orange   male  masculine Naboo  
4 Ric Olié        183    NA brown    fair     blue     NA    NA      Naboo  
5 Quarsh Panaka   183    NA black    dark     brown    NA    NA      Naboo  
6 Gregar Typho    185    85 black    dark     brown    male  masculine Naboo  
7 Cordé          157    NA brown    light    brown    female feminine Naboo  
8 Dormé          165    NA brown    light    brown    female feminine Naboo  
9 Padmé Amidala   165    45 brown    light    brown    female feminine Naboo
```

More succinctly, we can type:

```
> filter(starwars.cut, homeworld == "Naboo" & hair_color != 'grey' )
```

# Examples using *filter*

```
> starwars.cut %>% filter(height > 200 ) %>% filter(mass != 'NA')
```

```
# A tibble: 7 × 9
```

	name <chr>	height <int>	mass <dbl>	hair_color <chr>	skin_color <chr>	eye_color <chr>	sex <chr>	gender <chr>	homeworld <chr>
1	Darth Vader	202	136	none	white	yellow	male	masculine	Tatooine
2	Chewbacca	228	112	brown	unknown	blue	male	masculine	Kashyyyk
3	Roos Tarpals	224	82	none	grey	orange	male	masculine	Naboo
4	Lama Su	229	88	none	grey	black	male	masculine	Kamino
5	Grievous	216	159	none	brown, white	green, yellow	male	masculine	Kalee
6	Tarfful	234	136	brown	brown	blue	male	masculine	Kashyyyk
7	Tion Medon	206	80	none	grey	black	male	masculine	Utapau

# Grouping operations

# dplyr::group\_by

- Groups the data frame by a grouping variable(s). Becomes a grouped tibble where operations are performed by group until it is ungrouped
- NOT the same as sorting or ***dplyr::arrange***

```
> group_by(starwars.cut1, gender)
```

```
> ungroup(starwars.cut1)
```

# group\_by

```
> group_by(starwars.cut1, gender)
# A tibble: 87 × 11
# Groups:   gender [3]
  name          height mass hair_color skin_color eye_color birth_year sex gender homeworld height.feet
  <chr>         <int> <dbl> <chr>    <chr>    <chr>    <dbl> <chr> <chr>    <chr>    <dbl>
1 Luke Skywalker    172    77 blond     fair      blue      19   male masculine Tatooine    5.64
2 C-3PO             167    75 NA       gold      yellow    112  none masculine Tatooine    5.48
3 R2-D2              96    32 NA       white, blue red       33   none masculine Naboo      3.15
4 Darth Vader       202   136 none     white     yellow    41.9 male masculine Tatooine    6.63
5 Leia Organa       150    49 brown     light     brown     19   female feminine Alderaan   4.92
6 Owen Lars         178   120 brown, grey light     blue      52   male masculine Tatooine    5.84
7 Beru Whitesun lars 165    75 brown     light     blue      47   female feminine Tatooine    5.41
8 R5-D4              97    32 NA       white, red red       NA   none masculine Tatooine    3.18
9 Biggs Darklighter 183    84 black     light     brown     24   male masculine Tatooine    6.00
10 Obi-Wan Kenobi    182    77 auburn, white fair      blue-gray  57   male masculine Stewjon    5.97
# i 77 more rows
# i Use `print(n = ...)` to see more rows
> ungroup(starwars.cut1)
# A tibble: 87 × 11
  name          height mass hair_color skin_color eye_color birth_year sex gender homeworld height.feet
  <chr>         <int> <dbl> <chr>    <chr>    <chr>    <dbl> <chr> <chr>    <chr>    <dbl>
1 Luke Skywalker    172    77 blond     fair      blue      19   male masculine Tatooine    5.64
2 C-3PO             167    75 NA       gold      yellow    112  none masculine Tatooine    5.48
3 R2-D2              96    32 NA       white, blue red       33   none masculine Naboo      3.15
4 Darth Vader       202   136 none     white     yellow    41.9 male masculine Tatooine    6.63
5 Leia Organa       150    49 brown     light     brown     19   female feminine Alderaan   4.92
6 Owen Lars         178   120 brown, grey light     blue      52   male masculine Tatooine    5.84
7 Beru Whitesun lars 165    75 brown     light     blue      47   female feminine Tatooine    5.41
8 R5-D4              97    32 NA       white, red red       NA   none masculine Tatooine    3.18
9 Biggs Darklighter 183    84 black     light     brown     24   male masculine Tatooine    6.00
10 Obi-Wan Kenobi    182    77 auburn, white fair      blue-gray  57   male masculine Stewjon    5.97
# i 77 more rows
# i Use `print(n = ...)` to see more rows
.
```

# *dplyr::summarize*

- Creates a new data frame.
- It returns one row for each combination of grouping variables

Example with no grouping variable:

```
> starwars.cut %>%
```

```
  mutate(height.feet=height/30.48) %>%
```

```
  summarize(mean=mean(height.feet,na.rm=TRUE),  
             median=median(height.feet,na.rm=TRUE))
```

```
# A tibble: 1 × 2  
  mean median  
  <dbl>  <dbl>  
1  5.72   5.91
```

# *dplyr::summarize*

Some basic statistics you may compute:

Objective	Function
Basic	mean()
	median()
	sum()
variation	sd()
	IQR()
range	min(), max()
	quantile()

# *summarize*

Compute the average height in feet by gender

```
> starwars.cut %>%  
  mutate(height.feet=height/30.48) %>%  
  group_by(gender) %>%  
  summarize(mean=mean(height.feet,na.rm=TRUE),  
    count=n())
```

```
# A tibble: 3 × 3  
  gender    mean count  
  <chr>    <dbl> <int>  
1 feminine  5.40    17  
2 masculine  5.79    66  
3 NA        5.95     4
```

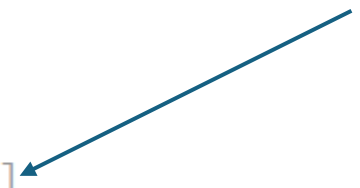


# *mutate*

Compute the deviation of each individual height (in feet) from the average for their gender

```
> starwars.cut %>%  
  mutate(height.feet=height/30.48) %>%  
  group_by(gender) %>%  
  mutate(height.grp.mean=mean(height.feet,na.rm=TRUE)) %>%  
  mutate(height.dev=height-height.grp.mean) %>%  
  select(name,gender,height.feet, height.dev, height.grp.mean,)
```

# *mutate*



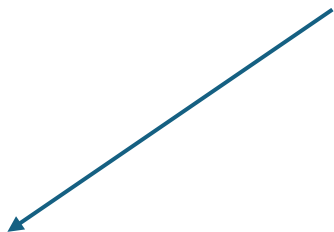
```
# A tibble: 87 x 5
# Groups:   gender [3]
  name          gender height.feet height.dev height.grp.mean
  <chr>         <chr>      <dbl>      <dbl>      <dbl>
1 Luke Skywalker masculine    5.64    -0.148      5.79
2 C-3PO         masculine    5.48    -0.312      5.79
3 R2-D2         masculine    3.15    -2.64       5.79
4 Darth Vader   masculine    6.63     0.836      5.79
5 Leia Organa   feminine    4.92    -0.482      5.40
6 Owen Lars     masculine    5.84     0.0487     5.79
7 Beru Whitesun lars feminine    5.41     0.0103     5.40
8 R5-D4         masculine    3.18    -2.61       5.79
9 Biggs Darklighter masculine    6.00     0.213      5.79
10 Obi-Wan Kenobi masculine    5.97     0.180      5.79
# i 77 more rows
# i Use `print(n = ...)` to see more rows
```

# *mutate*

Also compute the deviation from the overall mean:

```
> starwars.cut %>%  
  mutate(height.feet=height/30.48) %>%  
  group_by(gender) %>%  
  mutate(height.dev=height.feet - mean(height.feet,na.rm=TRUE))%>%  
  mutate(height.grp.mean=mean(height.feet,na.rm=TRUE)) %>%  
  ungroup(gender) %>%  
  mutate(height.overall.dev=height.feet - mean(height.feet,na.rm=TRUE)) %>%  
  select(name,gender,height.feet,height.dev,height.grp.mean,height.overall.dev)
```

# mutate



```
# A tibble: 87 × 6
  name                gender  height.feet height.dev height.grp.mean height.overall.dev
  <chr>              <chr>      <dbl>      <dbl>      <dbl>          <dbl>
1 Luke Skywalker    masculine    5.64      -0.148      5.79          -0.0774
2 C-3PO             masculine    5.48      -0.312      5.79          -0.241
3 R2-D2             masculine    3.15      -2.64       5.79          -2.57
4 Darth Vader       masculine    6.63       0.836      5.79           0.907
5 Leia Organa       feminine     4.92      -0.482      5.40          -0.799
6 Owen Lars         masculine    5.84       0.0487      5.79           0.119
7 Beru Whitesun lars feminine     5.41       0.0103      5.40          -0.307
8 R5-D4             masculine    3.18      -2.61       5.79          -2.54
9 Biggs Darklighter masculine     6.00       0.213      5.79           0.284
10 Obi-Wan Kenobi    masculine     5.97       0.180      5.79           0.251
# i 77 more rows
# i Use `print(n = ...)` to see more rows
```

# Example

Output the count for each gender and eye color.

```
gender  eye_color count
<chr>   <chr>    <int>
1 feminine black      2
2 feminine blue       6
3 feminine brown      5
4 feminine hazel      2
5 feminine red, blue   1
6 feminine yellow     1
7 masculine black      8
8 masculine blue     12
9 masculine blue-gray  1
10 masculine brown    15
# i 13 more rows
```

# Example

Output the count for each gender and eye color.

```
> starwars.cut %>%  
  group_by(gender, eye_color) %>%  
  summarize(count=n())
```

	gender	eye_color	count
	<chr>	<chr>	<int>
1	feminine	black	2
2	feminine	blue	6
3	feminine	brown	5
4	feminine	hazel	2
5	feminine	red, blue	1
6	feminine	yellow	1
7	masculine	black	8
8	masculine	blue	12
9	masculine	blue-gray	1
10	masculine	brown	15
# i 13 more rows			

# Example

Select all gender="feminine" individuals with height in feet greater than the overall group average height and then summarize the count in each homeworld for this subgroup

```
# A tibble: 11 x 12
  name      height mass hair_color skin_color eye_color birth_year sex  gender homeworld height.feet height.mean
  <chr>    <int> <dbl> <chr>    <chr>    <chr>    <dbl> <chr> <chr>  <chr>    <dbl>    <dbl>
1 Beru White...  165  75  brown    light    blue      47 fema... femin... Tatooine    5.41     5.40
2 Ayla Secura    178  55  none     blue     hazel     48 fema... femin... Ryloth     5.84     5.40
3 Adi Gallia    184  50  none     dark     blue      NA fema... femin... Coruscant   6.04     5.40
4 Luminara U...  170  56.2 black    yellow   blue      58 fema... femin... Mirial     5.58     5.40
5 Barriss Of...  166  50  black    yellow   blue      40 fema... femin... Mirial     5.45     5.40
6 Dormé         165  NA   brown    light    brown     NA fema... femin... Naboo      5.41     5.40
7 Zam Wesell    168  55  blonde   fair, gre... yellow   NA fema... femin... Zolan      5.51     5.40
8 Taun We       213  NA   none     grey     black     NA fema... femin... Kamino     6.99     5.40
9 Jocasta Nu    167  NA   white    fair     blue      NA fema... femin... Coruscant   5.48     5.40
10 Shaak Ti     178  57  none     red, blue... black     NA fema... femin... Shili      5.84     5.40
11 Padmé Amid...  165  45  brown    light    brown     46 fema... femin... Naboo      5.41     5.40
```

```
# A tibble: 8 x 2
  homeworld count
  <chr>    <int>
1 Coruscant    2
2 Kamino       1
3 Mirial       2
4 Naboo        2
5 Ryloth       1
6 Shili        1
7 Tatooine     1
8 Zolan        1
```

# Example

Select all gender="feminine" individuals with height in feet greater than the overall group average height and present a table summarizing the count of each homeworld in this subset

```
>starwars.cut %>%  
  filter(gender== 'feminine') %>%  
  mutate(height.feet=height/30.48) %>%  
  mutate(height.mean=mean(height.feet,na.rm=TRUE)) %>%  
  filter(height.feet > height.mean,na.rm=TRUE))
```



# Merging data frames

# dplyr::inner\_join

Only keeps observations from data frame A if they have a match in data frame B. Unmatched observations from A or B are not kept

Example:

A =	Name	Age
	Joe	19
	John	33
	Jack	41

B =	Name	Wage/hr
	Joe	15
	John	21
	Jack	24
	Ed	35

```
> inner_join(A,B,by='Name')
```

```
  Name Age Wage
1  Joe  19   15
2 John  33   21
3 Jack  41   24
```

# dplyr::left\_join and dplyr::right\_join

A =

Name	Age
Joe	19
John	33
Jack	41

B=

Name	Wage/hr
Joe	15
John	21
Jack	24
Ed	35

```
> left_join(A,B,by='Name')
  Name Age Wage
1  Joe  19   15
2 John  33   21
3 Jack  41   24
> right_join(A,B,by='Name')
  Name Age Wage
1  Joe  19   15
2 John  33   21
3 Jack  41   24
4   Ed  NA   35
```

# Helpful online communities

## R Studio

- <https://forum.posit.co/>

## R

- <https://stackoverflow.com/questions/tagged/R>

## Tidyverse:


- <https://posit.co/resources/videos/a-gentle-introduction-to-tidy-statistics-in-r/>
- <https://dplyr.tidyverse.org/reference/mutate-joins.html>

# BERD House

<https://www.einsteinmed.edu/centers/ictr/biostatistics-epidemiology-research-design-core/berd-house/>


[Main Site](#) [Research](#) [Education](#) [Health](#)

Follow Us: [f](#) [t](#) [i](#) [v](#) [in](#)

[Overview](#) [Leadership](#) [Services](#) [Education](#) [Community Engagement](#) [Q](#)

## BERD House

The BERD House is an online biostatistics resource managed by the BERD core. This resource includes guidelines for preparing a data set, statistical tools, links to statistics software, statistics training opportunities, and other statistics resources.



[Access Services](#)  
[Search Clinical Trials](#)

### Cite the CTSA

Ensure our resources are available for future research by citing the NIH CTSA grant.  
[CLICK HERE TO LEARN MORE](#)

UM1TR004400  
K12TR004411  
T32TR004537

### Getting started: Biostatistics support and guidelines


#### Who can I meet with to discuss my project?

The Division of Biostatistics offers biostatistics consulting and collaboration to enhance the quality and rigor of scientific research conducted by investigators at Einstein and Montefiore.

#### Biostatistics walk-in clinic:

Meet with a biostatistician without an appointment and obtain quick advice on your project.

#### Tweets from @EinsteinICTR



Nothing to