



Motivation and pre-requisites

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

About this course

- This course covers the basic ideas behind getting data ready for analysis
 - Finding and extracting raw data
 - Tidy data principles and how to make data tidy
 - Practical implementation through a range of R packages
- What this course depends on
 - The Data Scientist's Toolbox
 - R Programming
- What would be useful
 - Exploratory analysis
 - Reporting Data and Reproducible Research

What you wish data looked like

The screenshot shows a Microsoft Excel spreadsheet titled "solutions-jun3.csv". The data consists of 42 rows and 10 columns. The columns are labeled A through P. Column A contains row numbers from 1 to 42. Columns B through P contain various numerical and categorical values. The data appears to be a list of items, each with an ID, problem ID, subject ID, start time, stop time, time left, and an answer.

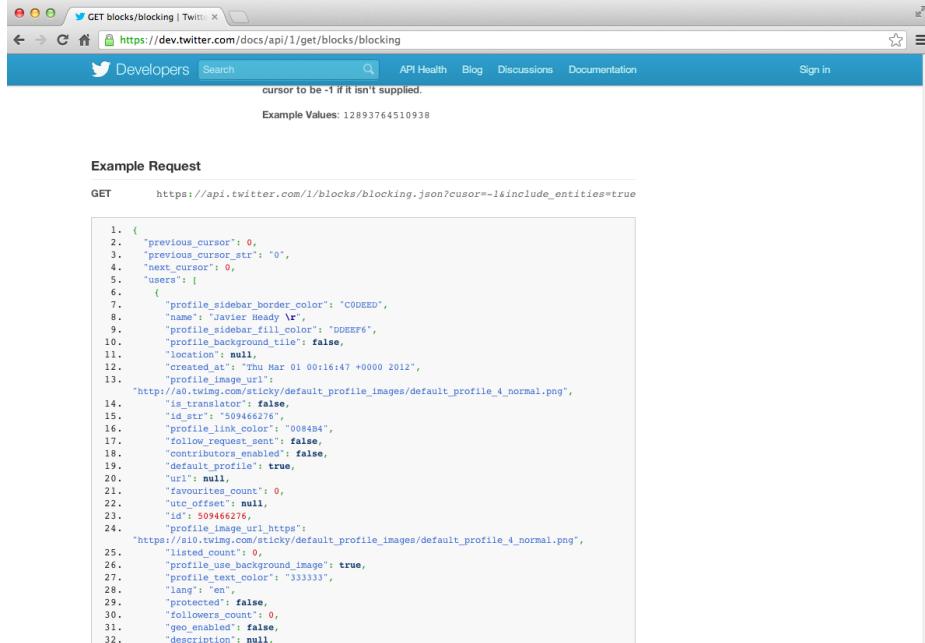
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	id	problem_id	subject_id	start	stop	time_left	answer									
1	1	498	17	1307119989	1307120016	2369	A									
2	2	150	15	1307119991	1307120009	2376	D									
3	3	313	16	1307119994	1307120009	2376	E									
4	4	12	13	1307119995	1307120019	2366	B									
5	5	273	14	1307119996	1307120028	2357	A									
6	6	101	19	1307119996	1307120021	2364	B									
7	7	105	18	1307119997	1307120020	2350	C									
8	8	162	12	1307120004	1307120042	2343	C									
9	9	70	15	1307120011	1307120038	2347	C									
10	10	300	16	1307120012	1307120092	2293	B									
11	11	494	17	1307120017	1307120075	2310	D									
12	12	357	13	1307120021	1307120118	2267	A									
13	13	522	19	1307120025	1307120152	2233	D									
14	14	232	14	1307120030	1307120158	2227	C									
15	15	344	15	1307120041	1307120117	2268	B									
16	16	160	17	1307120079	1307120149	2136	B									
17	17	516	16	1307120084	1307120159	2226	B									
18	18	472	12	1307120119	1307120170	2215	A									
19	19	43	15	1307120122	1307120140	2245	C									
20	20	353	13	1307120144	1307120199	2186	C									
21	21	218	15	1307120152	1307120272	2113	E									
22	22	69	16	1307120163	1307120188	2197	D									
23	23	562	16	1307120190	1307120301	2084	D									
24	24	121	19	1307120253	1307120294	2091	E									
25	25	297	15	1307120277	1307120342	2043	B									
26	26	495	13	1307120303	1307120303	2037	C									
27	27	96	14	1307120398	1307120343	2042	E									
28	28	22	18	1307120310	1307120365	2020	C									
29	29	64	19	1307120310	1307120385	2000	B									
30	30	502	16	1307120323	1307120336	2049	B									
31	31	44	16	1307120339	1307120352	2033	A									
32	32	315	14	1307120348	1307120362	2023	B									
33	33	385	15	1307120352	1307120553	1832	E									
34	34	550	13	1307120356	1307120444	1941	B									
35	35	92	14	1307120361	1307120397	1959	E									
36	36	395	16	1307120377	1307120436	1959	D									
37	37	267	17	1307120382	1307120515	1870	E									
38	38	257	14	1307120401	1307120427	1958	C									
39	39	312	19	1307120407	1307120548	1837	D									
40	40	321	18	1307120431	1307120449	1936	A									
41	41	220	16	1307120437	1307120510	1875	A									

What does data really look like?

```
@HWI-EAS121:4:100:1783:550#0/1
CGTTACGAGATCGGAAGAGCGGGTTCAGCAGGAATGCCGAGACGGATCTGTATGCCGTCTGCTGCGTGACAAGACAGGGG
+HWI-EAS121:4:100:1783:550#0/1
aaaaaa`b_aa`aa`YaX]aZ`aZM^Z]YRa]YSG[[ZREQLHESDHNDHNMEEDDM PENITKFLFEEDDDHEJQMEDDD
@HWI-EAS121:4:100:1783:1611#0/1
GGGTGGGCATTCCACTCGCAGTATGGGTTGCCGACAGCACGGCAGCGGTCAAGCCTGCGCTTGGCCTGGCCTTCGGAA
+HWI-EAS121:4:100:1783:1611#0/1
a``^__`_````a``^a_``_ja_]`\a_____`_``]X_]XTV_\]]NX_XVX]]_TTTG[VTHPN]VFDZ
@HWI-EAS121:4:100:1783:322#0/1
CGTTTATGTTTTGAATATGTCTTATCTAACGGTTATTTAGATGTTGGTCTTATTCTAACGGTCATATATTTCTA
+HWI-EAS121:4:100:1783:322#0/1
abaa``^aaaaabbbaabbbbbbbb`bbbb_bbbbbbbb`bbbaV^_a``^a``]``aT]a__V\]]_]^a`]a_abbaV_
@HWI-EAS121:4:100:1783:1394#0/1
GGGTCTTATTGGTCTGGT GATCCCCCATATTCTCCGGTTGTGGTTAACGATCATCGCGCATTACTCCGGCTGC
+HWI-EAS121:4:100:1783:1394#0/1
````[aa\b``^[]aabbb][`a_abbb`a``bbbbbabaabaaaab_Vza_``bab_X`[a\HV[_]_[^_X\T_VQQ
@HWI-EAS121:4:100:1783:207#0/1
CCCTGGGAGATCGGAAGAGCGGGTTCAGCAGGAATGCCGAGACCGATCTGTATGCCGTCTGCTTGAAAAAAAACA
+HWI-EAS121:4:100:1783:207#0/1
abba`Xa``^\\`aa]ba__bba[a_O_a`aa`aa`a]^V]X_a^YS\R_\H_[\ZTDUZZUSOPX]]POP\GS\WSHHD
@HWI-EAS121:4:100:1783:455#0/1
GGCTAATTCAAGGGACAATCTAATGGCTGCACAAAAAAATACATCTTCATGTTCCATTGCACCATTGACAAATACATATT
+HWI-EAS121:4:100:1783:455#0/1
abb_babbabaabbbbbbbbbbba``b``abbabbbbabbbaabbbbb`bb`ab_O_bab_Q_bbabaa_a
```

[http://brianknaus.com/software/srtoolbox/s\\_4\\_1\\_sequence80.txt](http://brianknaus.com/software/srtoolbox/s_4_1_sequence80.txt)

# What does data really look like?



The screenshot shows a web browser window with the URL [https://dev.twitter.com/docs/api/1/get\(blocks/blocking](https://dev.twitter.com/docs/api/1/get(blocks/blocking)) in the address bar. The page title is "GET blocks/blocking | Twitter". The main content area displays the "Example Request" for the API endpoint. It includes a "GET" method and the URL [https://api.twitter.com/1/blocks/blocking.json?cursor=-1&include\\_entities=true](https://api.twitter.com/1/blocks/blocking.json?cursor=-1&include_entities=true). Below the URL is a large block of JSON code representing the response structure. The JSON object has 32 numbered properties, each containing a field name and its corresponding value type or reference.

```
1. {
2. "previous_cursor": 0,
3. "previous_cursor_str": "0",
4. "next_cursor": 0,
5. "users": [
6. {
7. "profile_sidebar_border_color": "CODEDE",
8. "name": "Tavier Meady",
9. "profile_sidebar_fill_color": "DDDEF6",
10. "profile_background_tile": false,
11. "location": null,
12. "created_at": "Thu Mar 01 00:16:47 +0000 2012",
13. "profile_image_url": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
14. "is_translator": false,
15. "id": 509466276,
16. "profile_image_url_https": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
17. "follow_request_sent": false,
18. "contributors_enabled": false,
19. "default_profile": true,
20. "url": null,
21. "favourites_count": 0,
22. "utc_offset": null,
23. "id": 509466276,
24. "profile_image_url_https": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
25. "listed_count": 0,
26. "profile_use_background_image": true,
27. "profile_text_color": "333333",
28. "lang": "en",
29. "protected": false,
30. "followers_count": 0,
31. "geo_enabled": false,
32. "description": null
33. }
34.]
35. }
```

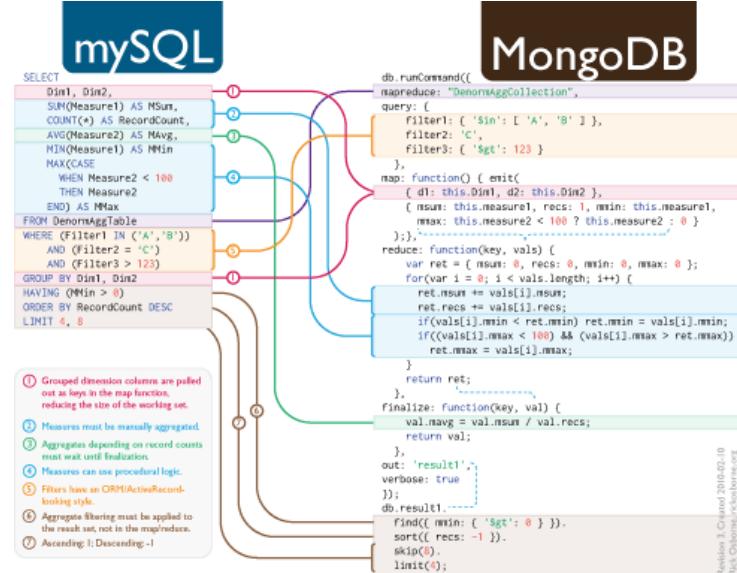
[https://dev.twitter.com/docs/api/1/get\(blocks/blocking](https://dev.twitter.com/docs/api/1/get(blocks/blocking)

# What does data really look like?

ALLERGIES		MEDICATION HISTORY
Last Updated: 01 Dec 2011 @ 0851		Last Updated: 11 Apr 2011 @ 1737
Allergy Name:	TRIMETHOPRIM	Medication: AMLODIPIINE BESYLATE 10MG TAB
Location:	DAYT29	Instructions: TAKE ONE TABLET BY MOUTH TAKE ONE-HALF TABLET FOR :
Date Entered:	09 Mar 2011	GRAPEFRUIT JUICE--
Action:		Status: Active
Allergy Type:	DRUG	Refills Remaining: 3
A Drug Class:	ANTI-INFECTIVES, OTHER	Last Filled On: 28 Aug 2010
Observed/Historical:	HISTORICAL	Initially Ordered On: 13 Aug 2010
Comments:	The reaction to this allergy was MILD (NO SQUELAE)	Quantity: 45
Allergy Name:	TRAMADOL	Days Supply: 90
Location:	DAYT29	Pharmacy: DAYTON
Date Entered:	09 Mar 2011	Prescription Number: 2718953
Action:	URINARY RETENTION	Medication: IBUPROFEN 600MG TAB
Allergy Type:	DRUG	Instructions: TAKE ONE TABLET BY MOUTH FOUR TIMES A DAY WITH FOOD
A Drug Class:	NON-OPIOID ANALGESICS	Status: Active
Observed/Historical:	HISTORICAL	Refills Remaining: 3
Comments:	gradually worsening difficulty emptying bladder	Last Filled On: 28 Aug 2010
		Initially Ordered On: 01 Jul 2010

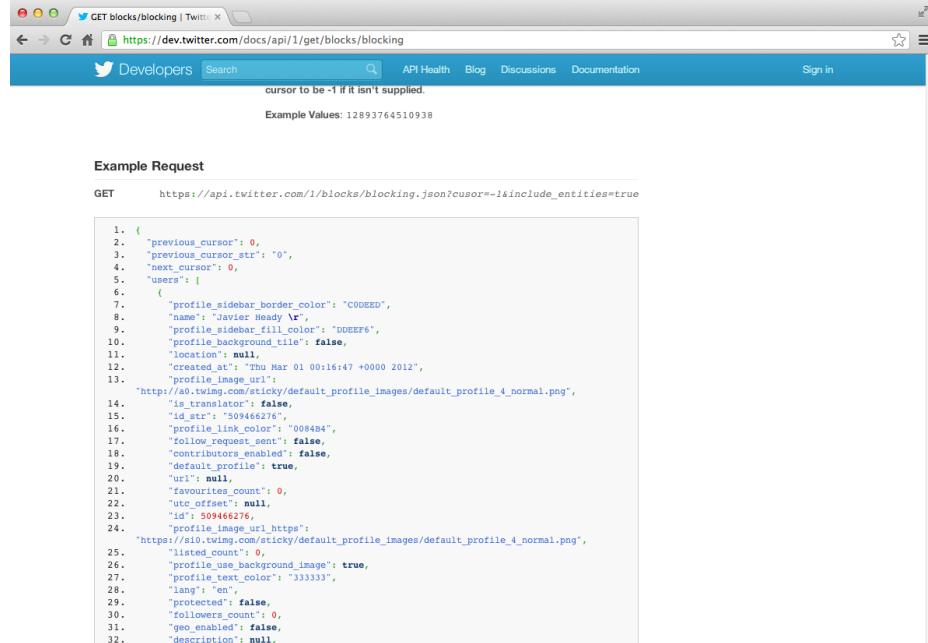
<http://blue-button.github.com/challenge/>

# Where is data?



<http://rickosborne.org/blog/2010/02/infographic-migrating-from-sql-to-mapreduce-with-mongodb/>

# Where is data?



The screenshot shows a web browser window with the URL [https://dev.twitter.com/docs/api/1/get\(blocks/blocking](https://dev.twitter.com/docs/api/1/get(blocks/blocking)) in the address bar. The page title is "GET blocks/blocking | Twitter". The main content area displays the "Example Request" for the API endpoint. It includes a "GET" method and a sample URL: `https://api.twitter.com/1/blocks/blocking.json?cursor=-1&include_entities=true`. Below the URL is a large block of JSON code representing the response from the API. The code is multi-line and contains numerous fields such as "previous\_cursor", "name", "profile\_sidebar\_border\_color", "profile\_sidebar\_fill\_color", "profile\_background\_tile", "created\_at", "location", "url", "profile\_image\_url", "is\_translator", "id\_str", "profile\_image\_url\_https", and "lang". The entire JSON object spans approximately 32 lines.

```
1. {
2. "previous_cursor": 0,
3. "previous_cursor_str": "0",
4. "next_cursor": 0,
5. "users": [
6. {
7. "profile_sidebar_border_color": "CODEED",
8. "name": "Tavian Ready",
9. "profile_sidebar_fill_color": "DDEEF6",
10. "profile_background_tile": false,
11. "created_at": "Thu Mar 01 00:16:47 +0000 2012",
12. "location": null,
13. "url": null,
14. "profile_image_url": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
15. "is_translator": false,
16. "id_str": "509466276",
17. "profile_image_url_https": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
18. "followers_request_sent": false,
19. "contributors_enabled": false,
20. "default_profile": true,
21. "utc_offset": null,
22. "favourites_count": 0,
23. "id": 509466276,
24. "profile_image_url_https": "http://a0.twimg.com/sticky/default_profile_images/default_profile_4_normal.png",
25. "listed_count": 0,
26. "profile_use_background_image": true,
27. "profile_text_color": "333333",
28. "lang": "en",
29. "protected": false,
30. "followers_count": 0,
31. "geo_enabled": false,
32. "description": null,
```

[https://dev.twitter.com/docs/api/1/get\(blocks/blocking](https://dev.twitter.com/docs/api/1/get(blocks/blocking)

# Where is data?

The screenshot shows the homepage of the Open Baltimore beta website. The header features the "OPEN BALTIMORE" logo with "beta" text above it. Below the logo is a navigation menu with links: Home, Residents, Business, Visitors, Government, Office of the Mayor, and Help. A "Sign Up" and "Sign In" button are located at the bottom left of the header. The main background image is a photograph of a computer monitor displaying a 3D model of a city building, overlaid with binary code (01010101...) and the City of Baltimore seal. A large black bar runs horizontally across the middle of the page, partially obscuring the navigation menu. At the bottom of the page, there are three small tables showing data samples.

**We Want Your Feedback!**

Have suggestions for a dataset? Please take a moment and visit the suggestion page at the bottom of this page. Or you can click the feedback tab to the left and join the discussion over at the forums. See you there!

Brought to you by

CITY OF BALTIMORE  
STEPHANIE RAWLINGS-BLAKE  
MAYOR

id	preposition	verb	object
1	in	drive	the car
2	on	drive	the road
3	at	drive	the intersection
4	over	drive	the bridge
5	under	drive	the overpass
6	past	drive	the gas station
7	through	drive	the tunnel
8	into	drive	the parking lot
9	out	drive	the parking lot
10	off	drive	the parking lot
11	onto	drive	the highway
12	over	drive	the bridge
13	under	drive	the overpass
14	past	drive	the gas station
15	through	drive	the tunnel
16	into	drive	the parking lot
17	out	drive	the parking lot
18	off	drive	the parking lot
19	onto	drive	the highway
20	over	drive	the bridge
21	under	drive	the overpass
22	past	drive	the gas station
23	through	drive	the tunnel
24	into	drive	the parking lot
25	out	drive	the parking lot
26	off	drive	the parking lot
27	onto	drive	the highway
28	over	drive	the bridge
29	under	drive	the overpass
30	past	drive	the gas station
31	through	drive	the tunnel
32	into	drive	the parking lot
33	out	drive	the parking lot
34	off	drive	the parking lot
35	onto	drive	the highway
36	over	drive	the bridge
37	under	drive	the overpass
38	past	drive	the gas station
39	through	drive	the tunnel
40	into	drive	the parking lot
41	out	drive	the parking lot
42	off	drive	the parking lot
43	onto	drive	the highway
44	over	drive	the bridge
45	under	drive	the overpass
46	past	drive	the gas station
47	through	drive	the tunnel
48	into	drive	the parking lot
49	out	drive	the parking lot
50	off	drive	the parking lot
51	onto	drive	the highway
52	over	drive	the bridge
53	under	drive	the overpass
54	past	drive	the gas station
55	through	drive	the tunnel
56	into	drive	the parking lot
57	out	drive	the parking lot
58	off	drive	the parking lot
59	onto	drive	the highway
60	over	drive	the bridge
61	under	drive	the overpass
62	past	drive	the gas station
63	through	drive	the tunnel
64	into	drive	the parking lot
65	out	drive	the parking lot
66	off	drive	the parking lot
67	onto	drive	the highway
68	over	drive	the bridge
69	under	drive	the overpass
70	past	drive	the gas station
71	through	drive	the tunnel
72	into	drive	the parking lot
73	out	drive	the parking lot
74	off	drive	the parking lot
75	onto	drive	the highway
76	over	drive	the bridge
77	under	drive	the overpass
78	past	drive	the gas station
79	through	drive	the tunnel
80	into	drive	the parking lot
81	out	drive	the parking lot
82	off	drive	the parking lot
83	onto	drive	the highway
84	over	drive	the bridge
85	under	drive	the overpass
86	past	drive	the gas station
87	through	drive	the tunnel
88	into	drive	the parking lot
89	out	drive	the parking lot
90	off	drive	the parking lot
91	onto	drive	the highway
92	over	drive	the bridge
93	under	drive	the overpass
94	past	drive	the gas station
95	through	drive	the tunnel
96	into	drive	the parking lot
97	out	drive	the parking lot
98	off	drive	the parking lot
99	onto	drive	the highway
100	over	drive	the bridge
101	under	drive	the overpass
102	past	drive	the gas station
103	through	drive	the tunnel
104	into	drive	the parking lot
105	out	drive	the parking lot
106	off	drive	the parking lot
107	onto	drive	the highway
108	over	drive	the bridge
109	under	drive	the overpass
110	past	drive	the gas station
111	through	drive	the tunnel
112	into	drive	the parking lot
113	out	drive	the parking lot
114	off	drive	the parking lot
115	onto	drive	the highway
116	over	drive	the bridge
117	under	drive	the overpass
118	past	drive	the gas station
119	through	drive	the tunnel
120	into	drive	the parking lot
121	out	drive	the parking lot
122	off	drive	the parking lot
123	onto	drive	the highway
124	over	drive	the bridge
125	under	drive	the overpass
126	past	drive	the gas station
127	through	drive	the tunnel
128	into	drive	the parking lot
129	out	drive	the parking lot
130	off	drive	the parking lot
131	onto	drive	the highway
132	over	drive	the bridge
133	under	drive	the overpass
134	past	drive	the gas station
135	through	drive	the tunnel
136	into	drive	the parking lot
137	out	drive	the parking lot
138	off	drive	the parking lot
139	onto	drive	the highway
140	over	drive	the bridge
141	under	drive	the overpass
142	past	drive	the gas station
143	through	drive	the tunnel
144	into	drive	the parking lot
145	out	drive	the parking lot
146	off	drive	the parking lot
147	onto	drive	the highway
148	over	drive	the bridge
149	under	drive	the overpass
150	past	drive	the gas station
151	through	drive	the tunnel
152	into	drive	the parking lot
153	out	drive	the parking lot
154	off	drive	the parking lot
155	onto	drive	the highway
156	over	drive	the bridge
157	under	drive	the overpass
158	past	drive	the gas station
159	through	drive	the tunnel
160	into	drive	the parking lot
161	out	drive	the parking lot
162	off	drive	the parking lot
163	onto	drive	the highway
164	over	drive	the bridge
165	under	drive	the overpass
166	past	drive	the gas station
167	through	drive	the tunnel
168	into	drive	the parking lot
169	out	drive	the parking lot
170	off	drive	the parking lot
171	onto	drive	the highway
172	over	drive	the bridge
173	under	drive	the overpass
174	past	drive	the gas station
175	through	drive	the tunnel
176	into	drive	the parking lot
177	out	drive	the parking lot
178	off	drive	the parking lot
179	onto	drive	the highway
180	over	drive	the bridge
181	under	drive	the overpass
182	past	drive	the gas station
183	through	drive	the tunnel
184	into	drive	the parking lot
185	out	drive	the parking lot
186	off	drive	the parking lot
187	onto	drive	the highway
188	over	drive	the bridge
189	under	drive	the overpass
190	past	drive	the gas station
191	through	drive	the tunnel
192	into	drive	the parking lot
193	out	drive	the parking lot
194	off	drive	the parking lot
195	onto	drive	the highway
196	over	drive	the bridge
197	under	drive	the overpass
198	past	drive	the gas station
199	through	drive	the tunnel
200	into	drive	the parking lot
201	out	drive	the parking lot
202	off	drive	the parking lot
203	onto	drive	the highway
204	over	drive	the bridge
205	under	drive	the overpass
206	past	drive	the gas station
207	through	drive	the tunnel
208	into	drive	the parking lot
209	out	drive	the parking lot
210	off	drive	the parking lot
211	onto	drive	the highway
212	over	drive	the bridge
213	under	drive	the overpass
214	past	drive	the gas station
215	through	drive	the tunnel
216	into	drive	the parking lot
217	out	drive	the parking lot
218	off	drive	the parking lot
219	onto	drive	the highway
220	over	drive	the bridge
221	under	drive	the overpass
222	past	drive	the gas station
223	through	drive	the tunnel
224	into	drive	the parking lot
225	out	drive	the parking lot
226	off	drive	the parking lot
227	onto	drive	the highway
228	over	drive	the bridge
229	under	drive	the overpass
230	past	drive	the gas station
231	through	drive	the tunnel
232	into	drive	the parking lot
233	out	drive	the parking lot
234	off	drive	the parking lot
235	onto	drive	the highway
236	over	drive	the bridge
237	under	drive	the overpass
238	past	drive	the gas station
239	through	drive	the tunnel
240	into	drive	the parking lot
241	out	drive	the parking lot
242	off	drive	the parking lot
243	onto	drive	the highway
244	over	drive	the bridge
245	under	drive	the overpass
246	past	drive	the gas station
247	through	drive	the tunnel
248	into	drive	the parking lot
249	out	drive	the parking lot
250	off	drive	the parking lot
251	onto	drive	the highway
252	over	drive	the bridge
253	under	drive	the overpass
254	past	drive	the gas station
255	through	drive	the tunnel
256	into	drive	the parking lot
257	out	drive	the parking lot
258	off	drive	the parking lot
259	onto	drive	the highway
260	over	drive	the bridge
261	under	drive	the overpass
262	past	drive	the gas station
263	through	drive	the tunnel
264	into	drive	the parking lot
265	out	drive	the parking lot
266	off	drive	the parking lot
267	onto	drive	the highway
268	over	drive	the bridge
269	under	drive	the overpass
270	past	drive	the gas station
271	through	drive	the tunnel
272	into	drive	the parking lot
273	out	drive	the parking lot
274	off	drive	the parking lot
275	onto	drive	the highway
276	over	drive	the bridge
277	under	drive	the overpass
278	past	drive	the gas station
279	through	drive	the tunnel
280	into	drive	the parking lot
281	out	drive	the parking lot
282	off	drive	the parking lot
283	onto	drive	the highway
284	over	drive	the bridge
285	under	drive	the overpass
286	past	drive	the gas station
287	through	drive	the tunnel
288	into	drive	the parking lot
289	out	drive	the parking lot
290	off	drive	the parking lot
291	onto	drive	the highway
292	over	drive	the bridge
293	under	drive	the overpass
294	past	drive	the gas station
295	through	drive	the tunnel
296	into	drive	the parking lot
297	out	drive	the parking lot
298	off	drive	the parking lot
299	onto	drive	the highway
300	over	drive	the bridge
301	under	drive	the overpass
302	past	drive	the gas station
303	through	drive	the tunnel
304	into	drive	the parking lot
305	out	drive	the parking lot
306	off	drive	the parking lot
307	onto	drive	the highway
308	over	drive	the bridge
309	under	drive	the overpass
310	past	drive	the gas station
311	through	drive	the tunnel
312	into	drive	the parking lot
313	out	drive	the parking lot
314	off	drive	the parking lot
315	onto	drive	the highway
316	over	drive	the bridge
317	under	drive	the overpass
318	past	drive	the gas station
319	through	drive	the tunnel
320	into	drive	the parking lot
321	out	drive	the parking lot
322	off	drive	the parking lot
323	onto	drive	the highway
324	over	drive	the bridge
325	under	drive	the overpass
326	past	drive	the gas station
327	through	drive	the tunnel
328	into	drive	the parking lot
329	out	drive	the parking lot
330	off	drive	the parking lot
331	onto	drive	the highway
332	over	drive	the bridge
333	under	drive	the overpass
334	past	drive	the gas station
335	through	drive	the tunnel
336	into	drive	the parking lot
337	out	drive	the parking lot
338	off	drive	the parking lot
339	onto	drive	the highway
340	over	drive	the bridge
341	under	drive	the overpass
342	past	drive	the gas station
343	through	drive	the tunnel
344	into	drive	the parking lot
345	out	drive	the parking lot
346	off	drive	the parking lot
347	onto	drive	the highway
348	over	drive	the bridge
349	under	drive	the overpass
350	past	drive	the gas station
351	through	drive	the tunnel
352	into	drive	the parking lot
353	out	drive	the parking lot
354	off	drive	the parking lot
355	onto	drive	the highway
356	over	drive	the bridge
357	under	drive	the overpass
358	past	drive	the gas station
359	through	drive	the tunnel
360	into	drive	the parking lot
361	out	drive	the parking lot
362	off	drive	the parking lot
363	onto	drive	the highway
364	over	drive	the bridge
365	under	drive	the overpass
366	past	drive	the gas station
367	through	drive	the tunnel
368	into	drive	the parking lot
369	out	drive	the parking lot
370	off	drive	the parking lot
371	onto	drive	the highway
372	over	drive	the bridge
373	under	drive	the overpass
374	past	drive	the gas station
375	through	drive	the tunnel
376	into	drive	the parking lot
377	out	drive	the parking lot
378	off	drive	the parking lot
379	onto	drive	the highway
380	over	drive	the bridge
381	under	drive	the overpass
382	past	drive	the gas station
383	through	drive	the tunnel

# The goal of this course

Raw data -> Processing script -> tidy data -> data analysis -> data communication



# Raw and processed data

Jeffrey Leek, Assistant Professor of Biostatistics  
Johns Hopkins Bloomberg School of Public Health

# Definition of data

“ Data are values of qualitative or quantitative variables, belonging to a set of items.”

<http://en.wikipedia.org/wiki/Data>

# Definition of data

“ Data are values of qualitative or quantitative variables, belonging to a **set** of items.”

<http://en.wikipedia.org/wiki/Data>

**Set of items:** Sometimes called the population; the set of objects you are interested in

# Definition of data

“ Data are values of qualitative or quantitative **variables**, belonging to a set of items.”

<http://en.wikipedia.org/wiki/Data>

**Variables:** A measurement or characteristic of an item.

# Definition of data

“ Data are values of **qualitative** or **quantitative** variables, belonging to a set of items.”

<http://en.wikipedia.org/wiki/Data>

**Qualitative:** Country of origin, sex, treatment

**Quantitative:** Height, weight, blood pressure

# Raw versus processed data

## Raw data

- The original source of the data
- Often hard to use for data analyses
- Data analysis *includes* processing
- Raw data may only need to be processed once

[http://en.wikipedia.org/wiki/Raw\\_data](http://en.wikipedia.org/wiki/Raw_data)

## Processed data

- Data that is ready for analysis
- Processing can include merging, subsetting, transforming, etc.
- There may be standards for processing
- All steps should be recorded

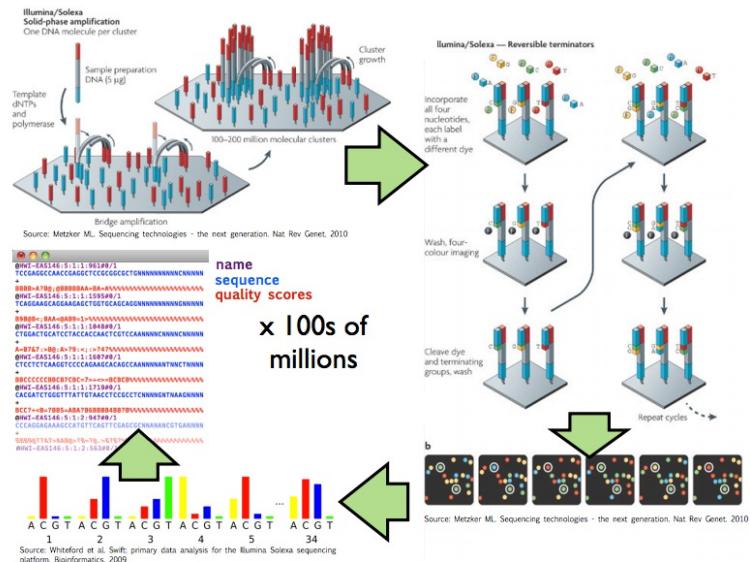
[http://en.wikipedia.org/wiki/Computer\\_data\\_processing](http://en.wikipedia.org/wiki/Computer_data_processing)

# An example of a processing pipeline



[http://www.illumina.com.cn/support/sequencing/sequencing\\_instruments/hiseq\\_1000.asp](http://www.illumina.com.cn/support/sequencing/sequencing_instruments/hiseq_1000.asp)

# An example of a processing pipeline



[http://www.cbcn.umd.edu/~hcorrada/CMSC858B/lectures/lect22\\_seqIntro/seqIntro.pdf](http://www.cbcn.umd.edu/~hcorrada/CMSC858B/lectures/lect22_seqIntro/seqIntro.pdf)



# The components of tidy data

Jeffrey Leek, Assistant Professor of Biostatistics  
Johns Hopkins Bloomberg School of Public Health

# The four things you should have

1. The raw data.
2. A tidy data set
3. A code book describing each variable and its values in the tidy data set.
4. An explicit and exact recipe you used to go from 1 -> 2,3.

# The raw data

- The strange binary file your measurement machine spits out
- The unformatted Excel file with 10 worksheets the company you contracted with sent you
- The complicated JSON data you got from scraping the Twitter API
- The hand-entered numbers you collected looking through a microscope

*You know the raw data is in the right format if you*

1. Ran no software on the data
2. Did not manipulate any of the numbers in the data
3. You did not remove any data from the data set
4. You did not summarize the data in any way

<https://github.com/jtleek/datasharing>

# The tidy data

1. Each variable you measure should be in one column
2. Each different observation of that variable should be in a different row
3. There should be one table for each "kind" of variable
4. If you have multiple tables, they should include a column in the table that allows them to be linked

*Some other important tips*

- Include a row at the top of each file with variable names.
- Make variable names human readable AgeAtDiagnosis instead of AgeDx
- In general data should be saved in one file per table.

<https://github.com/jtleek/datasharing>

# The code book

1. Information about the variables (including units!) in the data set not contained in the tidy data
2. Information about the summary choices you made
3. Information about the experimental study design you used

*Some other important tips*

- A common format for this document is a Word/text file.
- There should be a section called "Study design" that has a thorough description of how you collected the data.
- There must be a section called "Code book" that describes each variable and its units.

<https://github.com/jtleek/datasharing>

# The instruction list

- Ideally a computer script (in R :-), but I suppose Python is ok too...)
- The input for the script is the raw data
- The output is the processed, tidy data
- There are no parameters to the script

In some cases it will not be possible to script every step. In that case you should provide instructions like:

1. Step 1 - take the raw file, run version 3.1.2 of summarize software with parameters a=1, b=2, c=3
2. Step 2 - run the software separately for each sample
3. Step 3 - take column three of outputfile.out for each sample and that is the corresponding row in the output data set

<https://github.com/jtleek/datasharing>

# Why is the instruction list important?

Does High Public Debt Consistently Stifle Economic Growth? A Critique of Reinhart and Rogoff

Thomas Herndon\*

Michael Ash

Robert Pollin

April 15, 2013



<http://www.colbertnation.com/the-colbert-report-videos/425748/april-23-2013/austerity-s-spreadsheet-error>



# Downloading files

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# Get/set your working directory

- A basic component of working with data is knowing your working directory
- The two main commands are `getwd()` and `setwd()`.
- Be aware of relative versus absolute paths
  - **Relative** - `setwd("./data")`, `setwd("../")`
  - **Absolute** - `setwd("/Users/jtleek/data/")`
- Important difference in Windows `setwd("C:\\\\Users\\\\Andrew\\\\Downloads")`

# Checking for and creating directories

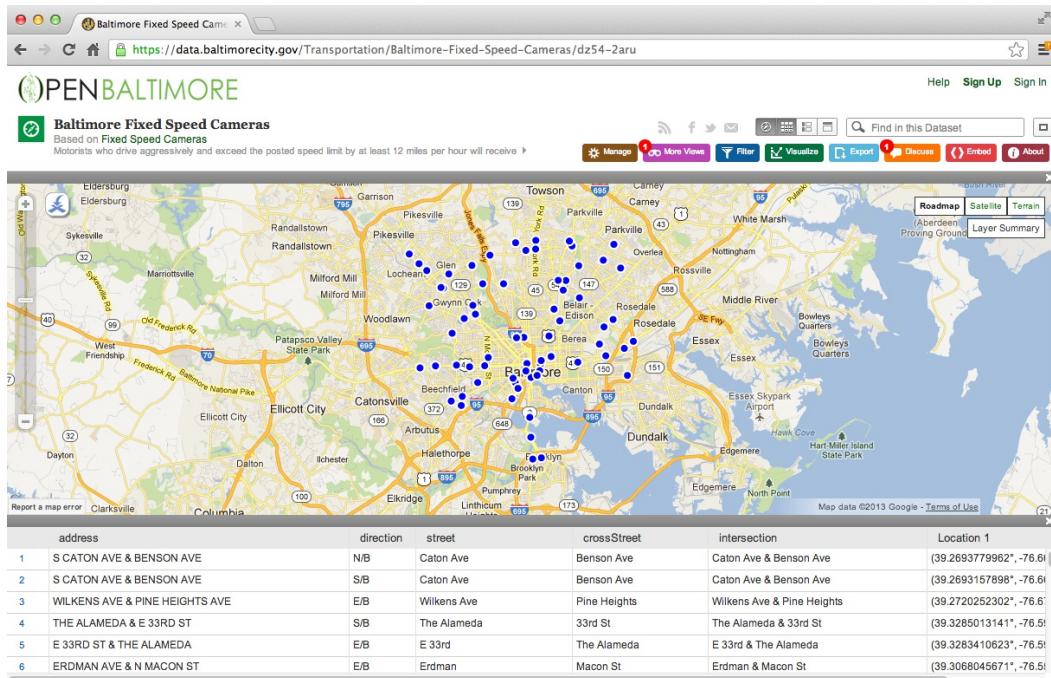
- `file.exists("directoryName")` will check to see if the directory exists
- `dir.create("directoryName")` will create a directory if it doesn't exist
- Here is an example checking for a "data" directory and creating it if it doesn't exist

```
if (!file.exists("data")) {
 dir.create("data")
}
```

# Getting data from the internet - download.file()

- Downloads a file from the internet
- Even if you could do this by hand, helps with reproducibility
- Important parameters are *url*, *destfile*, *method*
- Useful for downloading tab-delimited, csv, and other files

# Example - Baltimore camera data



BALTIMORE CITY + Socrata

Data Catalog Open Data Policy Privacy Policy Terms of Use Developers Help

<https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru>

# Example - Baltimore camera data

The screenshot shows a map of Baltimore and surrounding areas, specifically highlighting fixed speed cameras. The map includes major roads like I-95, I-695, and various state routes (e.g., 378, 100, 35, 147). Numerous blue dots represent the locations of these cameras across the city. Below the map is a table with six rows of address, direction, street, crossStreet, and intersection data.

	address	direction	street	crossStreet	intersection
1	S CATON AVE & BENSON AVE	N/B	Caton Ave	Benson Ave	Caton Ave & Be
2	S CATON AVE & BENSON AVE	S/B	Caton Ave	Benson Ave	Caton Ave & Be
3	WILKENS AVE & PINE HEIGHTS AVE	E/B	Wilkens Ave	Pine Heights	Wilkens Ave & i
4	THE ALAMEDA & E 33RD ST	S/B	The Alameda	33rd St	The Alameda &
5	E 33RD ST & THE ALAMEDA	E/B	E 33rd	The Alameda	E 33rd & The Al
6	ERDMAN AVE & N MACON ST	E/B	Erdman	Macon St	Erdman & Mac

At the bottom right of the map interface, a context menu is open over the 'Copy Link Address' option. The menu also includes 'Copy', 'Search Google for 'CSV'', 'Inspect Element', 'Look Up in Dictionary', 'Speech', 'Search With Google', and 'Add to iTunes as a Spoken Track'.

<https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru>

# Download a file from the web

```
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl, destfile = "./data/cameras.csv", method = "curl")
list.files("./data")
```

```
[1] "cameras.csv"
```

```
dateDownloaded <- date()
dateDownloaded
```

```
[1] "Sun Jan 12 21:37:44 2014"
```

# Some notes about download.file()

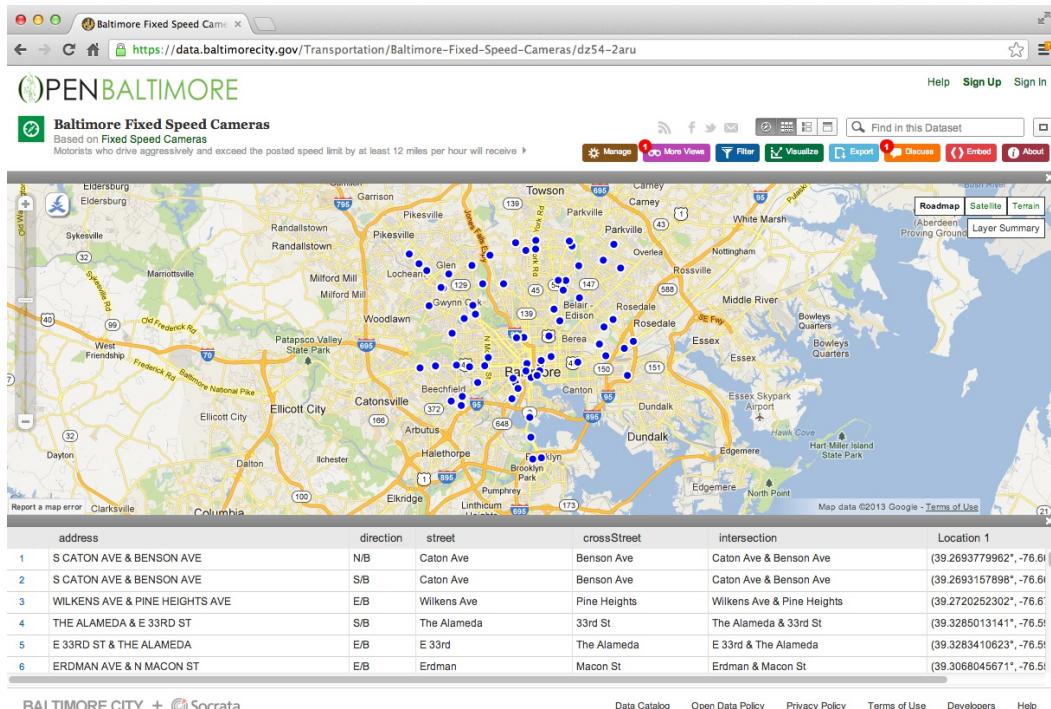
- If the url starts with *http* you can use download.file()
- If the url starts with *https* on Windows you may be ok
- If the url starts with *https* on Mac you may need to set *method="curl"*
- If the file is big, this might take a while
- Be sure to record when you downloaded.



# Reading local flat files

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# Example - Baltimore camera data



BALTIMORE CITY + Socrata

Data Catalog Open Data Policy Privacy Policy Terms of Use Developers Help

<https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru>

# Download the file to load

```
if (!file.exists("data")) {
 dir.create("data")
}
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl, destfile = "cameras.csv", method = "curl")
dateDownloaded <- date()
```

# Loading flat files - `read.table()`

- This is the main function for reading data into R
- Flexible and robust but requires more parameters
- Reads the data into RAM - big data can cause problems
- Important parameters *file*, *header*, *sep*, *row.names*, *nrows*
- Related: *read.csv()*, *read.csv2()*

# Baltimore example

```
cameraData <- read.table("./data/cameras.csv")
```

```
Error: line 1 did not have 13 elements
```

```
head(cameraData)
```

```
Error: object 'cameraData' not found
```

# Example: Baltimore camera data

```
cameraData <- read.table("./data/cameras.csv", sep = ",", header = TRUE)
head(cameraData)
```

```
address direction street crossStreet
1 S CATON AVE & BENSON AVE N/B Caton Ave Benson Ave
2 S CATON AVE & BENSON AVE S/B Caton Ave Benson Ave
3 WILKENS AVE & PINE HEIGHTS AVE E/B Wilkens Ave Pine Heights
4 THE ALAMEDA & E 33RD ST S/B The Alameda 33rd St
5 E 33RD ST & THE ALAMEDA E/B E 33rd The Alameda
6 ERDMAN AVE & N MACON ST E/B Erdman Macon St
intersection Location.1
1 Caton Ave & Benson Ave (39.2693779962, -76.6688185297)
2 Caton Ave & Benson Ave (39.2693157898, -76.6689698176)
3 Wilkens Ave & Pine Heights (39.2720252302, -76.676960806)
4 The Alameda & 33rd St (39.3285013141, -76.5953545714)
5 E 33rd & The Alameda (39.3283410623, -76.5953594625)
6 Erdman & Macon St (39.3068045671, -76.5593167803)
```

# Example: Baltimore camera data

read.csv sets *sep=","* and *header=TRUE*

```
cameraData <- read.csv("./data/cameras.csv")
head(cameraData)
```

```
address direction street crossStreet
1 S CATON AVE & BENSON AVE N/B Caton Ave Benson Ave
2 S CATON AVE & BENSON AVE S/B Caton Ave Benson Ave
3 WILKENS AVE & PINE HEIGHTS AVE E/B Wilkens Ave Pine Heights
4 THE ALAMEDA & E 33RD ST S/B The Alameda 33rd St
5 E 33RD ST & THE ALAMEDA E/B E 33rd The Alameda
6 ERDMAN AVE & N MACON ST E/B Erdman Macon St
intersection Location.1
1 Caton Ave & Benson Ave (39.2693779962, -76.6688185297)
2 Caton Ave & Benson Ave (39.2693157898, -76.6689698176)
3 Wilkens Ave & Pine Heights (39.2720252302, -76.676960806)
4 The Alameda & 33rd St (39.3285013141, -76.5953545714)
5 E 33rd & The Alameda (39.3283410623, -76.5953594625)
6 Erdman & Macon St (39.3068045671, -76.5593167803)
```

# Some more important parameters

- *quote* - you can tell R whether there are any quoted values quote="" means no quotes.
- *na.strings* - set the character that represents a missing value.
- *nrows* - how many rows to read of the file (e.g. nrows=10 reads 10 lines).
- *skip* - number of lines to skip before starting to read

*In my experience, the biggest trouble with reading flat files are quotation marks ` or " placed in data values, setting quote="" often resolves these.*



# Reading Excel files

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# Excel files

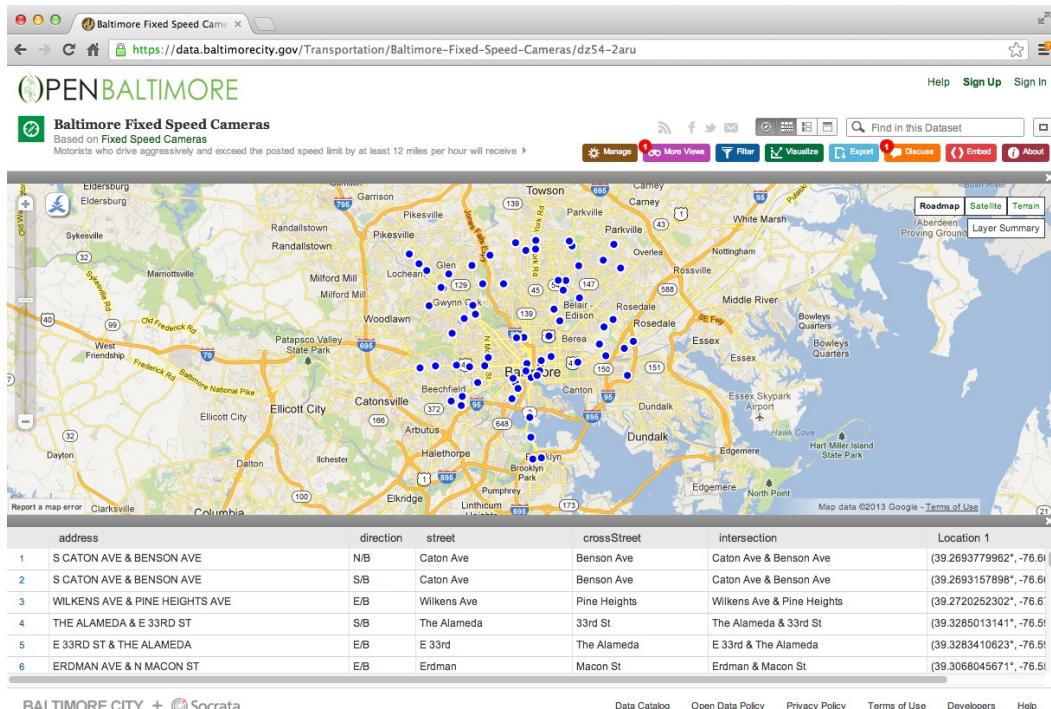
Still probably the most widely used format for sharing data

The screenshot shows the Microsoft Excel application running on a Mac OS X desktop. The window title is "Microsoft Excel - spreadsheets". The ribbon menu is visible at the top. A search bar at the top left says "Search all of Office.com". At the top right, there are buttons for "Buy with Office" and "Try 1 month FREE". The main content area displays a bar chart titled "Employee Travel Expense Trends" with data for months from Jan to Dec. The chart includes a legend for "Expenses", "Convenience fees", "Meals", and "Other". Below the chart is a data table with columns for Month, Type, and Value. At the bottom of the screen, there are three buttons: "Discover" (orange), "Visualize" (white), and "Share" (white).

Discover and reveal the insights hidden in your data

<http://office.microsoft.com/en-us/excel/>

# Example - Baltimore camera data



BALTIMORE CITY + Socrata

Data Catalog Open Data Policy Privacy Policy Terms of Use Developers Help

<https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru>

# Download the file to load

```
if(!file.exists("data")){dir.create("data")}
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.xlsx?accessType=DOWNLOAD"
download.file(fileUrl,destfile="./data/cameras.xlsx",method="curl")
dateDownloaded <- date()
```

# read.xlsx(), read.xlsx2() {xlsx package}

```
library(xlsx)
cameraData <- read.xlsx("./data/cameras.xlsx", sheetIndex=1, header=TRUE)
head(cameraData)
```

	address	direction	street	crossStreet	intersection
1	S CATON AVE & BENSON AVE	N/B	Caton Ave	Benson Ave	Caton Ave & Benson Ave
2	S CATON AVE & BENSON AVE	S/B	Caton Ave	Benson Ave	Caton Ave & Benson Ave
3	WILKENS AVE & PINE HEIGHTS AVE	E/B	Wilkens Ave	Pine Heights	Wilkens Ave & Pine Heights
4	THE ALAMEDA & E 33RD ST	S/B	The Alameda	33rd St	The Alameda & 33rd St
5	E 33RD ST & THE ALAMEDA	E/B	E 33rd	The Alameda	E 33rd & The Alameda
6					
1	(39.2693779962, -76.6688185297)				
2	(39.2693157898, -76.6689698176)				
3	(39.2720252302, -76.676960806)				
4	(39.3285013141, -76.5953545714)				
5	(39.3283410623, -76.5953594625)				
6	(39.3068045671, -76.5593167803)				

# Reading specific rows and columns

```
colIndex <- 2:3
rowIndex <- 1:4
cameraDataSubset <- read.xlsx("./data/cameras.xlsx", sheetIndex=1,
 colIndex=colIndex, rowIndex=rowIndex)
cameraDataSubset
```

	direction	street
1	N/B	Caton Ave
2	S/B	Caton Ave
3	E/B	Wilkens Ave

# Further notes

- The `write.xlsx` function will write out an Excel file with similar arguments.
- `read.xlsx2` is much faster than `read.xlsx` but for reading subsets of rows may be slightly unstable.
- The [XLConnect](#) package has more options for writing and manipulating Excel files
- The [XLConnect vignette](#) is a good place to start for that package
- In general it is advised to store your data in either a database or in comma separated files (.csv) or tab separated files (.tab/.txt) as they are easier to distribute.



# Reading XML

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# XML

- Extensible markup language
- Frequently used to store structured data
- Particularly widely used in internet applications
- Extracting XML is the basis for most web scraping
- Components
  - Markup - labels that give the text structure
  - Content - the actual text of the document

<http://en.wikipedia.org/wiki/XML>

# Tags, elements and attributes

- Tags correspond to general labels
  - Start tags <section>
  - End tags </section>
  - Empty tags <line-break />
- Elements are specific examples of tags
  - <Greeting> Hello, world </Greeting>
- Attributes are components of the label
  - 
  - <step number="3"> Connect A to B. </step>

<http://en.wikipedia.org/wiki/XML>

# Example XML file



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<!-- Edited by XMLSpy® -->
<!DOCTYPE breakfast_menu>
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
 <food>
 <name>Belgian Waffles</name>
 <price>$5.95</price>
 <description>
 Two of our famous Belgian Waffles with plenty of real maple syrup
 </description>
 <calories>650</calories>
 </food>
 <food>
 <name>Strawberry Belgian Waffles</name>
 <price>$7.95</price>
 <description>
 Light Belgian waffles covered with strawberries and whipped cream
 </description>
 <calories>900</calories>
 </food>
 <food>
 <name>Berry-Berry Belgian Waffles</name>
 <price>$8.95</price>
 <description>
 Light Belgian waffles covered with an assortment of fresh berries and whipped cream
 </description>
 <calories>900</calories>
 </food>
 <food>
 <name>French Toast</name>
 <price>$4.50</price>
 <description>
 Thick slices made from our homemade sourdough bread
 </description>
 <calories>600</calories>
 </food>
 <food>
 <name>Homestyle Breakfast</name>
 <price>$6.95</price>
 <description>
 Two eggs, bacon or sausage, toast, and our ever-popular hash browns
 </description>
 <calories>950</calories>
 </food>
</breakfast_menu>
```

<http://www.w3schools.com/xml/simple.xml>

# Read the file into R

```
library(XML)
fileUrl <- "http://www.w3schools.com/xml/simple.xml"
doc <- xmlTreeParse(fileUrl,useInternal=TRUE)
rootNode <- xmlRoot(doc)
xmlName(rootNode)
```

```
[1] "breakfast_menu"
```

```
names(rootNode)
```

```
food food food food food
"food" "food" "food" "food" "food"
```

# Directly access parts of the XML document

```
rootNode[[1]]
```

```
<food>
 <name>Belgian Waffles</name>
 <price>$5.95</price>
 <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
 <calories>650</calories>
</food>
```

```
rootNode[[1]][[1]]
```

```
<name>Belgian Waffles</name>
```

# Programmatically extract parts of the file

```
xmlSApply(rootNode, xmlValue)
```

"Belgian Waffles\$5.95Two of our famous Belgian Waffles with plenty of real

"Strawberry Belgian Waffles\$7.95Light Belgian waffles covered with strawberries and

"Berry-Berry Belgian Waffles\$8.95Light Belgian waffles covered with an assortment of fresh berries and

"French Toast\$4.50Thick slices made from our homemade so

"Homestyle Breakfast\$6.95Two eggs, bacon or sausage, toast, and our ever-popula

# Programmatically extract parts of the file

```
xmlSApply(rootNode, xmlValue)
```

"Belgian Waffles\$5.95Two of our famous Belgian Waffles with plenty of real

"Strawberry Belgian Waffles\$7.95Light Belgian waffles covered with strawberries and

"Berry-Berry Belgian Waffles\$8.95Light Belgian waffles covered with an assortment of fresh berries and

"French Toast\$4.50Thick slices made from our homemade so

"Homestyle Breakfast\$6.95Two eggs, bacon or sausage, toast, and our ever-popula

# XPath

- */node* Top level node
- *//node* Node at any level
- *node[@attr-name]* Node with an attribute name
- *node[@attr-name='bob']* Node with attribute name attr-name='bob'

Information from: <http://www.stat.berkeley.edu/~statcur/Workshop2/Presentations/XML.pdf>

# Get the items on the menu and prices

```
xpathSApply(rootNode, "//name", xmlValue)
```

```
[1] "Belgian Waffles" "Strawberry Belgian Waffles" "Berry-Berry Belgian Waffles"
[4] "French Toast" "Homestyle Breakfast"
```

```
xpathSApply(rootNode, "//price", xmlValue)
```

```
[1] "$5.95" "$7.95" "$8.95" "$4.50" "$6.95"
```

# Another example

Baltimore Ravens Football 

espn.go.com/nfl/team/\_/name/bal/baltimore-ravens

Clubhouse Stats Schedule Roster Splits Depth Chart Transactions Rankings Photos Stadium Blog

Sun Dec 29 Sun Dec 29 2013 Season

Final Paul Brown Stadium Record:

@  L 34-17 Cincinnati Bengals Overall: 8-8  
Pass: Dalton 281 yds vs AFC North: 3-3  
Rush: Green-Ellis 66 yds vs AFC: 6-6  
Rec: Hawkins 74 yds

Baltimore (8-8) @ Cincinnati (11-5)

Team leaders:

Pass: Flacco 3912.0 yds  
Rush: Rice 660.0 yds  
Rec: Smith 1128.0 yds

1 2 3 4 T  
BAL 6 0 11 0 17  
CIN 7 10 0 17 34

Recap » Box Score »

2013 OVERALL NFL RANKINGS

PASSING YDS	RUSHING YDS	OPP PASSING YDS	OPP RUSHING YDS
<b>18th</b> 224.4	<b>30th</b> 83.0	<b>12th</b> 230.1	<b>11th</b> 105.4
Overall	Overall	Overall	Overall

2013 REGULAR SEASON SCHEDULE

WK	DATE	OPPONENT	RESULT/TIME (ET)	RESOURCES
1	Thu, Sep 5	@  Denver	L 49-27	Box Score   Play-By-Play
2	Sun, Sep 15	VS  Cleveland	W 14-6	Box Score   Play-By-Play
3	Sun, Sep 22	VS  Houston	W 30-9	Box Score   Play-By-Play
4	Sun, Sep 29	@  Buffalo	L 23-20	Box Score   Play-By-Play
5	Sun, Oct 6	@  Miami	W 26-23	Box Score   Play-By-Play
6	Sun, Oct 13	VS  Green Bay	L 19-17	Box Score   Play-By-Play

BALTIMORE TEAMS  

 NFL Nation Buzz: Ravens

ESPN.com Ravens reporter Jamison Hensley reflects on an uneven season for the defending champs.

Tags: Joe Flacco, Baltimore Ravens, Jamison Hensley

VIDEO PLAYLIST 

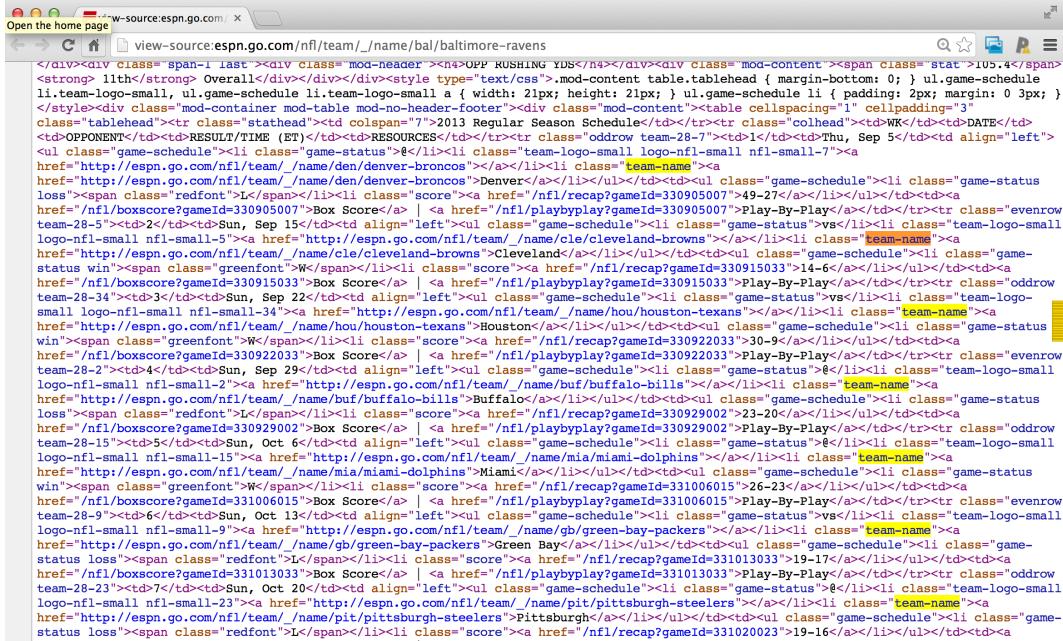
- NFL Nation Buzz: Ravens
- Harbaugh: Rice Will Rebound In '14
- Sunday Blitz: Patriots-Ravens Recap

2013 TEAM LEADERS

PASSING	ATT	COMP	YDS	TD
Joe Flacco	614	362	3912	19

[http://espn.go.com/nfl/team/\\_/name/bal/baltimore-ravens](http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens)

# Viewing the source



[http://espn.go.com/nfl/team/\\_/name/bal/baltimore-ravens](http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens)

# Extract content by attributes

```
fileUrl <- "http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens"
doc <- htmlTreeParse(fileUrl,useInternal=TRUE)
scores <- xpathSApply(doc,"//li[@class='score']",xmlValue)
teams <- xpathSApply(doc,"//li[@class='team-name']",xmlValue)
scores
```

```
[1] "49-27" "14-6" "30-9" "23-20" "26-23" "19-17" "19-16" "24-18"
[9] "20-17 OT" "23-20 OT" "19-3" "22-20" "29-26" "18-16" "41-7" "34-17"
```

```
teams
```

```
[1] "Denver" "Cleveland" "Houston" "Buffalo" "Miami" "Green Bay"
[7] "Pittsburgh" "Cleveland" "Cincinnati" "Chicago" "New York" "Pittsburgh"
[13] "Minnesota" "Detroit" "New England" "Cincinnati"
```

# Notes and further resources

- Official XML tutorials [short](#), [long](#)
- [An outstanding guide to the XML package](#)



# Reading JSON

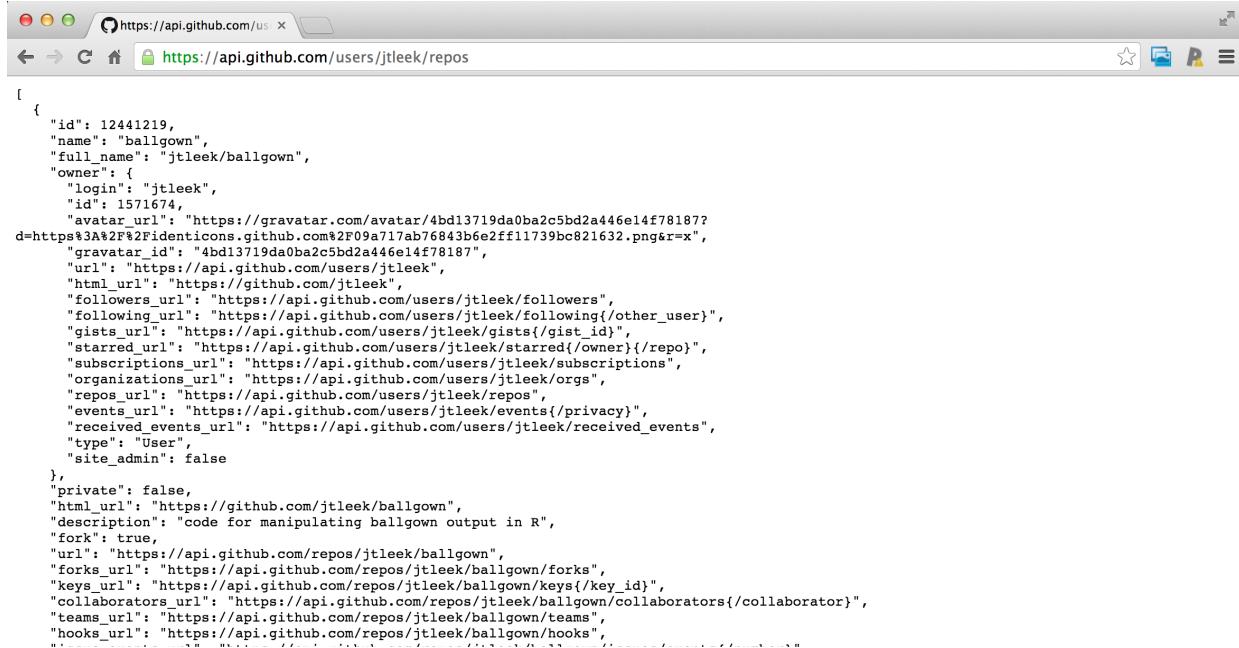
Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# JSON

- Javascript Object Notation
- Lightweight data storage
- Common format for data from application programming interfaces (APIs)
- Similar structure to XML but different syntax/format
- Data stored as
  - Numbers (double)
  - Strings (double quoted)
  - Boolean (*true* or *false*)
  - Array (ordered, comma separated enclosed in square brackets `[]`)
  - Object (unorderd, comma separated collection of key:value pairs in curley brackets `{}`)

<http://en.wikipedia.org/wiki/JSON>

# Example JSON file



A screenshot of a web browser window displaying a JSON response from the GitHub API. The URL in the address bar is <https://api.github.com/users/jtleek/repos>. The JSON data is shown in a monospaced font.

```
[
 {
 "id": 12441219,
 "name": "ballgown",
 "full_name": "jtleek/ballgown",
 "owner": {
 "login": "jtleek",
 "id": 1571674,
 "avatar_url": "https://gravatar.com/avatar/4bd13719da0ba2c5bd2a446e14f78187?
d=https%3A%2F%2Fidenticons.github.com%2F09a717ab76843b6e2ff11739bc821632.png&r=x",
 "gravatar_id": "4bd13719da0ba2c5bd2a446e14f78187",
 "url": "https://api.github.com/users/jtleek",
 "html_url": "https://github.com/jtleek",
 "followers_url": "https://api.github.com/users/jtleek/followers",
 "following_url": "https://api.github.com/users/jtleek/following{/other_user}",
 "gists_url": "https://api.github.com/users/jtleek/gists{/gist_id}",
 "starred_url": "https://api.github.com/users/jtleek/starred{/owner}{/repo}",
 "subscriptions_url": "https://api.github.com/users/jtleek/subscriptions",
 "organizations_url": "https://api.github.com/users/jtleek/orgs",
 "repos_url": "https://api.github.com/users/jtleek/repos",
 "events_url": "https://api.github.com/users/jtleek/events{/privacy}",
 "received_events_url": "https://api.github.com/users/jtleek/received_events",
 "type": "User",
 "site_admin": false
 },
 "private": false,
 "html_url": "https://github.com/jtleek/ballgown",
 "description": "code for manipulating ballgown output in R",
 "fork": true,
 "url": "https://api.github.com/repos/jtleek/ballgown",
 "forks_url": "https://api.github.com/repos/jtleek/ballgown/forks",
 "keys_url": "https://api.github.com/repos/jtleek/ballgown/keys{/key_id}",
 "collaborators_url": "https://api.github.com/repos/jtleek/ballgown/collaborators{/collaborator}",
 "teams_url": "https://api.github.com/repos/jtleek/ballgown/teams",
 "hooks_url": "https://api.github.com/repos/jtleek/ballgown/hooks",
 "...": "..."
 }]
]
```

# Reading data from JSON {jsonlite package}

```
library(jsonlite)
jsonData <- fromJSON("https://api.github.com/users/jtleek/repos")
names(jsonData)
```

```
[1] "id" "name" "full_name" "owner"
[5] "private" "html_url" "description" "fork"
[9] "url" "forks_url" "keys_url" "collaborators_url"
[13] "teams_url" "hooks_url" "issue_events_url" "events_url"
[17] "assignees_url" "branches_url" "tags_url" "blobs_url"
[21] "git_tags_url" "git_refs_url" "trees_url" "statuses_url"
[25] "languages_url" "stargazers_url" "contributors_url" "subscribers_url"
[29] "subscription_url" "commits_url" "git_commits_url" "comments_url"
[33] "issue_comment_url" "contents_url" "compare_url" "merges_url"
[37] "archive_url" "downloads_url" "issues_url" "pulls_url"
[41] "milestones_url" "notifications_url" "labels_url" "releases_url"
[45] "created_at" "updated_at" "pushed_at" "git_url"
[49] "ssh_url" "clone_url" "svn_url" "homepage"
[53] "size" "stargazers_count" "watchers_count" "language"
```

# Nested objects in JSON

```
names(jsonData$owner)
```

```
[1] "login" "id" "avatar_url" "gravatar_id"
[5] "url" "html_url" "followers_url" "following_url"
[9] "gists_url" "starred_url" "subscriptions_url" "organizations_url"
[13] "repos_url" "events_url" "received_events_url" "type"
[17] "site_admin"
```

```
jsonData$owner$login
```

```
[1] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
[11] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
[21] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
```

# Writing data frames to JSON

```
myjson <- toJSON(iris, pretty=TRUE)
cat(myjson)
```

```
[
 {
 "Sepal.Length" : 5.1,
 "Sepal.Width" : 3.5,
 "Petal.Length" : 1.4,
 "Petal.Width" : 0.2,
 "Species" : "setosa"
 },
 {
 "Sepal.Length" : 4.9,
 "Sepal.Width" : 3,
 "Petal.Length" : 1.4,
 "Petal.Width" : 0.2,
 "Species" : "setosa"
 },
 {
 "Sepal.Length" : 4.7,
```

# Convert back to JSON

```
iris2 <- fromJSON(myjson)
head(iris2)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

<http://www.r-bloggers.com/new-package-jsonlite-a-smarter-json-encoderdecoder/>

# Further resources

- <http://www.json.org/>
- A good tutorial on jsonlite - <http://www.r-bloggers.com/new-package-jsonlite-a-smarter-json-encoderdecoder/>
- [jsonlite vignette](#)



# Using data.table

Jeffrey Leek  
Johns Hopkins Bloomberg School of Public Health

# **data.table**

- Inherits from data.frame
  - All functions that accept data.frame work on data.table
- Written in C so it is much faster
- Much, much faster at subsetting, group, and updating

# Create data tables just like data frames

```
library(data.table)
DF = data.frame(x=rnorm(9),y=rep(c("a","b","c"),each=3),z=rnorm(9))
head(DF,3)
```

	x	y	z
1	0.4159	a	-0.05855
2	0.8433	a	0.13732
3	1.0585	a	2.16448

```
DT = data.table(x=rnorm(9),y=rep(c("a","b","c"),each=3),z=rnorm(9))
head(DT,3)
```

	x	y	z
1:	-0.27721	a	0.2530
2:	1.00158	a	1.5093
3:	-0.03382	a	0.4844

# See all the data tables in memory

```
tables()
```

```
NAME NROW MB COLS KEY
[1,] DT 9 1 x,y,z
Total: 1MB
```

# Subsetting rows

```
DT[2,]
```

```
 x y z
1: 1.002 a 1.509
```

```
DT[DT$y=="a",]
```

```
 x y z
1: -0.27721 a 0.2530
2: 1.00158 a 1.5093
3: -0.03382 a 0.4844
```

# Subsetting rows

```
DT[c(2,3)]
```

	x	y	z
1:	1.00158	a	1.5093
2:	-0.03382	a	0.4844

# Subsetting columns!?

```
DT[,c(2,3)]
```

```
[1] 2 3
```

# Column subsetting in data.table

- The subsetting function is modified for data.table
- The argument you pass after the comma is called an "expression"
- In R an expression is a collection of statements enclosed in curly brackets

```
{
 x = 1
 y = 2
}
k = {print(10); 5}
```

```
[1] 10
```

```
print(k)
```

```
[1] 5
```

# Calculating values for variables with expressions

```
DT[,list(mean(x),sum(z))]
```

```
 V1 V2
1: 0.05637 0.5815
```

```
DT[,table(y)]
```

```
Y
a b c
3 3 3
```

# Adding new columns

```
DT[, w:=z^2]
```

	x	y	z	w
1:	-0.27721	a	0.25300	0.064009
2:	1.00158	a	1.50933	2.278091
3:	-0.03382	a	0.48437	0.234619
4:	-0.70493	b	-1.22755	1.506885
5:	-1.36402	b	-0.64624	0.417631
6:	-0.26224	b	-0.51427	0.264475
7:	-0.10929	c	1.21445	1.474901
8:	1.40234	c	0.07493	0.005614
9:	0.85494	c	-0.56652	0.320948

# Adding new columns

```
DT2 <- DT
DT[, y:= 2]
```

	x	y	z	w
1:	-0.27721	2	0.25300	0.064009
2:	1.00158	2	1.50933	2.278091
3:	-0.03382	2	0.48437	0.234619
4:	-0.70493	2	-1.22755	1.506885
5:	-1.36402	2	-0.64624	0.417631
6:	-0.26224	2	-0.51427	0.264475
7:	-0.10929	2	1.21445	1.474901
8:	1.40234	2	0.07493	0.005614
9:	0.85494	2	-0.56652	0.320948

# Careful

```
head(DT,n=3)
```

	x	y	z	w
1:	-0.27721	2	0.2530	0.06401
2:	1.00158	2	1.5093	2.27809
3:	-0.03382	2	0.4844	0.23462

```
head(DT2,n=3)
```

	x	y	z	w
1:	-0.27721	2	0.2530	0.06401
2:	1.00158	2	1.5093	2.27809
3:	-0.03382	2	0.4844	0.23462

# Multiple operations

```
DT[,m:= {tmp <- (x+z); log2(tmp+5)}]
```

	x	y	z	w	m
1:	-0.27721	2	0.25300	0.064009	2.315
2:	1.00158	2	1.50933	2.278091	2.909
3:	-0.03382	2	0.48437	0.234619	2.446
4:	-0.70493	2	-1.22755	1.506885	1.617
5:	-1.36402	2	-0.64624	0.417631	1.580
6:	-0.26224	2	-0.51427	0.264475	2.078
7:	-0.10929	2	1.21445	1.474901	2.610
8:	1.40234	2	0.07493	0.005614	2.695
9:	0.85494	2	-0.56652	0.320948	2.403

# plyr like operations

```
DT[, a:=x>0]
```

	x	y	z	w	m	a
1:	-0.27721	2	0.25300	0.064009	2.315	FALSE
2:	1.00158	2	1.50933	2.278091	2.909	TRUE
3:	-0.03382	2	0.48437	0.234619	2.446	FALSE
4:	-0.70493	2	-1.22755	1.506885	1.617	FALSE
5:	-1.36402	2	-0.64624	0.417631	1.580	FALSE
6:	-0.26224	2	-0.51427	0.264475	2.078	FALSE
7:	-0.10929	2	1.21445	1.474901	2.610	FALSE
8:	1.40234	2	0.07493	0.005614	2.695	TRUE
9:	0.85494	2	-0.56652	0.320948	2.403	TRUE

# plyr like operations

```
DT[, b := mean(x+w), by=a]
```

	x	y	z	w	m	a	b
1:	-0.27721	2	0.25300	0.064009	2.315	FALSE	0.2018
2:	1.00158	2	1.50933	2.278091	2.909	TRUE	1.9545
3:	-0.03382	2	0.48437	0.234619	2.446	FALSE	0.2018
4:	-0.70493	2	-1.22755	1.506885	1.617	FALSE	0.2018
5:	-1.36402	2	-0.64624	0.417631	1.580	FALSE	0.2018
6:	-0.26224	2	-0.51427	0.264475	2.078	FALSE	0.2018
7:	-0.10929	2	1.21445	1.474901	2.610	FALSE	0.2018
8:	1.40234	2	0.07493	0.005614	2.695	TRUE	1.9545
9:	0.85494	2	-0.56652	0.320948	2.403	TRUE	1.9545

# Special variables

- .N An integer, length 1, containing the number r

```
set.seed(123);
DT <- data.table(x=sample(letters[1:3], 1E5, TRUE))
DT[, .N, by=x]
```

```
x N
1: a 33387
2: c 33201
3: b 33412
```

# Keys

```
DT <- data.table(x=rep(c("a", "b", "c"), each=100), y=rnorm(300))
setkey(DT, x)
DT['a']
```

	x	y
1:	a	0.25959
2:	a	0.91751
3:	a	-0.72232
4:	a	-0.80828
5:	a	-0.14135
6:	a	2.25701
7:	a	-2.37955
8:	a	-0.45425
9:	a	-0.06007
10:	a	0.86090
11:	a	-1.78466
12:	a	-0.13074
13:	a	-0.36984
14:	a	-0.18066
15:	a	-1.04973

# Joins

```
DT1 <- data.table(x=c('a', 'a', 'b', 'dt1'), y=1:4)
DT2 <- data.table(x=c('a', 'b', 'dt2'), z=5:7)
setkey(DT1, x); setkey(DT2, x)
merge(DT1, DT2)
```

```
x y z
1: a 1 5
2: a 2 5
3: b 3 6
```

# Fast reading

```
big_df <- data.frame(x=rnorm(1E6), y=rnorm(1E6))
file <- tempfile()
write.table(big_df, file=file, row.names=FALSE, col.names=TRUE, sep="\t", quote=FALSE)
system.time(fread(file))
```

	user	system	elapsed
	0.312	0.015	0.326

```
system.time(read.table(file, header=TRUE, sep="\t"))
```

	user	system	elapsed
	5.702	0.048	5.755

# Summary and further reading

- The latest development version contains new functions like `melt` and `dcast` for data.tables
  - <https://r-forge.r-project.org/scm/viewvc.php/pkg/NEWS?view=markup&root=datatable>
- Here is a list of differences between data.table and data.frame
  - <http://stackoverflow.com/questions/13618488/what-you-can-do-with-data-frame-that-you-can-t-in-data-table>
- Notes based on Raphael Gottardo's notes [https://github.com/raphg/Biostat-578/blob/master/Advanced\\_data\\_manipulation.Rpres](https://github.com/raphg/Biostat-578/blob/master/Advanced_data_manipulation.Rpres), who got them from Kevin Ushey.