DATA/STAT 234 Basic Syntax

The purpose of this series of handouts is to practice writing the basic syntax of some of the functions we will use “by hand.” Writing this syntax by hand can help with overall understanding of the code and gives us a resource to glance at when we move to the computer coding section of the material.

`ggplot2`

Suppose we have the following toy data set, named `df`. The first two columns are numeric while the third column is categorical.

x1	x2	cat1
1	3	Yes
7	20	Yes
4	2	No
...	...	...

Basic Plot Structure.

Inside vs. Outside `aes()` Aesthetics

Global vs. Local Aesthetics

`dplyr`

Suppose we have the following toy data set, named `df`. The first two columns are numeric while the third column is categorical.

x1	x2	cat1
1	3	Yes
7	NA	Yes
4	2	No

Choose Rows to Keep with `filter()` (based on a condition)

Choose Rows to Keep with `slice()` (based on the row index)

Choose Columns to Keep with `select()`

x1	x2	cat1
1	3	Yes
7	NA	Yes
4	2	No

Order/Sort Your Data Set with `arrange()`

Create New Variables with `mutate()` (Perhaps with `case_when()` or `if_else()`)

Obtain Numerical Summaries with `summarise()`

Obtain Numerical Summaries by Group with `group_by()` and `summarise()`

Quarto Options

option	description of the option	default	other choices
`echo`			
`eval`			
`warning`			
`output`			

Figure Options

option	description of the option	default	other choices
`fig-height`			
`fig-width`			

`tidyr`

Suppose we have the following toy data set, named `df`, on tennis players. The first column contains the player's max serve speed and handedness, the second contains their rank in the year 1980, and the third contains their rank in the year 1981.

xvar	Rank1980	Rank1981
100-RH	2	6
110-LH	30	19
99-RH	31	30

Split One Column into Two with `separate()`

`pivot_longer()` to Gather Multiple Columns

`unite()` and `pivot_wider()`

`R` Basics

Classes

Class Type	Name	description of the class type	other notes
`<chr>`			
`<fct>`			
`<date>`			
`<datetime>`			
`<dbl>`			
`<int>`			
`<lgl>`			

`forcats`

Suppose we have the following toy data set, named `df`, on the categorical variable `cat1` and the quantitative variable `x`.

cat	x
A	2
B	-1
C	14

- Change the names of factor levels with `fct_recode()`:

- Collapse many levels of a factor into fewer levels with `fct_collapse()`

- Order levels of a factor by a quantitative variable with `fct_reorder()`:

- Manually order the levels of a factor with `fct_relevel()`:

Joining with `dplyr`

Suppose we have the following two data sets. The first, `df1` has the variables `id_numb` and `xvar`. The second, `df2` has the variables `id` and `yvar`. `id_numb` and `id` serve as identification variables, possibly with duplicates, where observations from the first data set with `id_numb = 1` correspond to observations in the second data set with `id = 1`.

id_numb	xvar
1	16
1	-1
2	11
4	13

id	yvar
1	-1
2	-4
2	0
3	-9

Mutating Joins

- `left_join()`

- `right_join()`

- `inner_join()`

id_numb	xvar
1	16
1	-1
2	11
4	13

id	yvar
1	-1
2	-4
2	0
3	-9

- `full_join()`

Filtering Joins

- `semi_join()`

- `anti_join()`

`lubridate`

Suppose we have the following data set, named `df`, which has various date formats, as `<chr>` variables.

date1	date2	date3
January 14, 1992	1992-January-14	01/14/1992
October 19, 1991	1991-October-19	10/19/1991

`dmy()`, `dym()`, `mdy()`, `myd()`, `ydm()`, `ymd()` to Convert to a `<date>`

Now suppose we have a data set, called `df2`, that has a `<date>` variable:

date1
1992-01-14
1991-10-19

`year()`, `month()`, `mday()` `yday()`, and `wday()` to Pull Useful Variables from a `<date>`

`stringr`

Suppose we have the following data set with a variable of strings (of recent Wordle solutions) called `df1`. Our example is small, but you might think of each variable as containing lyrics to a song or the text of a book or essay.

strings_var1
Stale
dream
photo
Aloud
Inept

The following are just a few functions from the `stringr` package to manipulate strings.

`str_to_lower()` / `str_to_upper()` / `str_to_title()`

Most `stringr` functions will require you to specify a `pattern` in the string, called a **regex** (regular expression) that the `stringr` function will extract, detect, replace, etc.

`str_detect()` to Detect whether a String has a Certain Pattern

Using ^ and `$` in `str_detect()`

`str_replace_all()` to replace a regex pattern with a new string

`str_remove()` to Remove a Pattern from a String

`str_sub()` to grab certain parts of a string.

There are a lot of different patterns to regexes. For example, we can also use `.` to match any character, `\d` to match any digit, `\s` to match any whitespace, `[abc]` to match a, b, or c, and `[^abc]` to match anything except a, b, or c, etc.