pandas

Data in tables: load it, look at it, select, clean, summarise, combine

Python cheatsheet · MATH60033A · Quantitative Methods in IB · used in sessions 2 to 13

```
import pandas as pd
```

a DataFrame is a table; a Series is one column

1 · Get data in, and out

One line from a file to a DataFrame. Say which sheet, which columns.

`pd.read_csv("file.csv")`
a comma-separated file.

```
d = pd.read_csv("data/my_data.csv")
```

`pd.read_parquet(path)`
the course's own format: fast, keeps types.

```
core = pd.read_parquet("data/spine/core.parquet")
```

`pd.read_stata(path, columns=[...])`
an author's Stata `.dta` file; read only the columns you need.

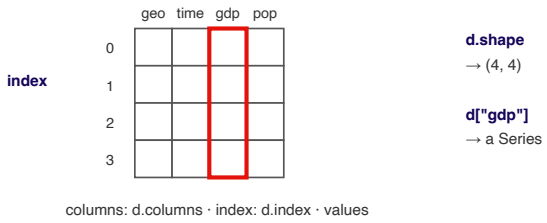
```
d = pd.read_stata(path, columns=["iso", "F0_index"])
```

`pd.read_excel(path, sheet_name=1)`
an Excel workbook; sheets count from 0.

`qmib.load("core")`
a course dataset by name: downloaded once, cached.

`d.to_csv("out.csv", index=False)`
write a table out; `index=False` drops the row numbers.

2 · Look before you model



columns: d.columns · index: d.index · values

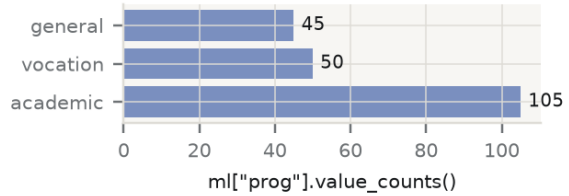
`d.head() · d.shape · d.dtypes`
the first rows · (rows, columns) · the type of each column.

`d.describe()`
count, mean, sd, min, quartiles, max of every numeric column.

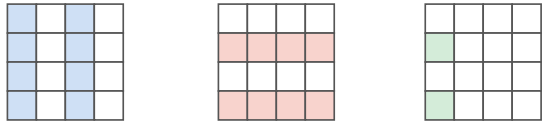
`d.isna().sum()`
how many values are missing, column by column.

`d["iso"].nunique()`
how many distinct values — countries, firms, groups.

`d["prog"].value_counts()`
how often each category occurs; `normalize=True` for shares.



3 · Select rows and columns



`d["gdp"] · d[["gdp", "pop"]]`
one column (a Series) · several (a DataFrame): note the double brackets.

`d[d.gdp > 30000]`
the rows that meet a condition; combine with & (and) and | (or), each in brackets.

`d.loc[rows, cols]`
select by **label** and by condition at once.

```
d.loc[d.time == 2022, ["geo", "gdp"]]
```

`d.iloc[0:5, 0:3]`
select by **position**: rows 0–4, columns 0–2.

`d[d["event"].isin(["growing", "shrinking"])]`
the rows whose value is in a list.

> Select a subset, then add columns? Take `.copy()` first, or pandas warns you are writing on a view.

4 · Clean

`d.dropna(subset=["y", "x"])`
drop rows missing **those** columns. Without `subset=`, a gap in *any* column drops the row.

`d["x"].astype(float)`
change a column's type: text to number, number to category.

`d.rename(columns={"old": "new"})`
rename columns.

`d.sort_values("gdp", ascending=False)`
sort the rows.

> Count rows before and after every cleaning step: what you dropped is part of your result.

5 · Make new columns

`d["log_gdp"] = np.log(d["gdp"])`
a new column from old ones; arithmetic works column by column.

`d["up"] = (d["ret"] > 0).astype(int)`
a 0/1 column from a condition — the outcome of a logit.

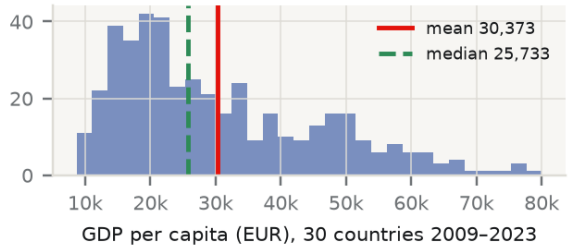
`d.loc[d.foreign == 1, "owner"] = "foreign"`
set a value only on the rows that meet a condition.

`pd.qcut(d["tfp"], 3, labels=["low", "mid", "high"])`
cut a number into equal-sized groups.

`pd.Categorical(x, categories=order, ordered=True)`
categories **with an order** — what an ordinal logit needs.

`pd.get_dummies(d["region"], drop_first=True)`
one 0/1 column per category, minus a reference — for a model without a formula.

6 · Summarise



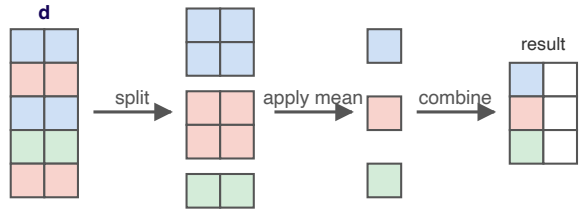
`d["gdp"].mean() .median() .std()`
centre and spread of one column (session 2).

`d["gdp"].quantile([0.25, 0.5, 0.75])`
quartiles: the IQR is the third minus the first.

`d[["gdp", "pop"]].corr()`
the correlation matrix — before a regression, look for near-twins.

`pd.crosstab(d["ses"], d["prog"])`
a contingency table: counts of each combination.

7 · Split, apply, combine — groupby

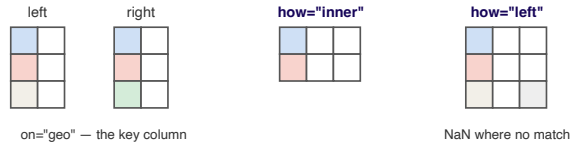


`d.groupby("geo")["gdp"].mean()`
one summary per group: here, each country's mean GDP.

`d.groupby("yr")[["tariff", "input"]].mean()`
several columns at once: the average tariffs by year.

`d.groupby("event")[X].median()`
compare the groups before modelling them.

8 · Combine tables



`left.merge(right, on=["geo", "time"], how="inner")`
join on key columns; `inner` keeps matches only, `left` keeps every left row.

`pd.concat([a, b])` · `pd.concat([a, b], axis=1)`
stack tables on top of each other · side by side.

> After a merge, check the number of rows: a duplicated key silently multiplies them.

9 · A first script, in order

The shape of almost every analysis in this course.

1 · load

```
core = qmib.load("core")
```

2 · keep what you need

```
d = core.dropna(subset=["gdp_pc_eur",
"productivity_idx"])
```

3 · look

```
print(d.shape); print(d.describe().round(1))
```

4 · summarise by group

```
d.groupby("geo")["gdp_pc_eur"].median()
```

5 · model (statsmodels, scikit-learn)

```
smf.ols("gdp_pc_eur ~ productivity_idx",
data=d).fit()
```

10 · When pandas complains

`KeyError: 'gdp'`
no column by that name: check `d.columns` for the exact spelling.

`SettingWithCopyWarning`
you added a column to a slice: take `.copy()` when you select.

`ValueError: ... NaN ...`
missing values where a number is needed: `dropna` first, or `fillna`.

`... in the printout`
pandas hid columns or rows: `pd.set_option` below, or select fewer.

11 · Display

`pd.set_option("display.max_columns", 10)`
how many columns print before pandas shows `...`.

`d.round(3)`
round every number for reading — not for computing.