

Code-a-long-1.3

Analyze Data with R

Goals

- Continue gaining practice loading and using data sets in R
- Create subsets of data based upon certain conditions (a.k.a. filtering)
- Learn how to generate summary statistics using the `group_by/summarize` functions

Loading Data

Let's start by reviewing some concepts from the last lesson:

```
# we need to make the tidyverse available with the library function:
```

```
library(tidyverse)
```

— Attaching core tidyverse packages — tidyverse 2.0.0 —

```
✓ dplyr      1.1.2    ✓ readr      2.1.4
✓ forcats    1.0.0    ✓ stringr    1.5.0
✓ ggplot2    3.4.2    ✓ tibble     3.2.1
✓ lubridate  1.9.2    ✓ tidyr      1.3.0
✓ purrr      1.0.2
```

— Conflicts — tidyverse_conflicts() —

```
* dplyr::filter() masks stats::filter()
```

```
* dplyr::lag()     masks stats::lag()
```

```
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# load the dataset to a variable
```

```
teamAntarcticaData<-read_csv("teamAntarcticaData.csv")
```

Rows: 75 Columns: 12

— Column specification —

Delimiter: ","

chr (7): Timestamp, school, swim, animals, parkaColor, teamFlag, distance

dbl (5): fishing, cold, remote, bedsideManner, cooking

```
i Use `spec()` to retrieve the full column specification for this data.
```

```
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# view it
```

```
teamAntarcticaData
```

```
# A tibble: 75 × 12
  Timestamp      school fishing swim   cold animals remote parkaColor teamFlag
  <chr>          <chr>    <dbl> <chr> <dbl> <chr>    <dbl> <chr>    <chr>
1 8/30/2022 16:0... Unive...     1 Yes     4 Yes     4 Gold   Penguin
2 8/30/2022 16:0... Unive...     2 Yes     4 Yes     5 Blue   Bear
3 8/30/2022 16:0... Unive...     2 Yes     4 Yes     3 Green  Penguin
4 8/30/2022 16:0... Unive...     1 Yes     1 Yes     1 Blue   Seal
5 8/30/2022 16:0... Unive...     1 Yes     3 Yes     3 White  Sea Spi...
6 8/30/2022 16:0... Unive...     1 Yes     3 Yes     3 hot pink Penguin
7 8/30/2022 16:0... Unive...     1 Yes     2 Yes     3 Blue   Sea Spi...
8 8/30/2022 16:0... Unive...     2 Yes     2 Yes     4 Blue   Penguin
9 8/30/2022 16:0... Unive...     2 Yes     2 Yes     5 White  Bear
10 8/30/2022 16:0... Unive...     5 Yes     5 Yes     5 Blue   Penguin
# i 65 more rows
# i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>
```

Sub-setting Data

In the last lesson, we looked at descriptive statistics for columns of data for the entire data set. But what if we were interested in pursuing answers to specific questions. Here's one:

- How does the cold tolerance differ for students at Lewis & Clark vs students at the University of Arizona?

One strategy we could take is to find the average cold tolerance responses for the LC students, and compare it to the average of the UA students. So we'll need to create two subsets of the original data set, using the filter function. The filter function works like this:

```
mySubset<-filter(.data=myDataFrame, columnName=="some value")
```

- 1st argument (.data) - identify the original, unfiltered data set
- 2nd argument - include one or more conditions

Now let's use the filter function to create a subset of data that only includes University of Arizona students:

```
uaStudents<-filter(.data=teamAntarcticaData, school=="University of Arizona")

uaStudents
```

```
# A tibble: 28 × 12
  Timestamp      school fishing swim   cold animals remote parkaColor teamFlag
  <chr>          <chr>    <dbl> <chr> <dbl> <chr>    <dbl> <chr>    <chr>
1 8/30/2022 16:0... Unive...     1 Yes     4 Yes     4 Gold   Penguin
2 8/30/2022 16:0... Unive...     2 Yes     4 Yes     5 Blue   Bear
3 8/30/2022 16:0... Unive...     2 Yes     4 Yes     3 Green  Penguin
4 8/30/2022 16:0... Unive...     1 Yes     1 Yes     1 Blue   Seal
5 8/30/2022 16:0... Unive...     1 Yes     3 Yes     3 White  Sea Spi...
6 8/30/2022 16:0... Unive...     1 Yes     3 Yes     3 hot pink Penguin
```

```

7 8/30/2022 16:0... Unive...      1 Yes      2 Yes      3 Blue      Sea Spi...
8 8/30/2022 16:0... Unive...      2 Yes      2 Yes      4 Blue      Penguin
9 8/30/2022 16:0... Unive...      2 Yes      2 Yes      5 White     Bear
10 8/30/2022 16:0... Unive...      5 Yes      5 Yes      5 Blue      Penguin

```

```
# i 18 more rows
```

```
# i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>
```

Below, create a subset of Lewis & Clark students, assign it to the variable `lcStudents`, and print it to the screen:

```
#create lcStudents below
```

```
lcStudents<-filter(.data=teamAntarcticaData, school=="Lewis & Clark College")
```

```
lcStudents
```

```
# A tibble: 47 × 12
```

	Timestamp <chr>	school <chr>	fishing <dbl>	swim <chr>	cold <dbl>	animals <chr>	remote <dbl>	parkaColor <chr>	teamFlag <chr>
1	8/31/2022 15:2...	Lewis...	2	Yes	4	Yes	1	White	Sea Spi...
2	8/31/2022 15:3...	Lewis...	1	Yes	3	Maybe	4	Blue	Penguin
3	8/31/2022 15:4...	Lewis...	5	Yes	5	Yes	5	Green	Bear
4	8/31/2022 16:0...	Lewis...	2	Yes	4	Yes	4	Blue	Seal
5	8/31/2022 16:3...	Lewis...	3	Yes	4	Yes	5	Black	Penguin
6	8/31/2022 17:0...	Lewis...	1	Yes	3	Yes	2	Blue	Penguin
7	8/31/2022 17:0...	Lewis...	4	Yes	5	Yes	5	White	Sea Spi...
8	8/31/2022 17:0...	Lewis...	5	Yes	5	Yes	5	Blue	Bear
9	8/31/2022 17:0...	Lewis...	2	Yes	4	Yes	4	Green	Penguin
10	8/31/2022 17:0...	Lewis...	2	Yes	2	Yes	3	Blue	Seal

```
# i 37 more rows
```

```
# i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>
```

So now we have two smaller data sets - one with U of A students (`uaStudents`), and one with LC students (`lcStudents`). How might we calculate the average cold tolerance of each? Using strategies from last week, try doing so below:

```
mean(uaStudents$cold)
```

```
[1] 3.285714
```

```
mean(lcStudents$cold)
```

```
[1] 3.425532
```

Relational operators

In the `filter` example above, we used `"=="` as part of our condition argument. The double equals is an example of a relational operator - it's a character (or multiple characters) that represents a logical action

or process. Practically speaking, the double equals means “is this field equal to this value?”. If the answer is “TRUE”, then the row is included as part of the filtered data set.

Here are some other relational operators:

- > (greater than)
- < (less than)
- <= (less than or equal to)
- >= (greater than or equal to)
- != (not equal to)

In the `filter` function, relational operators are used to define a condition.

Let’s say we’re interested in creating a subset of data that includes students with a self-reported aptitude in fishing of 4 or 5 (the students we should recruit to catch our fish). Create a subset of data called `goodFishing` that contains this list, and print to the screen:

```
# create goodFishing below

goodFishing<-filter(.data=teamAntarcticaData, fishing>3)

goodFishing
```

A tibble: 11 × 12

	Timestamp <chr>	school <chr>	fishing <dbl>	swim <chr>	cold <dbl>	animals <chr>	remote <dbl>	parkaColor <chr>	teamFlag <chr>
1	8/30/2022 16:0...	Unive...	5	Yes	5	Yes	5	Blue	Penguin
2	8/30/2022 16:1...	Unive...	5	Yes	4	Maybe	3	White	Bear
3	8/30/2022 16:1...	Unive...	4	Yes	3	Yes	4	Orange	Sea Spi...
4	8/30/2022 16:1...	Unive...	4	Yes	3	Yes	5	Orange	Seal
5	8/30/2022 16:1...	Unive...	4	Yes	4	Yes	5	White	Penguin
6	8/31/2022 15:4...	Lewis...	5	Yes	5	Yes	5	Green	Bear
7	8/31/2022 17:0...	Lewis...	4	Yes	5	Yes	5	White	Sea Spi...
8	8/31/2022 17:0...	Lewis...	5	Yes	5	Yes	5	Blue	Bear
9	8/31/2022 20:4...	Lewis...	5	Yes	3	Yes	4	Black	Bear
10	9/1/2022 13:50...	Lewis...	5	Yes	2	Yes	1	Orange	Penguin
11	9/1/2022 19:41...	Lewis...	5	Yes	4	Yes	5	Orange	Sea Spi...

i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>

Let’s say we also want to create a subset of data that includes students who are not particularly strong swimmers. Create a subset of data below called `nonSwimmers` that include students who did not answer “Yes” on the swimming question:

```
# create nonSwimmers below
```

```
nonSwimmers<-filter(.data=teamAntarcticaData, swim !="Yes")
```

Logical Operators

There may be cases in which we want to filter our dataset based on more than one condition. In these cases, we would use logical operators. Maybe we want to find the best University of Arizona chefs, or the students who want blue or orange parkas. Here are the main logical operators:

- & (and)
- | (or)

In the `filter` function, logical operators are used to join conditions together.

Here's an example of how to use a logical operator with the `filter` function:

```
uaChefs<-filter(.data=teamAntarcticaData, school=="University of Arizona" & cooking>3)
uaChefs
```

```
# A tibble: 16 × 12
```

	Timestamp <chr>	school <chr>	fishing <dbl>	swim <chr>	cold <dbl>	animals <chr>	remote <dbl>	parkaColor <chr>	teamFlag <chr>
1	8/30/2022 16:0...	Unive...	1	Yes	4	Yes	4	Gold	Penguin
2	8/30/2022 16:0...	Unive...	2	Yes	4	Yes	5	Blue	Bear
3	8/30/2022 16:0...	Unive...	2	Yes	4	Yes	3	Green	Penguin
4	8/30/2022 16:0...	Unive...	1	Yes	3	Yes	3	hot pink	Penguin
5	8/30/2022 16:0...	Unive...	1	Yes	2	Yes	3	Blue	Sea Spi...
6	8/30/2022 16:0...	Unive...	2	Yes	2	Yes	4	Blue	Penguin
7	8/30/2022 16:0...	Unive...	2	Yes	2	Yes	5	White	Bear
8	8/30/2022 16:0...	Unive...	5	Yes	5	Yes	5	Blue	Penguin
9	8/30/2022 16:1...	Unive...	3	Yes	3	Yes	3	Orange	Penguin
10	8/30/2022 16:1...	Unive...	4	Yes	3	Yes	4	Orange	Sea Spi...
11	8/30/2022 16:1...	Unive...	4	Yes	3	Yes	5	Orange	Seal
12	8/30/2022 16:1...	Unive...	1	Yes	4	Yes	3	Black	Penguin
13	8/30/2022 16:1...	Unive...	3	Yes	5	Yes	5	green	Sea Spi...
14	8/30/2022 16:1...	Unive...	2	Yes	2	Yes	4	Black	Bear
15	8/30/2022 16:1...	Unive...	1	Yes	3	Yes	3	Black	Penguin
16	8/30/2022 16:3...	Unive...	2	No	1	Yes	4	White	<NA>

```
# i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>
```

Below, use the `filter` function to create a data subset of students who want blue **or** orange parkas. Assign it to the variable `blueOrangeParkas`, and print to the screen.

```
blueOrangeParkas<-filter(.data=teamAntarcticaData, parkaColor=="Blue" | parkaColc
```

```
# A tibble: 32 × 12
```

	Timestamp	school	fishing	swim	cold	animals	remote	parkaColor	teamFlag
	<chr>	<chr>	<dbl>	<chr>	<dbl>	<chr>	<dbl>	<chr>	<chr>
1	8/30/2022 16:0...	Unive...	2	Yes	4	Yes	5	Blue	Bear
2	8/30/2022 16:0...	Unive...	1	Yes	1	Yes	1	Blue	Seal
3	8/30/2022 16:0...	Unive...	1	Yes	2	Yes	3	Blue	Sea Spi...
4	8/30/2022 16:0...	Unive...	2	Yes	2	Yes	4	Blue	Penguin
5	8/30/2022 16:0...	Unive...	5	Yes	5	Yes	5	Blue	Penguin
6	8/30/2022 16:1...	Unive...	3	Yes	3	Yes	3	Orange	Penguin
7	8/30/2022 16:1...	Unive...	4	Yes	3	Yes	4	Orange	Sea Spi...
8	8/30/2022 16:1...	Unive...	4	Yes	3	Yes	5	Orange	Seal
9	8/30/2022 16:1...	Unive...	2	I ca...	5	Yes	4	Blue	Penguin
10	8/30/2022 16:1...	Unive...	2	Yes	3	Yes	3	Blue	Penguin

```
# i 22 more rows
```

```
# i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>
```

The "Pipe"

The Tidyverse introduced a new convention to R called the "pipe":

```
%>%
```

or

```
|>
```

The purpose of the pipe is to string functions and data together. You can think of it as sort of the glue that joins pieces of an assembly line together. Another way to think of it is to read it as "AND THEN".

Below we can rewrite a command using the filter function with the pipe. After the assignment symbol (<-) we start with the data set, followed by the pipe, followed by the filter function. What's different about the arguments in the filter function in this case?

```
uaStudents2<-teamAntarcticaData |> filter(school=="University of Arizona")
```

```
uaStudents2
```

```
# A tibble: 28 × 12
```

	Timestamp	school	fishing	swim	cold	animals	remote	parkaColor	teamFlag
	<chr>	<chr>	<dbl>	<chr>	<dbl>	<chr>	<dbl>	<chr>	<chr>
1	8/30/2022 16:0...	Unive...	1	Yes	4	Yes	4	Gold	Penguin
2	8/30/2022 16:0...	Unive...	2	Yes	4	Yes	5	Blue	Bear
3	8/30/2022 16:0...	Unive...	2	Yes	4	Yes	3	Green	Penguin
4	8/30/2022 16:0...	Unive...	1	Yes	1	Yes	1	Blue	Seal
5	8/30/2022 16:0...	Unive...	1	Yes	3	Yes	3	White	Sea Spi...
6	8/30/2022 16:0...	Unive...	1	Yes	3	Yes	3	hot pink	Penguin
7	8/30/2022 16:0...	Unive...	1	Yes	2	Yes	3	Blue	Sea Spi...
8	8/30/2022 16:0...	Unive...	2	Yes	2	Yes	4	Blue	Penguin

```

9 8/30/2022 16:0... Unive...      2 Yes      2 Yes      5 White      Bear
10 8/30/2022 16:0... Unive...      5 Yes      5 Yes      5 Blue      Penguin

```

i 18 more rows

i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>

Try using the pipe in the code chunk below to create a data subset of students who answered "Maybe" in the animals question (call the variable `maybeAnimals`). Print it to the screen as well.

```

# create and print maybeAnimals:

maybeAnimals <-teamAntarcticaData |> filter(animals=="Maybe")

maybeAnimals

```

A tibble: 8 × 12

	Timestamp	school	fishing	swim	cold	animals	remote	parkaColor	teamFlag
	<chr>	<chr>	<dbl>	<chr>	<dbl>	<chr>	<dbl>	<chr>	<chr>
1	8/30/2022 16:09...	Unive...	1	Yes	4	Maybe	3	purple	Seal
2	8/30/2022 16:10...	Unive...	1	I ca...	3	Maybe	2	Black	Penguin
3	8/30/2022 16:10...	Unive...	5	Yes	4	Maybe	3	White	Bear
4	8/31/2022 13:04...	Unive...	2	Yes	4	Maybe	5	Black	Penguin
5	8/31/2022 15:33...	Lewis...	1	Yes	3	Maybe	4	Blue	Penguin
6	8/31/2022 18:57...	Lewis...	1	Yes	3	Maybe	3	Orange	Bear
7	8/31/2022 22:17...	Lewis...	1	Yes	2	Maybe	4	Blue	Penguin
8	8/31/2022 22:23...	Lewis...	2	Yes	4	Maybe	2	Blue	Sea Spi...

i 3 more variables: distance <chr>, bedsideManner <dbl>, cooking <dbl>

Generating summary statistics with `group_by` / `summarize`

One reason for introducing the `|>` now is because of how instrumental it is for chaining together two functions for generating summary statistics by group:

- `group_by`: a function that takes a data set and groups it by a variable/column
- `summarize`: uses the grouped data set from `group_by`, and lets you define summary statistics columns for that group

Let's say we want to calculate the mean and standard deviation of self-reported tolerance for cold, comparing Lewis & Clark to University of Arizona students. We sort of did this earlier, but let's try it again using `group_by` / `summarize`:

```

coldSummary<-teamAntarcticaData |>
  group_by(school) |>
  summarize(avgCold=mean(cold), sdCold=sd(cold))

coldSummary

```

A tibble: 2 × 3

school	avgCold	sdCold
--------	---------	--------

	<chr>	<dbl>	<dbl>
1	Lewis & Clark College	3.43	0.950
2	University of Arizona	3.29	1.08

Let's break down what's going on here:

- First declare our variable (`coldSummary`)
- Initially assign it to the `teamAntarcticaData` data set
- "Pipe" that data to `group_by` , where we choose to group the data by the school column
- Then "pipe" that to `summarize` , where we define two new columns:
 - `avgCold` , set equal to `mean(cold)`
 - `sdCold` , set equal to `sd(cold)`

When printing out `coldSummary` , we see it's a new data set with just summary statistics for cold tolerance, grouped by the school.

Try using the `group_by / summarize` technique by finding the mean and standard deviation of self-reported cooking skill, comparing Lewis & Clark to University of Arizona students. Print to the screen.

```
cookingSummary<- teamAntarcticaData |>
  group_by(school) |>
  summarize(meanCooking=mean(cooking, na.rm=TRUE),
            sdCooking=sd(cooking, na.rm=TRUE))

cookingSummary
```

```
# A tibble: 2 × 3
  school          meanCooking sdCooking
  <chr>          <dbl>      <dbl>
1 Lewis & Clark College      3.13      1.12
2 University of Arizona      3.56      1.09
```

We can also use this technique to calculate percentage. Let's say we want to know the different percentage of responses to the swimming question. We can calculate this by first defining the total number of rows (using `nrow` , below), and use it with `n()` in `summarize`

```
# first calculate total rows, to be used as denominator for percentage
totalRows<-nrow(teamAntarcticaData)

# n() generates the count of responses per/group
swimmingPercentage<-teamAntarcticaData |>
  group_by(swim) |>
  summarise(percent=n()/totalRows*100)
```

```
swimmingPercentage
```

```
# A tibble: 3 × 2
```

	swim	percent
	<chr>	<dbl>
1	I can dog paddle	8
2	No	1.33
3	Yes	90.7

Below, calculate the percentage of responses for each of the different parka colors:

```
parkaPercentage<-teamAntarcticaData |>
  group_by(parkaColor) |>
  summarise(percent=n()/totalRows*100)

parkaPercentage
```

```
# A tibble: 15 × 2
```

	parkaColor	percent
	<chr>	<dbl>
1	Baby Pink	1.33
2	Black	24
3	Blue	29.3
4	Gold	1.33
5	Green	5.33
6	Lavender/purple	1.33
7	Orange	13.3
8	Pink	1.33
9	Pink, if possible	1.33
10	Purple	1.33
11	White	14.7
12	green	1.33
13	hot pink	1.33
14	purple	1.33
15	<NA>	1.33

Next: [homework-1.3.qmd](#)