Welcome to The Carpentries Etherpad for Intro to R

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Users are expected to follow our code of conduct:
https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

---

Introduction to R

We will be following the Carpentry curriculum: R for Reproducible Scientific Analysis (https://swcarpentry.github.io/r-novice-gapminder/)

Workshop series website: https://ucsdlib.github.io/2021-09-13-uc-collab/
Collabortive document for series: https://codimd.carpentries.org/fwkQLM2ZTuCRirtE16WjBQ?view
This etherpad for today: https://pad.carpentries.org/UC-Carpentry-R-2021-09-20-concurrent
Lecture notes: https://uc-carps-intro-r.netlify.app/
Feedback form:
https://docs.google.com/forms/d/e/1FAIpQLSdiyAxUE3o0WnWOq4LMfPbDPqlG0akkGwOSNMf6NkGnINnmfw/viewform
ZOOM: https://ucla.zoom.us/j/94108574704?pwd=ZytBclJnVlJ1bUkvRlM2ZERMYWdWQT09
Meeting ID: 941 0857 4704
Passcode: 664202
Introduction to Vector Data in R 9/24 Register <https://ti.to/ucsd-carpentries/uc-carpentries-fall-workshop>
Introduction to Raster Data in R 10/1 Register <https://ti.to/ucsd-carpentries/uc-carpentries-fall-workshop>
Open lab tomorrow from 9-noon: https://ucla.zoom.us/j/95343916630?pwd=M1NRUlZObTgyWTlBSG5WdHc3MHp4Zz09

Installing R

- Be sure to have R and RStudio installed. Follow the instructions on the workshop page: https://ucsdlib.github.io/2021-09-13-uc-collab/
- If you can't get RStudio working on your machine, you can use https://rstudio.cloud/ in a pinch. You'll need to create an account.

Welcome to UC Carpentries Workshop - Introduction to R

- We will use this Etherpad for sharing links, code snippets, notes, ask/answer questions, etc.
- From the upper right corner of this screen, please change "anonymous" to your name and if you wish, please choose a color for your writing.
- There is a chat window at the bottom right corner. In case of zoom difficulties, please notify us using that.
- In this part at the left, we will keep adding our notes as we go through the workshop. You can also add your notes/questions here as well. This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows us to collaborate seamlessly on this document. Click on any suitable line and start typing!
- From the upper right corner, you can choose to save your own copy of this document by clicking the double arrow and choosing HTML or other formats.
- In case this page keeps showing error, please keep it closed for few minutes and open again.
- Both this page and the lesson pages will remain available online after the workshop. In case you need to review anything, please feel free to visit the links again!

Code of conduct

- Use welcoming and inclusive language
- Be respectful of different viewpoints and experiences
- Gracefully accept constructive criticism
- Focus on what is best for the community
- Show courtesy and respect towards other community members
- More details at https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html

Guidelines

- Zoom chat can be used to ask questions or provide comments or express difficulties during the lesson. Type “hand” in the chat, or type the question into the chat, or use a “raise hand”. To request assistance, message helper or indicate the red x "no" in the participants chat.
- Helpers may continue to reply there or via zoom private chat as needed.
- If necessary, helpers may temporarily move to a zoom breakout room with you to discuss in details using audio/screen sharing. You can return to the main session by leaving the breakout room.
- You can participate in the polls during the lesson by clicking Zoom Participants Panel and clicking on yes/no.
- For a single screen, it can be split to have the zoom window and the window where we'll be working, both open side by side.
- Please rename yourself to add the Operating System (OS) after your name in zoom. e.g. Annajiat (Linux)
- Having a Great Online Learning Experience: A Guide for Students, https://carpentries.org/blog/2020/04/great-online-learning-student

Day 1

Teaching team:

Name / pronouns / Affiliation / email
Jamie Jamison / UCLA / Data Science Center / she, her
Tim Dennis / UCLA / Data Science Center / he, him

Helpers:

Name / pronouns / Affiliation / email
Kenji Hayashi / he, him / UCLA / kthayashi@ucla.edu
Lisa McAulay / she/her / UCLA / emcaulay@library.ucla.edu
Maddi Cowen / she/her / UCLA / mcowen@g.ucla.edu

Learners, Please Sign In:

Name / pronouns / Affiliation / department / email

1. Cyndi Gutierrez / She, hers/ UCLA / cgutierrez00@g.ucla.edu
2. Aniruddha Adhikari/ He, him/ UCLA/ aniruddhacbe@ucla.edu
3. Edward Ramos/UCLA/Semel/weramos@ucla.edu
4. Debolina/She,hers/UCSD/Chem/desarkar@ucsd.edu
5. Kailash Venkatraman/He,him/UCSD/kvenkatr@ucsd.edu
6. Ashley Johnston/she,her/UCLA/am
7. Arsh Malik/ He,Him/ UCLA/ Henry Samueli School of Engineering/ arshmalik02@ucla.edu
8. Chelsea Blankenchip/she, her/UCSD/Biomedical Sciences/clblanke@health.ucsd.edu
9. Cameron Ziegler / he,him/UCLA/economics/cziegler661@gmail.com
10. Aileen Button/she, hers/UCSD/chem/acbutton@ucsd.edu
11. Federico Milani/federico.milani80@gmail.com
12. Wenyan Guan/she, her/UCM/MBSE/wguan3@ucmerced.edu
13. Deniz Orkun Eren / he,him/ UCLA / ECE / deren@g.ucla.edu
14. Ashley Mensing / she, her/ UCLA / Anthropology / anm2186@g.ucla.edu
15. Reuven Sinensky / he, him / UCLA / Anthropology / rsinensky@ucla.edu
16. Pranav Singamsetty / he, him/ UCLA / Engineering / pranavsingam@ucla.edu
17. Yan Zhou/she, her/UCLA/Asian languages and cultures/yz007@ucla.edu
18. Negli Gallardo/he, him/UNICAMP/Health Sciences/n234066@dac.unicamp.br

What is the mythical creature from where you grew up?

- Tim: Honey Island Swamp Monster, aka, Cajun Sasquash, near Slidell, Louisiana https://en.wikipedia.org/wiki/Honey_Island_Swamp_monster
- Jamie: I grew up in Los Angles, it's crazy enough here that we don't have mythical monsters. The real ones are sufficient.
- Lisa: Headless Horseman; I grew up in Upstate New York, and the author of The Legend of Sleepy Hollow (Washington Irving) was a New Yorker (eventually he moved a little outside the city, but not so far into the country as I lived)

- Maddi: I grew up in Miami, Florida. I think our not-so-very-mythical creature we all fear is the Florida Man
- Aniruddha: I grew up in Kolkata (Calcutta), India where everything is mythical.
- Ashley: In Florida,we've got our own smaller, stinkier version of Big Foot that lives in the swamp - the Skunk Ape.
- Edward: The California Bigfoot.....
- Aileen: I grew up in Vermont, where we have our own version of the Loch Ness monster, Champ, who fittingly lives in Lake Champlain
- Arsh: I grew up in New Delhi, India. Probably everything is mythical :)
- Kenji: I grew up in Kyoto, Japan - we have a lot of different mythical creatures (e.g. oni)
- Negli: I grew up in Guatemala, we have a lot of mythical cratures. El Sombrerón is a goblin that wears a big hat.

Agenda
9:00 - 9:15: Welcome, logistics, and introductions (Tim)
9:15 - 10:15: Introduction to R and RStudio, project management in R, getting help in R, & challenges (Tim)
10:15 - 10:25: Break
10:25 - 11:00: Data structures (Tim)
11:00 - 11:10: Break
11:10 - 12:10: Exploring data frames, subsetting data (Jamie)
12:10-12:20: Break
12:20 - 12:35: Breakout rooms - exercises
12:45 - 12:45: Go over exercises
12:45 - 1:00: Reflect/Q&A/Wrap up

Notes & Exercises (help us take notes)

Introduction to R and RStudio

R is a programming language and RStudio is a program (or application) that helps you use the R programming language by giving you "a console" interface with lots of help

Notes:

You can use the R console like a calculator and do basic math, although many of us want to use R for more complex uses. There are lots of built-in functions you can use for other kinds of computations.
“Function” = an operation that the program / programming language does for you
Arguments = the things you hand over to the function for the program to use in completing the function
A lot of programming languages use parentheses for the way you type in functions
Syntax is the term for the rules of how you write things in a programming language (spaces, punctuation, verbs, etc.)

an example of a function in R is "sum()"
Logical operators (or Boolean operators):
- comparison in R is "==" (this command asks whether two items are equal to each other)

- "!=" checks if two items are NOT equal
- can also use > or < (greater than or less than) or >= or <= (greater than or equal to; less than or equal to)

What is the meaning of the [<a number>] in front of the output lines? -- [1] denotes vector 1 (we'll get to defining vectors later); this is a built concept to R

Tip: if your screen gets all full of your experiments and you want a "clean slate", type CTRL+l (lower case "L")

Using R Projects is a great way to use R Studio. Once you've created an R Project, you can open it by double-clicking on the .Rproj file, and that will open a new R session where all of the files you create point to the directory where the .Rproj file is found. This makes it really easy to keep track of relevant files, and to work with collaborators (you can send your R Project to a collaborator and all of the files they need to run the code are in the R Project folder, self-contained).

Challenge #1: Creating a self-contained project

We're going to create a new project in RStudio:
1. Click the "File" menu button, then "New Project".
2. Click "New Directory".
3. Click "New Project".
4. Type in the name of the directory to store your project, e.g. "my_project".
5. If available, select the checkbox for "Create a git repository."
6. Click the "Create Project" button.

- simplest way to open rstudio and a project is thru your file system by double clicking on the .Rproj file in your poject folder.

Challenge #2: Opening an RStudio project through the file system

1. Exit RStudio.
2. Navigate to the directory where you created a project in Challenge 1.
3. Double click on the .Rproj file in that directory.

- now we have our project set up we can create an R script!

- **Go to FileNew FileR Script**

- ok who can tell me what a R Script is for?
- yes, we write our R code in it and it lets us run that code in a few ways
- we save our code in the file, etc.
- let's add some code and learn how to use R Scripts in RStudio

Challenge 3

Download the gapminder data from here. https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/gh-pages/_episodes_rmd/data/gapminder_data.csv

1. Download the file (right mouse click on the link above -> “Save link as” / “Save file as”, or click on the link and after the page loads, press Ctrl+S or choose File -> “Save page as”)
2. Make sure it’s saved under the name gapminder_data.csv
3. Save the file in the data/ folder within your project.

We will load and inspect these data later.

Running commands from an R script
- Click on "Run" button in R studio
- Cmd + Enter (Mac)
- Ctrl + Enter (Windows)

There's an analogy I really like for **what these different panes of R studio do**. It relates to cooking! You can think of the R script (top left pane) as the recipe you're following and/or writing. The Console (lower left pane) is where you're doing the actual cooking. The Environment (upper right pane) contains all the ingredients you're using and creating (e.g. your datasets and variables). And then the lower right pane has files (tells you where you're keeping your recipe and certain ingredients) and help files.

Variables
Assignment means to designate a variable and give it a value

The "<-" command assigns the item to the left of this command to the variable name to the right.
e.g. x <- 42 will assign 42 to the variable name x
Note that a single = will work similarly to <- but you'll see <- more often in R

R objects are variables and their values (you will see it in the environment pane to the right of your main pane)

Variable names

- - can contain letters, numbers, underscores, periods, but no spaces
- - they must start with a letter or period followed by a letter
- - you can't start a variable with a number! e.g. 47hello is not a good var name, but hello47 would work
- - variables beginning with a period are hidden variables
- e.g. this_is_my_long_var_name or thisIsMyLongVarName

Comments
- you can use # at the start of a line of code to comment it out (R will not try to evaluate what you've written in a commented-out line of code)
- it's good to take notes on what you've coded by adding in comments to your code! helps your future self and your collaborators

Packages

- these are bundles of functions that have been vetted by a group called CRAN
- install them with install.packages(): e.g. install.packages("dplyr")
- you only need to install once!
- then when you want to use the package, you can use the function library() to load the package into your R workspace and have access to those packages: e.g. library(dplyr)

Challenge 3

Which of the following are valid R variable names? (Add a plus after the correct answer(s))

min_height ++++++++++++++
max.height ++++++++
_age
.mass +++++
MaxLength +++++++
min-length
2widths ++
celsius2kelvin ++++++++++

Challenge 4

What will be the value of each variable after each statement in the following program?

```
mass <- 47.5
age <- 122
mass <- mass * 2.3
age <- age - 20
```

Challenge 5

Run the code from the previous challenge, and write a command to compare mass to age. Is mass larger than age?

Challenge 6

Install the following packages: ggplot2, gapminder
#install.packages(c("ggplot2", "gapminder"))
Note: the function "c()" means concatenate, so this is allowing us to run install.packages on both "ggplot2" and "gapminder". Can use c() to create a vector (see below)
You could also separately write
install.packages("ggplot2")
install.packages("gapminder")

Data Structures
- can see the data type of an object with function typeof()

- some data types: double, integer, logical, character
-vector: list/column of data that **has to be a same type**
- another helpful function: str() (stands for "structure") tells you the structure of the object. It'll show you some of (or all of, depending on the size of the object) what's in the object, and what data type it is.

here's a quick challenge
Assign the vector "quiz_vector" as follows?
quiz_vector <- c(2, 6, '3')

challenge: "What kind of data type is this vector?"
Answer: "character"
challenge: "Why?"
Answer: because there was one data value that was character data, so R converted the numbers to characters
coercion, is the way R converts data to conform to one type inside a vector

When you try to combine different types of data into one vector, R will "coerce" all of the items of the vector to be the same data type.
"coercion" is the term for the R programming language behavior of converting data types to being all one type inside a vector
Coercion rules: logical -> integer -> numeric -> complex -> character (where -> means "transformed into")
There are a variety of functions to force something to be a different type (e.g. as.numeric(), as.character())

You can wrap functions around each other (this will mean that the output of one function will be given as the input of the other function)
e.g. the line of code:
str(as.numeric(quiz_vector))
will first run as.numeric(quiz_vector), and then will run str() on the output of that

You can override existing objects (e.g. if you want to convert the format of a vector)
e.g. my_vector <- as.numeric(my_vector)

Data frames
a collection of vectors (there are many ways of making dataframes, but one way (which Tim used) is to use data.frame() to combine several vectors into a dataframe. When you do this, each vector will be a column in the dataframe)
A data frame is a spreadsheet in the R world
You can view your dataframe with the function View()
You can also click on it in the environment pane

Each column is a vector

~~break until 11:15~~

Data Frames and Subsetting

Reading in the gapminder data

There are at least three different ways to read in data:
If you followed along earlier on setting up a project and downloading your data into a folder named 'data/', you can run number 1:

1. **read the file from your data folder**

1. download the file in-code and then read in

```
download.file("https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/main/data/gapminder_data.csv", destfile = "data/gapminder_data.csv")

gapminder <- read.csv("data/gapminder_data.csv", stringsAsFactors = TRUE)
```

1. read directly from the internet

```
gapminder <- read.csv("https://raw.githubusercontent.com/swcarpentry/r-novice-gapminder/main/data/gapminder_data.csv", stringsAsFactors = TRUE)
```

Lasty, if you have a file type you're not sure about (SAS, SPSS, etc) you can use the 'Import Dataset'from the RStudio interface.

Factors: the data objects which are used to categorize the data and store it as levels. They can store both strings and integers. They are useful in the columns which have a limited number of unique values. These would be used in datasets that have demographic data such as age, sex, political party, etc

I know there was an outstanding question about factors so I'm going to try to answer it here. If you define a vector as a factor, say v <- factor(c("b", "a", "c")), the levels will automatically be coded in alphabetical order such that levels(v) should return "a" "b" "c", regardless of the order in which they appear in the vector. Under the hood, the levels are encoded as integers according to the order of the levels (so a = 1, b = 2, c = 3), which means from the computer's perspective the vector v should look like 2 1 3. This can be confirmed with the command str(v).

Vectors: Vector is a basic data structure in R.

- contains element of the same type. - Ttypehe data types can be logical, integer, double, character, complex or raw.

A vector's type can be checked with the typeof() function.
Another important property of a vector is its length. This is the number of elements in the vector and can be checked with the function length().

Challenge 8

It's good practice to also check the last few lines of your data and some in the middle. How would you do this?
Searching for ones specifically in the middle isn't too hard, but we could ask for a few lines at random. How would you code this?

-levels(gapminder$continent)
-str(gapminder)
-colnames(gapminder)
-summary(gapminder)
-head(gapminder)
-tail(gapminder,n=7)
-str(gapminder$lifeExp)
-sample(nrow(gapminder),5)
-dim(gapminder)

-Discussion on factor:
categories

Challenge 9

Read the output of str(gapminder) again; this time, use what you've learned about factors, lists and vectors, as well as the output of functions like colnames and dim to explain what everything that str prints out for gapminder means. If there are any parts you can't interpret, discuss with your neighbors!

Some useful operators:

- |: or
- %in%: match, e.g. A %in% B gives a TRUE or FALSE for each element in A that is in B
- &: and
- $: used to call column of a data frame by its column name, e.g. dataframe$columnname
- =: assigning value
- ==:

list of operators in R: https://www.statmethods.net/management/operators.html

Challenge 10

#notes
-[]: brackets for indexing purpose
-gapminder[1:3,]: show the first three rows
gapminder[1:3]: without the comma, it shows the columns
-head(gapminder[gapminder$year ==1957,]): get all columns of the year 1957
head(gapminder[gapminder$year > 1957 | gapminder$year <1987,]): get the year between 1957 and 1987
head(gapminder[gapminder$year %in% c(1957,1987),]):

Fix each of the following common data frame subsetting errors:

1. Extract observations collected for the year 1957

```
gapminder[gapminder$year = 1957,]
```

solution: gapminder[gapminder$year == 1957,]
2. Extract all columns except 1 through to 4
gapminder[,-1:4]

solution: gapminder[,-c(1:4)]
3. Extract the rows where the life expectancy is longer the 80 years
gapminder[gapminder$lifeExp 80]
solution: gapminder[gapminder$lifeExp > 80,]
4. Extract the first row, and the fourth and fifth columns (continent and lifeExp).
gapminder[1, 4, 5]
solution: gapminder[1,4:5]
5. Advanced: extract rows that contain information for the years 2002 and 2007
solution: gapminder[gapminder$year %in% c(2002,2007),]

- gapminder[gapminder$year == 2002 | gapminder$year == 2007,]

Challenge 11

1. Why does gapminder[1:20] return an error? How does it differ from gapminder[1:20,]?

2. Create a new data.frame called gapminder_small that only contains rows 1 through 9 and 19 through 23. You can do this in one or two steps.
Solutions:

1. gapminder is a data.frame so needs to be subsetted on two dimensions. gapminder[1:20,] subsets the data to give the first 20 rows and all columns.

2. gapminder_small <- gapminder[c(1:9, 19:23),]

Day 2

Teaching team:

Name / pronouns / Affiliation / email
Jamie Jamison / UCLA / Data Science Center / she, her
Maddi Cowen / UCLA / PhD Student in Ecology & Evolutionary Biology / she, her

Helpers:

Name / pronouns / Affiliation / email
Rick McCosh/he-him/UC San Diego/ rmccosh@ucsd.edu
Tim Dennis/ he-him / UCLA / timdennis@ucla.edu
Kenji Hayashi, he, him / UCLA / kthayashi@ucla.edu
Lisa McAulay / she,her / UCLA /emcaulay@library.ucla.edu

Learners, Please Sign In:

Name / pronouns / Affiliation / email

Cameron Ziegler / he him/ UCLA / cziegler661@gmail.com
Chelsea Blankenchip/she,her/UCSD/clblanke@health.ucsd.edu
Negli Gallardo/he, him/UNICAMP/n234066@dac.unicamp.br
Edward Ramos/./UCLA/weramos@ucla.edu
Reuven Sinensky / her, him / UCLA Anthropology / rsinensky@ucla.edu
Ashley Johnston /her,she/UCLA/amjohnston@g.ucla.edu
Wenyan Guan/her, she/UCM/wguan3@ucmerced.edu
Cyndi Gutierrez / her, she/ UCLA/cgutierrez00@g.ucla.edu
Arsh Malik/ He,Him / UCLA / arshmalik02@ucla.edu
Yan Zhou/She, her/UCLA/yz007@ucla.edu

Agenda

9:00 - 9:10: Welcome, logistics
9:10 - 9:20: Recap of yesterday (Jamie)
9:20 - 10:20: Introduction to dplyr (Jamie)
10:20 - 10:35: Breakout rooms - exercises
10:35 - 10:50: Go over exercises
10:50 - 11:00: Break
11:00 - 12:00: Introduction to ggplot2 (Maddi)
12:00 - 12:10: Break
12:10 - 12:40: ggplot2 continued (Maddi)
12:40 - 1:00: Wrap up and group discussion - "How could you use what we've learned in this workshop in your own research?" (feedback survey)

Lecture notes from Jamie: https://uc-carps-intro-r.netlify.app/
Lesson for today: https://swcarpentry.github.io/r-novice-gapminder/

- Dplyr & GGPLOT2

Open Lab (come with anything you want to work through--R, Python, unix, git, etc.) tomorrow 9/22 from 9-noon in this zoom:

- https://ucla.zoom.us/j/95343916630?pwd=M1NRUlZObTgyWTlBSG5WdHc3MHp4Zz09

Notes & Exercises (help us take notes)

Dplyr & ggplot2

Notes:

gapminder <- read.csv("data/gapminder_data.csv", stringsAsFactors = TRUE)
There is also a package for gapminder that you can install and then load with library(). That is another way to get access to your gapminder dataset.

Dplyr package

install.packages("dplyr") #surround dplyr with either single quotes or double quotes, but not one of each :)
library(dplyr)
- contains numerous really helpful functions to work with datasets (to pull out data that is relevant to you, to summarize it in nice ways, etc.)
- the functions we'll focus on today from this package: select(), filter(), group_by(), summarize(), mutate()

Select()
- use this to make a subset of your data where you specify only certain columns

year_country_gdp <- select(gapminder, year, country, gdpPercap)
select(gapminder, year, country, gdpPercap)
smaller_data <- select(gapminder, -continent) # all coulumns except continent

Pipe command
Keyboard shortcut for the pipe command (**%>%**) for Windows: Ctrl+Shift+M
chaining functions together
- this command is taking whatever is on the left of the %>% and giving that output as the first argument of the function to the right of the %>%
- for example:

year_country_gdp <- gapminder %>%

- select(year, country, gdpPercap)

View(year_country_gdp)

^^ this works because the first argument of select is the dataset (e.g. gapminder), so you can either write select(dataset, column, column) or you can write
dataset %>% select(column, column)

colnames(year_country_gdp) # display name of each colmn of data frame

Rename()
- use this to rename columns

tidy_gdp <- year_country_gdp %>% rename(gdp_Per_capita= gdpPercap)
#gdp_Per_capita is the new name of the column

filter()
-subset based on rows
- filter() selects certain rows that match the conditions you give it
gapminder %>% filter(continent == "Europe", year == 2007) %>% select (country, lifeExp)

group_by()
-gapminder %>% group_by(continent) %>% filter(year == 1957 | year == 1977)
- splits your dataset into different groups
- good visualization here:
https://swcarpentry.github.io/r-novice-gapminder/fig/13-dplyr-fig3.png

If you have a long command with lots of pipes, you can comment out some of the later lines if you want to test the first few lines of the command. ex:
gapminder %>%

- group_by(continent) # %>%
- # filter(year == 1957 | year == 1977)

summarize()
- Allows you to create a data frame that has columns with calcualted data
-mean() and min() are part of baseR (built in functions)
gapminder %>%

- group_by(continent) %>%
- summarize(mean_gdpPercap = mean(gdpPercap),
    - min_gdpPercap = min(gdpPercap) %>%
    - View

- n() is a really useful function to use with group_by and summarize (tells you how many lines per group). ex:
gapminder %>%

- group_by(continent) %>%
- summarize(number_of_lines = n())

count()
-returns number of observations
gapminder %>%

- filter(year == 2002) %>%
- count(continent)

#result shows the observations of 2002 by continent
gapminder %>%

- filter(year == 2002) %>%
- count(continent, sort=TRUE), name = 'num_observations_2002') %>%
- View()

mutate()
-Create new column that is added to your dataframe
-new data is saved only in the dataframe that is open in R, it does not save it to your file, so unless you save it, it will not be apart of your datafile next time you open R

gapminder %>%

- mutate(gdp_billion = gdpPercap * pop / 10^9)

```
gapminder %>%
  mutate(gdp_billion = gdpPercap * pop / 10^9) %>%
  group_by(continent, year) %>%
```

```
  summarize(
    mean_gdpPercap = mean(gdpPercap),
    mean_gdpbillion = mean(gdp_billion)) %>%
      View
```

-to save it, you need to assign the values to the new data frame.

Exercises for dplyr

https://uc-carps-intro-r.netlify.app/dplyr_challenges_solutions.html

Challenge 1

Write a single command (which can span multiple lines and includes pipes) that will produce a dataframe that has the African [or any other continent you pick] values for lifeExp, country and year, but not for the other Continents. How many rows does your data frame have, and why?

Challenge 2

Calculate the average life expectancy per country.
Which has the longest average life expectancy and which has the shortest average life expectancy?

Challenge 3

Create a new dataframe that has a new column with the ratio of life expectancy to GDP per capita. Keep only the country and ratio column. **Hint:** think about the order of operations!

Challenge 4

Calculate the average life expectancy in 2002 of 2 randomly selected countries for each continent. Then arrange the continent names in reverse order.
Hint: Use the dplyr functions arrange() and sample_n(), they have similar syntax to other dplyr functions.

Challenge 5

Filter for continent (your choice) AND (&) a life expectancy greater then 60
Then count the observatins for each group.

Notes on ggplot

Notes
-tidytuesday plots

```
install.packages('ggplot2')
library(ggplot2)
```

3 things:

- dataset,
- coordinate system (x,y),
- represent data points

```
ggplot(data = gapminder, mapping = aes(x=gdpPercap, y=lifeExp))  +
  geom_point()

  also works:
  ggplot(gapminder, aes(x=gdpPercap, y=lifeExp))  +
  geom_point()

gapminder %>%
ggplot(aes(x=gdpPercap, y=lifeExp)) + geom_point()
```

https://ggplot2.tidyverse.org/reference/
https://www.r-graph-gallery.com/ggplot2-package.html

```
gapminder %>%
  ggplot(data = gapminder, mapping = aes(x = year, y = lifeExp)) + geom_point()

gapminder %>%
  ggplot(mapping = aes(x = gdpPercap, y = lifeExp, color=continent)) + geom_point()

gapminder %>%
  ggplot(aes(x=year, y=lifeExp, color = continent, by = country)) +
  geom_line()
```

Add points on lines:

```
gapminder %>%
  ggplot(aes(x=year, y=lifeExp, color = continent, by = country)) +
  geom_line() +
  geom_point()

gapminder %>%
  ggplot(aes(x=year, y=lifeExp, by = country)) +
  geom_line(aes(color=continent)) +
  geom_point()
```

move points behind the lines:

```
gapminder %>%
  ggplot(aes(x=year, y=lifeExp, by = country)) +
  geom_point() +
```

```
  geom_line(aes(color=continent))

  gapminder %>%
  ggplot(aes(x=year, y=lifeExp, color = continent, by = country)) +
  geom_line(aes(color=continent))  +
  geom_point(color="blue")

gapminder %>%
  ggplot(aes(x=gdpPercap, y=lifeExp)) +
  geom_point() +
  scale_x_log10()

  # change transparancy
gapminder %>%
  ggplot(aes(x=gdpPercap, y=lifeExp)) +
  geom_point(alpha=0.5) +
  scale_x_log10()

# add line to show relationship between x and y

gapminder %>%
  ggplot(aes(x=gdpPercap, y=lifeExp)) +
  geom_point(color="blue", alpha=0.5) +
  scale_x_log10() +
  geom_smooth(method="lm", color="black")

#facet wrap

americas <- gapminder %>% filter(continent == "Americas")
americas %>%
ggplot(aes(x = year, y = lifeExp)) +
geom_point()+
geom_line() +
facet_wrap( ~ country)
```

Exercises for ggplot2

Challenge 6

Modify the example so that the figure shows how life expectancy has changed over time:

- `ggplot(data = gapminder, mapping = aes(x = gdpPercap, y = lifeExp)) + geom_point()`

Challenge 7

In the previous examples and challenge we've used the aes function to tell the scatterplot geom about the x and y locations of each point. Another *aesthetic* property we can modify is the point *color*. Modify the

code from the previous challenge to color the points by the “continent” column. What trends do you see in the data? Are they what you expected?

```
gapminder %>%
  ggplot(aes(x=year, y=lifeExp, color= continent))+geom_point()
```

Challenge 8

In the previous example (lifeExp by gdpPercap plot), modify the shape and size of the points on the point layer, and color the points by continent.

```
gapminder %>%
  ggplot(aes(x = gdpPercap, y = lifeExp)) +
  geom_point(alpha = 0.3) +
  scale_x_log10() +
  geom_smooth(method = "lm")

  gapminder %>%
  ggplot(aes(x=gdpPercap, y=lifeExp, color = continent)) +
  geom_point(alpha=0.3) +
  scale_x_log10() +
  geom_smooth(method="lm", color="black")

gapminder %>%
  ggplot(aes(x=gdpPercap, y=lifeExp))+
  geom_point(aes(color=continent), alpha=0.3,
             shape='triangle',size=5)+
  scale_x_log10()+
  geom_smooth(method = 'lm')
```

This challenge asked you to guess at what keywords to use to change the shape. You can look it up using the cheat sheet

Challenge 9

Modify geom_smooth so that there are different trend lines by continent.

```
gapminder %>%
  ggplot(aes(x=gdpPercap, y=lifeExp))+
  geom_point(aes(color=continent), alpha=0.3,
             shape='triangle',size=2)+
  scale_x_log10()+
  geom_smooth(aes(by=continent), method = 'lm', size=1.5)
```

ggplot cheat sheet: https://www.maths.usyd.edu.au/u/UG/SM/STAT3022/r/current/Misc/data-visualization-2.1.pdf

Using your data subsetting skills:

create a dataframe called americas
which consists of all the countries in the continent "Americas"

BREAK -- 12:00 - 12:10

Challenge 10

Combining dplyr and ggplot: Make the Americas facet_wrap plot starting from the dataset gapminder, without creating the intermediate variable "americas".

```
gapminder %>%
  filter(continent == "Americas") %>%
  ggplot(aes(x=year, y=lifeExp, by = country)) +
  geom_point() +
  geom_line() +
  facet_wrap( ~ country)  +
  theme(axis.text.x = element_text(angle=45, color="blue"))
```

Challenge 11

Plot population (y) against GDP (x) (not the per capita GDP) for only data from the year 2007, and color by continent.

Note: GDP = pop * gdpPercap

gapminder %>%

- mutate(gdp = pop * gdpPercap) %>%
- filter(year == 2007) %>%
- ggplot(aes(x = gdp, y = pop, color = continent)) +
- geom_point()

```
gapminder %>%
  # add variable for gdp
  mutate(gdp = pop * gdpPercap) %>%
  filter(year==2007) %>%
  ggplot(aes(x=gdp, y=pop, color=continent)) +
  geom_point() +
  facet_wrap( ~ continent)
```

Reflections

Something that you learned from the workshop
Will you be using this in your research, and if so, how?

One of the
I have used R in the past
demography and i think we'll be doing a lot with ggplot, so that section of the workshop will really help

dplyr will help me manipulate data
dplyr functions helps me work with my data and organize it

I'm new to R and quantitative studies in general. I read some intro readings on R but it wasn't as intangible as having this workshop.It helps a lot to go over the codes line by line. I feel more comfortable learning and using R now.

R is a better option since it is open source
I don't feel afriad about R stduio envirnonment -- at the beginning it is very challenging
now I have a better idea about how to sort data,
ggplot is very helpful - nice graphs and views of your data
right now, i am inBRazil and what I am trying to studying the social determinants of health related to pregnacny ; i can run a lot of the analysis through R

Everything that was taught in the past two days was really useful; the plotting is very helpful
The options make things look nice and that is very useful
I learned some new tips and tricks too

Learning from the exercises because there are things that

I learned a tool called R Markdown (another R feature) for making nice reports

Lesson notes are in R Markdown and the Lesson is in R markdown
(if you look at the lesson repository

End of Workshop

Feedback form: PLEASE Complete -- your feedback helps us grow and make sure we keep the good parts of the workshop while we continue to improve our teaching!
https://docs.google.com/forms/d/e/1FAIpQLSdiyAxUE3o0WnWOq4LMfPbDPqlG0akkGwOSNMf6NkGnINnmfw/viewform

Longer post-workshop survey for the Carpentries program:
https://carpentries.typeform.com/to/UgVdRQ?slug=2021-09-13-uc-collab - this helps the global Carpentries teaching organization improve these workshops

Tomorrow's Open Lab

(all technologies, any question, bring your own data or your own problem)

Zoom link: (9-12) https://ucla.zoom.us/j/95343916630?pwd=M1NRUlZObTgyWTlBSG5WdHc3MHp4Zz09