Lecture 6: Forward!

Last time

Well, there seemed to be some confusion about the use of one- versus two-tailed tests -- We will review that briefly today

We also presented the differences between Fisher and Neyman-Pearson when it came to inference -- The differences are interesting because some of the best disagreements in the field come at the junction between a mathematical framework and “the real world”

Hypothesis testing

Last time we considered a study by Cannon in which Vioxx and diclofenac were compared -- Before taking any data, the experimenters had to settle on their null and alternative hypotheses

Their null hypothesis was that both drugs carry the same risk of heart attack and their alternative was that the risks are not the same -- Before conducting the experiment, they had no reason to believe one drug would be more or less harmful than the other, and so **they were willing to consider a disproportionate number of heart attacks in either category** as evidence that the drugs are different

Hypothesis testing

The test statistic was taken to be the number of heart attacks in the Vioxx group -- Again, if we see a very large value here (large relative to the total number of heart attacks that might happen) or a very small number (which would mean diclofenac is more harmful), we would be convinced the drugs are different

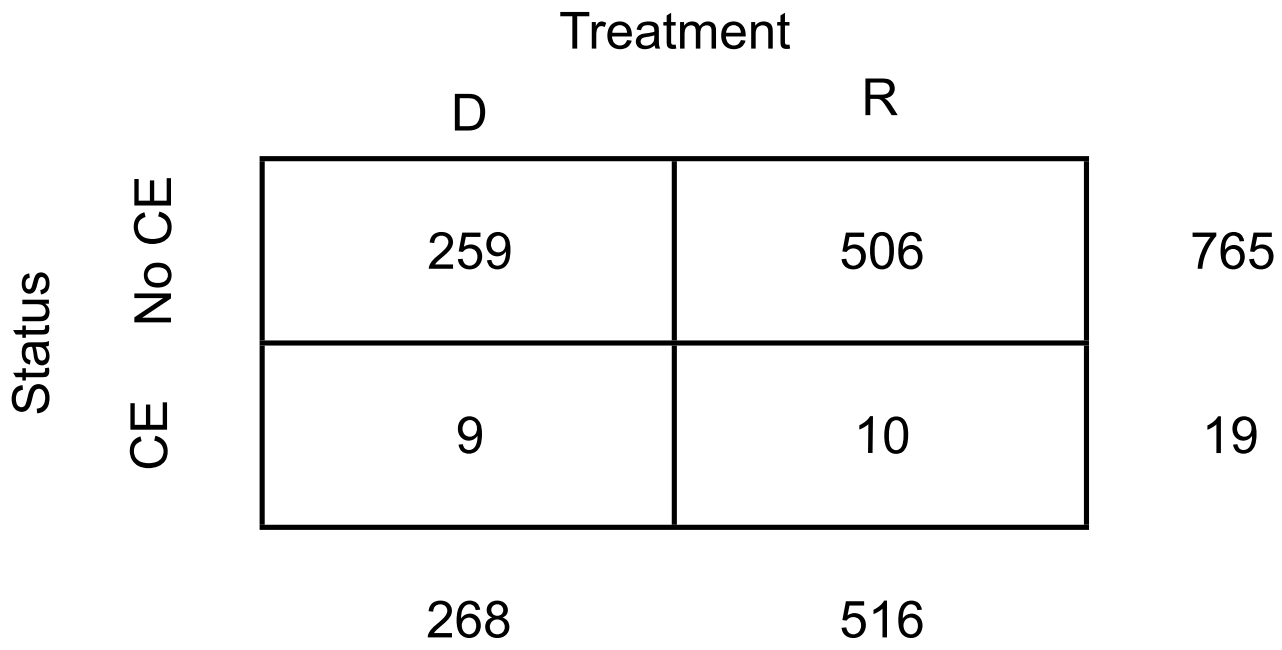
With that in mind, we will **define as "extreme counts" those that have occur under the null less frequently than the value of the test statistic we will observe from our experiment** -- This notion of extreme will help us judge significance if either Vioxx or diclofenac carry a larger risk of heart attacks

Keep in mind that **all of this has to be spelled out before you conduct your experiment** -- In this case we just need a protocol that will capture the notion of "extreme" no matter which drug might turn out to be more harmful

OK, with that protocol defined, they conducted the experiment...

Hypothesis testing

In this study, 9 out of 268 (3.4%) of the patients in the control group experienced CE, while 10/516 (1.9%) of patients in the Vioxx group experienced CE

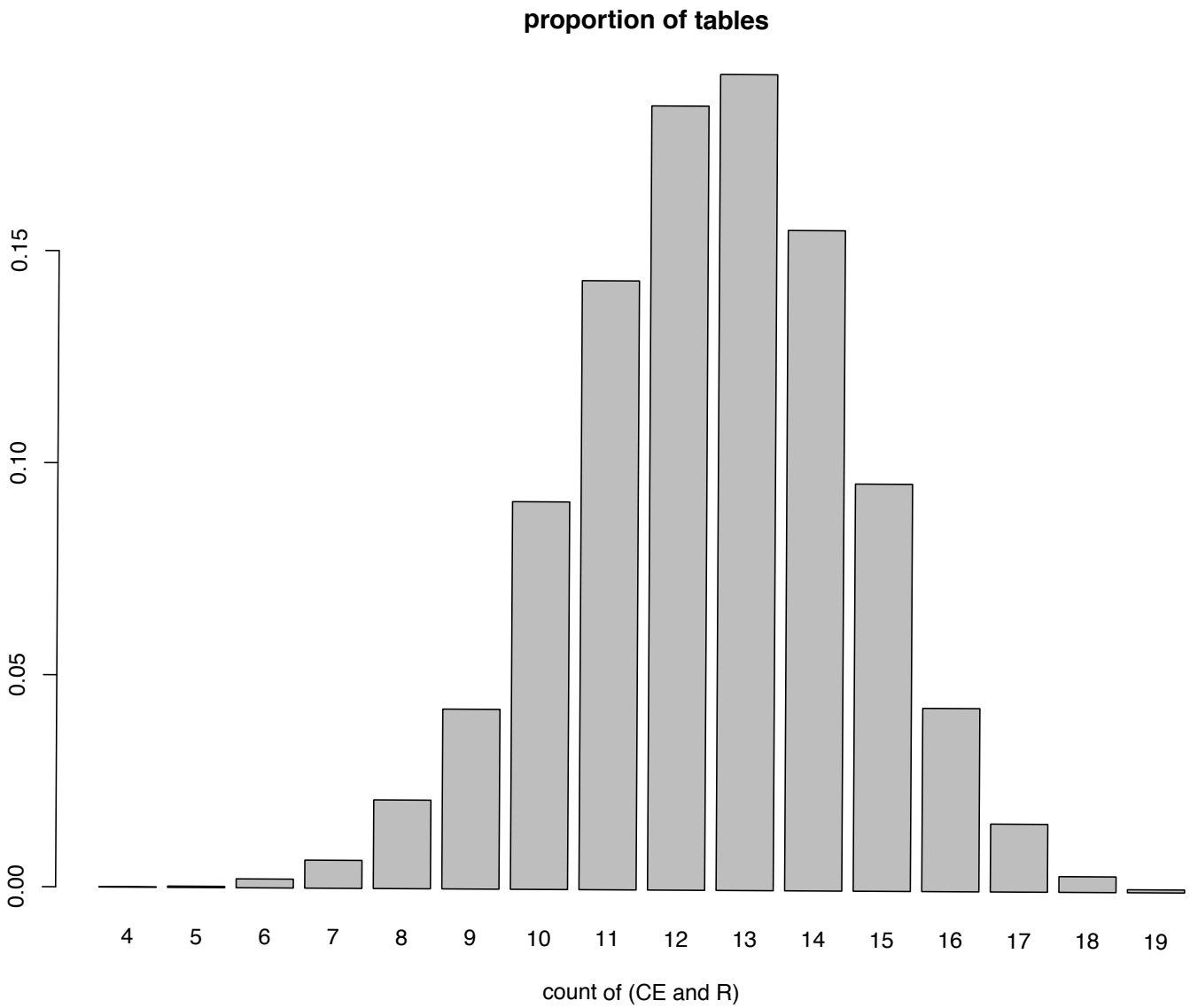


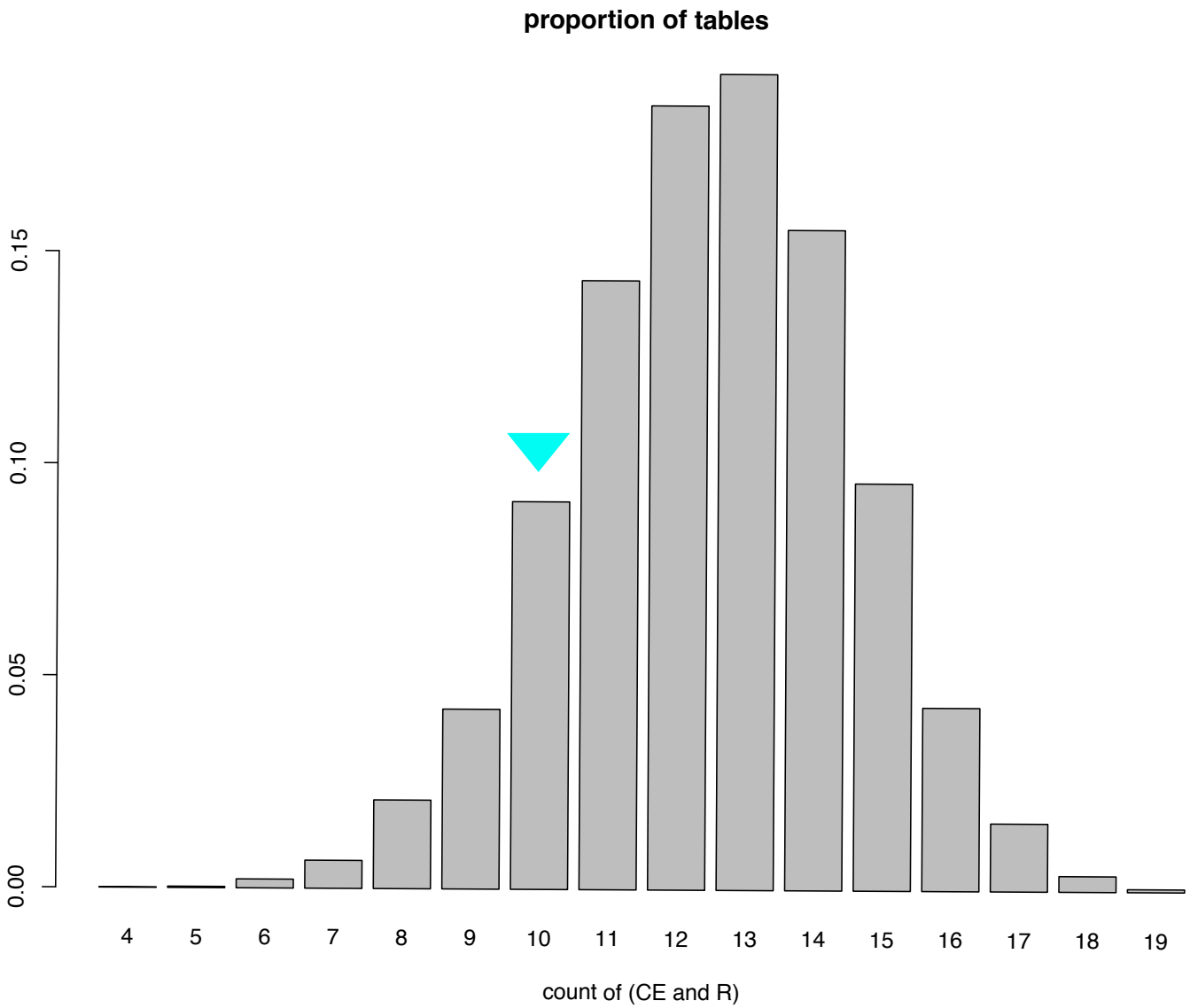
Hypothesis testing

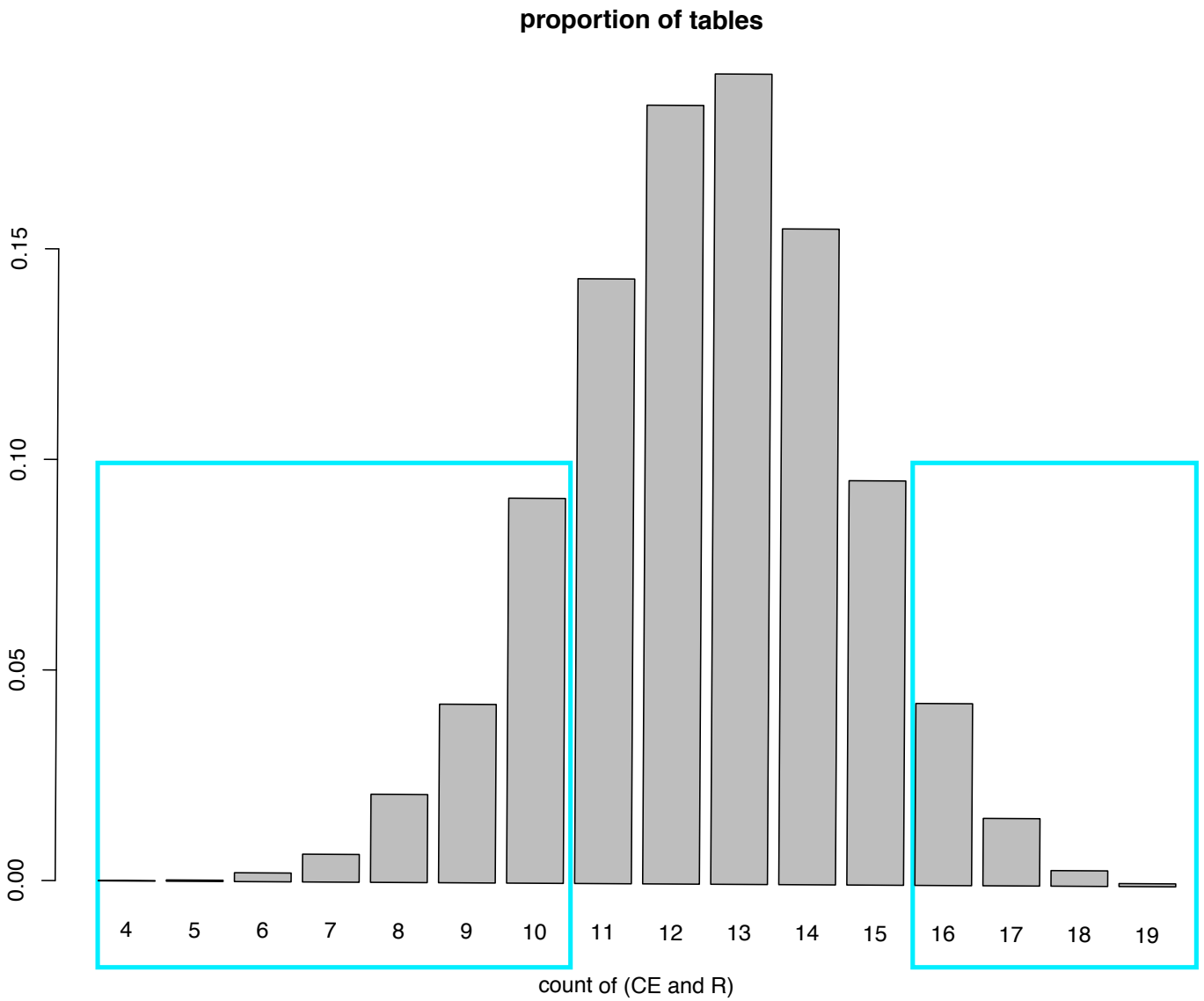
Now, based on the null hypothesis, the two drugs carry the same risk of causing a heart attack and so **the 19 heart attacks would have happened no matter how patients were randomized in the study**

We then generate a number of **re-randomizations with the goal of seeing a typical range of values for our test statistic when the null hypothesis is true** -- In short, we are having a look at the uncertainty coming from the randomization

On the next page we present the result of a few thousand re-randomizations -- Again, each re-randomization generates a new table and we record the number of heart attacks in the Vioxx group...


proportion of tables
0.00
0.05
0.10
0.15
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
count of (CE and R)


proportion of tables
0.00
0.05
0.10
0.15
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
count of (CE and R)


proportion of tables
0.15
0.10
0.05
0.00
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
count of (CE and R)

Hypothesis testing

To compute the chance of seeing something more extreme than what we found, we sum the probabilities in the cyan boxes that hover over **both tails of the distributions**

Fisher's P-value is 0.23 meaning an event as extreme as the one we saw occurs in about 1 in 4 randomizations and **seems consistent with what we expect from the null --** In this case we cannot reject the null hypothesis that the two drugs carry the same risk of heart attack

Today

We are going to finish our discussion of random number generation and illustrate how R creates sequences of uniform random numbers

We will then start one final randomized trial, this one not arising from the health sciences but is instead a little tech-y -- Once we finish this example we will move from randomized trials to techniques for analysis when our data are sampled from some larger population

THE WALL STREET JOURNAL.

Senators Offer Privacy Bill to Protect Personal Data

By JULIA ANGWIN

Sens. John Kerry and John McCain proposed legislation Tuesday to create a "privacy bill of rights" to protect people from the increasingly invasive commercial data-collection industry.

The bill, labeled the Commercial Privacy Bill of Rights Act of 2011, would impose new rules on companies that gather personal data, including offering people access to data about them, or the ability to block the information from being used or distributed. Companies would have to seek permission before collecting and sharing sensitive religious, medical and financial data with outside entities.

The bipartisan proposal would create the nation's first comprehensive privacy law and largely adopts recommendations made by the Obama Administration last year. Current laws cover only the use of certain types of personal data, such as financial and medical information.

European Pressphoto Agency

Sens. John Kerry and John McCain

The move comes amid widening scrutiny of the commercial data-gathering industry, which has been chronicled in The Wall Street Journal's "What They Know" series. In his comments, Sen. McCain, an Arizona Republican, read an excerpt from the Journal series revealing that 56 popular cellphone applications transmitted information about users to outsiders without users' awareness or consent. "Customers must have control of their data when it is transferred to a third party," Sen. McCain said.

Random number generation

So far, we have emphasized the use of **graphics and simple simulation** (re-randomization) to analyze a data set -- We will circle back to talk about probability more formally next lecture (probably), but before that we should (probably) **put simulation on firmer footing**

For example, we seem to be trusting that R can generate “randomizations” for us, dividing or redividing patients into treatment and control -- In short, this means that **we can depend on the computer to toss coins for us**

How does it do this?

Random number generation

In Hill's trial, the samples were small enough that you cold rely on actual "random" mechanisms like **drawing tickets from a hat** (and we'll see lots of examples of the heroic work behind pre-computer simulation!)

Even **Francis Galton** (right, and we'll see a fair bit from him later too!) in the late 1880s recognized the need for statisticians to have access to simulated data (and suggested various physical mechanisms)

Besides randomized trials, toward what other purposes might we apply a sequence of random numbers?

DICE FOR STATISTICAL EXPERIMENTS.

EVERY statistician wants now and then to test the practical value of some theoretical process, it may be of smoothing, or of interpolation, or of obtaining a measure of variability, or of making some particular deduction or inference. It happened not long ago, while both a friend and myself were trying to find appropriate series for one of the above purposes, that the same week brought me letters from two eminent statisticians asking if I knew of any such series suitable for their own respective and separate needs. The assurance of a real demand for such things induced me to work out a method for supplying it, which I have already used frequently, and finding it to be perfectly effective, take this opportunity of putting it on record.

The desideratum is a set of values taken at random out of a series that is known to conform strictly to the law of frequency of error, the probable error of any single value in the series being also accurately known. We have (1) to procure such a series, and (2) to take random values out of it in an expeditious way.

Suppose the axis of the curve of distribution (whose ordinates at 100 equidistant divisions are given in my "Natural Inheritance," p. 205) to be divided into n equal parts, and that a column is erected on each of these, of a + or a − height as the case may be, equal to the height of the ordinate at the middle of each part. Then the values of these heights will form a series that is strictly conformable to the law of frequency when n is infinite, and closely conformable when n is fairly large. Moreover the probable error of any one of these values irrespectively of its sign, is 1.

As an instrument for selecting at random, I have found nothing superior to dice. It is most tedious to shuffle cards thoroughly between each successive draw, and the method of mixing and stirring up marked balls in a bag is more tedious still. A tee-totum or some form of roulette is preferable to these, but dice are better than all. When they are shaken and tossed in a basket, they hurtle so variously against one another and against the ribs of the basket-work that they tumble wildly about, and their positions at the outset afford no perceptible clue to what they will be after even a single good shake and toss. The chances afforded by a die are more various than are commonly supposed; there are 24 equal possibilities, and not only 6, because each face has four edges that may be utilized, as I shall show.

I use cubes of wood $1\frac{1}{4}$ inch in the side, for the dice. A carpenter first planed a bar of mahogany squarely and

... at one time in the not too distant past, this problem was addressed in a very direct way!

A MILLION
Random Digits

WITH

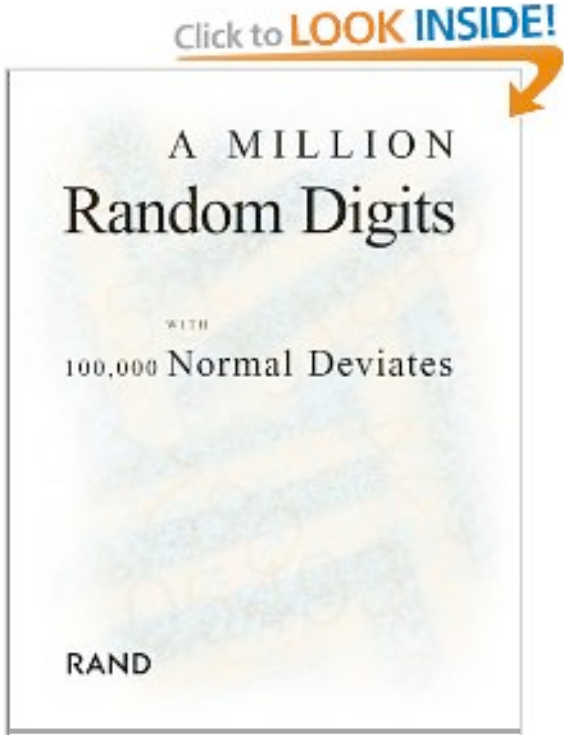
100,000 Normal Deviates

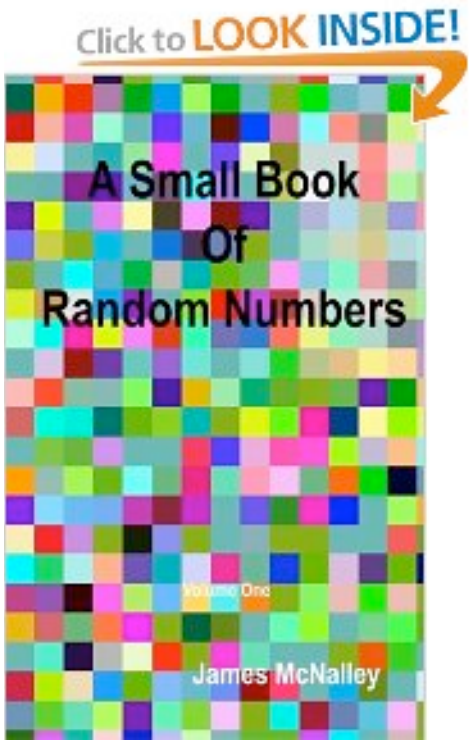
RAND

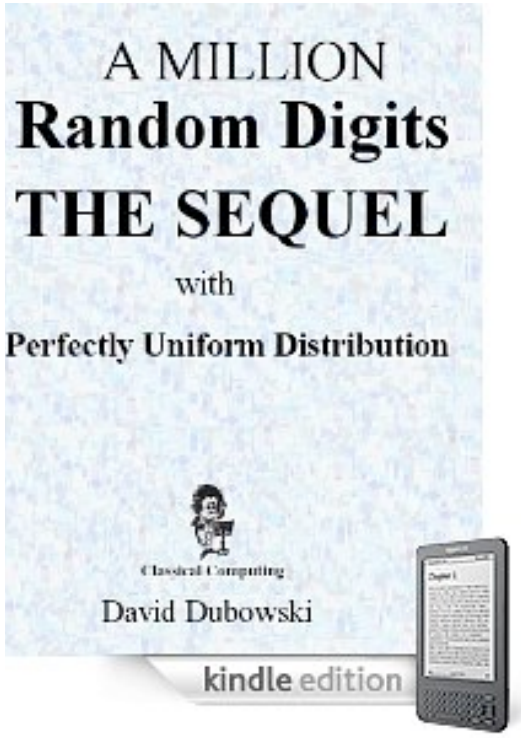
00000	10097	32533	76520	13586	34673	54876	80959	09117	39292	74945
00001	37542	04805	64894	74296	24805	24037	20636	10402	00822	91665
00002	08422	68953	19645	09303	23209	02560	15953	34764	35080	33606
00003	99019	02529	09376	70715	38311	31165	88676	74397	04436	27659
00004	12807	99970	80157	36147	64032	36653	98951	16877	12171	76833
00005	66065	74717	34072	76850	36697	36170	65813	39885	11199	29170
00006	31060	10805	45571	82406	35303	42614	86799	07439	23403	09732
00007	85269	77602	02051	65692	68665	74818	73053	85247	18623	88579
00008	63573	32135	05325	47048	90553	57548	28468	28709	83491	25624
00009	73796	45753	03529	64778	35808	34282	60935	20344	35273	88435
00010	98520	17767	14905	68607	22109	40558	60970	93433	50500	73998
00011	11805	05431	39808	27732	50725	68248	29405	24201	52775	67851
00012	83452	99634	06288	98083	13746	70078	18475	40610	68711	77817
00013	88685	40200	86507	58401	36766	67951	90364	76493	29609	11062
00014	99594	67348	87517	64969	91826	08928	93785	61368	23478	34113
00015	65481	17674	17468	50950	58047	76974	73039	57186	40218	16544
00016	80124	35635	17727	08015	45318	22374	21115	78253	14385	53763
00017	74350	99817	77402	77214	43236	00210	45521	64237	96286	02655
00018	69916	26803	66252	29148	36936	87203	76621	13990	94400	56418
00019	09893	20505	14225	68514	46427	56788	96297	78822	54382	14598
00020	91499	14523	68479	27686	46162	83554	94750	89923	37089	20048
00021	80336	94598	26940	36858	70297	34135	53140	33340	42050	82341
00022	44104	81949	85157	47954	32979	26575	57600	40881	22222	06413
00023	12550	73742	11100	02040	12860	74697	96644	89439	28707	25815
00024	63606	49329	16505	34484	40219	52563	43651	77082	07207	31790
00025	61196	90446	26457	47774	51924	33729	65394	59593	42582	60527
00026	15474	45266	95270	79953	59367	83848	82396	10118	33211	59466
00027	94557	28573	67897	54387	54622	44431	91190	42592	92927	45973
00028	42481	16213	97344	08721	16868	48767	03071	12059	25701	46670
00029	23523	78317	73208	89837	68935	91416	26252	29663	05522	82562
00030	04493	52494	75246	33824	45862	51025	61962	79335	65337	12472
00031	00549	97654	64051	88159	96119	63896	54692	82391	23287	29529
00032	35963	15307	26898	09354	33351	35462	77974	50024	90103	39333
00033	59808	08391	45427	26842	83609	49700	13021	24892	78565	20106
00034	46058	85236	01390	92286	77281	44077	93910	83647	70617	42941
00035	32179	00597	87379	25241	05567	07007	86743	17157	85394	11838
00036	69234	61406	20117	45204	15956	60000	18743	92423	97118	96338
00037	19565	41430	01758	75379	40419	21585	66674	36806	84962	85207
00038	45155	14938	19476	07246	43667	94543	59047	90033	20826	69541
00039	94864	31994	36168	10851	34888	81553	01540	35456	05014	51176
00040	98086	24826	45240	28404	44999	08896	39094	73407	35441	31880
00041	33185	16232	41941	50949	89435	48581	88695	41994	37548	73043
00042	80951	00406	96382	70774	20151	23387	25016	25298	94624	61171
00043	79752	49140	71961	28296	69861	02591	74852	20539	00387	59579
00044	18633	32537	98145	06571	31010	24674	05455	61427	77938	91936
00045	74029	43902	77557	32270	97790	17119	52527	58021	80814	51748
00046	54178	45611	80993	37143	05335	12969	56127	19255	36040	90324
00047	11664	49883	52079	84827	59381	71539	09973	33440	88461	23356
00048	48324	77928	31249	64710	02295	36870	32307	57546	15020	09994
00049	69074	94138	87637	91976	35584	04401	10518	21615	01848	76938

00050	09188	20097	32825	39527	04220	86304	83389	87374	64278	58044
00051	90045	85497	51981	50654	94938	81997	91870	76150	68476	64659
00052	73189	50207	47677	26269	62290	64464	27124	67018	41361	82760
00053	75768	76490	20971	87749	90429	12272	95375	05871	93823	43178
00054	54016	44056	66281	31003	00682	27398	20714	53295	07706	17813
00055	08358	69910	78542	42785	13661	58873	04618	97553	31223	08420
00056	28306	03264	81333	10591	40510	07893	32604	60475	94119	01840
00057	53840	86233	81594	13628	51215	90290	28466	68795	77762	20791
00058	91757	53741	61613	62269	50263	90212	55781	76514	83483	47055
00059	89415	92694	00397	58391	12607	17646	48949	72306	94541	37408
00060	77513	03820	86864	29901	68414	82774	51908	13980	72893	55507
00061	19502	37174	69979	20288	55210	29773	74287	75251	65344	67415
00062	21818	59313	93278	81757	05686	73156	07082	85046	31853	38452
00063	51474	66499	68107	23621	94049	91345	42836	09191	08007	45449
00064	99559	68331	62535	24170	69777	12830	74819	78142	43860	72834
00065	33713	48007	93584	72869	51926	64721	58303	29822	93174	93972
00066	85274	86893	11303	22970	28834	34137	73515	90400	71148	43643
00067	84133	89640	44035	52166	73852	70091	61222	60561	62327	18423
00068	56732	16234	17395	96131	10123	91622	85496	57560	81604	18880
00069	65138	56806	87648	85261	34313	65861	45875	21069	85644	47277
00070	38001	02176	81719	11711	71602	92937	74219	64049	65584	49698
00071	37402	96397	01304	77586	56271	10086	47324	62605	40030	37438
00072	97125	40348	87083	31417	21815	39250	75237	62047	15501	29578
00073	21826	41134	47143	34072	64638	85902	49139	06441	03856	54552
00074	73135	42742	95719	09035	85794	74296	08789	88156	64691	19202
00075	07638	77929	03061	18072	96207	44156	23821	99538	04713	66994
00076	60528	83441	07954	19814	59175	20695	05533	52139	61212	06455
00077	83596	35655	06958	92983	05128	09719	77433	53783	92301	50498
00078	10850	62746	99599	10507	13499	06319	53075	71839	06410	19362
00079	39820	98952	43622	63147	64421	80814	43800	09351	31024	73167
00080	59580	06478	75569	78800	88835	54486	23768	06156	04111	08408
00081	38508	07341	23793	48763	90822	97022	17719	04207	95954	49953
00082	30692	70668	94688	16127	56196	80091	82067	63400	05462	69200
00083	65443	95659	18288	27437	49632	24041	08337	65676	96299	90836
00084	27267	50264	13192	72294	07477	44606	17985	48911	97341	30358
00085	91307	06991	19072	24210	36699	53728	28825	35793	28976	66252
00086	68434	94688	84473	13622	62126	98408	12843	82590	09815	93146
00087	48908	15877	54745	24591	35700	04754	83824	52692	54130	55160
00088	06913	45197	42672	78601	11883	09528	63011	98901	14974	40344
00089	10455	16019	14210	33712	91342	37821	88325	80851	43667	70883
00090	12883	97343	65027	61184	04285	01392	17974	15077	90712	26769
00091	21778	30976	38807	36961	31649	42096	63281	02023	08816	47449
00092	19523	59515	65122	59659	86283	68258	69572	13798	16435	91529
00093	67245	52670	35583	16563	79246	86686	76463	34222	26655	90802
00094	60584	47377	07500	37992	45134	26529	26760	83637	41326	44344
00095	53853	41377	36066	94850	58838	73859	49364	73331	96240	43642
00096	24637	38736	74384	89342	52623	07992	12369	18601	03742	83873
00097	83080	12451	38992	22815	07759	51777	97377	27585	51972	37867
00098	16444	24334	36151	99073	27483	70939	85130	32552	54846	54759
00099	60790	18157	57178	65762	11161	78576	45819	52979	65130	04860

00100	03991	10461	93716	16894	66083	24653	84609	58232	88618	19161
00101	38555	95554	32886	59780	08355	60860	29735	47762	71299	23853
00102	17546	73704	92052	46215	55121	29281	59076	07936	27954	58909
00103	32643	52861	95819	06831	00911	98936	76355	93779	80863	00514
00104	69572	68777	39510	35905	14060	40619	29549	69616	33564	60780
00105	24122	66591	27699	06494	14845	46672	61958	77100	90899	75754
00106	61196	30231	92962	61773	41839	55382	17267	70943	78038	70267
00107	30532	21704	10274	12202	39665	23309	10061	68829	55986	66485
00108	03788	97599	75867	20717	74416	53166	35208	33374	87539	08823
00109	48228	63379	85783	47619	53152	67433	35663	52972	16818	60311
00110	60365	94653	35075	33949	42614	29297	01918	28316	98953	73231
00111	83799	42402	56623	34442	34994	41374	70071	14736	09958	18065
00112	32960	07405	36409	83232	99385	41600	11133	07586	15917	06253
00113	19322	53845	57620	52606	66497	68646	78138	66559	19640	99413
00114	11220	94747	07399	37408	48509	23929	27482	45476	85244	35159
00115	31751	57260	68980	05339	15470	48355	88651	22596	03152	19121
00116	88492	99382	14454	04504	20094	98977	74843	93413	22109	78508
00117	30934	47744	07481	83828	73788	06533	28597	20405	94205	20380
00118	22888	48893	27499	98748	60530	45128	74022	84617	82037	10268
00119	78212	16993	35902	91386	44372	15486	65741	14014	87481	37220
00120	41849	84547	46850	52326	34677	58300	74910	64345	19325	81549
00121	46352	33049	69248	93460	45305	07521	61318	31855	14413	70951
00122	11087	96294	14013	31792	59747	67277	76503	34513	39663	77544
00123	52701	08337	56303	87315	16520	69676	11654	99893	02181	68161
00124	57275	36898	81304	48585	68652	27376	92852	55866	88448	03584
00125	20857	73156	70284	24326	79375	95220	01159	63267	10622	48391
00126	15633	84924	90415	93614	33521	26665	55823	47641	86225	31704
00127	92694	48297	39904	02115	59589	49067	66821	41575	49767	04037
00128	77613	19019	88152	00080	20554	91409	96277	48257	50816	97616
00129	38688	32486	45134	63545	59404	72059	43947	51680	43852	59693
00130	25163	01889	70014	15021	41290	67312	71857	15957	68971	11403
00131	65251	07629	37239	33295	05870	01119	92784	26340	18477	65622
00132	36815	43625	18637	37509	82444	99005	04921	73701	14707	93997
00133	64397	11692	05327	82162	20247	81759	45197	25332	83745	22567
00134	04515	25624	95096	67946	48460	85558	15191	18782	16930	33361
00135	83761	60873	43253	84145	60833	25983	01291	41349	20368	07126
00136	14387	06345	80854	09279	43529	06318	38384	74761	41196	37480
00137	51321	92246	80088	77074	88722	56736	66164	49431	66919	31678
00138	72472	00008	80890	18002	94813	31900	54155	83436	35352	54131
00139	05466	55306	93128	18464	74457	90561	72848	11834	79982	68416
00140	39528	72484	82474	25593	48545	35247	18619	13674	18611	19241
00141	81616	18711	53342	44276	75122	11724	74627	73707	58319	15997
00142	07586	16120	82641	22820	92904	13141	32392	19763	61199	67940
00143	90767	04235	13574	17200	69902	63742	78464	22501	18627	90872
00144	40188	28193	29593	88627	94972	11598	62095	36787	00441	58997
00145	34414	82157	86887	55087	19152	00023	12302	80783	32624	68691
00146	63439	75363	44989	16822	36024	00867	76378	41605	65961	73488
00147	67049	09070	93399	45547	94458	74284	05041	49807	20288	34060
00148	79495	04146	52162	90286	54158	34243	46978	35482	59362	95938
00149	91704	30552	04737	21031	75051	93029	47665	64382	99782	93478


Click to LOOK INSIDE!
A MILLION
Random Digits
WITH
100,000 Normal Deviates
RAND


Click to LOOK INSIDE!
A Small Book
Of
Random Numbers
James McNalley


A MILLION
Random Digits
THE SEQUEL
with
Perfectly Uniform Distribution
David Dubowski
kindle edition

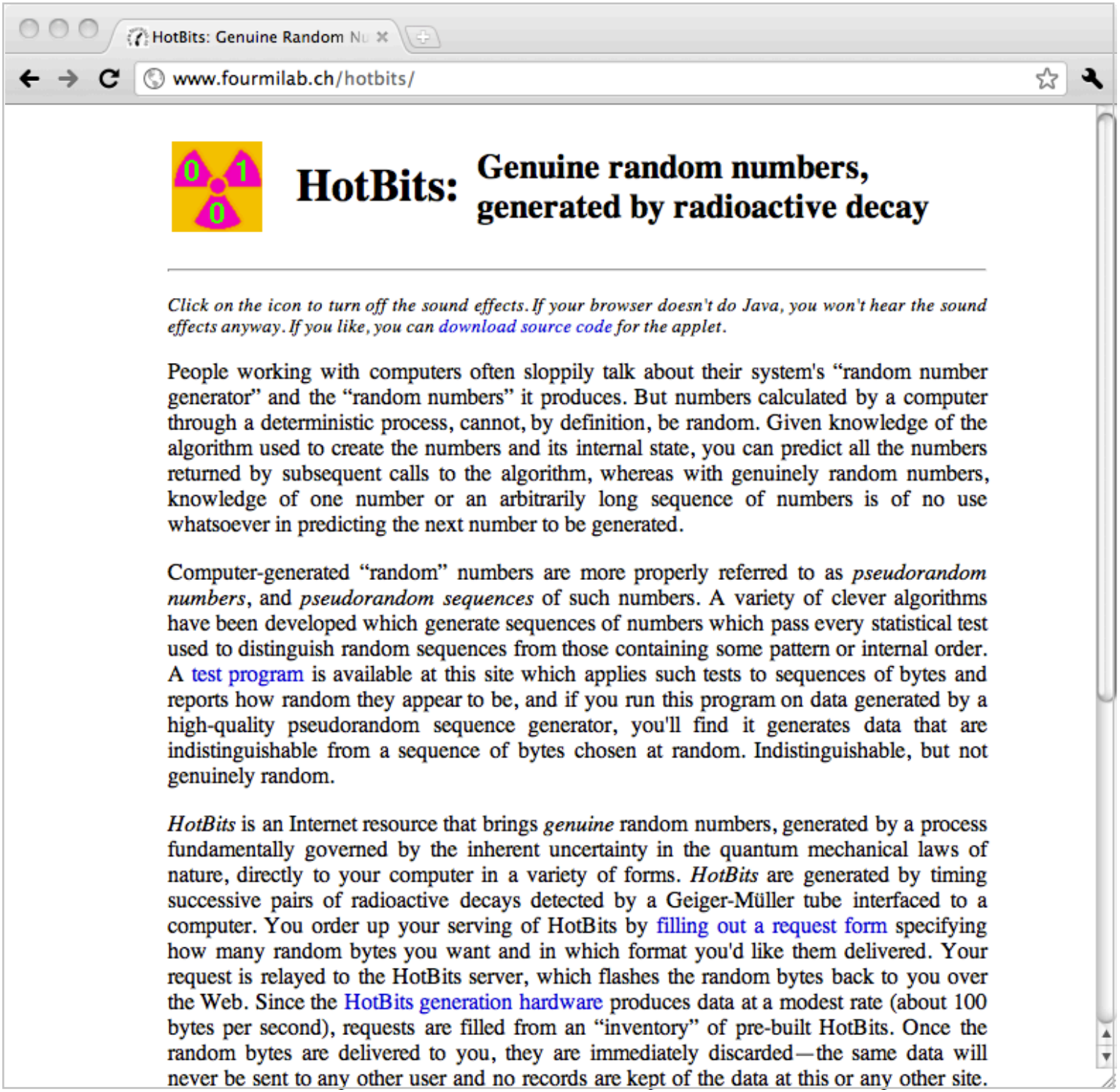
Random number generation

These days, there are two dominant techniques for generating random numbers

One is not really random according to any romantic notions of the word and are the result of **a mathematical formula** which is entirely predictable and repeatable -- These are often called **pseduo-random numbers**

The second, on the other hand, is often touted as **“true” random numbers** and are generated by **observing some physical process** -- You can think of a small coin-tossing device attached to your computer although the physical phenomena used tend to be more exotic

Let’s have a look at both, starting with the latter as it will give us to talk about data and the publication of data (a theme for this course)

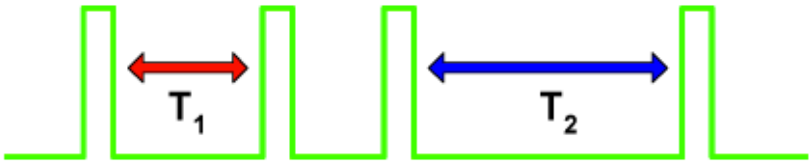

HotBits: Genuine Random Nu
www.fourmilab.ch/hotbits/
HotBits: Genuine random numbers, generated by radioactive decay
Click on the icon to turn off the sound effects. If your browser doesn't do Java, you won't hear the sound effects anyway. If you like, you can download source code for the applet.
People working with computers often sloppily talk about their system's "random number generator" and the "random numbers" it produces. But numbers calculated by a computer through a deterministic process, cannot, by definition, be random. Given knowledge of the algorithm used to create the numbers and its internal state, you can predict all the numbers returned by subsequent calls to the algorithm, whereas with genuinely random numbers, knowledge of one number or an arbitrarily long sequence of numbers is of no use whatsoever in predicting the next number to be generated.
Computer-generated "random" numbers are more properly referred to as pseudorandom numbers, and pseudorandom sequences of such numbers. A variety of clever algorithms have been developed which generate sequences of numbers which pass every statistical test used to distinguish random sequences from those containing some pattern or internal order. A test program is available at this site which applies such tests to sequences of bytes and reports how random they appear to be, and if you run this program on data generated by a high-quality pseudorandom sequence generator, you'll find it generates data that are indistinguishable from a sequence of bytes chosen at random. Indistinguishable, but not genuinely random.
HotBits is an Internet resource that brings genuine random numbers, generated by a process fundamentally governed by the inherent uncertainty in the quantum mechanical laws of nature, directly to your computer in a variety of forms. HotBits are generated by timing successive pairs of radioactive decays detected by a Geiger-Müller tube interfaced to a computer. You order up your serving of HotBits by filling out a request form specifying how many random bytes you want and in which format you'd like them delivered. Your request is relayed to the HotBits server, which flashes the random bytes back to you over the Web. Since the HotBits generation hardware produces data at a modest rate (about 100 bytes per second), requests are filled from an "inventory" of pre-built HotBits. Once the random bytes are delivered to you, they are immediately discarded—the same data will never be sent to any other user and no records are kept of the data at this or any other site.

Bit from It

This inherent randomness in decay time has profound implications, which we will now exploit to generate random numbers—HotBits. For if there's no way to know when a given Cæsium-137 nucleus will decay then, given an collection of them, there's no way to know when the *next* one of them will shoot its electron bolt and settle down to a serene eternity as Barium. That's uncertainty, with its origins in the deepest and darkest corners of creation—precisely what we're looking for to make genuinely random numbers.

If we knew the precise half-life of the radioactive source driving our detector (and other details such as the solid angle to which our detector is sensitive, the energy range of decay products and the sensitivity of the detector to them, and so on), we could generate random bits by measuring whether the time between a pair of beta decays was more or less than the time expected based on the half-life. But that would require our knowing the average beta decay detection time, which depends on a large number of parameters which can only be determined experimentally. Instead, we can exploit the inherent uncertainty of decay time in a parameter-free fashion which requires less arm waving and fancy footwork.

The trick I use was dreamed up in a conversation in 1985 with John Nagle, who is doing some fascinating things these days with artificial animals. Since the time of any given decay is random, then the *interval* between two consecutive decays is also random. What we do, then, is measure a pair of these intervals, and emit a zero or one bit based on the relative length of the two intervals. If we measure the same interval for the two decays, we discard the measurement and try again, to avoid the risk of inducing bias due to the resolution of our clock.



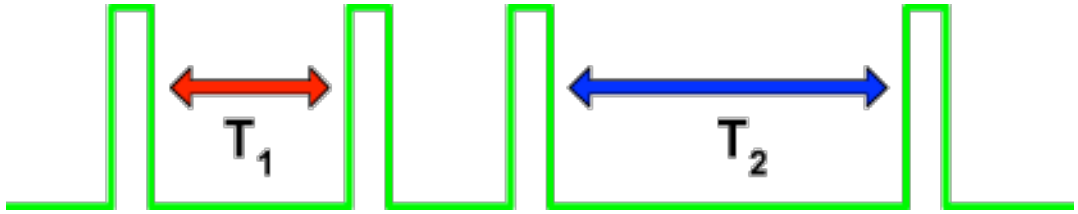
To create each random bit, we wait until the first count occurs, then measure the time, T_1, until the next. We then wait for a second pair of pulses and measure the interval T_2 between them, yielding a pair of durations. If they're the same, we throw away the measurement and try again. Otherwise if T_1 is less than T_2 we emit a zero bit; if T_1 is greater than T_2, a one bit. In practice, to avoid any residual bias resulting from non-

Bits

A “bit” stands for a “binary digit” and takes on the value 0 or 1 -- You can think of it as a coin toss where we map “heads” to 1, say, and “tails” to 0 (The term “bit” was actually coined by our man John Tukey; he also came up with the term “software”)

HotBits uses radioactive decay as a means for generating physically random or “true” random bits (coin tosses) -- **You can imagine listening to a Geiger counter and group the ticks into pairs**

If $T_1 > T_2$ they produce a 1, otherwise they generate a 0 -- **Voila! “true” random numbers!**



Bits

Another project, this run through the web site `random.org` uses atmospheric noise, suitably filtered, to accomplish the same task

Our interest in this site was because of the mechanism they used to “publish” their random bits, **offering a service that scientists around the world** -- They have implemented **a public API (application programming interface)** that let’s programs (like R) request data (random bits)


RANDOM.ORG – True Rando
www.random.org
Home Games Numbers Lists & More Drawings Web Tools Statistics Testimonials Learn More Login
RANDOM.ORG
Search RANDOM.ORG
Search
True Random Number Service
What's this fuss about true randomness?
Perhaps you have wondered how predictable machines like computers can generate randomness. In reality, most random numbers used in computer programs are pseudo-random, which means they are a generated in a predictable fashion using a mathematical formula. This is fine for many purposes, but it may not be random in the way you expect if you're used to dice rolls and lottery drawings.
RANDOM.ORG offers true random numbers to anyone on the Internet. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs. People use RANDOM.ORG for holding drawings, lotteries and sweepstakes, to drive games and gambling sites, for scientific applications and for art and music. The service has existed since 1998 and was built and is being operated by Mads Haahr of the School of Computer Science and Statistics at Trinity College, Dublin in Ireland.
As of today, RANDOM.ORG has generated 958.5 billion random bits for the Internet community.
True Random Number Generator
Min: 1
Max: 100
Generate
Result:
Powered by RANDOM.ORG
Like RANDOM.ORG?
Get the Newsletter
FREE services
Games and Gambling
Lottery Quick Pick is perhaps the Internet's most popular with over 130 lotteries
Keno Quick Pick for the popular game played at many casinos
Coin Flipper will give you heads or tails in many currencies
Dice Roller does exactly what it says on the tin
Playing Card Shuffler will draw cards from multiple shuffled decks
Birdie Fund Generator will create birdie holes for golf courses
PAID service
Random Drawings
Q3.1 in the FAQ explains how to pick a winner for your giveaway for FREE
Third-Party Draw Service is the premier solution to holding random drawings online
Step by Step Guide explains how to hold a drawing with the Third-Party Draw Service
Step by Step Video shows how to hold a drawing with the Third-Party Draw Service
Price Calculator tells exactly how much your drawing will cost
Drawing FAQ answers common questions about holding drawings

Another service

With the advent of Web 2.0, the dominant method of data distribution has changed **from simply serving up web pages** and, along with it, we have experienced a massive shift in our view of the web itself -- In 2005, the term Web 2.0 emerged to represent this new view, one of **“collective intelligence” of “data sharing” and “web services”**

To deliver on this version of the web as a system of **cooperating data services**, we need techniques to specify the kind of data we want and specify the format we expect to see it in -- For `random.org`, it’s pretty easy...

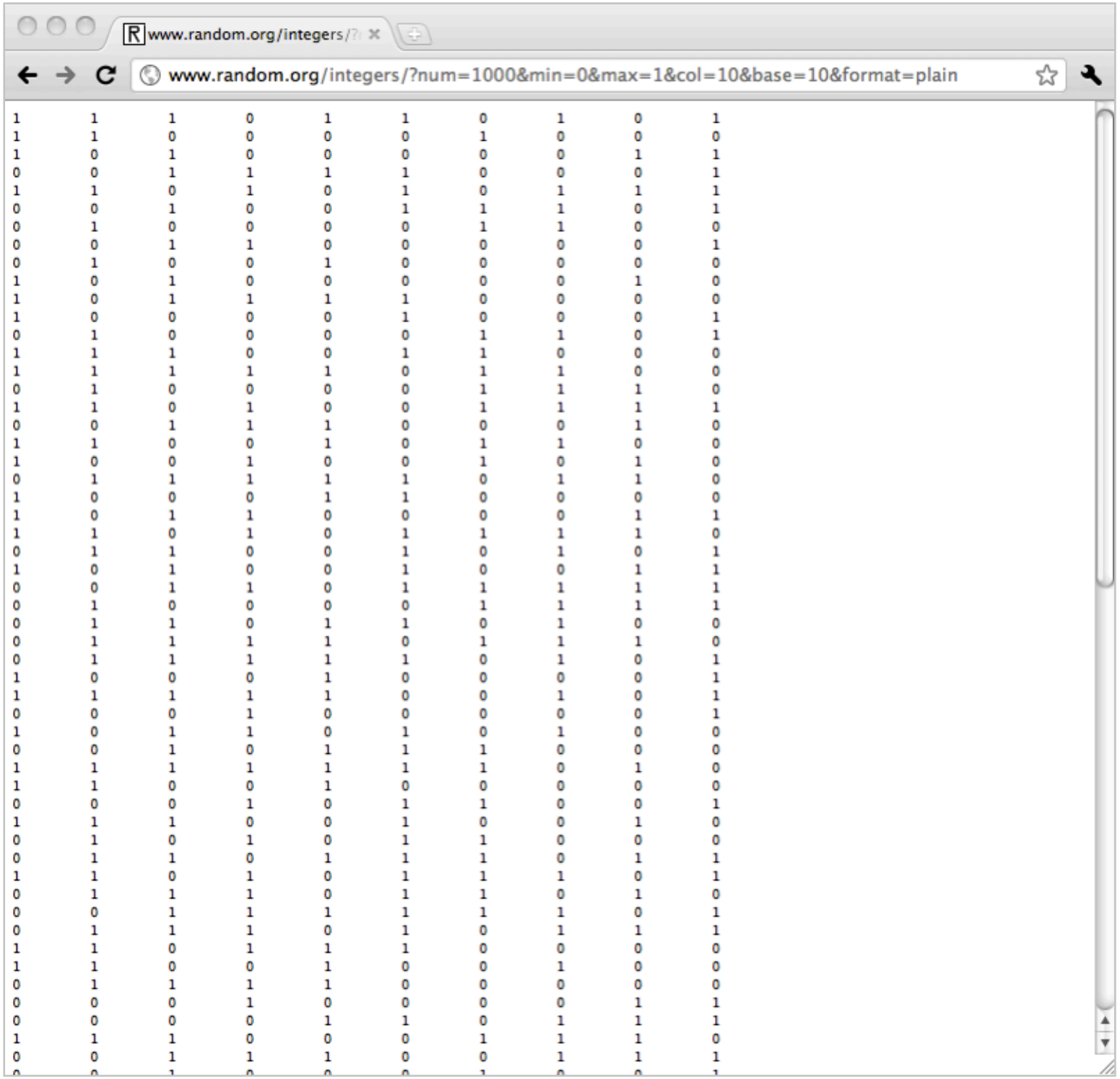

RANDOM.ORG – HTTP Interf
www.random.org/clients/http/
Home Games Numbers Lists & More Drawings Web Tools Statistics Testimonials Learn More Login
RANDOM.ORG
Search RANDOM.ORG
Search
True Random Number Service
HTTP Interface Description
RANDOM.ORG is a true random number service that generates randomness via atmospheric noise. This page explains how to interface to the service via the Hyper-Text Transfer Protocol (HTTP). There is also the HTTP Client Archive, which contains clients that other people have written.
Important note!
If you access RANDOM.ORG via an automated client, please make sure you observe the Guidelines for Automated Clients or your computer may be banned.
If you are writing a general-purpose client, please make sure it is easy for your users to run it in accordance with the guidelines.
This page contains documentation for the Integer Generator, the Sequence Generator, the String Generator and the Quota Checker, which allows you to examine your current bit allowance.
All the interfaces on this page return HTTP status code 503 (Service Unavailable) in the case of errors and code 200 (OK) when successful. Not all languages allow you to access the HTTP status codes in a straightforward manner. A reasonable workaround is to look for the string "Error:" (don't forget the colon) as the first line of the response. This will work for all the generators on this page, including the String Generator (which could by chance produce the string "Error" in a successful response, but which cannot produce the colon character).
Please note that the old CGI scripts (randbyte, randnum, etc.) are no longer supported and you should use the ones described below instead. In particular, the old scripts do not return the 503 status code in case of errrors (they return the 200 response code in all cases), so please use the new ones instead.
Integer Generator
The Integer Generator will generate truly random integers in configurable intervals. It is pretty easy to write a client to access the integer generator. The integer generator accepts only HTTP GET requests, so parameters are passed via encoding in the URL.
Parameters
Name
Possible Values
Description

Integer Generator

The Integer Generator will generate truly random integers in configurable intervals. It is pretty easy to write a client to access the integer generator. The integer generator accepts only HTTP GET requests, so parameters are passed via encoding in the URL.

Parameters

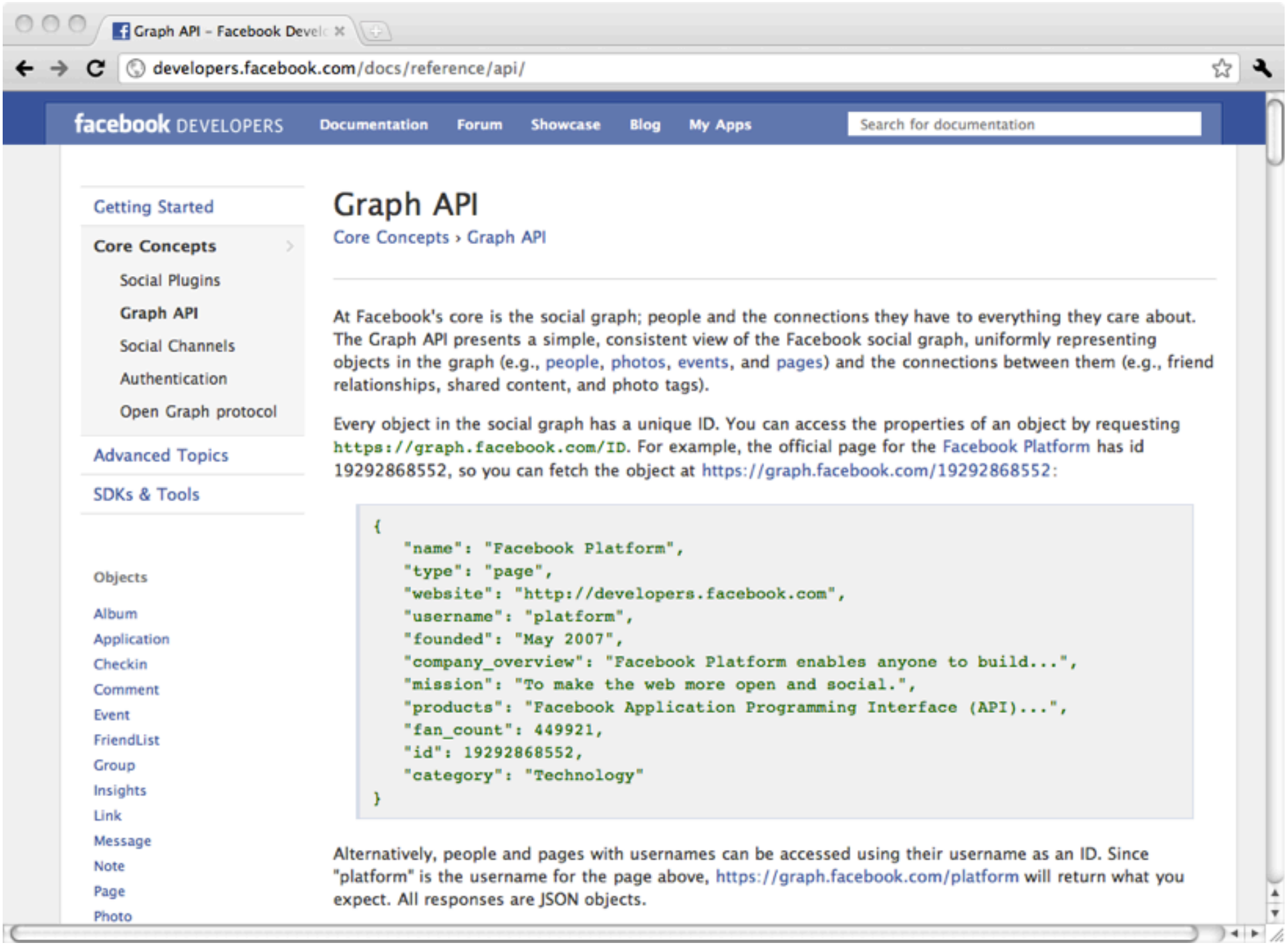
Name	Possible Values	Description
num	[1,1e4]	The number of integers requested.
min	[-1e9,1e9]	The smallest value allowed for each integer.
max	[-1e9,1e9]	The largest value allowed for each integer.
col	[1,1e9]	The number of columns in which the integers will be arranged. The integers should be read (or processed) left to right across columns.
base	**2** | **8** | **10** | **16**	The base that will be used to print the numbers, i.e., binary, octal, decimal or hexadecimal.
format	**html** | **plain**	Determines the return type of the document that the server produces as its response. If **html** is specified, the server produces a nicely formatted XHTML document (MIME type text/html), which will display well in a browser but which is somewhat cumbersome to parse. If **plain** is specified, the server produces as minimalistic document of type plain text (MIME type text/plain) document, which is easy to parse. If you are writing an automated client, you probably want to specify **plain** here.
rnd	**new** | **id.identifier** | **date.iso-date**	Determines the randomization to use to generate the numbers. If **new** is specified, then a new randomization will created from the truly random bitstream at RANDOM.ORG. This is probably what you want in most cases. If **id.identifier** is specified, the identifier is used to determine the randomization in a deterministic fashion from a large pool of pregenerated random bits. Because the numbers are produced in a deterministic fashion, specifying an id basically uses RANDOM.ORG as a pseudo-random number generator. The third (**date.iso-date**) form is similar to the second; it allows the randomization to be based on one of the daily pregenerated files. This form must refer to one of the dates for which files exist, so it must be the current day (according to UTC) or a day in the past. The date must be in ISO 8601 format (i.e., YYYY-MM-DD) or one of the two shorthand strings **today** or **yesterday**.

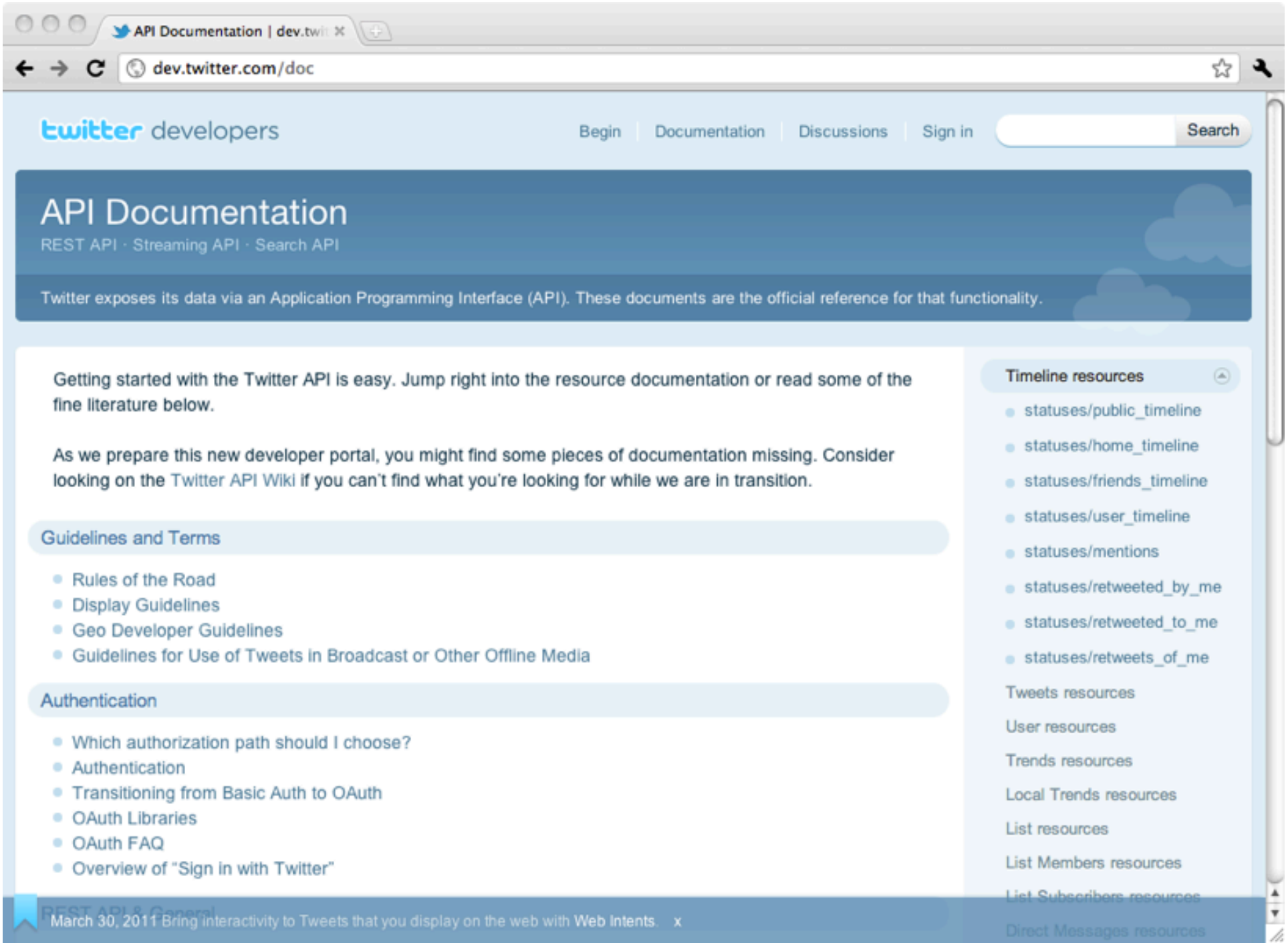

www.random.org/integers/
www.random.org/integers/?num=1000&min=0&max=1&col=10&base=10&format=plain
1 1 1 0 1 1 0 1 0 1
1 1 0 0 0 0 1 0 0 0
1 0 1 0 0 0 0 0 1 1
0 0 1 1 1 1 0 0 0 1
1 1 0 1 0 1 0 1 1 1
0 0 1 0 0 1 1 1 0 1
0 1 0 0 0 0 1 1 0 0
0 0 1 1 0 0 0 0 0 1
0 1 0 0 1 0 0 0 0 0
1 0 1 0 0 0 0 0 1 0
1 0 1 1 1 1 0 0 0 0
1 0 0 0 0 1 0 0 0 1
0 1 0 0 0 0 1 1 0 1
1 1 1 0 0 1 1 0 0 0
1 1 1 1 1 0 1 1 0 0
0 1 0 0 0 0 1 1 1 0
1 1 0 1 0 0 1 1 1 1
0 0 1 1 1 0 0 0 1 0
1 1 0 0 1 0 1 1 0 0
1 0 0 1 0 0 1 0 1 0
0 1 1 1 1 1 0 1 1 0
1 0 0 0 1 1 0 0 0 0
1 0 1 1 0 0 0 0 1 1
1 1 0 1 0 1 1 1 1 0
0 1 1 0 0 1 0 1 0 1
1 0 1 0 0 1 0 0 1 1
0 0 1 1 0 1 1 1 1 1
0 1 0 0 0 0 1 1 1 1
0 1 1 0 1 1 0 1 0 0
0 1 1 1 1 0 1 1 1 0
0 1 1 1 1 1 0 1 0 1
1 0 0 0 1 0 0 0 0 1
1 1 1 1 1 0 0 1 0 1
0 0 0 1 0 0 0 0 0 1
1 0 1 1 0 1 0 1 0 0
0 0 1 0 1 1 1 0 0 0
1 1 1 1 1 1 1 0 1 0
1 1 0 0 1 0 0 0 0 0
0 0 0 1 0 1 1 0 0 1
1 1 1 0 0 1 0 0 1 0
0 1 0 1 0 1 1 0 0 0
0 1 1 0 1 1 1 0 1 1
1 1 0 1 0 1 1 1 0 1
0 1 1 1 0 1 1 0 1 0
0 0 1 1 1 1 1 1 0 1
0 1 1 1 0 1 0 1 1 1
1 1 0 1 1 1 0 0 0 0
1 1 0 0 1 0 0 1 0 0
0 1 1 1 1 0 0 0 0 0
0 0 0 1 0 0 0 0 1 1
0 0 0 0 1 1 0 1 1 1
1 1 1 0 0 0 1 1 1 0
0 0 1 1 1 0 0 1 1 1

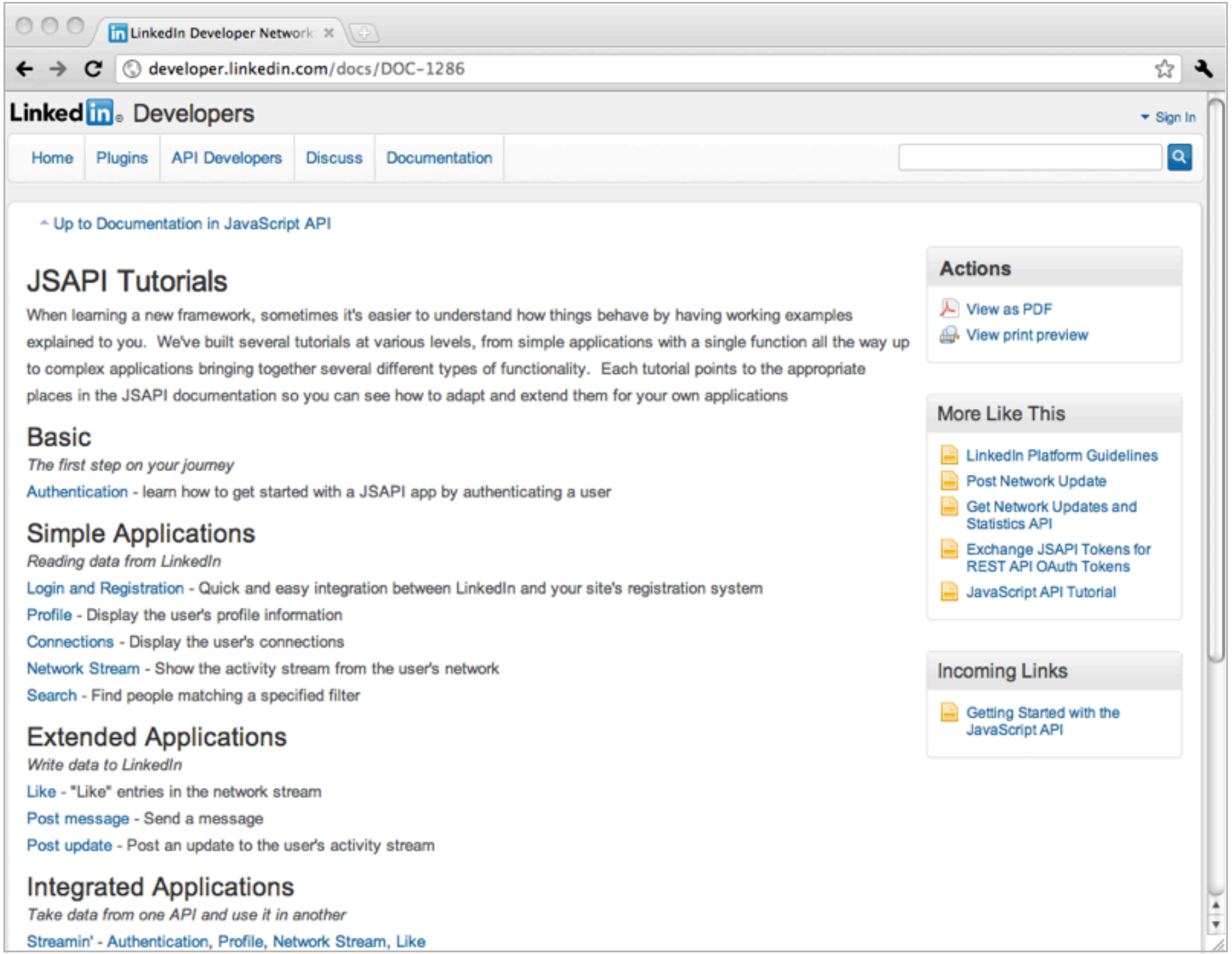
APIs

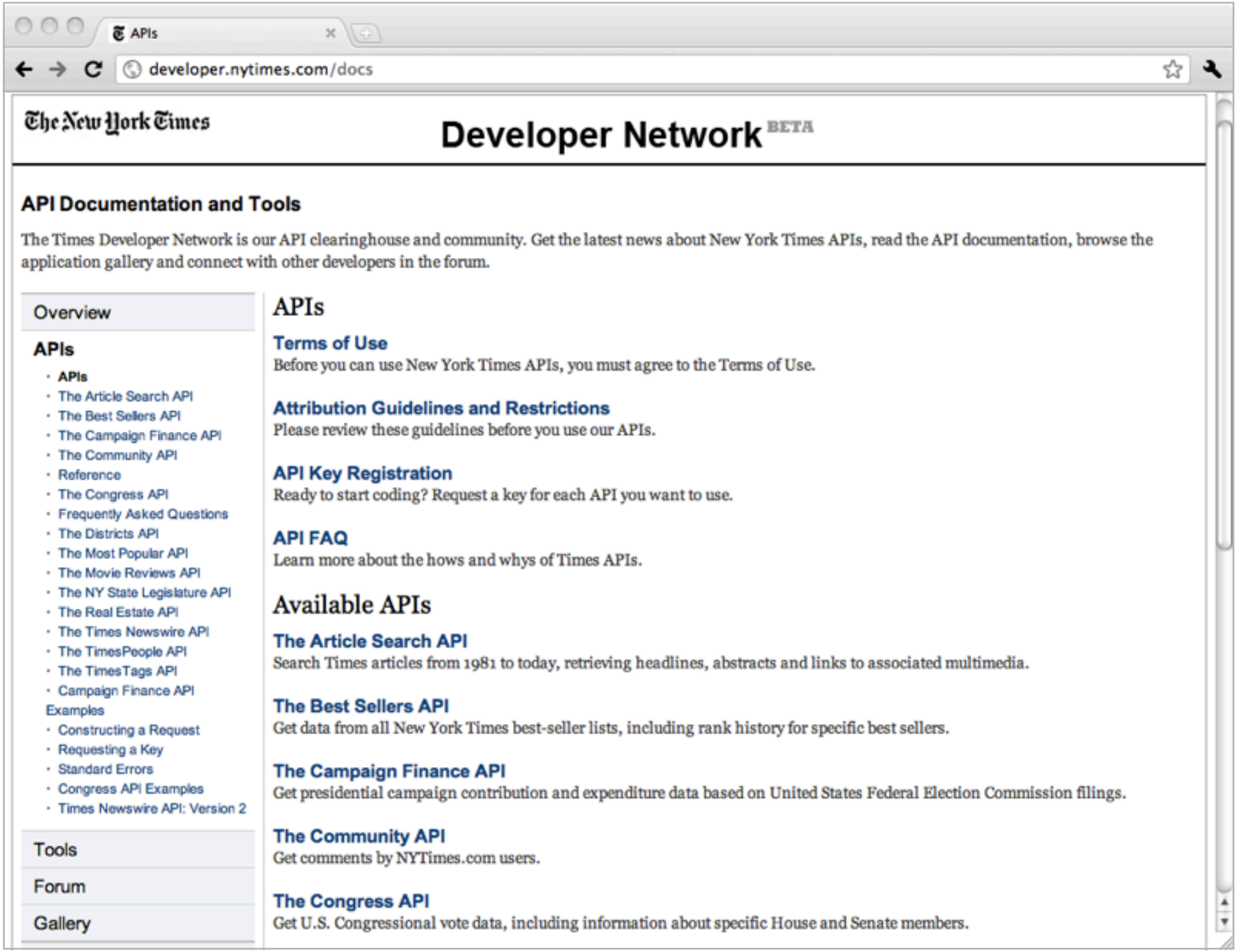
I mention this because many organizations offer their data via web services like this -- Specific HTTP requests yield not web pages designed for human consumption (reading), but instead **structured data that can be processed easily in an autonomous way by computers**

In this way, services build “mash-up” style, with data flowing easily between different organizations -- This is really an amazing development that opens up incredible possibilities for statistics, for computer science, for machine learning for the data geeks out there!


Graph API - Facebook Devel
developers.facebook.com/docs/reference/api/
facebook DEVELOPERS
Documentation
Forum
Showcase
Blog
My Apps
Search for documentation
Getting Started
Core Concepts
Social Plugins
Graph API
Social Channels
Authentication
Open Graph protocol
Advanced Topics
SDKs & Tools
Objects
Album
Application
Checkin
Comment
Event
FriendList
Group
Insights
Link
Message
Note
Page
Photo
Graph API
Core Concepts › Graph API
At Facebook's core is the social graph; people and the connections they have to everything they care about. The Graph API presents a simple, consistent view of the Facebook social graph, uniformly representing objects in the graph (e.g., people, photos, events, and pages) and the connections between them (e.g., friend relationships, shared content, and photo tags).
Every object in the social graph has a unique ID. You can access the properties of an object by requesting https://graph.facebook.com/ID. For example, the official page for the Facebook Platform has id 19292868552, so you can fetch the object at https://graph.facebook.com/19292868552:
{
"name": "Facebook Platform",
"type": "page",
"website": "http://developers.facebook.com",
"username": "platform",
"founded": "May 2007",
"company_overview": "Facebook Platform enables anyone to build...",
"mission": "To make the web more open and social.",
"products": "Facebook Application Programming Interface (API)...",
"fan_count": 449921,
"id": 19292868552,
"category": "Technology"
}
Alternatively, people and pages with usernames can be accessed using their username as an ID. Since "platform" is the username for the page above, https://graph.facebook.com/platform will return what you expect. All responses are JSON objects.


API Documentation | dev.twit
dev.twitter.com/doc
twitter developers
Begin
Documentation
Discussions
Sign in
Search
API Documentation
REST API · Streaming API · Search API
Twitter exposes its data via an Application Programming Interface (API). These documents are the official reference for that functionality.
Getting started with the Twitter API is easy. Jump right into the resource documentation or read some of the fine literature below.
As we prepare this new developer portal, you might find some pieces of documentation missing. Consider looking on the Twitter API Wiki if you can't find what you're looking for while we are in transition.
Guidelines and Terms
Rules of the Road
Display Guidelines
Geo Developer Guidelines
Guidelines for Use of Tweets in Broadcast or Other Offline Media
Authentication
Which authorization path should I choose?
Authentication
Transitioning from Basic Auth to OAuth
OAuth Libraries
OAuth FAQ
Overview of "Sign in with Twitter"
REST API & General
Timeline resources
statuses/public_timeline
statuses/home_timeline
statuses/friends_timeline
statuses/user_timeline
statuses/mentions
statuses/retweeted_by_me
statuses/retweeted_to_me
statuses/retweets_of_me
Tweets resources
User resources
Trends resources
Local Trends resources
List resources
List Members resources
List Subscribers resources
Direct Messages resources
March 30, 2011 Bring interactivity to Tweets that you display on the web with Web Intents.
x


LinkedIn Developer Network
developer.linkedin.com/docs/DOC-1286
LinkedIn Developers
Sign In
Home
Plugins
API Developers
Discuss
Documentation
Up to Documentation in JavaScript API
JSAPI Tutorials
When learning a new framework, sometimes it's easier to understand how things behave by having working examples explained to you. We've built several tutorials at various levels, from simple applications with a single function all the way up to complex applications bringing together several different types of functionality. Each tutorial points to the appropriate places in the JSAPI documentation so you can see how to adapt and extend them for your own applications
Basic
The first step on your journey
Authentication - learn how to get started with a JSAPI app by authenticating a user
Simple Applications
Reading data from LinkedIn
Login and Registration - Quick and easy integration between LinkedIn and your site's registration system
Profile - Display the user's profile information
Connections - Display the user's connections
Network Stream - Show the activity stream from the user's network
Search - Find people matching a specified filter
Extended Applications
Write data to LinkedIn
Like - "Like" entries in the network stream
Post message - Send a message
Post update - Post an update to the user's activity stream
Integrated Applications
Take data from one API and use it in another
Streamin' - Authentication, Profile, Network Stream, Like
Actions
View as PDF
View print preview
More Like This
LinkedIn Platform Guidelines
Post Network Update
Get Network Updates and Statistics API
Exchange JSAPI Tokens for REST API OAuth Tokens
JavaScript API Tutorial
Incoming Links
Getting Started with the JavaScript API


APIs
developer.nytimes.com/docs
The New York Times
Developer Network BETA
API Documentation and Tools
The Times Developer Network is our API clearinghouse and community. Get the latest news about New York Times APIs, read the API documentation, browse the application gallery and connect with other developers in the forum.
Overview
APIs
APIs
The Article Search API
The Best Sellers API
The Campaign Finance API
The Community API
Reference
The Congress API
Frequently Asked Questions
The Districts API
The Most Popular API
The Movie Reviews API
The NY State Legislature API
The Real Estate API
The Times Newswire API
The TimesPeople API
The TimesTags API
Campaign Finance API Examples
Constructing a Request
Requesting a Key
Standard Errors
Congress API Examples
Times Newswire API: Version 2
Tools
Forum
Gallery
APIs
Terms of Use
Before you can use New York Times APIs, you must agree to the Terms of Use.
Attribution Guidelines and Restrictions
Please review these guidelines before you use our APIs.
API Key Registration
Ready to start coding? Request a key for each API you want to use.
API FAQ
Learn more about the hows and whys of Times APIs.
Available APIs
The Article Search API
Search Times articles from 1981 to today, retrieving headlines, abstracts and links to associated multimedia.
The Best Sellers API
Get data from all New York Times best-seller lists, including rank history for specific best sellers.
The Campaign Finance API
Get presidential campaign contribution and expenditure data based on United States Federal Election Commission filings.
The Community API
Get comments by NYTimes.com users.
The Congress API
Get U.S. Congressional vote data, including information about specific House and Senate members.

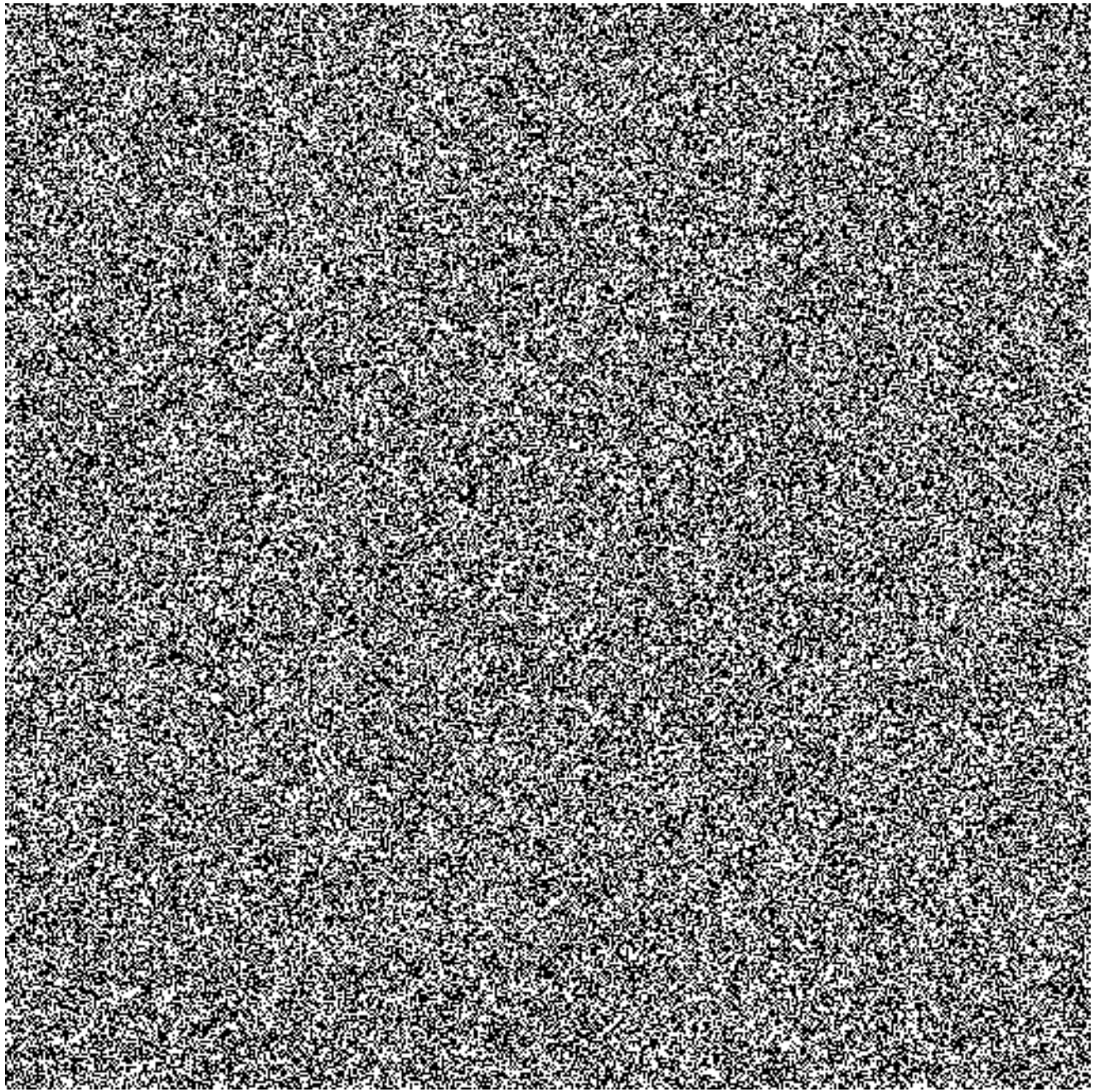
Testing?

OK, back from that distraction...

Just because a service advertises random bits (and they have a good story to go along with it) doesn't mean that it works -- To get a little technical, **we probably shouldn't think about a random number in isolation (is 1 random?) but instead talk about a sequence of random numbers**

Even this is a little vague -- What properties would we expect from a sequence of random bits (random coin tosses)? Intuitively, **what do you expect to see as you look across and down the web page on the previous slide?**

On the next slide we mapped each 1 to a black pixel and each 0 to a white one -- We then asked for 512*512 = 262,144 random bits from `random.org` and displayed them as a 512 by 512 image (putting the first bit in the upper lefthand corner, the second bit just below it and so on, filling up each column one at a time from left to right)



Testing?

Formally, there are a set of **classical statistical tests (yes, tests of hypothesis!) that could help us assess if a random number generator (true or otherwise) is performing as expected**

In this case, we can use the mathematics of probability to determine the null distribution for test statistics like **the fraction of 1's in the sequence (it should be about a half) or the length of "runs" of bits of the same kind (we shouldn't see long runs of 1s or 0s)**

These mathematical results help us avoid **a chicken and egg problem** -- If we needed simulation to test a null hypothesis, how could we ever test the simulator?!

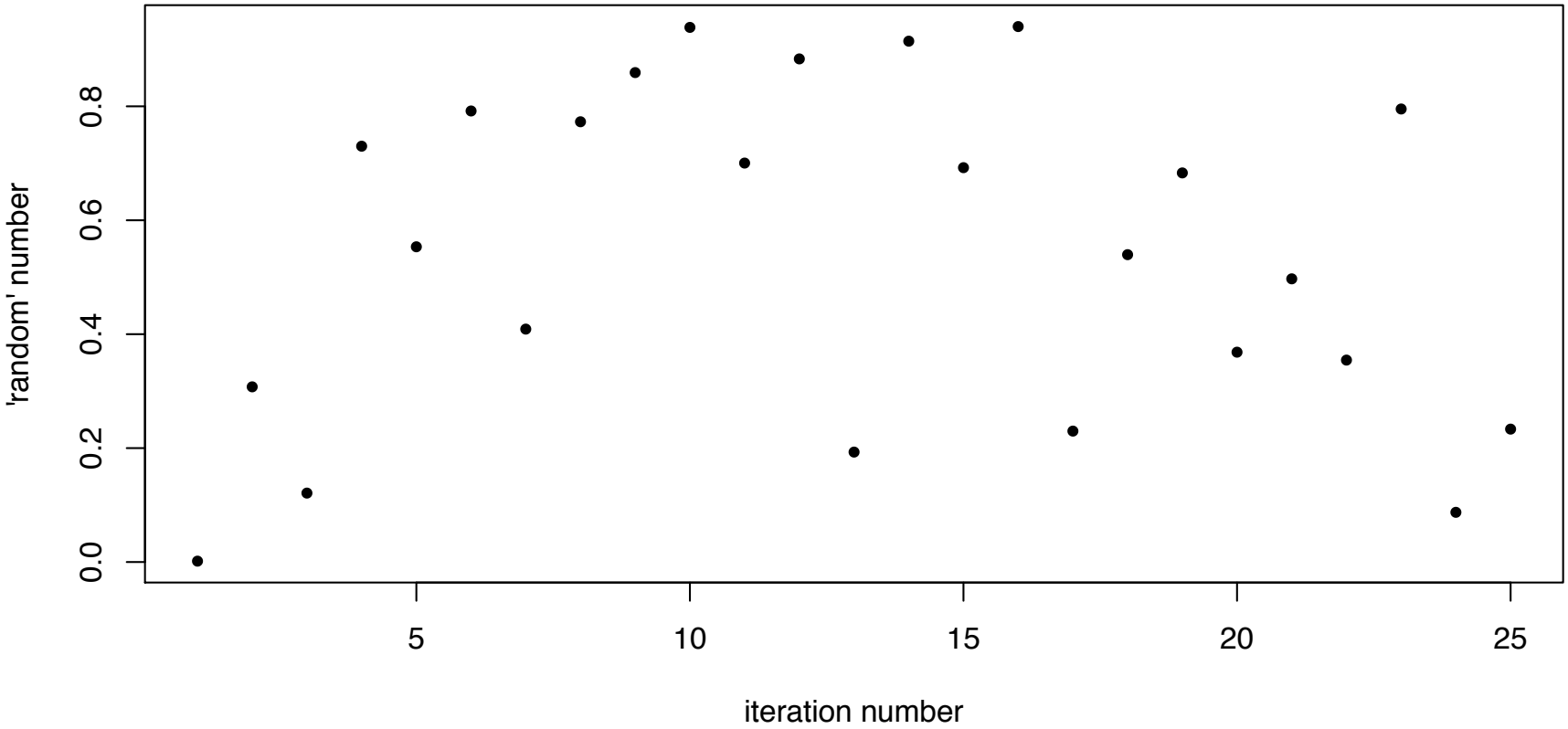
Random number generation

The second kind of random number generation comes from **a mathematical formula, a deterministic algorithm** that produces a repeatable, predictable series of numbers -- These are called **pseudo-random numbers**

Here is a snippet of code that implements a “classical” example of one of these algorithms -- We start by setting the variable `seed` to some number we choose (here I picked 200)

```
# initialize

> seed <- 200

# we then update the seed and generate
# a “random” number

> a <- 16807
> m <- 2147483647
> seed <- (a*seed)%%m
> random <- seed/m

# the values random will be in the
# interval (0,1)
```

```
# iteration 1
> seed <- 200
> (a*seed)
[1] 3361400
> (a*seed)%%m
[1] 3361400
> ((a*seed)%%m)/m
[1] 0.0015652738518851122

# iteration 2
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 56495049800
> (a*seed)%%m
[1] 660474978
> ((a*seed)%%m)/m
[1] 0.3075576286332484

# iteration 3
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 11100602955246
> (a*seed)%%m
[1] 259983903
> ((a*seed)%%m)/m
[1] 0.1210644390066454

# iteration 4
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 4369549457721
> (a*seed)%%m
[1] 1567719723
> ((a*seed)%%m)/m
[1] 0.7300263846898575

# iteration 5
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 26348665384461
> (a*seed)%%m
[1] 1188519418
> ((a*seed)%%m)/m
[1] 0.553447482433844

# iteration 6
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 19975445858326
> (a*seed)%%m
[1] 1700457579
> ((a*seed)%%m)/m
[1] 0.7918372656180697

# iteration 7
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 28579590530253
> (a*seed)%%m
[1] 878155977
> ((a*seed)%%m)/m
[1] 0.408923242897225

# iteration 8
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 14759167505439
> (a*seed)%%m
[1] 1659883255
> ((a*seed)%%m)/m
[1] 0.77294337366379

# iteration 9
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 27897657866785
> (a*seed)%%m
[1] 1845292255
> ((a*seed)%%m)/m
[1] 0.859281167322435

# iteration 10
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 31013826929785
> (a*seed)%%m
[1] 2015583458
> ((a*seed)%%m)/m
[1] 0.938579188165525

# iteration 11
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 33875911178606
> (a*seed)%%m
[1] 1504130828
> ((a*seed)%%m)/m
[1] 0.700415497971892

# iteration 12
> seed <- ((a*seed)%%m)
> (a*seed)
[1] 25279926826196
> (a*seed)%%m
[1] 1896817359
> ((a*seed)%%m)/m
[1] 0.883274413590913
```


'random' number
0.0
0.2
0.4
0.6
0.8
5
10
15
20
25
iteration number

Uniform random numbers

This procedure is also known by the mouthful of a name **“prime modulus multiplicative linear congruential generator”** (and often shortened to the equally difficult PMMLCG)

Technically the algorithm leaves the constants (`a` and `m`) unspecified, but our choice can be shown to have good properties relative to the statistical tests I alluded to

By construction, the numbers we highlighted in red are all between 0 and 1 -- They can be used anywhere we might need observations from the so-called **uniform distribution** on the interval [0,1]

Uniform random numbers

As its name suggests, we expect to see observations from the uniform distribution distributed, well, uniformly over [0,1] -- To be a little more precise, if we have a sample of 1200 such observations, we'd expect about 600 to be less than 0.5

Going farther, if we divided [0,1] into four equally sized subintervals (from 0 to 0.25, 0.25 to 0.5, 0.5 to 0.75 and 0.75 to 1) we would expect to see 300 observations of the 1200 in each bin (or so)

In general, under the uniform distribution, we expect **the proportion of our sample that falls in some subinterval we specify to be equal to the length of that interval** (the uniform distribution is a mathematical construction that we will examine more closely when we discuss probability in a future lecture)

So, using the algorithm two slides back, let's create some random bits -- We'll generate 512*512 = 262,144 numbers in [0,1] using this algorithm and **coloring a square black if the associated number is larger than 0.5 and color it white if it is less than or equal to 0.5**...

Uniform random numbers

Here is the sequence of seed values we get are

```
 [1]    3361400  660474978  259983903 1567719723 1188519418 1700457579
 [7]  878155977 1659883255 1845292255 2015583458 1504130828 1896817359
[13]  414612998 1963706518 1486760930 2018717665  493656702 1158862153
[19] 1467010828  791234989 1067717899  761374161 1707955101  187473058
[25]  501175657
```

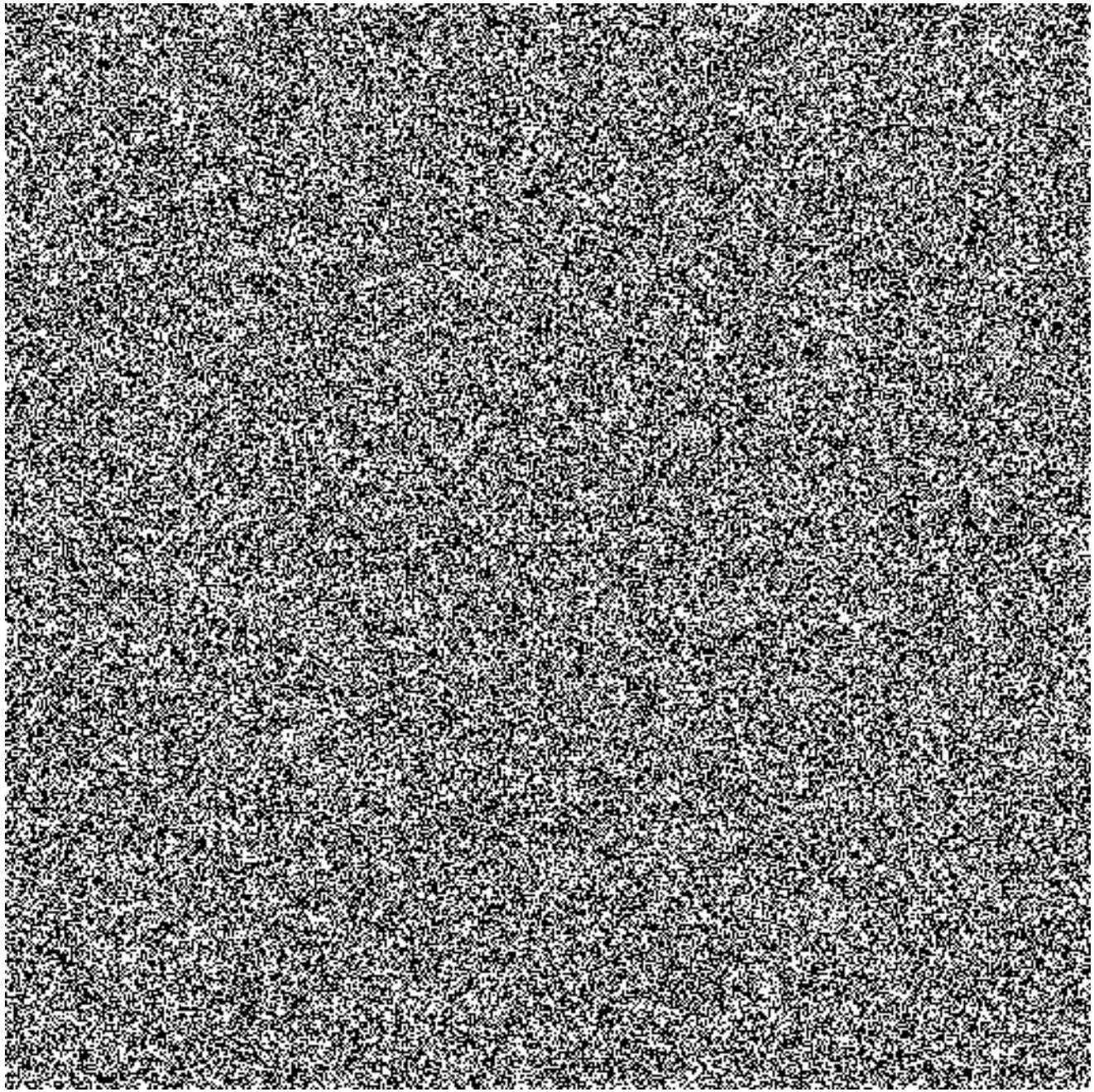
which when divided by `2147483647 = 2^31-1` gives the pseudo-random uniform observations

```
 [1] 0.001565273851885122 0.307557628633248425 0.121064439006645430
 [4] 0.730026384689857477 0.553447482433844118 0.791837265618069663
 [7] 0.408923242897225203 0.772943373663790179 0.859281167322434980
[10] 0.938579188165524547 0.700415497971892176 0.883274413590912855
[13] 0.193069222473105984 0.914422105492289194 0.692327008905041508
[16] 0.940038667032513153 0.229876815448457755 0.539637242229486502
[19] 0.683130150978979778 0.368447503712236668 0.497194891561379138
[22] 0.354542472099206640 0.795328571365833570 0.087298945564450212
[25] 0.233378101714597136
```

and the random bits

```
 [1] 0 0 0 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 1 0 0 0 1 0 0
```

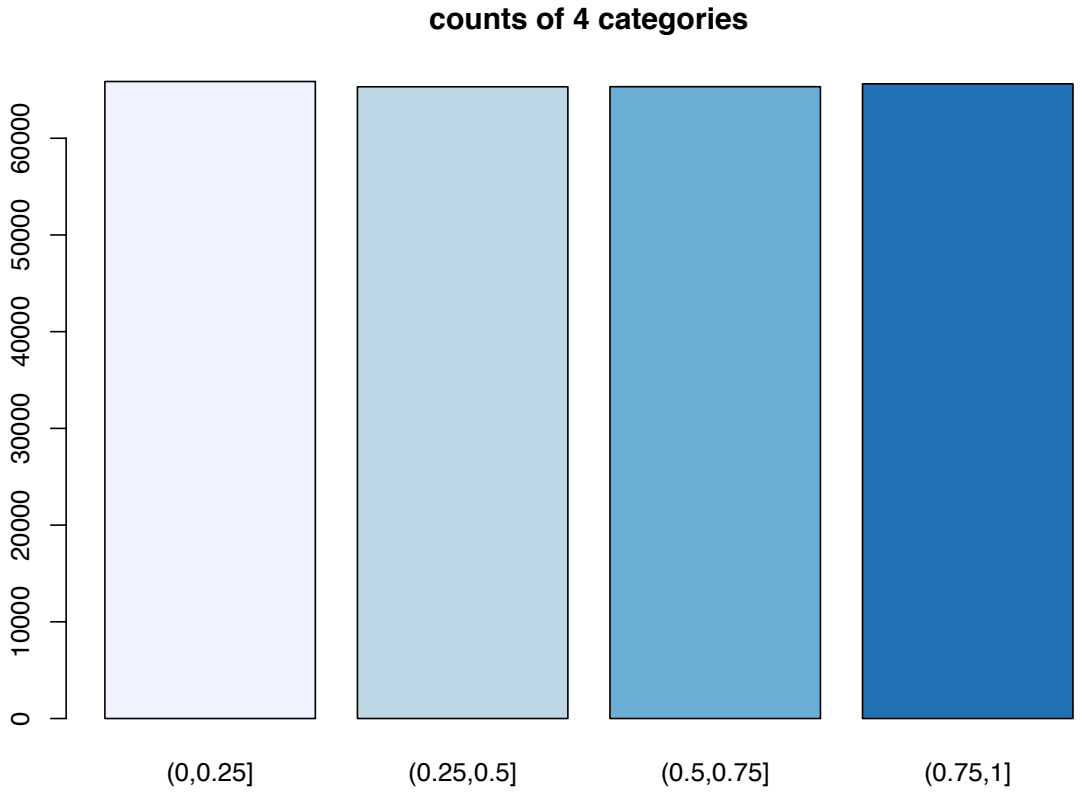
which, when continued for the full 512*512 values and arranged in an image gives us...

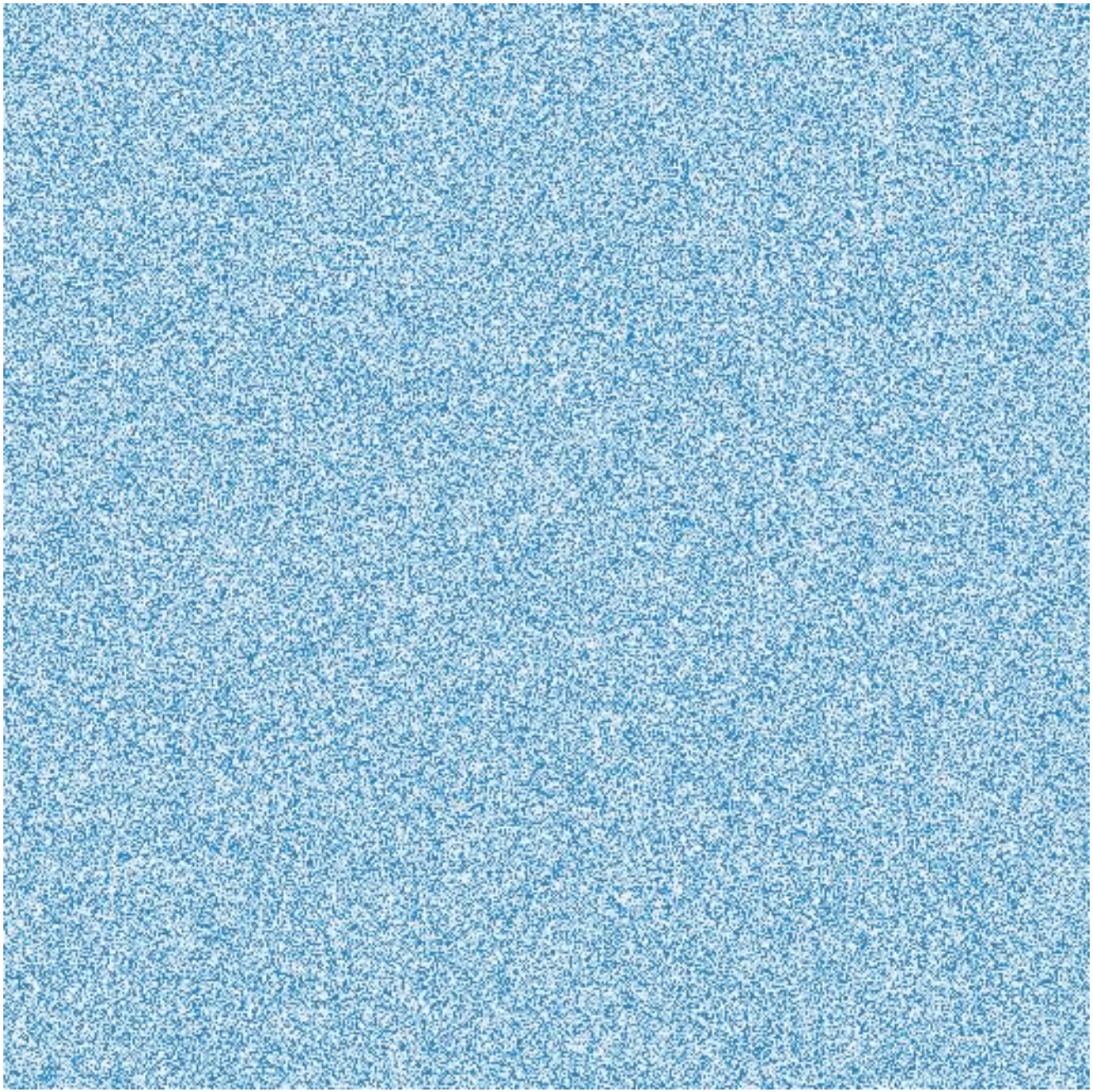


Uniform random numbers

The proportion of 1's constructed this way is 0.49955 -- We can take this farther and **divide the data into four pieces** (those between 0 and 0.25, between 0.25 and 0.5, between 0.5 and 0.75 and then larger than 0.75) and below we have a barplot of counts in each interval (and yes, it looks essentially flat)

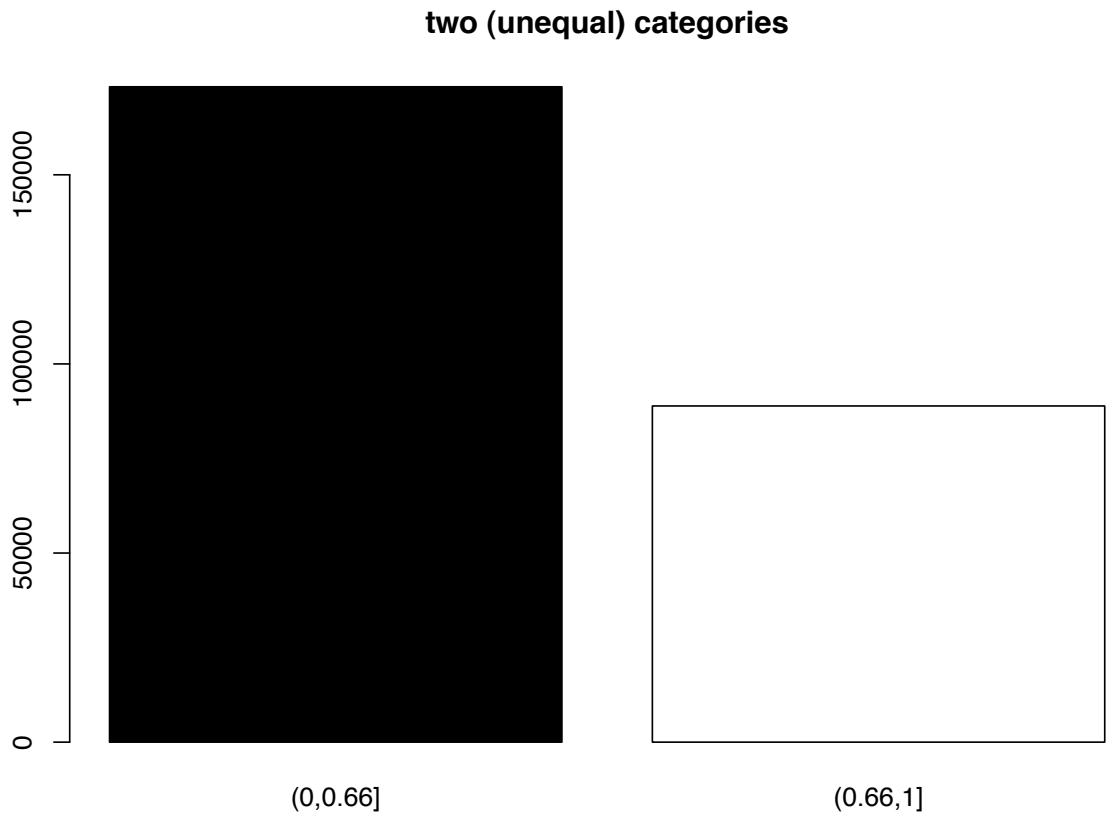
Essentially, an equal number of points falls in each interval -- We can color points falling into the four pieces using different colors (the colors of the four bars on the right) and pack them into an image again

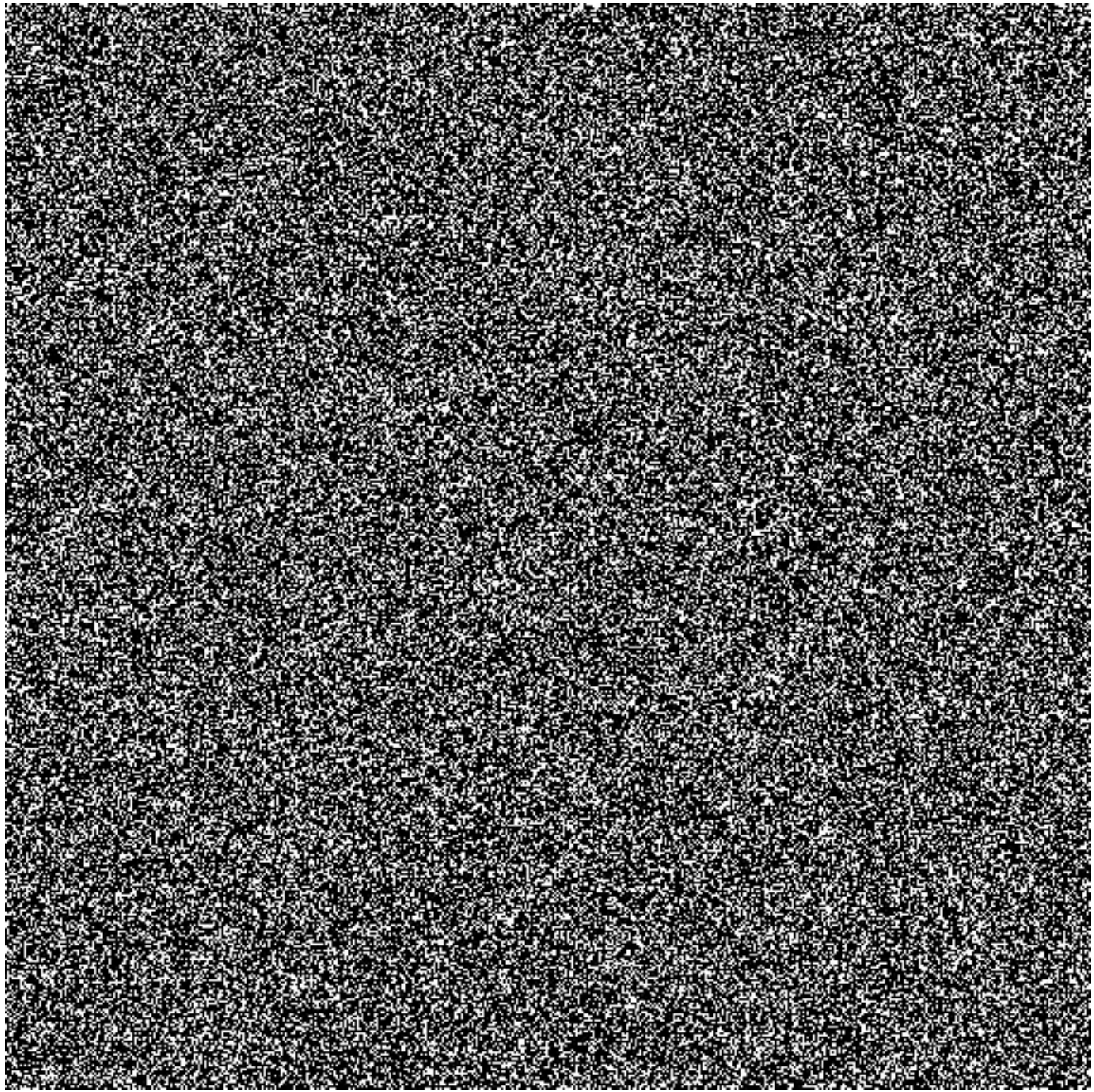




Uniform random numbers

Dividing the data using the intervals [0,0.66) and [0.66,1], we see essentially 2/3 of the data fall into the first category and 1/3 in the second -- On the next slide we play the image game again





To sum

There are **deterministic mathematical algorithms that allow us to generate observations that have many of the characteristics we'd expect to see from truly random data** (where our expectations are set by statistical tests like proportions falling into intervals or "runs" of particular kinds)

Starting with the uniform distribution, we can simulate a host of random phenomena, from permuting our data as in the case of our re-randomization analysis to tossing coins (a la Arbuthnot) to making observations from the normal (bell-shaped) distribution (later)

This idea might take some getting used to, but **the use of pseudo-random numbers is common practice** and even R depends on one such algorithm (the default is the so-called Mersenne-Twister)

Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin.

John von Neumann

Some advantages

While pseudo-random numbers are entirely deterministic, there are some advantages for scientific uses

Chief among them is reproducibility! If our analysis depends on simulation (like our re-randomization procedures for inference) we would like to be able to reproduce our results exactly (this comes in hand, say, when you want to debug more complex algorithms)

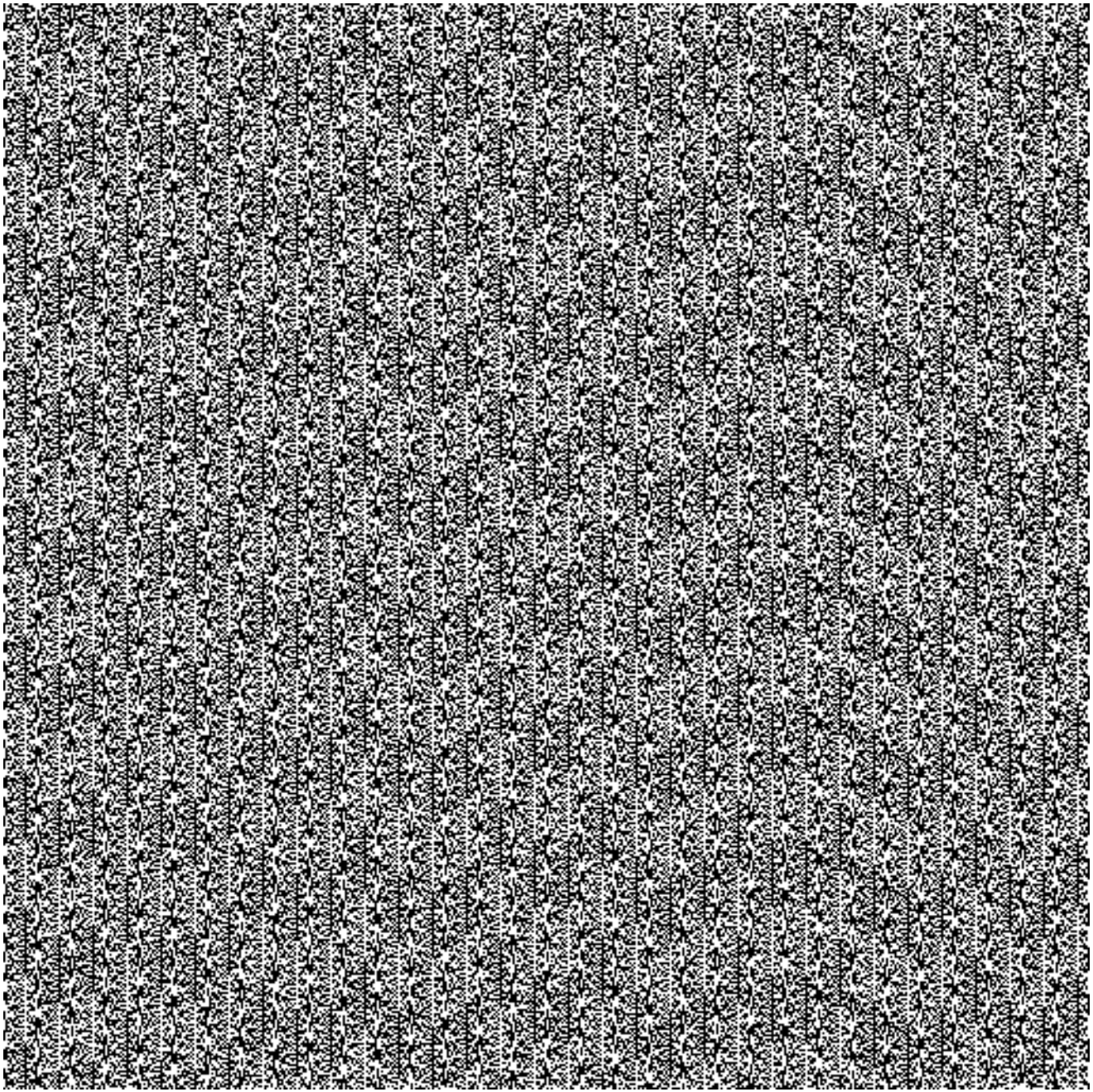
In R you can use the function `set.seed()` at any point to reset your sequence of random numbers (R does not use the algorithm described here, but it shares the properties of being a mathematical formula, deterministic and predictable)

```
> treat
 [1] "T" "T" "T" "T" "T" "C" "C" "C" "C" "C"

# treat holds a division into treatment and
# control -- let's use sample() to permute
# them or re-randomize

> set.seed(1000)
> sample(treat)
 [1] "T" "C" "T" "T" "C" "C" "T" "T" "C" "C"
> sample(treat)
 [1] "T" "C" "T" "C" "T" "T" "T" "C" "C" "C"
> sample(treat)
 [1] "T" "C" "T" "T" "T" "C" "C" "C" "C" "T"

# we can repeat our rerandomizations by
# resetting the seed

> set.seed(1000)
> sample(treat)
 [1] "T" "C" "T" "T" "C" "C" "T" "T" "C" "C"
> sample(treat)
 [1] "T" "C" "T" "C" "T" "T" "T" "C" "C" "C"
> sample(treat)
 [1] "T" "C" "T" "T" "T" "C" "C" "C" "C" "T"
```

Testing?

One final note -- **Not every service or system or program that advertises random numbers is any good!**

Here is the same bit picture for a combination of the programming language **PHP running on a Windows machine** -- Evidently this was the result of a bug that has since been corrected, but it does serve as a cautionary example

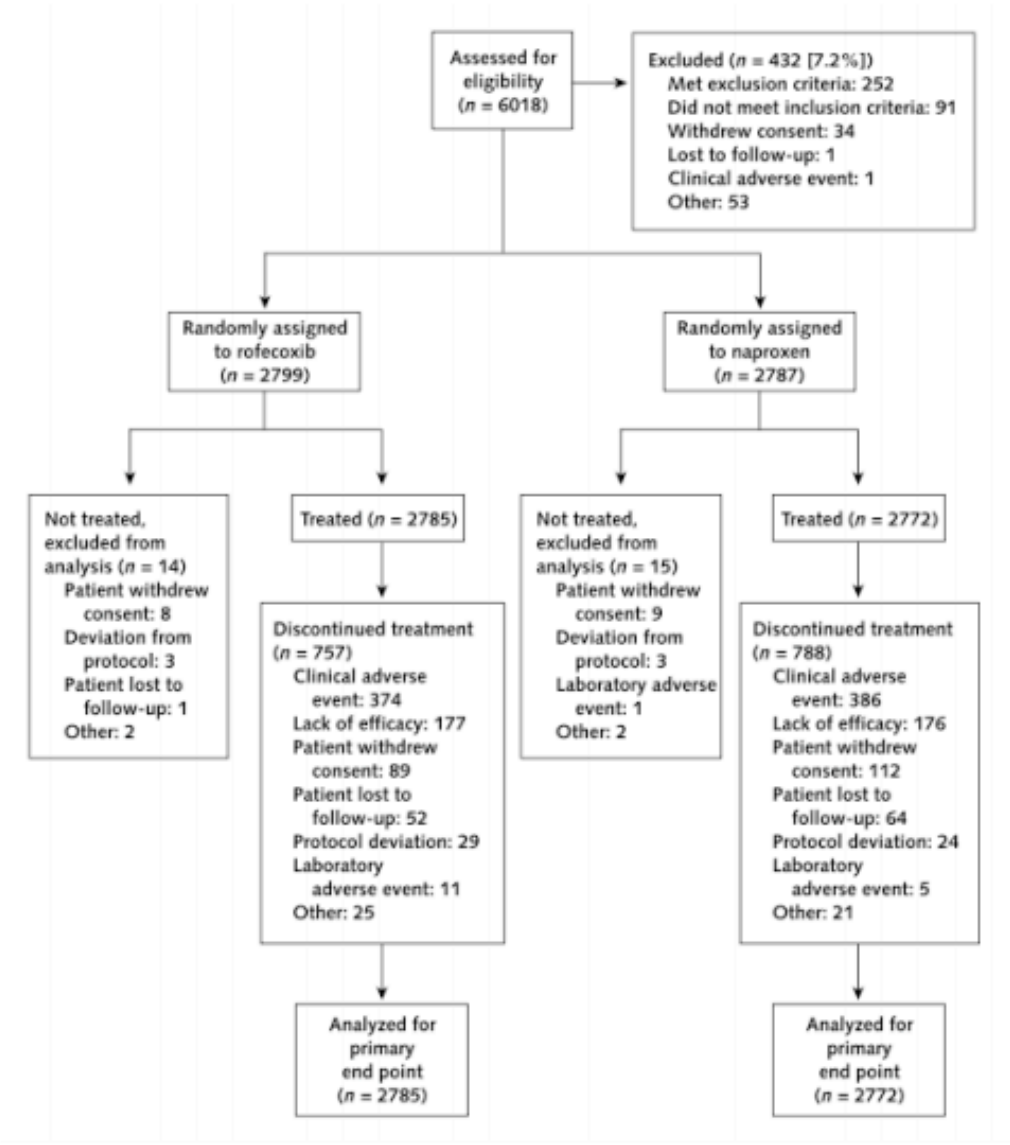


Other randomized experiments

So far, we have examined mainly experimental setups in health studies; the randomized controlled trial is a big deal and nicely highlights many of the features of hypothesis testing

But randomization in experimentation is extremely common; as we mentioned previously, Fisher was an aggressive advocate of its use in general

We'll now consider a more modern application of randomization, in experiments that "optimize" the operation of web sites

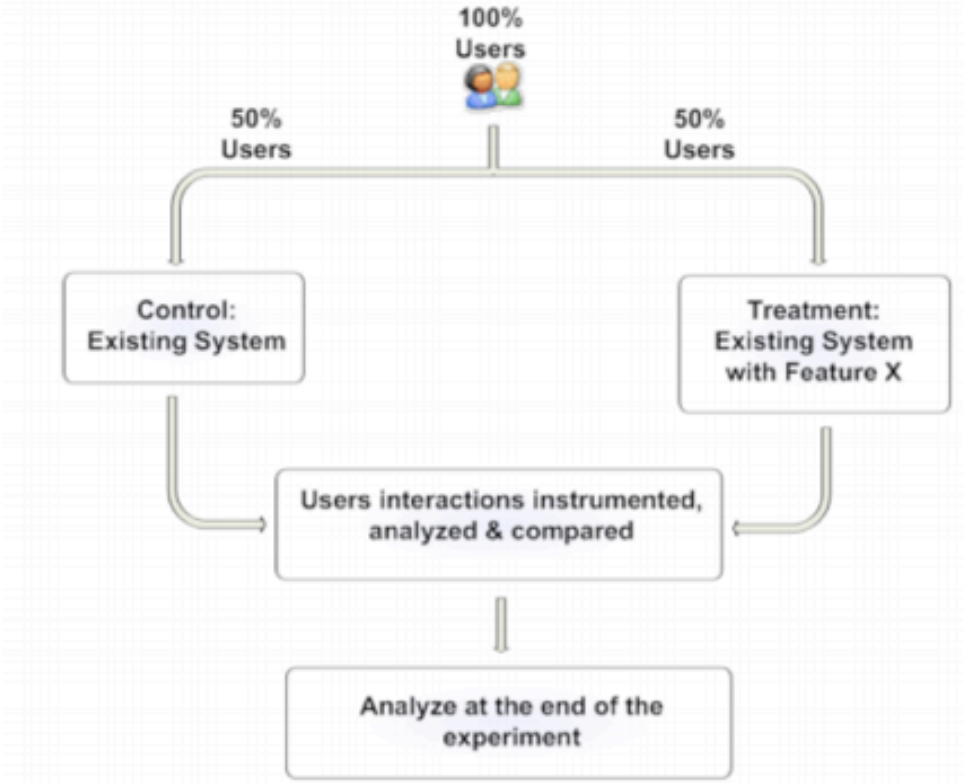


A/B testing

At the right we present a diagram for conducting an experiment on a web site; the structure is not that dissimilar from that for the ADVANTAGE trial (minus all the comments about "exclusions" and "discontinued" treatments

In an A/B test, visitors are presented with two different versions of the site (in some cases we literally have a treatment and control); sometimes the differences have to do with **the content on the page**, while in other cases they have to do with **the location of or visual arrangement of the content**

The outcomes that are measured depend on the site's overall objectives; an informational site might be interested in **the number of pages visitors read**, while commerce sites would be concerned with the **number of transactions made**

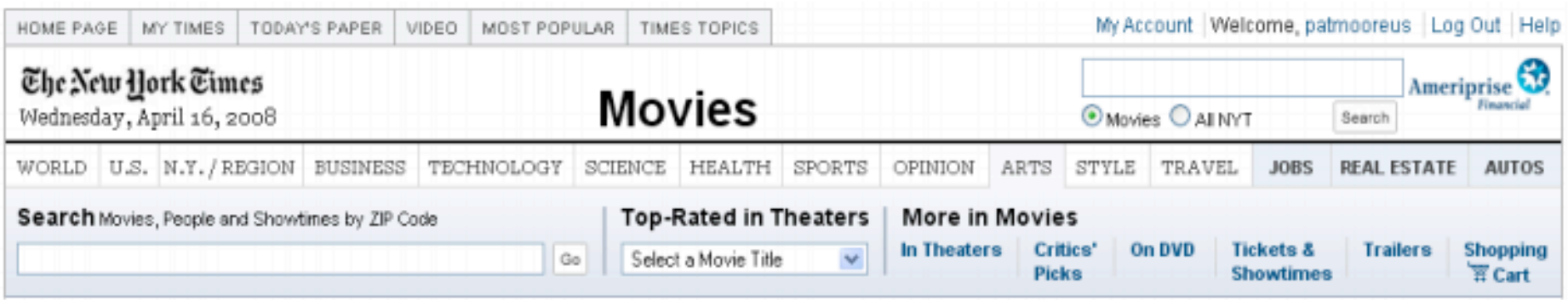


Kohavi, Henne and Sommerfield, *"Practical Guide to Controlled Experiments on the Web: Listen to Your Customers not to the HiPPO,"* KDD 2007

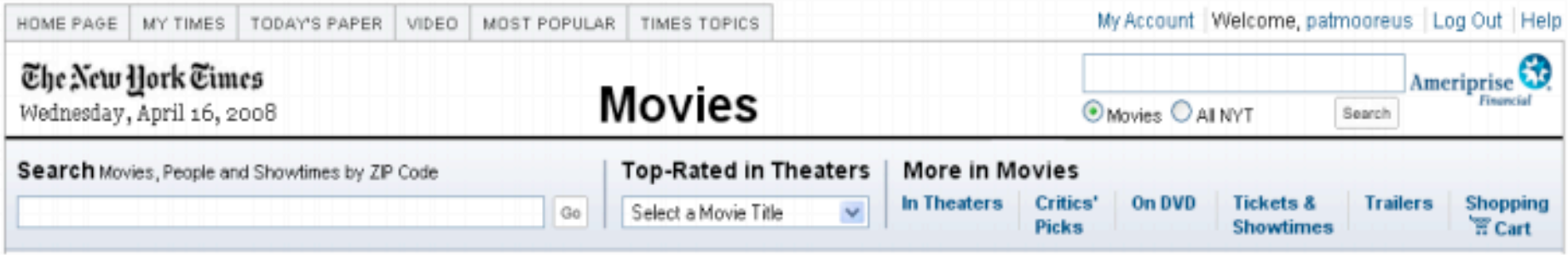
An example

Here is a simple experiment conducted by the New York Times web site (`nytimes.com`) in 2008 -- What is the difference between these two pages and what differences in visits might you be interested in comparing?

A: Control



B: Test

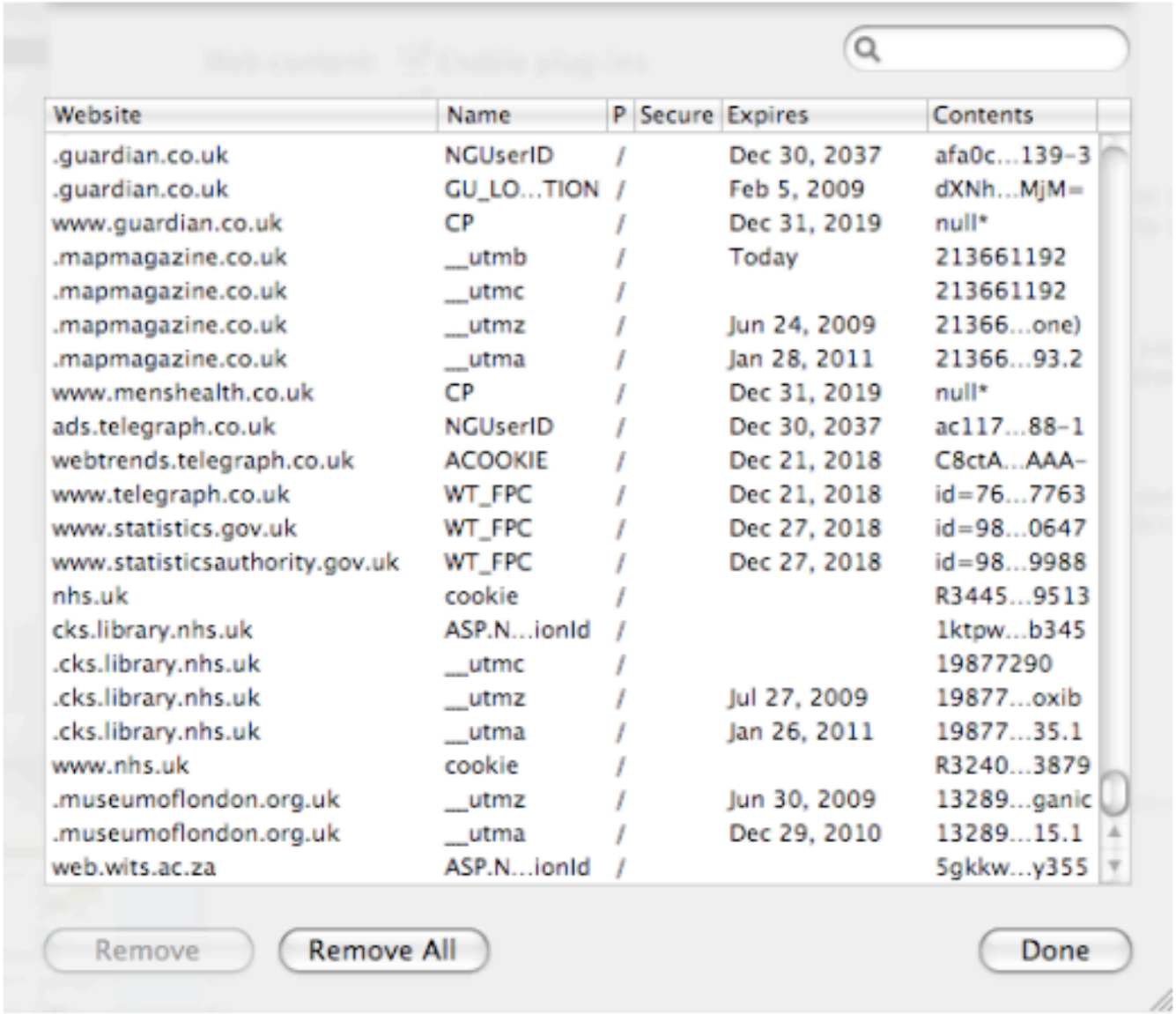


A/B testing

At a technical level (technical meaning from the standpoint of web servers), an A/B test means that visitors are assigned at random to group A or B (or C or D or E, depending on how many changes are being tested at one time)

Visitors assigned to group B, for example, will **see one version of the site consistently during the experimental period**; in the case of then Times example on the previous slide, it means that whenever a visitor in group B requests pages from The Movie Section, those pages are missing the middle navigation bar

What information does a web site need to know to make that happen? How is that information gathered?



Website	Name	P	Secure	Expires	Contents
.guardian.co.uk	NGUserID	/		Dec 30, 2037	afa0c…139-3
.guardian.co.uk	GU_LO…TION	/		Feb 5, 2009	dXNh…MjM=
www.guardian.co.uk	CP	/		Dec 31, 2019	null*
.mapmagazine.co.uk	__utmb	/		Today	213661192
.mapmagazine.co.uk	__utmc	/			213661192
.mapmagazine.co.uk	__utmz	/		Jun 24, 2009	21366…one)
.mapmagazine.co.uk	__utma	/		Jan 28, 2011	21366…93.2
www.menshealth.co.uk	CP	/		Dec 31, 2019	null*
ads.telegraph.co.uk	NGUserID	/		Dec 30, 2037	ac117…88-1
webtrends.telegraph.co.uk	ACOOKIE	/		Dec 21, 2018	C8ctA…AAA-
www.telegraph.co.uk	WT_FPC	/		Dec 21, 2018	id=76…7763
www.statistics.gov.uk	WT_FPC	/		Dec 27, 2018	id=98…0647
www.statisticsauthority.gov.uk	WT_FPC	/		Dec 27, 2018	id=98…9988
nhs.uk	cookie	/			R3445…9513
cks.library.nhs.uk	ASP.N…ionId	/			1ktpw…b345
.cks.library.nhs.uk	__utmc	/			19877290
.cks.library.nhs.uk	__utmz	/		Jul 27, 2009	19877…oxib
.cks.library.nhs.uk	__utma	/		Jan 26, 2011	19877…35.1
www.nhs.uk	cookie	/			R3240…3879
.museumoflondon.org.uk	__utmz	/		Jun 30, 2009	13289…ganic
.museumoflondon.org.uk	__utma	/		Dec 29, 2010	13289…15.1
web.wits.ac.za	ASP.N…ionId	/			5gkkw…y355

Remove | Remove All | Done

... the cookie jar

Cookies

Cookies can be set by the web sites you visit and are often used to "save state" during your session or between sessions -- This is how, for example, a site can implement a "shopping cart" or personalize a site based on your preferences or, as in this case, track your activities on the site

In this case, the cookie is set so that it can tell the site **which version of the site the visitor should see**

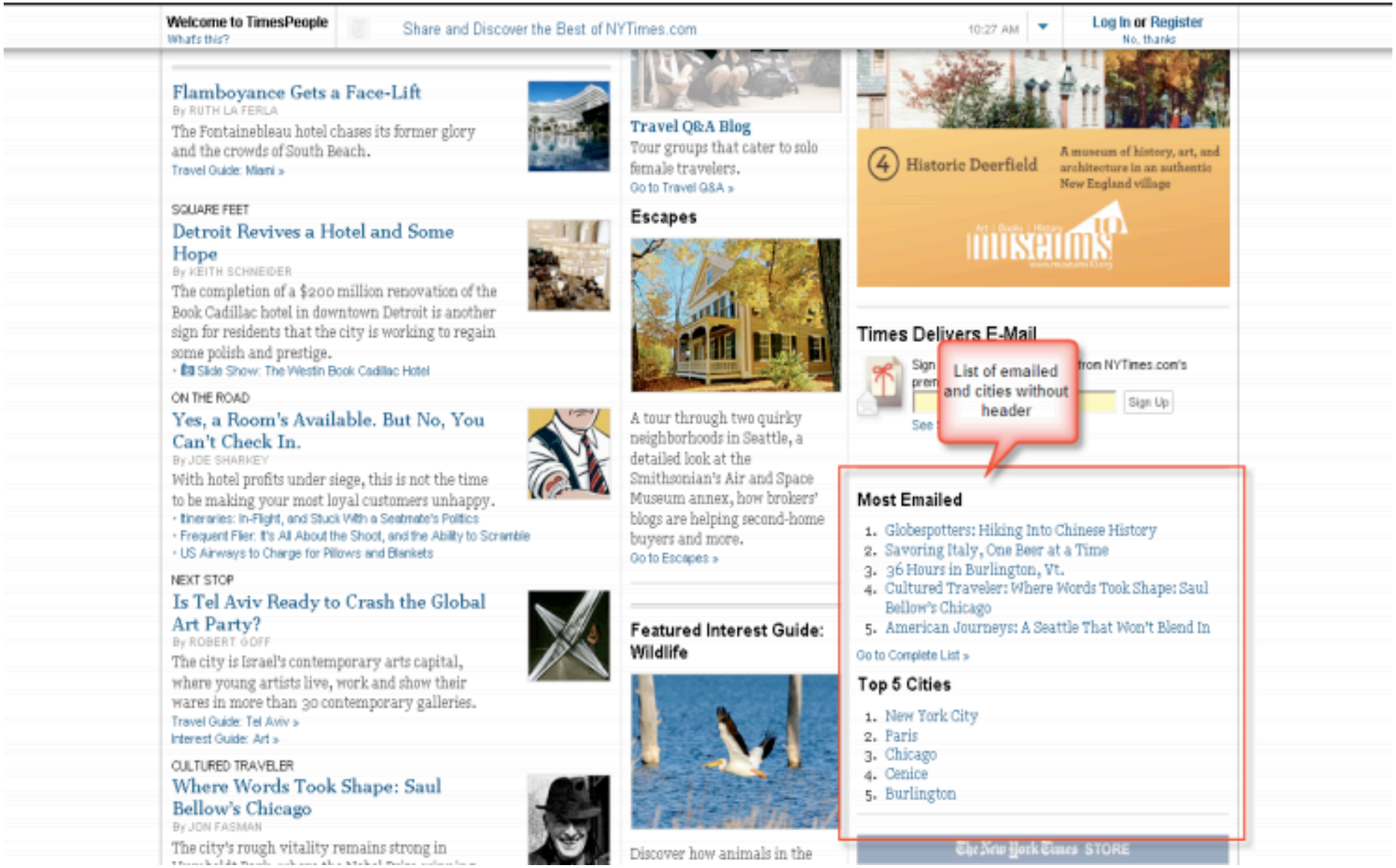
An experiment at `nytimes.com`

We will now consider a more recent example of an A/B test for **The Travel Section** of `nytimes.com` (we'll save the movie test for lab or your midterm or...)

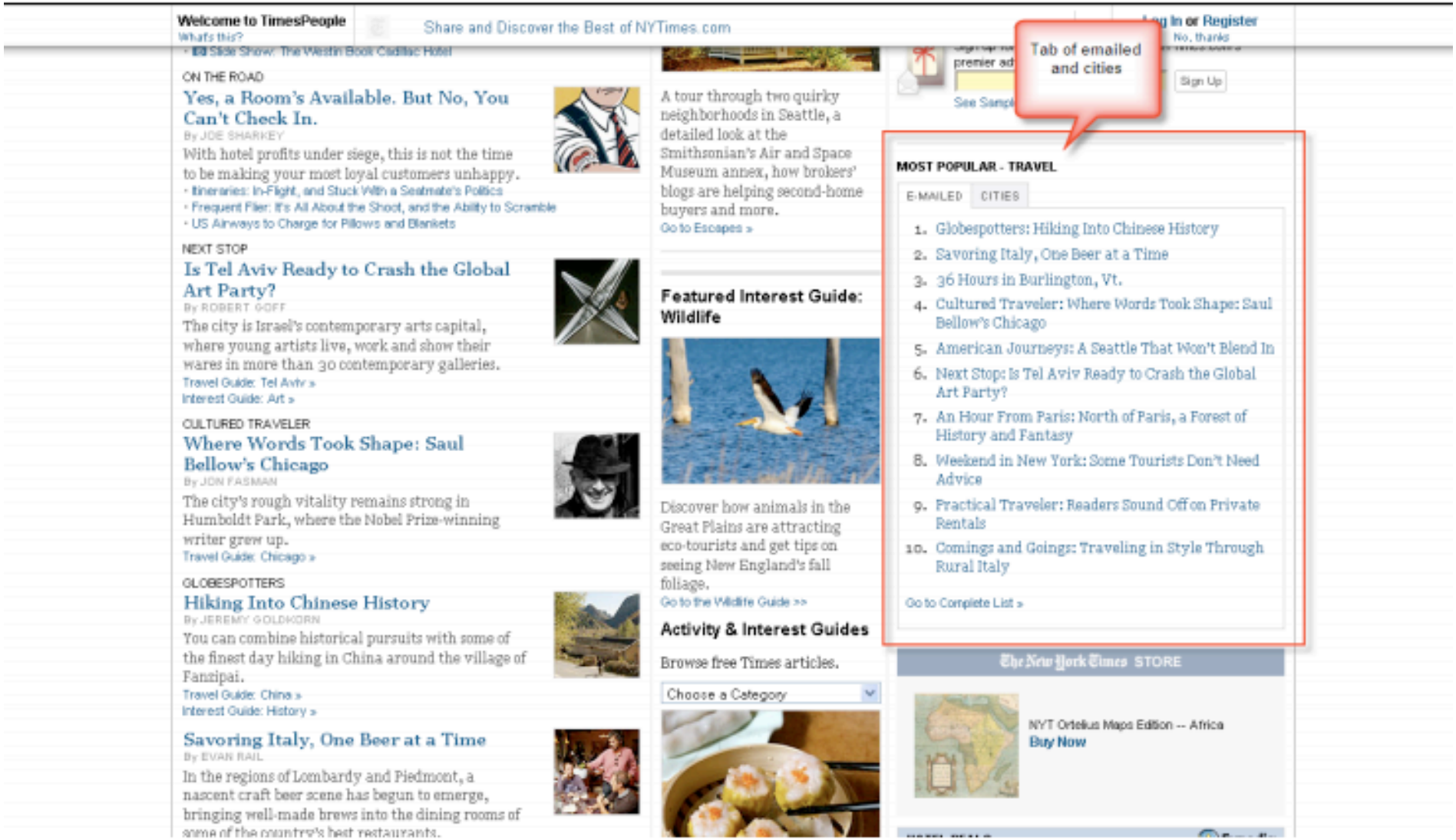
On the next two slides, we present samples of the A and B pages; the changes applied to all pages in The Travel Section, so **as a visitor browsed the site, they would consistently see either A or B**

Have a look at the two designs -- What differences do you see in terms of layout and content? What questions might the Times ask about how visitors react to these two options?

List: Variation 10858



Tabs: Variation 10859



An experiment at nytimes.com

The unit of observation during this study is a visit -- Imagine you are browsing the web and you land on `nytimes.com`, you browse a bit and then you leave to explore another site, maybe `wsj.com`

Your time at `nytimes.com` is referred to as a visit -- **The variables associated with a visit summarize your activities** and include when you arrived and how long you stayed

An experiment at `nytimes.com`

The Travel Section experiment lasted **five weeks**; the first visitor during the experiment arrived on Tuesday November 18 of 2008 at 8:40 am PST 2008, and the experiment closed with a final visit on Monday December 29 at about 8 pm PST

During that period, **nearly 200K visits were recorded as part of the experiment** (the experiment involved just a fraction of their overall traffic)

Given the nature of the change between A and B, and some of the objectives you wanted to examine, what sort of information would we like to collect about each visit? About each visitor?

The variables

Recall from last time the variables we have at our disposal; in all data were collected from about 130K users over a period of 6 weeks at the end of 2008

`User_ID` A unique number for each visitor

`UserVisit_ID` A unique number for each visit

`StartTime_SSE` Unix time for the start of the visit

`StartTime_English` A more human readable version of the time

`VisitLength` The number of seconds the visitor was reading Travel Section pages

`Variation` The version of the page they received

`RefererUrl` The page they clicked on (if any) to get to the page

`EntryPageUrl` The first page on `nytimes.com` they visited

`Pageviews` The number of page views to in the Travel Section

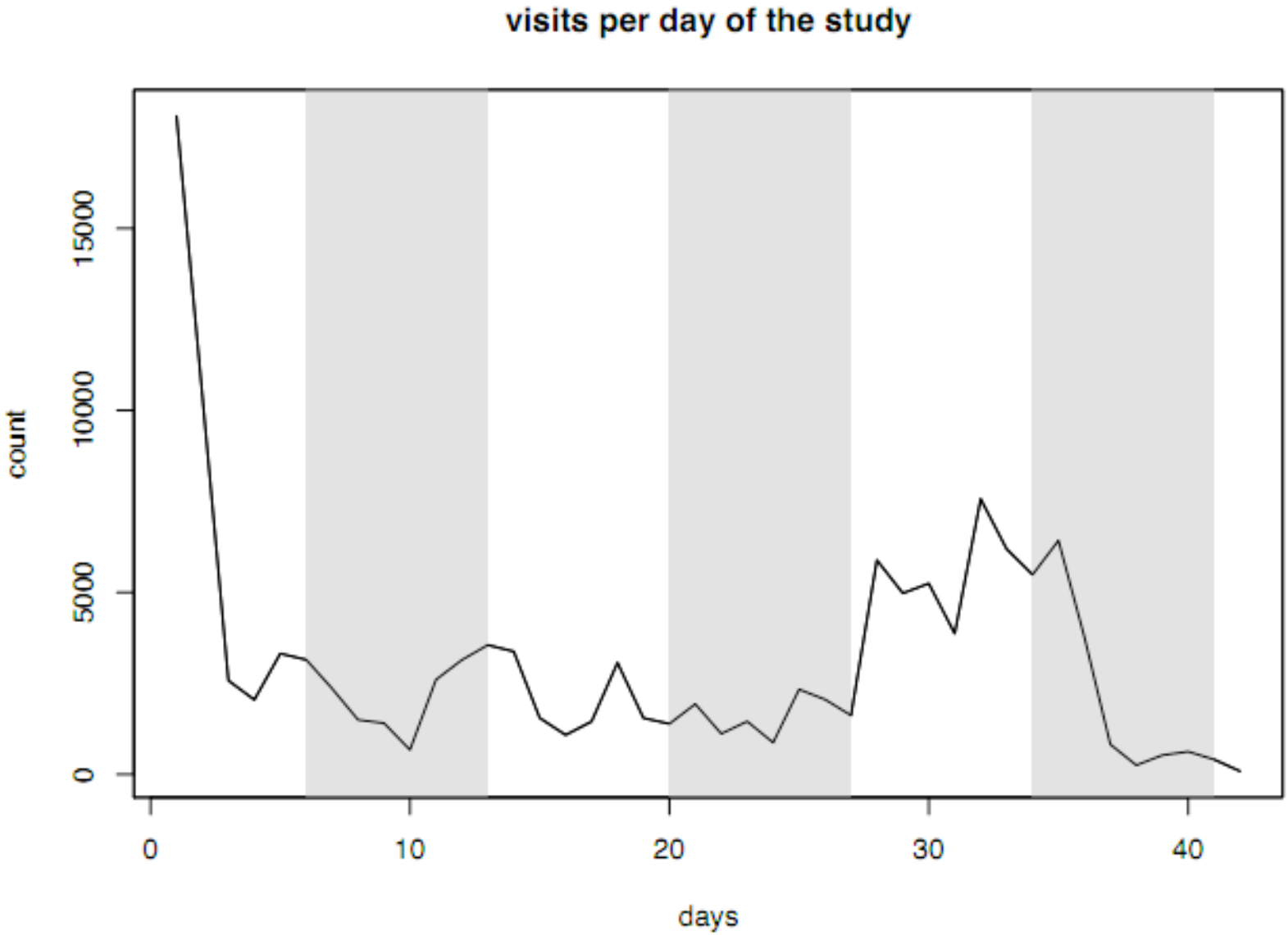
`TotalVisits` The total number of visits to the site

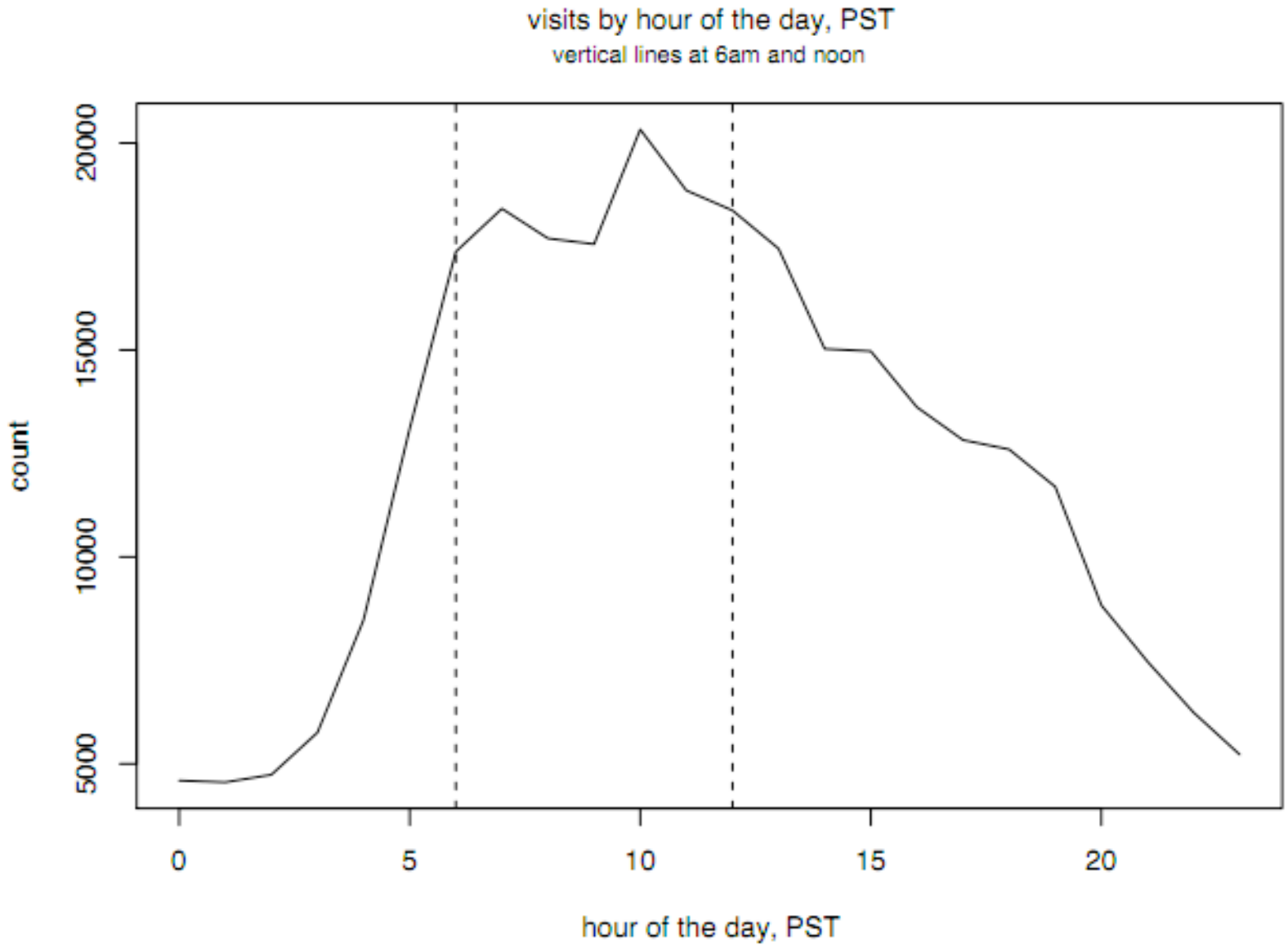
`TimeSinceFirstVisit` `(days)` How long it had been since their last visit

`UserAgent` Their browser

`TotalClicks` How many times did they click on the "most popular" field

`IfClicked` 0/1 did they click on the "most popular field" at least once


visits per day of the study
count
0
5000
10000
15000
0
10
20
30
40
days


visits by hour of the day, PST
vertical lines at 6am and noon
count
5000
10000
15000
20000
0
5
10
15
20
hour of the day, PST

Two entries from the data set

Here we have two visits, one of the first during the experiment (beginning at 8:40 am on November 18, PST) and one toward the end (starting at 6:15 am on December 23, PST)

How did they get to the site?

How long did they stay?

How many pages did they view?

Did they select anything from the “most popular” list?

* in case you were wondering... Google

User_ID 304431020
UserVisit_ID 374523135
StartTime_SSE 1227026436
StartTime_English Tue, Nov 18, 2008 - 16:40:36 (GMT)
VisitLength 23
Variation Tabs
RefererUrl http://www.google.com.jm/search?hl=en&q=clothes+in+paris
EntryPageUrl http://travel.nytimes.com/2006/05/21/travel/21forage.html
Pageviews 1
TotalVisits 1
TimeSinceFirstVisit (days) 40.99
UserAgent Mozilla/5.0 (Windows;U;Windows NT 5.0;en-US;rv:1.9.0.1) Firefox/3.0.1
TotalClicks 0
IfClicked 0

User_ID 248002503
UserVisit_ID 387427203
StartTime_SSE 1230041601
StartTime_English Tue, Dec 23, 2008 - 14:13:21 (GMT)
VisitLength 1297
Variation List
RefererUrl http://www.google.ro/search?hl=ro&q=Milan+Central+Station+grocery+stores
EntryPageUrl http://travel.nytimes.com/2007/06/17/travel/17hours.html
Pageviews 76
TotalVisits 4
TimeSinceFirstVisit (days) 136.31
UserAgent Mozilla/4.0 (compatible;MSIE 7.0;Windows NT 5.1;.NET CLR 1.1.4322;...
TotalClicks 1
IfClicked 1

"http://www.google.com/search?q=**tea+india**&sourceid=navclient-ff&ie=UTF-8&rlz=1B3GGGL_enUS242US

"http://www.google.com/search?hl=en&q=**fusion+menu**"

"http://www.google.com/search?q=**frequent+flier+mile+tracker**&ie=utf-8&oe=utf-8&aq=t&rls=org.moz

"http://www.google.com/search?q=**taos+ski+valley**&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:en-US:o

"http://www.google.com/search?hl=en&q=**Coco+Plum+Caba%C3%B1as+panama**"

"http://www.google.co.uk/search?hl=en&rlz=1T4ADBF_en-GBGB261GB261&q=d**rug+runners**&meta="

"http://www.google.ie/search?hl=en&q=**argan+oil+psoriasis**&start=30&sa=N"

"http://www.google.co.uk/search?hl=en&sa=X&oi=spell&resnum=0&ct=result&cd=1&q=**Port+of+Spain+tr**

"http://www.google.com/search?sourceid=navclient&ie=UTF-8&rls=GGLD,GGLD:2005-15,GGLD:en&**q=new+**

"http://www.google.com/search?q=**quaintest+towns+in+the+united+states**&btnG=Search&hl=en&sa=2"

"http://www.google.com/search?hl=en&q=**st.gotthard+pass+weather+december**&aq=f&oq="

"http://www.google.com/search?q=**seeko%27o+hotel+bordeaux**&rls=com.microsoft:*&ie=UTF-8&oe=UTF-8

"http://www.google.ca/search?q=**bed+and+breakfast+mont+tremblant+dog+sledding+packages**&start=20

"http://www.google.com/search?hl=en&q=**PALMYRA+NY**&btnG=Search"

"http://www.google.com/news?hl=en&tab=wn&nolr=1&q=**santa+barbara+fire+map**&btnG=Search"

"http://www.google.com/search?hl=en&q=**36+hours+in+milwaukee**&aq=f&oq="

"http://www.google.com/search?q=**new+york+city**&rls=com.microsoft:en-us:IE-Address&ie=UTF-8&oe=U

"http://www.google.com/ig/gmailmax?hl=en&mid=23"

"http://www.google.com/search?hl=en&rls=com.microsoft:en-us:IE-SearchBox&rlz=1I7RNWE&q=**outward**

"http://www.google.com/search?hl=it&lr=&client=firefox-a&channel=s&rls=org.mozilla:it:official

"http://www.google.co.in/search?hl=en&q=**sightseeing+around+england**&start=10&sa=N"

"http://news.google.com/news?q=**santa+fe**,+NM&rls=com.microsoft:en-us:IE-SearchBox&ie=UTF-8&oe=U

"http://www.google.com/search?hl=en&q=**week+end+in+new+york+last+minute**&aq=f&oq="

north america highest peaks climbing
biggest ship
lake little big wolf new york
hot springs in the northern california
twin tip skiis youth
black people and boston
biltmore neighborhood phoenix
new york times berlin
vermont ny
killington or okemo
compiegne june 22 photo
new york city tour
trips to mekong delta
how to curl reed for basket weaving
bicycle night ride around central park
small paragraph about marrakech
careyes mexico
best historic sites in the u.s
berlin rent a glass of wine
restauramts in italy
brasilia red light district
hyatt layoffs
travel time nyc providence
amsterdam
indian restaurant in costa rica
arizona desert survival courses
bolivian death road replacement
ny times prague
where to eat in new york near public library
introduction to the city of san cristobal
remote cabins ny
36 hours in buenos aires
pizza dough mario
bull wrestling
holiday markets new york city
ski resorts in new york
jordan travel guide
strip clubs in dubai
verbier switzerland
colonial days homes and buildings in new york
cinque terra italy
the chesterfield inn
nytimes mit museum
blood mountain loop trail
rock climbing in red rock canyon
charles de gaulle airport day room
star boys skansen
restaurants in paris
puebla mexico
huatulco cultures
nytimes travel
washington d c time
toronto new york flights
ho chi minh trail
outward bound children
ski underwear
sightseeing around england
hiking in napa valley
santa fe, nm
week end in new york last minute
2008 galapagos new york times
new york times san diego 36 hours
1 week on hawaii - what to see
santo domingo
argan oil
champlain map route
selling my chula vista condotel
new york city
currency exchange locations
state farm rental car insurance
donald m kendall sculpture gardens
toulouse art
best restaruants mexico city
ski resort near new york city

total: 35,936 words

cnt	word
1333	new
1174	york
1152	in
721	times
701	to
563	travel
355	best
352	the
309	ny
304	hotel
272	of
248	ski
248	city
226	restaurants
205	hours
193	36
190	nytimes
174	and
170	paris
166	restaurant
161	nyc
158	for
147	mexico
139	exchange
138	san
134	where
129	go
125	guide
124	from
122	italy
118	on
108	currency
107	de
105	hotels
103	winter
102	aires
101	buenos
95	a
92	gordon
89	oil
89	london
88	park
83	cinque
82	nyt
81	time
78	santa
77	resort
75	map
75	cartagena
74	india
72	skiing
72	how
71	terre
71	sightseeing
70	la
66	beach
65	what
65	reviews
64	argan
62	cruise
60	review
60	mountain
60	food
60	dubai
60	barcelona
59	places
59	do
59	at
58	stay
58	art
57	tour
57	is
56	place
55	shanghai
55	christmas
54	day
53	rental
53	house
53	colombia
52	street
52	resorts
52	france
51	ramsay
51	martin
51	kate
50	valley
50	st
50	middleton
49	south
49	los
48	spain
48	seattle
48	green
48	grand
48	bar
47	rio
47	kids
47	caribbean
47	car
46	national
46	lines
45	introduction
45	fe
44	with
44	top
44	socotra
44	ramsey
44	francisco
44	country
42	town
41	trip
41	chi
40	tokyo
40	ho
40	flight
39	minh
39	england
39	cheap
39	brazil
39	boston
38	puerto
37	tours
37	taos
37	puebla
37	one
37	island
36	shopping
36	itinerary
36	america
36	2008
35	st.
35	sea
35	portland
35	miami
35	club
35	bus
35	about
34	trail
34	small
34	near
34	montreal
34	bars
33	world
33	rome
33	indian
33	europe
32	steak
32	spa
32	rv
32	real
32	panama
32	market
32	islands
32	flights
32	berlin
31	towns
31	skis
31	or
31	night
31	le
31	istanbul
31	chicago
30	west
30	tremblant
30	rico
30	nude
30	museum
30	eat
30	central
30	beer
29	weekend
29	week
29	twin
29	las
29	lake
29	juan
29	estate
29	airlines
28	when
28	tayrona
28	state
28	ranch
28	locations
28	hot
27	road
27	peru
27	moroccan
27	long
27	el
27	bangkok
27	airline
26	riviera
26	rate
26	old
26	naples
26	music
26	madrid
26	janeiro
26	california
26	by
25	years
25	wine
25	united
25	tips
25	jose
25	inn
25	french
25	children
25	big

A slight adjustment

To remove the effect of frequent, repeat visitors, the Times chooses to look at **only a user's first visit to the site during the experimental period**; this reduces our data set from about 200K to 130K

One of the first questions we can ask of these data uses tools we've developed for "dichotomous responses"; that is, results that **we can display in a two-by-two table**

Do the different designs attract reader's interest differently? Put another way, are visitors equally likely to click on the two different representations for the "most popular" lists

Re-randomizing

Formally, our null hypothesis is that there is **no difference between the two designs** and that visitors would click on either equally often -- We can take as our **test statistic the number of visitors shown the Tabs design and also clicked on its links**

The alternative here is that **the two designs are different** -- At the moment, we have no idea which will be better so we can look to **large values** (meaning Tabs was better) or **small values** (meaning the List is better) in this cell

As we have done with the clinical trials, we can re-randomize our visitors into Tabs and List and see how these tables compare to our own -- So what did the experiment show?

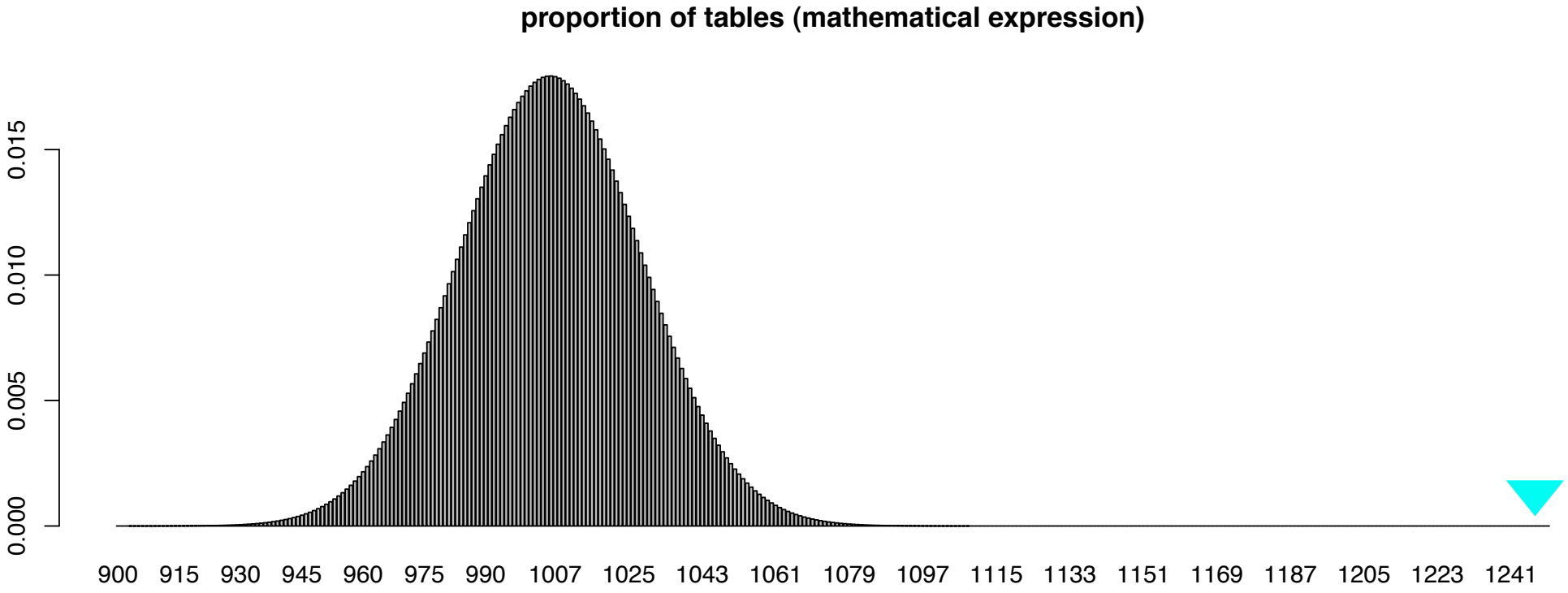
A first test

Not to belabor the two-by-two analysis below, we list a few R commands to read in the Times Travel Section data and make our (now) standard computations

What do you see?

```
> source("http://www.stat.ucla.edu/~cocteau/stat13/data/nyt.R")

> table(travel$IfClicked,travel$Variation)

     List  Tabs
  0 65315 64972
  1   766  1244

> 766/(65315+766)
[1] 0.01159183

> 1244/(1244+64972)
[1] 0.018787

> (1244/(1244+64972))/(766/(65315+766))
[1] 1.62071
```

It seems like the tabbed display is more popular (although both are fairly unused features) -- How can we make sure this wasn't just random, the result of our assignment to treatment and control, but an actual effect?


proportion of tables (mathematical expression)
0.000
0.005
0.010
0.015
900
915
930
945
960
975
990
1007
1025
1043
1061
1079
1097
1115
1133
1151
1169
1187
1205
1223
1241

A first test

Our observed test statistic is so far out in the tails that we had to appeal to a mathematical formula to draw the picture on the previous page -- The associated P-value (adding up the probability of having tables with more extreme values of the test statistic in BOTH tails) is 10^-16

In short, we can easily reject the null hypothesis that the two designs are the same in favor of the alternative hypothesis that they are different

Another test

We will look at the data in a lot more detail, but to emphasize a concept, **we'll consider a second test that the people who provided the data were interested in** -- Is there a difference between Tabs and Lists in terms of the number of Pageviews?

Let's quickly see how we can address that question...

Hypothesis testing

Before we propose anything formal, let's recall the steps...

1. We begin with **a null hypothesis**, a plausible statement (a model or scenario) which may explain some pattern in a given set of data but made for the purposes of argument; we also select a complementary alternative hypothesis

2. We then define **a test statistic**, some quantity calculated from our data that is used to evaluate how compatible the results are with those expected under the null hypothesis

3. We specify a **threshold or significance level,** α , of the test; at the end of the experiment, this threshold will be applied to determine if we can reject the null

4. We then consider **the distribution of the test statistic under the null hypothesis**; we can get at it either with some probability calculation (remember the table fun from last time) or through computer simulation

5. And finally, after the data are collected, we compute the P-value and apply the threshold α : if our P-value is less than α we reject the null, finding that the data contain evidence for the alternative; if not, we say that we cannot reject the null, and that the data do not contain sufficient evidence for the alternative

Pages per visit

To repeat, the question at hand is, is there some difference in the number of Page Views between groups A and B, between treatment and control, between Tab and Lists?

To turn this into a hypothesis testing exercise, we need a null hypothesis and an alternative; t**he null will be that there is no difference between treatment and control measured in terms of Page Views per visit**

Our **alternative will be that there is a difference,** and that one condition or the other tends to produce more Page Views per visit; notice that in the language we've been evolving over the last couple of lectures, this alternative is two-sided in the sense that we would be convinced of a difference if the Page Views turned out to be much larger for either treatment or control

In terms of significance, we will will choose the default 0.05; if this result proves to be significant, the management at the New York Times (business people) will want a traditional number like that

Now, a test statistic...

Page views per visit

We will work with the absolute value of the difference between the average Page Views per visit computed for each group; **we will use the absolute value to indicate that we are looking for a big difference in either direction**

When judging how extreme our data are, **we will consider how likely it is to see an absolute difference as big or bigger** than the one we see in our experimental data if the null distribution is true

The difference in averages is a reasonable metric for capturing a shift in one group or the other; **the average itself is something that is focused on in the "science" of web traffic, which also recommends it**

Page views per visit

With all that set, let's look at the data we collected

```
> mean((travel$Pageviews)[travel$Variation=="Tabs"])
[1] 1.997261
> mean((travel$Pageviews)[travel$Variation=="List"])
[1] 1.980060
```

The mean number of Pages viewed per visit for Lists is 1.980 while it is 1.997 for the Tabs option; the absolute difference, 0.017 is, well, tiny; and as a practical matter, it might not amount to anything important (although, as we have said, small differences multiplied over millions of visits might prove to be important

Page views per visit

Finally, we need to come up with some way to evaluate the distribution of our test statistic under the null; again, the null is that there is no difference between Tabs and Lists

If there really is no difference, then the value of 0.017 we saw is simply the result of the randomization that took place on the web server

So, if the labels really have nothing to do with the number of Page Views per visit, then **we can simulate other values of the test statistic under the null by simply reassigning visitors to treatment and control, to Tab and Lists**

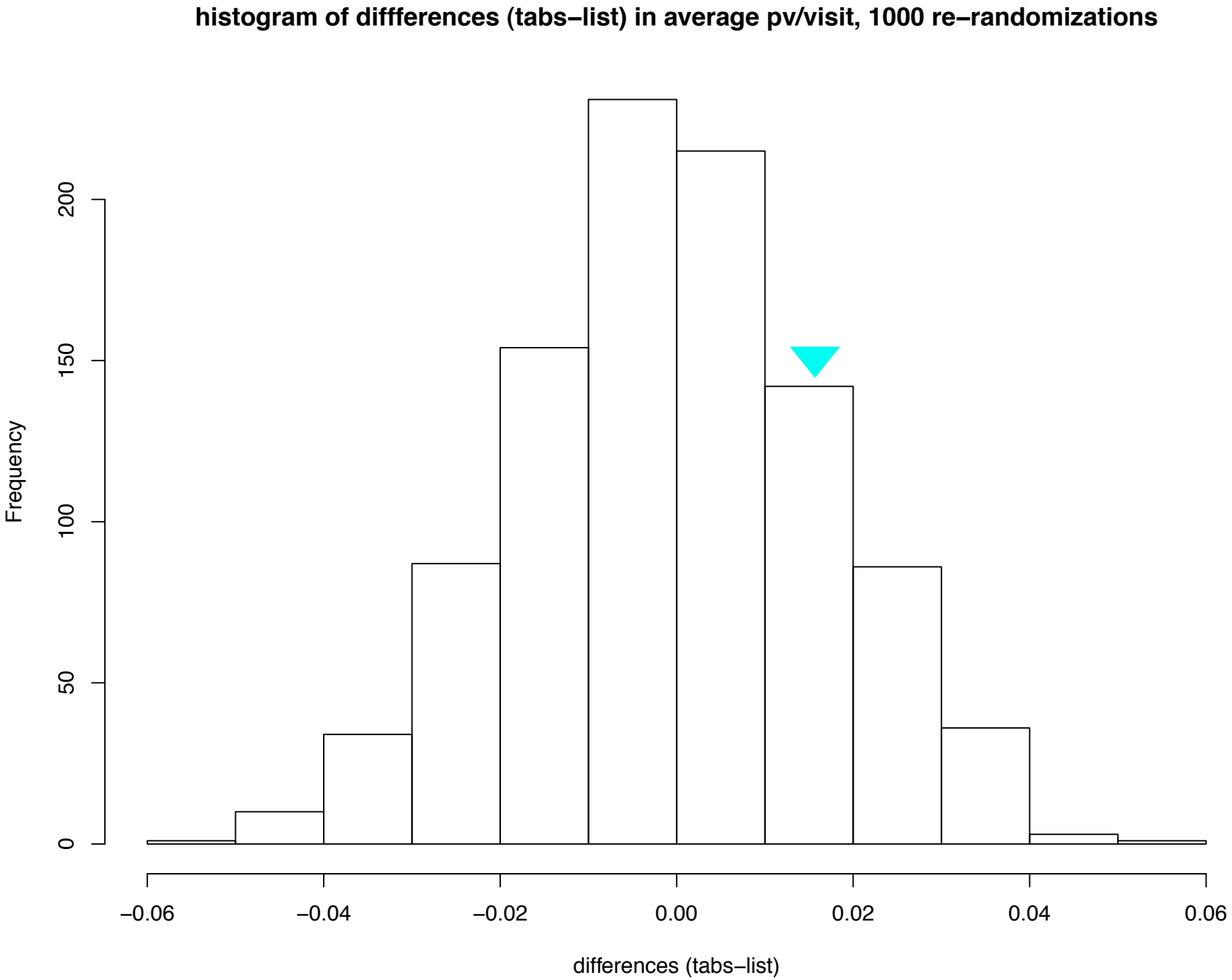
Again we re-randomize...

Page views per visit

On the next slide, we present the results of re-randomization; **here we plot the difference between the average Page Views per visit in the List group and the average Page Views per visit in the Tab group**

Our test statistic was really **the absolute value of the difference, but we present the signed value** (before the absolute value) to correspond with what we have been doing so far

Our observed value is 0.017 and for a difference in our null distribution to be more extreme, it has to have an absolute value of 0.017 or greater (meaning -0.017 or smaller and 0.017 or larger)...


histogram of diffferences (tabs−list) in average pv/visit, 1000 re−randomizations
Frequency
0
50
100
150
200
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06
differences (tabs−list)

Page views per visit

In this case, we don’t even have to be very formal about the fact that the observed difference of 0.017 is well within the null distribution, meaning any difference in mean we observed “looks” like it could be the result of our randomization process

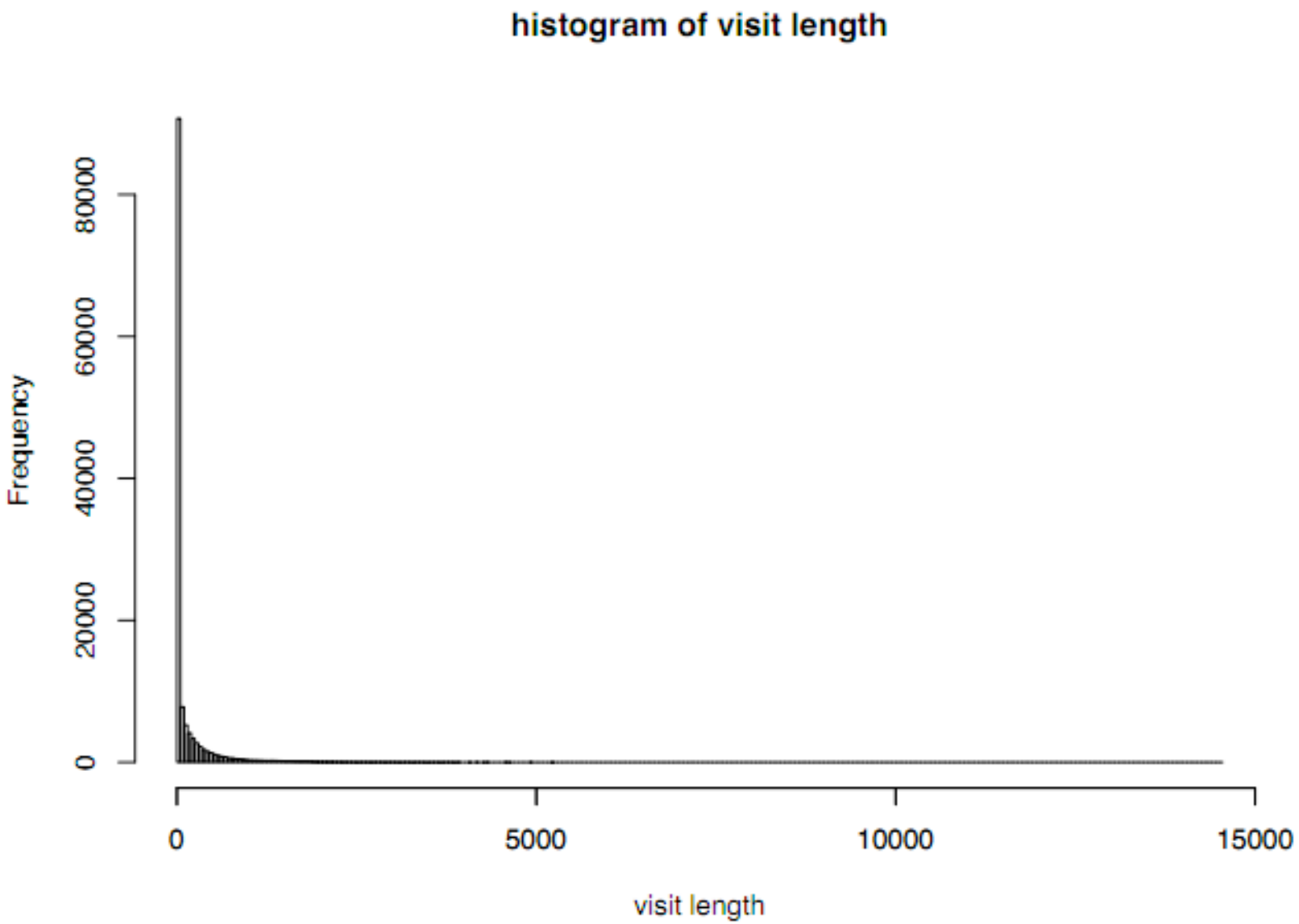
Formally, we would consider the proportion of tables having an absolute difference as large or larger than 0.017 -- That turns out to be 0.32 or 32% of our 1,000 re-randomized tables

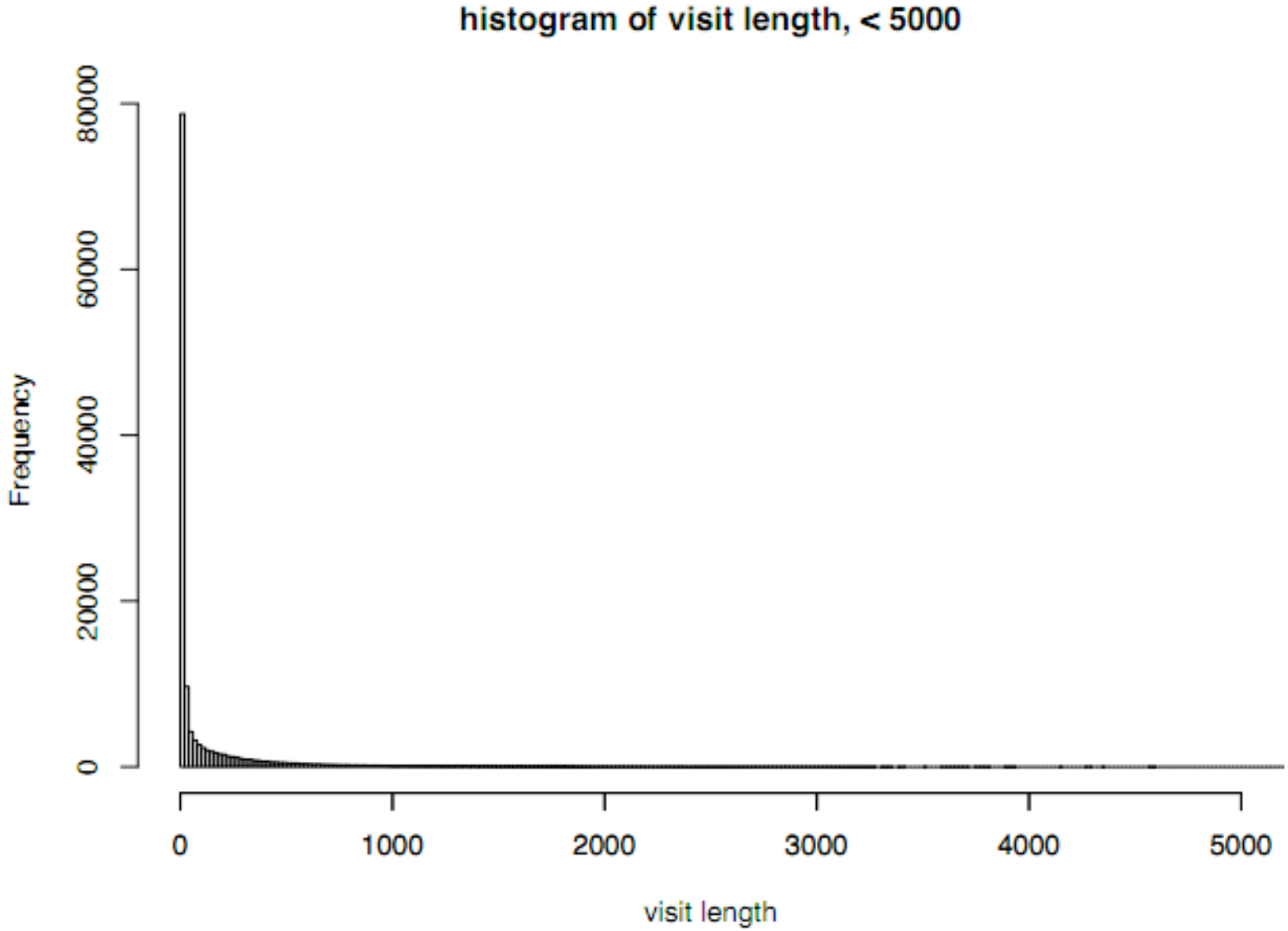
That means we cannot reject the null hypothesis that Tabs and List are performing differently in terms of Pageviews per visit

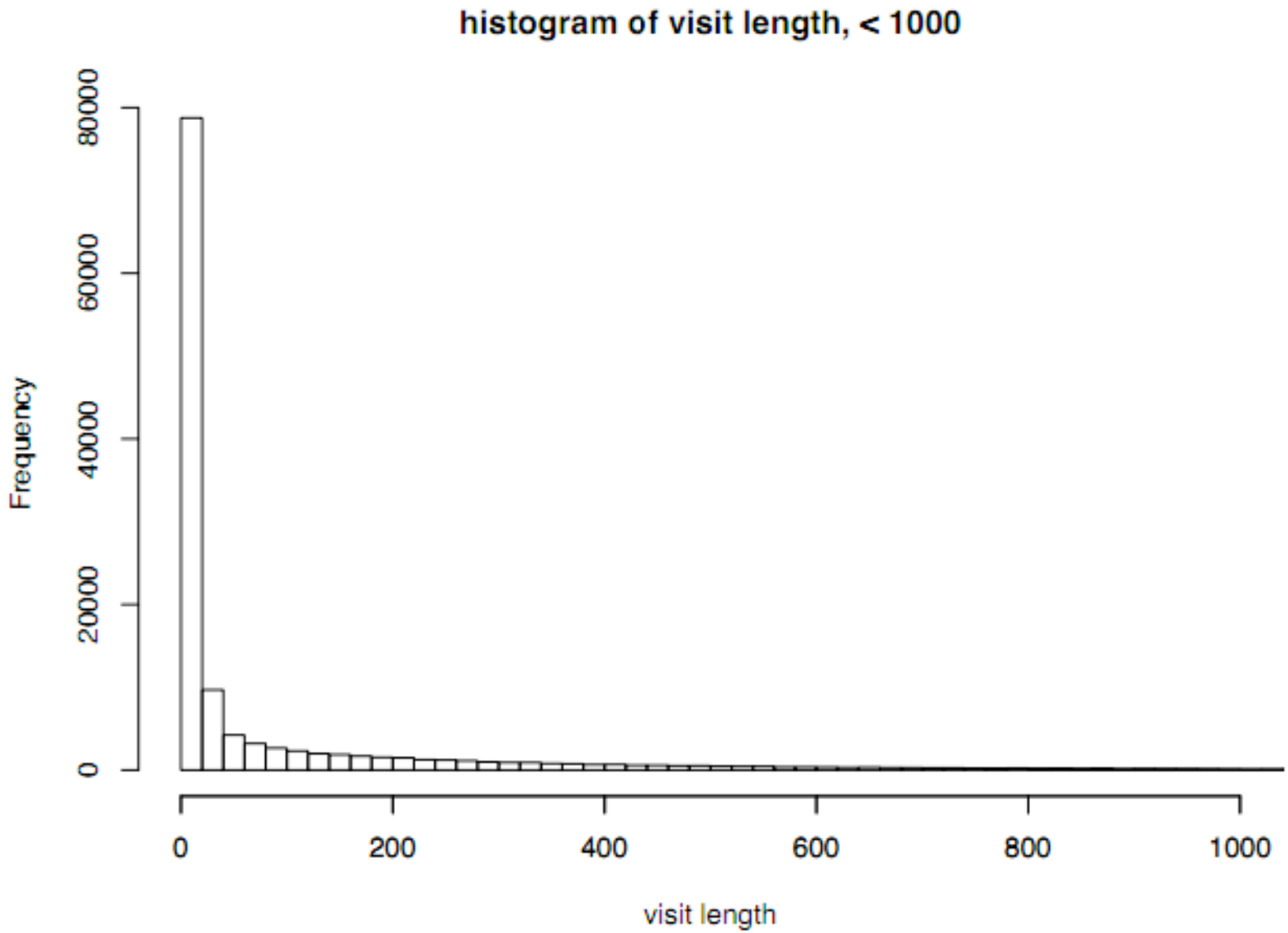
The data

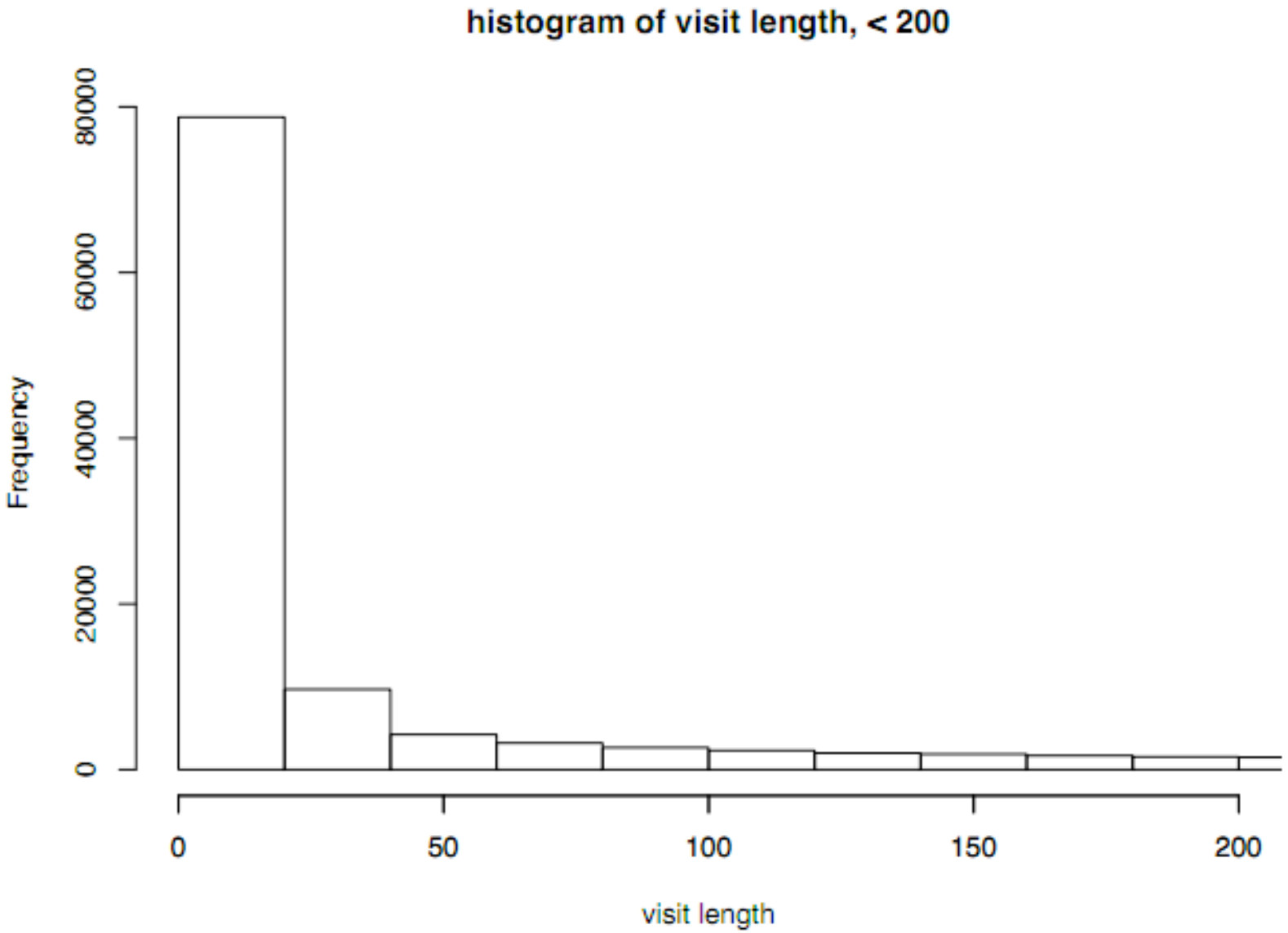
Testing is becoming a little tedious, so let's now move on to have a look at the rest of the data -- We will have one more test for you to perform in lab, but for now, let's consider some other topics

So, let's have a look at visit lengths -- As you see on the next page, no matter how tightly we restrict the x axis, we aren't getting a lot of new information; primarily we see a large number of points on the left and then a very long tail to the right


histogram of visit length
Frequency
0
20000
40000
60000
80000
0
5000
10000
15000
visit length


histogram of visit length, < 5000
Frequency
0
20000
40000
60000
80000
0
1000
2000
3000
4000
5000
visit length


histogram of visit length, < 1000
Frequency
0
20000
40000
60000
80000
0
200
400
600
800
1000
visit length


histogram of visit length, < 200
Frequency
0
20000
40000
60000
80000
0
50
100
150
200
visit length

Transformations

At a very practical level, transforming these data help us see more; that is, monotone transforms like square roots or the logarithm*, have a (relatively) **greater effect on large values, bringing them in closer**

With strongly skewed data, we want to consider transformations of this type simply to see what's going on...

* here log = natural log!

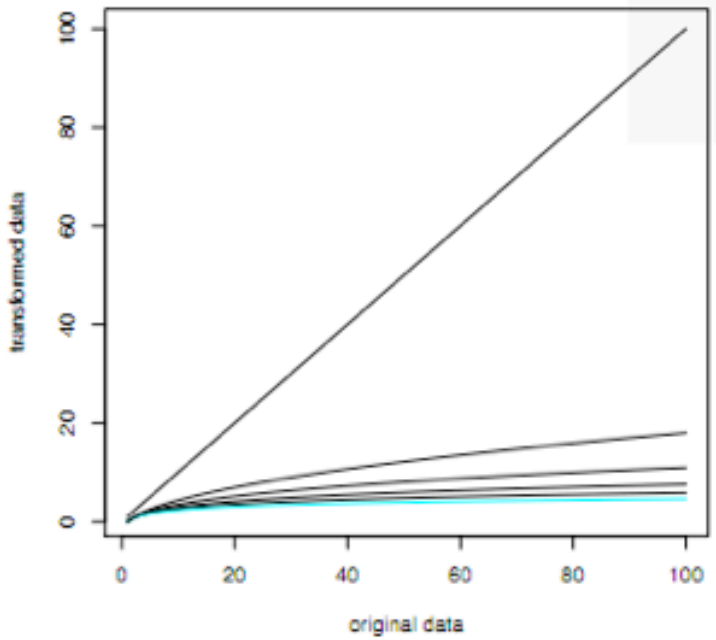
Transformations

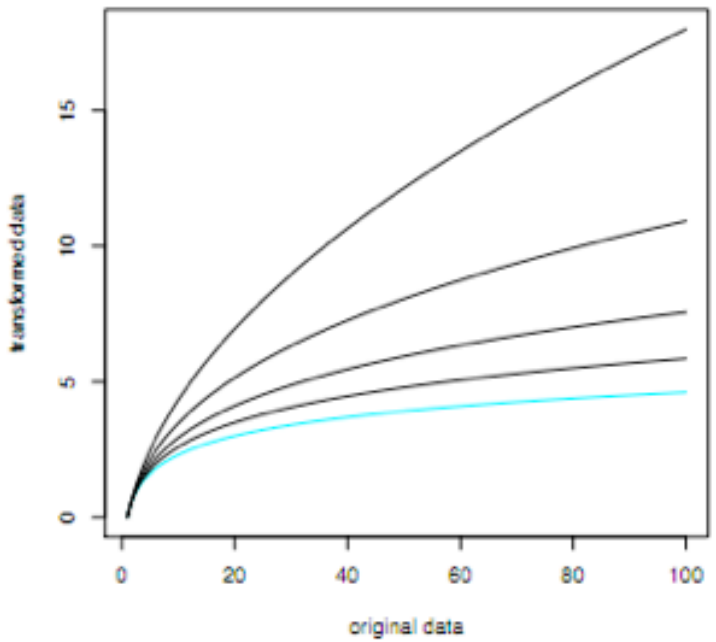
For data with a right skew, the square root, cube root, fourth root, and so on, can be used to define **a family of transformations** that all have the effect of taking big values and bringing them in closer to the rest of the data, to the smaller values

At the right we have a graph of this family (using the the square root, cube root and so on) to help you see what they are doing to the data; the top plot has the original scale (the straight line just being y=x) and the bottom plot is zoomed in on the curves -- What do you see?

Let's see how this works with our visit length data...

* technically the family is given by $f(x) = \begin{cases} \frac{x^\lambda - 1}{\lambda} & \lambda > 0 \\ \log(x) & \lambda = 0 \end{cases}$





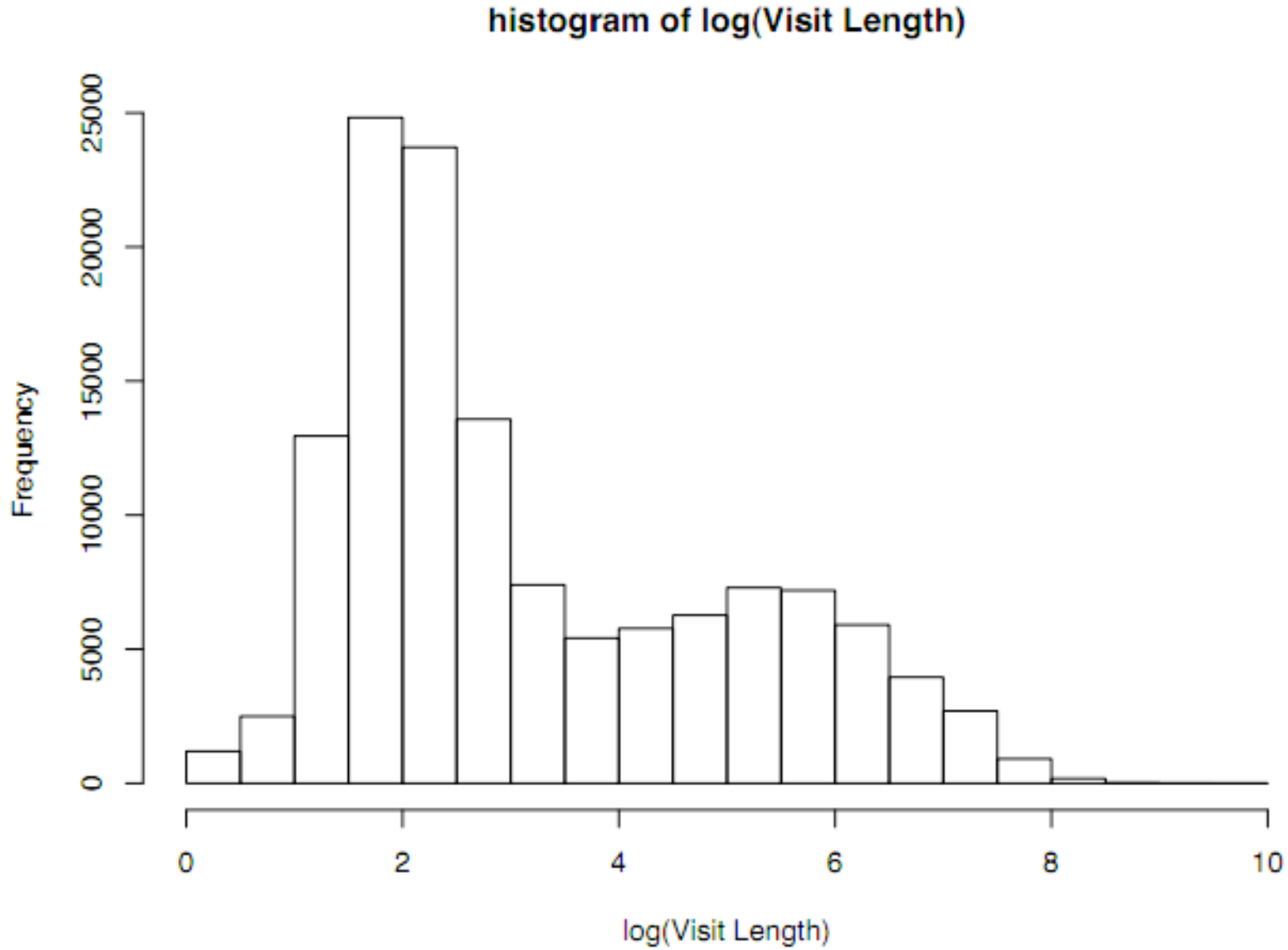
An aside

Notice that the members of this family are all monotonic transformations; that means if x < y then f(x) < f(y); put another way, **these transformation do not change the order of the data**

... and their effect on the median now becomes obvious: let $\tilde{x}$ be the median of the data points $x_1, x_2, \ldots, x_n$; the median of the transformed points $f(x_1), f(x_2), \ldots, f(x_n)$ is just $f(\tilde{x})$

Notice also, that as you work from the top curve to the bottom, the effect on the big values becomes more and more extreme; the square root of 10,000 is 100, the cube root is about 22, and the fifth root is about 6 -- in short, we are pulling the big values in closer and closer*

* To get the logarithm as a limit you need to use not just the simple roots but the family spelled out on the previous page

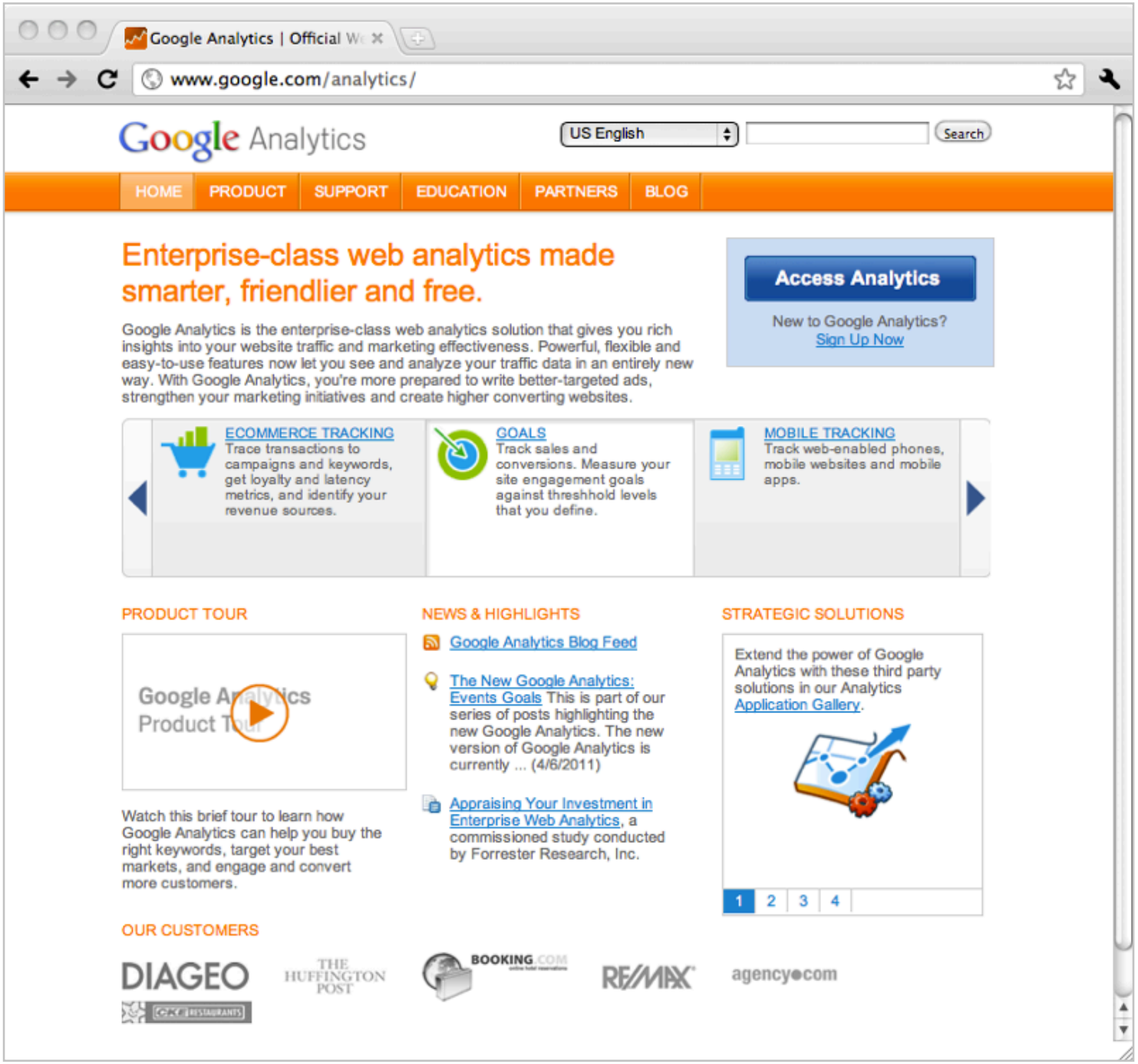

histogram of log(Visit Length)
Frequency
0
5000
10000
15000
20000
25000
0
2
4
6
8
10
log(Visit Length)

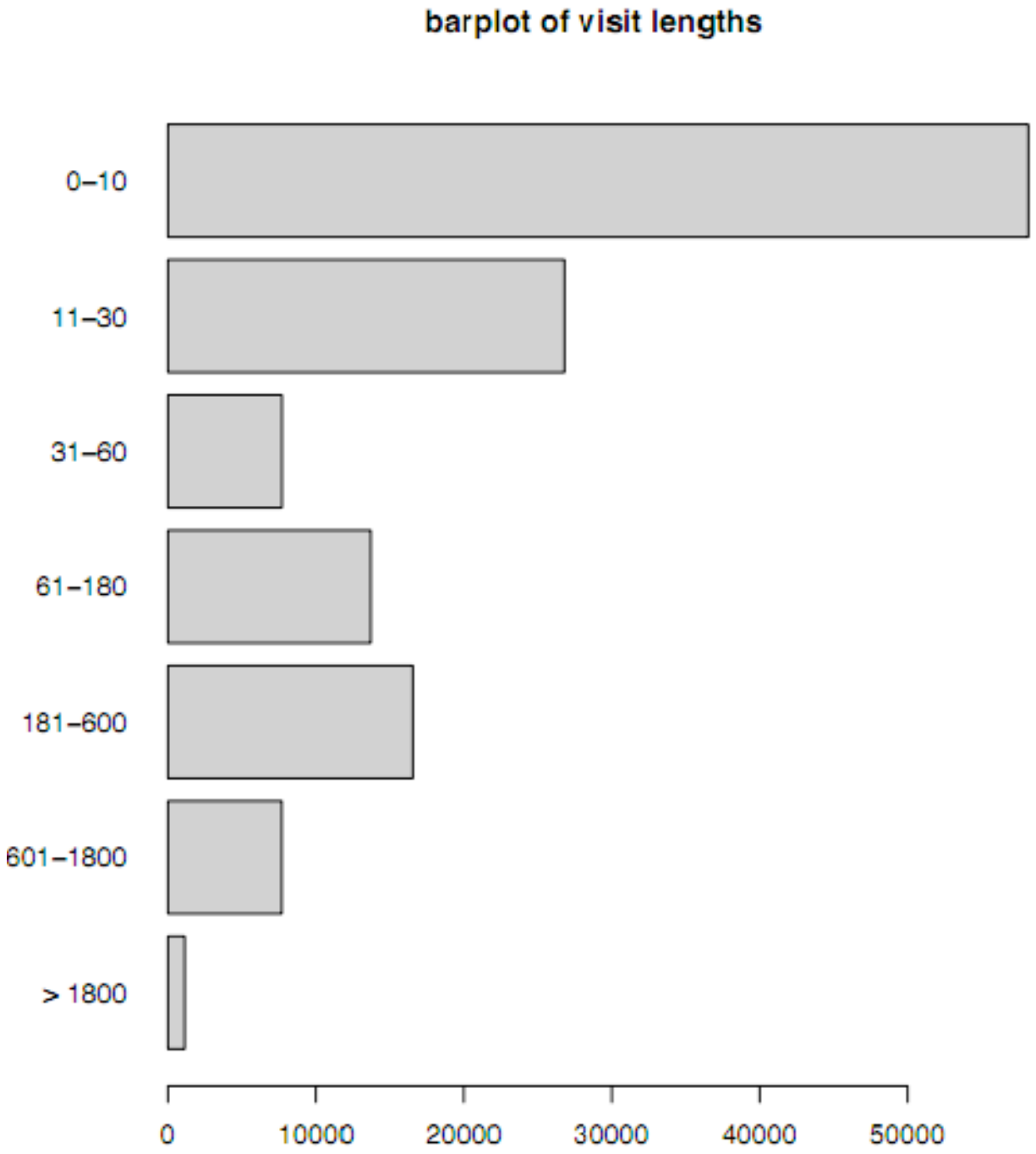
Alternatives

Yesterday, while dutifully analyzing data at the local Starbucks, I started talking to a man who was busy pouring over web site statistics; he owns a site related to retirement communities (although he is probably 40 years away from having to make use of his services)

He was using a platform offered by Google; by putting a piece of code on each of your web pages, you can have Google collect information about who is visiting your site, how long they stay and so on -- in short, the data we have from the New York Times

His view of Visit Length looked like this...


Google Analytics | Official W
www.google.com/analytics/
Google Analytics
US English
Search
HOME
PRODUCT
SUPPORT
EDUCATION
PARTNERS
BLOG
Enterprise-class web analytics made smarter, friendlier and free.
Google Analytics is the enterprise-class web analytics solution that gives you rich insights into your website traffic and marketing effectiveness. Powerful, flexible and easy-to-use features now let you see and analyze your traffic data in an entirely new way. With Google Analytics, you're more prepared to write better-targeted ads, strengthen your marketing initiatives and create higher converting websites.
Access Analytics
New to Google Analytics?
Sign Up Now
ECOMMERCE TRACKING
Trace transactions to campaigns and keywords, get loyalty and latency metrics, and identify your revenue sources.
GOALS
Track sales and conversions. Measure your site engagement goals against threshhold levels that you define.
MOBILE TRACKING
Track web-enabled phones, mobile websites and mobile apps.
PRODUCT TOUR
Google Analytics
Product Tour
Watch this brief tour to learn how Google Analytics can help you buy the right keywords, target your best markets, and engage and convert more customers.
NEWS & HIGHLIGHTS
Google Analytics Blog Feed
The New Google Analytics: Events Goals This is part of our series of posts highlighting the new Google Analytics. The new version of Google Analytics is currently ... (4/6/2011)
Appraising Your Investment in Enterprise Web Analytics, a commissioned study conducted by Forrester Research, Inc.
STRATEGIC SOLUTIONS
Extend the power of Google Analytics with these third party solutions in our Analytics Application Gallery.
1
2
3
4
OUR CUSTOMERS
DIAGEO
THE HUFFINGTON POST
BOOKING.COM
RE/MAX
agency●com


barplot of visit lengths
0–10
11–30
31–60
61–180
181–600
601–1800
> 1800
0
10000
20000
30000
40000
50000

Alternatives

The cutoffs used by common web site analysis packages have a (sort of) **logarithmic feel** to them -- from 10 seconds to 30 seconds to a minute to 1.5 minutes to 10 minutes to 30 minutes; notice that even in this display, the bimodal nature of the data are clear

My hope is that this kind of presentation will demystify the log-transform -- we can think of creating a display by either c**reating bins that get longer as you go into the right tail**, or using equally spaced bins on the transformed data

Transformations

With that out of the way...

Our observational unit is a visit, and specifically the first visit to the Travel Section; in the distribution on the previous page we see one mode centered around 2 log-seconds -- which in our original units is exp(2) = 7.4 seconds -- and the other at around 5.5 log-seconds -- which is exp(5.5) = 245 seconds or about 4 minutes

Notice what we've done here; **we've transformed back to the original units** when talking about the log-Visit Lengths

Transformations

In some situations, **it is easier to work on one particular scale or another**; think about currency or measures of weight or volume -- fairly straightforward transformations

In some applications a logarithm might be the de facto measurement standard; but it hasn't really caught on for Visit Length -- instead we appeal to a transformation to help us see things about the data, but we have to back-transform to the original scale for presentation

1 U.S. dollar = 5.78338895 Danish kroner

1 US gallon = 3.78541178 litres

1 short ton = 2000 pounds

Transformations

In fact, non-linear scaling is used across science

Should we measure acidity in the concentration of H^+ ions (linear) or PH (logarithmic)?

Should we measure the magnitude of an earthquake in mm of amplitude (linear) or on the Richter scale (logarithmic)?

Should eyeglass lenses be measured in terms of focal length in cm (linear) or dioptres (a reciprocal)?

Transformations

Now, let's consider what it means to work with the log of Visit Lengths -- In particular, let's assume we have some number of data values $x_1, \ldots, x_n$

$$\frac{\log x_1 + \log x_2 + \cdots + \log x_n}{n} = \frac{1}{n} \log x_1 x_2 \cdots x_n$$

$$= \log \left[(x_1 x_2 \cdots x_n)^{\frac{1}{n}} \right]$$

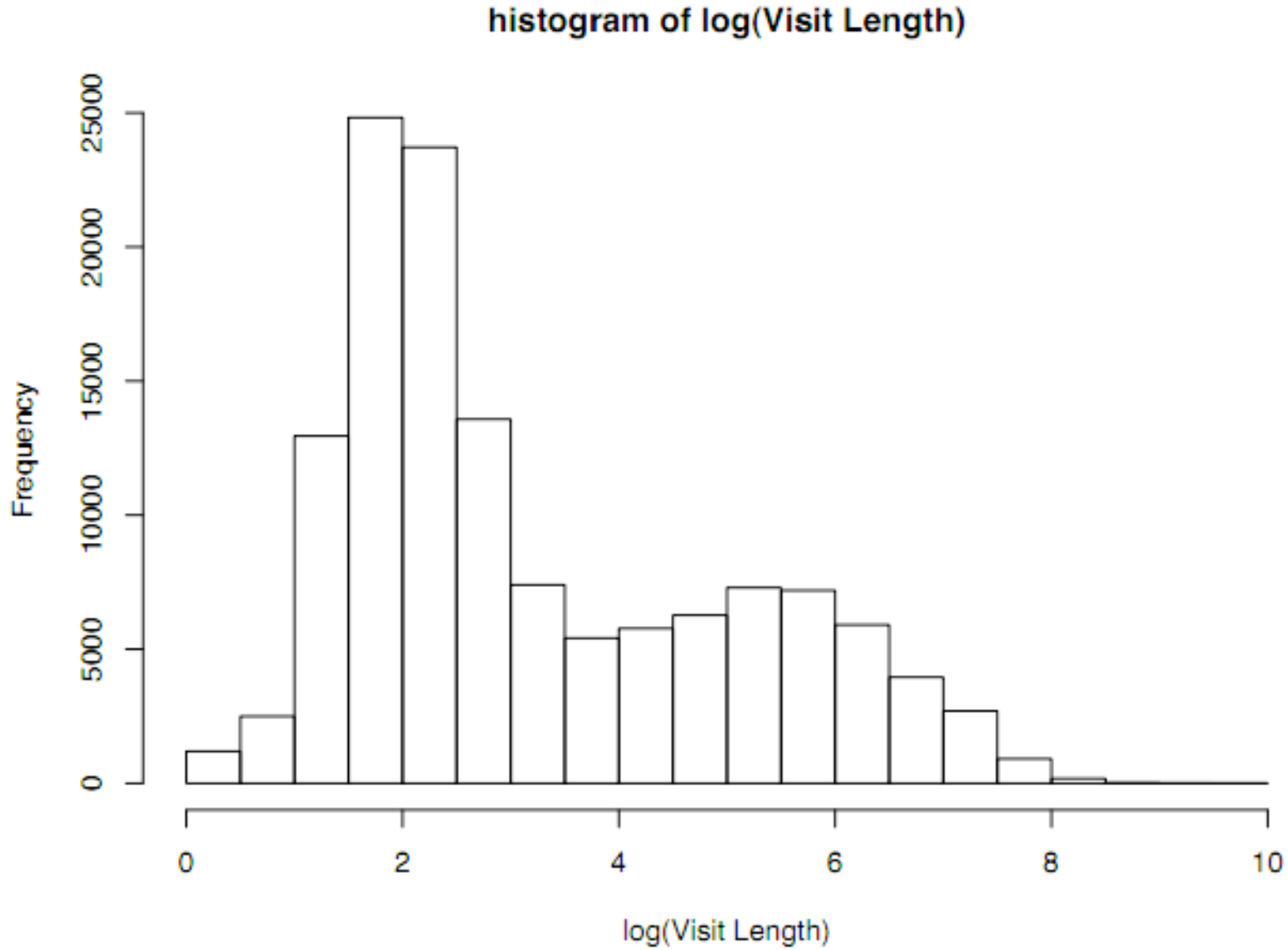
This means that the average of the logged values gives us the logarithm of the geometric mean -- when we back-transform (by exponentiating) we have the geometric mean of $x_1, x_2, \ldots, x_n$, a quantity that is back in the original units

* here log = natural log!

Question

Now, back to the log-Visit Length distribution: We have seen two modes, which implies that there are two kinds of visits taking place -- when faced with a distribution like this, there is really only one question that comes to mind...

"Why?" What variables in our data set might help us explain the two peaks?

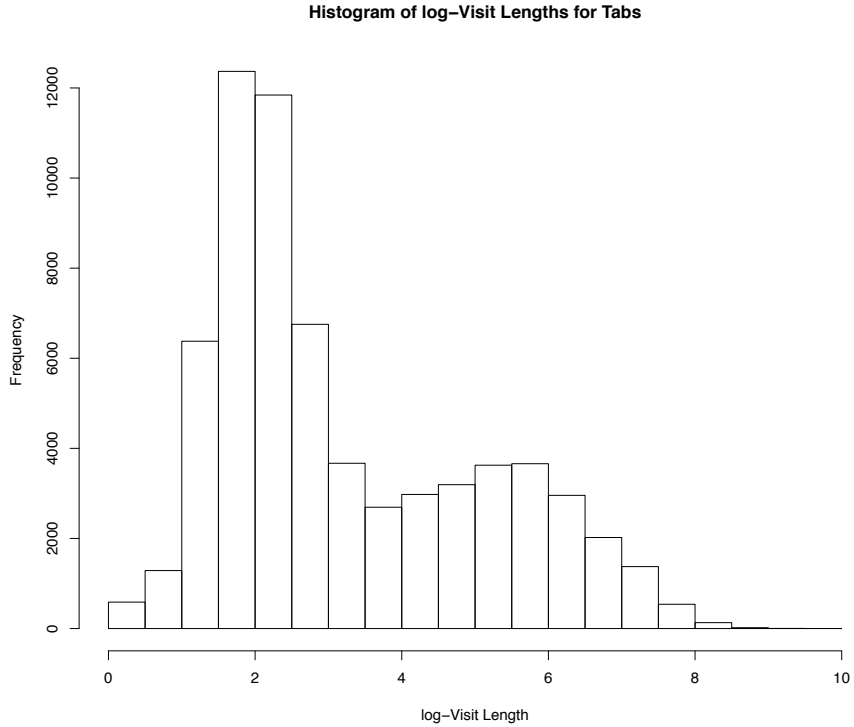

histogram of log(Visit Length)
Frequency
0
5000
10000
15000
20000
25000
0
2
4
6
8
10
log(Visit Length)

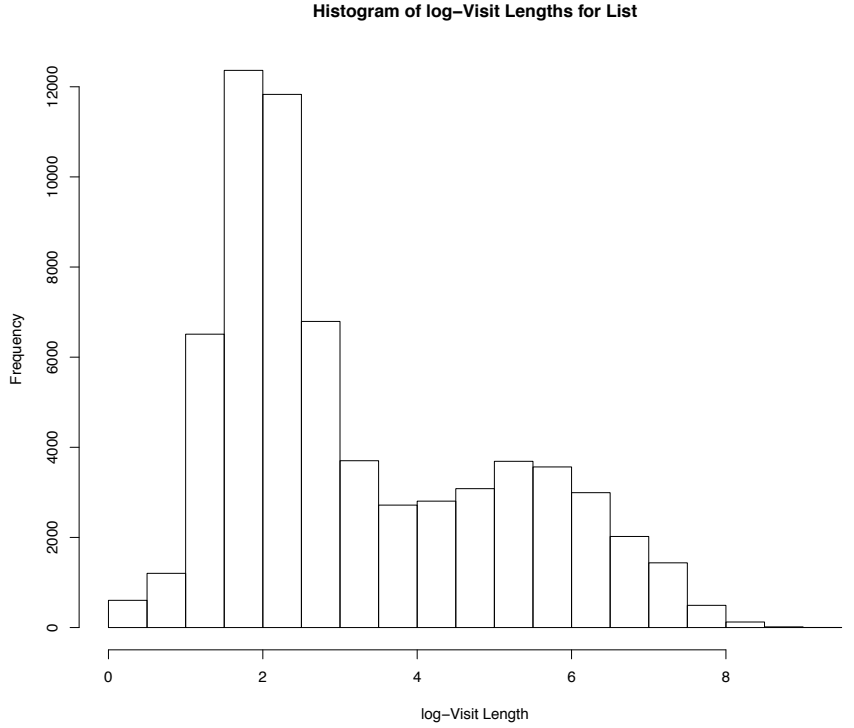
Two groups?

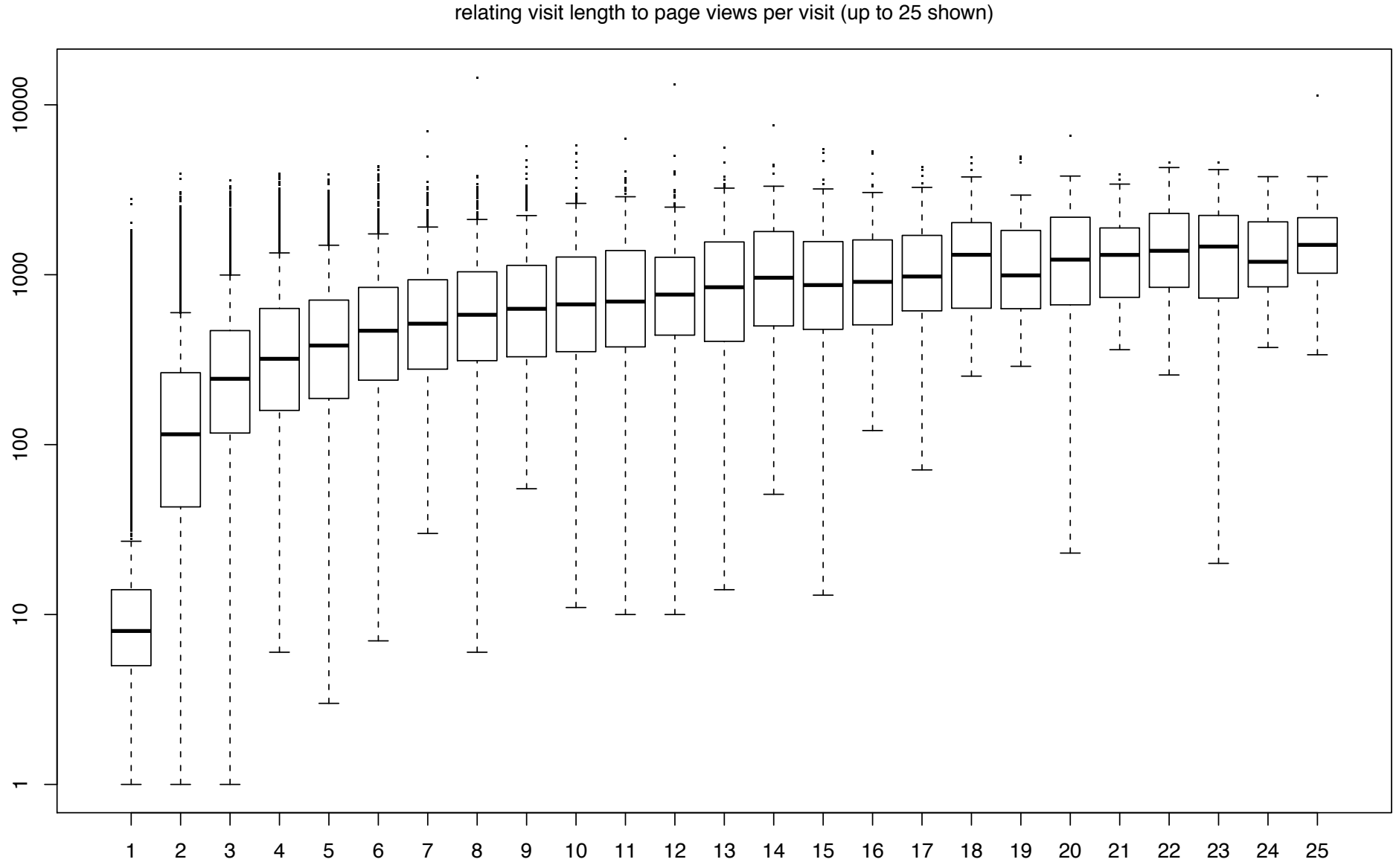
An obvious answer is that somehow the groups are the result of our experiment -- That one bump represents Tabs and the other Lists

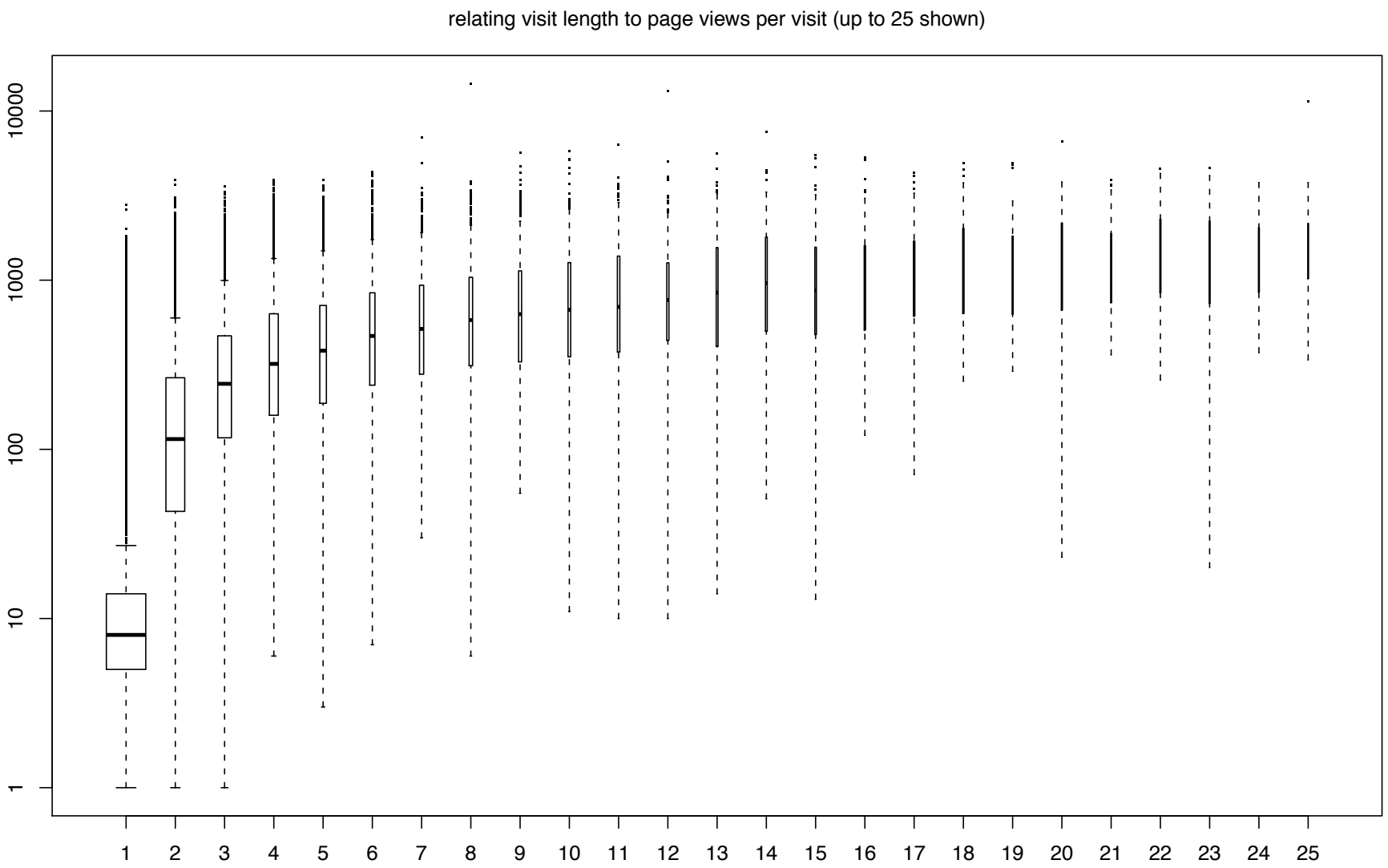
On the right we have two histograms, the top for just the log-visit lengths for those people seeing Tabs and the bottom for those seeing Lists

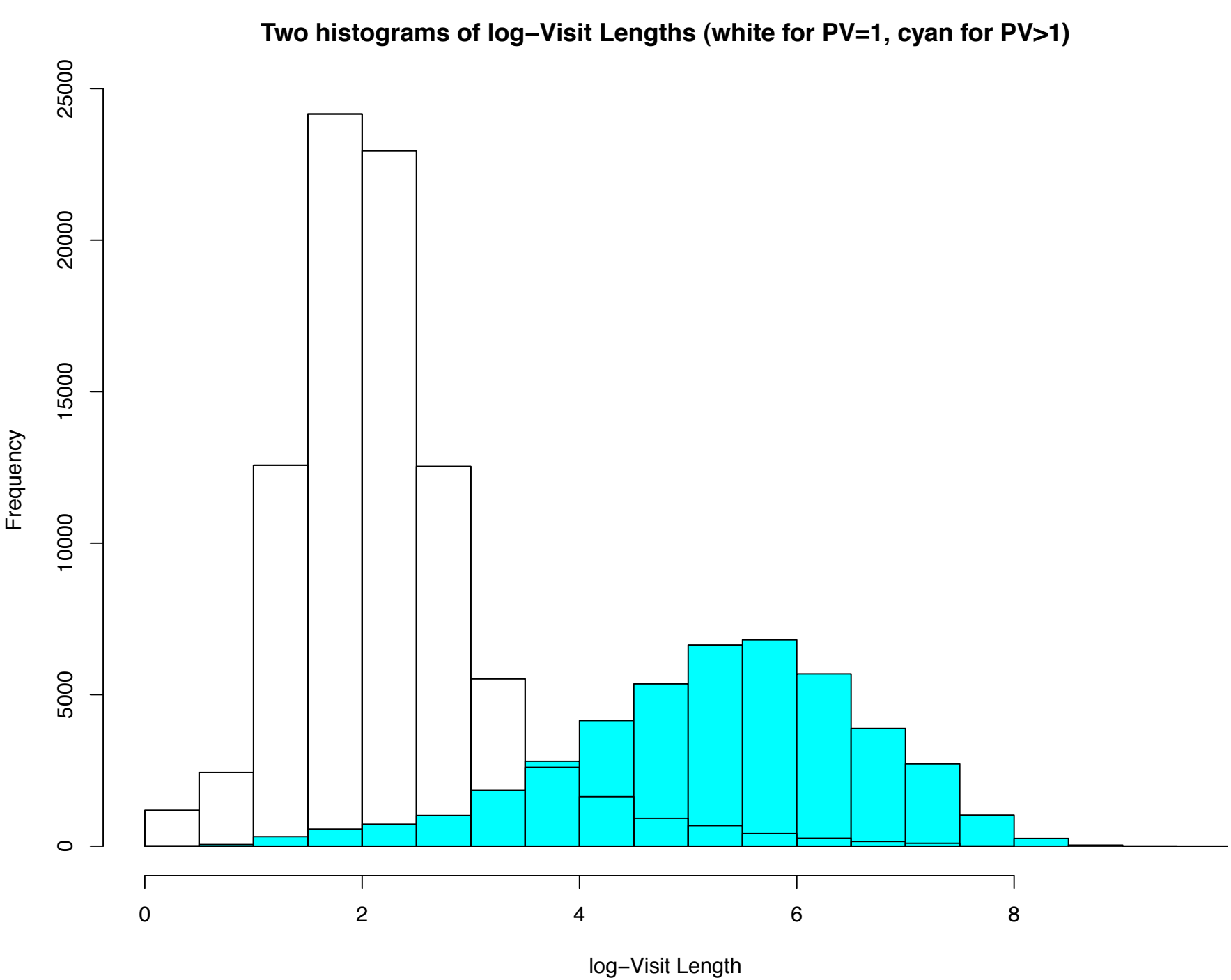
Any difference?






relating visit length to page views per visit (up to 25 shown)
10000
1000
100
10
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25


relating visit length to page views per visit (up to 25 shown)
10000
1000
100
10
1
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25


Two histograms of log−Visit Lengths (white for PV=1, cyan for PV>1)
Frequency
0
5000
10000
15000
20000
25000
0
2
4
6
8
log−Visit Length

Two groups

What we can see from this is that there are essentially two groups in the data -- Those that visit only one page (and their visit lengths correspond to the time it takes to load the page) and those that visit more pages

In general, when we identify groups or clusters in the data, we want to see what causes them -- It would have been amazing if our experiment created the groups, but in this case it has more to do with how visit length is calculated

With that out of the way, we can still legitimately ask if there is a difference between the two groups based on visit length or log-visit length...