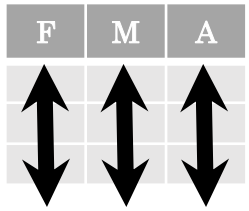
Data Wrangling

with pandas Cheat Sheet

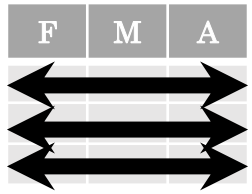
http://pandas.pydata.org

Tidy Data – A foundation for wrangling in p

In a tidy data set:



&



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

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

Tidy data complements pandas's ve
operations. pandas will automatical
observations as you manipulate vari
other format works as intuitively wit

Syntax – Creating DataFrames

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

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```

Specify values for each column.

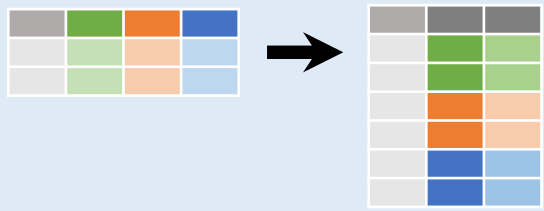
```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```

Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
```

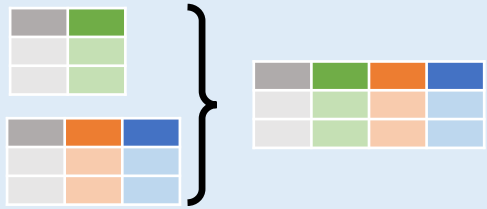
Reshaping Data – Change the layout of



pd.melt(df)
Gather columns into rows.

df.pivot(columns='var', values='val')
Spread rows into columns.

pd.concat([df1,df2])
Append rows of DataFrames



pd.concat([df1,df2], axis=1)
Append columns of DataFrames

df.s
Orc

df.s
Orc

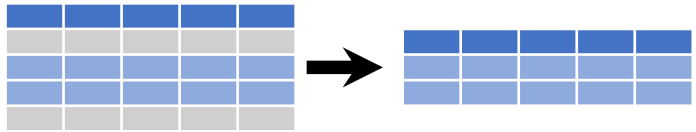
df.r
Re

df.s
Sor

df.r
Res
ind

df.c
Dr

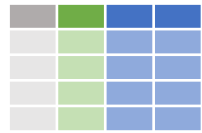
Subset Observations (Rows)



df[df.Length > 7]
Extract rows that meet logical

df.sample(frac=0.5)
Randomly select fraction of rows.

Subset V



df[['width','length
Select multiple columns
df['width'] *or* **df.**

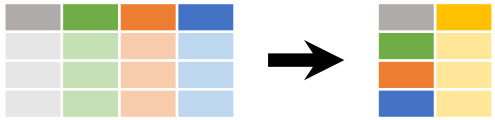
Summarize Data

`df['w'].value_counts()`
Count number of rows with each unique value of variable

`len(df)`
# of rows in DataFrame.

`df['w'].nunique()`
# of distinct values in a column.

`df.describe()`
Basic descriptive statistics for each column (or GroupBy)



pandas provides a large set of **summary functions** that operate on different kinds of pandas objects (DataFrame columns, Series, GroupBy, Expanding and Rolling (see below)) and produce single values for each of the groups. When applied to a DataFrame, the result is returned as a pandas Series for each column. Examples:

`sum()`
Sum values of each object.

`count()`
Count non-NA/null values of each object.

`median()`
Median value of each object.

`quantile([0.25,0.75])`
Quantiles of each object.

`apply(function)`
Apply function to each object.

`min()`
Minimum value in each object.

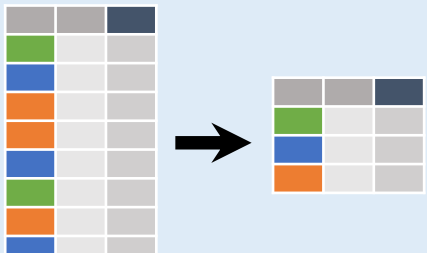
`max()`
Maximum value in each object.

`mean()`
Mean value of each object.

`var()`
Variance of each object.

`std()`
Standard deviation of each object.

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,

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.

`rank(method='dense')`

`shift(-1)`
Copy with values lagged by 1.

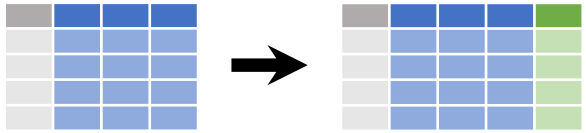
`cumsum()`

Handling Missing Data

`df.dropna()`
Drop rows with any column having NA/null data.

`df.fillna(value)`
Replace all NA/null data with value.

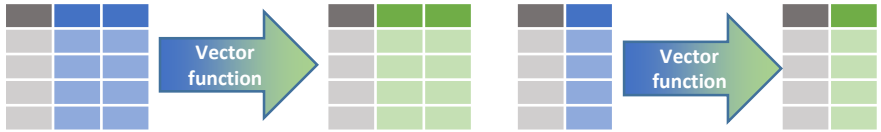
Make New Columns



`df.assign(Area=lambda df: df.Length*df.Height)`
Compute and append one or more new columns.

`df['Volume'] = df.Length*df.Height*df.Depth`
Add single column.

`pd.qcut(df.col, n, labels=False)`
Bin column into n buckets.



pandas provides a large set of **vector functions** that operate on all columns of a DataFrame or a single selected column (a pandas Series). These functions produce vectors of values for each of the columns, or a single Series for the individual Series. Examples:

`max(axis=1)`
Element-wise max.

`min(axis=1)`
Element-wise min.

`clip(lower=-10,upper=10)`
Trim values at input thresholds

`abs()`
Absolute value.

Com

adf

x1	x2
A	1
B	2
C	3

Standard Joins

x1	x2	x3
A	1	T
B	2	F
C	3	NaN

pd

x1	x2	x3
A	1.0	T
B	2.0	F
D	NaN	T

pd

x1	x2	x3
A	1	T
B	2	F

pd

x1	x2	x3
A	1	T
B	2	F
C	3	NaN
D	NaN	T

pd

Filtering Joins

x1	x2
A	1
B	2

ad

x1	x2
C	3

ad

ydf

x1	x2
A	1