Intro to R

Subsetting Data in R

Recap

- R functions as a calculator
- Use `c()` to **combine** vectors
- Use `<-` to save (assign) values to objects
- if you don't use `<-` to reassign objects that you want to modify, they will stay the same
- `length()`, `class()`, and `str()` tell you information about an object
- `head()` and `tail()` can also help you inspect an object
- `readr`has helpful functions like `read_csv()` that can help you import data into R

Cheatsheet

Overview

In this module, we will show you how to:

1. Look at your data in different ways
2. Create a data frame and a tibble
3. Create new variables/make rownames a column
4. Rename columns of a data frame
5. Subset rows of a data frame
6. Subset columns of a data frame
7. Add/remove new columns to a data frame
8. Order the columns of a data frame
9. Order the rows of a data frame

Setup

We will largely focus on the `dplyr` package which is part of the `tidyverse`.

Some resources on how to use `dplyr`:

- https://dplyr.tidyverse.org/
- https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html
- https://www.opencasestudies.org/

Why dplyr?

hadley commented on May 26, 2016 — Member

The d is for dataframes, the plyr is to evoke pliers. Pronounce however you like.

👍 1

The `dplyr` package is one of the most helpful packages for altering your data to get it into a form that is useful for creating visualizations, summarizing, or more deeply analyzing.

So you can imagine using pliers on your data.

Loading in dplyr and tidyverse

See this website for a list of the packages included in the `tidyverse`: https://www.tidyverse.org/packages/

```
library(tidyverse) # loads dplyr and other packages!
```

Getting data to work with

Here we use one of the datasets that comes with base R called `mtcars`. We will now create a toy data frame named `df` using this data. This way we can alter `df` without worrying about changing `mtcars`.

```
df <- mtcars # df is a copy of mtcars
head(df) # changing df does **not** change mtcars!
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Checking the data `dim()`

The `dim()`, `nrow()`, and `ncol()` functions are good options to check the dimensions of your data before moving forward.

```
dim(df) # rows, columns
```

```
[1] 32 11
```

```
nrow(df) # number of rows
```

```
[1] 32
```

```
ncol(df) # number of columns
```

```
[1] 11
```

Checking the data: `glimpse()`

In addition to `head()` and `tail()`, the `glimpse()`function of the `dplyr` package is another great function to look at your data.

```
glimpse(df)
```

```
Rows: 32
Columns: 11
$ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 17.8,
$ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4, 8,
$ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8, 16
$ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180, 180
$ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3.92,
$ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150, 3.
$ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90, 18
$ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
$ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
$ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, 4, 3, 3,
$ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, 1, 1, 2,
```

Checking your data: `slice_sample()`

What if you want to see the middle of your data? You can use the `slice_sample()` function of the `dplyr` package to see a random set of rows. You can specify the number of rows with the `n` argument or use a proportion with the `prop` argument.

```
slice_sample(df, n = 3)
```

```
                  mpg cyl  disp  hp drat   wt  qsec vs am gear carb
Maserati Bora    15.0   8 301.0 335 3.54 3.57 14.60  0  1    5    8
Hornet Sportabout 18.7   8 360.0 175 3.15 3.44 17.02  0  0    3    2
Merc 230         22.8   4 140.8  95 3.92 3.15 22.90  1  0    4    2
```

```
slice_sample(df, prop = .2)
```

```
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Merc 450SLC    15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 280       19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
```

skimr package

```
library(skimr)
skim(df)
```

```
> skim(df)
── Data Summary ────────────────────────
                           Values
Name                       df
Number of rows             32
Number of columns          11
_______________________
Column type frequency:
  numeric                  11
_______________________
Group variables            None

── Variable type: numeric ──────────────────────────────────────────────────────────────────────────────
   skim_variable n_missing complete_rate    mean      sd    p0    p25    p50    p75   p100 hist
 1 mpg                   0             1  20.1     6.03  10.4   15.4   19.2   22.8   33.9 ▃▇▅▁▂
 2 cyl                   0             1   6.19    1.79   4      4      6      8      8   ▆▁▃▁▇
 3 disp                  0             1 231.    124.    71.1  121.   196.   326    472   ▇▃▃▃▂
 4 hp                    0             1 147.     68.6   52     96.5  123    180    335   ▇▇▆▃▁
 5 drat                  0             1   3.60    0.535  2.76   3.08   3.70   3.92   4.93 ▆▃▇▅▁
 6 wt                    0             1   3.22    0.978  1.51   2.58   3.32   3.61   5.42 ▃▃▇▁▂
 7 qsec                  0             1  17.8     1.79  14.5   16.9   17.7   18.9   22.9  ▃▇▇▂▁
 8 vs                    0             1   0.438   0.504  0      0      0      1      1   ▇▁▁▁▆
 9 am                    0             1   0.406   0.499  0      0      0      1      1   ▇▁▁▁▆
10 gear                  0             1   3.69    0.738  3      3      4      4      5   ▇▁▆▁▂
11 carb                  0             1   2.81    1.62   1      2      2      4      8   ▇▆▁▂▁
```

Making data frames(base R) and tibbles (tidyverse)

Creating data frames using base R data frame function

```
data.frame(df)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2

Keep in mind…

Need to assign the output of the function to keep the result!

```
df_updated <- data.frame(df)
# this would overwrite the existing df object
df<-data.frame(df)
```

Or create a data frame when reading in the file

Or directly when reading in a csv with the `read.csv()` function (also base R)

```
# function comes from base R - no package loading required
df_example_readr <- read.csv(file = "documents/data_analysis/data_file.csv")
```

tibble

We can create a **fancier** version of the previous data frame which can be really helpful.

Creating a `tibble`

If we would like to create a `tibble` ("fancy" data frame), we can using the `tibble()` function.

```
tbl <- tibble(df)
tbl
```

```
# A tibble: 32 × 11
     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21        6  160    110  3.9   2.62  16.5     0     1     4     4
 2  21        6  160    110  3.9   2.88  17.0     0     1     4     4
 3  22.8      4  108     93  3.85  2.32  18.6     1     1     4     1
 4  21.4      6  258    110  3.08  3.22  19.4     1     0     3     1
 5  18.7      8  360    175  3.15  3.44  17.0     0     0     3     2
 6  18.1      6  225    105  2.76  3.46  20.2     1     0     3     1
 7  14.3      8  360    245  3.21  3.57  15.8     0     0     3     4
 8  24.4      4  147.    62  3.69  3.19  20       1     0     4     2
 9  22.8      4  141.    95  3.92  3.15  22.9     1     0     4     2
10  19.2      6  168.   123  3.92  3.44  18.3     1     0     4     4
# … with 22 more rows
```

Note don't necessarily need to use `head()`- tibbles conveniently print a portion of the data.

tibbles form read_csv()

Alternatively we can read data files using the `tidyverse` with the `read_csv()` function of the `readr` package from the `tidyverse` to make a tibble.

```
df_example_readr <- read_csv(file = "documents/data_analysis/data_file.csv")
```

You may start to notice how the tidyverse package work well together!

Summary of tibbles and data frames

Base R:
Using `read.csv()` and `data.frame()` you can make data frames

Tidyverse (fancier version):
Using `read_csv()` and `tibble()` you can make tibbles

We generally recommend using tibbles, but you can do a lot with data frames too.

Data frames vs tibbles

In the "tidy" data format, rownames are removed. For example, `df` has each car name as a row name. Here we use the `head()` function to see the first 2 rows of each using the `n` argument. In this case we would want to make the rownames a new column first before making into a tibble.

```
head(df, n = 2)
```

```
               mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4       21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag   21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```
head(tibble(df), n = 2)
```

```
# A tibble: 2 × 11
    mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1    21     6   160   110   3.9  2.62  16.5     0     1     4     4
2    21     6   160   110   3.9  2.88  17.0     0     1     4     4
```

rownames_to_column function

If you run into losing a variable contained in your row names, you can also use `rownames_to_column` (of `tibble` package) to add it before turning it into a `tibble` to keep them:

```
# general format! not code!
{data you are creating or changing} <- # reassign if you want to keep changes
      rownames_to_column({data you are using},
                         {Name of column you are making from rownames})
```

```
head(rownames_to_column(df, "car"),  n = 2)
```

```
            car mpg cyl disp  hp drat    wt  qsec vs am gear carb
1     Mazda RX4  21   6  160 110  3.9 2.620 16.46  0  1    4    4
2 Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```
head(tibble(rownames_to_column(df, "car")),  n = 2)
```

```
# A tibble: 2 × 12
  car             mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  ca
  <chr>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <db
1 Mazda RX4        21     6   160   110   3.9  2.62  16.5     0     1     4
2 Mazda RX4 W…     21     6   160   110   3.9  2.88  17.0     0     1     4
```

Renaming Columns

Renaming Columns of a data frame or tibble

To rename columns in `dplyr`, you can use the `rename` function.

For example, let's rename mpg to MPG. Notice the new name is listed **first**!

```
# general format! not code!
{data you are creating or changing} <- rename({data you are using},
                                              {New Name} = {Old name})
```

```
df <- rename(df, MPG = mpg)
head(df)
```

```
                   MPG cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Take Care with Column Names

When you can, avoid spaces, special punctuation, or numbers in column names, as these require quotes to refer to them.

See https://jhudatascience.org/intro_to_r/quotes_vs_backticks.html for more guidance.

```
df <- rename(df, MPG! = MPG) # this will cause an error
```

```
df_rename <-rename(df, `MPG!` = MPG) # this will work
head(df_rename, 2)
```

```
               MPG! cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4        21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag    21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

You will need to refer to a column like this with most functions.

Take Care with Column Names and Character Strings

These are the conventions, most options will work for most functions.

Backticks are typically for nonstandard variable names:

- those with spaces `col 1`
- those with punctuation `col.1`
- those that are just numbers `1`
- those that start with numbers `1st col`

Single or double quotes are typically used for character strings (data values that has characters):

- `"words"`
- `"phrases with spaces"`
- `'words'`
- `'phrases with spaces'`

Be careful about copy pasting code!

Curly quotes will not work!

```
df_rename <-rename(df, ‘MPG!’ = MPG) # this will cause an error!
```

```
df_rename <-rename(df, `MPG!` = MPG) # this will work!
```

Also true for double quotes

```
df_rename <-rename(df, “MPG!” = MPG) # this will cause an error!
```

```
df_rename <-rename(df, "MPG!" = MPG) # this will work!
```

Renaming All Columns of a data frame: dplyr

To rename all columns you use the `rename_with()`. In this case we will use `toupper()` to make all letters upper case. Could also use `tolower()` function.

```
df_upper <- rename_with(df, toupper)
head(df_upper, 3)
```

```
                MPG CYL DISP  HP DRAT    WT  QSEC VS AM GEAR CARB
Mazda RX4      21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

```
df <- rename_with(df, tolower)
head(df, 3)
```

```
                mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710     22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

Summary

- data frames are simpler version of a data table
- tibbles are fancier `tidyverse` version
- data frames are made with `data.frames()` and `read.csv()`
- tibbles are made with `tibble()` and `read_csv()` from `readr`
- If your original data has rownames, you need to use `rownames_to_column` before converting to tibble
- the `rename()` function of `dplyr` can help you rename columns
- avoid using punctuation (except underscores), spaces, and numbers (to start or alone) in column names
- If you must use backticks around those column names
- quotes can be used for character values
- avoid copy and pasting code from other sources - quotation marks will change!

Lab Part 1

Class Website
Lab

Subsetting Columns

Subset columns of a data frame - `tidyverse` way:

To grab (or "pull" out) the `carb` column the `tidyverse` way we can use the `pull` function:

```
pull(df, carb)
```

```
 [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8 2
```

Subset columns of a data frame: dplyr

The `select` command from `dplyr` allows you to subset (still a `tibble`!)

```
select(df, mpg)
```

```
                   mpg
Mazda RX4         21.0
Mazda RX4 Wag     21.0
Datsun 710        22.8
Hornet 4 Drive    21.4
Hornet Sportabout 18.7
Valiant           18.1
Duster 360        14.3
Merc 240D         24.4
Merc 230          22.8
Merc 280          19.2
Merc 280C         17.8
Merc 450SE        16.4
Merc 450SL        17.3
Merc 450SLC       15.2
Cadillac Fleetwood  10.4
Lincoln Continental 10.4
Chrysler Imperial   14.7
Fiat 128          32.4
Honda Civic       30.4
Toyota Corolla    33.9
Toyota Corona     21.5
Dodge Challenger  15.5
AMC Javelin       15.2
Camaro Z28        13.3
Pontiac Firebird  19.2
```

Subset columns of a data frame: dplyr

Note that if you want the values (not a `tibble`), use `pull` - as it pulls out the data:

```
pull(df, mpg)
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.
[31] 15.0 21.4
```

```
# pull with select works too!

pull(select(df, mpg))
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.
[31] 15.0 21.4
```

Select columns of a data frame: dplyr

The `select` command from `dplyr` allows you to subset columns matching patterns:

```
head(df, 2)
```

```
              mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

```
select(df, starts_with("c"))
```

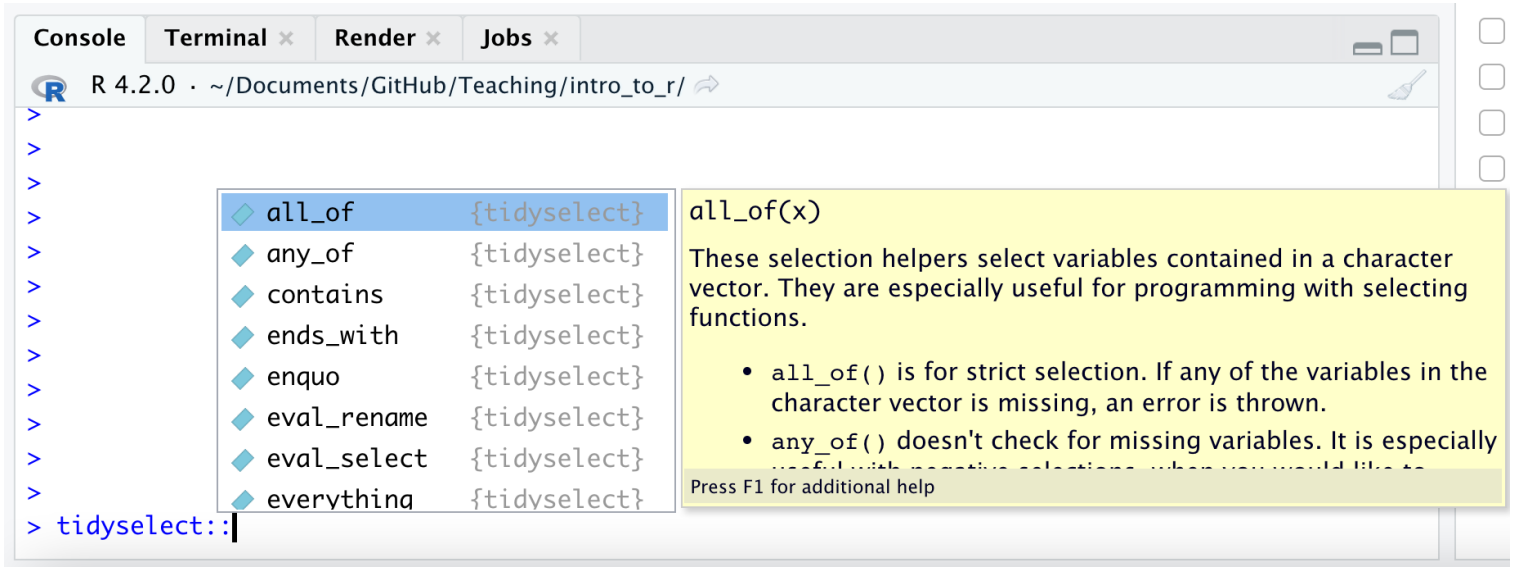
```
                  cyl carb
Mazda RX4           6    4
Mazda RX4 Wag       6    4
Datsun 710          4    1
Hornet 4 Drive      6    1
Hornet Sportabout   8    2
Valiant             6    1
Duster 360          8    4
Merc 240D           4    2
Merc 230            4    2
Merc 280            6    4
Merc 280C           6    4
Merc 450SE          8    3
Merc 450SL          8    3
Merc 450SLC         8    3
Cadillac Fleetwood  8    4
Lincoln Continental 8    4
Chrysler Imperial   8    4
```

See the Select "helpers"

Here are a few:

```
last_col()
starts_with()
ends_with()
contains() # like searching
```

Type `tidyselect::` in the **console** and see what RStudio suggests:



Summary

- `pull()` to get values out of a data frame/tibble
- `select()` is the `tidyverse` way to get a tibble with only certain columns
- you can `select()` based on patterns in the column names

Lab Part 2

Class Website
Lab

Subsetting Rows

Subset rows of a data frame: dplyr

The command in `dplyr` for subsetting rows is `filter`.

```
filter(df, mpg > 20)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

Subset rows of a data frame: dplyr

You can have multiple logical conditions using the following:

- `==` : equals to
- `!=`: not equal to (`!` : not/negation)
- `>` / `<`: greater than / less than
- `>=` or `<=`: greater than or equal to / less than or equal to
- `&` : AND
- `|` : OR

Subset rows of a data frame: dplyr

The `%in%` operator can be used find values from a pre-made list (using `c()`):

```
filter(df, mpg %in% c(20,21,22))
```

```
               mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4       21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag   21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

Subset rows of a data frame: dplyr

You can filter by two conditions using & or commas (must meet both conditions):

```
filter(df, mpg > 20 & cyl == 4)
```

```
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

```
filter(df, mpg > 20, cyl == 4)
```

```
                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
Toyota Corona  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

Subset rows of a data frame: dplyr

If you want OR statements (meaning the data can meet either condition does not need to meet both), you need to use the pipe | between conditions:

```
filter(df, mpg > 20 | cyl == 4)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

`which()` function

Instead of removing rows like filter, `which()` simply shows where the values occur if they pass a specific condition. We will see that this can be helpful later when we want to select and filter in more complicated ways.

```
which(select(df,carb) == 4)
```

```
 [1]  1  2  7 10 11 15 16 17 24 29
```

```
select(df, carb) == 4
```

```
                   carb
Mazda RX4          TRUE
Mazda RX4 Wag      TRUE
Datsun 710        FALSE
Hornet 4 Drive    FALSE
Hornet Sportabout FALSE
Valiant           FALSE
Duster 360         TRUE
Merc 240D         FALSE
Merc 230          FALSE
Merc 280           TRUE
Merc 280C          TRUE
Merc 450SE        FALSE
Merc 450SL        FALSE
Merc 450SLC       FALSE
Cadillac Fleetwood  TRUE
Lincoln Continental TRUE
Chrysler Imperial   TRUE
Fiat 128          FALSE
```

`distinct()` function

To filter for distinct values from a variable, multiple variables, or an entire tibble you can use the `distinct()` function from the `dplyr` package.

```
distinct(df, cyl)
```

```
                  cyl
Mazda RX4           6
Datsun 710          4
Hornet Sportabout   8
```

```
distinct(df, cyl, gear)
```

```
                  cyl gear
Mazda RX4           6    4
Datsun 710          4    4
Hornet 4 Drive      6    3
Hornet Sportabout   8    3
Toyota Corona       4    3
Porsche 914-2       4    5
Ford Pantera L      8    5
Ferrari Dino        6    5
```

Summary

- `filter()` can be used to filter out rows based on logical conditions
- `==` is the same as equivalent to
- `&` means both conditions must be met to remain after `filter()`
- `|` means either conditions needs to be met to remain after `filter()`
- `which()` shows you where values meet a condition
- `distinct()` helps you filter for unique values

Lab Part 3

Class Website Lab

Combining `filter` and `select`

You can combine `filter` and `select` to subset the rows and columns, respectively, of a data frame:

```
select(filter(df, mpg > 20 & cyl == 4), cyl, hp)
```

```
               cyl  hp
Datsun 710       4  93
Merc 240D        4  62
Merc 230         4  95
Fiat 128         4  66
Honda Civic      4  52
Toyota Corolla   4  65
Toyota Corona    4  97
Fiat X1-9        4  66
Porsche 914-2    4  91
Lotus Europa     4 113
Volvo 142E       4 109
```

In R, the common way to perform multiple operations is to wrap functions around each other in a "nested" way such as above.

Assigning Temporary Objects

One can also create temporary objects and reassign them:

```
df2 <- filter(df, mpg > 20 & cyl == 4)
df2 <- select(df2, cyl, hp)

head(df2,4)
```

```
           cyl hp
Datsun 710   4 93
Merc 240D    4 62
Merc 230     4 95
Fiat 128     4 66
```

Using the `pipe` (comes with `dplyr`):

The pipe `%>%` makes things such as this much more readable. It reads left side "pipes" into right side. RStudio `CMD/Ctrl + Shift + M` shortcut. Pipe `df` into `filter`, then pipe that into `select`:

```
df %>% filter(mpg > 20 & cyl == 4) %>% select(cyl, hp)
```

```
               cyl  hp
Datsun 710       4  93
Merc 240D        4  62
Merc 230         4  95
Fiat 128         4  66
Honda Civic      4  52
Toyota Corolla   4  65
Toyota Corona    4  97
Fiat X1-9        4  66
Porsche 914-2    4  91
Lotus Europa     4 113
Volvo 142E       4 109
```

Adding/Removing Columns

Adding columns to a data frame: dplyr (**tidyverse** way)

The `mutate` function in `dplyr` allows you to add or modify columns of a data frame.

```
# General format - Not the code!
{data object to update} <- mutate({data to use},
                                  {new variable name} = {new variable source})
```

```
df <- mutate(df, newcol = wt/2.2)
```

Removing columns of a data frame: dplyr

The `NULL` method is still very common.

The `select` function can remove a column with minus (`-`):

```
select(df, - newcol)
```

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant          18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Or, you can simply select the columns you want to keep, ignoring the ones you want to remove.

Removing columns in a data frame: dplyr

You can use `c()` to list the columns to remove.

Remove `newcol` and `drat`:

```
select(df, -c("newcol", "drat"))
```

	mpg	cyl	disp	hp	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	3.520	16.87	0	0	3	2

Ordering columns

Ordering the columns of a data frame: dplyr

The `select` function can reorder columns.

```
head(df)
select(df, cyl, mpg, wt, car) %>%
head()
```

Ordering the columns of a data frame: dplyr

The `select` function can reorder columns. Put `newcol` first, then select the rest of columns:

```
select(df, newcol, everything())
```

```
                  newcol  mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4       1.190909 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag   1.306818 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710      1.054545 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive  1.461364 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 1.563636 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant         1.572727 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

Ordering the columns of a data frame: dplyr

Put `newcol` at the end ("remove, everything, then add back in"):

```
select(df, -newcol, everything(), newcol)
```

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb   newcol
Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 1.190909
Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.306818
Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.054545
Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 1.461364
Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0    3    2 1.563636
Valiant          18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 1.572727
```

Ordering the column names of a data frame: alphabetically

```
order(colnames(df))
```

```
 [1]  9 11  2  3  5 10  4  1 12  7  8  6
```

```
df %>% select(order(colnames(df)))
```

	am	carb	cyl	disp	drat	gear	hp	mpg	newcol	qsec	vs
Mazda RX4	1	4	6	160.0	3.90	4	110	21.0	1.1909091	16.46	0
Mazda RX4 Wag	1	4	6	160.0	3.90	4	110	21.0	1.3068182	17.02	0
Datsun 710	1	1	4	108.0	3.85	4	93	22.8	1.0545455	18.61	1
Hornet 4 Drive	0	1	6	258.0	3.08	3	110	21.4	1.4613636	19.44	1
Hornet Sportabout	0	2	8	360.0	3.15	3	175	18.7	1.5636364	17.02	0
Valiant	0	1	6	225.0	2.76	3	105	18.1	1.5727273	20.22	1
Duster 360	0	4	8	360.0	3.21	3	245	14.3	1.6227273	15.84	0
Merc 240D	0	2	4	146.7	3.69	4	62	24.4	1.4500000	20.00	1
Merc 230	0	2	4	140.8	3.92	4	95	22.8	1.4318182	22.90	1
Merc 280	0	4	6	167.6	3.92	4	123	19.2	1.5636364	18.30	1
Merc 280C	0	4	6	167.6	3.92	4	123	17.8	1.5636364	18.90	1
Merc 450SE	0	3	8	275.8	3.07	3	180	16.4	1.8500000	17.40	0
Merc 450SL	0	3	8	275.8	3.07	3	180	17.3	1.6954545	17.60	0
Merc 450SLC	0	3	8	275.8	3.07	3	180	15.2	1.7181818	18.00	0
Cadillac Fleetwood	0	4	8	472.0	2.93	3	205	10.4	2.3863636	17.98	0
Lincoln Continental	0	4	8	460.0	3.00	3	215	10.4	2.4654545	17.82	0
Chrysler Imperial	0	4	8	440.0	3.23	3	230	14.7	2.4295455	17.42	0
Fiat 128	1	1	4	78.7	4.08	4	66	32.4	1.0000000	19.47	1
Honda Civic	1	2	4	75.7	4.93	4	52	30.4	0.7340909	18.52	1
Toyota Corolla	1	1	4	71.1	4.22	4	65	33.9	0.8340909	19.90	1
Toyota Corona	0	1	4	120.1	3.70	3	97	21.5	1.1204545	20.01	1
Dodge Challenger	0	2	8	318.0	2.76	3	150	15.5	1.6000000	16.87	0
AMC Javelin	0	2	8	304.0	3.15	3	150	15.2	1.5613636	17.30	0

Ordering the columns of a data frame: dplyr

In addition to `select` we can also use the `relocate()` function of dplyr to rearrange the columns.

For example, let say we just wanted `wt` to be first.

```
head(df)
```

```
                  mpg cyl disp  hp drat    wt  qsec vs am gear carb   newcol
Mazda RX4        21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 1.190909
Mazda RX4 Wag    21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.306818
Datsun 710       22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.054545
Hornet 4 Drive   21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 1.461364
Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0    3    2 1.563636
Valiant          18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 1.572727
```

```
df_carb <- relocate(df, wt,
                    .before = mpg)

df_carb
```

```
                     wt  mpg cyl  disp  hp drat  qsec vs am gear carb
Mazda RX4         2.620 21.0   6 160.0 110 3.90 16.46  0  1    4    4
Mazda RX4 Wag     2.875 21.0   6 160.0 110 3.90 17.02  0  1    4    4
Datsun 710        2.320 22.8   4 108.0  93 3.85 18.61  1  1    4    1
Hornet 4 Drive    3.215 21.4   6 258.0 110 3.08 19.44  1  0    3    1
Hornet Sportabout 3.440 18.7   8 360.0 175 3.15 17.02  0  0    3    2
Valiant           3.460 18.1   6 225.0 105 2.76 20.22  1  0    3    1
Duster 360        3.570 14.3   8 360.0 245 3.21 15.84  0  0    3    4
Merc 240D         3.190 24.4   4 146.7  62 3.69 20.00  1  0    4    2
```

Ordering rows

Ordering the rows of a data frame: dplyr

The `arrange` function can reorder rows By default, `arrange` orders in increasing order:

```
arrange(df, mpg)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1

Ordering the rows of a data frame: dplyr

Use the `desc` to arrange the rows in descending order:

```
arrange(df, desc(mpg))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3

Ordering the rows of a data frame: dplyr

You can combine increasing and decreasing orderings:

```
arrange(df, mpg, desc(hp))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1

Summary

- `select()` and `filter()` can be combined together
- you can do sequential steps in a few ways:
  1. nesting them inside one another using parentheses `()`
  2. creating intermediate data objects in between
  3. using pipes `%>%`
- `select()` and `relocate()` can be used to reorder columns
- `arrange()` can be used to reorder rows
- can remove rows with `filter()`
- can remove a column in a few ways:
  1. using `select()` with negative sign in front of column name(s)
  2. not selecting it (without negative sign)

Summary cont…

- `mutate()` can be used to create new variables or modify them

```
# General format - Not the code!
{data object to update} <- mutate({data to use},
                                  {new variable name} = {new variable source})
```

```
df <- mutate(df, newcol = wt/2.2)
```

A note about base R:

The `$` operator is similar to `pull()`. This is the base R way to do this:

```
df$carb
```

```
 [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8 2
```

Although it is easier (for this one task), mixing and matching the `$` operator with tidyverse functions usually doesn't work. Therefore, we want to let you know about it in case you see it, but we suggest that you try working with the tidyverse way.

Adding new columns to a data frame: base R

You can add a new column (or modify an existing one) using the `$` operator instead of `mutate`.

Just want you to be aware of this as it is very common.

```
df$newcol <- df$wt/2.2
head(df,3)
```

```
                mpg cyl disp  hp drat    wt  qsec vs am gear carb   newcol
Mazda RX4      21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 1.190909
Mazda RX4 Wag  21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.306818
Datsun 710     22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.054545
```

Even though `$` is easier for creating new columns, `mutate` is really powerful, so it's worth getting used to.

Lab Part 4

Class Website

Lab

Image by Gerd Altmann from Pixabay

Extra Slides - base R subsetting

Subset columns of a data frame:

We can grab the `carb` column using the `$` operator. This is the base R way to do this:

```
df$carb
```

```
 [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8 2
```

Remove a column in base R

```
df$mpg <- NULL
```

Renaming Columns of a data frame: base R

We can use the `colnames` function to extract and/or directly reassign column names of `df`:

```
colnames(df) # just prints
```

```
 [1] "mpg"    "cyl"    "disp"   "hp"     "drat"   "wt"     "qsec"   "vs"
 [9] "am"     "gear"   "carb"   "newcol"
```

```
colnames(df)[1:3] <- c("MPG", "CYL", "DISP") # reassigns
head(df)
```

```
                   MPG CYL DISP  hp drat    wt  qsec vs am gear carb   newcol
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 1.190909
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.306818
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.054545
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 1.461364
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 1.563636
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 1.572727
```

```
colnames(df)[1:3] <- c("mpg", "cyl", "disp") #reset - just to keep consistent
```

Renaming Columns of a data frame: base R

We can assign the column names, change the ones we want, and then re-assign the column names:

```
cn <- colnames(df)
cn[ cn == "drat"] <-"DRAT"
colnames(df) <- cn
head(df)
```

```
                   mpg cyl disp  hp DRAT    wt  qsec vs am gear carb   newcol
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 1.190909
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.306818
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.054545
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 1.461364
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 1.563636
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 1.572727
```

```
colnames(df)[ colnames(df) == "DRAT"] <- "drat" #reset
```

Subset rows of a data frame with indices:

Let's select **rows** 1 and 3 from `df` using brackets:

```
df[ c(1, 3), ]
```

```
           mpg cyl disp  hp drat   wt  qsec vs am gear carb   newcol
Mazda RX4  21.0   6  160 110 3.90 2.62 16.46  0  1    4    4 1.190909
Datsun 710 22.8   4  108  93 3.85 2.32 18.61  1  1    4    1 1.054545
```

Subset columns of a data frame:

We can also subset a data frame using the bracket `[, ]` subsetting.

For data frames and matrices (2-dimensional objects), the brackets are `[rows, columns]` subsetting. We can grab the `x` column using the index of the column or the column name ("`carb`")

```
df[, 11]
```

```
 [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8 2
```

```
df[, "carb"]
```

```
 [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8 2
```

Another difference between `tbl` and data frame:

Mostly, `tbl` (tibbles) are the same as data frames, except they don't print all lines. When subsetting only one column using brackets, a data frame will return the values, but a `tbl` will return a `tbl`

```
df[, 1]
```

```
 [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2 10.
[16] 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4 15.8 19.
[31] 15.0 21.4
```

```
tbl[, 1]
```

```
# A tibble: 32 × 1
     mpg
   <dbl>
 1  21
 2  21
 3  22.8
 4  21.4
 5  18.7
 6  18.1
 7  14.3
 8  24.4
 9  22.8
10  19.2
# … with 22 more rows
```

```
tbl[, "mpg"]
```

Subset columns of a data frame:

We can select multiple columns using multiple column names:

```
df[, c("mpg", "cyl")]
```

```
                   mpg cyl
Mazda RX4         21.0   6
Mazda RX4 Wag     21.0   6
Datsun 710        22.8   4
Hornet 4 Drive    21.4   6
Hornet Sportabout 18.7   8
Valiant           18.1   6
Duster 360        14.3   8
Merc 240D         24.4   4
Merc 230          22.8   4
Merc 280          19.2   6
Merc 280C         17.8   6
Merc 450SE        16.4   8
Merc 450SL        17.3   8
Merc 450SLC       15.2   8
Cadillac Fleetwood  10.4   8
Lincoln Continental 10.4   8
Chrysler Imperial   14.7   8
Fiat 128          32.4   4
Honda Civic       30.4   4
Toyota Corolla    33.9   4
Toyota Corona     21.5   4
Dodge Challenger  15.5   8
AMC Javelin       15.2   8
Camaro Z28        13.3   8
Pontiac Firebird  19.2   8
```