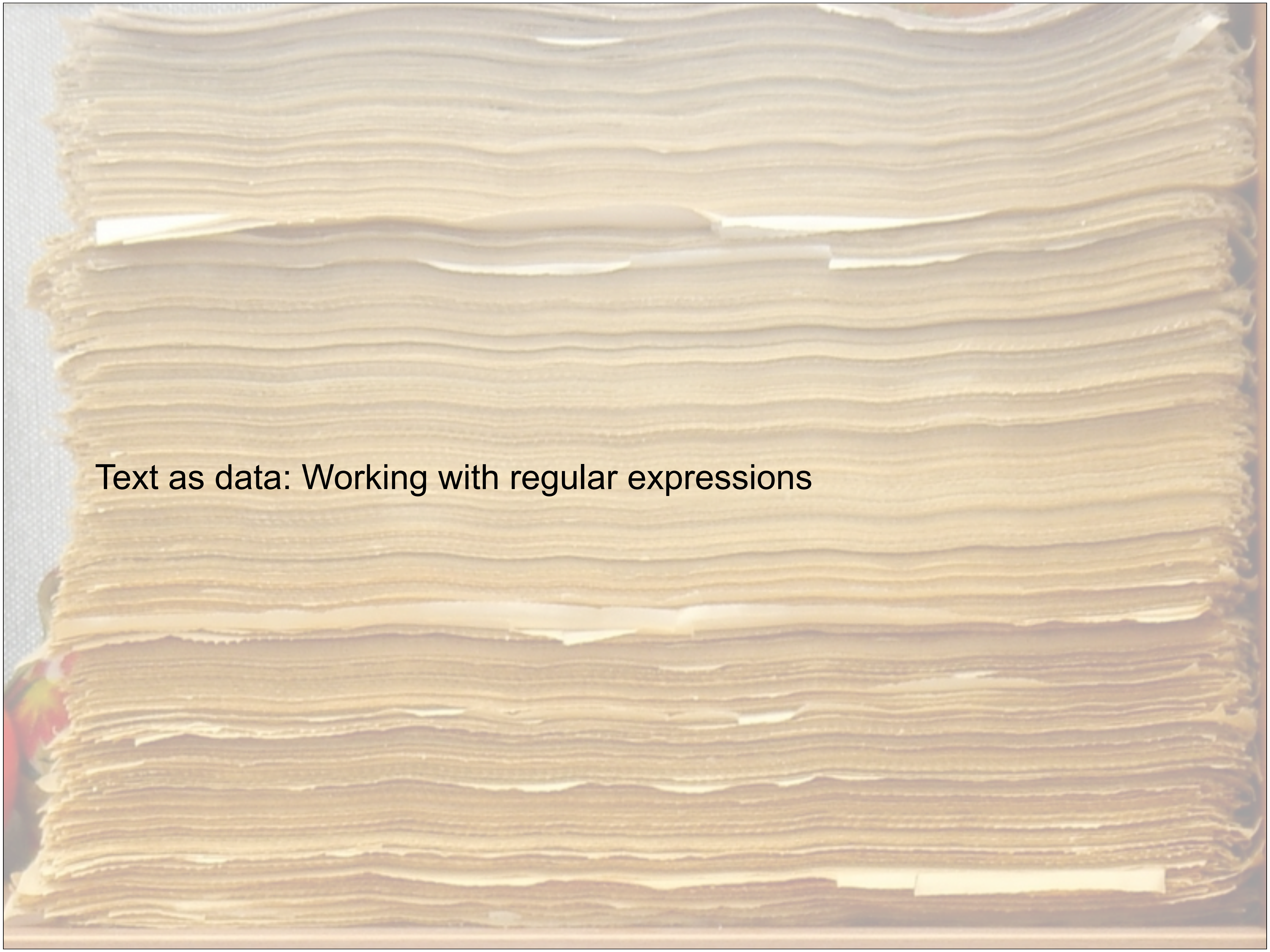


Text as data: Working with regular expressions

Text as data: Why we teach it

In the last three days, we’ve seen a couple of examples in which students have to manipulate text:

1. The Obama/Clinton county data set required record matching across disparate databases

2. The wireless data set involved extracting particular fields from a long data record

Text as data: Why we teach it

In my class, text manipulation seems to come up in almost every homework; formally, we have three specific text assignments

1. Analyze usage patterns of a web site based on a log file

   http://www.stat.ucla.edu/~cocteau/access_log.txt

2. Explore patterns of language usage in a day's worth of chat

   http://www.stat.ucla.edu/~cocteau/chat.txt

3. Identify the number of times reporters wrote for the NY Times

   http://www.stat.ucla.edu/~cocteau/1950.txt

```
134.226.32.57 - - [20/Sep/2007:07:54:29 -0700] "GET /~sczhu/icons/
daught.gif HTTP/1.0" 200 1898 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:29 -0700] "GET /~sczhu/icons/
bio.gif HTTP/1.0" 200 1681 "http://www.stat.ucla.edu/~sczhu/" "Mozilla/
5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/20070914
Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:30 -0700] "GET /~sczhu/
Zhu_LA_sm.gif HTTP/1.0" 200 39313 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:30 -0700] "GET /favicon.ico HTTP/
1.0" 200 318 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:
1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"

74.6.28.138 - - [20/Sep/2007:07:54:50 -0700] "GET /~nchristo/
statistics100B/syllabus100b.pdf HTTP/1.0" 200 47206 "-" "Mozilla/5.0
(compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/
slurp)"

164.67.132.219 - - [20/Sep/2007:07:54:55 -0700] "GET /robots.txt HTTP/
1.0" 200 559 "-" "gsa-crawler%20%28gsa1%2C%20contact%3A%20jhuang
%40ais.ucla.edu%29 (Enterprise; S5-J4JEBZS9PUJJA; jhuang@ais.ucla.edu)"

164.67.132.219 - - [20/Sep/2007:07:54:55 -0700] "GET /rss/feed.php?
unit=uclastat HTTP/1.0" 200 1739 "-" "gsa-crawler%20%28gsa1%2C
%20contact%3A%20jhuang%40ais.ucla.edu%29 (Enterprise; S5-J4JEBZS9PUJJA;
jhuang@ais.ucla.edu)"

134.226.32.57 - - [20/Sep/2007:07:55:03 -0700] "GET /%7Esczhu/
talks.html HTTP/1.0" 200 9489 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"
```

access_log.txt

apetalous - But, if that's the case, then again, it's a good market. Doesn't mean the stock is a good investment. That's the thing... you can't just invest in one stock and expect to make millions overnight, or even have your investment protected by the stock. It's a fluid game.

WHERE IS THE ISTHMUS OF CORINTH?

hey Pyro Pixie

BY WHAT PROCESS IS ROCK WORN DOWN BY THE WEATHER

what do you use when you script perl in windows?

englanddg, tin foil doesn't work.

ok devs, according to everything ive been told in every store ive gone to... 4 stores and 5 or 6 people... for what i need/want, 720 is fine

hugglez Witchen

that goes for all you opps

brb... need a break before i freakin choke LL

its all good

Heyya fellas

It must be copper foil.

How about movie?

31 f usa here

we're talking bout beer

```
By PAUL CROWELL
The New York Times (by Edward Hausner)
By LEE E.COOPER
By JOHN D. MORRIS Special to THE NEW YORK TIMES.
By KALMAN SEIGEL
By HAROLD FABER
The New York Times
By FELIX BELAIR Jr. Special to THE NEW YORK TIMES.
By LINDESAY PARROTT Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
By WALTER H. WAGGONER Special to THE NEW YORK TIMES.
By SANKA KNOX
By JAMES RESTON Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
By MILTON BRACKER Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
By TILLMAN DURDIN Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
By HANSON W.BALDWIN
By STANLEY LEVEY
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
Special to THE NEW YORK TIMES.
```

1950.txt

Text as data: Why we teach it

In general, we (as statisticians) are often asked to deal with rather complex patterns in data; sometimes they’re part of **the “clean up” process**, and sometimes the **presence or absence of patterns yield data**

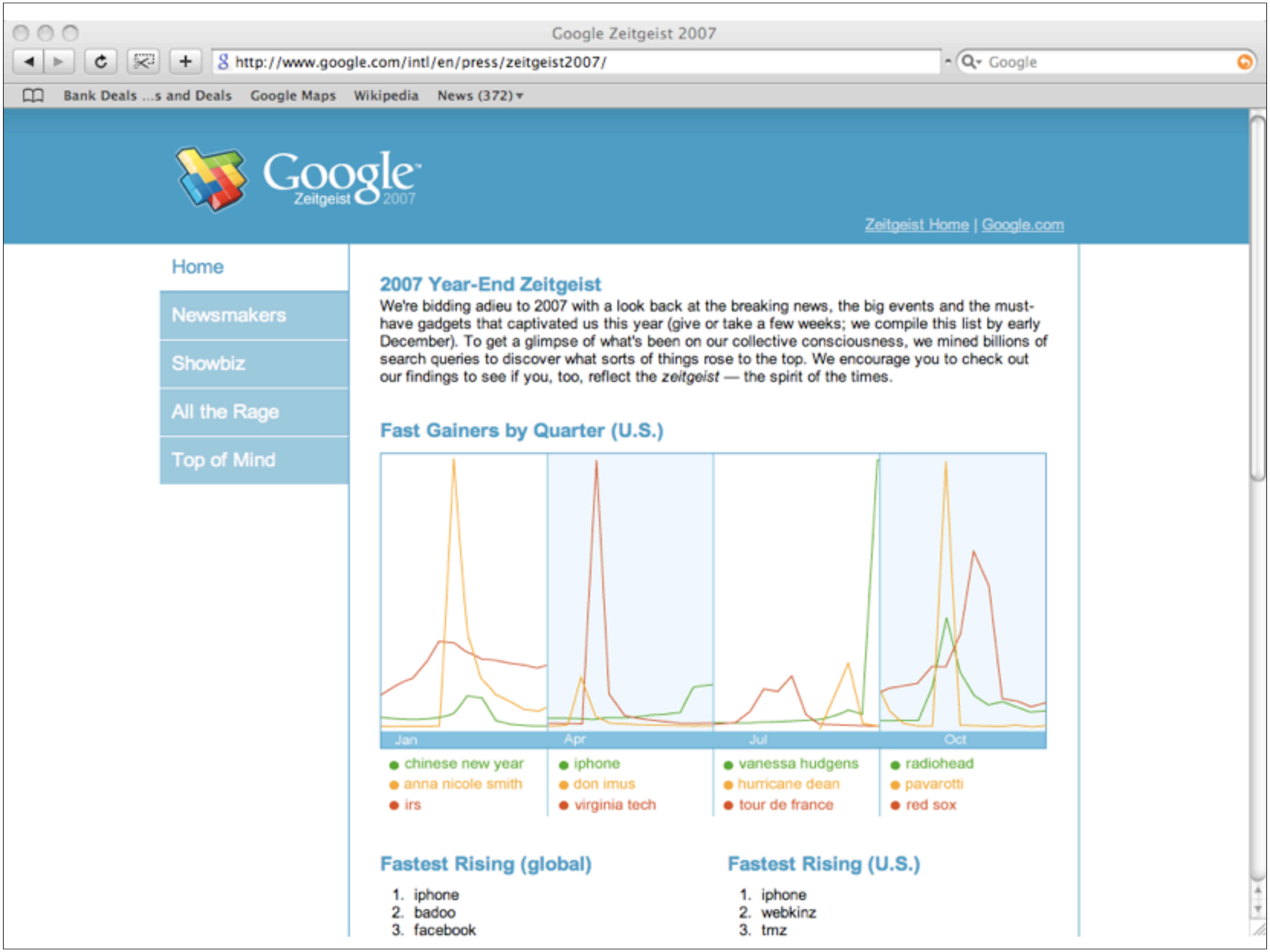
Text data is also plentiful it is relatively easy to create large amounts of interesting text data...

The Words That Were Used

Number of times President Bush used the following words or phrases in State of the Union addresses.

TERRORISM AND FOREIGN AFFAIRS

	2001 FEB. 27*	2002 JAN. 29	2003 JAN. 28	2004 JAN. 20	2005 FEB. 2	2006 JAN. 31
Afghanistan		13	3	5	3	2
Al Qaeda		1	8	3	2	3
Osama bin Laden						2
Democracy(ies)			1	4	9	7
Freedom	8	14	5	8	20	17
Saddam (Hussein)			19	5	2	
Iraq/Iraqi(s)		2	22	24	27	16
Sept. 11		5	3	3	1	2
Terror/terrorist(s)	1	33	19	19	26	17
Weapons of mass destruction	1	4	4	3	1	1



2007 Year-End Zeitgeist

We're bidding adieu to 2007 with a look back at the breaking news, the big events and the must-have gadgets that captivated us this year (give or take a few weeks; we compile this list by early December). To get a glimpse of what's been on our collective consciousness, we mined billions of search queries to discover what sorts of things rose to the top. We encourage you to check out our findings to see if you, too, reflect the *zeitgeist* — the spirit of the times.

Fast Gainers by Quarter (U.S.)

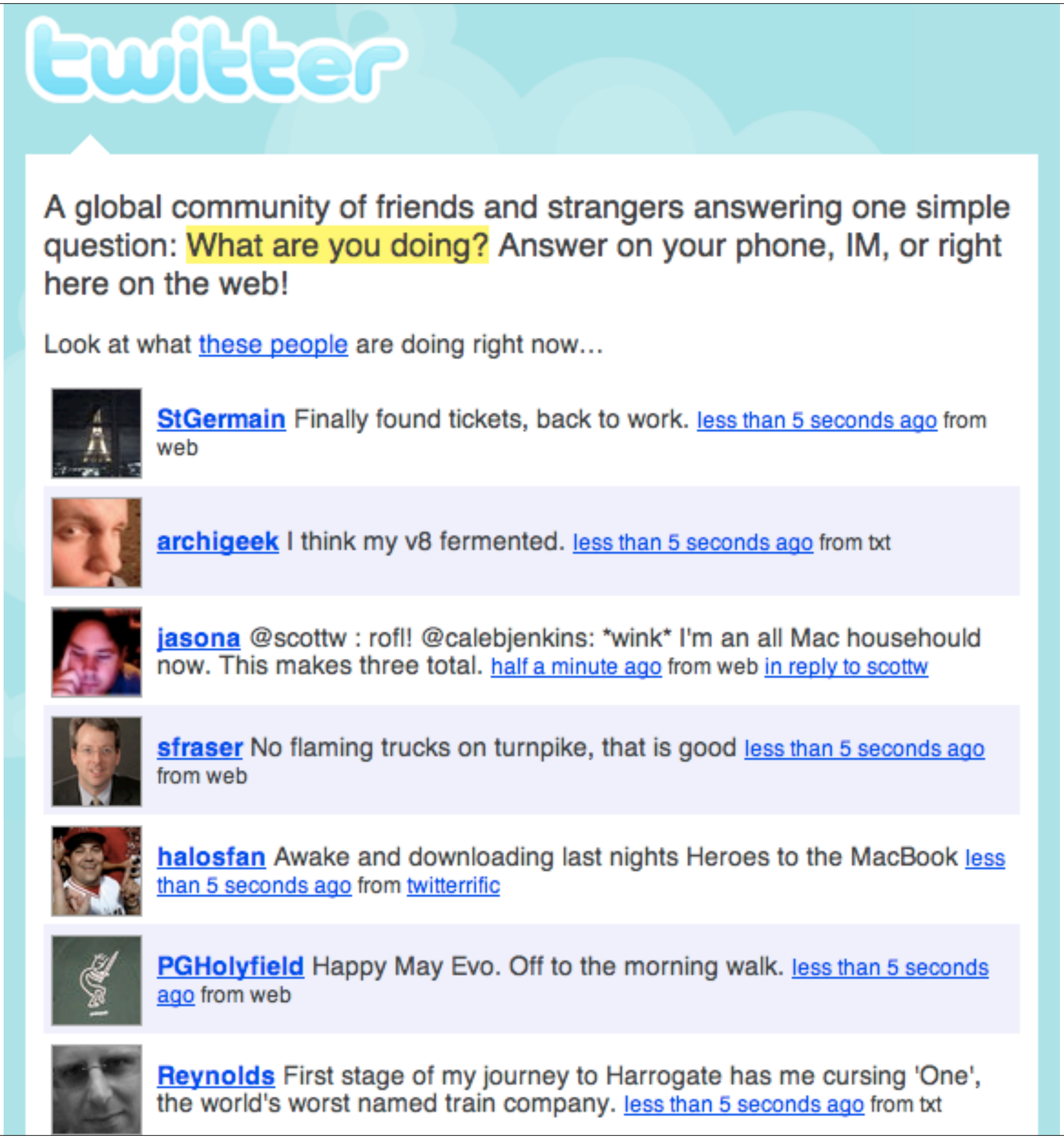
Jan	Apr	Jul	Oct
● chinese new year	● iphone	● vanessa hudgens	● radiohead
● anna nicole smith	● don imus	● hurricane dean	● pavarotti
● irs	● virginia tech	● tour de france	● red sox

Fastest Rising (global)

1. iphone
2. badoo
3. facebook

Fastest Rising (U.S.)

1. iphone
2. webkinz
3. tmz



A global community of friends and strangers answering one simple question: What are you doing? Answer on your phone, IM, or right here on the web!

Look at what these people are doing right now...

StGermain Finally found tickets, back to work. less than 5 seconds ago from web

archigeek I think my v8 fermented. less than 5 seconds ago from txt

jasona @scottw : rofl! @calebjenkins: *wink* I'm an all Mac househould now. This makes three total. half a minute ago from web in reply to scottw

sfraser No flaming trucks on turnpike, that is good less than 5 seconds ago from web

halosfan Awake and downloading last nights Heroes to the MacBook less than 5 seconds ago from twitterrific

PGHolyfield Happy May Evo. Off to the morning walk. less than 5 seconds ago from web

Reynolds First stage of my journey to Harrogate has me cursing 'One', the world's worst named train company. less than 5 seconds ago from txt

John Tukey

From Wikipedia, the free encyclopedia

John Wilder Tukey (June 16, 1915 - July 26, 2000) was an American statistician.

Contents [hide]

Biography [edit]

Tukey was born in New Bedford, Massachusetts in 1915, and obtained a B.A. in 1936 and M.Sc. in 1937, in Chemistry, from Brown University, before moving to Princeton University where he received his Ph.D. in mathematics.

During World War II, Tukey worked at the Fire Control Research Office and collaborated with Samuel Wilks and William Cochran. After the war, he returned to Princeton, dividing his time between the university and AT&T Bell Laboratories.

Among many contributions to civil society, Tukey served on a committee of the American Statistical Association that produced a report challenging the conclusions of the Kinsey Report, *Statistical Problems of the Kinsey Report on Sexual Behavior in the Human Male.*

He was awarded the IEEE Medal of Honor in 1982 "For his contributions to the spectral analysis of random processes and the fast Fourier transform (FFT) algorithm."

Tukey retired in 1985. In 2000, he died in New Brunswick, New Jersey.

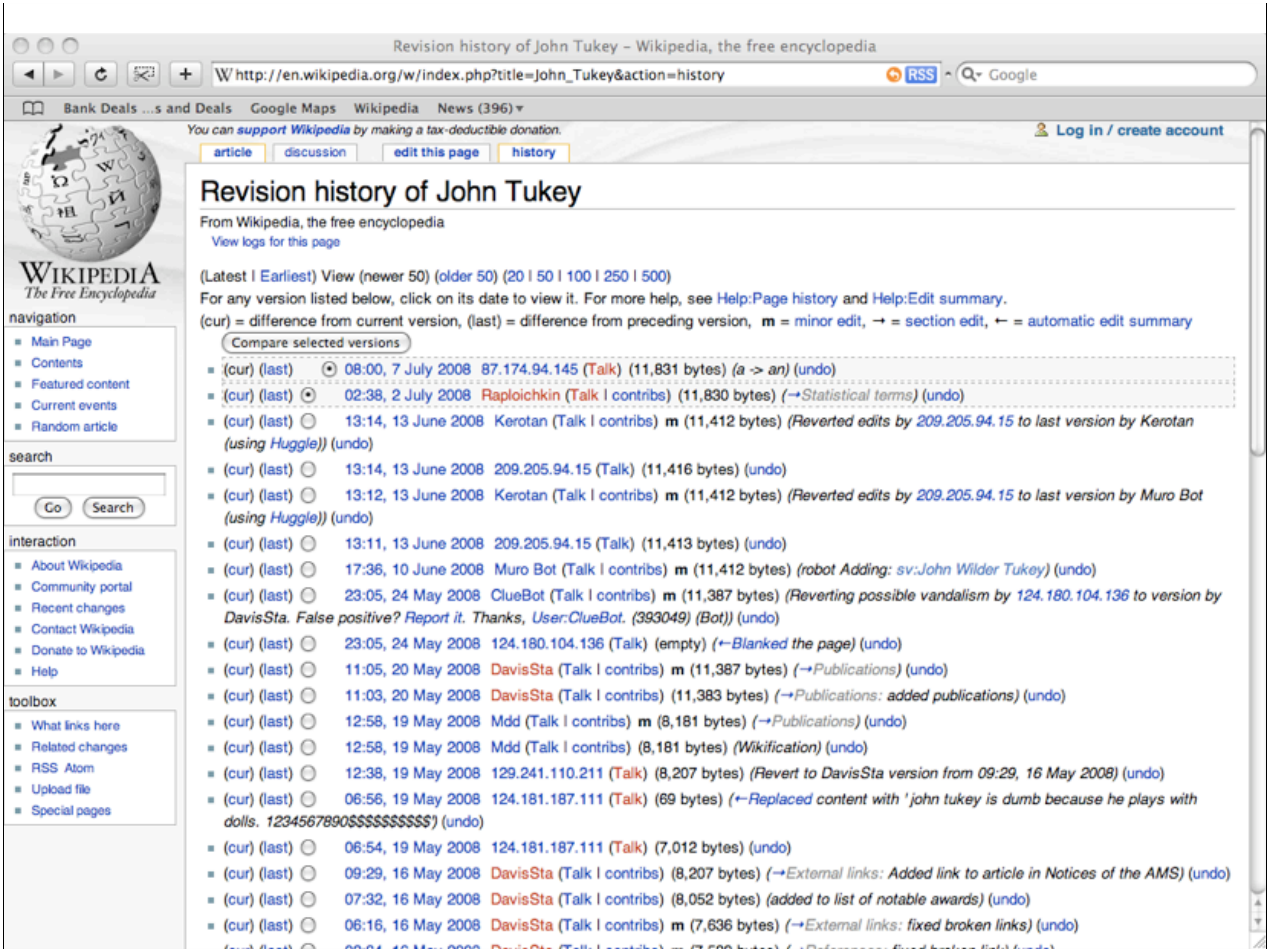
Scientific contributions [edit]

His statistical interests were many and varied. He is particularly remembered for his development

John Tukey

John Wilder Tukey

Born	June 16, 1915 New Bedford, Massachusetts, USA
Died	July 26, 2000 (aged 85) New Brunswick, New Jersey
Residence	United States
Nationality	American
Fields	Mathematician
Institutions	Bell Labs Princeton University
Alma mater	Brown University Princeton University
Doctoral advisor	Solomon Lefschetz
Doctoral students	Frederick Mosteller



You can support Wikipedia by making a tax-deductible donation.

Revision history of John Tukey

From Wikipedia, the free encyclopedia

View logs for this page

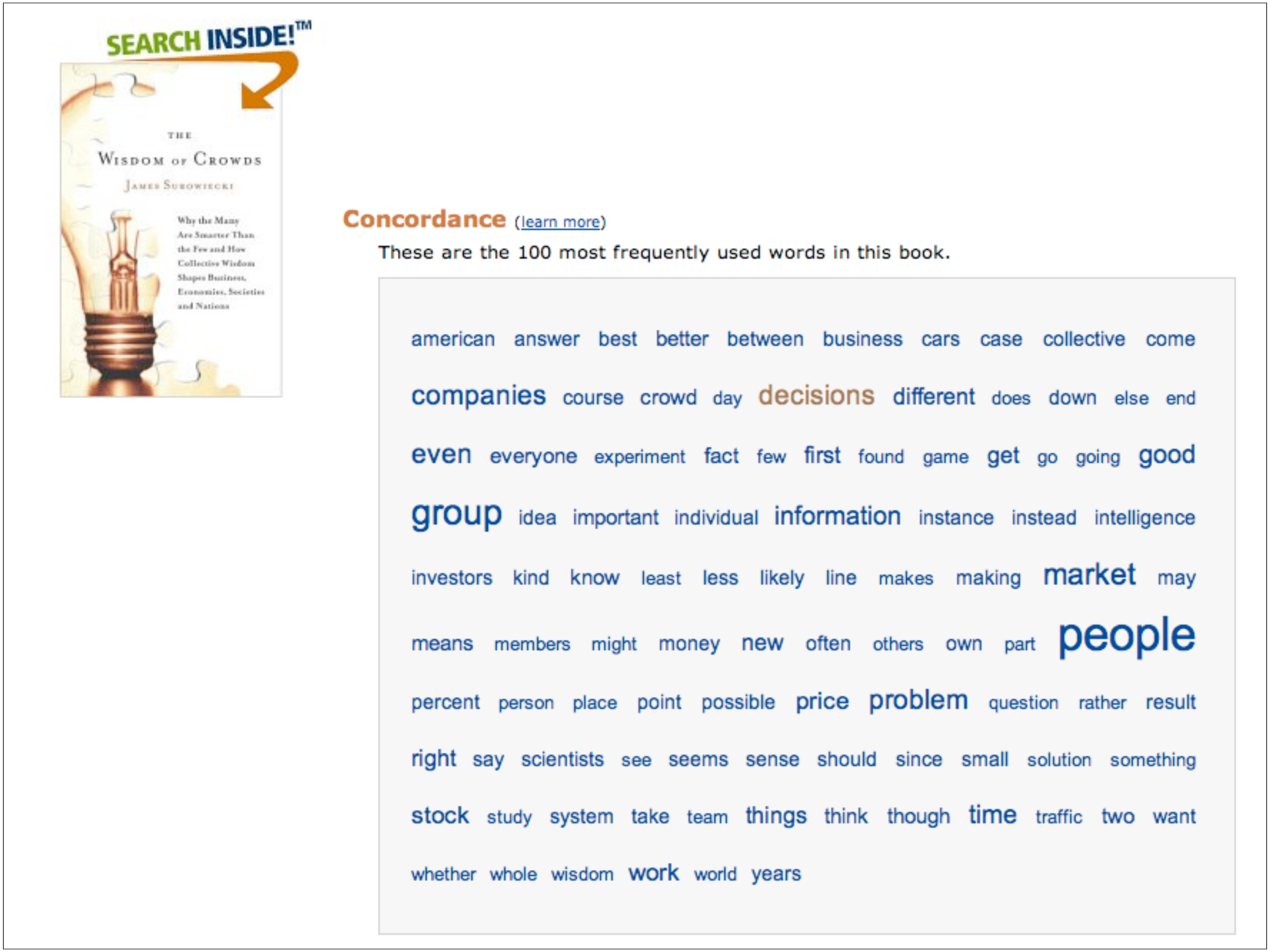
(Latest | Earliest) View (newer 50) (older 50) (20 | 50 | 100 | 250 | 500)

For any version listed below, click on its date to view it. For more help, see Help:Page history and Help:Edit summary.

(cur) = difference from current version, (last) = difference from preceding version, **m** = minor edit, → = section edit, ← = automatic edit summary

Compare selected versions

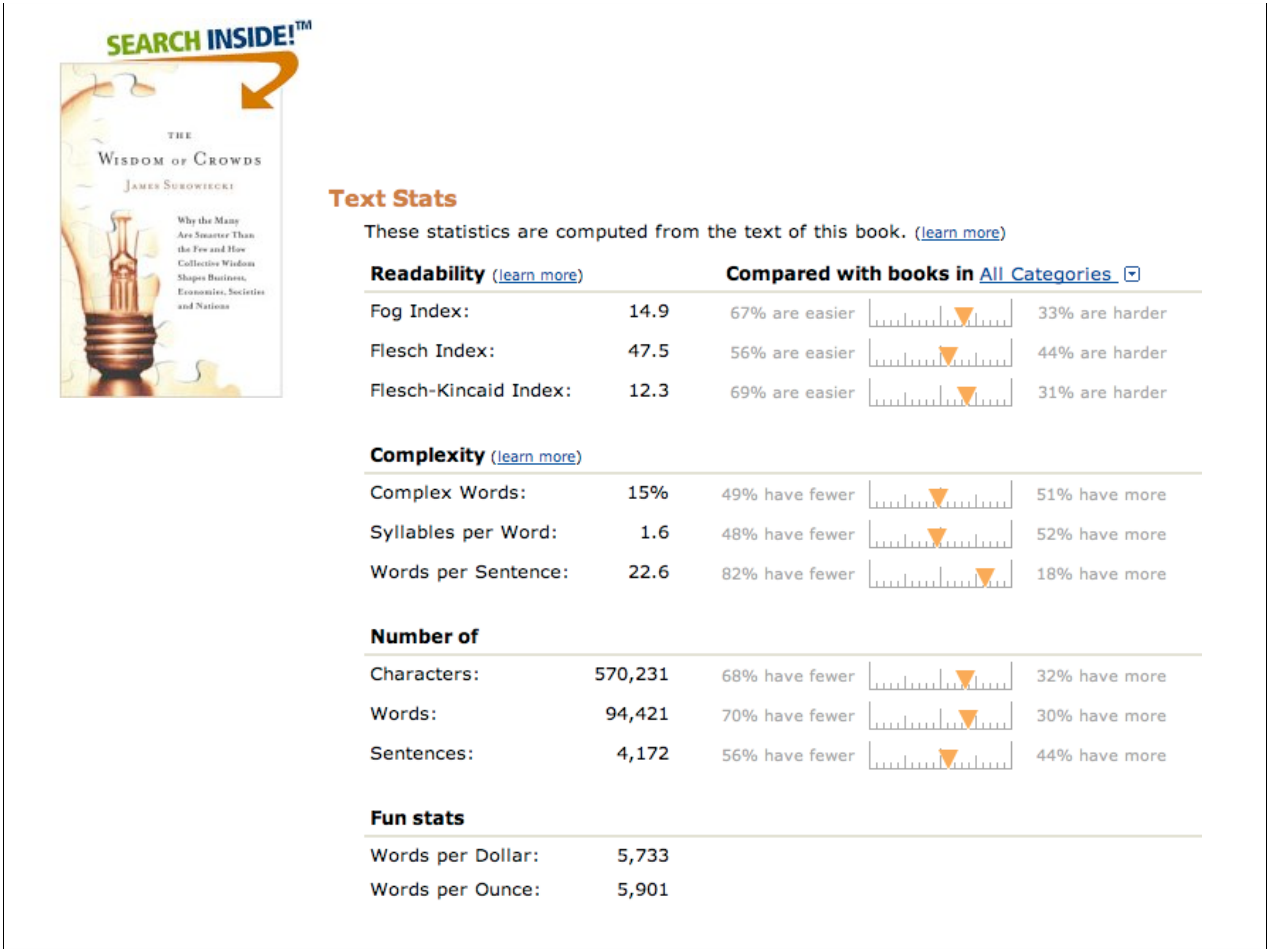
- (cur) (last) 08:00, 7 July 2008 87.174.94.145 (Talk) (11,831 bytes) *(a -> an)* (undo)
- (cur) (last) 02:38, 2 July 2008 Raploichkin (Talk | contribs) (11,830 bytes) *(→Statistical terms)* (undo)
- (cur) (last) 13:14, 13 June 2008 Kerotan (Talk | contribs) **m** (11,412 bytes) *(Reverted edits by 209.205.94.15 to last version by Kerotan (using Huggle))* (undo)
- (cur) (last) 13:14, 13 June 2008 209.205.94.15 (Talk) (11,416 bytes) (undo)
- (cur) (last) 13:12, 13 June 2008 Kerotan (Talk | contribs) **m** (11,412 bytes) *(Reverted edits by 209.205.94.15 to last version by Muro Bot (using Huggle))* (undo)
- (cur) (last) 13:11, 13 June 2008 209.205.94.15 (Talk) (11,413 bytes) (undo)
- (cur) (last) 17:36, 10 June 2008 Muro Bot (Talk | contribs) **m** (11,412 bytes) *(robot Adding: sv:John Wilder Tukey)* (undo)
- (cur) (last) 23:05, 24 May 2008 ClueBot (Talk | contribs) **m** (11,387 bytes) *(Reverting possible vandalism by 124.180.104.136 to version by DavisSta. False positive? Report it. Thanks, User:ClueBot. (393049) (Bot))* (undo)
- (cur) (last) 23:05, 24 May 2008 124.180.104.136 (Talk) (empty) *(←Blanked the page)* (undo)
- (cur) (last) 11:05, 20 May 2008 DavisSta (Talk | contribs) **m** (11,387 bytes) *(→Publications)* (undo)
- (cur) (last) 11:03, 20 May 2008 DavisSta (Talk | contribs) (11,383 bytes) *(→Publications: added publications)* (undo)
- (cur) (last) 12:58, 19 May 2008 Mdd (Talk | contribs) **m** (8,181 bytes) *(→Publications)* (undo)
- (cur) (last) 12:58, 19 May 2008 Mdd (Talk | contribs) (8,181 bytes) *(Wikification)* (undo)
- (cur) (last) 12:38, 19 May 2008 129.241.110.211 (Talk) (8,207 bytes) *(Revert to DavisSta version from 09:29, 16 May 2008)* (undo)
- (cur) (last) 06:56, 19 May 2008 124.181.187.111 (Talk) (69 bytes) *(←Replaced content with 'john tukey is dumb because he plays with dolls. 1234567890$$$$$$$$$$')* (undo)
- (cur) (last) 06:54, 19 May 2008 124.181.187.111 (Talk) (7,012 bytes) (undo)
- (cur) (last) 09:29, 16 May 2008 DavisSta (Talk | contribs) (8,207 bytes) *(→External links: Added link to article in Notices of the AMS)* (undo)
- (cur) (last) 07:32, 16 May 2008 DavisSta (Talk | contribs) (8,052 bytes) *(added to list of notable awards)* (undo)
- (cur) (last) 06:16, 16 May 2008 DavisSta (Talk | contribs) **m** (7,636 bytes) *(→External links: fixed broken links)* (undo)


SEARCH INSIDE!™
THE
Wisdom of Crowds
James Surowiecki
Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations

Concordance (learn more)

These are the 100 most frequently used words in this book.

american answer best better between business cars case collective come companies course crowd day decisions different does down else end even everyone experiment fact few first found game get go going good group idea important individual information instance instead intelligence investors kind know least less likely line makes making market may means members might money new often others own part people percent person place point possible price problem question rather result right say scientists see seems sense should since small solution something stock study system take team things think though time traffic two want whether whole wisdom work world years


SEARCH INSIDE!™
THE
Wisdom of Crowds
James Surowiecki
Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations

Text Stats

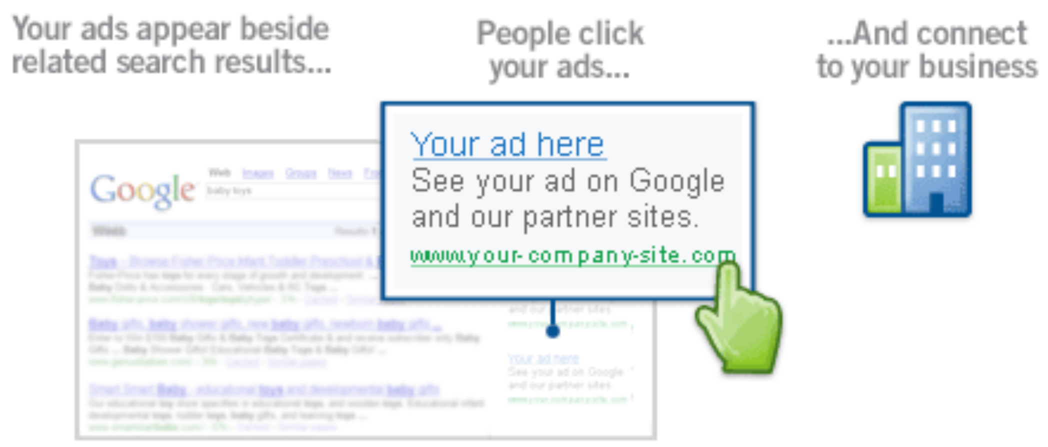
These statistics are computed from the text of this book. (learn more)

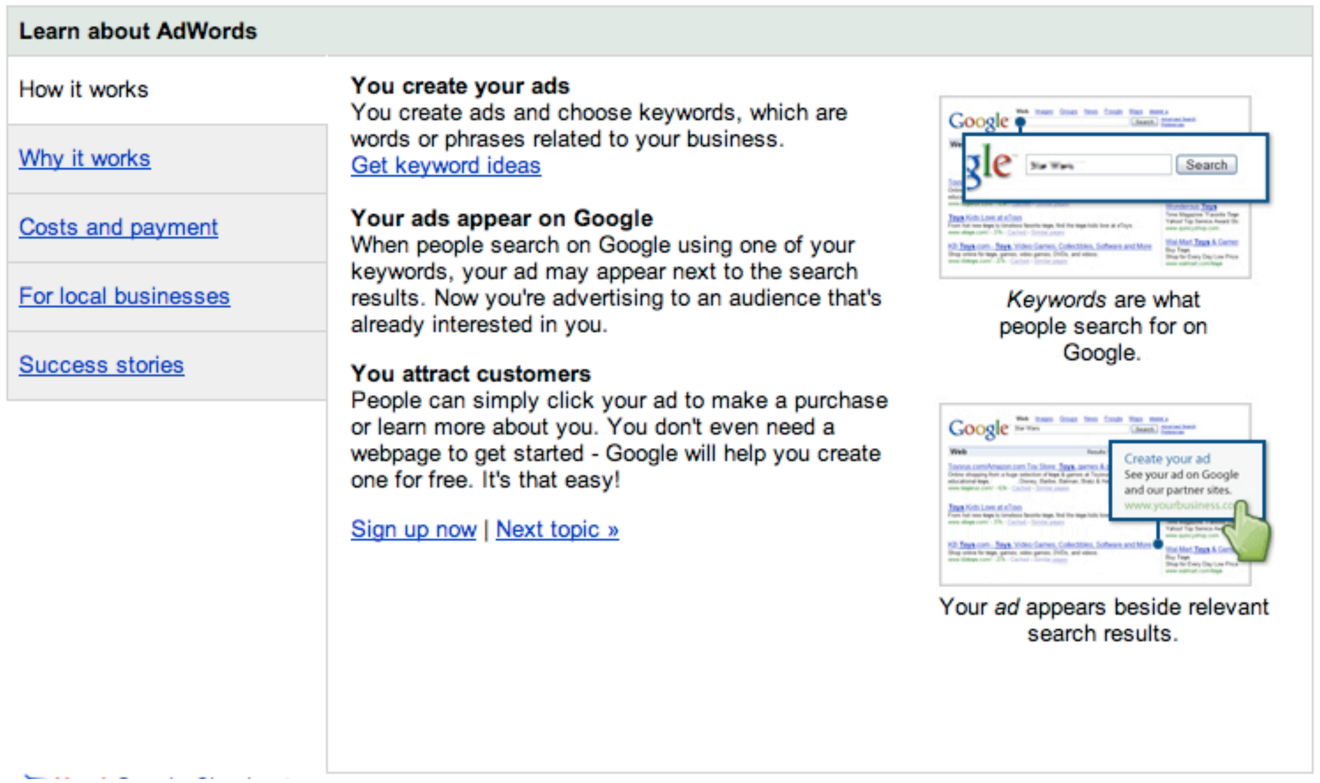
Readability (learn more)		**Compared with books in** All Categories		
Fog Index:	14.9	67% are easier		33% are harder
Flesch Index:	47.5	56% are easier		44% are harder
Flesch-Kincaid Index:	12.3	69% are easier		31% are harder

Complexity (learn more)				
Complex Words:	15%	49% have fewer		51% have more
Syllables per Word:	1.6	48% have fewer		52% have more
Words per Sentence:	22.6	82% have fewer		18% have more

Number of				
Characters:	570,231	68% have fewer		32% have more
Words:	94,421	70% have fewer		30% have more
Sentences:	4,172	56% have fewer		44% have more

Fun stats	
Words per Dollar:	5,733
Words per Ounce:	5,901


Your ads appear beside related search results...
People click your ads...
...And connect to your business
Google
Your ad here
See your ad on Google and our partner sites.
www.your-company-site.com



Learn about AdWords

How it works

Why it works

Costs and payment

For local businesses

Success stories

You create your ads
You create ads and choose keywords, which are words or phrases related to your business.
Get keyword ideas

Your ads appear on Google
When people search on Google using one of your keywords, your ad may appear next to the search results. Now you're advertising to an audience that's already interested in you.

You attract customers
People can simply click your ad to make a purchase or learn more about you. You don't even need a webpage to get started - Google will help you create one for free. It's that easy!

Sign up now | Next topic »

Keywords are what people search for on Google.

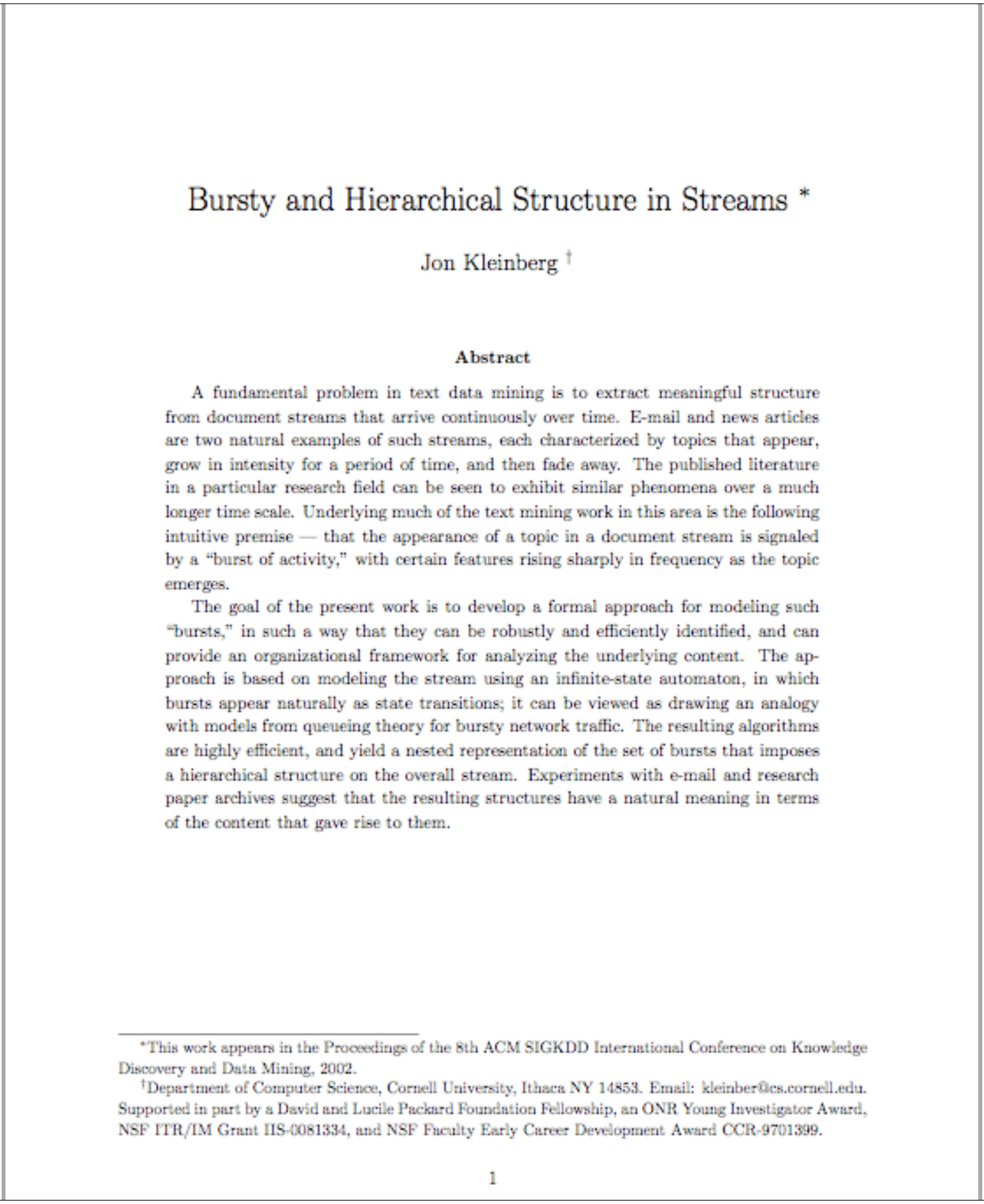
Your *ad* appears beside relevant search results.

Text as data: Why we teach it

From early problems in authorship attribution to more recent work in large-scale text mining, there’s plenty of interesting problems and data sources to analyze

Artifacts from computer-mediated communication (web logs, bulletin boards, chat transcripts, email) all provide complex and socially interesting data for students to work with

These sources can be more compelling for students; in some sense they are closer to home, are more recognizable than certain scientific data sources, and can kick off important discussions about privacy and computer technologies (OK, I might be the only one interested in that)



Bursty and Hierarchical Structure in Streams [*]

Jon Kleinberg [†]

Abstract

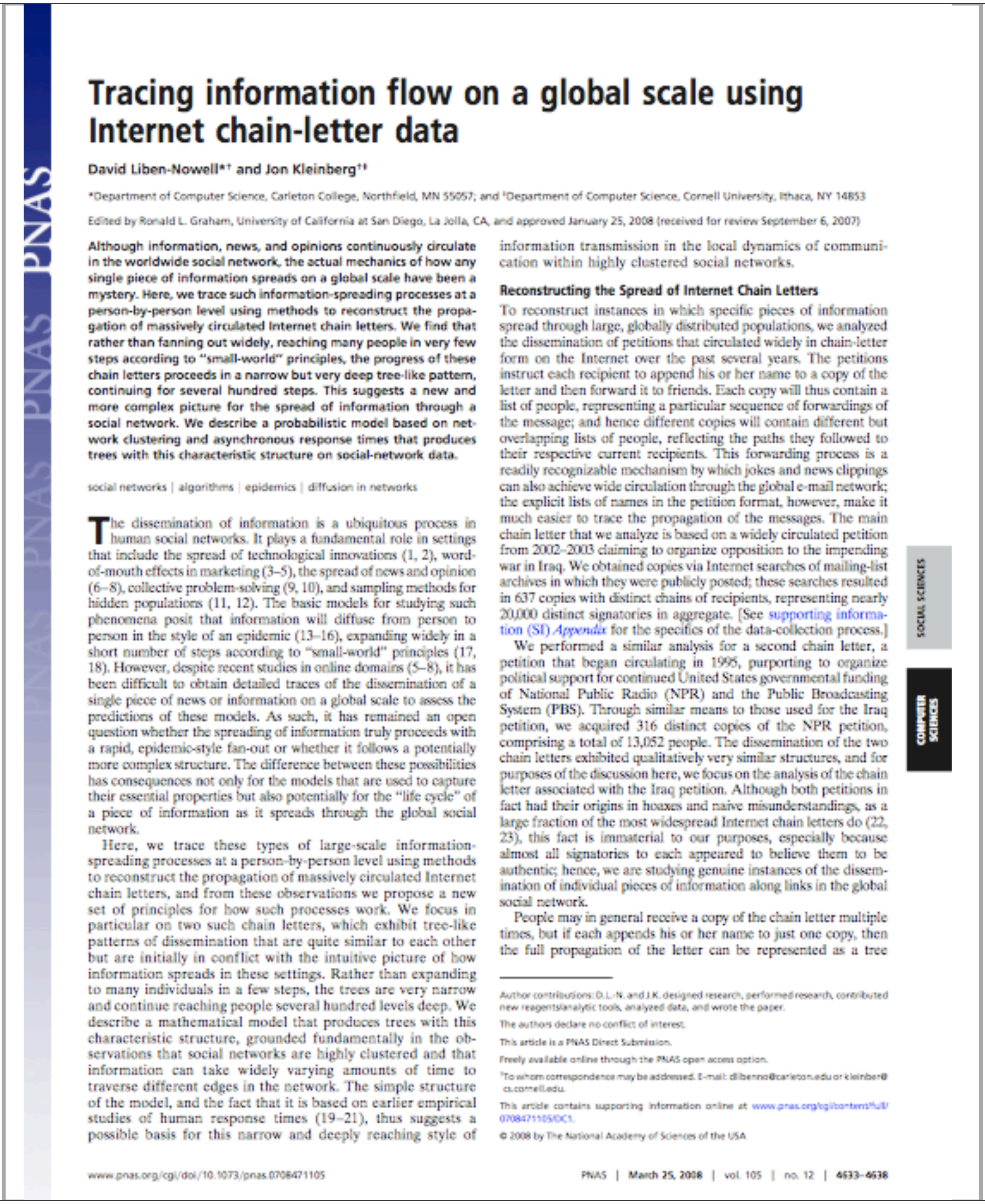
A fundamental problem in text data mining is to extract meaningful structure from document streams that arrive continuously over time. E-mail and news articles are two natural examples of such streams, each characterized by topics that appear, grow in intensity for a period of time, and then fade away. The published literature in a particular research field can be seen to exhibit similar phenomena over a much longer time scale. Underlying much of the text mining work in this area is the following intuitive premise — that the appearance of a topic in a document stream is signaled by a "burst of activity," with certain features rising sharply in frequency as the topic emerges.

The goal of the present work is to develop a formal approach for modeling such "bursts," in such a way that they can be robustly and efficiently identified, and can provide an organizational framework for analyzing the underlying content. The approach is based on modeling the stream using an infinite-state automaton, in which bursts appear naturally as state transitions; it can be viewed as drawing an analogy with models from queueing theory for bursty network traffic. The resulting algorithms are highly efficient, and yield a nested representation of the set of bursts that imposes a hierarchical structure on the overall stream. Experiments with e-mail and research paper archives suggest that the resulting structures have a natural meaning in terms of the content that gave rise to them.

[*]This work appears in the Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002.

[†]Department of Computer Science, Cornell University, Ithaca NY 14853. Email: kleinber@cs.cornell.edu. Supported in part by a David and Lucile Packard Foundation Fellowship, an ONR Young Investigator Award, NSF ITR/IM Grant IIS-0081334, and NSF Faculty Early Career Development Award CCR-9701399.

www.cs.cornell.edu/home/kleinber/



PNAS PNAS PNAS PNAS

Tracing information flow on a global scale using Internet chain-letter data

David Liben-Nowell*† and Jon Kleinberg†‡

*Department of Computer Science, Carleton College, Northfield, MN 55057; and †Department of Computer Science, Cornell University, Ithaca, NY 14853

Edited by Ronald L. Graham, University of California at San Diego, La Jolla, CA, and approved January 25, 2008 (received for review September 6, 2007)

Although information, news, and opinions continuously circulate in the worldwide social network, the actual mechanics of how any single piece of information spreads on a global scale have been a mystery. Here, we trace such information-spreading processes at a person-by-person level using methods to reconstruct the propagation of massively circulated Internet chain letters. We find that rather than fanning out widely, reaching many people in very few steps according to "small-world" principles, the progress of these chain letters proceeds in a narrow but very deep tree-like pattern, continuing for several hundred steps. This suggests a new and more complex picture for the spread of information through a social network. We describe a probabilistic model based on network clustering and asynchronous response times that produces trees with this characteristic structure on social-network data.

social networks | algorithms | epidemics | diffusion in networks

The dissemination of information is a ubiquitous process in human social networks. It plays a fundamental role in settings that include the spread of technological innovations (1, 2), word-of-mouth effects in marketing (3–5), the spread of news and opinion (6–8), collective problem-solving (9, 10), and sampling methods for hidden populations (11, 12). The basic models for studying such phenomena posit that information will diffuse from person to person in the style of an epidemic (13–16), expanding widely in a short number of steps according to "small-world" principles (17, 18). However, despite recent studies in online domains (5–8), it has been difficult to obtain detailed traces of the dissemination of a single piece of news or information on a global scale to assess the predictions of these models. As such, it has remained an open question whether the spreading of information truly proceeds with a rapid, epidemic-style fan-out or whether it follows a potentially more complex structure. The difference between these possibilities has consequences not only for the models that are used to capture their essential properties but also potentially for the "life cycle" of a piece of information as it spreads through the global social network.

Here, we trace these types of large-scale information-spreading processes at a person-by-person level using methods to reconstruct the propagation of massively circulated Internet chain letters, and from these observations we propose a new set of principles for how such processes work. We focus in particular on two such chain letters, which exhibit tree-like patterns of dissemination that are quite similar to each other but are initially in conflict with the intuitive picture of how information spreads in these settings. Rather than expanding to many individuals in a few steps, the trees are very narrow and continue reaching people several hundred levels deep. We describe a mathematical model that produces trees with this characteristic structure, grounded fundamentally in the observations that social networks are highly clustered and that information can take widely varying amounts of time to traverse different edges in the network. The simple structure of the model, and the fact that it is based on earlier empirical studies of human response times (19–21), thus suggests a possible basis for this narrow and deeply reaching style of information transmission in the local dynamics of communication within highly clustered social networks.

Reconstructing the Spread of Internet Chain Letters

To reconstruct instances in which specific pieces of information spread through large, globally distributed populations, we analyzed the dissemination of petitions that circulated widely in chain-letter form on the Internet over the past several years. The petitions instruct each recipient to append his or her name to a copy of the letter and then forward it to friends. Each copy will thus contain a list of people, representing a particular sequence of forwardings of the message; and hence different copies will contain different but overlapping lists of people, reflecting the paths they followed to their respective current recipients. This forwarding process is a readily recognizable mechanism by which jokes and news clippings can also achieve wide circulation through the global e-mail network; the explicit lists of names in the petition format, however, make it much easier to trace the propagation of the messages. The main chain letter that we analyze is based on a widely circulated petition from 2002–2003 claiming to organize opposition to the impending war in Iraq. We obtained copies via Internet searches of mailing-list archives in which they were publicly posted; these searches resulted in 637 copies with distinct chains of recipients, representing nearly 20,000 distinct signatories in aggregate. [See supporting information (SI) *Appendix* for the specifics of the data-collection process.]

We performed a similar analysis for a second chain letter, a petition that began circulating in 1995, purporting to organize political support for continued United States governmental funding of National Public Radio (NPR) and the Public Broadcasting System (PBS). Through similar means to those used for the Iraq petition, we acquired 316 distinct copies of the NPR petition, comprising a total of 13,052 people. The dissemination of the two chain letters exhibited qualitatively very similar structures, and for purposes of the discussion here, we focus on the analysis of the chain letter associated with the Iraq petition. Although both petitions in fact had their origins in hoaxes and naive misunderstandings, as a large fraction of the most widespread Internet chain letters do (22, 23), this fact is immaterial to our purposes, especially because almost all signatories to each appeared to believe them to be authentic; hence, we are studying genuine instances of the dissemination of individual pieces of information along links in the global social network.

People may in general receive a copy of the chain letter multiple times, but if each appends his or her name to just one copy, then the full propagation of the letter can be represented as a tree

SOCIAL SCIENCES

COMPUTER SCIENCES

Author contributions: D.L.-N. and J.K. designed research, performed research, contributed new reagents/analytic tools, analyzed data, and wrote the paper.

The authors declare no conflict of interest.

This article is a PNAS Direct Submission.

Freely available online through the PNAS open access option.

‡To whom correspondence may be addressed. E-mail: dlibenno@carleton.edu or kleinber@cs.cornell.edu.

This article contains supporting information online at www.pnas.org/cgi/content/full/0708471105/DC1.

www.cs.cornell.edu/home/kleinber/

Tools to manipulate text - R

There are a number of languages and tools that can manipulate text; your choice in an introductory computing course will depend on when you take up the subject, your audience, and your goals for the class

As an example, R has some basic facilities to deal with character strings and vectors; there are functions to extract or replace substrings, to split strings into pieces, and to identify patterns

Let’s have a look at these...

```
> txt = "here's an experiment with text."
> nchar(txt)              # the length of the string
[1] 31

> substr(txt,8,15)        # create a substring from char 8 to 15
[1] "an exper"

> strsplit(txt," ")       # divide on ‘whitespace’
[[1]]
[1] "here's"     "an"         "experiment" "with"       "text."

> strsplit(txt,"an")      # divide on ‘an’
[[1]]
[1] "here's "                "  experiment with text."

> sub("experiment","simple attempt",txt)     # substitution once
[1] "here's an simple attempt with text."

> gsub("e","i",txt)                          # ... and multiple substitutions
[1] "hiri's an ixpirimint with tixt."

> toupper(txt)                               # changing case
[1] "HERE'S AN EXPERIMENT WITH TEXT."
```

Tools to manipulate text - R

On the previous slide, we gave examples of the functions ***`sub`***, ***`gsub`***, and ***`strsplit`*** involving simple replacement and matching patterns; these functions operate with a more elaborate language for defining patterns, a construction we'll get to in a few minutes

As the "motivating" examples I presented at the beginning illustrate, we are usually not working quietly with **a single string**, but instead we have to "process" a number of lines stored in **a character vector**

In a simple example that both Deb and I use, we examine the logs created by a web server...

Example: Web server logs

Each time you request a file from a web site, a line is appended to the bottom of **a log file**; a file that quietly and continually records all the activity taking place

All of the actions from the millions (or thousands or hundreds or, as in the case of our own UCLA Statistics site, tens) of people browsing your site at any moment are added to this file **in time order**

What kinds of information might a log of this kind collect? What would you, as a site owner want to know?

```
> alog = scan("access_log.txt",what="",sep="\n")
Read 50000 items

> alog = scan(url("http://www.stat.ucla.edu/~cocteau/access_log.txt"),what="",sep="\n")
Read 50000 items

> alog[1]
[1] "134.226.32.57 - - [20/Sep/2007:07:54:29 -0700] \"GET /~sczhu/icons/daught.gif HTTP/
1.0\" 200 1898 \"http://www.stat.ucla.edu/~sczhu/\" \"Mozilla/5.0 (Windows; U; Windows NT
5.1; en-US; rv:1.8.1.7) Gecko/20070914 Firefox/2.0.0.7\""

> alog[50000]
[1] "165.134.208.6 - - [21/Sep/2007:12:05:03 -0700] \"GET /favicon.ico HTTP/1.1\" 200 318
\"-\" \"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.0.12) Gecko/20070508
Firefox/1.5.0.12\""
```

Combined log format

The format of this file is fairly standardized; each line contains a series of records about the transaction it represents

- IP address
- Identity
- Userid
- date
- Request
- Status
- Bytes
- Referrer
- Agent

Let’s start by considering the “Status” or success/failure of each transaction in our sample

Server status...

A fast Google search gives us a list of possible errors

Note that Error 200 actually means a success

Error 206 means that only part of the file was delivered; the user cancelled the request before it could be delivered

Error 304 is “not modified”; sometimes clients perform conditional GET requests

HTTP Error 101
Switching Protocols. Again, not really an "error", this HTTP Status Code means everything is working fine.

HTTP Error 200
Success. This HTTP Status Code means everything is working fine. However, if you receive this message on screen, obviously something is not right... Please contact the server's administrator if this problem persists. Typically, this status code (as well as most other 200 Range codes) will only be written to your server logs.

HTTP Error 201
Created. A new resource has been created successfully on the server.

HTTP Error 202
Accepted. Request accepted but not completed yet, it will continue asynchronously.

HTTP Error 203
Non-Authoritative Information. Request probably completed successfully but can't tell from original server.

HTTP Error 204
No Content. The requested completed successfully but the resource requested is empty (has zero length).

HTTP Error 205
Reset Content. The requested completed successfully but the client should clear down any cached information as it may now be invalid.

HTTP Error 206
Partial Content. The request was canceled before it could be fulfilled. Typically the user gave up waiting for data and went to another page. Some download accelerator programs produce this error as they submit multiple requests to download a file at the same time.

HTTP Error 300
Multiple Choices. The request is ambiguous and needs clarification as to which resource was requested.

HTTP Error 301
Moved Permanently. The resource has permanently moved elsewhere, the response indicates where it has gone to.

HTTP Error 302
Moved Temporarily. The resource has temporarily moved elsewhere, the response indicates where it is at present.

HTTP Error 303
See Other/Redirect. A preferred alternative source should be used at present.

```
> strsplit(alog[1]," ")[1]
[[1]]
 [1] "134.226.32.57"
 [2] "-"
 [3] "-"
 [4] "[20/Sep/2007:07:54:29"
 [5] "-0700]"
 [6] "\"GET"
 [7] "/~sczhu/icons/daught.gif"
 [8] "HTTP/1.0\""
 [9] "200"
[10] "1898"
[11] "\"http://www.stat.ucla.edu/~sczhu/\""
[12] "\"Mozilla/5.0"
[13] "(Windows;"
[14] "U;"
[15] "Windows"
[16] "NT"
[17] "5.1;"
[18] "en-US;"
[19] "rv:1.8.1.7)"
[20] "Gecko/20070914"
[21] "Firefox/2.0.0.7\""

> errs = sapply(strsplit(alog," "),function(x) x[9])        # isolate errors

> table(errs)                                               # table ‘em

  200   206   301   302   304   400   401   403   404   405
41093  1501  1431   223  3704     4     4   367  1669     4
```

```
> strsplit(alog[1]," ")[1]
[[1]]
 [1] "134.226.32.57"
 [2] "-"
 [3] "-"
 [4] "[20/Sep/2007:07:54:29"
 [5] "-0700]"
 [6] "\"GET"
 [7] "/~sczhu/icons/daught.gif"
 [8] "HTTP/1.0\""
 [9] "200"
[10] "1898"
[11] "\"http://www.stat.ucla.edu/~sczhu/\""
[12] "\"Mozilla/5.0"
[13] "(Windows;"
[14] "U;"
[15] "Windows"
[16] "NT"
[17] "5.1;"
[18] "en-US;"
[19] "rv:1.8.1.7)"
[20] "Gecko/20070914"
[21] "Firefox/2.0.0.7\""

> ips = sapply(strsplit(alog," "),function(x) x[1])    # pull out ip addrs

> sort(table(ips),decreasing=TRUE)[1:10]               # ... and sort ‘em

70.184.223.117  128.97.55.194 164.67.132.220 76.167.214.187  128.97.86.248
         13050           1626            757            614            520
208.68.136.250 164.67.132.219   66.249.73.99   138.9.25.242      24.5.6.46
           519            408            232            217            217
```

Moving on

Going along in this way, we could start to answer the sort of questions that are somewhat standard for traffic analysis services: When is the server active? What kinds of errors do we see? Who are my most frequent users?

In the end, we want to peer into these logs a little more deeply and examine what is being accessed; this data is found in the eleventh field of the split log-line

A deeper "content" analysis requires a more expressive language for describing patterns

NEED WEBSITE TRAFFIC?

Starting Now From.

$29

immense HITS

Blazing Traffic
Real Visitors for Your Website
Delivered 24 hours a day - GUARANTEED!

```
134.226.32.57 - - [20/Sep/2007:07:54:29 -0700] "GET /~sczhu/icons/
daught.gif HTTP/1.0" 200 1898 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:29 -0700] "GET /~sczhu/icons/
bio.gif HTTP/1.0" 200 1681 "http://www.stat.ucla.edu/~sczhu/" "Mozilla/
5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/20070914
Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:30 -0700] "GET /~sczhu/
Zhu_LA_sm.gif HTTP/1.0" 200 39313 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:30 -0700] "GET /favicon.ico HTTP/
1.0" 200 318 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:
1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"

74.6.28.138 - - [20/Sep/2007:07:54:50 -0700] "GET /~nchristo/
statistics100B/syllabus100b.pdf HTTP/1.0" 200 47206 "-" "Mozilla/5.0
(compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/
slurp)"

164.67.132.219 - - [20/Sep/2007:07:54:55 -0700] "GET /robots.txt HTTP/
1.0" 200 559 "-" "gsa-crawler%20%28gsa1%2C%20contact%3A%20jhuang
%40ais.ucla.edu%29 (Enterprise; S5-J4JEBZS9PUJJA; jhuang@ais.ucla.edu)"

164.67.132.219 - - [20/Sep/2007:07:54:55 -0700] "GET /rss/feed.php?
unit=uclastat HTTP/1.0" 200 1739 "-" "gsa-crawler%20%28gsa1%2C
%20contact%3A%20jhuang%40ais.ucla.edu%29 (Enterprise; S5-J4JEBZS9PUJJA;
jhuang@ais.ucla.edu)"

134.226.32.57 - - [20/Sep/2007:07:55:03 -0700] "GET /%7Esczhu/
talks.html HTTP/1.0" 200 9489 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"
```

```
134.226.32.57 - - [20/Sep/2007:07:54:29 -0700] "GET /~sczhu/icons/
daught.gif HTTP/1.0" 200 1898 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:29 -0700] "GET /~sczhu/icons/
bio.gif HTTP/1.0" 200 1681 "http://www.stat.ucla.edu/~sczhu/" "Mozilla/
5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/20070914
Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:30 -0700] "GET /~sczhu/
Zhu_LA_sm.gif HTTP/1.0" 200 39313 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"

134.226.32.57 - - [20/Sep/2007:07:54:30 -0700] "GET /favicon.ico HTTP/
1.0" 200 318 "-" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:
1.8.1.7) Gecko/20070914 Firefox/2.0.0.7"

74.6.28.138 - - [20/Sep/2007:07:54:50 -0700] "GET /~nchristo/
statistics100B/syllabus100b.pdf HTTP/1.0" 200 47206 "-" "Mozilla/5.0
(compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/
slurp)"

164.67.132.219 - - [20/Sep/2007:07:54:55 -0700] "GET /robots.txt HTTP/
1.0" 200 559 "-" "gsa-crawler%20%28gsa1%2C%20contact%3A%20jhuang
%40ais.ucla.edu%29 (Enterprise; S5-J4JEBZS9PUJJA; jhuang@ais.ucla.edu)"

164.67.132.219 - - [20/Sep/2007:07:54:55 -0700] "GET /rss/feed.php?
unit=uclastat HTTP/1.0" 200 1739 "-" "gsa-crawler%20%28gsa1%2C
%20contact%3A%20jhuang%40ais.ucla.edu%29 (Enterprise; S5-J4JEBZS9PUJJA;
jhuang@ais.ucla.edu)"

134.226.32.57 - - [20/Sep/2007:07:55:03 -0700] "GET /%7Esczhu/
talks.html HTTP/1.0" 200 9489 "http://www.stat.ucla.edu/~sczhu/"
"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.7) Gecko/
20070914 Firefox/2.0.0.7"
```

The need for a pattern language

While simple splits are effective in isolating fields, when we want to look into the file a little more deeply, we see that we need something more elaborate

For example, suppose we want to identify the kind of file being downloaded based on its suffix; or perhaps we want to determine the amount of traffic generated by Rob Gould (a faculty member in our department)

Looking at the next slide, you see pretty quickly that a more sophisticated approach to dealing with patterns in text is required...

```
/consult/paid/
/robots.txt
/frames/global.php?body=/departments/index_body.php&head=/departments/index_head.php
/departments/
/~rgould/120bs06/podcast/podcast.xml
/~rgould/120aw06/podcast/podcast.xml
/departments/index_head.php
/style_sheets/Global.css
/departments/index_body.php
/graphics/world.gif
/textbook/singles/describe_single/central/mean.html
/style_sheets/Global.css
/graphics/logo1_167x90.gif
/consult/tutor.php
/consult/free.php
/%7Ergould/11f03/skillcheck3.html
/%7Ergould/x401f01/worksheet1.html
/~dinov/courses_students.dir/02/Spr/STAT_XL10.dir/
/~vlew/stat11/WI01/old/labs/?M=D
/~dinov/courses_students.dir/02/Spr/STAT_XL10.dir/HWs.dir/HW1_XL10_sol.pdf
/forums/read.php?f=327&i=3&t=2
/%7Edinov/courses_students.dir/02/Fall/
/~rgould/152c1www/hints.html
/forums/read.php?f=327&i=3&t=2
/history/people/gould.gif.gz
/index.css
/css/uclastat/site.css
/~apresson/photos/other/ASHG06-New%20Orleans/slides/PA110889.html
/favicon.ico
/~sczhu/Lotus/images/research/fire_Ori.gif
/data/andrews-and-herzberg/T66.1
/~fredphoa/Spring2007_Stat13/PS7sol.pdf
/~sczhu/Lotus/images/research/fire_sketch_Ori.gif
/~cocteau/teaching/stat13/labs/lab3.pdf
/textbook/
/style_sheets/Global.css
/~cocteau/stat202a/lectures.2005/lecture19.pdf
```

Tools to manipulate text - Unix

As an aside, in my own class, I start with basic shell tools; the students are more familiar with commands like ***`split`***, ***`sort`***, ***`wc`***, ***`cut`*** and ***`uniq`***

Combinations of these utilities ("piped" together) are a fairly common way to work with large quantities of text data; these tools are also extremely useful for the data preparation and data cleaning phases of an analysis

Rudimentary pattern matching

Because I start with the shell, by the time we get to regular expressions, we have already encountered some basic pattern matching notions

For example, the command ***`wc *.txt`*** will provide a simple accounting of all the files that end with the characters **`.txt`**; in this expression ***`*`*** acts as a *wildcard* and matches 0 or more other characters

Rudimentary pattern matching

In the expression ***.txt** we can name two kinds of characters

The **.txt** is made up of *literal* or normal text characters

The ***** is a *metacharacter*

While specifying groups of files in this way is quite useful, it is not very expressive in terms of the patterns one can specify; the Unix shell does offer us richer notation, but rather than explain it in detail, I move to a more general construction

Regular expressions

Simply put, a regular expression is a sequence of characters that defines a pattern; most characters in the pattern simply “match” themselves in a target string

A few characters are used in patterns as metacharacters; these allow us to indicate **positioning**, **grouping** and **repetition**

To experiment with regular expressions, we’ll first need an engine that, given a (series of) strings and a pattern, can execute the matches; I use the shell as an engine, others use R...

Implementation - The shell

Because I start with the shell before R, regular expressions are introduced using the Unix utility ***`grep`*** (the name comes from an editing operation: g/re/p, see the next slide)

`grep` skims lines from a file (the strings) that have (or don’t have) a particular pattern, that match (or don’t match) a particular regular expression

"Grep was invented for me. I was making a program to read text aloud through a voice synthesizer. As I invented phonetic rules I would check Webster's dictionary for words on which they might fail. For example, how do you cope with the digraph `ui', which is pronounced many different ways: `fruit', `guile', `guilty', `anguish', `intuit', `beguine'? I would break the dictionary up into pieces that fit in ed's limited buffer and use a global command to select a list. I would whittle this list down by repeated scannings with ed to see how each proposed rule worked."

"The process was tedious, and terribly wasteful, since the dictionary had to be split (one couldn't afford to leave a split copy on line). Then ed copied each part into /tmp, scanned it twice to accomplish the g command, and finally threw it away, which takes time too."

"One afternoon I asked Ken Thompson if he could lift the regular expression recognizer out of the editor and make a one-pass program to do it. He said yes. The next morning I found a note in my mail announcing a program named grep. It worked like a charm. When asked what that funny name meant, Ken said it was obvious. It stood for the editor command that it simulated, g/re/p (global regular expression print)."

From an interview with Doug McIlroy

"Progress on my talking program accelerated dramatically. From that special-purpose beginning, grep soon became a household word. (Something I had to stop myself from writing in the first paragraph above shows how firmly naturalized the idea now is: `I used ed to grep out words from the dictionary.') More than any other single program, grep focused the viewpoint that Kernighan and Plauger christened and formalized in `Software Tools': make programs that do one thing and do it well, with as few preconceptions about input syntax as possible."

Implementation - The shell

As with the shell itself, regular expressions have different flavors depending on their implementation

The most direct way to use the tools we will present is with the command ***egrep*** rather than ***grep***; this specifies so-called extended regular expressions

To try out any of the expressions we will describe you can enter the command

```
    egrep ‘pattern’ file
```

Implementation - R

R also has facilities to manipulate text, and it is arguably better in an introductory course to describe regular expressions there

In general, if we have stored our text data (character vector) in ***txt*** and our pattern (character string) in ***pattern***, we can call

```
grep(pattern,txt)           # the indices of matches

grep(pattern,txt,value=TRUE)  # the matches
```

Regular expressions

As I mentioned before, most characters in a pattern “match” themselves; the simplest pattern consists only of these literals

For example, the literal ***Obama*** matches the following chat lines:

Israel can't risk Obama winning if it wants to attack Iran, because Obama would be actively against such an attack, while Bush would merely not support such an attack

I think Obama would support it.

Obama alone with Jesse Jackson and a boxcutter

I support Obama but I also believe Iran's nuclear programme must be stopped before it's too late

Obama would be rolled by Iran like a cigar

```
> chat = scan("chat.txt","",sep="\n")          # reading data
Read 137332 items

> post = chat[100]                             # just one string
> post
[1] "yer i'll stick around for a few more mins then bed"

> substr(post,10,12)                           # ... and subset it
[1] "sti"

> result = grep('Obama',chat,value=T)          # which refer to Obama?
> result[1:5]

[1] " Israel can't risk Obama winning if it wants to attack Iran, because
Obama would be actively against such an attack, while Bush would merely not
support such an attack"

[2] " I think Obama would support it."

[3] " Obama alone with Jesse Jackson and a boxcutter"

[4] " I support Obama but I also believe Iran's nuclear programme must be
stopped before it's too late"

[5] " Obama would be rolled by Iran like a cigar"
```

Regular expressions

It’s useful to see what’s going on when a “match” occurs; suppose we’re given the pattern ***`Obama`***, and the target string ***`I think Obama would support it.`***

The ***`grep`*** engine moves along the target string character-by-character, testing for a match of the first literal in the pattern, the ***`O`***; it fails to match until the 9th position

Once it matches the ***`O`***, it advances to the next literal and sees if the 10th character in the target string is a match, is a ***`b`***; it will continue in this way until it matches the compete pattern (possibly many times) or runs out of characters in the target string

Regular expressions

At a technical level, you can think of a regular expression as (finite) state machine that changes its state as it processes a string character-by-character according to rules specified by the pattern

We’ll have (a little) more to say about this when our patterns get a little more complicated

Regular expressions

To continue (and in the spirit of bipartisanship) the literal ***McCain*** would match to the following chat lines

McCain would actively support such an attack

what is this McCain and 'mental recession' my TV is speaking of

McCain says he knows nothing about the economy, seems his advisor doesn't either

i absolutely am scared to death of McCain - the man looks like Reagan just before they announced he had Alazheimers

What McCain should hammer home is fiscal responsibility.

McCain "Casper Milktoast" couldnt' hammer a thumbtack

now careful sunny, McCain will say you are whining

Regular expressions

Any character except for ***`[\^$.|?*+(){}`*** can be used to specify a literal; they match a single instance of themselves (the rest serve a role as metacharacters that we’ll describe in a second; we’ll also describe what to do if your pattern involves one of these)

Again, the metacharacters allow us to specify much more complicated patterns; for example, what if we only want the word “Obama”? or sentences that end in the word “McCain”, or “Mccain” or “mccain”?

Regular expressions

It’s clear that we need a way to express

white space

word boundaries

sets or classes of literals

the beginning and end of a line

alternatives (”war” or “peace”)

Metacharacters to the rescue!

Regular expressions

We’ll now present some simple metacharacter constructions

^,$,\b to specify positioning

[and] to express character classes (or equivalence classes)

(and), | to define subexpressions and alternatives

*,+,? to indicate multiplicities

Some metacharacters

^ represents the start of a line

```
^i think
```

will match the lines

i think thats 03 or 04

i think micheal jackson was 10 times better then prince

i think it was at 70 when i booted

i think they both kinda suck

i think i need to restart irsii for this to take effect

i think i have it wrong

i think cause i am over tired

Some metacharacters

$ represents the end of a line

`morning$`

will match the lines

He left this morning

He just left this morning

it'll only take a morning

i filled a mates 320gb in a a morning

sale this morning

Escaping metacharacters

`\` before one of the special characters ***`[\^$.|?*+(){}`*** lets us include these in a pattern; in technical terms, we have “escaped” the special meaning of these characters

```
\$1
```

will match the lines

$65 bucks for a 5 yr warrenty, and they called offering 3 more yrs for $125, so I took it

Six years ago, the majority of Americans was concerned that drilling in the ANWR Wildlife Refuge might upset the porcupine caribou. With the price of oil at $140 and rising, suddenly we're wondering; "Maybe it's all right for the porcupine caribou to just hop over the pipelines and enjoy the vast regions where there are none."

$1500 gets you a pimpin setup

$199 iphone launch today eh

Character classes with [ ]

A character class matches a single character out of all the possibilities contained in the brackets; there are certain rules that apply in these classes that we’ll get to in a second...

```
M[cC]cain
```

will match the lines

```
who here blamed Mccain for Grhams comments?

or maybe they aren't falling for Mccain, Goldbrick4, there's a diff

and Mccain disavowed the comments quickly

07/11/2008 Mccain 42 Obama 43

i hope Mccain will be the next
```

Character classes with [ ]

In terms of rules that work within character classes, you can specify a range of letters [a-z] or [a-zA-Z]; notice that the order within the class doesn’t matter

```
^[0-9][a-zA-Z]
```

will match the lines

6am here... and a lil tired

1k for a sli setup with quad core and 8800gt in sli

4gb ram, the works

1st i heard of it

4th time ive asked

Character classes with [ ]

There are also built-in classes that are convenient because you encounter them often and they offer a degree of cross-language portability for your code; ***`[:digit:]`***, ***`[:punct:]`***, ***`[:alpha:]`***, ***`[:lower:]`***, ***`[:upper:]`***, ***`[:space:]`***

```
[[:upper:]][[:punct:]]$
```

will match the lines

```
WHERE IS THE ISTHMUS OF CORINTH?

WHERE IS THE ISTHMUS OF CORINTH?

Nice going Lady L! The answer was GREECE.

WHAT IS THE LONGEST LIVING LAND MAMMAL AFTER MAN?

Well done Lady L! The answer was ELEPHANT.
```

Character classes with [ ]

When used at the beginning of a character class, the “^” is also a metacharacter and indicates matching characters NOT in the indicated class

```
[^?.]$
```

will match the lines

i like basketballs

6 and 9

dont worry... we all die anyway!

Not in Baghdad

helicopter under water? hmmm

Other examples

In a similar way, the symbols \d, \w, and \s can stand in for digits, word characters and spaces; similarly, \D, \W and \S are so-called negated versions, not digits, not words and not spaces

These can be used anywhere in a regular expression, not just within enclosing []'s in a character class

More metacharacters: .

"." is used to refer to any character. So

```
9.11
```

will match the lines

Asia: (SSEA 2996.284 -19.849 -0.66) (HSI 22184.551 +362.77 +1.66) (N225 13039.69 -27.52 -0.21) (STI 2926.84 +25.26 +0.87) (TWII 7244.76 +169.11 +2.39) (KS11 1567.51 +30.08 +1.96) (KLSE 1150.39 +14.90 +1.31) (BSESN 13469.85 -456.391 -3.28)

9/11 was before 9/12

WASHINGTON(AP) Senate Republicans on Wednesday scuttled an attempt by Sen. Hillary Clinton to establish an independent, bipartisan panel patterned after the 9/11 Commission to investigate what went wrong with federal, state and local governments response to Hurricane Katrina.

More metacharacters: |

This does not mean “pipe” in the context of regular expressions; instead it translates to “or”; we can use it to combine two expressions, the subexpressions being called *alternatives*

```
flood|fire
```

will match the lines

i cant count how many times I was fired years ago

by flood

heard that Schwartzenegger has called up The National Guard to help fight the fires in CA

no flood!

firecacker get a life

the world needs hatred, impatience, rage, fury, storms, fire

More metacharacters: |

We can include any number of alternatives...

```
flood|earthquake|hurricane
```

will match the lines

what was the name of the city in new jersey bob dylan sang about in the hurricane?

hurricanes would destroy me!

i will make a flood protection system for it

And yes, MN is not known for their earthquakes

More metacharacters: |

The alternatives can be real expressions and not just literals

```
^[Gg]ood|[Bb]ad
```

will match the lines

good luck with it

or you just have bad oppinions

miss u very badlyyy

it's a bad habit

not cus guns are bad

good music

More metacharacters: (and)

Subexpressions are often contained in parentheses to constrain the alternatives

```
^([Gg]ood|[Bb]ad)
```

will match the lines

bad words

bad bad boys

bad news

good evening/morning Melodyy and scout!

good Debs

good to seem I'm innocent

bad day here Orphic

More metacharacters: ?

The question mark indicates that the indicated expression is optional

```
[Gg]eorge( [Ww]\.)? [Bb]ush
```

will match the lines

Chief Judge Vaughn Walker of the US District Court in California has ruled that President George W. Bush is a felon.

george bush calls it the {"internets"}

george bush is a legend

One thing to note...

In the following

```
[Gg]eorge( [Ww]\.)? [Bb]ush
```

we wanted to match a “.” as a period; to do that, we had to “escape” the metacharacter, preceding it with a backslash

In general, we have to do this for any metacharacter we want to include in our match

More metacharacters: * and +

The * and + signs are metacharacters used to indicate repetition; * means “any number, including none, of the item” and + means “at least one of the item”

```
\(.*\)
```

will match the lines

anyone wanna chat? (24, m, germany)

hello, 20.m here... (east area + drives + webcam)

(he means older men)

()

NOTE: These were run on LAST YEAR’S DATA so you won’t get the same result with chat.txt

More metacharacters: * and +

The * and + signs are metacharacters used to indicate repetition; * means “any number, including none, of the item” and + means “at least one of the item”

```
[0-9]+ (.*)[0-9]+
```

will match the lines

working as MP here 720 MP battallion, 42nd birgade

so say 2 or 3 years at colleage and 4 at uni makes us 23 when and if we finish

fixing to spend over $15,00 to fill two 2.5 gallon gas jugs

it went down on several occasions for like, 3 or 4 *days*

Mmmm its time 4 me 2 go 2 bed

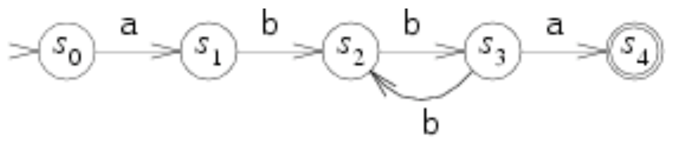
NOTE: These were run on LAST YEAR’S DATA so you won’t get the same result with chat.txt

A machine view again...

Recall that we can think of a regular expression as a kind of machine that moves along a string, examining each character; we can now revisit that

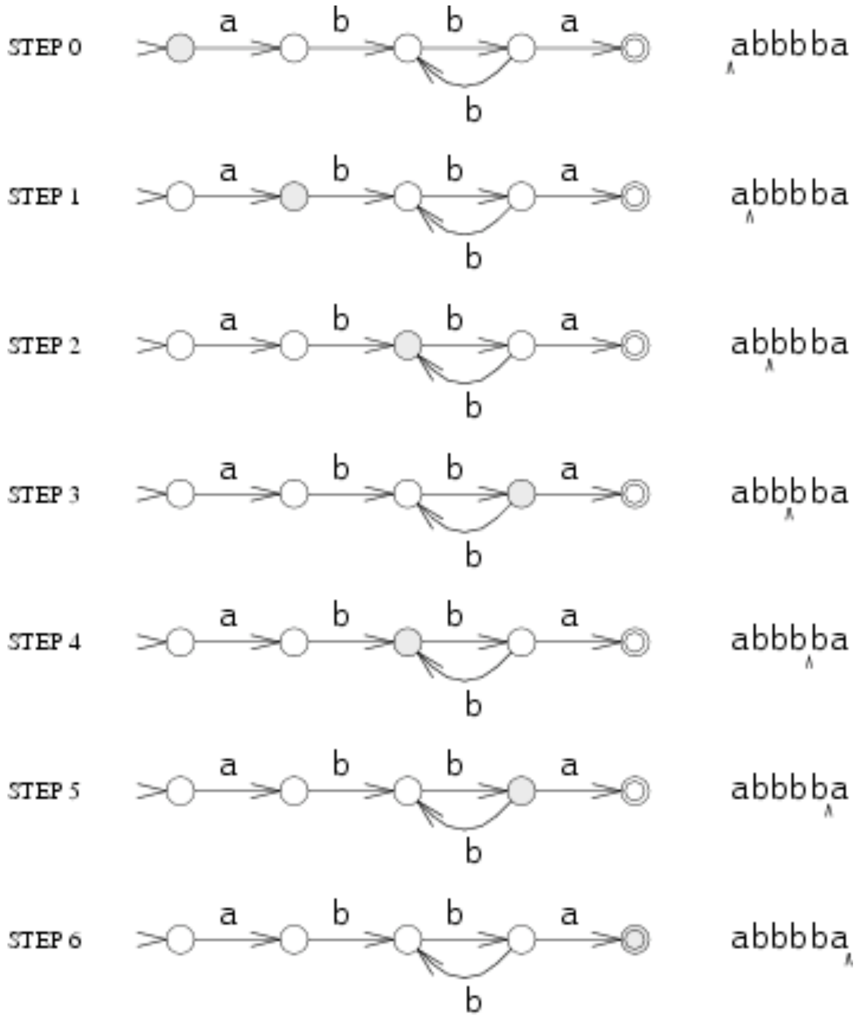
Suppose we have the simple pattern a(bb)+a, which means matching the the character a, followed by some number of double b’s, followed by a final a

The state machine that would do this is given graphically as



A machine view again...

And here is how it would process the string abbbba



More metacharacters: { and }

{ and } are referred to as interval quantifiers; the let us specify the minimum and maximum number of matches of an expression

```
[Bb]ush( +[^ ]+ +){1,5} debate
```

will match the lines

```
Bush has historically won all major debates he's done.

in my view, Bush doesn't need these debates..

bush doesn't need the debates? maybe you are right

That's what Bush supporters are doing about the debate.

Felix, I don't disagree that Bush was poorly prepared for the debate.

indeed, but still, Bush should have taken the debate more seriously.

Keep repeating that Bush smirked and scowled during the debate
```

NOTE: These were run on LAST YEAR'S DATA so you won't get the same result with chat.txt

More metacharacters: { and }

{m,n}	means at least m but not more than n matches
{m}	means exactly m matches
{m,}	means at least m matches

More metacharacters: (and) revisited

In most implementations of regular expressions, the parentheses not only limit the scope of alternatives divided by a “|”, but also can be used to “remember” text matched by the subexpression enclosed

We refer to the matched text with \1, \2, etc.

More metacharacters: (and) revisited

So the expression

```
‘ ([a-zA-Z]+) \1’
```

will match the lines

```
blah blah blah blah
i was standing all all alone against the world outside...
hi anybody anybody at home
what was that movie with with brad pitt & tom cruise
ha ha ha bionicwoman, how observant
```

NOTE: These were run on LAST YEAR’S DATA so you won’t get the same result with chat.txt

Some words of warning

So far, we have focused mainly on the presence of at least one instance of a pattern in a string; as I mentioned at the beginning, we are often looking to extract the matches from the target strings or perhaps replace them

Then we need to be a little more careful about how a match happens...

Greedy * and +

When you specify repetition with * and +, the engine performs a greedy search for your pattern; by that I mean it will continue to match arbitrary characters with . until the regular expression would fail

This is why it finds the last W in our string and not the second; we can flip this behavior with *? and +?