Introduction to dplyr

Dr. Natalia Costa Araujo
nat.costaaraujo@gmail.com

University of Georgia

07 December, 2019

library(dplyr)

```r
library(dplyr)
```

- Part of the `library(tidyverse)` which is a collection of R packages designed for data science
- Grammar of data manipulation - intuitive functions to organize data

tibble datasets

- tidyverse's version of data frames
- Does not convert character to factors automatically as `data.frame()`
- More informative print of data
- `tibble()` to define a new tibble dataset
- `as_tibble()` to make a data frame into a tibble

Air quality data

Daily air quality measurements in New York (May to September 1973)

```
data(airquality)
air_dplyr <- as_tibble(airquality)
air_dplyr
```

```
## # A tibble: 153 x 6
##    Ozone Solar.R  Wind  Temp Month   Day
##    <int>   <int> <dbl> <int> <int> <int>
##  1    41     190   7.4    67     5     1
##  2    36     118   8      72     5     2
##  3    12     149  12.6    74     5     3
##  4    18     313  11.5    62     5     4
##  5    NA      NA  14.3    56     5     5
##  6    28      NA  14.9    66     5     6
##  7    23     299   8.6    65     5     7
##  8    19      99  13.8    59     5     8
##  9     8      19  20.1    61     5     9
## 10    NA     194   8.6    69     5    10
## # ... with 143 more rows
```

Air quality data

- Ozone: Mean ozone in parts per billion from 1300 to 1500 hours at Roosevelt Island
- Solar.R: Solar radiation in Langleys in the frequency band 4000-7700 Angstroms from 0800 to 1200 hours at Central Park
- Wind: Average wind speed in miles per hour at 0700 and 1000 hours at LaGuardia Airport
- Temp: Maximum daily temperature in degrees Fahrenheit at La Guardia Airport.

Air quality data

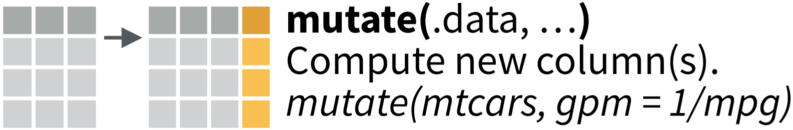
Goals:

- Interested in days with average wind speed of 5 miles per hour or more
- Convert maximum daily temperature from Fahrenheit to Celsius
- Compare estimates by month for different variables

mutate()

- `mutate()` creates new variables that are a function of existing variables

```
mutate(data,new_Var=f(variable))
```



mutate(.data, …**)**
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)

mutate()

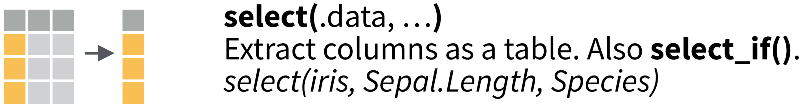
```
air_dplyr <- mutate(air_dplyr,Temp_c=(Temp-32)*(5/9))
air_dplyr
```

```
## # A tibble: 153 x 7
##    Ozone Solar.R  Wind  Temp Month   Day Temp_c
##    <int>   <int> <dbl> <int> <int> <int>  <dbl>
##  1    41     190   7.4    67     5     1   19.4
##  2    36     118   8      72     5     2   22.2
##  3    12     149  12.6    74     5     3   23.3
##  4    18     313  11.5    62     5     4   16.7
##  5    NA      NA  14.3    56     5     5   13.3
##  6    28      NA  14.9    66     5     6   18.9
##  7    23     299   8.6    65     5     7   18.3
##  8    19      99  13.8    59     5     8   15
##  9     8      19  20.1    61     5     9   16.1
## 10    NA     194   8.6    69     5    10   20.6
## # ... with 143 more rows
```

select()

Then we can keep the dataset more concise and delete the column with maximum daily temperatures in Fahrenheit using the `select()`

- `select()` picks variables to be remain in the data set or removes the unwanted variables



select(.data, …**)**
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

select()

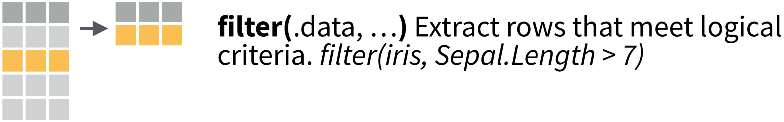
```
air_dplyr <- select(air_dplyr,-Temp)
air_dplyr
```

```
## # A tibble: 153 x 6
##    Ozone Solar.R  Wind Month   Day Temp_c
##    <int>   <int> <dbl> <int> <int>  <dbl>
##  1    41     190   7.4     5     1   19.4
##  2    36     118   8       5     2   22.2
##  3    12     149  12.6     5     3   23.3
##  4    18     313  11.5     5     4   16.7
##  5    NA      NA  14.3     5     5   13.3
##  6    28      NA  14.9     5     6   18.9
##  7    23     299   8.6     5     7   18.3
##  8    19      99  13.8     5     8   15
##  9     8      19  20.1     5     9   16.1
## 10    NA     194   8.6     5    10   20.6
## # ... with 143 more rows
```

filter()

We can now filter the days with average wind speed of least 5 miles per hour, using `filter()`

- `filter()` picks rows based on their values



filter(.data, …**)** Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

filter()

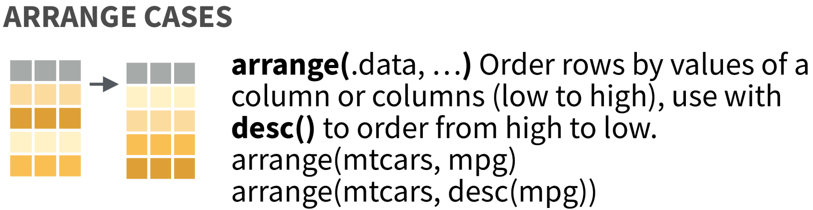
```
air_dplyr <- filter(air_dplyr,Wind>=5)
air_dplyr
```

```
## # A tibble: 143 x 6
##    Ozone Solar.R  Wind Month   Day Temp_c
##    <int>   <int> <dbl> <int> <int>  <dbl>
##  1    41     190   7.4     5     1   19.4
##  2    36     118   8       5     2   22.2
##  3    12     149  12.6     5     3   23.3
##  4    18     313  11.5     5     4   16.7
##  5    NA      NA  14.3     5     5   13.3
##  6    28      NA  14.9     5     6   18.9
##  7    23     299   8.6     5     7   18.3
##  8    19      99  13.8     5     8   15
##  9     8      19  20.1     5     9   16.1
## 10    NA     194   8.6     5    10   20.6
## # ... with 133 more rows
```

arrange()

We can rearrange the rows in any way we want using `arrange()`

- `arrange()` changes the ordering of the rows based in one or more variables



ARRANGE CASES

arrange(.data, …**)** Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

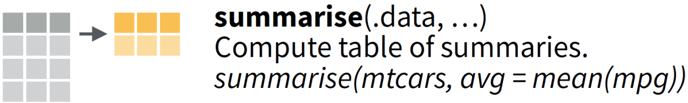
arrange()

```
air_dplyr <- arrange(air_dplyr,Month,desc(Wind))
air_dplyr
```

```
## # A tibble: 143 x 6
##    Ozone Solar.R  Wind Month   Day Temp_c
##    <int>   <int> <dbl> <int> <int>  <dbl>
##  1     8      19  20.1     5     9   16.1
##  2     6      78  18.4     5    18   13.9
##  3    11     320  16.6     5    22   22.8
##  4    NA      66  16.6     5    25   13.9
##  5    28      NA  14.9     5     6   18.9
##  6    NA     266  14.9     5    26   14.4
##  7    45     252  14.9     5    29   27.2
##  8    NA      NA  14.3     5     5   13.3
##  9    19      99  13.8     5     8   15
## 10    18      65  13.2     5    15   14.4
## # ... with 133 more rows
```

summarise() or summarize()

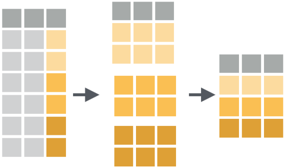
- `summarise()` or `summarize()` creates summary statistics for the specified variables



summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

group_by()

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

group_by(.data, ..., add = FALSE**)**
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …**)**
Returns ungrouped copy of table.
ungroup(g_iris)

group_by()

```
air_dplyr <- group_by(air_dplyr,Month)
air_dplyr
```

```
## # A tibble: 143 x 6
## # Groups:   Month [5]
##    Ozone Solar.R  Wind Month   Day Temp_c
##    <int>   <int> <dbl> <int> <int>  <dbl>
##  1     8      19  20.1     5     9   16.1
##  2     6      78  18.4     5    18   13.9
##  3    11     320  16.6     5    22   22.8
##  4    NA      66  16.6     5    25   13.9
##  5    28      NA  14.9     5     6   18.9
##  6    NA     266  14.9     5    26   14.4
##  7    45     252  14.9     5    29   27.2
##  8    NA      NA  14.3     5     5   13.3
##  9    19      99  13.8     5     8   15
## 10    18      65  13.2     5    15   14.4
## # ... with 133 more rows
```

Summarizing by Month

```
summarize(air_dplyr,Mean=mean(Temp_c),Median=median(Temp_c),
          StDev=sd(Temp_c))
```

```
## # A tibble: 5 x 4
##   Month  Mean Median StDev
##   <int> <dbl>  <dbl> <dbl>
## 1     5  18.6   18.9  3.81
## 2     6  26.3   25.8  3.77
## 3     7  28.9   28.9  2.48
## 4     8  28.6   27.8  3.69
## 5     9  24.3   24.4  4.10
```

Using pipe

```r
air_summary <- airquality %>%
  mutate(Temp_c=(Temp-32)*(5/9)) %>%
  select(-Temp) %>%
  filter(Wind>=5) %>%
  arrange(Month,desc(Wind)) %>%
  group_by(Month) %>%
  summarise(Mean=mean(Temp_c),Median=median(Temp_c),
            StDev=sd(Temp_c))
```

Using pipe

```
air_summary
```

```
## # A tibble: 5 x 4
##   Month  Mean Median StDev
##   <int> <dbl>  <dbl> <dbl>
## 1     5  18.6   18.9  3.81
## 2     6  26.3   25.8  3.77
## 3     7  28.9   28.9  2.48
## 4     8  28.6   27.8  3.69
## 5     9  24.3   24.4  4.10
```

- Some people report that when dealing with very large datasets, `library(dplyr)` can be slower than base R
- R code (attached) shows a similar approach to do the same things as we did here, but without `library(dplyr)`

Thank you for your attention

Any questions or ideas?

References

Figures and information from “Data Transformation with dplyr : CHEAT SHEET”, from RStudio

Rmarkdown template for slides from: https://github.com/rladies/resources