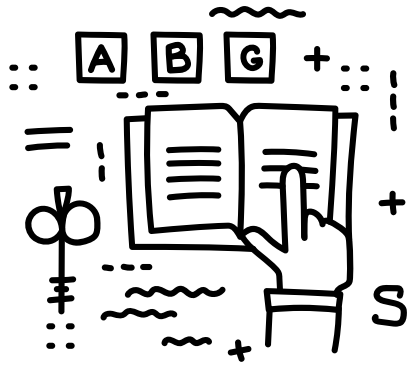
Political Text Analysis

Lecture 1

Kohei Watanabe



Introduction

What is this course about?

About me

- Kohei Watanabe
  - Assistant Professor at WIAS
    - Since June 2018
    - Office is building 9, room 804
  - Trained in Europe
    - 2010-2011 MA in Political Science from Central European University (Hungary)
    - 2013-2016 PhD in Social Research Methodology from London School of Economics (UK)
  - Research interests
    - Political communication (measuring bias in news, media agenda-setting)
    - International communication (propaganda, media globalization)
    - Further information is available at https://koheiw.net
  - Software development
    - Text analysis packages (quanteda, newsmap, LSS etc.)

Goals of this course

- Become a political text analyst
  - Learn technological and theoretical backgrounds of quantitative text analysis
    - Explain how and why text analysis has been used in political science in lectures
  - Acquire practical skills to manage, manipulate and analyse textual data
    - Teach how to use R as a programming language in seminars
- Apply quantitative text analysis in own fields
  - Quantitative text analysis can also be used in
    - Sociology, psychology, (corpus) linguistics, (digital) humanity, marketing etc.

Focus of this course

- There are two important things to understand
  - How textual data is stored and processed by the software programs
  - How and why quantitative text analysis is applied in political science research
    - There are questions that we cannot answer using other methods (survey, interview, qualitative text analysis etc.)
    - However, quantitative text analysis is also limited in many ways…

Tell me about yourself

- Name
- Home country
- Affiliation (if not School of Political Science and Economics)
- Academic interests
- Experience in
  - Statistical analysis
  - Computer programming
- Expectation to this course (if any)

Housekeeping

Logistic, schedule and grading

Lectures and seminars

- Wednesdays 4-5th slots (building 3, room 808)
  - Lectures 14:45-16:15
  - Seminars 16:30-18:00
  - Office hour by appointment (koheiw@aoni.waseda.jp)
- Lectures
  - Understand the technological and theoretical backgrounds
    - There is no all-in-one textbook on quantitative text analysis
    - I will suggest relevant papers and book chapters
- Seminars
  - Acquire practical skills to manage, manipulation and analyze textual data
    - You will learn computer programming in R
    - We will learn how to use quanteda

Course schedule

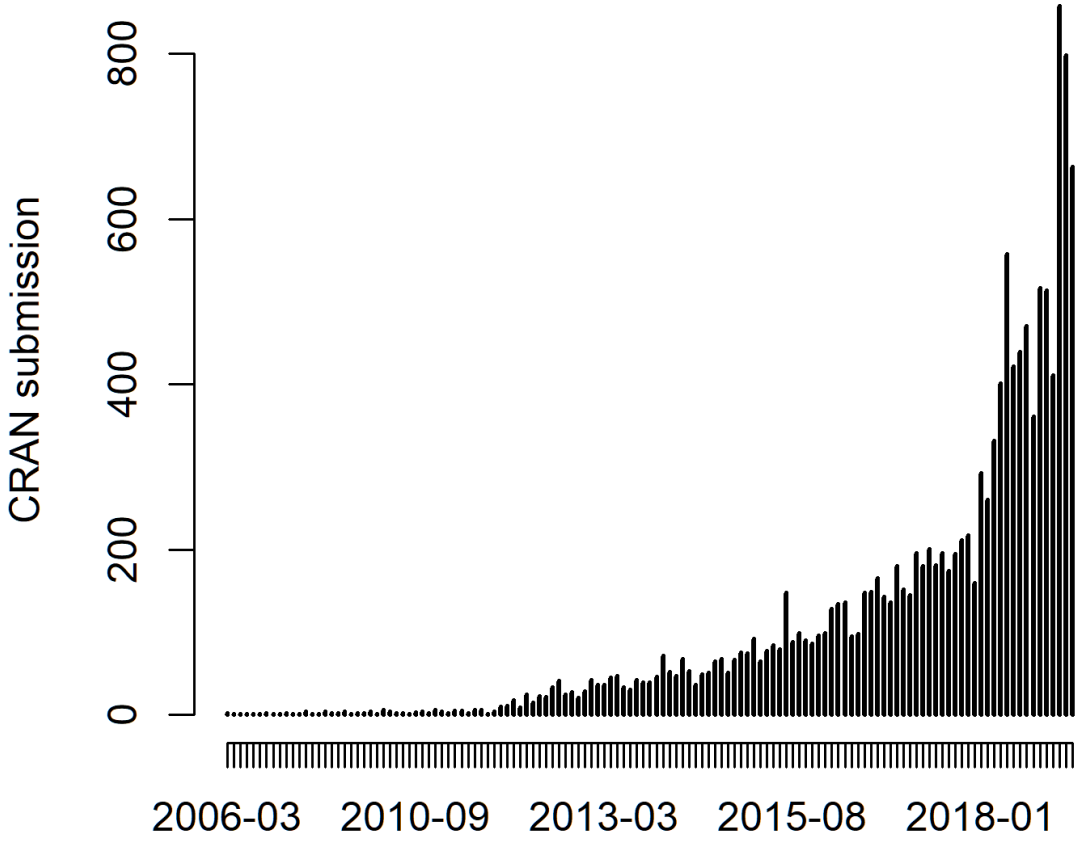
Week	Lectures	Seminars
1	Introduction	Programming in R (vector and data.frame)
2	Data collection and management	Programming in R (list)
3	Content analysis	Programming in R (matrix)
4	Statistical analysis	Data import and workflow
5	Machine learning 1	Basic and advanced operations
6	Machine learning 2	Statistical analysis
7	Advanced techniques	Scaling and classification
8	Exam: group presentation	

Why you have to learn programming?

- Programming skill is useful when you
  - Collect data from web-API or websites (scraping)
  - Preprocess textual data
  - Build complex analytic pipeline
- Programming helps you to understand
  - Underlying technologies
  - Textual data processing

What is R?

- R is a statistical language
- Development of R started to extend older statistical program S in early 1990s and it became an open source software in 1995
- R is widely used at political science departments in research and teaching
  - More and more statistical courses are taught using R instead of Stata or SPSS
  - Researchers develop packages and distribute on CRAN (~14,000 packages)



Programming in R

- Is R a programming language?
  - R is far more than Stata or SPSS commands
  - R base has very limited functions for textual data
    - Recently developed packages (stringi and RcppParallel) made processing of large textual data possible
  - R is only fast in vectorized operations
    - Serious R packages write their core functions in C++
    - R is more like shell script (interface to low-level languages)
- Why not Python?
  - It takes more time to master programming in Python
  - If you know R, you can understand other languages more easily

Seminars: week 1-3

- We make “mini quanteda” to understand
  - R language
    - Basic syntax
    - Object types (vector, data.frame, list, matrix)
    - Control flow (for loop, if statement)
  - Textual data processing
    - Fast data transformation
    - Pattern matching (regular expression)
    - Textual data structure

Seminars: week 4-7

- Lean how to use quanteda
  - Better to use quanteda to deal with real corpora
    - It's core functions are written in C++
    - It has a lot of functions
    - It is tested rigorously
  - Learning materials are available online
    - "quanteda tutorials" (https://tutorials.quanteda.io)

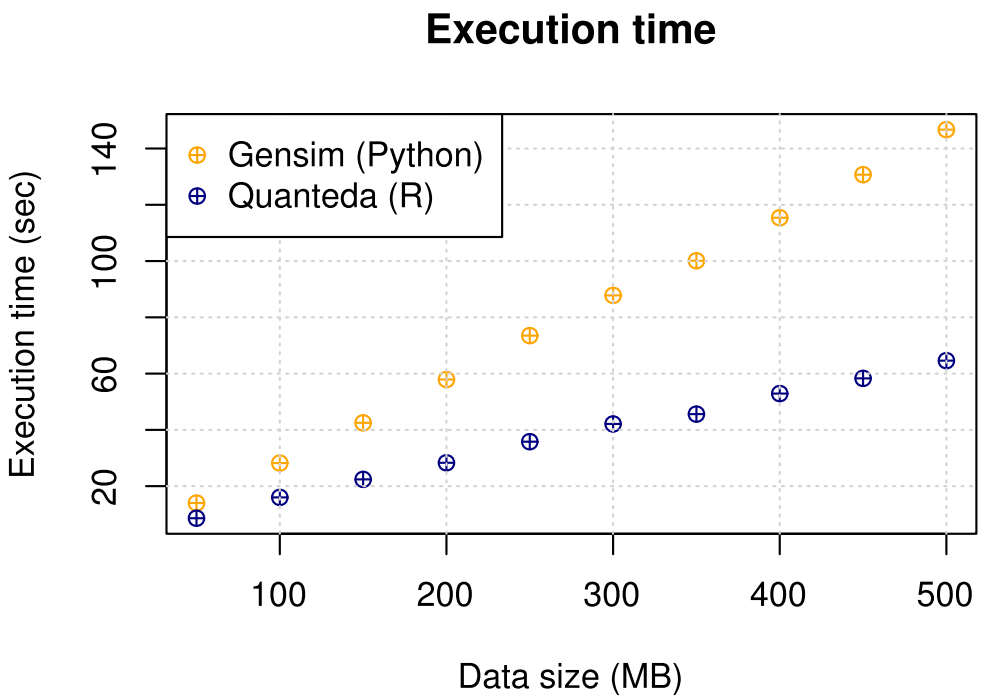
What you need in seminars

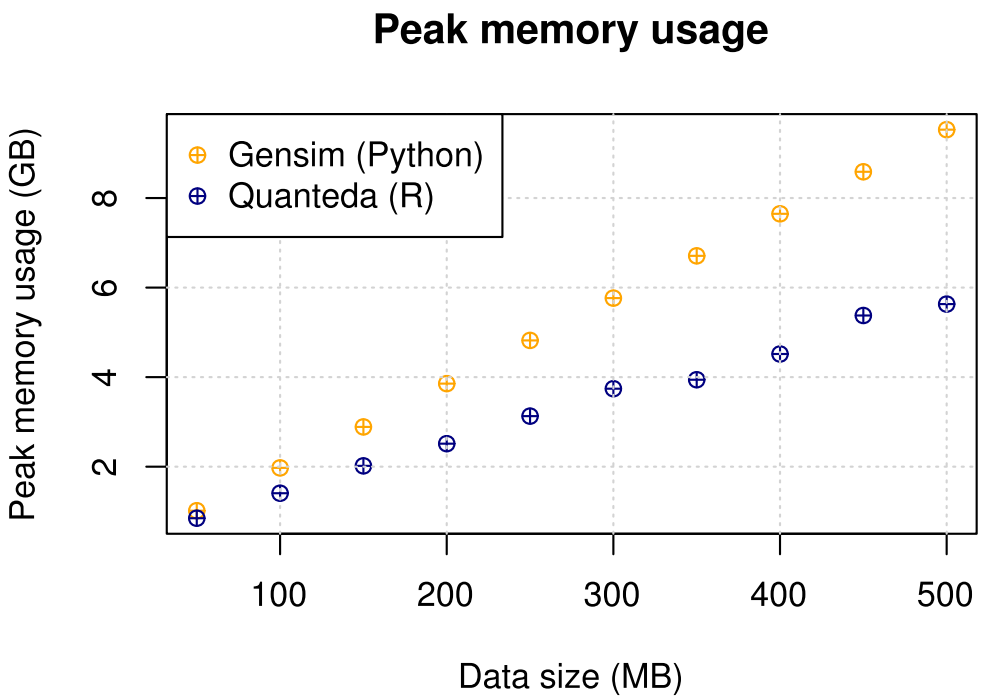
- Laptop
  - Windows, Mac, Linux
- R
  - R (v3.5 or later)
  - R Studio
  - R packages (latest)
- Text editor
  - Unicode support
- Browser

Tools for quantitative text analysis

- R
  - tm, tidytext, quanteda
- Python
  - NLTK, gensim
- Java
  - Mallet, Yoshikoder, Lexicoder
- GUI
  - KH Coder, Wordsmith, QDA Miner/Wordstat, LIWC

Comparison between R and Python





Exam: group presentation

- Form a group of 2-3
  - Write an essay (about 5,000 words) due in week 8
  - Present their project in about 10 min in week 8
  - Every member should do some analysis
  - Members share the same dataset and theoretical framework
- For good essay/presentation, you should
  1. Address theoretically important questions
  2. Address questions from a new perspective
  3. Analyze appropriate textual data
  4. Check robustness of your findings
  5. Clearly state above four points

Books and papers

Krippendorff, K. (2004). *Content Analysis: An Introduction to Its Methodology*. Sage.

Grimmer, J., & Stewart, B. M. (2013). Text as Data: The Promise and Pitfalls of Automatic Content Analysis Methods for Political Texts. *Political Analysis*, 1–31. https://doi.org/10.1093/pan/mps028

Welbers, K., Van Atteveldt, W., & Benoit, K. (2017). Text Analysis in R. *Communication Methods and Measures*, *11*(4), 245–265. https://doi.org/10.1080/19312458.2017.1387238

Manning, C. D., & Schütze, H. (2001). *Foundations of statistical natural language processing*. Cambridge (Mass.): MIT press.

Jurafsky, D., & Martin, J. H. (2009). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J.: Pearson Prentice Hall.

Hastie, T. J., Tibshirani, Robert J, & Friedman, Jerome H. (2013). *The elements of statistical learning: data mining, inference, and prediction*. New York, NY: Springer.

Relevant technologies

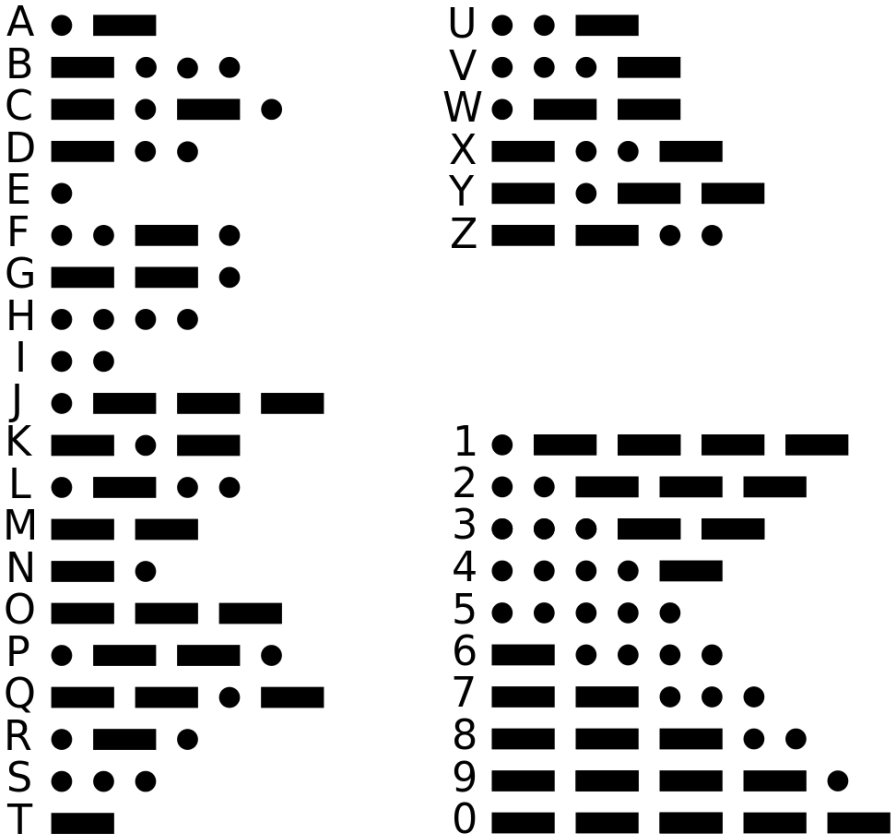
OS, character encoding, file format, etc.

Operating systems

- R is available for Windows, Mac and Linux
  - R and R packages work consistently across operating systems
  - However, input and output of texts depends on their design and settings
- The main problem is how they handle non-English texts
  - Mac and Linux
    - We usually have no problem with non-English texts
  - Windows 10
    - We too often face problem with non-English texts
    - All depends on settings in R and Windows
    - Windows version of R has its own problem that is difficult to fix

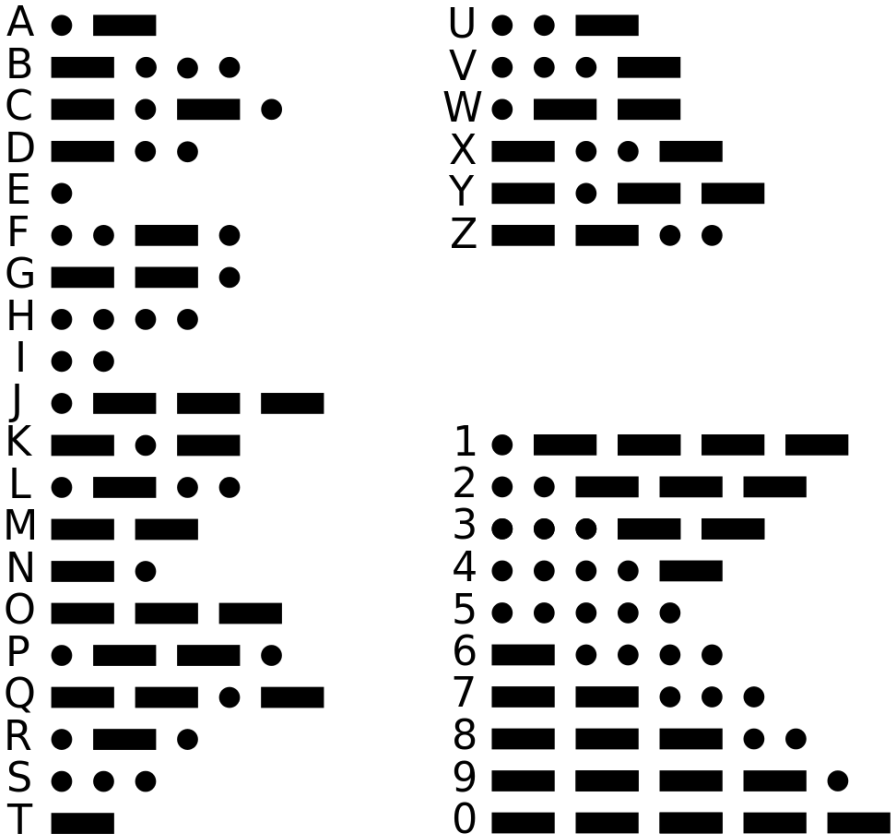
Character encoding

- Texts need to be expressed in simpler form (signal) to transmit electronically
- One of the character encoding is *Morse code* for telegraph
  - It combines dot (“.”), dash(“-”) and gap(“/”)
- You have to know the links between characters and codes to understand the text



Quiz

- Decode Morse code messages
  1. “... --- ...”
     - “SOS” (distress signal)
  2. “.. - / .. ... / .- / -... --- -.—”
     - “IT IS A BOY” (message sent in 1952 when a hydrogen bomb test was successful)



Character encoding

- Computers assign numeric ID to characters
  - ASCII (American Standard Code for Information Interchange) was created in 1963
    - Can only express 128 (= 2^7) characters in 7 bit
    - Do not even cover character with accent {á, é, ó, ú, ö, è, ã, ñ}

ASCII code

USASCII code chart

b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits ↓ b_4 b_3 b_2 b_1	Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Character encoding

- ISO 2022 was created to include European characters
  - 8 bit (1 bite) to include more characters
  - It became the basis of ISO 8859-x encodings (Latin-x)
    - ISO 8859-1 (Western European)
    - ISO 8859-2 (Central European)
    - ISO 8859-3 (South European)
    - ISO 8859-4 (North European)
    - ISO 8859-5 (Turkish)
    - ISO 8859-6 (Nordic)
  - It is also included in local encoding for Asian languages
    - e.g. Thai, Vietnamese, Indian, and Japanese

Character encoding

- There are too many local and system dependent character encodings
  - System dependent encoding
    - Computer companies (IBM, Microsoft, Apple etc.) started developing own character encoding
  - National encoding for CJK
    - National Institutions started creating original encodings
      - Japan: JIS (1970s-1990s)
      - China: GB 2312 (1981)
      - Korea: KS C5601 (1987)
      - Taiwan CNS 11643 (1992)
  - ISO 8859-x
  - EUC (Extended Unix Code)

Character encoding

- Fragmentation of character encoding makes exchange of data really difficult
  - Decoding by different scheme makes texts unreadable
  - Encoding into wrong scheme destroys the data

ISO8859-2	Orbán: Csak erős és öntudatos nemzetek állhatnak meg a birodalmi fenyegetésekkel szemben
ISO8859-1	Orbán: Csak erõs és öntudatos nemzetek állhatnak meg a birodalmi fenyegetésekkel szemben
Shift-JIS	Orb疣: Csak er□ 駸 □tudatos nemzetek 疝lhatnak meg a birodalmi fenyeget駸ekkel szemben

"Orbán: Only strong and self-conscious nations can stand up to imperial threats" (www.hirado.hu)

Character encoding

- Unicode is designed to support all the languages
  - In the 1980s, groups from the US, Japan, and Europe started to create international character codes
  - Unicode is saved in UTFs (Unicode Transformation Formats)
    - UTF-32
      - Cover all the Unicode space in 32-bit (2^32 = 4,294,967,296)
      - Takes a lot of storage and memory space
    - UTF-16 and UTF-8
      - Combine multiple code to cover Unicode
    - UTF-8 uses the same code for Ascii characters
      - It is fully backword compatible
      - File size remain the same for documents only with Ascii characters

File formats

- Old formats
  - Text (.txt)
    - We need guess character encoding
    - Can be unreadable or destroyed if it is not in UTF-8
    - Open in a text editor (not MS Word) or read directly from R
  - Comma (character) separated value (.csv/tsv)
    - Similar to text files
    - Open in Libre Office (not MS Excel) or read directly from R

File formats

- Legacy formats
  - HTML (.html)
    - Character encoding is specified in the header “<meta charset="UTF-8"/>”
    - Not easy to remove texts for buttons and advertisement
  - PDF (.pdf)
    - Looks great on screen but really difficult to extract texts
    - Texts are often only images
    - You might need to re-OCR

File formats

- Robust format
  - XML (.xml)
    - Character encoding is specified explicitly
    - Information is XML files defined clearly by tags
    - Difficult to parse when structure is complex
    - Used in Web-API and archives
  - JSON (.json)
    - Less robust but more compact than XML
    - Used in Web-API
- Human-friendly formats
  - YML/YAML (.yaml)
    - Similar to JSON but more human-friendly

File management

- Data backup
  - You can use Dropbox or OneDrive, but
    - Do not to use if you are analyzing sensitive data
    - Do not sync your R package directory
      - R installs packages in “C:\Users\your-name\Documents\R\win-library” (.libPaths() shows you where it is)
      - Uncheck “R” folder in OneDrive’s setting
- Version control
  - Git is becoming popular
    - You can trace changes in files and leave comments to collaborate with others
    - You should at least have Github account (https://github.com) to join online discussions

Hardware

- Textual data requires more computing than numeric data
  - Statistical estimation is parallelized
    - More CPU core the better (Core i3 < i5 < i7 < Xeon)
    - We recommend 4 or more cores (threads)
  - Data transformation needs space
    - More RAM is better (8GB < 16GB < 32GB)
      - RAM should be ~10 times more than the size of data on disk
    - We recommend 8GB or more RAM
  - When there is no space in RAM, it starts saving data into disks
    - SSD is much faster in random access

References

McEnery, A. M., & Xiao, R. Z. (2005). Character encoding in corpus construction. In M. Wynne (Ed.), *Developing Linguistic Corpora : A Guide to Good Practice*. Retrieved from http://eprints.lancs.ac.uk/60/

Gillam, R. (2002). *Unicode Demystified: A Practical Programmer's Guide to the Encoding Standard*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.