# Data Handling

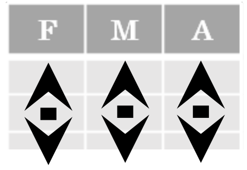
with Gretl Cheat Sheet

https://gretl.sourceforge.net/

Gretl command reference, function reference & User Guide

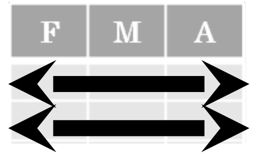
## Gretl Data

In a Gretl data set:



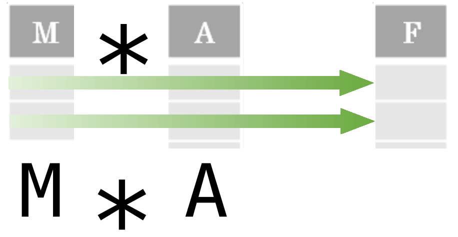
Each **variable** is saved in its own **column**

&



Each **observation** is saved in its own **row**

Tidy data complements Gretl's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.



## Creating Datasets

| | a | b | c |
|---|---|---|---|
| 1 | 4 | 7 | 10 |
| 2 | 5 | 8 | 11 |
| 3 | 6 | 9 | 12 |

```
nulldata 3
series a = {4; 5; 6}
series b = {7; 8; 9}
series c = {10; 11; 12}
```

Specify values for each column.

| | Index | a | b |
|---|---|---|---|
| 1:01 | 1 | 4 | 8 |
| 1:02 | 2 | 5 | 9 |
| 2:01 | 3 | 6 | 10 |
| 2:01 | 4 | 7 | 11 |

```
nulldata 4
series a = {4; 5; 6; 7}
series b = {8; 9; 10; 11}
setobs 2:2 --stacked-time-series
```

Create a panel dataset.

## Open and store data

`open denmark.gdt`

Open a local dataset. Supports various data types such as plain text, csv, MS Excel, Stata, SPSS, GEOJson etc.)

`store MyFile.csv`

Save data to some file. Support for native format, csv, txt, GNU Octave and Stata.

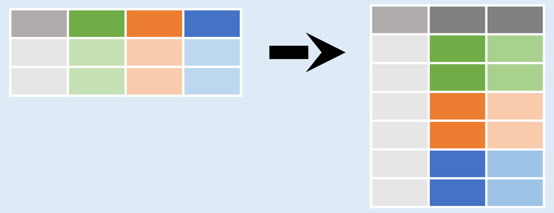
`store --matrix=mat MyFile.csv`

Save a matrix as a dataset.

`open dbnomics`

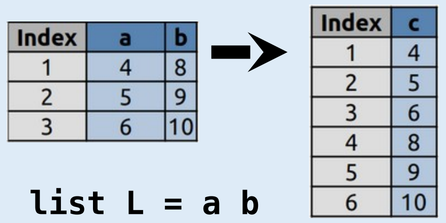
Connect to the dbnomics database.

## Reshaping Data – Change layout, sorting, reindexing, renaming



`dataset transpose`

Gather columns into rows.

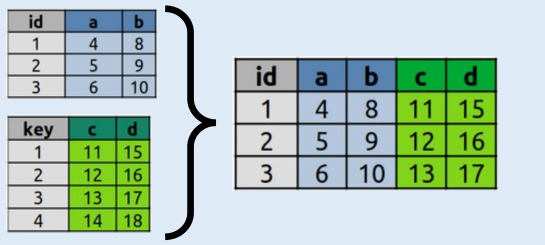


| Index | a | b |
|---|---|---|
| 1 | 4 | 8 |
| 2 | 5 | 9 |
| 3 | 6 | 10 |

| Index | c |
|---|---|
| 1 | 4 |
| 2 | 5 |
| 3 | 6 |
| 4 | 8 |
| 5 | 9 |
| 6 | 10 |

```
list L = a b
series c = stack(L, n)
```

Stack *n* observations from each series in *L*.



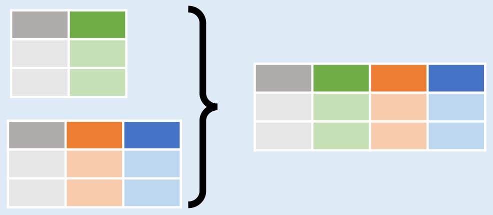
| id | a | b |
|---|---|---|
| 1 | 4 | 8 |
| 2 | 5 | 9 |
| 3 | 6 | 10 |

| key | c | d |
|---|---|---|
| 1 | 11 | 15 |
| 2 | 12 | 16 |
| 3 | 13 | 17 |
| 4 | 14 | 18 |

| id | a | b | c | d |
|---|---|---|---|---|
| 1 | 4 | 8 | 11 | 15 |
| 2 | 5 | 9 | 12 | 16 |
| 3 | 6 | 10 | 13 | 17 |

`join dfile --ikey=id --okey=key`

Left-join of another datafile *dfile*.



`append filename`

Append columns from another file *filename*

`dataset sortby mpg`

Order rows of dataset by values of series (low to high).

`dataset addobs n`

Adds *n* extra observations to the end of the dataset.

`rename y year`

Rename the series *y* of a dataset into *year*.

`delete L`

Drop list of series, *L*, from dataset.

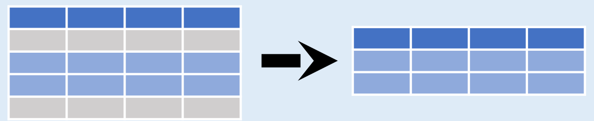
`setobs 1 1 --cross-section`

Reset index of dataset to row numbers.

`setobs 12 2000:1 --time-series`

Set index of dataset to monthly time-series.

## Subset Observations - rows



```
smpl Length > 7 && Width < 3 \
  --restrict
```

Restrict to rows that meet logical criteria.

`dataset resample n`

Randomly select *n* rows.

`smpl a >= values(a)[n] --restrict`

Select top *n* entries based on series *a*.

`smpl a <= values(a)[end-2] --restrict`

Select bottom *n* entries based on series *a*.

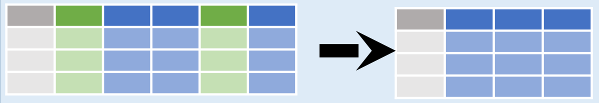
`smpl 1 n`

Select first *n* rows.

`smpl ($tmax – n) $tmax`

Select last *n* rows.

## Working with lists



`list L = Length Width`

Add multiple series with specific names to list.

`list L = 1 2 3`

Add multiple series using the ID number to list.

`list L = y_*`

Add series with the prefix „y_" to list using the wildcard character.

`list L delete`

Remove list *L* from memory.

`delete L`

Delete the series contained in list *L*.

`L += x`

Append series *x* to list *L*.

`L -= x`

Remove series *x* from list *L*.

`list L2 = mpg L1 Width`

Append to a list individual series as well as lists.

`list L3 = L1 || L2`

Union of two lists removing duplicates.

`list L3 = L1 & L2`

Intersection of two lists incl. eventual duplicates.

`list L3 = L1 - L2`

Remain all elements of *L1* that are not in *L2*.

`list L2 = L1[1:4]`

Only pass the first four members of *L1*.

`nelem(L)`

# of elements in list *L*.

`inlist(L, y)`

Return the 1-based position of series *y* if present in *L*, otherwise zero.

`list H = X ^ Z`

Compute interaction terms between $x_i$ and $z_i$.

**Logic in Gretl**

| | | | |
|---|---|---|---|
| < | Less than | `!=` | Not equal to |
| > | Greater than | `df.column.isin(values)` | Group membership |
| == | Equals | `missing(y)` | Is NaN |
| <= | Less than or equals | `ok(y)` | Is not NaN |
| >= | Greater than or equals | `&&,\|\|` | Logical and, or, not, xor, any, all |

**regex (Regular Expressions) Examples**

| | |
|---|---|
| `regsub(S, "\.", ",")` | Replace all ',' by '.' |
| `regsub(S, "Foo$", "")` | Delete 'Foo' if the string ends with 'Foo' |
| `regsub(S, "^My", "")` | Delete 'My' if the string starts with 'My' |
| | |

## Summarize Data

**nobs(y)**
# of observations in dataset.

**nelem(dataset)**
# of variables in dataset.

**values(y)**
Distinct values of a series sorted in ascending order.

**summary y x**
Basic descriptive and statistics for variables. The table of statistics produced can be retrieved in matrix form via the **$result** accessor.

Gretl provides a large set of summary functions that operate on different kinds of Gretl objects (series, list and matrix) depending on the function.

If *y* is a series, the following functions return a scalar values.

| | |
|---|---|
| **sum(y)** Sum values of series. | **min(y)** Minimum value of series. |
| **nobs(y)** # of non-NA values of series. | **max(y)** Maximum value of series. |
| **median(y)** Median value of series. | **mean(y)** Mean value of series. |
| **quantile(y, 0.25)** Quantiles of series. | **var(y)** Variance of series. |
| **skewness(y)** Skewness of series. | **sd(y)** Standard deviation of series. |

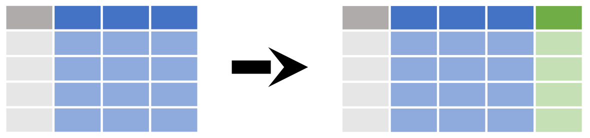
If *L* is a series, the functions return a series holding the applied summary statistics computed across all columns for each row.

## Handling Missing Data

**smpl L --no-missing**
Drop rows of list L with any column having *NA* data.

**series y = ok(y) ? y : value**
Replace all NA data with *value* for series y.

## Make New Series



**series y = NA**
Add a single series initialized with NA values.

**series Volume = Length*Height*Depth**
Add single series.

**series min_ab = min(deflist(a, b))**
Minimum across a list of series for each row.

| index | a | b |
|---|---|---|
| 1 | 4 | 8 |
| 2 | 8 | 5 |
| 3 | 6 | 6 |

→

| index | a | b | c |
|---|---|---|---|
| 1 | 4 | 8 | 4 |
| 2 | 8 | 5 | 5 |
| 3 | 6 | 6 | 6 |

## Group Data

**df.groupby(by="col")**
Return a GroupBy object, grouped by values in column named "col".

**df.groupby(level="ind")**
Return a GroupBy object, grouped by values in index level named "ind".

All of the summary functions listed above can be applied to a group. Additional GroupBy functions:

**size()**
Size of each group.

**agg(*function*)**
Aggregate group using function.

The examples below can also be applied to groups. In this case, the function is applied on a per-group basis, and the returned vectors are of the length of the original DataFrame.

| | |
|---|---|
| **shift(1)** Copy with values shifted by 1. | **shift(-1)** Copy with values lagged by 1. |
| **rank(method='dense')** Ranks with no gaps. | **cumsum()** Cumulative sum. |
| **rank(method='min')** Ranks. Ties get min rank. | **cummax()** Cumulative max. |
| **rank(pct=True)** Ranks rescaled to interval [0, 1]. | **cummin()** Cumulative min. |
| **rank(method='first')** Ranks. Ties go to first value. | **cumprod()** Cumulative product. |

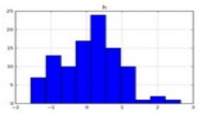
## Windows

**df.expanding()**
Return an Expanding object allowing summary functions to be applied cumulatively.

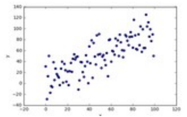
**df.rolling(n)**
Return a Rolling object allowing summary functions to be applied to windows of length n.

## Plotting

**freq y --plot=display**
Histogram for series

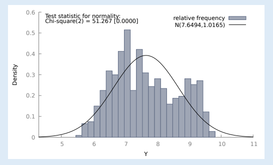


**gnuplot y x**
Scatter chart using pairs of points

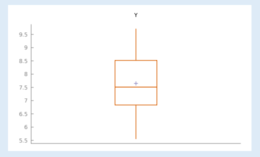


## Plotting

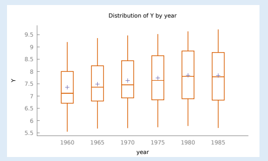
**freq y --normal --plot=display**
Histogram for series *y*.



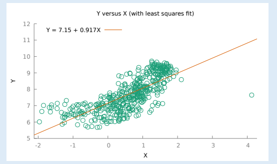
**boxplot y --output=display**
Boxplot for series *y*.



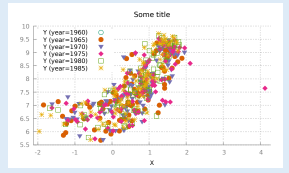
```
boxplot y year --factorized \
  --output=display
```
Boxplot for series *y* grouped by *year*.



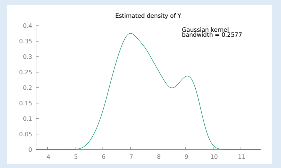
```
gnuplot y x --fit=linear \
  --output=display
```
Scatterplot with linear fit.



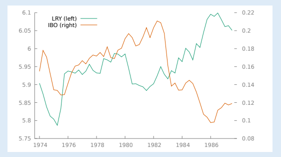
```
gnuplot y x year --dummy \
  --output=display \
  {set title "Some title" \
   Font ',14'; set grid lw 2;}
```
Scatterplot for each discrete value of *year* plus calling some gnuplot options for tweaking.



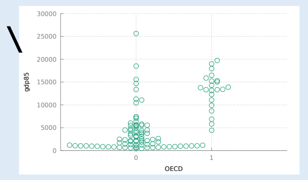
**kdplot y --output=display**
Kernel density plot.



```
gnuplot y x --with-lines \
  --time-series \
  --output=display
```
Time-series plot.



```
gnuplot y x --output=display \
  { set jitter overlap 0.5;\
  set grid;}
```
Scatter plot with jitter points.



```
open data4-10
strings MyPlots
gpbuild MyPlots
    gnuplot ENROLL CATHOL
    gnuplot ENROLL INCOME
    gnuplot ENROLL COLLEGE
    boxplot INCOME REGION --factorized
end gpbuild
gridplot MyPlots --output=display
```
Matrix of subplots.