

# **Effective Programming Practices for Economists**

## **Data management with pandas**

### **Loading and saving data**

Janoś Gabler and Hans-Martin von Gaudecker

# Example: Loading a csv file

```
>>> df = pd.read_csv(  
...      "gapminder.csv",  
...      engine="pyarrow",  
... )
```

```
>>> df
```

	country	continent	year	life_exp
0	Cuba	Americas	2002	77.16
1	Cuba	Americas	2007	78.27
2	Spain	Europe	2002	79.78
3	Spain	Europe	2007	80.94

gapminder.csv looks like this

```
country,continent,year,life_exp  
Cuba,Americas,2002,77.158  
Cuba,Americas,2007,78.273  
Spain,Europe,2002,79.780  
Spain,Europe,2007,80.941
```

- first argument is path
- `engine="pyarrow"` ensures we are getting modern pandas dtypes
- Many other optional arguments

# Other read functions

reader	extension	comment
<code>pd.read_csv</code>	<code>.csv</code>	Often need to use optional arguments to make it work
<code>pd.read_pickle</code>	<code>.pkl</code>	Good for intermediate files; Python specific.
<code>pd.read_feather</code>	<code>.arrow</code>	Very modern and powerful file format.
<code>pd.read_stata</code>	<code>.dta</code>	Stata's proprietary format. Avoid if you can.
<code>pd.read_fwf</code>	<code>.fwf</code>	Avoid this whenever you can!

Each read function has a corresponding write function

# Example: Write an Apache Arrow file

```
df.to_feather(path="gapminder.arrow")
```

- First argument is a file path
- More keyword arguments would allow for specifying compression level, format version
- Methods for other file formats tend to require more options

# File format recommendations

- Use `.pk1` format for processed datasets that you do not share with others
  - Very fast to read and write
  - Preserves every aspect of your DataFrame (e.g. dtypes)
- Use `.arrow` to save files you want to share with others
  - Can be read by many languages and programs
  - Efficient compression
- Use `.dta` iff sharing with Stata users