RPandas Part 001

2023.03.22

Choosing a file

Choosing a file in Python with simple Dialog: easygui

```
# python code
#To install:
# pip install easygui
import easygui
#filename =easygui.fileopenbox()
#print(filename)
#easygui.egdemo()
```

Choosing a file in Python with simple Dialog: plyer

```
# python code
#To install:
# pip install plyer
#import plyer
#filename =plyer.filechooser.open_file()
#print(filename)
```

```
# R code
#filename =file.choose()
#print(filename)
```

read csv and xlsx files

csv

```
# python code
import pandas as pd
#import easygui
#filename = easygui.fileopenbox()
filename = "G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris.csv"
df1=pd.read_csv(filename)
df1
```

```
##      Sepal_Length  Sepal_Width  Petal_Length  Petal_Width    Species
## 0             5.1          3.5           1.4          0.2     setosa
## 1             4.9          3.0           1.4          0.2     setosa
## 2             4.7          3.2           1.3          0.2     setosa
## 3             4.6          3.1           1.5          0.2     setosa
## 4             5.0          3.6           1.4          0.2     setosa
## ..            ...          ...           ...          ...        ...
## 145           6.7          3.0           5.2          2.3  virginica
## 146           6.3          2.5           5.0          1.9  virginica
## 147           6.5          3.0           5.2          2.0  virginica
## 148           6.2          3.4           5.4          2.3  virginica
## 149           5.9          3.0           5.1          1.8  virginica
##
## [150 rows x 5 columns]
```

```
# python code
#df1.dtypes
#df1.head()
#df1.tail(2)
#df1.columns
#df1.describe()
#df1["Petal_Length"]
#df1[0:2]
df1[df1.Petal_Length==1.4]
```

```
##     Sepal_Length  Sepal_Width  Petal_Length  Petal_Width Species
## 0            5.1          3.5           1.4          0.2  setosa
## 1            4.9          3.0           1.4          0.2  setosa
## 4            5.0          3.6           1.4          0.2  setosa
## 6            4.6          3.4           1.4          0.3  setosa
## 8            4.4          2.9           1.4          0.2  setosa
## 12           4.8          3.0           1.4          0.1  setosa
## 17           5.1          3.5           1.4          0.3  setosa
## 28           5.2          3.4           1.4          0.2  setosa
## 33           5.5          4.2           1.4          0.2  setosa
## 37           4.9          3.6           1.4          0.1  setosa
## 45           4.8          3.0           1.4          0.3  setosa
## 47           4.6          3.2           1.4          0.2  setosa
## 49           5.0          3.3           1.4          0.2  setosa
```

```
# R code
#filename = file.choose()
filename = "G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris.csv"
df1=read.csv(filename)
head(df1)
```

```
##   Sepal_Length Sepal_Width Petal_Length Petal_Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

xlsx

```
#R code
library(openxlsx)
filename = "G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris.xlsx"
df1=openxlsx::read.xlsx(filename)
head(df1)
```

R Code

```
##   Sepal_Length Sepal_Width Petal_Length Petal_Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
#R code
#read all sheets
library(openxlsx)
filename = "G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris.xlsx"
SheetNames <- openxlsx::getSheetNames(filename)
SheetNames
```

```
## [1] "iris"   "Sheet1"
```

```
SheetList <- lapply(SheetNames,openxlsx::read.xlsx,xlsxFile=filename)
names(SheetList) <- SheetNames
SheetList$Sheet1[1:4,]
```

```
##   sheet2 Sepal_Width Petal_Length Petal_Width Species
## 1 sheet2         3.5          1.4         0.2  setosa
## 2 sheet2         3.0          1.4         0.2  setosa
## 3 sheet2         3.2          1.3         0.2  setosa
## 4 sheet2         3.1          1.5         0.2  setosa
```

```
SheetList$iris[1:4,]
```

```
##   Sepal_Length Sepal_Width Petal_Length Petal_Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
```

```
# write xlsx files
library(openxlsx)
wb <- createWorkbook()  #wb <- loadWorkbook("RawExcel.xlsx")
addWorksheet(wb, sheetName = "sheetname1")
```

```
writeData(wb, sheet = "sheetname1", x = SheetList$iris[1:4,])
addWorksheet(wb, sheetName = "sheetname2")
writeData(wb, sheet = "sheetname2", x = SheetList$Sheet1[1:4,])
#saveWorkbook(wb, "G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris2.xlsx")
```

Python Code Python Code

```
import pandas as pd
xls = pd.ExcelFile('G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris2.xlsx')
xls.sheet_names
```

```
## ['sheetname1', 'sheetname2']
```

```
df1 = pd.read_excel(xls, xls.sheet_names[0])
df2 = pd.read_excel(xls, xls.sheet_names[1])
df1
```

```
##    Sepal_Length  Sepal_Width  Petal_Length  Petal_Width Species
## 0           5.1          3.5           1.4          0.2  setosa
## 1           4.9          3.0           1.4          0.2  setosa
## 2           4.7          3.2           1.3          0.2  setosa
## 3           4.6          3.1           1.5          0.2  setosa
```

```
import pandas as pd
xls = pd.ExcelFile('G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris2.xlsx')
xls.sheet_names
```

```
## ['sheetname1', 'sheetname2']
```

```
df1 = pd.read_excel(xls, xls.sheet_names[0])
df2 = pd.read_excel(xls, xls.sheet_names[1])
df1
```

```
##    Sepal_Length  Sepal_Width  Petal_Length  Petal_Width Species
## 0           5.1          3.5           1.4          0.2  setosa
## 1           4.9          3.0           1.4          0.2  setosa
## 2           4.7          3.2           1.3          0.2  setosa
## 3           4.6          3.1           1.5          0.2  setosa
```

```
dff=[pd.read_excel(xls, x) for x in xls.sheet_names]
```

```
import pandas as pd
dict_temp = pd.read_excel('G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris2.xlsx', sheet_name
dict_temp['sheetname1']
```

```
##    Sepal_Length  Sepal_Width  Petal_Length  Petal_Width Species
## 0           5.1          3.5           1.4          0.2  setosa
## 1           4.9          3.0           1.4          0.2  setosa
## 2           4.7          3.2           1.3          0.2  setosa
## 3           4.6          3.1           1.5          0.2  setosa
```

```
dict_temp['sheetname2']
```

```
##    sheet2  Sepal_Width  Petal_Length  Petal_Width Species
## 0  sheet2          3.5           1.4          0.2  setosa
## 1  sheet2          3.0           1.4          0.2  setosa
## 2  sheet2          3.2           1.3          0.2  setosa
## 3  sheet2          3.1           1.5          0.2  setosa
```

filter and select

filter

```
# python code
pl=1.4
qs="Petal_Length==@pl"
df1.query(qs)
```

```
##    Sepal_Length  Sepal_Width  Petal_Length  Petal_Width Species
## 0           5.1          3.5           1.4          0.2  setosa
## 1           4.9          3.0           1.4          0.2  setosa
```

```
# python code
pw=.3
sp=["setosa","setosa1"]
qs="Species in @sp"\
 " and Petal_Width <= @pw"
df1.query(qs)
```

```
##    Sepal_Length  Sepal_Width  Petal_Length  Petal_Width Species
## 0           5.1          3.5           1.4          0.2  setosa
## 1           4.9          3.0           1.4          0.2  setosa
## 2           4.7          3.2           1.3          0.2  setosa
## 3           4.6          3.1           1.5          0.2  setosa
```

```
# R code
library(dplyr)
```

```
##
## 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
```

```
filename = "G:\\Python tutorial\\pythontutorial\\pythontutorial\\iris.csv"
df1=read.csv(filename)
pw=.3
sp=c("setosa","setosa1")
df1 %>%
  dplyr::filter(
    Species %in% sp
    ,Petal_Width <= pw
  ) %>% head()
```

```
##   Sepal_Length Sepal_Width Petal_Length Petal_Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          4.6         3.4          1.4         0.3  setosa
```

select

```
# python code
cl=["Sepal_Length","Petal_Width"]
df1[cl]
```

```
##    Sepal_Length  Petal_Width
## 0           5.1          0.2
## 1           4.9          0.2
## 2           4.7          0.2
## 3           4.6          0.2
```

```
# R code
library(dplyr)
cl=c("Sepal_Length","Petal_Width")
df1%>% dplyr::select(all_of(cl))  %>% head()
```

```
##   Sepal_Length Petal_Width
## 1          5.1         0.2
## 2          4.9         0.2
## 3          4.7         0.2
## 4          4.6         0.2
## 5          5.0         0.2
## 6          5.4         0.4
```

```
# df1%>% dplyr::select(Sepal_Length,Petal_Width)
```

pivot/melt

melt - pivot_longer

```
# python code
import pandas as pd
df2 = pd.DataFrame({'A': {0: 'a', 1: 'b', 2: 'c'},
                   'B': {0: 1, 1: 3, 2: 5},
                   'C': {0: 2, 1: 4, 2: 6}})
df2.melt(id_vars='A')
```

```
##    A variable  value
## 0  a        B      1
## 1  b        B      3
## 2  c        B      5
## 3  a        C      2
## 4  b        C      4
## 5  c        C      6
```

```
df2.melt(id_vars='A', value_vars=['B','C'], var_name='BC', value_name='value')
```

```
##    A BC  value
## 0  a  B      1
## 1  b  B      3
## 2  c  B      5
## 3  a  C      2
## 4  b  C      4
## 5  c  C      6
```

```
# R code
library(dplyr)
library(tidyr)
df2 = data.frame(
  A=c('a','b','c')
  ,B=c(1,3,5)
  ,C=c(2,4,6)
  )
df2 %>%
  pivot_longer(B:C,names_to = 'BC',values_to = 'value')  %>% head()
```

```
## # A tibble: 6 x 3
##   A     BC    value
##   <chr> <chr> <dbl>
## 1 a     B         1
## 2 a     C         2
## 3 b     B         3
## 4 b     C         4
## 5 c     B         5
## 6 c     C         6
```

```
# R code
library(dplyr)
library(tidyr)
df2 = data.frame(
  A=c('a', 'a','b', 'b', 'c','c')
  ,B=c('A', 'B','A', 'B', 'A','B')
  ,D=c( 1, 3, 5,7,9,11)
  ,E=c(2, 4, 6,8,10,12)
  )
df2 %>%
  tidyr::pivot_longer(cols = any_of(c('D','E')),names_to = "DE",values_to  = "value")  %>% head()
```

```
## # A tibble: 6 x 4
##   A     B     DE    value
##   <chr> <chr> <chr> <dbl>
## 1 a     A     D         1
## 2 a     A     E         2
## 3 a     B     D         3
## 4 a     B     E         4
## 5 b     A     D         5
## 6 b     A     E         6
```

pivot_wider

```
# python code
import pandas as pd
df2 = pd.DataFrame({'A': {0: 'a', 1: 'b', 2: 'c'},
                   'B': {0: 1, 1: 3, 2: 5},
                   'C': {0: 2, 1: 4, 2: 6}})
#print(df2)
df2_melt=df2.melt(id_vars='A', value_vars=['B','C'], var_name='BC', value_name='value')
#print(df2_melt)
df_pivot=df2_melt.pivot(index='A', columns=['BC'])#, values='value')
df2_r = df_pivot.reset_index(None)
df2_r.columns = ['A', 'B', 'C']
print(df2_r)
```

```
##    A  B  C
## 0  a  1  2
## 1  b  3  4
## 2  c  5  6
```

```
df2_r.columns=df2.columns.values
print(df2.columns.values)
```

```
## ['A' 'B' 'C']
```

```
print(df2_r)
```

```
##    A  B  C
## 0  a  1  2
## 1  b  3  4
## 2  c  5  6
```

```r
# R code
library(dplyr)
library(tidyr)
df2 = data.frame(
  A=c('a','b','c')
  ,B=c(1,3,5)
  ,C=c(2,4,6)
)
df2_melt<-df2 %>%
  tidyr::pivot_longer(cols = any_of(c('B','C')),names_to = "BC",values_to  = "value")
df_pivot <- df2_melt %>%
  tidyr::pivot_wider(id_cols = A, names_from = BC,values_from = value )
df_pivot  %>% head()
```

```
## # A tibble: 3 x 3
##   A         B     C
##   <chr> <dbl> <dbl>
## 1 a         1     2
## 2 b         3     4
## 3 c         5     6
```

dplython

```python
# python code
#To install:
# pip install dplython
```

1

siuba

```python
# python code
#To install:
# pip install siuba
```

1

The across function

across

- across() makes it easy to apply the same transformation to multiple columns, allowing you to use select() semantics inside in "data-masking" functions like summarise() and mutate().
- if_any() and if_all() are used to apply the same predicate function to a selection of columns and combine the results into a single logical vector.
- across() supersedes the family of dplyr "scoped variants" like summarise_at(), summarise_if(), and summarise_all() and therefore these functions will not be implemented in poorman. across: Apply a function (or functions) across multiple columns

Usage

- across(.cols = everything(), .fns = NULL, . . . , .names = NULL)
- if_any(.cols, .fns = NULL, . . . , .names = NULL)
- if_all(.cols, .fns = NULL, . . . , .names = NULL)

Arguments

.fns Functions to apply to each of the selected columns. Possible values are:

- NULL, to returns the columns untransformed.
- A function, e.g. mean.
- A lambda, e.g. ~ mean(.x, na.rm = TRUE)
- A list of functions/lambdas, e.g. list(mean = mean, n_miss = ~ sum(is.na(.x))

Within these functions you can use cur_column() and cur_group() to access the current column and grouping keys respectively.

. . . Additional arguments for the function calls in .fns.

.names character(n). Currently limited to specifying a vector of names to use for the outputs.

cols, .cols Columns to transform. Because across() is used within functions like summarise() and mutate(), you can't select or compute upon grouping variables.

Value

- across() returns a data.frame with one column for each column in .cols and each function in .fns.
- if_any() and if_all() return a logical vector.

How to use across

There are four columns and I want to quickly get the mean of these columns for each category. First, here's how I might do this without across:

```r
# R code
iris  %>%
  group_by(Species) %>%
  summarise(
    Sepal.Length = mean(Sepal.Length, na.rm = TRUE),
    Sepal.Width = mean(Sepal.Width, na.rm = TRUE),
    Petal.Width = mean(Petal.Width, na.rm = TRUE),
    Petal.Length = mean(Petal.Length, na.rm = TRUE)
    )  %>% head()
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Width Petal.Length
##   <fct>             <dbl>       <dbl>       <dbl>        <dbl>
## 1 setosa             5.01        3.43       0.246         1.46
## 2 versicolor         5.94        2.77       1.33          4.26
## 3 virginica          6.59        2.97       2.03          5.55
```

Which works fine. But imagine if instead of four columns there were 10 or 20 or 100! It would quickly get tedious to add a new line for each column. Here's where across comes in:

```
# R code
iris  %>%
  group_by(Species) %>%

  summarise(across(c(Sepal.Length, Sepal.Width, Petal.Length,Petal.Width),mean,na.rm = TRUE)) %>% head(
```

```
## Warning: There was 1 warning in `summarise()`.
## i In argument: `across(...)`.
## i In group 1: `Species = setosa`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function instead.
##
##   # Previously
##   across(a:b, mean, na.rm = TRUE)
##
##   # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

Much more efficient. We give across a vector of column names followed by the function (in this case mean) followed by any other arguments we want to apply to the function.

:

: for selecting a range of consecutive variables.

```
# R code
iris  %>%
  group_by(Species) %>%
       summarise(across(c(Sepal.Length:Petal.Width), mean, na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

!

! for taking the complement of a set of variables.

```
# R code
iris  %>%
  group_by(Species) %>%
      summarise(across(!c(Petal.Width), mean, na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 4
##   Species    Sepal.Length Sepal.Width Petal.Length
##   <fct>             <dbl>       <dbl>        <dbl>
## 1 setosa             5.01        3.43         1.46
## 2 versicolor         5.94        2.77         4.26
## 3 virginica          6.59        2.97         5.55
```

& and |

& and |for selecting the intersection or the union of two sets of variables.

```
# R code
iris  %>%
  group_by(Species) %>%
      summarise(across(ends_with('Length') & !c(Petal.Length, Petal.Width), mean, na.rm = TRUE))  %>% h
```

```
## # A tibble: 3 x 2
##   Species    Sepal.Length
##   <fct>             <dbl>
## 1 setosa             5.01
## 2 versicolor         5.94
## 3 virginica          6.59
```

c()

c() for combining selections.

```
# R code
iris  %>%
  group_by(Species) %>%

  summarise(across(c(Sepal.Length, Sepal.Width, Petal.Length,Petal.Width),mean,na.rm = TRUE)) %>% head(
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

starts_with()

starts_with(): Starts with a prefix.

```
# R code
iris  %>%
  group_by(Species) %>%

  summarise(across(starts_with("S"),mean,na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 3
##   Species    Sepal.Length Sepal.Width
##   <fct>             <dbl>       <dbl>
## 1 setosa             5.01        3.43
## 2 versicolor         5.94        2.77
## 3 virginica          6.59        2.97
```

ends_with()

ends_with(): Ends with a suffix.

```
# R code
iris  %>%
  group_by(Species) %>%

  summarise(across(ends_with("dth"),mean,na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 3
##   Species    Sepal.Width Petal.Width
##   <fct>            <dbl>       <dbl>
## 1 setosa            3.43       0.246
## 2 versicolor        2.77       1.33
## 3 virginica         2.97       2.03
```

contains()

contains(): Contains a literal string.

```
# R code
iris  %>%
  group_by(Species) %>%
    summarise(across(contains('Length'),mean,na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 3
##   Species    Sepal.Length Petal.Length
##   <fct>             <dbl>        <dbl>
## 1 setosa             5.01         1.46
## 2 versicolor         5.94         4.26
## 3 virginica          6.59         5.55
```

matches()

matches(): Matches a regular expression.

```
# R code
iris  %>%
  group_by(Species) %>%
    summarise(across(matches('^(S|P)'),mean,na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

num_range()

num_range(): Matches a numerical range like x01, x02, x03.

```
# R code
df <- as.data.frame(matrix(1:24, nrow = 3))
df  %>% head()
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8
## 1  1  4  7 10 13 16 19 22
## 2  2  5  8 11 14 17 20 23
## 3  3  6  9 12 15 18 21 24
```

```
df %>% select(num_range("V", seq(1, 1000, by = 3)))  %>% head()
```

```
##   V1 V4 V7
## 1  1 10 19
## 2  2 11 20
## 3  3 12 21
```

```
# R code
df <- data.frame(id=c("a","a","b"), tot_1=4:6, tot_2=8:10, tot_3=11:13, tot_4=33:35,tot_5=22:24)
df  %>% head()
```

```
##   id tot_1 tot_2 tot_3 tot_4 tot_5
## 1  a     4     8    11    33    22
## 2  a     5     9    12    34    23
## 3  b     6    10    13    35    24
```

```
df %>% group_by(id) %>%
  mutate(across(.cols = num_range("tot_", seq(1, 5, by = 2)),mean,na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 6
## # Groups:   id [2]
##   id    tot_1 tot_2 tot_3 tot_4 tot_5
##   <chr> <dbl> <int> <dbl> <int> <dbl>
## 1 a       4.5     8  11.5    33  22.5
## 2 a       4.5     9  11.5    34  22.5
## 3 b       6      10  13      35  24
```

```
# R code
df %>% group_by(id) %>%
  summarise(across(.cols = num_range(prefix="tot_", range=seq(1, 5, by = 2)),mean,na.rm = TRUE))  %>% he
```

```
## # A tibble: 2 x 4
##   id    tot_1 tot_3 tot_5
##   <chr> <dbl> <dbl> <dbl>
## 1 a       4.5  11.5  22.5
## 2 b       6    13    24
```

all_of()

all_of(): Matches variable names in a character vector. All names must be present, otherwise an out-of-bounds error is thrown.

```
# R code
iris  %>%
  group_by(Species) %>%
    summarise(across(all_of(c('Sepal.Length','Sepal.Width','Petal.Length')),mean,na.rm = TRUE))  %>% hea
```

```
## # A tibble: 3 x 4
##   Species    Sepal.Length Sepal.Width Petal.Length
##   <fct>             <dbl>       <dbl>        <dbl>
## 1 setosa             5.01        3.43         1.46
## 2 versicolor         5.94        2.77         4.26
## 3 virginica          6.59        2.97         5.55
```

any_of()

any_of(): Same as all_of(), except that no error is thrown for names that don’t exist.

```
# R code
iris  %>%
  group_by(Species) %>%
    summarise(across(any_of(c('Sepal.Length','Sepal.Width','Petal.Length','Not_valid_name')),mean,na.rm
```

```
## # A tibble: 3 x 4
##   Species    Sepal.Length Sepal.Width Petal.Length
##   <fct>             <dbl>       <dbl>        <dbl>
## 1 setosa             5.01        3.43         1.46
## 2 versicolor         5.94        2.77         4.26
## 3 virginica          6.59        2.97         5.55
```

where()

where(): Applies a function to all variables and selects those for which the function returns TRUE.

```
# R code
iris  %>%
  group_by(Species) %>%
      summarise(across(where(is.numeric), mean, na.rm = TRUE))  %>% head()
```

```
## # A tibble: 3 x 5
##   Species    Sepal.Length Sepal.Width Petal.Length Petal.Width
##   <fct>             <dbl>       <dbl>        <dbl>       <dbl>
## 1 setosa             5.01        3.43         1.46       0.246
## 2 versicolor         5.94        2.77         4.26       1.33
## 3 virginica          6.59        2.97         5.55       2.03
```

Using in-line functions with across

Let's look at an example of summarizing the columns using a custom function (rather than n_distinct()). I usually do this using the tilde-dot shorthand for inline functions. The notation works by replacing

```
# R code
function(x) {
  x + 10
}
```

```
## function(x) {
##   x + 10
## }
```

with

```
# R code
~{.x + 10}
```

```
## ~{
##     .x + 10
## }
```

~ indicates that you have started an anonymous function, and the argument of the anonymous function can be referred to using .x (or simply .). Unlike normal function arguments that can be anything that you like, the tilde-dot function argument is always .x.

For instance, to identify how many missing values there are in every column, we could specify the inline function ~sum(is.na(.)), which calculates how many NA values are in each column (where the column is represented by .) and adds them up:

```
# R code
dat<-data.frame(a=c(1,2,3,NA,NA,6),b=1:6,d=c(NA,2:6))
dat
```

```
##     a b  d
## 1   1 1 NA
## 2   2 2  2
## 3   3 3  3
## 4  NA 4  4
## 5  NA 5  5
## 6   6 6  6
```

```
dat %>%
  summarise(across(everything(),  ~sum(is.na(.))))  %>% head()
```

```
##   a b d
## 1 2 0 1
```

```
# R code
dat<-data.frame(a=c(1:4),b=c(1:4)^2,d=c(1:4)^3)
dat  %>% head()
```

```
##   a  b  d
## 1 1  1  1
## 2 2  4  8
## 3 3  9 27
## 4 4 16 64
```

```
dat %>%
  summarise(across(everything(),  ~ .x +10))  %>% head()
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
##    a  b  d
## 1 11 11 11
## 2 12 14 18
## 3 13 19 37
## 4 14 26 74
```

Contact us

Contact me at masoudfaridi@modares.ac.ir or masoud1faridi@gmail.com