pyladies
MADRID

Pyladies, 17th January 2018

MADRID

INTRODUCTION TO PANDAS

Who am I?

PhD in Physics

Data scientist @
BBVA Data&Analytics

Graph Analytics
Complex Systems
General Risk assesment models

@DataLane

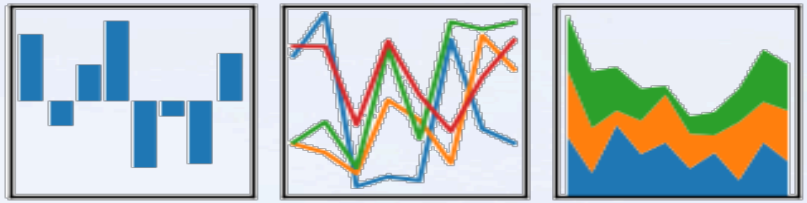
BBVA

DATA & ANALYTICS

Outline

- Motivation
- Pandas Data Structures: DataSeries & DataFrames
- Pandas tutorial
    - Reading and writing data: multiple formats
    - Selecting and indexing data
    - Data aggregation
    - Data mangling
    - Data visualisation
- Pandas Demo: case study

Motivation: Why pandas?



- Python:
  - opensource
  - readability,
  - large community of developers
- Fast numpy arrays (memory efficient, numerics library)
- Intuitive DataFrames from R (indexes + column labels)
- Other:
  - groupby like in SQL.
  - Input-output capabilities

Wes McKinney

Features ketplace Pricing This repository Search Sign in or Sign up

pandas-dev /

Watch 778 Star 12,486 Fork 4,849

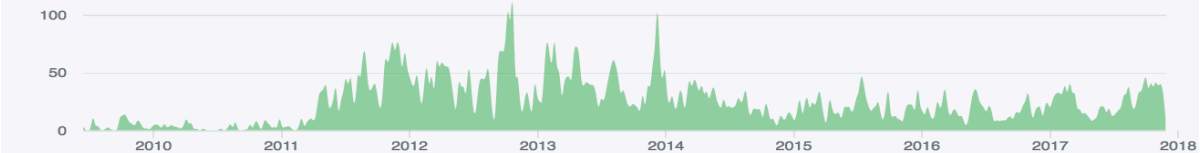
Code Issues 2,187 Pull requests 96 Projects 1 Wiki Insights

- Pulse
- Contributors
- Commits
- Code frequency
- Dependency graph
- Network
- Forks

Aug 2, 2009 – Jan 16, 2018

Contributions: Commits

Contributions to master, excluding merge commits

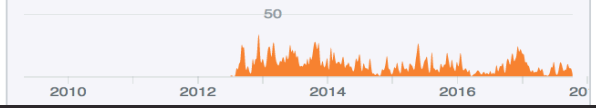


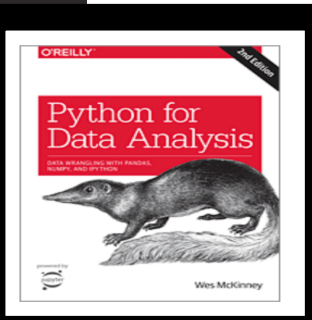
wesm #1

2,993 commits 458,980 ++ 340,580 --

jreback #2

2,646 commits 473,427 ++ 375,871 --





1000 contributors, 16000 commits

Pandas @ BBVA Data

conda create --name “your_env” python=3
source activate “your_env”

- Data preparation for machine learning
- Feature engineering

- Spark RDD’s → spark dataframes : gently moving to pd dataframe sintaxis by using alias such as .astype()

- Pyspark:
- Convert Spark DataFrame to Pandas: toPandas()
- Create a Spark DataFrame from Pandas: createDataFrame(pandas_df)

Pandas Series

- Series is a one-dimensional labeled array capable of holding any data type. The axis labels are the index.

```
s = Series(data, index=index)
```

Data can be: dict / ndarray / scalar

- Properties
  - Series is ndarray-like¶
    s[s > s.median()]
    s[0]
  - Series is dict-like¶
    s[‘category’]
  - Series admits vectorized operations¶
    s*2

Pandas Dataframes

- Pandas DataFrame: dict-like container of series.
- 2-dimensional, heterogeneous tabular data structure with labeled axes (rows and columns).
- Each column can have a different type
- Along with the data, you can pass index (row labels) and columns (column labels).
- Size mutable : insert and delete columns

DataFrame.from_records
DataFrame.from_dict
DataFrame.from_csv

Load and write data: I/O module

One line of code: very flexible and customisable!

```
pandas.read_csv(filepath_or_buffer, sep=', ', delimiter=None, header='infer', names=None, index_col=None,
usecols=None, squeeze=False, prefix=None, mangle_dupe_cols=True, dtype=None, engine=None,
converters=None, true_values=None, false_values=None, skipinitialspace=False, skiprows=None, nrows=None,
na_values=None, keep_default_na=True, na_filter=True, verbose=False, skip_blank_lines=True,
parse_dates=False, infer_datetime_format=False, keep_date_col=False, date_parser=None, dayfirst=False,
iterator=False, chunksize=None, compression='infer', thousands=None, decimal=b'.', lineterminator=None,
quotechar='"', quoting=0, escapechar=None, comment=None, encoding=None, dialect=None, tupleize_cols=None,
error_bad_lines=True, warn_bad_lines=True, skipfooter=0, skip_footer=0, doublequote=True,
delim_whitespace=False, as_recarray=None, compact_ints=None, use_unsigned=None, low_memory=True,
buffer_lines=None, memory_map=False, float_precision=None)                                    [source]
```

Read CSV (comma-separated) file into DataFrame

```
DataFrame.to_csv(path_or_buf=None, sep=', ', na_rep='', float_format=None, columns=None, header=True,
index=True, index_label=None, mode='w', encoding=None, compression=None, quoting=None, quotechar='"',
line_terminator='\n', chunksize=None, tupleize_cols=None, date_format=None, doublequote=True,
escapechar=None, decimal='.')                                                                 [source]
```

Write DataFrame to a comma-separated values (csv) file

Load and write data: I/O module

pd.read_excel()
pd.read_json()
pd.read_sql()
pd.read_parquet()
pd.read_hdfs()

pd.read_pickle() : useful to avoid loading headers.

Read directly from a URL!!!
Read from compressed formats

Describing the data

```
movies.index
```

```
RangeIndex(start=0, stop=979, step=1)
```

```
movies.columns
```

```
Index(['star_rating', 'title', 'content_rating', 'genre', 'duration',
       'actors_list'],
      dtype='object')
```

```
movies.dtypes
```

```
star_rating        float64
title               object
content_rating      object
genre               object
duration             int64
actors_list         object
dtype: object
```

```
movies.describe()
```

	star_rating	duration
count	979.000000	979.000000
mean	7.889785	120.979571
std	0.336069	26.218010
min	7.400000	64.000000
25%	7.600000	102.000000
50%	7.800000	117.000000
75%	8.100000	134.000000
max	9.300000	242.000000

Accesing the data

```
movies['content_rating'].head(5)
```

```
0          R
1          R
2          R
3      PG-13
4          R
Name: content_rating, dtype: object
```

```
movies[['content_rating','genre']].head(5)
```

	content_rating	genre
0	R	Crime
1	R	Crime
2	R	Crime
3	PG-13	Action
4	R	Crime

```
type(movies['content_rating']),type(movies[['content_rating','genre']])
```

```
(pandas.core.series.Series, pandas.core.frame.DataFrame)
```

Indexing

- Index: label of a series.
- Can be reset or replaced.
- One or more dimensiones
- It may not be unique

	star_rating	title	content_rating	genre
0	9.3	The Shawshank Redemption	R	Crime
1	9.2	The Godfather	R	Crime
2	9.1	The Godfather: Part II	R	Crime
3	9.0	The Dark Knight	PG-13	Action
4	8.9	Pulp Fiction	R	Crime
5	8.9	12 Angry Men	NOT RATED	Drama
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure
8	8.9	Schindler's List	R	Biography
9	8.9	Fight Club	R	Drama

```
movies.loc[4:6,:]
```

	star_rating	title	content_rating	genre
0	9.3	The Shawshank Redemption	R	Crime
1	9.2	The Godfather	R	Crime
2	9.1	The Godfather: Part II	R	Crime
3	9.0	The Dark Knight	PG-13	Action
4	8.9	Pulp Fiction	R	Crime
5	8.9	12 Angry Men	NOT RATED	Drama
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure
8	8.9	Schindler's List	R	Biography
9	8.9	Fight Club	R	Drama

```
movies.loc[4:6,"content_rating"]
```

	star_rating	title	content_rating	genre
0	9.3	The Shawshank Redemption	R	Crime
1	9.2	The Godfather	R	Crime
2	9.1	The Godfather: Part II	R	Crime
3	9.0	The Dark Knight	PG-13	Action
4	8.9	Pulp Fiction	R	Crime
5	8.9	12 Angry Men	NOT RATED	Drama
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure
8	8.9	Schindler's List	R	Biography
9	8.9	Fight Club	R	Drama

```
movies.iloc[4:6]
```

	star_rating	title	content_rating	genre
0	9.3	The Shawshank Redemption	R	Crime
1	9.2	The Godfather	R	Crime
2	9.1	The Godfather: Part II	R	Crime
3	9.0	The Dark Knight	PG-13	Action
4	8.9	Pulp Fiction	R	Crime
5	8.9	12 Angry Men	NOT RATED	Drama
6	8.9	The Good, the Bad and the Ugly	NOT RATED	Western
7	8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure
8	8.9	Schindler's List	R	Biography
9	8.9	Fight Club	R	Drama

```
movies.iloc[4:6,2:4]
```

Reseting the index

Selection returns copy of DataFrame

star_rating	title	content_rating	genre
9.3	The Shawshank Redemption	R	Crime
9.2	The Godfather	R	Crime
9.1	The Godfather: Part II	R	Crime
9.0	The Dark Knight	PG-13	Action
8.9	Pulp Fiction	R	Crime
8.9	12 Angry Men	NOT RATED	Drama
8.9	The Good, the Bad and the Ugly	NOT RATED	Western
8.9	The Lord of the Rings: The Return of the King	PG-13	Adventure
8.9	Schindler's List	R	Biography
8.9	Fight Club	R	Drama

```
movies.set_index('star_rating').iloc[4:6]
```

star_rating	title	content_rating	genre	duration
8.9	Pulp Fiction	R	Crime	154
8.9	12 Angry Men	NOT RATED	Drama	96

```
movies.set_index('star_rating').loc[4:6]
```

star_rating	title	content_rating	genre	duration	actors_list

Filtering rows

```
cond=movies.star_rating >= 9.1
cond.head(5)
```

```
0     True
1     True
2     True
3    False
4    False
Name: star_rating, dtype: bool
```

```
movies[cond]
```

	star_rating	title	content_rating	genre	duration	actors_list	new_genre
0	9.3	The Shawshank Redemption	R	Crime	142	[u'Tim Robbins', u'Morgan Freeman', u'Bob Gunt...	Crime
1	9.2	The Godfather	R	Crime	175	[u'Marlon Brando', u'Al Pacino', u'James Caan']	Crime
2	9.1	The Godfather: Part II	R	Crime	200	[u'Al Pacino', u'Robert De Niro', u'Robert Duv...	Crime

Filtering rows

```
cond=movies.star_rating >= 9.
cond2=movies.genre == "Action"
cond2.head(5)
```

```
0    False
1    False
2    False
3     True
4    False
Name: genre, dtype: bool
```

```
movies[cond&cond2]
```

	star_rating	title	content_rating	genre	duration	actors_list	new_genre
3	9.0	The Dark Knight	PG-13	Action	152	[u'Christian Bale', u'Heath Ledger', u'Aaron E...	Action

Transforming dataframes

df.drop(cols_to_drop, axis=1) df[['col1']].drop_duplicates() df[['col1', 'col2']].drop_duplicates()	→ Series → Rows → Rows
df.rename(columns={'col1': 'col_one'})	Accept “inplace” parameter to avoid creating a copy of the dataframe.
df.sample(n=10) df.sample(frac=0.01)	Data sampling
df.fillna(0) df.dropna() df2['four'].fillna('missing') df.fillna(method='pad', limit=1)	Cleaning / filling missing data¶ : NaN describe() and groupby ignore NaN We normaly impute them : fillna()
df.replace(1.5, np.nan) df.replace(999999, 0, inplace=True)	Numeric replacement
df[col].astype(‘int’)	Changing the data type
df[col].get_dummies()	One-hot-encoding

Merge, join, concatenate

`DataFrame.merge`(*right*, *how='inner'*, *on=None*, *left_on=None*, *right_on=None*, *left_index=False*, *right_index=False*, *sort=False*, *suffixes=('_x', '_y')*, *copy=True*, *indicator=False*, *validate=None*) [source]

Merge DataFrame objects by performing a database-style join operation by columns or indexes.

`pandas.concat`(*objs*, *axis=0*, *join='outer'*, *join_axes=None*, *ignore_index=False*, *keys=None*, *levels=None*, *names=None*, *verify_integrity=False*, *copy=True*) [source]

Concatenate pandas objects along a particular axis with optional set logic along the other axes.

Dataframe groupby

Split the data into groups		grouped=df.groupby([‘A’,’B’])
Apply a function to each group	Aggregate: group mean / size Transform: fill NAs within groups Filter: discard groups with few members	grouped.aggregate(np.sum) grouped.sum() grouped.aggregate({‘sum’:np.sum, ‘count’: np.size})
Combine the results		After combining, split variable is an index: we can .reset_index()

Groupby + unstack()

```
movies_grouped=movies.g
movies_grouped
```

split

```
r(['genre','star_rating_bin']).star_rating
```

apply + combine

```
genre        star_rating_bin
Action       5                57
             4                54
             3                18
             2                 6
             1                 1
Adventure    5                24
             4                33
             3                14
             2                 4
Animation    5                20
             4                28
             3                13
             2                 1
Biography    5                31
             4                34
             3                 9
             2                 3
```

```
movies_grouped.unstack(1)
```

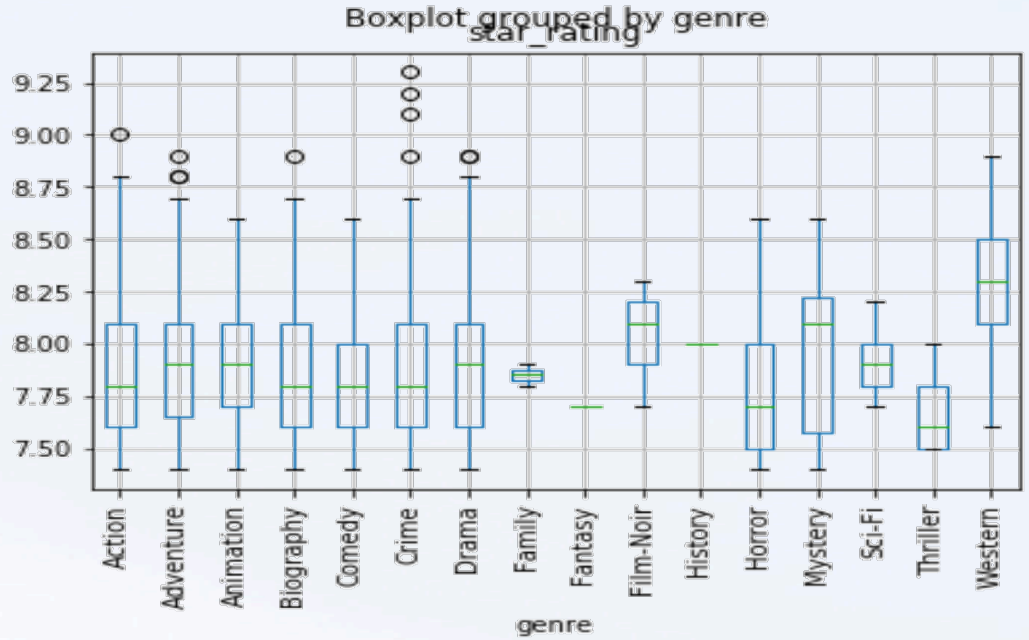
star_rating_bin	5	4	3	2	1
genre					
Action	57.0	54.0	18.0	6.0	1.0
Adventure	24.0	33.0	14.0	4.0	NaN
Animation	20.0	28.0	13.0	1.0	NaN
Biography	31.0	34.0	9.0	3.0	NaN
Comedy	77.0	55.0	21.0	3.0	NaN
Crime	52.0	44.0	20.0	5.0	3.0
Drama	105.0	117.0	46.0	10.0	NaN
Family	NaN	2.0	NaN	NaN	NaN

Data Visualization

`DataFrame.boxplot(column=None, by=None, ax=None, fontsize=None, rot=0, grid=True, figsize=None, layout=None, return_type=None, **kwds)` [source]

```
movies.boxplot(by='genre',column=['star_rating'],rot=90)
```

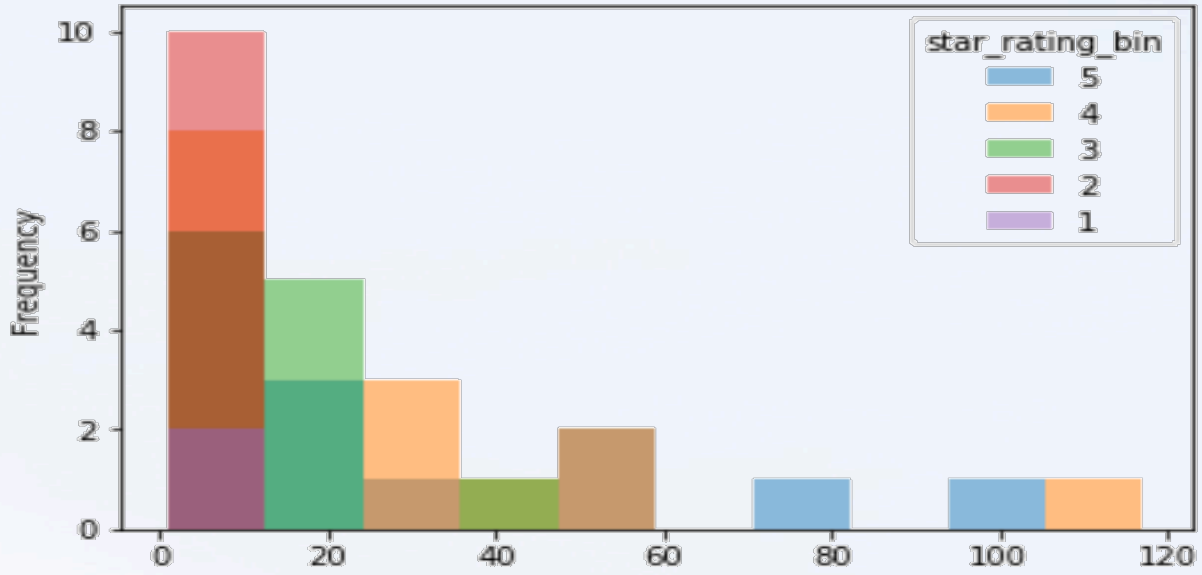
```
<matplotlib.axes._subplots.AxesSubplot at 0x10b5e8b00>
```



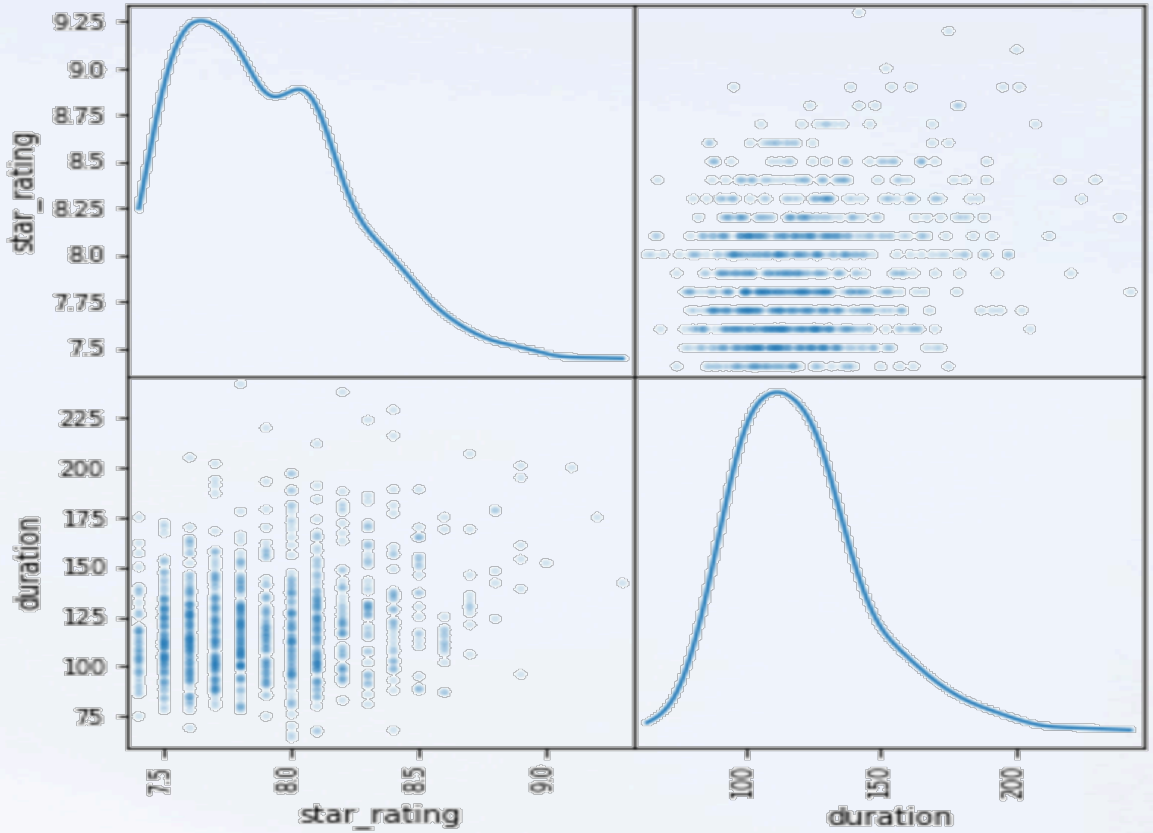
Data Visualization

```
%matplotlib inline
movies_grouped.unstack(1).plot.hist(alpha=0.5)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10b2a8240>
```



Data Visualization



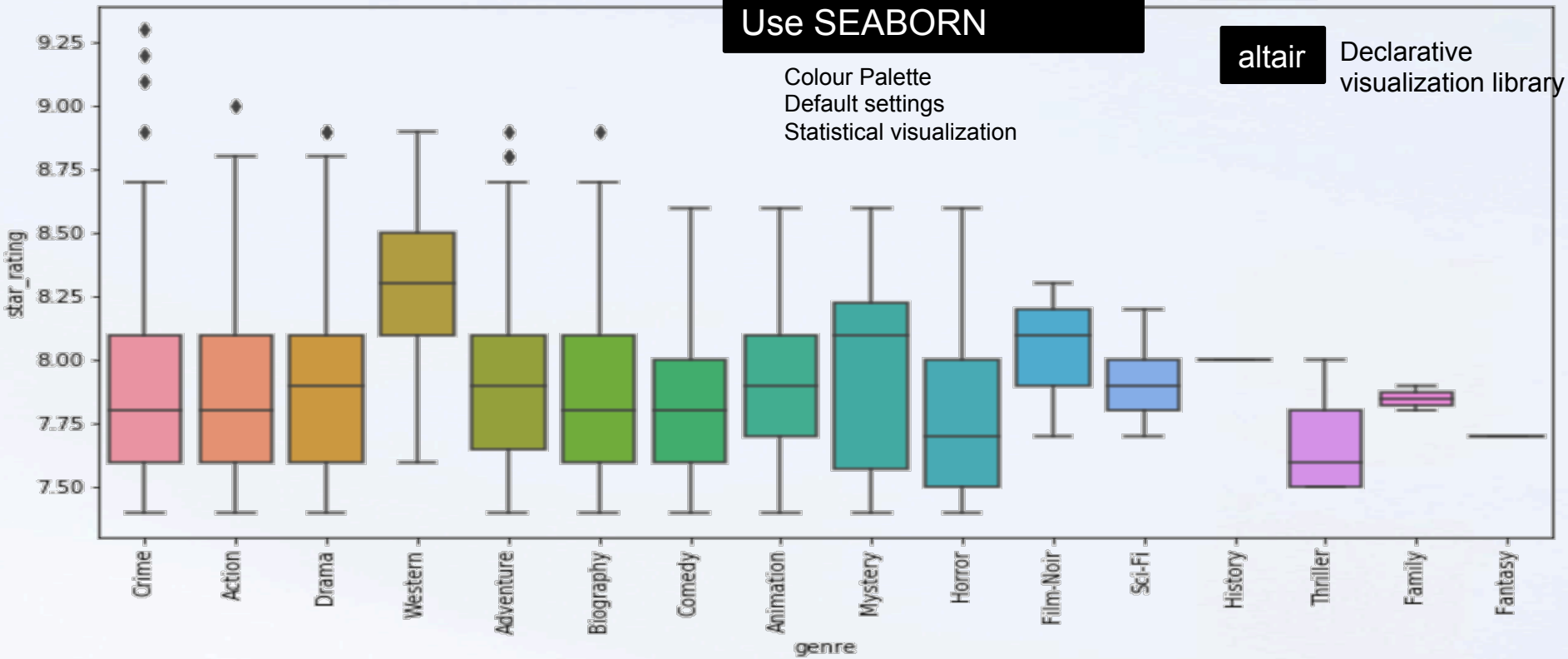
Data Visualization

bokeh Interactivity

Use SEABORN

altair Declarative visualization library

Colour Palette
Default settings
Statistical visualization



Tutorials

CodeBasics

Data School Pandas Tutorial

10 minutes to pandas

Modern pandas

Pandas documentation

Case Study