Day 2

BUSI 520: Python for Business Research

Kerry Back, JGSB, Rice University

Lambda functions

Convenient way to write short function definitions

In [9]:

```python
def f(x):
    return x**2
print(f(3))

f = lambda x: x**2
print(f(3))
```

```
9
9
```

Line continuations

- can use a backslash \ at the end of a line to continue it on the next
- or enclose in ()
- lists can be continued without a backslash

In [18]:

```
def f(x):
    return 2*x \
        + 3
print(f(2))

def f(x):
    return (
        2*x
        + 3
    )
print(f(2))

lst = [
    "a",
    "b",
    "c"
]
print(lst)
```

```
7
7
['a', 'b', 'c']
```

Reading and writing dataframes with pandas

- Can read csv, excel, sas, stata, sql, ...
- Can write also

Two ways to read stata dta files

1. pd.read_stata
2. StataReader

In [3]:

```python
import pandas as pd
df = pd.read_stata("WAGE1.DTA")
df
```

Out[3]:

	wage	educ	exper	tenure	nonwhite	female	married	numdep	smsa	n
0	3.10	11	2	0	0	1	0	2	1	
1	3.24	12	22	2	0	1	1	3	1	
2	3.00	11	2	0	0	0	0	2	0	
3	6.00	8	44	28	0	0	1	0	1	
4	5.30	12	7	2	0	0	1	1	0	
...	...	...	...	...	...	...	...	...	...	
521	15.00	16	14	2	0	1	1	2	0	
522	2.27	10	2	0	0	1	0	3	0	
523	4.67	15	13	18	0	0	1	3	0	
524	11.56	16	5	1	0	0	1	0	0	
525	3.50	14	5	4	1	1	0	2	0	

526 rows × 24 columns

In [5]:

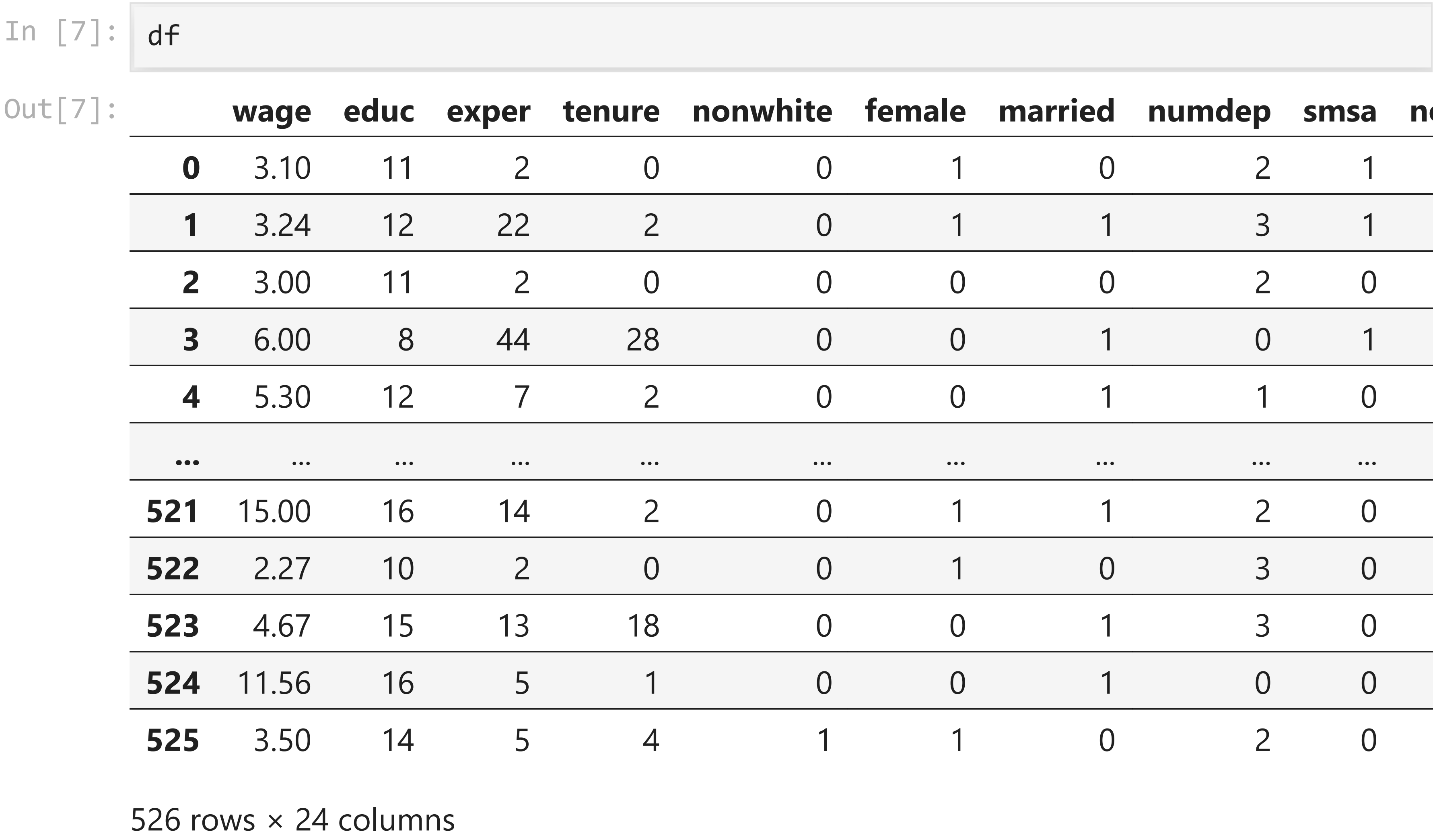
```
from pandas.io.stata import StataReader

file = StataReader("WAGE1.dta")
variables = file.variable_labels()
df = file.read()
```

In [6]:

```
variables
```

Out[6]:

```
{'wage': 'average hourly earnings',
 'educ': 'years of education',
 'exper': 'years potential experience',
 'tenure': 'years with current employer',
 'nonwhite': '=1 if nonwhite',
 'female': '=1 if female',
 'married': '=1 if married',
 'numdep': 'number of dependents',
 'smsa': '=1 if live in SMSA',
 'northcen': '=1 if live in north central U.S',
 'south': '=1 if live in southern region',
 'west': '=1 if live in western region',
 'construc': '=1 if work in construc. indus.',
 'ndurman': '=1 if in nondur. manuf. indus.',
 'trcommpu': '=1 if in trans, commun, pub ut',
 'trade': '=1 if in wholesale or retail',
 'services': '=1 if in services indus.',
 'profserv': '=1 if in prof. serv. indus.',
 'profocc': '=1 if in profess. occupation',
 'clerocc': '=1 if in clerical occupation',
 'servocc': '=1 if in service occupation',
 'lwage': 'log(wage)',
 'expersq': 'exper^2',
 'tenursq': 'tenure^2'}
```



In [7]:

```
df
```

Out[7]:

	wage	educ	exper	tenure	nonwhite	female	married	numdep	smsa	n
0	3.10	11	2	0	0	1	0	2	1	
1	3.24	12	22	2	0	1	1	3	1	
2	3.00	11	2	0	0	0	0	2	0	
3	6.00	8	44	28	0	0	1	0	1	
4	5.30	12	7	2	0	0	1	1	0	
...	...	...	...	...	...	...	...	...	...	
521	15.00	16	14	2	0	1	1	2	0	
522	2.27	10	2	0	0	1	0	3	0	
523	4.67	15	13	18	0	0	1	3	0	
524	11.56	16	5	1	0	0	1	0	0	
525	3.50	14	5	4	1	1	0	2	0	

526 rows × 24 columns

What can we do with pandas?

- Explore
- Select
- Transform
- Aggregate
- Sort, rank, and cut
- Filter
- Aggregate by group
- Merge
- Plot

Explore

- .info()
- .describe()
- .head()
- .tail()
- .columns
- .index

Select

- [] for columns
- or .column_name for columns
- .loc[] to get rows using labels
- .iloc[] to get rows using index 0, 1, ...

Transform

- mathematical operations to individual columns or rows
- combine columns or combine rows in mathematical operations (add, subtract, ...)
- create new columns or rows
- np.where(condition, if_true, if_false)
- .map(f) or .map(lambda x: ...)
- exercises:
    1. change the female column to be "F" if 1 and "M" if 0
    2. create a new column that is 1 if experience <=5, is 2 if 5 < experience <= 10, 3 if 10 < experience <= 20, 4 otherwise
    3. create a new column that is "northcen" if northcen=1, is "south" if south=1, is "west" if west=1, and is "east" otherwise.

Aggregate

- .sum(), .mean(), .median(), .quantile(), .std(), .var(), .min(), .max()
- .apply(lambda x: ...)

Sort, rank, and cut

- sort_values() or sort_index()
- .rank()
- pd.cut(data, bin_edges, right=True or False, labels=...)
- pd.qcut(data, number_of_groups, labels=...)

Filter

- df[df.wage<10]
- df[(df.wage<10) & (df.female==1)]
- df[(df.wage<10) | (df.female==1)]
- df.dropna() or df.dropna(subset=[...])
- not filtering, but .fillna is also useful

Aggregate by group

- df.groupby("female").wage.mean()
- df.groupby(["female", "married"]).wage.mean().unstack()
- can use any aggregation function including custom functions with .apply
- df.groupby("female").wage.apply(lambda x: x.quantile([0.2, 0.4, 0.6, 0.8]))
- df.groupby("female").wage.apply(lambda x: pd.qcut(x, 5, labels=range(1, 6)))

In [20]:

```
df["group"] = df.groupby(["female"], group_keys=False).wage.apply(
    lambda x: pd.qcut(x, 5, labels=range(1, 6))
)
df[["female", "wage", "group"]]
```

ask Julius

- to read WAGE1.DTA and describe it and then do some aggregation by groups with it
- to get the seaborn tips dataset and do some aggregation by groups with it
- to use pandas datareader to get the Fama-French factors from French's data library and merge them with MOM from French's data library
- to create boxplots for the Fama-French factors and MOM