Subsetting Data in R

Data Wrangling in R

Overview

We showed different ways to read data into R using:

```
readr::read_csv()
readr::read_delim()
readxl::read_excel()
```

In this module, we will show you how select rows and columns of datasets.

Setup

We will be using the dplyr package in the tidyverse.

Here are several resources on how to use dplyr:

- https://dplyr.tidyverse.org/
- https://r4ds.had.co.nz/
- https:
 - //cran.rstudio.com/web/packages/dplyr/vignettes/dplyr.html
- https://stat545.com/dplyr-intro.html

The dplyr package also interfaces well with tibbles.

Dataset

We will be using the diamonds dataset in the ggplot2 package as an example (so make sure you initiate the ggplot2 package if you are following along on your own).

head(diamonds)

```
# A tibble: 6 x 10
                color clarity depth table price
 carat cut
 <dbl> <ord>
                <ord> <ord>
                             <dbl> <dbl> <dbl> <dbl> <dl> <dl> <dbl> <dl
  0.23 Ideal E
                      ST2
                              61.5
                                      55
                                          326
                                               3.95
                                                   3
2 0.21 Premium E
                                      61
                                          326
                                               3.89 3
                      SI1
                              59.8
3 0.23 Good
              Ε
                      VS1
                              56.9
                                      65
                                          327
                                               4.05
                                                    4
  0.29 Premium I
                                      58
                                          334
                                               4.2
                                                     4
4
                      VS2
                              62.4
5 0.31 Good
                      SI2
                              63.3
                                      58
                                          335 4.34 4
6
  0.24 Very Good J
                      VVS2
                              62.8
                                      57
                                          336
                                               3.94
                                                     3
```

Selecting a single column of a data.frame:

To grab just the values from a single column, you would use the pull function. The output will be a vector (and not a tibble).

Since this is a long vector we will just show the first 6 values using the head function around the output of the pull function.

head(pull(diamonds,carat))

[1] 0.23 0.21 0.23 0.29 0.31 0.24

Using the pipe (comes with dplyr):

That was a lot of typing and nested functions, which can be confusing. 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.

Using the pipe (comes with dplyr):

Pipe diamonds into select, then pipe that into pull, and then show the head:

```
diamonds %>% pull(carat) %>% head()
```

```
[1] 0.23 0.21 0.23 0.29 0.31 0.24
```

Selecting a single column of a data.frame:

The pull function is equivalent to using the \$ method (in base R).

We can grab the carat column using the \$ operator.

head(pull(diamonds, carat))

[1] 0.23 0.21 0.23 0.29 0.31 0.24

head(diamonds\$carat)

[1] 0.23 0.21 0.23 0.29 0.31 0.24

Note this does *not* return a tibble (or data.frame) but rather a vector.

Selecting a single column of a data.frame:

The select function extracts one or more columns from a tibble or data.frame and returns a tibble (not a vector).

```
select(diamonds, carat)
```

```
# A tibble: 53,940 x 1
   carat
   <dbl>
   0.23
   0.21
 3
   0.23
   0.29
 5 0.31
```

0.24 0.24

0.26 0.22 10 0.23

8

... with 53,930 more rows

Selecting multiple columns of a data.frame:

The select command from dplyr is very flexible. You just need to list all columns you want to extract separated by commas. You can use this as a way to just keep the columns you want for example.

select(diamonds, carat, depth)

```
# A tibble: 53,940 x 2
   carat depth
   <dbl> <dbl>
 1 0.23 61.5
 2 0.21 59.8
 3
   0.23 56.9
 4
   0.29 62.4
 5
   0.31 63.3
 6
   0.24 62.8
   0.24 62.3
 8
   0.26 61.9
   0.22 65.1
   0.23
         59.4
10
```

See the Select "helpers"

Type tidyselect:: to see functions available.



Here are a few:

```
last_col()
ends_with()
starts_with()
contains() # search for a pattern
everything()
```

Tidyselect helpers

For example, we can take all columns that start with a "c":

```
diamonds %>% select(starts_with("c"))
```

```
# A tibble: 53,940 x 4
  carat cut color clarity
  <dbl> <ord> <ord> <ord>
1 0.23 Ideal E
                    ST2
2 0.21 Premium E
                    SI1
3 0.23 Good E VS1
4 0.29 Premium I VS2
5 0.31 Good
               J SI2
   0.24 Very Good J VVS2
6
   0.24 Very Good I VVS1
8
   0.26 Very Good H
                    SI1
           E VS2
  0.22 Fair
10 0.23 Very Good H
                    VS1
# ... with 53,930 more rows
```

Tidyselect helpers

Or we can take all columns that end with an "e":

```
diamonds %>% select(ends_with("e"))
```

```
# A tibble: 53,940 x 2
  table price
  <dbl> <int>
     55
          326
 2
     61 326
 3
     65 327
     58 334
 5
     58 335
 6
     57
          336
     57
          336
8
     55 337
     61
          337
10
     61
          338
     with 53,930 more rows
```

Tidyselect helpers

We are going to cover "fancier" ways of matching column names (and strings more generally) in the data cleaning lecture.

0.22 Fair

0.3 Good

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

The easiest way to filter is by testing whether numeric observations

```
are greater than or less than some cutoff:
filter(diamonds, depth > 60)
```

# A tibble: 48,	315 x 10					
carat cut	color	clarity	depth	table	price	x
<dbl> <ord></ord></dbl>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>

	carat	cut	color	clarity	depth	table	price	X
	<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>
1	0.23	Ideal	E	SI2	61.5	55	326	3.95
2	0.29	Premium	Т	VS2	62.4	58	334	4.2

	<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>
1	0.23	Ideal	E	SI2	61.5	55	326	3.95
2	0.29	Premium	I	VS2	62.4	58	334	4.2
3	0.31	Good	T	ST2	63.3	58	335	4 34

3	0.31	${\tt Good}$		J	SI2	63.3	58	335	4.34
4	0.24	Very	Good	J	VVS2	62.8	57	336	3.94
5	0.24	Very	Good	I	VVS1	62.3	57	336	3.95
6	0.26	Very	Good	H	SI1	61.9	55	337	4.07

VS2

SI1

65.1

64

61

55

337

339

3.87

4.25

Ε

					•	-		-	
	<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>		<dbl></dbl>	<dbl></dbl>	<int></int>	<d< td=""></d<>
1	0.23	Ideal	E	SI2		61.5	55	326	3
2	0.29	Premium	I	VS2		62.4	58	334	4
3	0.31	Good	T	CID		63 3	58	335	4

You can also using piping here:

0.23 Ideal

0.22 Premium F

10

```
diamonds %>% filter(depth > 60)
```

```
# A tibble: 48,315 x 10
               color clarity depth table price
  carat cut
                                            X
  <dbl> <ord>
              <ord> <ord>
                          <dbl> <dbl> <int> <dbl> <
                                 55
   0.23 Ideal E
                    SI2
                           61.5
                                     326 3.95
  0.29 Premium I
                   VS2
                           62.4
                                 58
                                     334 4.2
                           63.3 58 335 4.34
3
   0.31 Good
               J SI2
  0.24 Very Good J VVS2
                           62.8 57 336 3.94
5
  0.24 Very Good I VVS1
                           62.3 57
                                     336 3.95
   0.26 Very Good H
                           61.9
                                 55
                                         4.07
6
                    SI1
                                     337
   0.22 Fair
               Ε
                   VS2
                           65.1
                                 61
                                     337
                                         3.87
   0.3 Good J
                    SI1
                           64
                                 55
                                     339
                                         4.25
```

56

61

62.8

60.4

340 3.93

342 3.88

... with 48,305 more rows

VS1

SI1

0.7 Very Good F

0.7 Premium E

... with 1,694 more rows

0.71 Good

9

10

You can combine filtering on multiple columns by separating the filter arguments with commas:

```
diamonds %>% filter(depth > 60, table > 60, price > 2775)
```

```
# A tibble: 1,704 x 10
  carat cut
                  color clarity depth table price
                                                      X
```

<dbl> <ord> <ord> <ord> <dbl> <dbl> <int> <dbl> <

1 0.72 Premium F SI1 61.8 61 2777 5.82 ! 2 0.72 Very Good H VS1

60.6 63 2782 5.83

60.9

62.8

62.4

61

61

2812 5.66

5.66

64 2817 5.6

2818

3	0.81	Good	G	SI2	61	61	2789	5.94
4	0.71	Premium	F	VS1	60.1	62	2790	5.77
5	0 71	Premium	G	VS1	62 4	61	2803	5 7

-	0.71	1 I CIII I IIII	1	VDI	00.1	02	2130	0.11
5	0.71	Premium	G	VS1	62.4	61	2803	5.7
6	0.74	Fair	F	VS2	61.1	68	2805	5.82
7	0.7	Good	F	VS1	62.8	61	2810	5.57

VS2

SI1

VS2

F.

You can also filter character strings by a single value or category:

```
diamonds %>% filter(color == "I",
            clarity == "ST2", cut == "Premium")
```

0141	- 0 <i>j</i>	DIL , .	Juo	1 I Om I dm	,
# A tibble: 312	x 10				
carat cut	color	clarity	depth	table pr	ice

# A tibble: 312	x 10						
carat cut	color	clarity	depth	table	price	х	
/JL7\ /J\	/ J \	/ J \	/11-75	/11-15	/:	/11.7	1

#	A tibb]	le: 312 z	x 10						
	carat	cut	color	clarity	depth	table	price	х	
	<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>	<

	carat	cut	color	clarity	aeptn	table	price	X	
	<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>	<0
1	0.42	Premium	I	SI2	61.5	59	552	4.78	4

	<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl> <</dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>	<db.< th=""></db.<>
1	0.42	${\tt Premium}$	I	SI2	61.5	59	552	4.78	4.8
2	1	${\tt Premium}$	I	SI2	58.2	60	2795	6.61	6.5

1	0.42	Premium	Ι	SI2	61.5	59	552	4.78
2	1	Premium	I	SI2	58.2	60	2795	6.61
2	0 0	Danamina	т	CTO	60 0	EΩ	വരവഭ	6 11

2	1	Premium	I	SI2	58.2	60	2795	6.61	6
3	0.9	Premium	I	SI2	62.2	59	2826	6.11	6

2	1	Premium	1	S12	58.2	60	2795	6.61	6.8
3	0.9	Premium	I	SI2	62.2	59	2826	6.11	6.0
4	1.05	Premium	Т	ST2	58.3	57	2911	6 72	6 6

3	0.9	Premium	I	SI2	62.2	59	2826	6.11	6
4	1.05	Premium	I	SI2	58.3	57	2911	6.72	6

4	1.05	${\tt Premium}$	I	SI2	58.3	57	2911	6.72	6.
5	0.91	Premium	I	SI2	62	59	2913	6.18	6.

4	1.05	Premium	1	S12	58.3	57	2911	6.72	ь.
5	0.91	${\tt Premium}$	I	SI2	62	59	2913	6.18	6.
6	0.9	Premium	I	SI2	62.5	58	2948	6.15	6.

5	0.91	Premium	Ι	SI2	62	59	2913	6.18	6.
6	0.9	${\tt Premium}$	I	SI2	62.5	58	2948	6.15	6.
7	0 9	Premium	Т	ST2	60 6	60	2948	6 28	6

6	0.9	Premium	Ι	SI2	62.5	58	2948	6.15	6.
7	0.9	${\tt Premium}$	I	SI2	60.6	60	2948	6.28	6.
_	4 00	ъ .	-	a To	04 -		0000	0	_

-	1.00	1 I Om I am	_	DIZ	00.0	01	2011	0.12	
5	0.91	${\tt Premium}$	I	SI2	62	59	2913	6.18	
6	0.9	${\tt Premium}$	I	SI2	62.5	58	2948	6.15	

8

1.06 Premium I S12 61.5 57 2968 6.57

0.91 Premium I 6.3 SI2 60.2 59 2981 6.29 6.5

3001 6.23

10 60.6 0.9 Premium I SI2 60 # ... with 302 more rows

... with 22,249 more rows

Sometimes you want to be able to filter on matching several values or categories. The %in% operator is useful here:

```
diamonds %>% filter(clarity %in% c("SI1", "SI2"))
```

```
# A tibble: 22,259 x 10
   carat cut
                  color clarity depth table price
                               <dbl> <dbl> <int> <dbl> <
   <dbl> <ord>
                  <ord> <ord>
   0.23 Ideal
                  Ε
                       SI2
                                61.5
                                        55
                                            326
                                                 3.95
   0.21 Premium
                  Ε
                       SI1
                                59.8
                                        61
                                            326
                                                 3.89
 3
   0.31 Good
                       SI2
                                63.3
                                        58
                                            335
                                                 4.34
   0.26 Very Good H
                       SI1
                                61.9
                                        55
                                            337
                                                 4.07
 5
   0.3
        Good
                       SI1
                                64
                                        55
                                            339
                                                 4.25
                                                 3.88
   0.22 Premium
                  F
                       SI1
                                60.4
                                        61
                                            342
```

0.31 Ideal SI2 62.2 54 344 4.35 SI2 60.2 62 345 3.79 8 0.2 Premium Ε 9 0.3 Ι SI2 62 348 4.31 Ideal 54 0.3 63.4 10 Good SI1 54 351 4.23

10

1.02 Premium G

You can mix and match filtering on numeric and categorical/character columns in the same filter() command:

```
diamonds %>% filter(clarity %in% c("SI1", "SI2"),
                    cut == "Premium", price > 3000)
```

```
# A tibble: 3,976 x 10
                 color clarity depth table price
   carat cut
                                                  <dbl> <db?
```

	<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	
1	0.9	Premium	I	SI2	60.6	60	3001	

1	0.9	Premium	I	SI2	60.6	60	3001
_			_				

1	0.9	Premium	1	S12	60.6	60	3001	6.23
2	0.81	Premium	F	SI1	61.9	58	3004	5.99
_			_	~-~				

-	0.0	ı ı omı um	-	212	00.0	00	0001	0.20	•
2	0.81	${\tt Premium}$	F	SI1	61.9	58	3004	5.99	5
3	0.92	Premium	D	SI2	60.2	61	3004	6.32	6

2	0.81	${\tt Premium}$	F	SI1	61.9	58	3004	5.99	5
3	0.92	Premium	D	SI2	60.2	61	3004	6.32	6
4	09	Premium	D	ST1	62 2	60	3013	6 08	6

_			_						_
2	0.81	${\tt Premium}$	F	SI1	61.9	58	3004	5.99	5
3	0.92	${\tt Premium}$	D	SI2	60.2	61	3004	6.32	6
1	ΛΩ	Dromium	ח	QT1	62.2	60	2012	6 08	6

4	0.9	Premium	D	SI1	62.2	60	3013	6.08	6
5	0.96	${\tt Premium}$	E	SI2	62.8	60	3016	6.3	6
6	0.93	${\tt Premium}$	G	SI2	61.4	56	3019	6.27	6

3	0.92	Premium	D	SI2	60.2	61	3004	6.32	(
4	0.9	Premium	D	SI1	62.2	60	3013	6.08	(
5	0 96	Dromium	F	CID	62.8	60	3016	63	

5	0.96	Premium	E	S12	62.8	60	3016	6.3	6.2
6	0.93	Premium	G	SI2	61.4	56	3019	6.27	6.2
7	0.78	Premium	D	SI1	60.4	57	3019	6.02	5.9

4	0.9	Premium	D	SI1	62.2	60	3013	6.08	
5	0.96	Premium	E	SI2	62.8	60	3016	6.3	
6	0 93	Premium	G	ST2	61 4	56	3019	6 27	

0.75 Premium E SI1 61.7 60 3024 5.84 8

SI2

5.8 0.75 Premium D 58 3024 5.96 5.9 SI1 59.2

61.7

58

3027

6.46

6.4

Other useful logical tests:

&: AND

| : OR

<= : less than or equals

>= : greater than or equals

!=: not equals

0.7 Very Good F

... with 1,694 more rows

0.71 Good

0.7 Premium

10

The AND operator (&) is the what is being performed "behind the scenes" when chaining together filter statements with commas:

```
diamonds %>% filter(depth > 60 & table > 60 & price > 2775)
# A tibble: 1.704 \times 10
```

	, , , ,							
carat	cut	color	clarity	depth	table	price	x	
<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>	<0

carat	cut	color	clarity	depth	table	price	X	
<dbl></dbl>	<ord></ord>	<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>	<
1 0 72	Dromium	F	QT1	61.8	61	2777	5 82	

	carao	ouo		COTOT	orar roy	acpon	JUDIO	PTTCC	21
	<dbl></dbl>	<ord></ord>		<ord></ord>	<ord></ord>	<dbl></dbl>	<dbl></dbl>	<int></int>	<dbl></dbl>
1	0.72	Premiu	ım	F	SI1	61.8	61	2777	5.82
2	0.72	Very G	lood	H	VS1	60.6	63	2782	5.83

1	0.72	Premium	F	SI1	61.8	61	2777	5.82
2	0.72	Very Good	H	VS1	60.6	63	2782	5.83
3	0.81	Good	G	SI2	61	61	2789	5.94
1	0 71	D	E	TIC 1	CO 1	60	2700	E 77

2	0.72	Very Good	H	VS1	60.6	63	2782	5.83
3	0.81	Good	G	SI2	61	61	2789	5.94
4	0.71	Premium	F	VS1	60.1	62	2790	5.77

2	0.72	Very Good	H	VS1	60.6	63	2782	5.83
3	0.81	Good	G	SI2	61	61	2789	5.94
4	0.71	Premium	F	VS1	60.1	62	2790	5.77
_								

3	0.81	Good	G	SI2	61	61	2789	5.94
4	0.71	Premium	F	VS1	60.1	62	2790	5.77
5	0 71	Dromium	C	VQ1	62 /	61	2803	5 7

3	0.81 Good	G	217	0.1	ρŢ	2109	5.94	
4	0.71 Premium	F	VS1	60.1	62	2790	5.77	
5	0.71 Premium	G	VS1	62.4	61	2803	5.7	
_		_						

4	0.71 Premium	F	VS1	60.1	62	2790	5.77	
5	0.71 Premium	G	VS1	62.4	61	2803	5.7	
6	0.74 Fair	F	VS2	61.1	68	2805	5.82	

-	· · · -	I I Om I om	-		00.1	~_	_, 00	•••
5	0.71	Premium	G	VS1	62.4	61	2803	5.7
6	0.74	Fair	F	VS2	61.1	68	2805	5.82
7	0.7	Cood	E.	WC1	62.8	61	2010	5 57

5	0.71	Premium	G	VSI	62.4	61	2803	5.7
6	0.74	Fair	F	VS2	61.1	68	2805	5.82
7	0.7	Good	F	VS1	62.8	61	2810	5.57

60.9

62.8

62.4

61

64

61

2812

2818

2817

VS2

SI1

VS2

5.66

5.6

5.66

4	0.71 Premium	F	VS1	60.1	62	2790	5.7
5	0.71 Premium	G	VS1	62.4	61	2803	5.7
6	0 74 Fair	T.	WCO	61 1	60	2005	E 0'

The OR operator () is more permissive than the AND operator:

```
diamonds %>% filter(depth > 60 | table > 60 | price > 2775)
```

```
# A tibble: 52,198 x 10
  carat cut color clarity depth table price x
  <dbl> <ord> <ord> <dbl> <dbl> <int> <dbl> <
1 0.23 Ideal E SI2 61.5 55 326 3.95
```

2	0.21 Premium	E	SII	59.8	61	326	3.89	
3	0.23 Good	E	VS1	56.9	65	327	4.05	
4	0.29 Premium	I	VS2	62.4	58	334	4.2	
5	0 31 Good	Т	ST2	63.3	58	335	4 34	

3	0.23	Good	Ł	VSI	56.9	65	327	4.05
4	0.29	Premium	I	VS2	62.4	58	334	4.2
5	0.31	Good	J	SI2	63.3	58	335	4.34
6	0.24	Verv Good	J	VVS2	62.8	57	336	3.94

4	0.29 Premium	I	VS2	62.4	58	334	4.2
5	0.31 Good	J	SI2	63.3	58	335	4.34
6	0.24 Very Good	lЈ	VVS2	62.8	57	336	3.94
7	0.24 Very Good	lΙ	VVS1	62.3	57	336	3.95
8	0.26 Very Good	l H	SI1	61.9	55	337	4.07

65.1 61

9 0.22 Fair E VS2 59.4 61

10 0.23 Very Good H VS1 # ... with 52,188 more rows

337 3.87

338 4

A tibble: 2 x 10

The OR operator (|) can be a substitute for %in% (although it might take more typing):

```
diamonds %>% filter(clarity =="SI1" | clarity == "SI2") %>9
```

```
      carat cut
      color clarity depth table price
      x
      x
      x

      <dbl> <ord> <ord> <ord> <dbl> <dbl> <dbl> <int> <dbl> <dbl> <int> <dbl> <dbl> 

      1
      0.23 Ideal
      E
      SI2
      61.5
      55
      326
      3.95
      3.96

      2
      0.21 Premium
      E
      SI1
      59.8
      61
      326
      3.89
      3.84
```

diamonds %>% filter(clarity %in% c("SI1", "SI2")) %>% head

```
# A tibble: 2 x 10
carat cut color clarity depth table price x
<dbl> <ord> <ord> <ord> <ord> <dbl> <dbl> <int> <dbl> <int> <dbl> <3.95</d> 3.95</d> 3.95</d> 3.95</d>
```

2 0.21 Premium E SI1 59.8 61 326 3.89 3.84

Combining filter and select:

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

```
diamonds %>%
  filter(clarity == "SI2") %>%
  select(starts_with("c"))
```

```
# A tibble: 9,194 x 4
  carat cut color clarity
  <dbl> <ord> <ord> <ord> <ord>
1 0.23 Ideal E
                   SI2
2 0.31 Good
                   SI2
3 0.31 Ideal
              J SI2
4 0.2 Premium E ST2
5 0.3 Ideal
           I SI2
6 0.3 Good I
                   ST2
7 0.33 Ideal I
                   SI2
  0.33 Ideal I
                   SI2
   0.32 Good
                   SI2
```

Combining filter and select:

The order of these functions matters though, since you can remove columns that you might want to filter on.

```
diamonds %>%
  select(starts_with("c")) %>%
  filter(table > 60))
```

This will result in an error because the table column is now gone after the select() function!

Lab

 ${\sf Link}\ {\sf to}\ {\sf Lab}$