Reshaping Data

1405 Instructor: Ruiqing (Sam) Cao

Reshaping Data (Pivot & Melt)

Date	Type	Sales
2023Q1	A	100
2023Q2	A	200
2023Q1	B	150
2023Q2	B	250
2023Q1	C	125
2023Q2	C	225

Date	Sales		
	A	B	C
2023Q1	100	150	125
2023Q2	200	250	225

Pivot (Long to Wide)

`df=`

Date	Type	Sales
2023Q1	A	100
2023Q2	A	200
2023Q1	B	150
2023Q2	B	250
2023Q1	C	125
2023Q2	C	225

`df.pivot(index, columns, values)=`

Date	Sales		
	A	B	C
2023Q1	100	150	125
2023Q2	200	250	225

`index=Date columns=Type values=Sales`

Melt (Wide to Long)


```
df.melt(id_vars,value_name,var_name,value_vars)=
```

df=

Date	A	B	C
2023Q1	100	150	125
2023Q2	200	250	225

```
id_vars=Date value_name=Sales
var_name=Type value_vars=[A,B,C]
```

Date	Type	Sales
2023Q1	A	100
2023Q2	A	200
2023Q1	B	150
2023Q2	B	250
2023Q1	C	125
2023Q2	C	225

Pivot & Melt: Summary of Parameters

- Parameters for `pivot` (`index, columns, values`) and `melt` (`id_vars, value_name, var_name, value_vars`)

pivot (input)	pivot (output)	melt (input)	melt (output)	Example
`index`		`id_vars`		`['Date']`
`columns`			`var_name`	`['Type']`
		`value_vars`		`['A','B','C']`
`values`			`value_name`	`Sales`

Pivot (Long to Wide): Arguments

- Basic syntax: `DataFrame.pivot(index,columns,values)`
  - `index`: primary key of the pivoted table
  - `columns`: column(s) to pivot on, whose unique values will be new column names in the pivoted table
  - `values`: column(s) whose values populate the cells of the pivoted table

➢ `index`, `columns`, and `values` can be either a single column (one variable) or multiple columns (a list of variables)

Pivot (Long to Wide): an Example

```
data.pivot(index=['year','month'],columns=['is_group'],values='count')
```

	year	month	is_group	count
0	2011	02	False	1
1	2011	11	False	1
2	2012	01	False	1
3	2012	02	False	2
4	2012	04	False	1
...	...	...	...	...
91	2018	05	False	393
92	2018	05	True	1
93	2018	06	False	430
94	2018	07	False	558
95	2018	08	False	1

96 rows × 4 columns

	is_group	False	True
year	month		
2011	02	1.000	NaN
	11	1.000	NaN
2012	01	1.000	NaN
	02	2.000	NaN
	04	1.000	NaN
...	...	...	...
2018	04	355.000	1.000
	05	393.000	1.000
	06	430.000	NaN
	07	558.000	NaN
	08	1.000	NaN

81 rows × 2 columns

Melt (Wide to Long): Arguments

```
DataFrame.melt(id_vars,value_name,var_name,value_vars)
```

- `id_vars`: primary key of the original table
- `value_name`: the name of the value column in the melted table (defaults to 'value' if not specified)
- `var_name`: name of the column to store category labels (i.e., column names given in `value_vars`) in the melted table (defaults to 'variable' if not specified)
- `value_vars`: columns to melt in the original table, which become category labels stored in `var_name` in the melted table

- `value_name` & `var_name` must be atomic, `value_vars` must be a list, and `id_vars` can be either atomic or a list
- `value_vars` are the unique values of `var_name` in the melted table

Melt (Wide to Long): an Example

```
pivoted.melt(id_vars=['year','month'],value_name='count',var_name='grp_yn',value_vars=[False,True])
```

is_group	year	month	False	True
0	2011	02	1.000	NaN
1	2011	11	1.000	NaN
2	2012	01	1.000	NaN
3	2012	02	2.000	NaN
4	2012	04	1.000	NaN
...	...	...	...	...
76	2018	04	355.000	1.000
77	2018	05	393.000	1.000
78	2018	06	430.000	NaN
79	2018	07	558.000	NaN
80	2018	08	1.000	NaN

	year	month	grp_yn	count
0	2011	02	False	1.000
1	2011	11	False	1.000
2	2012	01	False	1.000
3	2012	02	False	2.000
4	2012	04	False	1.000
...	...	...	...	...
157	2018	04	True	1.000
158	2018	05	True	1.000
159	2018	06	True	NaN
160	2018	07	True	NaN
161	2018	08	True	NaN

162 rows × 4 columns

Exercise: Pivoting & Melting Data

Run the provided code and obtain the DataFrames **data** and **pivoted**

1. Pivot data using (`year`, `is_group`) as the primary key, `month` as the column to pivot on, and `cnt` as the values. Reset index to default. Examine columns. Then melt the table back to its original form
2. Pivot data using `year` as the primary key, (`month`, `is_group`) as the columns to pivot on, and `cnt` as the values. Reset index to default. Examine columns. Then melt the table back to its original from [warning: this one is hard]
3. More data cleaning (on **pivoted**)
    1) Clean the variable *True* in the DataFrame **`pivoted`**, by replacing the missing values (NaN) with 0 and casting non-missing values to integers. Store the result in a new column *True_cleaned* (hint: use lambda function)
    2) Check that *True* and *True_cleaned* indeed contain the same information. Then delete *True* from *pivoted_data*
    3) Cast the column *False* in the DataFrame **pivoted** to **int** type, and store the casted values in a new variable named *False_cleaned*. Drop the column *False* from **pivoted**
    4) Rename the column *True_cleaned* to *count1* and *False_cleaned* to *count0* in **pivoted**