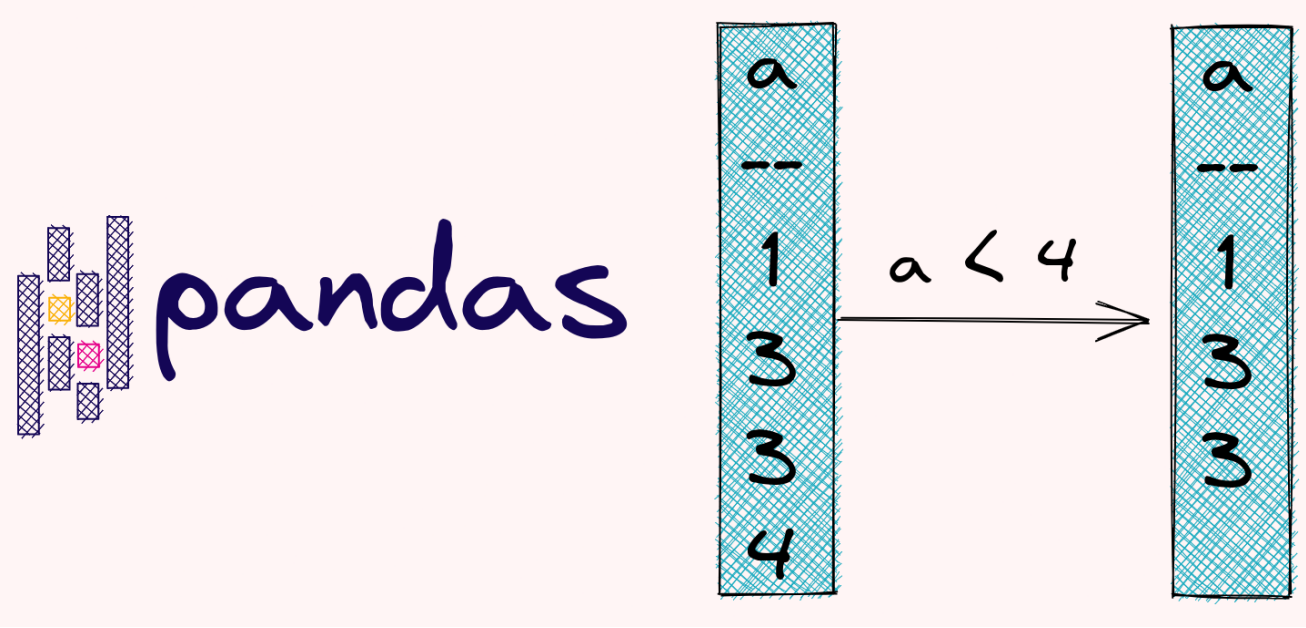
Efficient Python Tricks and Tools for Data Scientists - By Khuyen Tran

Filter Rows or Columns

GitHub View on GitHub Book View Book

This section shows some methods to filter rows or columns in a pandas DataFrame.



Pandas.Series.isin: Filter Rows Only If Column Contains Values From Another List

When working with a pandas Dataframe, if you want to select the values that are in another list, the fastest way is to use `isin`.

In the example below, 2 is filtered out because 3 is not in the list.

```
import pandas as pd

df = pd.DataFrame({'a': [1, 2, 3], 'b': [4, 5, 6]})
df
```

	a	b
0	1	4
1	2	5
2	3	6

```
l = [1, 2, 6, 7]
df.a.isin(l)
```

```
0     True
1     True
2    False
Name: a, dtype: bool
```

```
df = df[df.a.isin(l)]
df
```

	a	b
0	1	4
1	2	5

df.query: Query Columns Using Boolean Expression

It can be lengthy to filter columns of a pandas DataFrame using brackets.

```
import pandas as pd

df = pd.DataFrame(
    {"fruit": ["apple", "orange", "grape",
"grape"], "price": [4, 5, 6, 7]}
)
```

```
print(df[(df.price > 4) & (df.fruit ==
"grape")])
```

```
   fruit  price
2  grape      6
3  grape      7
```

To shorten the filtering statements, use `df.query` instead.

```
df.query("price > 4 & fruit == 'grape'")
```

	fruit	price
2	grape	6
3	grape	7

transform: Filter a pandas DataFrame by Value Counts

To filter a pandas DataFrame based on the occurrences of categories, you might attempt to use `df.groupby` and `df.count`.

```
import pandas as pd

df = pd.DataFrame({"type": ["A", "A", "O", "B", "O", "A"], "value": [5, 3, 2, 1, 4, 2]})
df
```

	type	value
0	A	5
1	A	3
2	O	2
3	B	1
4	O	4
5	A	2

```
df.groupby("type")["type"].count()
```

```
type
A    3
B    1
O    2
Name: type, dtype: int64
```

However, since the Series returned by the `count` method is shorter than the original DataFrame, you will get an error when filtering.

```
df.loc[df.groupby("type")["type"].count() > 1]
```

```
IndexingError: Unalignable boolean Series
provided as indexer (index of the boolean
Series and of the indexed object do not
match).
```

Instead of using `count`, use `transform`. This method will return the Series of value counts with the same length as the original DataFrame.

```
df.groupby("type")["type"].transform("size")
```

```
0    3
1    3
2    2
3    1
4    2
5    3
Name: type, dtype: int64
```

Now you can filter without encountering any error.

```
df.loc[df.groupby("type")
["type"].transform("size") > 1]
```

	type	value
0	A	5
1	A	3
2	O	2
4	O	4
5	A	2

df.filter: Filter Columns Based on a Subset of Their Names

If you want to filter columns of a pandas DataFrame based on characters in their names, use `DataFrame.filter`. In the example below, we only choose the columns that contain the word "cat".

```
import pandas as pd

df = pd.DataFrame({"cat1": ["a", "b"], "cat2":
["b", "c"], "num1": [1, 2]})
df
```

	cat1	cat2	num1
0	a	b	1
1	b	c	2

```
df.filter(like='cat', axis=1)
```

	cat1	cat2
0	a	b
1	b	c

pandas.clip: Exclude Outliers

Outliers are unusual values in your dataset, and they can distort statistical analyses.

```
import pandas as pd

data = {"col0": [9, -3, 0, -1, 5]}
df = pd.DataFrame(data)
df
```

	col0
0	9
1	-3
2	0
3	-1
4	5

If you want to trim values that the outliers, one of the methods is to use `df.clip`.

Below is how to use the 0.5-quantile as the lower threshold and .95-quantile as the upper threshold

```
lower = df.col0.quantile(0.05)
upper = df.col0.quantile(0.95)

df.clip(lower=lower, upper=upper)
```

	col0
0	8.2
1	-2.6
2	0.0
3	-1.0
4	5.0