Last mile software development

Writing modern software for bench scientists

Thomas Sibley

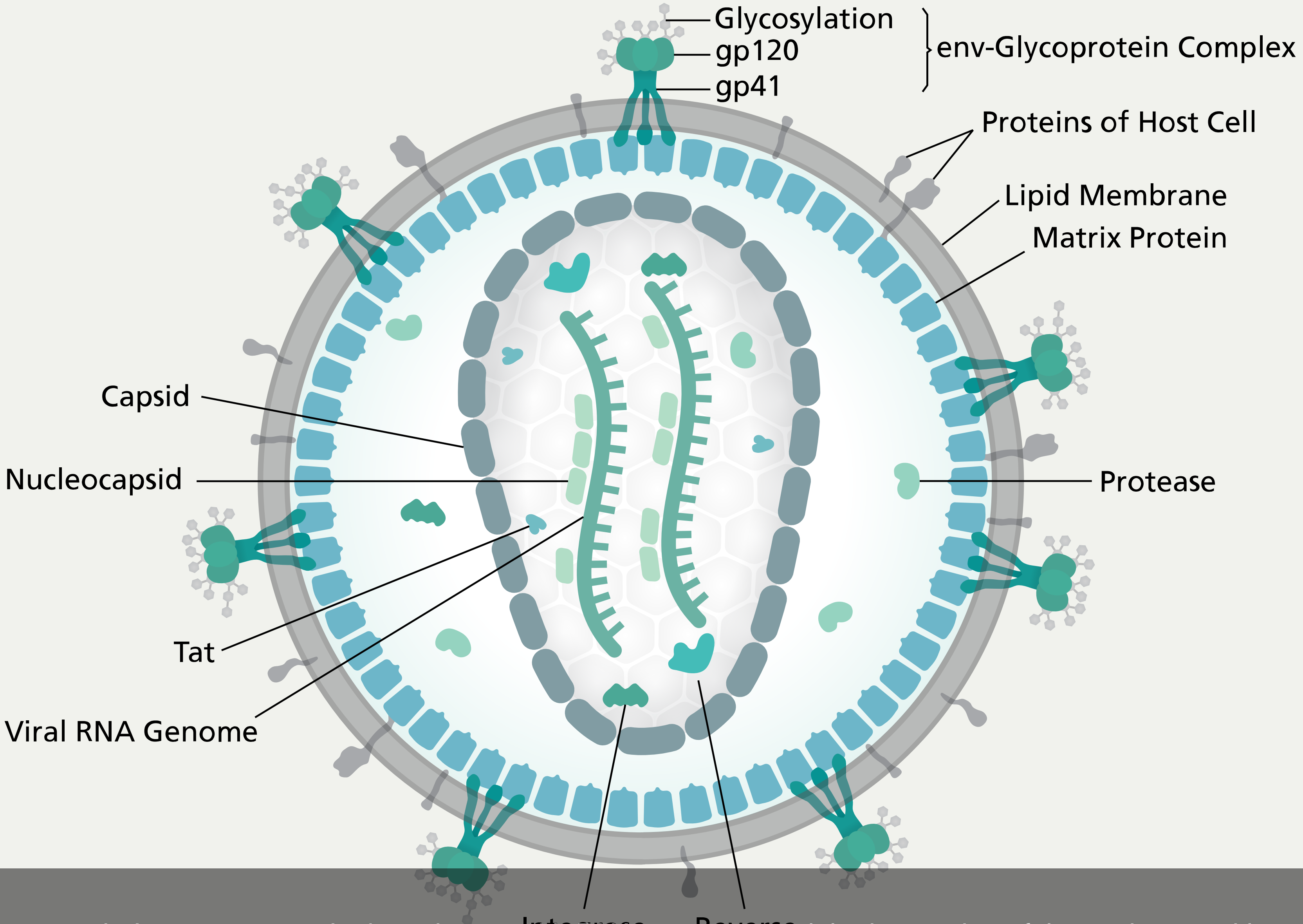
The Perl Conference 2017
Alexandria, VA

Hello! My name is Thomas Sibley. I'm here today to talk about modern software development in a biology research lab.

Mullins Molecular Retrovirology Lab

University of Washington

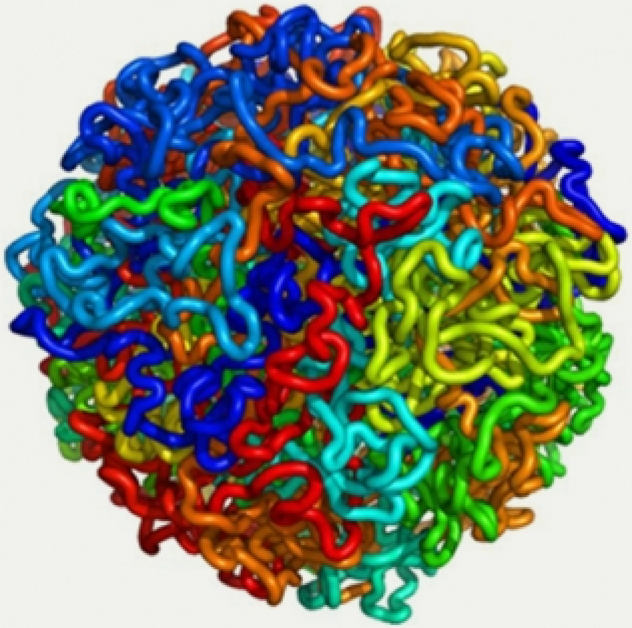
I work in the Mullins Molecular Retrovirology Lab at the University of Washington in Seattle.



Molecular retrovirology means we look at viruses with RNA genomes and the interaction of these viruses with molecules in the cell. We approach questions about the evolution of viruses and their interactions with human cells using a variety of wet lab techniques at the lab bench and "dry lab" bioinformatics techniques at the computer. Each informs the other, and often exploration of questions ping pongs back and forth between the two.


CLOCK
IN
OUT
CLEAR
MAX
MIN
LIGHT
SPRINGFIELD

My responsibilities cover everything involving a computer in the lab, from analyzing data to writing new apps to managing our racks of hardware. I've been in the lab for going on four years now and have helped modernize existing applications and kick off new ones.



You've probably heard horror stories about the kind of spaghetti, write-only code that academic research produces, or even worse, maybe you've looked at the BioPerl source code.

Ok, that's a cheap shot, but I'm here to tell you that not all software in science is terrible!

Act I: The Last Mile

Act II: Improving the Situation

Act III: Is this for you?

This will be a talk in three acts.

In the first act, I'll explore this idea of the last mile as I think it applies to software in science.

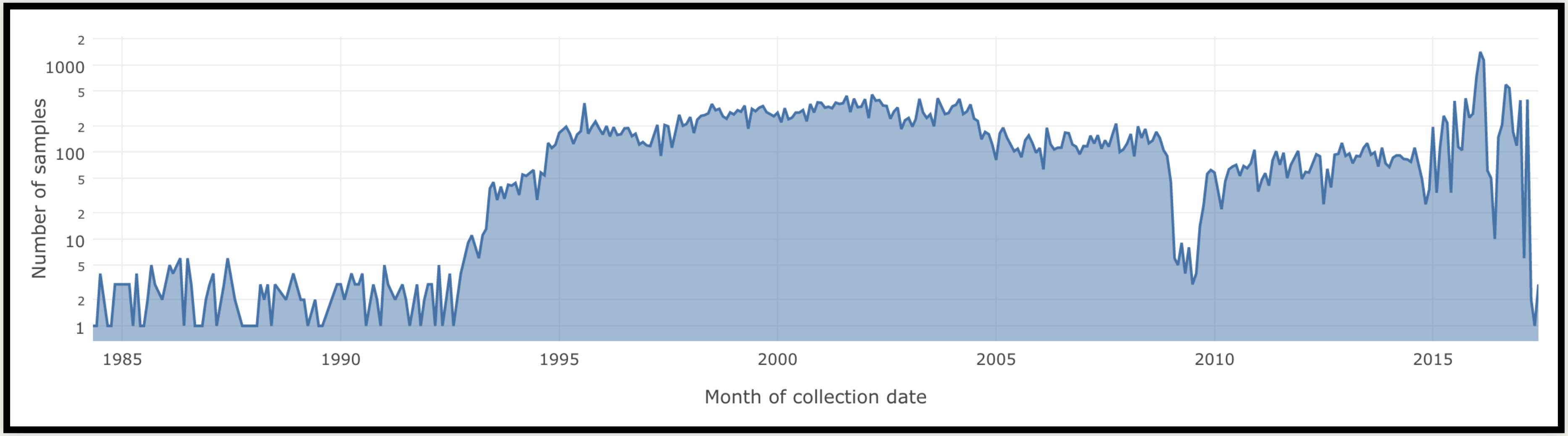
In the second act, I'll talk about the kind of work I do in the lab and show examples of improvements we've made to the computing practices, viewed through the lens of lessons learned.

In the final act, I'll talk about why you too might want to work in a science lab.

Act I

The Last Mile

Let's get started.



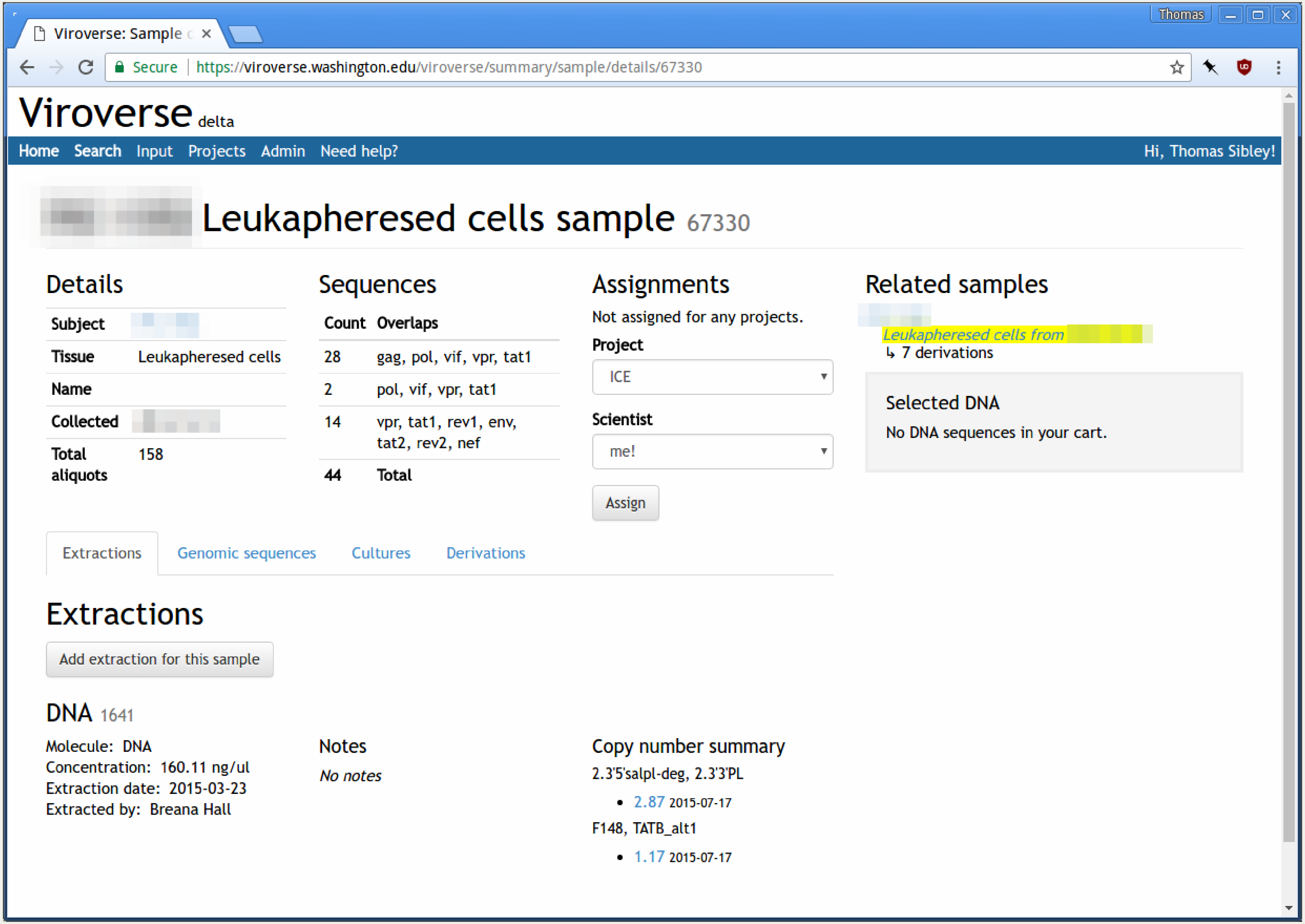
The Mullins Lab has been around for 23 years at UW and for 12 years before that at Stanford and Harvard. That's a lot of time to generate data! Some of the lab's ongoing projects span decades, with new data being collected from the start up until now.

This plot shows the collection dates of samples that the lab manages and works with.

The success of those projects is directly related to the lab's ability to make sense of the data over time and not lose it to the frequent turnover of students and postdocs or misplace it amidst shelves of lab notebooks.

Evan Silberman

Lab notebooks are an indispensible tool, but they don't scale.


Viroverse: Sample
Thomas
Secure
https://viroverse.washington.edu/viroverse/summary/sample/details/67330
Viroverse delta
Home Search Input Projects Admin Need help?
Hi, Thomas Sibley!
Leukapheresed cells sample 67330
Details
Subject
Tissue Leukapheresed cells
Name
Collected
Total aliquots 158
Sequences
Count Overlaps
28 gag, pol, vif, vpr, tat1
2 pol, vif, vpr, tat1
14 vpr, tat1, rev1, env, tat2, rev2, nef
44 Total
Assignments
Not assigned for any projects.
Project
ICE
Scientist
me!
Assign
Related samples
Leukapheresed cells from
↳ 7 derivations
Selected DNA
No DNA sequences in your cart.
Extractions
Genomic sequences
Cultures
Derivations
Extractions
Add extraction for this sample
DNA 1641
Molecule: DNA
Concentration: 160.11 ng/ul
Extraction date: 2015-03-23
Extracted by: Breana Hall
Notes
No notes
Copy number summary
2.3'5'salpl-deg, 2.3'3'PL
2.87 2015-07-17
F148, TATB_alt1
1.17 2015-07-17

Helping the lab make sense of data over the longer term and preserve it for future study is an in-house informatics application called Viroverse.

This is a quick example of a detail page for a sample in the Viroverse system.

Bit rot is a real concern though, and having the data doesn't matter if the software for accessing it doesn't work well.

CVS/mercurial

```
commit 47eca7460a6391be0bc532ab70e040736379439a
Author: [redacted] <[redacted]@uw.edu>
Date:   Tue Oct 20 23:23:21 2009 +0000

    synchronize Mercurial and CVS repositories

159 files changed, 14093 insertions(+), 1416 deletions(-)
```

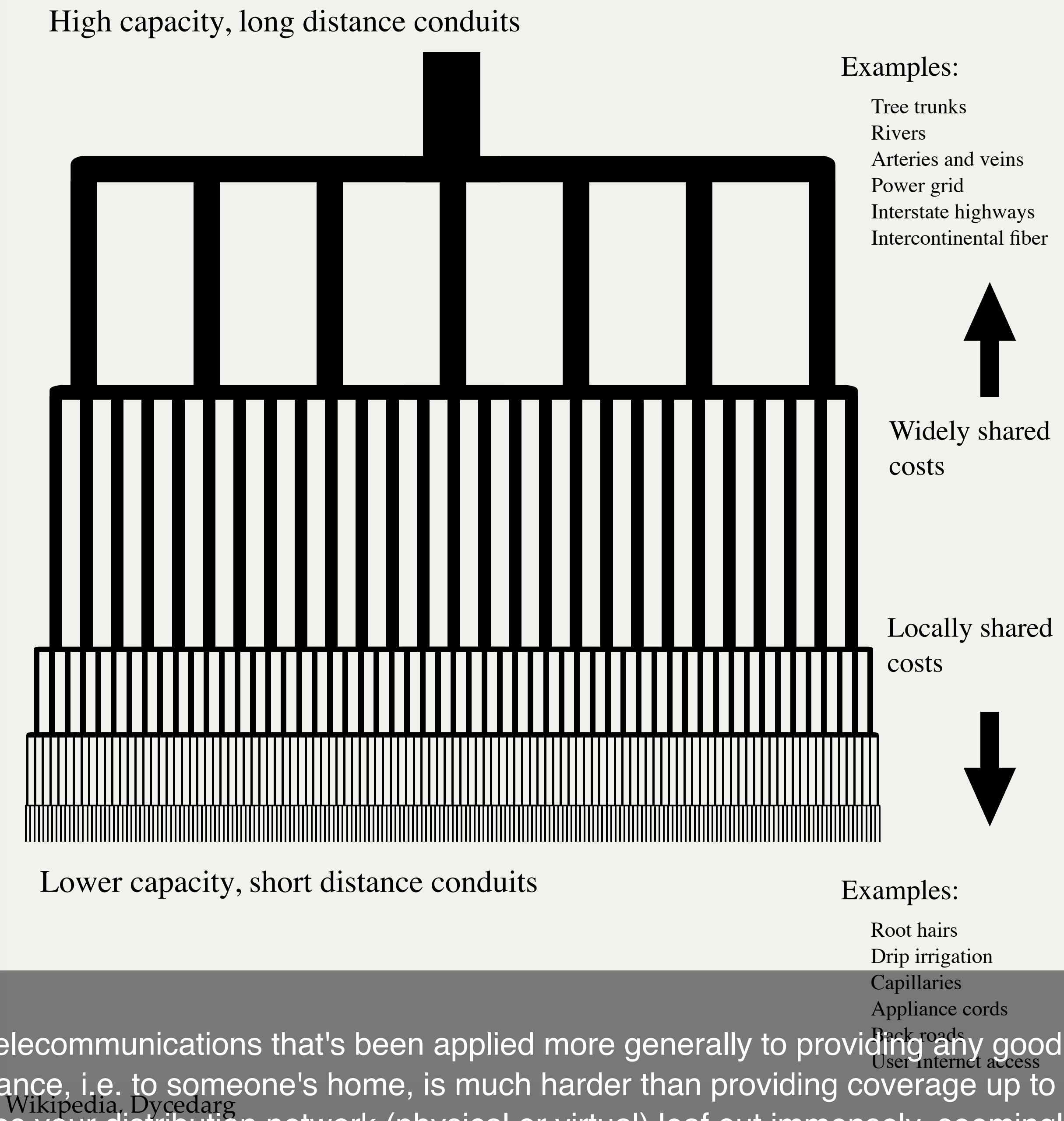
When I first started, Viroverse didn't look like the previous picture. It used cobbled together YUI2 components everywhere, was running on mod_perl, and using not just a homemade ORM but also Class::DBI *and* DBIx::Class. It was version controlled in an unholy combination of centralized CVS and private Mercurial repositories.

```
commit 2a7d6c4bdab7993e0f1d3ac792545ba05b9e406c
Author: [redacted] <[redacted]@uw.edu>
Date:   Fri Nov 12 22:13:46 2010 +0000

    (no message)

41 files changed, 3015 insertions(+), 377 deletions(-)
```

Over about a decade, various individuals had made their mark on the application. After a while you could pretty much tell who wrote what by how the code looked and how well it functioned.

Most of the people in my position before me had come to the job with a background primarily in biology not software. The development practices that had been used were *years* behind current best practices. Coming from an open-source and commercial software background, I saw many opportunities for modernization. It was clear that many improvements in the field, from better development tools to design practices to error handling to user experience, simply hadn't reached the lab.

I don't attribute this to a lack of caring on the part of the folks before me. Rather, I think for reasons ranging from the obtuseness of modern software stacks to the traditional funding structures in biology, that the advances in software and computing just hadn't *reached them yet*.



There's this idea in telecommunications that's been applied more generally to providing any good or service: covering the "last mile" of distance, i.e. to someone's home, is much harder than providing coverage up to that point. It's this "last mile" that necessitates your distribution network (physical or virtual) leaf out immensely, seemingly immeasurably compared to more concentrated service delivery points.

Mail services are a good example. Every day the US Postal Service touches, often *literally*, every mailbox in America. USPS would be a much smaller business if it just had to get mail to regional distribution centers or even local post offices. The difficulty and expense of bridging that "last mile" is the reason why private mail carriers like UPS and FedEx, as they handled more and more packages with the rise of online shopping, started using USPS for final delivery. USPS already had a "last mile" network because it's a much older organization that had the mandate to do so.

People *do* care

When I first started in the lab, I thought terrible software was just par for the course because no one cared as long as it appeared to work once.

I now see it as a last mile problem. It's not that the field doesn't care about producing bad, error-prone code that reinvents previously solved wheels, but that the field doesn't have access to modern practices and technology when it comes to software and, more broadly, computing.

The tech industry is busy building gleaming, glistening towers up in the clouds. While it's busy "innovating" by putting software in everything from toothbrushes to mugs, the industry doesn't seem to have much interest in actually trying to advance other fields by bringing to them the bread and butter tech we've all had for a while now, like snappy, reactive web apps.

If you feed the horse enough oats, some will pass through to the road for the sparrows.

—John Kenneth Galbraith

I don't see many people who I think of as tech ambassadors, people who try to keep one foot in tech and one foot in another field and facilitate knowledge transfer. It seems that everyone thinks tech will just trickle down eventually. An older name for trickle-down theory was horse-and-sparrow theory.

The tech industry is easy to blame, but it's not all its fault of course. Traditional funding structures in biology, for example, can make it hard to competively hire professional developers. Generational and institutional biases often devalue staff roles in science, making it harder to justify bringing in outside talent. Neither of these are universal, but they are impediments that are slowing breaking down.

Act II

Improving the Situation

While I can't affect funding structures, I *can* help dispense with the myths that all software in science has to be terrible and that the people writing it have to be trained scientists! Perhaps I can even pique your interest in bringing professional software development to research science.

From day one, my goal was to improve the situation I found. I didn't know much about biology at the time, but I knew what rotten software smelled like. The name of the game was to throw out what was rotten and keep what was sound, then build from there. Since I didn't have a big picture of what the lab needed, I hoped that by relentlessly improving the computing environment I would find ways to help out everyone.

```
# primerMatrix's tip is a commit which doesn't belong in time or
# topology; make a new branch and correct the primerMatrix branch
cd $(GIT); \
    git branch -m primerMatrix assembly-in-memory; \
    git branch primerMatrix assembly-in-memory~; \
    git rebase \
        --committer-date-is-author-date \
        --onto `git merge-base master assembly-in-memory` \
        primerMatrix assembly-in-memory; \
    git checkout master
cd $(GIT); \
    ATTIC_FIXUP_START=`git log --format=%H --diff-filter=D -- mvc
    git filter-branch \
    --index-filter "bash $(PWD)/filter-fixup-attic $(PWD)/$(GIT)
    --parent-filter "perl $(PWD)/filter-add-merge-parents" \
    --msg-filter "perl $(PWD)/filter-msg-cleanup" \
    --commit-filter '[[ -n `git rev-list $$GIT_COMMIT..start` ]]
    --tag-name-filter cat \
    -- \
    --all \
        && git reset --hard master \
        && git for-each-ref --format="%(refname)" refs/original/
            | xargs -n1 git update-ref -d \
        && git reflog expire --expire=now --all \
        && git gc --prune=now
```

Progress was slow at first. I spent about my first three weeks converting a mess of a global cvs repository to the best git repository I could manage.

More weeks were spent improving the deployment infrastructure for Viroverse, the lab's primary web application, so that I could deploy changes during working hours without accidentally eating someone's data they were in the middle of inputting. Gradually, tangible improvements compounded and real progress was easier to make.

A year later, the results were good enough to convince my boss, Jim, to let me hire a coworker to work alongside me. Since then, improvement's been much more rampant and together we've been able to feasibly tackle larger projects.

What Did We Learn?

Over these past years, I've learned a lot about how a lab works and how science works, and I want to share those lessons and examples with you. These are lessons I remind myself of and try to work by, because often my default reaction is the opposite. I think they give a nice glimpse into my day-to-day role where I strive to make excellent software for scientists. The first is...

Enable people to use better practices

Enable people to use better practices. What I mean by this is that our work should make it easier for people to use best practices rather than try to force them to do so.

A while back we had an issue in lab around how sequences were named. Scientists would generate sequences in the wet lab, attach mostly standardized names, and upload them into our database, Viroverse. But when it came time to do the analysis, instead of downloading the sequences en masse from Viroverse, people would collect all the data files from dozens of personal folders on the shared file server.

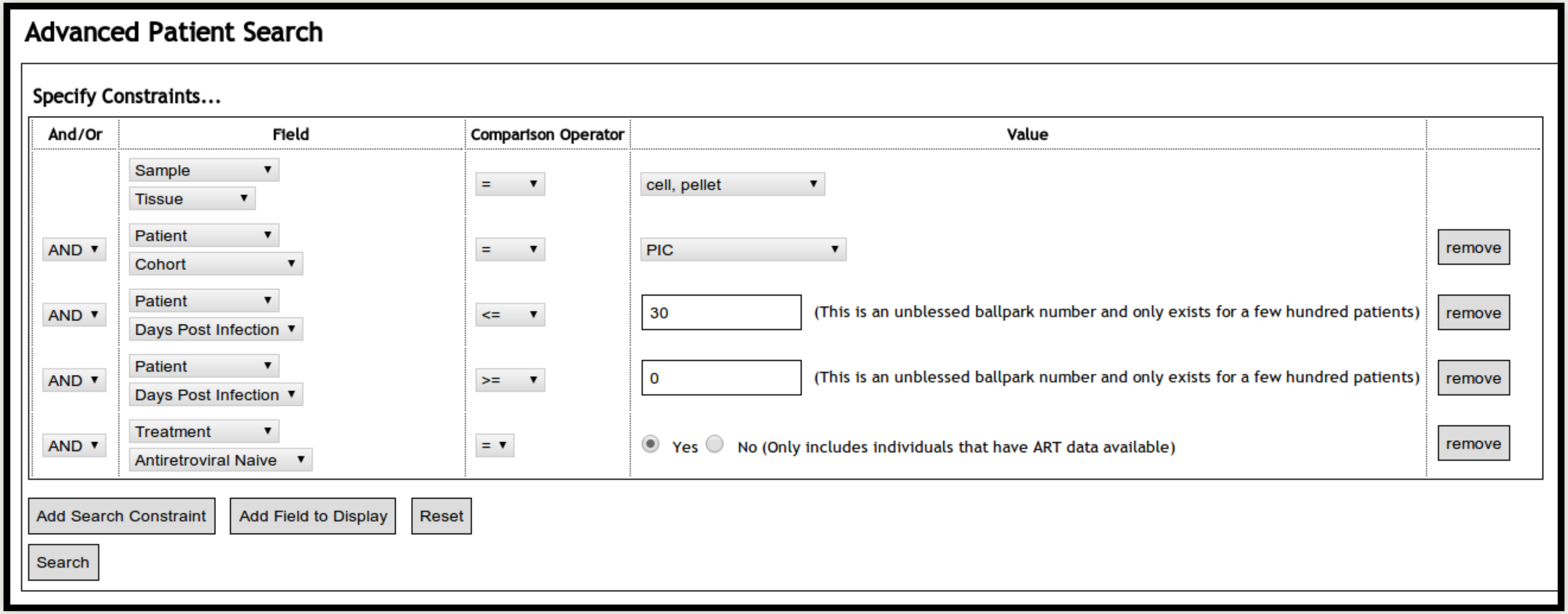
This wasn't ideal because Viroverse was supposed to be the authority for our sequencing data. It was important to keep the assigned accession number attached to sequences for data provenance, and sequences were also sometimes revised in Viroverse without the data files on the file server being updated.

XXWI30123780XXx990414XXpXXXX12873660

The problem was that it was hard to find and download all the sequences you needed, and when you did, each sequence was assigned a name that, as far as anyone was concerned, was just noise to them that got in the way during analysis.

Names looked like this gibberish, which is an insane mix of fixed length fields with some variable length fields thrown in the middle!

Since the sequences on the file server were more sanely named, people naturally opted to use those instead. Instead of trying to enforce a policy, we made it easier to get sequences from Viroverse than to trawl the file server for them. First, improved searching made it easier to get the batch of sequences you needed.



To replace a typically bad "advanced search" like this, which no one could use, I built...

Viroverse: Sequenc ×

Thomas

Secure | https://viroverse.washington.edu/viroverse/sequence?rows=12&page=66

Viroverse delta

Home Search Input Projects Admin Need help? Hi, Thomas Sibley!

Sequences

Sequence, PCR, or sample name contains...

Download results Share this search

Cohort sort by name

Unknown	12,448
PIC	9,927
Koronis	5,135
MACS	3,746
WIHS	3,545
NDRI	3,014
Partners in Prevention	2,065
AIEDRP	1,930
UCLA MICL	1,581
NNAB	1,290

Tissue sort by name

plasma	26,384
Unknown	13,605
PBMC	2,376
T cell lysate	953
lymph node	556
bone marrow	545
lung	527
intestine (large)	525
intestine (small)	516
GALT	425

NA Type sort by name

Unknown	35,539
RNA	7,537
DNA	7,324

Protocol sort by name

Genomic	48,961
Bisulfite	741
ISLA	518
Alu integration site	180

Scientist sort by name

Jill Stoddard	9,809
Laura Heath	9,640
Kim Wong	6,626
Indira Genowati	3,554
Hong Zhao	2,999
Dylan Westfall	2,738
Unknown	1,530
Yi Liu	1,464
Lennie Chen	1,158
Christine Rousseau	843

Region sort by name

env	27,329
gp120	26,153
v3	25,401
v4	23,796
v5	23,654
gag	22,028
p24	17,091
p17	16,904
pol	14,518
v2	12,862

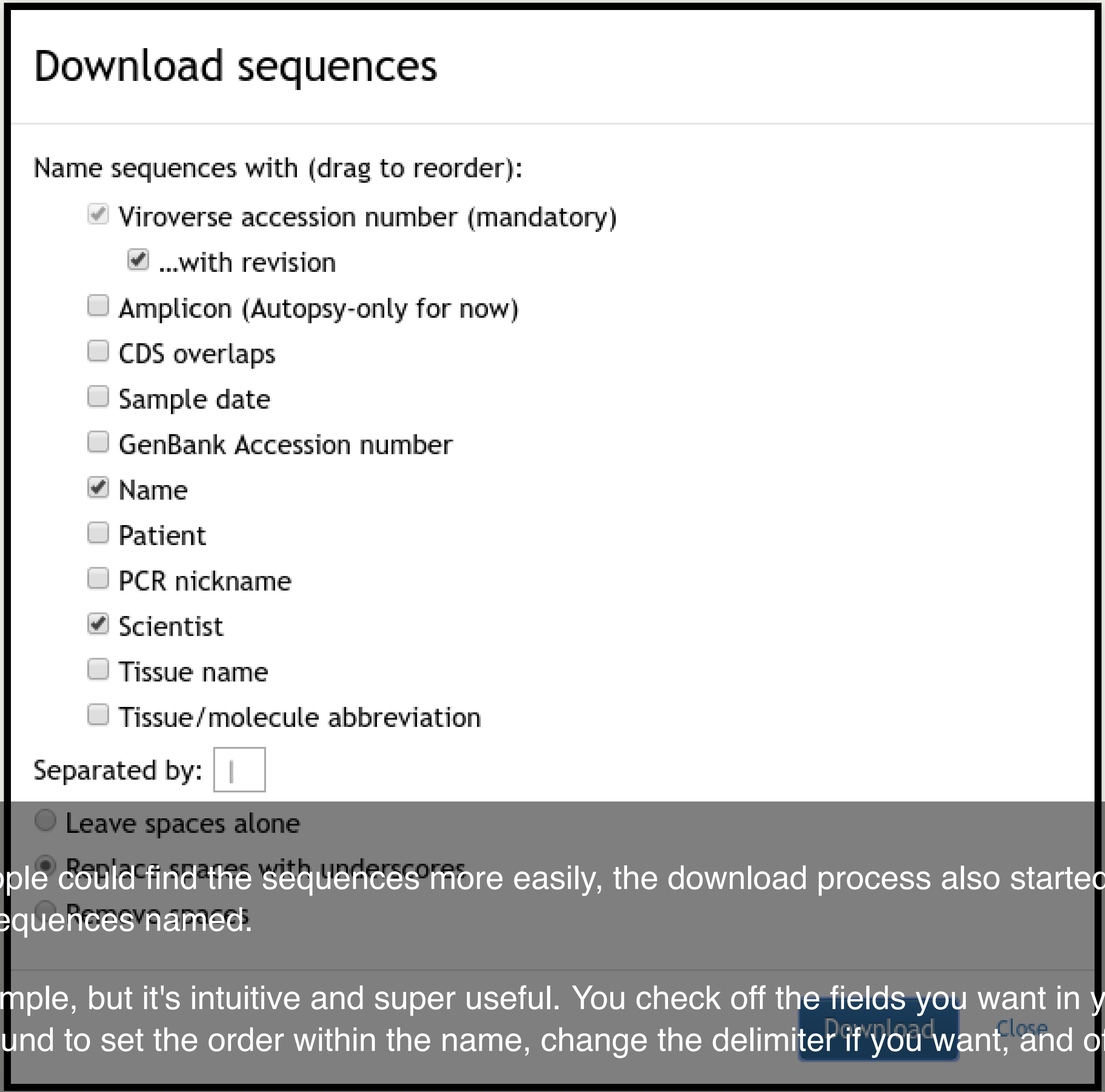
There are 50,400 sequences in all of Viroverse. Showing sequences 781 through 792, starting with the most recent.

« Previous Next »

#	Name	Scientist	NA Type	Protocol	Entered	Sample Name
56785.1	1517.E3	Alec Pankow	DNA	Genomic	2017-05-30	TC01517.E3d10
56787.1	1521.H8	Alec Pankow	DNA	Genomic	2017-05-30	TC01521.H8d10
56786.1	1522.E2	Alec Pankow	DNA	Genomic	2017-05-30	TC01522.E2d10
56788.1	1524.B5	Alec Pankow	DNA	Genomic	2017-05-30	TC01524.B5d10
56782.1	1093.H12-LH	Lennie Chen	DNA	Genomic	2017-05-30	TC01093.H12d10

...a faceted search interface which updates immediately when you click a facet value to start searching. It's immediately graspable by playing around, and you don't have to already know what kinds of values you can search for. They're all just shown! Faceted searching is one of those improvements that is all over the tech world, but that I rarely see outside of it.

We're also using a slimmed down version of this same component to replace some data tables since it works nicely as a general filter interface.



Crucially, now that people could find the sequences more easily, the download process also started letting you choose how you wanted the sequences named.

This widget is pretty simple, but it's intuitive and super useful. You check off the fields you want in your sequence names, you drag the fields around to set the order within the name, change the delimiter if you want, and off you go, with data tailored to your needs.

More people started fetching sequences from Viroverse, with the assigned ids intact, and they didn't have to spend time renaming sequences themselves via careful series of find-and-replace operations in TextWrangler.

Don’t collect data you don’t plan to use

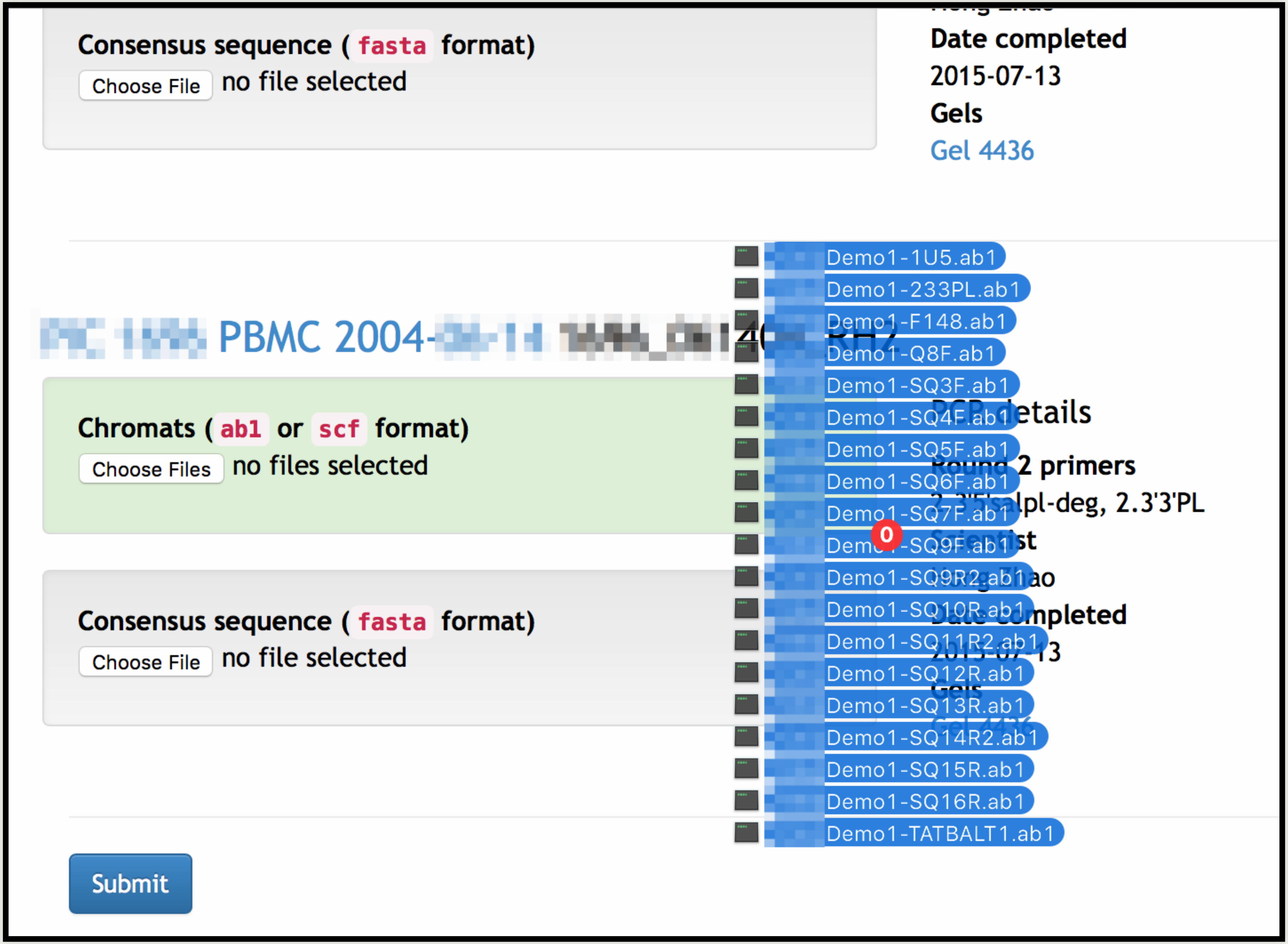
Don't collect data you don't plan to use.

While it's tempting to collect as many details as possible in the hope that it'll be useful someday, all it does in the short-term is add work for everyone. Bench scientists already keep detailed lab notebooks, so it's not as if the data is gone forever if you do find you need it later.

It also turns out that if you ask scientists to enter information that they know is never used, they won't bother to enter it accurately. You might as well just stop collecting it, which will be easier for everyone.

We ran into this with our sequence upload workflow. Previously people were required to tediously "paint" a diagram of how they were submitting their samples to the sequencing facility. This interface required repetitively dragging over the diagram to specify up to dozens of values for each sample sequenced. After doing so, they still had to manually prepare the same information in a different format for the sequencing facility.

When scientists eventually got back their sequencing results, they then had to upload the several dozen files and manually drag-and-drop match them with the correct location on their diagrams. It was slow and error-prone and ultimately collecting data that we didn't need. The result was that people fabricated their metadata diagrams in ways that made the data input steps easier, rather than sticking to what they really did.


Consensus sequence (fasta format)
Choose File
no file selected
Date completed
2015-07-13
Gels
Gel 4436
PBMC 2004-
Chromats (ab1 or scf format)
Choose Files
no files selected
Consensus sequence (fasta format)
Choose File
no file selected
Submit
Demo1-1U5.ab1
Demo1-233PL.ab1
Demo1-F148.ab1
Demo1-Q8F.ab1
Demo1-SQ3F.ab1
Demo1-SQ4F.ab1
Demo1-SQ5F.ab1
Demo1-SQ6F.ab1
Demo1-SQ7F.ab1
Demo1-SQ9F.ab1
Demo1-SQ9R2.ab1
Demo1-SQ10R.ab1
Demo1-SQ11R2.ab1
Demo1-SQ12R.ab1
Demo1-SQ13R.ab1
Demo1-SQ14R2.ab1
Demo1-SQ15R.ab1
Demo1-SQ16R.ab1
Demo1-TATBALT1.ab1
2 primers
lpl-deg, 2.3'3'PL

In the end, we replaced that part of the input workflow with this batch upload process incorporating heuristics for automatic data matching.

Confirm sequencing primers

[redacted]_PBD_121214_LH01

File	Primer
[redacted]Demo1-1U5.ab1	1.U5
[redacted]Demo1-233PL.ab1	2.3.3PLC
[redacted]Demo1-F148.ab1	F148
[redacted]Demo1-Q8F.ab1	1.3'3'(PRO)
[redacted]Demo1-SQ3F.ab1	SQ3F(SK38)
[redacted]Demo1-SQ4F.ab1	SQ4F
[redacted]Demo1-SQ5F.ab1	SQ5FC
[redacted]Demo1-SQ6F.ab1	SQ6F
[redacted]Demo1-SQ7F.ab1	SQ7F(2)
[redacted]Demo1-SQ9F.ab1	SQ9F
[redacted]Demo1-SQ9R2.ab1	SQ9R(2)
[redacted]Demo1-SQ10R.ab1	SQ10R(2)
[redacted]Demo1-SQ11R2.ab1	SQ11R(2)
[redacted]Demo1-SQ12R.ab1	SQ12R(2)
[redacted]Demo1-SQ13R.ab1	SQ13R(2)
[redacted]Demo1-SQ14R2.ab1	SQ14R(2)
[redacted]Demo1-SQ15R.ab1	SQ15R(SK39)_H
[redacted]Demo1-SQ16R.ab1	SQ16RC
[redacted]Demo1-TATBALT1.ab1	TATB_alt1

Submit

After upload, all our scientists need to do now is review the automatic matches, fix any mistakes, and click save.

People were so appreciative of this change because we *stopped wasting their time*.

Data is useless if not in front of someone's eyes

Data is useless unless it's in front of someone's eyeballs. When I first started, a lot of the data we had wasn't very visible to the lab. I had access to it, if I knew it existed, and could pull it up on demand, but the lab didn't always know what data we had or have the ability to see it.

Invisible data doesn't inspire questions and generate hypotheses. It doesn't register as available or pertinent when planning analyses. Probably about a third of my overall job is thinking critically about how to best surface and present the data we already have.

Scientists are naturally curious, and so if there's data in front of them, they'll look at it and ask questions of it. There are all sorts of good ways to get data in front of people.

Many people dismiss data tables as boring and immediately reach for plots and charts and diagrams, but I'd take a thoughtful, well-designed data table over a poorly thought-out visualization any day.

There are all kinds of data tables, and they adapt nicely to different needs and constraints.

Viroverse: PIC

trsibley@uw.edu

Secure https://viroverse.washington.edu/viroverse/cohort/1

Viroverse delta

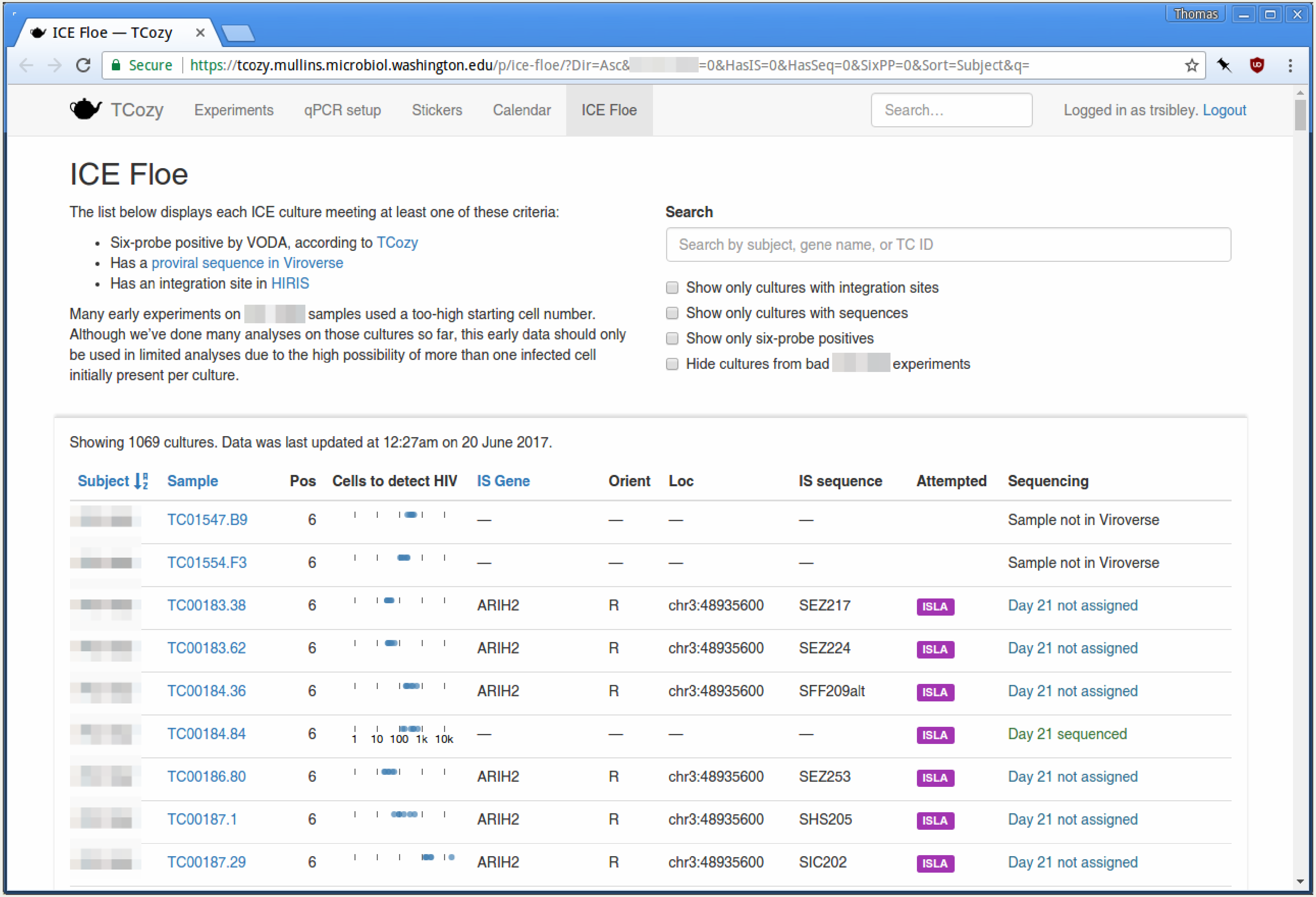
Home Search Input Projects Admin Need help? Hi, Thomas Sibley!

PIC

Showing **924** subjects from this cohort.

Subject ID	Est. Date Infected	First visit	Viral load history	Latest visit	Years Infected	Highest VL	Lowest VL	# Plasma	# PBMC	# Leuka	# Other
[illegible]	[illegible]	[illegible]24		[illegible]13	2.5	146,800	<LLOQ	5	2	0	2
[illegible]	[illegible]	[illegible]24		[illegible]05	3.0	272,700	10,120	4	10	0	6
[illegible]		[illegible]01		[illegible]01				0	0	0	1
[illegible]	[illegible]	[illegible]09		[illegible]23	5.5	423,990	3,623	8	11	0	15
[illegible]	[illegible]	[illegible]25		[illegible]23	9.9	6,095	<LLOQ	21	19	0	32
[illegible]		[illegible]01		[illegible]01				0	1	0	1
[illegible]	[illegible]	[illegible]25		[illegible]08	15.7	268,710	<LLOQ	53	72	1	78
[illegible]	[illegible]	[illegible]24		[illegible]06	3.4	7,663	<LLOQ	3	1	0	0
[illegible]	[illegible]	[illegible]15		[illegible]05	15.4	389,500	<LLOQ	45	68	0	70
[illegible]	[illegible]	[illegible]24		[illegible]24	6.5	279,100	26,290	0	1	0	0
[illegible]	[illegible]	[illegible]28		[illegible]25	7.0	1,600,000	540	25	21	0	38
[illegible]	[illegible]	[illegible]08		[illegible]14	12.7	117,336	<LLOQ	38	52	3	66
[illegible]	[illegible]	[illegible]08		[illegible]12	4.5	639,620	35,417	14	15	0	15
[illegible]	[illegible]	[illegible]27		[illegible]12	6.5	129,490	13,120	11	8	0	9
[illegible]	[illegible]	[illegible]09		[illegible]25	1.1	65,150	<LLOQ	1	1	0	0
[illegible]	[illegible]	[illegible]16		[illegible]19	0.2	142,700	10,010	2	0	0	0
[illegible]		[illegible]01		[illegible]01				0	0	0	1
[illegible]	[illegible]	[illegible]28		[illegible]17	3.3	210,000	10,820	12	11	0	20

You can start simple with a static but information-dense table for basic scanning. Sparklines are helpful for increasing information density in this table, showing longitudinal data that otherwise wouldn't fit but is important when skimming the table to find suitable study subjects.



ICE Floe — TCozy

Secure | https://tcozy.mullins.microbiol.washington.edu/p/ice-floe/?Dir=Asc& =0&HasIS=0&HasSeq=0&SixPP=0&Sort=Subject&q=

TCozy Experiments qPCR setup Stickers Calendar ICE Floe Search… Logged in as trsibley. Logout

ICE Floe

The list below displays each ICE culture meeting at least one of these criteria:

- Six-probe positive by VODA, according to TCozy
- Has a proviral sequence in Viroverse
- Has an integration site in HIRIS

Many early experiments on samples used a too-high starting cell number. Although we've done many analyses on those cultures so far, this early data should only be used in limited analyses due to the high possibility of more than one infected cell initially present per culture.

Search

Search by subject, gene name, or TC ID

- [ ] Show only cultures with integration sites
- [ ] Show only cultures with sequences
- [ ] Show only six-probe positives
- [ ] Hide cultures from bad experiments

Showing 1069 cultures. Data was last updated at 12:27am on 20 June 2017.

Subject	Sample	Pos	Cells to detect HIV	IS Gene	Orient	Loc	IS sequence	Attempted	Sequencing
	TC01547.B9	6		—	—	—	—		Sample not in Viroverse
	TC01554.F3	6		—	—	—	—		Sample not in Viroverse
	TC00183.38	6		ARIH2	R	chr3:48935600	SEZ217	ISLA	Day 21 not assigned
	TC00183.62	6		ARIH2	R	chr3:48935600	SEZ224	ISLA	Day 21 not assigned
	TC00184.36	6		ARIH2	R	chr3:48935600	SFF209alt	ISLA	Day 21 not assigned
	TC00184.84	6	1 10 100 1k 10k	—	—	—	—	ISLA	Day 21 sequenced
	TC00186.80	6		ARIH2	R	chr3:48935600	SEZ253	ISLA	Day 21 not assigned
	TC00187.1	6		ARIH2	R	chr3:48935600	SHS205	ISLA	Day 21 not assigned
	TC00187.29	6		ARIH2	R	chr3:48935600	SIC202	ISLA	Day 21 not assigned

From there you can add filtering and sorting by pertinent properties, and graphical labels for easier scanning.

Quantities

Filter table…

Clear highlighted | Negatives | Censored

All screened cultures that were positive for at least one HIV gene. There are no censored quantities.

Units | copies | cells/copy | copies/10⁶ cells

Sample name	Culture day	HTFR (cells)	PR (cells/copy)	INT (cells/copy)	TAT (cells/copy)
TC01482.E10	7	21,724	40,427.19	18,847.27	0
TC01482.E11	7	32,353	66,207.27	20,665.96	0
TC01482.F12	7	25,286	0	3,757.01	160,762.65
TC01482.H2	7	33,725	0	2,553.10	16,807.06
TC01483.G5	7	32,384	31,463.67	22,414.17	0
TC01483.H3	7	21,736	35,392.68	0	0

As the data becomes more complex, you can present different views of the same data with a toggle.

These buttons convert the units in the table because different units are easier for some tasks than others. The raw units are in total copies detected, but it's often useful to know the frequency of detection relative to the baseline.


TC00181 — TCozy
QPCR01864 — TCo
Thomas
Secure
https://tcozy.mullins.microbiol.washington.edu/qpcr/plate/QPCR01864
TCozy
Experiments
qPCR setup
Stickers
Calendar
ICE Floe
Search...
Logged in as trsibley. Logout
QPCR01864 #1021
← QPCR01863
QPCR01865 →
10
1
0.1
0.01
0.001
0.0001
Δ Rn
HTFR
PR
INT
TAT
STANDARD
UNKNOWN
NTC
0.03 HTFR, INT, PR, TAT
cycle
5
10
15
20
25
30
35
40
45
Properties
Experiments
total CD4+ T cells - 1
Probes
198
Reference
ROX
Finished
04:18:21 on 11 Feb 2017
Run by
tamara8
Instrument
sds7500
Filename
\\Hercules\ice\ICE Experiments\ _CD8-_T-cells - 1\2nd screen\eds\QPCR01864_TB.eds
Files
QPCR01864_TB_data.xls by tamara8 (#8581)
Standard curves
Target
R²
Efficiency
HTFR 0.989 104.4%
INT 0.986 84.9%
PR 0.993 97.2%
TAT 0.997 101.0%
Controls
Name
Quantity
Quantities
Filter table...
Clear highlighted
Negatives
Censored
Units
copies
cells/copy
copies/10⁶ cells
All screened cultures that were positive for at least one HIV gene. There are no censored quantities.
HTFR
PR
INT
TAT

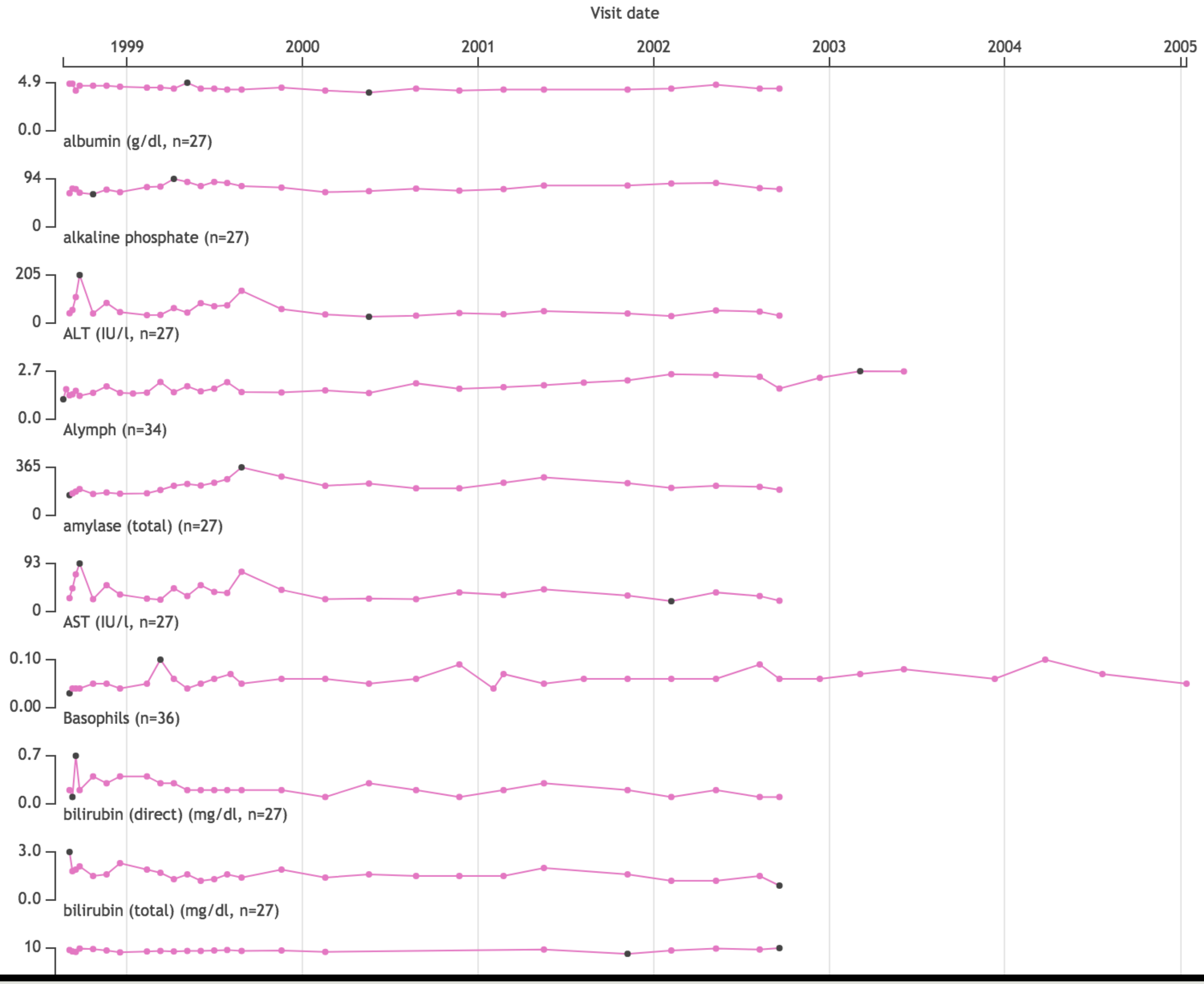
Sometimes its nice to interactively link a table with a visualization so that people can flip between the two as they dig into the data. The previous table and this plot of data are linked, so that selecting a curve in the plot selects the related data in the table, and vice versa.

Specimens | **Labs** | Treatment | Sequences | Virology | Epitopes

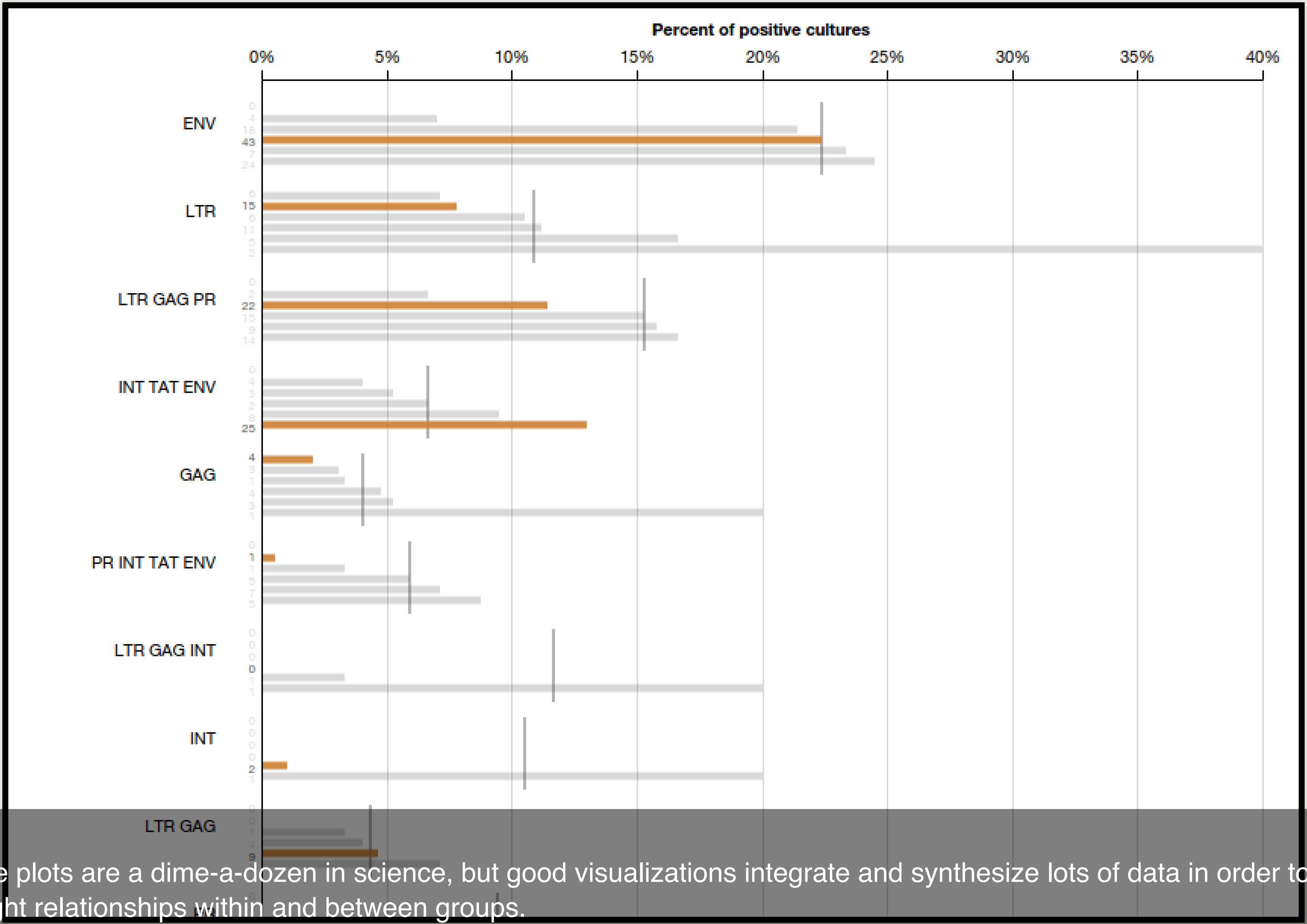
	1998-08-06	1998-08-10	1998-08-18	1998-09-01	1998-0
albumin	4.2	4.3			
alkaline phosphate		60			
ALT	21	23			
Alymph	1.529	4.018		2.568	2
amylase (total)	189	256			
AST	21	30			
Basophils	0.04				
bilirubin (direct)	0.4	0.3			
bilirubin (total)	0.6	0.5			
BUN	9	14			
Calcium	9.3	9.6			
CD3	963	2572		1721	
CD4	443	884		745	
CD4 calc	441	783		745	
CD8	489	1567		924	
Chloride	103	101			
Chol	132				
Chol/HDL ratio	4.9				
CK activity	44	68			

When a data table becomes unwieldy, like this all-scrolling, all-dancing horror, you can upgrade it directly to the visualization and regain comprehension.

Numeric lab assay results are shown in each panel. Units for each assay, when known, are shown in parentheses after the assay name. The earliest minimum and maximum values are denoted by black dots; later values may also reach the minimum or maximum but are not similarly denoted. The values of specific points are shown on hover.

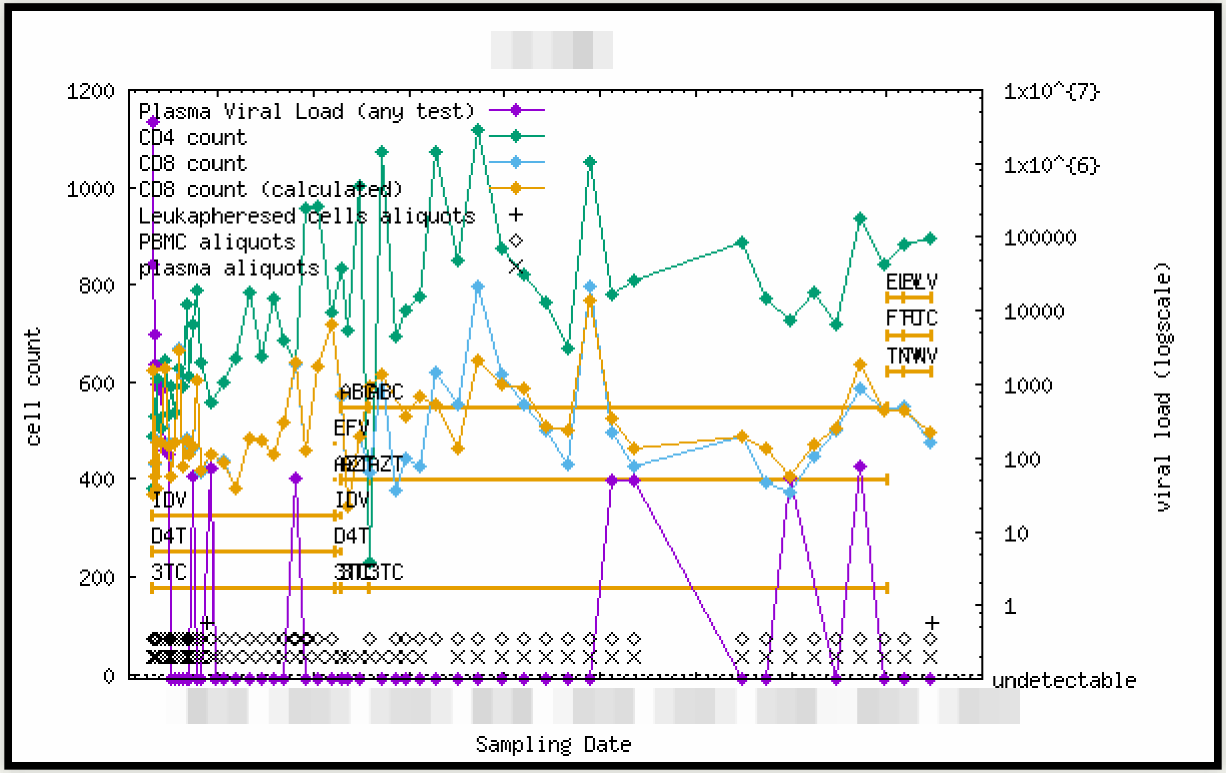


This is what is replacing that table, not only increasing information density but also clarity. And no more horizontal scrolling!

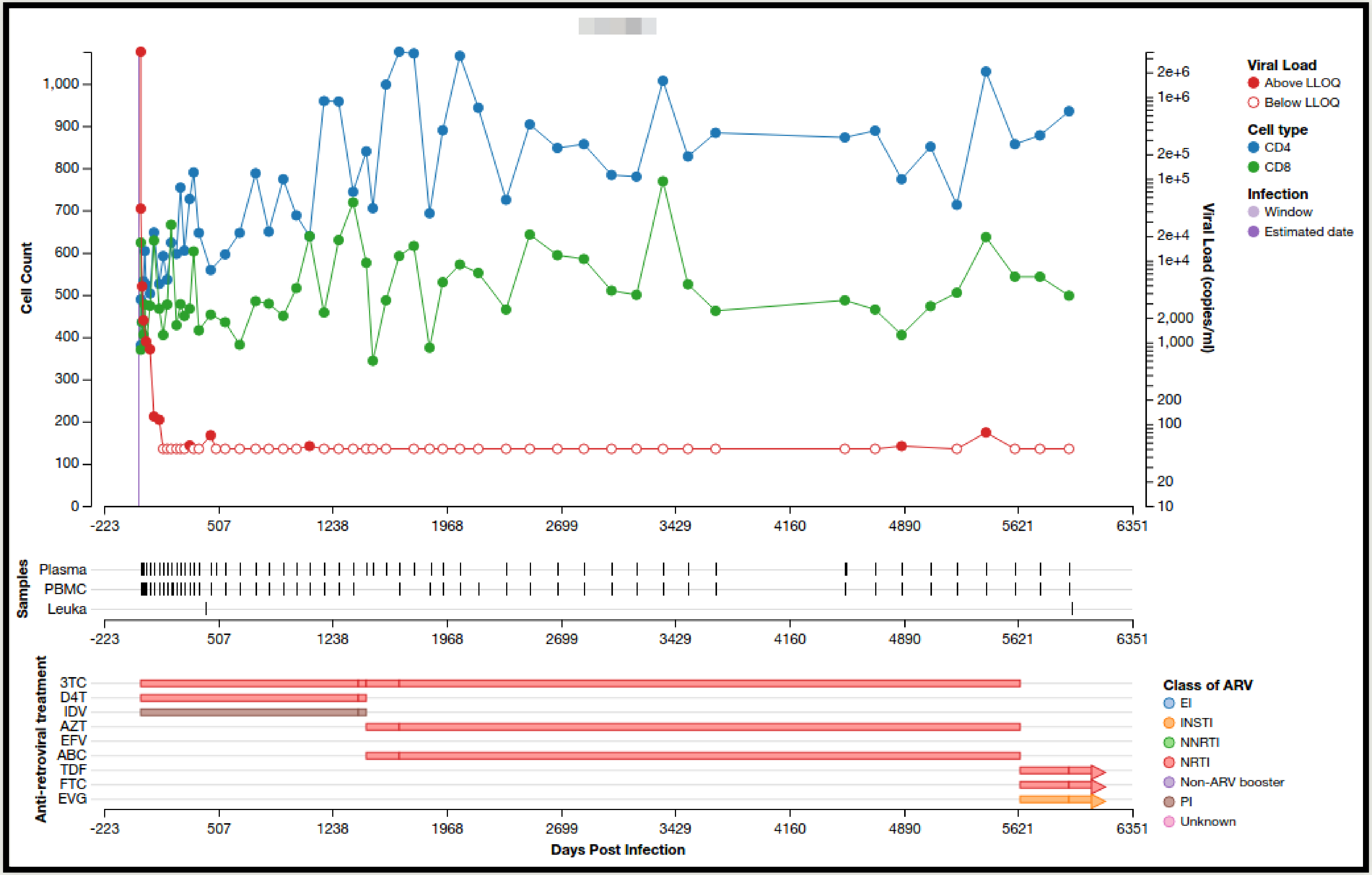


Simple plots are a dime-a-dozen in science, but good visualizations integrate and synthesize lots of data in order to highlight relationships within and between groups.

This viz compares the performance of the current experiment, highlighted, to related experiments, and is used for quality control.



Bad visualizations are hard to digest and leave you wondering who thought this was a good idea. They're like a pizza with too many toppings.



Good visualizations present the same data with clarity.

“Build just up to the edges”

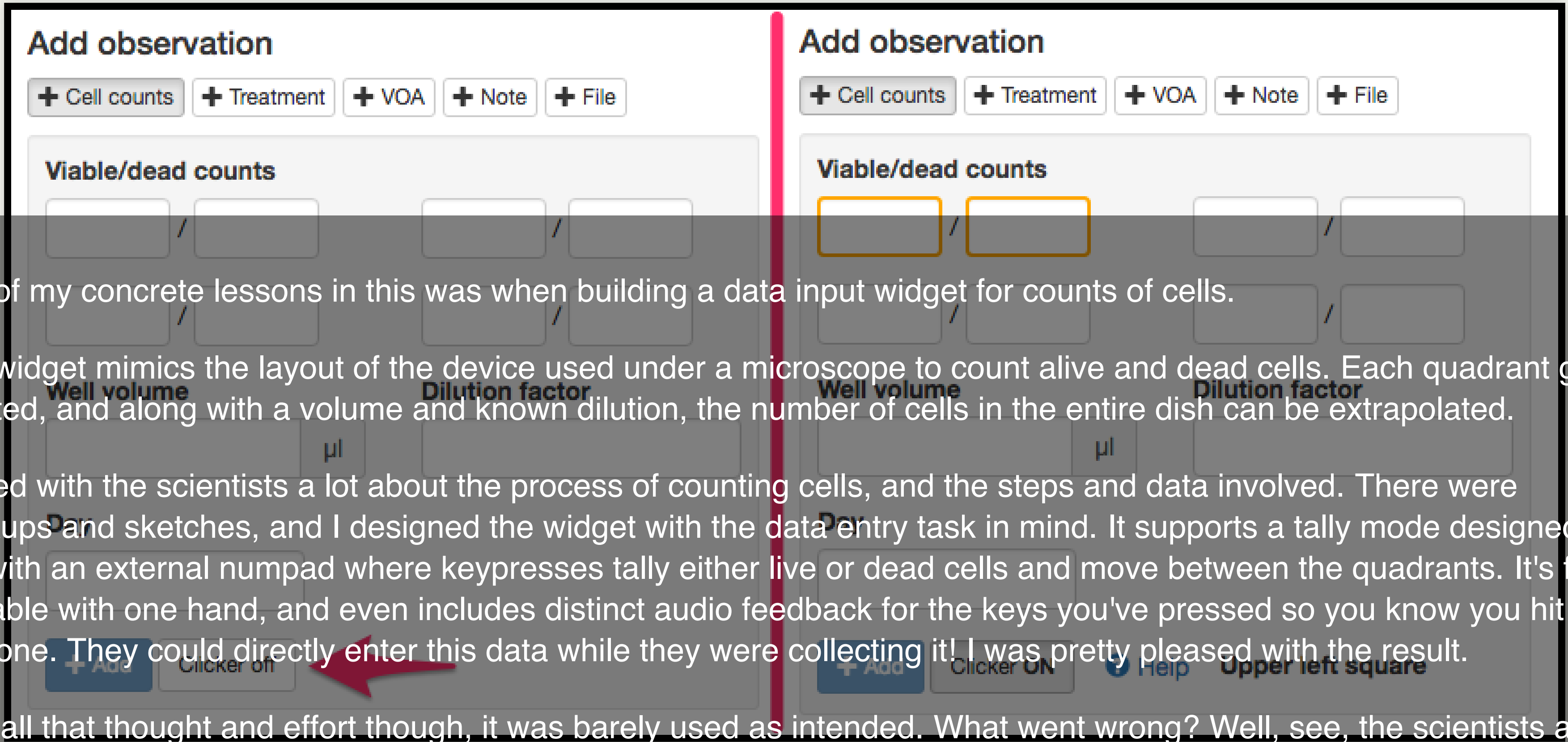
Designing and building research software is different than building a product to your vision. Research software must fit into the workflows that already exist, by and large, outside of the computer and not dictate them.

You're modeling real-world steps that happen in experiments, things that experiments produce, and actions that people perform. You don't typically get to decide what someone does at the bench, and so the challenges are different from designing many software products or applications.

The software is in service to the science and scientific goals. The job is to save labor, not create it by imposing new demands. My colleague Evan talks about this goal as "building right up to the edges of what researchers are already doing."

Sometimes I like to think of it as...

"I won't touch what already works for you"



One of my concrete lessons in this was when building a data input widget for counts of cells.

This widget mimics the layout of the device used under a microscope to count alive and dead cells. Each quadrant gets counted, and along with a volume and known dilution, the number of cells in the entire dish can be extrapolated.

I talked with the scientists a lot about the process of counting cells, and the steps and data involved. There were mockups and sketches, and I designed the widget with the data entry task in mind. It supports a tally mode designed for use with an external numpad where keypresses tally either live or dead cells and move between the quadrants. It's fully operable with one hand, and even includes distinct audio feedback for the keys you've pressed so you know you hit the right one. They could directly enter this data while they were collecting it! I was pretty pleased with the result.

After all that thought and effort though, it was barely used as intended. What went wrong? Well, see, the scientists are recording these counts while in a biosafety level 3 lab space. They're in gowns and hats, goggles, and are wearing two gloves on each hand. It's hot in the small, fully enclosed room. Counting cells is mundane and tedious and a lot of microscope time, so they streamline the process with two people: one preparing the cells to count and the other counting. Futzing with a laptop and software to do direct entry, even with the UX affordances was simply a non-starter. The goal is to get in and get out as soon as possible. It's much much faster to use a physical clicker, like the kind at movie theatres, and just scribble down numbers on a piece of paper as you go. The paper is easily transcribed to a spreadsheet when they're done, in the comfort of their offices.

Your spreadsheet of cell counts must be a CSV with all of the following columns. The order of columns doesn't matter, but the names must match exactly.

Column name	Description
`tc_plate`	Tissue culture plate, e.g. `TC001016`
`tc_well`	Tissue culture well name, e.g. `G12` or `B4`
`tc_day`	Culture day of the counted well, e.g. `10`
`volume`	Volume of the culture well in µl, used to extrapolate total count
`dilution_factor`	Factor by which the sample in the hemocytometer was diluted, used to extrapolate total count
`viable1` through `viable4`	Counts of viable cells in quadrants 1–4 (left to right, top to bottom)
`dead1` through `dead4`	Counts of dead cells in quadrants 1–4 (left to right, top to bottom)
`counted_by`	*(Optional)* The username of the person who counted this well, e.g. `trsibley`. If this column is omitted or is missing a value, the person who uploads the counts is considered the person who did the counting.

You can download a sample Excel file for entering data. Once you're done, click *File → Save As* and choose *Comma Separated Values*.

So what'd we do? Well, we removed nearly all of the specialized functionality for that widget, leaving just the bare bones input form for cases when a single count needed to be added, and added a way to bulk upload counts from a spreadsheet. This met the scientists on their own terms, right up to the edge of what they were already doing.

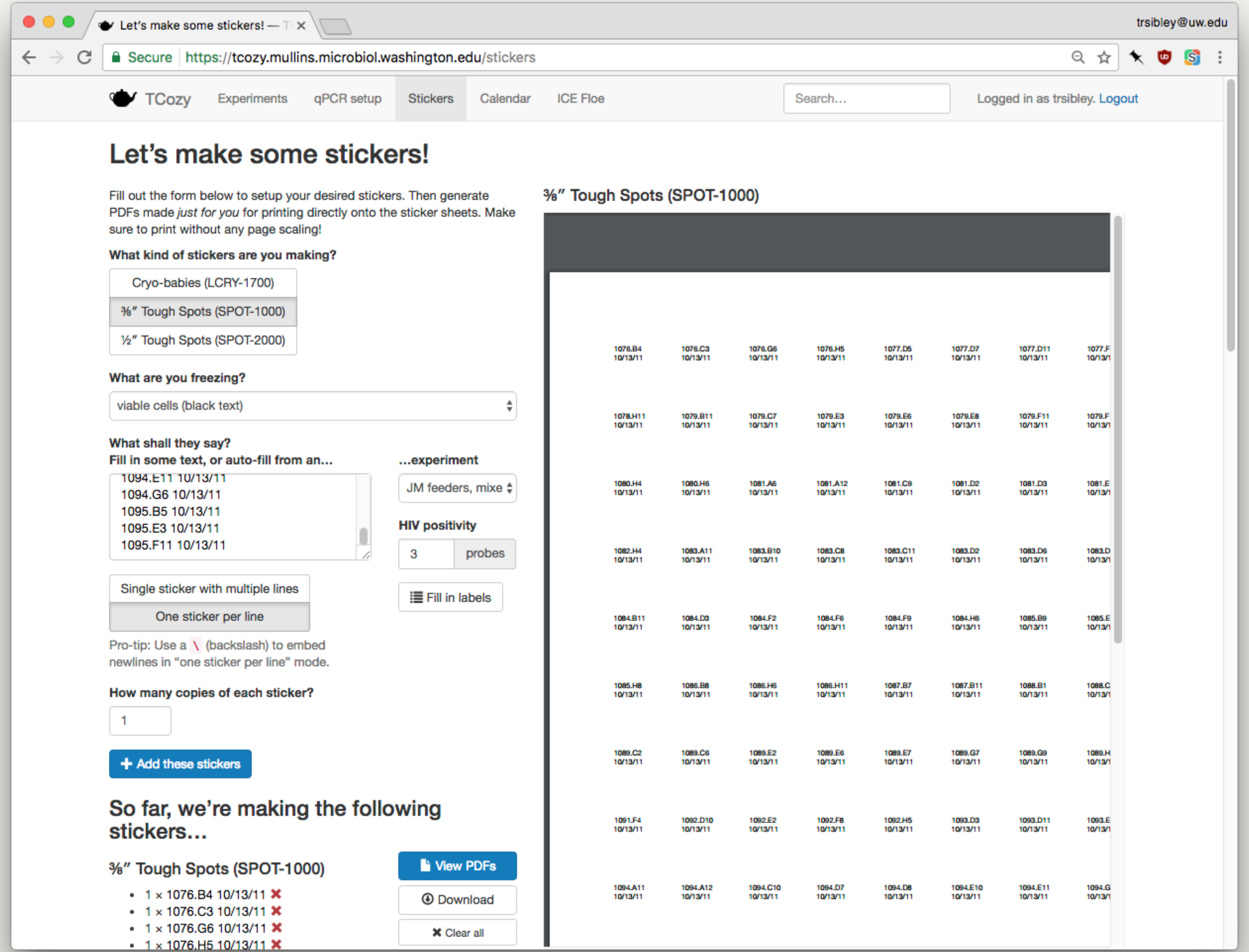
Evan Silberman

Another good example of this when we realized our scientists were hand labeling dozens of tiny stickers for every experiment they did. Each sticker's label had to be different, making even using a Word template laborious.

In this photo, each white dot is a sticker on top of a small vial. That's not a giant marker, the stickers are just *tiny*!

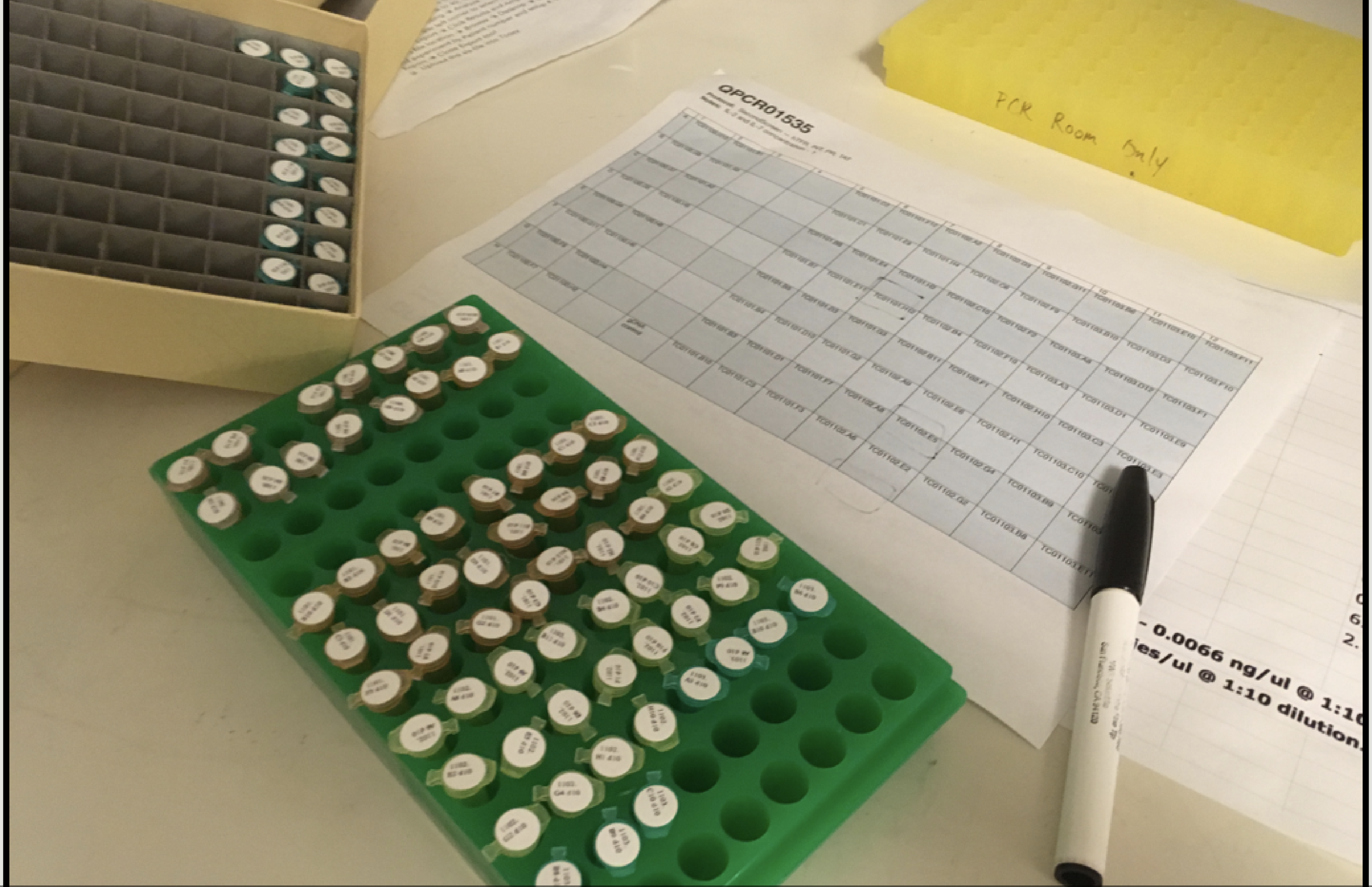
These aren't the normal printable stickers you'd buy at OfficeDepot either. They're cryo-safe stickers which will stay stuck to vials for years at minus 200°C (-328°F) while they chill out in the liquid nitrogen freezer.

While our scientists just buckled down to get the job done every time, it was obvious that we couldn't let that stand.


Let's make some stickers! — T
trsibley@uw.edu
Secure https://tcozy.mullins.microbiol.washington.edu/stickers
TCozy
Experiments
qPCR setup
Stickers
Calendar
ICE Floe
Search...
Logged in as trsibley. Logout
Let's make some stickers!
Fill out the form below to setup your desired stickers. Then generate PDFs made just for you for printing directly onto the sticker sheets. Make sure to print without any page scaling!
What kind of stickers are you making?
Cryo-babies (LCRY-1700)
⅜" Tough Spots (SPOT-1000)
½" Tough Spots (SPOT-2000)
What are you freezing?
viable cells (black text)
What shall they say?
Fill in some text, or auto-fill from an...
1094.E11 10/13/11
1094.G6 10/13/11
1095.B5 10/13/11
1095.E3 10/13/11
1095.F11 10/13/11
...experiment
JM feeders, mixe
HIV positivity
3
probes
Fill in labels
Single sticker with multiple lines
One sticker per line
Pro-tip: Use a \ (backslash) to embed newlines in "one sticker per line" mode.
How many copies of each sticker?
1
+ Add these stickers
So far, we're making the following stickers...
⅜" Tough Spots (SPOT-1000)
1 × 1076.B4 10/13/11
1 × 1076.C3 10/13/11
1 × 1076.G6 10/13/11
1 × 1076.H5 10/13/11
View PDFs
Download
Clear all
⅜" Tough Spots (SPOT-1000)

After only a couple days of hacking, testing, and discussing stickers, our scientists can now come to the lab's tissue culture app, TCozy, and generate labels based on the relevant data for the experiment they're wrapping up and freezing down.

Once they no longer had to worry about actually labeling the stickers themselves, they started suggesting improvements like color coding the label text and creating a second sticker for each vial which contained additional information.



There's also this piece of paper in the background of the picture. That's a plate setup guide which our app produces to help the scientists figure out what samples go where on 96-well plate. It matches the instrument setup files that our app also generates. Taking a couple hours of our time to produce those automatically is saving the scientists many hours of time in the long term and meets their workflows where they already are.

Learn to be comfortable adapting your models

Research science moves rapidly, much more rapidly than most software development can keep pace with, and especially so when the scientists are likely to outnumber the developers. Scientists will perform an experiment, learn something from it, and rinse and repeat ad infinitum, tweaking not just variables but also abandoning and adopting entire methods. You can't model every experiment. You can't capture all the data. You can't predict how the data being produced will fundamentally change over time as the way it's generated changes.

Software for bench scientists is so close to the physical world that no simplified, abstracted model of that world survives for very long before needing to be revised.

Because of this, I've learned to be comfortable making changes to our schemas, to be comfortable changing our object models, and to always consider if the problem will be easier by first adapting the model to fit the new reality. All software makes simplifications about the real world in order to make it tractable and understandable. Learn how to start simple and grow more nuanced from there as the needs arise.

We've found some good tools and techniques to help cope with rapid schema changes.

App::Sqitch

David Wheeler (THEORY)

sqitch.org

We use David Wheeler's sqitch to manage schema changes for our projects.

It organizes your schema changes into sets of deploy, revert, and verify scripts, with dependencies between your changesets declared in a plan file. There's nice command-line tooling to manage changesets and apply them or roll them back. The best part is that things like view definitions can be reworked in-place, leading to awesomely useful diffs in git.

It's pretty good, and I recommend it.

```
commit c3c755140031e1e8b80ece9a3c9b9bed992d503c
Author: [redacted] <[redacted]@uw.edu>
Date:   Wed Feb 9 23:36:37 2011 +0000

    Moving in Freezer System Whoo Hooo

    Requires the following DDL:

    begin transaction;
    CREATE SCHEMA freezer

    -- DROP TABLE freezer.freezer;
    CREATE TABLE freezer.freezer
    (
      freezer_id serial NOT NULL,
      "name" character varying(255) NOT NULL,
      owning_scientist_id integer,
      creating_scientist_id integer,
```

It's certainly better than putting your database migration scripts in your commit messages.

JSON document store…

While sqitch is nice for bringing sanity to the process of making schema changes, sometimes you need the flexibility to capture data before you can make the schema changes. Other times you want to capture data that's inherently variable and annoyingly hard to model relationally.

A JSON document store is a great option for this, but we didn't want another database service and we *did* want the documents themselves to tied to relational objects...

JSON document store… in Postgres

...so we use Postgres! We've found it works really well, and results in a very nice combination of flexibility without throwing out all relational integrity or adopting a fully destructured object-key-value relational table. It's easy to start using and provides straightforward upgrade paths to proper relational tables (managed with sqitch) once you're far enough along to know what's worth refactoring into the relational model and what's not.

It's also fast! The native JSON types in Pg are indexable and work well. You can even add CHECK constraints to do basic JSON document validation. The documents are manipulatable and traversable in SQL when you need it, which makes ad-hoc queries easy.

```
CREATE VIEW tcozy.cell_counts AS
SELECT
    observation_id,
    exp.name                                              as experime
    exp.experiment_id                                     as experime
    tcp.name                                              as tc_plate
    tcw.name                                              as tc_well,
    tpt.day                                               as tc_day,
    ( document#>>'{total,volume}' )::numeric              as volume,
    ( document#>>'{total,dilution_factor}' )::numeric as dilution
    ( document#>>'{viable,count}' )::numeric              as viable_c
    ( document#>>'{viable,density}' )::numeric            as viable_d
    ( document#>>'{dead,count}' )::numeric                as dead_cel
    ( document#>>'{dead,density}' )::numeric              as dead_den
    ( document#>>'{total,count}' )::numeric               as total_ce
    ( document#>>'{total,density}' )::numeric             as total_de
    CASE WHEN (document#>>'{total,count}')::numeric != 0 THEN
        round((document#>>'{viable,count}')::numeric
            / (document#>>'{total,count}')::numeric
            * 100, 2)
    END                                                   as viabilit
    username                                              as performe
FROM observation
JOIN timepoint tpt                   USING (timepoint_id)
JOIN experiment exp                     ON (tpt.experiment_id = exp
```

This also lets us create views designed for analysis that present the documents as flat tables joined into relevant rows, meaning our data scientists don't have to know about the document structure or JSON operators in SQL.

```
package TCozy::Document::CellCounts {
    extends 'TCozy::Document';

    has [qw[ viable dead total ]] => (
        is       => 'ro',
        isa      => Doc["CellCounts::Count"],
        coerce   => 1,
        required => 1,
    );

    has '+total' => (
        is => 'lazy',
    );

    sub _build_total {
        my $self = shift;
        return TCozy::Document::CellCounts::Count->new(
            ($self->viable->has_counts
                ? (counts => [
                    pairmap { $a + $b }
                        zip $self->viable->counts->@*,
                            $self->dead->counts->@* ])
                : (count => $self->viable->count + $self->dead->c
            volume => $self->viable->volume,
            dilution_factor => $self->viable->dilution_factor,
```

Our DBIx::Class models have helper methods for searching and filtering on the JSON documents, and the document columns themselves are automatically inflated to typed document classes so that we can constrain them on the application side and work with them as first-class objects.

This approach also yields nice options for future schema upgrades when you realize that perhaps some of the JSON data should be properly modeled. Since the data is already in the database, it's straightforward to write a migration script (in sqitch of course) moving data out of JSON documents and into proper relational tables.

We use JSON documents in Pg in two primary ways:

```
tcozy=> \d tcozy.experiment
                                  Table "tcozy.experiment"
    Column     |  Type   |                         Modifiers
---------------+---------+-------------------------------------------------
 experiment_id | integer | not null default nextval('experiment_e
 name          | text    |
 description   | text    |
 metadata      | jsonb   |
```

1. As a column directly on a primary record, usually to hold variable metadata

```
tcozy=> \d tcozy.observation
                                        Table "tcozy.observation"
        Column          |  Type   |
------------------------+---------+-----------------------------------
 observation_id         | integer | not null default nextval('ob
 document               | jsonb   | not null
 performed_by_user_id   | integer | not null
 timepoint_id           | integer |
 tissue_culture_plate_id| integer |
 tissue_culture_well_id | integer |
 qpcr_plate_id          | integer |
Indexes:
    "observation_pkey" PRIMARY KEY, btree (observation_id)
    "observation_document_idx" gin (document)
Check constraints:
    "observation_document_has_type" CHECK (document ? 'type'::tex
    "observation_has_one_object" CHECK (COALESCE(tissue_culture_p
```

1. As an better object-key-value pattern, an object-document pattern if you will. The document table refers to objects by real foreign keys, and can contain other metadata and relationships as necessary.

One tantalizing improvement to this approach that we may try in the future is using our application document models to produce JSON Schemas and JSON Schemas to produce database CHECK constraints so that we can validate docs regardless of how they enter the database.

Removing tedium makes space for new ideas and improvements

And finally, I think the most exciting and important lesson I've learned during my time in the lab is that when you remove the tedium from people's work, you help make space for them to think up new ideas and improvements to their workflows.

What's deemed reasonable, or even desireable, to do changes once tedious tasks vanish and people can think more creatively about the bigger picture.

Many of us have likely encountered fundamental misunderstandings about what's easy to make the computer do and what's hard. Every visible, successful example of automating away tedium is another example of how the computer can work for someone and a closing of that gap of understanding. They can better relate the possibilities to their own tasks. At first in lab I had to ask around for direct ways I could help people on their own work, but over time, people started approaching us to ask about making some process they were doing faster or easier or less error-prone.

Act III

Is this for you?

So is this for you?

The field of biology has a dire need for people who can think computationally and write good software at all levels. The field is currently grappling with how to build out these skills the last mile.

Currently demand vastly outstrips supply for computational skills in biology. There's a general consensus that the field needs to, as a whole, incorporate more bioinformatics and software development training into undergraduate and graduate biology curriculums. While this certainly must happen to a degree, it's a little like saying, "Well, to be a successful mechanical engineer, you now also need a law degree." The disciplines and practices of bioinformatics and software development are vast! While familiarity and literacy in both is a good goal, it's unreasonable to expect biology students to start mastering multiple fields.

There's another strategy that I think should be part of the solution, that's starting to come around in the field: create staff positions for and recruit professional software developers into research science. There are interesting and meaningful problems to solve in biology and a different culture than the tech industry, both of which can be attractive selling points.

- Learning new domains is fun
- Lots of room to operate in
- Fast pace

The rewards of working in a lab are many:

• You'll learn a new domain; scientists are happy to teach and explain.

• You'll be doing scientific research, where the problems are different than you're used to. There's broad space for your own thoughts, decision making, implementation, and feedback.

• The pace is often fast and exciting. You can never keep up with bench scientists, whether they're working on a new assay or churning through a rote set of experiments, thinking on your feet is necessary. You'll build "minimum viable products" to start capturing data now and then refine it to allow ongoing analysis.

xkcd

As programmers who know Perl, you're well-poised to think in terms of both high-level applications and raw data manipulation. You'll be able to deploy regular expressions to save the day.

Compassionate computing

You don't need a PhD to do this work, but you do need to have empathy and a determination to help others. I like to think of it as "compassionate computing", or software for humans.

Thanks!

Evan Silberman

Jim Mullins

@trs

tsibley.net/talks/last-mile-software-development/

That's all! I'd like to thank Evan Silberman, for his thoughts and conversations about these topics while we work, and Jim Mullins, for his support and allowing me wide discretion in the lab.

And thank *you* for your attention! I'll take questions now.