


Blue Spark Digital Microphone Impressions: Improving my Audio Quality!

 November 11, 2014 by Geet

Blue Spark Digital Microphone Impressions: Improving my Audio!



Moto 360 Review! 2 Months with an Android Wear Smartwatch

□ November 2, 2014 by Geet

Moto 360 Review! Habits After 2 Months



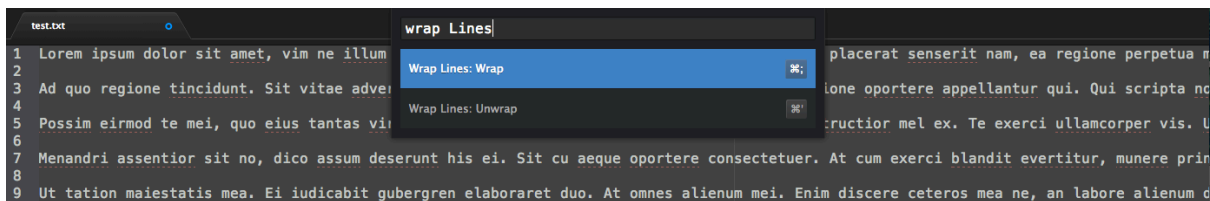
Inbox by Gmail First Impressions & Invitations!

□ November 2, 2014 by Geet



Atom package for wrapping lines

□ May 22, 2014 by Geet



Even though I have always seemed to fall back to Vim in a terminal for my coding, I have been using Atom (with SSHFS) for some time now and for reasons I don't quite understand yet, I've not really felt the urge to stop. My initial impressions were that Atom is a Sublime knock-off with some more flexibility, so I expected after a short time of playing with my invite, to go back to my tried and true Vim since that's what I did with Sublime. But here I am, and I'm more convinced to stick with it some more now that it has been open sourced. The 'ggq' wrap lines feature was one I sorely missed, so I thought I'd try to script it.

I was surprised at how easy and **quick** it was to code a package for it, and to subsequently make available to everyone else through their package management system (the image above takes you to the package). On a slightly related note, my friend Rob pointed me to this interesting Vim

Lamport Stresses Thinking Before Coding

□ April 13, 2014 by Geet

More specifically, he suggests some of the 'thought' before coding should be formalized as a specification (like a blueprint) written in the language of mathematics. The TLA specification language uses satisfiability of boolean formulas as a mathematical way to express initial states and the possibilities for a state to transition to other states. The framework provides the programmer a way to (1) think in depth about the possible states before coding and running test cases, (2) better document what an implemented piece of code does making it more convenient for updates later, and (3) automatically find bugs from logical inconsistencies.

This video provides more motivation and examples for what I said above, including a couple of anecdotes that mentioning the use of TLA by Amazon and Microsoft. Overall, I think he made a good argument that putting more effort in the specification can lead to better design decisions down the line.

A couple of notes:

(1) This approach is mostly about 'functional' specification, so it provides little assistance in implementing such a specification. In this way, it is largely independent of programming language choice/design, compilation, and optimization in practice. Many specific algorithms could satisfy a functional specification. In his talk, I got the impression that these other issues were trivial in comparison to specification, which I don't think is often the case. However, I certainly agree that more thought should be spent at the specification stage, if possible.

(2) Large-scale adoption of this formalism seems like it would be difficult. There is the obvious overhead in teaching people to understand and implement this formalism in their work. Even if knowledge of TLA was part of a programmer's toolkit, this type of formalism may be far more beneficial for specific, more 'predictable' problems. By predictable, I mean that a programmer (or computer) can consider the possibilities of desirable and undesirable states *a priori*. For example, the task "write a program to produce a pleasing sound" is not only a larger/more general problem, but it also very difficult to specify it exactly. This is an extreme example, but less extreme versions of it are a very common and practical form of programming (e.g. programs that are highly dependent on human satisfaction during use). For these applications, past intuition, seeing how you, a test group of people, and even your customers respond to your program may provide the most valuable information for improvement.

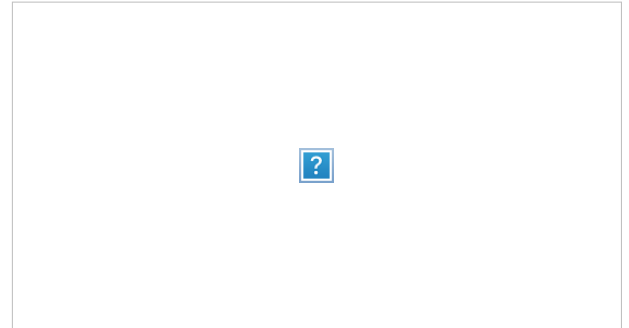
Leslie Lamport is a computer scientist working at Microsoft Research. He recently won the Turing Award (one of the highest honors in computer science), and ever since I learned about Lamport clocks, he has been one of my favorite CS academics.

Consider the Nimrod Programming Language

□ March 3, 2014 by Geet

Mid-April Update: Thanks for the thoughts everyone! The post quickly grew to >10,000 views and reached #10 on Hacker News for a while. I continue to enjoy the language for a multitude of applications from basic scripting to ‘hard work’ tasks.

I love to play around with new computer programming languages. Even though I spend most of my time in industry tested standards for their respective applications (e.g. Java, C, C++, Python, Javascript, ...), I think there are a lot of good reasons—especially these days—to learn and experiment with new languages. The impact of modern language development isn’t limited to a cottage industry of computer scientists and programmers. Take the growing Scala language as an example. Twitter transitioned from a framework primarily using Ruby to Scala to scale their service and to maintain a programming model they desired. I also believe we are finally beginning to see languages that are elegant, expressive, and, importantly, *fast*. For example, these days, a company using two industry standards like Python and C++ might do ‘heavy lifting’ in C++ and write a lot of ‘high level’ wrapper code and scripts in a language like Python. Why not use just one language? Why is Python ‘slow’ for some tasks and C++ ‘unpleasant’ for scripting tasks? A good language should be expressive enough to elegantly express domain-specific tasks while allowing the programmer to make the things that need to be fast, *fast*.



Why the competition may not *quite* fit the bill

I *could* just list out great Nimrod features and say: ‘consider it’, but I don’t think that these features are very useful without some explanation of why these features provide an overall better experience than other compelling languages. When it comes to picking a programming language that attempts a speed-elegance unification, there are a lot of choices. The five on the ‘short list’ that I discuss in this post are:

- Scala
- Rust
- Julia
- C++11 and beyond
- Nimrod

There are other options that I could put on this list like Haskell or Go, and I have my reasons for picking the 5 above, but I don’t want to discuss them right now. What I would like to do is convince you that Nimrod is a particularly nice language to consider since the design decisions they made, to

me, result in an elegant, expressive, and fast language (though I understand people have different syntactic preferences). These are my initial thoughts after nearly three weeks of coding a lot in Nimrod. I am writing this because I think the language needs to get more attention than it has, and it deserves to be taken seriously as a worthy competitor to the other four mentioned above.

Of course this is not a super-detailed comparison, but overall, I hope I provide some reasons you may want to consider Nimrod over these other very nice languages. They all have stuff to offer over Nimrod and vice-versa. And there are also a number of overlapping features. Ideally I would like to have a highly expressive, fast language that is what I call “K&R-memorable” which basically means that it is approximately as easy to understand it as it is to understand C (all you do is read K&R and you’re good).

C++11 has really brought C++ a long way. Coding with it results in a lot less boiler-plate code and it did a reasonable job of incorporating higher-order functions and handy value-semantic preserving features such as move semantics. However, there’s still a lot of boiler plate (e.g. it’s 2014 and I’m still writing header files separate from source because compilation time with header-only files is too slow?), and now I need to implement more operators for classes to preserve value semantics (don’t forget to implement the move assignment operator!). So C++11 is nice and incorporates some modern features, esp. because it works with all other C code, but it’s much too complex, and I think, far less elegant than the other alternatives.

Scala and Rust are both very interesting languages (in general, simpler to understand than the totality of C++11). I have had a good deal of experience with Scala and have played with Rust for a couple of minor tasks. Both languages implement traits. To me, traits are a far more elegant way of adding similar functionality to different objects when compared with multiple inheritance. But my experience with Scala has shown me that while it is easy to *use* libraries, it is harder to *design* them in the midst of a complex graph of how objects and traits are related to one another. I spent a lot of time engineering the types to be *just* right, which is great, but it was also frustrating and I felt that the safety I desire at compile time would be more easily achieved without such a complex system. I will discuss some design decisions made by Nimrod below that I think result in less time spent on type nitpicking and more time spent on getting the ‘job done right’ with reasonable safety features. Rust provides more built-in memory safety, which is great, but understanding how it all works and the ‘conversations’ with the compiler after coding can be frustrating. Believe it or not, sometimes hard guarantees with types/traits and memory are not worth the pain when programming (i.e. the programmer effort required, mental model and syntactic complexity). I think this is precisely why the adoption of a slow dynamically duck-typed language like Python has been so successful. They’re ‘easy’ to program in. I think Nimrod is a happier medium.

Julia’s motivation comes from two places. The language resembles typical scientific programming syntax (ala Matlab and Pylab) that executes *fast* when compiled, and offers extensive and intuitive metaprogramming capabilities since it is homoiconic like Lisp. (And the scientist in me really likes the IJulia notebook feature that they have apparently worked quickly to develop.) I will show some examples below on how Nimrod offers a powerful and elegant metaprogramming environment without necessarily being homoiconic. My only real concern with Julia is lower-level systems programming. Forced garbage collection can be a game-changer here, and I’m not sure I think its choice of being ‘largely’ dynamically typed is a good one in this setting either. Providing a library

developer some level of type annotation and type class restriction can be useful for engineering purposes and more helpful when dealing with compile-time errors. I work in the area of computational biology and I am left wondering: is Julia the right language to build the fastest read aligners, gene expression estimators, etc.? These tools are often written in C/C++, so Julia code would have to beat that! A similar sentiment applies to Scala: it's dependence on the JVM has actually resulted in very poor performance in even a simple multicore application, in my experience.

Quick start with Nimrod

OK, so you should read the tutorial and eventually the manual on the web site to get a quick start and get to know the language better, but I'll tell you how I started using it: as a scripting language. I know this isn't the best for 'performance' testing, but any language that has this 'unification' quality should be equally good at scripting as it is for high-performance applications. Here is a simple example:

```
1  import os
2
3  proc shell(cmd: string) =
4      if os.execShellCmd(cmd) != 0:
5          raise newException(EOS, cmd & "returned non-zero")
6
7  proc fexists(fname: string) : bool =
8      try: discard Open(fname)
9      except EIO: return false
10     return true
11
12  const fromScratch = false
13
14  shell "clear";
15
16  if fromScratch:
17      echo "Removing cached files and log";
18      shell "rm -rf nimcache log.txt";
19      echo "All output in log.txt";
20      echo "Compiling ...";
21      shell "g++ -fPIC -O3 -shared -std=c++11 bla.so bla.cpp";
22
23  # More of the pipeline, e.g.
24
25  if not fexists("blah.txt"): createBlah()
26  else: useBlah()
```

This, to me is a very clean way to do basic shell scripting while having the power of a full programming language.

Nimrod avoids 'over-objectifying'

OK, so that was relatively straightforward. Here is a simple example of how to create a matrix type (partially inspired from a [stackoverflow](#) post):

```

1  type Matrix[T] = object
2      nrows, ncols: int
3      data: seq[T]
4
5  proc index(A: Matrix, r,c: int): int {.inline.} =
6      if r<0 or r>A.nrows-1 or c<0 or c>A.ncols-1
7          raise newException(EInvalidIndex, &quot;matrix index out of bounds&quot;)
8      result = r*A.ncols+c
9
10 proc alloc(A: var Matrix, nrows,ncols: int) {.inline.} =
11     ## Allocate space for a m x n matrix
12     A.nrows = nrows
13     A.ncols = ncols
14     newSeq(A.data, nrows*ncols)
15
16 proc `[]`(A: Matrix, r,c: int): Matrix.T =
17     ## Return the element at A[r,c]
18     result = A.data[A.index(r,c)]
19
20 proc `[]`= `(A: var Matrix, r,c: int, val: Matrix.T) =
21     ## Sets A[r,c] to val
22     A.data[A.index(r,c)] = val
23
24 iterator elements(A: Matrix): tuple[i:int, j:int, x:Matrix.T] =
25     ## Iterates through matrix elements row-wise
26     for i in 0 .. &lt;A.nrows:
27         for j in 0 .. &lt;A.ncols:
28             yield (i,j,A[i,j])
29
30 proc `$`(A: Matrix) : string =
31     ## String representation of matrix
32     result = &quot;&quot;
33     for i in 0 .. &lt;A.nrows:
34         for j in 0 .. &lt;A.ncols:
35             result.add($A[i,j] &quot;&quot;)
36     result.add(&quot;\n&quot;)

```

The first thing to notice is that a matrix is an object type that contains data and its number of rows and columns. All the methods take a matrix as the first argument. This matrix is generic on any type `Matrix.T`. An alternative syntax where `[T]` comes after a procedure name may also be used. Nimrod uses a uniform call syntax that implies these two calls are equivalent:

```

1  A.alloc(nr,nc): ...
2  alloc(A,nr,nc)

```

Notice that `elements` is an iterator. This is a very efficient iterator called an `inline` iterator. You can read more about this in the tutorial and manual. The ``$`` operator before a variable is the standard `to string` operator. This allows you to do:

```

1  echo A

```


and a matrix will be printed out.

The uniform call syntax is a simple way to support a lot of ‘call-chaining’ like behavior commonly seen in object-functional programming and avoids forcing methods to be in objects. As an example, say I have a `BigInt`, and a little `int` and I want to be able to support addition of them. In Nimrod, you simply write a procedure that overloads the ‘+’ operator and it works (otherwise you get a compile time error). In a language like Scala, you define what’s called an ‘implicit conversion’ to do this for you. The added idea of an implicit conversion on objects and having to define them so explicitly seems more complex than just overloading the operator. Note that there are other cases where you would like to use implicit conversions and Nimrod provides this capability. Calls to procedures in Nimrod can be ‘pass by reference’ in C++ world:

```
1 | proc test(x:var int) =  
2 |   x=5  
3 |  
4 |   var x = 3  
5 |   echo x  
6 |   test(x)  
7 |   echo x
```

results in:

```
1 | 3  
2 | 5
```

The compiler will chose the appropriate method at compile time to call based on the types in the procedure. Nimrod also supports multiple dispatch.

Nimrod has an intuitive type system

As mentioned above, traits are a nice way of defining components of functionality tied to an object and the compiler will error out if certain traits are required, but missing, for example. I also mentioned that this can lead to complexities in library design and engineering (which may be good or bad depending on your perspective and the outcome).

One feature of Nimrod that’s appealing is that it offers the programmer *type classes* — the ability to group types into a single type (e.g. define `float` and `int` to be of type `number`), and *distinct types* — the ability to create two different types corresponding to the same underlying data type (e.g. `dollars` and `euros` are both `ints` in their tutorial example). Similar to type classes, Nimrod also allows constraints on generic types, and support for additional constraints is in the works. So the compiler will provide an error message if a method is not defined for a particular class of types its defined on or if a desired method is missing. Traits appear to be a formalism that could be useful, but might result in a lot of added complexity given the capabilities already provided by type classes and distinct types. Nimrod also supports an effects system which allows for additional compile-time safety checks.

You will want to metaprogram in Nimrod

Nimrod makes it easy to extend the language and the abstract syntax tree to generate the code you want. Say I wanted to do an openMP-like parallel for using Nimrod's threads over a shared sequence of data. The thread code looks a lot like this:

```
1 | template parallelFor[T](data:openarray[T], i:expr, numProcs:int, body
2 |   let numOpsPerThread = (data.len/numProcs).toInt
3 |   proc fe(j:int) {.thread.} =
4 |     for q in 0 .. numOpsPerThread-1:
5 |       let i = j*numOpsPerThread+q
6 |       body # so something with data[i]
7 |   var thr: array[0..numProcs-1, TThread[int]]
8 |   for j in 0 .. numProcs-1:
9 |     createThread(thr[j], fe, j)
10 |   joinThreads(thr)
```

But using the template as defined above, I can just do:

```
1 | parallelFor(sequence, i, numProcs):
2 |   # do something with sequence[i]
```

Note: this is just a toy example showing how to do some handy metaprogramming using a lower level version of threads. The developers are working on a much better way of handling threads and parallelism at a higher level. The tutorial defines a debug statement (a variation of it which I use a lot) that shows how you can actually modify the abstract syntax tree.

C code and Memory Allocation

The typical compilation scheme in Nimrod I use is to compile to C code. They introduce a nimcache/ directory with all the C and object code for me to inspect to see how efficient it is compared to what I would write in C. It's fairly straightforward to link into C code if you must.

The C code Nimrod generates often appears indistinguishable in speed when compared to hand-crafted C code I made in certain examples. Nimrod is much more pleasurable to program in than C, and the compile-time and run-time error messages are far better than C.

Also, I'd like to note that Nimrod allows for manually allocated memory and low-level operations to provide the developer 'C-like' control. In most cases the standard libraries using the GC are appropriate, but in some cases you may want to manage your own data on the heap and Nimrod allows for this.

Young language, helpful community

The Nimrod language is young and has a handful of developers working on making it to a 1.0 release. The Nimrod community has been very helpful to me and I think it has a lot of potential.

I'm writing this post based on my experiences so far. I would really appreciate any feedback if I'm wrong or misrepresented a language I discussed. The post will be modified accordingly with acknowledgement.

Thanks to [Rob Patro](#) and the Nimrod community for useful discussions.

Getting Closer to a Star Trek Computer?

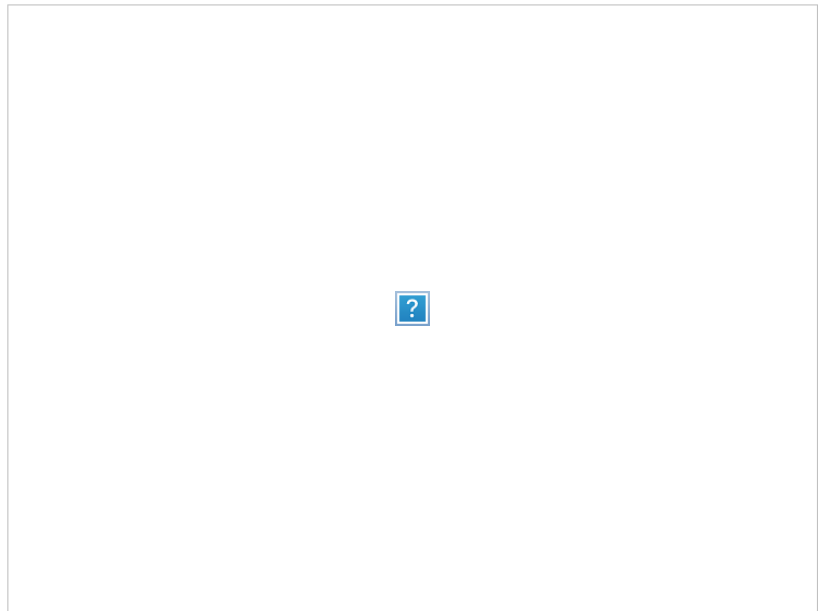
□ [December 26, 2013](#) by [Geet](#)

Reading this recent [Tech Crunch article](#) made me think that we may be a step closer to the day that we can talk to our computers and get information that we want as if we are conversing with the computer. That is, Google's [goal](#) (and arguably the goal of many other companies) of the Star Trek Computer.

Siri or Google Now isn't even close to this yet. Why? Because the problem is hard.

The article linked to above reminds me of a keynote at [KDD 2012](#) by Robin Li, Founder and Chairman of [Baidu](#), China's largest search engine. His talk was less about established problems in Machine Learning

and more about the problems that the company faces. One example is the ability to do voice recognition across the various dialects of Chinese and in the presence of background noise.



and more about the problems that the company faces. One example is the ability to do voice recognition across the various dialects of Chinese and in the presence of background noise.

In any case, I think the voice-interacting Star Trek computer is one of the largest unsolved problems in Tech. I look forward to the day that I can be driving in my car or walking the dog and say, "Computer, inform me about the history of public education in the United States ...". In a future post, I may compile a list of specific technical issues that have prevented us from this goal along with areas of research containing valiant attempts to solve parts of this problem. I welcome emails and comments from AI/Machine Learning buffs with pointers :).

Open Doors with Google Hangouts

□ [December 18, 2013](#) by [Geet](#)

In an [earlier post](#), I mentioned that active forms of communication like video chatting and telephone conversations can be pretty unnatural when you consider family and friends just 'hanging out' at home together, for instance. I also mentioned that maybe video chat can be treated in a more passive manner to encourage a more passive, natural chatting style.



Typically, video chat invitations need to be handled immediately: you either have to accept or reject the invitation to chat right away. One of my previous roommates and I had a neat system: the degree to which our doors were open indicated our openness to a 'spur of the moment' of chat.

Ideally, a similar approach could work for a video chat where the request to chat doesn't require an immediate response. You 'open your door' to other people and they will be informed. They can drop by so long as your door is open.

A nice Google Calendar feature allows you to create an event (you could even call it 'Door's open') and everyone who is listed on the event can follow a hyperlink to a Google+ Hangouts video chat. It's a subtle difference from the traditional video chat invite: instead of querying a friend and waiting for a response, you just send a link to them (e.g. via an email or text message). They can join if they want. Or not. No big deal. I think this is a more concrete suggestion for how to use tech to improve our communication with close family and friends than the high level thoughts I posted before. I'll be trying it out and posting my 'results'. Could be nice to streamline the process with an app, but this 'hack' allows me to try it out first!

Initial Thoughts on 'The Hobbit: The Desolation of Smaug'

📅 [December 15, 2013](#) by [Geet](#)

Just saw this movie. I really enjoyed the adventure, action, and suspense throughout it, and would recommend watching it in the theaters. After I saw the movie, I looked up reviews on it and agree with many others that Smaug was an awesome dragon to experience. Also, I like that it helps build up to the story told in *Lord of the Rings* and branches beyond the fun tale that I remember reading as a child.

Normally I don't write about my thoughts after a movie, but for some strange reason, I feel like I have to after this one.

Smaug and great action/adventure don't make a great story or trilogy. I was more touched by the relationship between Bruce Willis and Ben Affleck in *Armageddon* than any pair of characters so far in *The Hobbit*, maybe with the exception of Bilbo and Gollum in the first *Hobbit*. The *Lord of the Rings* (LOTR) movie trilogy is a masterpiece: it explored the a number of characters, pairs of characters, *triples*, etc. and their development together (of course *way* better than *Armageddon*). Action and adventure are great, but with just those two, I feel empty without character development. I'm surprised so many reviews focus on the adventure and action as opposed to character development, especially since it is natural to compare *The Hobbit* to LOTR. I also don't agree that this sequel is such a 'leap' above the first movie. To me, they're both about the same when it comes to what matters: the story.

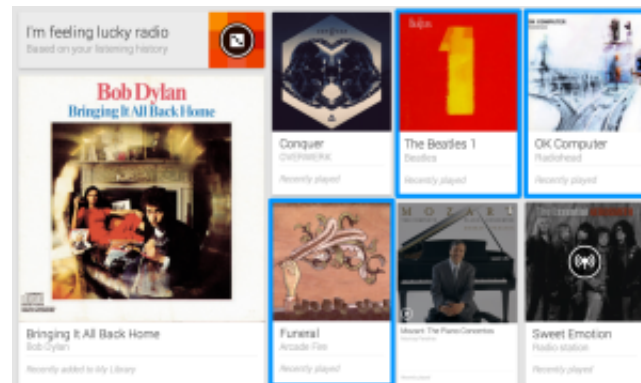
As it stands, the series so far is 'good'. It could have been 'great' by exploring a few characters in more depth (Thorin in particular). Since *The Hobbit* is such a short book, the creators of this movie series have a unique opportunity to contribute to the development of these characters by expanding a short book into three movies. They certainly have in many respects, but I argue that they should have focused on character development more. I don't mind a few poorly done special effects if what results is a better story with characters that I can relate to.



Automating Organization and 'Sticky Items'

□ December 11, 2013 by Geet

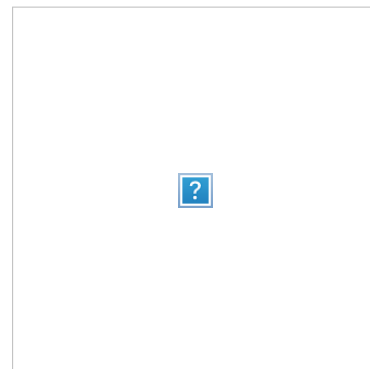
I had a nice in-person discussion after the last post. The basic idea is that there is an overwhelming amount of manual effort that goes into organizing a hierarchy. I think this is a really good point, and I really like the idea of ‘automating’ the organization process or algorithms providing suggested or ‘default’ organizations. This being said, I also like the idea of user input in the process. Even if you have a nice automated way to organize your data, what if you really want to keep something organized the way it is? This approach would allow you to access your data in a way that you specify. For example, in a snapshot of my Google Play music library above, what if I wanted the three items in blue ‘stuck’ or ‘pinned’ at those locations? That would be preferable to typing in a search bar or scrolling through a list of albums automatically sorted by some criteria. This hybrid approach allows user interfaces to provide automated suggestions for organization while at the same time letting the user keep certain things where they want them. Another example is phone apps. Right now we manually organize our apps on our home screen(s), but what if apps are organized, at least in part, by how often they are used while certain commonly used apps are ‘sticky’ because you like them there?



The Return of the Great Filesystem Hierarchy

□ December 10, 2013 by Geet

The problem: I have data of the same general ‘type’ spread across many web services and my personal files stored on my computer. For example, I have PDFs of books in a ‘Books’ directory in my file system and books from Amazon in my kindle. Or I have movies/videos on my computer, on the Play store, on Amazon, etc. There is no one place where I can see/browse my entire movie, music, or book collection across these services. I want a way to still be able to use certain apps and services while still maintaining a high-level view of where all my stuff is. For example, when I have a Kindle book, I want to still use the Amazon reader and all of its features, but for a PDF I may prefer to use OS X’s Preview or some other reader. I want a way to reconcile the fact that, for example, a ‘book’ is the type of thing I really care about organizing with the fact that applications like Amazon’s cloud reader want your ‘thing’ to be your entire library of books.



The file system is a solution: Remember the days when all of your stuff was on a single file system organized hierarchically? It’s not so convenient anymore to store all of your data in one place, and super awesome and useful cloud apps have contributed significantly to this. However, by using your file system as your primary view, I’d argue that most of your ‘things’ across can be organized again. One solution is to use file types that represent links. For example, in OS X these are ‘.webloc’

files (just drag a link from your browser to the Desktop and you will see how this looks). You can double click on one of these and it will open up the link in a browser. Between apps on my computer and cloud apps, using these links covers most of my cases (e.g. you can link to a specific book on Amazon or to an album on Google Play Music). There's even an app in Google Drive (by default using [.glink](#) extension) that properly handles [.webloc](#) files.

A natural extension to this approach is the ability to associate each thing with metadata (e.g. like a row in a database) allowing you to view your data in multiple ways.

In any case, using files and links to web apps seems to be a good interim solution to this problem (which first appeared as a [G+ post](#)).

Lenovo Yoga as a Chromebook?

□ [December 2, 2013](#) by [Geet](#)

A few weeks ago I played with this machine and was quite impressed with it. Seems like a great 'model' for a Chromebook.



State of My Gadget Union: HP Chromebook 11

□ [November 26, 2013](#) by [Geet](#)

I have made two new Googly purchases recently. The HP Chromebook 11 and the Nexus 5.

Nexus 5 thoughts soon, but after a few weeks with this Chromebook, I have found that this has effectively replaced my tablet and a surprisingly, many things I do on my Macbook Pro Retina 15". One of the reasons I wanted to try it out was that I found I often wanted to chat or write a quick email or do some basic keyboard-level content input with the tablet, but didn't want to pull out a full-fledged computer. This got me interested in the announcements for the Chromebook and the new

Microsoft Surface (or potentially a new Macbook Air 11"? That would be very tempting). So for me, while the Chromebook is more of a notebook than it is a tablet, I find that I use it more like a tablet: to quickly look up and input content.

Pros in my experience:

- **Surprisingly, this is currently my most used device.** My Macbook Pro Retina 15" is now used almost solely for larger, visually-intensive projects like creating a presentation or poster. And my tablets have become fancy remote controls lately.
- Super inexpensive
- Incredibly handy: can fit a tiny netbook/tablet bag
- Great for content input vs. a tablet
- Micro USB charging very handy: don't have to carry two chargers
- Display is really good (this is what got me interested in the first place): very nice contrast and viewing angles (even when compared to the Macbook Air)
- Font size scaling in Chrome is nice: iPad + keyboard, for example, doesn't scale fonts well, so it's hard to use when placed at a distance: i.e. towards your lap or knee vs. held up near your face. Chrome on a Macbook Air can scale fonts, but that'll run you about \$1,000: and you'll have a less appealing display at the moment.
- Very light: can easily hold it in one hand
- Nice to use for coding via SSH



Cons:

- Can be laggy/frame-droppy at times: especially when compared to a Chromebook Pixel, Mac or Google/Apple tablet experience. For a similar price, I believe the Asus and Samsung models may be faster, but the displays aren't as appealing.
- Native apps like Hangouts or Keep don't appear to support font size adjustment. I just use the browser instead.
- If you were to use this for presentations, I'm not sure how you would give one.
- Low battery life.

For the price, those are the only cons I can think of after weeks of use. This Chromebook may not have been targeted to a geek like me who already has much more powerful, beautiful machines, but it certainly has won me over. Things I'd like to see in the next iteration of entry-level Chromebooks by Google:

- Smoother. Please make this a smoother, more responsive laptop! Maybe the Asus or Samsung models are smoother, but I really liked the display on this one. I think many would pay the premium for this. One of the things I like about Apple products is that no matter what 'line' you get from them, you typically don't experience any major lag in performance for basic tasks, and I think this is a good standard for Google devices.
- Ramp up native Chrome apps to be more like a nice tablet experience. For example, make a native Youtube or Gmail App that immerses you in.

- Maybe a touchscreen like a lot of Windows models are going with (or the new Acer C720P!)?

Why Isn't 23andMe's Disclaimer Good Enough?

□ November 26, 2013 by Geet

The company 23andMe offers a different kind of 'tech' than I usually talk about on this blog, but it is meant to be a useful service for you (and for the company to collect information across many individuals). Give them \$100, and they will mail you a 'spit kit' with a tube that you literally spit in and send to it to them for analysis. They then use genotyping to determine whether you have certain variations in your DNA that are associated with your ancestry, disease, and other potentially genetically-derived traits.

[Image from 23andme.com]

I personally used their services about a year ago, mostly interested in ancestry. Honestly, I wasn't too interested in their disease associations (though I believe this will become more and more useful as we sequence more individuals and develop better computational/statistical methods to analyze the data). In the scientific community, genetic associations with diseases are still, for good reason, met with skepticism (see, for example, this recent article). This skepticism has worked its way up to politicians, resulting in various efforts to regulate laboratory developed tests. I believe genetic testing and screening has a lot of potential and am excited that so many efforts exist to provide individuals with these kind of services, but as many scientists would probably agree, we are just beginning to understand how complex traits and diseases are associated with genetic variations across individuals.

23andMe recently received a letter from the FDA to "immediately discontinue marketing the PGS [Personal Genome Service] until such time as it receives FDA marketing authorization for the device".

What surprised me when reading the letter was this statement:

Some of the uses for which PGS is intended are particularly concerning, such as assessments for BRCA-related genetic risk and drug responses (e.g., warfarin sensitivity, clopidogrel response, and 5-fluorouracil toxicity) because of the potential health consequences that could result from false positive or false negative assessments for high-risk indications such as these. For instance, if the BRCA-related risk assessment for breast or ovarian cancer reports a false positive, it could lead a patient to undergo prophylactic surgery, chemoprevention, intensive screening, or other morbidity-inducing actions, while a false negative could result in a failure to recognize an actual risk that may exist.

Perhaps because I have training in computational biology, I always assumed this information is meant to be more like a 'hint' or 'suggestion' that you might want to investigate a medical issue further with a physician who can conduct specialized follow-up tests and recommend further courses of action. For instance, I have a minor version of Beta-thalassemia that apparently doesn't require medication or affect my daily activities. 23andMe correctly identified that I have this trait. This disorder can be discovered via blood testing by a physician, so if I didn't already have these tests, my 23andMe report may have alerted me to investigate this further. I think this is useful information.

My 23andMe page also displays this disclaimer:

The genotyping services of 23andMe are performed in LabCorp's CLIA-certified laboratory. The tests have not been cleared or approved by the FDA but have been analytically validated according to CLIA standards. The information on this page is intended for research and educational purposes only, and is not for diagnostic use.

I do think that it could be made more prominent perhaps with a clause that states that some of these associations are in a stage of 'early research', but I am writing this post more to pose a general question to anyone that has some insight into the legal aspects of this recently newsworthy topic: *why isn't the disclaimer above (or a better one) good enough for the FDA?* Certainly, I understand that the general public doesn't have the same knowledge I do about genetic tests, but given a disclaimer like this, why can't 23andMe sell me their services and let me make follow-up decisions?

Some FDA links that are relevant:

- FDA device classes
- Classification of specialized medical devices

Using Tech to Enhance Remote Experiences

📅 November 16, 2013 by Geet

In an earlier post, I mentioned that, to me, one of the main benefits of tech is to "to inspire users to create and share ideas and enhance the overall experience of learning and growing as human beings". This is the first of hopefully a number of posts providing suggestions/examples of how we could do that (as opposed to play with the latest toys/features of software and hardware or browse social network streams until we drop).

One of the most essential needs for humans is the good company of family and friends. While we all also need isolation and time for reflection, the need for spending quality time with family and friends will never go away, and I have found that just being around this kind of company makes me happier and more productive.

Google Hangouts



Experiencing family and friends in person is great, but because of job opportunities or travel we are often forced to communicate remotely. What I have found is that this kind of communication is mostly 'active' with the exception of chat which is usually *very* passive. The problem with 'active' communication (e.g. phone or video chat) is that a constant stream of attention is required. I think this is what leads people to say "I'm not really a phone person", for instance: the desire to avoid awkward pauses or moments when there's not much content when the medium of communication sort of 'demands' a more active form of communication.

But when we are 'hanging out' with our family and friends in person, such an active form of communication is not expected of us. For example, when you are at home with the family it's perfectly natural for different members of the family to be reading, doing homework, etc. with the occasional interruption to share a thought or to eat. I believe that it is exactly this more relaxed form of communication that creates an atmosphere of comfort and openness to share ideas and fun experiences.

There's no reason that we can't use phone and video chat to experience a more passive, potentially more inspiring form of communication. One of the best experiences I had shortly after graduating college was sharing a video chat while watching *Arrested Development* episodes with a friend remotely. We didn't just watch the show. We spoke as if we were in the same room together and it was so fun that I still remember it fondly many years later.

It's a little odd: you have to get all parties to agree to leave their video/audio chat open while you cook/share a meal for example. It's not the norm, but I think doing more of this passive communication with family and friends would be a great use of tech that already exists: especially since we don't have to pay by the minute anymore!

Think Before You Tech

□ November 13, 2013 by Geet

First, this blog has been revived! The last post before the Nexus 7 Experience was a long time ago. Since then, I have become a Ph.D candidate, transferred from University of Maryland to Carnegie Mellon, and made a web page summarizing some of the work at my recently re-acquired domain. At first I was thinking of moving my blog over to Weebly which is what I used to host the web page. (I decided not to hand-code my web page this time and it was so much faster to get the design I wanted.) I decided to stick with WordPress because I think they do a really good job of distributing posts to a larger audience and I have had a number of useful interactions on their platform.


To inaugurate this second coming, I wanted to mention something more philosophical that has been on my mind lately, and that's the role of tech and gadgets in our lives. I occasionally post on various social media effusing my excitement on new gadgets or technologies. Are these just my toys? Short answer: YES! I enjoy playing with new tech toys. This being said, I believe it's important to emphasize a "think first" approach to new technologies. Social networks like Facebook and Twitter

are great: and new devices can make it that much easier to access all the new news feeds and information. But these feeds can also be thoughtless diversions from valuable human experiences: sharing a great conversation with someone, helping others out, and as someone I know pointed out recently on a private G+ conversation: even doing basic chores like vacuuming (this was in reference to an automatic iRobot cleaner)!

For me, the benefit of new technologies is to inspire users to create and share ideas and enhance the overall experience of learning and growing as human beings. Another benefit is to reduce needless suffering in the world (e.g. lack of shelter, hunger). I appreciate when big companies like Apple and Google focus on enhanced experiences as opposed to introducing just another new technology. For example, the recent Google Plus keynote hinted at this a little when Vic Gundotra gave the example of capturing a special moment with his children. I think we are just scraping the surface here. In the future, when I talk about my experiences with technologies and products, I will try to focus even more on the aspects I mentioned at the beginning of the paragraph.

TECHNOLOGY

Nexus 7 Experience

Video |  November 12, 2013 by Geet


Google Earth on the Nexus 7 2013



This is an awesome tablet. The Nexus 7 2012 was wildly popular. But I think this current Nexus 7 2013 model hits the sweet spot of size, performance, and cost. I would recommend it to anyone interested in getting a tablet. This post also has...

[Read More](#) 

Scala is pretty awesome

 [November 13, 2011](#) by [Geet](#)

After [hearing about it a lot](#), I have started using Scala recently and am finding it really fun and powerful to use. Scalala is a nice analog to numpy. The documentation needs to get better, but I think it's coming along.

I just found a good introductory video by the creator of the language, and if you're curious whether this is a language you're interested in playing with, I think it is a good place to start:

<http://www.youtube.com/watch?v=zqFryHC018k>

Grabbing individual colors from color maps in matplotlib

□ August 22, 2011 by Geet

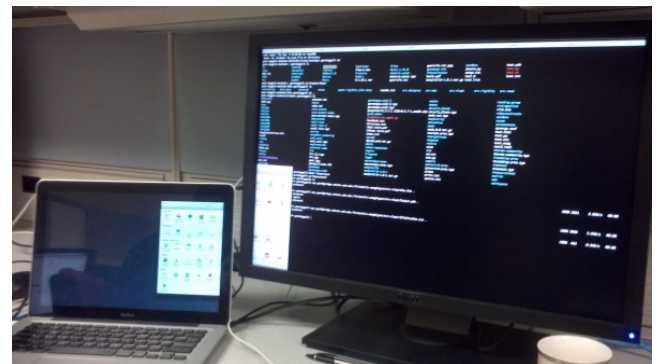
Want an individual color from a color map in matplotlib? You would think this would be easy (and it is as you will see below), but as I traversed the documentation, I could not find for the life of me how to do this. After looking at various matplotlib functions, Rob and I finally found that it's really simple (and apparently undocumented as far as we can tell). Just choose your colormap (e.g. 'jet') and call it like a function with a value between 0 and 1: e.g. `pylab.cm.jet(.5)`. This will return you the rgba 4-tuple that you can use for whatever.

Reclaiming your second monitor in OS X Lion

□ August 4, 2011 by Geet

Upgraded to Lion? Use dual monitors? Notice the full screen problem? My lab mate Rob and I found an interesting hack.

1. Open an application in full screen mode (e.g. Terminal)
2. Use three fingers on the trackpad and move *slightly* to the right or left so you are transitioning into the respective space (the 'transition position'). Hold this position.
3. While maintaining the transition position, launch another application. For example, use spotlight to launch TextEdit by pressing the apple key + space and search for TextEdit. Choose it, and press enter.
4. Still holding the transition position, wait until your application loads.
5. Let go of the transition position by lifting your three fingers.



I stumbled across this by accident while trying to open system preferences when in full screen mode as you can see in the attached picture.

Now your second screen is not a paperweight 😊

R and Python (RPy2): Rank-revealing QR Decomposition

□ January 27, 2011 by Geet

I used to use R all the time until I got into Python, NumPy, and SciPy.

In general, however, R has much more sophisticated packages for statistics (I don't think there's too much argument here). Unfortunately, R, in my opinion, is a cumbersome language for string manipulations or more general programming (though it is fairly capable).

You can get the best of both worlds with RPy.

I recently wanted to calculate the rank of a matrix via a QR decomposition as an alternative to computing the SVD. I haven't yet seen code to do this in numpy. R, on the other hand, computes this automatically in its 'qr' function. To get this running in my numpy/scipy code, I didn't even have to worry much about matrix format conversions between the languages. After a couple of 'imports' and within 5 minutes I calculated the rank in Python via the R code:

```
rqr = robjects.r['qr']  
print rqr(R)[1][0]
```

Wikipedia on your coffee table!

□ January 4, 2011 by Geet

So I recently thought I had this awesome idea that you could take the "featured articles" in Wikipedia and turn them into a book or volume of books that people could buy and just keep one on their coffee table. You may agree with me that there's a certain serendipity associated with traditional binded books or journals that is hard to replicate when browsing the web on a desktop, laptop or iPad-like device.

I was not too surprised, a little disappointed, and pretty excited to find out that this has been done. The most comical creation of a Wikipedia book has to go to Rob Matthews for actually attempting to bind all the featured articles. On a more serious front, I was happy to see that PediaPress, wiki-to-print publishing service, is partnered with the Wikimedia foundation to do exactly this. It's integrated quite well with Wikipedia so you can click on the articles you want to include in your own book or you can select books that have already been compiled by PediaPress.

As an alternative to purchasing a book, you can also generate a PDF that you can print at your own leisure. My only gripe when immediately trying this is that scalable content Latex equations are embedded in the PDFs as if they are images. It still looks decent, though, and is pretty cool!

Yet another awesome use of the Synergy app

□ November 25, 2010 by Geet

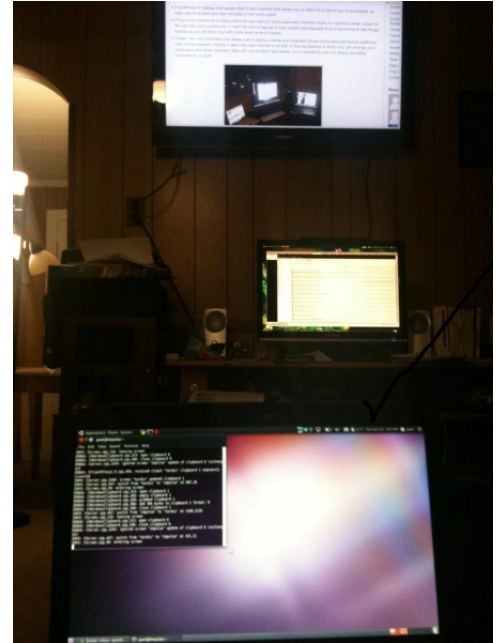
I've mentioned before that the Synergy app is awesome.

The image to the right is a configuration at my friend's house where my laptop (bottom) is controlling his dual-monitor computer setup (one regular monitor and an HD TV on top).

It works great as a remote and, yes, you can watch Hulu on it without being charged for watching Hulu on a TV device.

There are a lot of new internet TV solutions, but with just a little bit of extra work a very easy-to-control TV setup works that's far more flexible than what's on the market today.

That being said, I do own an Apple TV and what's nice about these types of devices is that they are small and focus on a simple user interface to get access to content.



Speed up your Python: Unladen vs. Shedskin vs. PyPy vs. Cython vs. C

□ November 25, 2010 by Geet

Lately I've found that prototyping code in a higher-level language like Python is more enjoyable, readable, and time-efficient than directly coding in C (against my usual instinct when coding something that I think needs to go fast). Mostly this is because of the simplicity in syntax of the higher level language as well as the fact that I'm not caught up in mundane aspects of making my code more optimized/efficient. That being said, it is still desirable to make portions of the code run efficiently and so creating a C/ctypes module is called for.

I recently created an application (I won't go into the details now) that had a portion of it that could be significantly sped up if compiled as a C module. This spawned a whole exploration into speeding up my Python code (ideally while making minimal modifications to it).

I created a C module directly, used the Shedskin compiler to compile my Python code into C++, and tried the JIT solutions PyPy and Unladen Swallow. The time results for running the first few iteration for this application were surprising to me:

cpython:	59.174s
shedskin:	1m18.428s
c-stl:	12.515s
pypy:	10.316s
unladen:	44.050s
cython:	39.824

While this is not an exhaustive test, PyPy consistently beats a handwritten module using C++ and STL! Moreover, PyPy required little modification to my source (itertools had some issues) [1]. I'm surprised that Uladen and Shedskin took so long (all code was compiled at O3 optimization and run on multiple systems to make sure the performance numbers were relatively consistent).

Apparently out-of-the-box solutions these days can offer nearly a 10x improvement over default Python for a particular app. and I wonder what aspects of PyPy's system accounts for this large performance improvement (their JIT implementation?).

[1] Uladen required no modifications to my program to run properly and Shedskin required quite a few to get going. Of course, creating a C-based version took a moment :-).

Update 1: Thanks for the comments below. I added Cython, re-ran the analysis, and emailed off the source to those who were interested.

Update 2: The main meat of the code is a nested for loop that does string slicing and comparisons and it turns out that it's in the slicing and comparisons that was the bottleneck for Shedskin. The new numbers are below with a faster matching function for all tests (note that this kind of addition requires call 'code twiddling', where we find ourselves fiddling with a very straightforward, readable set of statements to gain efficiency).

cpython:	59.593s
shedskin0.6:	8.602s
shedskin0.7:	3.332s
c-stl:	1.423s
pypy:	8.947s
unladen:	29.163s
cython:	26.486s (3.5s after adding a few types)

So C comes out the winner here, but Shedskin and Cython are quite competitive. PyPy's JIT performance is impressive and I've been scrolling through some of the blog entries on their website to learn more about why this could be. Thanks to Mark (Shedskin) and Maciej (PyPy) for their comments in general and to Mark for profiling the various Shedskin versions himself and providing a matching function. It would be interesting to see if the developers of Unladen and Cython have some suggestions for improvement.

I also think it's important not to look at this comparison as a 'bake-off' to see which one is better. PyPy is doing some very different things than Shedskin, for example. Which one you use at this point will likely be highly dependent on the application and your urge to create more optimized code. I think in general hand-writing C code and code-twiddling it will almost always get faster results, but this comes at the cost of time and headache. In the meanwhile, the folks behind these tools are making it more feasible to take our Python code and optimize it right out of the box.

Update 3: I also added (per request below :-)) just a few basic 'cdef's and types to my Cython version. It does a lot better, getting about 3.5s on average per run!

A couple of recent Python tidbits

□ November 2, 2010 by Geet

Five years ago the two most practical and used computer languages for me were C and Perl.

While I still use C for the nitty-gritty stuff that needs to be fast, I'm finding that a lot of stuff can get done, and get done very fast in a scripting language. Over a year ago I learned the wonders of Ruby (which, to me, is basically a superior replacement to Perl from a 'it's fun to code in' perspective and is easy to transition to from Perl).

But overwhelmingly, I've found myself enjoying and using Python. The biggest selling point from my perspective is the high quality scientific computing and plotting support (which in many cases has replaced my use of R-project for these types of things).

Here are three little tidbits that I've recently found handy and take virtually no effort to begin using:

(1) First, to speed up the things that need the speed, calling C functions from Python is super-handy. I really like the ctypes module because in many cases, as long as your C functions take the default types as inputs, you can simply expose your function via a dynamic library with no real extra effort.

(2) List comprehensions are a fun and very useful feature in Python. But many times, you don't want to create the whole list before iterating through it. Here's where generator expressions come in.

Instead of brackets around your comprehension, just put parens, and suddenly you can iterate over these elements without creating the entire list in advance (especially useful with a module like itertools).

(3) Finally, one thing that had bugged me a little in Python, especially since I'm used to C, is that there really is no real scanf equivalent [1]. What I end up doing is parsing out my variables from a string and on separate lines explicitly setting their type. This just takes too many lines (and I could often do it in fewer lines in C)! After some thought, a lab mate and I converged on:

```
>>> types = [float, str, int]
```

```
>>> fields = ['3.14159', 'Hello', 10]
```

```
>>> pi, hi, ten = [f(x) for f,x in zip(types,fields)]
```

[1] Note that while there is a suggestion in the Python docs on how to do this, it just suggests how to extract different types with regular expressions, not concisely convert them.

PubCasts for Journals

□ September 12, 2010 by Geet

To practice a presentation I recently gave on a paper, I recorded myself speaking (an odd experience).

It seems that a very common way people hear about research is through presentations, so why not take this to the next level and post a nice screencast with an article you publish?

Well, this actually happens quite a bit, and after a recent lunch conversation, I was motivated to look more into the sources of "screencasts for papers" that I have watched in the past.

While I think many students and researchers are familiar with video lectures (such as MIT's opencourseware or videlectures.net), one form of screencasts that focuses on research articles stood out to me as having technical enough content to properly motivate a full research paper: SciVee.

This website promotes YouTube-like screencasts of research papers and has coined the term "PubCast".

I'm definitely all for this, so what's the problem?

Thankfully, due in large part to the open access movement, I find myself visiting the journal sites themselves more and more for the content and supplementary information. Especially for open access journals like PLoS, researchers have less of a reason to post a PDF version of their article on their web page. Rather, why not just link directly to the open access content which makes the article visible in HTML and PDF?

While SciVee is partnered with PLoS (awesome), I noticed that the associated PLoS pages never link to the SciVee video (as far as I can tell). Moreover, PLoS Bio and Comp Bio videos seem to stop existing on SciVee after around 2008 or so.

This is a shame, because even if there's a lot of great video content out there associated with these papers, no one knows about it if they visit the journal's home page. Why not just stick the video link here?:



I think the basic idea is out there and there are some early-adopters of this technique (not necessarily limited to SciVee's services, of course). What I think it needs is better marketing and accessibility from journal websites. Intuitively, it seems like well-done screencasts promoted by the journals themselves (perhaps even made a part of the editorial process) could really be good for getting the journal and its papers more attention.

Mendeley: Try It

□ June 27, 2010 by Geet

Mendeley is pretty cool.

I've been using it for a week or so and it's definitely integrated itself well without being obtrusive.

Some things I really like:

- Cross-platform: OS X, Linux, Windows
- Drag-and-drop your PDFs and it figures out the metadata as best as it can. If it gets something slightly off, you press a button and a Google scholar check usually fixes it all up
- PDF viewing is quite speedy and natural for me with their tabbed interface
- Notes, highlights, etc: syncs with the cloud and can be integrated with other users that you share your documents with
- I find storing associated URLs to be a very handy feature (links to the web site, supplementary info, etc.)
- Cleanly store your PDFs in a directory of your choice (e.g. I use Dropbox to view them on my iPad)

Requests (maybe I can request some of these?):

- iPad app (supposedly on the way)*
- Sometimes highlighting/selecting text isn't as accurate and intuitive as it is in OS X's Preview application, which does selection brilliantly

- Parse PDFs for their citations and do *cool* things with this. What can you tell me about papers I may be interested in by looking at the citation network of my collection?
- If Mendeley is like iTunes for music, it would be nice to have a slick storefront to browse related papers of interest and import.

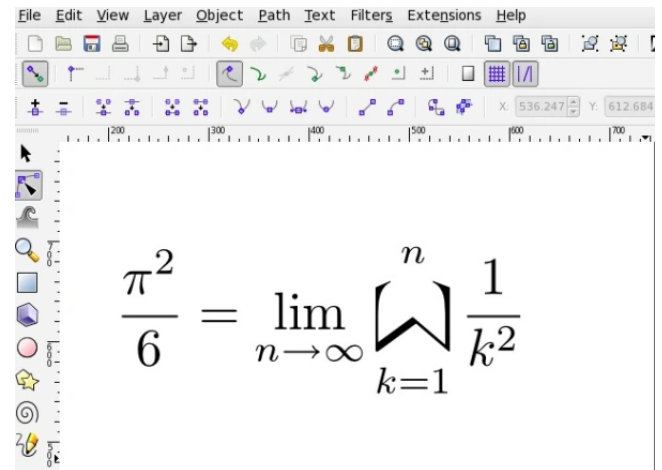
* Related posts

Inkscape and Latex

□ June 13, 2010 by Geet

I've been a fan of the open source Inkscape for some time now, especially for lower-level vector graphics drawing. But for diagramming purposes, I've tended to use OmniGraffle (made only for the Mac). I've found it pretty handy to use OmniGraffle with the program LatexIt (which comes bundled with the MacTex distribution).

Little did I know that Inkscape comes with *batteries included* Latex rendering support. And with its "connectors" tool, Inkscape is a very competitive alternative to diagramming that is cross-platform and is open source. For Latex rendering, it converts your favorite Latex equation into SVG (via Extensions->Render->LaTeX Formula).



$$\frac{\pi^2}{6} = \lim_{n \rightarrow \infty} \left(\sum_{k=1}^n \frac{1}{k^2} \right)$$

As you can see from the image, I rendered an equation and then rotated the summation symbol 90 degrees counter-clockwise since it is just another SVG object to play with in Inkscape.

This is a powerful feature that comes with the Inkscape distribution, but unfortunately you may not see it in your menu. You can Google around and figure how to get this to work based on various forum posts (though depending on your setup this may take a while).

Because it can be kind of a pain to figure out how to get this "default" feature to work properly, I thought I'd explain it for Ubuntu and OS X in one place so it would be potentially easier for others to get it going.

First of all, by "default" or "batteries included", I mean that this is a Python extension that is included by default in the Inkscape software distribution. According to the Python file itself, for the plugin to work properly:

eqtexsvg.py

functions for converting LaTeX equation string into SVG path

This extension need, to work properly:

– a TeX/LaTeX distribution (MiKTeX ...)

– pstoeedit software: <<http://www.pstoedit.net/pstoedit>>

eqtexsvg.py functions for converting LaTeX equation string into SVG path This extension need, to work properly: – a TeX/LaTeX distribution (MiKTeX ...) – pstoeedit software:

<<http://www.pstoedit.net/pstoedit>>

I'm assuming you have a Latex package installed (e.g. on Ubuntu, something like 'texlive-full' or 'lyx' or on OS X, the MacTex distribution).

The plugin basically takes the equation you feed in, runs latex and dvips on it to create a Postscript file. The real meat comes in the program 'pstoedit': it converts your postscript file to SVG.

But you have to make sure this program is installed properly. If you try to install it from source with the default settings, it may not work because for SVG output in pstoeedit, you need the GNU plotutils library.

The easy solution for both of these is to install pstoeedit from a package repository such as apt on Ubuntu and MacPorts on OS X and it should depend on plotutils. Unfortunately, the install on OS X may take some time because a larger list of dependencies are actually all compiled before pstoeedit itself is installed.

But after you install these packages, restart Inkscape and the plugin should show up in the menu and work.

Open Access and Keyboard Cats

□ May 11, 2010 by Geet

I thought this was especially funny:





WHO'S GONNA PAY FOR IT?

HERE IS WHAT I FOUND OUT:

IN 2001, SEVERAL PROMINENT SCIENTISTS PUBLISH AN OPEN LETTER THAT PROCLAIMS:

"SCIENTIFIC RESEARCH AND IDEAS SHOULD NEITHER BE OWNED NOR CONTROLLED BY PUBLISHERS. THEY SHOULD BELONG TO THE PUBLIC AND BE FREELY AVAILABLE."



34,000 SCIENTISTS SIGN THE PETITION.

SHORTLY BEFORE, VITEK TRACZ, FOUNDER OF BIOMED CENTRAL, ENVISIONS AN ONLINE OPEN REPOSITORY OF PAPERS:

SCIENCE CAN'T FUNCTION EFFICIENTLY ANYMORE WITHOUT OPEN, UNRESTRICTED ACCESS. IT IS INEVITABLE.

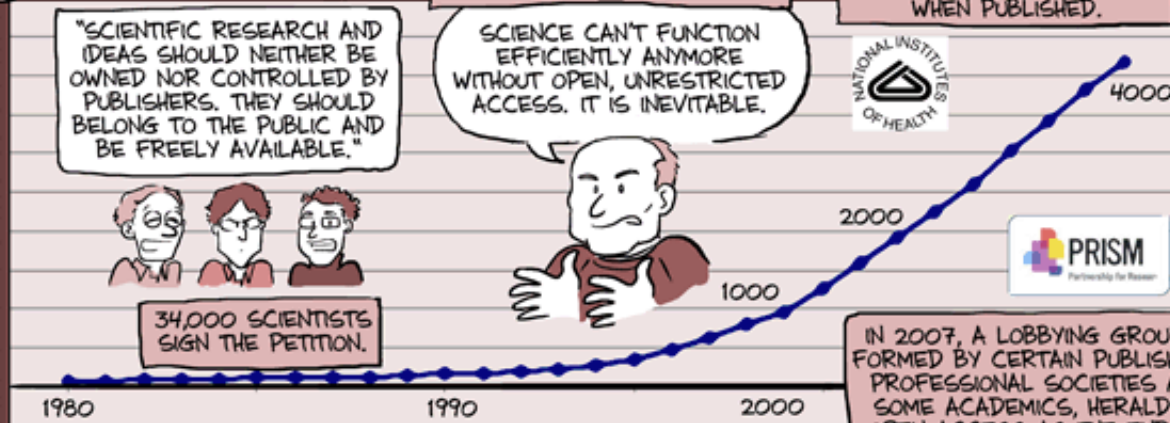


IN 2005, THE NATIONAL INST. OF HEALTH MANDATES THAT ALL NIH-FUNDED RESEARCH BE FREELY ACCESSIBLE WHEN PUBLISHED.



IN 2007, A LOBBYING GROUP IS FORMED BY CERTAIN PUBLISHERS, PROFESSIONAL SOCIETIES AND SOME ACADEMICS, HERALDING OPEN ACCESS AS THE END OF SCIENTIFIC INTEGRITY.

NUMBER OF OPEN ACCESS JOURNALS



Source: Directory of Open Access Journals

USING MY EXTENSIVE CONTACT NETWORK (READ: TWITTER), I LAND AN INTERVIEW WITH PETE BINFIELD, MANAGING EDITOR OF "PLOS ONE", CURRENTLY THE LARGEST OPEN ACCESS JOURNAL ON EARTH.

PETE IS A 15 YEAR VETERAN OF THE PUBLISHING BUSINESS.

ALL THAT TIME, I FELT THERE WAS SOMETHING WRONG WITH THE CURRENT SYSTEM.

"THERE IS SO MUCH WASTE IN THE SUBMISSION PROCESS AND IN THE MARKETING AND SALES DEPARTMENTS OF JOURNALS...

...THERE HAS TO BE A MORE INTELLIGENT WAY TO DO THINGS."

SCIENCE?

THIS ISN'T "GOTCHA" JOURNALISM, IS IT?

YOU'RE NOT GONNA QUOTE ME OUT OF CONTEXT, ARE YOU?

PEOPLE ARE AFRAID TO TALK TO ME?



PLOS ONE TAKES A STEP FURTHER AND ONLY PEER-REVIEWS FOR TECHNICAL CORRECTNESS, NOT "IMPACT". THEY USE THE WEB TO RANK EACH PAPER.

THE #1 PAPER THIS YEAR IS...

KEYBOARD CAT?

WHY SHOULD A FEW INDIVIDUALS AT THE TOP JOURNALS HAVE ALL THE POWER?

LET THE WORLD DECIDE WHAT IS SIGNIFICANT!

WILL SUBSCRIPTION JOURNALS SURVIVE? CAN OPEN ACCESS PUBLISHERS STAY AFLOAT? WHAT'S GONNA HAPPEN TO SCIENTIFIC PUBLISHING??

To download the full story, please sign-in to your subscription account or make a payment.

Login Required

User Name:

Password:

GO

[Forgotten username or password?](#)

[Don't really care? Sign out Here](#)



JORGE CHAM © 2009

Especially since I recently posted about a keyboard cat.

Quickly Creating Ajax Web Demos

□ May 6, 2010 by Geet

In about 50 lines of Python code, you can create the *skeleton* for a web demo that uses Ajax. This way you can have an interactive demo with HTML links, access to your executables and scripts, etc.; but you're now free from the system restrictions of JavaScript (which exist for security reasons).

The reason why the 50 lines is impressive is that in order to free yourself from the system restrictions of JavaScript, your web page needs to communicate data back and forth with a web server, and potentially with a database if you have lots of persistent data. Usually you have to install and configure all sorts of servers and settings to get this working.

Since I want my demo to exist within a self-contained directory, Python's batteries included philosophy comes to the rescue. With WSGI and SQLite, I can (1) create a minimalistic web server that delivers a JavaScript demo and can (2) select what to display from potentially massive amounts of data.

It's just one file that runs the server. I can then open my browser to "localhost:8000" and the demo can proceed. Of course, I just placed a skeleton below. You can make things more fancy regarding Ajax and JSON as well as by using more heavy-weight WSGI implementations than wsgiref.

Update: a Vim 'TOhtml' issue was fixed in the code below

```
from wsgiref.simple_server import make_server
```

```
ajax_html = """
```

```
<html>
```

```
<head>
```

```
<title>AJAX + wsgiref Demo</title>
```

```
<script language="Javascript">
```

```
function ajax_send()
```

```
{
    hr = new XMLHttpRequest();

    hr.open("POST", "/", true);
    hr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
```



```

    hr.onreadystatechange = function()
    {
        if (hr.readyState == 4)
            document.getElementById("result").innerHTML =
                hr.responseText;
        }
        hr.send(document.f.word.value);
    }
</script>
</head>

<body>
<center>
<form name="f" onsubmit="ajax_send(); return false;">
    <p>
        <input name="word" type="text">
        <input value="Do It" type="submit">
    </p>
    <div id="result"></div>
</form>
</center>
</body>
</html>
"""

```

```

def intact_app(environ, start_response):
    if environ["REQUEST_METHOD"] == "POST":
        start_response("200 OK", [("content-type", "text/html")])
        clen = int(environ["CONTENT_LENGTH"])
        return [environ["wsgi.input"].read(clen)]

    else:
        start_response("200 OK", [("content-type", "text/html")])
        return [ajax_html]

httpd = make_server("", 8000, intact_app)
httpd.serve_forever()

```

When planning your next vacation

□ April 27, 2010 by Geet

know that it's now possible to visit every bridge in current day Koenigsberg exactly once (previously this was impossible).

Changing just the math fonts in Beamer

□ April 18, 2010 by Geet

I really like beamer, but sometimes like the normal math fonts (rather than the sans default in beamer) for *only* the math portions. This is how you do it:

```
\renewcommand\mathfamilydefault{\rmdefault}
```

Uh oh, it's begun ...

□ February 4, 2010 by Geet

I just had the urge to sit down *without* my laptop and check out something on the net. And I thought, "it would be nice to browse in a space larger than a smart phone." I didn't want to code, type much on a keyboard+play with a mouse, or sit upright staring at my laptop.

It does seem like the iPad has gotten an unnecessarily bad reputation given that it's exploring a rather different space of human-computer interaction than your usual PC or smart phone. I was actually less interested in the MacBook Air, which seemed like a slicker, lighter-weight computer that I would feel less inclined to run my code on because it's significantly slower than my current 13" MacBook (which I paid much less for).

But the notion of a tablet that I use solely for internet and communication + some additional bonus stuff sounds not-so-bad. I'm tempted to get one just to see how much I would use it. I did this with a NetBook. I figured, being a computer-geek, that \$250 is worth seeing whether the thing improves my life. It turns out I didn't really find a NetBook that handy. Part of why I feel I didn't benefit much from the NetBook is similar to the reason I wasn't interested in the MacBook Air — it's still a computer that I open up, type on, and use a mouse — so the weight/slickness benefit doesn't outweigh the sacrifice in performance compared to my current laptop. The iPad seems to have some benefits over an e-Book reader in the sense that it allows a more interactive experience with the display that can show web pages like you would see them on your laptop. This, in combination with the fact that I lose the keyboard and mouse, might be enough to make it a useful secondary computing device.

In case I need a reminder

□ February 3, 2010 by Geet

that I'm a nerd ... GMail suggests:

Add to calendar

$\Pr[X=1] = 3/6$

Thu Jan 6, 2011 – add

I know it's been a while, but I'll be back shortly ...

Flex and Bison Fun

□ August 13, 2009 by Geet

So I've been writing parsers lately with Flex and Bison.

Some notes of interest:

- The “info” pages on both are really helpful and I've learned most of what I needed from them (as opposed to Googling around or finding a book)
- The Regular Expression Explorer has been quite handy
- While learning, I wrote a bunch of toy parsers, and I realized that I was copying “template code” a lot. This seemed unnecessary. A tool like this simply takes a BNF-like input and outputs into different languages. Seems pretty cool but haven't played with it too much yet.

Fun.

I've Spent the Last Fifteen Minutes

□ August 8, 2009 by Geet

YouTubing Nora the piano cat thanks to this video:

CATcerto. ORIGINAL PERFORMANCE. Mindaugas Piecaitis, Nora The Piano Cat

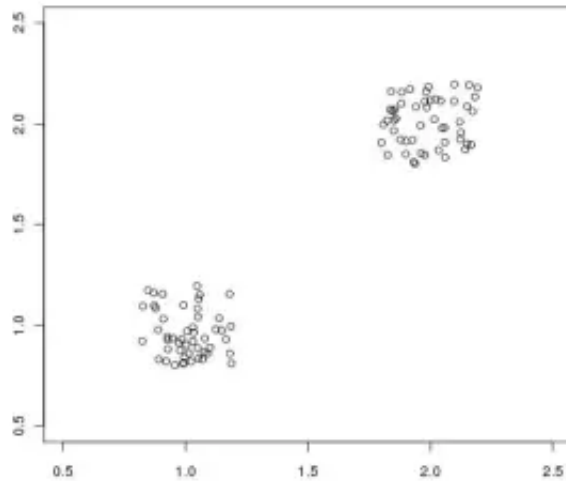


via Erik.

Is clustering just an art?

□ April 14, 2009 by Geet

Clustering data is the practice of grouping or partitioning a set of points based on how similar they are to one another. One of the most basic forms of this would be by eye when looking at a plot of data. For example, if we see two clumps on the x and y axes, then we say these data are grouped into two clusters.



```
plot(matrix(ncol=2,byrow=TRUE,jitter(c(rep(c(1,1),50),rep(c(2,2),50)))), xlim=c(.5,2.5),
      ylim=c(0.5,2.5),xlab=NA, ylab=NA)
```

Here, I artificially created two clusters in the R project tool grouped around (1,1), and (2,2). The job of clustering tools is to figure out that these are indeed clusters. This becomes harder when dealing with more than two dimensions, different types of data, and the question of what a cluster actually is.

Clustering tools are often used as a form of exploratory data analysis and fit within the class of unsupervised learning techniques. It's not hard to see their practical relevance when they can help form groups from data elements having many different properties.

For some time now, I've heard the phrase "clustering is an art not a science." This becomes even more evident when faced with the frustrating quantity of clustering algorithms for specialized tasks and intimate interaction with the data to realize what's actually happening. Moreover, I can't help sneaking suspicion on how certain techniques seem to overlap with one another. Now I'm not trying to knock specialized techniques. In fact, quite the opposite. For the thoughtful ones that seem to tackle a particular class of data, *at least they do that and can be used*. That being said, without being able to formally lay out what's actually happening, it's hard to have a sound basis on which to compare methods.

What I'd like to (quickly) highlight in this post is a couple of interesting notions that tickle the possibility of a cleaner way to think about clustering. I'm not going to focus on the usual classifications of algorithmic approaches (e.g. hierarchical vs. partitioning). While those distinctions are useful in choosing techniques and tools to cluster with, I think it might be valuable to look at the problem without concerning ourselves too much with traditional algorithms.

A data point can be quite flexible. It's just a tuple. It can even be of mixed types (e.g. the first dimension can be some point in the real numbers, the second can be one of n nominal elements of a set, ...).

All that matters is that we can define the distance between any two tuples. Even with mixed types, there are reasonable ways to approach this [1]. With this, we have a metric space.

We can stick this metric space in a Euclidean space. In the mid-80s, it was shown that any metric space with n elements (in our case tuples) can be embedded in a $O(\log n)$ -dimensional Euclidean space with some reasonable constraints on *distortion* [2]. Distortion is essentially the discrepancy between the true distances in the n element metric space and the distances in the Euclidean space. A decade later, randomized algorithms were developed to actually do this (e.g [3]).

The curse of dimensionality is a problem where the higher we go in dimension, the more likely it is for points to hover around the edges of some abstract hypersphere of our cluster. This is because as the dimension increases, most of the volume of the hypersphere is contained in some small distance from the edge of the sphere. Say our tuple was just a vector in the real number space, then it may be possible to reduce the dimensionality considerably and not bias our cluster shapes too much due to high dimension.

These facts (the first reference being quite practical and the last two more foundational) are quite relevant to clustering and I'd argue make the whole thing a little more of "a science" (eh hem, "a math").

There are, I feel, two big issues I've ignored 'till yet, and these seem to really make up the "art" part.

1. We still have to take these points in some lower-dimensional Euclidean space and partition them into groups. This requires some assumption of what a cluster actually is. We can assume they're spherical, but that's likely not always the case. What about different types of shapes? While clustering shape sounds minor and in some cases may not be practically that relevant, to my knowledge, most ways of defining distances between clusters tend to restrict all the clusters to relatively homogeneous shapes. This seems like a problem. Maybe it can be dealt with mathematically in some way?

2. Let's go back to the start where I talk about defining distances between points. This can be a potentially arbitrary process depending on the methods by which the data was gotten in the first place. But even aside from that, the structure one is clustering on may suggest a variety of alternate ways to define distance between two points. For example, we can define the distance between two nodes on a graph in many ways (not just the shortest path).

While for the moment I'm certainly more at home as a "user/developer" of tools in this regard, here's to theory.

[1] Kaufman L and Rousseeuw PJ. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons (1990).

[2] Bourgain J. "On Lipschitz embedding of finite metric spaces in Hilbert space". *Israel Journal of Math* 52, 46-52 (1985).

[3] Linial N, London E, and Rabinovich U. "The geometry of graphs and some of its algorithmic applications". *Combinatorica* 15, 215-245 (1995).

Git on that train

□ February 14, 2009 by Geet

Warning: As the title suggests, I'm on a train.

In the last week and a half or so I've been using Git for a project amongst coworkers and most recently for my own code and text files. I was a bit skeptical. But after going through their excellent documentation, seeing the videos, and most importantly, a lot of tinkering, I'm realizing that it's making life better for me.

There are a lot of resources that compare SCMs, so I don't want to worry here about which is better and why. But I'd like to share two things that I've really liked about using Git.

1. First off, using something like Git doesn't necessarily have to be for a big collaborative project. This may be a sort of different take than the usual, but **I like to see it as a tool that helps me see different "views" of my files depending on the job.** Anyone who's written a lot of text or code realizes that it's actually quite hard to make things as modular as one would like and that sometimes we're relegated to grabbing snippets of text here and there rather than making black boxes. While this is not encouraged as an engineering practice, it's sometimes very useful for play. With Git, I'm less concerned about making all my source code fit in a super-consistent modular framework and more concerned with focusing on doing a task cleanly. This is possible because I can branch and merge with relative ease (which I understand is a pain in the ass with other SCMs). In a branch I can move some files around and do whatever I want without affecting the source. I can then cherry-pick commits I like from that into another branch. These branches can look wholly different, but the code can still be updated. This flexibility makes me worry less about organization and directory structure and more about just choosing the right, minimalistic view for the job. I now see things in terms of diffs and commits and Git provides the machinery to do real work with them.
2. **Git is minimalistic, local, and fast.** Which is great. It's a small source code base that compiles quickly and gives me handy command line utilities. Proper use of them = power (though even with good documentation there's a learning curve involved). Unlike a lot of SCMs, Git is designed to be local. While it *can* do a lot of stuff over the network, it's *modus operandi* is in a local repository (which is just one .git/ directory in your root directory). I, in fact, don't even use the SSH/SSL features layered on top. Git helps me realize what I've changed and worked on and I build patches from there. You can email them to whoever, and applying them is easy.

And if I'm frustrated with some apparent inadequacy, it's likely I can find some post with Linus himself justifying it with a little intelligence (e.g.).

Handy Makefile

□ February 14, 2009 by Geet

The following is a snippet for a handy and concise Makefile that will make executables for all C files in a directory. It's good for "sandboxing" this illustrates some of Make's useful features without before consulting a [larger resource](#). Tip of the hat to [Erik](#) for helping me polish it up.

```
# Flags for gcc
FLAGS = -D_GNU_SOURCE -O3 -g

# All C files as sources, and chop off the .c for targets
SOURCES = $(wildcard *.c)

TARGETS = $(patsubst %.c, %, $(SOURCES))

all: $(TARGETS)

# All targets without an extension depend on their .c files
%: %.c
    @echo "Building $@"
    @gcc $(FLAGS) $< -o $@

# The "@" symbol suppresses Make actually displaying the command.

clean:
    @echo "Removing hidden files"
    @rm -rf *.swp *.dSYM .*_* 2> /dev/null
    @echo "Removing executables"
    @rm -rf $(TARGETS) 2> /dev/null
```

The nice thing about Make is that it's useful not only for things like C code. I've even used it (quite some time ago) to piece together tracks of music using [ecasound](#).

Wikipedia-like blog entries?

□ February 10, 2009 by Geet

This is a sort-of half-baked notion that may have some flame war started somewhere on the net, but I can't resist.

The combination of a noble call for better Wikipedia articles in a specific subject with a recent example of someone blogging to correct his own entry makes me fantasize: what if individuals wrote blog-like entries that were, in effect, articles like those on Wikipedia?

Why not just create an HTML page or contribute to Wikipedia? As far as contributing to Wikipedia, don't get me wrong: I think that's great. But as outlined in linked "noble call", there are some legitimate issues—most notably that of authorship. Blogging allows one to retain authorship and control but, unlike HTML, it facilitates commenting and allows RSS updates to interested readers and contributors.

In the second link, the creator of Bittorent is correcting his own Wikipedia entry. This makes sense at some level for the sake of clarification and posterity. There's no guarantee that if he edits his own entry (say anonymously), the changes will persist, and it may not be appealing to constantly keep watch of minor changes to articles either.

In the days of web one point yore (say before Wikipedia and Wolfram's Math/Physics pages), if I Googled "Maxwell's equations", any page put up by some schmuck like me would probably be dicey at best. But now, I can just feel lucky and get some pretty decent information from Wikipedia.

But due to the number of people editing the entry, there is a certain lack of authorship and potentially unwillingness to even contribute in the first place.

These days, we see lots of academic bloggers posting their lecture notes on specific topics online (some even making the lecture notes themselves posts rather than linking to PDFs), so parts of me feel like we're already seeing this kind of behavior, though lectures tend to be less encyclopedic.

The thing I like about this approach is that it's decentralized and personable in the sense that we can use the fact that we trust/value certain people's discourse more than others to our advantage. Potential downsides: (1) we're that much more reliant on a YourFavoriteArticleRank algorithm to rank the articles for us (for example, when you link to another article, do you link to your favorite one? the top-ranked one? eh hem, Wikipedia?), (2) more people/groups would need to get blogs, and (3) the notion of a "collaborative article" is substituted with a "first author" (the writer of the post) and "contributors" (commenters). But those concerns don't seem that dire.

Science in the recent stimulus

□ February 7, 2009 by Geet

Prodded by [prof Tao's post](#), I spent a little too much time sifting through official documents on the congressional budget as well as other articles related to NSF funding and the stimulus amendment.

Jake Young at Pure Pedantry has a nice [post](#) discussing the issue and outlines recent attention given to this subject by *Science* and *Nature*. Something I thought I'd forward on....

</politic>

Column-based text editing

□ [December 7, 2008](#) by [Geet](#)

Many text editors have a “block” or “column” mode where items can be copied, pasted, and edited vertically rather than horizontally. I’ve never come across an editor (or something like a Vi mod) that allows actual column-based editing. This implies no scrolling up or down. When you reach the bottom of the screen you go back to the top like a two-column paper. This would imply scrolling left or right. This mode would seem *somewhat* natural for a variety of reasons: it’s often easier to read lines of code and sentences when there is a more limited width (and this can also be more space-efficient for wider screens):



Assuming no explicit line-breaks within textual paragraphs and code that can fit within the column limits, I wonder if there are some other major issues with this...

Caribbean Christmas Tree

□ [December 2, 2008](#) by [Geet](#)

I'm not an avid traveler, but occasionally I'm prodded from where my rump rests to go somewhere. This time, my cousin who was born and raised in Trinidad left his Seattle-Google residence to get married in his home country.

On the trip, I had the chance to visit a bird sanctuary where the Scarlet Ibis is the main attraction. One of the neat things was that in the evening time, all the birds congregate around a similar area to sleep. On our tour, it was a mini island rock with a bunch of trees that looked much like this:



Pretty sharp contrast. Guess they do Christmas a little differently there.

A fun electromagnetic induction demo

□ November 22, 2008 by Geet

A couple of days ago, I had a lunchtime-preoccupation with antennas and electromagnetic induction. We observe a neat phenomenon if you put two wires near each other, and Maxwell stated the phenomenon quite eloquently in his treatise on electromagnetism. Basically, the instant you change the voltage across one of the wires, the wire next to it (but not touching it) responds with a current that quickly goes away.

By wiggling around the voltage in certain patterns, we can find interesting ways to transmit information back and forth between one another, and through induction we can do this without direct physical contact. These days it's easy to take for granted the all the internets sent to us via WiFi and forget about how highly developed an area radio is.

But to get taste of the physical communication, not much is required: basically some power sources and wires. A wire can in fact be practically used as an antenna for certain radio frequencies.

A simple way to check out induction with a laptop is to see if you can transmit a song that usually plays from your speaker through the air and back to yourself. Since the sound card does both audio input and output as well as the analog-to-digital conversion, a lot of the hairy parts are taken care of for us.

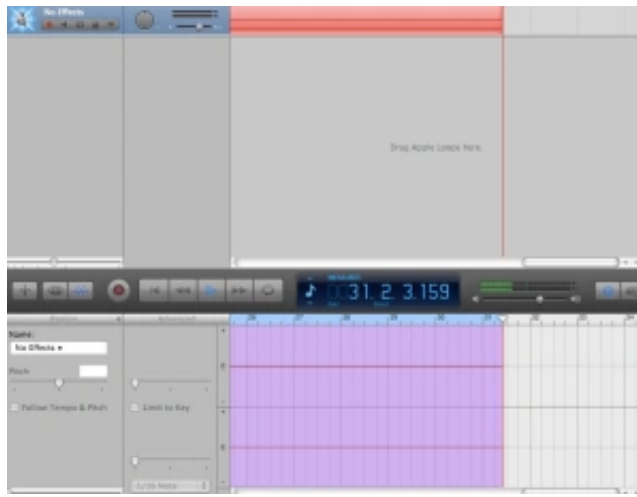
To do this, we need to have the sound output (i.e. from your headphone audio jack) go to a wire. Next to that wire is another wire that goes into the microphone or “line in” audio jack. Keep in mind, these are just wires—that’s all. I just forced myself to part with the headphone that came with my music player and another that I got on an airline flight.



As you can see, the ends of the two headphones lay prostrate next to the two wires connected into the microphone and headphone jacks:



So, now all you have to do is start up some music and record the input with some audio recording program (I used GarageBand since I’m on a Mac):



The green bars towards the middle-right wiggle around while recording indicating that the wire is receiving *something*. Doing the same thing without the wires in the line in and out yields no green bars wiggling around.

Because there's so much power loss, the signal is faint, so I digitally amplified it some in the software. But horray! I rather shittily transmitted sound back to myself over air via just two wires. This, I think, is a neat way to illustrate induction. Listen for yourself. You'll need to turn up your volume a bit.

MacBook and 64 bit address spaces

□ November 17, 2008 by Geet

I occasionally like to show off how I can do some pretty sophisticated calculations and experiments right on my Macbook. For these tasks, over-sped machines and special purpose hardware are sometimes assumed to be the right solution, but a judicious use of the resources on a MacBook may get the job done in a similar time.

One thing that would be nice is if the OS actually supported 64 bit address spaces (after all, the Core 2 Duo processor is a 64 bit one). This comes in handy when one wants to mmap large files (e.g. greater than a few gigabytes). On 32-bit architectures, the address space is limited by the word size (in this case 2^{32} bits or about 4 GB of space and even less-so with kernel limitations).

Supposedly the next version of OS X will have support for this (and will come on the MacBook Pro installations, but I assume not the MacBook).

Histograms and density estimation

□ November 15, 2008 by Geet

It is a common practice in the statistical analysis of data to group the samples into bins or categories. The goal is usually to extract some structure that is useful and often hidden from us when looking at direct plots of the sample set.

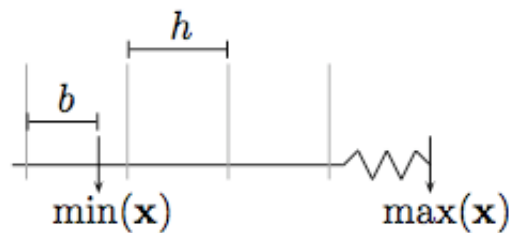
For instance, a time series plot of the change in stock price may appear erratic and contain no order, but observing the frequency of certain price-changes may help us realize that on average there is no change as well as give us an idea as to the practical bounds to how much the price will change.

The purpose of this note is to speak a little to the role of histograms in statistics and probability. While a variety of complications arise when using them for the estimation of a probability density function, they serve as a very practical (and efficient) tool in exploratory data analysis and provide a grounding for other forms of density estimation.

Histogram Construction

- For a given a sample, what bin does it belong to?
- How many samples are in a given bin?
- What is the numerical range or volume of a given bin?

From here forward, we consider data samples which can lie in some region of the real number line (e.g. integers, fractions, and roots of integers). We'll represent a sample of numbers by the vector \mathbf{x} , say $\mathbf{x} = (1.2, 3.002, 1, 1, 2)$.



The diagram outlines some relevant quantities of a histogram. The minimum and maximum values of the sample (1 and 3.002 in the example above) help define the range that we will distribute our bins in. We represent the bin width as h . It is not required that a histogram have fixed-width bins, but this is one of the most commonly used and useful formulations.

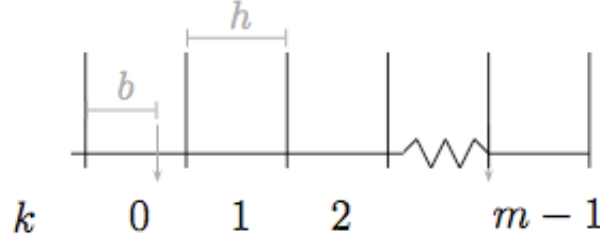
The starting offset ($0 \leq b < h$) is also a useful quantity. For instance, if the first bin starts at $\min(\mathbf{x})$ or $b = 0$, any number falling in the range $(\min(\mathbf{x}) \leq x < h)$ contributes to the quantity of elements in that bin. For $h = 0.3$, the range of the first bin in the example above is $(1 \leq x < 1.3)$. The two 1s and 1.2 will fall in the first bin.

Suppose we shift the bins a little to the left (i.e. $b = 0.1$). The range of the first bin is now $(.9 \leq x < 1.2)$. The two 1s fall in the first bin while 1.2 falls in the second. So it is easy to see how both the bin width and offset affect our view of the histogram.

The total number of bins in a histogram can be represented as:

$$m = \left\lceil \frac{\max(\mathbf{x}) - \min(\mathbf{x})}{h} \right\rceil + 1$$

where the $\lceil x \rceil$ represents the next integer larger than x , and this follows analogously for $\lfloor x \rfloor$. We add 1 to the traditional bin count to accommodate our offset b .



When constructing a histogram on a computer, we desire a function $\text{bin}(x)$ that maps a sample to an appropriate bin index $k = (0, 1, \dots, m-1)$. Assuming that we begin the bins at 0, the bin index can be determined by dividing the sample from the bin width: $\lfloor x/h \rfloor$. For $h = 0.3$, it's easy to see that numbers less than 0.3 will fall in bin 0, while a number twice as large will fall in the second bin, and so on. Accounting for the offset, we have:

$$\text{bin}(x) = \left\lfloor \frac{x - \min(\mathbf{x}) - b}{h} \right\rfloor$$

We also desire a function that will allow us to compute the start of the interval for a given bin index:

$$\text{interval}(k) = kh + \min(\mathbf{x}) - b$$

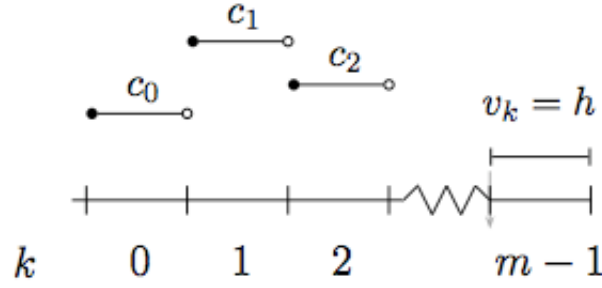
The formulation above allows us to experiment with the two essential qualities of a histogram (bin width and offset) and it should be straightforward to see how this simple approach generalizes for multi-dimensional histograms of equal bin widths, except we then consider more generally the bin volume (e.g in the three-dimensional case the volume $v = h^3$).

In the terminology of computer science, these are constant time procedures. If the bin widths are not fixed and arbitrarily set by the user then a more complicated method is required. One can use a specialized function or binary search for finding the appropriate bin in this case [1]. Histograms can also accept a variety of types (e.g. integers, single- or double-precision floating-point numbers) as inputs and if the compiler knows about this information ahead of time, it can optimize its code even further while providing a general interface of accumulators [2].

Histograms as density estimates

We now turn to a more mathematical perspective on histograms. It is natural to treat a histogram as the estimation of a probability density function. Once armed with the density function a variety of tools from probability and information theory are at our disposal, but we must take caution that the estimates of the density function hold true to the data sample and question whether we should be estimating the densities in the first place to solve the desired problem.

To get a taste for what this is like, we will derive the maximum likelihood estimator for a mathematical step function that represents the histogram.



Here, the function has m parameters: $\mathbf{c} = (c_0, c_1, \dots, c_{m-1})$. Provided our data sample and a given bin k , what is the optimal height, c_k^* , of the step function?

To find an estimate of the density function $p_H(x; \mathbf{c})$ we express the likelihood of our sample of data given the model parameters as:

$$\mathcal{L}(\mathbf{x}|\mathbf{c}) = \prod_{i=0}^{n-1} p_H(x_i; \mathbf{c}) = \prod_{k=0}^{m-1} c_k^{n_k}$$

Here, n_k represents the number of observed samples in bin k . In other words, we'd like find c_k^* that maximizes the probability that we'll find n_k items in bin k . This optimization is subject to the density function constraint:

$$\sum_{k=0}^{m-1} c_k v_k = 1$$

where v_k is the volume of the given histogram region.

Constrained optimizations of this form can be solved using the method of Lagrange multipliers. In this case, we have a multidimensional function:

$$f(\mathbf{c}) = \log \mathcal{L}(\mathbf{x}|\mathbf{c}) = \sum_{k=0}^{m-1} \log c_k^{n_k} = \sum_{k=0}^{m-1} n_k \log c_k$$

subject to the constraint that:

$$g(\mathbf{c}) = 1 - \sum_{k=0}^{m-1} c_k v_k = 0$$

Note that we took the log of the likelihood function so we can deal with sums rather than products. This will come in handy when taking derivatives in a moment and is a common practice.

The gradient of a multidimensional function results in a vector field where each point—in our case represented by \mathbf{c} —has an associated vector of steepest ascent or descent on the multidimensional surface [3]. Our goal is to find where the steepest ascent of $f(\mathbf{c})$ is equal in magnitude and direction to that of $g(\mathbf{c})$. This equality is represented as:

$$\nabla f(\mathbf{c}) = \lambda \nabla g(\mathbf{c})$$

And we can express this combined optimization with:

$$\nabla \Lambda(\mathbf{c}, \lambda) = \nabla f(\mathbf{c}) - \lambda \nabla g(\mathbf{c}) = 0$$

where $\Lambda(\mathbf{c}, \lambda)$ is called the “Lagrangian”. Computing the derivatives:

$$\frac{\partial \Lambda(\mathbf{c}, \lambda)}{\partial c_k} = \frac{n_k}{c_k} - \lambda v_k = 0$$

From this, we can see that

$$\frac{1}{\lambda} n_k = c_k v_k$$

Since $\sum_{k=0}^{m-1} c_k v_k = 1$ and $\sum_{k=0}^{m-1} n_k = n$, $\lambda = n$. Thus our optimal height and estimate of the density is:

$$c_k^* = \hat{p}_H(x) = \frac{n_k}{n v_k}, \text{ for all } x \text{ in bin } k$$

While this estimator is actually quite an intuitive result, it is important to observe that this only in the special case of step functions with varying volumes *predefined*. What if, for example, we wanted to see what the best estimator is given a piecewise linear function or a spline between bins? Or what if we want to find out what the bin volumes v_k should be as well? This question motivates a general discussion on the estimation of the probability density function given the data sample \mathbf{x} , and these “data-driven” approaches are dubbed non-parametric [4].

Similar issues as those brought up in density estimation also arise in the estimation of mutual information [5].

[1] Given a sample, start at the center-most bin. If the sample is greater than that value, then start at the center most bin of the right half, otherwise do the same to the left. Recurse this procedure until the desired bin is found. The running time of this scales like $\log_2(m)$. The GNU scientific library provides utilities in this spirit

[2] See the C++ Boost Accumulator library for more information on this.

[3] For a more detailed tour of why this is as well as an intuitive contour plot geometrical interpretation of this, consult an introductory applied math text. Setting $f = c$, we can study the function’s “surface contours”. By definition, a vector \mathbf{x} along this contour does not change the value of f , therefore $\nabla \cdot \mathbf{x} = 0$. Because of this, ∇f must be normal to \mathbf{x} .

[4] Scott DW, Sain SR. "Multidimensional density estimation". Handbook of Statistics: Data Mining and Computational Statistics 23. 297-306 (2004). Gentle JE. "Nonparametric estimation of probability density functions". Elements of Computational Statistics. 205-231 (2002).

[5] Paninski L. "Estimation of entropy and mutual information". Neural Computation 15. 1191-1253 (2003). Fraser AM, Swinney HL. "Independent coordinates for strange attractors from mutual information". Physical Review A 33. 1334-1140 (1986).

PSTricks and LaTeXdraw

□ November 12, 2008 by Geet

Those who are familiar with LaTeX know the power of PSTricks. I'd like to give my two-cents that LaTeXdraw is one of the cleanest and most-straightforward exporters (in this case from SVG) to PSTricks I've come across. If anyone knows of alternatives (other than Inkscape) that export particularly to PSTricks, I'd love to know.

How can I trust you, code?

□ November 12, 2008 by Geet

I've been programming a bit lately in C++ Boost for two reasons (1) I find the template metaprogramming aspects intriguing and the generic programming aspects super-convenient. Yet, when I'm really concerned with knowing precisely what's going on to avoid wasting CPU cycles and memory, I go back to C. But then of course C is doing all sorts of things for me as well (just see the difference in speed in your code with the -O3 optimization flag).

One trend I've noticed after reading a lot of code is that bit-twiddlers will often over-optimize. I'm more and more convinced that this not only affects the readability of the code, but it may also be indicative of a lack of trust in the compiler's optimization. The more control you have, the better you feel, right?

On the other end of things, some people trust objects and functions as black boxes and could care less what's going on underneath. This post is not for them.

I hate to pull the big CS "language" and "compiler" words out of my pocket again [1], but how can I trust you language and compiler? When do I know I'm over-twiddling and it's better that the compiler do the work for me? While there are guidelines, I think what it comes down to is a decent understanding of the compiler you're dealing with and then, basic experimentation.

Most of my questions about whether certain optimizations are actually occurring are typically answered by doing timing tests, creating little “experiment” code bits to test a simple hypothesis, or look directly at the assembly code (e.g. the “-S” flag in gcc).

This did get me thinking, though, with so many different compilers and architectures—even with standards and benchmarks—a lot of these little questions aren’t answered. This process of creating precise tests and getting the answers seems to be the best approach [2].

[1] But I do believe these topics have some of the most interesting research opportunities from a CS perspective. As I foray into scientific computing and its applications to the sciences, I hope I will have the chance to tackle a CSee problem or two along the way.

[2] Wouldn’t it be nice if there was a nice repository of contributed tests for a variety of architectures, compilers, and languages, and that these tests can be tagged with all this information? In the spirit of my last post, I say let’s start a massive conglomerate blog where people can post these tests. And then there could be another one where people just bitch about the goodness and badness of the tests. It’ll be like a somewhat moderated coding newsgroup for today’s age. Oh how I’ve blurred the line between honest desire and sarcasm here.

All my blog pipes are clogged

□ November 12, 2008 by Geet

It seems that there’s been a lot of hype lately about blogs selling out and the more personal touch being lost (so much so, that it’s not even worth linking to articles). It’s as if there’s this big trend and direction that we’re attempting to extract for where things are headed. To me, the partition between a blog with a personal twist and a more professional media one is quite clear (and it certainly doesn’t have to happen when you’re sponsored by a company).

For the obviousness it’s worth, I certainly prefer the random thoughts over the “professional updates”, and there’s of course still room for both.

IAS Princeton

□ September 26, 2008 by Geet

I visited the IAS this last week (“best known as the academic home of Albert Einstein, John von Neumann, and Kurt Gödel”) to meet with the prof I’m currently working with (she’s there for the year). It’s such a quiet and natural place that exists purely on “endowments, grants, and gifts”. That’s

quite the yearly budget considering they reportedly house 27 permanent faculty and award hundreds of year-long fellowships.

And “research is never contracted or directed; it is left to each individual researcher to pursue his or her own goals”. It certainly humbles any sense of self-discipline I’d like to think I have.

Pretty cellular pictures

□ September 3, 2008 by Geet

Electron microscopy for cellular transcription [1]:



Not saying that microscopy pictures are always the most instructive, but I was pleasantly surprised to see that the American Society for Cell Biology has a nice collection of images and videos available free for us online (and thankfully with decent descriptions/pointers).

[1] © 1969 J. Richard McIntosh. All rights reserved. Reprinted under license from The American Society for Cell Biology.

Network biology note

□ August 13, 2008 by Geet

I’ve been reading a bit more on biological networks lately (specifically those within the cell). There are a lot of interesting open questions and results that draw from molecular biology, network theory, and statistical physics. A few reviews stand out:

- Cell biology: networks, regulation, pathways (thanks Aaron!)
 - A good supplement to the paper above, though I've only sampled it, is the thesis *Information flow in biological networks*
- Network biology: understanding the cell's functional organization
- Getting connected: analysis and principles of biological networks

Compile-time optimization of data structures?

□ July 8, 2008 by Geet

Or: “**it’s all fun and games until you realize you’re a monkey**”. Bob brings up a big point about the expressiveness of a language and its relation to concise code. In a similar carpal tunnel vein (but certainly a bit less meaningful in scope) I find it difficult to find decent work done on compile-time optimization of data structures based on suggestions for space and time efficiencies and how they intend to be used.

In other words, if you enjoyed and/or revisited an undergraduate course on data structures, based on any given problem at hand, the chances are good that you can concoct some monstrosity of a hybrid data structure that keeps efficiency and your use of the code in mind.

The handwavy question I’m posing is, what aspects of concocting this crazy structure are actually due to clever thinking about the problem at hand vs. “this data structure is good for this, and this one for that” kind of algorithmic work? The end result of the algorithmic work is a tired hand and wordy, unexpressive code. I have this sneaking suspicion that a lot of the work is indeed algorithmic and it’s worth exploring what can be automated at compile-time based on potentially simple rules and patterns.

Perhaps a good example to keep things somewhat concrete is the notion of a set, essentially an unordered collection of unique elements. It has some basic binary operations like union and intersection and there are a variety of things one might imagine doing with a set: query for membership of an element, add an element to it, etc [1]. For instance, in a graph, every node might have an associated set of nodes it’s connected to [2], or it may alternatively be represented as a set of edges and a set of nodes where each edge is a set of two nodes. Now the last thing I’m advocating is to always use some mathematical structure to express what objects you’re dealing with in your program, but it does turn out that a “set object” can be quite handy. Here is some simple Python code that basically takes as an input a set of files and returns the number of unique files where each element of a file is a line (keep in mind there can be duplicate lines as well as entire duplicate files):

```

import sys
from sets import Set
uniqpars = Set()
for parfile in sys.stdin:
    parset = Set()
    pf = open(parfile.rstrip())
    for line in pf:
        parset.add(line)
    pf.close()
    uniqpars.add(parset)
print len(uniqpars)

```

The fact that the example above uses a set in a traditional object-oriented way [3] is somewhat beside the point. What sometimes matters are simple questions like how fast are membership and addition operations or are we doing a lot of unions or how much space does my set actually take? There are many ways to implement a set depending on how you're using it. A few examples:

Super-fast lookup, but potential memory or false positive concerns:

- A directly-mapped array or bitset if you have enough space to store your entire possible set of members.
- A valueless hash table if you have a decent amount of memory around, a potentially super-large set of possible members and want to store only the members you see. These also have super-quick lookup and insertion/deletion times.
- A Bloom filter if memory is a concern, you have a super-large set of possible members, don't mind not storing the members, and are willing to accept a small probability of false positives.

The structures above are great for fast operations given the set sizes don't fluctuate that much and you have the memory or are willing to deal with false positives.

- Binary search trees, especially the self-balancing kind, are great for sets with sizes that fluctuate a lot [4] or many union operations are being done, for example (they preserve an ordering of the elements that can be useful). If maximum or minimum element values need to be quickly accessed, binary heaps are often good ways to implement those kind of sets.

Obviously, there are a variety of ways to implement this kind of object. Object-oriented languages like C++ and Java have ways to separate the notion of an object from its implementation [5], but I reiterate, the object-orientedness is not necessarily the point. It's simply a convenient way to organize and express the fact that you're using a set object with commonly-used functions.

What seems to be deserving of more attention is if there are some handy structures with well-defined operations like a set, it's probably worth using those structures and optimizing for what we need to based on some helpful directives at compile-time [6]. In other words, suppose I'm using some objects with well-defined operations like a set with each other. Suppose I also have a way to express how I'm *using* my information with these objects. Finally, suppose I can use techniques that some resourceful computer scientists have come up with to piece together these objects in a relatively optimal way

based on how I plan to use the structures. And presto! We could potentially do a lot of work from the get-go to save the programmer some implementation details yet still wind up with some pretty efficient code.

OK, so there are a lot of assumptions there, but I imagine one can begin by citing more examples of commonly-used objects, they're uses, and common implementations by themselves and as hybrids with other data structures. Perhaps there are some interesting rules and techniques that could come about.

And all that just reminded of some Ph.D. work a friend is doing at the moment. Maybe this topic will be revisited.

[1] Note that I'm just focusing on data structures and basic operations as well as their optimization with other data structures and ignoring the more language-oriented aspects.

[2] Often implemented as an adjacency list

[3] That is, the way it can check equality of sets on insertion is the fact that each member is some object that has some defined way of saying it's "equal" to another object. In the example above, there is a set of lines and a "set of sets of those lines". With object-oriented techniques the "equals" function works for both even though the member types are different (i.e. a "line", and a "set of lines").

[4] As a brief motivator an allocator can be associated with a data structure to deal with this issue (though it can get hairy at times). A good review of allocation techniques as well as some implementation motivation can be found at this starting point.

[5] Though C++ is super-guilty of not using a multiple-inheritance trick to do this in practice.

[6] I don't think this is too far-fetched given we've done quite a lot of work in linear algebra and higher-level matrix-oriented languages tend to do a good job with running optimized routines based on a few directives.

Bob's blog.

□ June 28, 2008 by Geet

Two Crocodiles have emerged from the water.

R project map-reduce-filter tidbit

□ June 21, 2008 by Geet

The R-project is a great tool set for statistical computing (this is its speciality) and even just to have around for quick calculations, plots [1], and data manipulations. The community is large and the source is open. It provides a nifty Unix-like environment to work in and is available on three major operating systems. </advertisement>

Because of the large community size and the highly-interpreted nature of the language, there is definitely an “impure” feeling about using it since some packages have procedures that will call their own C or Fortran code while others use the language directly. I personally like to see it as a good platform for exploratory data analysis/prototyping ideas, and like to leave the more heavy-lifting to something..different.

That said, since its internals are kind of Scheme-like, the expressiveness of the language for data manipulation in particular can be quite handy [2]. The introduction describes a function “**tapply**” which is useful for applying functions to groups of items with the same category. In that neighborhood there is also “**lapply** / **sapply** / **mapply**” [3] which are like the traditional “map” function. “**subset**” is very much like the traditional “filter” function.

Not-so-advertised are the “**Map**”, “**Reduce**”, and “**Filter**” functions (tricky-little capital letters). The differences between the traditional FP functions above and the two R analogs listed in the last paragraph are mostly conveniences for the way R treats its data.

If you use R or are interested in experimenting with it, keep these functions in mind because they can make just a few lines of code do some pretty awesome things.

[1] Gnuplot and matplotlib are also pretty good open source alternatives to plotting, and there are of course any non-open source options. In my opinion, experimentation with R is definitely worth the time if you’re playing with plots, and willing to side-step a bit from the Python bandwagon, since Python does have some well-developed statistical and scientific computing tools.

[2] See their “Manuals” section for some good introductory documentation and language definition. Many scripting languages these days support higher order functions.

[3] “mapply” is a neat sort-of multimodal map.

Taking cover-ups too literally

□ June 13, 2008 by Geet

Are they ashamed of the prices? Perhaps hoping to lure in the gullible for a good deal? A statement by the station manager?

Texify

□ June 6, 2008 by Geet

Link advertisement: I came across Texify a while back. It is great for inserting TeX in HTML notes / documentation / blog entries without bothering with the more in-depth Latexing better suited for article creation.

Jot down that melody

□ May 20, 2008 by Geet

Have you ever conjured up a simple tune in your head but felt like big hammers were a bit inappropriate to jot the melodic thought down? There exists a wonderful thing called LilyPond which takes a TeX-like syntax for music composition and not only renders a neat little PDF or postscript of the score, but can also create a MIDI file of the tune that you can play very easily with a program like Timidity (this program allows you to avoid the sometimes tedious configuring of hardware and software to do MIDI synthesis that comes second nature to more seasoned PC audio folks). Here's a simple example of how easy it is:

Write a super-sophisticated melody to some file called test.ly:

```
score {  
  { c' e' g' e' }  
}
```

Now, just run:

```
$ lilypond test.ly
```

and it will produce a test.pdf (and postscript) file that looks something like:

Now to create a MIDI file simply add a directive to your file:

```
score {  
  { c' e' g' e' }  
  midi {}  
}
```

and rerun the command above. It will produce a MIDI file. If you'd like to hear it with software-synthesized goodness:

\$ timidity test.midi

Timidity can, of course, output to WAV or OGG and whatnot for you.

High-level UI translation

□ May 20, 2008 by Geet

With all the rave on generic higher-level interfaces looking more native to the host OS (as opposed the more free-form interface experimentation and design with generic low-level interfaces), I wonder if there has been much work done on actually translating the high-level widget description to OS (widget toolkit)-specific code with function-call bindings for events. This is clearly doable considering Firefox already defines these bindings for the three major OSes. OS-independent GUI code can then be translated into a native look and feel while not sacrificing efficiency because the interface description compiles to native OS widget code.

Not that I'm super-thrilled about the "widget-based" standards we've created for high level interfaces these days, but it's a thought....

Update: WxWidgets seems to do the trick.

Systematic learning vs. motivated problem solving

□ May 13, 2008 by Geet

Warning: what follows is a hand-wavy one-sided monologue, continue at your own risk!

It seems that the most appealing and productive way for someone like myself to learn and explore is via solving a somewhat concrete problem that motivates me (e.g. a sampling of the links on the sidebar of this log). Regardless of the fundamental and pure aspects of this motivation, I find that I learn a lot along the way and other interesting problems and doors open—natural extensions to continue doing more interesting work.

So it's tempting to say, "this is *ideally* how I should learn: through some form of personally-motivated problem solving," trusting that this approach will fan out more and more interesting work potentially even covering a lot of ground that I would come across in more artificial, in-order perusals of self-

contained texts. Fueling this temptation: since more introductions, review papers, and articles are available online, we may not be as dependent on self-contained texts and can actually tackle a subject at hand (for instance, read the preface [here](#)).

While this may be a sensible approach in principle, its limit in scope intuitively ignores two important things which motivate the rest of the entry: (1) like any sport, a push or even nudge of systematic self-contained learning goes a long way in growing more rapidly and increasing competence and (2) in reality abstractions are built and reused in ways that are dependent on one another, so students must take classes on well-defined subjects, professors must teach them, and professionals often end up becoming specialists/experts naturally, or are forced to take “mandatory” training to learn their craft.

Thus, regardless of what’s ideal, simply working for the man may require a more systematic approach to learning material. How this is approached by students seems to be a problem. Most students, for example, may see tough, self-contained classes as a chore or measure of performance, even if aspects of the material interest them. A systematic introduction to a topic meant to fuel thought and cover some potentially very good concrete examples in a breadth of topics in reality just implants the existence of a few keywords, or better-yet, a concept or two. My hunch is this is what the modern teacher/prof settles for in the general student (understandably so, there are often *a lot* of students per professor).

If I were to have a point (hopefully you don’t assume I need one, especially in a web log entry), it would be something like, “some students either blame themselves too much or not enough; they are either too hard on themselves for not understanding something they find difficult or, equally as bad, assume that something is wrong with the way a class is being taught.” If the systematic learning is approached *by the student* in less of a pressured or stand-offish way and in more of a, “well, I’m somewhat interested in/required by the man to learn about this, let me be open to the concrete examples and approach them curiously,” then systematic learning may not only prove to be less of a chore and more interesting, but also be a healthy push in an interesting direction.

Keep in mind that I say this as a student benefiting from this attitude for some time, though not nearly long enough.

The Structure and Interpretation of *

□ May 7, 2008 by Geet

For a while now I’ve been meaning to note a couple links that may be of interest. Some computer science undergrads are familiar with a happy-little purple book by Abelson and Sussman titled The Structure and Interpretation of Computer Programs (SICP, pronounced “sick-pea”) [1]. For those of you who don’t know, if you click on that last link, you’ll notice the entire text is available online, which is great because this book is a superb reference (at least it has been for me). SICP is known for its usage of a LISP-like variant: Scheme to introduce a precocious individual to computer science.

If you can get through that and an introductory classical mechanics course, then why not learn about classical mechanics vis-a-vis Scheme? Also available online, *The Structure and Interpretation of Classical Mechanics* (SICM or “sick-’em”, I presume).

[1] Sadly, most schools that adopt MIT’s SICP program probably teach the course with some pretension as the first (or one of the first) computer science course for undergrads. While the upper-echelon students enjoy and potentially fly through the material, most well-intentioned and eager students seem to become disenchanted with the complexity and challenge of the material. I appreciate the rationale for teaching this sooner than later: this book is a great intro to *real* computer science, not for-loops and try-catch statements. That being said, perhaps a lighter introduction with the book is in order, because even the best of students will just scrape at the surface of the cool stuff in it.

A little refresher on this never hurts,

□ May 2, 2008 by Geet

so here’s to thinking for yourself.

Analysis of Biological Networks (Book)

□ April 9, 2008 by Geet

A recent book on the application of complex networks tools to biological systems and vice versa has caught my eye: *Analysis of Biological Networks* by Bjorn Junker and Falk Schreiber. It seems to cover “middle ground” and contextual information very relevant for someone interested in the connection.

Update: this book is alright, but I feel later posts point to better papers.

Get more from your computers

□ April 3, 2008 by Geet

If you’re like me at all, you probably find that even with older/ alternative laptops and desktops you may have lying around, one of them is essentially the “main” one. Most of these machines (even ones that are many years old) still have it in them to do good for you—whether it be for the little monitor

space they provide or offloading some basic media or computations to them. While this is ideal, it's not all that practical and it may be tempting to sell or scrap your old and stick with the new. Inappropriate.

Taking advantage of a few simple, tried-and-true tools that have been shown to work well across platforms can go a long way. I find these have helped me use all my PCs in a surprisingly integrated and headache-free way so I can focus on stuff I'd rather do:

- A quick way to backup and update files to any machine that allows you to SSH into it (and if you're so inclined, an easy way to access your files remotely in the same spirit).
- If you're an occasional or heavy terminal user and run computationally-intensive tasks on machines better-suited for the job than your current one, it might be nice to log out of your system and log back in at a future time to see things exactly as you left them (but with more work done of course).
- Finally, the most important tool allows one to share a mouse and keyboard across computers and across platforms over a local network, making it seem like each monitor is all part of one big desktop in which you can arrange your computers and share clipboard data with one another seamlessly—and a refreshing use of a sleezy-sounding corporate buzz word.

The picture above shows three quite different machines that have been with me for years. They now share the same keyboard and mouse and act as a big desktop when around each other, but still function just as well as independent units.

Laptop wishlist

□ March 26, 2008 by Geet

The time has come where a biennial urge for a new laptop device has shadowed my good senses, intermittently diverting me from all the things that make a young spring so enjoyable. In keeping with previous patterns, I have decided to go with a Linux approach after a wonderful MacBook experience. I don't like to spend much on laptops. The most I've ever spent is on a Powerbook Titanium: a lovely machine, but still a bit steep for a laptop.

I do however tend to use them as my primary machine since I like having my setup go with me places. Fortunately, the file management and backup system I've chosen is quite portable whether I'm on a Mac or Linux system so long as it supports decent permissions and a UNIX-like semantic (e.g. ext3, HFS, ...). Moreover most of the software I use is either a web application or open source and available on all 3 major OSs.

I'm not one to join flame wars on laptop brands or concern myself with how to get everything working just right on a Linux laptop, but at the same token, experience with a variety of them has built a sort of rough "wishlist" over the years—not so much what a laptop *can or should be able to do*

(you can get laptops to *do anything* these days), just more on some general themes that would seem to improve my experience, but also things I don't necessarily have the time to work on myself. My views are biased to Linux and OS X since I stopped using Windows (for no principled reason whatsoever) years back.

Apple is officially the epitome of great laptop and software design that “works out the box” for me. They make great machines, and the MacBook series has super hardware for the price. Since everything is sold as a package, they've intertwined their hardware and software in very intelligent ways that leaves me typically very pleased and impressed. There are noticeable things like the snappiest sleep-wake process that I've seen: this makes it super-easy to jot notes down and tote the top away in seconds. And perhaps less appreciated things such as how all the audio / multitrack mixing / soundfont midi synthesis / etc works seamlessly (much more than I can say for Linux). Over the last few years they have made strides in making open source Linux-like scientific computing programming more possible and transparent (given that you have XCode installed with a little Fink or DarwinPorts).

Yet with all this, there are times I feel a little trapped in the Apple world. Though software like Gimp and Inkscape is available for OS X, it's usually a clunky X11 version that isn't in keeping with good OS X software design. That being said, there are many great OS X specific programs like TeX-Shop or OmniGraffle that do the job just as well if not better than a well-known open source alternative (sometimes at a price). If I can avoid proprietary standards and commitments to a single company / possibly ephemeral software life, I'd like to. This tends to lean me towards using well established open source software that writes files out to simple formats or standards.

The first wish is not directed to Apple so much as the open source community making OS X ports. For the programs written in GTK, moving towards something like GTK on OS X, like Windows did years ago would go a long way in making some heavier-duty open source programs look good and feel snappier on OS X. Speaking of which, the second wish is harder to describe. Things are typically not that snappy feeling and configurable on OS X in general. I can't pinpoint exactly why, though I've brainstormed several possible reasons. I used to attribute this flaw to an intentional design constraint imposed by Apple. While this may be the case for some things, I'm becoming more and more convinced that a bit more gracefully placed configuration settings and some window manager snappiness could go a long way for the next generation of OS X power users. A third wish is that the MacBook Pros would very much benefit from higher screen resolutions for the prices they're sold at.

Linux has come a long way in making their distributions work out of the box on laptops since my first experimentations with them. The community has done a great job at letting each other know how to resolve issues with Linux in general or with particular distributions (shamelessly). The fourth wish is for the sleep-wake functionality in Linux and laptops to work better (this is a long shot). I've had some luck in the past but, for the most part, the kernel always needs to modprobe something again, reload some service, etc., leaving me to a “just boot the computer every time you transport” heuristic. A fifth wish is for companies to continue the growing efforts to support general web apps and software resources (hardware drivers, Flash, legal DVD playback, etc.). The sixth and possibly most practical wish is to have a distro's community help contribute to a metapackage and post-install scripts that tailor configure settings such that the hardware and software are the most compatible with the machine.

Do it yourself for laptops, unlike desktops, has pretty much been a joke (other than an *alright* attempt at barebones laptops). The parts are surprisingly standard these days, except it appears you must buy the parts in bulk and invest in manufacturing chassis *en masse* before an earnest attempt at a barebones laptop company can be made. I sincerely think that a business along these lines could go a long way if marketed minimalistically, carefully, and in large enough quantity to wholesale the parts. Coat some decent utilitarian chassis designs with an elegant web-based ordering system (or reselling through something like newegg or mwave) and I would have a seventh wish satisfied: I would be able to piece together a high-resolution LCD laptop of my own with swappable motherboard, hard drive, and CD rom, all in a solid, hand-picked framework. Finally wish eight, Dell, though your designs and laptop quality has improved over the last few years, the machinery that is your customer service needs to strive for sending their customers along a directed *acyclic* path.

As far as the current purchase, I've gone for a very inexpensive deal on a pretty decked-out high resolution Inspiron 1720, since I feel comfortable developing on a Linux laptop and testing scientific computation programs on some decent portable processing and RAM.

Strogatz's Sync

□ February 28, 2008 by Geet

Evidently, as Feedburner suggests, there are quite a few more than the two or three suspected subscribers to this log's feed. I'll be amused to see if that number goes down as I post a bit more consistently, but for the moment it's nice to know there're ears out there.

I recently got done reading a book called *Sync* written in '03 by Steven Strogatz (doctoral advisor to the professor I'm working with at the moment). Disclaimer: if I write about something I read it's not really meant to be a "review" so much as "random wonderments" post-reading. My hope is that this style keeps the reading somewhat interesting and digestible while maintaining some semblance of coherence.

As an example, one of the first problems he motivates the entire book with is fathoming how exactly the many many tiny pacemaker cells of the heart fire in, you guessed it, *sync* to contribute to a heart beat:

...Each pacemaker cell is abstracted as an oscillating electrical circuit ... a constant input current causes the voltage across the capacitor to charge up, increasing its voltage steadily. Meanwhile, as the voltage rises, the amount of current leaking through the resistor [wired in parallel] increases, so the increase slows down. [By definition,] when the voltage reaches a threshold, the capacitor discharges and the voltage drops to zero.

...

Next, Peskin idealized the cardiac pacemaker as an enormous collection of those mathematical oscillators [all equal and able to communicate to one another]. ... Specifically, when one oscillator fires, it instantly kicks the voltages of all the others up by a fixed amount.

...

The effects of the firing are mixed: Oscillators that were close to the threshold are knocked closer to the firing oscillator but those close to baseline are knocked farther out of phase. In other words, a single firing has synchronizing effects for some oscillators and desynchronizing effects for others. The long-term consequences of all these rearrangements are impossible to fathom by common sense alone.

He goes on, with a colleague, to help extend Peskin's proof that two of them will indeed synchronize in the long-term to an arbitrary number of them (a bit more of a difficult task). Moreover, he emphasizes that this "so the increase slows down" part of the model described above is crucial for synchrony to occur.

The rest of the book extends on that type of oscillation problem quite a bit more. It undertakes a massive pop-sci sampling of examples of synchrony in a wide variety of areas. Sometimes he'd drift a bit off the topic of synchrony, but eventually some thread brought it right back into play. All of it was entertainingly explained, while he revealing his own personal connection to the work in an anachronistic way. The text is littered with some of the funniest scientific analogies I've seen. For instance, his description of a laser begins with:

Imagine you wake up one morning and find yourself on an alien planet, entirely deserted except for a watermelon and step stool beside it. ...

He also made it a point to illustrate where we've verified certain types of synchrony in practical science (e.g. Josephson arrays, circadian rhythms) while others are less pragmatic to verify (heart and neural studies, for example).

Something that definitely stuck in my mind afterwards is the way the more traditional research of nonlinear dynamical models and their applicability to practical systems evolved into the more current research of complex systems. Discussing some of his earlier work with Art Winfree:

In 1981, nonlinear dynamics had certainly not advanced to the stage where it could predict the behavior of such rotating waves in three dimensions. There was no hope of calculating their evolution in time, their lashing about, their swirling patterns of electrical turbulence. Even if the calculations were possible (assisted by a supercomputer, perhaps), any such attempt would be premature, since one wouldn't know how to interpret the findings. ... So Winfree felt that the first step would learn how to recognize them, to anticipate their features in his mind's eye; he would worry about their modus operandi later.

This sentiment appearing back in the '80s for exploring the dynamics of excitable media (with potential applications to an earnest start at modeling the systems-level chemistry of the intestines, stomach, and heart) also seems to apply to today's research in complex networks where the pure mathematics and statistics of dynamical null models for systems needs to be developed in tandem with useful methods to probe and explore a system of interest. I can't help but feeling that a good balance of the two vantage points where both feed each other is a good way to approach this young field.

Knuth and diamond signs

□ February 24, 2008 by Geet

You may show appreciation for Knuth's work by blogging or finding an error worth \$2.56 in his latest fascicle, but my friend Brian at Google's Maps team gave (and received!) a shout out by laying out the Knuth diamond signs photos on a Maps page.

It's been a while....but I'm glad to be back on the log as well am even more glad to be doing the College Park student thing.

What are the chances it will rain tomorrow?

□ October 21, 2007 by Geet

Leo has recently posted on our interpretation of the chances that it will rain. While I was bit caught up on how we may actually arrive at that probability, his question was even more fundamental in nature. The answer is surprisingly straightforward—so much so that you may kick yourself if you didn't think of it right away.

Let the psycho-robots rise.

□ October 21, 2007 by Geet

One of my favorite profs from school has resurfaced with a web log. Known for his robotics and vision work (e.g. Sony Aibo), he's currently spearheading the Animate Arts program at Northwestern.

One of his recent position papers presents some examples from psychopathology that may be valuable to look at when developing AI for interactive narratives. He argues that this is a good playground for thinking about these examples in concrete computational terms and that current architectures aren't particularly designed for this type of play.

The first example is self-medication:

What's interesting about self-medication is that although it is generally caused by some outside stressor, its goal is not to alleviate the stressor, so much as to regulate one's own affective response to the stressor. If you get drunk because your spouse left you, the goal of drinking isn't to get your spouse back, but to restore some sort of emotional equilibrium.

...

Although we can always add the rule to an agent's program that says (IF NO-DATE EAT-ICE-CREAM), current architectures don't account well for the systematicity of this general phenomenon. Not everyone will eat ice cream in response to perceived rejection, but nearly everyone will respond with some sort of self-soothing behavior.

The next example he brings up is limerence (the pattern of obsession, idealization, and fear most commonly associated with "falling in love"):

One of the most paradoxical characteristics of limerence is driven in large part by uncertainty. People don't become limerent toward those who indicate unambiguous interest or rejection toward them, but toward those whose behavior is ambiguous or inconsistent.... Once limerence begins, perceived rejection by the beloved actually increases the amount of time spent in limerent fantasy rather than reducing it (although sustained rejection will reduce and ultimately eliminate it).

...

this is interesting for AI because current architectures don't allow agents to [satisfy] goals (albeit temporarily and unsatisfactorily) through fantasy.

He notes how authoritarian personalities can predict behavioral patterns associated with group affiliation and how social rank, though this particular aspect was far less clear on my reading:

Authoritarianism also predicts certain aspects of behavior. For example, when asked to choose punishments for others' crimes, high authoritarians will in general choose more severe punishments than low authoritarians, and report greater pleasure in administering the punishment. However, their assignment of punishment will depend on the identity of the perpetrator; an accountant who started a fight with a "hippie panhandler" will be given less of a punishment than if the subject is told the hippie started the the fight with the accountant.

...

Modeling these traits computationally requires build[ing] reasoning systems in which (1) reasoning processes depend on the social status and affiliation of those being reasoned about, yet (2) the system itself is unaware of such dependencies.

Don't really appreciate the reasoning for item (2) up there.... I also feel that although this is a position paper on perhaps a more esoteric subject, I got little context about what the current AI architectures look like, or at the very least, a few more concrete details on *how* they're inappropriate for this type of thinking. Nonetheless, I think his point is that these things are important to start exploring at a computational level when considering AI models for narratives...interesting.

He also has recently posted an essay, "[What is computation](#)" that definitely seems worth a glance when I get the chance...it is written at an introductory level and should therefore be accessible to anyone interested.

Return from hiatus

□ [October 21, 2007](#) by [Geet](#)

So after a bit of a break from logging, I'm back. The home page has gotten a new look: that is, this log is my new home page. A significant addition is the start of a [del.icio.us account](#) (from which rss feeds are made available for particular tags or the bookmarks page in general). On the side bar you can see the tags I use often. I will also be using del.icio.us for bookmarking preprint papers of interest, so it serves a few handy functions for me. Satisfying a request, I will be extracting tags from del.icio.us for a new bookmarks and shoebox page shortly.

I want more monitors!

□ [July 25, 2007](#) by [Geet](#)

Breaking my promise on disallowing two-sentence log entries. It seems they just want more monitors :).

Desktop client vs. web app for web services

□ July 22, 2007 by Geet

The release of Google Gears will hopefully ignite some interest in focusing more on the distinction between a web application for a web service (i.e. essentially a GUI that runs completely within your browser communicating with a server hosting a web service) and a desktop client for a web service (e.g. Google Talk, ecto, IMAP clients, etc.).

While keeping the browser as the center of application activity has the benefits of being cross-platform, easily accessible, and having “the same look and feel wherever”, it would seem that we have gone through great pains simply to account for differences in browsers and to make the process a little more manageable.

This of course raises the question of how valuable a within-browser application is, especially for a well established company like Google. Would it be more valuable (though it will likely take more resources) to focus on desktop clients for web applications—clients that play friendly with the host operating system and desktop environment? Somewhat impractically, what if Mac Mail, for example, had a Google plugin where a few interface changes like conversation views (web applications like Gmail are of course not just a protocol for data exchange, they provide an interface specification) and Mac address-book synchronization took place? This kind of thinking isn't unheard of.

Gears certainly gives us a little more of a practical playing field for designing web applications to be resilient to “flaky” network activity. Though the underlying principles are not completely unexplored, it will be nice to see progress on the application-design standpoint. The various decisions on how to handle the synchronization activities from a UI perspective will be challenging. For example, it is fairly intuitive that the level of “local” activity and roughly the “rate” of synchronization has basic relations to the “collaborativeness” of the application. For highly collaborative applications (e.g. Google Docs), much emphasis would need to be placed on network storage and fast synchronization, whereas for more personal applications this is not the case. Getting this kind of application to work “intuitively” well, and better yet, withstand critique are both tall orders.

In any case, it'll be interesting to see where things head...whether the browser takes the monopoly for many apps or not. Microsoft's solution appears to be still heavily focused on the browser, and I too find GNOME's proposed plan a bit unclear.

Two friends, two more web logs

□ June 16, 2007 by Geet

I have recently discovered two of my good friends from school have recently made web logs. Mark is a math grad student at Cal Poly. He was an EE as an undergrad. He's slept through more midterms than me, and somehow takes the Putnam on a whim only to score remarkably well and gets unexpected awards from the math department. When away from math, he codes some particularly diverting javascript games, is on the development team for Fluxbox, and can occasionally be found sitting on a lawn chair *in* a tree. His log is far more entertaining than mine.

My other friend recently completed his CS graduate degree at Chapel Hill and has just started at Google. His log is a bit more sporadically updated, but perhaps he'll keep the world updated with his tomfooleries at Google in his log.

If your keyboard isn't good enough

□ May 8, 2007 by Geet

then make yourself one of these. And why not one of these (or dare I say, *this*) to compliment it?

Recently while taking a library-break, I came across a fun to walk through book. Haven't fully read it yet (still have to finish a couple on the nightstand), but was drawn from the beginning. Who knows, perhaps it is deserving of more study.

Monte carlo sampling the modularity space

□ May 8, 2007 by Geet

Warning: I'm not really doing this entry justice, but it should be good-enough for logging purposes.

For the past few years, the complex networks community has spawned a thread in community detection: very roughly the problem of detecting subgraphs within a network that have a greater density of edges within-community than not.

Though greedy algorithms optimizing a very successful criteria called modularity (obviously there are many alternative methods to optimizing modularity overviewed in the literature) are good for very large networks, one side-issue is properly analyzing the variety of partitions that could be generated that have similar modularities. This has been explored a bit (too many links for the spirit of this post).

One way to tackle this issue is to use a technique called Markov Chain Monte Carlo (MCMC), in which the transition probabilities in your Markov chain are from one possible partitioning of the network to another. Since the number of possible partitions of n nodes in the network is equivalent to the number of possible set partitions with n members, we can't possibly exhaustively explore the entire space for even moderately sized networks. But we can, if careful, sample from this space with proportional to a bias of our choice because it can be shown that the Markov chain will eventually reach a state of equilibrium.

This technique, among other things, helps us learn more about the properties of the space of partitions as well as analyze whether the partitioning buys us much (i.e. if the nodes really stay in one partition a lot or not for the ensemble of partitions generated at equilibrium). I've come across and have been pointed to literature where it's in statistics, quantum chemistry, and statistical physics, and am most certain it's used in many more fields of study. Part of the appeal in using the technique is that it allows for a surprising amount of flexibility in choosing the transition probabilities and is conceptually easy to grasp (I wonder how badly it's abused).

I can now link you to some code I wrote.

It uses the same transition scheme described in Massen-Doye '06, but samples directly proportional to the modularity rather than via parallel tempering on the Boltzmann distribution (an exploratory suggestion from Aaron Clauset). The results show that this works quite well itself, and the modularity of the partitions at equilibrium usually hover below the maximum determined modularity, as expected.

With this "starting point" (the cleaned up code) I have tried a variety of other transition schemes as well as done some thinking on what would be valuable in terms of the target distribution. This exploration serves as a good basis for continuing to play with modularity, its connection to hierarchical structure, and even dynamical systems.

On a less technical level, I have been slowly gathering notes on what's actually happening (a.k.a WAH?) in the field of statistical analysis of complex networks (both static and dynamic) as a whole and what certain analysis techniques buy us for particular networks. Right now I'm still engrossed in seminal work, but as they become more refined I'll post a link to them so that I can perhaps get more feedback on my "worldview" of the area.

And even more tangentially related, those of you who may reproducing research, there is a slight error in the reporting of the modularity of the karate club network for the fast algorithm (the author has been aware of this) and that the second pass of the CNM algorithm should be run at step MAXQ-1 to get the partition with optimal modularity.

Communities of papers in arXiv

□ April 6, 2007 by Geet



So as a first-pass exploration into the community structure aspect of complex networks, in particular arXiv, I read-up, then scripted up an exploration on some test citation networks. End result: some marginal progress as expected, but it bought me some increased maturity and satisfaction of concrete play. Also, who doesn't like to click on nodes and see the communities?

You can see the web page and a TeX-note if you're interested in killing a few minutes of your time.

Special thanks post-exploration to Aaron Clauset and Michelle Girvan who gave me helpful guidance and suggestions on continuing with this community analysis research after a particularly insightful talk given today. Now finally on to more interesting reading / work on a topic I've meant to learn properly for a while.

Community structure paper added to shoebox

□ March 19, 2007 by Geet

Though there are a variety of interesting papers on community structure (many of which I still have yet to read), I thought I'd place the Clauset-Newman-Moore paper as downloaded and in my shoebox because it seems that paper was one of the important ones in taking the developed idea and making it practical for very large networks. I like how it starts from the modularity optimization and takes it all the way to the data-structure specific solution into one packaged solution, whose code is readily available on the web—now that's good research. I may consider that also for Shalizi and company's CSSR, which has always intrigued me but I've never really sat down to really understand what's going on.

A shoebox

□ March 7, 2007 by Geet

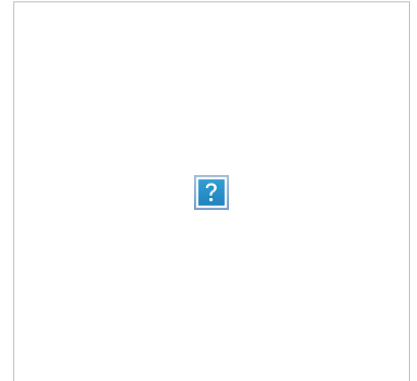
I have recently added a "shoebox" to my home page. The basic idea is any papers, images, code, etc. that I feel at some level is worth storing locally, I place there in reverse chronological order.

What makes something worthy of the shoebox? Beats me, but you may get an idea by looking in it.

What is an energy landscape?

□ January 22, 2007 by Geet

This note simply tries to get across as accurately and conceptually as possible what a potential energy landscape is. It is part of an inquiry into complex networks and physical structure. The field has quite a bit of activity, and the discussion can be seen as a simple introduction to the topic. It is difficult to find introductions at this level, so it may be of use to others, but mostly I just wrote it to log some progress. Thanks to a friend for his thoughts and I welcome other thoughts and better-yet corrections (though I don't expect any comments at such an early stage of me web logging).



You can download the PDF directly via this [link](#).

In the interim...

□ January 6, 2007 by Geet

I've completed the first draft of the next technical entry I'll post soon (revised title): "What is an energy landscape?". While I have a couple of friends look at that, I'm going to be developing some other thoughts. Technical entries make themselves obvious not only by their crappy titles, but also by the fact that they'll usually link to a "TeX note" (a variation of the term "tech note" borrowed from a friend) of some sort or contain words that mean nothing to a lot of people. Non-technical ones should be more engaging, I hope. I consider these first few entries me "warming" up to the habit of using the web log. Hope everyone is enjoying the new year.

Why do we care about the small world properties of atomic clusters?

□ December 31, 2006 by Geet

This summary is to try to keep a bigger picture in mind before getting too specific too quickly: a brief survey of a topic of interest in a text. It is specifically in the context of global optimization. I first wrote this post a few months back. Most of meat is obtained from Wales's *Energy Landscapes* and Doye and Massen's "Characterizing the network topology...atomic clusters". Finding the global minimum and local minima on an energy landscape (a purely theoretical thing determined from pure quantum-mechanical models or hybrid quantum-classical models) gives us cues as to how a set of molecules will structurally stabilize. Practically, this information is invaluable to molecular structure prediction, which helps us better understand how molecules are actually *built*.

The issue lies in finding the global minimum. I don't pretend to deeply know this area or detail it here by any means, but I may soon! Finding the global optimum in general is an NP-hard problem in computational complexity. In other words, "there is no known algorithm that is certain to solve the problem within a time that scales as a power of the system size". So what do we do? Exploit structural properties of a given atomic cluster to drastically improve the running time of the *algorithm*. (Wales, "Global optimisation")

This subfield in itself is very active with many different algorithms suited for all sorts of structures. But as far as small worlds go, it seems a basin hopping global optimization approach that could take advantage of the relatively scale-free properties of the landscape. But as implied (Wales, "Small worlds"), we should take caution in considering the cluster we're applying these algorithms to, how exactly the global optimization search is improved, and whether the small world results apply generally enough to a broader class of molecules or not:

It is not yet clear how general this result may be for other potential energy surfaces [than Lennard-Jones]. For example, the connectivity of the constrained polymer conformations was found to exhibit a Gaussian rather than a power law distribution for a noninteracting lattice polymer. For the Lennard-Jones polymer... Doye has found that the connectivity distribution does not really follow a power law, but the scaling is stronger than exponential...If low energy minima are generally highly connected then they should be easier to locate in global optimisation searches...However, the best way to utilise this information more efficiently is not yet clear.

Though it has been a few years since that was written, it seems to still hold true from my cursory understanding. Most of the work that has been done seems to be applied to Lennard-Jones clusters. Needless to say, I'm motivated to dig a little deeper.

The spirit of this log (and welcome!)

□ December 31, 2006 by Geet

Not so long ago, a friend brought up to me that the Atom feed on my home page that I edited by hand was infrequently updated even though I frequently added thoughts to my home page. My initial intent was to post only very significant changes, but the more thought I gave to the subject, the more I realized that the a web log with user comment capabilities and a significant user base is a good thing to participate in, and the personal home page is better for organizing the chronological information as I see fit. I feel that many bright people have web logs of interest to me and they have benefited me. In the rest of this post, I will try to explain (1) why I appreciate the way web logging has evolved from my perspective, and (2) if I'm fortunate enough to gain a reader or two, what you can expect of the quality of the entries to come. This entry in particular will avoid referencing other material and stay focused on my personal motivation for beginning the log.

A little history

This is not my first attempt at web logging. I used to keep logs in raw HTML on my server followed by spending some time with MovableType. I've spent many years learning how Internet news scoops, forums, and logs are truly used and feel that there are certain simple truths when approaching the subject. Trying to keep a system that kept all my thoughts categorized, organized, and consistent in one data store frustrated me at times. I have stubbornly (better now than never) come to accept the roles that certain web services play so I can use them to my advantage. It is OK to have duplicated information not all in one beautiful framework. The world is messy, our thoughts our messy, and the way we present them should aim to be valuable, and just a little less messy, but certainly not perfect or even close to it!

(Warning.) For some quick background so you know what kind of crank you're dealing with, being actively interested in the sciences as well as computers in high school, I went on to pursue a B.S. in computer science at Northwestern University. During the day, I'm in gov. service and deal with computer-related problems from mundane to very technical. During the evenings, when I get the chance, I like to exercise my drive towards the physical sciences and the statistical laws that govern so many systems not nearly as explored as we would like them to be. Consequently, I have geared myself towards a track of research that tends to relate those two topics.

Some of the content you can expect

Most recently, I've been interested in complex networks as they relate to physical structures—more specifically energy landscapes of many kinds. I'm also interested in hash-based data structures and the nature of the transition from discrete to continuous shapes in geometry. Most of the technical articles you will see on this log should relate to that. Though I don't expect a huge comment-base, if you have relevant thoughts in response to especially the technical subjects, I'd appreciate it. I don't promise pristine referencing, but I will try my best—especially on technical subjects—to be thorough with references.

While I promise no fleeting 2-sentence blog entries, some entries may be non-technical or avant-garde/controversial/even stupid in what they suggest. I'd like them to be about something—maybe topics more close-to-home about family and friends will hopefully be of interest to some.

Why web logging makes sense

Let's knock out some of the stupid criticisms of web logging services that I've run into. The first and most foolish responses to them are, "why not just keep an HTML file?" and "why post thoughts like that to everyone anyway?" To answer the latter, in the cases where little is being discussed and too much trolling going on, this is a reasonable criticism. I feel entries with little thought put into them and too much emotion sometimes stir controversy that is often not even worth arguing about and personal forum/comment battles ensue with little to be gained. But if that criticism has to do with posting undeveloped thoughts that aren't "articles", per se, I'd have to disagree whole-heartedly. The best learning goes on when simple, obvious questions are answered by people who are more knowledgeable and when those knowledgeable people are read like human beings with lives, charisma, and flaws, and not as almighty authorities. It is exactly that potential for mutual improvement why I think it's reasonable to post a thought every-so-often to the world to see if they have anything to say, or just to let something out there.

My response to the first criticism is simple. Web log services provide two things that your HTML file on your server doesn't help with: (1) a community of people partially networked and (2) technical features that help facilitate bulletin-board discussion amongst that community. Features like comments, trackback, and automatic syndication allow people to keep up with each other's thoughts and motivate discussion in them, and the interfaces, as ghastly as they can be, certainly make writing them easier than in HTML (and this comes from some time attempting to do the opposite).

Finally, I think web logs have a unique role in discussion on the net in the sense that unlike a forum, most web loggers have their identities revealed. Folklore of anonymous statements and hacking galore can be fun, but sometimes I like to cut the bullshit and see who I'm talking to, what their background is, and why are they honestly interested in responding to my thought.

In the end, a log is just a log...that's why I often don't call web logs "blogs" because of the loadedness of the term, but certainly I've seen a strong growing community of very bright people beginning to actually have discussions and the communities naturally form from social communities that already exist with the addition of a little help from your friendly search engine. (Better algorithmic methods to derive community structure from existing web logs and the content of their entries is a topic for another article, but they've been brewing, so they may find themselves an entry.) This new intensification, most notably in the academic specialties, is what motivates me to become a more frequent web logger myself. For better or worse, my shit is out there, and I'm hoping, for lack of a better system, it may be for the better.

[Blog at WordPress.com.](http://Blog.atWordPress.com)