On the Diagramatic Diagnosis of Data

Tools to make your data analysis and machine learning both easier and more reliable

Ian Ozsvald, PyConUK 2018

- http://ianozsvald.com (http://ianozsvald.com)
- @ianozsvald

Ian's background

- Senior data science coach (Channel 4, Hailo, QBE Insurance)
- Author of High Performance Python (O'Reilly)
- Co-founder of PyDataLondon meetup (8,000+ members) and conference (5 years old)
- Past speaker (Random Forests and ML Diagnostics) at PyConUK
- Blog - http://ianozsvald.com (http://ianozsvald.com)

We'll cover

- Google Facets
- Pandas pivot_table and styling
- Pandas Profiling
- Seaborn
- `discover_feature_relationships`
- The proposed "Data Stories" at the end might make you more confident when presenting your own ideas for investigation

Google Facets

- https://pair-code.github.io/facets/ (https://pair-code.github.io/facets/)
- Handles strings and numbers from CSVs
- 1d and up to 4d plots (!)

Facets overview (1D)

Sort by
Feature order

Reverse order

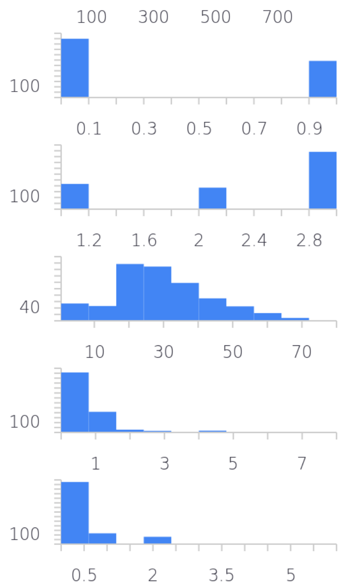
Feature search

Features: int(6) float(2) string(4)

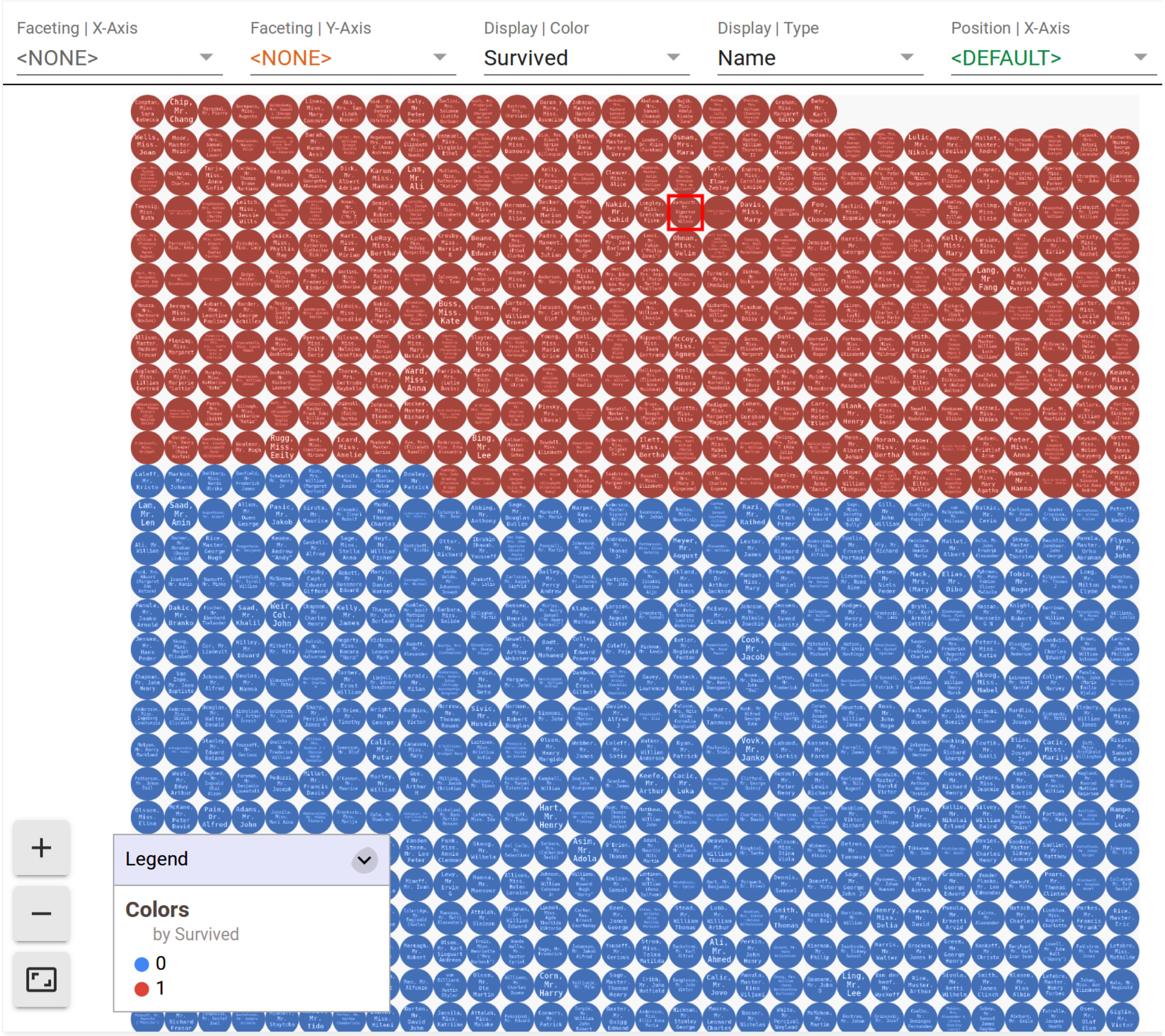
Numeric Features (8)

Chart to show
Standard
log expand

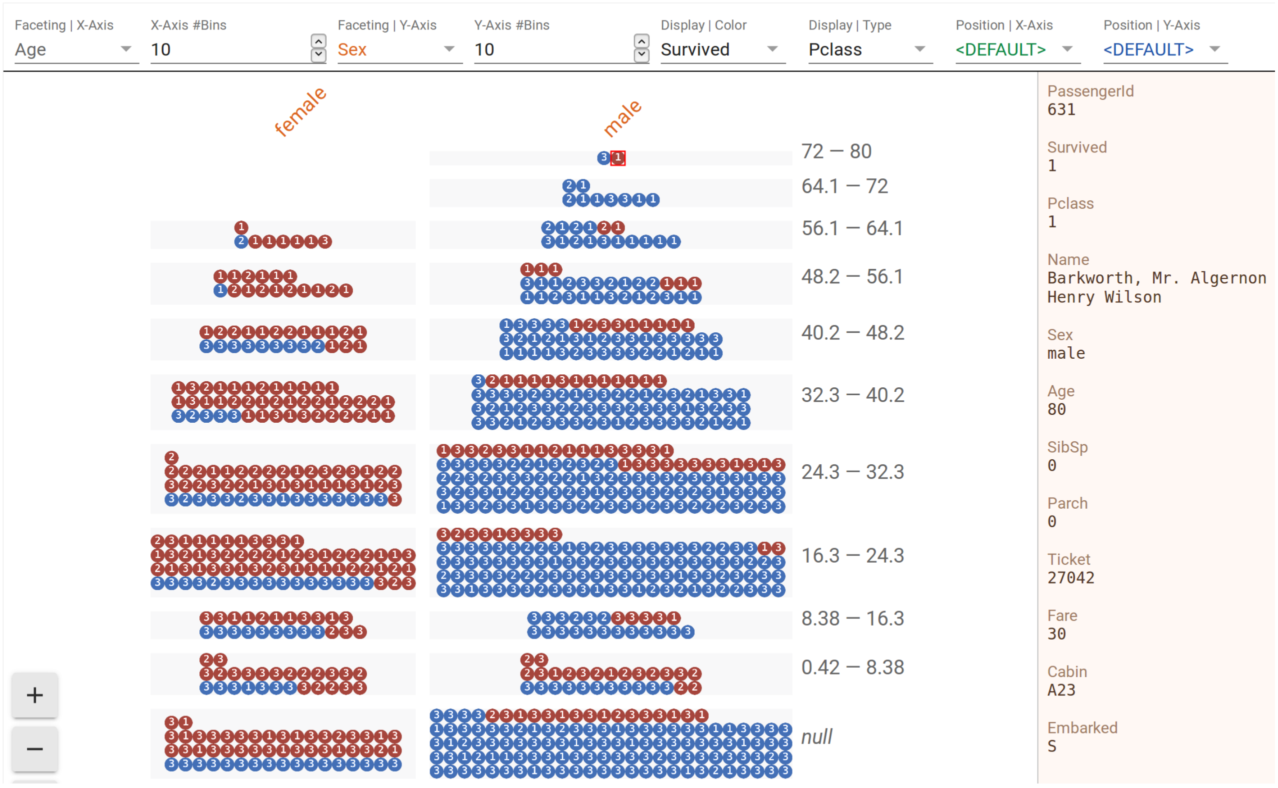
	count	missing	mean	std dev	zeros	min	median	max
Survived	891	0%	0.38	0.49	61.62%	0	0	1
Pclass	891	0%	2.31	0.84	0%	1	3	3
Age	714	19.87%	29.7	14.53	0%	0.42	28	80
SibSp	891	0%	0.52	1.1	68.24%	0	0	8
Parch	891	0%	0.38	0.81	76.09%	0	0	6



Facets Dive (2D)



Facets Dive (4D)



Facets

- Non-programmatic (you can't clean or add columns)
- You can upload your own CSV files after you add new features
- Interactivity is nice

Pandas pivot_table and styling

- Cut numeric columns into labeled bins
- Pivot_table to summarise
- Apply styling to add colours
- See [https://github.com/datapythonista/towards_pandas_1/blob/master/Towards%20pandas%201.0.ipynb (https://github.com/datapythonista/towards_pandas_1/blob/master/Towards%20pandas%201.0.ipynb)](https://github.com/datapythonista/towards_pandas_1/blob/master/Towards%20pandas%201.0.ipynb)
    - Via [https://twitter.com/datapythonista (https://twitter.com/datapythonista)](https://twitter.com/datapythonista)

In [4]:
```
titanic['age_'] = titanic.Age.fillna(titanic.Age.median())

titanic['has_family_'] = (titanic.Parch + titanic.SibSp) > 0
titanic.has_family_.value_counts()
```

Out[4]:
```
False    537
True     354
Name: has_family_, dtype: int64
```

In [5]:
```
titanic['age_labeled_'] = pd.cut(titanic['age_'],
                        bins=[titanic.age_.min(), 18, 40, titanic.age_.max()],
                        labels=['Child', 'Young', 'Over_40'])
titanic['age_labeled_'].value_counts()
```

Out[5]:
```
Young      602
Over_40    150
Child      138
Name: age_labeled_, dtype: int64
```

In [6]:
```
titanic[['Survived', 'Pclass', 'age_labeled_']].head(10)
```

Out[6]:

	Survived	Pclass	age_labeled_
PassengerId			
1	0.0	3	Young
2	1.0	1	Young
3	1.0	3	Young
4	1.0	1	Young
5	0.0	3	Young
6	0.0	3	Young
7	0.0	1	Over_40
8	0.0	3	Child
9	1.0	3	Young
10	1.0	2	Child

In [7]:
```
df_pivot = titanic.pivot_table(values='Survived', columns='Pclass', index='age_labe
led_', aggfunc='mean')
df_pivot
```

Out[7]:

Pclass	1	2	3
age_labeled_			
Child	0.875000	0.793103	0.344086
Young	0.669355	0.421488	0.232493
Over_40	0.513158	0.382353	0.075000

In [8]:

```
df_pivot = df_pivot.rename_axis('', axis='columns')
df_pivot = df_pivot.rename('Class {}'.format, axis='columns')
df_pivot.style.format('{:.2%}')
```

Out[8]:

	Class 1	Class 2	Class 3
age_labeled_			
Child	87.50%	79.31%	34.41%
Young	66.94%	42.15%	23.25%
Over_40	51.32%	38.24%	7.50%

In [9]:
```
# https://pandas.pydata.org/pandas-docs/stable/style.html
def highlight_max(s):
    '''
    highlight the maximum in a Series yellow.
    '''
    is_max = s == s.max()
    return ['background-color: yellow' if v else '' for v in is_max]

df_pivot.style.format('{:.2%}') \
        .apply(highlight_max, axis=1) \
        .set_caption('Survival rates by class and age')
```

Out[9]:

Survival rates by class and age

	Class 1	**Class 2**	**Class 3**
age_labeled_			
Child	87.50%	79.31%	34.41%
Young	66.94%	42.15%	23.25%
Over_40	51.32%	38.24%	7.50%

Pivot table and styling benefits

- Summarise relationships visually
- Highlight (and give background colours) to call out results
- Push the resulting DataFrame into a Seaborn `heatmap` (not shown) for a `.png` export

Pandas Profiling

- https://github.com/pandas-profiling/pandas-profiling (https://github.com/pandas-profiling/pandas-profiling)
- Take a look at the exported html: http://localhost:8000/titanic_pp.html (http://localhost:8000/titanic_pp.html)
- Add the exported html artefact to your source control

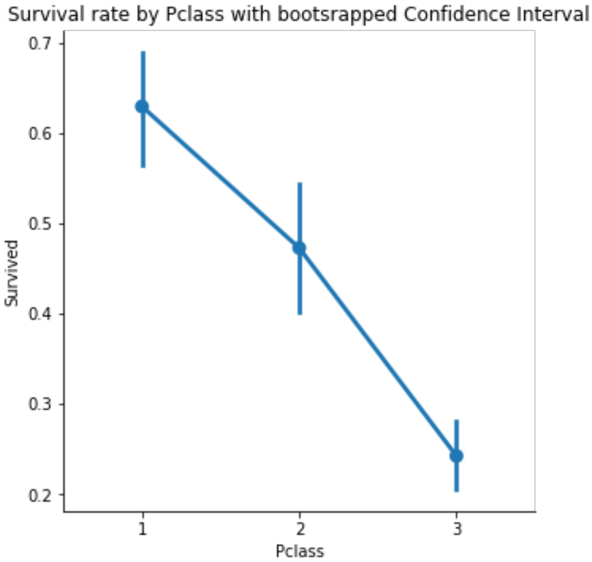
```
# report in the Notebook
pp.ProfileReport(titanic)

# report to an html file (i.e. generate an artefact)
profile = pp.ProfileReport(titanic)
profile.to_file(outputfile="./titanic_pp.html")
```

Seaborn

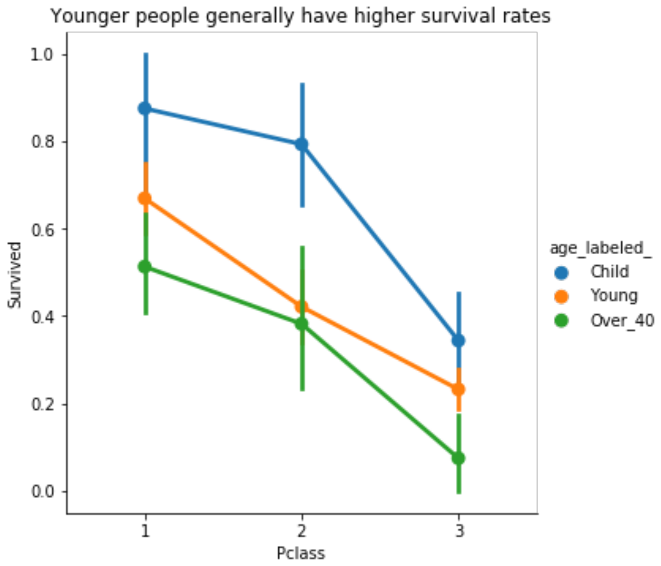
- Additional statistical plots on top of matplotlib and Pandas' own
- See https://www.kaggle.com/ravaliraj/titanic-data-visualization-and-ml (https://www.kaggle.com/ravaliraj/titanic-data-visualization-and-ml)

In [11]:

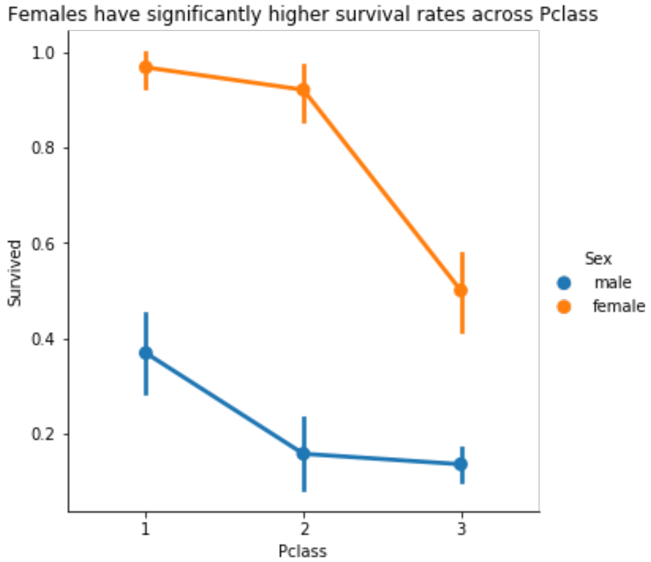
```
fg = sns.catplot('Pclass', 'Survived', data=titanic, kind='point')
fg.ax.set_title("Survival rate by Pclass with bootsrapped Confidence Interval");
```



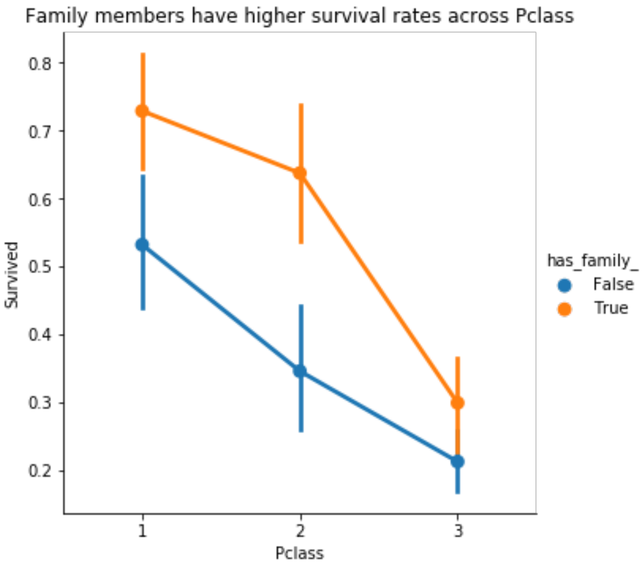
In [12]:
```
fg = sns.catplot('Pclass', 'Survived', data=titanic, hue='age_labeled_', kind='point');
fg.ax.set_title("Younger people generally have higher survival rates");
```



In [13]:
```
fg = sns.catplot('Pclass', 'Survived', data=titanic, hue='Sex', kind='point');
fg.ax.set_title("Females have significantly higher survival rates across Pclass");
```



In [14]:
```
fg = sns.catplot('Pclass', 'Survived', data=titanic, hue='has_family_', kind="point
");
fg.ax.set_title("Family members have higher survival rates across Pclass");
```



Seaborn benefits

- Visualise pivot-table results
- Clearly show 3D relationships
- Work using the DataFrame that you're manipulating (with new features and cleaner data)

Seaborn on the Boston dataset

- See also aplunket.com/data-exploration-boston-data-part-2/
- Smarter 2D scatter, rug and hex plots

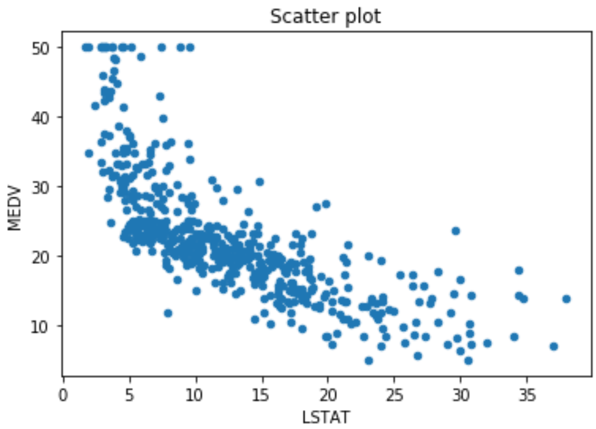
In [15]:

```python
from sklearn.datasets import load_boston
boston_data = load_boston()
boston = pd.DataFrame(boston_data.data, columns=boston_data.feature_names)
boston['MEDV'] = boston_data.target
boston.head()
```

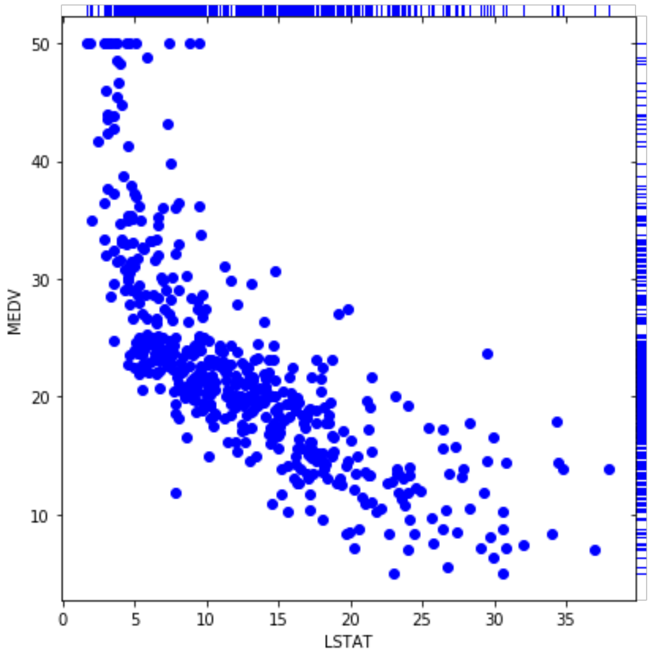
Out[15]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TA
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222

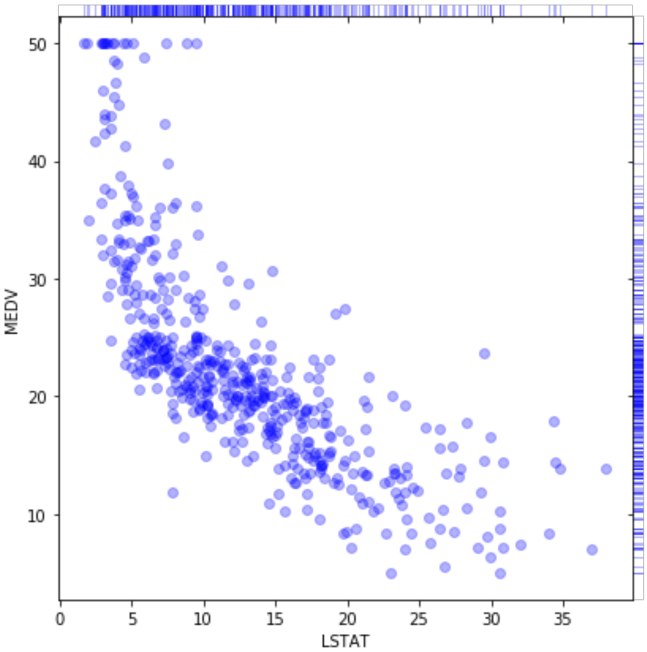
In [16]:
```
ax = boston[['LSTAT', 'MEDV']].plot(kind="scatter", x="LSTAT", y="MEDV");
ax.set_title("Scatter plot");
```



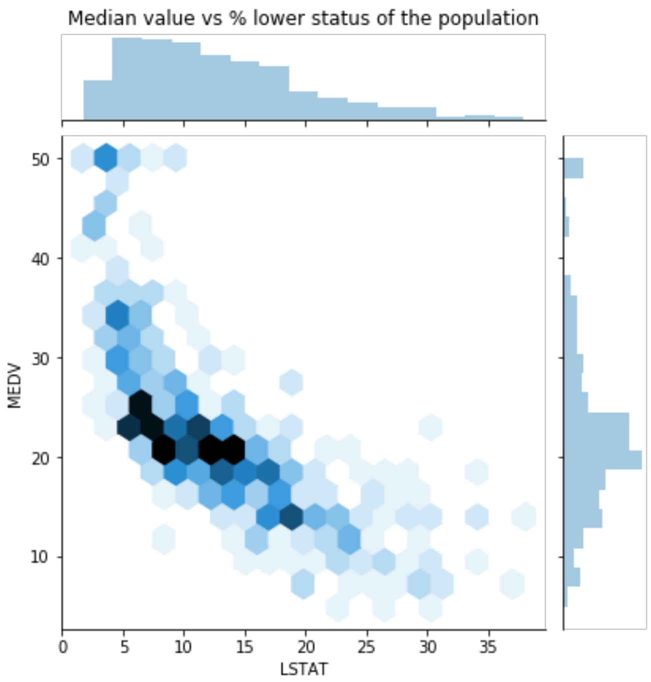
In [17]:
```
grid = sns.JointGrid(x='LSTAT', y='MEDV', data=boston, space=0, height=6, ratio=50)
grid.plot_joint(plt.scatter, color="b")
grid.plot_marginals(sns.rugplot, color="b", height=4);
```



In [18]:
```
grid = sns.JointGrid(x='LSTAT', y='MEDV', data=boston, space=0, height=6, ratio=50)
grid.plot_joint(plt.scatter, color="b", alpha=0.3)
grid.plot_marginals(sns.rugplot, color="b", height=4, alpha=.3);
```



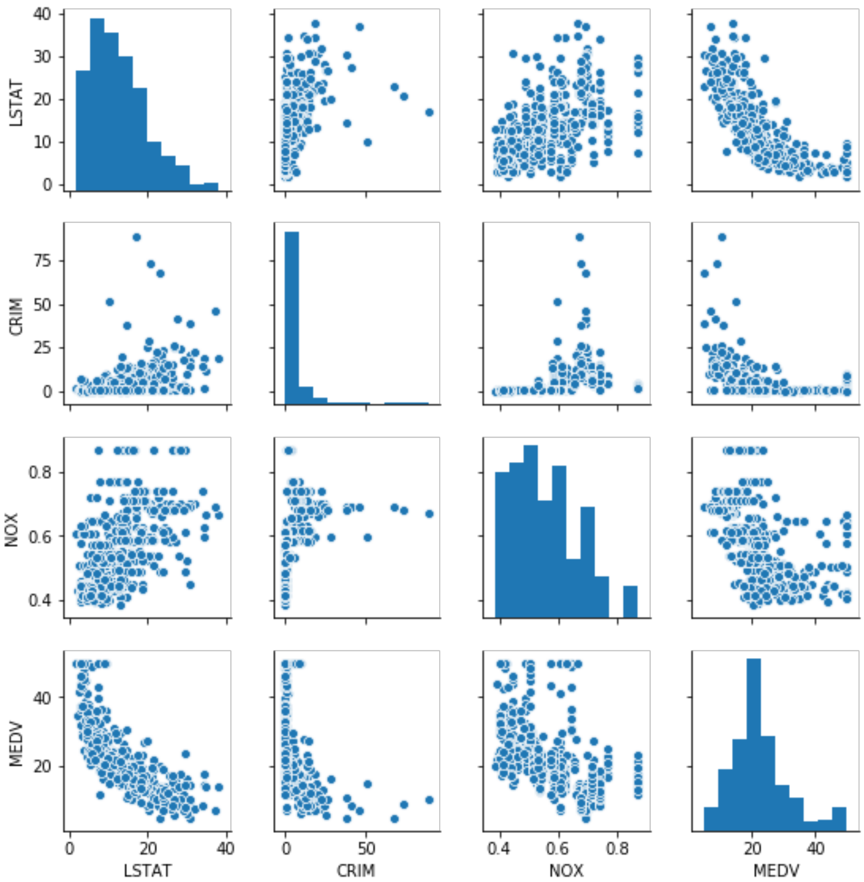
In [19]:
```
jg = sns.jointplot(boston.LSTAT, boston.MEDV, kind='hex')
jg.ax_marg_x.set_title("Median value vs % lower status of the population");
```



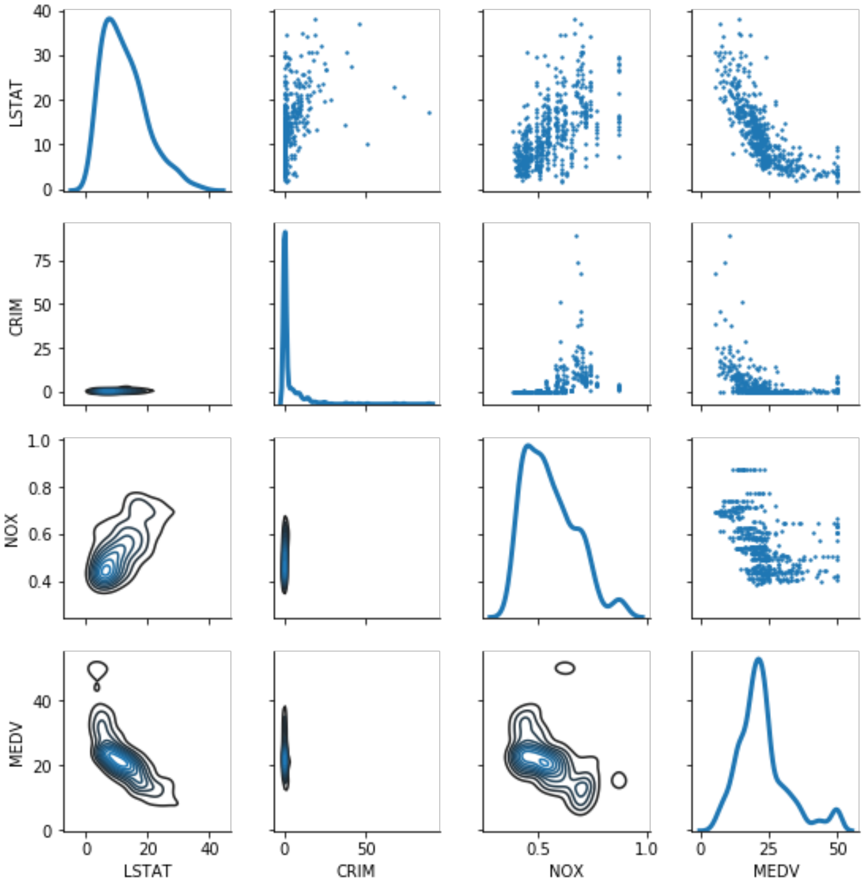
Pair plots

- Show scatter and kernel density (kde) plots for feature pairs
- See http://gael-varoquaux.info/interpreting_ml_tuto/content/01_how_well/02_cross_validation.html (http://gael-varoquaux.info/interpreting_ml_tuto/content/01_how_well/02_cross_validation.html)

In [20]:
```
boston_smaller = boston[['LSTAT', 'CRIM', 'NOX', 'MEDV']]
sns.pairplot(boston_smaller, height=2);
```



In [21]:
```
g = sns.PairGrid(boston_smaller, diag_sharey=False, height=2)
g.map_lower(sns.kdeplot)
g.map_upper(plt.scatter, s=2)
g.map_diag(sns.kdeplot, lw=3);
```



discover_feature_relationships

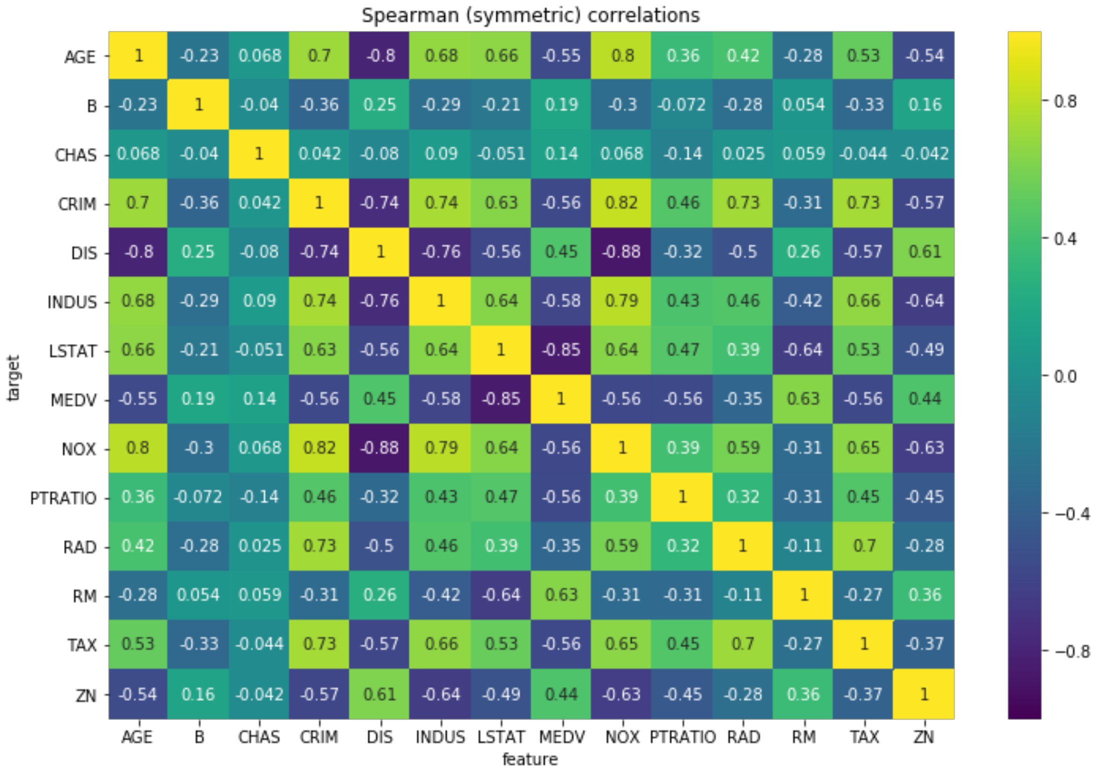
- Which features predict other features?
  - What relationships exist between all pairs of single columns?
  - Could we augment our data if we know the underlying relationships?
  - Can we identify poorly-specified relationships?
- Go beyond Pearson and Spearman correlations (but we can do these too)
- https://github.com/ianozsvald/discover_feature_relationships/ (https://github.com/ianozsvald/discover_feature_relationships/)

In [29]:

```
cols = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
classifier_overrides = set() # classify these columns rather than regress (in Boston everything can be regressed)
%time df_results = discover.discover(boston[cols].sample(frac=1), classifier_overrides, method="spearman")
```

```
CPU times: user 872 ms, sys: 0 ns, total: 872 ms
Wall time: 867 ms
```

In [31]:
```
fig, ax = plt.subplots(figsize=(12, 8))
sns.heatmap(df_results.pivot(index='target', columns='feature', values='score').fillna(1),
            annot=True, center=0, ax=ax, vmin=-1, vmax=1, cmap="viridis");
ax.set_title("Spearman (symmetric) correlations");
```



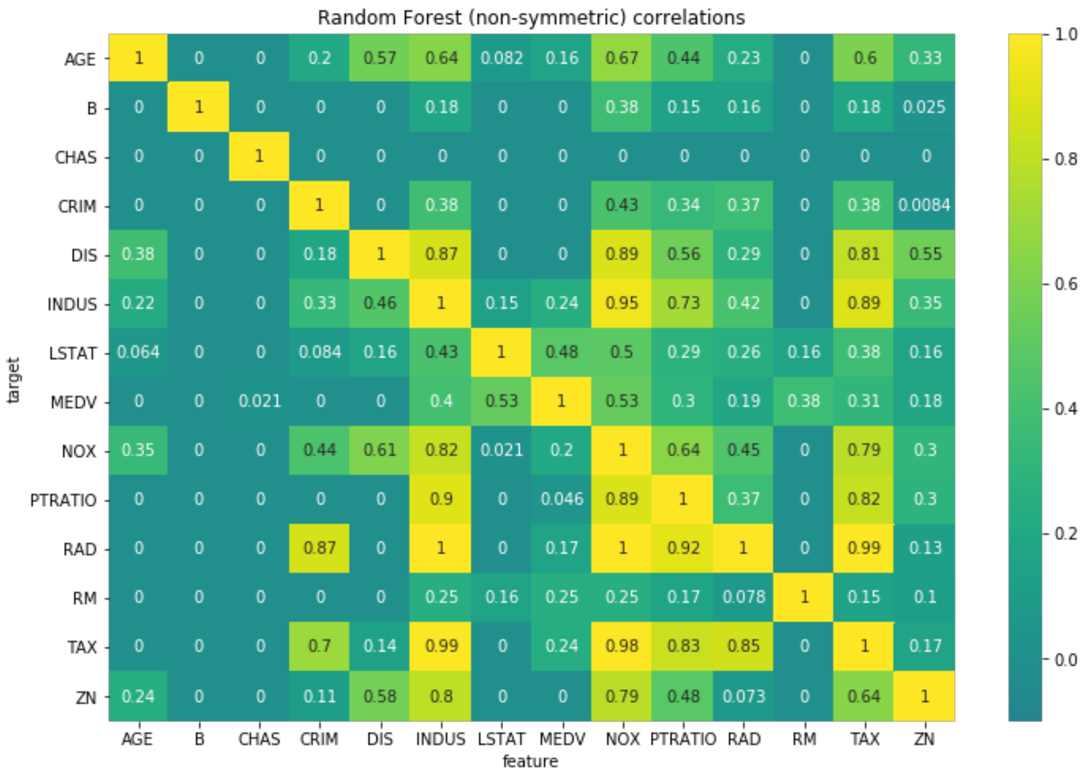
In [26]:

```
%time df_results = discover.discover(boston[cols].sample(frac=1), classifier_overrides)
```

```
CPU times: user 16.3 s, sys: 5.49 s, total: 21.7 s
Wall time: 1min 25s
```

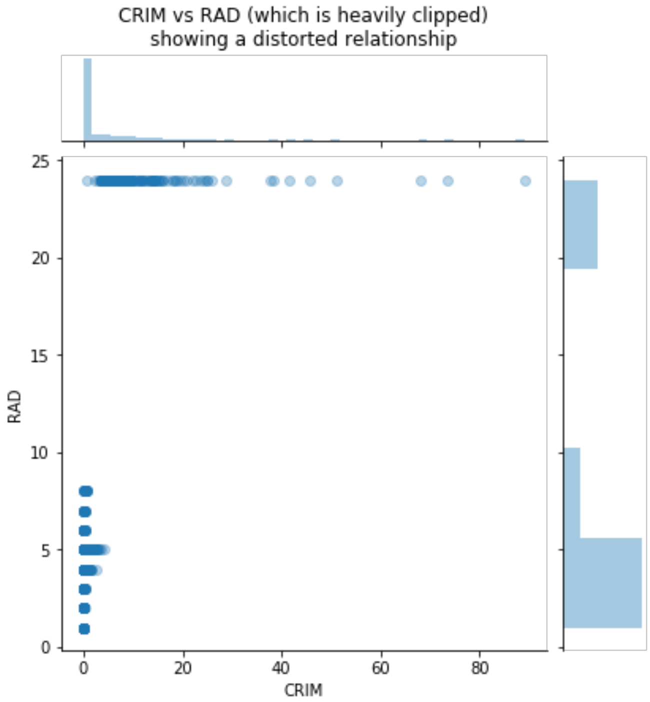
In [27]:

```
# CRIM predicts RAD but RAD poorly predicts CRIM - why?
# MEDV (target) is predicted by NOX, NOX is predicted by INDUS - could we get anyth
ing further by improving this?
fig, ax = plt.subplots(figsize=(12, 8))
sns.heatmap(df_results.pivot(index='target', columns='feature', values='score').cli
p_lower(0).fillna(1),
            annot=True, center=0, ax=ax, vmin=-0.1, vmax=1, cmap="viridis");
ax.set_title("Random Forest (non-symmetric) correlations");
```



In [43]:

```
# RAD figures are clipped which distorts the relationship with CRIM!
# we've identified some dodgy data - maybe we could look for better data sources?
jg = sns.jointplot(boston.CRIM, boston.RAD, alpha=0.3)
jg.ax_marg_x.set_title("CRIM vs RAD (which is heavily clipped)\nshowing a distorted relationship");
```



Data Stories

- Proposed by Bertil: https://medium.com/@bertil_hatt/what-does-bad-data-look-like-91dc2a7bcb7a (https://medium.com/@bertil_hatt/what-does-bad-data-look-like-91dc2a7bcb7a)
- A short report describing the data and proposing things we could do with it
- Use Facets and Pandas Profiling to describe the main features
- Use `discover_feature_relationships` and `PairGrid` to describe interesting relationships
- Note if there are parts of the data we don't trust (time ranges? sets of columns?)
  - Bonus - take a look at the `missingno` missing number library
- Propose experiments that we might run on this data which generate a benefit
- This presentation is a *Jupyter Notebook* in *presentation mode* (i.e. a source controlled code artefact)

Conclusion

- We've looked at a set of tools that enable Python engineers and data scientists to review their data
- Looking beyond 2D correlations we might start to dig further into our data's relationships
- A Data Story will help colleagues to understand what can be achieved with this data
- See my `Data Science Delivered` repo on github.com/ianozsvald
- Did you learn something? I love receiving postcards! Please email me and I'll send you my address
- Please try my tool - I'd love feedback: https://github.com/ianozsvald/discover_feature_relationships (https://github.com/ianozsvald/discover_feature_relationships)
- Please come to a PyData event and please thank your fellow volunteers here

Ian Ozsvald (http://ianozsvald.com (http://ianozsvald.com) , http://twitter.com/ianozsvald (http://twitter.com/ianozsvald))