What Is Exploratory Data Analysis?

> *But as much as EDA is a set of tools, it's also a mindset. And that mindset is about your relationship with the data… EDA happens between you and the data and isn't about proving anything to anyone else yet.*

- Cathy O'Neil (Doing Data Science)

What Is Exploratory Data Analysis?

- Developed at Bell Labs in the 1960's by John Tukey
- Techniques used to visualize and summarize data
  - Five-number summary: `describe()`
  - Distributions: box plots, stem and leaf, histogram, scatterplot

Goals of Exploratory Data Analysis

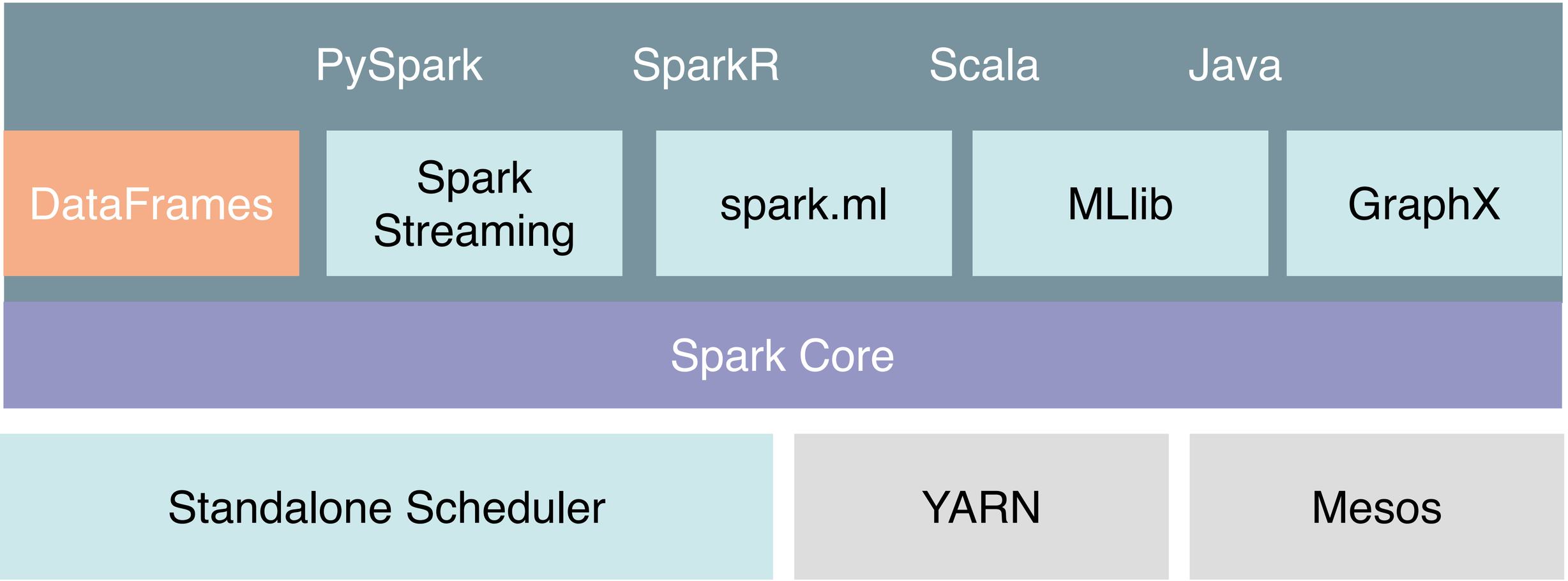
- Gain greater intuition
- Validate our data (consistency and completeness)
- Make comparisons between distributions
- Find outliers
- Treat missing data
- Summarize data (a statistic -> one number that represents many #'s)

How Can Spark Help?

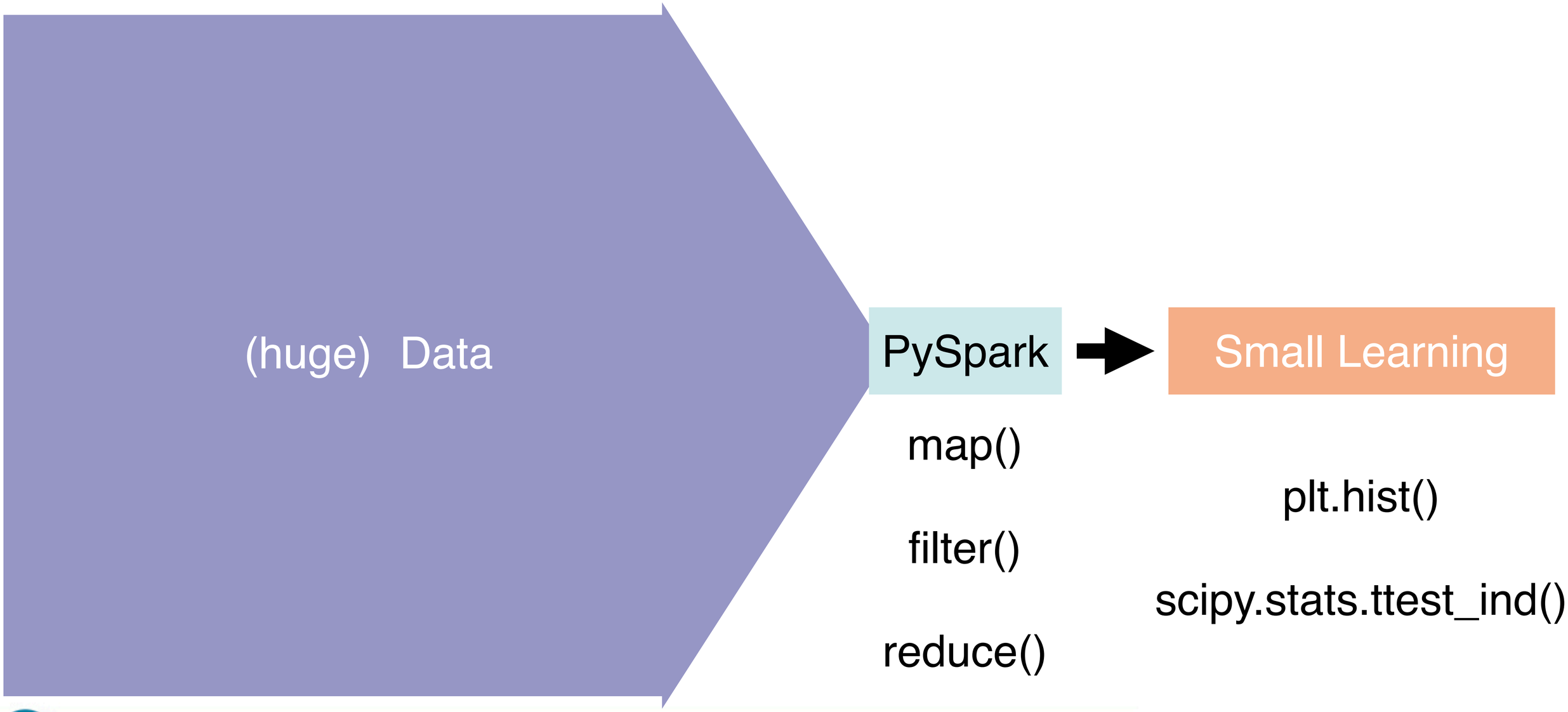
- Interactive REPL
- Rapid computation (especially aggregates) on large amounts of data
- High level abstractions for querying data
- “Condense” data for easier local exploration and visualization

Common Questions

- How many records in total are there?
- How many uniques values does each column contain?
- How many missing (or null) values are there?
- For numeric columns, what are its summary statistics
- What are values appear most often in each column?
- How are the values of each column distributed?


PySpark
SparkR
Scala
Java
DataFrames
Spark Streaming
spark.ml
MLlib
GraphX
Spark Core
Standalone Scheduler
YARN
Mesos

Big Data… Small Learning



Unique Values

- `rdd.distinct()`
- `rdd.countApproxDistinct(relative_accuracy)`

```
# HyperLogLog
rdd_dict.map(lambda row: row['_schoolid']).countApproxDistinct()
```

```
62989L
```

```
rdd_dict.map(lambda row: row['_schoolid']).distinct().count()
```

```
61402
```

http://dx.doi.org/10.1145/2452376.2452456

http://content.research.neustar.biz/blog/hll.html

Missing Values

- `column.isNull()`
- `dataframe.dropna(column_name)`
- `dataframe.fillna()`
- `dataframe.replace(to_replace, value)`
- `pyspark.accumulators`

https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrameNaFunctions

Missing Values

- `column.isNull()`
- `dataframe.fillna()`

```
df.filter(df_dates['students_reached'].isNull()).select('students_reached', 'funding_status').collect()
```

```
[Row(students_reached=None, funding_status=u'completed'),
 Row(students_reached=None, funding_status=u'completed'),
 Row(students_reached=None, funding_status=u'expired'),
 Row(students_reached=None, funding_status=u'completed'),
                            ...
```

```
df_no_null = df.fillna(0, ['students_reached'])
```

Frequently Occurring Values

- `dataframe.freqItems(columns, support)`

Note: this is an approximate algorithm that **always** returns all the frequent items, but may contain false positives.

required minimum proportion of rows

```
freq_items = df.freqItems(['school_city', 'primary_focus_area', \
                           'grade_level', 'poverty_level','resource'], 0.7).collect()
```

```
freq_items[0]
```

```
Row(school_city_freqItems=[u'Los Angeles'], primary_focus_area_freqItems=[u'Literacy & Language'], grade_level_freqIt
ems=[u'Grades PreK-2'], poverty_level_freqItems=[u'highest poverty'], resource_freqItems=[u'Supplies'])
```

http://dl.acm.org/citation.cfm?doid=762471.762473

Summary Statistics

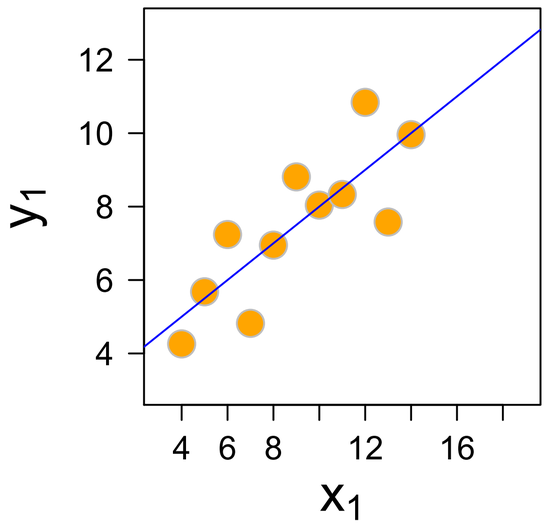
- `dataframe.describe(column_name)`

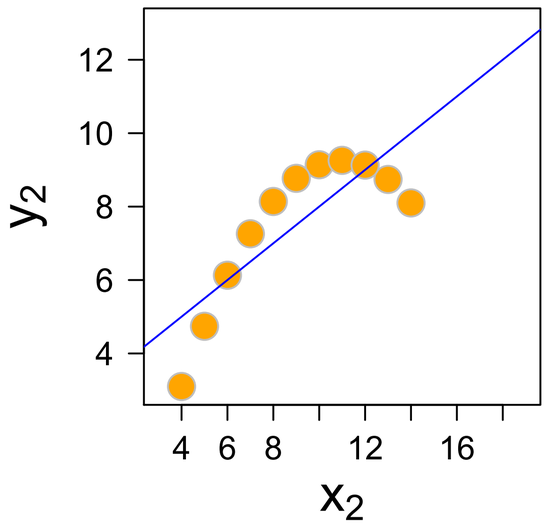
```
df.select('total_donations', 'num_donors', 'students_reached', \
                df_dates['total_price_excluding_optional_support'].alias('p_exclude'), \
                df_dates['total_price_including_optional_support'].alias('p_include')) \
  .describe().show()
```

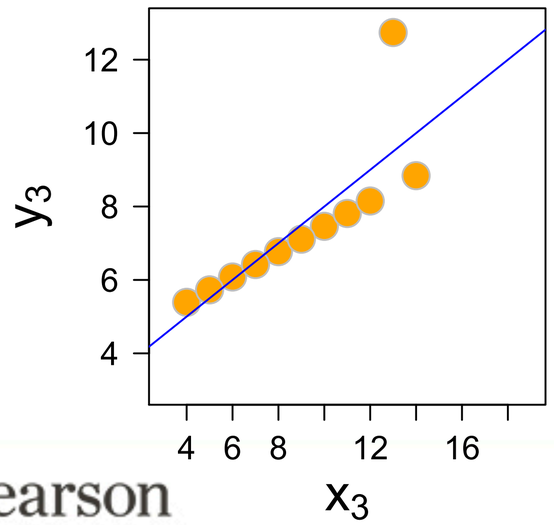
```
+-------+-----------------+-----------------+-----------------+------------------+-----------------+
|summary|  total_donations|       num_donors| students_reached|         p_exclude|         p_include|
+-------+-----------------+-----------------+-----------------+------------------+-----------------+
|  count|           771929|           771929|           771779|            771929|           771929|
|   mean|370.85023398481707|4.264279486843997|96.71620114048193| 569.6223687446723| 676.180708551764|
| stddev| 733.4647726421459|6.132976060232441|2118.592960253374|11763.955807309705|14344.347534777195|
|    min|              0.0|              0.0|              0.0|               0.0|              0.0|
|    max|         244778.0|            521.0|         999999.0|       1.0250017E7|       1.2500021E7|
+-------+-----------------+-----------------+-----------------+------------------+-----------------+
```

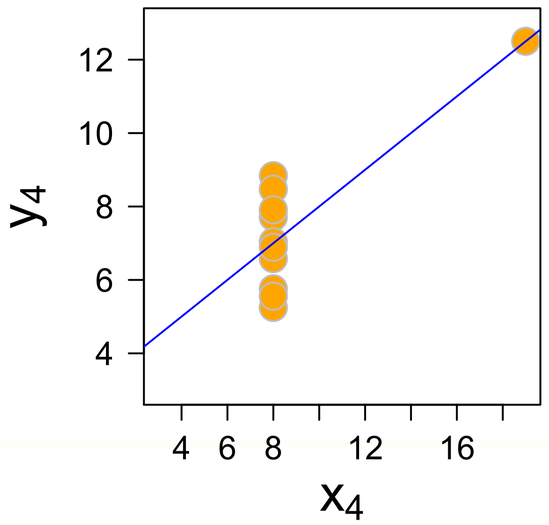
Interlude: Sometimes numbers aren’t enough!

Anscombe's Quartet









Mean (x)	9
Sample Variance (x)	11
Mean (y)	7.50
Sample Variance (y)	4.127
Correlation	0.816
Linear Regression	y = 3.00 + 0.500x

Distributions

RDD

- `rdd.histogram()`
- `rdd.stats()`

DataFrame

- `dataframe.groupby('column_name').count()`
- `dataframe.describe('column_name')`

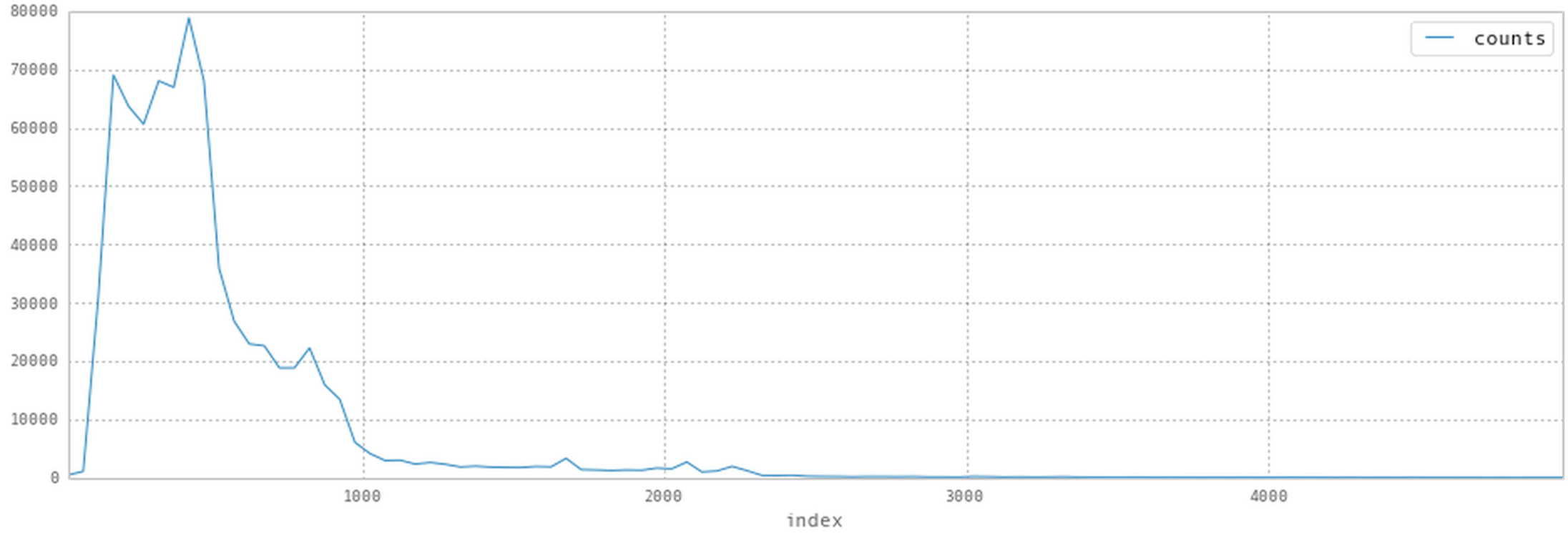
```
price_rdd = df_no_null.select('total_price_excluding_optional_support').rdd.map(lambda r: r.asDict().values()[0])
```

```
def plot_rdd_hist(hist):
    idx = []

    for i in range(len(hist[0]) - 1):
        idx.append((hist[0][i] + hist[0][i+1])/ 2)

    pd.DataFrame({'counts': hist[1], 'index': idx}).set_index('index').plot(figsize=(16,5))
```

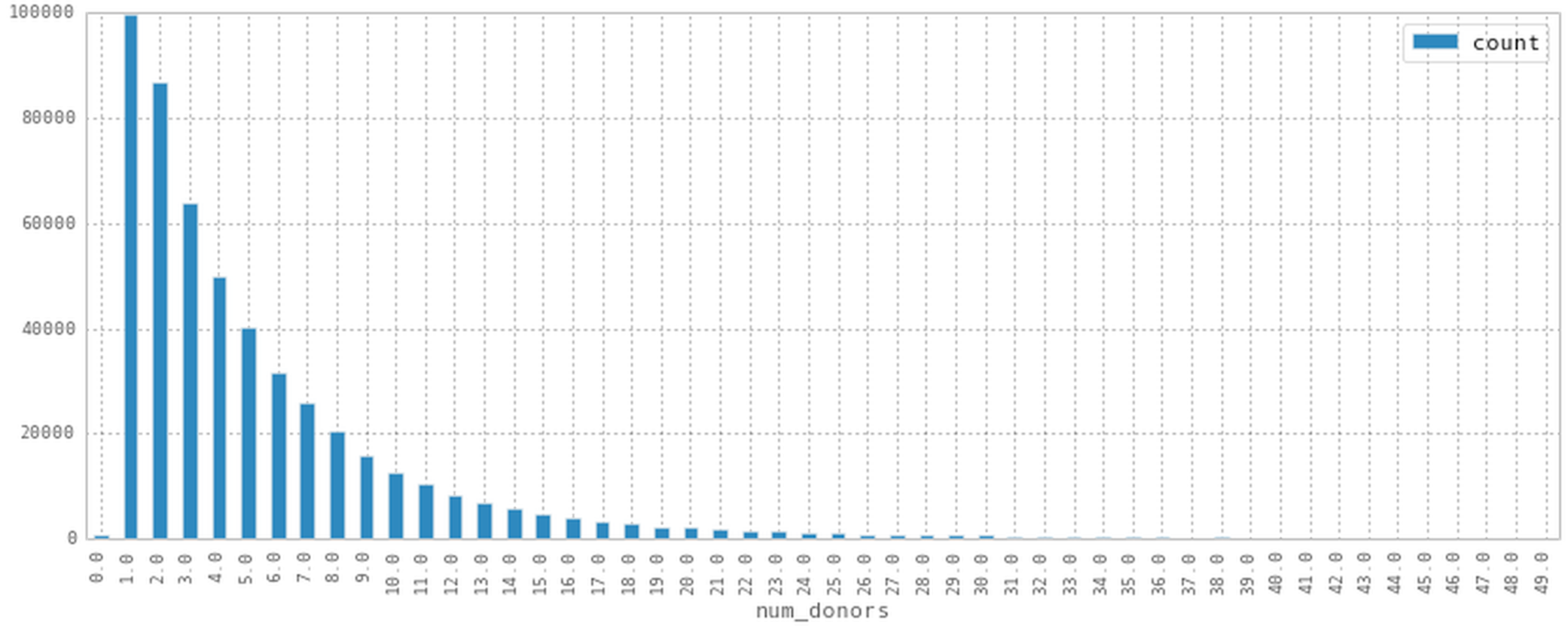
```
plot_rdd_hist(price_rdd.filter(lambda x: x < 5000).histogram(100))
```



```
def spark_histogram(df, column):
    donor_counts = df.groupby(column).count()
    donor_df = donor_counts.toPandas()
    donor_df[column] = donor_df.num_donors.astype(float)
    return donor_df.sort(column).set_index(column).iloc[:50,:].plot(kind='bar', figsize=(14,5))
```

```
spark_histogram(df_complete, 'num_donors')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x113b2be50>
```

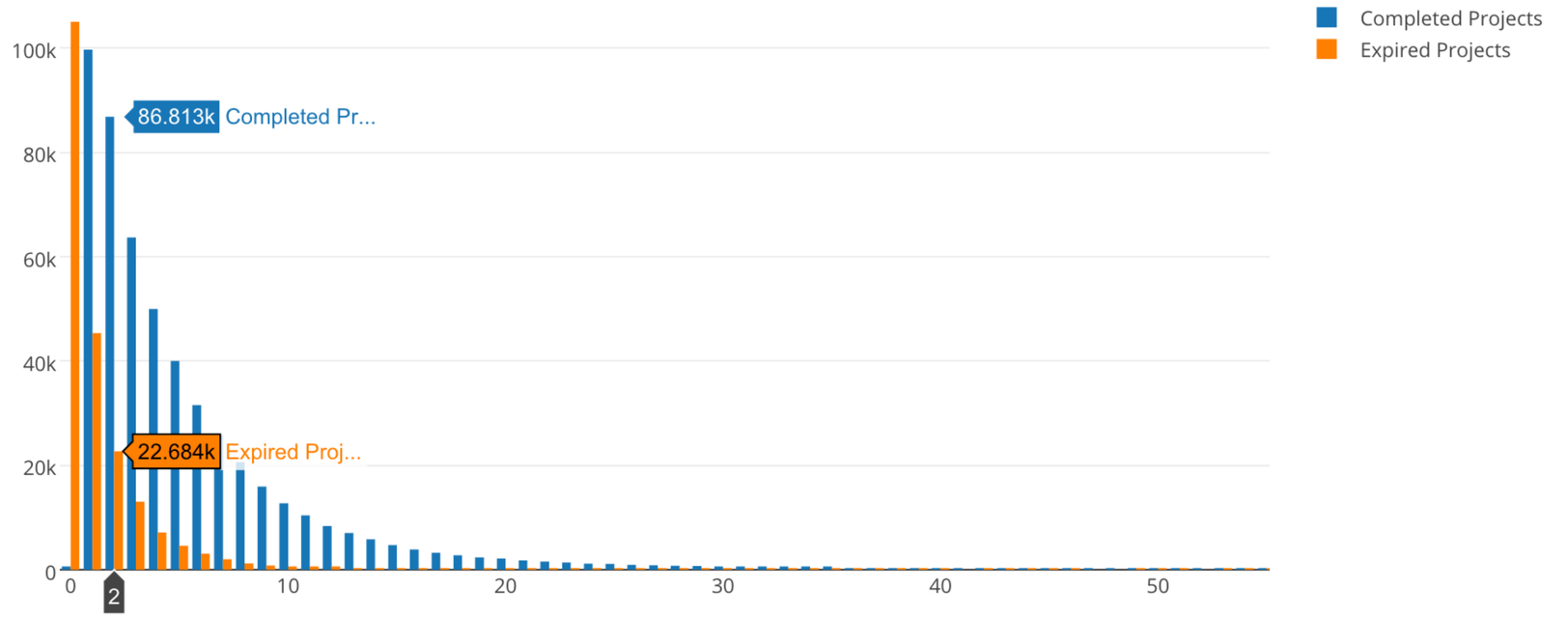


```
complete = df_complete.groupby('num_donors').count().toPandas()
expired = df_expired.groupby('num_donors').count().toPandas()
```

```
# And if we want an interactive plot, we can use a tool like plotly
# pip install plotly
import plotly.plotly as py
import plotly.tools as tls
from plotly.graph_objs import *

data = Data([
        Bar(x=complete['num_donors'], y= complete['count'], name="Completed Projects"),
        Bar(x=expired['num_donors'], y = expired['count'] , name="Expired Projects")
    ])

py.iplot(data)
```



Interactions

- dataframe.crosstab()
- dataframe.corr()

```
df_no_null.stat.corr('total_price_excluding_optional_support', 'num_donors')
```

```
0.007004254706419042
```

```
df_no_null.stat.corr('total_price_excluding_optional_support', 'students_reached')
```

```
0.0006159991686679948
```

```
df_no_null.stat.corr('total_price_excluding_optional_support', 'total_price_including_optional_support')
```

```
0.9999972199123168
```

```
df_dates.crosstab('resource_type', 'funding_status').show()
df_dates.crosstab('primary_focus_area', 'resource_type').show()
```

```
+----------------------------+-----+---------+-----------+-------+
|resource_type_funding_status| live|completed|reallocated|expired|
+----------------------------+-----+---------+-----------+-------+
|                        null|    2|       28|          0|     18|
|                       Other| 4542|    54610|        747|  22550|
|                       Books| 5982|   118810|       1527|  34554|
|                    Visitors|  102|      806|          6|    341|
|                    Supplies|11939|   185870|       2602|  63406|
|                       Trips|  347|     4381|         62|   1474|
|                  Technology|18957|   150500|       2256|  85510|
+----------------------------+-----+---------+-----------+-------+
```

```
+------------------------------+-----+--------+-----+----------+------+--------+----+
|primary_focus_area_resource_type|Trips|Visitors|Other|Technology| Books|Supplies|null|
+------------------------------+-----+--------+-----+----------+------+--------+----+
|              Literacy & Language|  630|     228|32795|    109605|127282|   75924|   4|
|                          null|    0|       0|    0|         0|     1|       0|  41|
|             Applied Learning| 1197|     104| 9429|     17869|  4863|   22596|   0|
|               Math & Science| 1902|     323|16353|     75189| 11746|   89101|   3|
|             Music & The Arts|  947|     441| 8305|     19289|  2883|   37804|   0|
|              Health & Sports|  159|      54| 4633|      3054|   432|   12970|   0|
|                 Special Needs|  241|      32| 7636|     19359|  4112|   17151|   0|
|             History & Civics| 1188|      73| 3298|     12858|  9554|    8271|   0|
+------------------------------+-----+--------+-----+----------+------+--------+----+
```

Review

- Interactive REPL
- Rapid computation (especially aggregates) on large amounts of data
- High level abstractions for efficient querying of data
- “Condense” data for easier local exploration and visualization

Natural Language Processing

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

[1, 3, 1, 1, 2, 0, 1, 0]
[0, 1, 4, 0, 0, 1, 1, 1]
[3, 0, 1, 1, 2, 2, 3, 2]
[0, 1, 1, 1, 0, 3, 2, 3]
[1, 2, 1, 2, 2, 0, 0, 0]
[1, 0, 1, 1, 0, 1, 1, 1]
[0, 2, 0, 0, 2, 2, 0, 0]
[1, 1, 1, 1, 0, 1, 1, 1]

The Unreasonable Effectiveness of Data

[Banko and Brill, 2001]

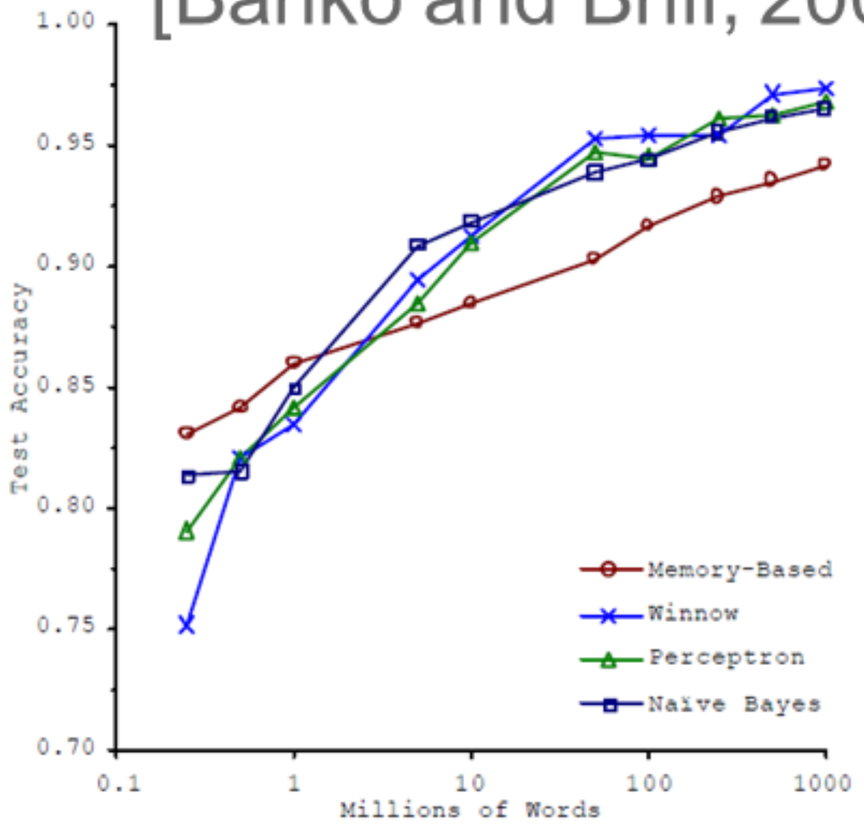


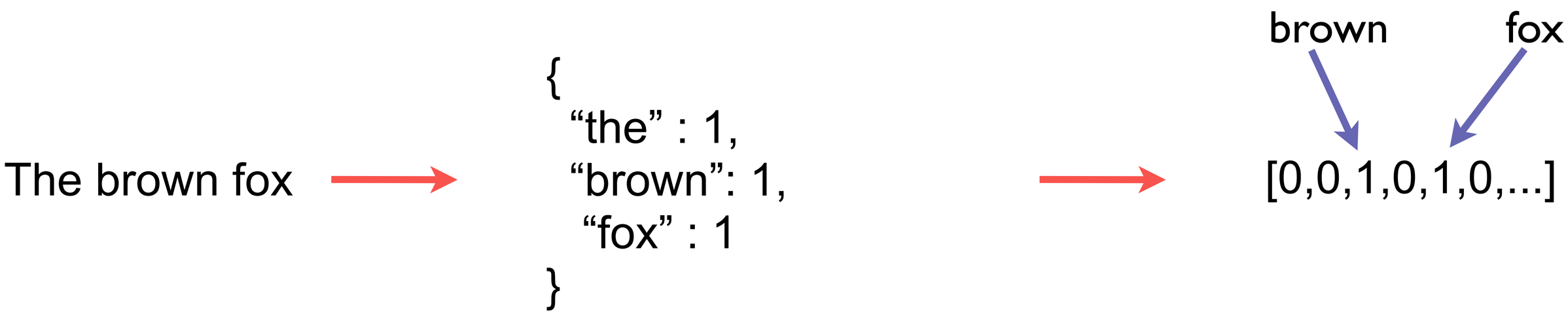
Figure 1. Learning Curves for Confusion Set Disambiguation

Source: https://www.microsoft.com/en-us/research/publication/scaling-to-very-very-large-corpora-for-natural-language-disambiguation/

Bag of Words

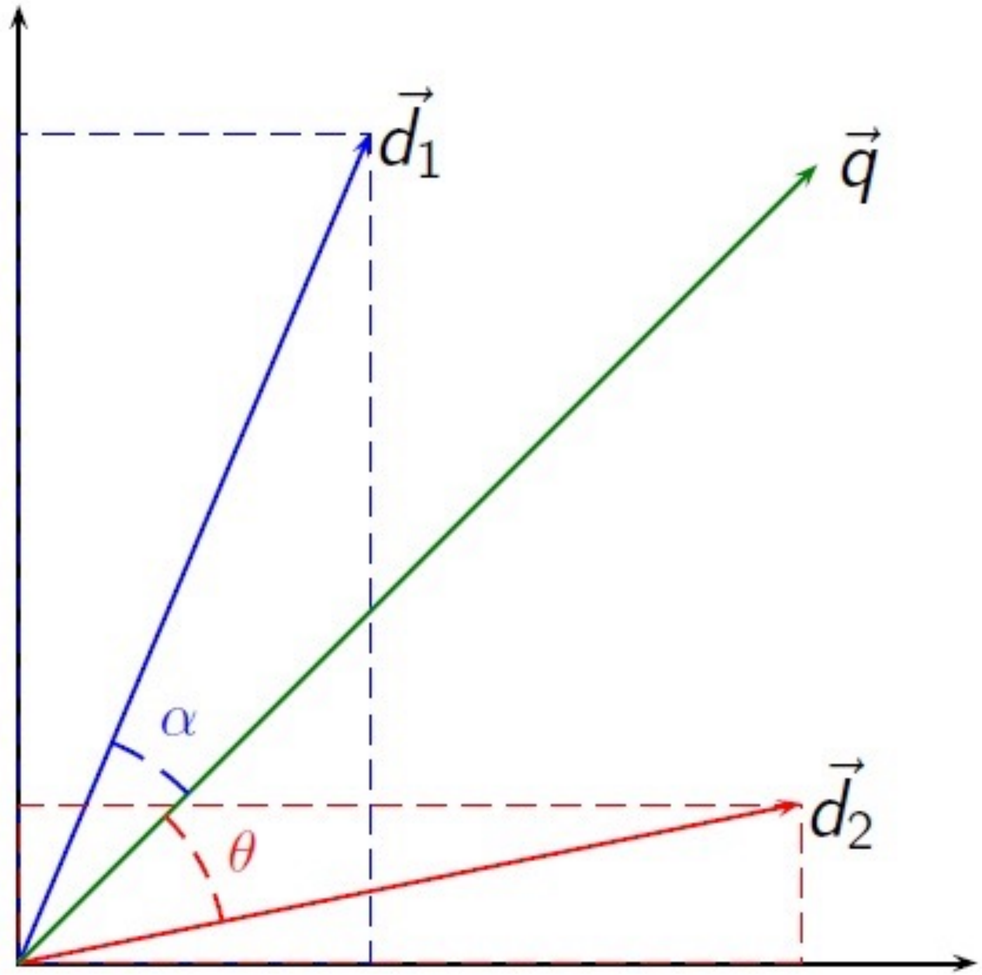
- **Document:** Single row of data/corpus
- **Corpus:** Entire set of all documents
- **Vocabulary:** Set of all words in corpus
- **Vector:** Mathematical representation of document (counts of word occurrences)

Bag of Words


The brown fox
{
"the" : 1,
"brown": 1,
"fox" : 1
}
brown
fox
[0,0,1,0,1,0,...]
Tokenization
Vectorization
original document
dictionary of word counts
feature vector

Vector Space Model

Similarity is a measure of “distance”

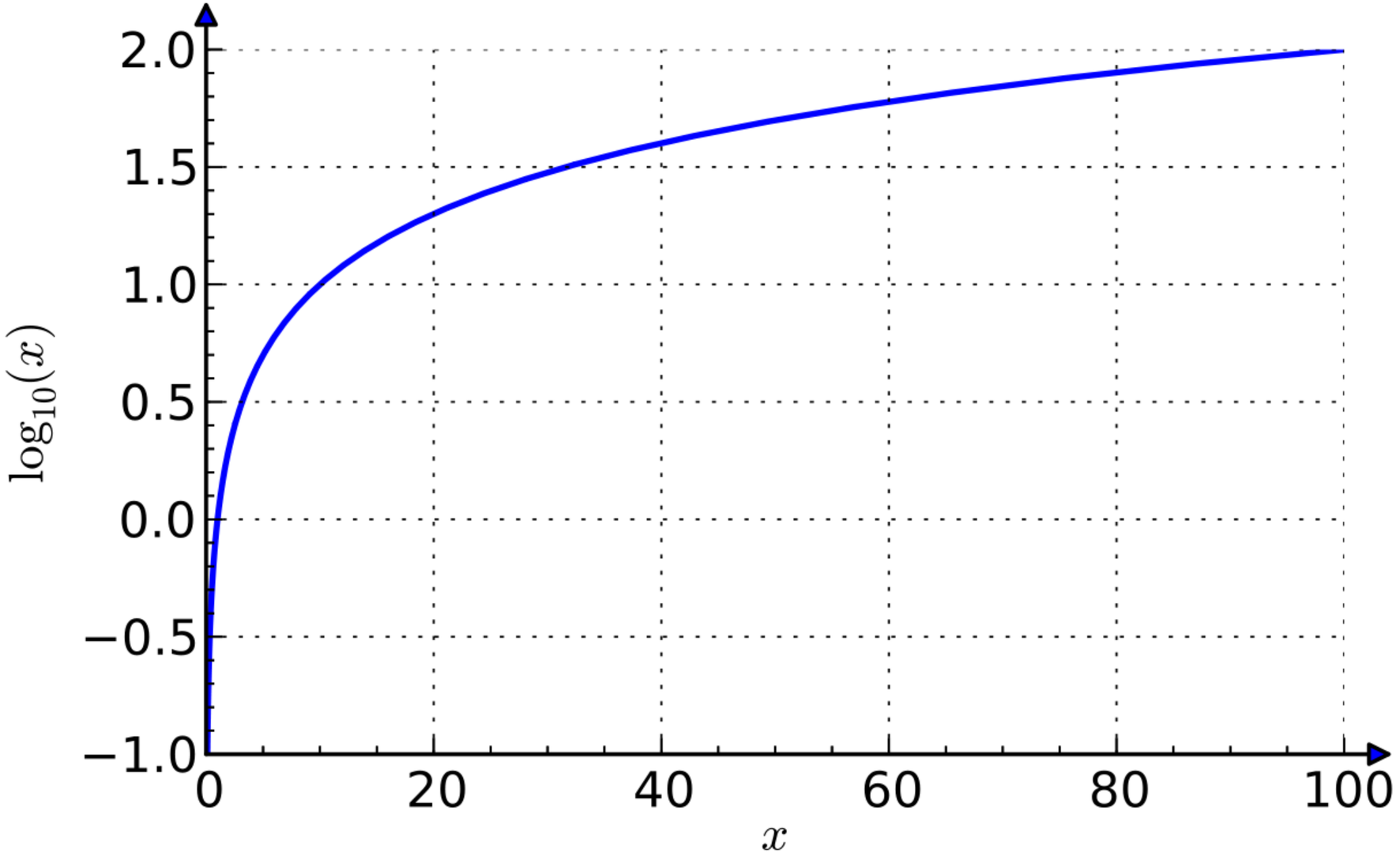


TF-IDF

- Measure of discriminatory power of word (feature)
- Highest when term occurs many times in a small number of documents
- Lowest when term occurs few times in document or many times in corpus
- Useful for information retrieval (queries) and keyword extraction (among other things)

$$tf(t,d) = \frac{f_d(t)}{|d|} \qquad idf(t,D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right)$$

TF-IDF



TF-IDF

Most Common

```
idf[:50]
```

```
[(u'students', 0.014067384597943282),
 (u'I', 0.15305316750494943),
 (u'school', 0.17010493952495984),
 (u'My', 0.3397655206814591),
 (u'The', 0.4149133167820112),
 (u'help', 0.4188088461791251),
 (u'classroom', 0.5361023876769617),
 (u'learning', 0.5748186189046272),
 (u'need', 0.5820538952580256),
 (u'They', 0.5941434194555928),
 (u'learn', 0.6187002265438729),
 (u'able', 0.7452815794748304),
 (u'use', 0.7494117483916651),
 (u"'", 0.755060153205684),
 (u'We', 0.7552806889430156),
 (u'This', 0.7749201702459683),
 (u'class', 0.7913652190100225),
 (u'would', 0.8149828303863013),
 (u'make', 0.8239845109910496),
 (u'many', 0.8273389184929604),
```

Least Common

```
idf[:-50:-1]
```

```
[(u'beer', 10.378594025517652),
 (u'worsen', 10.378594025517652),
 (u'theorist', 10.378594025517652),
 (u'Beneath', 10.378594025517652),
 (u'.how', 10.378594025517652),
 (u'unchanged', 10.378594025517652),
 (u'lessons-', 10.378594025517652),
 (u'on-stage', 10.378594025517652),
 (u'interactiveness', 10.378594025517652),
 (u'GoogleEarth', 10.378594025517652),
 (u'peers\u2019', 10.378594025517652),
 (u'pre-schools', 10.378594025517652),
 (u'PER', 10.378594025517652),
 (u'Davies', 10.378594025517652),
 (u'Spalding', 10.378594025517652),
 (u'7:15am', 10.378594025517652),
 (u'geneticists', 10.378594025517652),
 (u'20-year-old', 10.378594025517652),
 (u'inservice', 10.378594025517652),
 (u'Conquering', 10.378594025517652),
```

```
top_n = 10
summary = bag_of_words.map(lambda x: map(lambda idx: broadcast_idf.value[idx][0], np.argsort(x)[::-1][:top_n]))
```

```
summary.take(15)
```

```
[[u'science',
  u'Outreach',
  u'17-21',
  u'one-year',
  u'resource',
  u'magazine',
  u'periodical',
  u'http',
  u'York',
  u'competency'],
 [u'Worlds',
  u'Hidden',
  u'microscopes',
  u'cell',
  u'stressing',
  u'6th',
  u'single',
  u'cluster',
  u'intense',
  u'organisms'],
 [u'corner',
  u'Harlem',
  u'calming',
  u'rug',
  u'soft',
  u'world.In',
  u'began',
  u'populate',
  u'putting',
  u'stain'],
 [u'Music',
  u'music',
  u'Appreciation',
```

Summarization