ATAL Sponsored online five-Days Faculty Development Programme
on
“Data Analytics using Python/R programming”
Organized by
Department of CSE, NIT Puducherry

R packages for data science

Presented by: Siddharth R
Department of Computer Science and Engineering
Amrita School of Engineering, Amrita Vishwa Vidyapeetham,
Chennai Campus

August 27, 2021

Agenda

dplyr package

ggplot2 package

caret package

Section 1

dplyr package

Manipulating Data frames

- The data frame is a key data structure in statistics and in R.
- There is one observation per row and each column represents a variable, a measure, feature, or characteristic of that observation
- we have already discussed use of [] and $ operators to extract subsets of data frames
- The *dplyr* package provide a highly optimized set of routines specifically for dealing with data frames.

dplyr package

- The *dplyr* package was developed by Hadley Wickham of RStudio
- The dplyr package **does not** provide any **"new"** functionality to R per se
- Everything *dplyr* does could already be done with base R, but it greatly simplifies existing functionality in R.
- dplyr package provides a "grammar" (in particular,verbs) for data manipulation and for operating on data frames.
- The *dplyr* functions are very fast

Basic grammar in *dplyr*

- **select** - return a subset of the columns of a data frame, using a flexible notation
- **filter** - extract a subset of rows from a data frame based on logical conditions
- **arrange** - reorder rows of a data frame
- **rename** - rename variables in a data frame
- $\% > \%$: the “pipe” operator is used to connect multiple verb actions together into a pipeline

Common characteristics

All of the functions that we will discuss have a few common characteristics

- The first argument is a data frame.
- The subsequent arguments describe what to do with the data frame specified in the first argument, and you can refer to columns in the data frame directly without using the $ operator
- The return result of a function is a new data frame
- Data frames must be properly formatted and annotated - there should be one observation per row, and each column should represent a feature or characteristic of that observation.

Installing and loading dplyr package

- install.packages("dplyr") - To install the package
- library(dplyr) - load it into your R session
- You may get some warnings when the package is loaded because there are functions in the dplyr package that have the same name as functions in other packages.

select

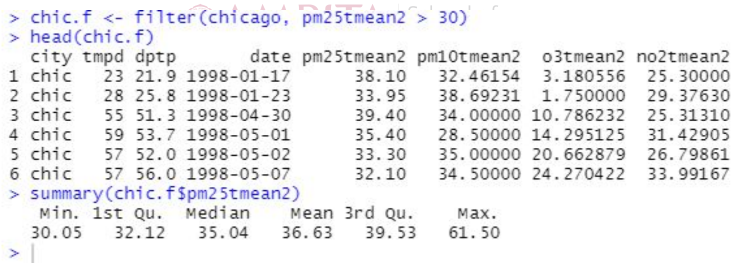
- The select() function can be used to select **columns** of a data frame that you want to focus on.
- Often you'll have a large data frame containing "all" of the data, but any given analysis might only use a subset of variables or observations.
- The select() function allows you to get the few columns you might need.
- Note that the **:** normally cannot be used with names or strings, but inside the select() function you can use it to specify a range of variable names.
- You can also omit variables using the select() function by using the negative sign.

Additional Options in select

- starts_with() = select all columns that start with the character string
- ends_with() = Select columns that end with a character string
- contains() = Select columns that contain a character string
- one_of() = Select columns names that are from a group of names

filter

- The filter() function is used to extract subsets of rows from a data frame

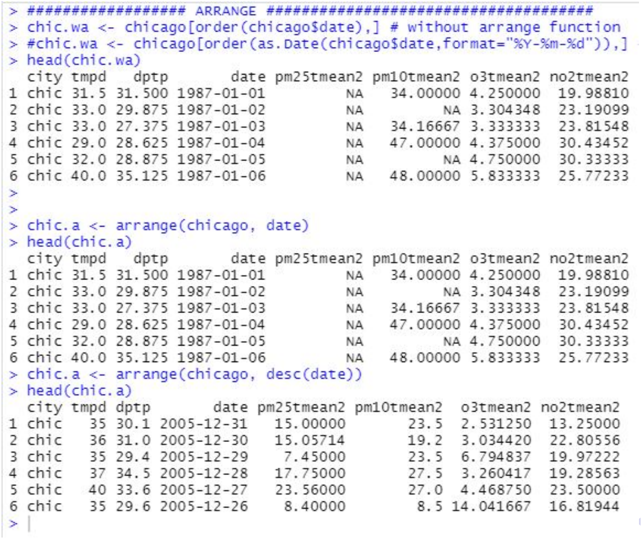


```
> chic.f <- filter(chicago, pm25tmean2 > 30)
> head(chic.f)
  city tmpd dptp       date pm25tmean2 pm10tmean2  o3tmean2 no2tmean2
1 chic   23 21.9 1998-01-17      38.10   32.46154  3.180556  25.30000
2 chic   28 25.8 1998-01-23      33.95   38.69231  1.750000  29.37630
3 chic   55 51.3 1998-04-30      39.40   34.00000 10.786232  25.31310
4 chic   59 53.7 1998-05-01      35.40   28.50000 14.295125  31.42905
5 chic   57 52.0 1998-05-02      33.30   35.00000 20.662879  26.79861
6 chic   57 56.0 1998-05-07      32.10   34.50000 24.270422  33.99167
> summary(chic.f$pm25tmean2)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  30.05   32.12   35.04   36.63   39.53   61.50 
> 
```

arrange

- Used to reorder rows of a data frame according to one of the variables
- Reordering rows of a data frame is normally a pain to do in R.
- The arrange() function simplifies the process quite a bit.
- Columns can be arranged in descending order too by using the special *desc()* operator.

Output Screenshot

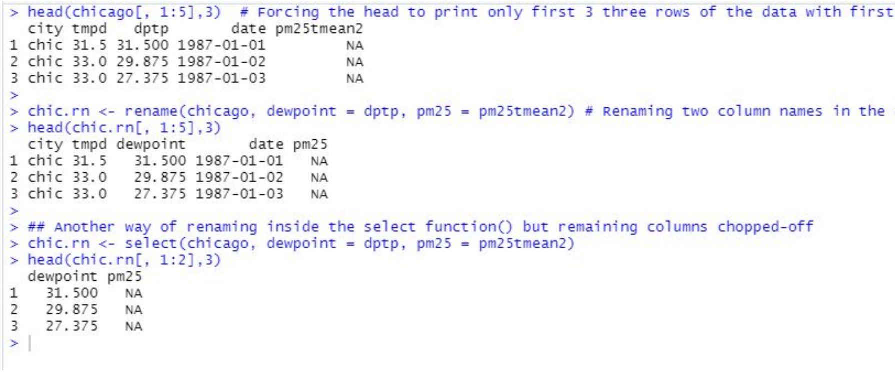


```
> ################## ARRANGE ####################################
> chic.wa <- chicago[order(chicago$date),] # without arrange function
> #chic.wa <- chicago[order(as.Date(chicago$date,format="%Y-%m-%d")),]
> head(chic.wa)
  city tmpd   dptp       date pm25tmean2 pm10tmean2 o3tmean2 no2tmean2
1 chic 31.5 31.500 1987-01-01         NA   34.00000 4.250000  19.98810
2 chic 33.0 29.875 1987-01-02         NA         NA 3.304348  23.19099
3 chic 33.0 27.375 1987-01-03         NA   34.16667 3.333333  23.81548
4 chic 29.0 28.625 1987-01-04         NA   47.00000 4.375000  30.43452
5 chic 32.0 28.875 1987-01-05         NA         NA 4.750000  30.33333
6 chic 40.0 35.125 1987-01-06         NA   48.00000 5.833333  25.77233
>
>
> chic.a <- arrange(chicago, date)
> head(chic.a)
  city tmpd   dptp       date pm25tmean2 pm10tmean2 o3tmean2 no2tmean2
1 chic 31.5 31.500 1987-01-01         NA   34.00000 4.250000  19.98810
2 chic 33.0 29.875 1987-01-02         NA         NA 3.304348  23.19099
3 chic 33.0 27.375 1987-01-03         NA   34.16667 3.333333  23.81548
4 chic 29.0 28.625 1987-01-04         NA   47.00000 4.375000  30.43452
5 chic 32.0 28.875 1987-01-05         NA         NA 4.750000  30.33333
6 chic 40.0 35.125 1987-01-06         NA   48.00000 5.833333  25.77233
> chic.a <- arrange(chicago, desc(date))
> head(chic.a)
  city tmpd dptp       date pm25tmean2 pm10tmean2  o3tmean2 no2tmean2
1 chic   35 30.1 2005-12-31   15.00000       23.5  2.531250  13.25000
2 chic   36 31.0 2005-12-30   15.05714       19.2  3.034420  22.80556
3 chic   35 29.4 2005-12-29    7.45000       23.5  6.794837  19.97222
4 chic   37 34.5 2005-12-28   17.75000       27.5  3.260417  19.28563
5 chic   40 33.6 2005-12-27   23.56000       27.0  4.468750  23.50000
6 chic   35 29.6 2005-12-26    8.40000        8.5 14.041667  16.81944
> |
```

rename

- Renaming a variable in a data frame in R is surprisingly hard to do.
- The rename() function is designed to make this process easier.
- The syntax inside the rename() function is to have the new name on the left-hand side of the = sign and the old name on the right-hand side.
- Another way - using select() function to rename
- **Exercise** - Rename a column without using dplyr package

Output Screenshot



```
> head(chicago[, 1:5],3)  # Forcing the head to print only first 3 three rows of the data with first
  city tmpd   dptp       date pm25tmean2
1 chic 31.5 31.500 1987-01-01         NA
2 chic 33.0 29.875 1987-01-02         NA
3 chic 33.0 27.375 1987-01-03         NA
>
> chic.rn <- rename(chicago, dewpoint = dptp, pm25 = pm25tmean2) # Renaming two column names in the
> head(chic.rn[, 1:5],3)
  city tmpd dewpoint       date pm25
1 chic 31.5   31.500 1987-01-01   NA
2 chic 33.0   29.875 1987-01-02   NA
3 chic 33.0   27.375 1987-01-03   NA
>
> ## Another way of renaming inside the select function() but remaining columns chopped-off
> chic.rn <- select(chicago, dewpoint = dptp, pm25 = pm25tmean2)
> head(chic.rn[, 1:2],3)
  dewpoint pm25
1   31.500   NA
2   29.875   NA
3   27.375   NA
> |
```

Pipe Operator

- The pipeline operator %>% is very handy for stringing together multiple dplyr functions in a sequence of operations.
- Every time we wanted to apply more than one function, the sequence gets buried in a sequence of nested function calls that is difficult to read third(second(first(x)))
- The %>% operator allows you to string operations in a left-to-right fashion
  first(x) %>% second %>% third

Agenda

dplyr package

ggplot2 package

caret package

Section 2

ggplot2 package

ggplot2

- ggplot2 is an R package for producing statistical, or data, graphics.
- Every ggplot2 plot has three key components:
  1. **data**
  2. A set of **aesthetic** mappings between variables in the data and visual properties
  3. At least one layer which describes how to render each observation. Layers are usually created with a **geom** function.
- data and aesthetic mappings are supplied in ggplot(), then layers are added on with +.

ggplot2(contd...)

- **faceting** - It creates tables of graphics by splitting the data into subsets and displaying the same graph for each subset.
- For a different geom function, you'd get a different type of plot.
- If you have a scatter plot with a lot of noise, it can be hard to see the dominant pattern. In this case it's useful to add a smoothed line to the plot with *geom_smooth()*

Agenda

dplyr package

ggplot2 package

caret package

Section 3

caret package

caret

- Caret is short for **C**lassification **A**nd **RE**gression **T**raining.
- Why caret? - With R having so many implementations of ML algorithms, it can be challenging to keep track of which algorithm resides in which package.
- No matter which package the algorithm resides, caret will remember that for you.

Preprocessing in caret

- **range:** Normalize values so it ranges between 0 and 1
- **center:** Subtract Mean
- **scale:** Divide by standard deviation
- **pca:** Replace with principal components
- **ica:** Replace with independent components

References

1. Roger D peng, ”R Programming for Data Science”, Lean Publishing, 2015
2. Hadley Wickham and Garrett Grolemund, ”R for Data Science”, O’Reilly
3. https://www.machinelearningplus.com/machine-learning/caret-package/

Thank You..