“What are some examples of using dplyr in data manipulation and how can I learn it through a tutorial?

Authored by

stats writer

June 24, 2024

RECOMMENDED CITATION

stats writer (2024). *“What are some examples of using dplyr in data manipulation and how can I learn it through a tutorial?*. PSYCHOLOGICAL SCALES. Retrieved from https://scales.arabpsychology.com/?p=150060

Dplyr is a powerful R package that simplifies the process of data manipulation and analysis. It offers a variety of functions that allow users to easily filter, arrange, group, and summarize data. Some common examples of using dplyr include merging multiple datasets, creating new variables, and performing calculations on specific subsets of data.

To learn dplyr, one option is to follow a tutorial. There are many resources available, such as online courses, video tutorials, and written guides, that provide step-by-step instructions on how to use dplyr for data manipulation. These tutorials typically cover the basics of dplyr and also provide examples of more advanced techniques for manipulating data. Through these tutorials, users can learn how to effectively use dplyr to streamline their data analysis process and gain valuable insights from their data.

In this R dplyr tutorial with examples, I will explain the R dplyr package, dplyr verbs, and how to use them with examples. All examples provided in this R dplyr tutorial are basic, simple, and easy to practice for beginners enthusiastic to learn R and advance their careers. I recommend reading the R Programming beginners tutorial with examples if you are new to R.

dplyr is like a toolbox for data wrangling. It gives you a bunch of handy tools (verbs) for changing and organizing your data, making it easier for data analysts to handle common tasks. All dplyr verbs take input as the data.frame and return the data.frame object.

When working with R data.frame, most of the R syntax takes $ to refer to the column name along with the data frame object (df$id) and uses notation, this syntax is not easy to read, and sometimes R code becomes confusing. Whereas R dplyr uses proper English verbs that are easily understandable by any programmer or analyst.

1. What is dplyr Package?

The `dplyr` package in R is a popular and powerful package for data manipulation and transformation. It provides a set of functions and tools that make data manipulation tasks more intuitive and efficient. The dplyr, `d` which stands for data.frame, `dplyr` can be read as pliers, which is referred to as a tool to manipulate the data frame.

Here are some advantages of using `dplyr`:

Readability:`dplyr` functions use a consistent and easy-to-read syntax, making your code more transparent and understandable. This can lead to more maintainable and error-free data manipulation code.**Efficiency:**`dplyr` is designed for performance. It uses optimized C++ code under the hood, which means it's often faster than equivalent base R code for data manipulation tasks. This is especially useful when working with large datasets.**Chaining:**`dplyr` supports method chaining, allowing you to apply multiple data manipulation operations in a sequence. This can

make your code more compact and easier to follow, as each step builds upon the previous one.

1.1 R dplyr Package Introduction

The `dplyr` is a R package that provides a grammar of data manipulation, and provides the most used verbs that help data science analysts to solve the most common data manipulation. Using methods from this package over R base function results in better performance of the operations.

To use dplyr verbs, you have to install it first using `install.packages('dplyr')` and load it using `library(dplyr)`. It provides the following methods and I will explain all these with examples.

R dplyr verbs	dplyr verb description
mutate()	Adds new variables
select()	Select variables.
filter()	Selects observations
arrange()	Ordering of the rows.
rename()	Rename variables name
slice()	Choose observations by position (location)
distinct()	Return distinct observation
rows_insert()	group_by() groups data. summarise() gives a summary.
inner_join(), left_join(), right_join(), full_join()	Join Operations
group_by() & summarise()	group_by() groups data. summarise() gives summary.

R dplyr verbs

Alternatively, by installing `tidyverse` package internally installs `dplyr` package.

1.2 Pipe Infix Operator %>%

All verbs in the dplyr package are taken `data.frame` as a first argument. When we use `dplyr` package, we mostly use the infix or pipe operator %>% in R from the magrittr, it passes the left-hand side of the operator to the first argument of the right-hand side of the operator. For example, x %>% f(y) converted into f(x, y) so the result from the left-hand side is then "piped" into the right-hand side. This pipe can be used to write multiple operations that you can read from left to right. For most of the examples in this R dplyr tutorial, I will be using this infix operator.

2. Install dplyr Package

To install the dplyr package, use install.packages() method. This method takes an argument as the package name you would like to install.

```
#Install just dplyr:
install.packages("dplyr")
```

# Alternatively, Install the entire tidyverse.
# tidyverse include dplyr.
install.packages("tidyverse")

3. Load dplyr Package

To use methods or verbs from `dplyr` package, first, you need to load the library using the R `library()`. Just input the package name in a string you want to load.

```
# Load dplyr library
library('dplyr')
```

4. dplyr Examples

We can start by creating an R data frame and then applying some dplyr functions to analyze the data. If you have existing data in a CSV format, you can import it into an R data frame. Additionally, you can find guidance on importing Excel files into R.

```
# Create DataFrame
df <- data.frame(
id = c(10,11,12,13,14,15,16,17),
name = c('sai','ram','deepika','sahithi','kumar','scott','Don','Lin'),
gender = c('M','M','F','F','M','M','M','F'),
dob = as.Date(c('1990-10-02','1981-3-24','1987-6-14','1985-8-16',
'1995-03-02','1991-6-21','1986-3-24','1990-8-26')),
state = c('CA','NY',NA,NA,'DC','DW','AZ','PH'),
row.names=c('r1','r2','r3','r4','r5','r6','r7','r8')
)
df
```

Yields below output.

```
# Output
id name gender dob state
r1 10 sai M 1990-10-02 CA
r2 11 ram M 1981-03-24 NY
r3 12 deepika F 1987-06-14 <NA>
r4 13 sahithi F 1985-08-16 <NA>
r5 14 kumar M 1995-03-02 DC
r6 15 scott M 1991-06-21 DW
r7 16 Don M 1986-03-24 AZ
r8 17 Lin F 1990-08-26 PH
```

4.1 dplyr::filter() Examples

By using the dplyr filter() function you can filter the R data frame rows by name, filter data frame by column value, by multiple conditions, etc. Here, `%>%` is an infix operator which acts as a pipe, it passes the left-hand side of the operator to the first argument of the right-hand side of the operator.

```
# Load dplyr library
library('dplyr')

# filter() by row name
df %>% filter(rownames(df) == 'r3')

# filter() by column Value
df %>% filter(gender == 'M')

# filter() by list of values
df %>% filter(state %in% c("CA", "AZ", "PH"))

# filter() by multiple conditions
df %>% filter(gender == 'M' & id > 15)
```

4.2 dplyr::select() Examples

dplyr select() function is used to select the columns or variables from the data frame. This takes the first argument as the data frame and the second argument is the variable name or vector of variable names. For more examples refer to select columns by name and select columns by index position.

```
# select() single column
df %>% select('id')
```

select() multiple columns
df %>% select(c('id','name'))

Select multiple columns by id
df %>% select(c(1,2))

4.3 dplyr::slice() Examples

slice() function is used to slice the data frame rows based on index position also, and it is used to drop rows based on an index. Following are some other slice verbs provided in the dplyr package.

Slice Verbs	Description
slice()	Slices the data.frame by row index
slice_head()	Select the first rows
slice_tail()	Select the last rows
slice_min()	Select the minimum of a column
slice_max()	Select the maximum of a column
slice_random()	Select random rows

Different slice functions from the dplyr package

Following are several examples of the usage of slice().

```
# Select rows 2 and 3
df %>% slice(2,3)
```

Select rows from list
df %>% slice(c(2,3,5,6))

select rows by range
df %>% slice(2:6)

Drop rows using slice()
df %>% slice(-2,-3,-4,-5,-6)

Drop by range

df %>% slice(-2:-6)

4.4 dplyr::mutate() Examples

Use the mutate() function and its other verbs mutate_all(), mutate_if(), and mutate_at() from the dplyr package to replace/update the values of the column (string, integer, or any type) in the R data frame (data.frame).

```
# Replace on selected column
df %>%
mutate(name = str_replace(name, "sai", "SaiRam"))
```

4.5 dplyr::rename() Examples

The rename() function of dplyr is used to change the column name present in the data frame. The first example from the following renames the column from the old name `id` to the new name `c1`. Similarly use dplyr to rename multiple columns.

```
#Change the column name - c1 to id
my_dataframe %>%
rename("c1" = "id")
```

# Rename multiple columns by name
my_dataframe <- my_dataframe %>% rename("c1" = "id",
"c2" = "pages",
"c3" = "name")

# Rename multiple columns by index
my_dataframe <- my_dataframe %>%
rename(col1 = 1, col2 = 2)

4.6 dplyr::distinct() Examples

distinct() function of dplyr is used to select the unique/distinct rows from the input data frame. Not using any column/variable names as arguments, this function returns unique rows by checking values on all columns.

```
# Create dataframe
```

```
df=data.frame(id=c(11,11,33,44,44),
pages=c(32,32,33,22,22),
name=c("spark","spark","R","java","jsp"),
chapters=c(76,76,11,15,15),
price=c(144,144,321,567,567))
df
```

# Load library dplyr
library(dplyr)

# Distinct rows
df2 <- df %>% distinct()
df2

# Distinct on selected columns
df2 <- df %>% distinct(id,pages)
df2

4.7 dplyr::arrange() Examples

dplyr arrange() function is used to sort the R data frame rows by ascending or descending order based on column values.

```
# Create data frame
df=data.frame(id=c(11,22,33,44,55),
name=c("spark","python","R","jsp","java"),
price=c(144,NA,321,567,567),
publish_date= as.Date(
c("2007-06-22", "2004-02-13", "2006-05-18",
"2010-09-02","2007-07-20"))
)
```

# Load dplyr library
library(dplyr)

# Using arrange in ascending order
df2 <- df %>% arrange(price)
df2

4.8 dplyr::group_by()

group_by() function in R is used to group the rows in a DataFrame by single or multiple columns and perform the aggregations.

```
# Create Data Frame
df = read.csv('/Users/admin/apps/github/r-examples/resources/emp.csv')
df
```

```
# Load dplyr
library(dplyr)

# group_by() on department
grp_tbl <- df %>% group_by(department)
grp_tbl

# summarise on groupped data.
agg_tbl <- grp_tbl %>% summarise(sum(salary))
agg_tbl
```

5. Conclusion

In this R `dplyr` tutorial, you have learned what is dplyr?, its usage, how to install, and load the library to use it in R programming, and finally explore different verbs with examples. Overall, the advantages of `dplyr` make it a valuable tool for data analysts and data scientists, allowing for efficient and readable data manipulation, which is often a critical component of the data analysis workflow.

Related Articles

References