Effective Programming Practices for Economists

Data management with pandas

Selecting rows and columns

Janoś Gabler and Hans-Martin von Gaudecker

Overview

- Selecting columns
- Selecting individual rows
- Selecting rows and columns
- Selecting rows using Boolean Series
- Selecting rows with queries

Selecting columns

```
>>> df["country"]

0      Cuba
1      Cuba
2     Spain
3     Spain
Name: country, dtype: string
```

```
>>> df[["country", "continent"]]
```

	country	continent
0	Cuba	Americas
1	Cuba	Americas
2	Spain	Europe
3	Spain	Europe

- Column selection is with square brackets
- For multiple columns you need double brackets:
  - Outer: selecting columns
  - Inner: defining a list of variables

Selecting individual rows

```
>>> df.loc[1]

country             Cuba
continent       Americas
year                2007
life_exp          78.273
Name: 1, dtype: object
```

```
>>> df = df.set_index(["country", "year"])
>>> df.loc["Cuba"]
```

	continent	life_exp
year		
2002	Americas	77.16
2007	Americas	78.27

```
>>> df.loc[("Cuba", 2002)]

continent    Americas
life_exp       77.158
Name: (Cuba, 2002), dtype: object
```

- Selection of rows needs `` `.loc[]` ``
- Selection is label based!
- For a MultiIndex you can specify some or all levels

Selecting rows and columns

```
>>> df.loc[1, "country"]
'Cuba'

>>> df.loc[[1, 3], ["country", "year"]]
```

	country	year
1	Cuba	2007
3	Spain	2007

- Use `` `.loc[rows, columns]` `` to select rows and columns
- Can use everything you have seen before

Selecting rows using Boolean Series

```
df["year"] >= 2005

0    False
1     True
2    False
3     True
Name: year, dtype: bool

>>> df[df["year"] >= 2005]
```

	country	continent	year	life_exp
1	Cuba	Americas	2007	78.27
3	Spain	Europe	2007	80.94

- Comparisons of Series produce Boolean Series!
- Complex conditions with `|` and `&`
- Boolean Series can be used for selecting rows
- Works also inside `.loc`

Selecting rows with queries

```
>>> df.query("year >= 2005")
```

	country	continent	year	life_exp
1	Cuba	Americas	2007	78.27
3	Spain	Europe	2007	80.94

```
>>> df.query("year >= 2005 & continent == 'Europe'")
```

	country	continent	year	life_exp
3	Spain	Europe	2007	80.94

- `` `.query` `` selects rows based on strings with conditions
- Can use index names just as column names
- Use single quotes (`` `'` ``) for string value inside the query
- More readable than selection via Boolean Series