

Intro to R

Subsetting Data in R

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. Create a `data.frame` and a `tibble`
2. Rename 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 largely focus on the `dplyr` package which is part of the `tidyverse`.

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.opencasestudies.org/> .

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!
```

— Attaching packages — tidyverse 1.3.1

✓ ggplot2 3.3.3	✓ purrr 0.3.4
✓ tibble 3.1.1	✓ stringr 1.4.0
✓ tidyr 1.1.3	✓ forcats 0.5.1
✓ readr 1.4.0	

— Conflicts — tidyverse_conflicts()

```
x dplyr::filter() masks stats::filter()
x dplyr::lag()     masks stats::lag()
```

Creating a `data.frame` to work with

Here we use one of the datasets that comes with the `jhur` package called `jhu_cars`, which is a (copy of another called `mtcars`). We will now create a toy `data.frame` named `df` using this data:

```
library(jhur)
data(jhu_cars)
df = jhu_cars # df is a copy of jhu_cars
head(df) # changing df does **not** change jhu_cars
```

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

Creating a tibble

If we would like to create a `tibble` ("fancy" `data.frame`), we can use the `as_tibble()` function (from the `tibble` package - part of the `tidyverse` too!).

```
tbl = as_tibble(df)
tbl
```

```
# A tibble: 32 x 12
```

	car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	ca
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Mazda RX4	21	6	160	110	3.9	2.62	16.5	0	1	4	
2	Mazda RX4 ...	21	6	160	110	3.9	2.88	17.0	0	1	4	
3	Datsun 710	22.8	4	108	93	3.85	2.32	18.6	1	1	4	
4	Hornet 4 D...	21.4	6	258	110	3.08	3.22	19.4	1	0	3	
5	Hornet Spo...	18.7	8	360	175	3.15	3.44	17.0	0	0	3	
6	Valiant	18.1	6	225	105	2.76	3.46	20.2	1	0	3	
7	Duster 360	14.3	8	360	245	3.21	3.57	15.8	0	0	3	
8	Merc 240D	24.4	4	147.	62	3.69	3.19	20	1	0	4	
9	Merc 230	22.8	4	141.	95	3.92	3.15	22.9	1	0	4	
10	Merc 280	19.2	6	168.	123	3.92	3.44	18.3	1	0	4	

```
# ... with 22 more rows
```

No rownames in tibbles!

In the “tidy” data format, all information of interest is a variable (not a name). **as of tibble 2.0, rownames are removed.** For example, `mtcars` has each car name as a row name. Here we use the `head()` function to see the first 2 rows of each. In this case we would want to make the rownames a new column first before making into a tibble.

```
head(mtcars, 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(as_tibble(mtcars), 2)
```

```
# A tibble: 2 x 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

Renaming Columns

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

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**!

```
df = dplyr::rename(df, MPG = mpg)
head(df)
```

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

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

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

```
df_upper = dplyr::rename_all(df, toupper)
head(df_upper, 3)
```

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

```
df = dplyr::rename_all(df, tolower)
head(df, 3)
```

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

Lab Part 1

[Website](#)

Subsetting Columns

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
```

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

To grab 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 (gives a `tibble`!)

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

	mpg
1	21.0
2	21.0
3	22.8
4	21.4
5	18.7
6	18.1
7	14.3
8	24.4
9	22.8
10	19.2
11	17.8
12	16.4
13	17.3
14	15.2
15	10.4
16	10.4
17	14.7
18	32.4
19	30.4
20	33.9
21	21.5
22	15.5
23	15.2
24	13.3

Subset columns of a `data.frame`: `dplyr`

Note that if you want a single vector (not a `tibble`), use `pull` or `$`:

```
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 10.0  
[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 19.0  
[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 10.0  
[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 19.0  
[31] 15.0 21.4
```


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
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
23	15.2	8

See the Select “helpers”

Here are a few:

```
one_of() # if they exist  
last_col()  
ends_with()  
contains() # like searching
```

Run the command to see them:

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

Lab Part 2

[Website](#)

Subsetting Rows

Subset rows of a `data.frame`: `dplyr`

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

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

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

- `==` : 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))
```

		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

Subset rows of a `data.frame`: `dplyr`

You can filter by two conditions using `&` or commas:

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

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

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

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

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

```
head(df2, 4)
```

	cyl	hp
1	4	93
2	4	62
3	4	95
4	4	66

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)
```

		car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	newcol	
1		Mazda	RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	1.190909
2	Mazda	RX4	Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	1.306818
3		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` (tidyverse way)

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)
```

		car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	newcol
1	Mazda	RX4	21	6	160	110	3.9	2.620	16.46	0	1	4	4	1.190909
2	Mazda	RX4	Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	1.306818

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

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

```
df$newcol = NULL
```

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)
```

	car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
2	Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
3	Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
4	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
5	Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
6	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 to a `data.frame`: `dplyr`

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

Remove `newcol` and `drat`:

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

	car	mpg	cyl	disp	hp	wt	qsec	vs	am	gear	carb	
1	Mazda RX4	21.0	6	160.0	110	2.620	16.46	0	1	4	4	
2	Mazda RX4 Wag	21.0	6	160.0	110	2.875	17.02	0	1	4	4	
3	Datsun 710	22.8	4	108.0	93	2.320	18.61	1	1	4	1	
4	Hornet 4 Drive	21.4	6	258.0	110	3.215	19.44	1	0	3	1	
5	Hornet Sportabout	18.7	8	360.0	175	3.440	17.02	0	0	3	2	
6	Valiant	18.1	6	225.0	105	3.460	20.22	1	0	3	1	
7	Duster 360	14.3	8	360.0	245	3.570	15.84	0	0	3	4	
8	Merc 240D	24.4	4	146.7	62	3.190	20.00	1	0	4	2	
9	Merc 230	22.8	4	140.8	95	3.150	22.90	1	0	4	2	
10	Merc 280	19.2	6	167.6	123	3.440	18.30	1	0	4	4	
11	Merc 280C	17.8	6	167.6	123	3.440	18.90	1	0	4	4	
12	Merc 450SE	16.4	8	275.8	180	4.070	17.40	0	0	3	3	
13	Merc 450SL	17.3	8	275.8	180	3.730	17.60	0	0	3	3	
14	Merc 450SLC	15.2	8	275.8	180	3.780	18.00	0	0	3	3	
15	Cadillac Fleetwood	10.4	8	472.0	205	5.250	17.98	0	0	3	4	
16	Lincoln Continental	10.4	8	460.0	215	5.424	17.82	0	0	3	4	
17	Chrysler Imperial	14.7	8	440.0	230	5.345	17.42	0	0	3	4	
18	Fiat 128	32.4	4	78.7	66	2.200	19.47	1	1	4	1	
19	Honda Civic	30.4	4	75.7	52	1.615	18.52	1	1	4	2	
20	Toyota Corolla	33.9	4	71.1	65	1.835	19.90	1	1	4	1	
21	Toyota Corona	21.5	4	120.1	97	2.465	20.01	1	0	3	1	35/54
22	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 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)
```

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

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

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

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

	car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
2	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
3	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
4	Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
5	Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
6	Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
7	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
8	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
9	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
11	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
12	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
13	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
14	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
15	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
16	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
17	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
18	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
19	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
20	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
21	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
22	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
23	Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
24	Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	40/542

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

Increasing and decreasing orderings:

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

	car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
2	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
3	Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
4	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
5	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
6	Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
7	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
8	AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
9	Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
10	Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
11	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
12	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
13	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
14	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
15	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
16	Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
17	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
18	Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
19	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
20	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
21	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
22	Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2
23	Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
24	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	41/542

Lab Part 4

[Website](#)

Extra Slides

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 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.
```

ifelse example

```
df$disp
```

```
[1] 160.0 160.0 108.0 258.0 360.0 225.0 360.0 146.7 140.8 167.6 167.6 275.8  
[13] 275.8 275.8 472.0 460.0 440.0  78.7  75.7  71.1 120.1 318.0 304.0 350.0  
[25] 400.0  79.0 120.3  95.1 351.0 145.0 301.0 121.0
```

```
#ifelse(test, yes, no)  
ifelse(df$disp<=200, "low", "high")
```

```
[1] "low" "low" "low" "high" "high" "high" "high" "low" "low" "low"  
[11] "low" "high" "high" "high" "high" "high" "high" "low" "low" "low"  
[21] "low" "high" "high" "high" "high" "low" "low" "low" "high" "low"  
[31] "high" "low"
```

Adding columns to a `data.frame`: `dplyr`

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

```
df = mutate(df,  
             disp_cat = ifelse(displacement <= 200, "Low", "High")  
             )  
  
head(df, 2)
```

		car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	newcol
1	Mazda	RX4	21	6	160	110	3.9	2.620	16.46	0	1	4	4	1.190909
2	Mazda	RX4 Wag	21	6	160	110	3.9	2.875	17.02	0	1	4	4	1.306818
		disp_cat												
1		Low												
2		Low												

Adding columns to a `data.frame`: dplyr

Alternatively, `case_when` provides a clean syntax as well:

```
df = mutate(df,  
             disp_cat2 = case_when(  
               disp <= 200 ~ "Low",  
               disp > 200 ~ "High",  
             ))  
head(df$disp_cat2)
```

```
[1] "Low"  "Low"  "Low"  "High" "High" "High"
```

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] "car"      "mpg"      "cyl"      "disp"     "hp"      "drat"
[7] "wt"      "qsec"     "vs"      "am"      "gear"     "carb"
[13] "newcol"   "disp_cat" "disp_cat2"
```

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

		MPG	CYL	DISP	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	
2	Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	
3	Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	
4	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	
5	Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	
6	Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	
	newcol	disp_cat	disp_cat2										
1	1.190909	Low	Low										
2	1.306818	Low	Low										
3	1.054545	Low	Low										
4	1.461364	High	High										
5	1.563636	High	High										
6	1.572727	High	High										

```
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	disp	hp	DRAT	wt	qsec	vs	am	gear	carb
1		Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
2		Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
3		Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
4		Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
5		Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
6		Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1
	newcol	disp_cat	disp_cat2										
1	1.190909	Low	Low										
2	1.306818	Low	Low										
3	1.054545	Low	Low										
4	1.461364	High	High										
5	1.563636	High	High										
6	1.572727	High	High										

```
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	disp	hp	drat	wt	qsec	vs	am	gear	carb	newcol
1	Mazda	RX4	21.0	6	160	110	3.90	2.62	16.46	0	1	4	4	1.190909
3	Datsun	710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	1	1.054545
		disp_cat	disp_cat2											
1		Low	Low											
3		Low	Low											

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 4 3 3 3 3 4 4 4 4 3 3 3 3 3 3 4 4 4 3 3 3 3 3 4 5 5 5 5 5 4
```

```
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.frame`s, 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] "Mazda RX4"           "Mazda RX4 Wag"       "Datsun 710"
[4] "Hornet 4 Drive"      "Hornet Sportabout"   "Valiant"
[7] "Duster 360"         "Merc 240D"           "Merc 230"
[10] "Merc 280"           "Merc 280C"           "Merc 450SE"
[13] "Merc 450SL"         "Merc 450SLC"         "Cadillac Fleetwood"
[16] "Lincoln Continental" "Chrysler Imperial"   "Fiat 128"
[19] "Honda Civic"         "Toyota Corolla"      "Toyota Corona"
[22] "Dodge Challenger"    "AMC Javelin"         "Camaro Z28"
[25] "Pontiac Firebird"    "Fiat X1-9"           "Porsche 914-2"
[28] "Lotus Europa"        "Ford Pantera L"      "Ferrari Dino"
[31] "Maserati Bora"       "Volvo 142E"
```

```
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
```

Subset columns of a `data.frame`:

We can select multiple columns using multiple column names:

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

		mpg	cyl
1	Mazda RX4	21.0	
2	Mazda RX4 Wag	21.0	
3	Datsun 710	22.8	
4	Hornet 4 Drive	21.4	
5	Hornet Sportabout	18.7	
6	Valiant	18.1	
7	Duster 360	14.3	
8	Merc 240D	24.4	
9	Merc 230	22.8	
10	Merc 280	19.2	
11	Merc 280C	17.8	
12	Merc 450SE	16.4	
13	Merc 450SL	17.3	
14	Merc 450SLC	15.2	
15	Cadillac Fleetwood	10.4	
16	Lincoln Continental	10.4	
17	Chrysler Imperial	14.7	
18	Fiat 128	32.4	
19	Honda Civic	30.4	
20	Toyota Corolla	33.9	
21	Toyota Corona	21.5	
22	Dodge Challenger	15.5	
23	AMC Javelin	15.2	
24	Camaro Z28	13.3	

No rownames in tibbles!

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

```
head(rownames_to_column(mtcars, var = "car"), 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(as_tibble(rownames_to_column(mtcars, var = "car")), 2)
```

```
# A tibble: 2 x 12
```

	car	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Mazda RX4	21	6	160	110	3.9	2.62	16.5	0	1	4	4
2	Mazda RX4 W...	21	6	160	110	3.9	2.88	17.0	0	1	4	4