

Anatomy of dplyr Functions

filter

- Filter the data frame to only show a particular set of values

Means “and then”



Must be a double equals

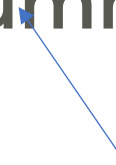


```
NewDataFrame <- DataFrame %>% filter(column ==  
value)
```

Argument Name



Number – no quotes
Word - quotes



filter %in%

- Filter the data frame to only show a particular set of values in a vector

Means “and then”

Create a vector of the values

Double parentheses

```
NewDataFrame <- DataFrame %>% filter(column %in%  
c(value, value))
```

Argument Name

Number – no quotes
Word - quotes

arrange

- Order the data frame by a variable's values

Means "and then"

desc: Largest first
asc: Smallest first

```
NewDataFrame <- DataFrame %>%  
  arrange(desc(column))
```

Argument Name

Double parentheses

The diagram illustrates the components of the `arrange` function syntax. It shows the code `NewDataFrame <- DataFrame %>% arrange(desc(column))`. Four annotations with arrows point to specific parts: 'Means "and then"' points to the `%>%` operator; 'desc: Largest first' and 'asc: Smallest first' point to the `desc` argument; 'Argument Name' points to the `column` argument; and 'Double parentheses' points to the closing parentheses of the `arrange` function call.

select

- Choose variables (columns)

Means “and then”

Name of the columns to keep

```
NewDataFrame <- DataFrame %>% select(column,  
column)
```

Argument Name

The diagram illustrates the syntax of the `select` function. It shows the code `NewDataFrame <- DataFrame %>% select(column, column)`. Three blue arrows point to specific parts of the code: one from the text 'Means "and then"' to the `%>%` operator, one from 'Name of the columns to keep' to the first `column` argument, and one from 'Argument Name' to the second `column` argument.

summarise

- Get an aggregate value for a column

Means “and then”

Name of your new column

```
NewDataFrame <- DataFrame %>% summarise(NewColumnName =  
function(column))
```

Argument Name

Options include:

- median
- mean
- max
- min
- And More!

group_by

- Rolling things up by a variable
- Must be used in conjunction with summarise to get the aggregation

```
NewDataFrame <- DataFrame %>%  
  group_by(column) %>%  
  summarise(NewColumnName =  
function(column))
```

The AMAZING power of dplyr

- Can mix and match and combine with %>% all in one new dataset!