Pre-work: Intro to R, RStudio and Tidyverse

Solutions

8/23/2021

The *purpose* of the pre-work is two-fold. First, we want you to make sure you have R installed and running before day 1. Second, we want you to start playing with code as soon as possible. Experimenting and tinkering with code is how you hone these skills!

Make sure you *watch the intro videos* intro part 1 and intro part 2 so you have context.

We *included a text file* with the code from this tutorial called *00_lab_intro_to_R_and_tidyverse.R*. If you copy and paste from this pdf, the syntax sometimes gets messed up and the code will not work. We have solutions available as well which you can check if you get stuck–but try to avoid them.

Installation completion

If you have not yet, install R and R Studio. We have instructions on how to do so here.

1. If you have not yet, run the following code to install the `tidyverse`.

   ```
   install.packages("tidyverse")
   ```

2. This code returns an error. Why? Fix it and install the packages. We'll see what they're for in the next lecture

We need to use strings (quotes) to install packages.

~~~
```r
install.packages("haven")
install.packages("readxl")
```
~~~

3. In order to have access to `tidyverse` functions, you need to load the library.

   ```
   library("tidyverse")
   ```

4. Run the following code to make sure you have installed successfully. Then, assign the resulting tibble to the name `big_storms` using `<-`.

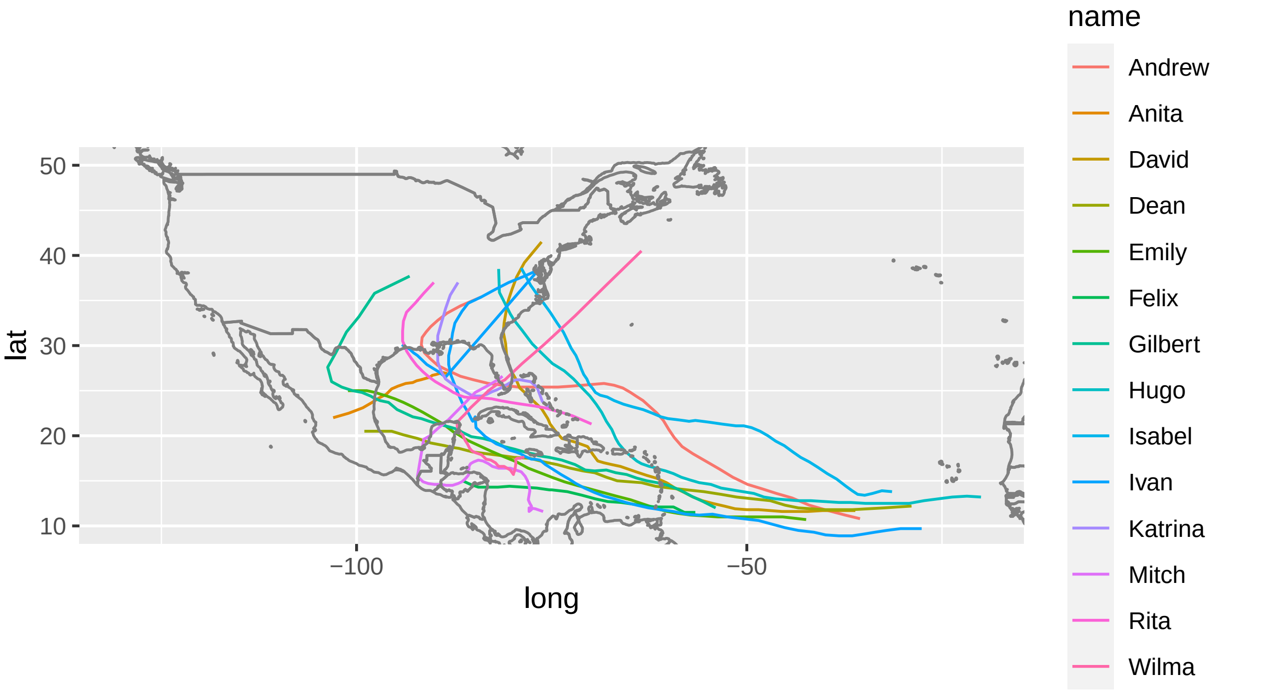
   ```
   # storms is a dataset that comes with the tidyverse
   # use View(storms) to see the full dataset
   big_storms <-
   storms %>%
     group_by(name, year) %>%
     filter(max(category) == 5)
   ```

5. This code makes the map seen below. If you run it you will get a message that says your code is missing a required package. R 4.1 will ask you to install the package. Say yes. [This package is not essential, so if you're using a lot of time (> 5 min) trying to install it. It's okay to move on.]

If you don't let R install the package for you. You need to run install.packages("maps")

6. The map is nice, but we cannot see where the storm paths begin. Adjust the code to expand our view to capture the full path.

```
  ggplot(aes(x = long, y = lat, color = name), data = big_storms) +
        geom_path() +
        borders("world") +
        coord_quickmap(xlim = c(-130, -20), ylim = c(10, 50))
```



Processing and analyzing data, an example:

Now you’ll see an abridged version of the data exploration I undertook to make the racial disparities of Covid-19 plot discussed in lecture.

As mentioned before, the code below is in the file `00_lab_intro_to_R_and_tidyverse.R`. (This code has syntax that copy-paste will mess up!) ’

Run the code to download and preprocess the data. The data will take a few minute to download since it is large-ish.

```
# Stuff after a # is a comment and will not affect what the code does.

covid_data <-
  # reads data directly from CDC website
  read_csv(paste0("https://data.cdc.gov/api/views/qfhf-uhaa/rows.csv?",
                  "accessType=DOWNLOAD&bom=true&format=true%20target="),
           col_types = cols(Suppress = col_character())) %>%
  # change column names from human readable to more code friendly
  # and create a new column (called diff) based on older ones
    mutate(week = `Week Ending Date`,
           race_ethnicity = `Race/Ethnicity`,
           n_deaths = `Number of Deaths`,
           diff = `Difference from 2015-2019 to 2020`,
           #  notice that diff + expected deaths = n_deaths
           expected_deaths = n_deaths - diff,
           perc_diff = `Percent Difference from 2015-2019 to 2020`,
           year = MMWRYear,
           week_no = MMWRWeek + ifelse(year == "2021", 52, 0),
           jurisdiction = Jurisdiction,
           state = `State Abbreviation`
           )  %>%
    # filter ROWs of data to make it more manageable
    filter(`Time Period` %in% c("2020", "2021"), Outcome == "All Cause", Type != "Unweighted") %>%
    # select COLUMNS of data to make it more manageable
    select(jurisdiction, state, week, year, week_no,
           race_ethnicity, n_deaths, expected_deaths, diff, perc_diff)
```

1. Now that you have `covid_data` loaded. Use `View()` to examine the data.

Analysis is easier with data that is "tidy", by which we mean

- each row is an observation
- each columns is a variable.

`covid_data` is tidy. What is the unit of observation for each row?

Observations in `covid_data` are uniquely determined by `state`, `week`, `race_ethnicity`. We could swap out `jurisdiction` for `state` and `week_no` for `week`

2. R has a built in function to summarize each column called `summary()`. Try calling the function by typing `summary(covid_data)`. Your data goes in the first position in the parenthesis. It's not the prettiest, but it gives you some information with little code. Look at `n_deaths`. What information are you getting from `summary()`.

We see the minimum and maximum values, the value of the first, second (median) and third quartiles along with the mean and number of NAs.

3. In `summary()`, you saw that the Max values are quite high compared to the third quartile. That's worthy of investigation. Run the following code and explain why we have some super large values in our data set.

```
covid_data %>%
  filter(n_deaths > 10000)
```

4. Ah, that's actually quite useful. We have US data already aggregated for us. Let's start by aggregating the data so we have a single number for each `race_ethnicity`. Use the `<-` to assign the results of the code to the name `us_deaths_by_race`. (The code is available in the .R file.)

```
us_deaths_by_race <-
covid_data %>%
  filter(state == "US") %>%
  group_by(race_ethnicity) %>%
  summarize(expected_deaths = sum(expected_deaths, na.rm = TRUE),
            total_additional_deaths = sum(diff, na.rm = TRUE),
            percent_diff = round(100 * total_additional_deaths / expected_deaths)
            )
```

5. It's often nice to see the data. Run the code to make a bar graph.

```
us_deaths_by_race %>%
  ggplot(aes(y = race_ethnicity, x = percent_diff)) +
  geom_col() +
labs(x = "Percent above expected death count",
     y = "",
     title = "Racial disparities of Covid-19 (USA)" )
```

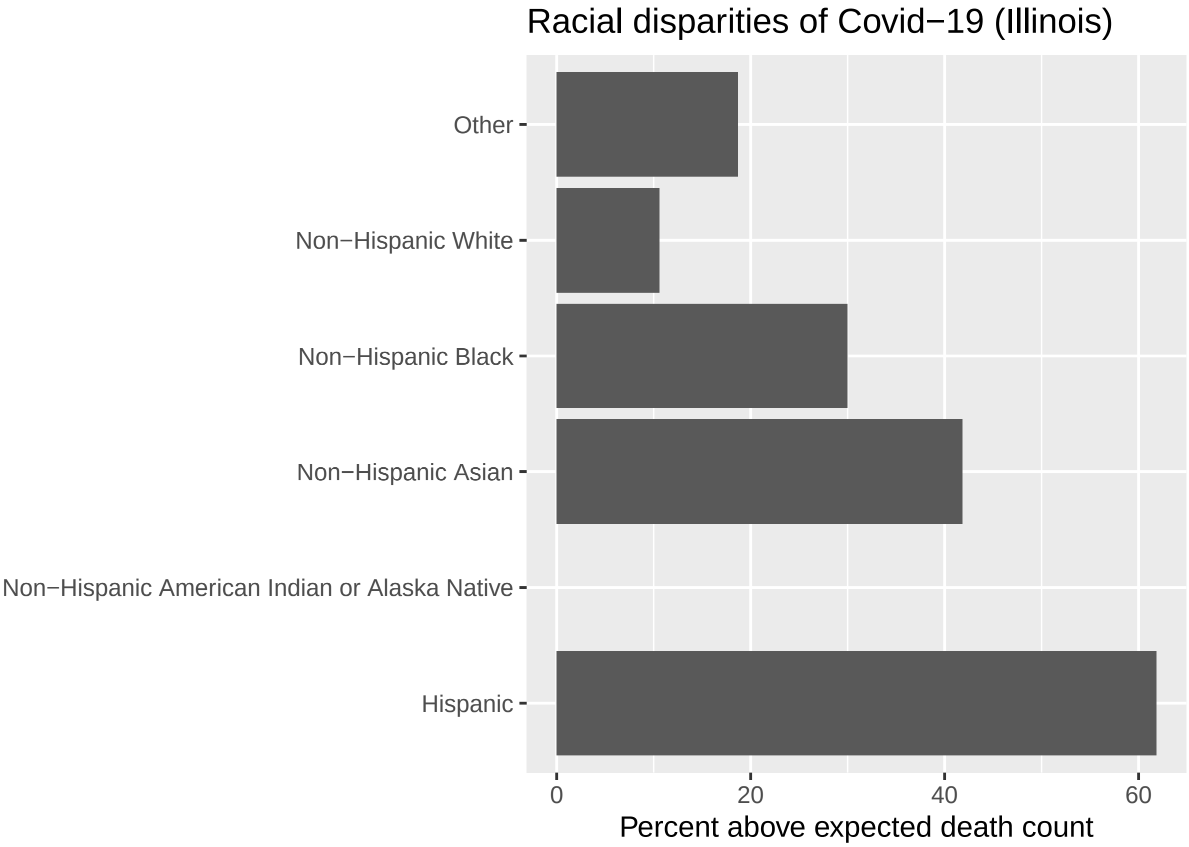
6. Remove the + `labs()` function from the above code. Describe what `labs()` does. What is the default plotting behavior?

labs() allows you to customize labels on the graph. If we don't customize, R uses column names as labels for axes and there is no title.

7. Remake the graph but for the state of IL. You should be able to do this by changing the above code slightly.

```
covid_data %>%
  # Filter the state
  filter(state == "IL") %>%
  group_by(race_ethnicity) %>%
  summarize(expected_deaths = sum(expected_deaths, na.rm = TRUE),
            total_additional_deaths = sum(diff, na.rm = TRUE),
            percent_diff = 100 * total_additional_deaths / expected_deaths
            )  %>%
  ggplot(aes(y = race_ethnicity, x = percent_diff)) +
  geom_col() +
labs(x = "Percent above expected death count",
     y = "",
     # Make sure to adjust the title
     title = "Racial disparities of Covid-19 (Illinois)" )
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```



8. Seeing this graph, I thought to myself. I can tell a better story and understand the data better if I included the time series element. This would allow us to see how the disparities change over time.

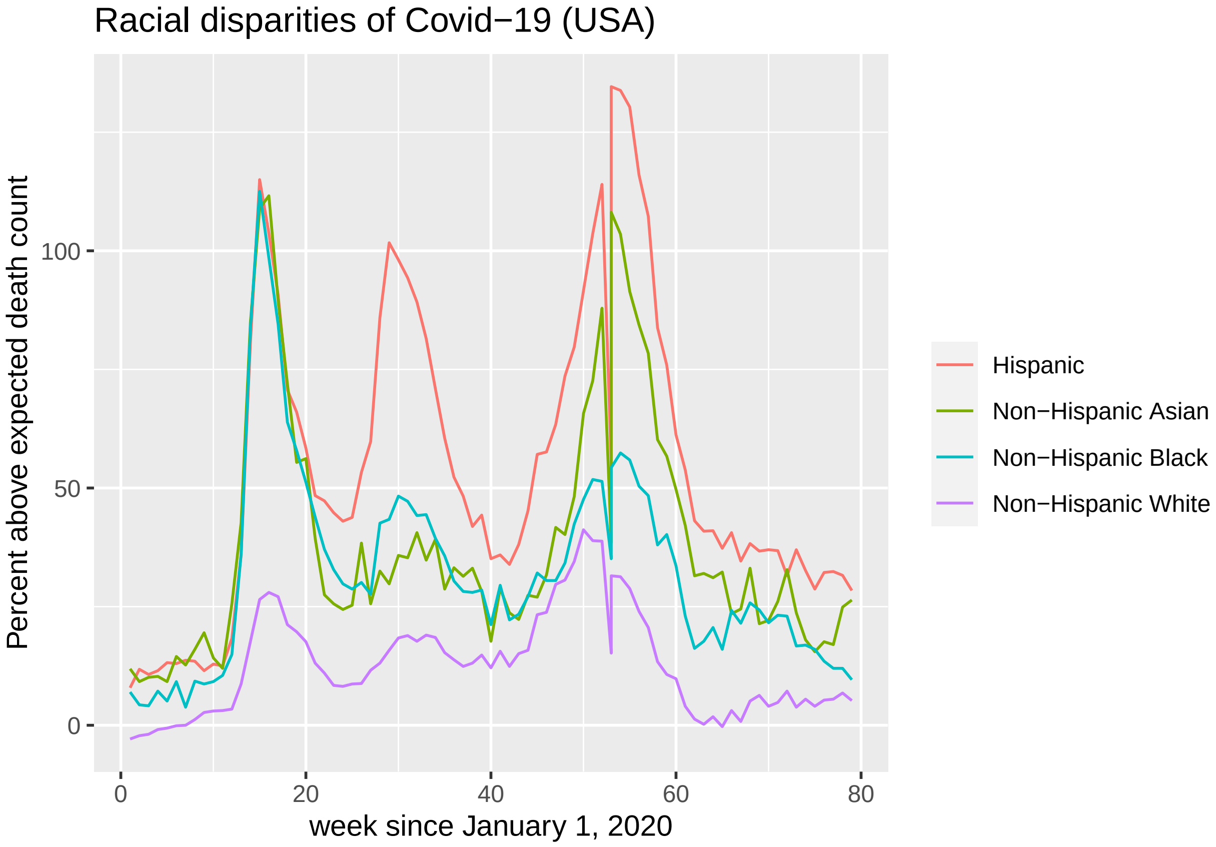
see 10

9. Notice that groups with smaller populations have noisier time-series. The unfortunately-named "Other" time series is sawtoothed, while the "Non-Hispanic White" line is relatively smooth. Remove the # from the code above and run it again. # is used for "comments" in R. R ignores code that comes after # until a new line is started.

see 10

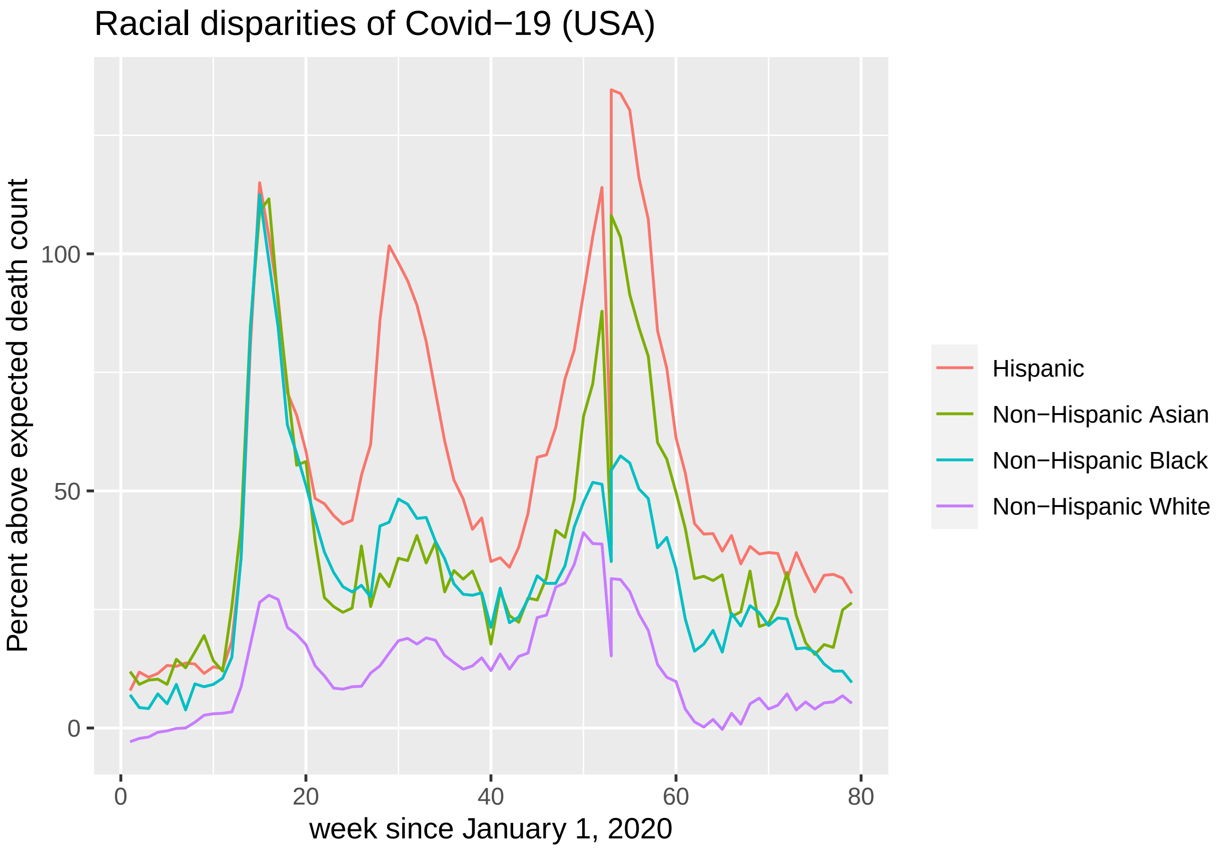
10. Finally, you might notice that it looks like there's a suspicious drop off in our "Percent above expected death count" after week 80. With a little investigation you discover that the CDC updates these counts over time and the last 6 weeks are likely to increase substantially. Add this line `filter(week_no < 80) %>%` to your code in an appropriate place (i.e. before making the plot!) to remove the misleading data.

```
covid_data %>%
  filter(state == "US") %>%
  filter(week_no < 80) %>%
  filter(race_ethnicity %in%
   c("Hispanic", "Non-Hispanic White", "Non-Hispanic Black", "Non-Hispanic Asian")) %>%
  ggplot(aes(x = week_no, y = perc_diff, color = race_ethnicity)) +
  geom_line()  +
labs(y = "Percent above expected death count",
     x = "week since January 1, 2020",
     title = "Racial disparities of Covid-19 (USA)",
     color = "" )
```



you could combine all the filter statements together too! I think this is more readable

```
covid_data %>%
  filter(state == "US",
         week_no < 80,
         race_ethnicity %in% c("Hispanic", "Non-Hispanic White",
                               "Non-Hispanic Black", "Non-Hispanic Asian")) %>%
  ggplot(aes(x = week_no, y = perc_diff, color = race_ethnicity)) +
  geom_line()  +
labs(y = "Percent above expected death count",
     x = "week since January 1, 2020",
     title = "Racial disparities of Covid-19 (USA)",
     color = "" )
```


Racial disparities of Covid−19 (USA)
Percent above expected death count
week since January 1, 2020
Hispanic
Non−Hispanic Asian
Non−Hispanic Black
Non−Hispanic White