

Cultural Analytics

ENGL 64.05

Fall 2019

Prof. James E. Dobson



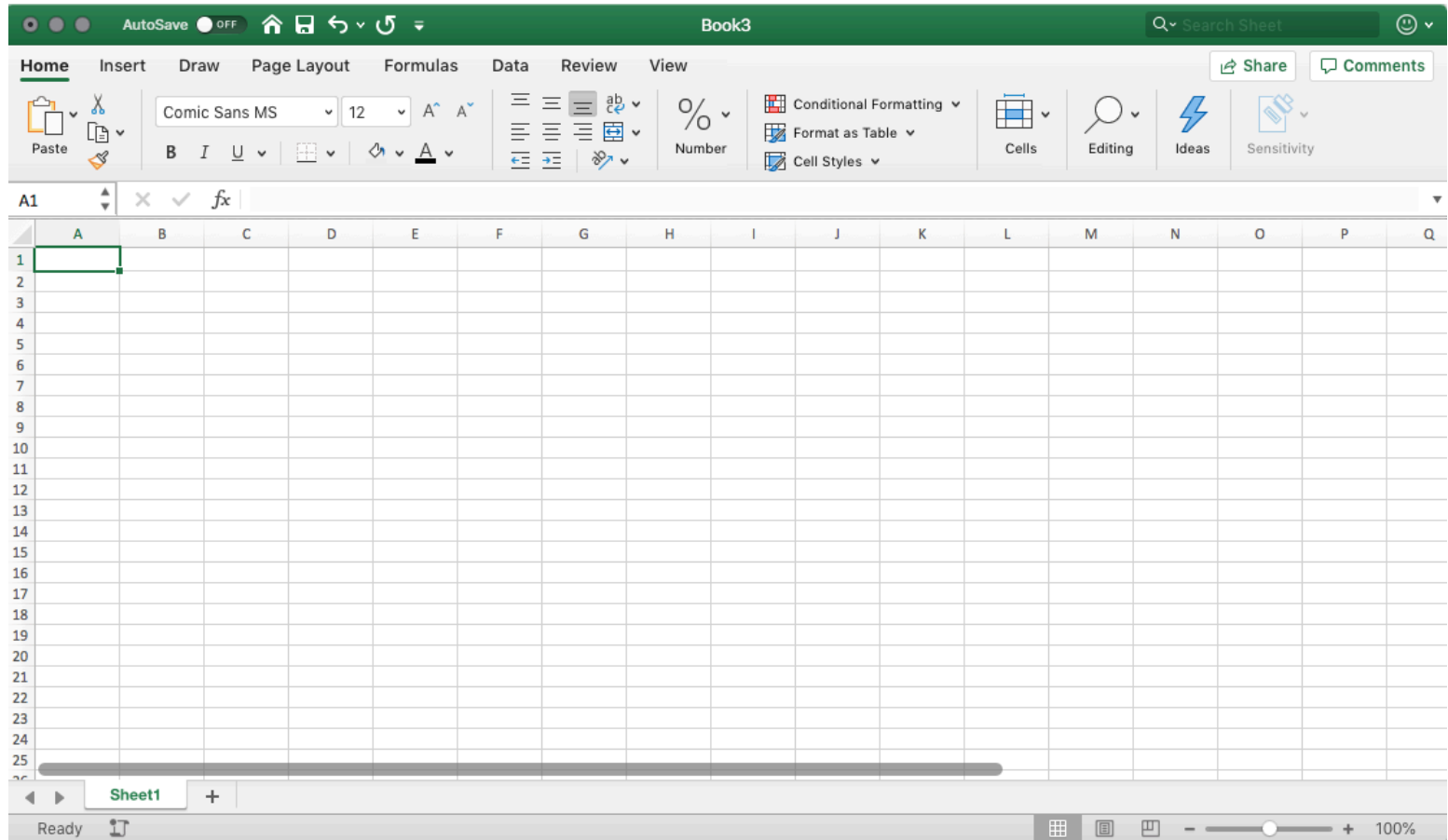
From Lists to Tables

“The crucial epistemological difference between list and table was that in the former, each datum referred to one specific object or category, whereas in the latter, groupings of different sorts related the data to one another, allowing for patterns and structures to emerge. In order to morph gathered data from list to table, each particular datum had to be unlocked from its fixed place in the list’s grid and moved around freely to eventually be subsumed under more general categories and—most importantly—various combinations thereof.”

Christine Von Oertzen,
“Datafication and Spatial Visualization
in Nineteenth-Century Census Statistics,”
574.




Tables are Relational








































































Binary Tables

Doodle Sign up Log in

Table view 

Hidden poll
This is a hidden poll. The participants and the result are only shown to the poll initiator.

		July 2017															
		Mon 17		Tue 18		Wed 19		Thu 20		Fri 21		Sat 22		Sun 23		Mon 24	
		AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM	AM	PM
	Participant 1																
	Participant 2																
	Participant 3																
	Participant 4																
	<input type="text" value="Your name"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Cannot make it Save

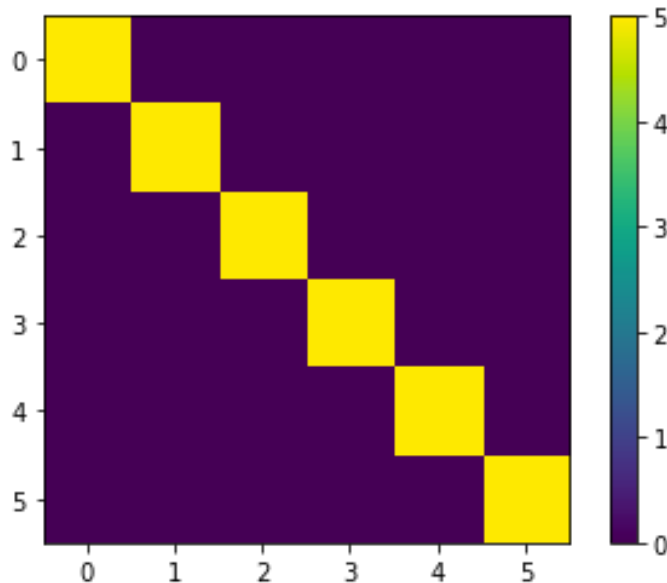
Meeting time algorithm: find column with largest value (assuming all participants are equally needed).

Vectors

```
import numpy as np
import matplotlib.pyplot as plt

data = np.zeros((6,6))
np.fill_diagonal(data, 5)
plt.imshow(data)
plt.colorbar()
plt.show()
```

```
[0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.]
```



```
[5.  0.  0.  0.  0.  0.]
[0.  5.  0.  0.  0.  0.]
[0.  0.  5.  0.  0.  0.]
[0.  0.  0.  5.  0.  0.]
[0.  0.  0.  0.  5.  0.]
[0.  0.  0.  0.  0.  5.]
```

“The table has been vectorized. We might say that the vectorization of the table is *epistopic*...In other words, as the term suggests, an *epistopic* connects general epistemic themes such as validity, precision, specificity, error, confidence, expectation, likelihood, uncertainty, or approximation with a place, a ‘local complex of activities’” (Mackenize, 59).

“Vectorizing computation produces the vector space, which we might understand as a resurgent form of the highly associative pre-Classical table, a table that tolerates many relations and similarities, operationally concrete and machinically abstract. It is no longer a visible diagram but a machinic process that multiples and propagates into the world along many diagonal lines” (73).

Vector Operations

- Mackenzie notes that “in vectorized languages such as R, transformations of a data structure expressed in one line of code simultaneously affect all the elements of the data structure” (68).
- He continues: “Operations no longer step through a series of coordinates that address data elements but instead wield planes, diagonals, cross-sections, and whole-space transformations ... The real stake in vectorizing data is not speed but a transformation in data practice. It makes working with data less like iteration through data structures (lists, indexes, arrays, fields, dictionaries, variables) and more like folding pliable material” (69).

Uses of Vectors

- Image processing
- Recommendation engines
- Spelling correction
- Essay grading
- Document classification
- Document segmentation
- Document clustering
- Topic modeling

Transformation of Text into Data

1. Native Python: Splitting of Strings

```
words = input_text.split()
```

2. NLTK: Tokenization

```
words = nltk.word_tokenize(input_text)
```

3. Scikit-Learn: Vectorization

```
words = CountVectorizer(input=input_text)
```

Introducing...



```
import sklearn
```

```
In [1]: import sklearn
        from sklearn.feature_extraction import text
```

```
In [2]: input_text = """
        Once upon a midnight dreary, while I pondered, weak and weary,
        Over many a quaint and curious volume of forgotten lore –
            While I nodded, nearly napping, suddenly there came a tapping,
        As of some one gently rapping, rapping at my chamber door.
        ‘Tis some visitor,” I muttered, “tapping at my chamber door –
            Only this and nothing more.”
        """
```

```
In [3]: vectorizer = text.CountVectorizer(lowercase='true',
        strip_accents='unicode',
        stop_words='english')
```

```
In [4]: counts = vectorizer.fit_transform([input_text])
        dc, vc = counts.shape
        print("document count:",dc,"vocabulary count:",vc)
```

document count: 1 vocabulary count: 23

```
In [6]: vocab = vectorizer.get_feature_names()
        print(vocab)

['came', 'chamber', 'curious', 'door', 'dreary', 'forgotten', 'gently',
'lore', 'midnight', 'muttered', 'napping', 'nearly', 'nodded', 'pondere
d', 'quaint', 'rapping', 'suddenly', 'tapping', 'tis', 'visitor', 'volum
e', 'weak', 'weary']
```

```
In [7]: # reproduce DTM
counts_as_array = counts.toarray()
for i,word in enumerate(vocab):
    print(word,counts_as_array[0][i])
```

```
came 1
chamber 2
curious 1
door 2
dreary 1
forgotten 1
gently 1
lore 1
midnight 1
muttered 1
napping 1
nearly 1
nodded 1
pondered 1
quaint 1
rapping 2
suddenly 1
tapping 2
tis 1
visitor 1
volume 1
weak 1
weary 1
```

```
from sklearn.feature_extraction import stop_words
print(stop_words.ENGLISH_STOP_WORDS)
```

```
frozenset({'throughout', 'your', 'last', 'who', 'which', 'namely', 'formerly', 'becoming', 'a', 'm', 'latterly', 'although', 'others', 'whatever', 'she', 'after', 'anything', 'except', 'myself', 'in', 'nevertheless', 'something', 'therein', 'wherein', 'whoever', 'had', 'will', 'fifteen', 'noone', 'about', 'many', 'that', 'part', 'other', 'sixty', 'otherwise', 'down', 'beforehand', 'full', 'nowhere', 'still', 'of', 'con', 'him', 're', 'could', 'couldnt', 'hasnt', 'fill', 'this', 'he', 'serious', 'go', 'made', 'would', 'name', 'its', 'afterwards', 'as', 'several', 'some', 'else', 'inc', 'few', 'thence', 'when', 'meanwhile', 'ever', 'it', 'where', 'found', 'through', 'much', 'below', 'on', 'any', 'there', 'thru', 'someone', 'front', 'detail', 'more', 'most', 'perhaps', 'whereas', 'whereupon', 'thereafter', 'well', 'side', 'along', 'see', 'cry', 'cant', 'everyone', 'no', 'almost', 'each', 'five', 'without', 'herein', 'were', 'another', 'either', 'give', 'wherever', 'me', 'empty', 'besides', 'his', 'ie', 'or', 'seems', 'same', 'always', 'everything', 'mostly', 'too', 'thus', 'hence', 'out', 'amount', 'less', 'somehow', 'nor', 'before', 'enough', 'hereupon', 'was', 'then', 'ltd', 'via', 'yours', 'us', 'latter', 'between', 'if', 'least', 'eg', 'describe', 'anyway', 'eleven', 'not', 'amongst', 'together', 'i', 'nine', 'rather', 'into', 'again', 'call', 'you', 'anyhow', 'until', 'one', 'former', 'under', 'and', 'both', 'our', 'yet', 'seem', 'ourselves', 'per', 'find', 'from', 'whither', 'move', 'thereupon', 'please', 'seemed', 'here', 'already', 'eight', 'hereby', 'how', 'beyond', 'take', 'can', 'so', 'the', 'due', 'them', 'a', 'also', 'during', 'herself', 'indeed', 'etc', 'never', 'for', 'get', 'off', 'should', 'sometime', 'by', 'their', 'sincerely', 'keep', 'moreover', 'bottom', 'nothing', 'alone', 'since', 'thereby', 'among', 'bill', 'now', 'everywhere', 'anywhere', 'neither', 'whether', 'because', 'at', 'but', 'has', 'over', 'such', 'beside', 'themselves', 'itself', 'often', 'third', 'up', 'becomes', 'only', 'two', 'cannot', 'across', 'do', 'fire', 'than', 'next', 'be', 'co', 'though', 'mine', 'hereafter', 'forty', 'above', 'himself', 'ten', 'whom', 'even', 'whereby', 'however', 'own', 'we', 'hers', 'an', 'top', 'twenty', 'what', 'six', 'de', 'three', 'those', 'is', 'while', 'show', 'behind', 'twelve', 'anyone', 'become', 'may', 'somewhere', 'her', 'elsewhere', 'put', 'therefore', 'they', 'thin', 'with', 'must', 'un', 'hundred', 'whereafter', 'against', 'upon', 'sometimes', 'four', 'system', 'amongst', 'my', 'every', 'mill', 'are', 'seeming', 'these', 'to', 'first', 'why', 'whenever', 'fifty', 'nobody', 'all', 'done', 'onto', 'thick', 'toward', 'who', 'le', 'whence', 'yourself', 'around', 'towards', 'being', 'yourselves', 'became', 'interest', 'might', 'once', 'very', 'back', 'further', 'within', 'none', 'whose', 'ours', 'have', 'been'})
```

What Stopwords?

Why Function/Stopwords?

Which is it?

- “Very frequent context words, such as the word *the*, unfortunately result in most vectors matching a non-zero coordinate. Such words are precisely the contexts that have little semantic discrimination power” (Turney & Pantel, 158).
- “Some forty years ago, Mosteller and Wallace suggested in their influential work on the Federalist Papers that a small number of the most frequent words in a language (‘function words’) could usefully serve as indicators of authorial style” (Argamon & Levitan 2005)

```
In [1]: import sklearn
        from sklearn.feature_extraction import text
```

```
In [2]: input_text = """
        Once upon a midnight dreary, while I pondered, weak and weary,
        Over many a quaint and curious volume of forgotten lore –
            While I nodded, nearly napping, suddenly there came a tapping,
        As of some one gently rapping, rapping at my chamber door.
        ‘Tis some visitor,” I muttered, “tapping at my chamber door –
            Only this and nothing more.”
        """
```

```
In [3]: vectorizer = text.CountVectorizer(lowercase='true',
        strip_accents='unicode')
```

```
In [4]: counts = vectorizer.fit_transform([input_text])
        dc, vc = counts.shape
        print("document count:",dc,"vocabulary count:",vc)

document count: 1 vocabulary count: 40
```

```
In [5]: vocab = vectorizer.get_feature_names()
        print(vocab)

['and', 'as', 'at', 'came', 'chamber', 'curious', 'door', 'dreary', 'forgotten', 'gently', 'lore', 'many', 'midnight', 'more', 'muttered', 'my', 'napping', 'nearly', 'nodded', 'nothing', 'of', 'once', 'one', 'only', 'over', 'pondered', 'quaint', 'rapping', 'some', 'suddenly', 'tapping', 'there', 'this', 'tis', 'upon', 'visitor', 'volume', 'weak', 'weary', 'while']
```

max_df : float in range [0.0, 1.0] or int, default=1.0. When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float, the parameter represents a proportion of documents, integer absolute counts. This parameter is ignored if vocabulary is not None.

min_df : float in range [0.0, 1.0] or int, default=1. When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float, the parameter represents a proportion of documents, integer absolute counts. This parameter is ignored if vocabulary is not None.

max_features : int or None, default=None. If not None, build a vocabulary that only consider the top max_features ordered by term frequency across the corpus. This parameter is ignored if vocabulary is not None.

Choices...

- strip_accents
- lowercase
- stop_words
- ngram_range
- max_df
- min_df
- max_features

ngrams

```
vectorizer = text.CountVectorizer(lowercase='true',
                                   ngram_range=(1,2),
                                   strip_accents='unicode',
                                   stop_words='english')
```

- Does meaning reside just in single words?
- What might common two-word combinations (a “bi-gram”) contribute to mean?

came 1	nodded 1
came tapping 1	nodded nearly 1
chamber 2	pondered 1
chamber door 2	pondered weak 1
curious 1	quaint 1
curious volume 1	quaint curious 1
door 2	rapping 2
door tis 1	rapping chamber 1
dreary 1	rapping rapping 1
dreary pondered 1	suddenly 1
forgotten 1	suddenly came 1
forgotten lore 1	tapping 2
gently 1	tapping chamber 1
gently rapping 1	tapping gently 1
lore 1	tis 1
lore nodded 1	tis visitor 1
midnight 1	visitor 1
midnight dreary 1	visitor muttered 1
muttered 1	volume 1
muttered tapping 1	volume forgotten 1
napping 1	weak 1
napping suddenly 1	weak weary 1
nearly 1	weary 1
nearly napping 1	weary quaint 1

Douglass Comparison

Single term:

min_df: 1 vocabulary count: 15,095

min_df: 2 vocabulary count: 12,024

min_df: 3 vocabulary count: 8,535

min_df: 4 vocabulary count: 3,874

Bigrams:

min_df: 1 vocabulary count: 138,776

min_df: 2 vocabulary count: 82,919

min_df: 3 vocabulary count: 33,245

min_df: 4 vocabulary count: 7,021

Semantic Space

- Vectorized data cannot capture/represent formal features or structure (organizational containers (sentences, paragraphs, chapters, etc) of the document are lost.
- But semantic information about the modeled space can be recalled from the model.
- Similar vectors should have similar meaning.
- The *distributional hypothesis*: words that occur in similar contexts tend to have similar meanings.
- Additional context means greater disambiguation of meaning.