

# Stacking DataFrames

RESHAPING DATA WITH PANDAS



**Maria Eugenia Inzaugarat**  
Data Scientist

# Row multi-indices

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

# Setting the index

churn

```
   credit_score  age  country  num_products  exited
0           619   43   France             1      Yes
1           608   34  Germany             0      No
2           502   23   France             1      Yes
```

# Setting the index

```
churn.set_index(['country', 'age'], inplace=True)
```

		credit_score	num_products	exited
age	country			
43	France	619	1	Yes
34	Germany	608	0	No
23	France	502	1	Yes

# MultiIndex from array

```
new_array = [['yes', 'no', 'yes'], ['no', 'yes', 'yes']]
churn.index = pd.MultiIndex.from_arrays(new_array, names=['member', 'credit_card'])
churn
```

		credit_score	age	country	num_products	exited
member	credit_card					
yes	no	619	43	France	1	Yes
no	yes	608	34	Germany	0	No
yes	yes	502	23	France	1	Yes

# Multindex DataFrames

		2019		2020	
		height	weight	height	weight
Last	First				
Wick	John	185	68	185	70
	Julien	164	61	164	60
Shelley	Mary	164	59	164	60
	Frank	155	65	155	58


# MultiIndex DataFrames

```
index = pd.MultiIndex.from_arrays([[ 'Wick', 'Wick', 'Shelley', 'Shelley'],
                                   [ 'John', 'Julien', 'Mary', 'Frank']],
                                names=[ 'last', 'first'])
columns = pd.MultiIndex.from_arrays([[ '2019', '2019', '2020', '2020'],
                                     [ 'age', 'weight', 'age', 'weight']],
                                   names=[ 'year', 'feature'])
patients = pd.DataFrame(data, index=index, columns=columns)
patients
```

	year		2019		2020
	feature	age	weight	age	weight
last	first				
Wick	John	25	68	26	72
	Julien	31	72	32	73
Shelley	Mary	41	68	42	69
	Frank	32	75	33	74

# The .stack() method

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58




Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

`df.stack()`



# The .stack() method

Rearrange a level of the columns to obtain a reshaped DataFrame with a new inner-most level row index



		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

# Stack into a series

```
churn
```

	credit_score	age	country	num_products	exited
0	619	43	France	1	Yes
1	608	34	Germany	0	No
2	502	23	France	1	Yes

```
churned_stacked = churn.stack()  
churned_stacked.head(10)
```

member	credit_card		
yes	no	credit_score	619
		age	43
		country	France
		num_products	1
		churn	Yes
no	yes	credit_score	608
		age	34
		country	Germany
		num_products	0
		churn	No

# Stack into a DataFrame

```
patients
```

	year		2019		2020
	feature	age	weight	age	weight
last	first				
Wick	John	25	68	26	72
	Julien	31	72	32	73
Shelley	Mary	41	68	42	69
	Frank	32	75	33	74

```
patients_stacked = patients.stack()  
patients_stacked
```

		year	2019	2020
last	first	feature		
Wick	John	age	25	26
		weight	68	72
	Julien	age	31	32
		weight	72	73
Shelley	Mary	age	41	42
		weight	68	69
	Frank	age	32	33
		weight	75	74

# Stack a level by number

```
patients
```

		year		2019		2020	
		feature	age	weight	age	weight	
last	first						
Wick	John	25	68	26	72		
	Julien	31	72	32	73		
Shelley	Mary	41	68	42	69		
	Frank	32	75	33	74		

```
patients.stack(level=0)
```

		feature	age	weight
last	first	year		
Wick	John	2019	25	68
		2020	26	72
	Julien	2019	31	72
		2020	32	73
Shelley	Mary	2019	41	68
		2020	42	69
	Frank	2019	32	75
		2020	33	74

# Stack a level by name

```
patients
```

	year		2019		2020
	feature	age	weight	age	weight
last	first				
Wick	John	25	68	26	72
	Julien	31	72	32	73
Shelley	Mary	41	68	42	69
	Frank	32	75	33	74

```
patients.stack(level='year')
```

		feature	age	weight
	last	first	year	
	Wick	John	2019	25 68
			2020	26 72
		Julien	2019	31 72
			2020	32 73
Shelley		Mary	2019	41 68
			2020	42 69
		Frank	2019	32 75
			2020	33 74

**Let's practice!**  
RESHAPING DATA WITH PANDAS

# Unstacking DataFrames


RESHAPING DATA WITH PANDAS



**Maria Eugenia Inzaugarat**  
Data Scientist

# Review

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58



Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

`df.stack()`



# Undoing stacking process

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58



		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

# The .unstack() method

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58




		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

`df.unstack()`

# The `.unstack()` method

Rearrange a level of the row index into the columns to obtain a reshaped DataFrame with a new inner-most level column index.



Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

# Unstack Series

churn\_stacked

member	credit_card		
yes	no	credit_score	619
		age	43
		country	France
		num_products	1
		churn	Yes
no	yes	credit_score	608
		age	34
		country	Germany
		num_products	0
		churn	No
yes	yes	credit_score	502
		age	23
		country	France
		num_products	1
		churn	Yes

# Unstack Series

```
churned_stacked.unstack()
```

		credit_score	age	country	num_products	exited	
member	credit_card						
	no	yes	608	34	Germany	0	No
	yes	no	619	43	France	1	Yes
		yes	502	23	France	1	Yes

# Unstacking a DataFrame

```
patients_stacked
```

		year	2019	2020
first	last	feature		
Wick	John	age	25	26
		weight	68	72
	Julien	age	31	32
		weight	72	73
Shelley	Mary	age	41	42
		weight	68	69
	Frank	age	32	33
		weight	75	74


# Unstacking a DataFrame

```
patients_stacked.unstack()
```

			2019		2020
	feature	age	weight	age	weight
	last	first			
	Shelley	Frank	32	75	33 74
		Mary	41	68	42 69
	Wick	John	25	68	26 72
		Julien	31	72	32 73

# Unstack a level

Last	First		
Johnson	Louis	age	32
		weight	68
	Mary	age	42
		weight	61
Smith	Louis	age	20
		weight	59
	Mary	age	32
		weight	58



	First	Louis	Mary
Last			
Johnson	age	32	42
	weight	68	61
Smith	age	20	32
	weight	59	58

`df.unstack(level=1)` or `df.unstack(level='First')`



# Unstack level by number

```
churn_stacked.head(10)
```

member	credit_card		
yes	no	credit_score	619
		age	43
		country	France
		num_products	1
		churn	Yes
no	yes	credit_score	608
		age	34
		country	Germany
		num_products	0
		churn	No

```
churn_stacked.unstack(level=0)
```

	member	no	yes
credit_card			
	no	credit_score	NaN 619
		age	NaN 43
		country	NaN France
		num_products	NaN 1
		churn	NaN Yes
	yes	credit_score	608 502
		age	34 23
		country	Germany France
		num_products	0 1
		churn	No Yes

# Unstack level by name

```
churn_stacked.head(10)
```

member	credit_card		
yes	no	credit_score	619
		age	43
		country	France
		num_products	1
		churn	Yes
no	yes	credit_score	608
		age	34
		country	Germany
		num_products	0
		churn	No

```
churn_stacked.unstack(level='credit_card')
```

	credit_card	no	yes
member			
	no	credit_score	NaN
		age	NaN
		country	NaN
		num_products	NaN
		churn	NaN
	yes	credit_score	619
		age	43
		country	France
		num_products	1
		churn	Yes

# Sort index

```
patients_stacked.unstack().sort_index(ascending=False)
```

	year		2019		2020	
	feature		age	weight	age	weight
	last	first				
	Wick	Julien	31	72	32	73
		John	25	68	26	72
Shelley		Mary	41	68	42	69
		Frank	32	75	33	74

# Rearranging levels

```
patients_stacked
```

		year	2019	2020
first	last	feature		
Wick	John	age	25	26
		weight	68	72
	Julien	age	31	32
		weight	72	73
Shelley	Mary	age	41	42
		weight	68	69
	Frank	age	32	33
		weight	75	74

```
patients_stacked.unstack(level=1).stack(level=0)
```

first			Frank	John	Julien	Mary
	last	feature	year			
Shelley	age	2019	32.0	NaN	NaN	41.0
		2020	33.0	NaN	NaN	42.0
	weight	2019	75.0	NaN	NaN	68.0
		2020	74.0	NaN	NaN	69.0
Wick	age	2019	NaN	25.0	31.0	NaN
		2020	NaN	26.0	32.0	NaN
	weight	2019	NaN	68.0	72.0	NaN
		2020	NaN	72.0	73.0	NaN

**Let's practice!**  
RESHAPING DATA WITH PANDAS

# Working with multiple levels

RESHAPING DATA WITH PANDAS



**Maria Eugenia Inzaugarat**  
Instructor

# Review

- Stack and unstack DataFrames and Series
- Choose a level to stack or unstack by name or number
- Rearrange levels by combining unstack and stack

# Rearranging multiple levels

- Swap levels
- Stack and unstack multiple levels at the same time



# Swap levels

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

	First	Last	
height	John	Wick	185
weight			68
height	Julien		164
weight			61
height	Mary	Shelley	164
weight			59
height	Frank		155
weight			58

```
df.swaplevel( 0, 2 )
```

# Swap levels

cars

			2019	2020
price	Golf	VW	25	26
sold	Golf	VW	68	72
price	Passat	VW	31	32
sold	Passat	VW	72	73
price	A-class	Mercedes	41	42
sold	A-class	Mercedes	68	69
price	C-class	Mercedes	32	33
sold	C-class	Mercedes	75	74

# Swap levels

```
cars
```

			2019	2020
price	Golf	VW	25	26
sold	Golf	VW	68	72
price	Passat	VW	31	32
sold	Passat	VW	72	73
price	A-class	Mercedes	41	42
sold	A-class	Mercedes	68	69
price	C-class	Mercedes	32	33
sold	C-class	Mercedes	75	74

```
cars.swaplevel(0, 2)
```

			2019	2020
VW	Golf	price	25	26
		sold	68	72
	Passat	price	31	32
		sold	72	73
Mercedes	A-class	price	41	42
		sold	68	69
	C-class	price	32	33
		sold	75	74

# Swap levels and unstack

```
cars
```

			2019	2020
price	Golf	VW	25	26
sold	Golf	VW	68	72
price	Passat	VW	31	32
sold	Passat	VW	72	73
price	A-class	Mercedes	41	42
sold	A-class	Mercedes	68	69
price	C-class	Mercedes	32	33
sold	C-class	Mercedes	75	74

```
cars.swaplevel(0, 2).unstack()
```

			2019		2020
		price	sold	price	sold
Mercedes	A-class	41	68	42	69
	C-class	32	75	33	74
VW	Golf	25	68	26	72
	Passat	31	72	32	73

# Swap levels and unstack

`cars`

			2019	2020
price	Golf	VW	25	26
sold	Golf	VW	68	72
price	Passat	VW	31	32
sold	Passat	VW	72	73
price	A-class	Mercedes	41	42
sold	A-class	Mercedes	68	69
price	C-class	Mercedes	32	33
sold	C-class	Mercedes	75	74

`cars.unstack()`

		Mercedes	VW	Mercedes	VW
		2019	2019	2020	2020
price	A-class	41.0	NaN	42.0	NaN
	C-class	32.0	NaN	33.0	NaN
	Golf	NaN	25.0	NaN	26.0
	Passat	NaN	31.0	NaN	32.0
sold	A-class	68.0	NaN	69.0	NaN
	C-class	75.0	NaN	74.0	NaN
	Golf	NaN	68.0	NaN	72.0
	Passat	NaN	72.0	NaN	73.0

# Swap levels and unstack

```
cars
```

			2019	2020
price	Golf	VW	25	26
sold	Golf	VW	68	72
price	Passat	VW	31	32
sold	Passat	VW	72	73
price	A-class	Mercedes	41	42
sold	A-class	Mercedes	68	69
price	C-class	Mercedes	32	33
sold	C-class	Mercedes	75	74

```
cars.unstack().swaplevel(0, 1, axis=1)
```

			2019		2020
		Mercedes	VW	Mercedes	VW
price	A-class	41.0	NaN	42.0	NaN
	C-class	32.0	NaN	33.0	NaN
	Golf	NaN	25.0	NaN	26.0
	Passat	NaN	31.0	NaN	32.0
sold	A-class	68.0	NaN	69.0	NaN
	C-class	75.0	NaN	74.0	NaN
	Golf	NaN	68.0	NaN	72.0
	Passat	NaN	72.0	NaN	73.0

# Swap levels and stack

`cars`

			2019	2020
price	Golf	VW	25	26
sold	Golf	VW	68	72
price	Passat	VW	31	32
sold	Passat	VW	72	73
price	A-class	Mercedes	41	42
sold	A-class	Mercedes	68	69
price	C-class	Mercedes	32	33
sold	C-class	Mercedes	75	74

`cars.stack()`

price	Golf	VW	2019	25
			2020	26
sold	Golf	VW	2019	68
			2020	72
price	Passat	VW	2019	31
			2020	32
sold	Passat	VW	2019	72
			2020	73
price	A-class	Mercedes	2019	41
			2020	42
sold	A-class	Mercedes	2019	68
			2020	69

# Swap levels and stack

```
cars
```

			2019	2020
price	Golf	VW	25	26
sold	Golf	VW	68	72
price	Passat	VW	31	32
sold	Passat	VW	72	73
price	A-class	Mercedes	41	42
sold	A-class	Mercedes	68	69
price	C-class	Mercedes	32	33
sold	C-class	Mercedes	75	74

```
cars.stack().swaplevel(0, 2)
```

VW	Golf	price	2019	25
			2020	26
		sold	2019	68
			2020	72
	Passat	price	2019	31
			2020	32
		sold	2019	72
			2020	73
Mercedes	A-class	price	2019	41
			2020	42
		sold	2019	68
			2020	69



# Multiple levels

		day				night	
		2019		2020		2020	
		high	low	high	low	high	low
Last	First						
Wick	John	110	68	120	70	110	70
	Julien	120	61	121	60	115	60
Shelley	Mary	90	59	90	60	100	60
	Frank	100	65	92	58	105	58

# Unstacking multiple levels

cars

year			2019	2020
brand	model	feature		
VW	Golf	price	25	26
		sold	68	72
	Passat	price	31	32
		sold	72	73
Mercedes	A-class	price	41	42
		sold	68	69
	C-class	price	32	33
		sold	75	74

# Unstacking levels by number

```
cars.unstack(level=[0, 1])
```

	year				2019				2020			
brand	VW		Mercedes		VW		Mercedes		VW		Mercedes	
model	Golf	Passat	A-class	C-class	Golf	Passat	A-class	C-class	Golf	Passat	A-class	C-class
feature												
price	25	31	41	32	26	32	42	33				
sold	68	72	68	75	72	73	69	74				

# Unstacking levels by name

```
cars.unstack(level=['brand', 'model'])
```

	year				2019				2020			
brand	VW				Mercedes				VW			
model	Golf	Passat	A-class	C-class	Golf	Passat	A-class	C-class	Golf	Passat	A-class	C-class
feature												
price	25	31	41	32	26	32	42	33				
sold	68	72	68	75	72	73	69	74				

# Stacking multiple levels

`cars_unstacked`

year	2019				2020			
brand	VW		Mercedes		VW		Mercedes	
model	Golf	Passat	A-class	C-class	Golf	Passat	A-class	C-class
feature								
price	25	31	41	32	26	32	42	33
sold	68	72	68	75	72	73	69	74

# Stacking by name or number

```
cars_unstacked.stack(level=[0, 1])
```

		model	A-class	C-class	Golf	Passat
feature	year	brand				
price	2019	Mercedes	41.0	32.0	NaN	NaN
		VW	NaN	NaN	25.0	31.0
	2020	Mercedes	42.0	33.0	NaN	NaN
		VW	NaN	NaN	26.0	32.0
sold	2019	Mercedes	68.0	75.0	NaN	NaN
		VW	NaN	NaN	68.0	72.0
	2020	Mercedes	69.0	74.0	NaN	NaN
		VW	NaN	NaN	72.0	73.0

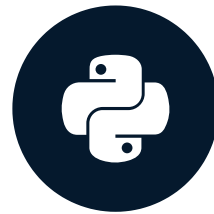
```
cars_unstacked.stack(level=['year', 'brand'])
```

		model	A-class	C-class	Golf	Passat
feature	year	brand				
price	2019	Mercedes	41.0	32.0	NaN	NaN
		VW	NaN	NaN	25.0	31.0
	2020	Mercedes	42.0	33.0	NaN	NaN
		VW	NaN	NaN	26.0	32.0
sold	2019	Mercedes	68.0	75.0	NaN	NaN
		VW	NaN	NaN	68.0	72.0
	2020	Mercedes	69.0	74.0	NaN	NaN
		VW	NaN	NaN	72.0	73.0

**Let's practice!**  
RESHAPING DATA WITH PANDAS

# Handling missing data

RESHAPING DATA WITH PANDAS



**Maria Eugenia Inzaugarat**  
Data Scientist



# Review

- Stack and unstack DataFrames:
  - All columns index levels
  - A row index level
  - Choose which levels to stack or unstack

# Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals
```

```
      jump  run  fly
class  order  name
Mammalia carnivora  dog      No  Yes  No
          Diprotodontia Kangaroo Yes  No  No
Aves      hervibora  bird     No  No  Yes
```

# Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals
```

```
           jump  run  fly
class  order      name
Mammalia carnivora   dog      No  Yes  No <--
          Diprotodontia Kangaroo Yes  No  No
Aves      hervibora   bird      No  No  Yes
```

# Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals.unstack(level='class')
```

			jump		run		fly	
	class		Aves	Mammalia	Aves	Mammalia	Aves	Mammalia
	order	name						
	Diprotodontia	Kangaroo	NaN	Yes	NaN	No	NaN	No
	carnivora	Dog	NaN	No	NaN	Yes	NaN	No
	Charadriiformes	Avocet	No	NaN	No	NaN	Yes	NaN

# Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals.unstack(level='class')
```

			jump		run		fly	
	class		Aves	Mammalia	Aves	Mammalia	Aves	Mammalia
	order	name						
	Diprotodontia	Kangaroo	NaN	Yes	NaN	No	NaN	No
-----								
	carnivora	Dog	NaN <--	No	NaN	Yes	NaN	No
-----								
	Charadriiformes	Avocet	No	NaN	No	NaN	Yes	NaN

# Handling NaN with unstack

```
animals.unstack(level='class', fill_value= )
```

# Handling NaN with unstack

```
animals.unstack(level='class', fill_value='No')
```

# Handling NaN with unstack

```
animals.unstack(level='class', fill_value='No').sort_index(level=['order', 'name'], ascending=[True, False])
```

			jump		run		fly	
clas			Aves	Mammalia	Aves	Mammalia	Aves	Mammalia
order	name							
Diprotodontia	Kangaroo		No	Yes	No	No	No	No
carnivora	Dog		No	No	No	Yes	No	No
Charadriiformes	Avocet		No	No	No	No	Yes	No



# Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers
```

```
      petals Stigma  
      number  size  
rose      40   NaN  
Lily       8     5
```

# Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers.stack()
```

```
      Stigma  petals
rose number   NaN   40.0
Lily number   NaN    8.0
      size      5    NaN
```

# Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers.stack(dropna=True)
```

	Stigma	petals
rose number	NaN	40.0
Lily number	NaN	8.0
size	5	NaN

# Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers.stack(dropna=False)
```

```
      Stigma  petals
rose number  NaN    40.0
      size    NaN    NaN <--
Lily number  NaN     8.0
      size     5     NaN
```

# Handling NaN with stack

```
flowers.stack(dropna=False).fillna(0)
```

	Stigma	petals
rose number	0	40.0
size	0	0
Lily number	0	8.0
size	5	0

**Let's practice!**  
RESHAPING DATA WITH PANDAS