ST445 Managing and Visualizing Data

Introduction to Data

Week 1 Lecture, MT 2017 - Kenneth Benoit

Data is Fundamental

"You can have data without information, but you cannot have information without data." – Daniel Keys Moran, an American computer programmer and science fiction writer.

"In God we trust. All others must bring data." – W. Edwards Deming, statistician, professor, author, lecturer, and consultant.

Structured data: Index cards

- Origins in the 19th century, with botanist Carl Linnaeus (https://en.wikipedia.org/wiki/Carl_Linnaeus), who needed to record species that he was studying
- This was a form of *database*
  - each piece of information about a species formed a *field*
  - each species' entry in the system formed a *record*
  - the records were *indexed* using some reference system

Heyday: Use in libraries to catalog books

a record looked like this

photo by Abbie Weinberg

Dewey decimal system

- a proprietary library classification system first published in the United States by Melvil Dewey in 1876
- scheme is made up of ten classes, each divided into ten divisions, each having ten sections
- the system's notation uses Arabic numbers, with three whole numbers making up the main classes and sub-classes and decimals creating further divisions

- Example:

```
500 Natural sciences and mathematics
    510 Mathematics
        516 Geometry
            516.3 Analytic geometries
                516.37 Metric differential geometries
                    516.375 Finsler Geometry
```

How to index?

- Problem: Could only sort the cards one way
- Re-referencing was literally a manual operation

- Contrast with the idea of electronic indexes, where assets are stored once, and many indexing and reference systems can be applied

- In most photographic software, edits are "non-destructive", and stored separately from the original images

Modern database manager

- "Normalizes" data into relational tables, linked by "keys" (more on this in Week 3 (https://lse-st445.github.io/#week-3-creating-and-managing-databases))

School Table

ID	Name
S001	University of Technology
S002	University of Applied Science

Student Table

School ID	ID	Name	DOB
S001	UT-1000	Tommy	05/06/1995
S001	UT-1000	Better	16/04/1995
S002	UAS-1000	Linda	02/09/1995
S002	UAS-1000	Jonathan	22/06/1995

Punch cards (and legacy systems)

- How we used to enter programs into the computer

- Pre-computer origins: 18-th century use in textile looms
- Responsible for the 80-character legacy

Who knows what this is?

The point: Information is more than data

Data

- raw, unorganized facts that need to be processed
- unusable until it is organized

Information

- created when data is processed, organized, structured
- needs to be situated in an appropriate *context* in order to become useful

There are important differences in how humans and computers treat data as information

Changes in the world of data

- volume of data in the modern world: 90% of the world's data generated in the last *two years* (https://www.sciencedaily.com/releases/2013/05/130522085217.htm)

- and that was *in 2013*

- SKA: Square Kilometer Array
  - a southern hemisphere radio telescope with a total of 1 km^2 of data sensors
  - will generate 1 exabyte *daily* = 1×10^{18} bytes
  - = 1,000,000,000,000,000,000 bytes

- compare this with the Apollo Guidance Computer (1966), which guided the first humans to the moon
  - 16-bit wordlength, 2048 words RAM (magnetic core memory) = *4KB*
  - 36,864 words ROM (core rope memory) = *73KB*

Basic units of data

- Bits
  - smallest unit of storage, a 0 or 1
  - anything that can store two states - now "transistors", used to be vacuum tubes
  - with n bits, can store 2^n patterns - so one byte can store 256 patterns

- Bytes
  - eight *bits* = one *byte*
  - "eight bit encoding" - represented characters, such as `A` represented as 65

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

multi-byte units

unit	abbreviation	total bytes	nearest decimal equivalent
kilobyte	KB	1,024^1	1000^1
megabyte	MB	1,024^2	1000^2
gigabyte	GB	1,024^3	1000^3
terabyte	TB	1,024^4	1000^4
petabyte	PB	1,024^5	1000^5
exabyte	EB	1,024^6	1000^6
zettabyte	ZB	1,024^7	1000^7
yottabyte	YB	1,024^8	1000^8

- this is why 1GB is greater than 1 billion bytes

Comparison of decimal and binary units

Data types: Generically

- objects are *bound* to an identifier, e.g.

In [2]:
```
temperature = 98.6
print(temperature)
```

```
98.6
```

- here, `temperature` is a variable name assigned to the literal floating-point object with the value of 98.6
- in Python, this is an instance of the **float** class
- identifiers in R and Python are *case-sensitive*
- some identifiers are typically reserved, e.g.

```
False, True, None, or, and  # Python
  FALSE, TRUE, NA             # R
```

- All programming languages use comments, for humans to read
  - this is anything that follows the # character in both Python and R

"Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do." -- Donald Knuth, Literate Programming (1984)

Instantiation

- objects have *classes*, meaning they represent a "type" of object
- *instantiation* means creating a new instance of that class

- "immutable" objects cannot be subsequently changed

Python class	Immutable	Description	R class
bool	Yes	Boolean value	logical
int	Yes	integer number	integer
float	Yes	floating-point number	numeric
list	No	mutable sequence of objects	list
tuple	Yes	immutable sequence of objects	-
str	Yes	character string	character
set	No	unordered set of distinct objects	-
frozenset	Yes	immutable form of set class	-
dict	No	dictionary	(named) list

How to index data

- Endian debate
  - Big Endian Byte Order: The most significant byte (the "big end") of the data is placed at the byte with the lowest address
    - Little Endian Byte Order: The least significant byte (the "little end") of the data is placed at the byte with the lowest address
    - Comes from *Gulliver's Travels*

(indexing data cont.)

- index from 0 or from 1?
  - where an index begins counting, when addressing elements of a data object
  - most languages index from 0 (https://en.wikipedia.org/wiki/Comparison_of_programming_languages_%28array%reference_list)
  - human ages - do they index from 0?

In [6]:
```
string_example = "Hello World"
string_example[0:5]
```

Out[6]:
```
'Hello'
```

- Python indexes from 0, R from 1. See R example here (R_indexing.ipynb)

git

- `git`: a revision control system
- Allows for complete history of changes, branching, staging areas, and flexible and distributed workflows

- simplified workflow (from Anita Cheng's excellent blog post (http://anitacheng.com/git-for-non-developers))

your computer

- Works through the command line, or through GUI clients, or through most IDEs and (good) editors

lectures — -bash — 94×25

```
KB-Office-iMac:lectures kbenoit$ cd ~/Dropbox/GitHub/lse-st445/lectures/
KB-Office-iMac:lectures kbenoit$ git add *
KB-Office-iMac:lectures kbenoit$ git commit -m "Update week 1 lectures"
[master ca47e74] Update week 1 lectures
 10 files changed, 1402 insertions(+)
 create mode 100644 week01/R_example.ipynb
 create mode 100644 week01/ST445_wk1_admin.ipynb
 create mode 100644 week01/ST445_wk1_class.ipynb
 create mode 100644 week01/ST445_wk1_lecture.ipynb
 create mode 100644 week01/figs/cardcatalog.jpg
 create mode 100644 week01/figs/cardcatalog2.jpg
 create mode 100644 week01/figs/photos.png
 create mode 100644 week01/figs/pianoroll.png
 create mode 100644 week01/figs/punchard.jpg
 create mode 100644 week01/figs/relational_data.png
KB-Office-iMac:lectures kbenoit$ git push
warning: redirecting to https://github.com/lse-st445/lectures/
Counting objects: 14, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (14/14), 5.57 MiB | 3.14 MiB/s, done.
Total 14 (delta 0), reused 0 (delta 0)
To http://github.com/lse-st445/lectures
   2d26aa9..ca47e74  master -> master
KB-Office-iMac:lectures kbenoit$
```

- or through editors (here, the excellent Atom (https://atom.io) editor)

GitHub

- a website and hosting platform for git repositories

- and so much more...
  - publishing websites: http://kenbenoit.net (http://kenbenoit.net), whose source code is at https://github.com/kbenoit/kbenoit.github.io (https://github.com/kbenoit/kbenoit.github.io)
  - "continous integration" hooks: https://github.com/kbenoit/spacyr (https://github.com/kbenoit/spacyr) (for instance - see the badges)
  - Issue tracking: https://github.com/kbenoit/spacyr/issues (https://github.com/kbenoit/spacyr/issues)
  - Inspecting code: (e.g.) https://github.com/kbenoit/spacyr/blob/master/R/python-functions.R (https://github.com/kbenoit/spacyr/blob/master/R/python-functions.R)

- GitHub classroom (https://classroom.github.com)
- Free stuff for students! https://education.github.com/pack (https://education.github.com/pack)

More great resources for using git/GitHub

- An easy git Cheatsheet (http://rogerdudler.github.io/git-guide/files/git_cheat_sheet.pdf), by Nina Jaeschke and Roger Dudler
- git - the simple guide (http://rogerdudler.github.io/git-guide/) by Roger Dudler

Some people have entire, open-source, user-commented books online, such as:

- *R for Data Science* (http://r4ds.had.co.nz)
  - with source code here (https://github.com/hadley/r4ds)
  - with GitHub issues here (https://github.com/hadley/r4ds/issues)
  - and pull requests - examples here (https://github.com/hadley/r4ds/pulls)

git Example

Fixing a broken R Jupyter notebook

This Jupyter notebook is broken and needs to be fixed:

https://github.com/lse-st445/lectures/blob/master/week01/R_example.ipynb (https://github.com/lse-st445/lectures/blob/master/week01/R_example.ipynb)

How to fix it:

- clone the repository
- edit the file
- stage the changes
- commit the changes
- issue a "pull request"

Markdown (and other markup languages)

- Idea of a "markup" language: HTML, XML, LaTeX
- "Markdown"
  - Created by John Gruber as a simple way for non-programming types to write in an easy-to-read format that could be converted directly into HTML
  - No opening or closing tags
  - Plain text, and can be read when not rendered
- Markdown has many "flavours" (https://github.com/commonmark/CommonMark/wiki/Markdown-Flavors)

Markdown example

This is a markdown example.

- bullet list 1
- bullet list 2

> *"I love deadlines. I like the whooshing sound they make as they fly by. (https://www.brainyquote.com/quotes/quotes/d/douglasada134151.html?src=t_funny)"*
> *-- Douglas Adams*

```
# Markdown example

This is a markdown example
* bullet list 1
* bullet list 2

> "[I love deadlines. I like the whooshing sound they make as they fly by.](http
s://www.brainyquote.com/quotes/quotes/d/douglasada134151.html?src=t_funny)"
-- _Douglas Adams_
```

Upcoming

- **Lab**: Working with Jupyter, working with Github, setting up a web page
- **Next week**: The shape of data