R Example DPLYR Generate Sorted Index, Ordinal Deviation Negative and Positive Index, and Expand Value from Lowest Index to All Rows

Fan Wang

Go back to fan's REconTools Package, R4Econ Repository, or Intro Stats with R Repository.

```
rm(list = ls(all.names = TRUE))
options(knitr.duplicate.label = 'allow')
```

```
library(tidyverse)
library(knitr)
library(kableExtra)
library(REconTools)
# file name
st_file_name = 'fs_index_populate'
# Generate R File
try(purl(paste0(st_file_name, ".Rmd"), output=paste0(st_file_name, ".R"), documentation = 2))
# Generate PDF and HTML
# rmarkdown::render("C:/Users/fan/R4Econ/summarize/index/fs_index_populate.Rmd", "pdf_document")
# rmarkdown::render("C:/Users/fan/R4Econ/summarize/index/fs_index_populate.Rmd", "html_document")
```

Generate Sorted Index within Group and Spread

Generate Sorted Index within Group with Repeating Values

There is a variable, sort by this variable, then generate index from 1 to N representing sorted values of this index. If there are repeating values, still assign index, different index each value.

- r generate index sort
- dplyr mutate equals index

```
# Sort and generate variable equal to sorted index
df_iris <- iris %>% arrange(Sepal.Length) %>%
              mutate(Sepal.Len.Index = row_number()) %>%
              select(Sepal.Length, Sepal.Len.Index, everything())

# Show results Head 10
df_iris %>% head(10) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

Sepal.Length

Sepal.Len.Index

Sepal.Width

Petal.Length

Petal.Width

Species

4.3

1

3.0

1.1

0.1

setosa

4.4

2

2.9

1.4

0.2

setosa

4.4

3

3.0

1.3

0.2

setosa

4.4

4

3.2

1.3

0.2

setosa

4.5

5

2.3

1.3

0.3

setosa

4.6

6

3.1

1.5

0.2

setosa

4.6

7

3.4

1.4

0.3

setosa

4.6

8

3.6

1.0

0.2

setosa

4.6

9

3.2

1.4

0.2

setosa

4.7

10

3.2

1.3

0.2

setosa

Populate Value from Lowest Index to All other Rows

We would like to calculate for example the ratio of each individual's highest to the the person with the lowest height in a dataset. We first need to generated sorted index from lowest to highest, and then populate the lowest height to all rows, and then divide.

Search Terms:

- r spread value to all rows from one row
- r other rows equal to the value of one row
- Conditional assignment of one variable to the value of one of two other variables
- dplyr mutate conditional
- dplyr value from one row to all rows
- dplyr mutate equal to value in another cell

Links:

- see: dplyr rank
- see: dplyr case_when

Short Method: mutate and min We just want the lowest value to be in its own column, so that we can compute various statistics using the lowest value variable and the original variable.

```
# 1. Sort
df_iris_m1 <- iris %>% mutate(Sepal.Len.Lowest.all = min(Sepal.Length)) %>%
                select(Sepal.Length, Sepal.Len.Lowest.all, everything())


# Show results Head 10
df_iris_m1 %>% head(10) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

Sepal.Length

Sepal.Len.Lowest.all

Sepal.Width

Petal.Length

Petal.Width

Species

5.1

4.3

3.5

1.4

0.2

setosa

4.9

4.3

3.0

1.4

0.2

setosa

4.7

4.3

3.2

1.3

0.2

setosa

4.6

4.3

3.1

1.5

0.2

setosa

5.0

4.3

3.6

1.4

0.2

setosa

5.4

4.3

3.9

1.7

0.4

setosa

4.6

4.3

3.4

1.4

0.3

setosa

5.0

4.3

3.4

1.5

0.2

setosa

4.4

4.3

2.9

1.4

0.2

setosa

4.9

4.3

3.1

1.5

0.1

setosa

Long Method: row_number and case_when This is the long method, using row_number, and case_when. The benefit of this method is that it generates several intermediate variables that might be useful. And the key final step is to set a new variable (A=*Sepal.Len.Lowest.all*) equal to another variable's (B=*Sepal.Length*'s) value at the index that satisfies condition based a third variable (C=*Sepal.Len.Index*).

```
# 1. Sort
# 2. generate index
# 3. value at lowest index (case_when)
# 4. spread value from lowest index to other rows
# Note step 4 does not require step 3
df_iris_m2 <- iris %>% arrange(Sepal.Length) %>%
              mutate(Sepal.Len.Index = row_number()) %>%
              mutate(Sepal.Len.Lowest.one =
                       case_when(row_number()==1 ~ Sepal.Length)) %>%
              mutate(Sepal.Len.Lowest.all =
                       Sepal.Length[Sepal.Len.Index==1]) %>%
              select(Sepal.Length, Sepal.Len.Index,
                     Sepal.Len.Lowest.one, Sepal.Len.Lowest.all)


# Show results Head 10
df_iris_m2 %>% head(10) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

Sepal.Length

Sepal.Len.Index

Sepal.Len.Lowest.one

Sepal.Len.Lowest.all

4.3

1

4.3

4.3

4.4

2

NA

4.3

4.4

3

NA

4.3

4.4

4

NA

4.3

4.5

5

NA

4.3

4.6

6

NA

4.3

4.6

7

NA

4.3

4.6

8

NA

4.3

4.6

9

NA

4.3

4.7

10

NA

4.3

Generate Sorted Index based on Deviations

Generate Positive and Negative Index based on Ordered Deviation from some Number

There is a variable that is continuous, substract a number from this variable, and generate index based on deviations. Think of the index as generating intervals indicating where the value lies. 0th index indicates the largest value in sequence that is smaller than or equal to number x, 1st index indicates the smallest value in sequence that is larger than number x.

The solution below is a little bit convoluated and long, there is likely a much quicker way. The process below shows various intermediary outputs that help arrive at deviation index *Sepal.Len.Devi.Index* from initial sorted index *Sepal.Len.Index*.

search:

- dplyr arrange ignore na
- dplyr index deviation from order number sequence
- dplyr index below above
- dplyr index order below above value

```
# 1. Sort and generate variable equal to sorted index
# 2. Plus or minus deviations from some value
# 3. Find the zero, which means, the number closests to zero including zero from the negative side
# 4. Find the index at the highest zero and below deviation point
# 5. Difference of zero index and original sorted index
sc_val_x <- 4.65
df_iris_deviate <- iris %>% arrange(Sepal.Length) %>%
               mutate(Sepal.Len.Index = row_number()) %>%
               mutate(Sepal.Len.Devi = (Sepal.Length - sc_val_x)) %>%
               mutate(Sepal.Len.Devi.Neg =
                        case_when(Sepal.Len.Devi <= 0 ~ (-1)*(Sepal.Len.Devi))) %>%
               arrange((Sepal.Len.Devi.Neg), desc(Sepal.Len.Index)) %>%
               mutate(Sepal.Len.Index.Zero =
                        case_when(row_number() == 1 ~ Sepal.Len.Index)) %>%
               mutate(Sepal.Len.Devi.Index =
                        Sepal.Len.Index - Sepal.Len.Index.Zero[row_number() == 1]) %>%
               arrange(Sepal.Len.Index) %>%
               select(Sepal.Length, Sepal.Len.Index, Sepal.Len.Devi,
                      Sepal.Len.Devi.Neg, Sepal.Len.Index.Zero, Sepal.Len.Devi.Index)


# Show results Head 10
df_iris_deviate %>% head(20) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "responsive"))
```

Sepal.Length

Sepal.Len.Index

Sepal.Len.Devi

Sepal.Len.Devi.Neg

Sepal.Len.Index.Zero

Sepal.Len.Devi.Index

4.3

1

-0.35

0.35

NA

-8

4.4

2

-0.25

0.25
NA
-7
4.4
3
-0.25
0.25
NA
-6
4.4
4
-0.25
0.25
NA
-5
4.5
5
-0.15
0.15
NA
-4
4.6
6
-0.05
0.05
NA
-3
4.6
7
-0.05
0.05
NA
-2
4.6
8
-0.05

0.05
NA
-1
4.6
9
-0.05
0.05
9
0
4.7
10
0.05
NA
NA
1
4.7
11
0.05
NA
NA
2
4.8
12
0.15
NA
NA
3
4.8
13
0.15
NA
NA
4
4.8
14
0.15

NA
NA
5
4.8
15
0.15
NA
NA
6
4.8
16
0.15
NA
NA
7
4.9
17
0.25
NA
NA
8
4.9
18
0.25
NA
NA
9
4.9
19
0.25
NA
NA
10
4.9
20
0.25

NA

NA

11