Week 4: Extracting Vectors

Ellen Bledsoe

2024-02-07

Vectors and Data Frames

We've learned about two general ways to store data: vectors and data frames.

Vectors store a single set of values with the same type. Data frames store multiple sets of values, one in each column, that can have different data types.

As I've mentioned previously, these two ways of storing data are related to one another. A data frame is a bunch of equal length vectors that are grouped together.

Because of this, we can extract vectors from data frames, and we can also make data frames from vectors.

Extracting Vectors from Data Frames

There are several ways to extract a vector from a data frame.

Let's look at how this works with the `species` data frame.

We'll start by loading the species table into R if it isn't there already.

```r
library(readr)
species <- read_csv("species.csv")
```

```
## Rows: 54 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): species_id, genus, species, taxa
##
## 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.
```

Square Brackets

One way to pull out a vector is by using square brackets. Remember that `[]` basically means "give me a piece of something."

Let's get the species_id column. The column name has to be in quotes this time because we aren't using `dplyr`.

```r
species["species_id"]
```

```
## # A tibble: 54 x 1
##    species_id
##    <chr>
##  1 AB
##  2 AH
##  3 AS
##  4 BA
##  5 CB
##  6 CM
##  7 CQ
##  8 CS
##  9 CT
## 10 CU
## # i 44 more rows
```

Well, this *almost* returned a vector. What it actually returns a one column data frame, not a vector. Often this isn't a big deal, but sometimes functions will only accept vectors, not data frames.

To extract a single column as a vector, we use two sets of square brackets. We can think of the second set of `[]` as getting the single vector from inside the one column data frame

```
species[["species_id"]]
```

```
##  [1] "AB" "AH" "AS" "BA" "CB" "CM" "CQ" "CS" "CT" "CU" "CV" "DM" "DO" "DS" "DX"
## [16] "EO" "GS" "NL" "NX" "OL" "OT" "OX" "PB" "PC" "PE" "PF" "PG" "PH" "PI" "PL"
## [31] "PM" "PP" "PU" "PX" "RF" "RM" "RO" "RX" "SA" "SB" "SC" "SF" "SH" "SO" "SS"
## [46] "ST" "SU" "SX" "UL" "UP" "UR" "US" "ZL" "ZM"
```

The `$` Operator

Another common approach to extracting a column into a vector is to use `$`.

The `$` in R is short hand for `[[]]` in cases where the piece we want to get has a name.

As usual, we start with the object from which we want to pull a part; in this case, it would be the `surveys` data frame. We follow that with the `$`, which will then list all the names of our columns. We can then type the name of the column we want to choose (without quotes, just to make things extra confusing).

```
species$species_id
```

```
##  [1] "AB" "AH" "AS" "BA" "CB" "CM" "CQ" "CS" "CT" "CU" "CV" "DM" "DO" "DS" "DX"
## [16] "EO" "GS" "NL" "NX" "OL" "OT" "OX" "PB" "PC" "PE" "PF" "PG" "PH" "PI" "PL"
## [31] "PM" "PP" "PU" "PX" "RF" "RM" "RO" "RX" "SA" "SB" "SC" "SF" "SH" "SO" "SS"
## [46] "ST" "SU" "SX" "UL" "UP" "UR" "US" "ZL" "ZM"
```

If you click on the little arrow next to the `surveys` data frame in the environment, you will notice that all of the column names are preceded by `$`, really hammering home the fact that each column is really a vector.

Combining Vectors into Data Frames

We can also combine vectors to make a data frame.

We can make a data frame using the `data.frame` function. It takes one argument for each column in the data frame. Like in `mutate()` and `summarize()`, the argument includes the name of the column we want in the data frame, an equal sign, and the name of the vector whose values we want in that column.

```r
states <- c("AZ", "AZ", "NM", "CA")
count <- c(9, 16, 3, 10)
area <- c(3, 5, 1.9, 2.7)
count_data <- data.frame(states = states, counts = count, regional_area = area)
```

We can also add columns to the data frame that only include a single value without first creating a vector. We do this by providing a name for the new column, an equals sign, and the value that we want to occur in every row of the column.

For example, if all of these data were collected in the same year, and we wanted to add that year as a column in our data frame, we could do it like this:

```r
count_data_year <- data.frame(year = 2022,
                              states = states,
                              counts = count,
                              regional_area = area)
```

In the first argument, `year =` sets the name of the column in the data frame and 2000 is that value that will occur on every row of that column. If we wanted to fill a column with character data, we would need to put the data in quotation marks.

Let's Practice!

Work on Questions 8 and 9 in the Assignment.