Data Structures III: Tibbles

UNIVERSITY OF
SAN FRANCISCO

Abbie M Popa

BSDS 100 - Intro to Data Science with R

Outline

- Tibbles
- Tibbles versus Data Frames

Tibbles with tibble

- Dataframes can be fickle, with their stringsAsFactors = FALSE requirements, etc.
- Tibbles - from the `tibble` package - **are data frames**, with some minor modifications which make operating with tibbles a more pleasant and worry-free experience than dealing with raw dataframes.
- Creating a tibble is the same as creating a dataframe, but instead we use the function tibble() instead of data.frame().
- Coercing a dataframe (or other data structure) is done with as_tibble() function

Tibbles with tibble

- Tibbles have some convenient advantages
- When importing (e.g., `csv`'s) or coercing data into a tibble()
    1. **will not** automatically convert strings to factors
    2. **will not** change the names of variables
    3. **will not** create row names
    4. **will not** do any partial matching when subsetting
    5. when subsetting a tibble, it is consistent in returning a tibble or a vector
- Subsetting tibbles can be done with either `$` or `[[`

Creating and Manipulating Dataframes

- Recall that a dataframe is a list, which means that typeof(myDataFrame) will output a list
- Instead use is.data.frame()
- An object can be coerced to a (base) dataframe using as.data.frame() or tibble::as_data_frame()

Subsetting Dataframes

Dataframes (both base and tibbles) can be subset like lists or like matrices

```
> (my_data_frame <- data.frame(x = 1:3, y = -4:-6, z = LETTERS[1:3]))
    x   y   z
1   1   -4  A
2   2   -5  B
3   3   -6  C


> my_data_frame[1:2]
    x   y
1   1   -4
2   2   -5
3   3   -6
> my_data_frame[1, 3]
[1] A
Levels: A B C
```

Subsetting Dataframes

- If you subset using a single vector, the result behaves as a list
- If you subset using two vectors, the results behaves as a matrix
- Try the following:
  1. If you didn't already, set up the dataframe: `my_data_frame <- data.frame(x = 1:3, y = -4:-6, z = LETTERS[1:3])`
  2. What happens if you call `my_data_frame[1]`?
  3. What happens if you call `my_data_frame[1, ]`?
  4. What happens if you call `my_data_frame["x"]`?
  5. What happens if you call `my_data_frame[ ,"x"]`?
  6. How can you return the first and third columns of the dataframe?
  7. What type of structure is returned when you index as a list to return one column?
  8. What type of structure is returned when you index as a matrix to return one column?

Attributes of Dataframes

- You can use colnames(df) to access the column names of dataframe df (the variables, generally)
- You can use row.names(df) to access the row names of dataframe df (the observations, generally)
    - Note, tibbles discourage special row names
- You can use dim() to find the number of columns and rows in a dataframe (be wary of length())
- str() is infinitely useful for understanding your dataframe

$ for Dataframes

$ is an operator that can access a variable within a dataframe

e.g., For the iris dataset, `iris$Sepal.Length` will return all sepal length values and `iris$Sepal.Length[1:3]` will return the first three sepal length values.

Partial Column Matching

- Another quirk of "base" dataframes: partial column matching
  - Try it, typing `iris$Spec` returns the full column `iris$Species`
- More confusing, if partial matching would return two columns a value of "NULL" is produced
  - Try it, typing `iris$Sepal` returns `NULL`
- Tibbles do NOT have partial matching
  - First load the tibble library with `library(tibble)`
  - Now save the iris dataset as a tibble `iris2 <- as_tibble(iris)`
  - Try the above partial matches with `iris2` what happens?

Column Names as Variables

- Sometimes you might want to save a column name as a variable (e.g., `sl <- "Sepal.Length"`)
- Can be useful for long or hard to type column names, or if you need to reference the column many places
- $ only works with the column's "real" name, so if you use a variable for a column name you will need to reference with [ ]
    - Try the above, set the variable `sl <- "Sepal.Length"`
    - What happens if you reference by `iris$sl`?
    - What about `iris[ , sl]`?

One More Way to Subset

- We have seen subsetting with [ ] and $, but there is also [[ ]]
- Recall, when [ ] is used to subset a list, it returns a list, whereas when [[ ]] is used to subset a list, it returns it's contents.
- Similarly, you can use [[ ]] to extract the contents of a particular column from a dataframe
- You can use column names `iris[["Species"]]` or ordinal locations `iris[[1]]` in the double square brackets
- Note: for both tibbles and dataframes [[ ]] subsetting always returns the contents of the column, not another dataframe
- Note: when using [[ ]] you can only select one (not multiple) columns

$ versus [[ ]]

- $ does partial matching in data frames (not tibbles)
- [[ ]] never does partial matching

Simplifying versus Preserving Subsetting

- **Simplifying** subsets returns the simplest possible data structure that can represent the output
- **Preserving** subsets keep the output structure the same as the input
- The drop option when subsetting is one of the most common sources of programming error
- By default, a data frame (not a tibble) will **simplify** when returning a single column, but **preserve** when returning a single row
- You can set ‘drop = F‘ to **preserve** even when returning a column

```
tester[1, ]
class(tester[1, ])
tester[ ,1]
class(tester[ ,1])
tester[ ,1, drop = F]
class(tester[ , 1, drop = F])
```

Column Names in Tibbles

- Data frames will not allow you to use special characters like spaces or punctuation in column names
- Tibbles let you name columns anything, though if they contain special characters you must surround them with back ticks

Tribbles

- Tibbles also allow you to input rows using the tribble() function
- This can be particularly useful when adding new observations to a tibble

Displaying Large Tibbles

- Tibbles also give you more control when viewing
- Note traditional data frames will print the whole data frame, regardless of size
- Tibbles will by default print 10 rows and however many columns fit on your screen
- But this can be adjusted

Summary: Tibbles versus Data Frames

- Data frames are one of the (the?) most popular and useful data structures in R
- Tibbles aim to fix some of the finicky features of data frames, speficially
  - Tibbles will default to keeping strings as strings (data frames convert strings to factors by default)
  - Tibbles preserve all subsets made with [ ] by default (data frames simplify single column subsets by default)
    - Note, both will simplify when subsetting with $
  - Tibbles discourage the use of row names
  - Tibbles have a pretty output by default

Activity: Comparing Data Frames to Tibbles

Go to https://github.com/abbiepopa/bsds100 and select Tibbles versus Data Frames under **Class Code**