

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.

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` function 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
```

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
tbl = as.tbl(df)
```

Renaming Columns

Renaming Columns of a data.frame: base R

We can use the `colnames` function to directly reassign column names of `df`:

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

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

```
colnames(df)[1:3] = c("mpg", "cyl", "disp") #reset
```

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
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0

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

Renaming Columns of a data.frame: dplyr

```
library(dplyr)
```

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

Renaming Columns of a data.frame: dplyr

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, ...)  
{  
  filter_(.data, .dots = lazyeval::lazy_dots(...))  
}  
<environment: namespace:dplyr>
```

Renaming Columns of a data.frame: dplyr

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:

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
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0

```
df = dplyr::rename(df, mpg = MPG) # reset
```

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
```

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
```

```
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
```

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  
[15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2  
[29] 15.8 19.7 15.0 21.4
```

```
tbl[, 1]
```

```
# A tibble: 32 × 1
```

```
    mpg  
  <dbl>
```

```
1    21.0
```

```
2    21.0
```

```
3    22.8
```

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

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

Select columns of a data.frame: dplyr

The select command from dplyr allows you to subset columns of

```
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

Subsetting Rows

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	car
Mazda	RX4	21.0	6	160	110	3.90	2.62	16.46	0	1	4	
Datsun	710	22.8	4	108	93	3.85	2.32	18.61	1	1	4	

Subset rows of a data.frame:

Let's select the rows of `df` where the `mpg` column is greater than 20 or is less than 14. Without any index for columns, all columns are returned:

```
df[ df$mpg > 20 | df$mpg < 14, ]
```

	mpg	cyl	displacement	hp	drat	wt	qsec	vs
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1

Subset rows of a data.frame:

We can subset both rows and columns at the same time:

```
df[ df$mpg > 20 | df$mpg < 14, c("cyl", "hp")]
```

	cyl	hp
Mazda RX4	6	110
Mazda RX4 Wag	6	110
Datsun 710	4	93
Hornet 4 Drive	6	110
Merc 240D	4	62
Merc 230	4	95
Cadillac Fleetwood	8	205
Lincoln Continental	8	215
Fiat 128	4	66
Honda Civic	4	52
Toyota Corolla	4	65
Toyota Corona	4	97
Camaro Z28	8	245

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

Subset rows of a data.frame: dplyr

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)
```

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

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

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
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4

Removing columns to a data.frame: base R

You can remove a column by assigning to NULL:

```
df$newcol = NULL
```

or selecting only the columns that were not newcol:

```
df = df[, colnames(df) != "newcol"]  
head(df,3)
```

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

Adding new columns to a data.frame: base R

You can also “column **bind**” a data.frame with a vector (or series of vectors), using the `cbind` command:

```
cbind(df, newcol = df$wt/2.2)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0
Merc 450SE	17.3	8	275.8	180	3.07	3.730	17.60	0

Adding columns to a data.frame: dplyr

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

```
print({df = mutate(df, newcol = wt/2.2)})
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	m
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4	1.19
2	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4	1.30
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1	1.05
4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1	1.46
5	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2	1.56
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1	1.57
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4	1.62
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2	1.45
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2	1.43
10	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4	1.56
11	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4	1.56
12	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3	1.85
13	17.8	8	275.8	180	3.07	3.730	17.20	0	0	3	3	1.85

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
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	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
6	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
10	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
11	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4

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
1	21.0	6	160.0	110	2.620	16.46	0	1	4	4
2	21.0	6	160.0	110	2.875	17.02	0	1	4	4
3	22.8	4	108.0	93	2.320	18.61	1	1	4	1
4	21.4	6	258.0	110	3.215	19.44	1	0	3	1
5	18.7	8	360.0	175	3.440	17.02	0	0	3	2
6	18.1	6	225.0	105	3.460	20.22	1	0	3	1
7	14.3	8	360.0	245	3.570	15.84	0	0	3	4
8	24.4	4	146.7	62	3.190	20.00	1	0	4	2
9	22.8	4	140.8	95	3.150	22.90	1	0	4	2
10	19.2	6	167.6	123	3.440	18.30	1	0	4	4
11	17.8	6	167.6	123	3.440	18.90	1	0	4	4
12	16.4	8	275.8	180	4.070	17.40	0	0	3	3
13	17.3	8	275.8	180	3.730	17.60	0	0	3	3
14	15.2	8	275.0	180	3.730	17.60	0	0	3	3

Ordering columns

Ordering the columns of a data.frame: base R

We can use the `colnames` function to get the column names of `df` and then put `newcol` first by subsetting `df` using brackets:

```
cn = colnames(df)
df[, c("newcol", cn[cn != "newcol"])]
```

	newcol	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
1	1.1909091	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
2	1.3068182	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
3	1.0545455	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
4	1.4613636	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
5	1.5636364	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
6	1.5727273	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
7	1.6227273	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
8	1.4500000	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4
9	1.4318182	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4
10	1.5636364	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4
11	1.5636364	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4
12	1.0500000	16.4	8	275.0	180	3.07	4.070	17.40	0	0	3

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
1	1.1909091	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
2	1.3068182	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
3	1.0545455	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
4	1.4613636	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
5	1.5636364	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
6	1.5727273	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
7	1.6227273	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
8	1.4500000	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4
9	1.4318182	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4
10	1.5636364	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4
11	1.5636364	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4
12	1.8500000	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3

Ordering rows

Ordering the rows of a data.frame: base R

We use the `order` function on a vector or set of vectors, in increasing order:

```
df[ order(df$mpg), ]
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	m
15	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4	2.38
16	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4	2.46
24	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4	1.74
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4	1.62
17	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4	2.42
31	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8	1.62
14	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3	1.71
23	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2	1.56
22	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2	1.60
29	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4	1.44
12	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3	1.85
13	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3	1.69
44	17.8	8	265.0	130	3.21	3.440	16.99	0	0	4	4	1.51

Ordering the rows of a data.frame: base R

The decreasing argument will order it in decreasing order:

```
df[ order(df$mpg, decreasing = TRUE), ]
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	m
20	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1	0.83
18	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1	1.00
19	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2	0.73
28	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2	0.68
26	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1	0.87
27	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2	0.97
8	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2	1.45
3	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1	1.05
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2	1.43
21	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1	1.12
4	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1	1.46
32	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2	1.26
1	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4	1.19
2	21.0	6	161.3	110	3.85	2.875	17.02	0	1	4	4	1.19

Ordering the rows of a data.frame: base R

You can pass multiple vectors, and must use the negative (using -) to mix decreasing and increasing orderings (sort increasing on x and decreasing on y):

```
df[ order(df$mpg, -df$hp), ]
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	m
16	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4	2.46
15	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4	2.38
24	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4	1.74
7	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4	1.62
17	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4	2.42
31	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8	1.62
14	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3	1.71
23	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2	1.56
22	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2	1.60
29	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4	1.44
12	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3	1.85
18	17.0	8	275.8	130	3.21	3.570	17.30	0	0	3	3	1.41

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	m
1	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4	2.38
2	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4	2.46
3	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4	1.74
4	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4	1.62
5	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4	2.42
6	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8	1.62
7	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3	1.71
8	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2	1.56
9	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2	1.60
10	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4	1.44
11	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3	1.85
12	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3	1.69
13	17.8	8	260.0	150	3.21	3.170	16.99	0	1	5	4	1.46
14	18.1	4	295.0	150	3.54	3.440	15.42	0	1	5	4	1.58
15	18.7	4	265.0	150	3.92	3.300	16.99	0	1	5	4	1.52
16	19.2	4	264.0	150	3.92	3.300	16.99	0	1	5	4	1.52
17	19.4	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
18	19.6	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
19	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
20	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
21	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
22	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
23	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
24	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
25	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
26	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
27	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
28	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
29	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52
30	19.7	4	279.0	150	3.92	3.300	16.99	0	1	5	4	1.52

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	m
1	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1	0.83
2	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1	1.00
3	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2	0.73
4	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2	0.68
5	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1	0.87
6	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2	0.97
7	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2	1.45
8	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1	1.05
9	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2	1.43
10	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1	1.12
11	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1	1.46
12	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2	1.26
13	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4	1.19
14	20.9	6	133.0	115	3.23	2.875	17.30	0	1	4	4	1.16

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	m
1	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4	2.46
2	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4	2.38
3	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4	1.74
4	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4	1.62
5	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4	2.42
6	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8	1.62
7	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3	1.71
8	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2	1.56
9	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2	1.60
10	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4	1.44
11	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3	1.85
12	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3	1.69
13	17.8	8	299.0	150	3.21	3.440	16.99	0	0	3	2	1.61
14	18.1	4	260.0	150	3.00	3.170	14.50	0	1	5	4	1.44
15	18.7	4	265.0	150	3.00	3.170	14.50	0	1	5	4	1.44
16	19.2	4	264.0	150	3.00	3.170	14.50	0	1	5	4	1.44
17	19.4	4	279.0	150	3.00	3.170	14.50	0	1	5	4	1.44
18	19.6	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
19	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
20	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
21	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
22	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
23	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
24	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
25	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
26	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
27	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
28	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
29	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44
30	19.7	4	276.0	150	3.00	3.170	14.50	0	1	5	4	1.44

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	1.6733333	16.4	123