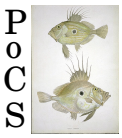

Principles of Complex Systems, Vols. 1, 2, & 3D
CSYS/MATH 6701, 6713, & a pretend number
University of Vermont, Fall 2023
Assignment 13
"Don't answer that, you'll get it wrong."

Due: Friday, February 9, by 11:59 pm
https://pdodds.w3.uvm.edu/teaching/courses/2023-2024pocsverse/assignments/13/
Some useful reminders:
Deliverator: Prof. Peter Sheridan Dodds (contact through Teams)
Assistant Deliverator: Chris O'Neil (contact through Teams)
Office: The Ether
Office hours: See Teams calendar
Course website: https://pdodds.w3.uvm.edu/teaching/courses/2023-2024pocsverse
Overleaf: LaTeX templates and settings for all assignments are available at https://www.overleaf.com/read/tsxfwwmwdgxj. If this link doesn't work, try https://www.overleaf.com/read/tsxfwwmwdgxj#456832

All parts are worth 3 points unless marked otherwise. Please show all your workingses clearly and list the names of others with whom you ~~conspired~~ collaborated.

For coding, we recommend you improve your skills with Python, R, and/or Julia. The (evil) Deliverator uses (evil) Matlab.

Graduate students are requested to use LaTeX (or related TeX variant). If you are new to LaTeX, please endeavor to submit at least n questions per assignment in LaTeX, where n is the assignment number.

Assignment submission:

Via Brightspace or other preferred death vortex.

Please submit your project's current draft in pdf format via Brightspace by the same time specified for this assignment. For teams, please list all team member names clearly at the start.

Semester goal: A paper per text studied, building through assignments.

Three stories to analyze:

- **Pride and Prejudice**
  https://www.gutenberg.org/ebooks/1342
- **Frankenstein; or the Modern Prometheus**
  https://www.gutenberg.org/files/84/84-h/84-h.htm

- **Moby Dick; Or, The Whale**
  https://www.gutenberg.org/ebooks/2701

Tasks:

1. (9 points, 3 points for each novel)

   Take the UTF-8 text versions of each of these three novels and parse them into narrative time series of 1-grams which include all punctuation, numbers, and words.

   See below for instructions.

   For each novel, present your output for the first paragraph, rendered as a single, wrapped line.

2. (9 points, 3 points for each novel)

   For each novel, produce size rank distributions of 1-grams according to counts.

   Display the 1-grams and counts for the first 100 1-grams.

3. (9 points, 3 points for each novel)

   For each novel, plot the rank-count distribution for 1-grams.

Outputs provided for comparison:

The basic data format is as a time series with one 1-gram per line (links below).

For each story, also linked to below are the rank distributions of 1-grams by counts.

https://pdodds.w3.uvm.edu/permanent-share/pride_and_prejudice_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/pride_and_prejudice_1grams.txt

https://pdodds.w3.uvm.edu/permanent-share/frankenstein_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/frankenstein_1grams.txt

https://pdodds.w3.uvm.edu/permanent-share/moby-dick_narrativetimeseries.txt
https://pdodds.w3.uvm.edu/permanent-share/moby-dick_1grams.txt

General instructions:

- Using an editor, remove the start and finish material for each of the three novels that Gutenberg adds to books somewhat inconsistently.

- Using a judicious selection of regex operations, create a script* (Python is strongly recommended but you can use whatever you like) to break up the text into meaningful 1-grams that may be words, punctuation, and numbers.

  Regex = Regular Expression.

  * If you want to use tokenizer, you can. But the idea is to be careful and really understand what's happening the text as you smash it into pieces (storyons).

- See Perl code below for an example.

```
## for Frankenstein
$text =~ s/D--n/Damn/g;

## separate out some basic punctuation
$text =~ s/([\!\?\,\.])/ \1 /g;
$text =~ s/:/ : /g;
$text =~ s/;/ \; /g;

## remove underscores used for emphasis
$text =~ s/_//g;

## isolate parentheses
$text =~ s/\(/ ( /g;
$text =~ s/\)/ ) /g;

## dash madness
$text =~ s/----/ --- /g; ## long dash
$text =~ s/--/ --- /g; ## em dash
$text =~ s/;-/ --- /g; ## em dash
$text =~ s/-/ --- /g; ## em dash

## handle specific salutations
$text =~ s/Mr \./Mr./g;
$text =~ s/Mrs \./Mrs./g;
$text =~ s/Dr \./Dr./g;

## clean up white space duplication
$text =~ s/\s+/ /g;

## separate quotes
## opening quotes should have a space before them (except for em dashes, treated
```

```
$text =~ s/\s"/ " /g;
## closing quotes should be what's left:
$text =~ s/"/ " /g;
$text =~ s/"/ " /g;
$text =~ s/"/ " /g;

## separate off opening single quotation mark
$text =~ s/‘/ ‘ /g;

## dyspunctional catapostrophes:
## clean up apostrophes and opening and closing single quote mark
## closing single quote mark should generally be isolated
## leave alone to preserve contractions
$text =~ s/‘/ ' /g;

$text =~ s/’/'/g;
## opening quote mark
## \p{L} stands for any UTF-8 letter
$text =~ s/(\s)'(\p{L})/\1' \2/g;
## closing quote mark (will be a problem with contractions)
$text =~ s/(\p{L})'(\s)/\1 '\2/g;

## split off possession indicator
$text =~ s/'s/ 's/g;

## remove any white space at the front
$text =~ s/^\s+//;

## add new line at the end
$text = $text."\n";

## now create time series text by replacing spaces with returns
($timeseriestext = $text) =~ s/ /\n/g;

## last: remove any white space redundancies
$timeseriestext =~ s/\s+/\n/ms;
```