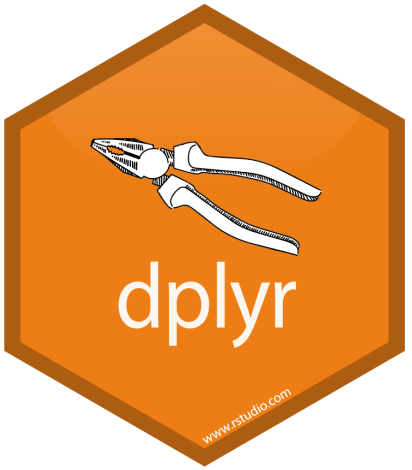
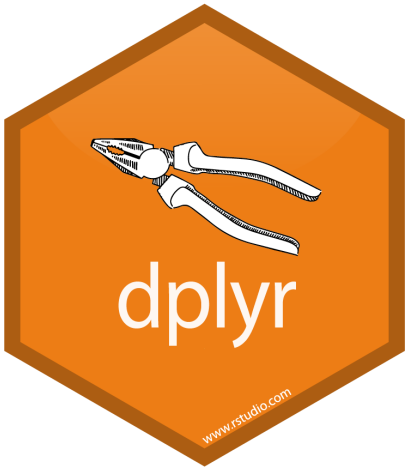
Introduction to dplyr

Tugba Ozturk


dplyr
www.rstudio.com



`dplyr` package is a set of tools using split-apply-combine strategy. Here is a list of important `dplyr` verbs:

- `arrange()` for changing the ordering of the rows.
- `select()` for selecting one or more variables.
- `mutate()` for creating a new variable as a function of existing variables.
- `filter()` for row-wise subsetting.
- `summarise()` for reducing multiple values down to a single summary.
- `group_by()` for group operations.

Installation

Installation

```
install.packages("dplyr") #to install
library("dplyr") #to load
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Data set

We will use a data set called `Orange` having 35 rows and 3 columns of records of the growth of orange trees.

Data set

We will use a data set called `Orange` having 35 rows and 3 columns of records of the growth of orange trees.

```
head(Orange)
```

```
##   Tree  age circumference
## 1    1  118            30
## 2    1  484            58
## 3    1  664            87
## 4    1 1004           115
## 5    1 1231           120
## 6    1 1372           142
```

Click here for more information about `Orange`.

arrange(dataframe, colname)

To order the rows by a particular column:

arrange(dataframe, colname)

To order the rows by a particular column:

```
head(arrange(Orange, circumference))
```

```
##   Tree age circumference
## 1    1 118            30
## 2    3 118            30
## 3    5 118            30
## 4    4 118            32
## 5    2 118            33
## 6    5 484            49
```

arrange(dataframe, colname)

To order the rows by a particular column:

```
head(arrange(Orange, circumference))
```

```
##   Tree age circumference
## 1    1 118            30
## 2    3 118            30
## 3    5 118            30
## 4    4 118            32
## 5    2 118            33
## 6    5 484            49
```

```
{
    {
        head(arrange(Orange, desc(circumference)))
    }
}
```

```
##   Tree  age circumference
## 1    4 1582           214
## 2    4 1372           209
## 3    2 1372           203
## 4    2 1582           203
## 5    4 1231           179
## 6    5 1582           177
```

arrange(dataframe, colname)

Did we **actually** change the data frame?

arrange(dataframe, colname)

Did we **actually** change the data frame?

No, we did not. To make the effect of `arrange` function permenant:

```
Orange <- arrange(Orange, circumference)
```

Pipe Operator (%>%)

`dplyr` imports this operator from `magrittr`. `%>%` pipes the output of one function to the input of another function.

Pipe Operator (%>%)

`dplyr` imports this operator from `magrittr`. `%>%` pipes the output of one function to the input of another function.

```
Orange %>% arrange(circumference) %>% head(4)
```

```
##   Tree age circumference
## 1    1 118            30
## 2    3 118            30
## 3    5 118            30
## 4    4 118            32
```

Pipe Operator (%>%)

`dplyr` imports this operator from `magrittr`. `%>%` pipes the output of one function to the input of another function.

```
Orange %>% arrange(circumference) %>% head(4)
```

```
##   Tree age circumference
## 1    1 118            30
## 2    3 118            30
## 3    5 118            30
## 4    4 118            32
```

In the **descending** order:

```
Orange %>% arrange(desc(circumference)) %>% head(4)
```

```
##   Tree  age circumference
## 1    4 1582           214
## 2    4 1372           209
## 3    2 1372           203
## 4    2 1582           203
```

select()

To select a particular column:

```
Orange %>% select(circumference) %>% head(3)
```

```
##   circumference
## 1            30
## 2            58
## 3            87
```

select()

To select a particular column:

```
Orange %>% select(circumference) %>% head(3)
```

```
##   circumference
## 1            30
## 2            58
## 3            87
```

Additional `options`:

- `ends_with()` to select columns that end with a character string
- `contains()` to select columns that contain a character string
- `matches()` to select columns that match a regular expression
- `one_of()` to select columns names that are from a group of names

select()

To select a particular column:

```
Orange %>% select(circumference) %>% head(3)
```

```
##   circumference
## 1            30
## 2            58
## 3            87
```

Additional `options`:

- `ends_with()` to select columns that end with a character string
- `contains()` to select columns that contain a character string
- `matches()` to select columns that match a regular expression
- `one_of()` to select columns names that are from a group of names

```
Orange %>% select(starts_with("a")) %>% head(3)
```

```
##   age
## 1 118
## 2 484
## 3 664
```

```
Orange %>% select(contains("e")) %>% head(3)
```

select()

To select a column and save it as a **vector**:

select()

To select a column and save it as a **vector**:

```
ages <- as.vector(Orange %>% select(age))
ages <- Orange %>% .$age
ages <- Orange %>% pull(age)
```

select()

To select a column and save it as a **vector**:

```
ages <- as.vector(Orange %>% select(age))
ages <- Orange %>% .$age
ages <- Orange %>% pull(age)
```

To reorder the columns:

select()

To select a column and save it as a **vector**:

```
ages <- as.vector(Orange %>% select(age))
ages <- Orange %>% .$age
ages <- Orange %>% pull(age)
```

To reorder the columns:

```
Orange %>% select(circumference,age,Tree) %>% head()
```

```
##   circumference  age Tree
## 1            30  118    1
## 2            58  484    1
## 3            87  664    1
## 4           115 1004    1
## 5           120 1231    1
## 6           142 1372    1
```

Add a minus before the column name to remove it!

mutate()

To create new columns:

mutate()

To create new columns:

```
Orange %>% mutate(radius=round(circumference/(2*pi),1)) %>% head()
```

```
##   Tree  age circumference radius
## 1    1  118            30    4.8
## 2    1  484            58    9.2
## 3    1  664            87   13.8
## 4    1 1004           115   18.3
## 5    1 1231           120   19.1
## 6    1 1372           142   22.6
```

mutate()

To create new columns:

```
Orange %>% mutate(radius=round(circumference/(2*pi),1)) %>% head()
```

```
##   Tree  age circumference radius
## 1    1  118            30    4.8
## 2    1  484            58    9.2
## 3    1  664            87   13.8
## 4    1 1004           115   18.3
## 5    1 1231           120   19.1
## 6    1 1372           142   22.6
```

Use `transmute()` to get rid of the old variables:

mutate()

To create new columns:

```
Orange %>% mutate(radius=round(circumference/(2*pi),1)) %>% head()
```

```
##   Tree  age circumference radius
## 1    1  118            30    4.8
## 2    1  484            58    9.2
## 3    1  664            87   13.8
## 4    1 1004           115   18.3
## 5    1 1231           120   19.1
## 6    1 1372           142   22.6
```

Use `transmute()` to get rid of the old variables:

```
Orange %>% transmute(radius=round(circumference/(2*pi),1)) %>% head()
```

```
##   radius
## 1    4.8
## 2    9.2
## 3   13.8
## 4   18.3
## 5   19.1
## 6   22.6
```

filter()

To select rows:

filter()

To select rows:

```
Orange %>% filter(circumference==100) %>% head()
```

```
## [1] Tree          age           circumference
## <0 rows> (or 0-length row.names)
```

```
Orange %>% filter(circumference>=100) %>% head(4)
```

```
##   Tree  age circumference
## 1    1 1004           115
## 2    1 1231           120
## 3    1 1372           142
## 4    1 1582           145
```

filter()

To select rows with **multiple** conditions:

filter()

To select rows with **multiple** conditions:

```
Orange %>% filter(circumference < 100 & age > 500 ) %>% head(3)
```

```
##   Tree age circumference
## 1    1 664            87
## 2    3 664            75
## 3    5 664            81
```

filter()

To select rows with **multiple** conditions:

```
Orange %>% filter(circumference < 100 & age > 500 ) %>% head(3)
```

```
##   Tree age circumference
## 1    1 664            87
## 2    3 664            75
## 3    5 664            81
```

If you are interested in the `Tree` column:

```
Orange %>% filter(circumference < 100 & age > 500 ) %>% select(Tree) %>% head()
```

```
##   Tree
## 1    1
## 2    3
## 3    5
```

Practice

Tree	age	circumference
1	118	30
1	484	58
1	664	87
1	1004	115
1	1231	120
1	1372	142

- Create a new column which is the ratio of area to circumference and name it as growth
- Find out the age corresponds to the minimum of value in the growth column for the third tree.

Practice

Tree	age	circumference
1	118	30
1	484	58
1	664	87
1	1004	115
1	1231	120
1	1372	142

- Create a new column which is the ratio of area to circumference and name it as growth
- Find out the age corresponds to the minimum of value in the growth column for the third tree.

```
answer <- Orange %>% mutate(growth=circumference/age) %>%
  filter(Tree==3) %>% filter(growth==min(growth))
print(answer$age)
```

```
## [1] 1582
```

summarize()

To collapse a data frame to a single value:

summarize()

To collapse a data frame to a single value:

```
Orange %>% summarize(r_ave=mean(circumference/(2*pi)))
```

```
##      r_ave
## 1 18.43924
```

summarize()

To collapse a data frame to a single value:

```
Orange %>% summarize(r_ave=mean(circumference/(2*pi)))
```

```
##      r_ave
## 1 18.43924
```

In case of `NA` values, use the following:
`summarize(r_ave=mean(circumference/(2*pi),na.rm = TRUE))`.

summarize()

To collapse a data frame to a single value:

```
Orange %>% summarize(r_ave=mean(circumference/(2*pi)))
```

```
##      r_ave
## 1 18.43924
```

In case of `NA` values, use the following:
`summarize(r_ave=mean(circumference/(2*pi),na.rm = TRUE))`.

`summarize()` is often used with `group_by`:

```
Orange %>% group_by(Tree) %>% head(2)
```

```
## # A tibble: 2 x 3
## # Groups:   Tree [1]
##   Tree    age circumference
##   <ord> <dbl>         <dbl>
## 1 1       118            30
## 2 1       484            58
```

summarize() and group_by()

An example combining both:

```
Orange %>% group_by(Tree) %>% summarize(count=n(),
   age_ave=mean(age), cir_ave=mean(circumference))
```

```
## # A tibble: 5 x 4
##   Tree  count age_ave cir_ave
##   <ord> <int>   <dbl>   <dbl>
## 1 3         7    922.    94
## 2 1         7    922.    99.6
## 3 5         7    922.   111.
## 4 2         7    922.   135.
## 5 4         7    922.   139.
```

summarize() and group_by()

An example combining both:

```
Orange %>% group_by(Tree) %>% summarize(count=n(),
   age_ave=mean(age), cir_ave=mean(circumference))
```

```
## # A tibble: 5 x 4
##   Tree  count age_ave cir_ave
##   <ord> <int>   <dbl>   <dbl>
## 1 3         7    922.    94
## 2 1         7    922.    99.6
## 3 5         7    922.   111.
## 4 2         7    922.   135.
## 5 4         7    922.   139.
```

- `n()` calculates the number of observations (rows) in the group.

summarize() and group_by()

An example combining both:

```
Orange %>% group_by(Tree) %>% summarize(count=n(),
   age_ave=mean(age), cir_ave=mean(circumference))
```

```
## # A tibble: 5 x 4
##   Tree  count age_ave cir_ave
##   <ord> <int>   <dbl>   <dbl>
## 1 3         7    922.    94
## 2 1         7    922.    99.6
## 3 5         7    922.   111.
## 4 2         7    922.   135.
## 5 4         7    922.   139.
```

- `n()` calculates the number of observations (rows) in the group.
- Other aggregate functions which can be used: `max()`, `mean()`, `median()`, `min()`, `sd()`, `sum()` and the interquartile range (`IQR()`).

A lesser known function: `full_join()`

Tree	age	circumference
1	118	30
1	484	58
1	664	87
1	1004	115
1	1231	120

Tree	Country
1	US
2	CAN
3	FRA
4	CAN
5	UK

A lesser known function: `full_join()`

Tree	age	circumference
1	118	30
1	484	58
1	664	87
1	1004	115
1	1231	120

Tree	Country
1	US
2	CAN
3	FRA
4	CAN
5	UK

```
full_join(Orange,df) %>% head(3)
```

```
## Joining, by = "Tree"

##   Tree age circumference Country
## 1    1 118            30      US
## 2    1 484            58      US
## 3    1 664            87      US
```

A lesser known function: `full_join()`

Tree	age	circumference
1	118	30
1	484	58
1	664	87
1	1004	115
1	1231	120

Tree	Country
1	US
2	CAN
3	FRA
4	CAN
5	UK

```
full_join(Orange,df) %>% head(3)
```

```
## Joining, by = "Tree"
```

```
##   Tree age circumference Country
## 1    1 118            30      US
## 2    1 484            58      US
## 3    1 664            87      US
```

Check the documentation for two table verbs.

Acknowledgment

Here is a list of the resources I've used for this talk:

- The official `dplyr` documentation
- Jenny Bryan's STAT 545 course page
- Another course page