Subsetting Data in R

Introduction to R for Public Health Researchers

Overview

We showed one way to read data into R using `read_csv` and `read.csv`. In this module, we will show you how to:

1. Select specific elements of an object by an index or logical condition
2. Renaming columns of a `data.frame`
3. Subset rows of a `data.frame`
4. Subset columns of a `data.frame`
5. Add/remove new columns to a `data.frame`
6. Order the columns of a `data.frame`
7. Order the rows of a `data.frame`

Setup

We will show you how to do each operation in base R then show you how to use the `dplyr` package to do the same operation (if applicable).

Many resources on how to use `dplyr` exist and are straightforward:

- https://cran.rstudio.com/web/packages/dplyr/vignettes/
- https://stat545-ubc.github.io/block009_dplyr-intro.html
- https://www.datacamp.com/courses/dplyr-data-manipulation-r-tutorial

The `dplyr` package also interfaces well with tibbles.

Loading in dplyr and tidyverse

```
library(tidyverse)
```

```
── Attaching packages ─────────────────────────────────────── tidyverse 1.2.1 ──

✔ ggplot2 3.1.0     ✔ readr   1.1.1
✔ tibble  1.4.2     ✔ purrr   0.2.5
✔ tidyr   0.8.2     ✔ stringr 1.3.1
✔ ggplot2 3.1.0     ✔ forcats 0.3.0

── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
```

Note, when loading `dplyr`, it says objects can be "masked"/conflicts. That means if you use a function defined in 2 places, it uses the one that is loaded in **last**.

Loading in dplyr and tidyverse

For example, if we print `filter`, then we see at the bottom `namespace:dplyr`, which means when you type `filter`, it will use the one from the `dplyr` package.

```
filter
```

```
function (.data, ...)
{
    UseMethod("filter")
}
<bytecode: 0x7f92b664f720>
<environment: namespace:dplyr>
```

Loading in dplyr and tidyverse

A `filter` function exists by default in the `stats` package, however. If you want to make sure you use that one, you use `PackageName::Function` with the colon-colon ("`::`") operator.

```
head(stats::filter,2)
```

```
1 function (x, filter, method = c("convolution", "recursive"),
2     sides = 2L, circular = FALSE, init = NULL)
```

This is important when loading many packages, and you may have some conflicts/masking.

Creating a `data.frame` to work with

Here we use one of the datasets that comes with `R` called `mtcars` create a toy data.frame named `df` using random data:

```
data(mtcars)
df = mtcars
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

Creating a `data.frame` to work with

If we would like to create a `tibble` ("fancy" data.frame), we can using `as.tbl`. In the "tidy" data format, all information of interest is a variable (not a name).

```
tbl = as.tbl(df)
tbl = rownames_to_column(tbl, var = "car")
```

Renaming Columns

Renaming Columns of a `data.frame`: dplyr

To rename columns in `dplyr`, you use the `rename` command

```
df = dplyr::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
```

```
df = rename(df, mpg = MPG) # reset - don't need :: b/c not masked
```

Renaming All Columns of a `data.frame`: dplyr

To rename all columns you use the `rename_all` command (with a function)

```
df_upper = dplyr::rename_all(df, toupper)
head(df_upper)
```

```
                   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
```

Lab Part 1

Website

Subsetting Columns

Subset columns of a `data.frame`:

We can grab the `carb` column using the `$` operator.

```
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

```
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
```

Select columns of a `data.frame`: dplyr

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

```
select(df, 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
```

See the Select "helpers"

Run the command:

```
??tidyselect::select_helpers
```

Here are a few:

```
one_of()
last_col()
ends_with()
contains() # like searching
matches() # Matches a regular expression - cover later
```

Lab Part 2

Website

Subsetting Rows

Subset rows of a `data.frame`: dplyr

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

```
filter(df, mpg > 20 | mpg < 14)
```

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
5  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
6  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
7  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
8  10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
9  32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
10 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
11 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
12 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
13 13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
14 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
15 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
16 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
17 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Note, no `$` or subsetting is necessary. R "knows" `mpg` refers to a column of `df`.

Subset rows of a `data.frame`: dplyr

You can have multiple logical conditions using the following:

- `&` : AND
- `|` : OR

By default, you can separate conditions by commas, and `filter` assumes these statements are joined by `&`:

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

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
2  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
3  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
4  32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
5  30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
6  33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
7  21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
8  27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
9  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
10 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
11 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

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

Subset rows of a `data.frame`: dplyr

If you want OR statements, you need to do the pipe `|` explicitly:

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

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
5  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
6  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
7  32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
8  30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
9  33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
10 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
11 27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
12 26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
13 30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
14 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
```

Lab Part 3

Website

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
1    4  93
2    4  62
3    4  95
4    4  66
5    4  52
6    4  65
7    4  97
8    4  66
9    4  91
10   4 113
11   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)
```

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

Recently, 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
1    4  93
2    4  62
3    4  95
4    4  66
5    4  52
6    4  65
7    4  97
8    4  66
9    4  91
10   4 113
11   4 109
```

Adding/Removing Columns

Adding new columns to a `data.frame`: base R

You can add a new column, called `newcol` to `df`, using the `$` operator:

```
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
```

Adding columns to a `data.frame`: dplyr

The `$` method is very common.

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

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

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb    newcol
1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4 1.1909091
2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4 1.3068182
3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1 1.0545455
4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1 1.4613636
5  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 1.5636364
6  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 1.5727273
7  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4 1.6227273
8  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2 1.4500000
9  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2 1.4318182
10 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4 1.5636364
11 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4 1.5636364
12 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3 1.8500000
13 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3 1.6954545
14 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3 1.7181818
15 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4 2.3863636
16 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4 2.4654545
17 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4 2.4295455
18 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1 1.0000000
```

Creating conditional variables

One frequently-used tool is creating variables with conditions.

A general function for creating new variables based on existing variables is the `ifelse()` function, which "returns a value with the same shape as test which is filled with elements selected from either yes or no depending on whether the element of test is `TRUE` or `FALSE`."

```
ifelse(test, yes, no)

# test: an object which can be coerced
    to logical mode.
# yes: return values for true elements of test.
# no: return values for false elements of test.
```

Adding columns to a `data.frame`: dplyr

Combined with `ifelse(condition, TRUE, FALSE)`, it can give you didffer

```
df = mutate(df,
            disp_cat = ifelse(
              disp <= 200,
              "Low",
              ifelse(disp <= 400,
                     "Medium",
                     "High")
            )
)
head(df$disp_cat)
```

```
[1] "Low"    "Low"    "Low"    "Medium" "Medium" "Medium"
```

Removing columns to a `data.frame`: base R

You can remove a column by assigning to `NULL`:

```
df$newcol = NULL
```

Removing columns to a `data.frame`: dplyr

The `NULL` method is still very common.

The `select` function can remove a column with a minus (`-`), much like removing rows:

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

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb disp_cat
1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4      Low
2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4      Low
3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1      Low
4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1   Medium
5  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2   Medium
6  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1   Medium
7  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4   Medium
8  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2      Low
9  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2      Low
10 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4      Low
11 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4      Low
12 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3   Medium
13 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3   Medium
14 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3   Medium
15 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4     High
16 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4     High
17 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4     High
18 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1      Low
```

Removing columns to a `data.frame`: dplyr

Remove `newcol` and `drat`

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

```
    mpg cyl  disp  hp    wt  qsec vs am gear carb disp_cat
1  21.0   6 160.0 110 2.620 16.46  0  1    4    4      Low
2  21.0   6 160.0 110 2.875 17.02  0  1    4    4      Low
3  22.8   4 108.0  93 2.320 18.61  1  1    4    1      Low
4  21.4   6 258.0 110 3.215 19.44  1  0    3    1   Medium
5  18.7   8 360.0 175 3.440 17.02  0  0    3    2   Medium
6  18.1   6 225.0 105 3.460 20.22  1  0    3    1   Medium
7  14.3   8 360.0 245 3.570 15.84  0  0    3    4   Medium
8  24.4   4 146.7  62 3.190 20.00  1  0    4    2      Low
9  22.8   4 140.8  95 3.150 22.90  1  0    4    2      Low
10 19.2   6 167.6 123 3.440 18.30  1  0    4    4      Low
11 17.8   6 167.6 123 3.440 18.90  1  0    4    4      Low
12 16.4   8 275.8 180 4.070 17.40  0  0    3    3   Medium
13 17.3   8 275.8 180 3.730 17.60  0  0    3    3   Medium
14 15.2   8 275.8 180 3.780 18.00  0  0    3    3   Medium
15 10.4   8 472.0 205 5.250 17.98  0  0    3    4     High
16 10.4   8 460.0 215 5.424 17.82  0  0    3    4     High
17 14.7   8 440.0 230 5.345 17.42  0  0    3    4     High
18 32.4   4  78.7  66 2.200 19.47  1  1    4    1      Low
19 30.4   4  75.7  52 1.615 18.52  1  1    4    2      Low
20 33.9   4  71.1  65 1.835 19.90  1  1    4    1      Low
21 21.5   4 120.1  97 2.465 20.01  1  0    3    1      Low
22 15.5   8 318.0 150 3.520 16.87  0  0    3    2   Medium
```

Ordering columns

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 disp_cat
1  1.1909091 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4       Low
2  1.3068182 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4       Low
3  1.0545455 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1       Low
4  1.4613636 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1    Medium
5  1.5636364 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2    Medium
6  1.5727273 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1    Medium
7  1.6227273 14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4    Medium
8  1.4500000 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2       Low
9  1.4318182 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2       Low
10 1.5636364 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4       Low
11 1.5636364 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4       Low
12 1.8500000 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3    Medium
13 1.6954545 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3    Medium
14 1.7181818 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3    Medium
15 2.3863636 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4      High
16 2.4654545 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4      High
17 2.4295455 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4      High
18 1.0000000 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1       Low
19 0.7340909 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2       Low
20 0.8340909 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1       Low
```

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 disp_cat    newcol
1  21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4       Low 1.1909091
2  21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4       Low 1.3068182
3  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1       Low 1.0545455
4  21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1    Medium 1.4613636
5  18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2    Medium 1.5636364
6  18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1    Medium 1.5727273
7  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4    Medium 1.6227273
8  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2       Low 1.4500000
9  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2       Low 1.4318182
10 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4       Low 1.5636364
11 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4       Low 1.5636364
12 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3    Medium 1.8500000
13 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3    Medium 1.6954545
14 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3    Medium 1.7181818
15 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4      High 2.3863636
16 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4      High 2.4654545
17 14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4      High 2.4295455
18 32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1       Low 1.0000000
19 30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2       Low 0.7340909
20 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1       Low 0.8340909
21 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1       Low 1.1204545
22 15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2    Medium 1.6000000
```

Ordering rows

Ordering the rows of a `data.frame`: dplyr

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

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

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb    newcol disp_cat
1  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4 2.3863636     High
2  10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4 2.4654545     High
3  13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4 1.7454545   Medium
4  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4 1.6227273   Medium
5  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4 2.4295455     High
6  15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8 1.6227273   Medium
7  15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3 1.7181818   Medium
8  15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2 1.5613636   Medium
9  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2 1.6000000   Medium
10 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4 1.4409091   Medium
11 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3 1.8500000   Medium
12 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3 1.6954545   Medium
13 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4 1.5636364      Low
14 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 1.5727273   Medium
15 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 1.5636364   Medium
16 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4 1.5636364      Low
17 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2 1.7477273   Medium
18 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6 1.2590909      Low
19 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4 1.1909091      Low
20 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4 1.3068182      Low
```

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    newcol disp_cat
1  33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1 0.8340909      Low
2  32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1 1.0000000      Low
3  30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2 0.7340909      Low
4  30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2 0.6877273      Low
5  27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1 0.8795455      Low
6  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2 0.9727273      Low
7  24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2 1.4500000      Low
8  22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1 1.0545455      Low
9  22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2 1.4318182      Low
10 21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1 1.1204545      Low
11 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1 1.4613636   Medium
12 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2 1.2636364      Low
13 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4 1.1909091      Low
14 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4 1.3068182      Low
15 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6 1.2590909      Low
16 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4 1.5636364      Low
17 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2 1.7477273   Medium
18 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 1.5636364   Medium
19 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 1.5727273   Medium
20 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4 1.5636364      Low
21 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3 1.6954545   Medium
22 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3 1.8500000   Medium
```

Ordering the rows of a `data.frame`: dplyr

It is a bit more straightforward to mix increasing and decreasing orderings:

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

```
    mpg cyl  disp  hp drat    wt  qsec vs am gear carb    newcol disp_cat
1  10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4 2.4654545     High
2  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4 2.3863636     High
3  13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4 1.7454545   Medium
4  14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4 1.6227273   Medium
5  14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4 2.4295455     High
6  15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8 1.6227273   Medium
7  15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3 1.7181818   Medium
8  15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2 1.5613636   Medium
9  15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2 1.6000000   Medium
10 15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4 1.4409091   Medium
11 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3 1.8500000   Medium
12 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3 1.6954545   Medium
13 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4 1.5636364      Low
14 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1 1.5727273   Medium
15 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2 1.5636364   Medium
16 19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2 1.7477273   Medium
17 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4 1.5636364      Low
18 19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6 1.2590909      Low
19 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4 1.1909091      Low
20 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4 1.3068182      Low
21 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1 1.4613636   Medium
22 21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2 1.2636364      Low
```

Transmutation

The `transmute` function in `dplyr` combines both the `mutate` and `select` functions. One can create new columns and keep the only the columns wanted:

```
transmute(df, newcol2 = wt/2.2, mpg, hp)
```

```
     newcol2  mpg  hp
1  1.1909091 21.0 110
2  1.3068182 21.0 110
3  1.0545455 22.8  93
4  1.4613636 21.4 110
5  1.5636364 18.7 175
6  1.5727273 18.1 105
7  1.6227273 14.3 245
8  1.4500000 24.4  62
9  1.4318182 22.8  95
10 1.5636364 19.2 123
11 1.5636364 17.8 123
12 1.8500000 16.4 180
13 1.6954545 17.3 180
14 1.7181818 15.2 180
15 2.3863636 10.4 205
16 2.4654545 10.4 215
17 2.4295455 14.7 230
18 1.0000000 32.4  66
19 0.7340909 30.4  52
20 0.8340909 33.9  65
21 1.1204545 21.5  97
```

Lab Part 4

Website

Bracket Subsetting

Select specific elements using an index

Often you only want to look at subsets of a data set at any given time. As a review, elements of an R object are selected using the brackets (`[` and `]`).

For example, `x` is a vector of numbers and we can select the second element of `x` using the brackets and an index (2):

```
x = c(1, 4, 2, 8, 10)
x[2]
```

```
[1] 4
```

Select specific elements using an index

We can select the fifth or second AND fifth elements below:

```
x = c(1, 2, 4, 8, 10)
x[5]
```

```
[1] 10
```

```
x[c(2,5)]
```

```
[1]  2 10
```

Subsetting by deletion of entries

You can put a minus (-) before integers inside brackets to remove these indices from the data.

```
x[-2] # all but the second
```

```
[1]  1  4  8 10
```

Note that you have to be careful with this syntax when dropping more than 1 element:

```
x[-c(1,2,3)] # drop first 3
```

```
[1]  8 10
```

```
# x[-1:3] # shorthand. R sees as -1 to 3
x[-(1:3)] # needs parentheses
```

```
[1]  8 10
```

Select specific elements using logical operators

What about selecting rows based on the values of two variables? We use logical statements. Here we select only elements of `x` greater than 2:

```
x
```

```
[1]  1  2  4  8 10
```

```
x > 2
```

```
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

```
x[ x > 2 ]
```

```
[1]  4  8 10
```

Select specific elements using logical operators

You can have multiple logical conditions using the following:

- `&` : AND
- `|` : OR

```
x[ x > 2 & x < 5 ]
```

```
[1] 4
```

```
x[ x > 5 | x == 2 ]
```

```
[1]  2  8 10
```

which function

The `which` functions takes in logical vectors and returns the index for the elements where the logical value is `TRUE`.

```
which(x > 5 | x == 2) # returns index
```

```
[1] 2 4 5
```

```
x[ which(x > 5 | x == 2) ]
```

```
[1]  2  8 10
```

```
x[ x > 5 | x == 2 ]
```

```
[1]  2  8 10
```

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"
 [7] "qsec"     "vs"       "am"       "gear"     "carb"     "newcol"
[13] "disp_cat"
```

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

```
   MPG CYL DISP  hp drat    wt  qsec vs am gear carb   newcol disp_cat
1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 1.190909      Low
2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.306818      Low
3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.054545      Low
4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 1.461364   Medium
5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 1.563636   Medium
6 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 1.572727   Medium
```

```
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 disp_cat
1 21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 1.190909      Low
2 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.306818      Low
3 22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.054545      Low
4 21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 1.461364   Medium
5 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 1.563636   Medium
6 18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 1.572727   Medium
```

```
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 disp_cat
1 21.0   6  160 110 3.90 2.62 16.46  0  1    4    4 1.190909      Low
3 22.8   4  108  93 3.85 2.32 18.61  1  1    4    1 1.054545      Low
```

Subset columns of a `data.frame`:

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

For `data.frame`s 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
```

Biggest 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 a vector, 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
[15] 10.4 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
[29] 15.8 19.7 15.0 21.4
```

```
tbl[, 1]
```

```
# A tibble: 32 x 1
   car
   <chr>
 1 Mazda RX4
 2 Mazda RX4 Wag
 3 Datsun 710
 4 Hornet 4 Drive
 5 Hornet Sportabout
 6 Valiant
 7 Duster 360
 8 Merc 240D
 9 Merc 230
10 Merc 280
# ... with 22 more rows
```

Subset columns of a `data.frame`:

We can select multiple columns using multiple column names:

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

```
    mpg cyl
1  21.0   6
2  21.0   6
3  22.8   4
4  21.4   6
5  18.7   8
6  18.1   6
7  14.3   8
8  24.4   4
9  22.8   4
10 19.2   6
11 17.8   6
12 16.4   8
13 17.3   8
14 15.2   8
15 10.4   8
16 10.4   8
17 14.7   8
18 32.4   4
19 30.4   4
20 33.9   4
21 21.5   4
22 15.5   8
```