Jonny L. Saunders@
University of Oregon
Institute of Neuroscience, Department of Psychology
Eugene, OR 97403, United States

Swarmpunk

Rough Consensus and Running Code in Brains, Machines, and Society

July 16, 2022

This dissertation would not have been possible without the love and support of many people.

I would like to thank my parents for tolerating me for so many years, I've finally found a way to take all the angst and DMCA shutoffs and bricked computers and go pro.

Thank you as well to my undergraduate mentor Emma Coddington who was willing to take me into her lab on a last-hour discovery of purpose and made it possible for me to be a part of science at all. Thanks to a few others from those years: Bill Duvall for teaching me the power of knowing the contingency of history, Steve Heller for being a model of navigating empowerment and profit in science, and a few friends who made it through those messy years with me and I hope to see again soon, Emma Jonas, Emily Weatherford, and Lance Rossi.

I remember arriving late, underdressed, underslept, and with a headful of nonsense ideas to my initial interview to work with Mike Wehr, I am forever grateful for him for taking a risk on me despite my having absolutely no business being here. Not much has changed, and it has been a lovely, tumultuous ride as Mike supported me through a kaleideskope of projects that only occasionally made sense, giving me the space to chase the tail of whatever idea I was obsessing over and teaching me a bit about when to give them up. This lab has been a lovely place to figure out how science works. I am especially grateful to Iryna Yavorska for being my close friend and mentor in my first years here, and to Lucas Ott who has been a constant friend, colleague, and commiserateur. Thank you to my committee members Melissa Baese-Berk, Santiago Jaramillo, and Matt Smear who were willing to go along with this bizarre mash of disciplines and ideas. I am so honored to have been able to spend time with so many wonderful people while doing this work: my labmates Aldis Weible, Kip Keller, Nick Sattler, Sam Mehan, Molly Shallow, Tillie Morris, Brynna Paros, Netanya Beard, Rocky Penick, Matt Nardoci, and Alexa Wright; my colleagues and cohortmates particularly Judit Pungor, Anna Lakunina, Sarah Stednitz, Phil Parker, Jo Tomorsky, Joe Wekselblatt, Denver Ncube, Eartha Mae Guthman, Austin Seroka, Rachel Lukowicz, Jackie O'Brien, Dani Cosme, John Flournoy, Cory Costello, Rita Ludwig; and a few of my union cousins and co-conspirators Sam Shepherd, Ellen Kress, Rhi Lindgren, Emily Sutton, and many more that have impacted my life, subtly or not.

I would not have made it through grad school without the Janet Smith House and all of the people that I have had the blessing of living with here. There is nothing that has given me hope for the future more than what I have learned in cooperation with you all. It doesn't feel fair to name some and not all of you, but I am sitting at home looking at the wall of your photos and being flooded with memories right now.

For the Speech project: thanks to Aldis Weible, Lucas Ott, Conor O'Sullivan, Erik Flister, Kaori Idemaru, Brynna Paros, and all the wonderful people who let me record, chop up, and pitch shift their voices.

For the Autopilot project: thanks to Lucas Ott and Tillie Morris for doing most of the behavioral training and being so patient with the bugs, Brynna Paros and Nick Sattler for their help with constructing our behavioral boxes, Chris Rogers who has

been brave enough to adopt and contribute to Autopilot in its roughest state, Arne Meyer, Mikkel Roald-Arbøl, and David Robbe who have contributed code and advice, Mackenzie Mathis, Alex Mathis, Gonçalo Lopes, and Gary Kane who collaborated on the DeepLabCut-Live project and provided many a mentorship along the way, Jeremy Delahanty for his inspiring tenacity and humbleness in thinking about better research tools, Matt Smear and Reese Findley for loaning us their Bpod for far longer than they intended to, John Boosinger and the rest of the staff in the machine shop for all their advice and letting me use all their tools, Erik Flister whose Ratrix software inspired some of the design features of Autopilot [1], Santiago Jaramillo whose TASKontrol[2] gave inspiration for GUI design and served as an early scaffolding to learn Python, my labmates Molly Shallow and Sam Mehan who kept me afloat in my last months of dissertation writing, Rocky Penick for her help strapping Autopilot onto the back of Evan Vickers' mesoscope rig (and Evan for letting us play with his rig)

For the Infrastructure project: Thank you to Gabriele Hayden, Joel Chan, Tomasz Pluskiewicz, James Meickle, Christine Lemmer-Webber, Arnold Schrijver, Aad Versteden, Leslie Harka, the NWB & DANDI teams, Os Keyes, Avery Everhart, Eartha Mae Guthman, Olivia Guest, Andrew Hoffman, Lauren E. Wool, Kris Chauvin, Phil Parker, Ceci Herbert, Chris Rogers, Petar Todorov, Jeremy Delahanty, Andrey Andreev, Ralph Emilio Peterson, Manuel Schottdorf, Dan Goodman, Jakob Voigts for participating in the glue wiki, Gonçalo Lopes, Mackenzie Mathis, Mark Laubach & Open Behavior Team, Irene Knapp, Nire Bryce, Danny Mclanahan, Björn Brembs, Sanjay Srivastava & Metascience Class, Joon An, The Emerging ONICE team.

Thank you to my dear Rumbly Tumbly Lawnmower for being the light of my life.

Contents

Introduction

This is a piece about the power of swarms.

I arrived in neuroscience after bottoming out in politics and economics as I learned that studying the emergent dynamics of belief and power was not, in fact, what those academic disciplines studied. Somehow, my intellectual history professor Bill Duvall was able to identify that my rudderless interests were pointing towards neuroscience, and somehow he was right.

I was blessed with spending my first years outside of the canonical training of neuroscience with Emma Coddington, who was more interested in the multiscale interplay between neural, endocrine, and emotional systems than the reductive filterbank model of the brain. In neuroscience I found an endless abyss of systems creating themselves, loose boundaries between layers of organization and chaos. I still find myself in the fringes, thinking about the brain in its nonlinear dynamics, immune to averaging and estimation, an organ with its own ideas about its activity without a clear "code" or purpose aside from its own persistence. It is all the noise and interdependence and local organization that results in some messy superstructure that keeps me near it.

It has been impossible for me to ignore the systems that structure the practice of science long enough for me to spend much time doing it, though. The same things that draw me towards studying the brain make me look upward at the messy, anarchic processes that emerge as science — and the higher-order structuring forces that condition it. We are all little neurons, only aware of our immediate n-depth neighbors against the backdrop of the structures that our local awareness creates. The reality of swarms is the slippery interdependence that merges that local autonomy with the overriding circumstance that binds them together. The miracle of the brain is the miracle of society, how the blend of autonomy and independence makes something more spectacular than its parts. The challenge for understanding both is valuing the messiness and unplannedness of their agents alongside their necessary interdependence, without which they would lose meaning.

"Rough consensus and running code," cribbed from internet protocol architects[3], captures the tautology that what works is whatever works. Neither a grand planned architecture nor a libertarian focus on the disconnected autonomy of its agents describes systems capable of emergent behavior. What works is a fluid and evolving consensus based on the agents organizing together to meet their needs. This is the thread that binds my work: ill-defined categories computed by neural assemblies, loose design in decoupled systems in experimental tools, and the linked interoperability of digital infrastructure. It's not clear to me whether this is an anarchist's view of the brain, or a neuroscientist's view of politics, and it's not necessarily important to me to resolve that.

These first two pieces focus on the ability for auditory cortex to learn from continuous sounds to create the ill-defined perceptual categories of phonemes, the initial plan for my work that was quickly interrupted.

Part I

Speech

1

Mice Can Learn Phonetic Categories

Originally published as doi:10.1121/1.5091776 [4]

We perceive speech as a series of relatively invariant phonemes despite extreme variability in the acoustic signal. To be perceived as nearly-identical phonemes, speech sounds that vary continuously over a range of acoustic parameters must be perceptually discretized by the auditory system. Such many-to-one mappings of undifferentiated sensory information to a finite number of discrete categories are ubiquitous in perception. Although many mechanistic models of phonetic perception have been proposed, they remain largely unconstrained by neurobiological data. Current human neurophysiological methods lack the necessary spatiotemporal resolution to provide it: speech is too fast and the neural circuitry involved is too small. Here we demonstrate that mice are capable of learning generalizable phonetic categories, and can thus serve as a model for phonetic perception. Mice learned to discriminate consonants, and generalized consonant identity across novel vowel contexts and speakers, consistent with true category learning. A mouse model, given the powerful genetic and electrophysiological tools for probing neural circuits available for them, has the potential to powerfully augment our mechanistic understanding of phonetic perception.

1.1 Introduction

1.1.1 Lack of acoustic invariance in phonemes

We perceive speech as a series of relatively invariant phonemes despite extreme variability in the acoustic signal. This lack of order within phonemic categories remains one of the fundamental problems of speech perception [5]. Plosive stop consonants (such as /b/ or /g/) are the paradigmatic example of phonemes with near-categorical perception [6, 7, 8] without invariant acoustic structure [9, 10]. The problem is not just that phonemes are acoustically variable, but rather that there is a fundamental lack of invariance in the relation between phonemes and the acoustic signal [10]. Despite our inability to find a source of invariance in the speech signal, the auditory system learns some acoustic-perceptual mapping such that a plosive stop like /b/ is perceived as nearly identical across phonetic contexts. A key source of variability is coarticulation, which causes the sound of a spoken consonant to be strongly affected by neighboring segments, such as vowels. Coarticulation occurs during stop production because the articulators (such as the tongue or lips) have not completely left the positions from the preceding phoneme, and are already moving to anticipate the following phoneme [11, 12]. Along with many other sources of acoustic variation like speaker identity, sex, accent, or environmental noise; coarticulation guarantees that a given stop consonant does not have a uniquely invariant acoustic structure across phonetic contexts. In other words, there is no canonical acoustic /b/ [11, 6]. Phonetic perception therefore cannot be a simple, linear mapping of some continuous feature space to a discrete phoneme space. Instead it requires a mapping that flexibly uses evidence from multiple, imperfect cues depending on

context [6, 13]. This invariant perception of phonemes, despite extreme variability in the physical speech signal, is referred to as the non-invariance problem [14].

1.1.2 Generality of phonetic perception

The lack of a simple mapping between acoustic attributes and phoneme identity has had a deep influence on phonetics, in part motivating the hypothesis that speech is mechanistically unique to humans [15], and the development of non-acoustic theories of speech perception (most notably motor theories [13, 11, 16]). However, it has been clear for more than 30 years that at least some auditory components of speech perception are not unique to humans, suggesting that human speech perception exploits evolutionarily-preserved functions of the auditory system [10, 17, 18, 19]. For example, nonhuman animals like quail [10, 20], chinchillas [21], rats [22], macaques [23], and songbirds [24] are capable of learning phonetic categories that share some perceptual qualities with humans [25, 26]. This is consistent with the idea that categorizing phonemes is just one instance of a more general problem faced by all auditory systems, which typically extract useable information from complex acoustic environments by reducing them to a small number of 'auditory objects' (for review, see [27]).

1.1.3 Neurolinguistic theories of phonetic perception

Many neurolinguistic theories of phonetic perception have been proposed [28, 29, 30, 16, 31], but neurophysiological evidence to support them is limited. One broad class of models follows the paradigm of hierarchical processing first described by Hubel and Weisel in the visual system [28, 32, 29]. In these models, successive processing stages in the auditory system extract acoustic features with progressively increasing complexity by combining the simpler representations present in preceding stages. Such hierarchical processing is relatively well-supported by experimental data. For example, the responses of neurons in primary auditory cortex (A1) to speech sounds are more diverse than those in inferior colliculus [33] (but see [34]). While phoneme identity can be classified post-hoc from population-level activity in A1 [35, 36, 37], neurons in secondary auditory cortical regions explicitly encode higher-order properties of speech sounds [38, 39, 40, 41, 42].

Another class of models proposes that phonemes have no positive acoustic "prototype", and that we instead learn only the acoustic features useful for telling them apart [30]. Theoretically, these discriminative models provide better generalization and robustness to high variance [43]. Theories based on discrimination rather than prototype-matching have a long history in linguistics [44], but have rarely been implemented as neurolinguistic models. A possible neural implementation of discriminative perception is that informative contrast cues could evoke inhibition to suppress competing phonetic percepts, similar to predictive coding [45, 30, 46]. Neurophysiological evidence supports the existence of discriminative predictive coding, but its specific implementation is unclear [47, 48].

These two very different classes of models illustrate a major barrier faced by phonetic research: both classes can successfully predict human categorization performance, making it difficult to empirically validate or refute either of them using psychophysical experiments alone. Mechanistic differences have deep theoretical consequences — for example, the characterizations made by the above two classes of models re-

garding what phonemes *are* precisely oppose one another: are they positive acoustic prototypes, or sets of negative acoustic contrasts? Perceptually, do listeners identify phonemes, or discriminate between them? Neurobiological evidence regarding how the brain actually solves these categorization problems could help overcome this barrier.

1.1.4 The utility of a mouse model for speech research

Neurolinguistic research in humans faces several limitations that could be overcome using animal models.

First, most current human neurophysiological methods lack the spatiotemporal resolution to probe the fine spatial scale of neuronal circuitry and the millisecond timescale of speech sounds. A causal, mechanistic understanding of computation in neural circuits is also greatly aided by the ability to manipulate individual neurons or circuit components, which is difficult in humans. Optogenetic methods available in mice provide the ability to activate, inactivate, or record activity from specific types of neurons at the millisecond timescales of speech sounds.

Second, it is difficult to isolate the purely auditory component of speech perception in humans. Humans can use contextual information from syntax, semantics or task structure to infer phoneme identity [49, 50]. It is also difficult to rule out the contribution of multimodal information [51], or of motor simulation predicted by motor theories. Certainly, these and other non-auditory strategies are used during normal human speech perception. Nevertheless, speech perception is possible without these cues, so any neurocomputational theory of phonetic perception must be able to explain the purely auditory case. Animal models allow straightforward isolation of purely auditory phonetic categorization without interference from motor, semantic, syntactic, or other non-auditory cues.

Third, it is difficult to control for prior language experience in humans. Experience-dependent effects on phonetic perception are present from infancy [52]. It can therefore be challenging to separate experience-driven effects from innate neurocomputational constraints imposed by the auditory system. Completely language-naive subjects (such as animals) allow the precise control of language exposure, permitting phonetics and phonology to be disentangled in neurolinguistics.

Animal models of phonetic perception are a useful way to avoid these confounds, and provide an important alternative to human studies for empirically grounding the development of neurolinguistic theories. The mouse is particularly well-suited to serve as such a model. A growing toolbox of powerful electrophysiological and optogenetic methods in mice has allowed unprecedented precision in characterizing neural circuits and the computations they perform.

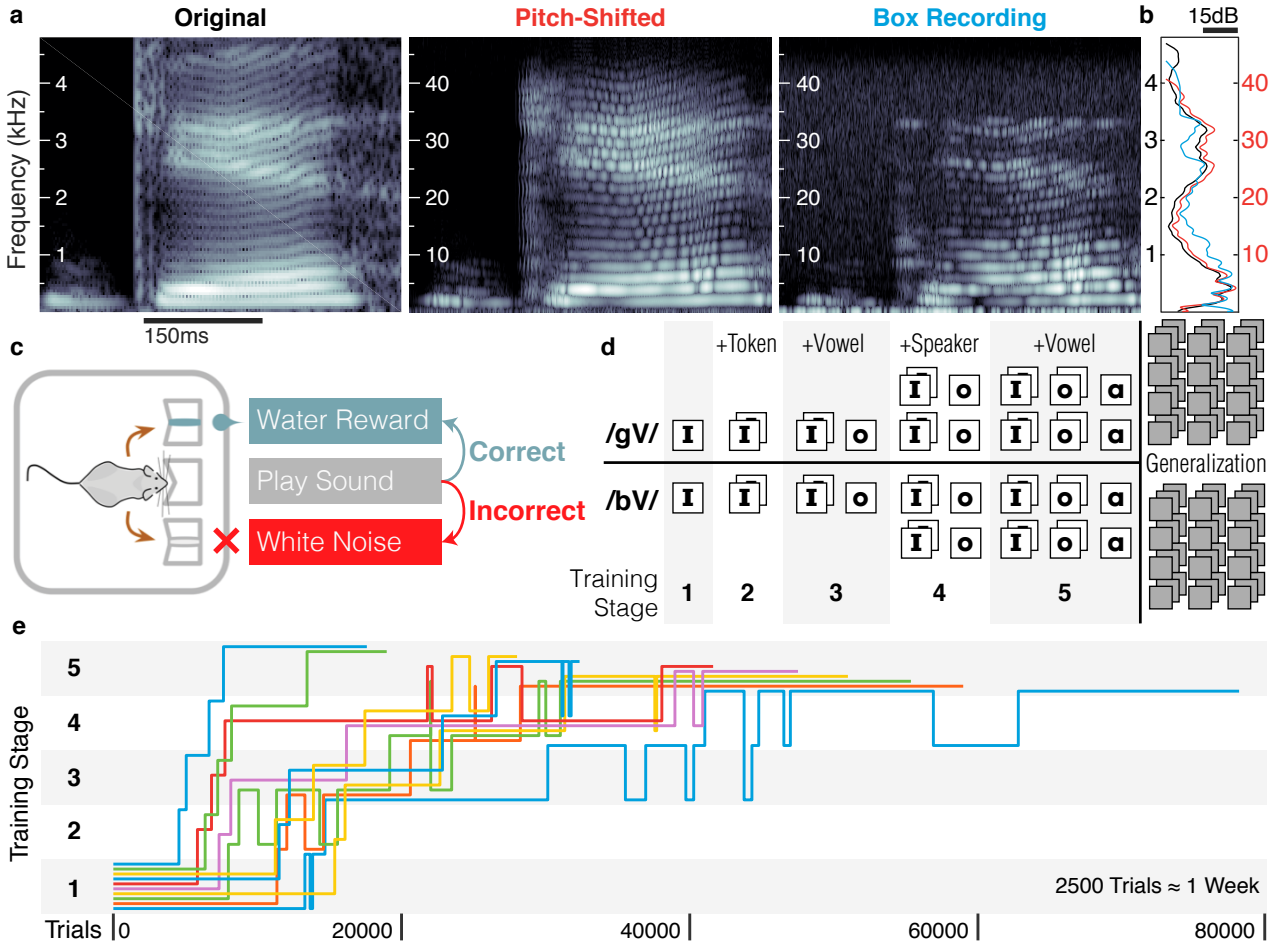
1.1.5 The utility of phonetics for auditory neuroscience

Conversely, auditory neuroscience stands to benefit from the framework provided by phonetics for studying how sound is transformed to meaning. Understanding how complex sounds are encoded and processed by the auditory system, ultimately leading to perception and behavior, remains a challenge for auditory neuroscience. For example, it has been difficult to extrapolate from simple frequency/amplitude receptive fields to understand the hierarchical organization of complex feature selec-

tivity across brain areas. A great strength of neuroethological model systems such as the songbird is that both the stimulus (e.g., the bird's own song) and the behavior (song perception and production) are well understood. This has led to significant advances in understanding the hierarchical organization and function of the song system [53, 54]. The long history of of speech research in humans has produced a deep understanding of the relationships between acoustic features and phonetic perception [55]. These insights have enabled specific predictions about what kinds of neuronal selectivity for features (and combinations of features) might underlie phonetic perception [5]. Although recognizing human speech sounds is not a natural ethological behavior for mice, phonetics nevertheless provides a valuable framework for studying how the brain encodes and transforms complex sounds into perception and behavior.

Here we trained mice to discriminate between pitch-shifted recordings of naturally produced consonant-vowel (CV) pairs beginning with either /g/ or /b/. Mice demonstrated the ability to generalize consonant identity across novel vowel contexts and speakers, consistent with true category learning. To our knowledge this is the first demonstration that any animal can generalize consonant identity across both novel vowel contexts and novel speakers. These results indicate that mice can solve the non-invariance problem, and suggest that mice are a suitable model for studying

Figure 1.1: Stimuli and Task Design. a) Spectrograms of stimuli. Left: Example of an original recording of an isolated consonant-vowel token (/gI/). Center: the same token pitch-shifted upwards by 10x (3.3 octaves) into the mouse hearing range. Right: Recording of the pitch-shifted token presented in the behavior box. Stimuli retained their overall acoustic structure below 34kHz (the upper limit of the speaker frequency response). **b)** Power spectra (dB, Welch's method) of tokens in **a**. Black: Original (left frequency axis), red: Pitch-shifted (right frequency axis), blue: Box Recording (right frequency axis). **c)** Mice initiated a trial by licking in a center port and responded by licking on one of two side ports. Correct responses were rewarded with water and incorrect responses were punished with a mildly-aversive white noise burst. **d)** The difficulty of the task was gradually expanded by adding more tokens (squares), vowels (labels), and speakers (rows) before the mice were tested on novel tokens in a generalization task. **e)** Mice (colored lines) varied widely in the duration of training required to reach the generalization phase. Mice were returned to previous levels if they remained at chance performance after reaching a new stage.



phonetic perception.

1.2 Results

1.2.1 Generalization performance

We began training 23 mice to discriminate between consonant-vowel (CV) pairs beginning with either /b/ or /g/ in a two-alternative forced choice task. CV tokens were pitched-shifted up into the mouse hearing range (Fig. 1.1a-b). Each mouse began training with a pair of tokens (individual recordings) in a single vowel context (ie. /bI/ and /gI/) from a single speaker, and then advanced through stages that progressively introduced new tokens, vowels, and speakers (Fig. 1.1c-d, see Methods). Training was discontinued in 13 (56.5%) of these mice because their performance on the first stage was not significantly better than chance after two months. The remaining 10 (43.5%) mice progressed through all the training stages to reach a final generalization task, on average in 14.9 $(\sigma \pm 7.8)$ weeks (Fig. 1.1e). This success rate and training duration suggest that the task is difficult but achievable.

We note that this training time is similar to that reported previously for rats $(14 \pm 0.3$weeks [22]). Previous studies have not generally reported success rates. Human infants also vary in the rate and accuracy of their acquisition of phonetic categories [**?**], so we did not expect perfect accuracy from every mouse. The cause of such differences in ability is itself an opportunity for future study.

Generalization is an essential feature of categorical perception. By testing whether mice can generalize their phonetic categorization to novel stimuli, we can distinguish whether mice actually learn phonetic categories or instead just memorize the reward contingency for each training token. Four types of novelty are possible with our stimuli: new tokens from the speakers and vowel contexts used in the training set, new vowels, new speakers, and new vowels from new speakers (colored groups in Fig. 1.2a). In the final generalization stage, we randomly interleaved tokens from each of these novelty classes on 20% of trials, with the remaining 80% consisting of tokens from the training set. We interleaved novel tokens with training tokens for two reasons: (1) to avoid a sudden increase in task difficulty, which can degrade performance, and (2) to minimize the possibility that mice could learn each new token by widely separating them in time (on average, generalization tokens were repeated only once every five days).

We looked for 4 hallmarks of generalization: (1) Mice should be able to accurately categorize novel tokens, (2) performance should reflect the quality of the acoustic-phonetic criteria learned in training, (3) performance on novel tokens should be correspondingly worse for tokens that differ more from those in the training set, and (4) accurate categorization of novel tokens should not require additional reinforcement.

All 10 mice were able to categorize tokens of all generalization types with an accuracy significantly greater than chance. We estimated the impact of each generalization class on performance as a fixed factor nested within each mouse as a random factor in a mixed-effects logistic regression (see Methods). The predicted accuracy for each generalization class is shown in Table 1.1, each providing an estimate of the difficulty of that class after accounting for the random effects of individual mice.

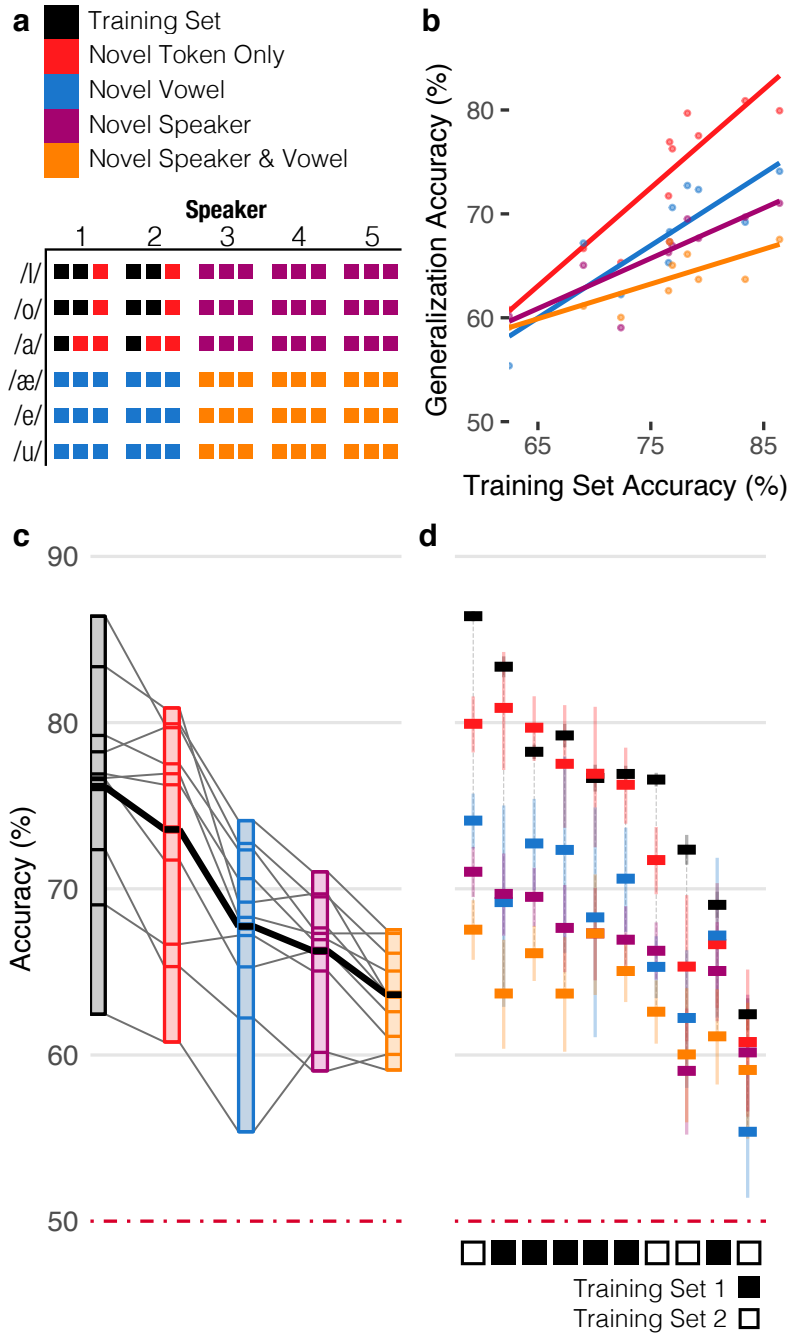


Figure 1.2: Generalization accuracy by novelty class. Mice generalized stop consonant discrimination to novel CV recordings. **a)** Four types of novelty are possible with our stimuli: novel tokens from the speakers and vowels used in the training set (red), novel vowels (blue), novel speakers (purple), and novel speakers with novel vowels (orange). Tokens in the training set are indicated in black. Colors same throughout. **b)** Mice that performed better on the training set were better at generalization. Each point shows the performance for a single mouse on a given novelty class, plotted against that mouse's performance on training tokens presented on during the generalization phase (both averaged across the entire generalization phase). Lines show linear regression for each novelty class. **c)** Mean accuracy for each novelty class (gray lines indicate individual mice). **d)** Mean accuracy for individual mice (colored bars indicate each novelty class). Error bars in **d** are 95% binomial confidence intervals. Mice were assigned one of two sets of training tokens, black and white boxes in **d**.

Performance on all generalization types was strongly and positively correlated with performance on the training set (Fig. 1.2b, adj. $R^2 = 0.74, F(4,5) = 7.4, p < 0.05$). If mice were "overfitting," that is, memorizing the training tokens rather than learning categories, then we would expect the opposite (i.e., above some threshold, mice that performed better on the training set would perform correspondingly worse on the generalization set). It appears instead that better prototypes or decision boundaries learned in the training stages allowed better generalization to novel tokens.

Mice were better at some types of generalization than others (Fig. 1.2c). The estimates of their relative difficulty (Fig. 1.2c) provide a ranking of the perceptual novelty of the stimulus classes based on their similarity to the training tokens. From easiest to hardest, these were: novel token, novel vowel, novel speaker (which was not significantly more difficult than novel vowel), novel speaker & vowel. The effects of generalizing to novel vowels and novel speakers were not significantly different from each other, but pairwise comparisons between each of the other types of generalization were (Tukey's method, all $p < 0.001$, also see confidence intervals in Table 1.1).

Although the effect of each generalization type on performance was significantly different between mice (Likelihood Ratio Test, $\chi^2(14) = 407.22, p \ll 0.001$), they were highly correlated (see Table 1.1). The relative consistency of novelty type difficulty across mice (ie. the correlation of fixed effects, Fig. 1.2c) is striking, but our results cannot distinguish whether it is due to the mice or the stimuli: it is unclear whether the acoustic/phonetic criteria learned by all mice are similarly general, or whether the "cost" of each type of generalization is similar across an array of possible acoustic/phonetic criteria.

True generalization requires that one set of discrimination criteria can be successfully applied to novel cases without reinforcement. It is possible that the mice were instead able to rapidly learn the reward contingency of novel tokens during the generalization stage. If mice were learning rapidly rather than generalizing, this would predict that novel token performance (1) would be indistinguishable from chance on the first presentation, and (2) would increase relative to performance on already-learned tokens with repeated presentations.

Performance on the first presentation of novel tokens was significantly greater than chance (Fig. 1.3, all mice, all tokens from all novelty classes: one-sided binomial test, $n = 1410, P_{correct} = 0.61$, lower 95% CI $= 0.588, p \ll 0.001$; all mice, worst novelty class: $n = 458, P_{correct} = 0.581$, lower 95% CI $= 0.541, p < 0.001$). This demonstrates that mice were able to generalize immediately without additional reinforcement. Although performance on novel tokens did increase with repetition, so did performance on training tokens (Fig. 1.3). We noted that performance on all tokens (both novel and previously learned tokens) transiently dropped after each transition between task stages, suggesting a non-specific effect of an increase in task difficulty. To distinguish an increase in performance due to learning from an increase due to acclimating to a change in the task, we compared performance on generalization and training tokens over the first 40 presentations of each token. If the mice were learning the generalization tokens, the increase in performance with repeated presentations should be significantly greater than that of the already trained tokens.

Performance was well fit by a logistic regression of correct/incorrect responses from each mouse against the novelty of a token (trained vs. novel tokens), and the number of times it had been presented (Fig. 1.3). The effect of the number of presentations on accuracy was not significantly different for novel tokens compared to trained tokens (interaction between novelty and the number of presentations: Wald test, $z = 1.239, 95\% CI = [-0.022, 0.1], p = 0.215$). This was also true when the model was fit with with the generalization types themselves rather than trained vs. novel tokens (most significant interaction, generalization to novel speakers x number of presentations: Wald test, $z = 1.425, 95\% CI = [-0.018, 0.117], p = 0.154$) and with different numbers of repetitions (10: $z = -0.219, 95\% CI =$

$[-0.161, 0.13], p = 0.827$; 20: $z = -0.521, 95\% CI = [-0.116, 0.068], p = 0.602$). This indicates that the asymptotic increase in performance on novel tokens was a general effect of adapting to a change in the task rather than a learning period for the novel stimuli.

In summary, the behavior of the mice is consistent with an ability to generalize some learned acoustic criteria to novel stimuli. It is unlikely that the mice rapidly learned the novel tokens because (1) performance on the first presentation of novel tokens was significantly above chance, (2) performance on subsequent presentations of novel tokens did not improve compared to trained tokens, and (3) learning each token would have to take place over unrealistically long timescales: there were an average of 2355 trials (5 days) between the first and second presentation of each novel token.

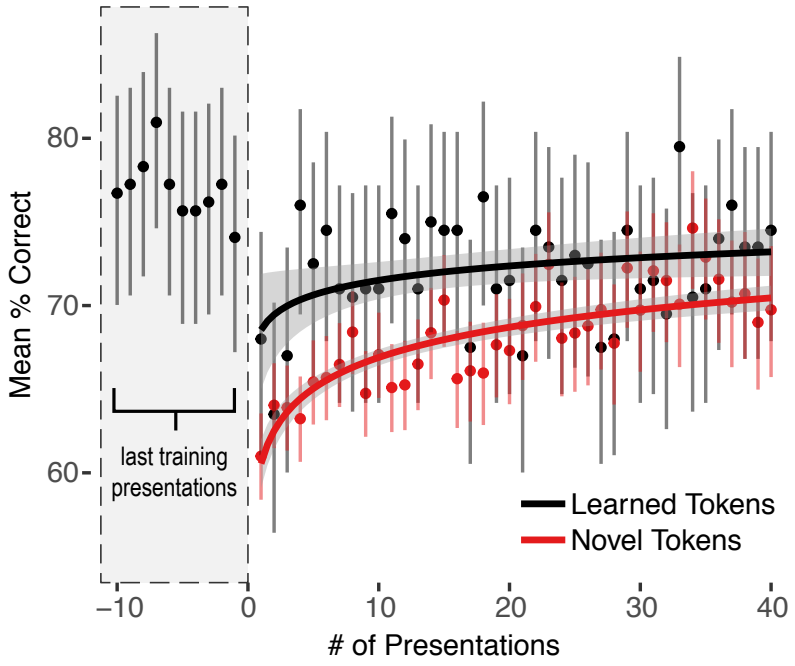


Figure 1.3: Learning curve for novel tokens. Performance for both novel and training set tokens dropped transiently and recovered similarly after the transition to the generalization stage. Presentation 0 corresponds to the transition to the generalization stage. The final ten trials before the transition are shown in the gray dashed box. Mean accuracy and 95% binomial confidence intervals are collapsed across mice for novel (red, all novelty classes combined) or learned (black) tokens, by number of presentations in the generalization task. Logistic regression of binomial correct/incorrect responses fit to log-transformed presentation number (lines, shading is smoothed standard error).

1.2.2 Training Set Differences

One strength of studying phonetic perception in animal models is the ability to precisely control exposure to speech sounds. To test whether and how the training history impacted the pattern of generalization, we divided mice into two cohorts trained with different sets of speech tokens. In the first cohort ($n = 6$ mice), mice were trained with tokens from speakers 1 and 2 (speaker number in Fig. 1.4a), whereas the second cohort ($n = 4$ mice) were trained on speakers 4 and 5.

The two training cohorts had significantly different patterns of which tokens were accurately categorized (Fig. 1.4a, Likelihood-Ratio test, regression of mean accuracy on tokens with and without token x cohort interaction: $\chi^2_{161}, p \ll 0.001$). Put another way, accuracy patterns were markedly similar within training cohorts: cohort differences accounted for fully 40.6% of all accuracy variance (sum of squared-error) between tokens.

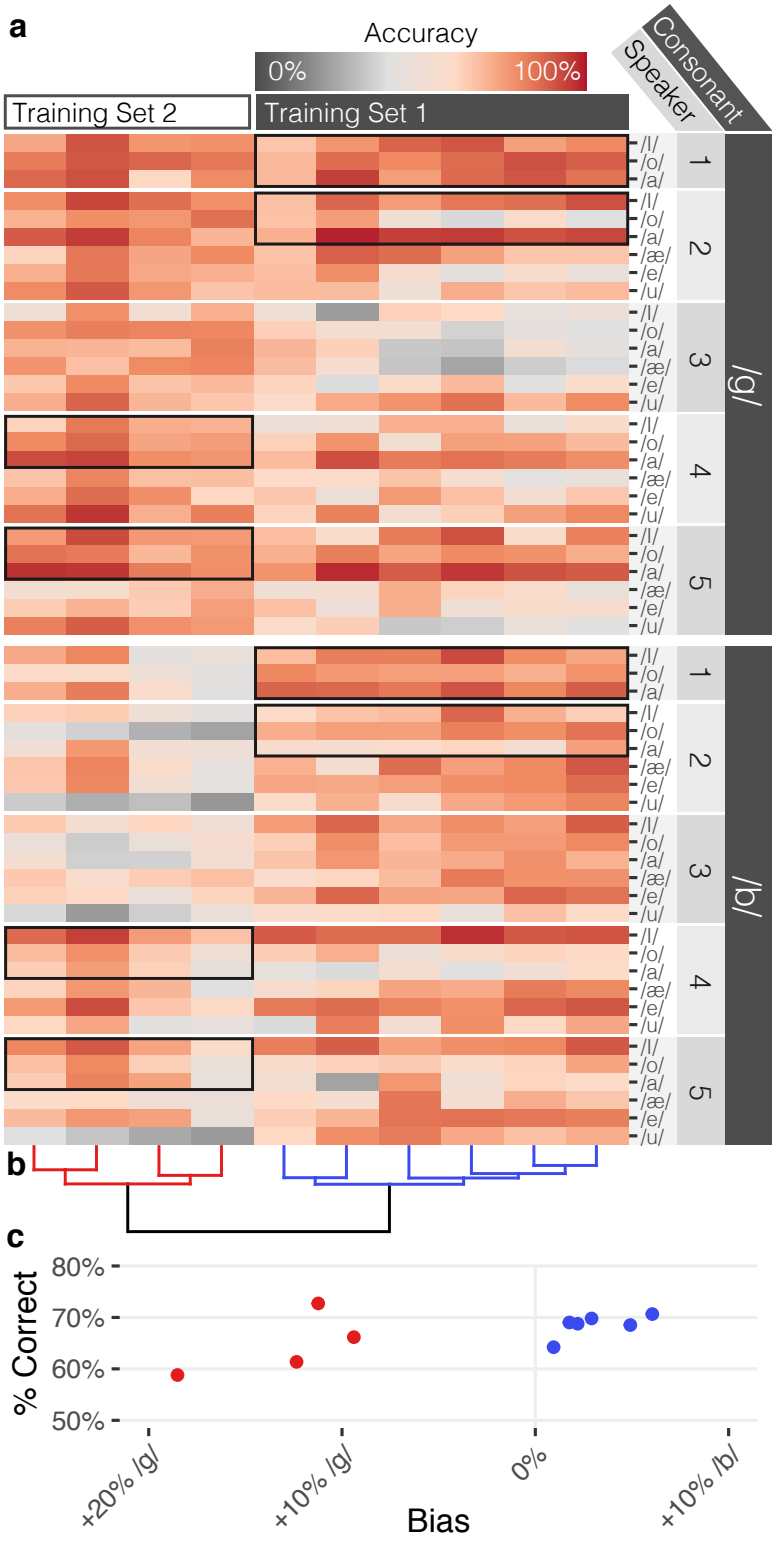


Figure 1.4: Patterns of individual and group variation. a) Mean accuracy (color, scale at top) for each mouse (columns) on tokens grouped by consonant, speaker, and vowel (rows). The different training sets (cells outlined with black boxes) led to different patterns of accuracy on the generalization set. **b)** Ward clustering dendrogram, colored by cluster. **c)** Training set cohorts differed in bias but not mean accuracy.

Mice from the second training cohort were far more likely to report novel tokens as a /g/ than the first cohort (Fig. 1.4b), an effect that was not significantly related to their overall accuracy ($b = 0.351, t(8) = 2.169, p = 0.062$). Since the only

difference between these mice were the tokens they were exposed to during training (they were trained contemporaneously in the same boxes), we interpret this response bias as the influence of the training tokens on whatever acoustic cues the mice had learned in order to perform the generalization task. This suggests that the acoustic properties of training set 2 caused the /g/ "prototype" to be overbroad.

We searched for additional sub-cohort structure with hierarchical clustering (Ward's Method, dendrogram in fig 1.4b). Within each training cohort there appeared to be two additional clusters of accuracy patterns. Though our sample size was too small to meaningfully interpret these clusters, they raise the possibility that even when trained using the same set of sounds mice might learn multiple sets of rules to distinguish between consonant classes.

1.2.3 Acoustic-behavioral correlates

Humans can flexibly use several acoustic features such as burst spectra and formant transitions to discriminate plosive consonants, and we wondered to what extent mice were sensitive to these same features.

One dominant acoustic cue for place of articulation in stop consonants is the transition of the second formant following the plosive burst [56, 5, 57]. Formant transitions are complex and dependent on vowel context, but tokens for a given place of articulation cluster around a line – or "locus equation" – relating F2 frequency at release to its mid-vowel steady-state [5, 57] (Fig. 1.5a). If mice were sensitive to this cue, the distance from both locus equation lines should influence response. For example, a /g/ token between the locus equation lines should have a greater rate of misclassification than a token at an equal distance above the red /g/ line. Therefore we tested how classification depended on the difference of distances from each line (/g/ distance - /b/ distance, which we refer to as "locus difference").

Mean responses to tokens (ranging from 100% /g/ - 100% /b/) were correlated with locus differences (black line, Fig. 1.5b). However, it is important to note that this correlation does not necessarily demonstrate that mice relied on this acoustic cue. Because multiple acoustic features are correlated with consonant identity, performance that is correlated with one such cue would also be correlated with all the others. The mice learned some acoustic property of the consonant classes, and since the acoustic features are all highly correlated with one another, they are all likely to correlate with mean responses.

To distinguish whether mice specifically relied on F2 locus distance, we therefore measured the marginal effect of this acoustic cue within a consonant class. This is shown by the slopes of the red and blue lines in Fig. 1.5b. For example, is a /g/ token that is further away from the blue /b/ line more likely to be identified as a /g/ than one very near the /b/ line? Mean responses to /g/ tokens were negatively correlated with locus distance (Mean response /g/ to /b/ between 0 and 1, $b = -0.028kHz, 95\%CI = [-0.035, -0.022], p \ll 0.001$). In other words, tokens that should have been more frequently confused with /b/ were actually more likely to be classified as /g/. Note the red points at locus distance of zero in Fig. 1.5b: these tokens have an equal distance from both the /b/ and /g/ locus equation prototypes but are some of the most accurately categorized /g/ tokens. /b/ tokens obeyed the predicted direction of locus distance ($b = 0.049, 95\%CI = [0.039, 0.06], p \ll 0.001$), but the effect was very small: moving one standard de-

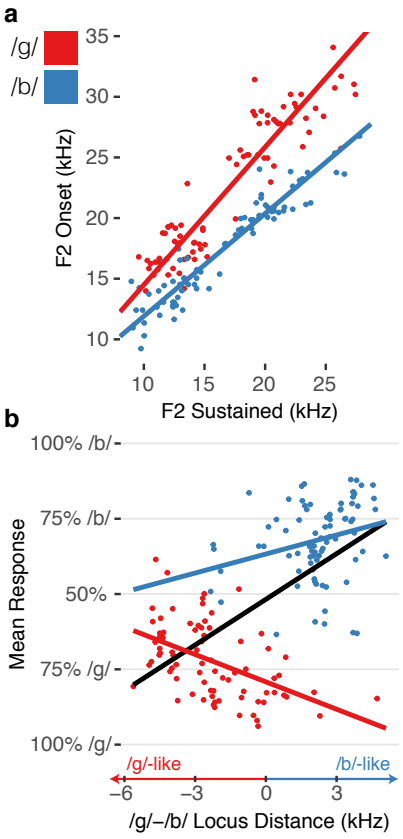


Figure 1.5: Acoustic-Behavior Correlates F2 Onset-Vowel transitions do not explain observed response patterns. **a)** Locus equations relating F2 at burst onset and vowel steady state (sustained) for each token (points), split by consonant (colors, same as **b)**). **b)** As the difference of a token's distance from the ideal /g/ and /b/ locus equation lines increased (x axis, greater distance from /g/, smaller distance from /b/), /b/ tokens obeyed the predicted categorization while /g/ tokens did not (slopes of colored lines).

viation ($\sigma_{/b/} = 1.618kHz$) towards the /g/ line only changed responses by 7.9%. These results suggest that mice did not rely on F2 transitions to categorize these consonants.

We repeated this analysis separately for each training cohort to test whether the two cohorts could have developed different acoustic templates that better explained their response patterns. We derived cohort-specific locus-equation lines and distances using only the tokens from each of their respective training sets. These models were qualitatively similar to the model that included all tokens and mice and did not improve the model fit (Cohort 1: /g/: $b = -0.051, 95\%CI = [-0.064, -0.038]$, /b/: $b = 0.041, 95\%CI = [0.022, 0.059]$; Cohort 2: /g/: $b = -0.022, 95\%CI = [-0.031, -0.014]$, /b/: $b = 0.055, 95\%CI = [0.042, 0.069]$).

We conclude that while our stimulus set had the expected F2 formant transition structure, this was unable to explain the behavioral responses we observed both globally and within training cohorts. There are, of course, many more possible acoustic parameterizations to test, but the failure of F2 transitions to explain our behavioral data is notable because of its perceptual dominance in humans and its common use in parametrically synthesized speech sounds. This demonstrates one advantage of using natural speech sounds: mice trained on synthesized speech that varied parametrically only on F2 transitions would likely show sensitivity to this cue, but this does not mean that mice show the same feature sensitivity when trained with natural speech. Preserving the complexity of natural speech stimuli is important for developing a general understanding of auditory category learning.

1.3 Discussion

These results demonstrate that mice are capable of learning and generalizing phonetic categories. Indeed, this is the first time to our knowledge that mice have been trained to discriminate between any classes of natural, non-species-specific sounds. Thus mice join a number of model organisms that have demonstrated categorical learning with speech sounds [10, 25, 26, 21, 22, 23, 24], making a new suite of genetic and electrophysiological tools available for phonetic research.

Two subgroups of our mice that were trained using different sets of speech tokens demonstrated distinct patterns of consonant identification, presumably reflecting differences in underlying acoustic prototypes. The ability to precisely control exposure to speech sounds provides an opportunity to probe the neurocomputational constraints that govern the possible solutions to consonant identification.

Here we opted to use naturally recorded speech tokens in order to demonstrate that mice could perform a "hard version" of phonetic categorization that preserves the full complexity of the speech sounds and avoids *a priori* assumptions about the parameterization of phonetic contrasts. Although our speech stimuli had the expected F2 formant transition structure, that did not explain the response patterns of our mice. This suggests that the acoustic rules that mice learned are different from those that would be learned from synthesized speech varying only along specifically chosen parameters.

Future experiments using parametrically synthesized speech sounds are a critical next step, and will support a qualitatively different set of inferences. Being able to carefully manipulate reduced speech sounds is useful to probe the acoustic cue structure of learned phonetic categories, but the reduction in complexity that makes

them useful also makes it correspondingly more difficult to probe the learning and recognition mechanisms for a perceptual category that is defined by multiple imperfect, redundant cues. It is possible that the complexity of natural speech may have caused our attrition rate to be higher, and task performance lower, than other sensory-driven tasks. Neither of those concerns, however, detracts from the possibility for the mouse to shed mechanistic insight on phonetic perception. Indeed, error trials may provide useful neurophysiological data about how and why the auditory system fails to learn or perceive phonetic categories.

We hope in future experiments to directly test predictions made by neurolinguistic models regarding phonetic acquisition and discrimination. For example, one notable model proposes that consonant perception relies on combination-sensitive neurons that selectively respond to specific combinations of acoustic features [5]. This model predicts that mice trained to discriminate stop consonants would have neurons selective for the feature combinations that drive phoneme discrimination, perhaps in primary or higher auditory cortical areas. Combination-selective neurons have been observed in A1 [58, 59], and speech training can alter the response properties of A1 neurons in rats [22], but it is unclear whether speech training induces combination-selectivity that would facilitate phonetic discrimination.

The ability to record from hundreds of neurons in awake behaving animals using tetrode electrophysiology or 2-photon calcium imaging presents exciting opportunities to test predictions like these. Should some candidate population of cells be found with phonetic selectivity, the ability to optogenetically activate or inactivate specific classes of neurons (such as excitatory or inhibitory cell types, or specific projections from one region to another) could shed light on the circuit computations and transformations that confer that selectivity.

1.4 Methods

1.4.1 Animals

All procedures were performed in accordance with National Institutes of Health guidelines, as approved by the University of Oregon Institutional Animal Care and Use Comittee.

We began training 23 C57BL/6J mice to discriminate and generalize stop consonants in CV (consonant-vowel) pairs. 13 mice failed to learn the task (see Training, below). 10 mice (43.5%) progressed through all training stages and reached the generalization task in an average 14.9 ($\sigma = 7.8$) weeks. Mean age at training onset was 8.1 ($\sigma = 2$) weeks, and at discontinuation of training was 50.6 ($\sigma = 11.2$) weeks. Sex did not significantly affect the probability of passing or failing training (Fisher's Exact Test: $p = 0.102$), nor did the particular behavioral chamber used for training ($p = 0.685$), nor age at the start of training (Logistic regression: $z = 1.071, p = 0.284$). Although this task was difficult, our training time (14 ± 0.3 weeks as in [22]), and accuracy (generalization: 76%[10], training tokens only: 84.1% [22]) are similar to comparable experiments in other animals.

1.4.2 Speech stimuli

Speech stimuli were recorded in a sound-attenuating booth with a head-mounted microphone attached to a Tascam DR-100mkII handheld recorder sampling at 96kHz/24bit. Each speaker produced a set of 3 recordings (tokens) of each of 12 CV pairs beginning with either /b/ or /g/, and ending with /I/, /o/, /a/, /æ/, /ɛ/, /u/. To reduce a slight hiss that was present in the recordings, they were denoised using a Daubechies wavelet with two vanishing moments in MATLAB. The typical human hearing range is 20 Hz - 20 kHz, whereas the mouse hearing range is 1 kHz - 80 kHz [60]. The F_0 of our recorded speech sounds ranged from 100 - 200 Hz, which is well below the lower frequency limit of the mouse hearing range. We therefore pitch shifted all stimuli upwards by 10x (3.3 octaves) in MATLAB [**?**]. This shifted all spectral information equally upwards into an analogous part of mouse hearing range while preserving temporal information unaltered.

Tokens from five speakers (one male - speaker 1 throughout, four female - speakers 2-5 throughout) were used. Three vowel contexts (/æ/, /ɛ/, and /u/) were not recorded from one speaker. It is unlikely that this had any effect on our results, as our primary claims are based on the ability to generalize at all, rather than generalization to tokens from a particular speaker. Tokens were normalized to a common mean amplitude, but were otherwise unaltered to preserve natural variation between speakers — indeed, preserving such variation was the reason for using naturally recorded rather than synthesized speech.

Formant frequency values were measured manually using Praat [61]. F2 at onset was measured at its center as soon as it was discernible, typically within 20 ms of burst onset, and at vowel steady-state, typically 150-200ms after burst onset.

1.4.3 Training

We trained mice to discriminate between CV pairs beginning with /b/ or /g/ in a two-alternative forced choice task. Training sessions lasted approximately 1 hour, 5 days a week. Each custom-built sound-attenuating training chamber contained two free-field JBL Duet speakers for stimulus presentation with a high-frequency rolloff of 34 kHz, and a smaller 15 x 30 cm plastic box with three "lick ports." Each lick port consisted of a water delivery tube and an IR beam-break sensor mounted above the tube. Beam breaks triggered water delivery by actuating a solenoid valve. Water-restricted mice were trained to initiate each trial with an unrewarded lick at the center port, which started playback of a randomly selected stimulus, and then to indicate their stimulus classification by licking at one of the ports on either side. Tokens beginning with /g/ were always on the left, with /b/ on the right. Two cohorts were trained on two separate sets of tokens. Training set 1 started with speaker 1 (Fig. 4a) and had speaker 2 introduced on the fourth stage, where Training set 2 started training with speaker 5 and had speaker 4 introduced on the fourth stage. Correct classifications received ∼10 µL water rewards, and incorrect classifications received a 5s time-out that included a mildly aversive 60 dB SPL white noise burst.

Training advanced in stages that progressively increased the number of tokens, vowel contexts, and speakers. Mice first learned a simple pure-tone frequency discrimination task to familiarize them with the task and shape their behavior; the tones were gradually replaced with the two CV tokens of the first training stage. CV discrimination training proceeded in 5 stages outlined in Table 2. Mice automatically gradu-

ated from each stage when 75% of the preceding 300 trials were answered correctly. In a few cases, a mouse was returned to the previous stage if its performance fell to chance for more than a week after graduating. Training was discontinued after two to three months if performance in the first stage never rose above chance. Mice that reached the final training stage were allowed to reach asymptotic performance, and then advanced to a generalization task.

In the generalization task, stimuli from the set of all possible speakers, vowel contexts, and tokens (140 total, not including the stage 5 stimulus set) were randomly presented on 20% of trials and the stage 5 stimulus set was used on the remaining 80%. Training tokens were drawn from a uniform random distribution so that each was equally likely to occur during both the stage 5 training and generalization phases. Novel tokens were drawn uniformly at random by their generalization class, but since there were unequal numbers of tokens in each class (Novel token only: 16 tokens, Novel Vowel: 36, Novel Speaker: 54, Novel Speaker + Vowel: 54), tokens in each class had an unequal number of presentations. We note that the logistic regression analysis with restricted maximum likelihood that we used is robust to unequal sample sizes [62].

1.4.4 Data analysis

Data were excluded from days on which a mouse had a $>$ 10% drop in accuracy from their mean performance on the previous day ($44/636 = 7\%$ of sessions). Anecdotally, mice are sensitive to environmental conditions (e.g., thunderstorms), so even though all efforts were made to minimize variation between days, even the best performing mice had "bad days" where they temporarily fell to near-chance performance and exhibited strong response bias. We thus assume these "bad days" were the result of temporary environmental or other performance issues, and were unrelated to the difficulty of the task itself.

All analyses were performed in R (R version 3.5.3 (2019-03-11))[63] using RStudio (1.1.456)[64]. Generalization performance was modeled using a logistic generalized linear mixed model (GLMM) using the R package "lme4"[65]. Binary correct/incorrect responses were fit hierarchically to models of increasing complexity (see Table 1.3), with a final model consisting of the generalization class (as in Fig. 2a: training tokens, novel tokens from the speakers and vowels in the training set, novel speaker, novel vowel, and novel speaker and vowel) as a fixed effect with random slopes and intercepts nested within each mouse as a random effect. There was no evidence of overdispersion (i.e., deviance $\approx$ degrees of freedom, or less than ~ 2 times degrees of freedom), and the profile of the model showed that the deviances by each fixed effect were approximately normal. Accordingly, we report Wald confidence intervals. We also computed bootstrapped confidence intervals, which had only minor disagreement with the Wald confidence intervals and agreed with our interpretation in the text.

Clustering was performed with the "cluster"[66] package. Ward clustering split the mice into two notable clusters, which are plotted in Fig. 1.4.

We estimated locus equations relating F2 onset and F2 vowel using total least squares linear regression. The locus equations of the /b/ and /g/ tokens accounted for 97.3% and 95.9% of the variance in the F2 measurements of our tokens, respectively.

Spectrograms in Figure 1.1a were computed with the "spectrogram" function in

MATLAB 2017b, and power spectra in Figure 1.1b were computed with the "pwelch" function in MATLAB 2018b with the same window and overlap as 1.1a spectrograms.

The remaining analyses are described in the text and used the "binom"[67], "reshape"[68], and "plyr"[69] packages. Data visualization and tabulation was performed with the "ggplot2,"[70] and "xtable"[71] packages.

1.5 Tables

	Accuracy	95% Wald CI	Corr			
Learned	0.767*	[0.748, 0.785]				
Token	0.739*	[0.713, 0.763]	0.5			
Vowel	0.678*	[0.655, 0.701]	0.81	0.91		
Speaker	0.666*	[0.651, 0.68]	0.98	0.68	0.92	
Vow+Spk	0.637*	[0.624, 0.651]	0.98	0.64	0.9	1

Table 1.1: Impact of each generalization class on performance. Accuracy values provide an estimate of the difficulty of that class after accounting for the random effects of individual mice. Accuracies are logistic GLMM coefficients transformed from logits, and model coefficients are logit differences from training set accuracy, which was used as an intercept. Correlation values are between fixed effects (novelty classes) across random effects (mice). * indicates significance ($p(>|z|) \ll .001$).

Stage	Speakers	Vowels	Total Tokens
1	1	1	2
2	1	1	4
3	1	2	6
4	2	2	12
5	2	3	20
Generalization	5	6	160 (20 training, 140 novel)

Table 1.2: Token structure of training stages

	DF	χ^2	DF_{χ^2}	$\Pr(>\chi^2)$
Mouse	2			
Mouse + Type	6	2534.46	4	$\ll 0.001$
Type | Mouse	20	407.22	14	$\ll 0.001$

Table 1.3: hierarchical GLMM: To reach the appropriate complexity of model, we first modeled correct/incorrect answers as a function of each mouse as a fixed effect (row 1), then added the generalization type (as in Fig. 1.2) as a fixed effect (row 2), and finally modeled generalization type as a fixed effect nested within each mouse as a random effect (row 3). Since the final model had the best fit, it was used in all reported analyses related to the GLMM.

Language Games

I had intended to finish my dissertation with an experiment that was the next logical conclusion of the mouse model of phonetics, doing longitudinal mesoscopic calcium imaging of auditory cortex as the phonemes were being learned in order to model the changes in network activity. "Interdisciplinary" neuroscience tends to look more like neuroscientists swinging into other disciplines without taking the time to appreciate their complexity, and so this was an attempt at a synthesis of decades of work across several disciplines in order to inform our work. The picture the model attempts to paint is one of a loose basis set of low-level acoustic representations that fluidly restructure themselves according to their recent informational history to achieve a sort of rough categorical consensus around the ill-defined category of a phoneme — an attempt to move us beyond synthesized speech and trial averaging to naturalistic sounds computed by the dynamic activity negotiated by the brain.

It comes out of chronological order in the spring of 2021, after my work with Autopilot and a covid-induced awakening of the possibility of public engineering with the People's Ventilator Project[72]. I was restless and not ready to return to basic research while the world was still so broken, and so it was abandoned in favor of the last piece on digital infrastructure. Accordingly, it ends relatively abruptly, without satisfying conclusion. I include it here in its unfinished form, roughly edited, warts and all, as something I intend to pick up perhaps one day when basic science is more possible.

2

Phonemes are a Game the Brain Plays

> "Consider for example the proceedings that we call "games". [...] For if you look at them you will not see something that is common to all, but similarities, relationships, and a whole series of them at that. [...] Are they all 'amusing'? Compare chess with noughts and crosses. Or is there always winning and losing, or competition between players? Think of patience. [...] Look at the parts played by skill and luck; and at the difference between skill in chess and skill in tennis.
>
> And the result of this examination is: we see a complicated network of similarities overlapping and criss-crossing: sometimes overall similarities, sometimes similarities of detail. [...] And we extend our concept as in spinning a thread we twist fibre on fibre. And the strength of the thread does not reside in the fact that some one fibre runs through its whole length, but in the overlapping of many fibres."
>
> *-Wittgenstein, Philosophical Investigations: 66-67[73]*

Cognitive reality is characterized by its discreteness: rather than a continuous undifferentiated gradient wash of sensation and cognition, we experience objects, concepts, and categories. Speech is a continuous, high-dimensional, high-variability acoustic signal, yet it is perceived as a small number of relatively-discrete phonemes[74]. The acoustic structure of phonemes is a sort of "Family Resemblance"[73] — the truly extravagant variability of speech has thus far defied any simple, definite acoustic parameterization of its phonemes. Instead, individual utterances within a phonetic category vary along high numbers of feature-dimensions, none of which are necessary nor sufficient for a listener to identify it[75, 76].

There are different types of category structure, and what typifies family resemblance structures is 1) they are **multiply defined** - category membership is assesed across many imperfect 'features' none of which is necessary nor sufficient, 2) **prototypicality** - some instances are better 'examples' of a category than others, category membership is not binary, 3) **context dependence** - which feature is important depends on the features present in the instance and the context in which it is being compared. [77]

2.1 A Very Simple Model...

Category representation theories are intimately related (and occasionally literally isometric to [78]) to theories of the measurement of similarity, which is dominated by geometric models[79]. These models nearly universally presuppose that categories exist in a feature space such that there exist some number of features that describe each instance of an object to be categorized.

To begin perhaps purposely naively, we will formulate a very simple geometric model of perceptual categories:

Suppose that some sensory stimulus **s** was composed of some set of physical attributes a_i in the d-dimensional "stimulus space" **S** capable of fully representing all stimuli for a given sensory modality (as opposed to a particular set of eg. parameterized stimuli)

$$\mathbf{s} = \{a_0, a_i, \dots a_d \ : a \in \mathbf{S}\} \tag{2.1}$$

For example, a digital sound is fully defined by the amplitudes of the waveform at each of its samples, or an image is defined as the wavelength and intensity of light at each pixel. Since a_i are arbitrary, **S** can represent a set of static attributes, or a set of attributes through time.

The sensory stimulus **s** is processed into some percept **p** composed of perceptual attributes b_i in the e-dimensional "perceptual space" **P**

$$\mathbf{p} = \{b_0, b_i, \dots b_e \ : b \in \mathbf{P}\} \tag{2.2}$$

such that some perceptual computation M maps **S** to **P**.

$$M = f : \mathbf{S} \rightarrow \mathbf{P} \tag{2.3}$$

$$\mathbf{p} = M(\mathbf{s}) \tag{2.4}$$

Like **S**, the form of **P** is arbitrary, so while the discussion that follows treats it like a continuously-valued metric space, it could also consist of a collection of binary/discrete properties (like traditional phonetic descriptions like [± voiced]), as in, for example [79, 80]

The objective of the observer is to infer the category c_s given **s**'s representation as **p**.

$$c_s = max(\{p(c_i|\mathbf{p}) : c_i \in \mathbf{C}\}) \tag{2.5}$$

The form of the sensory-perceptual mapping M, the perceptual space **P** it constructs, and the inference of category identity c_s it supports serve as a loom for a few threads of the speech perception problem scattered across a few disciplines and vocabularies.

2.2 ...and its history

A prominent strain of phonetics research in the US, largely associated with the Haskins Labs ([81] and see [82, p. 51]), has characterized the speech perception problem as resolving a set of acoustic "cues" into phonetic identity:

> "Liberman, Cooper, and Pierre Delattre began to study the acoustic speech signal, to determine how it represents the consonants and vowels of spoken words, and to discover the acoustic structure (the 'cues') essential for their identification by listeners. [...] By selectively including and eliminating elements of acoustic structure,l Liberman and his colleagues could determine what bits of structure provided information for the different phonetic properties of spoken words."

> -Carol Fowler & Katherine S. Harris in [82, p. 51]

The "cue discovery" paradigm of phonetics research posits that, for the auditory component of phonetic perception, the elements in **P** are linear combinations of the features in **S** whose manipulation can influence the identity of the perceived phoneme. These features represent familiar phonetic parameterizations like voice onset times or formant frequency ratios. The mapping M that constructs **p** is taken to be a fixed, innate feature of the auditory system: "this version of the auditory theory takes the perceived boundary between one phonetic category and another to correspond to a naturally-occurring discontinuity in perception of the relevant acoustic continuum." [83].

The conclusion of cue-based research is summarized neatly by Philip Rubin, Robert Remez, and Jennifer Pardo with respect to their sinewave synthesis experiments: "Question: Which acoustic elements are essential for the perception of speech? Answer: None[84]." The failure to find a simple parameterization of phonetic categories as acoustic cues motivated an abandonment of an acoustic account of phonetic perception entirely in favor of a motor theory of perception that posited a special, evolved "speech module" that linked the wily acoustics of speech sounds to the action of the articulatory system:

> "For if phonetic categories were acoustic patterns, and if, accordingly, phonetic perception were properly auditory, one should be able to describe quite straightforwardly the acoustic basis for the phonetic category and its associated percept. According to the motor theory, by contrast, one would expect the acoustic signal to serve only as a source of information about the gestures; hence the gestures would properly define the category" [83]

Purely motor theories of speech have been diversely problematized, not least of all by the many demonstrations that animals that conspicuously lack a human articulatory system are capable of phonetic categorization[17, 25, 26]. The acoustic problem of speech perception was simply too difficult to be solved by an evolutionarily plausible auditory system – how could the family resemblance structure of phonetic categories be learned without some explicit, innate knowledge of the acoustic consequences of articulation?[76]

Research on infant acquisition of speech sounds has since demonstrated the profound plasticity of the auditory system and its ability to learn the complex statistical dependencies between the acoustic attributes of speech[85]. A family of models based primarily on the work of Patricia Kuhl and colleagues describe the stimulus space **S** as acoustic features based on the "basic cuts" of sensitivity in the auditory system[86]. Infants exploit the statistical regularity and patterns of feature co-occurance to learn some mapping M that constructs a "warped" perceptual space **P** that clusters features in **S** into acoustic "prototypes."[85]

Phonetic category identity then consists of some density in **P**, the center of which is the "ideal" phonetic exemplar most likely to be identified with a particular categoery, and proceeding from this center point one transitions from off-target imperfect examplars to overlapping densities of other phonetic categories. Extensions to the model make this formulation explicit, like Kronrod, Coppess, and Feldman's[7] bayesian model that offers a unified explanation of the strong categorical perception of stop consonants and the weaker categorical perception of vowels. Their model describes phonetic identification as an inference problem that depends on both the

acoustic properties of a stimulus and prior knowledge of phonetic categories, defined as some mean and variance in an arbitrary perceptual space.

In this model, the difficulty of the acoustic problem of speech perception carefully described by cue-centric phonetic research is resolved by suggesting the auditory system relies on sharp internal representations of category identity for phonemes that have a large degree of uninformative variance, like stop consonants.

The degree of arbitrariness is problematic for the model, however. The proposition that there is some stimulus space **P** that supports linearly-separable phonetic categories is emphatically counterevidenced by the 70 years of cue-based research that has attempted to find one[87, 75]. These prototype models, without weighting for the informativeness of a particular dimension in context (as opposed to some global weight) would be vulnerable to misidentifying speech when the most dominant cue was made redundant, when in fact human listeners will adapt to using a more informative cue. A lot of the supporting research relies on carefully parameterized speech, so one might expect such a single-density-based prototype model would fail if the stimulus set contained instances where there was some mixture or inversion of informativeness of cues, as in real speech. Having nonlinear blobby parameterizations of prototypes doesn't really solve the problem either, as you would then just require an additional downstream 'readout' layer that could compute the conditions where a particular dimension.

They describe future directions of research as the studying the process by which listeners identify and learn the underlying dimensional structure, and so in that spirit we can extend our model by continuing Kronrod's emphasis on the information contained in each perceptual dimension and allowing it to vary by context.

2.3 An extension to our model...

Instead of a static perceptual space **P** where a given stimulus **s** is mapped to a single percept **p** (ie. M is injective), we can extend our very simple model by introducing some notion of reweighting perceptual dimensions. Rather than inferring category directly from **P** as in eq. 2.5, the features $b_e \in \mathbf{P}$ are reweighted by some weight vector **w** computed as some function W of the representation $\mathbf{p} = M(\mathbf{s})$ and some prior knowledge of the category structure of **C**

$$\mathbf{w} = W(\mathbf{p}, \mathbf{C}) \tag{2.6}$$

$$c_s = max\big(\{p(c_i|\mathbf{p} \cdot \mathbf{w}) : c_i \in \mathbf{C}\}\big) \tag{2.7}$$

Recall that since the features $a \in \mathbf{S}$ are arbitrary, they can include time-varying features, so the weighting function W can, for example, incorporate contextual effects from the recent perceptual past. Category inference being dependent on W has equivalent interpretations in the parlance of artificial neural networks and geometry: as a self-attention mechanism (eg. [88]) giving higher weight to more informative features, or as "collapsing" or "expanding" un/informative dimensions.

2.4 ... and its implications

The notion of different perceptual features having different weights or importance depending on the acoustic context and the category structure of the phonemes for a particular language is of course far from new (eg. [81]).

A parallel line of thought to the generative models that posit phonetic identity as some positive description of cues or perceptual features are discriminative models that focuses on the features that can be used to tell phonemes apart. A prominent family of discriminative models in phonetics are those that describe a hierarchy of contrastive features[46, 89, 90]. Though they are diverse in their details, in these models M is again typically some fixed feature of the auditory system, and the perceptual space **P** that it constructs is some set of high-level descriptions like voicing, frication, or articulator configuration. Typically these features are binary (eg. +/- voiced), rather than continuous.

As an example, consider the proposed contrastive feature hierarchy for russian vowels from [91] (Figure 2.1). Vowel identification is dominated by the primary contrast of +/- back, and successive constrastive features eliminate candidate phonemes until the true phoneme is identified. W's dependence on **C** is exemplified (fix passive voice..) by its treatment of "round": -back vowels [e i] are fully determined by +/- high, so for a percept **p** with -back, the weight of "round" should be 0. Put another way, the importance of a given feature is dependent on the phonemes that are left ambiguous without it. Any given feature's importance depends on both the set of available features and the set of available categories (the dependence of W, and thus roughly the "meaning" of **P**, on **C** can also be thought of as the "task demand" on phonetic perception, see for example the discussion of [80] in [92]). The only problem is that features like +rhotic don't correspond to anything in the input space, and presuppose either an articulatory or intrinsic auditory model of perception [93].

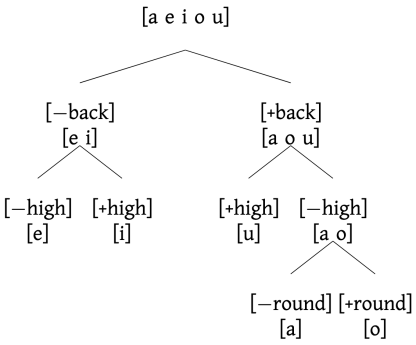


Figure 2.1: Contrastive hierarchy for Russian Vowels, reproduced from [91] without permission

The designedness of parameterized stimuli has the same problem: shouldn't it matter how bad they sound for claims about natural speech perception? The question for perception, rather than computing the values of pre-programmed cues, is how the different perceptual axes are normalized/selected/weighted. This is necessarily a multi-timescale process: during learning, how does the auditory system learn the space of features? When there is only one feature present, as in parameterized speech, the auditory system is performing a qualitatively different task than during the perception of natural speech — and thus elide a strong assumption on the nature of the problem that the auditory system is solving.

But, since there is some "basic cuts" argument to be made about the auditory system and the types of cues that it selects, there is some reason to study speech sounds in particular, as opposed to stimuli with some arbitrary category structure: speech sounds come pre-optimized for mammalian auditory systems a la adaptive dispersion. This is noted by researchers studying learning arbitrary feature spaces:

> "I should emphasize, nevertheless, that there is a great deal of evidence that practice, even large amounts of it, does not produce efficient perception of acoustic alphabets. This is clear, not only in the example of the Morse code, but even more convincingly, perhaps, in the repeatedly unsuccessful attempts to find non-speech sounds that will work well as part of a reading machine for the bling. Many sound alphabets have been given a thorough trial, but none has proved adequate. It must surely give us pause to know that, while sounds are the universal carriers

> of language, only one set of sounds — those of speech — serves well."[94]

Their conclusions — that this means that speech is special and has its own processing modality — invert the problem. But the observation does indeed point to the joint optimization of a phonetic space over an auditory space as being constitutive of language, and a potent reason to use speech sounds for category learning.

The notion of the informativeness of different featural dimensions has been given a rather polemic treatment in Keith Kluender and Christian Stilp's application of information theory to phonetic perception[95, 96, 97, 98]. They summarize their argument, elegantly as always

> "If one's problem is finding the right fencing to corral a unicorn, then there is really no problem at all. Instead the problem is dissolved upon discovery that unicorns do not exist.
>
> Here, we ask the reader to consider the possibility that there are no objects of perception [...]. Like unicorns, they do not exist at all. Instead, there are *objectives* for perception. [...] Perceptual success does not require recovery or representations of the world per se." [95]

They argue that the central operation of sensory systems is to adapt to regularity at multiple scales in order to efficiently extract meaningful information from their environment. Rather than a faithful representation of articulatory maneuvers (as in motor theory) or a warped, but still bijective relationship between the acoustic space and perceptual space (as in perceptual warping), they argue that sensory systems discard information that is predictable based on (multiscale) context, and instead represent just the unpredictable, "information-bearing" in an appropriately Shannonistic sense, dimensions.

Though theoretically all configurations of frequencies and amplitudes are possible, naturally produced sounds are strongly constrained by the physics of their production – much of the variation in natural sounds is predictable. Rather than representing the fullness of acoustic variation, the auditory system adapts to redundancies and regularities in sounds to preferentially represent only the unpredictable, informative variation in an "efficient code" [99, 100]. In the case of phonetic perception, where the objective of the listener is to identify the phoneme intended by the speaker rather than perceiving a sound qua sound, the listener attempts to learn auditory features that are maximally informative of phonetic identity[101, 102, 95, 96].

This information-theoretic account provides a mechanism for learning the dimensions of $\mathbf{P}$ and the form of W. Rather than some a priori, fixed inventory of articulatory/acoustic cues, a listener should learn some set of perceptual features that support the identification of phonemes given the phonemic inventory of their language and the acoustic variability (eg. accent, environment, timbre, etc.) that they are exposed to. Individual listeners do indeed use different combinations of cues with different weights[103] which are stable over time[104]. Rather than learning some category center and spread over some pre-existing perceptual feature space, the task of the listener is to learn the feature space itself.

The difference between learning $\mathbf{P}$ and the operation of W is a matter of timescale: over short timescales, W reweights the features in $\mathbf{P}$ depending on those features that are contextually informative of phonetic identity. While the observation that individual cues are informative, uninformative, and anti-informative depending on

the context of surrounding phonemes is a central feature of argument for a motor theory[76], an information-theoretic view interprets this problem as a reweighting of individual features: /s/ differs from /f/ along different featural axes than /s/ differs from /k/, so /s/ shouldn't necessarily rely on the same inventory of acoustic features in all contexts — and particularly when cues are rendered uninformative, the auditory system should adapt to emphasize those that still are(eg. as in [101] and [105]). Contextual effects on phonetic categorization are of course well known (see [74]). Where perceptual warping accounts cannot explain results where some or all of the typical acoustic features are replaced, like sine-wave speech[106], noise-vocoded speech[107], or joint spectrotemporal degradation[108]; an information-theoretic view argues that listeners will adapt to use any cues that are still present (as in [101]).

The auditory system does *not* seem to operate in an entirely information-maximizing way when identifying phonemes, however. Consider a category structure like that used by Couchman, Coutinho, and Smith (2010, [109]) depicted in figure 2.2. Each stimulus is composed of four binary features (columns), and stimulus identity is defined by the first feature (0 = category A, 1 = category B). The remaining three features are "epiphenomenal," but stimuli in category B have a greater sum than those in category A. A perfect, information-maximizing observer would learn to only attend to the first dimension, but in speech and many other perceptual categories observers use many, even uninformative dimensions[109, 77] (but see [110]). Non-speech sounds that are strictly uninformative of phonetic identity like pure tones and sweeps can nevertheless strongly influence the perceived phoneme[111, 112], even when the sounds are not immediately adjacent[113].

Category A	Category B
0 0 0 0	1 1 1 1
0 1 0 0	1 0 1 1
0 0 1 0	1 1 0 1
0 0 0 1	1 1 1 0

Figure 2.2: Category structure reproduced from [109] without permission. Each stimulus (row of four digits) is composed of four features (columns). Category identity is determined by the first feature (0 = A, 1 = B), but three other "irrelevant" features are present.

The messy compromise between learning maximally informative dimensions and the influence of non-informative dimensions has its reflections in infant speech acquisition as well. Infants are able to discriminate between the phonemes of any language, but during language acquisition become specifically attuned to the phonemes of the language(s) they are taught. This is typically discussed as learning the statistical regularities of speech sounds [114, 86], and the act of *emphasizing* the statistical regularity must necessarily mean *collapsing* those phonetic contrasts that are not present in the language – they aren't informative because no one uses that contrast. Infants that are better at discriminating the phonemes in their language are worse at discriminating those in a non-native language[114], but the ability doesn't drop to zero, indicating they are still present in some form, perhaps as a balance between "exploration" of potentially unaccounted for variation with "exploitation" of learned invariances.

Such an influence of many, imperfect stimulus dimensions on perception is our signpost to indicate we've arrived back in the bewildering little shire of category structures with family resemblance.

2.5 Neural Mechanisms

Until now our very simple model has been entirely theoretical, describing the general requirements of the computation of phonetic category identity, but the form of any biological computation is necessarily constrained by the substrate of its implementation (roughly, Marr's levels, for a recent discussion see [115]). Though the model could be retained in its current form by recasting **P** as the neural repre-

sentation of perceptual dimensions from which category $c \in \mathbf{C}$ is inferred, this would require strong assumptions about the form of the neural representation of perceptual dimensions, and in a practical modeling context assumes we have enough information to infer it. To preserve generality at the cost of complexity, we add an additional "layer" to the model,

$$\mathbf{n} = \{n_0, n_i, ...n_{dn} : n \in N^{dm} \subseteq \mathbb{R}^{dn}\} \tag{2.8}$$

where a neural state **n**, a dn-dimensional instantaneous firing rate of neurons n_i in some neural manifold $\mathbf{N}^{dm}$ of dimension dm embedded within $\mathbb{R}^{dn}$. The manifold embedding N reflects the intrinsic constrants network structure poses on the possible states $\mathbf{n} \in \mathbf{N} \subseteq \mathbb{R}$, but the embedding is arbitrary.

The neural layer is incorporated by modifying equation 2.6 such that

$$M_n = f(\mathbf{s}, \mathbf{p}) : \mathbf{S} \rightarrow \mathbf{N} \tag{2.9}$$
$$M_p = f(\mathbf{n}) : \mathbf{N} \rightarrow \mathbf{P} \tag{2.10}$$

where some sensory input **s** is mapped to some neural state **n**, which supports some percept **p** from which phonetic category is computed. The dependence of M_n on **p** reflects the possibility of top-down influence on the neural representation of a given stimulus.

What we're doing here is largely accounting for incomplete observation of the implementation of perceptual representation. For example there might be some real perceptual dimension that is not independently represented in the neural space, but is computed "downstream" by some structure that we're not observing. In the case of making a claim on the structure of neural representation (eg. that short-run firing rates are meaningful) with full observation, $\mathbf{N} = \mathbf{P}$, where **P** is then the perceptual space represented by the brain from which category identity is computed. This is the typical assumption of "decoding analyses." Instead our expansion of the model allows us to consider a "basis set" of neural representations that are non-isomorphic with their perceptual representations.

Arguably a computational strategy common to all sensory systems is to exploit regularities in the statistical structure of the natural world to form an efficient sensory representation[116, 117, 99, 98, 118, 119]. Though the task of phonetic perception is a truly monstrous one, work since the heyday of motor theories has demonstrated the remarkable ability of the auditory system to perform the fundamental computations of phonetic categorization has given the problem an air of tractability. And though we still are methodologically limited in our ability to study spech perception in humans at the spatiotemporal scales of its computation, work in animal models as well as recent advances in human brain electrophysiology have given some of the first glimpses.

Several features of our model are happily known to be true of neurons in mammalian auditory cortex.

Neurons in primary auditory cortex jointly encode multiple dimensions of sound [117]. In ferrets presented with an array of stimuli that varied by pitch, timbre, and azimuth [120], more A1 neurons were observed to be sensitive to two or three dimensions (36% and 29%, respectively) than a single dimension (23%). In a subset of neurons, these responses were temporally complex such that the dimensions could be partially recovered by separating sustained from onset responses [121]. Similar results have been observed in marmosets (combined sensitivity to amplitude modulation, frequency modulation, etc. [59]) and in studies that estimated the dimensionality of receptive fields from complex stimuli like dynamic ripples in cats [122].

This is perhaps unsurprising, as cortical neurons being sensitive to multiple dimensions of a stimulus is a trivial reformulation of the well-known model of hierarchical processing throughout the auditory system (for a review, see [123]): cortical neurons representing "higher order" properties of a stimulus necessarily implies sensitivity to multiple features of the stimulus (provided a generously-enough low-level description of the stimulus feature space).

Maciello and colleagues recently argued that joint, rather than independent encoding of multiple stimulus dimensions is computationally advantageous [124]. Though sensitivity to multiple features makes response patterns ambiguous with respect to the value of any individual dimension, joint encoding provides more information about all represented dimensions to a downstream decoder. If it is the case that joint encoding is constitutive of auditory representations, and individual stimulus or perceptual dimensions are never (or rarely) represented independently, behavior that reflects sensitivity to family resemblance structure rather than optimal rule-based categorization is parsimonious. If all features are estimated simultaneously, influence of "nontarget" dimensions becomes unsurprising.

A rich body of research has described the many conditions that auditory representations are modulated by context (for a review, see [125]) at timescales as short as hundreds of milliseconds[126, 127]. Processes like forward masking, stimulus-specific adaptatation (SSA), and suppression of background noise all reflect the general principle that auditory representations adapt to predictable acoustic statistics in order to form robust, invariant representations of auditory objects[128] by emphasizing the maximally informative dimensions[122].

Adaptation to noise or stimulus statistics can be characterized as a short-term 'reweighting' of features through processes like synaptic depression[129, 130] or microcircuit interactions[131, 132]. In tasks based on simple parametric sounds, representations of task-relevant stimuli are enhanced on the order of minutes[133]. Animals trained on multiple tasks had neurons that adapted their receptive fields to facilitate the different task demands[134] and reward structures[135]. David and Shamma [136] argue that short-term integration of auditory context could also be a substrate for representing and comparing auditory features that occur through time.

The auditory system is also plastic on longer timescales to represent the dimensions of sound that are maximally informative to the demands placed on it. Rats trained using a single set of stimuli had differential enhancement of sensitivity to frequency or intensity depending on which they were trained to attend to[137]. Bieszczad and Weinberger observed that such enhancement correlated with the strength of a learned memory trace[138].

Speech in the Brain

The Superior Temporal Gyrus (STG) in humans, or secondary parabelt regions in some other species, of auditory cortex is the primary candidate for representation of higher-order auditory features used in speech perception. Damage to the left posterior Superior Temporal Gyrus, containing BA 22 "Wernicke's area," has long been associated with receptive aphasia, but a variety of human and animal studies have given further insight on the character of speech processing within the STG.

A series of studies from Edward Chang and colleagues recording electrophysiological activity in human temporal lobe using high-density multi-electrode arrays have contributed greatly to our understanding of the encoding of speech sounds, partic-

ularly in the superior temporal gyrus (STG) [139].

Recordings of high-gamma (70-150Hz) power show individual electrode sites in middle to posterior STG are selective to acoustically similar groups of phonemes (eg. obstruent vs. sonorant selectivity, plosive vs. fricative selectivity, etc.) in humans passively listening to natural speech samples [140]. These phonetic sensitivites were reflective of sensitivity to multiple complex acoustic features that are correlated within phonetic categories and that "maximiz[e] vowel discriminability in the neural domain." [140]. Lower frequency (<50Hz) macrocortigraphy recordings also show that subpopulations of pSTG neurons carry information that allows discrimination of consonant-vowel token category analogously to behavioral categorization [141].

In the anterior STG (aSTG), individual sorted units recorded from one person demonstrated complex, speech-specific reponses when one subject was presented with a wide array of sounds [142]. Many (66 of 141) units demonstrated selectivity to one or a few words that was invariant across speaker. Speech selectivity was only partially explained by a linear combination of acoustic features (linear spectrograms and MFCCs), and did not (over-)generalize to noise-vocoded speech, time-reversed speech. Unit responses to individual phonemes also differed by the recent phonetic past, all together suggesting that some units in aSTG are selective to the fine spectrotemporal structure of speech sounds at single-to-few phoneme timescales [142].

Though acoustic response profiles are spatially heterogeneous across the STG and between individuals [140, 143], there does appear to be some functional distinction between anterior and posterior STG with respect to speech sound processing. In macroelectrode recordings in humans listening to natural sentences, pSTG electrodes selectively track phrase-level onsets, while aSTG electrodes have more sustained responses through a phrase. The dissociation between onset and sustained responses was not reflective of the discontinuous vs. continuous nature of consonants and vowels, as selectivity to groups of phonemes (vowels, plosives, nasals, etc.) was mixed in both anterior and posterior STG [143]. Information useful for discrimination of phonetic identity in the pSTG develops and reaches a peak 100-150ms or so after speech sound onset [140, 141], and neural state space projections onto axes representing the activity of neurons sensitive to sound onset or sustained sound show a reliable sweep between posterior and anterior STG on the order of seconds. In short, the picture that emerges is multiple "codes" or "codecs" that overlap in multiple regions at multiple timescales.

Animal research of neural mechanisms of speech sound processing is quite sparse, and so our understanding is relatively coarse and by analogy from more general auditory research. Speech training in rats evokes a complex set of changes to acoustic response properties in several auditory cortical fields loosely analogous to secondary cortical areas in humans[144]. Neurons in the anterior auditory field (AAF) and A1 were more responsive to the initial consonant in consonant-vowel (/CV/) pairs in trained vs. control rats (27% and 57% more spiking activity, respectively). Additionally, the proportion of neurons that were responsive to 2kHz tones (the spectral peak in the speech tokens used) increased by 65% in AAF and 38% in A1 after speech training compared to control rats. In contrast, in response to vowels VAF and PAF were less responsive following speech training (42% and 30% fewer spikes, respectively, vs. controls). In neurons that had similar frequency tuning, responses to consonants were more correlated in AAF and VAF, and responses to vowels were less correlated in AAF, A1, and VAF after speech training (vs. controls)[144].

These results [144] may not establish definitive roles for secondary auditory fields in rodent auditory cortex, but in sum do suggest that speech training induces long-lasting plasticity in auditory cortex, and suggests that processing may be distinct for different acoustic features in anterior vs. posterior fields as in humans. Mice trained to discriminate speech sounds were returned to chance following lesions of auditory cortex [4], indicating its necessity. Task-specific plasticity [145] and contribution to processing task-relevant auditory stimulus categories [146] has been previously demonstrated in AAF, which is thought to operate as a parallel processing system, with response latencies comparable to or lower than A1 in cats [147] and mice [148]. PAF is a secondary auditory cortical area and thought to be downstream from both A1 and AAF [149, 150]. Though their functional specialization of computational role might not be equivalent in humans, it is parsimonious to assume that primary and secondary auditory cortical areas in nonhuman mammalian auditory systems process acoustic information in such a way that supports the recognition of phonetic identity.

2.6 Towards a Research Program

In lightly constraining the constitution of **N**, loosely the neural "representation" of phonetic information, the human and animal results hint at the dissociation between **N** and **P** in our model — en passant to the statement of the research problem.

Suppose that one dimension $b_{vot} \in \mathbf{P}$ is the voice onset time, which dissociates voiced from unvoiced consonants (eg. /b/ vs. /p/) as the time between the onset of phonation and the occlusion of the stop. Further suppose a neural system analogous to the temporal landmark model suggested by [143] where the high-energy plosive of the occlusion is "encoded" by the activity of some region analogous to the phrase-onset sensitivity of pSTG, and the sonorant, spectral quality of the voicing is encoded by another region. In this scheme, some downstream region infers VOT by comparing the relative timing of gross spiking activity between these two regions. In this hopelessly naïve instantiation of our model, the dimension b_{vot} is some real-valued (though not necessarily linear) value from negative to positive voice onset times. Such a dimension is not present in **N** as characterized by the n-dimensional space of, say, instantaneous firing rate of n neurons, requiring M_p, the mapping between them.

The dissociation of the descriptions of **N** and **P** thus, in our model, defines the research problem:

1) We characterize the problem the brain faces in auditory phonetic perception is to learn some perceptual space **P** of maximally informative perceptual dimensions that supports the identification of received phonemes by flexibly adapting to the information present in the phoneme.

2) Understanding the neural mechanisms of auditory phonetic perception is describing the way **P** is implemented by some neural state manifold **N** in such a way that the information loss by the model projecting **N** to **P** is minimized.

It is not necessarily the case that we should expect to find neurons, or even collections of neurons, whose time-averaged firing rate is the literal measurement of the perceptual dimensions used to compute phonetic identity. We also don't expect to be able to estimate the full manifold of all neurons that are involved with the process, so there will ultimately always be some gap between **P** and **N**. Roughly, kept

independent, **P** is the level of "representation" — the basis from which the brain derives its use of phonetic information, though we dont' characterize **P** as the unique source of information, as information is represented at multiple scales (syntactic, semantic) and is bidirectional (predictive as well as receptive) — while **N** is the level of "implementation".

This distinction may read as trivial, but it precludes a majority of the common methodological kinks of contemporary cognitive neuroscience. The implicit assumption of "decoding"-based analysis strategies is that neural representation is encoded in the language of time-averaged firing rate, and that the accuracy of some (usually uninterrogated) classification algorithm on the timeseries of firing rates (or BOLD level, or EEG bandpass amplitude, etc.) is reflective of the presence or absence of category information in the data. The same assumption is made in the case of so-called "Representational Similarity Analysis," and any number of other analytical ruts that uncritically characterize the geometry of the brain and the perceptual reality it supports as euclidean spaces with the axes of whatever recording methodology is handy for the dataset.

In both, the geometry of the perceptual space is also typically uninterrogated, where the parameters that were used to synthesize the stimuli, or the category labels imposed by the researcher are analyzed as if they were faithfully represented by the brain. This, despite the creation of non-isomorphic representations of physical phenomena being the entire goal of efficient perception (see [97, 151]) — if representation operated like an isomorphism then perceptual learning would be entirely unnecessary.

Rather than assuming the perceptual structure of phonemes by prespecifying cues and synthesizing sounds, or assuming the representational language of the brain to be time-averaged firing rate, we take the role of empirical geometers and attempt to preserve as much of the natural complexity of the problem and derive both from the data. The combination of apparent category structure without clear parameterization of naturally produced phonemes is, I argue, a promising way to interrogate the similarly loose category structure of the neural computations that support their perception.

(Here is where I would have continued on to describe the experiments that would follow, if I did not abruptly change disciplines.)

Part II

Autopilot

Neuroscience needs behavior, and behavioral experiments require the coordination of large numbers of heterogeneous hardware components and data streams. Currently available tools strongly limit the complexity and reproducibility of experiments. Here we introduce Autopilot, a complete, open-source Python framework for experimental automation that distributes experiments over networked swarms of Raspberry Pis. Autopilot enables qualitatively greater experimental flexibility by allowing arbitrary numbers of hardware components to be combined in arbitrary experimental designs. Research is made reproducible by documenting all data and task design parameters in a human-readable and publishable format at the time of collection. Autopilot provides a high-level set of programming tools while maintaining submillisecond performance at a fraction of the cost of traditional tools. Taking seriously the social nature of code, we scaffold shared knowledge and practice with a publicly editable semantic wiki and a permissive plugin system. Autopilot's flexible, scalable architecture allows neuroscientists to work together to design the next generation of experiments to investigate the behaving brain.


DOCS


SOURCE


WIKI


PAPER SOURCE


PAPER PLUGIN

We would like to acknowledge and thank Lucas Ott and Tillie Morris for doing most of the behavioral training and being so patient with the bugs, Brynna Paros and Nick Sattler for their help with constructing our behavioral boxes, Chris Rogers who has been brave enough to adopt and contribute to Autopilot in its roughest state, Arne Meyer, Mikkel Roald-Arbøl, and David Robbe who have contributed code and advice, Mackenzie Mathis, Alex Mathis, Gonçalo Lopes, and Gary Kane who collaborated on the DeepLabCut-Live project and provided many a mentorship along the way, Jeremy Delahanty for his inspiring tenacity and humbleness in thinking about better research tools, Matt Smear and Reese Findley for loaning us their Bpod for far longer than they intended to, John Boosinger and the rest of the staff in the machine shop for all their advice and letting me use all their tools, Erik Flister whose Ratrix software inspired some of the design features of Autopilot [1], Santiago Jaramillo whose TASKontrol[2] gave inspiration for GUI design and served as an early scaffolding to learn Python, my labmates Molly Shallow and Sam Mehan who kept me afloat in my last months of dissertation writing, Rocky Penick for her help strapping Autopilot onto the back of Evan Vickers' mesoscope rig (and Evan for letting us play with his rig), several artists on `flaticon.com` *(Freepik, Nikita Golubev, Those Icons) whose work served as stems for some of the figures, and the Janet Smith House for the endless support and relentless criticism of the figures. This material is based on work supported by NIH NIDCD R01 DC-015828, NSF Graduate Research Fellowship No. 1309047, and a University of Oregon Incubating Interdisciplinary Initiatives award.*

Contribution Statement: JLS designed and wrote the software, documentation, wiki, figures, ran the tests, and wrote and edited the paper. LO trained the animals and did an extensive amount of beta testing, bugfinding, and made some of the hardware designs on the wiki. MW mentored, edited the paper, and beta tested the software.

3

Introduction

Animal behavior experiments need precision and patience, so we make computers do them for us. The complexity of contemporary behavioral experiments, however, presents a stiff methodological challenge. For example, researchers might wish to measure pupil dilation[152], respiration[153], and running speed[154], while tracking the positions of body parts in 3 dimensions[155] and recording the activity of large ensembles of neurons[156], as subjects perform tasks with custom input devices such as a steering wheel[157] while immersed in virtual reality environments using stimuli synthesized in real time[158, 159]. Coordinating the array of necessary hardware into a coherent experimental design—with the millisecond precision required to study the brain—can be daunting.

Historically, researchers have developed software to automate behavior experiments as-needed within their lab or relied on purchasing proprietary software (eg. [160]). Open-source alternatives have emerged recently, often developed in tandem with hardware peripherals available for purchase [161, 162]. However, the diverse hardware and software requirements for behavioral experiments often lead researchers to cobble together multiple tools to perform even moderately complex experiments. Understandably, most software packages do not attempt to simultaneously support custom hardware operation, behavioral task logic, stimulus generation, and data acquisition. The difficulty of designing and maintaining lab-idiosyncratic systems thus defines much of the everyday practice of science. Idiosyncratic systems can hinder reproducibility, especially if the level of detail reported in a methods section is sparse[163]. Additionally, development time and proprietary software are expensive, as are the custom hardware peripherals that are required to use most available open-source behavior software, stratifying access to state-of-the-art techniques according to inequitable funding distributions.

Technical challenges are never merely technical: they reflect and are structured by underlying *social* challenges in the organization of scientific labor and knowledge work. Lab infrastructure occupies a space between technology intended for individual users and for large organizations: that of *groupware*[1] [165, 164]. Experimental frameworks thus face the joint challenge of technical competency while also embedding in and supporting existing cultures of practice. Behind every line of code is an unwritten wealth of technical knowledge needed to make use of it, as well as an unspoken set of beliefs about how it is to be used — labs aren't born fresh on release day ready to retool at a moment's notice, they're held together by decades of duct tape and run on ritual. The boundaries of this "contextual knowledge" extend fluidly beyond individual labs, structuring disciplinary, status, and role systems in scientific work[166]. Given their position at the intersection of scientific theory, technical work, data production, and social organization, experimental frameworks are an elusive design challenge, but also an underexplored means of realizing some of our loftier dreams of open, accessible, and collaborative science.

[1] "Our original definition of groupware was 'intentional group processes plus software to support them.' It has both *computer* and *human* components: software of the computer and 'software' of the people using it. [...] Recently this definition has been extended to include other more expressly cultural factors including myth, values and norms. The computer software should reflect and support a group's purpose, process and culture."

Peter and Trudy Johnson-Lenz (1991)[164]

Here we present Autopilot, a complete open-source software and hardware framework for behavioral experiments. We leverage the power of distributed computing using the surprisingly capable Raspberry Pi 4[2] to allow researchers to coordinate arbitrary numbers of heterogeneous hardware components in arbitrary experimental designs.

[2] See Table 5.2

Autopilot takes a different approach than existing systems to overcome the technical challenges of behavioral research: *just use more computers*. Specifically, the advent of inexpensive single-board computers (ie. the Raspberry Pi) that are powerful enough to run a full Linux operating system allows a unified platform to run on every Pi or other computer in the system so that they can work together seamlessly. At the core of its architecture are networking classes (Section 5.8) that are fast enough to stream electrophysiological or imaging data and flexible enough to make the mutual coordination of hardware straightforward.

This distributed design also makes Autopilot extremely scalable, as the Raspberry Pi's $35-$75 price tag makes it an order of magnitude less costly than comparable systems (Section 4.3.4). Its low cost doesn't come at the expense of performance or useability: Autopilot provides an approachable, high-level set of tools that still have input and output precision between dozens of microseconds to a few milliseconds (Sections 4.1.2 and 6).

Autopilot balances experimental flexibility with support. Its task design infrastructure is flexible enough to perform arbitrary experiments, but also provides support for data management, plotting task progress, and custom training regimens. We try to bridge multiple modalities of use: use its modular framework of tools out of the box, or use its complete low-level API documentation[3] to hack it to do what you need. Rather than relying on costly proprietary hardware modules, users can take advantage of the wide array of peripherals and extensive community support available for the Raspberry Pi. Autopilot is designed to be *permissive*: build your whole experiment with it or just use its networking modules, adapt it to existing hardware, integrate your favorite analysis tool. We designed Autopilot to *play nice* with other software libraries and existing practices rather than force you to retool your lab around it.

[3] https://docs.auto-pi-lot.com

Finally, we have designed Autopilot to help scientists do reproducible research and be good stewards of the human knowledge project. Experiments are not written as scripts that are reliant on the particularities of each researcher's hardware configuration. Instead, we have designed the system to encourage users to write reusable, portable experiments that can be incorporated into a public central library while also allowing space to iterate and refine without needing to learn complicated programming best-practices to contribute. Every parameter that defines an experiment is automatically saved in publication-ready format, removing ambiguity in reported methods and facilitating exact replication with a single file. Its plugin system is built atop a densely-linked semantic wiki[4] that fluidly combines human- and computer-readable, communally editable technical knowledge that surrounds your experiments with the software that performs them.

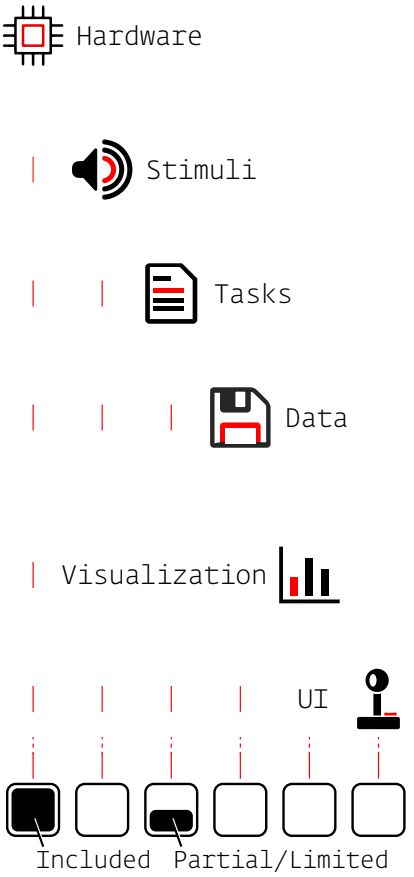
[4] https://wiki.auto-pi-lot.com

We begin by defining the requirements of a complete behavioral system and evaluating two current examples (Sections 3.1 and 3.2). We then describe Autopilot's design principles (Section 4) and how they are implemented in the program's structure (Section 5). We close with a demonstration of its current capabilities and our plans to expand them (Sections 6 and 7).

3.1 *Existing Systems for Behavioral Experiments*

At minimum, a complete system to automate behavioral experiments has 6 requirements:

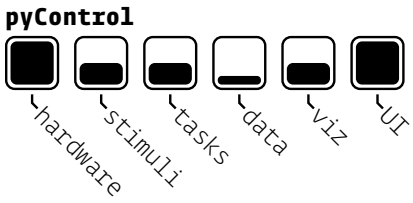
1. Hardware to interact with the experimental subject, including **sensors** (eg. photodiodes, cameras, rotary encoders) to receive input and **actuators** (eg. lights, motors, solenoids) to provide feedback.
2. Some capability to synthesize and present sensory stimuli. Ideally both discrete stimuli, like individual tone pips or grating patches, and continuous stimuli, like those used in virtual reality experiments, should be possible.
3. A framework to coordinate hardware and stimuli as a task. Task definition should be flexible such that it facilitates rather than constrains experimental design.
4. A data management system that allows fine control of data collection and format. Data should be human readable and include complete metadata that allows independent analysis and reproduction. Ideally the program would also allow some means of realtime data processing of sensor values for use in a task.
5. Some means of visualizing data as it is collected in order to observe task status. It should be possible to customize visualization to the needs and structure of the task.
6. Finally, a user interface to control task operation. The UI should make it possible for someone who does not program to operate the system.



We will briefly describe two other systems that meet this definition of completeness: pyControl and Bpod.

3.1.1 *pyControl*

pyControl[167] is a behavioral framework built in Python by the Champalimaud Foundation. It uses the micropython microcontroller ("pyboard") as its primary hardware device along with several extension boards sold by openephys. The pyboard has four I/O ports, or eight with a multiplexing expander board. Schematics are available for many other hardware components like solenoid valve drivers and rotary encoders. Multiple pyboards can be connected to a computer via USB and run independent tasks simultaneously with a GUI.



There is limited support for some parametrically defined sound stimuli, presented from a separate amplifier connected using the I2C protocol. Visual stimuli are unsupported.

Like most behavioral software, pyControl uses a finite-state machine formalism to define its tasks. A task is a set of discrete states, each of which has a set of events that transition the task from one state to another. pyControl also allows timed transitions between states, and one function that is called on every event for a rough sort of parallelism. pyControl also allows the use of external variables to control state logic, making these state machines more flexible than strict finite state machines.

```
D 0 2
D 8976 3
D 8976 1
P 8976 Print Statement
D 10162 3
D 10163 2
```

Figure 3.1: pyControl data is stored as plain text, each line having a type (like **D**ata or **P**rint), timestamp, and state

All events and states are stored alongside timestamps as a plain text log file, one file per subject per session (Figure 3.1). Analog data are stored in a custom binary serialization that alternates 4-byte data and timestamp integers.

There is only one plot type available in the GUI, a raster plot of events, and no facility for varying the plot by task type. The GUI is otherwise quite capable, including the ability to batch run subjects, redefine task variables, and configure hardware.

3.1.2 Bpod

Bpod is primarily a collection of hardware designs and an assembly service run by Sanworks LLC. Similar to pyControl, each Bpod behavior box is based on a finite-state machine microcontroller with four I/O ports. Additional hardware modules provide extended functionality. Bpod is controlled using its own MATLAB package, though there are at least two other third-party software packages, BControl and pyBpod, that can control Bpod hardware. A task is implemented as a MATLAB script that constructs a new state machine for each trial, uploads it to the Bpod, and waits for the trial to finish. As a result, only one Bpod can be used per host computer, or at least per MATLAB session. Data are stored as trial-split events in a MATLAB structure.

There are a few basic plots for two-alternative forced choice tasks, but there doesn't seem to be a prescribed way to add additional plots. Bpod has a reasonably complete GUI for managing the hardware and running tasks, but it is relatively technical (Figure 3.2).

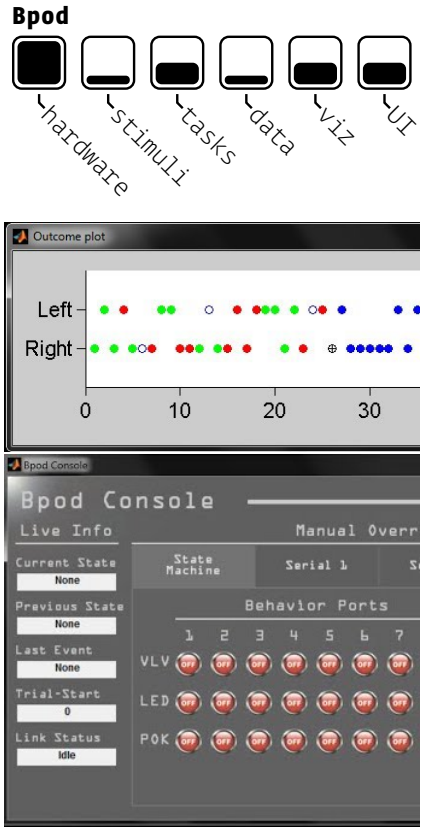


Figure 3.2: A Bpod event plot (above) showing the results of individual behavioral trials, and the Bpod GUI (below).

For brevity we have omitted many other excellent tools that perform some subset of the operations of a complete behavioral system, or otherwise have a substantial difference in scope.[5]

3.2 Limitations of Existing Systems

We see several limitations with these and other behavioral systems:

- **Hardware** - Both Pycontrol and Bpod strongly encourage users to purchase a limited set of hardware modules and add-ons from their particular hardware ecosystem. If a required part is not available for purchase, neither system provides a clear means of interacting with custom hardware aside from typical digital inputs and outputs, requiring the user to 'tack on' loosely-integrated components. There is also a hard limit on the *number* of hardware peripherals that can be used in any given task, as there is no ability to use additional pyboards or Bpod state machines in a single task. The microcontrollers used in these systems also impose strong limits on their software: neither run a full, high-level programming language[6]. We will discuss this further in section 4.2.1. A broader limitation of existing systems is the difficulty of flexibly integrating diverse hardware with the analytical tools necessary to perform the next generation of behavioral neuroscience experiments that study "naturalistic, unrestrained, and minimally shaped behavior"[175].

[5] **Other tools:**
- Bonsai[168] - site, git
- Expyriment[169] - site, git
- PsychoPy[170] - site, git
- OpenSesame[171] - site, git
- SMiLE - docs
- ArControl[172] - git
- and see OpenBehavior

[6] Bpod runs custom firmware written in C++ on a Teensy 3.6 microcontroller. pyControl's pyboard runs micropython, a subset of Python that excludes canonical libraries like numpy[173] or scipy[174]

- **Stimuli** - Stimuli are not tightly integrated into either of these systems, requiring the user to write custom routines for their synthesis, presentation, and description in the resulting data. Neither are capable of delivering visual stimuli. Since the publication of the initial version of this manuscript, Bpod has added support for a HiFiberry sound card that we also describe here[176], but the sound generation API appears to be unchanged, with a single method for generating sine waves. Some parametric audio stimuli are included in the pyControl source code but we were unable to find any documentation or examples of their use.

- **Tasks** - Tasks in both systems require a large amount of code and effort duplication. Neither system has a notion of reusable tasks or task 'templates,' so every user typically needs to rewrite every task from scratch. Bpod's structure in particular tends to encourage users to write long task scripts that contain the entire logic of the task including updating plots and recreating state machines (Figure 3.3). Since there is little notion of how to share and reuse common operations, most users end up creating their own secondary libraries and writing them from scratch. Another factor that contributes to the difficulty of task design in these systems is the need to work around the limitations of finite state machines, which we discuss further in section 5.3.3.

- **Data** - Data storage and formatting is basic, requiring extensive additional processing to make it human readable. For example, to determine whether a subject got a trial correct in an example Bpod experiment, one would use the following code:

  ```
  SessionData.RawEvents.Trial{1,1}.States.Punish(1) ~= NaN
  ```

  As a result, data format is idiosyncratic to each user, making data sharing dependent on manual annotation and metadata curation from investigators.

- **Visualization & GUI** - The GUIs of each of these systems are highly technical, and are not designed to be easily used by non-programmers, though pyControl's documentation offsets much of this difficulty. Visualization of task progress is quite rigid in both systems, either a timeseries of task states or plots specific to two-alternative forced choice tasks. In the examples we have seen, adapting plots to specific tasks is mostly ad-hoc use of external tools.

- **Documentation** - Writing good documentation is challenging, but particularly for infrastructural systems where a user is likely to need to modify it to suit their needs it is important that it be possible to understand its lower-level workings. PyControl has relatively good user documentation for how to use the system, but no API-level documentation. Bpod's documentation is a bit more scattered, and though it does have documentation for a subset of its functions, there is little indication of how they work together or how someone might be able to modify them.

- **Reproducibility** - As of November 2020, pyControl has versioned task files that append a hash to each version of a task and save it along with any produced data, tying the data to exactly the code that produced it. PyControl's most recent releases have explicit version numbers, but these don't appear to be saved along with the data. Bpod stores neither code nor task versions in its data. Neither

```
for currentTrial = 1:MaxTrials
% new state machine every trial
sma = NewStateMachine();

% add states and transitions
sma = AddState(sma,
    'Name', 'Wait', ...
    'Timer', 0, ...
    'StateChangeConditions', ...
    {'Port2In', 'Delay'}, ...
    'OutputActions', ...
    {'AudioPlayer1','*'});
% add more states ...

% upload and run task
SendStateMatrix(sma);
RawEvents = RunStateMatrix;

% manually gather data and params
BpodSystem.Data = AddTrialEvents(
    BpodSystem.Data, RawEvents);

% plotting in the main loop
UpdateSideOutcomePlot( ... );
UpdateTotalRewardDisplay( ... );

% manually save data
SaveBpodSessionData;
end
```

Figure 3.3: Bpod's general task structure.

system saves experimental parameter changes by default —and the GUIs of both allow parameters to be changed at will— and so critical data could be lost and experiments made unreproducible unless the user writes custom code to save them. Bpod has an undocumented plugin system, but neither system has a formal system for sharing plugins or task code, requiring work to be duplicated across all users of the system.

- **Integration and Extension** - Integration with other systems that might handle some out-of-scope function is tricky in both of these example systems. All systems have some limitation, so care must be taken to provide points by which other systems might interact with them. One particularly potent example is the use of Bpod in the International Brain Laboratory's standardized experimental rig[177], which relies on a single-purpose 93 page PDF to describe how to use the iblrig library, which consists of a large amount of single-purpose code for stitching together pybpod with bonsai for controlling video acquisition. Even if a system takes a large amount of additional work to integrate with another, hopefully the system allows it to be done in a way such that it can be reused and shared with others in the future so they can be spared the trouble. The relatively sparse documentation and the high proportion of ibl-specific code present in the repository make that seem unlikely.

Some of these limitations are cosmetic—fixable with additional code or hardware—but several of the most crucial are intrinsic to the design of these systems.

These systems, among others, have pioneered the development of modern behavioral hardware and software, and are to be commended for being open-source and highly functional. One need look no further for evidence of their usefulness than to their adoption by many labs worldwide. At the time that these systems were developed, a general-purpose single-board computer with performance like the Raspberry Pi 4 was not widely available. The above two systems are not unique in their limitations[7], but are reflective of broader constraints of developing experimental tools: solving these problems is *hard*. We are only able to articulate the design principles that differentiate Autopilot by building on their work.

[7] And Autopilot, of course, also has many of its own weaknesses

4

Design

Autopilot distributes experiments across a network of Raspberry Pis,[1] a type of inexpensive single-board computer.

[1] Raspberry Pi model 4B, see Table 5.2

Autopilot has three primary design principles:

1. **Efficiency** - Autopilot should minimize computational overhead and maximize use of hardware resources.

2. **Flexibility** - Autopilot should be transparent in all its operations so that users can expand it to fit their existing or desired use-cases. Autopilot should provide clear points of modification and expansion to reduce local duplication of labor to compensate for its limitations.

3. **Reproducibility** - Autopilot should maximize system transparency and minimize the potential for the black-box of local reprogramming. Autopilot should maximize the information it stores about its operation as part of normal data collection.

4.1 Efficiency

Though it is a single board, the Raspberry Pi operates more like a computer than a microcontroller. It most commonly runs a custom Linux distribution, Raspbian, allowing Autopilot to use Python across the whole system. Using an interpreted language like Python running on Linux has inherent performance drawbacks compared to compiled languages running on embedded microprocessors. In practice these drawbacks are less profound than they appear on paper: Python's overhead is negligible on modern processors[2], jitter and performance can be improved by wrapping compiled code, etc. While we view the gain in accessibility and extensibility of a widely used high-level language like Python as outweighing potential performance gains from using a compiled language, Autopilot is nevertheless designed to maximize computational efficiency.

[2] and improvements to CPython in Python 3.11 and onwards will bring overhead close to zero[178]

4.1.1 Concurrency

Most behavioral software is single-threaded (Figure 4.1), meaning the program will only perform a single operation at a time. If the program is busy or waiting for an input, other operations are blocked until it is finished.

Autopilot distributes computation across multiple processes and threads to take advantage of the Raspberry Pi's four CPU cores. Most operations in Autopilot are

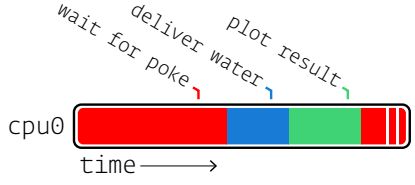


Figure 4.1: A single-threaded program executes all operations sequentially, using a single process and cpu core.

executed in **threads**. Specifically, Autopilot spawns separate threads to process messages and events, an architecture described more fully in section 5.8. Threading does not offer true concurrency[3], but does allow Python to distribute computational time between operations so that, for example, waiting for an event does not block the rest of the program, and events are not missed because the program is busy (Figure 4.2).

Critical operations that are computationally intensive or cannot be interrupted are given their own dedicated **processes**. Linux allows individual cores of a processor to be reserved for single processes, so individual Raspberry Pis are capable of running four truly parallel processing streams. For example, all Raspberry Pis in an Autopilot swarm create a messaging client to handle communication between devices which runs on its own processor core so no messages are missed. Similarly, if an experiment requires sound delivery, a realtime sound engine in a separate process (Figure 4.3) also runs on its own core.

Since even moderately complex experiments can consume more resources than are available on a single processor, the topmost layer of concurrency in Autopilot is to use additional **computers**. Autopilot uses the Raspberry Pi as a low-cost hardware controller, but only its GPIO control system is unique to them: the rest of the code can be used on any type of computer, so computationally expensive or GPU-intensive operations can be offloaded to any number of high performance machines. Computers divide labor *autonomously* (see 4.2.4 and 5.7), so for example one computer running a task can send and receive messages from another running the GUI and plots, but does not *depend* on that input as it would in a system that couples a microcontroller with a managing computer. The ability to coordinate multiple, autonomous computers with heterogeneous responsibilities and capabilities in a shared task is Autopilot's definitive design decision.

[3] See David Beazley's 'Understanding the Global Interpreter Lock' and associated visualizations.

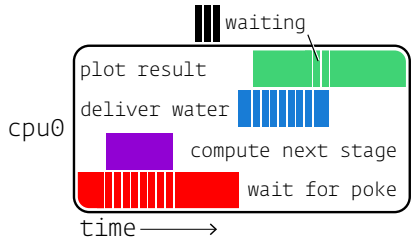


Figure 4.2: A multi-threaded program divides computation time of a single process and cpu core across multiple operations so that, for example, waiting for input doesn't block other operations.

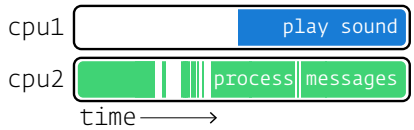


Figure 4.3: A multi-process program is truly concurrent, allowing multiple cpu cores to operate in parallel.

4.1.2 Leveraging Low-Level Libraries

Autopilot uses Python as a "glue" language, where it wraps and coordinates faster low-level compiled code[179]. Performance-critical components of Autopilot are thin wrappers around fast C libraries (Table 4.1). As Autopilot's API matures, we intend to replace any performance-limiting Python code like its sound server and networking operations with compiled code exposed to python with tools like the C Foreign Function Interface (CFFI).

Table 4.1: A few libraries Autopilot uses

jack	realtime audio
pigpio	GPIO control
ZeroMQ	networking
Qt	GUI

Since Autopilot coordinates its low-level components in parallel rather putting everything inside one "main loop," Autopilot actually has *better* temporal resolution than single-threaded systems like Bpod or pyControl, despite the realtime nature of their dedicated processors (Table 4.2).

Table 4.2: Using pigpio as a dedicated I/O process gives autopilot greater measurement precision

	Precision
Autopilot (pigpio)	$5\mu s$
Bpod	$100\mu s$
pyControl	$1000\mu s$

4.1.3 Caching

Finite-state machines are only aware of the current state and the events that transition it to future states. They are thus incapable of exploiting the often predictable structure of behavioral tasks to precompute future states and precache stimuli. Further, to change task parameters between trials (eg. changing the rewarded side in a two-alternative forced-choice task), state machines need to be fully reconstructed

and reuploaded to the device that runs them each time.

Autopilot precomputes and caches as much as possible. Rather than wait "inside" a state, Autopilot prepares each of the next possible events and saves them for immediate execution when the appropriate trigger is received. Static stimuli are prepared once at the beginning of a behavioral session and stored in memory. Before their presentation, they are buffered to minimize latency.

By providing full low-level documentation, we let researchers choose the balance between ease of use and performance themselves: it's possible to just call a sound's `play()` method, explicitly buffer it with its `buffer()` method, or generate samples on the fly with its `play_continuous()` method. Similarly, messages can be sent with a networking node's `send()` method, or prepared beforehand by explicitly making a `Message` and calling its `serialize()` method.

Autopilot's efficient design lets it access the best of both worlds—the speed and responsiveness of compiled code on dedicated microprocessors and the accessibility and flexibility of interpreted code.

4.2 Flexibility

4.2.1 Single-language

Behavior software that uses dedicated microprocessors must have some routine for compiling the high-level abstraction of the experiment into machine code. This gives those systems a theoretical advantage in processing speed, but the compiler becomes the bottleneck of complexity: only those things that can be compiled can be included in the experiment. This may in part contribute to the ubiquity of state-machine formalisms in behavior software.

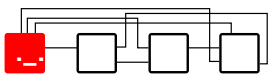
Because Python is used throughout the system, extending Autopilot's functionality is straightforward. Task design (see section 5.3) is effectively arbitrary—anything that can be expressed in Python is a valid task. This also allows Autopilot to easily be extended to make use of external libraries (eg. our integration with DeepLabCut-Live[180] and our planned integration with OpenEphys).

4.2.2 Modularity

Although Autopilot deeply integrates with the Raspberry Pi's hardware, we have also worked to make its components modular. There is a tension between providing a full-featured behavioral system and the flexibility of its components — as additional features are added to a system, they can constrain the functionality of existing components that they rely on. To address this tension, we have continuously worked to decouple Autopilot into subcomponents with clear inheritance hierarchies and APIs that can used quasi-independently.

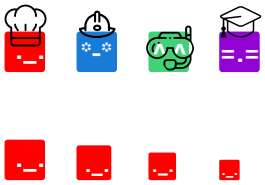
Modularity has 3 primary advantages:

1. **Modularity makes code more flexible** by reducing the constraints imposed by unstructured code interdependencies



2. **Modularity makes code more intelligible** by logically distributing tasks to discrete classes

3. **Modularity reduces effort-duplication** by allowing multiple, similar classes to be created with inheritance rather than copying and pasting.



There is no such thing as "incompatible hardware" with Autopilot because the classes that control hardware are independent from the code that provides other core functionality. In systems without modular design, hardware implementation is spread across the codebase. For example to add a new type of hardware output to a Bpod system, one would need to write new firmware for it in C (eg. the valve driver module), modify Bpod's existing firmware, hunt through the code to modify how states are added and state machines are assembled, add its controls explicitly to the GUI, and so on.

Tasks specify what type of hardware is needed to run them, but are agnostic about the way the hardware is implemented, making their descriptions more portable. Tasks that have the same structure but differ in hardware (eg. a freely moving two-alternative forced choice task in which a mouse visits several IR sensors, or a head-fixed two-alternative forced choice task in which a mouse runs on a wheel to indicate its choice) can be implemented by a trivial subclass that modifies the hardware description rather than completely rewriting the task.

4.2.3 Plugins & Code Transparency

We call Autopilot a software framework because in addition to providing classes and methods to run experiments out of the box, it also provides explicit structure that scaffolds any additional code that is needed by the user. Our goal is to clearly articulate in the documentation how modules should interact so that anyone can write code that works on any apparatus.

As groupware intended to be used differently by lab members with different responsibilities, Autopilot is designed for users with a range of programming expertise, from those who only want to interact with a GUI, to those who wish to fundamentally rewrite core operations for their particular experiment. As such, it is extensively documented: this paper provides a high-level introduction to its design and structure, its user guide describes how to use the program and provides examples, and its API-level documentation describes in granular detail how the program actually works[4]. Nothing is "off-limits" to the user—there isn't any hidden, undocumented hardware code behind the curtain[5]. We want users to be able to understand how and why everything works the way it does so that Autopilot can be adapted and expanded to any use-case.

A broader goal of Autopilot is to build a library of flexible task prototypes that can be tweaked and adapted, hopefully reducing the number of times the wheel is reinvented. We have attempted to nudge users to write reusable tasks by designing Autopilot such that rather than writing tasks as local unstructured scripts, they use its plugin system that scaffolds development by extending any of its basic types. Plugins are registered using a form in the Autopilot Wiki which makes them available to anyone while also embedding them in a semantically annotated information system that allows giving explicit credit to contributors, programmatically linking to any

[4] The user guide and API documentation are available at `docs.auto-pi-lot.com`

[5] For readability of the docs, we omit generating HTML documentation for some private methods and functions, but they are documented in the source and their function is made clear from their context and the documentation of public methods.

derivative publications that use the plugin, and further documentation of any tasks, hardware, or other extensions included within the plugin. Inheriting from parent classes give plugins structure and a set of basic features[6] while also being maximally permissive — anything can be overridden and modified.

[6] Like inheriting from the `GPIO` class gives GPIO plugins a systematic means of interacting with the underlying pigpiod daemon.

4.2.4 *Message Handling*

Modular software needs a well-defined protocol to communicate between modules, and Autopilot's is heavily influenced by the concurrency philosophy[7] of ZeroMQ[181]. All communication between computers and modules happens with ZeroMQ messages, and handling those messages is the main way that Autopilot handles events. A key design principle is that Autopilot components should not "share state"—they can communicate, but they are not *dependent* on one another. While this may seem like a trivial detail, having networking and message-handling at its core has three advantages that make Autopilot a fundamental departure from previous behavioral software.

[7] "ZeroMQ [...] has a subversive effect on how you develop network-capable applications. [...] message processing rapidly becomes the central loop, and your application soon breaks down into a set of message processing tasks."

"If there's one lesson we've learned from 30+ years of concurrent programming, it is: *just don't share state.*"

-The ZeroMQ Guide

First, new software modules can be added to any system by simply dropping in a standalone networking object. There is no need to dramatically reorganize existing code to make room for new functionality. Instead new modules can receive, process, and send information by just connecting to another module in the swarm. For example, each plot opens a network connection to stream incoming task data independently from the stream that is saving the data.

Second, Autopilot can be made to interact with other software libraries that use ZeroMQ. For example, The OpenEphys GUI for electrophysiology can send and receive ZMQ messages to execute actions such as starting or stopping recordings. Interaction with other software is also useful in the case that some expensive computation needs to happen mid-task. For example, one could send frames captured from a video camera on a Raspberry Pi to a GPU computing cluster for tracking the position of the animal. Since ZeroMQ messages are just TCP packets it is also possible to communicate over the internet for remote control or to communicate with a data server.

Third, making every component network-capable allows tasks to be distributed over multiple Raspberry Pis. Chaining multiple Pis distributes the computational load, allowing, for example, one Raspberry Pi to record and process video while another runs a sound server and delivers rewards. Autopilot expands with the complexity of your task, simultaneously eliminating limitations on quantity of hardware peripherals while ensuring latency is minimal. More interestingly, distributing tasks allows the arbitrary construction of what we call "behavioral topologies," which we describe in section 5.7.1.

4.3 *Reproducibility*

We take a broad view on reproducibility: including not only the ability to share data and recreate experiments, but also integrating into a broader ecosystem of tools that reduces labor duplication and encourages sharing and organizing technical knowledge. For us, reproducibility means building a set of tools that make every experiment and every technique available to anyone, anywhere.

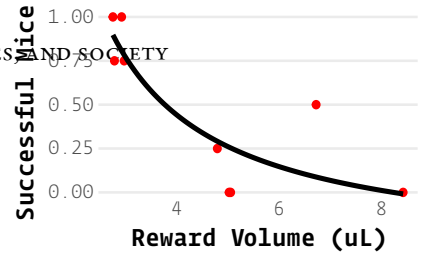


Figure 4.4: "Minor" details have major effects. Proportion of mice (each point, n=4) that were successful learning the first stage of the speech task described in [4] across 10 behavior boxes with variable reward sizes. A $2\mu L$ difference in reward size had a surprisingly large effect on success rate.

4.3.1 *Standardized task descriptions*

The implementation and fine details of a behavioral experiment matter. Seemingly trivial details like milliseconds of delay between trial phases and microliters of reward volume can be the difference between a successful and unsuccessful task (Figure 4.4). *Reporting* those details can thus be the difference between a reproducible and unreproducible result. Researchers also often use "auxiliary" logic in tasks—such as methods for correcting response bias—that are never completely neutral for the interpretation of results. These too can be easily omitted due to brevity or memory in plain-English descriptions of a task, such as those found in Methods sections. Even if all details of an experiment were faithfully reported, the balkanization of behavioral software into systems peculiar to each lab (or even to individuals within a lab) makes actually performing a replication of a behavior result expensive and technically challenging. Widespread use of experimental tools that are not explicitly designed to preserve every detail of their operation presents a formidable barrier to rigorous and reproducible science[163].

Autopilot splits experiments into a) the **code** that runs the experiment, which is intended to be standardized and shared across implementations, and b) the **parameters** (Figure 4.5) that define your particular experiment and system configuration. For example, two-alternative forced choice tasks have a shared structure regardless of the stimulus modality, but only your task plays pitch-shifted national anthems. This division of labor, combined with Autopilot's structured plugin system, help avoid the ubiquitous problem of rig-specific code and hard-coded variables making experimental code only useful on the single rig it was designed for — enabling the possibility of a shared library of tasks as described in section 4.2.3

The practice of reporting exactly the parameter description used by the software to run the experiment removes any chance for incompleteness in reporting. Because all task parameters are included in the produced data files, tasks are fully portable and can be reimplemented exactly by anyone that has comparable hardware to yours.

```
{
"step_name"    : "tone_discrim",
"task_type"    : "2AFC",
"bias_mode"    : 0,
"punish_sound" : false,
"stim" : {
  "sounds" : {
    "L": {
      "duration"  : 100,
      "frequency" : 10000,
      "type"      : "tone",
      "amplitude" : 0.01},
    "R": {"...":"..."}}},
"reward": {
  "type"     : "volume",
  "volume"   : 20},
"graduation" : {
    "type"      : "accuracy",
    "threshold" : 0.75,
    "window"    : 400},
}
```

Figure 4.5: Task parameters are stored as portable JSON, formatting has been abbreviated for clarity.

4.3.2 *Self-Documenting Data*

A major goal of the open science movement is to normalize publishing well documented and clearly formatted data alongside every paper. Typically, data are acquired and stored in formats that are lab-idiosyncratic or ad-hoc, which, over time, sprout entire software libraries needed just to clean and analyze it. Idiosyncratic data formats hinder collaboration within and between labs as the same cleaning and analysis operations gain multiple, mututally incompatible implementations, duplicating labor and multiplying opportunities for difficult to diagnose bugs. Over time these data formats and their associated analysis libraries can mutate and become incompatible with prior versions, rendering years of work inaccessible or uninterpretable. In one worst-case scenario, the cleaning process unearths some critically missing information about the experiment, requiring awkward caveats in the Methods section or months of extra work redoing it. In another, the missing information or bugs in analysis code are never discovered, polluting scientific literature with inaccuracies.

The best way to make data publishable is to avoid cleaning data altogether and *design good data hygiene practices into the data acquisition process.* Autopilot automatically

```
root
├── current_task
├── data
│   └── task_name
│       ├── S00_free_water
│       │   └── trial_data
│       └── S01_tone_discrim
│           └── trial_data
├── history
│   ├── git_hashes
│   ├── parameter_history
│   ├── past_protocols
│   │   └── old_task
│   └── weights
└── info
    ├── animal_id
    ├── birth_date
    ├── genotype
    └── etc_additional_info
```

Figure 4.6: Example data structure. All information necessary to reconstruct an experiment is automatically stored in a

stores all the information required to fully reconstruct an experiment, including any changes in task parameters or code version that happen throughout training as the task is refined.

Autopilot data is stored in HDF5 files, a hierarchical, high-performance file format. HDF5 files support metadata throughout the file hierarchy, allowing annotations to natively accompany data. Because HDF5 files can store nearly all commonly used data types, data from all collection modalities—trialwise behavioral data, continuous electrophysiological data, imaging data, etc.—can be stored together from the time of its acquisition. Data is always stored with the full conditions of its collection, and is ready to analyze and publish immediately (Figure 4.6). No Autopilot-specific scripts are needed to import data into your analysis tool of choice—anything that can read HDF5 files can read Autopilot data[8].

[8] Though our `Subject` class provides a simplified interface to access and manipulate Autopilot data

As of v0.5.0, we have built a formal data modeling system into Autopilot, allowing for unified declaration of data for experimental subjects, task parameters, and resulting data with verifiable typing and human-readable annotations. These abstract data models can be used with multiple storage interfaces, paving the way for export to, for example, the Neurodata Without Borders standard[182], further enabling Autopilot data to be immediately incorporated into existing processing pipelines (see section 5.2).

4.3.3 *Testing & Continuous Integration*

Open-source scientific software does away with prior limitations to access and inspection imposed by proprietary tools. It also exposes the research process to bugs in software written by semi-amateurs that can yield errors in the resulting data, analysis, and interpretation[183, 184, 185, 186]. Autopilot tries to bring best practices in software development to experimental software, including a set of automated tests for continuous integration.

We are still formalizing our contribution process, and our tests are still far from achieving full coverage[9], but we currently require tests and documentation for all new code added to the library. Writing good tests is hard, and we are in the process of building a set of hardware simulators and test fixtures to ease contribution.

[9] Coverage statistics for Autopilot are available on coveralls.io at https://coveralls.io/github/auto-pi-lot/autopilot

```python
def test_set_gpio():
    """
    The `set` method of a Digital_Out
    object sets the pin state
    """
    pin = Digital_Out(pin=17)

    # Turn GPIO pin on
    pin.set(True)
    assert pin.state == True

    # Turn GPIO pin off
    pin.set(False)
    assert pin.state == False
```

Figure 4.7: A test like `test_set_gpio` is a provable statement about the functionality of a program, in this case that "the `Digital_Out.set()` method sets the state of a GPIO pin."

Tests are effectively provable statements about how a program functions (Figure 4.7), which are particularly important for a library that aspires to be baseline lab infrastructure like Autopilot. Tests make it possible to use and contribute to the library with confidence: all tests are run on every commit, making it possible to determine if some new contribution breaks existing code without manually reading and testing every line. As we work to complete our test coverage, we hope to provide researchers with a tool that they can trust and elevates the verifiability of scientific results at large.

4.3.4 *Expense*

Autopilot is an order of magnitude less expensive than comparable behavioral systems (Table 4.3). We think the expense of a system is important for two reasons: scientific equity and statistical power.

The distribution of scientific funding is highly skewed, with a large proportion of

research funding concentrated in relatively few labs[187]. Lower research costs benefit all scientists, but lower instrumentation costs directly increase the accessibility of state-of-the-art experiments to labs with less funding. Since well-funded labs also tend to be concentrated at a few (well-funded) institutions, lower research costs also broaden the base of scientists outside traditional research institutions that can stay at the cutting edge[188, 189, 190].

Neuroscience also stands to benefit from the lessons learned from the replication crisis in Psychology[192]. In neuroscience, underpowered experiments are the rule, rather than the exception[193]. Statistical power in neuroscience is arguably even worse than it appears, because large numbers of observations (eg. neural recordings) from a small number of animals are typically pooled, ignoring the nested structure of observations collected within individual animals. Increasing the number of cells recorded from a small number of animals dramatically increases the likelihood of Type I errors (Figure 4.8)—indeed, for values of within-animal correlation typical of neuroscientific data, high numbers of observations make Type I errors more likely than not[191]. For this reason, perhaps paradoxically, recent technical advances in multiphoton imaging and silicon-probe recordings will actually make statistical rigor in neuroscience *worse* if we don't use analyses that account for the multilevel structure of the data and correspondingly record from the increased number of animals that they require.

Although the expense of multi-photon imaging and high-density electrophysiology will always impose an experimental bottleneck, behavioral training time is often the greater determinant of study sample size. Typical behavioral experiments require daily training sessions often carried out over weeks and months, while far fewer imaging or electrophysiology sessions are carried out per animal. Training large cohorts of animals in parallel is thus the necessary basis of a well-powered imaging or electrophysiology experiment.

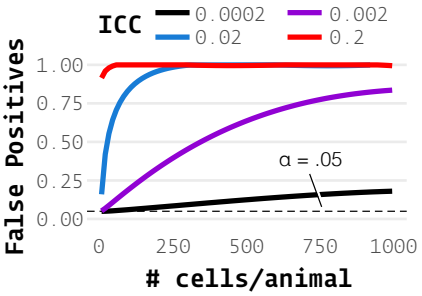


Figure 4.8: When comparing a value across groups, eg. a genetic knockout vs. wildtype, even a modest intra-animal (or, more generally, intra-cluster) correlation (ICC) causes the false positive rate to be far above the nominal $\alpha = 0.05$. Shown are false positive rates for simulated data with various numbers of "cells" recorded for comparisons between two groups of 5 animals each with a real effect size of 0. We note that 741 simultaneously recorded cells were reported in [156] and a mean ICC of 0.19 across 18 neuroscientific datasets was reported in [191]

	Autopilot	pyControl	Bpod
Behavior CPU	$45	$270	$925
Nosepoke (3x)	$216	$369	$810
Total for One	**$261**	**$639**	**$1735**
Five Systems	$1305	$3195	$8675
Host CPU(s)	$1000	$1000	$5000
Total for Five	**$2305**	**$4195**	**$13625**
Total for Ten	**$3610**	**$8390**	**$27350**

Table 4.3: Cost for Basic 2AFC System
"Nosepoke" includes a solenoid valve, IR sensor, water tube, LED, housing, and any necessary driver PCBs. For PyControl and Autopilot, we included the cost of one Lee LHDA0531115H solenoid valve per nosepoke ($63.35). For PyControl, we estimated a typical USB hub with 5 ports to control 5 pyControl systems from one computer. We note that the Bpod and PyControl systems both include cost of assembly for the control CPUs and nosepokes, but also that Autopilot does not require assembly for its control CPU and its default nosepoke is a snap-together 3D printed part and PCB without surface mounted components that can be assembled by an amateur in roughly half an hour.

5

Program Structure

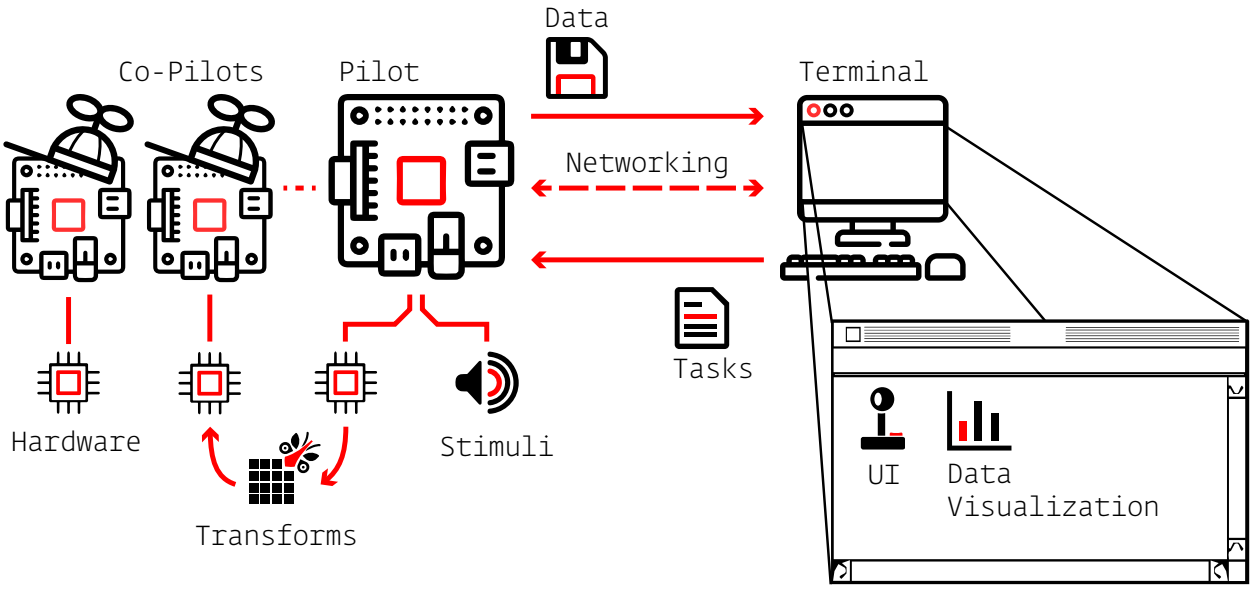


Figure 5.1: Overview of major Autopilot components

Autopilot consists of software and hardware modules configured to create a behavioral topology. Independent agents linked by flexible networking objects fill different roles within a topology, such as hosting the user interface, controlling hardware, transforming incoming and outgoing data, or delivering stimuli. This infrastructure is ultimately organized to perform a behavioral task.

5.1 Directory Structure

On setup, Autopilot creates a user directory that contains all local files that define its operation (Figure 5.2). The subdirectories include:

- **`calibration`** — Calibration for hardware objects like audio or solenoids that, for example, map opening durations to volumes of liquids dispensed
- **`data`** — Data for experimental subjects
- **`launch_autopilot.sh`** — Launch script that includes launching external processed like the jack audio daemon (will be removed and integrated into a more formal agent structure in future versions)
- **`logs`** — Every Autopilot object is capable of full debug logging, neatly formatted by object type and instance ID and grouped within module-level logging files. Logs are both written to disk, and output to `stderr` using the rich logging handler for clean and readable inspection during program operation (Figure 5.3).

```
./autopilot
├── calibration
├── data
│   ├── subject_1.h5
│   └── subject_2.h5
├── launch_autopilot.sh
├── logs
│   ├── core.terminal.log
│   └── plugins.my_plugin.log
├── pilot_db.json
├── plugins
│   └── my_plugin
│       └── my_task.py
├── prefs.json
├── protocols
│   ├── 2afc_easy.json
│   └── 2afc_hard.json
└── sounds
```

Figure 5.2: Example user directory structure, typically in ~/autopilot.

Logs can be parsed back into python objects to make it straightforward to diagnose problems or recover data in the case of an error.

```
[22-03-17T20:10:39] INFO     [core.terminal] parent, module-level logger created: core.terminal       loggers.py:159
                    INFO     [core.terminal.Terminal] Logger created: core.terminal.Terminal          loggers.py:163
                    INFO     [core.terminal.Terminal] successfully loaded pilot_db.json file from    terminal.py:437
                             /Users/jonny/autopilot/pilot_db.json
                    DEBUG    [core.terminal.Terminal] OrderedDict([('added_pilot',                   terminal.py:438
                                        OrderedDict([('subjects', ['myid']),
                                                     ('ip', ''),
                                                     ('prefs', OrderedDict())])),
                                       ('platform',
                                        OrderedDict([('subjects', ['test_nafc']),
                                                     ('ip', '192.168.0.101'),
                                                     ('prefs', OrderedDict())]))])
[22-03-17T20:10:39] INFO     [core.gui] parent, module-level logger created: core.gui                 loggers.py:159
                    INFO     [core.gui.Control_Panel] Logger created: core.gui.Control_Panel          loggers.py:163
[22-03-17T20:10:39] INFO     [core.plots] parent, module-level logger created: core.plots             loggers.py:159
                    INFO     [core.plots.Plot_Widget] Logger created: core.plots.Plot_Widget          loggers.py:163
                    INFO     [core.plots.Plot] Logger created: core.plots.Plot                        loggers.py:163
```

Figure 5.3: Logs printed to `stderr` are formatted and colorized by the rich logging handler. Logfiles are created by module, and log entries are identified by the individual objects instantiated from them. Logfiles are rotated and size-limited for configurable backups.

- **`pilot_db.json`** — A .json file that stores information about associated Pilots, including the contents of their prefs files, which hash/version of Autopilot they are running, and any Subjects that are associated with them.
- **`plugins`** — Plugins, which are any Python files that contain subclasses of Autopilot objects, that are automatically made available by Autopilot's registry system (eg. `autopilot.get('hardware', 'My_Hardware')` would retrieve a custom hardware object). Plugins can be documented and made available to other Autopilot users by registering them on the wiki
- **`prefs.json`** — Configuration options for this particular Autopilot instance, including configurations of local hardware objects, audio output, etc. In the future this will likely be broken into multiple files for different kinds of preferences[1].
- **`protocols`** — Protocols, which consist of parameterizations of individual Tasks as well as criteria for graduating betwewen them. These are also stored in individual subject data files, and updated whenever the source protocol files change.
- **`sounds`** — Any sound files that are requested by the File sound class.

[1] with care for backwards compatibility

5.2 Data

As of v0.5.0, Autopilot uses pydantic to create explicitly typed and schematized data models. Submodules include `data` abstract `modeling` tools that define base model types like `Tables`, `Groups`, and sets of `Attributes`. These base modeling classes are then built into a few core data models like subject `Biography` information, `Protocol` declaration, and the `Subject` data model itself that combines them. Modeling classes then have multiple `interfaces` that can be used to create equivalent objects in other formats, like pytables for hdf5 storage, pandas dataframes for analysis, or exported to Neurodata Without Borders.

For example, consider a simplified version of the `Biography` model:

Listing 1: **data - Biography**

```python
from autopilot.data.modeling import Data, Attributes, Field
from typing import Optional, Union
from datetime import datetime, timedelta

class Enclosure(Data):
    """Where does the subject work?"""
    box:  Optional[Union[str, int]] = Field(
        default=None,
        description="The box this Subject is run in")
    room: Optional[Union[str, int]] = Field(
        default=None,
        description="The room number that the animal is run in")

class Biography(Attributes):
    """Biography of an Experimental Subject"""
    id:  str = Field(...
        description="The indentifying name of this subject.")
    dob: datetime = Field(...
        description="The Subject's date of birth")
    enclosure: Optional[Enclosure] = None

    @property
    def age(self) -> timedelta:
        """Difference between now and :attr:`.dob`"""
        return datetime.now() - self.dob
```

Data models use builtin Python type hints. Type hints are colon delimited annotations like `x:int` that indicate the type (integer, string, etc.) of the variable. Though typically Python does not, Pydantic both validates that a type matches its hint and coerces it to the correct type if possible.

The `Union` type means that a field can be one of several possible types, in this case the box can be identified with either a string or integer.

Optional fields can have default fields, either a single value like **`None`** or a function that computes a default value like the current date.

Descriptions are stored in the data model schema to make shared data self-documenting, and also used by GUI widgets for tooltips that clarify what fields mean.

The class that our model inherits from indicates how Autopilot should treat it in a given storage interface. For HDF5 files, subclasses of the Attributes class are stored as node metadata, while subclasses of Table make tables.

Autopilot's format interfaces define mappings from nonstandard types to types supported by the format. For HDF5 files, datetime objects are converted to ISO 8601 formatted strings

Data models can be recursive, or use other models as types for their own fields. In this case the subject's Enclosure can be optionally specified in its Biography.

Properties are model fields that are automatically computed based on the values of other fields. The age of the animal can be accessed like a normal instance attribute that returns a timedelta object.

A new subject could then be created with a biography like this, storing it in the HDF5 file and made accessible through the Subject interface:

Listing 2: **data - New Subject**

```python
from autopilot.data import Subject
from autopilot.data.models import Biography, Enclosure
bio = Biography(
    id="my_subject",
    dob="2022-01-01T00:00:00",
    enclosure=Enclosure(box=100, room="Building 200")
)
sub = Subject.new(bio)
assert sub.info == bio
```

The `Subject` class is the primary means by which Autopilot stores, organizes, and interacts with data.

Several basic models are built into Autopilot, and in future versions it will be possible to extend and replace these models with plugins, making storage formats fully customizable while still being explicit and understandable.

If we don't give the type specified in the model, it will try and coerce it to the correct type and raise an error if it can't.

An **`assert`**ion declares that some logical statement is **`True`** and raises an exception if it isn't. Autopilot's unit tests ensure that subject data can be stored and retrieved without losing information or changing types.

The models are declared using a combination of python type hints and `Field` objects that provide defaults and descriptions. Because these models can be recursive, as in the case of using the `Enclosure` model as a type within the `Biography` model, we can build expressive, flexible, but still strict representations of complex data.

Out of the box, pydantic models can create explicit and interoperable schemas in JSON Schema and OpenAPI formats, and Autopilot extends them with additional interfaces and representations. Autopilot can create a GUI form for filling in fields for models, for example, to create a new Subject or declare parameters for a task (Figure 5.4). `Attribute` models that consist of scalar key-value pairs can be reliably stored and retrieved from metadata attribute sets in HDF5 groups, but Autopilot knows that `Table` models should be created as HDF5 tables as they will have multiple values for each field. An additional `Trial_Data` class that inherits from `Table` can be exported to NWB trial data, and the `Subject.get_trial_data` method uses the model to load trial data and convert it to a correctly typed pandas[194] DataFrame.

Though the data modeling system is new in v0.5.0[2], we have laid the groundwork for Autopilot's plugin system to allow researchers to declare custom schema for all data produced by Autopilot, and to preserve both interoperability and reproducibility by combining them with datasets potentially produced by multiple incompatible tools (see Section 7.4).

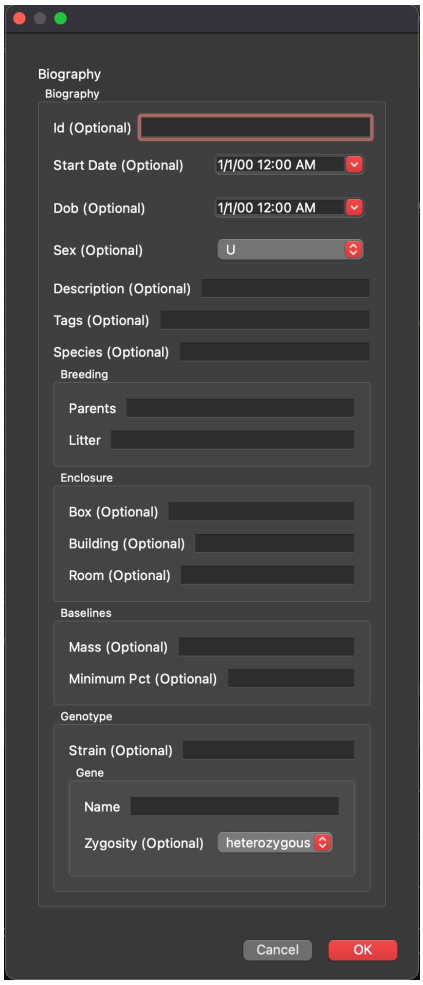


Figure 5.4: An Autopilot Data model can automatically generate a GUI form to fill in its properties, in this example to define a new experimental Subject's biography.

[2] Released as an alpha version at the time of writing

5.3 *Tasks*

Behavioral experiments in Autopilot are centered around **tasks**. Tasks are Python classes that describe the parameters, coordinate the hardware, and perform the logic of the experiment. Tasks may consist of one or multiple **stages** like a stimulus presentation or response event, completion of which constitutes a **trial** (Figure 5.5). Stages are analogous to states in the finite state machine formalism.

Multiple tasks are combined to make **protocols**, in which animals move between tasks according to "graduation" criteria like accuracy or number of trials. Training an animal to perform a task typically requires some period of shaping where they are familiarized to the apparatus and the structure of the task. For example, to teach animals about the availability of water from "nosepoke" sensors, we typically begin with a "free water" task that simply gives them water for poking their nose in them. Having a structured protocol system prevents shaping from relying on intuition or ad hoc criteria.

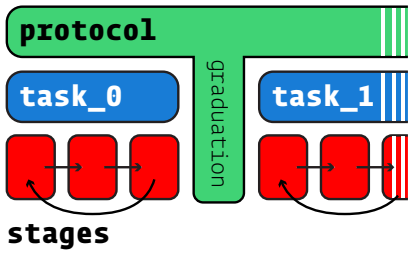


Figure 5.5: Protocols consist of one or multiple tasks, tasks consist of one or multiple stages. Completion of all of a task's stages constitutes a trial, and meeting some graduation criterion like accuracy progresses a subject between tasks.

5.3.1 *Task Components*

The following is a basic two-alternative choice (2AFC) task—a sound is played and an animal is rewarded for poking its nose in a designated target nosepoke. While simple, it is included here in full to show how one can program a task, including an explicit data and plotting structure, in roughly 60 lines of generously spaced Python.

Every task begins by describing four elements:

1) the task's parameters, 2) the data that will be collected, 3) how to plot the data, and 4) the hardware that is needed to run the task.

Listing 3: task - parameters

```
class Nafc(Task):
    class Params(Task_Params):
        stim: Sounds = Field(...,
            description = "Sound Stimuli")
        reward: units.mL = Field(...,
            description = "Reward Volume (mL)"
        )

    class TrialData(Trial_Data):
        target: Side = Field(...,
            description="Side (L, R) of the correct response")
        correct: bool = Field(...,
            description="Response matched target")

    PLOT = {}
    PLOT['data']   =  {'target'  : 'point',
                       'correct' : 'rollmean'},
    # n trials to roll window over
    PLOT['params'] = {'roll_window' : 50}

    HARDWARE = {
        'POKES':{
            'L': 'Digital_In',
            'R': 'Digital_In'
        },
        'PORTS':{
            'C': 'Solenoid',
        }
    }
```

1) A `Task_Params` model defines what parameters are needed to run the task.

We use `Field` objects as in listing 1, and can also use some special types like `Sounds` to declare complex parameters

`units` work like numbers but avoid ambiguity, so eg. the `Solenoid` class below knows this is a volume, rather than a duration

2) A `Trial_Data` model defines what data will be returned from the task.

3) A `PLOT` dictionary maps the data output to graphical elements in the GUI. (In future versions this will be incorporated into the `Fields` of `TrialData`)

4) A `HARDWARE` dictionary that describes what hardware will be needed to run the task.

The specific implementation of the hardware (eg. where it is connected, how to interact with it) is independent of the task. The task just knows about a `PORT` named `'C'` that is a `Solenoid`.

Created tasks receive some common methods, like input/trigger handling and networking, from an inherited metaclass. Python inheritance can also be used to make small alterations to existing tasks[3] rather than rewriting the whole thing. The GUI will use the `Params` model and the `PLOT` dictionary to generate forms for parameterizing the task within a protocol and display the data as it is collected. The `Subject` class will use the `TrialData` model to create HDF5 tables to store the data, and the `Task` metaclass will instantiate the described `HARDWARE` objects from their system-specific configuration in the `prefs.json` file so they are available in the rest of the class like `self.hardware['POKES']['L'].state`

[3] An example of subclassing a generic 'Task' class is included in Autopilot's user guide

5.3.2 Stage Methods

The logic of tasks is described in one or a series of methods (stages). The order of stages can be cyclical, as in this example, or can have arbitrary logic governing the transition between stages.

Listing 4: task - methods

```
30  def __init__(self, params:'Nafc.Params'):
31      self.stim_mgr = Stim_Manager(params.stim)
32      self.reward   = Reward_Manager(params.reward)
33
34      stage_list  = [self.discrim, self.reinforcement]
35      self.stages = itertools.cycle(stage_list)
36
37      self.init_hardware()
38      next(self.stages)()
39
40  def discrim(self):
41      target, wrong, stim = self.stim_mgr.next()
42      self.target = target
43
44      self.triggers[target] = [
45          self.hardware['PORTS']['C'].open,
46          lambda: next(self.stages)()]
47      self.triggers[wrong] = lambda: next(self.stages)()
48
49      self.node.send('DATA', {'target':target})
50
51      stim.play()
52
53  def reinforcement(self, response):
54      if response == self.target:
55          self.node.send('DATA', {'correct':True})
56      else:
57          self.node.send('DATA', {'correct':False})
58
59      next(self.stages)()
```

In Python, `def` defines new methods. The `__init__` method is called when a new object is initialized

Managers control stimulus and reward delivery, so users can, for example, continually synthesize new stimuli or implement adaptive rewards

Stages are combined into an object that (in this case) continually cycles through them when its `next()` method is called.

This starts the task by retrieving the first stage and then calling it.

The stimulus manager returns which port will be the target and the sound to be played.

A sequence of triggers is set: if the target port is poked, a reward will be delivered and the next stage will be called. A `lambda` function indicates not to call the method *now*, but only when triggered.

The task has a networking object that asynchronously streams data back to the user-facing terminal

In this example, the response port is passed from the trigger handling function. If it matches the stored target variable, the animal answered correctly.

Finally, the task is repeated by calling the next stage.

Autopilot is not prescriptive about how tasks are written. The same task could have two separate methods for correct and incorrect answers rather than a single reinforcement method, or only a single stage that blocks the program while it waits for a response.

Publishing data from this task requires no additional effort: a hash that uniquely identifies the code version (as well as any local changes) is automatically stored at the time of collection, as is a JSON-serialized version of the parameter model (Figure 5.6). If this task was incorporated into the central task library, anyone using Autopilot would be able to exactly replicate the experiment from the published data.

```
{
"step_name": "Simple 2AFC",
"stim" : {
  "sounds" : {
    "L": {
      "type"      : "tone",
      "frequency" : 4000},
    "R": {
      "type"      : "tone",
      "frequency" : 8000}
  }
},
"reward": 10
}
```

Figure 5.6: Simplified example of parameters for the above task

5.3.3 *The limitations of finite state machines*

The 2AFC task described above could be easily implemented in a finite-state machine. However, the difficulty of programming a finite-state machine is subject to combinatoric explosion with more complex tasks. Specifically, finite-state machines can't handle any task that requires any notion of "state history."

As an example, consider a maze-based task. In this task, the animal has to learn a particular route through a maze—it is not enough to reach the endpoint, but the animal has to follow a specific path to reach it (Figure 5.7). The arena is equipped with an actimeter that detects when the animal enters each area.

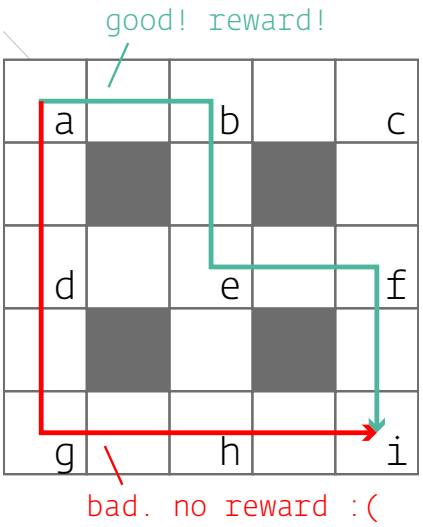


Figure 5.7: The subject must reach point `i` but only via the correct (green) path.

In Autopilot, we would define a hardware object that logs positions from the actimeter with a `store_position()` method. If the animal has entered the target position ("i" in this example), a `task_trigger()` that advances the task stage is called. The following code is incomplete, but illustrates the principle.

Listing 5: **maze - hardware**

```
1  class Actimeter(Hardware):
2      def __init__(self):
3          # ... some code to access the hardware ...
4          self.positions = []
5          self.target_position = "i"
6
7      def store_position(self, position):
8          self.positions.append(position)
9
10         if position == self.target_position:
11             self.finished_cb(self.positions)
12             self.positions = []
```

See line 18 below

The task follows, with parameters and network methods for sending data omitted for clarity.

Listing 6: **maze - task**

```
13 class Maze(Task):
14     def __init__(self):
15         self.target_path = ['a', 'b', 'e', 'f', 'i']
16
17         self.actimeter = Actimeter()
18         self.actimeter.finished_cb = self.finished
19
20     def finished(self, positions):
21         if positions == self.target_path:
22             self.reward()
```

The actimeter is given a reference to the Maze task's `finished()` method, which it calls when the target position is reached

The sequence of `positions` is compared to the `target_path` with ==. If they match, the subject is rewarded!

How would such a task be programmed in a finite-state machine formalism? Since the path matters, each "state" needs to consist of the current position and all the positions before it. But, since the animal can double back and have arbitrarily many state transitions before reaching the target corner, this task is impossible to represent with a finite-state machine, as a full representation would necessitate infinitely many states (this is one example of the *pumping lemma*, see [195]).

Even if we dramatically simplify the task by 1) assuming the animal never turns back and visits a space twice, and 2) only considering paths that are less than or equal to the length of the correct path, the finite state machine would be as complex as figure 5.8.

While finite-state machines are relatively easy to implement and work well for simple tasks, they quickly become an impediment to even moderately complex tasks. Even for 2AFC tasks, many desirable features are difficult to implement with a finite state machine, such as: (1) graduation to a more difficult task depending on performance history, (2) adjusting reward volume based on learning rate, (3) selecting or synthesizing upcoming stimuli based on patterns of errors[196], etc.

Some of these problems are avoidable by using extended versions of finite state machines that allow for extra-state logic, but require additional complexity in the code running the state machines to accomodate, and with enough exceptions the clean systematicity that is the primary benefit of finite state machines is lost. Autopilot attempts to avoid these problems by providing *tools* to program tasks and describe them without *requiring a specified format*, balancing the increased complexity by scaffolding the broader ecosystem of the experiment like its output data, hardware control, etc. When possible, we have tried to avoid forcing people to change the way they think about their work to fit our "little universe"[4] and instead try to provide a set of tools that let researchers decide how they want to use them.

[4] We take inspiration from Aaron Swartz' description of another engineering project, the Semantic Web, that became too precious about its formalisms:

"Instead of the "let's just build something that works" attitude that made the Web (and the Internet) such a roaring success [...] they formed committees to form working groups to write drafts of ontologies that carefully listed (in 100-page Word documents) all possible things in the universe and the various properties they could have, and they spent hours in Talmudic debates over whether a washing machine was a kitchen appliance or a household cleaning device. [...] And instead of spending time building things, they've convinced people interested in these ideas that the first thing we need to do is write *standards.* (To engineers, this is absurd from the start—standards are things you write *after* you've got something working, not before!)"[197]

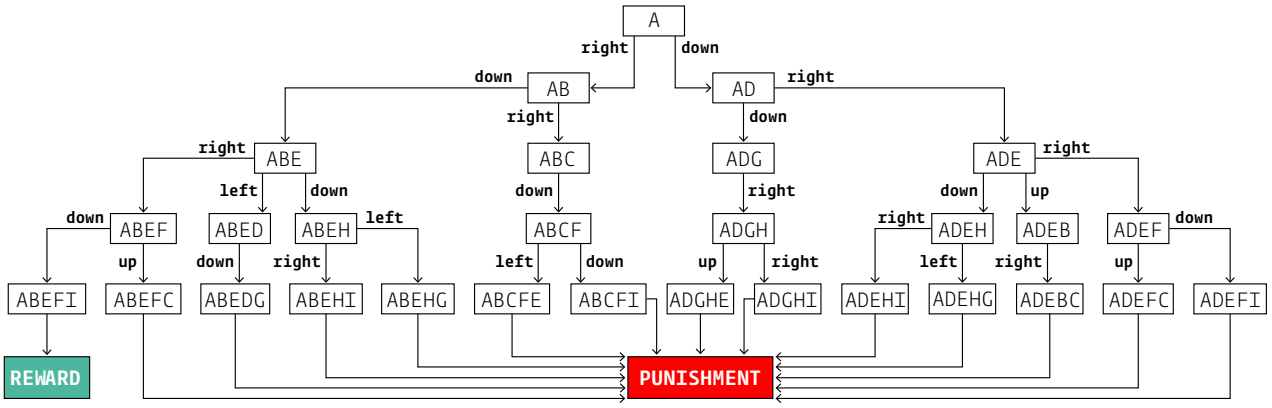


Figure 5.8: State transition tree for a simplified maze task.

5.4 *Hardware*

The Raspberry Pi can interface with nearly all common hardware, and has an extensive collection of guides, tutorials, and an active forum to support users implementing new hardware. There is also an enormous amount of existing hardware for the Raspberry Pi, including sound cards, motor controllers, sensor arrays, ADC/DACs, and touchscreen displays, largely eliminating the need for a separate ecosystem of purpose-built hardware (Table 5.1).

Autopilot controls hardware with an extensible inheritance hierarchy of Python classes intended to be built into a library of hardware controllers analogously to tasks. Autopilot uses pigpio to interact with its GPIO pins, giving Autopilot $5\mu s$ measurement precision and enabling protocols that require high precision (such as Serial, PWM, and I2C) for nearly all of the pins. Currently, Autopilot also has a family of objects to control cameras (both the Raspberry Pi Camera and high-speed GENICAM-compliant cameras), i2c-based motion and heat sensors, and USB mice. In the future we intend to improve performance further by replacing time-critical hardware operations with low-level interfaces written in Rust.

To organize and make available the vast amount of contextual knowledge needed to build and use experimental hardware, we have made a densely linked and publicly editable semantic wiki. The Autopilot wiki contains, among others, reference information for off-the-shelf parts, schematics for 2D and 3D-printable components, and guides for building experimental apparatuses and custom parts. The wiki combines unrestrictive freeform editing with structured, computer-readable semantic properties, and we have defined a collection of schemas for commonly documented items coupled with submission forms for ease of use. For example, the wiki page for the Lee Company solenoid we use has fields from a generic Part schema like a datasheet, price, and voltage, but also that it's a 3-way, normally-closed solenoid.

The wiki's blend of structure and freedom breaks apart typically monolithic hardware documentation into a collaborative, multimodal technical knowledge graph. Autopilot can access the wiki through its API, and we intend to tighten their integration over time, including automatic configurations for common parts, usage and longevity benchmarks, detecting mutually incompatible parts, and automatically re-

Table 5.1: Cost of common peripherals. The native hardware of the Raspberry Pi, low-level hardware control of Autopilot, and availability of inexpensive off-the-shelf components compatible with the raspi make most custom-built peripherals unnecessary.

Device	Raspi	Bpod
HiFiBerry DAC2 Pro	$45	$445
ADC	$30	$495
I2C	$0	$225
Ethernet	$0	$285
Rotary Encoder	$0	$145

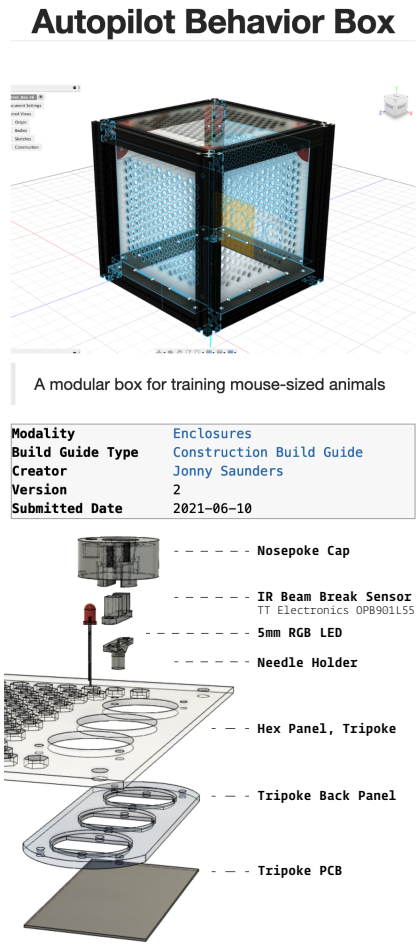


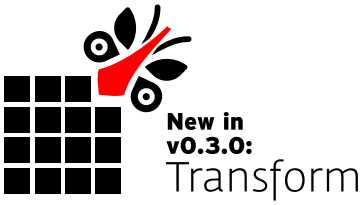
Figure 5.9: Two examples of parts with assembly guides available on the autopilot wiki: A modular behavior box with magnetic snap-in panels (top), and a three-nosepoke panel (bottom).

	Raspberry Pi 4B	Teensy 3.6	pyboard
CPU Clock	1.5GHz	180MHz	168MHz
CPU Cores	4	1	1
Architecture	ARMv8-A, 64-bit	ARMv7 32-bit	ARMv7 32-bit
RAM Size	2, 4, or 8GB	256KB	192KB
Storage	MicroSD (any size)	1024KB	1024KB
GPU	Broadcom VideoCore VI	—	—
GPIO Pins	40	58	29
USB Ports	2x USB 2.0, 2x USB 3.0	2x USB 2.0	1x USB 2.0
Ethernet	1Gbps	100Mbps	—
WiFi	2.4/5 GHz b/g/n/ac	—	—
Camera	15-pin Serial Interface	—	—
Bluetooth	✓	—	—

Table 5.2: Specifications of reviewed behavior hardware. BPod's state machine uses the Teensy 3.6 microcontroller, and PyControl uses the Micropython Pyboard.

solving any additional plugins or dependencies needed to use a part.

5.5 Transforms



In v0.3.0, we introduced the `transform` module, a collection of tools for transforming data. The raw data off a sensor is often not in itself useful for performing an experiment: we want to compare it to some threshold, extract positions of objects in a video feed, and so on. Transforms are like building blocks, each performing some simple operation with a standard object structure, and then composed into a pipeline (Figure 5.10). Pipelines are portable, and can be created on the fly from a JSON representation of their arguments, so it's easy to offload expensive operations to a more capable machine for distributed realtime experimental control (See [180]).

In addition to computing derived values, we use transforms in a few ways, including

- **Bridging Hardware** — Different hardware devices use different data types, units, and scales, so transforms can rescale and convert values to make them compatible.

- **Integrating External Tools** — The number of exciting analytical tools for realtime experiments keep growing, but in practice they can be hard to use together. The transform module gives a scaffolding for writing wrappers around other tools and exposing them to each other in a shared framework, as we did with DeepLabCut-Live[180], making closed-loop pose tracking available to the rest of Autopilot's ecosystem. We don't need to rally thousands of independent developers to agree to write their tools in a shared library, instead transforms make wrapping them easy.

- **Extending Objects** — Transforms can be used to augment existing objects and create new ones. For example, a motion sensor uses the spheroid transform to calibrate its accelerometer, and the gammatone filter[5] extends the Noise sound to make a gammatone filtered noise sound.

Transform - DLC Live!

```
from autopilot import transform as t
# track points on a human body
dlc = t.image.DLC(
    model_zoo="full_human")
# select one of the knees
dlc += t.selection.DLCSlice(
    select="knee2")
# Test if it's in an ROI
dlc += t.logical.Condition(
    minimum=(0,0),
    maximum=(128,128))

# Process frames with the pipeline
# set pin High if knee2 in ROI
while True:
    pin.set(dlc.process(
        cam.q.get()))
```

Figure 5.10: Transforms can be chained together (here with the in-place addition operator `+=`) to make pipelines that encapsulate the logical relationship between some input and a desired output. Here `pin` is a Digital_Out object, and `cam` is a PiCamera with `queue` enabled.

[5] a thin wrapper around scipy's `signal.gammatone`

Like Tasks and Hardware, the transform module provides a scaffolding for writing reference implementations of algorithms commonly needed for realtime behavioral experiments. For example, neuroscientists often want to quickly measure a research subject's velocity or orientation, which is possible with inexpensive inertial motion sensors (IMUs), but since anything worth measuring will be swinging the sensor around with wild abandon the readings first need to be rotated back to a geocentric coordinate frame. Since the readings from an accelerometer are noisy, we found a few whitepapers describing using a Kalman filter for fusing the accelerometer and gyroscope data for a more accurate orientation estimate ([198, 199]), but couldn't find an implementation. We wrote one and integrated it into the IMU class (Figure 5.11). Since it's an independent transform, it's available to anyone even if they use nothing else from Autopilot.

Transforms were made to be composed, so we broke it into independent sub-operations: A Kalman filter, rotation, and a spheroid correction to calibrate accelerometers. Then

Geocentric Velocity

```
# rotate input in x and y
# by some pitch and roll
reorient = t.geometry.Rotate(
    dims='xy')
# select the z axis
reorient += t.selection.Slice(
    select=2)
# remove gravity
reorient += t.math.Add(
    -9.8)

# using I2C_9DOF ...
angle = imu.rotation
z_accel = reorient.process(
    imu._acceleration, angle)
```

Figure 5.11: Using the IMU_Orientation transform built into the IMU's `rotation` property, a processing chain to reorient the accelerometer reading and subtract gravity for geocentric z-axis acceleration

we combined it with the DLC-Live transform for a fast but accurate motion estimate from position, velocity, and acceleration measurements from three independent sensors. Since each step of the transformation is exposed in a clean API, it was straightforward to extend the Kalman filter to accomodate the the wildly different sampling rates of the camera and IMU. It's still got its quirks, but that's the purpose of plugins — to make the code available and documented without formally integrating it in the library.

5.6 *Stimuli*

A hardware object would control a speaker, whereas stimulus objects are the individual sounds that the speaker would play. Like tasks and hardware, Autopilot makes stimulus generation portable between users, and is released with a family of common sounds like tones, noises, and sounds from files. The logic of sound presentation is contained in an inherited metaclass, so to program a new stimulus a user only needs to describe how to generate it from its parameters (Figure 5.12). Sound stimuli are better developed than visual stimuli as of v0.5.0, but we present a proof-of-concept visual experiment (Section 6.5) using psychopy[170].

An Autopilot Tone

```
my_tone = sounds.Tone(
    frequency = 500,
    duration  = 200)
my_tone.play()
```

Figure 5.12: Autopilot stimuli are parametrically defined and inherit all the playback logic that makes them easy to integrate in tasks

Autopilot controls the realtime audio server jack from an independent Python process that dumps samples directly into jack's buffer (Figure 5.13), giving it a trigger-to-playback latency very near the theoretical minimum (Section 6.2). Sounds can be pre-buffered in memory or synthesized on demand to play continuous sounds. Because the realtime server is independent from the logic of sound synthesis and storage, stimuli can be controlled independently from different threads without interrupting audio or dropping frames.

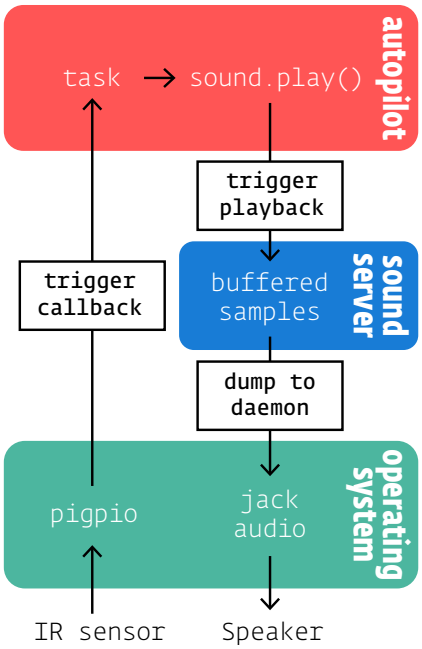


Figure 5.13: Our sound server keeps audio samples buffered until a `.play()` method is called, and then dumps them directly into the jack audio daemon.

We use the Hifiberry Amp 2, a combined sound card and amplifier, which is capable of 192kHz/24Bit audio playback. Autopilot and Jack can output to any sound hardware, however, including the builtin audio of the Raspberry Pi if fidelity isn't important. There are no external video cards for the Raspberry Pi 4b[6], but its embedded video card is capable of presenting video and visual stimuli (Section 6.5) especially if the other computationally demanding parts of the task are distributed to other Raspberry Pis (Section 5.7.1). If greater video performance is needed, Autopilot is capable of running on typical desktops as well as other single-board computers with GPUs (as we did with the Nvidia Jetson in [180]).

5.7 *Agents*

All of Autopilot's components can be organized into a single system as an "agent," the executable that coordinates everyday use. An agent encapsulates:

- **Runtime Logic** — an initialization routine that starts any needed system processes and any subsequent operations that define the behavior of the agent.
- **Networking Station** — Agents have networking objects called `Stations` that are intended to be the "load bearing" networking objects (described more below).
- **Callbacks** — An action vocabulary that maps different types of messages to methods for handling them. Called `listens` to disambiguate from other types of callbacks.
- **Dependencies** — Required packages, libraries, and system reconfigurations needed to operate. Python dependencies are currently defined for agents as groups of optional packages[7], and system configuration is done with scripts which shorthand common operations like compiling OpenCV with optimizations for the raspi or enabling a soundcard.

[6] though the Raspberry Pi compute module has a PCI lane that supports GPUs.

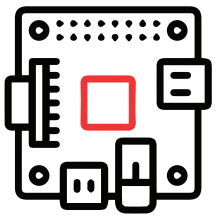
[7] As of v0.5.0, Autopilot is packaged with Poetry, so they are `[tool.poetry.extras]` entries within the `pyproject.toml` file, installed with pip like `pip install auto-pi-lot[pilot]` or poetry like `poetry install -E pilot`

Together, these define an agent's *role* in the swarm.

There are currently two agents in Autopilot:

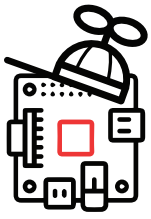
- **Terminal** - The user-facing control agent.
- **Pilot** - A Raspberry Pi that runs tasks, coordinates hardware, and optionally coordinates a set of child Pis.

Terminal agents serve as a root node (see Section 5.8) in an Autopilot swarm. The terminal is the only agent with a GUI, which is used to control its connected pilots and visualize incoming task data. The terminal also manages data and keeps a registry of all active experimental subjects. The terminal is intended to make the day-to-day use of an Autopilot swarm manageable, even for those without programming experience. The terminal GUI is described further in Section 5.9.



Pilot agents are the workhorses of Autopilot—the agents that run the experiments. Pilots are intended to operate as always-on, continuously running system services. Pilots make a network connection to a terminal and wait for further instructions. They maintain the system-level software used for interfacing with the hardware connected to the Raspberry Pi, receive and execute tasks, and continually return data to the terminal for storage.

Each agent runs autonomously, and so a Pilot can run a task without a Terminal and store data locally, a Terminal can be used without Pilots to define protocols and manage subjects, and so on. This decoupling lets each agent have more freedom in its behavior at the expense of the complexity of configuring and maintaining them (see Sec. 7.10 and 7.13). All interaction is based on the "listen" callbacks known by the agents, so to start a task a Terminal will send a Pilot a `"START"` message containing information about a `Task` class that it is to run along with its parameterization. The Pilot then attempts to run the task, sends a message to the Terminal alerting it to a `"STATE"` change, and begins streaming data back to it in messages with a `"DATA"` key.



Each pilot is capable of mutually coordinating with one or many **Copilots**[8]. We are still experimenting with, and thus openminded to the best way to structure multi-pilot tasks. Like many things in Autopilot, there is no one right way to do it, and the strategy depends on the particular constraints of the task. We include a few examples in the network latency and go/no-go tasks in the plugin that accompanies this paper, and expand on this a bit further in a few parts of section 7, as it is a major point of active development.

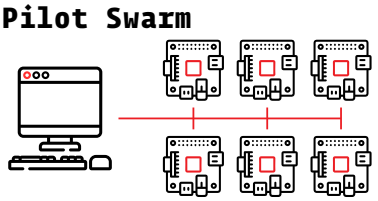
5.7.1 Behavioral topologies

We think one of the most transformative features of Autopilot's distributed structure is the control that users have over what we call "behavioral topology." The logic of hardware and task operation within an agent, the distribution of labor between agents performing a task, and the pattern of connectivity and command within a swarm of agents constitute a topology.

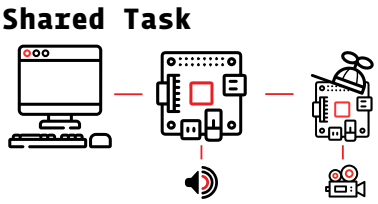
Below we illustrate this idea with a few examples:

[8] A previous version of this paper described a third, subordinate "Child" agent that performed auxiliary operations in a task. We now view such a hierarchy as unnecessary, and that distribution of labor within a task is better served by a fluid combination of multiple Pilots than thinking of them as qualitatively different agents. We now refer one among multiple agents performing a task together as a "copilot."

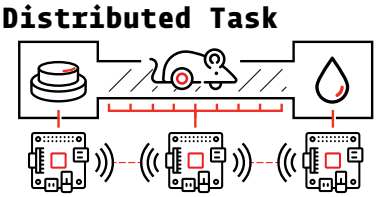
- **Pilot Swarm** - The first and most obvious topological departure from traditional behavioral instrumentation is the use of a single computer to independently coordinate tasks in parallel. Our primary installation of Autopilot is a cluster of 10 behavior boxes that can independently run tasks dispatched from a central terminal which manages data and visualization. This topology highlights the expandability of an Autopilot system: adding new pilots is inexpensive, and the single central terminal makes controlling experiments and managing data simple.



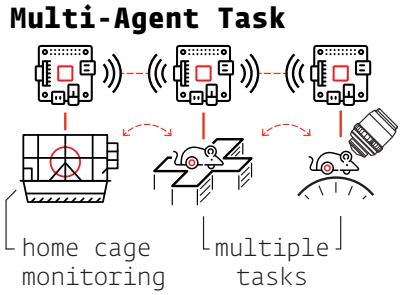
- **Shared Task** - Tasks can be shared across a set of copilots to handle tasks with computationally intensive operations. For example, in an open-field navigation task, one pilot can deliver position-dependent sounds while another records and analyzes video of the arena to track the animal's position. The terminal only needs to be configured to connect to the parent pilot, but the other copilot is free to send data to the Terminal marked for storage in the subject's file as well.



- **Distributed Task** - Many pilots with overlapping responsibilities can cooperate to perform distributed tasks. We anticipate this will be useful when the experimental arenas can't be fully contained (such as natural environments), or when experiments require simultaneous input and output from multiple subjects. Distributed tasks can take advantage of the Pi's wireless communication, enabling, for example, experiments that require many networked cameras to observe an area, or experiments that use the Pis themselves as an interface in a multisubject augmented reality experiment.



- **Multi-Agent Task** - Neuroscientific research often consists of multiple mutually interdependent experiments, each with radically different instrumentation. Autopilot provides a framework to unify these experiments by allowing users to rewrite core functionality of the program while maintaining integration between its components. For example, a neuroethologist could build a new "Observer" agent that continually monitors an animal's natural behavior in its home cage to calibrate a parameter in a task run by a pilot. If they wanted to manipulate the behavior, they could build a "Compute" agent that processes Calcium imaging data taken while the animal performs the task to generate and administer patterns of optogenetic stimulation. Accordingly, passively observed data can be combined with multiple experimental datasets from across the subject's lifespan. We think that unifying diverse experimental data streams with interoperable frameworks is the best way to perform experiments that measure natural behavior in the fullness of its complexity in order to understand the naturally behaving brain[175].



5.8 Networking

Agents use two types of object to communicate with one another: core **station** objects and peripheral **node** objects (Figure 5.14). Each agent creates one station in a separate process that handles all communication *between* agents. Stations are capable of forwarding data and maintaining agent state so the agent process is not unnecessarily interrupted. Nodes are created by individual modules run within an

agent—eg. tasks, plots, hardware—that allow them to send and receive messages within an agent, or make connections directly to other nodes on other agents after the station discovers their network addresses. Messages are TCP packets[9], so there is no distinction between sending messages within a computer, a local network, or over the internet[10].

Both types of networking objects are tailored to their hosts by a set of callback functions — **listens** — that define how to handle each type of message. Messages have a uniform key-value structure, where the key indicates the listen used to process the message and the value is the message payload. This system makes adding new network-enabled components trivial:

Listing 7: A new networked LED

```
class LED_RGB(Hardware):
    def __init__(self):
        # call self.color for a 'COLOR' message
        self.listens = {'COLOR': self.color}
        self.node = networking.Node(
            id      = 'BEST_LED',
            listens = self.listens)

    def color(msg):
        self.set_color(msg.value)

# elsewhere in the code, we change the color to red!
node.send(to='BEST_LED', key='COLOR', value=[255,0,0])
```

Messages are serialized[11] with JSON, and can handle arrays, including on-the-fly compression with blosc. Net Nodes can create additional sockets to stream data that is stashed in a queue, and can take advantage of message batching and compressing multiple arrays together when latency is less critical.

Network connectivity is currently treelike by default (Figure 5.15) — each independent networking object can have many children but at most one parent. This structure makes an implicit assumption about the anisotropy of information flow: 'higher' nodes don't need to send messages to the 'lowest' nodes, and the 'lowest' nodes send all their messages to one or a few 'higher' nodes. It enforces simplified delegation of responsibilities in both directions: a terminal shouldn't need to know about every hardware object connected to all of its connected pilots, it just sends messages to the pilots, who handle it from there. A far-downstream node shouldn't need to know exactly how to send its data back to the terminal, so it pushes it upstream until it reaches a node that does.

This treelike structure is useful for getting started quickly with the default full system configuration, but for experimenting with different configurations it is also possible to directly connect network nodes. The Station backbone is then a useful way of connecting objects across agents, as messages can be sent as multihop messages through a connected tree of stations[12] to make an initial connection without hard-coding an IP or Port. In the future we plan to simplify this further by directly implementing a peer to peer discovery model, see Section 7.6.

[9] Autopilot uses ZeroMQ[181] and tornado to send and process messages

[10] Though automatically configuring the use of faster protocols like IPC for communication within an agent or different backends like redis or gstreamer for data streams that would benefit from them is part of our development goals

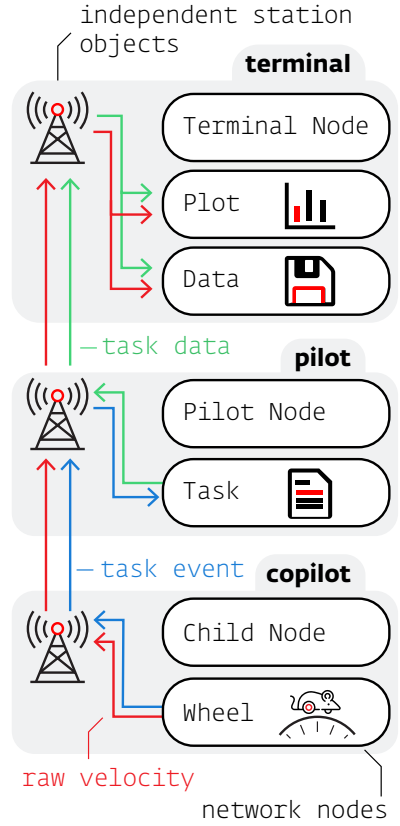


Figure 5.14: Autopilot segregates data streams efficiently—eg. raw velocity (red) can be plotted and saved by the terminal while only the task-relevant events (blue) are sent to the primary pilot. The pilot then sends trial-summarized data to the terminal (green).

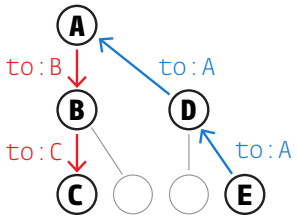


Figure 5.15: Treelike network structure—downstream messages are addressed by successive nodes, but upstream messages can always be pushed until the target is found.

[11] converted to binary suitable for sending between computers

[12] For example, to send a message from `E` to `C` in the diagram above:

```
node.send(to=["A", "B", "C"])
```

5.9 GUI & Plots

The terminal's GUI controls day-to-day system operation[13]. It is intended to be a nontechnical frontend that can be used by those without programming experience.

For each pilot, the terminal creates a control panel that manages subjects, task operation, and plots incoming data. Subjects can be managed through the GUI, including creation, protocol assignment, and metadata editing. Protocols can also be created from within the GUI. The GUI also has a set of basic maintenance and informational routines in its menus, like calibrating water ports or viewing a history of subject weights.

The simple callback design and network infrastructure makes adding new GUI functionality straightforward, and in the future we intend to extend the plugin system such that plugins can provide additional menu actions, plots, and utilities.

[13] Autopilot uses PySide, a wrapper around Qt, to build its GUI.

Trial Plot

```
{"data": {
    "target"   : "point",
    "response" : "segment",
    "correct"  : "rollmean"
},
"roll_window" : 50}
```

Continuous Plot

```
{"data": {
    "target"   : "point",
    "response" : "segment",
    "velocity" : "shaded"
},
"continuous": true}
```

Figure 5.16: `PLOT` parameters for Figure 5.17. In both, "target" and "response" data are mapped to "point" and "segment" graphical primitives, but timestamps rather than trial numbers are used for the x-axis in the "continuous" plot (Figure 5.17, bottom). Additional parameters can be specified, eg. the trial plot (Figure 5.17, top) computes rolling accuracy over the past 50 trials

5.9.1 Plotting

Realtime data visualization is critical for monitoring training progress and ensuring that the task is working correctly, but each task has different requirements for visualization. A task that has a subject continuously running on a ball might require a continuous readout of running velocity, whereas a trial-based task only needs to show correct/incorrect responses as they happen. Autopilot approaches this problem by assigning the data returned by the task to graphical primitives like points, lines, or shaded areas as specified in a task's `PLOT` dictionary (taking inspiration from Wilkinson's grammar of graphics[200]).

The GUI is now some of the oldest code in the library, and we are in the process of decoupling some of its functionality from its visual representation and moving to a model where it is a thinner wrapper around the data modeling tools. Following the lead of formal models with strict typing will, for example, make plotting more fluid where the researcher can map incoming data to the set of graphical elements that are appropriate for its type. We discuss this further in section 7.8

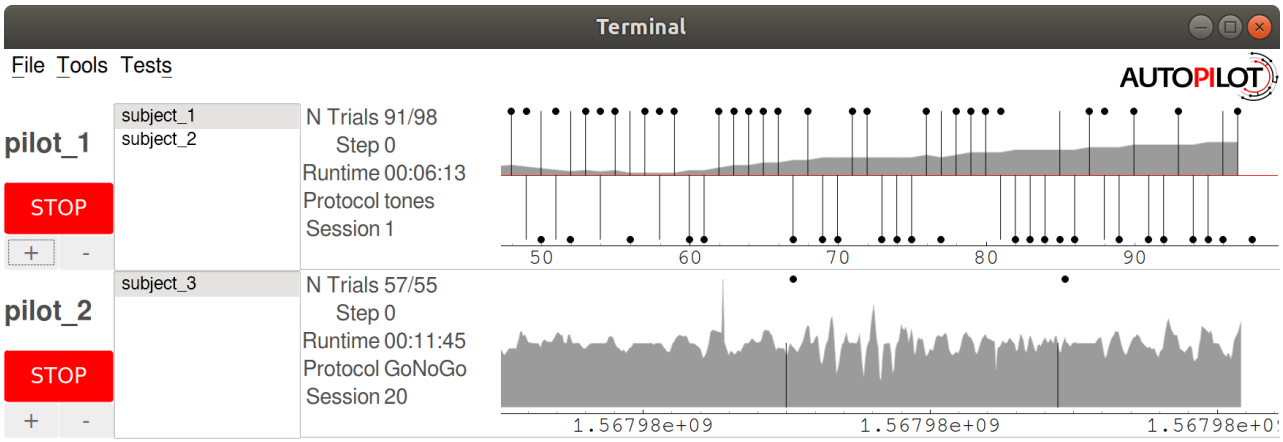


Figure 5.17: Screenshot from a terminal GUI running two different tasks with different plots concurrently. `pilot_1` runs 2 subjects: (`subject_1` and `subject_2`), while `pilot_2` runs `subject_3`. See Figure 5.16 for plot description

6

Tests

We have been testing and refining Autopilot since we built our swarm of 10 training boxes in the spring of 2019. In that time 178 mice[1] have performed over 6 million trials on a range of tasks. While Autopilot is still relatively new, it is by no means untested.

In this section we will present a set of basic performance benchmarks while also showing several of the different ways that Autopilot can be used. The code for all of the following tests is available as a plugin that is further documented on the wiki, and runs on a prerelease of v0.5.0. Materials tables (Table 6.1) for each test link more specifically to the test code and provide additional hardware and version documentation, where appropriate.

[1] All procedures were performed in accordance with National Institutes of Health guidelines, as approved by the University of Oregon Institutional Animal Care and Use Committee.

Table 6.1: General Materials

Hardware	
Raspi	Raspberry Pi 4b
Oscilloscope	Rigol DS1054Z
Software	
Autopilot	v0.5.0a
Plugin	Autopilot_Paper
Python	3.9.12
RaspiOS	Bullseye 22-04-04 (lite)
Analysis	
R	4.2.0
ggplot2[201]	3.3.5
dplyr[202]	1.0.9
purrr[203]	0.3.4
pandas[194]	1.4.2
numpy[173, 204]	1.21.6

6.1 GPIO Latency

Neurons compute at millisecond timescales, so any task that links neural computation to behavior needs to have near-millisecond latency. We start by characterizing Autopilot's GPIO control latency in "script mode" — using the GPIO control classes on their own, without using any of the rest of Autopilot's modules.

6.1.1 Output Latency

We first tested the software measured latency between when a command to write a value to a GPIO pin is issued and when it completes (Figure 6.1, Table 6.2). By default, the pigpio interface we use to control GPIO pins issues a command and then confirms the request was successful by querying the pigpio daemon for the status of the pin (Write+Read). We extended pigpio to just issue the command without confirmation to estimate the true time between when the command is issued and when the voltage of the pin changes (Write). Each of these operations takes roughly 40μs with minimal jitter (Median $\pm$ IQR — Write Only: 40.8$\mu s$$\pm$0.41, Write and Read: 88.4 $\mu s$$\pm$2.02, n=100,000 each).

Table 6.2: GPIO Latency Materials. (Parameters in {} are input in separate runs)

Code	test_gpio.py
replicate	`python test_gpio.py -w {0,1,2} -n 100000`
gpiozero	1.6.2
pigpio	3c23715

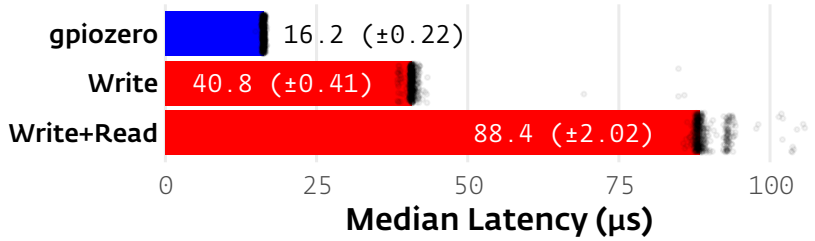


Figure 6.1: Software latency from GPIO write to completion of command. Values are presented as medians $\pm$ IQR with n=100,000 tests for each. A random subsample of 500 (for tractability of plotting) of each type of test are presented (black points) after filtering to the bottom 99th percentile to exclude extreme outliers. Commands sent using pigpio (red) took roughly 40μs each (write and read are effectively two separate commands, Write Only: 40.8$\mu s$$\pm$0.41, Write and Read: 88.4 $\mu s$$\pm$2.02). The prototype gpiozero wrapper (blue) using RPi.GPIO as its backend was faster, taking 16.2$\mu s$$\pm$0.22 to complete.

Pigpio is useful as a general purpose controller because of its ability to run scripts within its daemon, use hardware PWM via direct memory access, and consistently poll for pin state, but takes a latency penalty because the python interface communicates with it through a local TCP socket. To demonstrate the flexibility of Autopilot

in incorporating additional software libraries, we wrote a thin wrapper around gpiozero, which can use RPi.GPIO to directly write to the GPIO registers. For simple output, this wrapper proved to be faster (16.2 μs ± 0.22, n=100,000), and with 63 lines of code is now available in the plugin accompanying this paper to be used, repurposed, and extended.

6.1.2 Roundtrip Input/Output Latency

Output commands usually aren't issued in isolation, but as a response to some external or task-driven trigger. We measured the roundtrip latency from a 5V square pulse from an external function generator to when an output pin was flipped from low to high on an oscilloscope (Table 6.3).

Typically that is as much methodological detail as you would expect in a scientific paper, but actually making those measurements via oscilloscope requires knowing how to set up such a test as well as how to extract the measured data afterwards — which is not altogether trivially available technical knowledge. As an example of how integrating semantically linked documentation with experimental tools enables a fundamentally deeper kind of reproducibility and methodological transparency, we instead documented these operations, including a code sample and a guide to unlocking additional features on our oscilloscope on the autopilot wiki. The code to extract traces from the oscilloscope is also included in this paper's plugin, which links to the oscilloscope page with a `[[Controls Hardware::Rigol DS1054Z]]` tag, so it is possible to bidirectionally find code examples from the oscilloscope page as well as find further documentation about the hardware used in this paper from the plugin page. The same can be true for any hardware used by any plugin in any paper using Autopilot.

We used the `assign_cb` method of the `Digital_In` class to test the typical roundtrip latency that Autopilot objects can deliver. This gave us a median 474 μs (IQR: 52.5μs) latency (Red in Figure 6.3). GPIO callbacks are flexible, and can use arbitrary python functions, but if all that's needed is to trigger one pin off of another with some simple logic like a parametric digital waveform or static "on" time, pigpio also allows us to directly program pin to pin logic as a "pigs" script (literally Figure 6.2) that runs within the pigpio daemon. The pigs script gave us roughly three orders of magnitude lower latency (Median ± IQR: 370ns ± 140, blue in 6.3).

Table 6.3: Roundtrip Latency Materials

Function Generator	Koolertron CJDS98
Code	test_gpio.py
Replicate	`python test_gpio.py -w {3,4}`

Pigs Trigger Script

```
" ".join([
  "tag 999",
  # read input pin
  f"r {pin_in.pin_bcm}",
  # if off, goto 998
  f"jz 998",
  # else, turn on
  f"w {pin_out.pin_bcm} 1",
  # then goto 999
  "jp 999",
  "tag 998",
  # turn off
  f"w {pin_out.pin_bcm} 0",
  # jump to beginning
  f"jp 999"
])
```

Figure 6.2: The pigs script used to trigger one pin (pin_out), from another (pin_in). At the expense of a little bit of complexity having to write a script in its scripting language, we are able to reduce latency by three orders of magnitude.

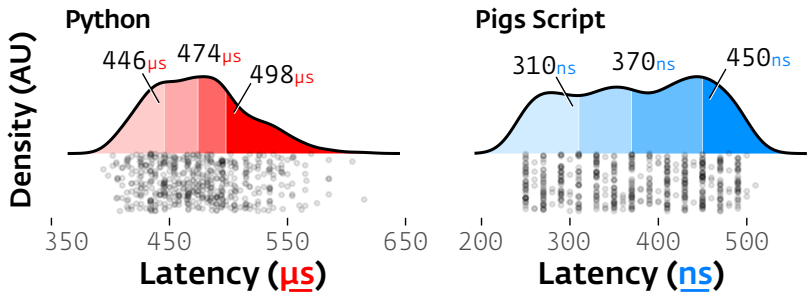


Figure 6.3: Roundtrip latency from external trigger to digital output using two methods: Typical Autopilot callback function given to a `Digital_In` object that turns a `Digital_Out` pin on for 1ms when an input pin changes state (Red, Left, Median ± IQR: 474 μs± 52.5). Pigs script that runs entirely within the pigpio daemon (Blue, Right, 380ns ± 140). For each, black points represent individual measurements (n=525), annotations are quartiles.

6.2 *Sound Latency*

We measured end to end, hardware input to sound output latency by measuring the delay between an external digital input and the onset of a 10kHz pure tone (Table 6.4). Sound playback was again triggered by the `Digital_In` class's callback method, and sound samples were buffered in a deque held in a separate process by the jack audio client between each trial. A `Digital_Out` pin was wired to the `Digital_In` pin in order to deliver the trigger pulse (but the `Digital_Out` pin was uninvolved in the software trigger for sound output).

Autopilot's jack audio backend was configured with a `192kHz` sampling rate with a buffer with two periods of of `32` samples each for theoretical minimum latency of `0.33ms`[2]. We observed a median 1.35ms (± 0.72 IQR) latency across 521 samples — roughly 4x the theoretical minimum (Figure 6.4). This suggests that Autopilot eliminates most perceptible end-to-end latency, which is necessary for tasks that require realtime feedback. One clear future direction is to write the sound processing loop in a compiled language exposed with a foreign function interface (FFI) to decrease both latency and jitter.

Table 6.4: Sound Latency Materials

Sound Card	Hifiberry Amp2
jack	1.9.22
Code	test_sound.py
Replicate	`python test_sound.py`

[2] A previous version of this paper included benchmarking and comparison to Bpod and pyControl's sound onset latency, but since then both packages have changed substantially, including Bpod creating a new hifi sound module based off HiFiBerry hardware very similar to the card used here, making those benchmarks obsolete. In this version we have omitted comparative benchmarks in favor of allowing the maintainers of those packages to publish their own benchmarks.

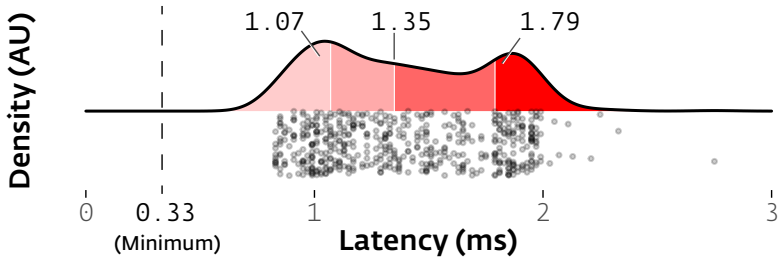


Figure 6.4: Autopilot has a median 1.35ms (± .72 IQR) latency between an external trigger and sound onset. Individual trials (dots, n=521) are shown beneath a density plot (red area under curve) colored by quartile (shades, numbers above are median, first, and third quartile). This latency is roughly 4x the theoretical minimum (0.33ms, dashed line).

6.3 *Network Latency*

To support data-intensive tasks like those that require online processing of video or electrophysiological data, the networking modules at the core of Autopilot need high bandwidth and low latency.

To test the latency of Autopilot's networking modules, we switch from "script mode" to "Task mode" (Table 6.5). Tasks are useful for encapsulating multistage routines across multiple devices that would be hard to coordinate with scripts alone. Our `Network_Latency` task consists of one "leader" pilot sending timestamped messages to a "follower" pilot which returns the timestamp marking when it received the message. The two pis communicate via two directly connected `Net_Nodes` (rather than routing each message through agent-level `Station` objects) after the leader pi initiates the follower with a multihop "START" message routed through a Terminal agent containing the task and networking parameters. We measured latency using software timestamps while synchronizing the clocks of the two pis with Chrony, an NTP daemon previously measured to synchronize Raspberry Pis within dozens of microseconds[205][3], with the leader pi hosting an NTP server and the follower pi synchronizing its clock solely from the leader. We documented this on the wiki too, since synchronization is a universal problem in multi-computer experiments.

Point to point latency was 0.975ms (median, ± 0.1 IQR, n=10,000, Figure 6.5) with some clear bimodality where a subset of messages (2,300 of 10,000) took longer

Table 6.5: Network Test Materials

Router	TP-Link AC1750
Chrony	4.0
zmq	22.3.0
Code	Network_Latency
Replicate	Assign task to subject from Terminal, start Task.

[3] Our sync is likely to be near to or better than that reported in [205]: in addition to a quiet network, we configured chrony to poll more frequently and tolerate a smaller error than default

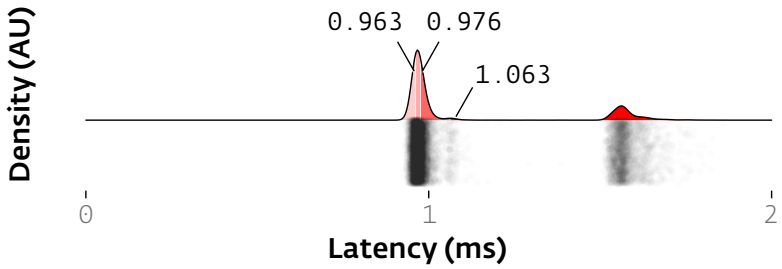


Figure 6.5: Network latency from when a message is sent from one pilot to when it is received by another. Messages took 0.975ms to send and receive (median, ± 0.1 IQR, n=10,000, overlaid numbers and red shading in density plot indicate quartiles). There is a clear bimodality in latencies for individual messages (black dots, jittered in y-axis) with unclear cause.

(median 1.567ms). The source of the bimodality is unclear to us, though it could be due to network congestion or interruption by other processes as the networking modules are not run in their own process like the sound server. This latency includes message serialization and deserialization by the builtin JSON library, which is on the order of roughly $100\mu s$ each for even the very small messages sent in this test. In future versions we will explore other serialization tools like msgpack and offer them as alternate serialization backends.

6.4 Network Bandwidth

To test Autopilot's bandwidth, we demonstrate yet another modality of use, using Autopilot's `Bandwidth_Test` widget, an action available from the Terminal GUI's `tests` menu that corresponds to a callback "listen" method in the Pilot (Table 6.6). This test requests that one or several pilots send messages at a range of selected frequencies and payload sizes back to the terminal. The messages pass through four networking objects en route: the stations and network nodes running the test for both the terminal and pilots (See Figure 5.14).

Table 6.6: Bandwidth Test Materials.

Terminal	Macbook Pro 2019, macOS 12.3.1, 2.4 GHz 8-Core Intel Core i9
blosc2	v0.2.0
Code	Bandwidth_Test, Pilot.l_bandwidth
Replicate	Terminal > Tools > Test Bandwidth

The needs for streaming experimental data vary depending on what is being streamed. Electrophysiological data is an n-electrode length vector sampled at a rate of dozens to hundreds of kilohertz, so each individual message isn't very large but there are a lot of them. Video data is a width by height (and for color video, by channel) array that can be relatively large[4], but it is captured at dozens to hundreds of hertz. Different data streams also have different degrees of compressibility: noisy, quasirandom electrical signals compress relatively poorly, while the typical behavioral neuroscientist's video of an animal that takes up 1/10th of the frame against a white background can have compression ratios in the hundreds.

Autopilot tries to provide flexibility for streaming different data types by offering message batching and optional on-the-fly compression with blosc. The bounds on bandwidth are then the speed at which an array can be compressed and the rate at which messages of a given size can be sent.

Autopilot's networking modules were able to send an "empty" (402 byte) message with headers describing the test but no payload at a maximum observed rate of 1,818Hz[5]. Approximately 15% of the duration is spent in message serialization, as a "frozen" preserialized message can be sent at 2,100Hz, though we imagine the need to send the same message thousands of times is rare.

We tested four types of messages with nonzero array[6] payloads: since the entropy of an array determines how compressible it is, we sent random and all-zero arrays with

[4] (1920 * 1080 * 3 * 8 bits) / 8 = 6 megabytes per frame of a 1080p color video, which is why video is rarely streamed uncompressed

[5] maximum average rate of 5000 messages for each of the equivalent empty message tests in the four conditions described below

[6] In all cases, float64 numpy arrays encoded in base64

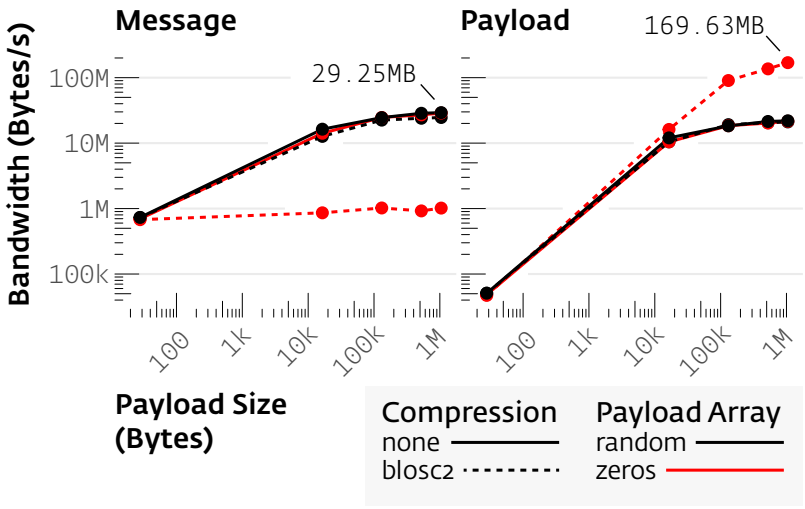


Figure 6.6: Bandwidth measurements between a pilot and terminal for compressed (solid lines) or uncompressed (dotted lines) arrays of random numbers (black) or zeros (red, each point n=5000 messages). As message size increased, the bandwidth for the rate of bytes transferred in serialized messages ("message bandwidth," left) plateaued at 29.25MBytes/s, while the effective bandwidth of arrays before and after compression ("payload bandwidth," right) reached 169.63MBytes/s. Real data will fall somewhere in this effective bandwidth range, depending on its compressibility.

and without compression. The random and all-zero arrays are the floor and ceiling of compressibility, respectively. Compression gives us two notions of bandwidth: the literal number of bytes that can be passed through a connection, and the effective bandwidth of the size of the arrays that can be transferred with a given compression ratio. We refer to these as "message" and "payload" bandwidth, respectively in Figure 6.6. Message bandwidth reflects the hardware limitations of the Raspberry Pi, but payload bandwidth is the number that matters in practice, as it measures the actual "speed of data" that can be used by the receiver.

As we increased the size of the array payload[7], the message bandwidth plateaued at a maximum of 29.25MByte per second (Figure 6.6, left). After this plateau, increasing the message size trades off linearly with the rate of messages sent. For all but the compressed array of zeros, the payload bandwidth mirrored the message bandwith with some trivial overhead from the base64 encoding. The compressed array of zeros, however, had an effective payload bandwidth of 169.6MBytes/s, a compromise between the speed of compression with the smaller message size[8]. The compressed random array had only negligible differences in payload and message bandwidth compared to the uncompressed random array, indicating that the overhead for blosc is trivial.

The ability to batch messages allows researchers to tune the size of an individual message to their particular need for high bandwidth or low latency. Since the compressibility of real data varies across the entire entropic range from randomness to arrays of all zeros, Autopilot doesn't have a single "bandwidth", but one that ranges between 30 and 170MByte/s[9].This bandwidth makes Autopilot capable of streaming raw Calcium imaging[10] and electrophysiological data from modern high-density probes[11]. Its flexible architecture allows researchers to decide how to build their experiments by distributing different components over different combinations of computers: stream data from a raspberry pi to a more powerful computer for processing, use GPIO rather than network triggers for time-critical operations — meeting the tooling challenge of complex, hardware-intensive, multimodal experiments that define contemporary systems neuroscience.

[7] n=5,000 for each condition at each size

[8] A message with a 1MByte zero array payload compressed to 6KBytes.

[9] In this dataset. There is additional payload bandwidth headroom with larger messages, and we include an additional dataset with a 200MByte/s bandwidth in the supplement.

[10] 2-Photon: 5.9MB/s
(12 bits * 512x512 resolution * 15Hz)

[11] Neuropixels: 14.4MB/s[156]
(10 bits * 30kHz * 384 channels)

6.5 *Distributed Go/No-go Task*

We designed a visual go/no-go task as a proof of concept for distributing task elements across multiple Pis, and also for the presentation of visual stimuli (Figure 6.7, Table 6.7). While the rest of the tests presented have been re-run, in the time since the intitial publication of the preprint we have not done substantial work on Autopilot's visual stimulus module, and so this section is presented as previously written using the v0.1.0 initial release.

In this task, a head-fixed subject would[12] be running on a wheel in front of a display with a lick-detecting water port able to deliver reward. Above the port is an LED. Whenever the LED is green, if the subject drops below a threshold velocity for a fixation period, a grating stimulus at a random orientation is presented on the monitor. After a random delay, there is a chance that the grating changes orientation by a random amount. If the subject licks the port in trials when the orientation is changed, or refrains from licking when it is not, the subject is rewarded.

One pilot controlled the operation of the task, including the coordination of a copilot. The pilot was connected to the LED and solenoid valve for reward delivery, as well as a monitor[13] to display the gratings[14]. The copilot continuously streamed velocity data (measured with a USB optical mouse against the surface of the wheel) back to the terminal for storage (see also Figure 5.14, which depicts the network topology for this task). The copilot waited for a message from the pilot to initiate measuring velocity, and when a rolling average of recent velocities fell below a given threshold the copilot sent a TTL trigger back to the pilot to start displaying the grating. This split-pilot topology allows us to poll the subject velocity continuously (at `125Hz` in this example) without competing for resources with psychopy's rendering engine.

We measured trigger (TTL pulse from the copilot) to visual stimulus onset latency using the measurement cursors of our oscilloscope as before. To detect the onset of the visual stimulus, we used a high-speed optical power meter attached to the top-left corner of our display monitor. The stimulus was a drifting Gabor grating drawn to fill half the horizontal and vertical width of the screen (`960 x 540px`), with a spatial frequency of `4cyc/960px` and temporal (drift) frequency of `1Hz`.

We observed a bimodal distribution of latencies (Quartiles: `28, 30, 36ms, n=50`, Figure 6.8), presumably because onsets of visual stimuli are quantized to the refresh rate (`60Hz, 16.67ms`) of the monitor. This range of latencies corresponds to the second and third frame after the trigger is sent (2/3 of observations fall in the 2nd frame, 1/3 of observations in the 3rd frame). We observed a median framerate of `36.2 FPS (IQR: 0.7)` across 50 trials (`8863` frames, Figure 6.9).

We further tested the Pi's framerate by using Psychopy's `timeByFrames` test—a script that draws stimuli without any Autopilot components running—to see if the framerate limits were imposed by the hardware of the Raspberry Pi or overhead from Autopilot (Table 6.8). We tested a series of Gabor filters and random dot stimuli (dots travel in random directions with equal velocity, default parameters) at different screen resolutions and stimulus complexities. The Raspberry Pi was capable of moderately high framerates (`>60 FPS`) for smaller, lower resolution stimuli, but struggled (`<30 FPS`) for full HD, fullscreen stimuli.

Table 6.7: Go/No-go Materials

Hardware	
Beam Break	TT Electronics OPB901L55
Monitor	Acer S230HL
Lick Port	Autopilot Tripoke v1
Light Sensor	Thorlabs PM100D
Software	
psychopy	v3.1.5
glfw	v1.8.3

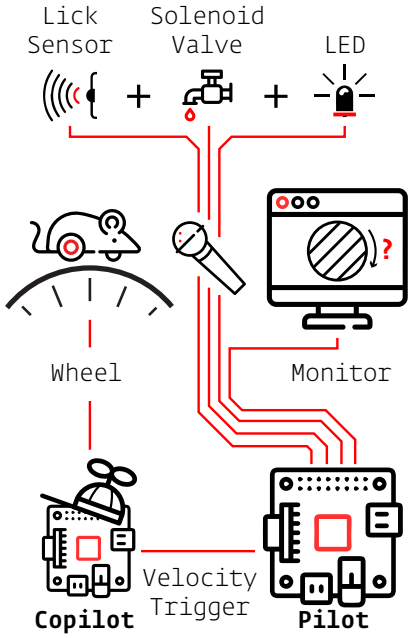


Figure 6.7: Hardware distribution for the distributed go/no-go task. Red lines indicate physical connections between hardware components. The lick sensor, solenoid valve, and LED are physically bundled into one component represented as the mouse's microphone.

[12] No mice were trained on this task

[13] (`1920×1080px, 60Hz`)

[14] Visual stimuli were presented with Psychopy using the glfw backend while Autopilot was run in a dedicated X11 server.

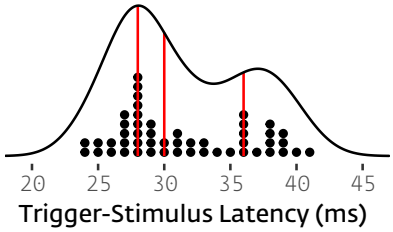


Figure 6.8: Stacked dots are a histogram of individual observations (n=50) underneath the probability density (black line), red lines indicate quartiles.

Autopilot is appropriate for realtime rendering of simple stimuli, and the proof-of-concept API we built around Psychopy doesn't impose discernible overhead (Mean framerate for a `960 x 540px` grating at `1080p` in Autopilot: `36.2 fps`, vs. `timeByFrames`: `35.0 fps`). In the future we will investigate prerendering and caching complex stimuli in order to increase performance. A straightforward option for higher-performance video would be to deploy an Autopilot agent running on a desktop computer with a high-performance GPU, or to use a single-board computer with a GPU like the NVIDIA Jetson ($99)[15].

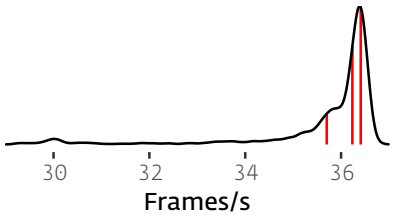


Figure 6.9: Probability density of framerates for 960 x 540px grating rendered at 1080p. Red lines indicate quartiles

[15] as we did in [180]

Stimulus	Resolution	Size / # Dots	Mean FPS	σ FPS
Gabor Filter	1280 x 720	300 x 300px	106.4	5.5
Gabor Filter	1920 x 1080	300 x 300px	75.2	3.5
Gabor Filter	1280 x 720	640 x 360px	53.5	2.2
Gabor Filter	1920 x 1080	960 x 540px	35.0	1.0
Gabor Filter	1280 x 720	720 x 720px	41.5	2.2
Gabor Filter	1920 x 1080	1080 x 1080px	20.1	0.7
Random Dots	1280 x 720	100 dots	98.0	3.8
Random Dots	1920 x 1080	100 dots	67.6	3.0
Random Dots	1280 x 720	1000 dots	20.9	0.25
Random Dots	1920 x 1080	1000 dots	19.5	0.36

Table 6.8: Tests performed over 1000 frames with PsychoPy's `timeByFrames` test.

7

Limitations and Future Directions

We will likely never view Autopilot as "finished." Autopilot—like all open-source software—is an evolving project, and this paper captures it as a snapshot at v0.5.0. We are invested in its development, and will be continually working to fix bugs, make its use more elegant, and add new features in collaboration with other researchers.

We expect that as the codebase matures and other researchers use Autopilot in new, unexpected ways that some fundamental elements of its structure may evolve. We have built version logging into the structure of the system so that changes will not compromise the replicability of experiments (see 7.5 below). While there will inevitably be breaking changes, these will be transparently documented, announced in release notes, and indicated with semantic versioning in order to alert users and describe how to adapt as needed.

We recognize the risk and inertia of retooling lab infrastructure, and there is still much work to be done on Autopilot. We welcome all issues and questions from anyone interested in contributing, or just curious to try it out — trying Autopilot is ultimately as risky as buying a Raspberry Pi.

The current major planned changes (also see the todo page in the docs) include:

1. **Python, Meet Rust** - Python is very useful as a high-level glue language, and its accessibility to a large number of scientific programmers is important to us, but it has its own very real performance limitations. As Autopilot's modules mature and stabilize, we are interested in rewriting core routines like sound presentation and networking in rust and exposing them to python with tools like PyO3

2. **Real Realtime** - Beneath user space decisions like programming language, the timing of CPU operations in linux is still determined by the kernel — this is one of the major reasons why other projects are based around dedicated microcontrollers. For almost everything that most scientists want to do, the standard linux kernel is perfectly fine, but we are interested in investigating what it would take to provide true deterministic realtime performance via Autopilot's high-level object system. One approach might be to provide prebuilt realtime kernel images along with tools to easily deploy them, though no firm plan has been made.

3. **Integration** - We will continue to collaborate with other programming teams to be interoperable with a broader array of other tools. Our next set of planned integrations include recording electrophysiological data by integrating with Open Ephys[206], optical imaging data from the Miniscope project [207, 208], and shared processing and control pipelines with Bonsai[168].

4. **Data Ingest & Export** - We are releasing Autopilot's data modeling system in v0.5.0 as an alpha release alongside this paper, and it includes prototype export interfaces to Neurodata Without Borders[182] and Datajoint[209]. Over the next several releases, we will continue to improve our data model so that researchers can easily structure their data and choose among different backends for storage. We are also working on a separate project to make tools to ingest data from the more ad-hoc directory-based data formats widely used in science and ingest them into Autopilot's and other tools formal modeling systems. In the longer term, we are interested in making Autopilot interoperable with linked data systems as part of a broader vision of digital infrastructure.

5. **Provenance** - Autopilot stores version information and local configuration in multiple places, and it is technically possible to faithfully replicate an experiment, but recording of provenance can still be consolidated and improved. By formalizing our object and data model, we will also systematize the many changes in configuration and version possible across the system for complete provenance tracking.

6. **P2P Networking** - The default tree structure of Autopilot's networking modules has proven to be unnecessarily limiting over time. In part, we had preoptimized for processing messages in a separate processes assuming that would help problems from dropped messages and overflowing send buffers, but in practice messages are almost never dropped and network nodes are as effective as stations in sending and receiving large amounts of data. As part of unifying Autopilot's object system, we will implement a fully peer-to-peer networking system such that each instantiated object has a unique ID so that messages can be easily addressed from any object to any other in its swarm. We will learn from previous p2p addressing systems like distributed hash tables to allow net nodes to join the swarm and discover all other nodes automatically without manually configuring IP addresses and ports. In the longer term we are interested in peer to peer data transfer as well, so that an object serving as a data source can efficiently stream to many consumers without needing to duplicate each message for every consumer.

7. **Slots, Signals, and Streaming** - We will be supplementing a more general network structure with a system of specifying which attributes of each object are data sources, which are sinks, and what kind of connection they accept. Similar to Qt's signal and slot model, we want to make it as easy as using a `.connect()` method to control one piece of hardware with another. The transforms module should also be able to support branching and forking operations so multiple data sources can be combined for elaborated hardware control. ZeroMQ is an excellent tool for sending and receiving control messages, but formalized signals and slots could also specify different streaming tools like redis or gstreamer that might be better suited for high-bandwidth linear streams like video. Applied generally, this could also solve related problems like the relatively implicit handling of event triggers in the Task class and the need for manual configuration of connections between pilots and copilots.

8. **Rebuild the GUI** - The GUI is some of the oldest code in the library, was written before most of the other modules existed, and needs to be rebuilt. We have started by remaking its central widgets to be generated from pydantic models used increasingly throughout the system, but the rest of the GUI still needs to be rearchitected into a structure that decreases code duplication and allows us to do things like provide GUI extensions via plugins. We will likely continue to use Qt for the near future, but are also exploring the idea of webassembly tools to make browser-based web interfaces for remote control.

9. **Plugins** - We want Autopilot's plugin system to be permissive and as natural as the scripting style that most experimental code is written in, but we still need some means of specifying dependencies on other packages and plugins, among other improvements. We will be making a plugin generator that makes a folder of plugin boilerplate, as well as tools for installing, uploading, and synchronizing versions with git and the wiki. Over time we will make all object types within Autopilot able to be extended with plugins, as well as make it possible to override and extend built-in objects.

10. **Knowledge Organization** - We have been extending our thinking from code itself to more broadly consider the social systems that surround research code. The wiki was our first step, and we will continue to make more points of integration for smoothly incorporating contextual knowledge typically stored in lab notebooks into a public, collectively curated information system. We want to make it easier not only for individual researchers to use Autopilot, but make it easier for labs to coordinate work across projects without needing to rely on proprietary SaaS platforms with additional tooling for managing swarms, and moving beyond a single Autopilot wiki to a federated system of wikis for fluid continuity between "private" local coordination and "public" shared knowledge.

11. **Tests** - Our collection of tests doesn't cover the whole codebase, and so as we formalize our contribution process will move towards a system where all new code must have tests and documentation to be integrated. We also want to integrate our tests more closely with our documentation so that researchers know which part of the code has explicit tests guaranteeing functionality.

12. **Security** - Autopilot is a networked program, and while it doesn't execute arbitrary code from network messages, there is no security model to speak of. So far this hasn't beren a problem, as we encourage only using Autopilot on a local network behind a router, but as we we build out our networking modules we will investigate how to incorporate identity verification systems to protect swarms from malicious messages.

13. **Metastructure & API Maturity** - The scope and structure of Autopilot is still in flux relative to other, more mature Python packages. To reach a stable v1.0.0 API, we are in the process of unifying Autopilot's object structure so everything is clearly typed, all configuration is explicit, and all code written to handle special cases is absorbed into more general systems. Different

parts of Autopilot have had different degrees of care over time, and so we will be working to catch the oldest modules up, trim unused ones, and make sure every line in the library is documented and useful. For the time being, flexibility is useful because frequently used or requested features trace a desire path outlining how its users believe Autopilot should behave. Each shortcoming we fix in Autopilot's modules makes it more straightforward to fix the rest, and so once the major remaining work is completed we will transition to a more conservative pace of development that ensures the longevity of the project.

8
Glossary

Agent	5.7	The executable part of Autopilot. A set of startup routines (eg. opening a GUI or starting an audio server), runtime behavior (eg. opening as a window or running as a background system process), and event handling methods (ie. **listens**) that constitute the role of the particular Autopilot instance in the **swarm**.
Copilot	5.7	An **agent** that performs some auxiliary, supporting role in a **task**—primarily used for offloading some hardware responsibilities from a **pilot**.
Graduation	5.3	Moving between successive **tasks** in a **protocol** when some criterion is met.
Listen	5.8	A method belonging to the **station** or **node** of a particular **agent** that defines how to process a particular type of message (ie. a message with a particular `key`).
Node	5.8	A networking object that some module (eg. hardware, **tasks**, GUI routines) or method (eg. a **listen**) uses to communicate with other **nodes**. Messages to other **agents** in the swarm are relayed through their **Station**
Pilot	5.7	An **agent** that runs on a Raspberry Pi, the primary experimental agent of Autopilot. Typically runs as a system service, receives **tasks** from a **terminal** and runs them. Can organize a group of **children** if requested by the **task**.
Protocol	5.3	A (`.json`) file that contains a list of **task** parameters and the **graduation** criteria to move between them. The **tasks** in a protocol are also known as its **levels**.
Stage	5.3	**Stages** are methods that implement the logic of a **task**. They can be used analogously to states in a finite-state machine (eg. wait for **trial** initiation, play stimulus, etc.) or asynchronously (whenever x input is received, rotate stimulus by y degrees).
Station	5.8	Each **agent** has a single **station**, a networking object that is run in its own process and is responsible for communication between **agents**. The **station** also routes messages from **children** or other **nodes**.
Swarm		Informally, a group of connected **agents**.
Task	5.3	A formalized description of an experiment: the parameters it takes, the data that it collects, the hardware it needs, and a collection of **stages** that describe what happens during the experiment.
Terminal	5.7	A user-facing **agent** that provides a GUI for operating and maintaining a **swarm**.
Topology	5.7.1	A particular combination of **agents**, their designated responsibilities, and the networking connections between them invoked by a **task** (eg. task requires one pilot to record video, one to process the video, and one to administer reward) or by usage (eg. 10 pilots are connected to a single terminal and are typically used to run 10 independent tasks, though they could run shared tasks together).
Trial	5.3	If a **task** is structured such that its **stages** form a repeating series, a **trial** is a single completion of that series.

Part III

Infrastructure

> If we can make something decentralised, out of control, and of great simplicity, we must be prepared to be astonished at whatever might grow out of that new medium.
>
> Tim Berners-Lee (1998): Realising the Full Potential of the Web

> A good analogy for the development of the Internet is that of constantly renewing the individual streets and buildings of a city, rather than razing the city and rebuilding it. The architectural principles therefore aim to provide a framework for creating cooperation and standards, as a small "spanning set" of rules that generates a large, varied and evolving space of technology.
>
> RFC 1958: Architectural Principles of the Internet

> In building cyberinfrastructure, the key question is not whether a problem is a "social" problem or a "technical" one. That is putting it the wrong way around. The question is whether we choose, for any given problem, a primarily social or a technical solution
>
> Bowker, Baker, Millerand, and Ribes (2010): Toward Information Infrastructure Studies [210]

> Billionaires have squatted on the Magna Cum Lauded / [...] Methodically they plotted against those who fought it / [...] / Now the scientific process got hijacked for profits / It flows in the direction that a silver spoon prodded / We'll get science for the people when we run the economics.
>
> The Coup (2012) The Gods of Science

9

Introduction

We work in an archipelago of technical islands: researchers, labs, consortia, and a few well-funded institutions reinventing the wheel in parallel. Our knowledge dissemination systems are as nimble as static pdfs[1] served by an extractive publishing turned surveillance industry we can't seem to quit. Experimental instrumentation except for that at the polar extremes of technological complexity or simplicity is designed and built custom, locally, and on-demand[2]. Software for performing experiments is a patchwork of libraries that satisfy some of the requirements of the experiment, sewn together by some uncommented script written years ago by a grad student who left the lab long-since. The technical knowledge to build both instrumentation and software is fragmented and unavailable as it sifts through the funnels of word-limited methods sections and never-finished documentation. Our data is born into this world without coherent form to speak of, indexable only by passively-encrypted notes in a paper lab notebook, and analyzed once before being mothballed in ignominy on some unlabeled external drive.

[1] (save some complicated half-in flirtation with social media).

[2] At least in systems neuroscience, appropriate caveats below.

These problems are typically treated in isolation, but all are symptomatic of a broader deficit in **digital infrastructure** for science. Every routine need that requires heavy technical development, an appeal to a hostile publishing system, or yet another platform subscription is an indicator that infrastructural deficits *define the daily reality of science.* We *should* be able to easily store, share, and search for data; be able to organize and communicate with each other; be able to write and review our work, but we are hemmed in on all sides by looming tech profiteers and chasms of underdevelopment.

If the term infrastructure conjures images of highways and plumbing, then surely digital infrastructure would be flattered at the association. Roughly following Star and Ruhleder's (1996) dimensions [211] , by analogy they illustrate many of its promises and challenges: when designed to, it can make practically impossible things trivial, allowing the development of cities by catching water where it lives and snaking it through tubes and tunnels sometimes directly into your kitchen. Its absence or failure is visible and impactful, as in the case of power outages. There is no guarantee that it "optimally" satisfies some set of needs for the benefit of the greatest number of people, as in the case of the commercial broadband duopolies. It exists not only as its technical reality, but also as an embodied and shared set of social practices, and so even when it does exist its form is not inevitable or final; as in the case of bottled water producers competing with municipal tap water on a behavioral basis despite being dramatically less efficient and more costly. Finally it is not socially or ethically neutral, and the impact of failure to build or maintain it is not equally shared, as in the expression of institutional racism that was the Flint, Michigan water crisis [212] .

Infrastructural deficits are not our inevitable and eternal fate, but the course of infrastructuring is far from certain. It is not the case that "scientific digital infrastructure" will rise from the sea monolithically as a natural result of more development time and funding, but instead has many possible futures[213] , each with their own

advocates and beneficiaries. Without concerted and strategic development based on a shared and liberatory ethical framework, science will continue to follow the same path as other domains of digital technology down the dark road of platform capitalism. The prize of owning the infrastructure that the practice of science is built on is too great, and it is not hard to imagine tech behemoths buying out the emerging landscape of small scientific-software-as-a-service startups and selling subscriptions to Science Prime.

The possibility of future capture of nascent infrastructure is still too naive a framing: operating as obligate brokers of (usually surveillance) data[214, 215, 216] , prestige, and computational resources naturally relies on *displacing* the possibility of alternative infrastructure. Our predicament is doubly difficult: we both have digital infrastructural deficits, but are also being actively *deinfrastructured.* The harms of deinfrastructuring are bidirectional, comprising both the missed opportunities from decades of free knowledge exchange, and the impacts of the informational regime that exists in its place. One can only imagine what the state of science and medicine might be if NIH's 1999 push to displace for-profit journals[217, 218, 219, 220] had succeeded and we had more than 20 years of infrastructural development built atop a fundamentally free system of scientific knowledge. Instead, our failure to seize the digital infrastructure of science has led to a system where what should be our shared intellectual heritage is yoked to the profit engine of surveillance conglomerates (formerly known as publishers) [214, 221] that repackage it along with a deluge of mined personal data in a circular economy of control [222, 223, 224] that makes us directly complicit in the worst abuses of informational capitalism [225, 226, 227, 228, 229] .

We need to move beyond conceptualizing the problems of scientific infrastructure as being unique to science, a sighing hope for some future that "might be nice" to have (built by an always-anonymous "*someone else*"), but one to be pursued gradually after staid and cautious scholars are convinced no risk will come to our precious systems of prestige and peer review. We need to start seeing ours as one of many stories in the digital enclosure movement where adversarial economic entities take ownership of basic digital infrastructure and wipe out a domain of knowledge work, reducing it to a captive market and content farm [216, 230] . We need to see taking control of our digital infrastructure as *essential* to the continued existence of science as we know it.

This paper is an argument that **decentralized** digital infrastructure[3] is the best means of alleviating the harms of infrastructural deficits and building a digital landscape that supports, rather than extracts from science. I will draw from several disciplines and knowledge communities, across and outside academia to articulate a vision of an infrastructure in three parts: **shared data, shared tools, and shared knowledge.** These domains reflect three of the dominant modes of digital enclosure prerequisite for platform capture: **storage, computation, and communication.** The systems we will describe are in conversation with and a continuation of a long history of reimagining the relationship between these domains for a healthier web (see eg. [231, 232]). We depart from it to describe a system of fluid, peer-to-peer social affiliation and folksonomic linked data with lessons primarily from early wikis and Wikipedia, the fissures of the semantic web and linked data communities, the social structure of private bittorrent trackers, and the federation system of Activity-

[3] Recently the notion of decentralized digital infrastructure has been co-opted by a variety of swindlers and other motivated parties to refer to blockchain-based technologies like cryptocurrencies, decentralized autonomous organizations, and the like. This work will not discuss them, as they have not been demonstrated to do anything that peer-to-peer technology with adjoining social systems can't do except use a collosal quantity of fossil fuels and drain a lot of credulous people's bank accounts.

Pub and the Fediverse. Approaching this problem from science has its constraints — like the structuring need to rebuild systems of credit assignment — as well as the powerful opportunity of one of the last systems of labor largely not driven by profit developing technology and seeding communities that could begin to directly address the dire, societywide need for digital freedom.

The problems we face are different than they were at the dawn of the internet, but we can learn from its history: we shouldn't be waiting for a new journal-like **platform,** software package, or subscription to save us. We need to build **protocols** for communication, interoperability, and self-governance (see, recently [233]).

I will start with a brief description of what I understand to be the state of our digital infrastructure and the structural barriers and incentives that constrain its development. I will then propose a set of design principles for decentralized infrastructure and possible means of implementing it informed by prior successes and failures at building mass digital infrastructure. I will close with contrasting visions of what science could be like depending on the course of our infrastructuring, and my thoughts on how different actors in the scientific system can contribute to and benefit from decentralization.

I insist that what I will describe is *not utopian* but is eminently practical — the truly impractical choice is to do nothing and continue to rest the practice of science on a pyramid scheme [234] of underpaid labor. With a bit of development to integrate and improve the tools, **every class of technology I propose here already exists and is widely used.** A central principle of decentralized systems is embracing heterogeneity: harnessing the power of the diverse ways we do science instead of constraining them. Rather than a patronizing argument that everyone needs to fundamentally alter the way they do science, the systems that I describe are specifically designed to be easily incorporated into existing practices and adapted to variable needs. In this way I argue decentralized systems are *more practical* than the dream that any one system will be capable of expanding to the scale of all science — and as will hopefully become clear, inarguably *more powerful* than a disconnected sea of centralized platforms and services.

An easy and common misstep is to categorize this as solely a *technical* challenge. Instead the challenge of infrastructure is also *social* and *cultural* — it involves embedding any technology in a set of social practices, a shared belief that such technology should exist, that its form is not neutral, and a sense of communal valuation and purpose that sustains it [235] .

The social and technical perspectives are both essential, but make some conflicting demands on the construction of the piece: Infrastructuring requires considering the interrelatedness and mutual reinforcement of the problems to be addressed, rather than treating them as isolated problems that can be addressed piecemeal with a new package or by founding a new journal alternative. Such a broad scope trades off with a detailed description of the relevant technology and systems, but a myopic techno-zealotry that does not examine the social and ethical nature of scientific practice risks reproducing or creating new sources of harm. That, and techno-solutionism never *works* anyway. As a balance I will not be proposing a complete technical specification or protocol, but describing the general form of the tools and some existing

examples that satisfy them; I will not attempt a full history or treatment of the problem of infrastructuring, but provide enough to motivate the form of the proposed implementations.

My understanding of this problem is, of course, uncorrectably structured by my training largely centered in systems neuroscience and my position as an early career researcher (ECR). While the core of my argument is intended to be a sketch compatible with sciences and knowledge systems generally, my examples will sample from, and my focus will skew to my experience. In many cases, my use of "science" or "scientist" could be "neuroscience" or "neuroscientist," but I will mostly use the former to avoid the constant context switches. This document is also an experiment in public collaboration on a living scientific document: to try and ease our way out of disciplinary tunnelvision, we invite annotation and contribution with no lower bound — if you'd like to add or correct a sentence or two (or a page or ten), you're welcome as coauthor. I ask the reader for a measure of patience for the many ways this argument requires elaboration and modification for distant fields.

10

The State of Things

10.1 The Costs of Infrastructure Deficits

A diagnosis of digital infrastructure deficits gives a common framework to consider many technical and social harms in scientific work that are typically treated separately, and allows us to problematize other symptoms have become embedded as norms.

I will list some of the present costs to give a sense of the scale of need, as well as scope for the problems we intend to address here. These lists are grouped into rough and overlapping categories, but make no pretense at completeness.

Impacts on the **daily experience** of researchers include:

- A prodigious duplication and dead-weight loss of labor as each lab, and sometimes each person within each lab, will reinvent basic code, tools, and practices from scratch. Literally it is the inefficiency of the Harberger's triangle in the supply and demand system for scientific infrastructure caused by inadequate supply. Labs with enough resources are forced to pay from other parts of their grants to hire professional programmers and engineers to build the infrastructure for their lab[1], but most just operate on a purely amateur basis. Many PhD students will spend the first several years of their degree re-solving already-solved problems, chasing the tails of the wrong half-readable engineering whitepapers, in their 6th year finally discovering the technique that they actually needed all along. That's not an educational or training model, it's the effect of displacing the undone labor of unbuilt infrastructure on vulnerable graduate workers almost always paid poverty wages.

[1] (and usually their lab or institute only)

- At least the partial cause of the phenomenon where "every scientist needs to be a programmer now" as people who aren't particularly interested in being programmers — which is *fine* and *normal* — need to either suffer through code written by some other unlucky amateur or learn several additional disciplines in order to do the work of the one they chose.

- A great deal of pain and alienation for early- career researchers not previously trained in programming before being thrown in the deep end. Learning data hygeine practices like backup, annotation, etc. "the hard way" through some catastrophic loss is accepted myth in much of science. At some scale all the very real and widespread pain, guilt, and shame felt by people who had little choice but to reinvent their own data management system must be recognized as an infrastructural, rather than a personal problem.

- The high cost of "openness" and the dearth of mass-scale collaboration. It is still rare to publish full, raw data and analysis code, often because the labor of cleaning it is too great. We can't expect openness from everyone while it is still so *hard*. The "Open science" movement, roughly construed, has reached a few

hard limits from present infrastructure that have forced its energy to leak from the sides as bullying leaderboards or sets of symbols that are mere signifiers of cultural affiliation to openness. "Openness" is not a uniform or universal goal for all science, and even the framing of openness as inspection of results collected and analyzed in private isolation illustrates how infrastructural deficts bound our imagination. Our dreams can be bigger than being able to police each other's data, towards a more continuously collaborative process that renders the need for post-hoc openness irrelevant with mutually beneficial information sharing baked into every stage.

Impacts on the **system of scientific inquiry** include:

- A profoundly leaky knowledge acquisition system where entire PhDs worth of data can be lost and rendered useless when a student leaves a lab and no one remembers how to access the data or how it's formatted.
- The inevitability of continual replication crises because it is often literally impossible to replicate an experiment that is done on a rig that was built one time, used entirely in-lab code, and was never documented
- Reliance on communication platforms and knowledge systems that aren't designed to, and don't come close to satisfying the needs of scientific communication. In the absence of some generalized means of knowledge organization, scientists ask the void[2] for advice or guidance from anyone that algorithmically stumbles by. Often our best recourse is to make a Slack about it, which is incapable of producing a public, durable, and cumulative resource: and so the same questions will be asked again... and again...
- A perhaps doomed intellectual endeavor as we attempt to understand the staggering complexity of the brain by peering at it through the camera obscura of just the most recent data you or your lab have collected rather than being able to index across the many measurements of the same phenomena. The unnecessary reduplication of experiments becomes not just a methodological limitation, but an ethical catastrophe as researchers have little choice but to abandon the elemental principle of sacrificing as few animals as possible.
- A near-absence of semantic or topical organization of research that makes cumulative progress in science proabilistic at best, and subject to the malformed incentives of publication and prestige gathering at worst. Since engaging with prior literature is a matter of manually reconstructing a caricature of a field of work in every introduction, continuing lines of inquiry or responding to conflicting results is *strictly optional.*
- A hierarchy of prestige that devalues the labor of many groups of technicians, animal care workers, and so on. Authorship is the coin of the realm, but many workers that are fundamental to the operation of science only receive the credit of an acknowledgement. We need a system to value and assign credit for the immense amount of technical and practical knowledge and labor they contribute.

[2] (Twitter)

Impacts on the relationship between **science and society**:

- An insular system where the inaccessibility of all the "contextual" knowledge [236, 166] that doesn't have a venue for sharing but is necessary to perform experiments, like "how to build this apparatus," "what kind of motor would work here," etc. is a force that favors established and well-funded labs who can rely on local knowledege and hiring engineers/etc. and excludes new, lesser-funded labs at non-ivy institutions. The concentration of technical knowledge magnifies the inequity of strongly skewed funding distributions such that the most well-funded labs can do a completely different kind of science than the rest of us, turning the positive-feedback loop of funding begetting funding ever faster.

- An absconscion with the public resources we are privileged enough to receive, where rather than returning the fruits of the many technical challenges we are tasked with solving to the public in the form of data, tools, collected practical knowledge, etc. we largely return papers. Since those papers are often impenetrable outside of their discipline or paywalled outside of academia, we multiply the above impacts of labor duplication and knowledge inaccessibility by the scale of society.

- The complicity of scientists in rendering our collective intellectual heritage nothing more than another regiment in the ever-advancing armies of platform capitalism. If our highest aspirations are to shunt all our experiments, data, and analysis tools onto Amazon Web Services, our failure of imagination will be responsible for yet another obligate funnel of wealth into the system of extractive platforms that dominate the flow of global information. For ourselves, we stand to have the practice of science filleted at the seams into a series of mutually incompatible subscription services. For society, we squander the chance for one of the very few domains of non-economic labor to build systems to recollectivize the basic infrastrucutre of the internet: rather than providing an alternative to the information overlords and their digital enclosure movement, we will be run right into their arms.

Considered separately, these are serious problems, but together they are a damning indictment of our role as stewards of our corner of the human knowledge project.

We arrive at this situation not because scientists are lazy and incompetent, but because we are embedded in a system of mutually reinforcing disincentives to cumulative infrastructure development. Our incentive systems are, in turn, coproductive with a raft of economically powerful entities that would really prefer owning it all themselves, thanks. Put bluntly, "we are dealing with a massively entrenched set of institutions, built around the last information age and fighting for its life" [210]

There is, of course, an enormous amount of work being done by researchers and engineers on all of these problems, and a huge amount of progress has been made on them. My intention is not to shame or devalue anyone's work, but to try and describe a path towards integrating it and making it mutually reinforcing.

Before proposing a potential solution to some of the above problems, it is important to motivate why they haven't already been solved, or why their solution is not necessarily imminent. To do that, we need a sense of the social and technical challenges that structure the development of our tools.

10.2 (Mis)incentives in Scientific Software

The incentive systems in science are complex, subject to infinite variation everywhere, so these are intended as general tendencies rather than statements of irrevocable and uniform truth.

10.2.1 Incentivized Fragmentation

Scientific software development favors the production of many isolated, single-purpose software packages rather than cumulative work on shared infrastructure. The primary means of evaluation for a scientist is academic reputation, primarily operationalized by publications, but a software project will yield a single paper (if any). Traditional publications are static units of work that are "finished" and frozen in time, but software is never finished: the thousands of commits needed to maintain and extend the software are formally not a part of the system of academic reputation.

Howison & Herbsleb described this dynamic in the context of BLAST[3]

> In essence we found that BLAST innovations from those motivated to improve BLAST by academic reputation are motivated to develop and to reveal, but not to integrate their contributions. Either integration is actively avoided to maintain a separate academic reputation or it is highly conditioned on whether or not publications on which they are authors will receive visibility and citation. [237]

[3] "Basic Local Alignment Search Tool" - a tool to compare genetic or protein sequences to find potential matches or analogues.

For an example in Neuroscience, one can browse the papers that cite the DeepLabCut paper [238] to find hundreds of downstream projects that make various extensions and improvements that are not integrated into the main library. While the alternative extreme of a single monolithic ur-library is also undesirable, working in fragmented islands makes infrastructure a random walk instead of a cumulative effort.

After publication, scientists have little incentive to **maintain** software outside of the domains in which the primary contributors use it, so outside of the most-used libraries most scientific software is brittle and difficult to use [239, 240, 241] .

Since the reputational value of a publication depends on its placement within a journal and number of citations (among other metrics), citation practices for scientific software are far from uniform and universal, and relatively few "prestige" journals publish software papers at all, the incentive to write scientific software in the first place is low compared to its near-universal use [242] .

10.2.2 Domain-Specific Silos

When funding exists for scientific infrastructure development, it typically comes in the form of side effects from, or administrative supplements to research grants. The NIH describes as much in their Strategic Plan for Data Science [243] :

> from 2007 to 2016, NIH ICs used dozens of different funding strategies to support data resources, most of them linked to research-grant mechanisms that prioritized innovation and hypothesis testing over user service, utility, access, or efficiency. In addition, although the need for open and efficient data sharing is

> clear, where to store and access datasets generated by individual laboratories—and how to make them compliant with FAIR principles—is not yet straightforward. Overall, it is critical that the data-resource ecosystem become seamlessly integrated such that different data types and information about different organisms or diseases can be used easily together rather than existing in separate data "silos" with only local utility.

The National Library of Medicine within the NIH currently lists 122 separate databases in its search tool, each serving a specific type of data for a specific research community. Though their current funding priorities signal a shift away from domain-specific tools, the rest of the scientific software system consists primarily of tools and data formats purpose-built for a relatively circumscribed group of scientists. Every field has its own challenges and needs for software tools, but there is little incentive to build tools that serve as generalized frameworks to integrate them.

10.2.3 "The Long Now" of Immediacy vs. Idealism

Digital infrastructure development takes place at multiple timescales simultaneously — from the momentary work of implementing it; through longer timescales of planning, organization, and documenting; to the imagined indefinite future of its use — what Ribes and Finholt call "The Long Now. [244] " Infrastructural projects constitutively need to contend with the need for immediately useful results vs. general and robust systems; the need to involve the effort of skilled workers vs. the uncertainty of future support; the balance between stability and mutability; and so on. The tension between hacking something together vs. building something sustainable for future use is well-trod territory in the hot-glue and exposed wiring of systems neuroscience rigs.

Deinfrastructuring divides the incentives and interests of junior and senior researchers. ECRs might be interested in developing tools they'll use throughout their careers, but given the pressure to establish their reputation with publications rarely have the time to develop something fully. The time pressure never ends, and established researchers also need to push enough publications through the door to be able to secure the next round of funding. The time preference of scientific software development is thus very short: hack it together, get the paper out, we'll fix it later.

The constant need to produce software that *does something* in the context of scientific programming which largely lacks the institutional systems and expert mentorship needed for well-architected software means that most programmers *never* have a chance to learn best practices commonly accepted in software engineering. As a consequence, a lot of software tools are developed by near-amateurs with no formal software training, contributing to their brittleness [245] .

The problem of time horizon in development is not purely a product of inexperience, and a longer time horizon is not uniformly better. We can look to the history of the semantic web, a project that was intended to bridge human and computer-readable content on the web, for cautionary tales. In the semantic web era, thousands of some of the most gifted programmers and some of the original architects of the internet worked with an eye to the indefinite future, but the raw idealism and neglect of the pragmatic reality of the need for software to *do something* drove many to abandon the effort (bold is mine, italics in original):

> **But there was no *use* of it.** I wasn't using any of the technologies for anything, except for things related to the technology itself. The Semantic Web is utterly inbred in that respect. The problem is in the model, that we create this metaformat, RDF, and *then* the use cases will come. But they haven't, and they won't. Even the genealogy use case turned out to be based on a fallacy. The very few use cases that there are, such as Dan Connolly's hAudio export process, don't justify hundreds of eminent computer scientists cranking out specification after specification and API after API.
>
> When we discussed this on the Semantic Web Interest Group, the conversation kept turning to how the formats could be fixed to make the use cases that I outlined happen. "Yeah, Sean's right, let's fix our languages!" But **it's not the languages which are broken,** except in as much as they are entirely broken: because **it's the *mentality* of their design which is broken.** You can't, it has turned out, make a metalanguage like RDF and then go looking for use cases. We thought you could, but you can't. It's taken eight years to realise. [246]

Developing digital infrastructure must be both bound to fulfilling immediate, incremental needs as well as guided by a long-range vision. The technical and social lessons run in parallel: We need software that solves problems people actually have, but can flexibly support an eventual form that allows new possibilities. We need a long-range vision to know what kind of tools we should build and which we shouldn't, and we need to keep it in a tight loop with the always-changing needs of the people it supports.

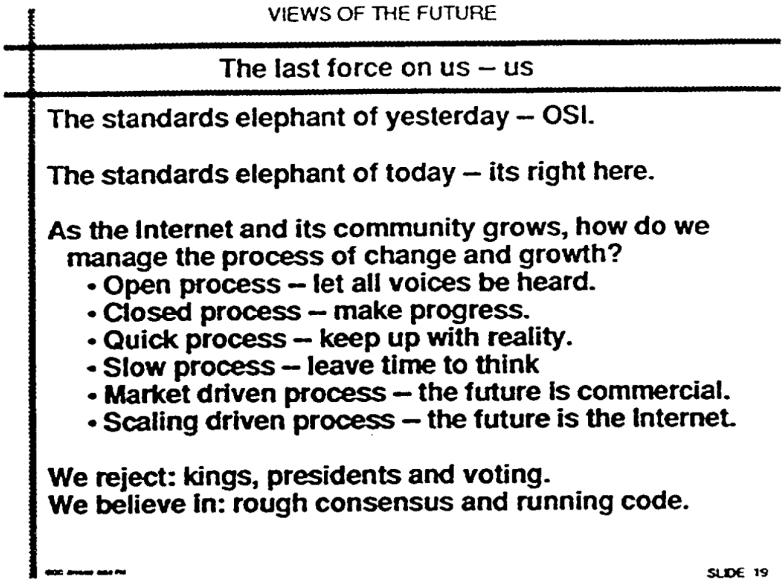
In short, to develop digital infrastructure we need to be *strategic.* To be strategic we need a *plan.* To have a plan we need to value planning as *work.* On the valuation of this kind of work, Ribes and Finholt are instructive:

> "On the one hand, I know we have to keep it all running, but on the other, LTER is about long-term data archiving. If we want to do that, we have to have the time to test and enact new approaches. But if we're working on the to-do lists, we aren't working on the tomorrow-list" (LTER workgroup discussion 10/05).
>
> The tension described here involves not only time management, but also the differing valuations placed on these kinds of work. The implicit hierarchy places scientific research first, followed by deployment of new analytic tools and resources, and trailed by maintenance work. [...] While in an ideal situation development could be tied to everyday maintenance, in practice, maintenance work is often invisible and undervalued. As Star notes, infrastructure becomes visible upon breakdown, and only then is attention directed at its everyday workings (1999). Scientists are said to be rewarded for producing new knowledge, developers for successfully implementing a novel technology, but the work of maintenance (while crucial) is often thankless, of low status, and difficult to track. *How can projects support the distribution of work across research, development, and maintenance?* [244]

10.2.4 "Neatness" vs "Scruffiness"

Closely related to the tension between "Now" and "Later" is the tension between "Neatness" and "Scruffiness." Lindsay Poirier traces its reflection in the semantic web community as the way that differences in "thought styles" result in different "design logics" [247] . On the question of how to develop technology for representing the ontology of the web – the system of terminology and structures with which everything should be named – there were (very roughly) two camps. The "neats" prioritized consistency, predictability, uniformity, and coherence – a logically complete and formally valid System of Everything. The "scruffies" prioritized local systems of knowledge, expressivity, "believing that ontologies will evolve organically as everyday webmasters figure out what schemas they need to describe and link their data. [247] "

This tension is as old as the internet, where amidst the dot-com bubble a telecom spokesperson lamented that the internet wasn't controllable enough to be profitable because "it was devised by a bunch of hippie anarchists." [248] The hippie anarchists probably agreed, famously rejecting "kings, presidents and voting" in favor of "rough consensus and running code" during an attempted ISO coup to replace TCP/IP with a proprietary protocol. Clearly, the difference in thought styles has an unsubtle relationship with beliefs about who should be able to exercise power and what ends a system should serve [249] .



A slide from David Clark's 1992 "Views of the Future"[3] that contrasts differing visions for the development process of the future of the internet. The struggle between engineered order and wild untamedness is summarized forcefully as "We reject: kings, presidents and voting. We believe in: rough consensus and running code"

Practically, the differences between these thought communities impact the tools they build. Aaron Swartz put the approach of the "neat" semantic web architects the way he did:

> Instead of the "let's just build something that works" attitude that made the Web (and the Internet) such a roaring success, they brought the formalizing mindset of mathematicians and the institutional structures of academics and defense contractors. They formed committees to form working groups to write drafts of ontologies that carefully listed (in 100-page Word documents) all possible things in the universe and the various properties they could have, and they spent hours in Talmudic debates over whether a washing machine was a kitchen appliance or a household cleaning device.
>
> With them has come academic research and government grants and corporate R&D and the whole apparatus of people and institutions that scream "pipedream." And instead of spending time building things, they've convinced people interested in these ideas that the first thing we need to do is write standards. (To engineers, this is absurd from the start—standards are things you write after you've got something working, not before!) [197]

The outcomes of this cultural rift are subtle, but the broad strokes are clear: the "scruffies" largely diverged into the linked data community, which has taken some of the core semantic web technology like RDF, OWL, and the like, and developed a broad range of downstream technologies that have found purchase across information sciences, library sciences, and other applied domains[4]. The linked data developers, starting by acknowledging that no one system can possibly capture everything, build tools that allow expression of local systems of meaning with the expectation and affordances for linking data between these systems as an ongoing social process.

The vision of a totalizing and logically consistent semantic web, however, has largely faded into obscurity. One developer involved with semantic web technologies (who requested not be named), captured the present situation in their description of a still-active developer mailing list:

> I think that some people are completely detached from practical applications of what they propose. [...] I could not follow half of the messages. these guys seem completely removed from our plane of existence and I have no clue what they are trying to solve.

This division in thought styles generalizes across domains of infrastructure, though outside of the linked data and similar worlds the dichotomy is more frequently between "neatness" and "people doing whatever" – with integration and interoperability becoming nearly synonymous with standardization. Calls for standardization without careful consideration and incorporation of existing practice have a familiar cycle: devise a standard that will solve everything, implement it, wonder why people aren't using it, funding and energy dissipiates, rinse, repeat[5]. The difficulty of scaling an exacting vision of how data should be formatted, the tools researchers should use for their experiments, and so on is that they require dramatic and sometimes total changes to the way people do science. The alternative is not between standardization and chaos, but a potential third way is designing infrastructures that allow the diversity of approaches, tools, and techniques to be expressed in a common framework or protocol along with the community infrastructure to allow the continual negotiation of their relationship.

[4] This isn't a story of "good people" and "bad people," as a lot of the linked data technology also serves as the backbone for abusive technology monopolies like google's acquisition of Freebase [250] and the profusion of knowledge graph-based medical platforms.

[5] There is, of course, an XKCD for that to which we make obligatory reference: https://xkcd.com/927/

10.2.5 Taped-on Interfaces: Open-Loop User Testing

The point of most active competition in many domains of commercial software is the user interface and experience (UI/UX). To compete, software companies will exhaustively user-test and refine them with pixel precision to avoid any potential customer feeling even a thimbleful of frustration. Scientific software development is largely disconnected from usability testing, as what little support exists is rarely tied to it. This, combined with the preponderance of semi-amateurs and above incentives for developing new packages – and thus reduplicating the work of interface development – make it perhaps unsurprising that most scientific software is hard to use!

I intend the notion of "interface" in an expansive way: In addition to a graphical user interface (GUI) or set of functions and calling conventions exposed to the end-user, I am referring generally to all points of contact with users, developers, and other software. Interfaces are intrinsically social, and include the surrounding documentation and experience of use — part of using software is being able to figure out how to use it! The favored design idiom of scientific software is the black box: I implemented an algorithm of some kind, here are the two or three functions needed to use it, but beneath the surface there be dragons.

Ideally, software would be designed with developer interfaces and documentation at multiple scales of complexity to enable clean entrypoints for developers with differing levels of skill and investment to contribute. When this kind of design and documentation is underdeveloped, even widely used projects with excellent top-level interfaces like poetry struggle to respond to the pile of issues thousands deep as even users who have spent time reading the source have difficulty understanding what exactly needs to be fixed and maintainers have to spend their time triaging them and manually re-explaining the software hundreds of times[6].

Additionally, it would include interfaces for use and integration with other software — or APIs. While the term "API" most commonly refers to web APIs, the term generally refers to the means by which other programs can interact with a given program. All programs have some limit to their function, the question is how other programs are expected to handle them. One particularly successful approach to program interface design is the Unix philosophy as articulated by Doug McIlroy and colleagues [251] — which was originally designed to help build research software. Its first "make each program do one thing well" and second "expect the output of every program to become the input to another, as yet unknown, program" principles inspired a set of simple tools that can be composed together for complex tasks. When a program is monolithic and isn't designed to provide access to its component parts, it becomes difficult to reuse in downstream projects, potentially reskin with a more friendly user interface, and ultimately more likely to be a dead-end in a system of shared infrastructure.

Without care given to any of these types of interfaces, the barrier to use is likely to remain high, the community of co-developers is likely to remain small, and the labor they expend is less likely to be useful outside that single project. This, in turn, closes the loop with incentives to develop new packages and makes another vicious cycle reinforcing fragmentation[7].

[6] For one example of many, see Issue #3855, where several users try to make sense of the way poetry resolves packages from multiple sources — a conversation that has been happening for more than a year at the time of writing across multiple related issues.

[7] Incentivized to develop new packages -> need to reinvent interfaces -> hard to develop and extend -> incentivized to develop new packages

10.2.6 Platforms, Industry Capture, and the Profit Motive

Publicly funded science is an always-irresistable golden goose for private industry. The fragmented interests of scientists and the historically light touch of funding agencies on encroaching privatization means that if some company manages to capture and privatize a corner of scientific practice they are likely to keep it. Industry capture has been thoroughly criticized in the context of the journal system (eg. recently, [233]), and that criticism should extend to the rest of our infrastructure as information companies seek to build a for-profit platform system that spans the scientific workflow (eg. [252]). The mode of privatization of scientific infrastructure follows the broader software market as a proliferation of software as a service (SaaS), from startups to international megacorporations, that rent access to some, typically proprietary software without selling the software itself.

While in isolation SaaS can make individual components of the infrastructural landscape easier to access — and even free!!* — the business model is fundamentally incompatible with integrated and accessible infrastructure. The SaaS model derives revenue from subscription or use costs, often operating as "freemium" models that make some subset of its services available for free. Even in freemium models, though, the business model requires that some functionality of the platform is enclosed and proprietary. To keep the particular domain of enclosure viable as a profit stream, the proprietor needs to actively defend against competitors as well as any technology that might fill the need for the proprietary technology[8] (See a more thorough treatment of platform capitalism in science in [213])

[8] eg. see the complaint in State of Texas et al. v. Google that alleges Google rigs ad markets designed to lessen its dominance and uses its control over Chrome and Android to create a single, always-on tracking ecosystem owned only by them [253]

As isolated services, one can imagine the practice of science devolving along a similar path as the increasingly-fragmented streaming video market: to do my work I need to subscribe to a data storage service, a cloud computing service, a platform to host my experiments, etc. For larger software platforms, however, vertical integration of multiple complementary services makes their impact on infrastructure more insidious. Locking users into more and more services makes for more and more revenue, which encourages platforms to be as mutually incompatible as they can get away with [254] . To encourage adoption, platforms that can offer multiple services may offer one of the services – say, data storage – for free, forcing the user to use the adjoining services – say, a cloud computing platform.

Since these platforms are often subsidiaries of information industry monopolists, scientists become complicit in their often profoundly unethical behavior of by funneling millions of dollars into them. Longterm, unconditional funding of wildly profitable journals has allowed conglomerates like Elsevier to become sprawling surveillance companies [255, 214] that are sucking as much data up as they can to market derivative products like algorithmic ranking of scientific productivity [222] and making data sharing agreements with ICE [226] . Or our reliance on AWS and the laundry list of human rights abuses by Amazon [256] . In addition to lock-in, dependence on a constellation of SaaS allows the opportunity for platform-holders to take advantage of their limitations and *sell us additional services to make up for what the other ones purposely lack* — for example Elsevier has taken advantage of our dependence on the journal system and its strategic disorganization to sell a tool for summarizing trending research areas for tailoring maximally-fundable grants [257] .

Funding models and incentive structures in science are uniformly aligned towards the platformatization of scientific infrastructure. Aside from the corporate dou-

blespeak "technology transfer" rhetoric that pervades the neoliberal university, the relative absence of major funding opportunities for scientific software developers competitive with the profit potential from "industry" often leaves it as the only viable career path. The preceding structural constraints on local infrastructural development strongly incentivize labs and researchers to rely on SaaS that provides a readymade solution to specific problems. Distressingly, rather than supporting infrastructural development that would avoid obligate payments to platform-holders, funding agencies seem all too happy to lean into them (emphases mine):

> NIH will **leverage what is available in the private sector,** either through strategic partnerships or procurement, to create a workable **Platform as a Service (PaaS)** environment. [...] NIH will partner with cloud-service providers for cloud storage, computational, and related infrastructure services needed to facilitate the deposit, storage, and access to large, high-value NIH datasets. [...]
>
> NIH's cloud-marketplace initiative will be the first step in a phased operational framework that **establishes a SaaS paradigm for NIH and its stakeholders.** (-NIH Strategic Plan for Data Science, 2018 [243])

The articulated plan being to pay platform holders to house data while also paying for the labor to maintain those databases veers into parody, haplessly building another triple-pay industry [258] into the economic system of science — one can hardly wait until they have the opportunity to rent their own data back with a monthly subscription. This isn't a metaphor: the STRIDES program, with the official subdomain cloud.nih.gov, has been authorized to pay $85 million to cloud providers since 2018. In exchange, NIH hasn't received any sort of new technology, but "extramural" scientists receive a maximum discount of 25% on cloud storage and "data egress" fees as well as plenty of training on how to give control of the scientific process to platform giants [259] [9]. Without exaggeration, we are paying them to let us pay for something that makes it so we need to pay them more later.

It is unclear to me whether this is the result of the cultural hegemony of platform capitalism narrowing the space of imaginable infrastructures, industry capture of the decision-making process, or both, but the effect is the same in any case.

[9] Their success stories tell the story of platform non-integration where scientists have to handbuild new tools to manage their data across multiple cloud environments: "We have been storing data in both cloud environments because we wanted the ecosystem we are creating to work on both clouds" [260]

10.2.7 Protection of Institutional and Economic Power

Aside from information industries, infrastructural deficits are certainly not without beneficiaries within science — those that have already accrued power and status.

Structurally, the adoption of SaaS on a wide scale necessarily sacrifices the goals of an integrated mass infrastructure as the practice of research is carved into small, marketable chunks within vertically integrated technology platforms. Worse, it stands to amplify, rather than reduce, inequities in science, as the labs and institutes that are able to afford the tolls between each of the weigh stations of infrastructure are able to operate more efficiently — one of many positive feedback loops of inequity.

More generally, incentives across infrastructures are often misaligned across strata of power and wealth. Those at the top of a power hierarchy have every incentive to maintain the fragmentation that prevents people from competing — hopefully

mostly unconsciously via uncritically participating in the system rather than maliciously reinforcing it.

This poses an organizational problem: the kind of infrastructure that unwinds platform ownership is not only unprofitable, it's **anti-profitable** – making it impossible to profit from its domain of use. That makes it difficult to rally the kind of development and lobbying resources that profitable technology can, requiring organization based on ethical principles and a commitment to sacrifice control in order to serve a practical need.

The problem is not insurmountable, and there are strategic advantages to decentralized infrastructure and its development within science. Centralized technologies and companies might have more concerted power, but we have *numbers* and can make tools that let us combine small amounts of labor from many people. We often start (and end) our dreams of infrastructure with the belief that they will necessarily cost a lot of *money,* but that's propaganda. Of course development isn't *free,* but the cost of decentralized technologies is far smaller than the vast sums of money funnelled into industry profits, labor hours spent compensating for the designed inefficiencies of the platform model, and the development of a fragmented tool ecosystem built around them.

Science, as one of few domains of non-economic labor, has the opportunity to be a seed for decentralized technologies that could broadly improve not only the health of scientific practice, but the broader information ecosystem. We can develop a plan and mobilize to make use of our collective expertise to build tools that have no business model and no means of development in commercial domains — we just need to realize what's at stake and agree that the health of science is more important than the convenience of the cloud[10] or which journal our papers go into.

[10] Though the system of engineered helpless that convinces us that we're incapable of managing our own web infrastructure is not actually as reliable and seamless as it claims, as the long history of dramatic outages at AWS can show us [261, 262]

10.3 The Ivies, Institutes, and "The Rest of Us"

Given these constraints, who can build new digital infrastructure? Constraints, motivations, and strategies all depend on the circumstance of those doing the development. The undone work of infrastructure is being nibbled at around the edges[11] by several different kinds of organization already ranging in scale and structure. A short survey to give us some notion of how we should seek to organize infrastructure building:

[11] aka doing hard development work in sometimes adverse conditions.

10.3.1 Institutional Core Facilities

Centralized "core" facilities are maybe the most typical form of infrastructure development and resource sharing at the level of departments and institutions. These facilities can range from minimal to baroque extravagance depending on institutional resources and whatever complex web of local history brought them about.

A subproject within a PNI Systems Core grant echoes a lot of the thoughts here, particularly regarding effort duplication[12]:

> Creating an Optical Instrumentation Core will address the problem that much of the technical work required to innovate and maintain these instruments has shifted to students and postdocs, because it has exceeded the capacity of existing

[12] Thanks a lot to the one-and-only stunning and brilliant Dr. Eartha Mae Guthman for suggesting looking at the BRAIN initiative grants as a way of getting insight on core facilities.

> staff. This division of labor is a problem for four reasons: (1) lab personnel often do not have sufficient time or expertise to produce the best possible results, (2) the diffusion of responsibility leads people to duplicate one another's efforts, (3) researchers spend their time on technical work at the expense of doing science, and (4) expertise can be lost as students and postdocs move on. For all these reasons, we propose to standardize this function across projects to improve quality control and efficiency. Centralizing the design, construction, maintenance, and support of these instruments will increase the efficiency and rigor of our microscopy experiments, while freeing lab personnel to focus on designing experiments and collecting data.

While core facilities are an excellent way of expanding access, reducing redundancy, and standardizing tools *within* an instutition, as commonly structured they can displace work spent on efforts that would be portable *outside* of the institution. Elite institutions can attract the researchers with the technical knowledge to develop the instrumentation of the core and infrastructure for maintaining it, but this development is only occasionally made usable by the broader public. The Princeton data science core is an excellent example of a core facility that does makes its software infrastructure development public[13], which they should be applauded for, but also illustrative of the problems with a core-focused infrastructure project. For an external user, the documentation and tutorials are incomplete – it's not clear to me how one would set this up for my institute, lab, or data, and there are several places of hard-coded princeton-specific values that I am unsure how exactly to adapt[14]. I would consider this example a high-water mark, and the median openness of core infrastructure falls far below it. I was unable to find an example of a core facility that maintained publicly-accessible documentation on the construction and operation of its experimental infrastructure or the management of its facility.

This might be unsurprising given the economic structure of most core facilities: an institution pays for a core to benefit the institution, and downstream public benefits are a nice plus but not high up in the list of concerns (if present at all). Core facilities are thus unlikely to serve as the source of mass infrastructure, but they do serve as a point of local coordination within institutions, and so given some larger means of coordination may still be useful.

[13] Project Summary: Core 2, Data Science [...] In addition, the Core will build a data science platform that stores behavior, neural activity, and neural connectivity in a relational database that is queried by the DataJoint language. [...] This data-science platform will facilitate collaborative analysis of datasets by multiple researchers within the project, and make the analyses reproducible and extensible by other researchers. [...] https://projectreporter.nih.gov/project_info_description.cfm?aid=944412

[14] Though again, this project is examplary, built by friends, and would be an excellent place to start extending towards global infrastructure.

10.3.2 Centralized Institutes

Outside of universities, the Allen Brain Institute is perhaps the most impactful reflection of centralization in neuroscience. The Allen Institute has, in an impressively short period of time, created several transformative tools and datasets, including its well-known atlases [263] and the first iteration of its Observatory project which makes a massive, high-quality calcium imaging dataset of visual cortical activity available for public use. They also develop and maintain software tools like their SDK and Brain Modeling Toolkit (BMTK), as well as a collection of hardware schematics used in their experiments. The contribution of the Allen Institute to basic neuroscientific infrastructure is so great that, anecdotally, when talking about scientific infrastructure it's not uncommon for me to hear something along the lines of "I thought the Allen was doing that."

Though the Allen Institute is an excellent model for scale at the level of a single organization, its centralized, hierarchical structure cannot (and does not attempt to)

serve as the backbone for all neuroscientific infrastructure. Performing single (or a small number of, as in its also-admirable OpenScope Project) carefully controlled experiments a huge number of times is an important means of studying constrained problems, but is complementary with the diversity of research questions, model organisms, and methods present in the broader neuroscientific community.

Christof Koch, its director, describes the challenge of centrally organizing a large number of researchers:

> Our biggest institutional challenge is organizational: assembling, managing, enabling and motivating large teams of diverse scientists, engineers and technicians to operate in a highly synergistic manner in pursuit of a few basic science goals [264]
>
> These challenges grow as the size of the team grows. Our anecdotal evidence suggests that above a hundred members, group cohesion appears to become weaker with the appearance of semi-autonomous cliques and sub-groups. This may relate to the postulated limit on the number of meaningful social interactions humans can sustain given the size of their brain [265]

These institutes too are certainly helpful in building core technologies for the field, but they aren't necessarily organized for developing mass-scale infrastructure. They reflect the capabilities and needs of the institute itself, which are likely to be radically different than a small lab. They can build technologies on a background of expensive cloud storage and computation and rely on a team of engineers to implement and maintain them. So while the tools they make are certainly *useful* we shouldn't count on them to build the systems we need for scientists at large.

10.3.3 Meso-scale collaborations

Given the diminishing returns to scale for centralized organizations, many have called for smaller, "meso-scale" collaborations and consortia that combine the efforts of multiple labs [266] . The most successful consortium of this kind has been the International Brain Laboratory [267, 236] , a group of 22 labs spread across six countries. They have been able to realize the promise of big team neuroscience, setting a new standard for performing reproducible experiments across many labs [268] and developing data management infrastructure to match [269] [15]. Their project thus serves as the benchmark for large-scale collaboration and a model from which all similar efforts should learn from.

[15] Seriously, don't miss their extremely impressive data portal.

Critical to the IBL's success was its adoption of a flat, non-hierarchical organizational structure, as described by Lauren E. Wool:

> IBL's virtual environment has grown to accommodate a diversity of scientific activity, and is supported by a flexible, 'flattened' hierarchy that emphasizes horizontal relationships over vertical management. [...] Small teams of IBL members collaborate on projects in Working Groups (WGs), which are defined around particular specializations and milestones and coordinated jointly by a chair and associate chair (typically a PI and researcher, respectively). All WG chairs sit on the Executive Board to propagate decisions across WGs, facilitate operational and financial support, and prepare proposals for voting by the General Assembly, which represents all PIs. [236]

They should also be credited with their adoption of a form of consensus decision-making, sociocracy, rather than a majority-vote or top-down decisionmaking structure. Consensus decision-making systems are derived from those developed by Quakers and some Native American nations, and emphasize collective consent rather than the will of the majority.

The infrastructure developed by the IBL is impressive, but its focus on a single experiment makes it difficult to expand and translate to widescale use. The hardware for the IBL experimental apparatus is exceptionally well-documented, with a complete and detailed build guide and library of CAD parts, but the documentation is not modularized such that it might facilitate use in other projects, remixed, or repurposed. The experimental software is similarly single-purpose, a chimeric combination of Bonsai [270] and PyBpod scripts. It unfortunately lacks the API-level documentation that would facilitate use and modification by other developers, so it is unclear to me, for example, how I would use the experimental apparatus in a different task with perhaps slightly different hardware, or how I would then contribute that back to the library. The experimental software, according to the PDF documentation, will also not work without a connection to an alyx database. While alyx was intended for use outside the IBL, it still has IBL-specific and task-specific values in its source-code, and makes community development difficult with a similar lack of API-level documentation and requirement that users edit the library itself, rather than temporary user files, in order to use it outside the IBL.

My intention is not to denigrate the excellent tools built by the IBL, nor their inspiring realization of meso-scale collaboration, but to illustrate a problem that I see as an extension of that discussed in the context of core facilities — designing infrastructure for one task, or one group in particular makes it much less likely to be portable to other tasks and groups. This argument is much more contingent on the specific circumstances of the consortium than the prior arguments about core facilities and institutes: when organized with mass-infrastructure in mind, collaborations between semi-autonomous groups across institutions could be a powerful mode of tool development.

It is also unclear how replicable these consortia are, and whether they challenge, rather than reinforce technical inequity in science. Participating in consortia systems like the IBL requires that labs have additional funding for labor hours spent on work for the consortium, and in the case of graduate students and postdocs, that time can conflict with work on their degrees or personal research which are still far more potent instruments of "remaining employed in science" than collaboration. In the case that only the most well-funded labs and institutions realize the benefits of big team science without explicit consideration given to scientific equity, mesoscale collaborations could have the unintended consequence of magnifying the skewed distribution of access to technical expertise and instrumentation.

The central lesson of the IBL, in my opinion, is that governance matters. Even if a consortium of labs were to form explicitly to build mass-scale digital infrastructure, without a formal system to ensure contributors felt heard and empowered to shape the project it would soon become unfocused or unsustainable. Even if this system is not perfect, with some labor still falling unequally on some researchers, it is a promising model for future collaborative consortia.

10.3.4 The rest of us...

Outside of ivies with rich core facilities, institutes like the Allen, or nascent multi-lab consortia, the rest of us are largely on our own, piecing together what we can from proprietary and open source technology. The world of open source scientific software has plenty of energy and lots of excellent work is always being done, though constrained by the circumstances of its development described briefly above. Anything else comes down to whatever we can afford with remaining grant money, scrape together from local knowledge, methods sections, begging, borrowing, and (hopefully not too much) stealing from neighboring labs.

The state of broader scientific deinfrastructuring is perhaps to be expected given our relationship to informational monopolies that in some part depend on it, but unlike many other industries or professions there is reason for hope in science. Science is packed with people with an enormous diversity of skills, resources, and perspectives. Publicly funded science is relatively unique as a labor system that does not strictly depend on profit. There is widespread discontent with the systems of scientific practice, and so the question becomes how we can organize our skill, labor, and energy to rebuild the systems that constrain us.

A third option from the standardization offered by centralization and the blooming, buzzing, beautiful chaos of disconnected open-source development is that of decentralized systems, and with them we might build the means by which the "rest of us" can mutually benefit by organizing our knowledge and labor.

We don't need to wait for permission from a memo from a funding body or the founding of some new organization. We do have to recognize that while we might have very different roles to play, we are all responsible for the state of digital scientific infrastructure. We should take courage and purpose in knowing that we are not alone, and that our problems are just one reflection of the model of digital enclosure and surveillance that defines the information economy. There is no need for distance or animosity with the other modes of organization described above, as if what we intend to build is truly useful to *everyone* except those that profit from its absence, then that certainly includes them. Shunting the vision of a better future onto some as-yet formed effort is precisely the trap we should avoid: our existing organizations *should* be a part of the work of rebuilding our infrastructure precisely because we should be reconsidering the ways that *we, ourselves* work. Seeing a subscription to this platform monopolist's cloud, or that knowledge baron's prestige hierarchy as not being a value-neutral decision begs an alternative from people, labs, and institutions alike. The diversity in what that means for different groups is a *strength,* not a weakness, but it does require some shared vision and notion of how to get there. The rest of the paper is an attempt to draft one.

11

A Draft of Decentralized Scientific Infrastructure

What should we build?

The infrastructural systems I will describe here are similar to previous notions of "grass-roots" science articulated within systems neuroscience [266] , "small tech" [271] or the anti software software club's manifesto [272] in the web development world , and shares some of the motivations of the Solid project [273] , but ultimately draws from a set of ideas with broad and deep history in many domains of computing. My intention is to provide a more prescriptive scaffolding for their design and implementation as a way of painting a picture of what science could be like. This sketch is not intended to be final, but a starting point for further negotiation and refinement.

Throughout this section, when I am referring to any particular piece of software I want to be clear that I don't intend to be dogmatically advocating that software *in particular*, but software *like it* that *shares its qualities* — no snake oil is sold in this document. Similarly, when I describe limitations of existing tools, without exception I am describing a tool or platform I love, have learned from, and think is valuable — learning from something can mean drawing respectful contrast! Many of these technologies have long and torrid social histories, and so when invoked as examples I don't necessarily mean to import along with them all the unmentioned baggage that might accompany them[1].

[1] As one example, while I will write about linked data, I don't necessarily mean it in precisely the original instantiation as an irrevocable URI/RDF/SPARQL-only web, but do draw on its triplet link structure.

11.1 Design Principles

I won't attempt to derive a definition of decentralized systems from first principles here, but from the constraints described above, some design principles that illustrate the idea emerge naturally. For the sake of concreteness, in some of these I will draw from the architectural principles of the internet protocols (specifically TCP/IP): the most successful decentralized digital technology project to date.

11.1.1 Protocols, not Platforms

Much of the basic technology of the internet was developed as protocols that describe the basic attributes and operations of a process. A simple and common example is email over SMTP (Simple Mail Transfer Protocol) [274] . SMTP describes a series of steps that email servers must follow to send a message: the sender initiates a connection to the recipient server, the recipient server acknowledges the connection, a few more handshake steps ensue to describe the senders and receivers of the message, and then the data of the message is transferred. Any software that implements the protocol can send emails to and from any other. The protocol basis of email is the reason why it is possible to send an email from a gmail account to a hotmail account (or any other hacky homebrew SMTP client) despite being wholly different pieces of software.

In contrast, *platforms* provide some service with a specific body of code usually with-

out any pretense of generality. In contrast to email over SMTP, we have grown accustomed to not being able to send a message to someone using Telegram from WhatsApp, switching between multiple mutually incompatible apps that serve nearly identical purposes. Platforms, despite being *theoretically* more limited than associated protocols, are attractive for many reasons: they provide funding and administrative agencies a single point of contracting and liability, they typically provide a much more polished user interface, and so on. These benefits are short-lived, however, as the inevitable toll of lock-in and shadowy business models is realized.

By virtue of being intended for use by many independent organizations rather than under the sole control of a platform-holder, protocols are a complicated political effort that embed and facilitate systems of belief and power (see re: TCP/IP [249] , ActivityPub [275]). For example, in order to arrive at a version of TCP/IP that kept the intermediate relays relatively simple at the expense of reliability, the manufacturer of the "smart" relays had to be excluded from the group. TCP/IP's success was not inevitable: it was one of several protocols, becoming the default over proprietary competitors from telecommunication and network hardware companies because of some combination of timing, its relative absence of bureaucracy, and institutional adoption (depending on who does the accounting)[249] .

Seemingly prosocial protocols can be used by industries to pre-empt an alternative that would undermine their profit model — a notable example for academics being the DOI system, created in order for publishers to preserve control over their intellectual property [276] . The STM association[2] hastily[3] threw its weight behind the DOI-X initiative at its 1999 meeting. The impending creation of PubMed Central by the National Library of Medicine (and see then-NIH Director Harold Varmus' and others self-described "radical" departure from publishers with what became PLoS [218, 217]) posed an existential threat to for-profit publishing. At the time there was no unified means of linking to scholarly work[4], and bilateral publisher-publisher linking deals threatened the smooth operation of business, so an NIH-owned platform might have made journals might lose their status as the obligate dissemination platform. According to Bob Campbell, STM chair at the time: "our consensus was that publishers should be the ones doing the linking." Unlike the anarchic URI/URL, The DOI system requires a registrar (denoted by the prefix before the slash, `doi:10.xxxx/yyyyy`) to create DOI names [278] . In the US, that means being an institution with an approved CrossRef membership, which requires members not to link to intellectual property infringing content, and to use DOIs as their default reference links to other works. Effectively, though it is an "open[5]" standard, the DOI system ensures that publishers remain in control of what counts as scholarly work [277] .

When approaching protocols, we should do so with humility and caution: work in smaller teams with shared visions with the intention of rough consensus around multiple instances of working code. We should refuse participation by the wide range of industries and interest groups circling each domain of infrastructure, their protocols and standards are siren songs.

11.1.2 Integration, not Invention

At the advent of the internet protocols, several different institutions and universities had already developed existing network infrastructures, and so the "top level goal" of IP was to "develop an effective technique for multiplex utilization of exist-

[2] The global trade association of publishers that serves as its lobbying and propaganda arm.

[3] The description provided by the "official" CrossRef 10 year retrospective paints a picture of panicked executives making an announcement for something they weren't quite sure what it would be, but it would be *something* to compete with pubmed:

> We decided to issue an announcement of a broad STM reference linking initiative. It was, of course, a strategic move only, since we had neither plan nor prototype."
>
> A small group led by Arnoud de Kemp of Springer-Verlag met in an adjacent room immediately following the Board meeting to draft the announcement, which was distributed to all attendees of the STM annual meeting the following day and published in an STM membership publication.
>
> Campbell recalled running into Bolman and Swanson (neither of whom was then on the STM Board) in the hotel lobby immediately after the drafting of the announcement. Their astonishment at hearing what had just transpired was matched by Campbell's own on learning what they had been working on. [...]
>
> Bolman and Swanson chose to seize the moment, and called an ad hoc meeting the following evening, Tuesday, October 12, to announce their venture and assemble a coalition of publishers to launch it. [...]
>
> The potential benefit of the service that would become CrossRef was immediately apparent. Organizations such as AIP and IOP (Institute of Physics) had begun to link to each other's publications, and the impossibility of replicating such one-off arrangements across the industry was obvious. As Tim Ingoldsby later put it, "All those linking agreements were going to kill us." [277]

[4] It is hard to appreciate in retrospect how radical URLs/URIs were at the time — it might seem trivial to us now to be able to arbitrarily link to different locations on the internet, but before the internet linking was a carefully controlled process

ing interconnected networks," and "come to grips with the problem of integrating a number of separately administered entities into a common utility" [279] . As a result, IP was developed as a 'common language' that could be implemented on any hardware, and upon which other, more complex tools could be built. This is also a cultural practice: when the system doesn't meet some need, one should try to extend it rather than building a new, separate system — and if a new system is needed, it should be interoperable with those that exist.

This point is practical as well as tactical: to compete, an emerging protocol should integrate or be capable of bridging with the technologies that currently fill its role. A new database protocol should be capable of reading and writing existing databases, a new format should be able to ingest and export to existing formats, and so on. The degree to which switching is seamless is the degree to which people will be willing to switch.

This principle runs directly contrary to the current incentives for novelty and fragmentation and the dominant economic model of software platforms, which must be counterbalanced by design choices elsewhere.

11.1.3 Embrace Heterogeneity, Be Uncoercive

In addition to integrating with existing systems, it must be straightforward for unanticipated future development to be integrated to accomodate unanticipated needs and practices. This idea is related to "the test of independent invention", summarized with the question "if someone else had already invented your system, would theirs work with yours?" [280] . Rather than attempting to *a priori* divine a single perfect universal protocol, we should design multiple with extensibility in mind (see this discussion of the extensibility models of ActivityPub to XMPP [281] and Christopher Yoo's description of the tradeoffs of the internet's layered protocols [282]) to leave open the opportunity for porting functionality between them.

This principle also has tactical elements. An uncoercive system allows users to gradually adopt it rather than needing to adopt all of its components in order for any one of them to be useful. We shouldn't rely on potential users making dramatic changes to their existing practices. For example, an experimental framework should not insist on a prescribed set of supported hardware and rigid formulation for describing experiments. Instead it should provide affordances that give a clear way for users to extend the system to fit their needs [283] .There always needs to be a *benefit* to adopting further components of the system to encourage *voluntary* adoption, but it should never be *compulsory*. For example, again from experimental frameworks, it should be possible to use it to control experimental hardware without needing to use the rest of the experimental design, data storage, and interface system. To some degree this is accomplished with a modular system design where designers are mindful of keeping the individual modules independently useful.

A noncoercive architecture also prioritizes the ease of leaving. Though this is somewhat tautological to protocol-driven design, specific care must be taken to enable export and migration to new systems. Multiplicity of design and making leaving easy help ensure that early missteps in development of the system are not fatal, preventing lock-in to a component that becomes fixed and stagnant.

11.1.4 Empower People, not Systems

Because IP was initially developed as a military technology by DARPA, a primary design constraint was survivability in the face of failure. The model adopted by internet architects was to move as much functionality as possible from the network itself to the end-users of the network — rather than the network itself guaranteeing a packet is transmitted, the sending computer will do so by requiring a response from the recipient [279] .

For infrastructure, we should make tools that don't require a central team of developers to maintain, a central server-farm to host data, or a small group of people to govern. Whenever possible, data, software, and hardware should be self-describing[6], so one needs minimal additional tools or resources to understand and use it. It should never be the case that funding drying up for one node in the system causes the entire system to fail.

[6] AKA you shouldn't need to resort to some external source to understand it. Data should come packaged with clear metadata, software should have its own docs, etc.

Practically, this means that the tools of digital infrastructure should be deployable by individual people and be capable of recapitulating the function of the system without reference to any central authority. Researchers need to be given control over the function of infrastructure: from controlling sharing permissions for eg. clinically sensitive data to assurance that their tools aren't spying on them. Formats and standards must be negotiable by the users of a system rather than regulated by a central governance body.

11.1.5 Infrastructure is Social

The alternative to centralized governing and development bodies is to build the tools for community control over infrastructural components. This is perhaps the largest missing piece in current scientific tooling. On one side, decentralized governance is the means by which an infrastructure can be maintained to serve the ever-evolving needs of its users. On the other, a sense of community ownership is what drives people to not only adopt but contribute to the development of an infrastructure. In addition to being a source of all the warm fuzzies of socially affiliative "communityness," any collaborative system needs a way of ensuring that the practice of maintaining, building, and using it is designed to *visibly and tangibly benefit* those that do, rather than be relegated to a cabal of invisible developers and maintainers [284, 285] .

Governance and communication tools also make it possible to realize the infinite variation in application that infrastructures need while keeping them coherent: tools must be built with means of bringing the endless local conversations and modifications of use into a common space where they can become a cumulative sense of shared memory.

I will return to this idea in Archives Need Communities in the context of social dynamics of private bittorrent trackers, as well as propose a set of basic communication and governance tools in Rebuilding Scientific Communication.

11.1.6 Usability Matters

It is not enough to build a technically correct technology and assume it will be adopted or even useful, it must be developed embedded within communities of

practice and *be useful for solving problems that people actually have.* We should learn from the struggles of the semantic web project. Rather than building a fully prescriptive and complete system first and deploying it later, we should develop tools whose usability is continuously improved *en route* to a (flexible) completed vision.

The adage from RFC 1958[7] "nothing gets standardized until there are multiple instances of running code" [283] captures the dual nature of the constraint well. Workable standards don't emerge until they have been extensively tested in the field, but development without an eye to an eventual protocol won't make one.

We should read the gobbling up of open protocols into proprietary platforms that defined "Web 2.0" as instructive[8][286] . *Why* did Slack outcompete IRC?[9] The answer is relatively simple: it was relatively simple to use. Using a contemporary example, to set up a Synapse server to communicate over Matrix one has to wade through dozens of shell commands, system-specific instructions, potential conflicts between dependent packages, set up an SQL server... and that's just the backend, we don't even have a frontend client yet! In contrast, to use Slack you download the app, give it your email, and you're off and running.

The control exerted by centralized systems over their system design does give certain structural advantages to their usability, and their for-profit model gives certain advantages to their development process. There is no reason, however, that decentralized systems *must* be intrinsically harder to use, we just need to focus on user experience to a degree comparable to centralized platforms: if it takes a college degree to turn the water on, that ain't infrastructure.

People are smart, they just get frustrated easily and have other things to do on a deadline. We have to raise our standards of design such that we don't expect users to have even a passing familiarity with programming, attempting to build tools that are truly general use. We can't just design a peer-to-peer system, we need to make the data ingestion and annotation process automatic, effortless, and expressive. We can't just build a system for credit assignment, it needs to happen as an automatic byproduct of using the system. We can't just make tools that *work,* they need to *feel good to use.*

Centralized systems also have intrinsic limitations that provide openings for decentralized systems, like cost, incompatibility with other systems, restrictions on independent extension, and opacity of function. The potential for decentralized systems to capture the independent development labor of all of its users, rather than just that of a core development team, is one means of competition. If a system is sufficiently easy to adopt, at least comparable to prior tooling, and gives people a satisfying means of having their work accepted and valued, the social and technical joy might be enough to outweigh the inertia of change and the convenience of centralized systems.

With these principles in mind, and drawing from other knowledge communities solving similar problems: internet infrastructure, library/information science, peer-to-peer networks, and radical organizing, I conceptualize a system of distributed infrastructure for (neuro)science as three objectives: **shared data**, **shared tools**, and **shared knowledge**.

[7] A "request for comment" from the Network Working Group of the Internet Engineering Task Force on the architecture of the internet. The IETF designs many of the protocols that serve as the backbone of the internet.

[8] (in addition to a demonstration of the raw power of concentrated capital, of course)

[9] IRC, internet relay chat, was a messaging system that served many of the same functions as the group messaging program Slack serves now. Also see its more active cousin XMPP

11.2 Shared Data

11.2.1 Formats as Onramps

The shallowest onramp towards a generalized data infrastructure is to make use of existing discipline-specific standardized data formats. As will be discussed later, a truly universal pandisciplinary format is impossible and undesirable, but to arrive at the alternative we should first congeal the wild west of unstandardized data into a smaller number of established formats.

Data formats consist of some combination of an abstract specification, an implementation in a particular storage medium, and an API for interacting with the format. I won't dwell on the particular qualities that a particular format needs, assuming that most that would be adopted would abide by FAIR principles.

There are a dizzying number of scientific data formats [287] , so a comprehensive treatment is impractical here and I will use Neurodata Without Borders:N (NWB)[288] as an example. NWB is the de facto standard for systems neuroscience, adopted by many institutes and labs, though far from universally. NWB consists of a specification language, a schema written in that language, a storage implementation in hdf5, and an API for interacting with the data. They have done an admirable job of engaging with community needs [289] and making a modular, extensible format ecosystem.

The major point of improvement for NWB, and I imagine many data standards, is the ease of conversion and use. The conversion API requires extensive programming, knowledge of the format, and navigation of several separate tutorial documents. This means that individual labs, if they are lucky enough to have some partially standardized format for the lab, typically need to write (or hire someone to write) their own software library for conversion.

Without being prescriptive about its form, substantial interface development is needed to make mass conversion possible. It's usually untrue that unstandardized data had *no structure,* and researchers are typically able to articulate it – "the filenames have the collection date followed by the subject id," and so on. Lowering the barriers to conversion mean designing tools that match the descriptive style of folk formats, for example by prompting them to describe where each of an available set of metadata fields are located in their data. It is not an impossible goal to imagine a piece of software that can be downloaded and with minimal recourse to reference documentation allow someone to convert their lab's data within an afternoon.

NWB also has an extension interface, which allows, for example, data from common hardware and software tools to be more easily described in the format. These are registered in an extensions catalogue, but at the time of writing it is relatively sparse. The preponderance of lab-specific conversion packages relative to extensions is indicative of an interface and community tools problem: presumably many people are facing similar conversion problems, but because there is not a place to share these techniques in a human-readable way, the effort is duplicated in dispersed codebases. We will return to some possible solutions for knowledge preservation and format extension when we discuss tools for shared knowledge.

For the sake of the rest of the argument, let us assume that some relatively trivial conversion process exists to subdomain-specific data formats and we reach some reasonable penetrance of standardization. The interactions with the other pieces

of infrastructure that may induce and incentivize conversion will come later.

11.2.2 Peer-to-peer as a Backbone

We should adopt a *peer-to-peer* system for storing and sharing scientific data. There are, of course many existing databases for scientific data, ranging from domain-general like figshare and zenodo to the most laser-focused subdiscipline-specific. The notion of a database, like a data standard, is not monolithic. As a simplification, they consist of at least the hardware used for storage, the software implementation of read, write, and query operations, a formatting schema, some API for interacting with it, the rules and regulations that govern its use, and especially in scientific databases some frontend for visual interaction. For now we will focus on the storage software and read-write system, returning to the format, regulations, and interface later.

Centralized servers[10] are fundamentally constrained by their storage capacity and bandwidth, both of which cost money. In order to be free, database maintainers need to constantly raise money from donations or grants in order to pay for both. Funding can never be infinite, and so inevitably there must be some limit on the amount of data that someone can upload and the speed at which it can serve files[11]. Centralized servers are also intrinsically out of the control of their users, requiring them to abide whatever terms of use the server administrators set. Even if the database is carefully backed up, it serves as a single point of infrastructural failure, where if the project lapses then at worst data will be irreversibly lost, and at best a lot of labor needs to be expended to exfiltrate, reformat, and rehost the data. The same is true of isolated, local, institutional-level servers and related database platforms, with the additional problem of skewed funding allocations making them unaffordable for many researchers.

[10] This applies to centrally managed traditional servers as well as rented space on larger CDNs like AWS, but in the case of the CDN the constraint is from their pricing model.

[11] As I am writing this, I am getting a (very unscientific sample of n=1) maximum speed of 5MB/s on the Open Science Framework

Peer-to-peer (p2p) systems solve many of these problems, and I argue are the only type of technology capable of making a database system that can handle the scale of all scientific data. They are also not new for science, used in projects like AcademicTorrents.com [290, 291] or the now defunct BioTorrents [292] . Whether we acknowledge it or not, most scientific work is already available on p2p networks via sci-hub and library genesis [293, 294, 295] .

There is an enormous degree of variation between p2p systems[12], but they share a set of architectural advantages. The essential quality of any p2p system is that rather than each participant in a network interacting only with a single server that hosts all the data, everyone hosts data and interacts directly with each other.

[12] Peer to peer systems are, maybe predictably, a whole academic subdiscipline. See [296] for reference.

For the sake of concreteness, we can consider a (simplified) description of Bittorrent [297] , arguably the most successful p2p protocol. To share a collection of files, a user creates a `.torrent` file with their Bittorrent client which consists of a cryptographic hash, or a string that is unique to the collection of files being shared; and a list of "trackers." A tracker, appropriately, keeps track of the `.torrent` files that have been uploaded to it, and connects users that have or want the content referred to by the `.torrent` file. The uploader (or seeder) then leaves a torrent client open waiting for incoming connections. Someone who wants to download the files (a leecher) will then open the `.torrent` file in their client, which will then ask the tracker for the IP addresses of the other peers who are seeding the file, directly connect to them, and begin downloading. So far so similar to standard client-server systems, but say another person wants to download the same files before the first person has finished

downloading it: rather than *only* downloading from the original seeder, the new leecher downloads from *both* the original seeder and the first leecher by requesting pieces of the file from each until they have the whole thing. Leechers are incentivized to share among each other to prevent the seeders from spending time reuploading the pieces that they already have, and once they have finished downloading they become seeders themselves.

From this very simple example, we can articulate a number of attractive qualities of p2p systems:

- First, p2p systems are extremely **inexpensive to maintain** since they take advantage of the existing bandwidth and storage space of the computers in the swarm. Near the height of its popularity in 2009, The Pirate Bay, a notorious bittorrent tracker [298] , was estimated to cost $3,000 per month to maintain while serving approximately 20 million peers [299] . According to a database dump from 2013 [300] , multiplying the size of each torrent by the number of seeders (ignoring any partial downloads from leechers), the approximate instantaneous amount of data stored by The Pirate Bay was ~26 Petabytes. The comparison to centralized services is not straightforward, since it is hard to evaluate the distributed costs of additional storage media (as well as the costs avoided by being able to take advantage of existing storage infrastructure within labs and institutes), but for the sake of illustration: hosting 26PB would cost $546,000/month with standard AWS S3 hosting ($0.021/GB/month). On AWS, downloads cost extra ($0.05/GB), so the much smaller academictorrents.com which has served nearly 18PB in 1.3m downloads since 2016 would have cost $900,000 in bandwidth costs alone — as opposed to the literally zero dollars it costs to operate.

- The **speed** of a bittorrent swarm *increases,* rather than decreases, the more people are using it since it is capable of using all of the available bandwidth in the system.

- The network is extremely **resilient** since the data is shared across many independent peers in the system. If our goal is to make a resilient and robust data architecture, we would benefit by paying attention to the tools used in the broader archival community, especially the archival communities that are frequent targets of governments and intellectual property holders[301] . Despite more than 15 years of concerted effort by governments and intellectual property holders, The Pirate Bay is still alive and kicking[13] [302] . This is because even if the entire infrastructure of the tracker is destroyed, as it was in 2006, the files are distributed across all of its users, the actual database of `.torrent` metadata is quite small, and the tracker software is extraordinarily simple to rehost [303] – The Pirate Bay was back online in 2 days. When another tracker, what.cd (which we will return to soon) was shut down, a series of successors popped up using the open source tools Gazelle and Ocelot that what.cd developers built. Within two weeks, one successor site had recovered and reindexed 200,000 of its torrents resubmitted by former users [304] . Bittorrent is also used by archival groups with little funding like Archive Team, who struggled – but eventually succeeded – to disseminate their geocities archive over a single "crappy cable modem" [305] .

[13] knock on wood

- The network is extremely **scalable** since there is no cost to connecting new peers and the users of a system expand the storage capacity of the system depending

on their needs. Rather than having one extremely fast data center, the model of p2p systems is to leverage many approachable peer/servers.

Peer-to-peer systems are not mutually exclusive with centralized servers: servers are peers too, after all. A properly implemented p2p system will always be *at least* as fast and have *at least* as much storage as any alternative centralized server because peers can use *both* the bandwidth of the server *and* that of any peers that have the file. In the bittorrent ecosystem large-bandwidth/storage peers are known as "seedboxes"[306] when they use the bittorrent protocol, and "web seeds"[307] when they use a protocol built on top of traditional HTTP. Archive.org has been distributing all of its materials with bittorrent by using its servers as web seeds since 2012 and makes this point explicitly: "BitTorrent is now the fastest way to download items from the Archive, because the Bittorrent client downloads simultaneously from two different Archive servers located in two different datacenters, and from other Archive users who have downloaded these Torrents already." [308]

p2p systems complement centralized servers in a number of ways beyond raw download speed, increasing the efficiency and performance of the network as a whole. Spotify began as a joint client/server and p2p system [309] , where when a listener presses play the central server provides the data until the p2p system locates peers with a cached copy to download from. The central server is able to respond quickly and reliably, and is the server of last resort in the case of rare files that aren't being shared by anyone else in the network. The p2p system alleviates pressure on the central server, improving the performance of the network and reducing server costs.

A peer to peer system is a particularly natural fit for many of the common circumstances and practices in science, where centralized server architectures seem (and prove) awkward and inefficient. Most labs, institutes, or other organized bodies of science have some form of local or institutional storage systems. In the most frequent cases of sharing data within a lab or institute, sending it back and forth to some nationally-centralized server is like walking across the lab by going the long way around the Earth. That's the method invoked by a Dropbox or AWS link, which keeps a time-tested p2p system relevant: walking a flash drive across the lab. The system makes less sense when several people in the same place need to access the same data at the same time, as is frequently the case with multi-lab collaborations, or scientific conferences and workshops. Instead of needing to wait on the 300kb/s conference wifi bandwidth as it's cheese-gratered across every machine, we instead could directly beam it between all computers in range simultaneously, full blast through the decrepit network switch that won't have seen that much excitement in years.

If we take the suggestion of Andrey Andreev et al. and invest in server clusters within institutes [310, 311] , their impact could be multiplied manyfold by fluidly combining them in a p2p swarm. While the NIH might be shy to start up another server farm for all scientific data and prefer to contract with AWS, the rest of us don't have to be. Nervous university administrators concerned about bandwidth costs should also favor p2p systems: instead of needing to serve entire datasets to each person who wants them, the load can be spread out across many institutes naturally based on the use of the file, and sharing the dataset internally would cost nothing at all.

So far I have relied on the Extraordinarily Simplified Bittorrent[14] depiction of a peer to peer system, but there are many improvements and variants that can address different needs for scientific data infrastructure.

[14] ™

One obvious need that bittorrent can't currently support is version control[15], but

[15] Though the Bittorrent V2 protocol specification [312] adopts a Merkle tree data structure which could theoretically support versioned torrents, v2 torrents are still not widely supported.

more recent p2p systems do. IPFS functions like "a single BitTorrent swarm, exchanging objects within one Git repository." [313] [16] Dat [314] , specifically designed for data synchronization and versioning, handles versioning and more. A full description of IPFS is out of scope, and it has plenty of problems [315] , but for now it suffices to say p2p systems can handle version control.

[16] Git, briefly, is a version control system that keeps a history of changes of files (blobs) as a Merkle DAG: files can be updated, and different versions can be branched and reconciled.

Bittorrent swarms are vulnerable to data loss if all the peers seeding a file disconnect (though the tail is longer than typically assumed, see [316]), but this too can be addressed with updated p2p system design. A first-order solution to this problem is a variant of IPFS' notion of 'pinning.' Since backup to lab-level or institutional servers is already commonplace, one peer could be able to 'pin' another and automatically download all the data that they share. This concept could scale to institutes and national infrastructure as scientists can request the datasets they'd like to be saved permanently be pinned.

Another could be something akin to Freenet [317] . Peers could allocate a certain amount of their unused storage space to be used to automatically download, cache, and rehost shards of other datasets. Distributing chunks and encrypting them at rest so the rehoster can't inspect their contents would make it possible to maintain privacy and network availability for sensitive data (see, for example, ERIS). IPFS has an analogous concept – BitSwap – that is makes it into a barter system. Peers who seek to download will have to 'earn' it by finding some chunk of data that the other peers want, download, and share them, though it seems like an empirical question whether or not a barter system works or is necessary.

Solid is a project that almost exactly meets all these needs [318, 273, 319] . Solid allows people to share data in Pods, which let them control access and distribution across storage system with a unified identity system. It is implementation-agnostic, and so can support any peer-to-peer storage and transfer system that complies with its protocol specification.

There are a number of additional requirements for a peer to peer scientific data infrastructure, but even these seemingly very technical problems of versioning and distributed storage show the clear need to consider the structure of the surrounding social system. What control do we give to researchers over the version history of their data? Should people that aren't the originating researcher be able to issue new versions? What structure of distributed/centralized storage works? How should we incentivize sharing of excess storage and resources?

Even before considering additional social systems, a p2p structure in itself implies a different relationship to infrastructure. Scientists always unavoidably make their data available to at least one person: themselves; on at least one computer: theirs, and that computer is usually connected to the internet. With a p2p system that integrates metadata from domain-specific data formats, that's it, that's all, the data is already hosted by merely existing. Dust your palms off: open data achieved. A peer-to-peer backbone for scientific infrastructure realizes the unnecessarily radical notion that our infrastructure can be integrated into our daily practices, rather than existing exogenously as something "out there." It helps us internalize the slyly subversive notion that *we can build it ourselves* instead of renting something out of our control from someone else.

Scientists don't need to reinvent the notion of distributed, community curated data archives from scratch. In addition to scholarly work on the social systems of digital infrastructure, we can learn from communities of practice, and there has been no

more important and impactful decentralized archival project than internet piracy.

11.2.3 Archives Need Communities

Why do hundreds of thousands of people, completely anonymously, with zero compensation, spend their time to do something that is as legally risky as curating pirated cultural archives?

Scholarly work, particularly from Economics, tends to focus on understanding piracy in order to prevent it[320, 321] , taking the moral good of intellectual property markets as an *a priori* imperative and investigating why people behave *badly* and "rend [the] moral fabric associated with the respect of intellectual property." [321] . If we put the legality of piracy aside, we may find a wealth of wisdom and insight to draw from for building scientific infrastructure.

The world of digital piracy is massive, from entirely disorganized efforts of individual people on public sites to extraordinarily organized release groups [320] , and so a full consideration is out of scope (see [322]), but many of the important lessons are taught by the structure of bittorrent trackers.

An underappreciated element of the BitTorrent protocol is the effect of the separation between the data transfer protocol and the "discovery" part of the system — or "overlay" — on the community structure of torrent trackers (for a more complete picture of the ecosystem, see [316]). Many peer to peer networks like KaZaA or the gnutella-based Limewire had searching for files integrated into the transfer interface. The need for torrent trackers to share .torrent files spawned a massive community of private torrent trackers that for decades have been iterating on cultures of archival, experimenting with different community structures and incentives that encourage people to share and annotate some of the world's largest, most organized libraries.

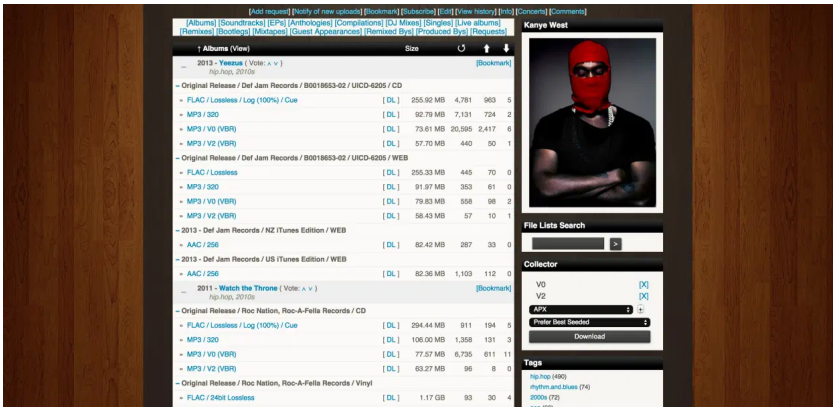
One of these private trackers was the site of one of the largest informational tragedies of the past decade: what.cd[17], which I will use as an example to describe some of these community systems.

[17] for a detailed description of the site and community, see Ian Dunham's dissertation [323]

What.cd was a bittorrent tracker that was arguably the largest collection of music that has ever existed. At the time of its destruction in 2016, it was host to just over one million unique releases, and approximately 3.5 million torrents[18] [323] . Every torrent was organized in a meticulous system of metadata communally curated by its roughly 200,000 global users. The collection was built by people who cared deeply about music, rather than commercial collections provided by record labels notorious for ceasing distribution of recordings that are not commercially viable — or just losing them in a fire [324] . Users would spend large amounts of money to find and digitize extremely rare recordings, many of which were unavailable anywhere else and are now unavailable anywhere, period. One former user describes one example:

[18] Though spotify now boasts its library having 50 million tracks, back of the envelope calculations relating number of releases to number of tracks are fraught, given the long tail of track numbers on albums like classical music anthologies with several hundred tracks on a single "release."

> "I did sound design for a show about Ceaușescu's Romania, and was able to pull together all of this 70s dissident prog-rock and stuff that has never been released on CD, let alone outside of Romania" [325]



The what.cd artist page for Kanye West (taken from here in the style of pirates, without permission). For the album "Yeezus," there are ten torrents, grouped by each time the album was released on CD and Web, and in multiple different qualities and formats (.flac, .mp3). Along the top is a list of the macro-level groups, where what is in view is the "albums" section, there are also sections for bootleg recordings, remixes, live albums, etc.

What.cd was a "private" bittorrent tracker, where unlike public trackers that anyone can access, membership was strictly limited to those who were personally invited or to those who passed an interview (for more on public and private trackers, see [326]). Invites were extremely rare, and the interview process was demanding to the point where extensive guides were written to prepare for them.

The what.cd incentive system was based on a required ratio of data uploaded vs. data downloaded [327] . Peer to peer systems need to overcome a free-rider problem where users might download a torrent ("leeching") and turn their computer off, rather than leaving their connection open to share it to others (or, "seeding"). In order to download additional music, then, one would have to upload more. Since downloading is highly restricted, and everyone is trying to upload as much as they can, torrents had a large number of "seeders," and even rare recordings would be sustained for years, a pattern common to private trackers [328] .

The high seeder/leecher ratio made it so it was extremely difficult to acquire upload credit, so users were additionally incentivized to find and upload new recordings to the system. What.cd implemented a "bounty" system, where users with a large amount of excess upload credit would be able to offer some of it to whoever was able to upload the album they wanted. To "prime the pump" and keep the economy moving, highlight artists in an album of the week, or direct users to preserve rare recordings, moderators would also use a "freeleech" system, where users would be able to download a specified set of torrents without it counting against their download quantity [329, 330] .

The other half of what.cd was the more explicitly social elements: its forums, comment sections, and moderation systems. The forum was home to roiling debates that lasted years about the structure of some tagging schema, whether one genre was just another with a different name, and so on. The structure of the community was an object of constant, public negotiation, and over time the metadata system evolved to be able to support a library of the entirety of human musical culture[19]. To support the good operation of the site, the forums were also home to a huge amount of technical knowledge, like guides on how to make a perfect copy of a CD or how to detect a fake upload, that eased new users into being able to use and con-

[19] Though music metadata might seem like a trivial problem (just look at the fields in an MP3 header), the number of edge cases are profound. How would you categorize an early Madlib casette mixtape remastered and uploaded to his website where he is mumbling to himself while recording some live show performed by multiple artists, but on the b-side is one of his Beat Konducta collections that mix together studio recordings from a collection of other artists? Who is the artist? How would you even identify the unnamed artists in the live show? Is that a compilation or a bootleg? Is it a cassette rip, a remaster, or a web release?

tribute to the system.

A critical problem in maintaining coherent databases is correcting metadata errors and departures from schemas. Finding errors was rewarded. Users were able to discuss and ask questions of the uploader in a comment section below each upload, which would allow "polite" resolution of low-level errors like typos. More serious problems could be reported to the moderation team, which caused the upload to be visibly marked as under review, and the report could then be discussed either in the comment sections or the forum. The system wasn't perfect: being an anonymous, gray-area community, there was of course plenty of power to be abused. Rather than being a messy hodgepodge of fake, low-quality uploads, though, what.cd was always teetering just shy of perfection.

These structural considerations do not capture the most elusive but indisputably important feature of what.cd's community infrastructure: *the sense of commmunity*. The What.cd forums were the center of many user's relationships to music. Threads about all the finest scales of music nichery could last for years: it was a rare place people who probably cared a little bit too much about music could talk to people with the same condition. What made it more satisfying than other music forums was that no matter what music you were talking about, everyone else in the conversation would always have access to it if they wanted to hear it. Beyond any structural incentives, people spent so much time building and maintaining what.cd because it became a source of community and a sink of personal investment.

Structural norms supported by social systems converge as a sort of *reputational* incentive. Uploading a new album to fill a bounty both makes the network more functional and complete, but also *people respect you for it* because it's prominently displayed on your profile as well as in the bounty charts and that *feels good*. Becoming known on the forums for answering questions, writing guides, or even just having a good taste in music *feels good* and also contributes to the overall health of the system. Though there are plenty of databases, and even plenty of different communication venues for scientists, there aren't any databases (to my knowledge) with integrated community systems.

The tracker overlay model mirrors and extends some of the recommendations made by Benedikt Fecher and colleagues in their work on the reputational economy surrounding data sharing [331] . They give three policy recommendations: Increasing reputational benefits, reducing transaction costs, and "increasing market transparency by making open access to research data more visible to members of the research community." One way to accomplish implement them is to embed a data sharing system within a social system that is designed to reward communitarian behavior.

Many features of what.cd's structure are undesirable for scientific infrastructure, but they demonstrate that a robust archive is not only a matter of building a database with some frontend, but also building a community [332] . Of course, we need to be careful with building the structural incentives for a data sharing system: the very last thing we want is another coercive leaderboard that turns what should be a collaborative effort punitive. In contrast to what.cd, for infrastructure we want extremely low barriers to entry, and be agnostic to resources — researchers with access to huge server farms should not be unduly favored. We should think carefully about using downloading as the "cost," because downloading and analyzing huge amounts of data can be *good* and exactly what we *want* in some circumstances, but a threat to

privacy and data governance in others.

This model has its own problems, including the lack of interoperability between different trackers, the need to recreate a new set of accounts and database for each new tracker, among others. It's also been tried before: sharing data in specific formats (as our running example, Neurodata Without Borders) on indexing systems like bittorrent trackers amounts to something like BioTorrents [292] or AcademicTorrents [290] . Even with our extensions of version control and some model of automatic mirroring of data across the network, we still have some work to do. To address these and several other remaining needs for scientific data infrastructure, we can take inspiration from *federated systems.*

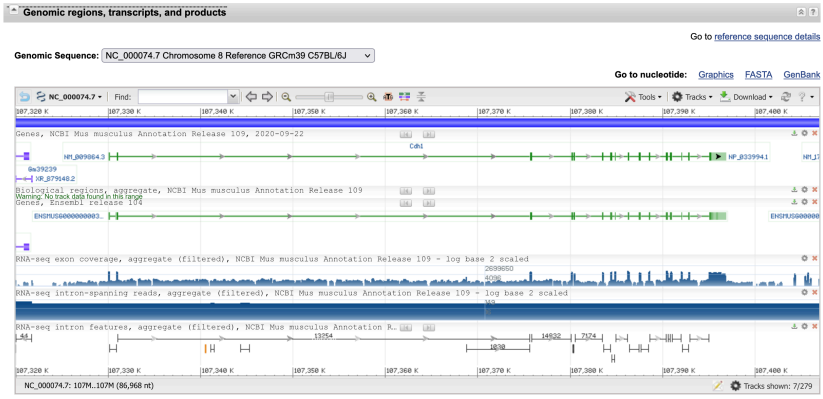
11.2.4 Linked Data or Surveillance Capitalism?

> Having become a dense and consistent historical reality, language forms the locus of tradition, of the unspoken habits of thought, of what lies hidden in a people's mind; it accumulates an ineluctable memory which does not even know itself as memory. Expressing their thoughts in words of which they are not the masters, enclosing them in verbal forms whose historical dimensions they are unaware of, men believe that their speech is their servant and do not realize that they are submitting themselves to its demands.
>
> Michel Foucault — *The Order of Things* [333]

There is no shortage of databases for scientific data, but their traditional structure chokes on the complexity of representing multi-domain data. Typical relational databases require some formal schema to structure the data they contain, which have varying reflections in the APIs used to access them and interfaces built atop them. This broadly polarizes database design into domain-specific and domain-general[20]. This design pattern results in a fragmented landscape of databases with limited interoperability. How shall we link the databases? In this section we'll consider the Icarian promise of creating the great unified database of everything as a way of motivating an alternative that blends *linked data* [334] with *federated systems* against our peer to peer backbone in the next section.

[20] To continue the analogy to bittorrent trackers, an example domain-specific vs. domain-general dichotomy might be What.cd (with its specific formatting and aggregation tools for representing artists, albums, collections, genres, and so on) vs. ThePirateBay (with its general categories of content and otherwise search-based aggregation interface)

Domain-specific databases require data to be in one or a few specific formats, and usually provide richer tools for manipulating and querying by metadata, visualization, summarization, aggregation that are purpose-built for that type of data. For example, NIH's Gene tool has several visualization tools and cross-referencing tools for finding expression pathways, genetic interactions, and related sequences (Figure xx). This pattern of database design is reflected at several different scales, through institutional databases and tools like the Allen brain atlases or observatory, to lab- and project-specific dashboards. This type of database is natural, expressive, and powerful — for the researchers they are designed for. While some of these databases allow open data submission, they often require explicit moderation and approval to maintain the guaranteed consistency of the database, which can hamper mass use.



NIH's Gene tool included many specific tools for visualizing, cross-referencing, and aggregating genetic data. Shown is the "genomic regions, transcripts, and product" plot for Mouse Cdh1, which gives useful, common summary descriptions of the gene, but is not useful for, say, visualizing reading proficiency data.

General-purpose databases like figshare and zenodo[21] are useful for the mass aggregation of data, typically allowing uploads from most people with minimal barriers. Their general function limits the metadata, visualization, and other tools that are offered by domain-specific databases, however, and are essentially public, versioned, folders with a DOI. Most have fields for authorship, research groups, related publications, and a single-dimension keyword or tags system, and so don't programmatically reflect the metadata present in a given dataset.

[21] No shade to Figshare, which, among others, paved the way for open data and are a massively useful thing to have in society.

The dichotomy of fragmented, subdomain-specific databases and general-purpose databases makes combining information from across even extremely similar subdisciplines combinatorically complex and laborious. In the absence of a formal interoperability and indexing protocol between databases, even *finding* the correct subdomain-specific database often comes down to pure luck. It also puts researchers who want to be good data stewards in a difficult position: they can hunt down the appropriate subdomain specific database and risk general obscurity; use a domain-general database and make their work more difficult for themselves and their peers to use; or spend all the time it takes to upload to multiple databases with potentially conflicting demands on format.

What can be done? There are a few naïve answers from standardizing different parts of the process: If we had a universal data format, then interoperability becomes trivial. Conversely, we could make a single ur-database that supports all possible formats and tools.

The notion of a universal database system almost immediately runs aground on the reality that organizing knowledge is intrinsically political. Every subdiscipline has conflicting *representational* needs, will develop different local terminology, allocate differing granularity and develop different groupings and hierarchies for the same phenomena. At their mildest, differences in representational systems can be incompatible, but at their worst they can reflect and reinforce prejudices and become the site of expression for intellectual and social power struggles [335, 336, 337, 338] . Every subdiscipline has conflicting *practical* needs, with infinite variation in privacy demands, different priorities between storage space, bandwidth, and computational power, and so on. In all cases the boundaries of our myopia are impossible to gauge: we might think we have arrived at a suitable schema for biology, chemistry, and physics... but what about the historians?

Matthew J Bietz and Charlotte P Lee articulate this tension in their ethnography of metagenomics databases:

> "Participants describe the individual sequence database systems as if they were shadows, poor representations of a widely-agreed-upon ideal. We find, however, that by looking across the landscape of databases, a different picture emerges. Instead, **each decision about the implementation of a particular database system plants a stake for a community boundary. The databases are not so much imperfect copies of an ideal as they are arguments about what the ideal Database should be.** [...]
>
> In the end, however, **the system was so tailored to a specific set of research questions that the collection of data, the set of tools, and even the social organization of the project had to be significantly changed.** New analysis tools were developed and old tools were discarded. Not only was the database ported to a different technology, the data itself was significantly restructured to fit the new tools and approaches. While the database development projects had begun by working together, in the end they were unable to collaborate. **The system that was supposed to tie these groups together could not be shielded from the controversies that formed the boundaries between the communities of practice.**" [339]

The pursuit of unified representation is an intimate part of the history of linked data, which relies on "ontologies" or controlled vocabularies that describe a set of objects (or classes) and the properties they can have. For example, schema.org maintains a widely used set of hierarchical vocabularies to describe the fundamental things that exist in the world, in particular the unfamiliar world in which a Person has a gender and net worth but lacks a race [247] . At one extreme in the world of ontology builders, the ideological nature of demarcating what is allowed to exist is as clear as a klaxon (emphasis in original):

> An exception is the Open Biomedical Ontologies (OBO) Foundry initiative, which accepts under its label only those ontologies that adhere to the principles of ontological realism. [...] Ontologies, from this perspective, are representational artifacts, comprising a taxonomy as their central backbone, whose representational units are intended to designate *universals* (such as *human being* and *patient role*) or *classes defined in terms of universals* (such as *patient,* a class encompassing *human beings* in which there inheres a *patient role*) and certain relations between them. [...]
>
> BFO is a realist ontology [15,16]. This means, most importantly, that representations faithful to BFO can acknowledge only those entities which exist in (for example, biological) reality; thus they must reject all those types of putative negative entities - lacks, absences, non-existents, possibilia, and the like [340]

In practice, because of the difficulty of changing the representation and encompassing database systems on a dime, using these ontologies to link disparate datasets tends to follow the pattern of metadata *overlays* where the structure of individual databases are mapped onto one "unifying" ontology to allow for aggregation and translation. This approach appears gentler than standardization at the level of individual databases, but has the same problems kicked up one level of abstraction.

To concretize the problems with a globally unified database or metadata overlay, the remainder of this section will trace the compromises and outcomes of the The NIH's "Biomedical Data Translator" project. The Translator project was initially described in the 2016 Strategic Plan for Data Science as a means of translating between biomedical data formats:

> Through its Biomedical Data Translator program, the National Center for Advancing Translational Sciences (NCATS) is supporting research to develop ways to connect conventionally separated data types to one another to make them more useful for researchers and the public. [243]

The original funding statement from 2016 is similarly humble, and press releases through 2017 also speak mostly in terms of querying the data – though some ambition begins to creep in. By 2019, the vision for the project had veered sharply away from anything a basic researcher might recognize as a means of translating between data types. In their piece "Toward a Universal Biomedical Translator," then in a feasibility assessment phase, the members of the Translator Consortium assert that universal translation between biomedical data is impossible [341] . The impossibility they saw was not that of conflicting political demands on the structure of organization (as per [338]), but of the sheer numeracy of the data and vocabularies needed to describe them. The risk posed by a lack of a universal "language" was not being able to index all possible data, rather than inaccuracy or inequity.

Undaunted by their stated belief in the impossibility of a universalizing ontology, the Consortium created one in their biolink model [342, 343] . Biolink consists of a hierarchy of basic classes: eg. a BiologicalEntity like a Gene, or a ChemicalEntity like a Drug. Classes can then linked by any number of properties, or "Slots," like a therapeutic procedure that treats a disease.

The translator does not attempt to respond to the needs of researchers or labs who might want to link their raw data splayed out across flash drives and file structures whose chaos borders on whimsy. Instead, the Translator operates at the level of "knowledge," or "generally accepted, universal assertions derived from the accumulation of information" [344] . Rather than translating *between data types*, the meaning of "translation" shifted to meaning *"translating data into knowledge"* [341] .

To feed the Translator, Biolink sits "on top of" a collection of database APIs that serve structured biomedical data, each called a "knowledge source." Individual APIs declare that they are able to provide data for a particular set of classes or slots, like drugs that affect genetic expression, and are then made browsable from the SmartAPI Knowledge Graph. Queries to individual APIs do not return "raw" data, but return assertions of fact in the parlance of the Biolink model: this procedure treats that disease, etc.

Because individual researchers do not typically represent their data in the form of factual assertions, knowledge sources are constrained to "highly curated biomedical databases" or other aggregated systems. The NIH RePORTER tool gives an

overview of the way these knowledge sources are prepared when none already exist for a given Biolink class or predicate: automated text mining tools and a series of domain-specific data provider projects, rather than via tools provided to researchers.

The collection of knowledge sources, linked to nodes and edges in the Biolink model, are designed to be queried as a graph. To answer a query like "what drug treats this disease?" the translator considers the graph of entities linked to the disease: what symptoms does the disease have? what genes are linked to those symptoms? which drugs act on those genes? and so on [345] . The form of the Translator as a graph-based question answering machine bounds its application as a platform for researchers to guide their research and clinicians to guide their care [346] , rather than a tool for linking data.

One primary example currently featured by NCATS is using the translator to propose novel treatments for drug-induced liver injury (DILI) [347] detailed in a 2021 conference paper [348] . To find a candidate drug, the researchers manually conducted three API queries: first they searched for phenotypes associated with DILI and selected "one of them"[22] — "red blood cell count". Then they queried for genes associated with red blood cell count to find telomerase reverse transcriptase (TERT), and then finally for drugs that affect TERT to find Zidovudine. The directionality of each of these relationships, high vs. low, increases vs. decreases, is unclear in each case. A more recent report on the Translator repeated this pattern of manual querying, arriving at a handful of different genes and drugs [344] .

[22] Using the only API listed with a "related to" link between disease and phenotypic feature, SEMMEDDB, I was unable to find "Red blood cell count" with DILI (C0860207) as either the subject or object, and it is unclear why one would prefer that to any number of other phenotypes like "Fever" or the ominious symptom named "Symptoms" (C1457887).

While the current examples are highly manual, providing an array of results for each query along with links to associated papers on pubmed, some algorithmic system for ranking results is necessary to make use of the information in the extended knowledge graph. Rather than just the first-order connections, it should be possible to make use of second, third, and n-th order connections to weight potential results. Algorithmic medical recommendation systems have been thoroughly problematized elsewhere (eg. [349, 350, 351, 352]). The primary ranking algorithm is developed by a defense contractor (CoVar) who has[23] named it ROBOKOP [353] [24]. Though ROBOKOP functions with a simple weighted graph metric based on citations and abstract text, the ranking system is intended to be extended with machine learning tools [353] that can be trained based on the way the provided answers are used [341] . Algorithmic recommendation platforms are in a regulatory gray area [354, 355] , but would arguably need to have interpretable results with clear provenance to pass scrutiny. The DILI example uses a language model which explained the recommendation of Zidovudine with all the clarity of "one of 'DOWNREGULATOR,' 'INHIBITOR,' 'INDIRECT DOWNREGULATOR'."

[23] seemingly unironically

[24] which seems totally fine and normal.

The arrival at a biomedical question answering platform built atop an algorithmic ranking system for a knowledge graph that queries 200+ aggregated data sources has several qualities that should give us pause.

First, as with any machine-learning based system, the algorithm can only reflect the structure of its input data, including its bias. The "mass of data" approach ML tools lend themselves to, in this case, querying hundreds of independently operated databases, makes dissecting the provenance of every entry from every data provider effectively impossible. For example, one of the providers, mydisease.info was more than happy to respond to a query for the outmoded definition of "transsexualism" as a disease [356] along with a list of genes and variants that supposedly "cause" it - see for yourself. At the time of the search, tracing the source of that entry first led to the

disease ontology DOID:1234 which traced back into an entry in a graph aggregator Ontobee (Archive Link), which in turn listed this github repository **maintained by a single person** as its source[25]. This is, presumably, the fragility and inconsistency in input data that the machine learning layer is intended to putty over.

If the graph encodes being transgender as a disease, it is not farfetched to imagine the ranking system attempting to "cure" it. In a seemingly prerelease version of the translator's query engine, ARAX, it does just that: in a query for entities with a `biolink:treats` link to gender dysphoria[26], it ranks the standard therapeutics [**?**] Testosterone and Estradiol 6th and 10th of 11, respectively — behind a recommendation for Lithium (4th) and Pimozide (5th) due to an automated text scrape of two conversion therapy papers[27]. Queries to ARAX for treatments for gender identity disorder helpfullly yielded "zinc" and "water," offering a paper from the translator group that describes automated drug recommendation as the only provenance [**?**] . A query for treatments for `DOID:1233` "transvestism" was predictably troubling.

Even if the curators do their best to prevent harmful queries and block searches for "cures" to being trans, the graph-based nature of the system means that any given entry will have unpredictable consequences on recommendations made from the surrounding network of objects like genes, treatment history, and so on. If the operation of the ranking algorithm is uninterpretable, as most are, or the algorithm it itself proprietary, harmful input data could have long-range influence on both the practice of medicine as well as the course of basic research *without anyone being able to tell*. The Consortium also describes a system whereby the algorithm is continuously updated based on usage of results in research or clinical practice [341] , which stands to magnify the problem of algorithmic bias by uncritically treating harmful treatment and research practices as training data.

The approach creates a fundamental tradeoff between algorithmic interpretability and the system being useful at all. The paper cited in the 2021 DILI example as evidence that the system gives plausible results is for a specific subclass of liver injuries caused by anti-tuberculosis drugs [357] , highlighting the danger of automated recommendations from noisy data, but also calling into question what novel contribution the Translator made if telomeres were already implicated in DILI. The 2022 report gives examples where the results were already expected by the researchers, or provided a series of papers that seems difficult to imagine being much more informative than a PubMed search. If the algorithmic recommendations are unexpected — ie. the system provides novel information — the process of confirming them appears to be near-identical to the usual process of reading abstracts and hopping citation trees.

Perhaps most worrisome is the eventual fate of the project in the hands of the broader ecosystem of orbiting information conglomerates. Centralized infrastructure projects can be an opportunity for for-profit companies to "dance until the music stops" and then scoop up any remaining technology when the funding dries up (so far roughly $81.6 million since 2016 for the Translator [358] , and $84.7 million for the discontinued NIH Data Commons pilot which morphed into the STRIDES program). I have little doubt that the scientists and engineers working on the Translator are doing so with the best of intentions — the real question is what happens to it after it's finished.

[25] I submitted a pull request to remove it, but it has not been merged more than 8 months later. A teardrop in the ocean.

[26] To its credit, ARAX does transform the request for `DOID:10919` to `MONDO:0001153` - gender dysphoria.

[27] as well as a recommendation for "date allergenic extract" from a misinterpretation of "to date" in the abstract of a paper that reads "Cross-sex hormonal treatment (CHT) used for gender dysphoria (GD) could by itself affect well-being without the use of genital surgery; however, **to date,** there is a paucity of studies investigating the effects of CHT alone"

Knowledge graphs in particular are promising targets for platform holders. Perhaps the most well known example is Google's 2010 acquisition of Freebase (via Metaweb) [359] , a graph of structured data with a wealth of properties for common people, places and things. Google incorporated it into their Knowledge Graph [360] to populate its factboxes and make its search results more semantically aware in its Hummingbird upgrade in 2013, the largest overhaul of its search engine since 2001 [361] , cementing its dominance as a search engine. The connection between swallowing up knowledge organization systems into search engines is not incidental, but reflective of the broader pattern of enclosing basic digital infrastructure behind opaque platforms. Searching has a different set of cognitive expectations than browsing a database: we expect search results to be "best effort," not necessarily complete or accurate, where when browsing a database it's relatively clear when information is missing or inaccurate. For products packaged up into search platforms by for-profit companies, *it doesn't have to actually work* as long as it seems like it does.

The platformatization of the knowledge graph, along with carefully worded terms of service, is a clean means by which "good enough" results could be jackknifed into an expanded system of biomedical surveillance. Since the algorithm needs continual training, the translator has every incentive to suck up as much personal data as it can[28]. For-profit platform providers as a rule depend on developing elaborate personal profiles for targeted advertising algorithmically inferred from available data[29], that naturally includes diagnosed or inferred disease — a practice they explicitly describe in the patents for the targeting technology[363] , have gone to court to defend [364, 365] , and formed secretive joint projects with healthcare systems to pursue [366] .

So while an algorithmic recommendation tool may have limited use for the basic researchers it was originally intended for, it is likely to be extremely useful for the booming business of "personalized medicine." Linking biomedical and patient data in a single platform is a natural route towards a multisided market where records management apps are sold to patients, treatment recommendation systems are sold to clinicians, research tools and advertising opportunities are sold to pharmaceutical companies, risk metrics are sold to insurance companies, and so on.

Multiple information conglomerates are poised to capitalize on the translator project. Amazon already has a broad home surveillance portfolio [367] , and has been aggressively expanding into health technology [368] and even literally providing health care [369] , which could be particularly dangerous with the uploading of all scientific and medical data onto AWS with entirely unenforceable promises of data privacy through NIH's STRIDES program [370] .

RELX, parent of Elsevier, is as always the terrifying elephant in the room. In addition to distribution rights for a large proportion of scientific knowledge and a collection of research databases, it also sells a clinical reference platform in ClinicalKey, point of service products for planning patient care with ClinicalPath, medical education tools, and pharmaceutical advertisements designed to look like scientific papers [371] , among others [372] . It also is explicitly expanding into "clinical decision support applications" [372] and recently embedded its medication management product into Apple's watchOS 9 [223] . Subsidiaries in RELX's "Risk" market segment sell risk profiles to insurance companies based on what they claim to be highly comprehensive profiles of harvested personal data. The Translator infrastructure is a perfect keystone to unify these products: after the NIH fronts the money to develop it

[28] A 2020 presentation in one of the Translator's github repositories describes methods for mining individual clinical data [362]

[29] A patent from Google is telling about how they view privacy concerns: whatever we can't get explicitly, we'll infer to sell better ads! > One possible method to improve ad targeting is for ad targeting systems to obtain and use user profiles. For example, user profiles may be determined using information voluntarily given by users (e.g., when they subscribe to a service). This user attribute information may then be matched against advertiser specified attributes of the ad (e.g., targeting criteria). Unfortunately, user profile information is not always available since many Websites (e.g., search engines) do not require subscription or user registration. Moreover, even when available, the user profile may be incomplete (e.g., because the information given at the time of subscription may be limited to what is needed for the service and hence not comprehensive, because of privacy considerations, etc.). Furthermore, advertisers may need to manually define user profile targeting information. In addition, even if user profile information is available, advertisers may not be able to use this information to target ads effectively. [363]

and lends the credibility of basic research, RELX can cheaply expand its surveillance apparatus to enhanced medical risk profiles to insurers, priority placement in candidate drug rankings to pharmaceutical companies, and augment its ranking systems for funders and employers to include some proprietary metric of "promisingness" to encourage researchers to follow its research recommendations. This isn't speculative — it can just strap whatever clinical data Translator gains access to into its existing biomedical knowledge graph.

Even assuming the Translator works perfectly and has zero unanticipated consequences, the development strategy still reflects the inequities that pervade science rather than challenge them. Biopharmaceutical research, followed by broader biomedical research, being immediately and extremely profitable, attracts an enormous quantity of resources and develops state of the art infrastructure, while no similar infrastructure is built for the rest of science, academia, and society.

The eventual form of the Translator follows from a series of decisions centered around the intended universality of the system. From the funding statement in 2016, the system was conceptualized as an "informatics platform" intended to "bring together all biomedical and health data types." The surrounding background of cloud-based database storage imagined by the Strategic Plan for Data Science immediately constrained the design to consist of APIs that served small quantities of aggregated data, rather than potentially large quantities of raw data. Together with a platform, rather than tool-based approach, a system that allowed individual researchers to link and make sense of the subtlety of own their data was precluded from the start.

From these constraints, the form of the BioLink model comes into focus: high-level classes and logical relationships between them as asserted by a large number of separate knowledge sources. Since the data from each of these sources is heterogeneous, relatively uncurated, and potentially numerous for any given graph-based query, the need for a machine learning layer to make sense of it follows. The conceptualization of BioLink as a universal ontology seems to follow the lineage of the "neat" thought style [247] that emphasizes "deductive inference through logical rules" [343] or otherwise computing derived information from the structure of the knowledge graph rather than browsing the graph itself. Together, these constraints and design logics bring us to the form of the Translator as a graph-based query engine.

The Translator Consortium justifiably takes pride in its social organizing systems [373] — coordinating 200 researchers and engineers from dozens of institutions is no small feat. This system of social organization seems to have lent itself towards developing the individual components with an eye for them to be understood by the rest of the *consortium* rather than with the intention of inviting collaboration from the broader research community[30]. The very notion of a platform indicates that it is something that *they build* and *we use*: There is no explicit means for proposing changes to the BioLink model, to pick and choose how answers are ranked or queries are performed, etc. This is broadly true of platform-based scientific tools, especially databases, and contributes to how they *feel*: they feel disconnected with our work, don't necessarily help us do it more easily or more effectively, and contributing to them is a burdensome act of charity (if it is possible at all).

Given the real need for *some* means of combining heterogeneous data from disparate sources, what could have been done differently?

Problematizing the need for a system intended to link *all* or even *most* biomedical data in a single mutually coherent system opens the possibility for a very differ-

[30] The descriptions of difficulty in interfacing the components of the project internally are littered throughout their public-facing documents: eg. "A lot of the work has been about defining standards, so that the components that each of the 15 teams are building can talk to each other" [345] , "In part due to the speed with which the program has progressed, team members also have found it challenging to coordinate milestones and deliverables across teams and align the goals of the Translator program with the goals of their own nonTranslator research projects." [373] . These problems are, of course, completely reasonable. My comment here merely suggests that solving these problems, particularly on the self-described tight timeline of the Translator's development, may have edged out concerns for engagemenent with the broader research community.

ent data linking infrastructure. Perhaps paradoxically, any universal, logically complete schema intended to support algorithmic inference projects a relatively circumscribed group of people for whom it would be useful: nearly all of the publicly described use-cases are oriented around finding new drugs or targets to treat disease, presumably in part because that's what preoccupies the ontology. Rather than a set of generalizable *tools* for linking data, the need for universality strongly constrains the form of data that can be represented by the system, and its platform structure constrains its uses to only those imagined by the platform designers. Every infrastructural model is an act of balancing constraints, and prioritizing "all data" seems to imply "for some people." Who is supposed to be able to upload data? change the ontology? inspect the machine learning model? Who is in charge of what? Who is a knowledge-graph query engine useful for?

Another conceptualization might be building systems for *all people* that can *embed with existing practices* and *help them do their work* which typically involves accessing *some data.* We can imagine a system designed to integrate data with schemas written in the *vernacular* of communities of knowledge work. Rather than the dichotomy of one singular database vs. many fragmented and incompatible databases, we can imagine a *pluralistic* system capable of supporting multiple overlapping and potentially conflicting representations, governable and malleable in local communities of practice. Taking seriously the notion of "translation," we could stand to learn from linguistics and translation studies: rather than attempting to project the dialects of each subdiscipline into some "true" meta-framework (a decidedly colonial project [374]), we could resist the urge for homogenization and preserve the multiplicity of representation, embracing the imperfection of mappings between heterogeneous representational systems at multiple scales without resigning ourselves to completely isolated incompatibility.

Maybe we don't *want* a universal system that presents itself with the authority of truth to be mined and spun off into derivative platforms by information conglomerates. We might abandon the techno-utopianism of a globally consistent schema that supports arbitrary logical inference by acknowledging that those inferences would always be colored by the decisions embedded in the structure of the system, unknowable beneath the shrouding weights of its ranking model.

Instead can we imagine a properly *human* data infrastructure? One that preserves the seams and imperfections in our representational systems, that is designed to represent precisely the contingency of representation itself? We might start with the propositional nature of links and mappings between formats — that rather than a divine received truth, the relationships between things are contextual and created. We could find grounding in *use,* that the schemas and mappings between them should arise from the need to link representations within the context of some problem, rather than to resolve their difference.

Picking up the thread of our peer to peer data sharing backbone, we might start to imagine the boistrous multiplicity of an infrastructure based around communication and expression, rather than platformatized perfection.

11.2.5 Folk Federation

> Human language thrives when using the same term to mean somewhat different things, but automation does not. *Tim Berners-Lee (1999) The Semantic Web*

[375]

> Wittgenstein's contribution to communism was his robust proof of the proposition that there is no private language, but in our time, privatized languages are everywhere. And not just languages: Images, codes, algorithms, even genes can become private property, and in turn private property shapes what we imagine the limits and possibilities of this information to be. *McKenzie Wark (2021) Capital Is Dead: Is This Something Worse?* [216]

To structure our p2p data sharing system, we should use *Linked Data*. Linked data is at once exceptionally simple and deceptively complex, a set of technologies and social histories. In this section we will introduce the notion of linked data, extend it for a p2p context, and then add a twist from *federated systems.*[31] Our goal will be to articulate the foundation for a "protocol of protocols," a set of minimal operations by which individual people can create, extend, borrow, and collectively build a space of linked folk schemas and ontologies, or *folksonomies.*

When last we left it, we had developed the notion of a p2p system to the point where we had big torrentlike piles of files with a few additional features like versioning and sharded storage. We need to add an additional layer of *metadata* that exposes information about the contents of each of these file piles. But what is that metadata *made of?*

The core format of linked data is the Resource Document Format (RDF) [377] and its related syntaxes like Turtle [378] . Typical hyperlinks are *duplet* links — linking from the source to the target. The links of linked data are instead **triplet** links that consist of a **subject**, a **predicate** that *describes* the link, and an **object** that is linked to. Subjects and objects (generally, nodes) have particular types like a number, or a date, or something more elaborate like an Airline or Movie that have particular sets of predicates or properties: eg. a `Movie` has a `director` property which links to a `Person`. A `Person` has an `address` which links to a `PostalAddress`, and so on. Types and properties are themselves defined in **vocabularies** (or, seemingly interchangeably [379] , ontologies and schemas) by a special subset of RDF schema modeling classes and properties [380] . Linked data thus consists of semantically annotated **graphs** of linked nodes[32].

Linked data representations are very general and encompass many others like relational [381] and object-oriented models, but have a few properties that might be less familiar. The first is that triplet links have the status of an utterance or a proposition: much like typical duplet hyperlinks, anyone can make whatever links they want to a particular object to say what they'd like about it. As opposed to object-oriented models where a class is defined beforehand and its attributes or data are stored "within" the object, RDF schemas are composed of links just like any other, and the link, object, and predicate can all be stored in separate places by different people [382] . For example:

> One person may define a `vehicle` as having a `number of wheels` and a `weight` and a `length`, but not foresee a `color`. This will not stop another person making the assertion that a given car is red, using the color vocabulary from elsewhere. [382]

[31] There is a lot of subtlety to the terminology surrounding "federated" and the typology of distributed systems generally, I am using it in the federated messaging sense of forming groups of people, rather than the strict term "federated databases" which do imply a standardized schema across a federation. The conception of distributed, autonomous databases described by the DataLad team [376] is a bit closer to my meaning. In the ActivityPub world, federations refer to a single homeserver under which many people can sign up. We mean something similar but distinct: people that have autonomous "homeservers" in a peer to peer system, typically multiple identities for a single person rather than many people on a single server, that can combine into federations with particular governance structures and technological systems attached.

[32] Or, precisely, a "directed labeled graph" (DLG).

Linked data has an ambivalent history of thought regarding the location and distribution of ontology building. Its initial formulation came fresh from the recent incendiary success of the internet, where without any system of organization "people were frightened of getting lost in it. You could follow links forever." [382] Linked data was conceptualized to be explicitly without authoritative ontologies, but intended to evolve like language with local cultures of meaning meshing and separating at multiple scales [375] . Perhaps one of the pieces that went missing when moving between writing about the semantic web and its realization in standards and protocols is that this language-like conception of links requires **quartet,**

rather than triplet links: **author**, subject, object, predicate. The author is encoded implicitly in the source of the vocabulary: "Users are given [...] a single URI [...] for each persona they want to have," [231] so theoretically ontologies have the status of "schema.org says this." Without a first-class notion of author in the links themselves there is little means of "forking" a vocabulary, or having multiple versions of a term with the same name but different authors.

The dream of mass automaticity, however, with computational "agents" capable of seamlessly crawling consistent graphs of linked data to extract surplus meaning necessarily requires that the meaning of terms does not "mutate" between different uses. For many early linked data architects the resolution was more automation, to use additional semantic structure about the equivalence between different ontologies as a means of estimating how trustworthy a particular result was. This tension is sewn into one of its most well known ontologies, the Simple Knowledge Organization System (skos) [383] , which is intended to represent relationships between terms and vocabularies [384] .

The fluidity of the original vision for linked data never emerged, however, and is remembered instead as being monstrously overcomplicated [246, 385] . While HTML, CSS, and Javascript developed a rich ecosystem of abstractions that let people create websites without directly writing HTML, the same never materialized for RDF. While linked data entities are intended to be designated by the very general notion of a URI, in practice URIs are near-synonymous with URLs, and maintaining a set of URLs is hard. In the absence of interfaces for manipulating linked data and the pain of hosting them, the dream of a distributed negotiation over language-like ontologies was largely confined to information scientists and what became corporate knowledge graphs. For those war-weary RDF vets, I will again clarify that we are describing the desirable *qualities* of RDF while trying to learn from its failures.

In our revival of this dream we are describing a system where heterogeneous data is indicated by its metadata, rather than representing all data in a uniform format — similarly to the mixture of RDF and non-RDF data in the linked data platform standard [386] . We want to handle a broad span of heterogeneity: data with different naming schemes, binary representations, sizes, nested structures, and so on. The first task is to describe some means of accessing this heterogeneous data in a reasonably standard way despite these differences.

While that may seem a tall order, researchers already do it, it's just mostly done manually whenever we want to use anyone else's data. One way of characterizing the task at hand is systematizing the idiosyncratic paths by which a researcher might dump out a .csv file from a sql database to load into MATLAB to save in the .mat format with the rest of their data. To do that we can draw from a parallel body of thought on *federated databases.*

Like our p2p system, federated systems consist of *distributed*, *heterogeneous*, and *autonomous* agents that implement some minimal agreed-upon standards for mutual communication and (co-)operation. Federated databases were proposed in the early 1980's [387] and have been developed and refined in the decades since as an alternative to either centralization or non-integration [388, 389, 390] . Their application to the dispersion of scientific data in local filesystems is not new [391, 392, 393] , but their implementation is more challenging than imposing order with a centralized database or punting the question into the unknowable maw of machine learning.

Amit Sheth and James Larson, in their reference description of federated database systems, describe **design autonomy** as one critical dimension that characterizes them:

> Design autonomy refers to the ability of a component DBS to choose its own design with respect to any matter, including
>
> - (a) The **data** being managed (i.e., the Universe of Discourse),
> - (b) The **representation** (data model, query language) and the **naming** of the data elements,
> - (c) The conceptualization or **semantic interpretation** of the data (which greatly contributes to the problem of semantic heterogeneity),
> - (d) **Constraints** (e.g., semantic integrity constraints and the serializability criteria) used to manage the data,
> - (e) The **functionality** of the system (i.e., the operations supported by system),
> - (f) The **association and sharing with other systems**, and
> - (g) The **implementation** (e.g., record and file structures, concurrency control algorithms).

Susanne Busse and colleagues add an additional dimension of **evolvability,** or the ability of a particular system to adapt to inevitable changing uses and requirements [391] .

In order to support such radical autonomy and evolvability, federated systems need some means of translating queries and representations between heterogeneous components. The typical conceptualization of federated databases have five layers that implement different parts of this reconciliation process [394] :

- A **local schema** is the representation of the data on local servers, including the means by which they are implemented in binary on the disk
- A **component schema** serves to translate the local schema to a format that is

compatible with the larger, federated schema

- An **export schema** defines permissions, and what parts of the local database are made available to the federation of other servers

- The **federated schema** is the collection of export schemas, allowing a query to be broken apart and addressed to different export schemas. There can be multiple federated schemas to accomodate different combinations of export schemas.

- An **external schema** can further be used to make the federated schema better available to external users, but in this case since there is no notion of "external" it is less relevant.

This conceptualization provides a good starting framework and isolation of the different components of a database system, but a peer-to-peer database system has different constraints and opportunities [395] . In the strictest, "tightly coupled" federated systems, all heterogeneity in individual components has to be mapped to a single, unified federation-level schema. Loose federations don't assume a unified schema, but settle for a uniform query language, and allow multiple translations and views on data to coexist. A p2p system naturally lends itself to a looser federation, and also gives us some additional opportunities to give peers agency over schemas while also preserving some coherence across the system. I will likely make some database engineers cringe, but the emphasis for us will be more on building a system to support distributed social control over the database, rather than guaranteeing consistency and transparency between the different components.

Let us take the notion of a loosely coupled systems to its extreme, and invert the meaning of federation as it is used in other systems like ActivityPub: rather than a server-first federation, where peers create accounts on servers that define their operation and the other servers they federate with, ours will be peer-first federation. In this system, individual peers will maintain their own vocabularies and be able to make them available to other peers. Peers can directly connect to one another, but can also federate into groups, which can federate into groups of groups, and so on. A peer will implement the local, component, and export schema with a client that handles requests for vocabularies and and datasets according to their scheme of permissions. Translation from a metadata-based query to a particular binary representation of a file, whether it be in a relational database, binary, file, or otherwise, will also be supported by vocabularies that indicate the necessary code.

Clearly, we need some form of *identity* in the system so that a peer can have their links unambiguously identified and discovered. This is a challenging problem that we leave open here, but strategies ranging from URI-based resolution like `username@domain.com`, to locally-held cryptographic key based identity, to decentralized systems like the w3c's Decentralized Identifiers [396] would suffice. For the sake of example, let's make identity simple and flat, denoted in pseudocode as `@username`. Someone would then be able to use their `@name`space as a root, under which they could refer to their data, schemas, and so on, which will be denoted `@name:subobject` (see this notion of personal namespaces for knowledge organization discussed in early wiki culture here [397]). Let us also assume that there is no categorical difference between `@usernames` used by individual researchers, institutions, consortia, etc. — everyone is on the same level.

To illustrate the system by example, we pick up where we left off earlier with a peer who has their data in some discipline-specific format, which let us assume for the sake of concreteness has a representation as an OWL schema.

That schema could be "owned" by the `@username` corresponding to the standard-writing group — eg `@nwb` for neurodata without borders. In all the following examples, we will use a turtle-ish syntax that is *purposely pseudocode* with the intention of demonstrating general qualities without being concerned with syntactic correctness or indicating one syntax in particular. Our dataset might look like this:

```
@base @jonny

<#my-data>
  a @nwb:NWBFile
  @nwb:general:experimenter @jonny
  @nwb:ElectricalSeries
    .electrodes [1, 2, 3]
    .rate 30000
    .data [ ... ]
```

Unpacking the pseudocode, this indicates:

- We declare a `@base` context underneath my identity, `@jonny`,
- Underneath the base, individual objects are declared with their name like `<#object-name>`, a shorthand for `<@base:object-name>`. In this case I have made a dataset identified as `@jonny:my-data`.
- I have identified the type of this object with the `a` token, in this case a `@nwb:NWBFile`
- Subsequent lines indicate particular properties of the indicated type and their value, specifically I have indicated that the `@nwb:general:experimenter` is me, `@jonny`, and that the dataset also contains a `@nwb:ElectricalSeries`. While my identity object might have additional links like an `@ORCID:ID`, we can assume some basic inference that resolves my identity to a string as specified in the NWB specification, or else specify it explicitly as `@jonny:name`
- Additional subproperties are assigned with a leading `.`, so `.electrodes` would resolve to `@nwb:ElectricalSeries:electrodes`.

How would my client know how to read and write the data to my disk so i can use and share it? In a system with heterogeneous data types and database implementations, we need some means of specifying different programs to use to read and write, different APIs, etc. This too can be part of the format specification. Suppose the HDF5 group (or anyone, really!) has a namespace `@hdf` that defines the properties of an `@hdf:HDF5` file, basic operations like `Read`, `Write`, or `Select`. NWB could specify that in their definition of `@nwb:NWBFile`:

```
<@nwb:NWBFile>
  a @hdf:HDF5
    .isVersion "x.y.z"
    .hasDependency "libhdf5"=="x.y.z"
  usesContainer @nwb:NWBContainer
```

So when I receive a request for the raw data of my electrical series, my client knows to use the particular methods from the HDF5 object type to index the data contained within the file.

I have some custom field for my data, though, which I extend the format specification to represent. Say I have invented some new kind of solar-powered electrophysiological device — the SolarPhys2000 — and want to annotate its specs alongside my data.

```
<#SolarEphys>
  extends @nwb:NWBContainer

  UsedWith @jonny:hw:SolarPhys2000

  ManufactureDate
    a @schema:Date

  InputWattageSeries
    extends @nwb:ElectricalSeries

    sunIntensity
      a @nwb:TimeSeries
```

Here I create a new extension `@jonny:SolarEphys` that `extends` the `@nwb:NWBContainer` schema. We use `extends` rather than `a` because we are adding something new to the *description* of the container rather than *making* a container to store data. I declare that this container is `UsedWith` our SolarPhys2000 which we have defined elsewhere in our `hw` namespace using some hardware ontology. I then add two new fields, `ManufactureDate` and `InputWattageSeries`, declaring types from, for example `@schema:Date` and `@nwb`.

The abstraction around the file implementation makes it easier for others to consume my data, but it also makes it easier for *me* to use and contribute to the system. Making an extension to the schema wasn't some act of charity, it was the most direct way for me to use the tool to do what I wanted. Win-win: I get to use my fancy new instrument and store its data by extending some existing format standard. We are able to make my work part of a cumulative schema building effort by *aligning the modalities of use and contribution.*

For the moment our universe is limited only to other researchers using NWB. Conveniently, the folks at NWB have set up a federating group so that everyone who uses it can share their format extensions. In the same way that we can use schemas to refer to code as with our HDF5 files, we can use it to indicate the behavior of clients and federations. Say we want to make a federating peer that automatically `Accepts` request to `Join` and indexes any schema that inherits from their base `@nwb:NWBContainer`. Let's say `@fed` defines some basic properties of our federating system — it constitutes our federating "protocol" — and loosely use some terms from the ActivityStreams vocabulary as `@as`

```
<@nwbFederation>
  a @fed:Federation
  onReceive
```

```
    @as:Join @as:Accept
  allowSchema
    extensionOf @nwb:NWBContainer
```

Now anyone that is a part of the `@nwbFederation` would be able to see the schemas we have submitted, sort of like a beefed up, semantically-aware version of the existing neurodata extensions catalog. In this system, many overlapping schemas could exist simultaneously under different namespaces, but wouldn't become a hopeless clutter because similar schemas could be compared and reconciled based on their semantic properties.

Now that I've got my schema extension written and submitted to the federation, time to submit my data! Since it's a p2p system, I don't need to manually upload it, but I do want to control who gets it. By default, I have all my NWB datasets set to be available to the `@nwbFederation` , and I list all my metadata on, say the Society for Neuroscience's `@sfnFederation`.

```
<#globalPermissions>
  a @fed:Permissions
  permissionsFor @jonny

  federatedWith
    name @nwbFederation
    @fed:shareData
      is @nwb:NWBFile

  federatedWith
    name @sfnFederation
    @fed:shareMetadata
```

Let's say this dataset in particular is a bit sensitive — say we apply a set of permission controls to be compliant with `@hhs.HIPAA` — but we do want to make use of some public server space run by our Institution, so we let it serve an encrypted copy that those I've shared it with can decrypt.

```
<#datasetPermissions>
  a @fed:Permissions
  permissionsFor @jonny:my-data

  accessRuleset @hhs:HIPAA
    .authorizedRecipient <#hash-of-patient-ids>

  federatedWith
    name @institutionalCloud
    @fed:shareEncrypted
```

Now I want to make use of some of my colleagues data. Say I am doing an experiment with a transgenic dragonfly and collaborating with a chemist down the hall. This transgene, known colloquially in our discipline as `@neuro:superstar6` (which the chemists call `@chem:SUPER6`) fluoresces when the dragonfly is feeling bashful, and we have plenty of photometry data stored as `@nwb:Fluorescence` objects. We

think that its fluorescence is caused by the temperature-dependent conformational change from blushing. They've gathered NMR and Emission spectroscopy data in their chemistry-specific format, say `@acs:NMR` and `@acs:Spectroscopy`.

We get tired of having our data separated and needing to maintain a bunch of pesky scripts and folders, so we decide to make a bridge between our datasets. We need to indicate that our different names for the gene are actually the same thing and relate the spectroscopy data.

Let's make the link explicit, say we use an already-existing vocabulary like the "simple knowledge organization system" for describing logical relationships between concepts: `@skos`?

```
<#links:super6>
  @neuro:superstar6
    @skos:exactMatch @chem:SUPER6
```

Our `@nwb:Fluorescence` data has the emission wavelength in its `@nwb:Fluorescence:excitation_lambda` property[33], which is the value of their `@acs:Spectroscopy` data at a particular value of its `wavelength`. Unfortunately, `wavelength` isn't metadata for our friend, but does exist as a column in the `@acs:Spectroscopy:readings` table, so where we typically have a singular value they have a set of measurements. Since the same information has a structurally different meaning across disciplines, we dont expect there to be an automated 1:1 mapping between them, but presumably their data format also specifies some means of reading the data akin to the HDF5 methods indicated by our NWB data format so we can add an additional translation later like `@math:mean` and pick it up in our analysis tools.

[33] not really where it would be in the standard, but again, for the sake of example...

```
<#links:lambda>
  @acs:Spectroscopy:readings:wavelength
    @skos:narrowMatch @nwb:Fluorescence:excitation_lambda
      @skos:note
        "Multiple spectrographic readings are
        aggregated to a single excitation lambda"
      @translate:aggregate @math:mean
```

This makes it much easier for us to index our data against each other and solves a few real practical problems we were facing in our collaboration. We don't need to do as much cleaning when it's time to publish the data since it can be released as a single linked entity.

Though this example is relatively abstract (which metadata from spectroscopy readings would need to match which in a fluorescence series to compare wavelengths to lambda?), it serves as an example in its own right of the quasi-inversion of reasoning that we can make use of in our particular version of linked data with code. We refer to the general notion of taking a `@math:mean`, but don't specify a particular implementation of it. Other package maintainers could indicate that their function implements it, so we could be prompted to choose one when resolving the link. Alternatively, if we specified our aggregation used `@numpy:mean`, we could trace it backwards to find which general operation it implements and choose a different one. Since the objects of any triplet link have their own type, we can use the *context* of the link to infer how to use it.

Rinse and repeat our sharing and federating process from our previous schema extension, add a little bit of extra federation with the `@acs` namespace, and in the normal course of our doing our research we've contributed to the graph structure linking two common data formats. Our link is one of many, and is a proposition that other researchers can evaluate in the context of our project rather than as an authoritative reference link. We might not have followed the exact rules, but we have also changed the nature of rules — rather than logical coherence guaranteed *a priori* by adherence to a specification language, much like language the only rules that matter are those of *use.* We may have only made a few links rather than a single authoratative mapping, but if someone is interested in compiling one down the line they'll start off a hell of a lot further than if we hadn't contributed it! Rather than this format translation happening ad-hoc across a thousand lab-specific analysis libraries, we have created a space of *discourse* where our translation can be contextually compared to others and negotiated by the many people concerned, rather than handed down by a standards body.

Queries across what amounts to the federated schema, in the federated database parlance, are by design less seamless than they would be with centrally governed schema — which is a feature, not a bug. While this example deals with relatively dry fluorescence and spectrographic data, if this system were to expand to clinical, cultural, and personal data, the surveillance economy that emerged subsequent to they heydey of the semantic web has made it abundantly clear that *we don't necessarily want* arbitrary actors to be able to index across all available data. It is much more valuable to have low-barrier, vernacular expression usable by collections of subdisciplines and communities of people than a set of high-barrier, fixed, logically correct schemas. Researchers and people alike typically are only concerned with using the information within or a few hops outside of their local systems of meaning, so who is a totalizing database of everything *for?* This framing of linked data, by rejecting the goal of global inference altogether, could be considered beyond even Lindsay Poirier's conception of "scruffiness" to something we might properly call *vulgar linked data.*

The act of translation is always an act of creation, and by centering the multiplicity of links between extensible schemas we center the dialogic reality of that creation: *who says* those things are equivalent? Since the act of using translating links between schemas itself creates links — ie. I link to `@<user>`'s link to link my dataset and another — we are both able to assess the status of consensus around which links are used, as well as bring a currently invisible form of knowledge work into a system of credit. As we will develop in the following two sections, this multiplicity also naturally lends itself to a fluid space of tools that implement translations and analyses, as well as a means of discussing and contextualizing the results.

We have been intentionally vague about the technical implementation here, but there are many possible strategies and technologies for each of the components.

For making our peers and the links within their namespace discoverable we could use a distributed hash table, or **DHT**, like bittorrent, which distributes references to information across a network of peers (eg. [398]). We could use a strategy like the

Matrix messaging protocol, where peers could federate with "relay" servers. Each server is responsible for keeping a synchronized copy of the messages sent on the servers and rooms it's federated with, and each server is capable of continuing communication if any of the others failed. We could use **ActivityPub** (AP) [399] , a publisher-subscriber model where users affiliated with a server post messages to their

'outbox' and are sent to listening servers (or made available to HTTP GET requests). AP uses JSON-LD [400] , so is already capable of representing linked data, and the related ActivityStreams vocabulary [401] also has plenty of relevant action types for creating, discussing, and negotiating over links (also see cpub). We could use a strategy like IPFS where peers will voluntarily rehost each other's data in order to gain trust with one another. To preserve interoperability with existing systems, we will want to make links referenceable from a URI (as IPFS does) as well as be able to resolve multiple protocols, but beyond that the space of possible technologies is broad.

Indexing and querying metadata across federated peers could make use of the SPARQL query language [402] as has been proposed for biology many times before [403, 392, 393] . The distinction between metadata and data is largely practical — a query shouldn't require transferring and translating terabytes of data — so we will need some means of resolving references to data from metadata as per the linked data platform specification [386] . A mutable/changeable/human-readable name and metadata system that points to a system of unique content addressed identifiers has been one need that has hobbled IPFS, and is the direction pointed to by DataLad[34] [376] . A parallel set of conversations has been happening in the broader linked data community with regard to using ActivityPub as a way to index data on Solid.

[34] DataLad [404, 376] and its application in Neuroscience as DANDI are two projects that are *very close* to what I have been describing here — developing a p2p backend for datalad might even be a promising development path towards it.

The design of federations of peers is intended to resolve several of the problems of prior p2p protocols. Rather than a separate swarm for every dataset per bittorrent, or a single global swarm per IPFS, this system would be composed of peers that can voluntarily associate and share metadata structure at multiple scales. Bittorrent requires trackers to aggregate and structure metadata, but they become single points of failure and often function as means of gatekeeping by the beloved petty tyrants who host them. IPFS has turned to filecoin to incentivize donating storage space among quasi-anonymous peers, a common design pattern among the radical zero-trust design of many cryptocurrencies and cryptocurrency-like systems.

Voluntary federations are instead explicitly social systems that can describe and organize their own needs: peers in a federation can organize tracker or serverlike rehosting of their data for performance, discoverability, guaranteed longevity. A federation can institute a cooperative storage model akin to private bittorrent trackers that requires a certain amount of rehosted data per data shared. A small handful of researchers can form a small federation to share data while collaborating on a project in the same way that a massive international consortioum could. Without enumerating their many forms, federations can be a way to realize the evolvable community structure needed for sustained archives. As may become clearer as we discuss systems for communication, in the context of science they might be a way of reconceptualizing scientific societies as something that supports the practice of science beyond their current role as ostensibly nonprofit journal publishers and event hosts.

So far we have described a system for sharing data with a p2p system integrated with linked data. We have given a few brief examples of how linked data can be used for standardized and vernacular metadata, integrating with heterogeneous local storage systems, and to perform actions like creating and joining federations of peers. As described, though, the system would still be decidedly unapproachable for most scientists and doesn't offer the kind of strong incentives that would create a broad base of use. We clearly need one or several *interfaces* to make the creation and use metadata easy. We will return to those in Shared Knowledge and also describe a set of communication and governance systems sorely needed in science. To get there, we

will first turn to a means of integrating our shared data system with analytical and experimental tools to make each combinatorically more useful than if considered alone.

11.3 Shared Tools

Straddling our system for sharing data are the tools to gather and analyze it — combining tools to address the general need for *storage* with *computational resources*. Considering them together presents us with new opportunities only possible with cross-domain interoperability. In particular, we can ask how a more broadly integrated system makes each of the isolated components more powerful, enables a kind of deep provenance from experiment to results, and further builds us towards reimagine the form of the community and communication tools for science. Where the previous section focused on integrating linked metadata with data, here our focus is how to make linked data *do things* by integrating it with code.

This section will be relatively short compared to shared data. We have already introduced, motivated, and exemplified many of the design practices of the broader infrastructural system. There is much less to argue against or "undo" in the spaces of analytical and experimental tools because so much more work has been done, and so much more power has been accrued in the domain of data systems. Distributed computing does have a dense history, with huge numbers of people working on the problem, but its dominant form is much closer to the system articulated below than centralized servers are to federated semantic p2p systems. I also have written extensively about experimental frameworks before [405] , and develop one of them so I will be brief at risk of repeating myself or appearing self-serving.

Integrated scientific workflows have been written about many times before, typically in the context of the "open science" movement. One of the founders of the Center for Open Science, Jeffrey Spies, described a similar ethic of toolbuilding as I have in a 2017 presentation:

> Open Workflow: 1. Meet users where they are 2. Respect current incentives 3. Respect current workflow
>
> - We could... demonstrate that it makes research more efficient, of higher quality, and more accessible.
> - Better, we could... demonstrate that researchers will get published more often.
> - Even better, we could... make it easy.
> - Best, we could... make it automatic [406]

Similar to the impossibility of a single unified data format, it is unlikely that we will develop one tool to rule them all. We will take the same tactic of thinking about *frameworks* to integrate tools and make them easier to build, rather than building any tool in particular.

11.3.1 Analytical Frameworks

The first natural companion of shared data infrastructure is a shared analytical framework. A major driver for the need for everyone to write their own analysis code largely from scratch is that it needs to account for the idiosyncratic structure of everyone's data. Most scientists are (blessedly) not trained programmers, so code for loading and loading data is often intertwined with the code used to analyze and plot it. As a result it is often difficult to repurpose code for other contexts, so the same analysis function is rewritten in each lab's local analysis repository. Since sharing raw data and code is still a (difficult) novelty, on a broad scale this makes results in scientific literature as reliable as we imagine all the private or semi-private analysis code to be.

Analytical tools (anecdotally) make up the bulk of open source scientific software, and range from foundational and general-purpose tools like numpy [407] and scipy [408] , through tools that implement a class of analysis like DeepLabCut [238] and scikit-learn [409] , to tools for a specific technique like MoSeq [410] and DeepSqueak [411] . The pattern of their use is then to build them into a custom analysis system that can then in turn range in sophistication from a handful of flash-drive-versioned scripts to automated pipelines.

Having tools like these of course puts researchers miles ahead of where they would be without them, and the developers of the mentioned tools have put in a tremendous amount of work to build sensible interfaces and make them easier to use. No matter how much good work might be done, inevitable differences between APIs is a relatively sizable technical challenge for researchers — a problem compounded by the incentives for fragmentation described previously. For toolbuilders, many parts of any given tool from architecture to interface have to be redesigned each time with varying degrees of success. For science at large, with few exceptions of well-annotated and packaged code, most results are only replicable with great effort.

Discontinuity between the behavior and interface of different pieces of software is, of course, the overwhelming norm. Negotiating boundaries between (and even within) software and information structures is an elemental part of computing. The only time it becomes a conceivable problem to "solve" interoperability is when the problem domain coalesces to the point where it is possible to articulate its abstract structure as a protocol, and the incentives are great enough to adopt it. That's what we're trying to do here.

It's unlikely that we will solve the problem of data analysis being complicated, time consuming, and error prone by teaching every scientist to be a good programmer, but we can build experimental frameworks that make analysis tools easier to build and use.

Specifically, a shared analytical framework should be

- **Modular** - Rather than implementing an entire analysis pipeline as a monolith, the system should be broken into minimal, composable modules. The threshold of what constitutes "minimal" is of course to some degree a matter of taste, but the framework doesn't need to make normative decisions like that. The system should support modularity by providing a clear set of hooks that tools can provide: eg. a clear place for a given tool to accept some input, parameters, and so on. Since data analysis can often be broken up into a series of relatively independent

stages, a straightforward (and common) system for modularity is to build hooks to make a directed acyclic graph (DAG) of data transformation operations. This structure naturally lends itself to many common problems: caching intermediate results, splitting and joining multiple inputs and outputs, distributing computation over many machines, among others. Modularity is also needed within the different parts of the system itself – eg. running an analysis chain shouldn't require a GUI, but one should be available, etc.

- **Pluggable** - The framework needs to provide a clear way of incorporating external analysis packages, handling their dependencies, and exposing their parameters to the user. Development should ideally not be limited to a single body of code with a single mode of governance, but should instead be relatively conservative about requirements for integrating code, and liberal with the types of functionality that can be modified with a plugin. Supporting plugins means supporting people developing tools for the framework, so it needs to make some part of the toolbuilding process easier or otherwise empower them relative to an independent package. This includes building a visible and expressive system for submitting and indexing plugins so they can be discovered and credit can be given to the developers. Reciprocal to supporting plugins is being interoperable with existing and future systems, which the reader may have assumed was a given by now.

- **Deployable** - For wide use, the framework needs to be easy to install and deploy locally and on computing clusters. A primary obstacle is dependency management, or making sure that the computer has everything needed to run the program. Some care needs to be taken here, as there are multiple emphases in deployability that can be in conflict. Deployable for who? A system that can be relatively challenging to use for routine exploratory data analysis but can distribute analysis across 10,000 GPUs has a very circumscribed set of people it is useful for. This is a matter of balancing design constraints, but we should prioritize broad access, minimal assumptions of technological access, and ease of use over being able to perform the most computationally demanding analyses possible when in conflict. Containerization is a common, and the most likely strategy here, but the interface to containers may need a lot of care to make accessible compared to opening a fresh .py file.

- **Reproducible** - The framework should separate the *parameterization* of a pipeline, the specific options set by the user, and its *implementation*, the code that constitutes it. The parameterization of a pipeline or analysis DAG should be portable such that it, for example, can be published in the supplementary materials of a paper and reproduced exactly by anyone using the system. The isolation of parameters from implementation is complementary to the separation of metadata from data and if implemented with semantic triplets would facilitate a continuous interface from our data to analysis system. This will be explored further below and in shared knowledge

Thankfully a number of existing projects that are very similar to this description are actively being built. One example is DataJoint [412] , which recently expanded its facility for modularity with its recent Elements project [413] . Datajoint is a system for creating analysis pipelines built from a graph of processing stages (among other

features). It is designed around a refinement on traditional relational data models, which is reflected throughout the system as most operations being expressed in its particular schema, data manipulation, and query languages. This is useful for operations that are expressed in the system, but makes it harder to integrate external tools with their dependencies — at the moment it appears that spike sorting (with Kilosort [414]) has to happen outside of the extracellular electrophysiology elements pipeline.

Kilosort is an excellent and incredibly useful tool, but its idiomatic architecture designed for standalone use is illustrative of the challenge of making a general-purpose analytic framework that can integrate a broad array of existing tools. It is built in MATLAB, which requires a paid license, making arbitrary deployment difficult, and MATLAB's flat path system requires careful and usual manual orchestration of potentially conflicting names in different packages. Its parameterization and use are combined in a "main" script in the repository root that creates a MATLAB struct and runs a series of functions — requiring some means for a wrapping framework to translate between input parameters and the representation expected by the tool. Its preprocessing script combines I/O, preprocessing, and plotting, and requires data to be loaded from disk rather than passed as arguments to preserve memory — making chaining in a pipeline difficult.

This is not a criticism of Datajoint or Kilosort, which were both designed for different uses and with different philosophies (that are of course, also valid). I mean this as a brief illustration of the design challenges and tradeoffs of these systems.

We can start getting a better picture for the way a decentralized analysis framework might work by considering the separation between the metadata and code modules, hinting at a protocol as in the federated systems sketch above. In the time since the heydey of the semantic web there has been a revolution in containerization and dependency management that makes it possible to imagine extending the notion of linked data to being able to not only indicate binary data but also *executable code.* Software dependencies form a graph structure, with one top level package specifying a version range from a 1st-order dependent, which in turn has its own set of 2nd-order packages and versions, and so on. Most contemporary dependency managers (like Python's poetry, Javascript's yarn, Rust's cargo, Ruby's Bundler, etc.) compute an explicit dependency graph from each package's version ranges to create a 'lockfile' containing the exact versions of each package, and usually the repositories where they're located and the content hashes to verify them. More general purpose package managers like spack [415] , or nix [416] can also specify system-level software outside of an individual programming language, and containerization tools like docker can create environments that include entire operating systems.

Since we're considering modular analysis elements, each module would need some elemental properties like the parameters that define it, its inputs, outputs, as well as some additional metadata about its implementation (eg. this one takes *numpy arrays* and this one takes *matlab structs*). The precise implementation of a modular protocol also depends on the graph structure of the analysis pipelining system. We invoked DAGs before, but analysis graph structure of course has its own body of researchers refining them into eg. Petri nets which are graphs whose nodes necessarily alternate between "places" (eg. intermediate data) and "transitions" (eg. an analysis operation), and their related workflow markup languages (eg. WDL or [417]). In that scheme, a framework could provide tools for converting data between types, caching intermediate data, etc. between analysis steps, as an example of how differ-

ent graph structures might influence its implementation.

The graph structure of our linked data system could flexibly extend to be continuous with these dependency pipeline graphs. With some means for a client to resolve the dependencies of a given analysis node, it would be possible to reconstruct the environment needed to run it. By example, how might a system like this work?

Say we use `@analysis` as the namespace for our specifying each analysis node's properties, and someone has provided bindings to objects in `numpy` (we'll give an example of how these bindings might work below, but for now assume they work analogously to the module structure of numpy, ie. `@numpy:ndarray = numpy.ndarray`). We can assume they are provided by the package maintainers, but that's not necessary: this is my node and it takes what I want it to!

In pseudocode, I could define some analysis node for, say, converting an RGB image to grayscale under my namespace as `@jonny:bin-spikes` like this:

```
<#bin-spikes>
  a @analysis:node
    Version "⩾1.0.0"

  hasDescription
    "Convert an RGB Image to a grayscale image"

  inputType
    @numpy:ndarray
      # ... some spec of shape, dtype ...

  outputType
    @numpy:ndarray
      # ... some spec of shape, dtype ...

  params
    bin_width int
      default 10
```

I have abbreviated the specification of shape and datatype to not overcomplicate the pseudocode example, but say we successfully specify a 3 dimensional (width x height x channels) array with 3 channels as input, and a a 2 dimensional (width x height) array as output. An optional `bin_width` parameter with default "10" can also be provided.

The code doesn't run on nothing! We need to specify our node's dependencies. Say in this case we need to specify an operating system image `ubuntu`, a version of `python`, a system-level package `opencv`, and a few python packages on `pip`. We are pinning specific versions with semantic versioning, but the syntax isn't terribly important. Then we just need to specify where the code for the node itself comes from:

```
  dependsOn
    @ubuntu:"^20.*":x64
    @python:"3.8"
    @apt:opencv:"^4.*.*"
    @pip:opencv-python:"^4.*.*"
```

```
      .extraSource "https://pywheels.org/"
    @pip:numpy:"^14.*.*"

  providedBy
    @git:repository
      .url "https://mygitserver.com/binspikes/fast-binspikes.git"
      .hash "fj9wbkl"
    @python:class "/main-module/binspikes.py:Bin_Spikes"
      method "run"
```

Here we can see the practical advantage of the "inverted" link-based system rather than an object-oriented-like approach. `@ubuntu` refers to a specific software image that would have a specific `providedBy` value, but both `@apt` and `@pip` can have different repositories that they pull packages from, and for a given version and repository there will be multiple possible software binaries for different CPU architectures, python versions, etc. Rather than needing to specify a generalized specification format, each of these different types of links could specify their own means of resolving dependencies: a `@pip` dependency requires some `@python` version to be specified. Both require some operating system and architecture. If we hadn't provided the `.extraSource` of pywheels (for ARM architectures), someone who had defined some link between a given architecture and `@pip` could be proposed as a way of finding the package.

Our `@analysis.node` protocol gives us several slots to connect different tools together, each in turn presumably provides some minimal functionality expected by that slot: eg. `inputType` can expect `@numpy:ndarray` to specify its own dependencies, the programming language it is written in, shape, data type, and so on. Coercing data between chained nodes then becomes a matter of mapping between the `@numpy` and, say a `@nwb` namespace of another format. In the same way that there can be multiple, potentially overlapping between data schemas, it would then be possible for people to implement mappings between intermediate data formats as-needed. This gives us an opportunity to build pipelines that use tools from multiple languages, a problem typically solved by manually saving, loading, and cleaning intermediate data.

This node also becomes available to extend, say someone wanted to add an additional input format to my node:

```
<@friend#bin-spikes>
  extends @jonny:bin-spikes

  inputType
    @pandas:DataFrame

  providedBy
     ...
```

They don't have to interact with my potentially messy codebase at all, but it is automatically linked to my work so I am credited. One could imagine a particular analysis framework implementation that would then search through extensions of a particular node for a version that supports the input/output combinations appropriate

for their analysis pipeline, so the work is cumulative. This functions as a dramatic decrease in the size of a unit of work that can be shared.

This also gives us healthy abstraction over implementation. Since the functionality is provided by different, mutable namespaces, we're not locked into any particular piece of software — even our `@analysis` namespace that gives the `inputType` etc. slots could be forked. We could implement the dependency resolution system as, eg. a docker container, but it also could be just a check on the local environment if someone is just looking to run a small analysis on their laptop with those packages already installed.

The relative complexity required to define an analysis node, as well as the multiple instances of automatically computed metadata like dependency graphs hints that we should be thinking about tools that avoid needing to write it manually. We could use an `Example_Framework` that provides a set of classes and methods to implement the different parts of the node (a la luigi). Our `Bin` class inherits from `Node`, and we implement the logic of the function by overriding its `run` method and specify an `output` file to store intermediate data (if requested by the pipeline) with an `output` method. Our class is within a typical python package that specifies its dependencies, which the framework can detect. We also specify a `bin_width` as a `Parameter` for our node, as an example of how a lightweight protocol could be bidirectionally specified as an interface to the linked data format: we could receive a parameterization from our pseudocode metadata specification, or we could write a framework with a `Bin.export_schema()` that constructs the pseudocode metadata specification from code.

```
from Example_Framework import Node, Param, Target

class Bin(Node):
  bin_width = Param(dtype=int, default=10)

  def output(self) → Target:
    return Target('temporary_data.pck')

  def run(self, input:'numpy.ndarray') → 'numpy.ndarray':
    # do some stuff
    return output
```

Now that we have a handful of processing nodes, we could then describe some `@workflow`, taking some `@nwb:NWBFile` as input, as inferred by the `inputType` of the `bin-spikes` node, and then returning some output as a `:my-analysis:processed` child beneath the input. We'll only make a linear pipeline with two stages, but there's no reason more complex branching and merging couldn't be described as well.

```
<#my-analysis>
  a @analysis:workflow

  inputType
    @jonny:bin-spikes:inputType

  outputName
```

```
    input:my-analysis:processed

  step Step1 @jonny:bin-spikes
  step Step2 @someone-else:another-step
    input Step1:output
```

Since the parameters are linked from the analysis nodes, we can specify them here (or in the workflow). Assuming literally zero abstraction and using the tried-and-true "hardcoded dataset list" pattern, something like:

```
<#my-project>
  a @analysis:project

  hasDescription
    "I gathered some data, and it is great!"

  researchTopic
    @neuro:systems:auditory:speech-processing
    @linguistics:phonetics:perception:auditory-only

  inPaper
    @doi:10.1121:1.5091776

  workflow Analysis1 @jonny:my-analysis
    globalParams
      .Step1:params:bin_width 10

    datasets
      @jonny.mydata1:v0.1.0:raw
      @jonny.mydata2:^0.2.*:raw
      @jonny.mydata3:⩾0.1.1:raw
```

And there we are! The missing parameters like `outputName` from our workflow can be filled in from the defaults. Our project is an abstract representation of the analysis to be performed and where its output will be found - in this case as `:processed` beneath each dataset link. From this very general pseudocode example it's possible to imagine executing the code locally or on some remote server, pulling the data from our p2p client, installing the environment, and duplicating the resulting data to the clients configured to mirror our namespace. This system would work similar to the combination of configuration and lockfiles from package managers: we would give some abstract specification for a project's analysis, but then running it would create a new set of links with the exact dependency graph, links to intermediate products, and so on. We get some inkling of where we're going later by also being able to specify the paper this data is associated with, as well as some broad categories of research topics so that our data as well as the results of the analysis can be found.

From here we could imagine how existing tools might be integrated without needing to be dramatically rewritten. In addition to wrapping their parameters, functions, and classes with the above `Node` class, we could imagine our analysis linking framework providing some function to let us indicate code within a package and prompt us for any missing pieces like dependency specification from, for example, old style

python packages that don't require it. For packages that don't have an explicit declarative parameterization, but rely on programmatically created configuration files, we could imagine a tool ingestion function being able to extract default fields and then refer to them with a `fromConfig @yaml` link. A single tool need not be confined to a single analysis node: for example a tool that requires some kind of user interaction could specify that with an `@analysis:interactive` node type that feeds its output into a subsequent analysis node. There are infinitely more variations to be accounted for — but adapting to them is the task of an extensible linking system.

As soon as we extend our relatively static protocol to the realm of arbitrary code we immediately face the question of security. Executing arbitrary code from many sources is inherently dangerous and worthy of careful thought, but any integrative framework becomes a common point where security practices could be designed into the system as opposed to the *relative absence of security practices of any kind* in most usages of scientific software. There is no reason to believe that this system is intrinsically more dangerous than running uninspected packages from PyPI, which, for example, have been known to steal AWS keys and environment variables [418] . Having analysis code and its dependency graph specified publicly presents opportunities for being able to check for identified vulnerabilities at the time of execution — a role currently filled by platform tools like GitHub's dependabot or npm's audit. Running code by default in containers or virtual environments could be a way towards making code secure by default.

So that's useful, but comparable to some existing pipelining technologies. The important part is in the way this hypothetical analysis framework and markup interact with our data system — it's worth unpacking a few points of interaction.

A dataset linked to an analysis pipeline and result effectively constitutes a "unit of analysis." If I make my data publicly available, I would be able to see all the results and pipelines that have been linked to it. Within a single pipeline, comparing the results across a grid of possible parameterizations gives us a "multiverse analysis [419] " for estimating the effects of each parameterization for free. Conversely, "rules of thumb" for parameter selection can be replaced by an evaluation of parameters and results across prior applications of the pipeline. Since some parameters like model weights in neural networks are not trivial to reproduce, and their use is linked to the metadata of the dataset they are applied to, all analyses contribute to a collection of models like the DeepLabCut model zoo decreasing the need for fine tuning on individual datasets and facilitating metalearning across datasets.

Across multiple pipelines, a dataset need no longer be dead on publication, but can instead its meaning and interpretation can continuously evolve along with the state of our tools and statistical practices. Since pipelines themselves are subject to the same kind of metadata descriptions as datasets are, it becomes to find multiple analysis nodes that implement the same operation, or to find multiple pipelines that perform similar operations despite using different sets of nodes. Families of pipelines that are applied to semantically related datasets would then become the substrate for a field's state of the art, currently buried within disorganized private code repositories and barely-descriptive methods sections. Instead of a 1:1 relationship where one dataset is interpreted once, we could have a many-to-many relationship where a cumulative body of data is subject to an evolving negotiation of interpretation over time — ostensibly how science is *"supposed to"* work.

This system also allows the work of scientific software developers to be credited

according to use, instead of according to the incredibly leaky process of individual authors remembering to search for all the citations for all the packages they may have used in their analysis. Properly crediting the work of software developers is important not only for equity, but also for the reliability of scientific results as a whole. A common admonishment in cryptography is to "never roll your own crypto," but that's how most homebrew analysis code works, and the broader state of open source scientific code is not much better without incentives for maintenance. Bugs in analysis code that produce inaccurate results are inevitable and rampant [186, 183, 184, 185] , but impossible to diagnose when every paper writes its own pipeline. A common analysis framework would be a single point of inspection for bugs and means of providing credit to people who fix them. When a bug is found, rather than irreparably damaging collective confidence in a field, it would then be trivial to re-run all the analyses that were impacted and evaluate how their results were changed.

Finally, much like how we are building towards the social systems to support federations for sharing data, integrating analysis pipelines into a distributed network of servers is a means of realizing a generalized Folding@Home-style distributed computing grid [420, 421] . Existing projects like F@H and the Pacific Research Platform [422] show the promise of these distributed computing systems for solving previously-intractable problems, but they require large amounts of coordination and are typically centrally administered towards a small number of specific projects with specific programming requirements. With some additional community systems for governance, resource management, and access, they become tantalizingly in-reach from the system we are describing here. We will return to that possibility after discussing experimental tools.

11.3.2 Experimental Frameworks

Across from the tools to analyze data are those to collect it, and tools to integrate the diversity of experimental practice are a different challenge altogether: *everyone needs completely different things!* Imagine the different stages of research as a cone of complexity: at the apex we can imagine the relatively few statistical outcomes from a family of tests and models. For every test statistic we can imagine a thousand analysis scripts, for every analysis script we might expect a thousand data formats, and so the complexity of the thousand experimental tools used to collect each type of data feels ... different.

Beyond a narrow focus of the software for performing experiments itself, the surrounding contextual knowledge work largely lacks a means of communication and organization. Methods sections have been increasingly marginalized, abbreviated, pushed to the end, and relegated to the supplement. The large body of work that is not immediately germane to experimental results, like animal care, engineering instruments, lab management, etc. have effectively no formal means of communication — and so little formal means of credit assignment.

Extending our ecosystem to include experimental tools has a few immediate benefits: bridging the gap between collection and sharing of data would resolve the need for format conversion as a prerequisite for inclusion in the linked system, allowing the expression of data to be a fluid part of the experiment itself. It would also serve as a means of building a body of cumulative contextual knowledge in a creditable system.

I have previously written about the design of a generalizable, distributed experimental framework [405] , so to avoid repeating myself, and since many of the ideas from the section on analysis tools apply here as well, I will be relatively brief.

We don't have the luxury of a natural formalism like a DAG to structure our experimental tools. Some design constraints on experimental frameworks might help explain why:

- They need to support a wide variety of instrumentation, from **off-the-shelf parts,** to **proprietary instruments** as are common in eg. microscopy, to **custom, idiosyncratic designs** that might make up the existing infrastructure in a lab. Writing and testing embedded code that controls external hardware is a wholly different kind of difficulty than writing analysis tools.

- To be supportive, rather than constraining, they need to be able to **flexibly perform many kinds of experiments** in a way that is **familiar to patterns of existing practice.** That effectively means being able to coordinate heterogeneous instruments in some "task" with a flexible syntax.

- They need to be **inexpensive to implement,** in terms of both money and labor, so it can't require buying a whole new set of hardware or dramatically restructuring existing research practices.

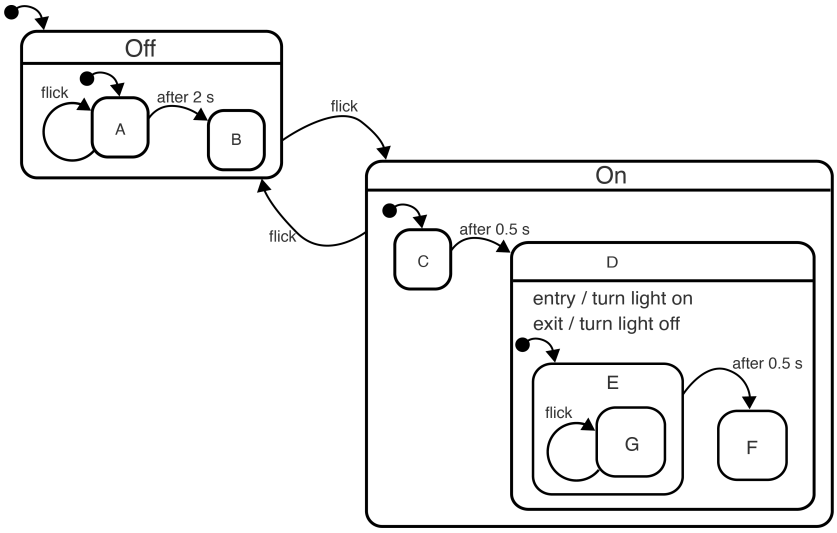
- They need to be **accessible and extensible,** with many different points of control with different expectations of expertise and commitment to the framework. It needs to be useful for someone who doesn't want to learn it to its depths, but also have a comprehensible codebase at multiple scales so that reasearchers can **easily extend** it when needed.

- They need to be designed to support **reproducibility and provenance,** which is a significant challenge given the heterogeneity inherent in the system. On one hand, being able to produce *data that is clean at the time of acquisition* simplifies automated provenance, but enabling experimental replication requires multiple layers of abstraction to keep the idiosyncracies of an experiment separable from its implementation: it shouldn't require building *exactly* the same apparatus with *exactly* the same parts connected in *exactly* the same way to replicate an experiment.

- Ideally, they need to support **cumulative labor and knowledge organization,** so an additional concern with designing abstractions between system components is allowing work to be made portable and combinable with others. The barriers to contribution should be extremely minimal, not requiring someone to be a professional programmer to make a pull request to a central library, and contributions should come in many modes — code is not the only form of knowing and it's far from the only thing needed to perform an experiment.

Here, as in the domains of data and analysis, the temptation to universalize is strong, and the parts of the problem that are emphasized influence the tools that are produced. A common design tactic for experimental tools is to design them as state machines, a system of states and transitions not unlike the analysis DAGs above. One such nascent project is BEADL [423] from a Neurodata Without Borders working group. BEADL is an XML-based markup for standardizing a behavioral task as an abstraction of finite state machines called statecharts. Experiments are fully abstract from their hardware implementation, and can be formally validated in simulations. The working group also describes creating a standardized ontology and metadata schema for declaring all the many variable parameters for experiments, like reward sizes, stimuli, and responses [424] . This group, largely composed of members from the Neurodata Without Borders team, understandably emphasize systematic description and uniform metadata as a primary design principle.

Personally, I *like* statecharts. The problem is that it's not necessarily natural to express things as statecharts as you would want to, or in the way that your existing, long-developed local experimental code does. There are only a few syntactical features needed to understand the following statechart: blocks are states, they can be inside each other. Arrows move between blocks depending on some condition. Entering and exiting blocks can make things happen. Short little arrows from filled spots are where you start in a block, and when you get to the end of the chart you go back to the first one. See the following example of a statechart for controlling a light, described in the introductory documentation and summarized in the figure caption:



"When you flick a lightswitch, wait 0.5 seconds before turning the light on, then once it's on wait 0.5 seconds before being able to turn it back off again. When you flick it off, wait 2 seconds before you can turn it on again.

They have an extensive set of documents that defend the consistency and readability of statecharts on their homepage, and my point here is not to disagree with them. My point is instead that tools that aspire to the status of generalized infrastructure can't ask people to dramatically change the way they think about and do science. There are many possible realizations of any given experiment, and each is more or less natural to every person.

The problem here is really one of emphasis, BEADL seeks to solve problems with

inconsistencies in terminology by standardizing them, and in order to do that seeks to standardize the syntax for specifying experiments.

This means of standardization has many attractive qualities and is being led by very capable researchers, but I think the project is illustrative of how the differing structures of problems constrain the possible space of tooling. Analysis tasks are often asynchronous, where the precise timing of each node's completion is less important than the path dependencies between different nodes. Analysis tasks often have a clearly defined set of start, end, and intermediate cache points, rather than branching or cyclical decision paths that change over multiple timescales. Statecharts are a hierarchical abstraction of finite state machines, the primary advantage of which is that they are better able to incorporate continuous and history-dependent behavior, which cause state explosion in traditional finite-state machines.

The difficulty of a controlled ontology for experimental frameworks is perhaps better illustrated by considering a full experiment. In Autopilot, a full experiment can be parameterized by the `.json` files that define the task itself and the system-specific configuration of the hardware. An example task from our lab consists of 7 behavioral shaping stages of increating difficulty that introduce the animal to different features of a fairly typical auditory categorization task. Each stage includes the parameters for at most 12 different stimuli per stage, probabilities for presenting lasers, bias correction, reinforcement, criteria for advancing to the next stage, etc. So just for one relatively straightforward experiment, in one lab, in one subdiscipline, there are **268 parameters** – excluding all the default parameters encoded in the software.

How might we approach this problem differently, to accommodate diversity of thought styles and to be complementary to our data and analysis systems? The primary things we need from our experimental frameworks are a) to be able to link a particular realization of an experiment with the metadata that describes it, and b) to be able to produce similarly metadata-rich data. Rather than linked data indicating code as in our analysis frameworks, we might invert our strategy and think about code that draws from linked data.

As an example, Autopilot [405] approaches the problem by avoiding standardizing *experiments* themselves, instead providing smaller building blocks of experimental tools like hardware drivers, data transformations, etc. and emphasizing understanding their use in *context.* This approach sacrifices some of the qualities of a standardized system like being a logically complete or guaranteeing a standardized vocabulary in order to better support integrating with existing work patterns and making work cumulative. Because we can't possibly predict the needs and limitations of a totalizing system, we split the problem along a different set of concerns, those of the elements of experimental practice, and give facility for describing how they are used together.

For concrete example, we might imagine the lightswitch in an autopilot-like framework like this:

```
from autopilot.hardware.gpio import Digital_Out
from time import sleep
from threading import Lock

class Lightswitch(Digital_Out):
  def __init__(self,
```

```
    off_debounce: float = 2,
    on_delay:     float = 0.5,
    on_debounce:  float = 0.5):
    """
    Args:
      off_debounce (float):
        Time (s) before light can be turned back on
      on_delay (float):
        Time (s) before light is turned on
      on_debounce (float):
        Time (s) after turning on that light can't be turned off
    """
    self.off_debounce = off_debounce
    self.on_delay     = on_delay
    self.on_debounce  = on_debounce

    self.on = False
    self.lock = Lock()

  def switch(self):
    # use a lock to make sure if
    # called while waiting, we ignore it
    if not self.lock.acquire():
      return

    # if already on, switch off
    if self.on:
      self.on = False
      sleep(self.off_debounce)

    # otherwise switch on
    else:
      sleep(self.on_delay)
      self.on = True
      sleep(self.on_debounce)

    self.lock.release()
```

The class `Lightswitch` inherits from the `Digital_Out` class, which in turn inherits from `GPIO` and eventually `Hardware`. This hierarchy of inheritance carries with it a progressive refinement of meaning about what this class does. The terms `off_debounce`, `on_delay`, and `on_debounce` are certainly not part of a controlled ontology, but the context of their use bounds their meaning. Rather than being bound by, for example, the abstract `Latency` term from interlex, we have defined terms that we need to make a hardware object do what we need it to. These terms don't have too much meaning on their own — there isn't even much in this class to uniquely identify it as a "lightswitch" beyond its name, it is just a timed digital output. What makes them meaningful is how they are used.

The way Autopilot handles various parameters are part of set of layers of abstraction that separate idiosyncratic logic from the generic form of a particular `Task`

or `Hardware` class. The general structure of a two-alternative forced choice task is shared across a number of experiments, but they may have different stimuli, different hardware, and so on. Autopilot `Task`s use abstract references to classes of hardware components that are required to run them, but separates their implementation as a system-specific configuration so that it's not necessary to have *exactly the same* components plugged into *exactly the same* GPIO pins, etc. Task parameters like stimuli, reward timings, etc. are similarly split into a separate task parameterization that both allow `Task`s to be generic and make provenance and experimental history easier to track. `Task` classes can be subclasses to add or modify logic while being able to reuse much of the structure and maintain the link between the root task and its derivatives — for example one task we use that starts a continuous background sound but otherwise is the same as the root `Nafc` class. The result of these points of abstraction is to allow exact experimental replication on inexactly replicated experimental apparatuses.

This separation of the different components of an experiment is a balance between reusable code and clear metadata: we might allow freedom of terminology for each individual class, but by designing the system to encourage reuse of flexible classes we reduce the number of times unique terms need to be redefined. For example, we can imagine a trivial use of our lightswitch inside a task measuring an experimental subject's estimation of time intervals: we toggle the switch once some analog sensor reaches a certain threshold, and then the subject tries to press a button at the same time as the light turns on after a fixed delay. While this is very similar to how Autopilot currently works, note that we are using pseudocode to indicate how it might extend the system we're describing.

```
from autopilot import Task
from autopilot.data.modeling import Field
from datetime import datetime, timedelta

class Controlled_Switch(Task):
  """
  A [[Discipline::Psychophysics]] experiment
  to measure [[Research Topic::Interval Estimation]].
  """

  class Params(Task.Param_Spec):
    on_delay: '@si:seconds' = Field(
        description="Delay (s) before turning light on",
        parameterizes="@jonny:hardware:Lightswitch")
    threshold: float = Field(
        description="Flick switch above this value",
        is_a="@interlex:Threshold")

  class TrialData(Task.TrialData_Spec):
    switch_time: datetime = Field(
        description="Time the switch was flicked")
    target_time: datetime = Field(
        description="Time the subject should respond")
    response_time: datetime = Field(
        description="Time the subject did respond")
```

```
  error: timedelta = Field(
      description="Difference between target and response",
      is_a="@psychophys:ReactionTime")


HARDWARE = {
  'sensor': 'Analog_In',
  'button': 'Digital_In',
  'lightswitch': '@jonny:hardware:Lightswitch'
}

def __init__(self,
    on_delay:float,
    threshold:float):
  self.on_delay = on_delay
  self.threshold = threshold

  super(Controlled_Switch, self).__init__()
  self.poll()

def poll(self):
  while self.running:
    if self.hardware['sensor'].value > self.threshold:
      self.hardware['lightswitch'].switch()
      switch_time = datetime.now()
      target_time = switch_time + self.on_delay

      # Wait for the subject to press the button
      response_time = self.hardware['button'].wait()

      # Send the data for storage
      self.node.send(key="DATA", value={
          'switch_time': switch_time,
          'target_time': target_time,
          'response_time': response_time,
          'error': target_time - response_time
        })
```

In this example, we first define a data model (see section 3.2 - Data in [425]) for the Tasks `Params`, the data that the task produces as `TrialData`, and the `HARDWARE` that the task uses. Our `Params` each have a type hint indicating what type of data they are, as well as a `Field` that gives further detail about them. Specifically, we have exposed the Lightswitch's `on_delay` parameter, indicated that it will be in seconds by referring to some namespace that defines SI units `@si` and that it parameterizes the lightswitch object that we defined above. The `TrialData` is similarly annotated, and by default Autopilot will use this specification to create an hdf5 table to store the values. The `HARDWARE` dictionary makes abstract references the hardware objects that will be made available in the task, each of which would have its configuration — which GPIO pin they are plugged into, the polarity of the signal, etc. — using some local system configuration. Finally, the single `poll()` method continuously compares the value of the sensor to the threshold, switches the lightswitch when

the threshold is crossed, records the time the button was pressed, and sends it for storage with its network node.

As before, we are using our experimental framework as an interface to our linked data system. Currently, Autopilot uses a semantic wiki to organize technical knowledge and to share plugins - `https://wiki.auto-pi-lot.com`. In this case, I would write my task and hardware classes inside a git repository and then add them to Autopilot's plugin registry, which uses a form to fill in semantic properties and allows further annotation in free text and semantic markup.

We could instead imagine being able to document the task in its docstring, including describing the relevant subdiscipline, research topic, and any other relevant metadata. Rather than manually entering it in the wiki, then, we might export the triplet annotations directly from the class and make them available from my `@jonny` namespace and mirroring that to the wiki. Since the plugin specifies its dependencies using standard Python tools, it would then be possible for other researchers to use its task and hardware objects by referring to them as above.

In our pseudocode, the (abbreviated) exported metadata for this task might look like this:

```
<#tasks:Controlled_Switch>
  a @autopilot:Task

  hasDescription
    "A Psychophysics experiment
    to measure Interval Estimation."

  Discipline "Psychophysics"
  Research_Topic "Interval Estimation"

  Params
    on_delay @si:seconds
      hasDescription " ... "
      parameterizes @jonny:hardware:Lightswitch
    ...

  TrialData
    switch_time @python:datetime
    ...

  usesHardware
    @autopilot:hardware:Analog_In
      hasID "sensor"
    @autopilot:hardware:Digital_In
      hasID "button"
    @jonny:hardware:Lightswitch
      hasID "lightswitch"
```

and we might combine it with metadata that describes our particular use of it like this, where we combine that task with a series of other `levels` that shape the behavior, make it more challenging, or measure something else entirely:

```
<#projects:my-project>
  a @autopilot:protocol
  experimenter @jonny
   ...

  level @jonny:tasks:Controlled_Switch
    on_delay 2
    threshold 0.5
    graduation @autopilot:graduation:ntrials
      n_trials 200

  level @jonny:tasks:Another_Task
     ...

  hardwareConfig
    button @autopilot:hardware:Digital_In
      gpioPin 17
      polarity 1
    sensor @autopilot:hardware:Analog_In
      usesPart @apwiki:parts:<Part_Number>
       ...
```

On the other side, our output data can be automatically exported to NWB[35]. Our experimental framework knows that data contained within a `TrialData` model is a `@nwb:behavior:BehavioralEvents` object, and can combine it with the metadata in our task docstring and system configuration. If we needed more specific data export - say we wanted to record the timeseries of the analog sensor - we could use the same `is_a` parameter to declare it as a `@nwb:TimeSeries` and create an extension to store the metadata about the sensor alongside it[36].

So while our code is mildly annotated and uses a mixture of standard and nonstandard terminology, we make use of the structure of the experimental framework to generate rich provenance to understand our data and task in context. It's worth pausing to consider what this means for our infrastructural system as a whole

To start, we have a means of integrating our task with the knowledge that precedes it in the hardware and system configuration that runs it. In addition to documenting plugins, among others, the Autopilot wiki also has schema for custom built and off-the-shelf hardware like sensors and sound cards. These correspond to local hardware configuration entries that link them to the hardware classes required to use them[37]. That link can be used bidirectionally: metadata about the hardware used to perform an experiment can be used in the experiment and be included with the produced data data, but the data from experiments can also be used to document the hardware. That means that usage data like calibrations and part longevity can be automatically collected and contributed to the wiki and then used to automatically configure hardware in future uses. This makes using the experimental framework more powerful, but also makes building a communal library of technical knowledge a normal part of doing experiments. Though the wiki is a transitional medium towards what we will discuss in the next section, since contributions are tracked and versioned that allows a currently undervalued class of knowledge work to be creditable.

[35] Recall that we're using NWB for the sake of concreteness, but this argument applies to any standardized data format.

[36] Though this is a description of something we could build towards, v0.5.0 (at the time of writing released as alpha) of Autopilot has a data modeling framework that should make this possible in future versions.

[37] Not yet, but this is planned development for future versions.

This gives us a different model of designing and engineering experiments than we typically follow. Rather than designing most of it from scratch or decoding cryptic methods sections, researchers could start with a question and basic class of experiment, browse through various implementations based on different sets of tools, see which hardware they and analogous experiments use, which is then linked to the code needed to run it. From some basic information researchers would then be most of the way to performing an experiment: clone the task, download the necessary system configuration information to set up the hardware, make incremental modifications to make the experiment match what they had designed, all the while contributing and being credited for their work.

Much of this is possible because of the way that Autopilot isolates different components of an experiment: hardware is defined separately from tasks, both are separate from their local configuration. In addition to thinking about how to design tools for our infrastructural system, we can also think of the way it might augment existing tools. Another widely used and extremely capable tool, Bonsai [426, 427] , is based on XML documents that combine the pattern of nodes that constitute an experiment with specific parameters like a crop bounding box. That makes sharing and reusing tasks difficult without exactly matching the original hardware configuration, but we could use our metadata system to *generate* code for Bonsai in addition to consuming data from it. Given some schematic pattern of nodes that describes the operation of the experiment, we could combine that with the same notion of separable parameterization and hardware configurations as we might use in Autopilot to generate the XML for a bonsai workflow. As with analytical tools, our infrastructural system could be used to make a wide array of experimental tools interoperable with an evolving set of vernacular metadata schema.

Together, our data, experimental, and analytical infrastructures would dramatically reshape what is possible in science. What we've described is a complete provenance chain that can be traced from analyzed results back through to the code and hardware used to perform the experiment. Trivially, this makes the elusive workflow where experimental data is automatically scooped up and analyzed as soon as it is collected that is typically a hard-won engineering battle within a single lab the normal mode of using the system. Developing tools that give researchers control over the mode of exported data renders the act of cleaning data effectively obsolete. The role of our experimental tool is to be able to make use of collected technical knowledge, but also to lower the barriers to using the rest of the system by integrating it with normal experimental practice.

The effects on collaboration and metascience are deeper though. Most scientific communication describes collecting and analyzing a single dataset. Making sense of many experiments is only possible qualitatively as a review or quantitatively as meta-analysis. Even if we have a means of linking many datasets and analysis pipelines as in the previous section, the subtle details in how a particular experiment is performed matter: things as small as milliseconds of variation in valve timings through larger differences in training sequences or task design powerfully influence the collected data. This makes comparing data from even very similar experiments — to say nothing of a class of results from a range of different experiments — a noisy and labor-intensive statistical process, to the degree that it's possible at all. This system extends the horizon of meta-analysis to the experiment itself and turns experimental heterogeneity into a strength rather than a weakness. Is some result a byproduct of some unreported parameter in the experimental code? Is a result only visible when

comparing across these different conditions? Individual experiments only allow a relatively limited set of interpretations and inferences to be drawn, but being able to look across the variation in experimental design would allow phenomena to be described in the full richness supported by available observations.

This would also effectively dissolve the "file drawer problem." [428, 429] Though malice is not uncommon in science, I think it's reasonably fair to assume that most researchers do not withhold data given a null result in order to "lie" about an effect, but because there is no reward for a potentially laborious cleaning and publication process. Collecting data that is clean and semantically annotated at the time of acquisition resolves the problem. Even without the analysis or paper, being able to index across experiments of a particular kind would make it possible to have a much fairer look at a landscape distorded by the publication process and prevent us from repeating the same experiments because no one has bothered to publish the null. This would also open new avenues for collaboration as we will explore in the next section.

To review:

We have described a system of three component modalities: **data, analytical tools, and experimental tools** connected by a **linked data** layer. We started by describing the need for a **peer-to-peer** data system that makes use of **data standards** as an onramp to linked metadata. To interact with the system, we described an identity-based linked data system that lets individual people declare linked data resources and properties that link to **content addressed** resources in the p2p system, as well as **federate** into multiple larger organizations. We described the requirements for **DAG-based analytical frameworks** that allow people to declare individual nodes for a processing chain linked to code, combine them into workflows, and apply them to data. Finally, we described a design strategy for **component-based experimental frameworks** that lets people specify experimental metadata, tools, and output data.

This system as described is a two-layer system, with a few different domains linked by a flexible metadata linking layer. The metadata system as described is not merely *inert* metadata, but metadata linked to code that can *do something* — eg. specify access permissions, translate between data formats, execute analayis workflows, parameterize experiments, etc. Put another way, we have been attempting to describe a system that *embeds the act of sharing and curation in the practice of science.* Rather than a thankless post-hoc process, the system attempts to provide a means for aligning the daily work of scientists so that it can be cumulative and collaborative. To do this, we have tried to avoid rigid specifications of system structure, and instead described a system that allows researchers to pluralistically define the structure themselves.

11.4 Shared Knowledge

The remaining set of problems implied by the infrastructural system sketched so far are the *communication* and *organization* systems that make up the interfaces to maintain and use it. We can finally return to some of the breadcrumbs laid before:

the need for negotiating over distributed and conflicting data schema, for incentivizing and organizing collective labor, and for communicating information within and without academia.

The communication systems that are needed double as *knowledge organization* systems. Knowledge organization has the rosy hue of something that might be uncontroversial and apolitical — surely everyone involved in scientific communication wants knowledge to be organized, right? The reality of scientific practice might give a hint at our naivete. Despite being, in some sense, itself an effort to organize knowledge, *scientific results effectively have no system of explicit organization.* There is no means of, say, "finding all the papers about a research question."[38] The problem is so fundamental it seems natural: the usual methods of using search engines, asking around on Twitter, and chasing citation trees are flex tape slapped over the central absence of a system for formally relating our work as a shared body of knowledge.

[38] Also see Eve Marder's recent short and characteristically refreshing piece which in part discusses the problem of keeping up with scientific literature the context of maintaining the joy of discovery [430] .

Information capitalism, in its terrifying splendor, here too pits private profit against public good. Analogously to the necessary functional limitations of SaaS platforms, artificially limiting knowledge organization opens space for new products and profit opportunities. In their 2020 shareholder report, RELX, the parent of Elsevier, lists increasing the number of journals and papers as a primary means of increasing revenue [255] . This represents a shift in their business model from subscriptions to deals like open access, which according to RELX CEO Erik Nils Engström "is where revenue is priced per article on a more explicit basis" [431] .

In the next breath, they describe how "in databases & tools and electronic reference, representing over a third of divisional[39] revenue, we continued to drive good growth through content development and enhanced machine learning [ML] and natural language processing [NLP] based functionality."

[39] RELX is a huge information conglomerate, and scientific publication is just one division.

What ML and NLP systems are they referring to? The 2019 report is a bit more revealing (emphases mine):

> Elsevier looks to enhance quality by building on its premium brands and **grow article volume** through **new journal launches,** the expansion of open access journals and growth from emerging markets; and add value to core platforms by implementing capabilities such as **advanced recommendations on ScienceDirect and social collaboration through reference manager and collaboration tool Mendeley.**
>
> **In every market, Elsevier is applying advanced ML and NLP techniques** to help researchers, engineers and clinicians perform their work better. For example, in research, ScienceDirect Topics, a free layer of content that enhances the user experience, uses **ML and NLP techniques to classify scientific content and organise it thematically,** enabling users to get faster access to relevant results and related scientific topics. The feature, launched in 2017, is proving popular, generating 15% of monthly unique visitors to ScienceDirect via a topic page. **Elsevier also applies advanced ML techniques that detect trending topics per**

> **domain,** helping researchers make more informed decisions about their research.
>
> **Coupled with the automated profiling and extraction of f unding body in-**
>
> **formation from scientific articles,** this process supports the whole researcher journey; from planning, to execution and funding. [432]

Reading between the lines, it's clear that the difficulty of finding research is a feature, not a bug of their system. Their explicit business model is to increase the number of publications and sell organization back to us with recommendation services. The recommendation system might be free[40], but the business is to maintain the self-reinforcing system of prestige where researchers compete for placement in highly visible journals to stand out among a wash of papers, in the process reifying the mythology [433] of the "great journals." With semantic structure to locate papers, it becomes much more difficult to sell high citation count as a product — people can find what they need, rather than needing to pay attention to a few high-profile journals. Without it, which papers might a paper discovery system created by a publisher recommend? The transition from a strictly journal-based discovery system to a machine learning powered search and feed model mirrors the strategic displacement of explicit organization by search in the rest of the digital economy, and presents similar opportunities for profit. Every algorithmically curated feed is an opportunity to sell ad placement[41] — which they proudly describe as looking very similar to their research content [436, 371] .

[40] "free"

[41] a strategy that the reprehensible digital marketing disciplines call "native advertising" [434, 435]

The extended universe of profitmaking from knowledge disorganization gets more sinister: Elsevier sells multiple products to recommend 'trending' research areas likely to win grants, rank scientists, etc., algorithmically filling a need created by knowledge disorganization. The branding varies by audience, but the products are the same. For pharmaceutical companies "scientific opportunity analysis" promises custom reports that answer questions like "Which targets are currently being studied?" "Which experts are not collaborating with a competitor?" and "How much funding is dedicated to a particular area of research, and how much progress has been made?" [437] . For academics, "Topic Prominence in Science" offers university administrators tools to "enrich strategic research planning with portfolio overviews of their own and peer institutions." Researchers get tools to "identify experts and potential cross-sector collaborators in specific Topics to strengthen their project teams and funding bids and identify Topics which are likely to be well funded." [438] This reflects RELX's transition "from electronic reference, information reference tools, databases to [...] analytics and decision tools." [**?**] Publishing is old news, the real money is in tools for extending control through the rest of the process of research.

These tools are, of course, designed for a race to the bottom — if my colleague is getting an algorithmic leg up, how can I afford not to? Naturally only those labs that *can* afford them and the costs of rapidly pivoting research topics will benefit from them, making yet another mechanism that reentrenches scientific inequity for profit. Knowledge disorganization, coupled with a little surveillance capitalism that monitors the activity of colleagues and rivals [233, 439] , has given publishers powerful control over the course of science, and they are more than happy to ride algorithmically amplified scientific hype cycles in fragmented research bubbles all the way to the bank.

One more turn of the screw: the ability of the (former) publishers to effectively invent the metrics that operationalize "prestige" in the absence of knowledge organization sytems gives them broad leverage with governments and funding agencies. In an environment of continuously dwindling budgets and legislative scrutiny, seemingly mutually beneficial platform contracts offer the sort of glossy comfort that only predictive analytics can. In 2020 the National Research Foundation of Korea (NRF) and Elsevier published a joint report that used a measurement derived from citation counts - "Field-weighted citation impact", or FWCI - to argue for the underrated research prestige of South Korea [440] . While I don't dispute the value of South Korea's research program, the apparent bargain that was struck is chilling. South Korea gets a very fancy report arguing that more scientists in other countries should work with theirs, and Elsevier gets to cement itself into the basic operation of science. Elsevier controls the journals that can guarantee high citation counts *and* the metrics built on top of them. The Brain Korea program Phase II report [42] [441] , issued just before the 2009 formation of the NRF argued that rankings and funding should be dependent on citation counts. The NRF now relies on SciVal and their FWCI measurement as a primary means of ranking researchers and determining funding, built into the Brain Korea 21 funding system [442, 443] . Without exaggeration, scientific disorganization and reliance on citation counts allowed Elsevier to buy control over the course of research in South Korea.

[42] the result of another corporate collaboration with the Rand corporation.

The consequences for science are hard to overstate. In addition to literature search being an unnecessarily huge sink of time and labor, science operates as a wash of tail-chasing results that only rarely seem to cumulatively build on one another. The need to constantly reinforce the norm that purposeful failure to cite prior work is research misconduct is itself a symptom of how engaging with a larger body of work is both extremely labor intensive and *strictly optional* in the communication regime of journal publication. The combination of more publications translating into more profit and the strategic disorganization of science contributes to conditions for scientific fraud. An entirely fraudulent paper can be undetectable even by domain experts. Since papers can effectively be islands — given legitimacy by placement in a journal strongly incentivized to accept all comers — and there is no good means of evaluating them in context with their immediate semantic neighbors, investigating fraud is extremely time consuming and almost entirely without reward. And since traditional peer review happens once, rather than as a continual public process, the only recourse outside of posting on PubPeer is to wait on journal editorial boards to self-police by reviewing each individual complaint. Forensic peer-reviewers have been ringing the alarm bell, saying that there is "no net" to bad research [444] , and brave and highly-skilled investigators like Elisabeth Bik have found thousands of papers with evidence of purposeful manipulation [445, 446] . The economic structure of for-profit journals pits their profit model against their function as providing a venue for peer review — the one function most scientists are still sympathetic to. Trust in science is critical for addressing our most dire problems from global pandemics to climate change [447] , but attitudes towards scientists are lukewarm at best [448] . Even when it isn't fake news, why would anyone trust us when it's *effectively impossible* to find or assess the quality of scientific information? [449] Not even scientists can: despite the profusion of papers, by some measures progress in science has slowed to a crawl [450] .

While Chu and Evans [450] correctly diagnose *symptoms* of knowledge disorganization like the need to "resort to heuristics to make continued sense of the field" and reliance on canonical papers, by treating the journal model as a natural phenomenon

and citation as the only means of ordering research, they misattribute root *causes.* The problem is not people publishing *too many papers,* or a *breakdown of traditional publication hierarchies,* but the *staggering profitability of knowledge disorganization.* Knowledge disorganization is precisely the precondition of information-as-capital and the outcome of its concentration by our century's robber barons (see [451]). Their prescription for "a clearer hierarchy of journals" misses the role of organizing scientific work in journals ranked by prestige, rather than by the content of the work, as a potentially major driver of extremely skewed citation distributions. It also misses the publisher's stated goals of *publishing more papers* within an ecosystem of algorithmic recommendations, and there is nothing recommendation algorithms love recommending more than things that are already popular. Without diagnosing knowledge disorganization as a core part of the business model of scientific publishers, we can be led to prescriptions that would make the problem worse.

It's hard to imagine an alternative to journals that doesn't look like, well, journals. While a full treatment of the journal system is outside the scope of this paper, the system we describe here renders them *effectively irrelevant* by making papers as we know them *unnecessary.* Rather than facing the massive collective action problem of asking everyone to change their publication practices on a dime, by reconsidering the way we organize the surrounding infrastructure of science we can flank journals and replace them "from below" with something qualitatively more useful.

Beyond journals, the other technologies of communication that have been adopted out of need, though not necessarily design, serve as desire paths that trace other needs for scientific communication. As a rough sample: Researchers often prepare their manuscripts using platforms like Google Drive, indicating a need for collaborative tools in perparation of an idea. When working in teams, we often use tools like Slack to plan our work. Scientific conferences reflect the need for federated communication within subdisciplines, and we have adopted Twitter as a de facto platform for socializing and sharing our work to a broader audience. We use a handful of blogs and other sites like OpenBehavior [452] , Open Neuroscience, and many others to index technical knowledge and tools. Last but not finally, we use sites like PubPeer and ResearchGate for comment and criticism.

These technologies point to a few overlapping and not altogether binary axes of communication systems.

- **Durable vs Ephemeral** - journals seek to represent information as permanent, archival-grade material, but scientific communication also necessarily exists as contextual, temporally specific snapshots.

- **Structured vs Chronological** - scientific communication both needs to present itself as a structured basis of information with formal semantic linking, but also needs the chronological structure that ties ideas to their context. This axis is a gradient from formally structured references, through intermediate systems like forums with hierarchical topic structure that embeds a feed, to the purely chronological feed-based social media systems.

- **Messaging vs Publishing** - Communication can be person-to-person, person-to-group with defined senders and recipients, or person-to-all statement to an undefined public. This ranges from private DMs through domain-specific tool

indexes like OpenBehavior through the uniform indexing of Wikipedia.

- **Public vs. Private** - Who gets to read, who gets to contribute? Communication can be composed of entirely private notes to self, through communication in a lab, collaboration group, discipline, and landing in the entirely public realm of global communication.

- **Formal vs. Informal** - Journal articles and encyclopedia-bound writing that conforms to a particular modality of expression vs. a vernacular style intended to communicate with people outside the jargon culture.

- **Push vs. Pull** - Do you go to get information from a reference location, or does information come to you as an alert or message? Or, generally, where is the information "located," is an annotation pushed and overlaid on a document, or stored elsewhere requiring the audience to explicitly pull it?

"Peer reviewed vs. unrefereed" is purposely excluded as an axis of communication tools, as the ability to review and annotate multiple versions of a document — subject to the context of the medium — should be a basic part of any communication system. Fear over losing the at once immutable but also paradoxically fragile ecosystem of journal-led peer review is one of the first strawmen that stops consideration of radically reorganizing scientific communication[43]. The belief that peer review as we know it is an intrinsic part of science is ahistorical (eg. [454]), and the belief that journal-led peer review is somehow a unique venue for evaluating scientific work ignores the immense quantity of criticism and discussion that happens in almost every communicative context, scientific and otherwise. The notion that the body of scientific knowledge is best curated by passing each paper through a gauntlet of three anonymous reviewers, after which it becomes Fact is ridiculous on its face. Focusing on preserving peer review is a red herring that unnecessarily constrains the possible forms of scientific communication. Instead we will try and sketch systems that address the needs for communication and knowledge organization left unmet precisely because of the primacy of peer reviewed journal publications.

[43] For a recent example, see the responses to Dan Goodman's argument why he has stopped doing pre-publication peer review altogether [453**?**]

Clearly a variety of different types of communication tools are needed, but there is no reason that each of them should be isolated and inoperable with the others. We have already seen several of the ideas that help bring an alternative into focus. Piracy communities demonstrate ways to build social systems that can sustain distributed infrastructure. Federated and protocol-based systems show us that we don't need to choose between a single monolithic system or many disconnected ones, but can have a heterogeneous space of tools linked by a basic protocol. The semantic web gives us the unfilled promise of triplet links as a very general means of structuring data and building interfaces for disparate systems. We can bridge these lessons with some from wiki culture to get a more practical sense of distributed governance and organization. Together with our sketches of data, analytical, and experimental tools we can start imagining a system for coordinating them — as well as displacing some of the more intractable systems that misstructure the practice of science.

11.4.1 The Wiki Way

> If we take radical collaboration as our core, then it becomes clear that extending Wikipedia's success doesn't simply mean installing more copies of wiki software for different tasks. It means figuring out the key principles that make radical collaboration work. What kinds of projects is it good for? How do you get them started? How do you keep them growing? What rules do you put in place? What software do you use? [455]

> So that's it — insecure but reliable, indiscriminate and subtle, user hostile yet easy to use, slow but up to date, and full of difficult, nit-picking people who exhibit a remarkable community camaraderie. Confused? Any other online community would count each of these "negatives" as a terrible flaw, and the contradictions as impossible to reconcile. Perhaps wiki works because the other online communities don't. [456**? ?**] and in WhyWikiWorks

a[44]

Aside from maybe the internet itself, there is no larger public digital knowledge organization effort than Wikipedia. While there are many lessons to be learned from Wikipedia itself, it emerged from a prior base of thought and experimentation in radically permissive, self-structuring read/write — sometimes called "peer production" [457] — communities. Wikis are now quasi-ubiquitous[45], perhaps largely thanks to Wikipedia, but its specific history and intent to be an *encyclopedia* entwines it with a very particular technological and social system that obscures some of the broader dreams of early wikis.

Aaron Swartz recounts a quote from Jimmy Wales, co-founder of Wikipedia:

> "I'm not a wiki person who happened to go into encyclopedias," Wales told the crowd at Oxford. "I'm an encyclopedia person who happened to use a wiki." [458]

And further describes how this origin and mission differentiates it from other internet communities:

> But Wikipedia isn't even a typical community. Usually Internet communities are groups of people who come together to discuss something, like cryptography or the writing of a technical specification. Perhaps they meet in an IRC channel, a web forum, a newsgroup, or on a mailing list, but the focus is always something "out there", something outside the discussion itself.
>
> But **with Wikipedia, the goal is building Wikipedia.** It's not a community set up to make some other thing, it's a community set up to make itself. And since Wikipedia was one of the first sites to do it, we know hardly anything about building communities like that. [455]

We know a lot more now than in 2006, of course, but Wikipedia still has outsized structuring influence on our beliefs about what Wikis can be. Wikipedia has since spawned a large number of technologies and projects like Wikidata and Wikimedia Commons, each with their own long and occasionally torrid histories. I won't dwell on the obvious and massive feat of collective organzation that the greater Wikipedia

[44] Interestingly, this quote is almost, but not exactly the same as that on Ward's wiki: "So that's it - insecure, indiscriminate, user-hostile, slow, full of difficult, nit-picking people, and frivolous. Any other online community would count each of these strengths as a terrible flaw. Perhaps wiki works because the other online communities do not." I can't tell if Ward Cunningham wrote the original entry in the wiki, but in any case seems to have found a bit of optimism in the book.

[45] though their corporate manifestations would probably be unrecognizable to the project early wiki users imagined.

project represents — we should build on and interoperate with its projects and respect the amount of work the foundation and its editors have put in to preserve free access to information, but learning from its imperfections is more useful to us here, especially for things that aren't encyclopedias. The dream of a centralized, but mass-edited "encyclopedia of everything" seems to be waning, and its slow retreat from wild openness has run parallel to a long decline in contributors [457, 459] . Throughout that time, there has been a separate (and largely skeptical) set of wiki communities holding court on what a radically open web can be like, inventing their worlds in real time. These communities have histories that are continuous with one another, and in their mutual reaction and inspiration sometimes teach similar lessons from across the divides of their very different structure.

The first wiki was launched in 1995[46] (it's still up) and came to be known as Ward's wiki after its author WardCunningham. Technically, it was extremely simple: a handful of TextFormattingRules and use of WikiCase where if you JoinCapitalizedWords you create a link to a (potentially new) WikiPage — and the ability for anyone to edit any page. These very simple WikiDesignPrinciples led to a sprawling and continuous conversation that spanned more than a decade and thousands[47] of pages that, because of the nature of the medium, is left fully preserved in amber. Those conversations are a history of thought on what makes wiki communities work (eg. WhyWikiWorks, WhyWikiWorksNot), and what is needed to sustain them.

[46] it's complicated: http://wiki.c2.com/?WardsWikiTenthAnniversary

[47] 23,244 unique page names according to the edit history, but the edit history was also purposely pruned from time to time.

One tension that emerged early without satisfying resolution is the balance between "DocumentMode" writing that serves as linearly-readable reference material, similar to that of Wikipedia, and "ThreadMode" writing that is a nonlinear representation of a conversation. Order vs spontaneity is a fundamental challenge of inventing culture in plaintext. The purpose of using a wiki as opposed to other technologies that existed at the time like bulletin boards, newsgroups, IRC, etc. was that it provided a means of fluid structure[48]. The parallel need to communicate and attribute work made it a seeming inevitability that even if you went out of your way to restructure a lot of writing into a sensible DocumentMode page, someone would soon after create a new horizontal divider and start a fresh ThreadMode section.

[48] Giving a means of organizing the writing of the Portland Pattern Repository was the reason for creating Ward's Wiki in the first place.

Ward Cunningham and other more organizationally-oriented contributors opposed ThreadMode (eg. ThreadModeConsideredHarmful, InFavorOfDissertation) for a number of reasons, largely due to the ThreadMess and WikiChaos it had the potential of creating.

> I occasionally suggest how this site should be used. My GoodStyle suggestions have been here since the beginning and are linked from the edit page should anyone forget. I have done my best to discourage dialog InFavorOfDissertation which offers a better fit to this medium. I've been overruled. I will continue to make small edits to pages for the sake of brevity. – WardCunningham [460]

Most pages are thus a combination of both, usually with some DocumentMode text at the top with ThreadMode conversations interspersed throughout without necessarily having any clean delineation between the two. Far from just being raw disorder, this mixed mode of writing gave it a peculiar character of being *both* a folk reference for a library of concepts *as well as* a history of discussion that made the contingency of that reference material plain. Beka Valentine put it well:

> c2wiki is an exercise in dialogical methods. of laying bare the fact that knowledge and ideas are not some truth delivered from On High, but rather a social process, a conversation, a dialectic, between various views and interests [461]

This tension and its surrounding discussions point to the need for multiple representations of a single idea: that both the social and reference representations of a concept are valuable, but aren't necessarily best served by being represented in the same place. There was relatively common understanding that the intended order of things was to have many ThreadMode conversations that would gradually be converted to DocumentMode in a process of BrainStormFirstCleanLater. Many proposed solutions orbit around making parallel pages with similar names (like <pagename>Discussion) to clean up a document while preserving the threads (though there were plenty of interesting alternatives, eg. DialecticMode)[49].

[49] Contemporary wikis have continued this conversation, see DocumentsVsMessages on communitywiki.org

Wikipedia, in its attendant WikiEngine MediaWiki, cut the Gordian Knot by splitting each page into a separate *Article* and *Talk* pages, with the talk page in its own **Namespace** – eg. Gordian_Knot vs Talk:Gordian_Knot. Talk pages resemble a lot of the energy of early wikis: disorganized, sometimes silly, sometimes angry, and usually charmingly pedantic. Namespaces extend the traditional "everything is a page" notion encoded in the WikiCase link system by giving different pages different roles. In addition to having parallel conversations on articles and talk pages, it is possible to have template pages that can be included on wiki pages with `{% raw %}{{double curly bracket}}{% endraw %}` syntax – eg. Template:Citation_Needed renders `{% raw %}{{Citation needed}}{% endraw %}` as [citation needed]. Talk pages have their own **functional differentiation,** with features for threading and annotating discussions that aren't present on the main article pages (see Wikipedia:Flow [462]). Generalized beyond the context of wikis, functional differentiation of a single item into its multiple representations is relatively common in computing: eg. this document exists as a git repository, the rendered page, a pdf, hypothes.is annotations, etc.

The complete segregation of discussion to Talk pages is driven by Wikipedia's exclusivity as an encyclopedia, with reminders that it is the "sole purpose" peppered throughout the rules and guidelines. The presence of messy subjective discussions would of course be discordant with the very austere and "neutral" articles of an encyclopedia. There are no visible indications that the talk pages even exist in the main text, and so even deeply controversial topics have no references to the conversations in talk pages that contextualize them — despite this being a requested feature by both administrators and editors [463] .

Talk pages serve as one of the primary points of coordination and conflict resolution on Wikipedia, and also provide a low-barrier entrypoint for questions posed to a space they perceive to be "an approchable community of experts" [464] . The separation of Talk pages and the labyrinthine rules governing their use function to obscure the dialogical and collective production of knowledge at the heart of wikis and Wikipedia. The body of thought that structures Wikipedia, most of which is in its Wikipedia:* namespace, is immense and extremely valuable, but is largely hidden except from those who care to look for it. Since Wikipedia is "always already there" often without trace of its massively collective nature, relatively few people ever contribute to it. Reciprocally, since acknowledging personal contribution is or point of view is explicitly against some of its core policies and traditions, there is little public

credit outside the Wikipedia community itself for the labor of maintaining it.

The forking of Wards Wikis into the first SisterSites teaches a parallel strain of lessons. Ward's Wiki started as a means of organizing knowledge for the Portland Pattern Repository[50], a programming community (referred to as DesignPatterns below), and in 1998 they were overwhelmed with proponents of ExtremeProgramming (or XP), which caused the first fissure in the wiki:

> XP advocates seemed to be talking about XP at every possible opportunity and seemingly on every page with content the least bit related to software development. This annoyed a number people who were here to discuss patterns, leading to the tag XpFreeZone, as a request not to talk about ExtremeProgramming on that page.
>
> It was difficult to pick out the DesignPatterns discussion on RecentChanges[51], because most of the activity was related to ExtremeProgramming. Eventually, most of the DesignPatterns people left, to discuss patterns in a "quieter" environment, and people started referring to this site as WardsWiki instead of the PortlandPatternRepository [460]

One of the first and most influential Sister Sites was Meatball Wiki, described on Wards Wiki:

> SunirShah founded MeatballWiki to absorb and enlarge the discussion of what wiki and wiki like sites might be. That discussion still simmers here. But here it can take on a negative tone sounding more like complaining. On meatball, under Sunir's careful leadership, the ideas, wild or not, stay amazingly upbeat. - SisterSites

MeatballWiki became the spiritual successor to Ward's Wiki, which at that point had its own momentum of culture less interested in being the repository of wiki thought[52]. Meatball has its own prolific history of thought, including reflections on its very existence as a SisterSite. These were a series of discussions that melded thoughts from open source computing with social systems; in part: RightToFork, RightToLeave, EnlargeSpace, and TransClusion.

What can be done when the internal divisions in a wiki community and the weight of its history make healthy contribution impossible? The simplest is to exercise the RightToLeave, as it is almost always possible to just stop being part of a digital community. This approach is clearly the most destructive, as it involves abandoning the emotional bonds of a community, prior work (see the WikiMindWipe where a user left and took all their contributions with them), and doesn't necessarily provide an alternative that alleviates the cause of the tension. The next idea is to *fork* the community, where its body — in the case of wikis the pages and history — can be duplicated so that it can proceed along two parallel tracks. Exercising the right to fork is, according to Meatball, "people exercising their RightToLeave whilst maintaining their emotional stake" [466] .

The discussion around the Right to Fork on Meatball is far from uniformly positive, and is certainly colored by the strong presence of its BenevolentDictator Sunir Shah who viewed it as a last resort after all attempts at ConflictResolution have failed. They point to the potentially damaging effects of a fork, like bitterness, disputes over content ownership (see MeatballIsNotFree), and potentially an avoidance of conflict resolution that is a normal and healthy part of any community. Others place it more in the realm of a radical *political* action rather than a strictly social action. Writing about the fork of OpenOffice to LibreOffice, Terry Hancock writes:

[50] The initial motivations are actually stunningly close to the kinds of communication and knowledge organization problems we are still solving today (even in this piece) > Cunningham had developed a database to collect the contributions of the listserv members. He had noticed that the content of the listserv tended to get buried, and therefore the most recent post might be under-informed about posts which came before it. The way around this problem was to collect ideas in a database, and then edit those ideas rather than begin anew with each listserv posting. Cunningham's post states that "The plan is to have interested parties write web pages about the People, Projects and Patterns that have changed the way they program. Short stories that hint at patterns are welcome too." As to the rhetorical expectations, Cunningham added "The writing style is casual, like email or netnews, but doesn't have to be so repetitive since the things being discussed don't disappear. Think of it as a moderated list where anyone can be moderator and everything is archived. It's not quite a chat, still, conversation is possible." - [465]

[51] Recent Changes was the dominant, if not controversial means of keeping track with recent wiki traffic, see RecentChangesJunkie

[52] There seems to have been an overriding belief that theoretical ideas about wikis and wiki culture belong on Meatball Wiki, from WikiWikiWebFaq: > Q: Do two separate wikis ever merge together to create one new wiki? Has this happened before? Keep in mind that I don't just mean two different pages within a wiki. (And for that matter, where is an appropriate page where I can post questions about the history of all wikis, not just this one?) > > A1: I don't know of any such wiki merge, nor of any discussion of the history of all wikis. Such a discussion should probably reside (if created) on MeatballWiki.

> [In] proprietary software [a] political executive decision can kill a project, regardless of developer or user interest. But with free software, the power lies with the people who make it and use it, and the freedom to fork is the guarantee of that power. [...] The freedom to fork a free software project is [a] "tool of revolution" intended to safeguard the real freedoms in free software. [467]

Forking digital communities can be much less acrimonious than physically-based communities because of the ability to EnlargeSpace given by the medium:

> In order to preserve GlobalResources, create more public space. This reduces limited resource tension. Unlike the RealWorld, land is cheap online. In effect, this nullifies the TragedyOfTheCommons by removing the resource pressure that created the "tragedy" in the first place. **You can't overgraze the infinity.** - [468]

Enlarging space has the natural potential to make the broader social scene bewildering with a geyser of pages and communities, but can be made less damaging by having mechanisms to link histories, trace their divergence, and potentially resolve a fork as is common in open source software development. Forking is then a natural process of community regeneration, allowing people to regroup to make healthier spaces when needed, where the fork is itself part of the history of the community rather than an unfathomable rift.

Forking communities is not the same as forking community resources: "you can't fork a community [...] what you can do is fork the content and to *split* the community" [469] . As described so far, a fork divides people into unreconciled and separate communities. In some cases this makes forking difficult, in others it makes it impossible: the prime example, again, is Wikipedia. It is simply too large and too culturally dominant to fork. Even though it is technically possible to fork Wikipedia, if you succeeded, then what? Who would come with you to build it, and who would that be useful for? This is partly a product of its totalizing effort to be an encyclopedia of everything (what good would *another* encyclopedia of everything be?) but also the weight of history: you won't get enough long-encultured Wikipedians to join you, and you probably won't be able to recruit a new generation of them on your own.

The last major effort to fork Wikipedia was in 2002 with an effort led by Edgar Enyedy to move the Spanish Wikipedia to The Enciclopedia Libre Universal en Español [470, 471] . Though it was brief and unsuccessful, Enyedy claims that because Jimmy Wales was worried about other non-English communities following their lead, he and the other admins capitulated to the demands for no advertising and a transfer to a .org domain, among others[53]. Even a politically symbolic fork is dependent on the perceived threat to the original project, and that window seems to have been closed after 2002.

[53] Jimmy Wales, naturally, disputes this characterization of events.

The cultural tensions and difficulties that lead other wikis and projects to fork have taken their toll on the editorship and culture of Wikipedia. The community is drawn into dozens of conflicting philosophical camps: the Deletionists[54] vs. the Inclusionists, Eventualists vs. Immediatists, Mergists vs. Separatists, and yes even a stub page for Wikisecessionism. Editorship has steadily declined from a peak in 2007. Its relatively invisible community systems make it mostly a matter of chance or ideology that new contributors are attracted in the first place. In its calcification of norms, largely to protect against legitimate challenges to the integrity of the ency-

[54] Also see Association of Wikipedians Who Dislike Making Broad Judgments About the Worthiness of a General Category of Article, and Who Are in Favor of the Deletion of Some Particularly Bad Articles, but That Doesn't Mean They Are Deletionists

clopedia, any newcomers that do find their way into editing now have little chance to catch a foothold in the culture before they are frustrated by (sometimes algorithmic) rejection [457, 459] .

Arguably all internet communities have some kind of life cycle, so the question becomes how to design systems that support healthy forking without replicating the current situation of fragmentation. Wikis, including Meatball and MediaWiki, as well as other projects like Xanadu often turn to **transclusion** — or being able to reference and include the content of one wiki (or wiki page) in another. Rather than copying and pasting, the remote content is kept updated with any changes made to it.

Transclusion naturally brings with it a set of additional challenges: Who can transclude my work? Whose work can I transclude? Can my edits be propagated back to their work? What can be transcluded, at what level of granularity, and how? While before we had characterized splitting communities as an intrinsic part of a fork, that need not be the case in a system built for transclusion. Instead relationships post-fork are then made an *explicit social process* within the system, where even if a community wants to work as separate subgroups, it is possible for them to arrive at some agreement over what they want to share and what they want to keep separate. This kind of decentralized work system resembles radical organizing tactics like affinity groups where many autonomous groups fluidly work together or separately on an array of shared projects without aspiring to create "one big movement" [472] . Murray Bookchin describes:

> The groups proliferate on a molecular level and they have their own "Brownian movement." Whether they link together or separate is determined by living situations, not by bureaucratic fiat from a distant center. [...]
>
> [N]othing prevents affinity groups from working together closely on any scale required by a living situation. They can easily federate by means of local, regional or national assemblies to formulate common policies and they can create temporary action committees (like those of the French students and workers in 1968) to coordinate specific tasks. [...] As a result of their autonomy and localism, the groups can retain a sensitive appreciation of new possibilities. Intensely experimental and variegated in lifestyles, they act as a stimulus on each other as well as on the popular movement. [473]

To cherrypick a few lessons from more than 25 years of thought from tens of thousands of people: The differing models of document vs. thread modes and separate article vs. talk pages show us that using **namespaces** is an effective way to bridge multimodal expression on the same topic across perceived timescales or other conflicting communicative needs. This is especially true when the namespaces have **functional differentiation**[55] like the tools for threading conversations on Wikipedia Talk pages and the parsing and code generation tools of Templates. These namespaces need to be **visibly crosslinked** both to preserve the social character of knowledge work, but also to provide a means of credit assignment and tool development between namespaces. Any communication system needs to be designed to **prioritize ease of leaving** and **ease of forking** such that a person can take their work

[55] Tim Berners-Lee described this notion of functional differentiation in a much more general way in describing the nature of the URI: > The technology should define mechanisms wherever possible without defining policy. > > because we recognize here that many properties of URIs are social rather than technical in origin. > > Therefore, you will find pointers in hypertext which point to documents which never change but you will also find pointers to documents which change with time. You will find pointers to documents which are available in more than one format. You will find pointers to documents which look different depending on who is asking for them. There are ways to describe in a machine or human readable way exactly what sort of repeatability you would expect from a URI, but the architecture of the Web is that that is for

and represent it on some new system or start a new group to encourage experimentation in governance models and technologies. One way of accomplishing these goals might be to build a system around **social transclusion** such that work across many systems and domains can be linked into a larger body of work without needing to create a system that becomes too large to fork. The need for communication across namespaces and systems, coupled with transclusion further implies the need for **bidirectional transclusion** so that in addition to being able to transclude something in a document, there is visible representation on the original work being transcluded (eg. commented on, used in an analysis, etc.) by allowed peers and federations.

These lessons, coupled with those from private bittorrent trackers, linked data communities, and the p2p federated system we have sketched so far give us some guidelines and motivating examples to build a varied space of communication tools to communicate our work, govern the system, and grow a shared, cumulative body of knowledge.

11.4.2 Rebuilding Scientific Communication

It's time to start thinking about interfaces. We have sketched our system in turtle-like pseudocode, but directly interacting with our linking syntax would be labor intensive and technically challenging. Instead we can start thinking about tools for interacting with it in an abstract way. Beneath every good interface we're familiar with, a data model lies in wait. A .docx file is just a zipped archive full of xml, so a blank word document that contains the single word "melon" is actually represented (after some preamble) like:

```
<w:body>
  <w:p
    w14:paraId="0667868A"
    w14:textId="50600F77"
    w:rsidR="002B7ADC"
    w:rsidRDefault="00A776E4">
    <w:r>
        <w:t>melon</w:t>
    </w:r>
  </w:p>
</w:body>
```

Same thing with jupyter notebooks, where a block of code:

```
>>> rating = 100
>>> print(f'I rate this dream {rating}')
'I rate this dream 100'
```

is represented as JSON (simplified for brevity):

```
{
  "cell_type": "code",
```

```
  "id": "thousand-vermont",
  "outputs": [{
    "name": "stdout",
    "output_type": "stream",
    "text": [
      "I rate this dream 100\n"
    ]
  }],
  "source": [
    "rating = 100\n",
    "print(f'I rate this dream {rating}')"
  ]
}
```

So we are already used to working with interfaces to data models, we just need to think about what kind of interfaces we need for a scientific communication system.

Let's pick up where we left off with our linked data and tools. Recall that we had a `project` named `#my-project` that linked an experiment, a few datasets that it produced, and an analysis pipeline that we ran on it. We *could* just ship the raw numbers from the analysis, wash our hands of it, and walk straight into the ocean without looking back, but usually scientists like to take a few additional steps to visualize the data and write about what it means.

To explore the communicative tools that might be useful, we can start by considering traditional documents, and attempt to generalize them by separating their form as "units" or "cells" of information with accompanying metadata from their representation in interfaces for interacting and communicating about them.

Documents & Notebooks

Say we have reached the stage where we are writing a brief summary of our experiment and analysis, but not yet at the stage of writing a "formal" scientific paper. We might do so in a notebook-like [474] environment with different kinds of "cells," specifically cells that execute *code* and cells that render *markdown.* We want to plot some of the results of our analysis, so to do that we might load the data and use matplotlib [475] to make our point:

{% include notebook.html html="/infrastructure/assets/notebooks/smile.html" %}

Our notebook file would then include an array of JSON objects that describe the contents of its cells. For example, our data loading cell would look something like this:

```
{
   "cell_type": "code",
   "execution_count": 2,
   "id": "rapid-information",
   "metadata": {
    "scrolled": true
   },
   "outputs": [
```

```
    " ... "
   ],
   "source": [
    "x, y, sizes = get_data('@jonny:my-project:Analysis1')"
   ]
}
```

The "`outputs`" description has been abbreviated above, but it describes to the jupyter notebook server how to display it. Regular text piped through stdout is represented like this:

```
{
  "name": "stdout",
  "output_type": "stream",
  "text": [
    "Downloading dataset @jonny:my-dataset\n",
    "--------------------\n"
  ]
}
```

And multiple output types can be combined in a single cell, for example a widget like our loading progress bar is described like this:

```
{
  "data": {
    "application/vnd.jupyter.widget-view+json": {
      "model_id": "5799ac2959084a4596ffbad3f9940f48",
      "version_major": 2,
      "version_minor": 0
    },
    "text/plain": [
      "  0%|          | 0/100 [00:00<?, ?it/s]"
    ]
    },
    "metadata": {},
  "output_type": "display_data"
}
```

where the `model_id`, `version_major`, and `version_minor` describe which rendering code to use for the cell, similarly to the "metadata that indicates code" that we discussed in analytical frameworks.

Notice that there is already a metadata field! In order to link our notebook to our analysis — and thus to our extended graph of data, experiment, etc. — we could do it manually, but since we're thinking about interfaces we can also imagine that our `p2p_framework` is capable of filling it in for us. We don't need to invent a new metadata protocol for JSON, JSON-LD is already quite similar to the syntax we've been using already. For simplicity, say we use a `@comms` ontology to denote various features of our communication system. Our data loading function might then populate a field in our cell like this:

```
"metadata": {
  "scrolled": true,
  "@comms:usesData": "@jonny:my-project:Analysis1"
}
```

Other frameworks might make their own metadata annotations, like an indication that we're plotting some feature of the data, or performing some statistical analysis on the data. These annotations might be responsive to the parameterization of the function call or its results, but if we emphasize a design process that makes interfaces at multiple levels we could also imagine using something like iPython "magic commands" to declare metadata for our cell. For example, each cell is automatically assigned a random combination of words as an ID, but if we wanted to be able to specifically refer to a cell we could give it an explicit one:

```
%%meta @comms:cellID smilePlot
plt.scatter(x, y, s=sizes)
```

We're familiar with two types of cells, code and markdown, but we can extend our thinking to arbitrary cell types. What is a cell? A cell has a a) **type** that indicates its capabilities and representation, b) **metadata** that describes it, we can also generalize that to include *arguments* that parameterize it, and c) the content, or information contained by the cell. The jupyter document model more or less reflects this already, but in its base model only has code, markdown, and raw cell types, and the metadata field is unstructured. Its extension system allows for additional cell types as well as restructuring the program more generally, but since we're focused on self-contained documents we'll limit our discussion to additional cell types.

From this it's relatively trivial to imagine additional cell types that serve common needs in academic writing: a citation cell type[56] that takes a BibTeX object (or its fields) as arguments and then preserves the full metadata as well as renders it in a chosen style. A figure cell type that takes an image or plot and a caption. A contributor cell type that takes an author's name, affiliation, ORCID, email, and so on. Currently jupyter extensions use the NPM registry, but we could imagine being able to use other people's cell types directly by referring to them like `@jonny:celltypes:citation`.

[56] The original Jupyter Notebook paper describes the need for this near the end [474] .

Notebooks have multiple levels of metadata, so we can also specify document-level metadata that describe the type of our document (like a `@schema:ScholarlyArticle`), its `creativeWorkStatus` as a `Draft`, our authorship information, permissions, and whatever else we'd like. But what is a document? In the case of our jupyter notebook, it's a series of cell descriptions in a JSON array. Trivially, a document is a cell that contains other cells. What about in the other direction? The contents of our cells are *also* a cell-like system. The very notion of a programming language is a means of mapping structured syntax to machine instructions, and to do that code (in some languages) is interpreted or compiled by parsing it into an abstract syntax tree that relates data and its structuring metadata. Markdown can also be thought of as a series of subcells, where using a `# header` indicates how the text is to be represented as compared to `*italic*` text or `[links](https://link.com)`. The use of a programming language or markup syntax is represented by the `cell_type` field, which the notebook server knows to translate `"code"` to mean Python and `"markdown"` to mean its particular flavor of markdown (of which there are several).

This points towards a model of **recursive** cells that can contain other cells. An editor could, for example, draw from templating engines like liquid, where an abstract representation of the content of a cell could include a `{{ content }}` marker that indicates that additional cells can be included inside of it. Recursive models, coupled with structuring metadata that indicates the relationship between a parent and child cell could then be used to model compound concepts. Another simple example using citation might be to have a cell with one child cell containing a reference to another work that ours `@cito:disagrees_with` [476] , and another child cell that in turn contains some writing in markdown and a plot. Recursive cells also naturally

lend themselves to **transclusion** by making each of the individual subcomponents of a document referenceable with full granularity. We will expand on both compount concepts and transclusion in a moment in talking about the extension of our cellular system to wikis.

Before we go beyond a document system that would be unrecognizable to most scientists, and thus yet another nice pipedream, it's important to pause on the continuity with existing document systems. Microsoft Word, or Word-like WYSIWYG editors like LibreOffice or Google docs are the dominant mode of preparing academic documents. Word-like editors are *already* create recursive cell-like documents, though their interface obscures them. They support semantic markup like heading styles (though their use compared to manual formatting is far from universal [477]), and every paragraph can be considered a cell, with the default paragraph styling as its metadata and additional styled elements like bolded words as sub-cells. It should then be possible to import existing word documents into a cellular document system. Care should also be taken to smooth the cognitive transition from word-like editors: Jupyter currently treats cells as being strictly separate, and new cells need to be created manually. Instead it should be possible for cells to "recede into the background" and be created with common gestures like a double return to make a new paragraph. The "insert" menu used to create things like tables or images is already a familiar tool in word-like editors, so the notion of adding elaborated types like citations shouldn't be that big of a lift.

The other major document preparation tool modalities are markup syntaxes and their associated builders like LaTeX. Though TeX-like tools have an exceedingly opinionated and obscure design history [478] , they have three major affordances: 1) document-level structure provided by document classes, packages, and the options they provide, 2) environments that enclose some section of text between `\begin{}` and `\end{}` and provide some specific functionality or formatting like lists, and 3) commands that accept arguments and modify some smaller unit of text like creating a link with `\href{https://url.com}{link text}`. Each of these maps onto a cellular document system, with document-level metadata or the templates commonly used to render markdown, and cells that take arguments to approximate environments and commands. Markdown extensions like MyST [479] make this translation even more straightforward with direct analogies to LaTeX commands and environments and their "role" and "directive" counterparts in reStructuredText. Since the goal should be a 1:1 relationship between source code and visual editor, the difference between representing a cell visually versus in markup should be left as a matter of author preference.

Bidirectional translation from a WYSIWYG editor to its markup is not a trivial task — the mediawiki team started writing theirs in 2011 and rolled it out as a default fea-

ture in 2020 [480] . It's a careful balance between ease of use, power of syntax, and accomodation of historical usage patterns. Markdown is on one extreme of ease with only a handful of markup elements to learn, but has a relatively steep learning curve to do anything more complex. On the other end is the wonderful dokieli [481] (and see Sarven's masterpiece [482] , spiritual cousin to this document), which does essentially everything that we want our linked documents to do, but requires authors to write their documents in HTML and manually manage the semantic markup. Extending Notebooks to use recursive cells with reusable types sacrifices some of the ability to directly edit the source of a document as a potential way to balance familiarity and expressiveness.

Notebooks, with some architectural and interfaces then become a straightforward way of breaking up the scientific paper as a singular unit of knowldge work when embedded in a linked data system. Their use in scholarly publishing has been proposed many times before, but our linking system lets us resolve some of the largest outstanding limitations [483] : dependency management [484] , archiving [485] , and discovery, among others. The same gradient of access control rules we discussed in controlling access to sensitive data would support a process of gradual publication of smaller units of work, from a private demo in our lab meeting to a public part of scientific discourse.

What happens when we invite other people to respond?

Forums & Feeds

What if we think of our documents as "threads" and their cells as "posts?" What makes a cellular document a document is some (relatively arbitrary) notion of a 'root' cell that contains the others — ie. for notebooks a JSON array of cells. That could be trivially reformulated as cells with metadata indicating that they are `PartOf` a document, each indicating their `position` or linked to the cells they are before and after. If we also allow cells to be `inReplyTo` each other, we have the basis of a threaded communication system continuous with documents. Where cells in a linear document have at most one preceeding and succeeding cell, multiple replies allow a tree structure that maps onto the patterns of most contemporary social media. Metadata that describes category and content extends this to include the structure of forums, and could be the basis of a rich continuum of media spanning order and chaos, permanence and ephemerality, between the *magnum opus* and the shitpost: media absent but sorely needed in academic communication.

Traditional forums like phpBB and contemporary social media operate from a single host with a fixed interface and representation of posts. What would a communication system that decouples hosting, identity, interface, and format look like? We can draw inspiration from the "fediverse," a collection of interoperable software platforms and protocols. The fediverse makes it possible to communicate across radically different interfaces: someone using Funkwhale, which resembles music software like spotify, can communicate with people on PeerTube, a p2p video streaming program like YouTube, and Mastodon, a microblogging medium like Twitter. Rather than a single host, instances of each of these programs are hosted independently and can choose to federate with other instances to enable communication between them. Most of these programs use the ActivityPub [399] protocol, which defines a standard set of capabilities for client-server and server-server communication.

Mastodon posts (or "toots") already resemble the kind of document-interoperable medium hinted at above. For example this post is represented in (abbreviated) JSON:


```
{
  "to":[
    "https://www.w3.org/ns/activitystreams#Public"
  ],
  "cc":[
    "https://social.coop/users/jonny/followers"
  ],
  "id": "107328829457619549",
  "created_at": "2021-11-23T22:52:49.044Z",
  "in_reply_to_id": "107328825611826508",
  "in_reply_to_account_id": "274647",
  "visibility": "public",
  "url": "https://social.coop/@jonny/107328829457619549",
  "content": "<p>and making a reply to the post to show the in_reply_to and context fields</p>",
  "account":
  {
      "id": "274647",
      "username": "jonny",
      "fields":
      [ ... ]
  },
  "media_attachments": [],
  "mentions": [],
  "tags": [],
}
```


As described previously, ActivityPub supports linked data with JSON-LD – a remarkable feat despite the justifiable angst with the protocol [486, 281] given the historical grudges between linked data and indieweb communities (See this retrospective by one of its authors, Christine Lemmer-Webber [275]). So we could imagine that post using a reference to a document or one of its cells in its `in_reply_to` field.

Mastodon might be a good transitional medium, but we can extend it to make use of our linked p2p system. The fediverse decouples the network from a single platform, but instances still bundle together the underlying data of a post with an interface, host, and account (but see hubzilla). p2p helps us decouple accounts from hosts (see this discussion on a p2p ActivityPub [487]), but we would also like to decouple interfaces from the underlying data so that we have a continuous communication medium where different interfaces are just *views* on the data. To do that we would want to start by replacing Mastodon's flat "`content`" field with the kind of typed cells in our documents that indicate what kind of message they are. For example a simple text-based message might use the ActivityStreams `Note` type:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "type": "Note",
  "name": "My Message",
```

```
  "content": "A note I send to you!"
}
```

But we might equivalently send a `@jupyter:Notebook` as a message, or some compound object like a `Collection`:

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "summary": "A Compound Message!",
  "type": "Collection",
  "totalItems": 2,
  "items": [
    {
      "type": "Note",
      "name": "Hey how ya doin here's a notebook"
    }
    {
      "@context": "https://jupyter.com/",
      "type": "Notebook",
      "content": "..."
    },
  ]
}
```

So the *existence* of a particular type of message is not bound to the ability of any given program's ability to render it. Our notebook program might not be able to understand what it means to have people responding to and making threads about its cells, but we would still be able to receive them and open them with an interface that does, and we could further imagine the ability for a type to recommend a program to us for rendering it as we did with the ability for analysis nodes to specify the code to execute them. We will set aside for a moment the issues of moderation and permission for which messages can link to our work and the practicalities of sending, receiving, storing, and serving messages and return to them in the context of annotations and trackers, respectively.

Where do our posts go? For concreteness, we can start with a forum called "NeuroChat." `@neurochat` is a peer like any other, and it supports some of the basic ActivityStreams vocabulary. We can request to join it by sending a `@as:Join` request, which gives it permission to index our public posts and issue links on our behalf through its web interface. It has a few broad categories like "Neuromodulation" and "Sensory Neuroscience," within which are collections of threads full of chronologically-sorted posts. Threads are objects that indicates a category like `@neurochat:categories:Neuromod`, and when we post in them we create links that are `@as:attributedTo` us with the `@as:context` of the thread we're posting in and any `@as:inReplyTo` links to preceding or quoted posts.

We want to announce and describe some recent results in our document `@jonny:my-project:Writeup`. This kind of post is common in `@neurochat`, and so instead of a generic citation we use a `@neurochat:AnnouncesResult` link to indicate the relevant document. In our forum pseudocode we'll use a `#prefix` macro to give a short name to our project and semantic wikilinks with a `[[predicate::object]]` syntax for the purpose of

demonstration, though ideally these would be part of the forum's interface. We think we really have something that challenges some widely held previous results:

```
#prefix project @jonny:my-project
#prefix nc @neurochat

Hi everyone, happy to present my new work
[[nc:AnnouncesResult :: project:Writeup]].

I think it raises a number of interesting questions,
in particular @rival's longstanding argument
[[@cito:disputes :: @rival:TheBrainIsInTheLiver]].

I also wonder what this means about the conversation
we've been more generally about
[[@cito:discusses :: @discipline:whereAreTheOrgans]].

Anyway, write back soon, xoxo.
```

Our rival takes the criticism in stride but wants to run their own analysis. They follow the links back to find our data, and reanalyze it. Their analysis framework has already issued a link indicating that it reanalyzes our data, and rather than do an independent writeup our rival returns to the thread to continue the discussion.

```
Interesting result, you old scoundrel.

That indeed [[disputes :: @doi:<id>]],
in particular its section [[.:results:main]]
and my re-analysis adds another wrinkle to the problem!
Take a look:

[[nc:embed :: @rival:reanalysis]]

This really complicated another project of mine,
[[@rival:projects:NeuronsCanSwim]]
```

Our forum's `embed` link knows how to embed the notebook our rival used to do their reanalysis and in the underlying message indicates the the current version so if they update it in the future the message will still be comprehensible. Our rival doesn't use a predicate for their link to their side-project and our forum uses its default `Mentions` predicate. It's still more informative than a duplet link because the context of being a discussion in our forum the links in the surrounding posts. We could imagine additional capabilities we give to our forum, like the ability to automatically trigger a re-analysis by someone mentioning a different pipeline for a given dataset, but we'll leave those as an exercise to the reader.

This example is a relatively trivial instance of scientific communication: sharing results, relating them to previous findings, and thinking about the broader implications on the field. However in our current regime of scientific communication, even in the most progressive publication venues that allow communication directly on a work, this kind of communication is *entirely invisible* to the broader state of

our understanding. With our system of linked communication, however, the entire provenance chain from our experiment through its analysis and contextualizing discussion is related to immediately related work as well as the standing questions in our field. Our work is enriched by the additional analysis from our rival, and their work is continuously contextualized as the state of our understanding develops. We were capable of making incremental refinements to our shared understanding using units of work that were much smaller than the traditional scientific paper. It would be possible for someone entirely outside our field to browse through the general links from basic research questions to relevant work and its surrounding discussion. If they were to ask questions, our answers would represent the latent diffusion of understanding to other disciplines based on the graph context of our respective work — and we could be credited the time we spent doing so! In short, scientific communication could actually be *cumulative.*

Forums are just one point in a continuous space of threaded media. If we were to take forum threads out of their categories, pour them into our water supply, and drink whatever came our way like a dog drinking out of an algorithmic fire hydrant, we would have Twitter. Remove the algorithm and arrange them strictly chronologically and we have Mastodon. In both, the "category" that organizes threads is the author of the initial post. Algorithmic, rather than purposefully organized threaded systems have their own sort of tachycardic charm. They are effective at what they aim to do, presenting us whatever maximizes the amount of time we spend looking at them in a sort of hallucinatory timeless now of infinite disorganization — at the expense of desirable features of a communication system like a sense of stable, autonomously chosen community, perspective on broader conversation, and cumulative collective memory.

Nevertheless the emergence of a recognizable "Science Twitter" points towards a need for relatively informal all-to-all communication. Serendipitously being able to discover unlikely collaborators or ideas is a beautiful dream, if one ill-served by the for-profit attention economy. Our formulation of the `@neurochat` forum was as an equal peer that mirrored, collected, and organized posts that otherwise are issued from other peers such as ourselves. In the same way that we might use the ActivityStreams `Join` action to have our posts mirrored by it, we might also use `@as:Follow` to receive posts from any peer, and in the case of a federation that might include posts from its members sent to the federation. Notice in the example mastodon post above how it uses JSON-LD and the activitystreams ontology: a "me to the world" tweetlike message is addressed to `activitystreams#Public` and cc'd to the URL that corresponds symbolically to the list of `@jonny`'s followers.

We can take advantage of the graph structure and rich metadata of our social network in ways that are impossible in corporate social media networks that require the expectation of disorder to be able to sell "native" ad placement. The instance-to-instance federation model of the fediverse, and the accompanying absence of any "global" scope of all posts, results in the need for multiple views on the network: in Mastodon, a "local" timeline that shows only posts from within the host instance, and a "federated" timeline that shows posts from all instances that the host instance has federated with. Since our network allows peer-to-peer, federation-to-federation, and peer-to-federation interaction, we can extend that further. We can construct views of the network based on granular control over graph depth: instead of seeing just the posts from the peers that we follow, we can request to see n-depth posts, from the peers that our peers follow, and so on. This could be done at the level of

a "view" or at the level of the follow link itself — since I know this person well, I want to see a graph depth of 2 from them, and a depth of 1 from others. At the federation level, we might imagine that `@neurochat` is federated with another `@linguisticsChat` group and the two mirror and rehost each other's posts. We could then make use of our extended social graph and prioritize posts from people who are part of overlapping subsets of the federations we are a part of. The peer-based nature of our social network serves as the basis for a system of fluid scoping and filtering of the kind of communication we are looking for at any given time. So rather than a disorganized public melee or the empty rooms and new logins from yet another closed Slack, our communication could be part of a coherent scientific conversation.

Across from filtering what we receive, the same could be done to what we send by choosing where our posts are addressed and who can see them. The same multi-modality of "following" used to indicate the graph depth of the posts we see could let us indicate different kinds of relationships. We should be able to send global, undirected messages on a public feed, but we don't necessarily want to talk to our friends in the same way that we talk to strictly professional colleagues. We might want to organize privately with a few colleagues, or prevent trolls or hostile groups from accessing or making use of our work. Effectively, we should be able to direct our messages to different groups of peers to support the multiple registers of our communication.

The need for rapid and informal scientific communication being mediated by corporate social networks has the unfortunate byproduct of needing to carefully manage a "personal brand." To be seen as a "serious," we need to maintain some proximity to the stilted academic voice, forfeiting any approachability to science that might be gained from public communication. If we are to expand the scope of what we consider as the labor of scientific communication, we should also take seriously its many registers and contexts. Informal media like alt accounts, mailing lists, groupchats, zines, and whisper networks are also an integral part of science, particularly for marginalized and vulnerable scientists [488] . Parallel to organizing our communication in empirical professional communication, we might build systems that support our organization into federations to more effectively bargain over our working conditions and protect ourselves. The venues that organize our communication being limited to journals, and the accompanying regulation over the registers of communication that count as "real" science, is even more limiting than its profound effects on scientific literature proper. The absence of infrastructure to support the multiregister communication of science limits our ability to organize over the broader state of our work, form extended communities, and reduces what should be the collective project of making our work broadly understandable to the individualistic projects of "scicomm influencers." It shouldn't take a lot of additional critical analysis to say "shitposts are good, actually, for science."

There's a balance to be struck between a system of granular control over the messages we send and receive with the ease of a monolithic algorithmic feed. Mastodon sorts all posts purely chronologically, which translates into relatively steep limits on the size of communities as feeds become unintelligible washes of posts. Instead of forgoing algorithmic organization altogether, another means by which we could take advantage of the graph structure of our network is by being able to *choose* the sorting algorithms we use. We might want to prioritize posts from someone who we don't necessarily follow but is interacting with people that we do in contexts that we share,

or be able to deprioritize posts that are "close" to us in our social graph in order to discover new things. This too could be a cumulative, community-driven project, where we might want to try out our friend's `@friends:sorting:NewAlgorithm`, tweak it a bit for our preferences, and republish a new version.

Generally, the impact of having a communication system that decouples hosting, identity, interface, and format on an underlying linked data grpah gives us a broad space to build different views and tools to use the underlying data. Specifically, without predicting the infinite future of communication media, our system of linked, cell-like communication generalizes threadlike media like forums and feeds into a continuous system that can blend their features as needed. Durable, cumulative discussion about the state of our understanding should be able to live side-by-side with ephemeral, informal conversations. It should be possible for us to serendipitously discover people and information as well as for a newcomer to have a place to ask questions and build their understanding. It should be possible for us to form and dissolve communities fluidly without substantial technical start-up costs and the total loss of memory when they close. A system that supports the fullness of continuous communication would be an unfathomably richer way of building reliable, accessible, and multivalent understanding of our reality than the current system of a gladitorial thumbs up/down indictment on years of your life that is journal-based peer review.

Overlays & Adversarial Interoperability

We can't expect the entire practice of academic publishing to transition to cell-based text editors in a p2p linked data swarm overnight. In the same way that we discussed frameworks for integrating heterogeneous analytical and experimental tools,

we need some means of **bridging** communication tools and **overlays** for interacting with existing communication formats. There are many examples of bridging communication protocols, eg. the many ways to use Matrix with Slack, email, Signal, etc. The overlays for websites, pdfs, and other more static media that we'll discuss are means to bring them into the system whether they support it or not: our interoperability should be willing to be adversarial if it needs to be [489, 490] . In representing the intrinsically interactive and social nature of reading (eg. see [491]), overlays as interfaces also supplement the "horizontal" connections between cells by injecting information into them or transcluding it elsewhere: creating a fuzzy boundary between writing *on* something vs *about* something.

We don't need to look far to find a well-trod interface for annotation overlays for document-like media: the humble highlighter. Hypothes.is, enabled on this page, lets readers highlight and annotate any webpage with a browser extension or javascript bookmarklet. This interface is a near match to the highlighting and review tools of Microsoft Word and Google Docs used for the same purpose. At its heart is a system for making anchors, references to specific places in a text, and the means of matching them even when the text changes or the reference is ambiguous [492] . For example, this anchor has three features, a `RangeSelector` that anchors it given the position within the paragraph, an absolute `TextPositionSelector`, and a contextual `TextQuoteSelector` that you can see with an API call. Anchors like these, along with references to the code that resolves them, could be the objects to which we could link from the rest of our communication system.

On its own, it serves to give a `Talk:` page to every website. With an integration into a system of linked data and identity, it also serves as a means of extending the notion of bidirectional transclusion described above to work that is not explicitly formatted for it. Most scientific work is represented as `.pdfs` rather than `.html` pages, and hypothes.is already supports annotating PDFs. With an integration into pdf reading software, for example Zotero's PDF reader, there would be a relatively low barrier to integrating collaborative annotation into existing workflows and practices.

Digital publishing makes imagining the social regulation of science as a much more broadly based and continuous process much easier, but the problem of moderation remains (as it has since at least the coiner of the terms "Gold" and "Green" open access lost faith in ahierarchical scientific communication after someone said poo-poo words at him on Internet while defending the use of they/them as gender-ambiguous pronouns [493, 494, 495]). Some movement has been made towards public peer review: eLife has integrated hypothes.is since 2016 [496] , and bioRxiv had decided to integrate it as well in 2017 [497] before getting cold feet about the genuinely hard problem of moderation (among others [498]) and instead adopting the more publisher-friendly TRiP system of refereed peer-reviews [499] .

Overlays raise basic questions about control over the representation of our work, about who is able to write what on it. As with potential incompatibility between interfaces, we should be able to control what comments appear *on* our work, but there is no way to control – even in our current communication systems – what someone says *about* it. Our system gives us some ability to identify bad actors and regulate the avenues of communication without overcorrecting into a system where criticism becomes impossible – even if we don't want to represent someone's comments on our work, it is possible to make them and for others to find them, but it's also possible to contextualize their context if they're made in bad faith.

Though a description of the norms and tools needed to maintain healthy public annotation is impossible here, our system *provides a space for having that conversation.* Authors could, for example, allow the display of annotations from a professional society like `@sfn` that has a code of conduct and moderation team, or annotations associated with comments on `@pubpeer`, or from a looser organization of colleagues and other `@neurofriends`. Conversely, being able to make annotations and comments from different federations gives us a rough proxy to different registers of communication and preserves the plurality of our expression. Social tools like these are in the hypothes.is team's development roadmap, but I intend it as a well-developed and mature example of a general type of technology[57] rather than a recommendation.

[57] cf. the genius.com overlay.

In addition to annotating other works, overlays can come in the form of bots or other tools for interacting with existing systems in a way that's compatible with a new one. One particularly impressive example of aggressive interoperability in this domain is Eduardo "flancian" Ivanec's agora [500, 501] . An agora is a wiki-like project with pages (or nodes) for each named concept, but it also allows for multiple representations of a given node: so notes from multiple people across multiple mediums will be present on the same page. Accompanying the agora is the anagora bot (on Mastodon and Twitter), which makes links to, and backlinks from pages mentioned as `[[wikilinks]]` by accounts that follow them (for example: a post, the bot's reply, and one of the linked pages, `[[wikilinks everywhere]]`). This becomes natural quickly: it's common for people associated with the agora (or flancians) to speak with wikilinks, or index links and conversations that they come across for mutual discovery.

The agora makes linked annotation a basic part of using the web without requiring fundamental changes in communication practices. The agora is an exercise in radically permissive protocol-like thinking: rather than creating a new app or platform, theoretically any bot could be made to crawl different mediums for wikilinks and index them. It illustrates that interfaces can precede formal protocols and serve as a testing and development ground for them.

Another bridging overlay for more author-focused scientific communication would be to explicitly archive the threads that increasingly serve as companions to published work — or original works of scholarship on their own (eg. [502]). I have started experimenting with this with the `@threadodo_bot`, a bot that converts a thread to a PDF[58] and uploads it to Zenodo when it is tagged beneath one. This bot is being programmed as a generalizable framework for bots that can accept parameterized commands. For example, someone can set their authorship information by tweeting "identify" at threadodo, which accepts a series of key-value pairs to set your name, affiliation, and orcid. Future versions will support automatic reference generation for linked works, including previously archived threads, as well as setting prefixes for OWL schema for use in semantic `[[predicate::object]]` wikilinks.

[58] complete with markdown renderin!

When some recognizably different communication medium begins to coalesce, it should support bidirectional *crossposting* to and from existing mediums. Crossposting substantially eases transition — for example between Twitter and Mastodon — as patterns of usage that have been trained for years on hyperoptimized attention-capturing platforms are hard to break. Together with bridges, bots, and overlays for annotation, linking, and archiving, the dream of rewriting the norms of academic communication looks less like some "if you build it they will come" pipe dream and more like a transitional period of demonstrating what we can dream of together. Adversarial interoperability not only *works* [490] , it's also a gift of the hacker mindset that teaches us how to make building a better world an act of unrepentant *joy*.

Trackers, Clients, & Wikis

The final set of social interfaces are those for collective governance of the system. So far we have generalized documents "vertically" into recursive typed cells, "horizontally" into linked cells for communication, and then blurred their independence by and extended them into incompatible media with overlays. The remaining piece we need are multi-authored documents: **wikis**. We'll pick up the threads left hanging from our description of bittorrent trackers and knit them in with those from the wiki way to describe how systems for surfacing procedural and technical knowledge work can also serve as a basis of searching, indexing, and governing the rest of the system. Where the rest of our interfaces were means of creating particular kinds of structured links, we'll also describe wikis as a means of interacting directly with links to negotiate the relationships between the multiplicity of our folksonomic schema.

In the process we'll give some structure to the **clients** and **trackers** that serve and organize them.

Our notion of recursive cell-like documents is already a good basis for wiki pages.

Multi-author documents should already be possible with a permission system that we have invoked previously to limit read access, and so the most radically open, pub-

licly editable wikis would just have edit permissions open to anyone. The **version history** that makes the notion of SoftSecurity possible should also be a general property of links in our system. The other concept we'll borrow from traditional wikis is the model where **pages represent topics.** Practically, let's suppose this means that within documents beneath some namespace like `@jonny:wiki`, we can make wikilinks to [[New Pages]] that imply links to `@jonny:wiki:New_Pages` — though for the sake of simplicity in this section we will assume that our wiki starts at the root of the `@jonny` namespace.

We want to preserve two types of multiplicity: the multiplicity of *representations* (as in `Talk:` pages) and *instances* of a given topic, or the ability for multiple peers to have linked versions that potentially transclude content from other peers, but are ultimately independent. Both can use different components of a namespace: for multiplicity of representation we might follow the example of mediawiki and use parallel namespaces like `@jonny:talk`, and multiplicity of instances follows naturally from parallel peers by linking `@jonny:wiki:My_Page` to `@rumbly:wiki:My_Page`.

Wikis that represent multiple instances of a given page are already a subject of active experimentation. Flancian's Agora is one example, which is based on markdown files in git repositories, and markdown files with the same name in federated repositories are presented on the same page. A much older project[59], everything2 is built around multiple "writeups" for a given "node." Multiple instances of a page are also a defining feature of Ward Cunningham's federated wiki, which has a vertical "strip" based interface where clicking the colored squares at the bottom of a given page will open another strip to show another user's instance of the page. We'll borrow Ward's terminology and refer to this kind of wiki as a federated wiki.

[59] everything2 (or e2) users tend to be, uh, floridly sarcastic, and so its history is not as clearly laid out as the other old wikilike sites.

Federated wikis already have some broader purchase as "personal knowledge graphs," [503] where people use tools like Notion or Obsidian to keep a set of linked, semistructured personal notes. Rather than thinking of a wiki as wikipedia, with pages that aspire to be uniformly named and written, personal knowledge graphs take whatever form is useful to the person maintaining them. This maps neatly onto our namespaces and recursive documents as a means of *organizing our system of links.*

Say we have a very simple project structure that consists of a dataset with two tables and a document with the date of the experiment and some short description of the data. In our pseudocode:

```
<#project>
  a @jonny:Project

  dataset
    @format:csv
      table1
      table2

  document
    a @jupyter:notebook

    @schema:Date dateCollected
    Description
```

```
"This is the data that I collected"
```

This has a natural representation in our wiki as a set of nested cells: the `@jonny:project` page has two child cells, one for the dataset and one for the document, which have their own child cells that represent the tables, date, and description according to their types. Since the relationships between our cells can also typed, ie. have an associated predicate like `before`, `after`, or `inReplyTo`, we'll use two additional types to differentiate nested cells:

- `child` (and its inverse `parent`) cells correspond to a cell's position in our namespace, so we could find our data at `@jonny:project:dataset`.
- `transcludes` (and its inverse `transcluded`) indicates some other cell that we represent on a given wiki page, as we might want to do if we wanted to embed one of our plots in a post.
- And other cells linked with bare `[[wikilinks]]` are untyped.

This gives us a bidirectional representation of our link structure: and with it an interface for browsing and managing all the various types of objects that we have described so far.

Since schemas, or abstract representations of the links a type might have, are themselves made of links, these too can be managed with a wiki. Semantic mediawiki and its page schemas extension implement a system like this. For example, the Autopilot wiki has a form to submit build guides for experimental apparatuses. Build guides have a schema and an associated template that lays out the form input on the created page and makes the semantic wikilinks that declare its properties like `[[Is Version::2]]`.

This system is semantically rich while also being flexible, as everything reduces down to semantic wikilinks on a page, so free text can be used fluidly along with structured schemas, forms, and templates. The wide open structuring space of the wiki handles the messy iteration of technical knowledge work well while also having enough structure to be computer readable. A page for an amplifier makes the datasheet, serial protocol, and the GPIO pins it needs available via an API call while also carrying on a continuous effort to crudely defeat its low-pass output filter. A plugin page can credit the papers it was used in by DOI and the python packages needed to run it while also describing how to void the warranty of your oscilloscope to unlock additional functionality.

The page-centric model of semantic wikis poses a problem, though. The guide for building the Autopilot Behavior Box has semantic annotations describing the CAD schematics, materials, and tools that it uses. This works fine for other assembled parts or schematics like 3d printed parts that have pages of their own, because their pages can contain the additional properties that describes them like the associated `.stl` files. Materials like screws are trickier. Each screw varies along about a dozen dimensions, and so that either requires making a separate page for each individual screw or use workarounds[60] that reduce the maximum depth of representation to two layers and add other nasty complexities.

[60] like subobjects or record types

A recursive cellular system avoids these problems and provides a uniform interface to complex representations. We can create schema for experiments that allow for

a build guide, which can contain assembled component descriptions, which can contain materials, etc. When using that schema to describe a new experiment, the researcher can be prompted for any of the possible available fields in the recursive model while also allowing for free space to write in the semi-structure of the building blocks. Extending an existing schema is just a matter of transcluding it and then modifying it as needed. With the ability for our interface to assign fixed IDs for these objects or generate unique hashes based on their contents, the tension of ephemeral object declaration with unique addresses disappears.

The tension of arbitrarily flexible personal knowledge graphs with multiscale organization with other peers remains, though. Approaching from the other side of discovery, rather than declaration of information leads back to considering the structure of our p2p client and tracker-like systems. The most immediate problem we face is the need to reconcile the differences between multiple instantiations of overlapping representations of concepts that change through time. That sounds a lot like version control system, and a VCS like git or mercurial should be a natural part of our client. Where IPFS is "a single bittorrent swarm, exchanging objects within one Git repository," [313] we make a mild modification and think of a single bittorrent swarm with a git repository per peer (also see IPLD [504]). Git stores files as content-addressed "blobs" of binary indexed by "trees" that represent the file hierarchy [505] . Our client can do something similar, except using the triplet link structure for trees rather than typical duplet links. Another peer querying our data would then resolve our identity to the top of the tree, our client would then either serve the parts of our tree that the peer has access to or else let them traverse some subsection of it, and they could then request any file "blobs" that the tree points to[61].

By itself this would have a lot of overhead as a large number of peers would need to be queried to find a particular subset of matching metadata. We can mediate that in a few ways. First, our clients could take advantage of the embedded social network to cache and rehost other peer's trees — either in their entirety or as shards distributed among other peers — depending on our relationship to them. Second, when making links, we could notify relevant and subscribed peers that we have made it (eg. see [508]). Combined with distributed caching, that would allow the peer responsible for the schema to direct queries to peers already known to have a particular kind of file: eg. the `@nwb` peer could track when `@nwb` datasets are declared.

We don't necessarily *want* to have an entirely autonomous protocol though, following the example of wikis and bittorrent trackers we want social systems for shared governance and maintenace of the system. Trackers first serve the technical need of indexing a particular community's data, eg. as `@dandihub` does with `@nwb`, in case peers go offline. We don't want to just track datasets, however, we want to track the many different kinds of metadata in our swarm. The second role of trackers is collective curation and negotiation over schema.

Say a group of my colleagues and I organize to set up a server as our tracker. As an interface, our tracker might allow us to browse schemas as a tree. For a given node, we might see "horizontally" across all the schemas that have modifications or extensions to that node, and "vertically" up and down their parent and children nodes. We notice that our colleague has made an extension to a schema that looks very similar to ours. We do a `diff` to see which nodes are similar and which are different between our schema. Both of us have some good ideas that the other doesn't have, so we open a conversation thread by creating a node that references both of our schemas

[61] That's sufficient detail for a sketch, but there is of course a great deal of subtlety that would need to be resolved in an implementation. For example, see [506, 507] .

as candidates for merging and send it to our colleague. We negotiate over a way to resolve their differences, similar to a pull request, and then `merge` them. Part of our merging process is indicating how to change either of our existing structures to become the third merged structure, so our clients are able to handle those changes for us and the update propagates through the network.

As our tracker grows and maybe even becomes the de-facto tracker for our subdiscipline, things start becoming a bit messier. Aside from the "tree" view for browsing metadata, we've built views that help it function as a forum for threaded conversations and a wiki for organization, tracking projects, and setting policies. The durable but plastic nature of wikis is exceptionally well suited for this. From Butler, Joyce, and Pike (emphasis mine):

> Providing tools and infrastructure mechanisms that support the development and management of policies is an important part of creating social computing systems that work. [...]
>
> When organizations invest in [collaborative] technologies, [...] their first step is often to put in place a collection of policies and guidelines regarding their use.
>
> **However, less attention is given to the policies and guidelines created by the groups that use these systems which are often left to "emerge" spontaneously.** The examples and concepts described in this paper highlight the complexity of rule formation and suggest that support should be provided to help collaborating groups create and maintain effective rulespaces.
>
> [...] **The true power of wikis lies in the fact that they are a platform that provides affordances which allow for a wide variety of rich, multi-faceted organizational structures.** Rather than assuming that rules, policies, and guidelines are operating in only one fashion, wikis allow for, and in fact facilitate, the creation of policies and procedures that serve a wide variety of functions
>
> *Don't Look Now, But We've Created a Bureaucracy: The Nature and Roles of Policies and Rules in Wikipedia* (2008) [509]

So we might have a set of policies that encourages a reporting system to notify other peers if their data is misformatted. Or we might reward contribution with a "peer of the week" award that highlights their work like What.cd's album of the week or Wikipedia's barnstars [510] . We might adopt a cooperative model where each peer pays their share of the server fees, or has to take shifts on moderation and cleanup duty for the week. Each tracker can adopt different policies to reflect their communities.

Trackers-as-wikis don't have to exist in isolation. Trackers for adjacent disciplines or purposes should be able to federate together to transclude pages: organizing multiple perspectives on the same topic, or supplementing each other into a broader base of knowledge.

What if consensus fails? Our system attempts to mitigate the potential damage of tyrannical moderators by making it extremely easy to *fork*. Since every link in the

system "belong" to someone underneath a `@namespace`, links and the schemas they build are always a proposition: "something someone said that I don't necessarily have to agree with." If another peer doesn't like the `merge` that we did, they can fork the previous version and continue using it — for other peers the link to the merged version lets them translate between them. If we want to jump ship and go find a different tracker that better reflects our values, all our data, including relationships to the people that we liked there, guides we wrote on the wiki, etc. are still our own. The tracker just tracks, it isn't a platform.

Our joint tracker-wikis have many applications for scientific communication, and it's worth exploring a few.

11.4.3 Applications

Continuing the example of the Autopilot wiki, we could make an array of **technical knowledge wikis.** Wikis organized around individual projects could federate together to share information, and broader wikis could organize the state of our art which currently exists hollowed out in supplemental methods sections. The endless stream of posts asking around for whoever knows how to do some technique that should be basic knowledge for a given discipline illustrate the need. Across disciplines, we are drenched in widely-used instrumentation and techniques without coherent means of discussing how we use them. Organizing the technical knowledge that is mostly hard-won by early career researchers without robust training mechanisms would dramatically change their experience in science, whittling away at inequities in access to expertise. Their use only multiplies with tools that are capable of using the semantically organized information to design interface or simplify their operation as described in experimental frameworks.

Technical wikis could change the character of technical work. By giving a venue for technical workers to describe their work, they would be welcomed into and broaden the base of credit currently reserved only for paper authors. Even without active contribution, they would be a way of describing the unseen iceberg of labor that science rests on. Institutional affiliations are currently just badges of prestige, but they could also represent the dependence of scientific output on the workers of that institution. If I do animal research at a university, and someone has linked to the people responsible for maintaining the animal facility, then they should be linked to all of my work. Making technical knowledge broadly available might also be a means of inverting the patronizing approach to "crowdsourcing" "citizen science" by putting it directly in the hands of nonscientists, rather than at the whim of some gamified platform (see [511]).

Technical wikis blend smoothly into **methods wikis** for cataloguing best practices in experimental design and analysis. It is a damning indictment of our systems of training or review (or, more likely, both) that it is possible to publish a paper based on badly misused t-tests, yet the scientific literature is flooded with analytical and interpretive errors [512, 513, 514] . Analytical errors are not just a matter of lack of education, but also a complex network of incentives and disciplinary subcultures. Having the ability to discuss and contextualize different analytical methods elevates all the exasperated methods critiques and exhortations to "not use this technique that renders meaningless results" into something *structurally expressed in the prac-*

tice of science. See the `@methodswiki` page that summarizes this general category of techniques and the discussion surrounding their application in the relevant body of research. For implementation of analytical libraries, to move beyond fragile code reduplicated in every lab we need some means of reaching fluid consensus on a set of quasi-canonical implementations of fundamental analysis operations. Given a system where analysis chains are linked to the data they are used with, that consensus might come by negotiating over a semantically dense map of the analysis paths used in a research domain.

Analysis wikis would also be a natural means of organizing the previously mentioned Folding@Home-style distributed computing grids. Groups of researchers could organize computational resources and govern and document their use. For example, a tracker could implement a "compute ratio" where donated computing resources function as credit for "bounties." Analogously to private torrent trackers, where a bounty system might allow peers to trade their excess upload in exchange for someone uploading a rare album, linked tracker/wikis could translate that model to one where someone who has donated a lot of excess compute time could trade it for someone uploading or collecting a particular dataset. Since the kind of wikis we are describing combine free text with computer-readable data structures, policies for use could be directly implemented in the wiki in the same place they were discussed. This too is a means of collectivizing support for open-source initiatives that support basic infrastructure by donation and the mercy of cloud providers by integrating them in the basic social practices of science [479] .

Review wikis could replace journals almost as an afterthought. Though an adequate infrastructure of scientific communication immediately antiquates traditional peer review, review wikis could facilitate it without recourse to an extractive information industry. In response to the almost unique profitability of publishing, some researchers have reacted, perhaps justifiably, by demanding payment for their reviews (eg [515]). An alternative might be to organize review *ourselves.* Like the ratio requirements of private bittorrent trackers, we might establish a review ratio system, where for every review your work receives you need to review n other works.

This would effectively function as a **reviewer co-op** that can make the implicit labor of reviewing explicit, and tie the reviews required for frequent publication with explicit norms around reciprocal reviewing.

Library wikis focused on curation, contextualization, and organization of information could be one modality of resisting the neoliberal drive to reduce librarians to stewards of subscriptions and surveillance data [228, 516] . Knowledge organization is hard practical and theoretical work, and reimagining the space of scientific communication as one that we actively *create* instead of one that we merely *suffer through* is a wide-open invitation for the comradeship and leadership of librarians. Linked data has been a mixed blessing for librarians, its promise obscured by intellectual property oligopolies and the complexity of linked data standards (see [385]). Given fresh tooling and a path away from structuring influence of for-profit publishers, the rest of us should be prepared to learn from those that have already been doing the work of curating our archives:

> [M]ake it easy to rely on linked data, easier than it is to rely on MARC, and the library world will shift, from the smallest and poorest libraries upward... and David will at last stone Goliath to death with his linked-data slingshot.

| *Stoning Goliath* (2022) The Library Loon [385]

Finally, **theory wikis** could "close the theoretical-experimental loop" to turn the buckshot of results into cumulative understanding of complex phenomena. In many (or maybe just the non-realist) scientific epistemologies, results do not directly reflect some truth about reality, but instead are embedded in a system of meaning through a process of active interpretation (eg. [517, 518]). The model of grounding new research in existing understanding given by contemporary regimes of scientific communication is for each paper to synthesize and re-interpret the entire body of relevant prior research (formally, the "introduction"), which is bluntly impossible. We do the best we can alongside strong countervailing incentives to selectively engage with work in order to tell a publishable story in which we are the hero. Since the space of argumentation is built from scratch each time, cumulative progress on a shared set of theories is more of a myth for undergraduate introductions to the scientific method than a reality. Most fall far from the supposed ideal of hard refutation and can have long lives as "zombie theories." van Rooij and Baggio describe the "collecting seashells" approach of gathering many results and leaving the theory for later with an analogy:

> "In a sense, trying to build theories on collections of effects is much like trying to write novels by collecting sentences from randomly generated letter strings. Indeed, each novel ultimately consists of strings of letters, and theories should ultimately be compatible with effects. Still, the majority of the (infinitely possible) effects are irrelevant for the aims of theory building, just as the majority of (infinitely possible) sentences are irrelevant for writing a novel." [519]

They and others (eg. [520]) have argued for an iterative process of experiments informed by theory and modeling that confirm or constrain future models. Their articulation of the need for multiple registers of formality and rigidity is particularly resonant here. van Rooij and Baggio again, emphasis mine:

> **We should interpret any data in the context of our larger "web of beliefs,"** which may contain anything we know or believe about the world, including scientific or commonsense knowledge. One does not posit a function f in a vacuum. [...] One can either cast the net wide to capture intuitive phenomena and refine and formalize the idea in a well-defined f or, alternatively, make a first guess and then adjust it gradually on the basis of the constraints that one later imposes: The first sketch of an f need not be the final one; what matters is how the initial f is constrained and refined and how the rectification process can actually drive the theory forward. **Theory building is a creative process involving a dialectic of divergent and convergent thinking, informal and formal thinking.** [519]

Durable but plastic, referential and dialogic, structured and free mediums like our wiki-trackers could be a practical means of integrating theory in a loop with experimentation and interpretation. Many theories are formalizable, and our linked data system is a relatively arbitrary means of expressing complex constraints and inference logics. Others are not, and our mixed-format media also supports the dialectic of informal and formal, mathematized and non-mathemetized theories.

In the most optimistic case, where we have a full provenance chain from interpretation of analytical results back through the viscera of their acquisition, we have a living means of formally evaluating the empirical contingencies that serve as the evidence for scientific theories. For a given theory, what kinds of evidence exist? As the state of the art in analytical tooling changes, how are the interpretations of prior results changed by different analyses? How do different experimental methodologies influence the form of our theories?

The points of conflicting evidence and unevaluated predictions of theory are then a means of distributed coordination of future experiments: guided by a distributed body of evidence and interpretation, rather than the number of papers individual researchers are able to hold in mind, what are the most informative experiments to do? This would be a fundamentally different way of approaching a new "unit" of scientific work that dissolves the scientific paper as such. Many calls for smaller units of scientific work amount to faster turnaround for shorter papers that preserve the unitary binding of an experiment, results, and interpretation. Instead new experiments could start *in medias res,* filling in some cracks in an ongoing experimental/interpretational network. A new node could be contributed already contextualized by the "introduction" of its position in a broader graph of understanding, its interpretation posed against a broader background of prior thought than the immediate data at hand. Given the means of directly applying accumulated technical knowledge, it would be possible for more than just the most resourced labs to be responsive to the nicks and burrs in the cutting edge.

The pessimistic case where we only have scientific papers in their current form to evaluate is not that much worse — it requires the normal reading and evaluation of experimental results of a review paper, but the process of annotating the paper to describe its experimental and analytical methods as a shared body of links makes that work cumulative. Even more pessimistic, where for some reason we aren't able to formulate theories even as rough schematics but just link experimental results to rough topic domains is still vastly better than the current state of proprietary disorganization in service of a surveillance-backed analytics industry.

A meta-organization of experimental results would change the way researchers and non-researchers alike interact with academic literature. It currently takes many years of implicit knowledge to understand any scientific subfield: finding canonical papers, knowing which researchers to follow, which keywords to search in table of contents alerts. Being able to locate a question in a continuous space of discussion, data, results, and theories — to say nothing of building a world without paywalls — would profoundly lower barriers to access to primary scientific knowledge for *everyone.* We might avoid the efforts to weaponize this gap into an ostensibly "helpful" algorithmic search platform that re-entrenches the very industries that make such a platform necessary by constraining the modes of our communication. We might instead arrive at a fluid, boisterous, collective project of explicitly organizing understanding. One sounds like science, the other sounds like industry capture.

11.4.4 Credit Assignment

> I also think one of the big obstacles to freeing up scientific information remains the way in which we continue to pay allegiance to the idea that the most important work is published in so-called 'high-impact' journals [...]. These journals con-

> tinue to thrive, despite a kind of anti-social policy, because **so many academic scientists evaluate each other's work and measure abilities and accomplishments based on where people have published.**
>
> **The only way by which we'll eventually get out of the current situation is by changing the formula dramatically.** That means that we'll probably have to move to a world where the authors have full control – their work will be presented online together with expert reviews and perhaps accompanied by a new evaluation system in which members of the scientific community will provide qualitative and perhaps quantitative measures of the value of the paper. The current world of high- and low-impact journals will eventually dissolve, it's just taking a lot longer than I thought.
>
> Harold Varmus, former director of the NIH (2019) *Of Oncogenes and Open Science* [521]

> The reason we are (once again) having a fight about whether the producers of publicly available/published data should be authors on any work using said data is that we have a completely dysfunctional system for crediting the generation of useful data. [522] The same is true for people who generate useful reagents, resources and software. [523] And like everything, the real answer lies on how we assess candidates for jobs, grants, etc... **So long as people treat authorship as the most/only valuable currency, this debate will fester. But it's in our power to change it.** [524]
>
> Michael Eisen, EIC eLife (2021)

The critical anchor for changes to the scientific infrastructure is the system of professional incentives that structure it. As long as the only way we operationalize scientific value is paper authorship and the prestige of the journals they are placed in, the system stays: Blog posts, software, analysis pipelines, wikis, forums, reviews, are nice, but they don't count as *science.*

Imagining different systems of credit assignment is easy: just make a new DOI-like identifier for my datasets that I can put on my CV. Integrating systems of credit assignment into commonly-held beliefs about what is valuable is harder. One way to frame solutions to the credit assignment problem is as a collective action problem: everyone/funding agencies/hiring committees just need to *decide* that publishing data, reviewing, criticism et al. is valuable without any serious changes to broader scientific infrastructure. As is hopefully obvious, the approach favored here is to *displace* the system of credit assignment by aligning the interests of the broad array of researchers, technicians, and students that it directly impacts to build an alternative that makes it *irrelevant.*

The sheer quantity of work that is currently uncredited in science is a structural advantage to any more expansive system of credit assignment. The strategic question is how to design a system that aligns the interests of enough people excluded by the current system. Belief, as always, is a tricky circular process: how would the

people being evaluated come to believe in its value enough to contribute to it, and how would the people doing the evaluation believe in its value enough to ignore the analytics products by deeply embedded industries?

Everything that exists in this system is attributable to one or many equal peers. Rather than attempting to be an abstract body of knowledge, clean and tidy, that conceals its social underpinnings, we embrace its messy and pluralistic personality. We have *not* been focused on some techno-utopian dream of automatically computing over a system of universally linked data, but on representing and negotiating over a globally discontinuous body of work and ideas linked to people and groups. We have *not* been imagining new platforms and services to suit a limited set of needs, but on a set of tools and frameworks to let people work together to cumulatively build what they need. What is different about this set of ideas is that it is not a new metric, journal, or platform intended to be the new standard that replaces some small element of the system, leaving the rest unchanged. We are taking a broad view on the infrastructural deficits that define scientific work, learning from the broad histories of attempts to remedy them, and trying to chart a course to building systems that fill basic needs. The hope is to seed a critical mass of solidarity by organizing the work to fill the unmet needs that structure the current system of evaluation, in the process building a real alternative that makes the existing system look as ridiculous as it is.

Credit is woven through the heart of this system: the basic operations of interacting with someone else's work are tied to crediting it. While credit is currently meted out by proprietary scientometric tools like altmetric or Plum; downloading a dataset, using an analysis tool, and so on should be directly attributable to one or several digital identities that you control in the manner that you want.

The first-order effects for the usual suspects in need of credit are straightforward: counting the number of analyses and papers our datasets are cited in, seeing the type of experiments our software was used to perform. Control over the means of credit assignment also opens the possibility of surfacing the work that happens invisibly but is nonetheless essential for the normal operation of research. Why shouldn't the animal care technician receive credit for caring for the animals that were involved with a study, its results, and its impact on science more broadly?

A name prominently displayed on a wiki page and a permalink for a CV is ok, but clearly not enough. Foundational work like technical, communicative, and organizational work is useful in itself, but its impact is mostly felt *downstream* in the work it enables. Beyond first-order credit, a linked credit assignment system lets us evaluate *higher-order* effects of work that *more closely resemble* its impact. Say we find someone else's 3D Model, modify it for our use, and then use it to collect a dataset and publish a paper. Someone else sees it and links a colleague to it, and they too use it in their work. Over time someone else updates the design and puts it in some derivative component. Most of the linking is automatic, built into the interfaces of the relevant tools, and soon the network of links is dense and deep.

The incentive to "freeload" by making the use of the system without credit is changed by breaking apart the notion of unitary credit where one or a few people are responsible for "all" of a work. Our current obsession with utter novelty and closed credit removes incentives to extend someone else's work: why would I help patch their code? I won't be added as an author on their paper. For us, instead of just getting professional credit for our paper, we also get credit for extending someone else's

work, for documenting it, and for the potentially large number of nth-order derivative uses. Our credit extends multimodally, including papers that cite papers that use our tool, and the "amount" of credit can be contextualized because the type of link between them is explicit – as opposed to the non-semantic links of citation. Our colleague that recommended our part gets credit as well, as they should since helpful communication is presumably something we want to reward. Rather than the scarcity mindset of authorship, a link-based system can push us towards abundance: "good" work is work that engages with and extends a broad array of techniques, technologies, and expertise.

From the perspective of the worker, their extended contribution graph will always be a superset of the things they would otherwise be credited for. The goal should make it be something we *prefer* to share because it's more reflective of our work. Unlike proprietary metrics that will be increasingly based on surveillance data, our system gives us control over which information we want to be part of our evaluative profile, and it's something that we own to do what we will with rather than the product of some platform.

It's easy to imagine extended credit scenarios for a broad array of workers: A grad student rotating in a lab might not get enough data to make a paper, but they might make some tangible improvement to lab infrastructure, which they can document and receive credit for. Open source software developers might get some credit from a code paper, but will be systematically undervalued from failure to cite it and undercounted in derivative packages. The many groups of workers whose work is formally excluded from scientific valuation are those with the most to gain by reimagining credit systems, and an infrastructural plan that actively involves them and elevates their work has a much broader base of labor, expertise, and potential for buy-in.

From the perspective of the evaluator, our contribution graph provides a much richer space of evaluation while also eroding the notion of a scalar-valued ranking. Some of my more communitarian colleagues might share my distaste for metricizing knowledge work — but hiring committees and granting agencies are going to use *some* metric, the question is whether it's a good reflection of our work and who controls it. Our problems with the h-index (eg. [525, 526]) are problems with paper citations being a bad basis for evaluating scientific "value", and their primacy is in turn a consequence of the monopoly over scientific communication and organization by publishers and aggregators. Their successors, black box algorithmic tools like SciVal with valuation criteria that are bad for science (but good for administrators) like 'trendiness' are here whether we like it or not. A transparent graph of scientific credit at least gives the *possibility* for reimagining the more fundamental questions of scientific valuation: assigning credit for communication, maintenance, mentorship, and so on. So some misguided reductions of the complexity of scientific labor to a single number are inevitable, but at least we'll be able to *see what they're based on* and *propose alternatives.* The presence of many simultaneous metrics on the same underlying graph would be itself a demonstration of the inability of any single metric to capture the value of our work. Conversely, spamming the graph to increase your "high score" with a large number of trivial contributions would be straightforward to detect because of the likely shallowness of the graph, so micro-commodification of labor is less likely. The incentives are aligned to do work that is useful to others and positively affect the state of our understanding.

It's true that some of these extended metrics are already possible to compute. One could crawl package dependencies for code, or download the 100GB Crossref database

[527] and manually crunch our statistics, but being *able* to compute some means of credit is very different than making it a *normal part* of doing and evaluating research. The multimodality of credit assignment that's possible with a linked data system is part of its power: our work *actually does* have impacts across modalities, and we should be able to represent that as part of our contribution to science.

Reaching a critical mass of linked tools and peers is not altogether necessary for them to be useful, but critical mass may trigger a positive feedback loop for the development of the system itself. Even in isolation, a semantic wiki is a better means of assigning credit than a handful of google docs, experimental tools that automatically annotate data are better than a pile of `.csv` files, etc. Bridging two tools to share credit is better than one tool in isolation, and more people using them are better than fewer for any given user of the system. Lessons learned from STS, Computer-Supported Cooperative Work (CSCW), pirates, wikis, forums, et al. make it clear that *the labor of maintaining and building the system can't be invisible.*

12

Conclusion

To take stock:

To approach the deficits in the basic digital infrastructure of science, we divided them into three domains: systems for sharing **data, tools, and knowledge.** These map onto three rough patterns of infrastructure that define the current cloud orthodoxy era of the internet: **storage, computation, and communication.**

We traced the historical development of prior digital infrastructure projects to learn from their successes and failures, conditioned as they are by the contingency and combinatorics of the technologies that existed at the time. We started close at hand within science, and ranged more broadly into lessons from internet protocols, pirates, the semantic web and linked data, early wikis, and the fediverse/indieweb.

Our goal throughout was to sketch a **realistic plan** by which existing technologies could make an emergent interoperable system that was *expansive and evolving* beyond the isolated use of its quasi-independent parts. Our sketch was intended to be *specific* enough to be an actionable blueprint for dispersed groups to work in parallel, but *general* enough to allow refinement through inevitable complexity. We attempted to balance several constraints, primarily **technical capability** and **social compatibility,** but also simplicity and expressiveness, structure and permissiveness; systems that are personal and scalable, respect privacy and empower mutual organization. We are neither politically nor economically neutral, and see the infrastractural deficits of science as reflective of information's broader role as the currently dominant mode of capital accumulation. Accordingly we are searching for system design that can dismantle regimes of surveillance, extraction, and the commodification of information to **re-decentralize** our digital technologies for *people* not *profit.*

The system we arrived at is based on **p2p folksonomic linked data.** Using existing data formats as an initial onramp, and overlays to bridge to incompatible media, our p2p system blends ideas from bittorrent, IPFS, and the Linked Data Platform with metadata beneath a peer's **namespace** indicating content-addressed binary data. Our metadata uses **triplet links** as a means of specifying multimodal schema for data, tools, and social systems. We integrate our data in a complete provenance chain from collection to use with **metadata indicating code** in analytical frameworks and **code indicating metadata** in experimental frameworks. The **social reality** of infrastructure is designed into the core of our system, with peers forming overlapping **federations** with tracker-like overlays. A generalization

of documents as systems of **recursive typed cells** serve as an interface to, and representation of the underlying data and metadata. From them we construct a fluid and continuous system of **documents**, **feedlike media**, and **wikis** for communication and governance of the system. With this system, we satisfy the design goal of a decentralized, protocol-driven infrastructure of linked data, tools, and knowledge.

So how do we build it?

12.1 Tactics & Strategy

> Don't scab for the bosses / don't listen to their lies / us poor folks haven't got a chance / unless we organize
>
> Which side are you on?
>
> Florence Reece (1931) *Which Side Are You On?*

> **Oh but they will mock us and they will mistreat us til they can replace us all with an app or a kiosk,** [...]
>
> All of the energy that I end up expending, I will get back in spades when the systems that necessitate all of this work fall apart... **And we can work for ourselves for a change!**
>
> **So we gotta work!** Cuz none of our visions of a better tomorrow will come to fruition without **a whole lot of work!**
>
> RENT STRIKE (2021) Work! (Future Perfect) [528]

The primary ingredient needed to build decentralized infrastructure is **will.** The incentive and professional systems of science are designed to make us build our own cage: play along, or lose your job. We need to recognize that *the contemporary practice of science is unsustainable* without radical infrastructural, social, and economic reorganization. As the logic of the digital enclosure movement transforms old enemies into new ones, publishers into surveillance conglomerates, the comfortable familiarity of science as we know it will evaporate into the cloud as we cede control over the direction of our work to for-profit companies with their gamified metrics and platforms that commodify every part of it. The worst parts of scientific work are neither natural nor inevitable, but reflect the overwhelming structuring power of orbiting conglomerates. We are *part of this world,* and the world is drowning in an algorithmic sea owned and operated by a rapidly consolidating cluster of information giants. We need to see our place in a shared struggle, the relationship between our deinfrastructuring and the operation of science — and have the courage to do the work to counteract it.

The work doesn't need to be as dreary as its motivation: rebuilding our infrastructure will be ***joyful.*** We have been starved for social and labor organization, for com-

radeship and compassion, isolated as we are on our workplace and disciplinary islands, crushed under the weight of cutthroat publish-or-perish schemes, secretive and distrustful from our culture of the heroic individual rushing through the gauntlet of credit before our enemies do. What we might lose in prestige we will regain in collaboration with new and unexpected colleagues building tools to make our work *more fun.* We can trade artificial scarcity for abundance.

12.1.1 Starting Points

Much of the tactical and strategic vision for our new infrastructure is embedded in its design. We have taken pains to articulate its components as elaborations of existing and widely-used systems, keeping them separable so each can be independently useful before a fuller system is realized, exemplifying them with the real problems that they can remedy. Still, some more scaffolding for how to get there from here is useful.

The core of our strategy should be to organize alongside each other in a series of independent groups working in parallel. We don't need a new leadership council to become a single point of failure. We should try and organize the many existing groups working in different related areas to pull in the same direction towards interoperability. We should avoid the pitfalls of designing our infrastructures "in theory," building crystal palaces removed from the reality of their use. We should seek to embed in existing projects, using their existing mass to lessen the need to prospect for abstract "early adopters."

We should look outside our usual circles for collaborators, and there we might find unexpected energy and expertise. Though the miserable academic fleeing to the greener pasture of "industry" is now a well-trod trope, there is plenty of disaffection on the other side. We shouldn't underestimate the number of extremely talented engineers that would do *anything* to not have to build tools so that Facebook can mine your thoughts to target ads [529] or maximize the time people spend watching YouTube by recommending them increasingly toxic videos [530] . Academic science is relatively unique in that it can marshall funding and labor for projects not bound to profit. Resources and applications are two potent missing ingredients in developing technologies that are intended to be anti-profitable, and we should work to provide them. We should trawl the places where the decentralized messaging, former semantic web, indieweb, and small tech people are already working on building better infrastructure and invite them to work with us.

The three broad domains of our infrastructure could, but don't necessarily need to, correspond to a division of development labor. The serial order of this piece is primarily a byproduct of the constraint of the medium, and there is no reason we can't proceed in parallel. I want to avoid being too prescriptive here in order to invite input from the many people that might potentially be involved — the purpose of this document as a pre-development plan is to provide direction and a high-level design so that the details can be sorted out as we work. For the sake of illustration, though I'll drop down from the level of strategy to tactics to flesh out some of the more proximal possibilities for development, but the remainder of this section should be considered non-normative.

A promising context to develop a p2p linked data client is existing collaborations or tools that have a base of users that handle overlapping but variable data. For example,

the users or developers of a tool like OpenEphys [206] or Miniscope [208] that has data acquisition software that ouputs semi-structured data might be interested in making it possible for everyone who uses the tool to share data with one another from the time of acquisition. The situation is similar for other types of tools like analysis tools, or for collaborations where people are sharing data frequently. Since the output data is relatively simple (eg. videos and timestamps) with some variation (eg. configuration and notes), it would be a smaller climb to prototype generating a metadata model linked to the data. Since the group of people that would be sharing data might initially be relatively small with room to grow, the several components of the p2p client could be worked out separately: eg. manually index repositories of metadata from some frontend while figuring out how to strap a git server to a p2p client like hypercore, etc. Being able to be plugged into a group of people sharing data by using a tool might be a reasonably attractive idea to get people to adopt the tool, so it would be worthwhile to the developers while being a useful feature for the users. Being able to do very tight loops of development and field testing might make the tool more robust than if it were developed strictly in-house, and would be a good small-scale demonstration of the utility of p2p.

At the same time, work could happen in the other direction from data standards towards p2p. Datalad [404] would be an excellent candidate to add linked data and p2p support to, as it already supports JSON-LD with a metadata extension and has a generalizable data storage backend. In neuroscience, DANDI hosts data formatted in NWB, and interoperability with IPFS is on its development timeline. Working from multiple directions towards aligned projects could encourage a small set of modular tools that can accomodate the variation in each of the approaches, and the process would be useful for navigating the fine-scale constraints to the system without putting all of the development eggs in one basket, so to speak.

Aside from p2p, a toolset that's desperately needed across disciplines is an generalizable, approachable means of modeling and ingesting data. The work of building an interface that lets researchers create a JSON-LD metadata model and a declarative description of where that metadata can be located in whatever lab-idiosyncratic format already exists would supplement all other parts of the system. There is no reason for each format to develop a separate schema language and storage mechanism, and this might be one way to spark collaboration between format maintainers.

One of the major reasons for bootstrapping the system with existing formats is to be able encourage analytical and experimental tool interoperability before the means of creating and negotiating over arbitrary schema are developed. This is already starting to happen to a degree in neuroscience with datajoint elements [413] and NWB (see pynapple), but since the conversion tooling for NWB at the moment is still relatively opaque there isn't strong incentive for analysis libraries to support it for seamless input. A wrapper framework to be able to specify an analysis pipeline from metadata that combines a few of these tools might be useful to kick off the positive feedback loop of analysis toolbuilders building towards interoperability, incentivizing format conversion, etc.

The other major starting point for development I see is generalizing cellular documents with JSON-LD and mixing them with ActivityPub. With some relatively minor extensions to the jupyter document format we could add the ability to create new cell types with elaborated linked metadata. From there, we could build an ActivityPub client that allows researchers to post their notebooks and invite comment on them in a document/threaded communication medium. The support of

an existing organization would be useful here too: they could apply to be a crossref member and make use of the very general specification [278] such that each post can be given a hierarchical DOI like `doi:10.<registrar>/user/post/version`. Along with the ability to automatically submit to legacy journals with the conversations attached as supplemental material, this might attract a reasonable critical mass towards a model that would make the move towards a p2p document/communication a much smaller step. Neuromatch [531, 532, 533] has expressed interest in work in this area, though at the time of writing their plans are still in development.

I'll leave the remainder of the organization project to the work of the future.

12.1.2 To Whom It May Concern...

This project should benefit everyone, but we all have different roles to play. Without enumerating every possible category, a few love letters:

PIs: Infrastructure is everyone's responsibility! Diverting time towards organizing the development of basic infrastructure seems expensive and risky, but the truly expensive thing is to do nothing. The quantity of time spent rebuilding everything from scratch, debugging local code, contending with the journal system, resurrecting old data, etc. for all but the most efficient labs is truly staggering. The absence of collective organization makes PIs a sitting duck for profiteering: seeing the difficult resistance posed by library consortia, the open access model shifted towards payments from individual PIs because they have little choice but to pay them on their own. It is in your best interest to commit time to organize with others in your discipline to build generalizable tools that you can share with other labs. We will need you to help shake down funding to pay for development — it will be worth it.

It is also in your best interest to start closing ranks and collectively disavowing the for-profit journals. The rationalization that you need prestige publications for the sake of your trainees is plainly self-fulfilling: what it actually accomplishes is guaranteeing they have to endure the same grim circumstances you do. The best way to support your trainees is to fight to fix our broken infrastructure! Individually you may have little power, but if you organized your colleagues, starting in your department and working out, to agree to never publish for-profit, the problem starts looking very different. In tenure and hiring decisions, having no Nature papers looks very different when you have been loudly organizing with your colleagues for the health of science. Except for those at the extreme heights of the prestige economy, you have perhaps the most to gain by getting off the treadmill.

Early Career Researchers: We don't have much, but we can have each other! We don't have the leverage to make huge changes quickly, but but since we're the ones doing most of the work of building the tools for our research anyway, we should also start organizing with our colleagues to share techniques and methods. As we build infrastructure, coalescing into institutional collaboratives makes it that much easier to organize across institutions. We shouldn't fall into the trap laid for us working to the bone for a prestige publication — if we want to make academic science something that we would actually want to work in, we can help shake the researchers trained in prior generations out of complacency. A better world is out there!

Scientific Open Source Developers: We've got work to do! First, we need to start

making alliances with people we're not necessarily used to, but that's the fun part! For those of us working outside the few major projects, the best thing we can do is to start organizing our tools as broader infrastructure instead of one-off, single-purpose tools. We should focus on designing our tools in such a way that they can be integratable: as a small sample, that means spending time on good packaging rather than throwing everything in a docker container, making APIs that are clear in what they expect and return, and well-contained configuration and parameterization. If our tools aren't already part of a broader framework, we should work on that first! We should avoid cloud dependency when possible: if it is necessary, make sure that it can also be deployed locally just as easily. We should also emphasize multi-scale interfaces: instead of just exposing a set of top-level functions, it should also be possible for someone else to understand its internal operations, otherwise interoperability becomes a distant dream. At the risk of being preachy as a younger developer, the most important thing we can do is organize and be organizable.

Funding Agencies: You are being swindled! Partnership with the cloud industry is a recipe for burning ever-larger portions of your funding allocations on systems that only become harder to walk away from with time. Open source is your friend. Rather than funding projects piecemeal, or funding massive top-down projects with little user engagement, it needs to be possible to receive funding for projects that fill fundamental cross-disciplinary infrastructural gaps. We need to figure out some way of breaking the catch-22 of scientific software development where only projects with demonstrated uptake can receive funding, but it is difficult to start projects intended to address large problems without funding. Scientific funders already do fund a large amount of open source development, and I am not proposing an alternative model here, except to say that the energy and expertise is there to build open-source infrastructure that avoids creating another triple-dip industry.

University Administrators: You're also being swindled! The disorganized smattering of SaaS that serves as the infrastructure of many universities [534] is a short-run bandaid that makes operations more fragile in the long-term! Putting your resources behind organizing institutional and regional infrastructure is a better PR story and far more attractive when recruiting than how large of an AWS subscription you have [310] . University libraries shoulder a huge burden of the cost of the for-profit publishing system, and so you should have every incentive to cut ties — instead of open access mandates, we need you lobbing on behalf of all of us to end the for-profit system.

12.2 Limitations

To get a few big and obvious limitations out of the way first: - Everyone could ignore this piece entirely and it is dead on arrival. - This project would be a direct threat to some of the most powerful entities in science, and they will likely actively work against it. - Despite my best efforts, I could be completely misinformed and missing something fundamental about the underlying technologies. - The social tensions between the relevant development communities could be too great to overcome.

Beyond those, there are several open questions that deserve further consideration, particularly those things concerning cryptography as it is squarely outside my domain of expertise:

Identity: Identity is extremely challenging for any decentralized system. An identity needs to be unique, difficult to counterfeit, easy to verify, easy to manage or recover, and also recognizable if not memorable — and several of these requirements are clearly in conflict. A satisfying resolution of identity will require guidance from cryptographers, but the design of our system has some features that make identity a less-than-intractable problem. The actual raw identifier itself will likely need to be a cryptographic hash of a public key (as in IPFS) for uniqueness and verifiability, but they are very far from memorable. One approach might be to have each peer provide a signed identification object that can be publicly queried with a shorter handle or username, which can then be stored by the peers that follow or befriend them. When peer A referrs to peer B's namespace, then, it would be in reference to peer A's follow/friends list. Another approach is to use an RDF-like prefixing idea: in a given context, a short name for a peer's hash is always explicitly declared first before using it. Neither of these are entirely satisfying, and will require a bit of experimentation to get right.

The problem of managing keys and recoverability is also tricky: there's no "forgot password" link if you lose your private key. Since our system is designed to be intrinsically social, we can relax some of the more stringent requirements of zero-trust, totally-anonymous networks like IPFS and lean more on a "web of trust." We might share additional private keys with other peers that we trust to verify or recover our identity, which might be particularly useful in the case of a more stable federation of peers. We want to avoid peers operating like identity systems, as that lends itself to centralization of power and returning to a more activitypub-like style of identity, so it would be a tricky balance. Another strategy might be to use an out-of-band mechanism, like storing a URL in the signed identity object that can be used to update the public key associated with a particular identity – if you lose yours, you can generate a new keypair and update the public key stored at the URL, which another peer could check to verify that you are who you say you are. These too are not very satisfying, and so more work will be needed to draft a satisfying identity system, the practicality and usability of which will be critical for its success.

Privacy: Closely related to identity, in a p2p system any message that isn't intended to be public will need to be encrypted so that secondary peers can't just reshare something we intended to be only for them. In our system, it's not desirable to be able for some data-greedy entity to scrape all the data, we want peers to be able to make friction as-needed. To some degree this is not a solvable problem, as it's always possible to take a screenshot of the most secure end-to-end encrypted chat. It's possible even in analog social systems for secrets to slip, or for people to lie about something that another person said, so arguably the question is how to protect the things that can be verified to be from a person. Another practical problem is communicating which peers are allowed to see something so that a secondary peer knows whether or not they can help seed something: we don't want to have to transmit a list of a thousand peers along with every message, and if they have to ask the primary peer every time then the redundancy of the system is lost. Capability-based security, where permissions for a given object are conferred by having a hard to guess reference to it rather than by checking an easy to guess reference against a permissions list, seems like a good approach (see [535]). This would look a bit like generating (revokable) sharing links for different groups. Here too we might lean a bit on the social nature of our system, where peers that routinely violate the privacy requirements of other peers can be labeled untrustworthy.

Security: Most parts of this system are relatively low-risk as they are based on metadata that only relies on defined actions programmed into the receiving client — you don't get viruses merely by torrenting something. Several of the more interesting applications, though, involve a message or link being able to self-identify some code used to run or display it, and whenever you introduce running arbitrary code you introduce significant security risks. This is largely mitigated by our emphasis on non-automaticity: the default for any of these cases should be to *not* do the action. That's cold comfort, though, given the high clickthrough rates for phishing emails. More mitigation can be had by executing code in containers or virtual machines, but that too is not total. We manage to get by extraordinarily well with a very informal reputation system in open source computing. For the most part, people don't think twice about running some Python package without reading the full source code. Our system is one very conducive to soft security [536] , which is based more on accountability and resiliance than strict guarantees of security. Where typically an untrustworthy platform will do whatever it can to avoid people being able to leave comments or talk about it, in our linked data system it's always possible to issue a link saying that something is not to be trustworthy. The ability to mirror shards of our data makes any particular attack more likely to be recoverable, but special care will be needed to ensure the whole network is not subject to rolling waves of ransomware. Like cryptographers, we'll need consultation and input from the infosec crowd to make it safe, but there's nothing intrinsically more dangerous than, say, pip being able to run arbitrary code inside a `setup.py` file.

RDF Standards: RDF is highly polarizing, and many people have written it off as a lost cause because it is too complex. Much of the computing world runs off of table and relational databases rather than graphs. Though we tried to be careful to avoid endorsing any particular technology in favor of thinking about triplet links as such, the question of the literal implementation of the standards is an inevitable one. JSON-LD is, thankfully, a relatively humane standard that should be the first point of exploration. We should consider interconvertibility and interoperability with existing standards a top priority of the system in general, so we will need to make interfaces to make it trivial to interact with commonly used formats, even if it is just a wrapper that indicates the format rather than one that can convert it to JSON-LD. Interface design is one of the major missing pieces in the linked data story, and that too should be a top priority so that as little of the system as possible needs to rely on direclty interfacing with the underlying data model.

Performance: We have more or less explicitly cast performance aside as the wrong thing to optimize for: we want to have *autonomy* more than be able to blaze through the network in microseconds. Still, it's possible for technologies to be so inefficient to be nonfunctional. In a world where we have been conditioned to expect to be able to speak with a manager when our apps are not immediately responsive, or to be able to just buy whatever server performance we want, it will take some collective unlearning to rethink the internet along the lines of cooperatively managed resources. Unlearning performance will take time and has no boardroom-friendly KPIs to measure. There's no reason to believe the system will be slow before it exists, and we ultimately don't imagine this system running from residential connections and personal computers, but being a mixture of institutional and personal resources. Decentralization is a continuum, rather than a binary: we don't have to *ban* large servers from the network, but instead want to make sure that there is a healthy mix

so that the system doesn't *depend* on them. This is another place where it is useful to seed this from academia: internet service providers have historically leaned on their oligopolistic control over the underlying hardware of the internet to crush threatening technologies [537] , and we should expect no different this time. We will need to have access to commercial connections, and will likely need to convince our institutions to lobby on our behalf.

Manipulation: What if people lie? What if people purposefully rig the system by seeding it with a bunch of fake data and bad papers? People already lie! People already game the system! What we are hoping to change is to make a system where manipulation isn't built into the system as a self-reinforcing partnership between its proprieters and beneficiaries. The real dangerous thing is a system that *presents* itself as being infallible or neutral through its automaticity and glittering PR campaign. This is why we have baked the social contingency of the system so thoroughly into its design (and should investigate making triplet links into quadruple links with each having an explicit author to make it even more concrete). This is why, I believe, it is so difficult for some people to imagine a world without pre-publication peer review vouched for by a journal: the social contingency of information is scary! It is, however, preferable to the economic contingency of factuality-as-a-service.

The last set of concerns are diffuse rumblings about uptake and whether or not it is even still *possible* to challenge entrenched economic powers in science. It is true that people are cynical, and busy, and some benefit immensely from the present system, and so on. It is also true that we are likely to be met with stiff resistance if we start posing a credible threat to their dominance — the danger of opposing a set of companies who are the primary data brokers to federal law enforcement agencies, credit rating, and insurance agencies is not lost on me. I don't have any good answers to these sets of questions except that the work from here is about organizing people, adapting and responding to their needs, and making something that is useful enough for even the most complacent to adopt. I don't present this blueprint for infrastructure as infallible, and intend it to be mutated and merged with other ideas as we progress. The only thing that *isn't* an option is doing *nothing*.

12.3 In Closing

Infrastructure isn't just a technical, or even social project: it's also ethical. We started by outlining the harms of our infrastructural deficits for science, many of which are widely seen as normal, or otherwise inevitable. Some harms are only possible to recognize when it's possible to imagine an alternative to the system that causes them. This project was an attempt to help us imagine what science can be like as a guide and inspiration for us to organize to make it real. I didn't get everything right, and I probably raised more problems than I addressed. My goal more than to be right was to give a fistful of threads to pull for those that are eager to, and to make it impossible to say that a better future for science is impossible. If all we can imagine science to be is a system where we scrape by, forcing a chain of papers through a rapacious machine that turns curiosity into a treadmill of funding and prestige, playing out the clock as our working conditions deteriorate to the point where publicly funded science is nothing more than a training program for pharmaceutical and advertising companies — what are we even doing here?

Infrastructure isn't a distraction from science or something to put off as the work

of a diffuse *someone else.* It's not even an ill-defined alternative cynics use to grandstand about how much they know about how bad everything is. Collectively built infrastructure is the best way for us to make science continue to be possible. We often focus on the problems of science in isolation: what do we do about the *journals,* how do we make *scientific software* more sustainable, why is it so hard to share *data.* My central argument is that the only way we will address any of these problems is by considering the system as a whole. Rather than being a utopian vision of ripping it out from the root and starting anew in one fell swoop, considering the whole system is how we turn nibbles around the edges into coordinated mass movement. It's less about this vision matching exactly what we end up building, but making it possible for the many diverse and dispersed people who care about different facets of the problem to see it as a shared project.

Beyond any technology, my hope is that by organizing together to build something that helps us organize better, that we can re-commit to working for, instead of against each other. Some of our deeper problems like the neoliberalization of universities will only be possible to approach with a renewed sense of ourselves as often privileged, but nonetheless exploited labor. I am the first to admit my naïveté, but I think a nontrivial part of the lack of labor consciousness in science is the way our systems of work, communication, and evaluation feed back into an individualist celebration of the hero of knowledge. Maybe by rebuilding those systems to support the abundance of cooperation and make collective organization a central part of our work can help us both do better science and make science better (also see [538]).

Science for science's sake also misses the point. The dominant stories we tell of how science can give back to society are also shot through with market individualism: become a scicomm influencer, or found a start-up. Instead of giving back to a society that we are somehow separate from, we can take our part in shared struggle seriously, like the graduate workers at Columbia and Harvard who did us all proud by fighting like hell through the strike wave this year and last [539, 540] . Even for the most basic research-oriented, the problem of informational dominance in the 21st century tolls for thee. Systems like those described here could serve as a basis for a new kind of digital infrastructure that challenges the basic platform model of the internet more broadly [541] . What are the three to five remaining websites but data storage, computation, and communication systems? By organizing to make our own work better, we might also seed the systems that help reclaim digital infrastructure as something that empowers everyone, rather than uses our urge to connect with each other to control us.

It was scientists[1] looking for a better way to communicate that created the internet in the first place, radically rewriting the course of history [542] — and we can do it again.

[1] With funding from the military

12.4 Contrasting Visions of Science

Through this text I have tried to sketch in parallel a potentially liberatory infrastructural future with the many offramps and alternatives that could lead us astray, but to make two of those futures clearer, it's worth imagining them outright.

12.4.1 What if we do nothing?

You're a researcher with dead-center median funding at an institute with dead-center median prestige, and you have a new idea.

The publishing industry has built its surveillance systems into much of the practice of science: their SeamlessAccess login system and browser fingerprinting harvest your reading patterns across the web[543, 544, 545, 546, 547] , Mendeley watches what you highlight and how you organize papers, and with a data sharing agreement with Google crossreference and deanonymize your drafts in progress [214] . Managing constant surveillance is a normal part of doing science now, so when reading papers you are careful to always use a VPN, stay off the WiFi whenever possible, randomly scroll around the page to appear productive while the PDF is printing to read offline. The publishers have finally managed to kill sci-hub with a combination of litigation and lobbying universities to implement mandatory multifactor authentication, cutting off their ability to scrape new papers. The few papers you're able to find, and fewer that you're able to access, after several weeks of carefully covering your tracks while hopping citation trees make you think your hunch might be right — you're on to something.

This is a perfect project for a collaboration with an old colleague from back in grad school. Their SciVal Ranking is a little low, so you're taking a risk by working with them, but friendship has to be worth something right? "Don't tell me I never did nothing for you." You haven't spoken in many years though, so you have to be careful on your approach. The repackaged products of all their surveillance are sold back to the few top-tier labs able to afford the hype-prediction products that steer all of their research programs [548, 549] . The publishers sell tips on what's hot, and since they sell the same products to granting agencies and control the publishing process, every prediction can be self-fulfilling — the product is plainly prestige, and the product is good. If you approach your colleague carelessly, they could turn around and plug the idea into the algorithm to check its score, tipping off the larger labs that can turn their armies of postdocs on a dime to pounce. There is no keeping up with the elites anymore.

Even if you do manage to keep it a secret, it'll be a hard road to pull off the experiment at all. There are a few scattered open source tools left, but the rest have been salami sliced into a few dozen mutually incompatible platforms (compatibility only available with the HyperGold Editions). The larger labs are able to afford all the engineers they need to build tools, but have little reason to share any of the technical knowledge with the rest of us — why should they spoil the chance to spin it off into a startup? There aren't any jobs left in academia anyway.

Industry capture has crept into ever more of the little grant funding you have, all the subscriptions and fees add up, so you can only afford to mentor one grad student at a time while keeping plausibly up to date with new instrument technology. You can't choose who they are anymore really. The candidate ranking algorithms have thoroughly baked the exclusionary biases of the history of science into the pool of applicants[214, 222] , so the only ones left are those who have been playing to the algorithm since they were in middle school. Advocates for any sort of diversity in academia are long gone. We've never been able to confirm it, but everyone knows that the publishers tip the scales of the algorithm to downrank anyone who starts organizing against them.

Your colleague and you manage to coordinate. they're the same as they've always been, trustworthy. You really need someone from a different field at least in consultation, but there isn't really a good way to find who would be a good fit. Somehow Twitter is still the best way to communicate at large, but you've never really gotten how it works and the discourse has gotten *dark* so you don't have enough followers to reach outside your small bubble of friends. You decide to go it your own, and find the best papers you can from what you think is the right literature base, but there's no good way of knowing you're following the right track. Maybe that part of the paper is for the supplement.

Data is expensive, if you can find it. Who can pay the egress costs for several TB anymore? You forego some modeling that would help with designing the experiment because you don't have the right subscription to access the data you need. You'll have to wait until there is a promotional event to to get some from a Science Influencer.

You experiment in public silence until you've collected your data. Phew, probably safe from getting scooped. You start the long slog of batch analysis with the scraps of Cloud Compute time you can afford.

Papers are largely unchanged, still the same old PDFs. They're a source of grim nostalgia, at least we'll always have PDF. What has changed is citation: since it's the major component of the ranking algorithm, nobody cites to reference ideas anymore, just to try and keep their colleagues afloat. The researchers who still care about the state of science publish a parallel list of citations for those who still care to read them, but most just ignore them — the past is irrelevant anyway, the only way to stay afloat is hunting hype. You know this is distorting the literature base, feeding the algorithm junk data that will steer the research recommendations off course, but you don't want to see your colleague down the hall fired [222] . Their rankings have been sinking lately.

Uploading preprints is expensive now too, and they charge by the version, so you make sure you've checked every letter before sending it off. It's a really compelling bit of science, some of that old style science, fundamental mechanisms, basic research kind of stuff. You check your social media metrics to perfectly time your posts about it, click send, and wait. Your friends reply with their congratulations, glad you managed to pull it off, but there's not really a lot that can be made a meme of, and it's not inflammatory enough to bait a sea of hot takes. You watch your Altmetric idle and sigh. You won't get a rankings boost, but at least it looks like you're safe from sinking for awhile.

You're going to take a few weeks off before starting the multi-year process of publication. Few researchers are willing to review for free anymore, everyone is sick of publisher profiteering, but we didn't manage to build an alternative in time, and now it's too dangerous to try. Triage at the top of the journal prestige hierarchy is ruthless. Most submissions not pre-coordinated with the editor are pre-desk rejected after failing any one of the dozen or so benchmarks for "quality" and trendiness crunched by their black box algorithms. Instead we ping-pong papers down the hierarchy, paying submission fees all along the way. Don't worry, there's always some journal that will take any work — they want the publication fees in any case. If you're cynically playing the metrics game, you can rely on the class of blatantly sacrificial junk journals that can be hastily folded up when some unpaid PubPeer blogger manages to summon enough outrage on social media. We haven't managed

to fix the problems with peer review that favor in-crowd, clickbait-friendly, though not necessarily reproducible, research. It turned out to have been a feature, not a bug for their profit model all along.

You're not sure if you've made a contribution to the field, there isn't any sense of cumulative consensus on basic problems. People study things that are similar to you, lots of them, and you talk. You forget what they've been doing sometimes, though, and you catch what you can. You like your work, and even find value in it. You can forget about the rest when you do it. And you like your lab. The system isn't perfect but everyone knows that. Some good science still gets done, you see it all the time from the people you respect. It's a lot of work to keep track of, at least without the subscription. But you managed to make it through another round. That feels ok for now. And it's not your job, your job is to do science.

The attention span of your discipline has gotten shorter and shorter, twisting in concentric hype cycles, the new *rota fortuna.* It's good business, keeping research programs moving helps the other end of the recommendation system. It started with advertising that looked like research [371] , but the ability to sell influence over the course of basic science turned out to be particularly lucrative. Just little nudges here and there, you know, just supply responding to demand. They turn a blind eye to the botnets hired to manipulate trending research topics by simulating waves of clicks and scrolls. More clicks, more ads, the market speaks, everybody wins.

The publishers are just one piece of the interlocking swarm of the information economy. The publishers sell their data to all the others, and buy whatever they need to complete their profiles. They move in lockstep: profit together, lobby together. The US Supreme Court is expected to legalize copyrighting facts soon, opening up new markets for renting licenses to research by topic area. No one really notices intellectual property expansions anymore. There are more papers than ever, but the science is all "fake news." Nobody reads it anyway.

12.4.2 What we could build

You're a researcher with dead-center median funding at an institute with dead-center median prestige, and you have a new idea.

You are federated with a few organizations in your subdiscipline that have agreed to share their full namespaces, as well as a broader, public multidisciplinary indexing federation that organizes metadata more coarsely. You navigate to a few nodes in the public index that track work from some related research questions. You're able to find a number of forum conversations, blog posts, and notebooks in the intersection between the question nodes, but none that are exactly what you're thinking about. There's no such thing as paywalls anymore, but some of the researchers have requested to be credited on view, so you accept the prompts that make a `read` link between you and their work. You can tell relatively quickly that there is affirmatively a gap in understanding here, rather than needing to spend weeks reading to rule it out by process of elimination — you're on to something.

You request access to some of the private sections of federations that claim to have data related to the question nodes. They have some writing, data, and code public, but the data you're after is very raw and was never written up — just left with a reference to a topic in case someone else wanted to use it later. Most accept you since they can see your affiliation in good standing with people and federations they know and

trust. Others are a little more cagey, asking that you request again when you have a more developed project rather than just looking around so they can direct your permissions more finely, or else not responding at all. The price of privacy, autonomy, and consent: we might grumble about it sometimes, but all things considered are glad to pay it.

Your home federations have a few different names for things than those you've joined, so you spend a few hours making some new mappings between your communities, and send them along with some terms they don't have but you think might be useful for them and post them to their link proposals inbox. They each have their own governance process to approve the links and associate them with their namespace, but in the meantime they exist on yours so you use them to start gathering and linking data from a few different disciplines to answer some preliminary questions you have. In the course of feeling out a project, you've made some new connections between communities, concepts, and formats, and made incremental improvements on knowledge organization in multiple fields. You're rehosting some of their data as a gesture of good faith, because you're using it and it's become part of your project, (and because a few of the federations have ratio requirements).

You do some preliminary analysis to refine your hypotheses and direct the experimental design. You are able to find some analysis code from your new colleagues in a notebook linked to the data of theirs that you're using. It doesn't do *exactly* what you want, but you're able to extend it to do a variation on the analysis and link it from their code in case anyone else wants to do something similar.

You post a notebook of some preliminary results from your secondary analysis and a brief description of your idea and experimental plan in a thread that is transcluded between the forums of the now several federations involved in your project. There's little reason to fear being scooped: since you're in public conversation with a lot of the people in the relevant research areas, and have been linking your work to the concepts and groups that any competitor also would have to, it doesn't really make sense to try and rush out a result faster than you to take credit for your ideas. All the provenance of your conversations and analyses is already public, and so if someone did try and take credit for your idea, you would be able to link to their work with some "uncredited derivation" link.

In the thread, several people from another discipline point out that they have already done some of what you planned to do, so you link to their post to give them credit for pointing you in the right direction and transclude the relevant work in your project. Others spitball some ideas for refinements to the experiment, and try out alternate analysis strategies on your preliminary results. It's interesting and useful, you hadn't thought about it that way. They give you access to some of their nonpublic datasets that they never had a chance to write up. It'll be useful in combination with your experimental results, and in the process you'll be helping them analyze and interpret their unused data.

You're ready to start your experiment. They say an hour in the library is worth a thousand at the bench, and your preliminary work has let you skip about a third of what you had initially planned to do. The project gives credit and attribution to the many people whose work you are building on and who have helped you so far, and has been made richer from the discussion and half dozen alternative analyses proposed and linked from your thread.

Some of the techniques and instruments are new to you, but you're able to piece

together how they work by surfing between the quasi-continuous wikis shared between federations. Hardware still costs money, but since most people able to make do with less specialized scientific instruments because of the wealth of DIY instrument documentation, and scientists are able to maintain grant funded nonprofit instrument fabrication organizations because their work is appropriately credited by the work that uses them, it's a lot less expensive. You try out some parameter sets and experiment scripts in your experimental software linked by some technical developers in the other fields. You get to skip a lot of the fine tuning by making use of the contextual knowledge: less dead ends on the wrong equipment, not having to rediscover the subtleties of how the parameters interact, knowing that the animals do the experiment better if the second phase is delayed by a second or two more than you'd usually think. Your experimental software lets you automatically return the favor, linking your new parameters and experimental scripts as extensions of the prior work.

While you were planning and discussing your experiment you had been contributing your lab's computing hardware to a computational co-op so other people could deploy analyses on it while it was idle. Now you have some credit stored up and distribute the chunks of your analysis across the network. It takes a little bit of tweaking to get some of the more resource-intensive analysis steps to work on the available machines. You don't have time to organize a full pull request to the main analysis code, but if someone wants to do something similar they'll be able to find your version since it's linked to the main library as well as the rest of your project.

You combine the various intermediary results you have posted and been discussing in the forums into a more formal piece of writing. You need to engage with the legacy scientific literature for context, so you highlight the segments you need and make direct reference to and transclude the arguments that they are making in your piece. While you're writing you annotate inline how your work `[[extends::@oldWork]]` because it `[[hasPerspective::@newDiscipline]]`. Some of your results `[[contradict::@oldWork:a-claim]]` and so the people who have published work affirming it are notified and invited to comment.

There isn't any need for explicit peer review to confirm your work as "real science" or not. The social production of science is very visible already, and the smaller pieces you have been discussing publicly are densely contextualized by affirmative and skeptical voices from the several disciplines you were engaging with. You have `@public` annotations enabled on my writing, so anyone reading my work is able to see the inbound links from others highlighting and commenting on it. Submitting in smaller pieces with continual feedback has let you steer your work in more useful directions than your initial experimental plan, so you've already been in contact with many of the people who would otherwise have been your biggest skeptics and partially addressed their concerns. People are used to assessing the social context of a work: the interfaces make it visually obvious that work that has few annotations, a narrow link tree, or has a really restricted circle of people able to annotate it has relatively less support. When a previously well-supported set of ideas is called into question by new methods or measurements, it's straightforward to explore how its contextual understanding has changed over time.

It's rare for people to submit massive singular works with little public engagement beforehand. There isn't a lot of reward for minimal authorship because the notion of "authorship" has been dissolved in favor of fluid and continuous credit assignment — engaging with little prior work and making few contributions to the data

and tooling where it would have been obvious to do so is generally seen as antisocial. They are in the unenviable position of having sunk several years of work into a flawed experimental design that many people in the community could have warned about and helped with, but now since the criticisms are annotated on their work they likely will have to do yet more work if they can't be adequately addressed or dismissed. We don't miss the old system of peer review.

It's clear that you have made a contribution to not only your field, but several that you collaborated with. Your project is a lot more than a single PDF: you can see (and be credited for) the links between data formats, communities, forum posts, notebooks, analytical tools, theories, etc. that you created. It's clear how your work relates to and extends prior work because you were engaging with the structure of scientific research throughout. Your work implies further open questions in the open spaces in the concept graphs of several different research communities, and can organize future experiments without the need for explicit coordination.

There are a dozen or so metrics that are used to evaluate research and researchers. None of them are exactly neutral, and there is ongoing debate about the meaning and use of each since there are so many modalities of credit in a given person's graph. There isn't such a thing as a *proprietary* metric though, because no company has a monopoly on proprietary information that they could say makes it unique, and why would you trust a random number given by a company when there are plenty of ways to measure the public credit graphs? It's relatively hard to game the system, there aren't any proprietary algorithms to fool, and trust is a social process based on mutual affiliation instead of a filter bubble.

The public inspectability of scientific results, the lowered barriers to scientific communication, and ability to find research and researchers without specialized training has dramatically changed between science and the public at large. It's straightforward to find a community of scientists for a given topic and ask questions in the public community forums. Scientific communication resembles the modes of communication most people are familiar with, and have shed some of the stilted formality that made it impenetrable. There isn't such a firm boundary between 'scientist' and 'nonscientist' because anyone can make use of public data and community clusters to make arguments on the same forums and feeds that the scientists do with the same mechanism of credit assignment.

Scientists, building new systems of communication and tooling and then seeding them with their communities has provided alternatives to some of the platforms that dominated the earlier web. The scientists were able to use some of their labor and funding to overcome the development problems of prior alternatives, so they are just as easy to use as (and much more fun than) platforms like Twitter and Facebook. Their well-documented and easily deployed experimental hardware and software has empowered a new generation of DIY enthusiasts, making it possible for many people to build low-cost networked electronics to avoid the surveillance of the ad-based "Internet of Things," air quality sensors, medical devices, wireless meshnets, and so on. The scientists helped make controlling and using personal data much more accessible and fluid. We now control our own medical data and selectively share it as-needed with healthcare providers. Mass genetics databases collected by companies like 23andme and abused by law enforcement slowly fall out of date because we can do anything the geneticists can do.

By taking seriously the obligation conferred by their stewardship of the human knowl-

edge project, the scientists rebuilt their infrastructure to serve the public good instead of the companies that parasitize it. In the process they did their part ending some of the worst harms of the era of global information oligopoly.

Most things aren't completely automatic or infinite, but you don't want them to be. It's nice to negotiate with your federations and communities, it makes you feel like a person instead of a product. Being in a silent room where algorithms shimmer data as a dark wind friction-free through the clouds sounds lonely. Now we are the winds and clouds and the birds that gossip between them, and all the chatter reminds us that we forgot what we were taught to want. You take the hiccups and errors and dead links as the work of the world we built together.

Everything is a little rough, a little gestural, and all very human.

[?]

13

Bibliography

[1] Philip Meier, Erik Flister, and Pamela Reinagel. Collinear features impair visual detection by rats. *Journal of Vision*, 11 (3), March 2011. ISSN 1534-7362. https://doi.org/10.1167/11.3.22. (document), II

[2] Santiago Jaramillo, Anna Lakunina, Lan Guo, and Nick Ponvert. TASKontrol, January 2022. (document), II

[3] David Clark. A Cloudy Crystal Ball - Visions of the Future. In *Proceedings of the Twenty-Fourth Internet Engineering Task Force*, pages 539–543, July 1992. (document), 10.2.4

[4] Jonny L. Saunders and Michael Wehr. Mice can learn phonetic categories. *The Journal of the Acoustical Society of America*, 145(3):1168–1177, March 2019. ISSN 0001-4966. https://doi.org/10.1121/1.5091776. 1, 2.5, 4.4

[5] Harvey M Sussman, David Fruchter, Jon Hilbert, and Joseph Sirosh. Linear correlates in the speech signal: The orderly output constraint. *The Behavioral and brain sciences*, 21(2):241–59; discussion 260–99, 1998. ISSN 0140-525X. https://doi.org/10.1017/S0140525X98001174. 1.1.1, 1.1.5, 1.2.3, 1.3

[6] L. L. Holt and A. J. Lotto. Speech perception as categorization. *Attention, Perception & Psychophysics*, 72(5):1218–1227, July 2010. ISSN 1943-3921. https://doi.org/10.3758/APP.72.5.1218. 1.1.1

[7] Yakov Kronrod, Emily Coppess, and Naomi H. Feldman. A unified account of categorical effects in phonetic perception. *Psychonomic Bulletin & Review*, 23(6):1681–1712, December 2016. ISSN 1069-9384. https://doi.org/10.3758/s13423-016-1049-y. 1.1.1, 2.2

[8] A M LIBERMAN, K S HARRIS, H S HOFFMAN, and B C GRIFFITH. The discrimination of speech sounds within and across phoneme boundaries. *Journal of experimental psychology*, 54(5):358–68, November 1957. ISSN 0022-1015. 1.1.1

[9] J L Elman and D Zipser. Learning the hidden structure of speech. *The Journal of the Acoustical Society of America*, 83(4): 1615–26, April 1988. ISSN 0001-4966. 1.1.1

[10] K R Kluender, R L Diehl, and P R Killeen. Japanese quail can learn phonetic categories. *Science (New York, N.Y.)*, 237 (4819):1195–1197, September 1987. ISSN 0036-8075. https://doi.org/10.1126/science.3629235. 1.1.1, 1.1.2, 1.3, 1.4.1

[11] A M Liberman, F S Cooper, D P Shankweiler, and M Studdert-Kennedy. Perception of the speech code. *Psychological review*, 74(6):431–61, November 1967. ISSN 0033-295X. 1.1.1, 1.1.2

[12] E. Farnetani. V-C-V Lingual Coarticulation and Its Spatiotemporal Domain. In *Speech Production and Speech Modelling*, pages 93–130. Springer Netherlands, Dordrecht, 1990. https://doi.org/10.1007/978-94-009-2037-8_5. 1.1.1

[13] Randy L. Diehl, Andrew J. Lotto, and Lori L. Holt. Speech Perception. *Annual Review of Psychology*, 55(1):149–179, February 2004. ISSN 0066-4308. https://doi.org/10.1146/annurev.psych.55.090902.142028. 1.1.1, 1.1.2

[14] Joseph S. Perkell, Dennis H. Klatt, Kenneth N. Stevens, and Symposium on Invariance and Variability of Speech Processes (1983 : Massachusetts Institute of Technology). *Invariance and Variability in Speech Processes*. Lawrence Erlbaum Associates, 1986. ISBN 0-89859-545-2. 1.1.1

[15] Philip. Lieberman. *The Biology and Evolution of Language*. Harvard University Press, 1984. ISBN 0-674-07413-0. 1.1.2

[16] Alvin M. Liberman and Ignatius G. Mattingly. The motor theory of speech perception revised. *Cognition*, 21(1):1–36, October 1985. ISSN 00100277. https://doi.org/10.1016/0010-0277(85)90021-6. 1.1.2, 1.1.3

[17] Kathy M. Carbonell and Andrew J. Lotto. Speech is not special... again. *Frontiers in psychology*, 5(June):427, June 2014. ISSN 1664-1078. https://doi.org/10.3389/fpsyg.2014.00427. 1.1.2, 2.2

[18] Asif A. Ghazanfar and Marc D. Hauser. The neuroethology of primate vocal communication: Substrates for the evolution of speech. *Trends in Cognitive Sciences*, 3(10):377–384, 1999. ISSN 13646613. https://doi.org/10.1016/S1364-6613(99)01379-0. 1.1.2

[19] Ina Bornkessel-Schlesewsky, Matthias Schlesewsky, Steven L. Small, and Josef P. Rauschecker. Neurobiological roots of language in primate audition: Common computational properties. *Trends in Cognitive Sciences*, 19(3):142–150, March 2015. ISSN 13646613. https://doi.org/10.1016/j.tics.2014.12.008. 1.1.2

[20] K R Kluender and a J Lotto. Effects of first formant onset frequency on [-voice] judgments result from auditory processes not specific to humans. *The Journal of the Acoustical Society of America*, 95(2):1044–52, 1994. ISSN 0001-4966. https://doi.org/10.1121/1.408466. 1.1.2

[21] Patricia K. Kuhl and James D. Miller. Speech perception by the chinchilla: Identification functions for synthetic VOT stimuli. *The Journal of the Acoustical Society of America*, 63(3):905–917, 1978. ISSN 00014966. https://doi.org/10.1121/1.381770. 1.1.2, 1.3

[22] Crystal T. Engineer, Kimiya C. Rahebi, Elizabeth P. Buell, Melyssa K. Fink, and Michael P. Kilgard. Speech training alters consonant and vowel responses in multiple auditory cortex fields. *Behavioural Brain Research*, 287:256–264, 2015. ISSN 18727549. https://doi.org/10.1016/j.bbr.2015.03.044. 1.1.2, 1.2.1, 1.3, 1.4.1

[23] P K Kuhl and D M Padden. Enhanced discriminability at the phonetic boundaries for the place feature in macaques. *The Journal of the Acoustical Society of America*, 73(3):1003–1010, 1983. ISSN 0001-4966. https://doi.org/10.3758/BF03204208. 1.1.2, 1.3

[24] Robert J Dooling, Catherine T. Best, and Susan D. Brown. Discrimination of synthetic full-formant and sinewave /ra–la/ continua by budgerigars (Melopsittacus undulatus) and zebra finches (Taeniopygia guttata). *The Journal of the Acoustical Society of America*, 97(3):1839–1846, 1995. ISSN 00014966. https://doi.org/10.1121/1.412058. 1.1.2, 1.3

[25] AJ Lotto, KR Kluender, and LL Holt. Animal models of speech perception phenomena. *Chicago Linguistic Society*, 1997. 1.1.2, 1.3, 2.2

[26] Keith R. Kluender. Contributions of nonhuman animal models to understanding human speech perception. *The Journal of the Acoustical Society of America*, 107(5):2835–2835, May 2000. ISSN 0001-4966. https://doi.org/10.1121/1.429153. 1.1.2, 1.3, 2.2

[27] Jennifer K Bizley and Yale E Cohen. The what, where and how of auditory-object perception. *Nature reviews. Neuroscience*, 14(10):693–707, October 2013. ISSN 1471-0048. https://doi.org/10.1038/nrn3565. 1.1.2

[28] Josef P Rauschecker and Sophie K Scott. Maps and streams in the auditory cortex: Nonhuman primates illuminate human speech processing. *Nature Neuroscience*, 12(6):718–724, June 2009. ISSN 1097-6256. https://doi.org/10.1038/nn.2331. 1.1.3

[29] Ted J. Strauss, Harlan D. Harris, and James S. Magnuson. jTRACE: A reimplementation and extension of the TRACE model of speech perception and spoken word recognition. *Behavior Research Methods*, 39(1):19–30, February 2007. ISSN 1554-351X. https://doi.org/10.3758/BF03192840. 1.1.3

[30] Keith R Kluender, Christian E Stilp, Michael Kiefte, K R Kluender, C E Stilp, and M Kiefte. Perception of Vowel Sounds Within a Biologically Realistic Model of Efficient Coding. In *Vowel Inherent Spectral Change*, pages 117–151. 2013. ISBN 978-3-642-14209-3. https://doi.org/10.1007/978-3-642-14209-3_6. 1.1.3

[31] M. Gareth Gaskell and William D. Marslen-Wilson. Integrating Form and Meaning: A Distributed Model of Speech Perception. *Language and Cognitive Processes*, 12(5-6):613–656, October 1997. ISSN 0169-0965. https://doi.org/10.1080/016909697386646. 1.1.3

[32] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154.2, January 1962. ISSN 0022-3751. https://doi.org/10.1523/JNEUROSCI.1991-09.2009. 1.1.3

[33] K. G. Ranasinghe, W. A. Vrana, C. J. Matney, and M. P. Kilgard. Increasing diversity of neural responses to speech sounds across the central auditory pathway. *Neuroscience*, 252:80–97, 2013. ISSN 03064522. https://doi.org/10.1016/j.neuroscience.2013.08.005. 1.1.3

[34] Edward L. Bartlett. The organization and physiology of the auditory thalamus and its role in processing acoustic features important for speech perception. *Brain and language*, 126(1):29–48, July 2013. ISSN 10902155. https://doi.org/10.1016/j.bandl.2013.03.003. 1.1.3

[35] T. M. Centanni, A. M. Sloan, A. C. Reed, C. T. Engineer, R. L. Rennaker, and M. P. Kilgard. Detection and identification of speech sounds using cortical activity patterns. *Neuroscience*, 258:292–306, 2013. ISSN 03064522. https://doi.org/10.1016/j.neuroscience.2013.11.030. 1.1.3

[36] Crystal T Engineer, Claudia A Perez, YeTing H Chen, Ryan S Carraway, Amanda C Reed, Jai A Shetake, Vikram Jakkamsetti, Kevin Q Chang, and Michael P Kilgard. Cortical activity patterns predict speech discrimination ability. *Nature neuroscience*, 11(5):603–8, May 2008. ISSN 1097-6256. https://doi.org/10.1038/nn.2109. 1.1.3

[37] Mitchell Steinschneider, Yonatan I Fishman, and Joseph C Arezzo. Representation of the voice onset time (VOT) speech parameter in population responses within primary auditory cortex of the awake monkey. *The Journal of the Acoustical Society of America*, 114(1):307–321, 2003. ISSN 00014966. https://doi.org/10.1121/1.1582449. 1.1.3

[38] Nima Mesgarani, Connie Cheung, Keith Johnson, and Edward F Chang. Phonetic feature encoding in human superior temporal gyrus. *Science (New York, N.Y.)*, 343(6174):1006–10, 2014. ISSN 1095-9203. https://doi.org/10.1126/science.1245994. 1.1.3

[39] P Belin, R J Zatorre, P Lafaille, P Ahad, and B Pike. Voice-selective areas in human auditory cortex. *Nature*, 403(6767):309–312, 2000. ISSN 0028-0836. https://doi.org/10.1038/35002078. 1.1.3

[40] Edward F Chang, Jochem W Rieger, Keith Johnson, Mitchel S Berger, Nicholas M Barbaro, and Robert T Knight. Categorical speech representation in human superior temporal gyrus. *Nature neuroscience*, 13(11):1428–32, November 2010. ISSN 1546-1726. https://doi.org/10.1038/nn.2641. 1.1.3

[41] Brian N. Pasley, Stephen V. David, Nima Mesgarani, Adeen Flinker, Shihab A. Shamma, Nathan E. Crone, Robert T. Knight, and Edward F. Chang. Reconstructing speech from human auditory cortex. *PLoS Biology*, 10(1), 2012. ISSN 15449173. https://doi.org/10.1371/journal.pbio.1001251. 1.1.3

[42] Gavin M. Bidelman, Sylvain Moreno, and Claude Alain. Tracing the emergence of categorical speech perception in the human auditory system. *NeuroImage*, 79:201–212, 2013. ISSN 10538119. https://doi.org/10.1016/j.neuroimage.2013.04.093. 1.1.3

[43] Andrew Ng and Michael I. Jordan. On generative vs. discriminative classifiers: A comparison of logistic regression and naive bayes. *Proceedings of Advances in Neural Information Processing*, 28(3):169–187, 2002. ISSN 13704621. https://doi.org/10.1007/s11063-008-9088-7. 1.1.3

[44] Ferdinand de Saussure. *Cours de Linguistique Générale*. Payot, Lausanne, Paris, 1916. 1.1.3

[45] Ueli Rutishauser, Jean Jacques Slotine, and Rodney Douglas. Computation in Dynamically Bounded Asymmetric Systems. *PLoS Computational Biology*, 11(1):e1004039, 2015. ISSN 15537358. https://doi.org/10.1371/journal.pcbi.1004039. 1.1.3

[46] B Elan Dresher. The contrastive hierarchy in phonology. *Contrast in phonology: theory, perception, acquisition*, 13:11, 2008. ISSN 1718-3510. https://doi.org/10.1017/CBO9780511642005. 1.1.3, 2.4

[47] Helen Blank and Matthew H Davis. Prediction Errors but Not Sharpened Signals Simulate Multivoxel fMRI Patterns during Speech Perception. *PLoS Biology*, 14(11):e1002577, November 2016. ISSN 1545-7885. https://doi.org/10.1371/journal.pbio.1002577. 1.1.3

[48] Pierre Gagnepain, Richard N. Henson, and Matthew H. Davis. Temporal Predictive Codes for Spoken Words in Auditory Cortex. Technical Report 7, 2012. 1.1.3

[49] Neal P. Fox and Sheila E. Blumstein. Top-down effects of syntactic sentential context on phonetic processing. *Journal of Experimental Psychology: Human Perception and Performance*, 42(5):730–741, May 2016. ISSN 1939-1277. https://doi.org/10.1037/a0039965. 1.1.4

[50] Bert Schouten, Ellen Gerrits, and Arjan Van Hessen. The end of categorical perception as we know it. In *Speech Communication*, volume 41, pages 71–80, 2003. ISBN 0167-6393. https://doi.org/10.1016/S0167-6393(02)00094-8. 1.1.4

[51] Lawrence D. Rosenblum. Speech Perception as a Multimodal Phenomenon. *Current Directions in Psychological Science*, 17(6):405–409, December 2008. ISSN 0963-7214. https://doi.org/10.1111/j.1467-8721.2008.00615.x. 1.1.4

[52] PK Kuhl, KA Williams, F Lacerda, KN Stevens, and B Lindblom. Linguistic experience alters phonetic perception in infants by 6 months of age. *Science*, 255(5044), 1992. 1.1.4

[53] E. A. Brenowitz, D. Margoliash, and K. W. Nordeen. An introduction to birdsong and the avian song system. *Journal of Neurobiology*, 33(5):495–500, 1997. ISSN 00223034. 1.1.5

[54] Frédéric E Theunissen and Julie E Elie. Neural processing of natural sounds. *Nature reviews. Neuroscience*, 15(6):355–66, 2014. ISSN 1471-0048. https://doi.org/10.1038/nrn3731. 1.1.5

[55] Gordon E Peterson and Harold L Barney. Control methods used in a study of the vowels. *The Journal of the Acoustical Society of America*, 24(2):175–184, 1952. ISSN 00014966. https://doi.org/10.1121/1.1906875. 1.1.5

[56] Richard Wright. A review of perceptual cues and cue robustness. In Bruce Hayes, Donca Steriade, and Robert Kirchner, editors, *Phonetically Based Phonology*, pages 34–57. Cambridge University Press, Cambridge, 2004. ISBN 978-0-521-82578-8. https://doi.org/10.1017/CBO9780511486401.002. 1.2.3

[57] Björn Lindblom and Harvey M. Sussman. Dissecting coarticulation: How locus equations happen. *Journal of Phonetics*, 40(1):1–19, 2012. ISSN 00954470. https://doi.org/10.1016/j.wocn.2011.09.005. 1.2.3

[58] S. Sadagopan and X. Wang. Nonlinear Spectrotemporal Interactions Underlying Selectivity for Complex Sounds in Auditory Cortex. *Journal of Neuroscience*, 29(36):11192–11202, September 2009. ISSN 0270-6474. https://doi.org/10.1523/JNEUROSCI.1286-09.2009. 1.3

[59] Xiaoqin Wang, Thomas Lu, Ross K Snider, and Li Liang. Sustained firing in auditory cortex evoked by preferred stimuli. *Nature*, 435(7040):341–6, 2005. ISSN 1476-4687. https://doi.org/10.1038/nature03565. 1.3, 2.5

[60] Kelly E Radziwon, Kristie M June, Daniel J Stolzberg, Matthew A Xu-Friedman, Richard J Salvi, and Micheal L Dent. Behaviorally measured audiograms and gap detection thresholds in CBA/CaJ mice. *Journal of comparative physiology. A, Neuroethology, sensory, neural, and behavioral physiology*, 195(10):961–9, October 2009. ISSN 1432-1351. https://doi.org/10.1007/s00359-009-0472-1. 1.4.2

[61] Paul Boersma. Praat, a system for doing phonetics by computer. *Glot International*, 5(9/10):341–347, 2001. ISSN 0196-0202. https://doi.org/10.1097/AUD.0b013e31821473f7. 1.4.2

[62] H. D. Patterson and R. Thompson. Recovery of Inter-Block Information when Block Sizes are Unequal. *Biometrika*, 58 (3):545, December 1971. ISSN 00063444. https://doi.org/10.2307/2334389. 1.4.3

[63] R Core Team. R: A language and environment for statistical computing., 2016. 1.4.4

[64] RStudio Team. RStudio: Integrated Development for R., 2015. 1.4.4

[65] Douglas Bates, Martin Machler, Ben Bolker, and Steve Walker. Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. https://doi.org/10.18637/jss.v067.i01. 1.4.4

[66] Martin Maechler, Peter Rousseeuw, Anja Struyf, Mia Hubert, and Kurt Hornik. Cluster: Cluster Analysis Basics and Extensions, 2017. 1.4.4

[67] Dorai-Raj Sundar. Binom: Binomial Confidence Intervals For Several Parameterizations, 2014. 1.4.4

[68] Hadley Wickham. Reshaping Data with the reshape Package. *Journal of Statistical Software*, 21:1–20, November 2007. ISSN 1548-7660. https://doi.org/10.18637/jss.v021.i12. 1.4.4

[69] Hadley Wickham. The Split-Apply-Combine Strategy for Data Analysis. *Journal of Statistical Software*, 40(1):1–29, 2011. 1.4.4

[70] Hadley Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag, New York, NY, 2009. ISBN 978-0-387-98140-6. 1.4.4

[71] David B. Dahl. Xtable: Export Tables to LaTeX or HTML, 2016. 1.4.4

[72] Julienne LaChance, Manuel Schottdorf, Tom J. Zajdel, Jonny L. Saunders, Sophie Dvali, Chase Marshall, Lorenzo Seirup, Ibrahim Sammour, Robert L. Chatburn, Daniel A. Notterman, and Daniel J. Cohen. PVP1—The People's Ventilator Project: A fully open, low-cost, pressure-controlled ventilator research platform compatible with adult and pediatric uses. *PLOS ONE*, 17(5):e0266810, May 2022. ISSN 1932-6203. https://doi.org/10.1371/journal.pone.0266810. 1.5

[73] Ludwig Wittgenstein. *Philosophical Investigations*. Basil Blackwell, Oxford, 1968. ISBN 978-0-631-11900-5. 2

[74] Lori L. Holt and Andrew J. Lotto. Speech perception as categorization. *Attention, Perception, & Psychophysics*, 72(5): 1218–1227, July 2010. ISSN 1943-393X. https://doi.org/10.3758/APP.72.5.1218. 2, 2.4

[75] Leigh Lisker. Rapid versus rabid: A catalogue of acoustic features that may cue the distinction. *The Journal of the Acoustical Society of America*, 62(S1):S77, 1977. ISSN 00014966. https://doi.org/10.1121/1.2016377. 2, 2.2

[76] P J Bailey and Q Summerfield. Information in speech: Observations on the perception of [s]-stop clusters. *Journal of experimental psychology. Human perception and performance*, 6(3):536–563, August 1980. ISSN 0096-1523. https://doi.org/10.1037/0096-1523.6.3.536. 2, 2.2, 2.4

[77] Eleanor Rosch and Carolyn B Mervis. Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology*, 7(4):573–605, October 1975. ISSN 0010-0285. https://doi.org/10.1016/0010-0285(75)90024-9. 2, 2.4

[78] S Edelman. Representation is representation of similarities. *The Behavioral and brain sciences*, 21(4):449–67; discussion 467–98, August 1998. ISSN 0140-525X. 2.1

[79] Amos Tversky. Features of Similarity. 84(4), 1977. 2.1, 2.1

[80] Michael D. Lee and Daniel J. Navarro. Extending the ALCOVE model of category learning to featural stimulus domains. *Psychonomic Bulletin & Review*, 9(1):43–58, March 2002. ISSN 1531-5320. https://doi.org/10.3758/BF03196256. 2.1, 2.4

[81] Jessamyn Schertz and Emily J. Clare. Phonetic cue weighting in perception and production. *WIREs Cognitive Science*, 11(2):e1521, 2020. ISSN 1939-5086. https://doi.org/10.1002/wcs.1521. 2.2, 2.4

[82] John J Ohala, Arthur Bronstein, J, M. Grazia Busà, Julie A Lewis, and William F Weigel, editors. *A Guide to the History of the Phonetic Sciences in the United States*. 1999. 2.2

[83] Alvin M. Liberman and Ignatius G. Mattingly. The motor theory of speech perception revised. *Cognition*, 21(1):1–36, 1985. ISSN 00100277. https://doi.org/10.1016/0010-0277(85)90021-6. 2.2

[84] Haskins Laboratories. https://web.archive.org/web/20200809223413/http://www.haskins.yale.edu/featured/sws/sws.html, August 2020. 2.2

[85] Patricia K. Kuhl. A new view of language acquisition. *Proceedings of the National Academy of Sciences*, 97(22):11850–11857, October 2000. ISSN 0027-8424, 1091-6490. https://doi.org/10.1073/pnas.97.22.11850. 2.2

[86] Patricia K. Kuhl. Early language acquisition: Cracking the speech code. *Nature Reviews Neuroscience*, 5(11):831–843, November 2004. ISSN 1471-0048. https://doi.org/10.1038/nrn1533. 2.2, 2.4

[87] Robert E. Remez, Philip E. Rubin, Stefanie M. Berns, Jennifer S. Pardo, and Jessica M. Lang. On the perceptual organization of speech. *Psychological Review*, 101(1):129–156, January 1994. ISSN 0033-295X. https://doi.org/10.1037/0033-295X.101.1.129. 2.2

[88] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. 2.3

[89] G.N. Clements. Feature Organization. In *Encyclopedia of Language & Linguistics*, pages 433–440. Elsevier, 2006. ISBN 978-0-08-044854-1. https://doi.org/10.1016/B0-08-044854-2/00055-9. 2.4

[90] Morris Halle, Bert Vaux, and Andrew Wolfe. On Feature Spreading and the Representation of Place of Articulation. *Linguistic Inquiry*, 31(3):387–444, July 2000. ISSN 0024-3892. https://doi.org/10.1162/002438900554398. 2.4

[91] Pavel Iosad. Vowel reduction in Russian: No phonetics in phonology. 2012. https://doi.org/10.1017/S0022226712000102. 2.1, 2.4

[92] Danielle J. Navarro. Between the Devil and the Deep Blue Sea: Tensions Between Scientific Judgement and Statistical Model Selection. *Computational Brain & Behavior*, 2(1):28–34, March 2019. ISSN 2522-087X. https://doi.org/10.1007/s42113-018-0019-z. 2.4

[93] Mona Lindau. The story of /r/. *The Journal of the Acoustical Society of America*, 67(S1):S27–S27, April 1980. ISSN 0001-4966. https://doi.org/10.1121/1.2018134. 2.4

[94] A. M. Liberman. Some characteristics of perception in the speech mode. *Research Publications - Association for Research in Nervous and Mental Disease*, 48:238–254, 1970. ISSN 0091-7443. 2.4

[95] Keith R. Kluender, Christian E. Stilp, and Fernando Llanos Lucas. Long-standing problems in speech perception dissolve within an information-theoretic perspective. *Attention, Perception, & Psychophysics*, 81(4):861–883, May 2019. ISSN 1943-393X. https://doi.org/10.3758/s13414-019-01702-x. 2.4

[96] Keith R. Kluender, Christian E. Stilp, and Michael Kiefte. Perception of Vowel Sounds Within a Biologically Realistic Model of Efficient Coding. In Geoffrey Stewart Morrison and Peter F. Assmann, editors, *Vowel Inherent Spectral Change*, pages 117–151. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-14208-6 978-3-642-14209-3. https://doi.org/10.1007/978-3-642-14209-3_6. 2.4

[97] Christian E. Stilp and Keith R. Kluender. Efficient Coding and Statistically Optimal Weighting of Covariance among Acoustic Attributes in Novel Sounds. *PLoS ONE*, 7(1):e30845, January 2012. ISSN 1932-6203. https://doi.org/10.1371/journal.pone.0030845. 2.4, 2.6

[98] C. E. Stilp, T. T. Rogers, and K. R. Kluender. Rapid efficient coding of correlated complex acoustic properties. *Proceedings of the National Academy of Sciences*, 107(50):21914–21919, December 2010. ISSN 0027-8424, 1091-6490. https://doi.org/10.1073/pnas.1009020107. 2.4, 2.5

[99] Evan C. Smith and Michael S. Lewicki. Efficient auditory coding. *Nature*, 439(7079):978–982, February 2006. ISSN 1476-4687. https://doi.org/10.1038/nature04485. 2.4, 2.5

[100] Maria N Geffen, Judit Gervain, Janet F Werker, and Marcelo O Magnasco. Auditory perception of self-similarity in water sounds. *Front Integr Neurosci*, 5(May):15, 2011. ISSN 1662-5145. https://doi.org/10.3389/fnint.2011.00015. 2.4

[101] Michael Kiefte and Keith R. Kluender. Absorption of reliable spectral characteristics in auditory perception. *The Journal of the Acoustical Society of America*, 123(1):366–376, January 2008. ISSN 0001-4966. https://doi.org/10.1121/1.2804951. 2.4

[102] Shi Tong Liu, Pilar Montes-Lourido, Xiaoqin Wang, and Srivatsun Sadagopan. Optimal features for auditory categorization. *Nature Communications*, 10(1):1302, March 2019. ISSN 2041-1723. https://doi.org/10.1038/s41467-019-09115-y. 2.4

[103] Paul Iverson and Patricia K. Kuhl. Influences of phonetic identification and category goodness on American listeners' perception of /r/ and /l/. *The Journal of the Acoustical Society of America*, 99(2):1130–1140, February 1996. ISSN 0001-4966. https://doi.org/10.1121/1.415234. 2.4

[104] Pamela Souza, Richard Wright, Frederick Gallun, and Paul Reinhart. Reliability and Repeatability of the Speech Cue Profile. *Journal of speech, language, and hearing research: JSLHR*, 61(8):2126–2137, August 2018. ISSN 1558-9102. https://doi.org/10.1044/2018_JSLHR-H-17-0341. 2.4

[105] Dave F. Kleinschmidt and T. Florian Jaeger. Robust speech perception: Recognize the familiar, generalize to the similar, and adapt to the novel. *Psychological Review*, 122(2):148–203, April 2015. ISSN 1939-1471, 0033-295X. https://doi.org/10.1037/a0038695. 2.4

[106] R. E. Remez, P. E. Rubin, D. B. Pisoni, and T. D. Carrell. Speech perception without traditional speech cues. *Science*, 212(4497):947–949, May 1981. ISSN 0036-8075, 1095-9203. https://doi.org/10.1126/science.7233191. 2.4

[107] Matthew H. Davis, Ingrid S. Johnsrude, Alexis Hervais-Adelman, Karen Taylor, and Carolyn McGettigan. Lexical information drives perceptual learning of distorted speech: Evidence from the comprehension of noise-vocoded sentences. *Journal of Experimental Psychology. General*, 134(2):222–241, May 2005. ISSN 0096-3445. https://doi.org/10.1037/0096-3445.134.2.222. 2.4

[108] Taffeta M. Elliott and Frédéric E. Theunissen. The Modulation Transfer Function for Speech Intelligibility. *PLOS Computational Biology*, 5(3):e1000302, March 2009. ISSN 1553-7358. https://doi.org/10.1371/journal.pcbi.1000302. 2.4

[109] Justin J. Couchman, Mariana V. C. Coutinho, and J. David Smith. Rules and Resemblance: Their Changing Balance in the Category Learning of Humans (Homo sapiens) and Monkeys (Macaca mulatta). *Journal of experimental psychology. Animal behavior processes*, 36(2):172–183, April 2010. ISSN 0097-7403. https://doi.org/10.1037/a0016748. 2.2, 2.4

[110] Stephen E. G. Lea and A. J. Wills. Use of multiple dimensions in learned discriminations. *Comparative Cognition & Behavior Reviews*, 3, 2008. ISSN 19114745. https://doi.org/10.3819/ccbr.2008.30007. 2.4

[111] L. L. Holt, A. J. Lotto, and K. R. Kluender. Neighboring spectral content influences vowel identification. *The Journal of the Acoustical Society of America*, 108(2):710–722, August 2000. ISSN 0001-4966. https://doi.org/10.1121/1.429604. 2.4

[112] Lori L. Holt. The mean matters: Effects of statistically defined nonspeech spectral distributions on speech categorization. *The Journal of the Acoustical Society of America*, 120(5 Pt 1):2801–2817, November 2006. ISSN 0001-4966. https://doi.org/10.1121/1.2354071. 2.4

[113] Lori L. Holt. Temporally nonadjacent nonlinguistic sounds affect speech categorization. *Psychological Science*, 16(4):305–312, April 2005. ISSN 0956-7976. https://doi.org/10.1111/j.0956-7976.2005.01532.x. 2.4

[114] Patricia K Kuhl, Barbara T Conboy, Sharon Coffey-Corina, Denise Padden, Maritza Rivera-Gaxiola, and Tobey Nelson. Phonetic learning as a pathway to language: New data and native language magnet theory expanded (NLM-e). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1493):979–1000, March 2008. https://doi.org/10.1098/rstb.2007.2154. 2.4

[115] Iris van Rooij and Giosuè Baggio. Theory before the test: How to build high-verisimilitude explanatory theories in psychological science, February 2020. 2.5

[116] Patricia K. Kuhl. Brain Mechanisms in Early Language Acquisition. *Neuron*, 67(5):713–727, September 2010. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2010.08.038. 2.5

[117] Andrew J. King, Sundeep Teki, and Ben D. B. Willmore. Recent advances in understanding the auditory cortex. *F1000Research*, 7, 2018. ISSN 2046-1402. https://doi.org/10.12688/f1000research.15580.1. 2.5

[118] Jennifer K. Schiavo and Robert C. Froemke. Capacities and neural mechanisms for auditory statistical learning across species. *Hearing Research*, 376:97–110, May 2019. ISSN 0378-5955. https://doi.org/10.1016/j.heares.2019.02.002. 2.5

[119] H B Barlow. Single Units and Sensation: A Neuron Doctrine for Perceptual Psychology? *Perception*, 1(4):371–394, December 1972. ISSN 0301-0066. https://doi.org/10.1068/p010371. 2.5

[120] Jennifer K. Bizley, Kerry M. M. Walker, Bernard W. Silverman, Andrew J. King, and Jan W. H. Schnupp. Interdependent encoding of pitch, timbre, and spatial location in auditory cortex. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 29(7):2064–2075, February 2009. ISSN 1529-2401. https://doi.org/10.1523/JNEUROSCI.4755-08.2009. 2.5

[121] Kerry M. M. Walker, Jennifer K. Bizley, Andrew J. King, and Jan W. H. Schnupp. Multiplexed and robust representations of sound features in auditory cortex. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 31 (41):14565–14576, October 2011. ISSN 1529-2401. https://doi.org/10.1523/JNEUROSCI.2074-11.2011. 2.5

[122] Craig A. Atencio and Tatyana O. Sharpee. Multidimensional receptive field processing by cat primary auditory cortical neurons. *Neuroscience*, 359:130–141, September 2017. ISSN 1873-7544. https://doi.org/10.1016/j.neuroscience.2017.07.003. 2.5

[123] Tatyana O Sharpee, Craig A Atencio, and Christoph E Schreiner. Hierarchical representations in the auditory cortex. *Current Opinion in Neurobiology*, 21(5):761–767, October 2011. ISSN 0959-4388. https://doi.org/10.1016/j.conb.2011.05.027. 2.5

[124] Matthew V. Macellaio, Bing Liu, Jeffrey M. Beck, and Leslie C. Osborne. Why sensory neurons are tuned to multiple stimulus features. *bioRxiv*, page 2020.12.29.424235, December 2020. https://doi.org/10.1101/2020.12.29.424235. 2.5

[125] C Angeloni and MN Geffen. Contextual modulation of sound processing in the auditory cortex. *Current Opinion in Neurobiology*, 49:8–15, April 2018. ISSN 0959-4388. https://doi.org/10.1016/j.conb.2017.10.012. 2.5

[126] Isabel Dean, Ben L. Robinson, Nicol S. Harper, and David McAlpine. Rapid Neural Adaptation to Sound Level Statistics. *Journal of Neuroscience*, 28(25):6430–6438, June 2008. ISSN 0270-6474, 1529-2401. https://doi.org/10.1523/JNEUROSCI.0470-08.2008. 2.5

[127] Neil C. Rabinowitz, Ben D. B. Willmore, Jan W. H. Schnupp, and Andrew J. King. Contrast gain control in auditory cortex. *Neuron*, 70(6):1178–1191, June 2011. ISSN 1097-4199. https://doi.org/10.1016/j.neuron.2011.04.030. 2.5

[128] Neil C. Rabinowitz, Ben D. B. Willmore, Andrew J. King, and Jan W. H. Schnupp. Constructing noise-invariant representations of sound in the auditory pathway. *PLoS biology*, 11(11):e1001710, November 2013. ISSN 1545-7885. https://doi.org/10.1371/journal.pbio.1001710. 2.5

[129] Nima Mesgarani, Stephen V. David, Jonathan B. Fritz, and Shihab A. Shamma. Mechanisms of noise robust representation of speech in primary auditory cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 111(18):6792–6797, May 2014. ISSN 1091-6490. https://doi.org/10.1073/pnas.1318017111. 2.5

[130] Stephen V. David, Nima Mesgarani, Jonathan B. Fritz, and Shihab A. Shamma. Rapid synaptic depression explains nonlinear modulation of spectro-temporal tuning in primary auditory cortex by natural stimuli. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 29(11):3374–3386, March 2009. ISSN 1529-2401. https://doi.org/10.1523/JNEUROSCI.5249-08.2009. 2.5

[131] Ryan G Natan, John J Briguglio, Laetitia Mwilambwe-Tshilobo, Sara I Jones, Mark Aizenberg, Ethan M Goldberg, and Maria Neimark Geffen. Complementary control of sensory adaptation by two types of cortical interneurons. *eLife*, 4: e09868, October 2015. ISSN 2050-084X. https://doi.org/10.7554/eLife.09868. 2.5

[132] Ryan G. Natan, Winnie Rao, and Maria N. Geffen. Cortical Interneurons Differentially Shape Frequency Tuning following Adaptation. *Cell Reports*, 21(4):878–890, October 2017. ISSN 2211-1247. https://doi.org/10.1016/j.celrep.2017.10.012. 2.5

[133] Jonathan Fritz, Shihab Shamma, Mounya Elhilali, and David Klein. Rapid task-related plasticity of spectrotemporal receptive fields in primary auditory cortex. *Nature Neuroscience*, 6(11):1216–1223, November 2003. ISSN 1097-6256. https://doi.org/10.1038/nn1141. 2.5

[134] Jonathan Fritz, Mounya Elhilali, and Shihab Shamma. Active listening: Task-dependent plasticity of spectrotemporal receptive fields in primary auditory cortex. *Hearing Research*, 206(1-2):159–176, August 2005. ISSN 0378-5955. https://doi.org/10.1016/j.heares.2005.01.015. 2.5

[135] Stephen V. David, Jonathan B. Fritz, and Shihab A. Shamma. Task reward structure shapes rapid receptive field plasticity in auditory cortex. *Proceedings of the National Academy of Sciences of the United States of America*, 109(6):2144–2149, February 2012. ISSN 1091-6490. https://doi.org/10.1073/pnas.1117717109. 2.5

[136] Stephen V. David and Shihab A. Shamma. Integration over multiple timescales in primary auditory cortex. *The Journal of Neuroscience: The Official Journal of the Society for Neuroscience*, 33(49):19154–19166, December 2013. ISSN 1529-2401. https://doi.org/10.1523/JNEUROSCI.2270-13.2013. 2.5

[137] D. B. Polley. Perceptual Learning Directs Auditory Cortical Map Reorganization through Top-Down Influences. *Journal of Neuroscience*, 26(18):4970–4982, 2006. ISSN 0270-6474. https://doi.org/10.1523/JNEUROSCI.3771-05.2006. 2.5

[138] Kasia M. Bieszczad and Norman M. Weinberger. Representational gain in cortical area underlies increase of memory strength. *Proceedings of the National Academy of Sciences of the United States of America*, 107(8):3793–3798, February 2010. ISSN 1091-6490. https://doi.org/10.1073/pnas.1000159107. 2.5

[139] Han Gyol Yi, Matthew K. Leonard, and Edward F. Chang. The Encoding of Speech Sounds in the Superior Temporal Gyrus. *Neuron*, 102(6):1096–1110, June 2019. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2019.04.023. 2.5

[140] N. Mesgarani, C. Cheung, K. Johnson, and E. F. Chang. Phonetic Feature Encoding in Human Superior Temporal Gyrus. *Science*, 343(6174):1006–1010, February 2014. ISSN 0036-8075, 1095-9203. https://doi.org/10.1126/science.1245994. 2.5

[141] Edward F. Chang, Jochem W. Rieger, Keith Johnson, Mitchel S. Berger, Nicholas M. Barbaro, and Robert T. Knight. Categorical speech representation in human superior temporal gyrus. *Nature Neuroscience*, 13(11):1428–1432, November 2010. ISSN 1546-1726. https://doi.org/10.1038/nn.2641. 2.5

[142] Alexander M. Chan, Andrew R. Dykstra, Vinay Jayaram, Matthew K. Leonard, Katherine E. Travis, Brian Gygi, Janet M. Baker, Emad Eskandar, Leigh R. Hochberg, Eric Halgren, and Sydney S. Cash. Speech-Specific Tuning of Neurons in Human Superior Temporal Gyrus. *Cerebral Cortex*, 24(10):2679–2693, October 2014. ISSN 1047-3211. https://doi.org/10.1093/cercor/bht127. 2.5

[143] Liberty S. Hamilton, Erik Edwards, and Edward F. Chang. A Spatial Map of Onset and Sustained Responses to Speech in the Human Superior Temporal Gyrus. *Current Biology*, 28(12):1860–1871.e4, June 2018. ISSN 0960-9822. https://doi.org/10.1016/j.cub.2018.04.033. 2.5, 2.6

[144] Crystal T. Engineer, Kimiya C. Rahebi, Elizabeth P. Buell, Melyssa K. Fink, and Michael P. Kilgard. Speech training alters consonant and vowel responses in multiple auditory cortex fields. *Behavioural Brain Research*, 287:256–264, July 2015. ISSN 0166-4328. https://doi.org/10.1016/j.bbr.2015.03.044. 2.5

[145] H. Takahashi, R. Yokota, A. Funamizu, H. Kose, and R. Kanzaki. Learning-stage-dependent, field-specific, map plasticity in the rat auditory cortex during appetitive operant conditioning. *Neuroscience*, 199:243–258, December 2011. ISSN 0306-4522. https://doi.org/10.1016/j.neuroscience.2011.09.046. 2.5

[146] Zhiyue Shi, Sumei Yan, Yu Ding, Chang Zhou, Shaowen Qian, Zhaoqun Wang, Chen Gong, Meng Zhang, Yanjie Zhang, Yandong Zhao, Huizhong Wen, Penghui Chen, Qiyue Deng, Tiantian Luo, Ying Xiong, and Yi Zhou. Anterior Auditory Field Is Needed for Sound Categorization in Fear Conditioning Task of Adult Rat. *Frontiers in Neuroscience*, 13, 2019. ISSN 1662-453X. https://doi.org/10.3389/fnins.2019.01374. 2.5

[147] Andres Carrasco and Stephen G. Lomber. Neuronal activation times to simple, complex, and natural sounds in cat primary and nonprimary auditory cortex. *Journal of Neurophysiology*, 106(3):1166–1178, June 2011. ISSN 0022-3077. https://doi.org/10.1152/jn.00940.2010. 2.5

[148] Jennifer F. Linden, Robert C. Liu, Maneesh Sahani, Christoph E. Schreiner, and Michael M. Merzenich. Spectrotemporal Structure of Receptive Fields in Areas AI and AAF of Mouse Auditory Cortex. *Journal of Neurophysiology*, 90(4): 2660–2675, October 2003. ISSN 0022-3077. https://doi.org/10.1152/jn.00751.2002. 2.5

[149] Pritesh K. Pandya, Daniel L. Rathbun, Raluca Moucha, Navzer D. Engineer, and Michael P. Kilgard. Spectral and Temporal Processing in Rat Posterior Auditory Cortex. *Cerebral Cortex*, 18(2):301–314, February 2008. ISSN 1047-3211. https://doi.org/10.1093/cercor/bhm055. 2.5

[150] Andres Carrasco and Stephen G. Lomber. Evidence for Hierarchical Processing in Cat Auditory Cortex: Nonreciprocal Influence of Primary Auditory Cortex on the Posterior Auditory Field. *Journal of Neuroscience*, 29(45):14323–14333, November 2009. ISSN 0270-6474, 1529-2401. https://doi.org/10.1523/JNEUROSCI.2905-09.2009. 2.5

[151] Xiaoqin Wang. Neural coding strategies in auditory cortex. *Hearing Research*, 229(1):81–93, July 2007. ISSN 0378-5955. https://doi.org/10.1016/j.heares.2007.01.019. 2.6

[152] Jacob Reimer, Matthew J. McGinley, Yang Liu, Charles Rodenkirch, Qi Wang, David A. McCormick, and Andreas S. Tolias. Pupil fluctuations track rapid changes in adrenergic and cholinergic activity in cortex. *Nature Communications*, 7:13289, November 2016. ISSN 2041-1723. https://doi.org/10.1038/ncomms13289. 3

[153] Ana Parabucki, Alexander Bizer, Genela Morris, Antonio E. Munoz, Avinash D. S. Bala, Matthew Smear, and Roman Shusterman. Odor Concentration Change Coding in the Olfactory Bulb. *eNeuro*, 6(1), February 2019. ISSN 2373-2822. https://doi.org/10.1523/ENEURO.0396-18.2019. 3

[154] Cristopher M. Niell and Michael P. Stryker. Modulation of Visual Responses by Behavioral State in Mouse Visual Cortex. *Neuron*, 65(4):472–479, February 2010. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2010.01.033. 3

[155] Tanmay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols*, 14(7):2152–2176, July 2019. ISSN 1750-2799. https://doi.org/10.1038/s41596-019-0176-0. 3

[156] James J. Jun, Nicholas A. Steinmetz, Joshua H. Siegle, Daniel J. Denman, Marius Bauza, Brian Barbarits, Albert K. Lee, Costas A. Anastassiou, Alexandru Andrei, Çağatay Aydın, Mladen Barbic, Timothy J. Blanche, Vincent Bonin, João Couto, Barundeb Dutta, Sergey L. Gratiy, Diego A. Gutnisky, Michael Häusser, Bill Karsh, Peter Ledochowitsch, Carolina Mora Lopez, Catalin Mitelut, Silke Musa, Michael Okun, Marius Pachitariu, Jan Putzeys, P. Dylan Rich, Cyrille Rossant, Wei-Lung Sun, Karel Svoboda, Matteo Carandini, Kenneth D. Harris, Christof Koch, John O'Keefe, and Timothy D. Harris. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, November 2017. ISSN 1476-4687. https://doi.org/10.1038/nature24636. 3, 4.8, 11

[157] Christopher P. Burgess, Armin Lak, Nicholas A. Steinmetz, Peter Zatka-Haas, Charu Bai Reddy, Elina A. K. Jacobs, Jennifer F. Linden, Joseph J. Paton, Adam Ranson, Sylvia Schröder, Sofia Soares, Miles J. Wells, Lauren E. Wool, Kenneth D. Harris, and Matteo Carandini. High-Yield Methods for Accurate Two-Alternative Visual Psychophysics in Head-Fixed Mice. *Cell Reports*, 20(10):2513–2524, September 2017. ISSN 2211-1247. https://doi.org/10.1016/j.celrep.2017.08.047. 3

[158] Kay Thurley and Aslı Ayaz. Virtual reality systems for rodents. *Current Zoology*, 63(1):109–119, February 2017. ISSN 1674-5507. https://doi.org/10.1093/cz/zow070. 3

[159] Anna R. Chambers, Kenneth E. Hancock, Kamal Sen, and Daniel B. Polley. Online stimulus optimization rapidly reveals multidimensional selectivity in auditory cortical neurons. *The Journal of Neuroscience*, 34(27):8963–8975, July 2014. ISSN 1529-2401. https://doi.org/10.1523/JNEUROSCI.0260-14.2014. 3

[160] Chance Elliott, Vipin Vijayakumar, Wesley Zink, and Richard Hansen. National Instruments LabVIEW: A Programming Environment for Laboratory Automation and Measurement. *JALA: Journal of the Association for Laboratory Automation*, 12(1):17–24, February 2007. ISSN 1535-5535. https://doi.org/10.1016/j.jala.2006.07.012. 3

[161] Open Ephys. pyControl. http://www.open-ephys.org/store/pycontrol, February 2019. 3

[162] Josh Sanders. Sanworks - BPod. https://www.sanworks.io/shop/products.php?productFamily=bpod. 3

[163] Matthew B. Wall. Reliability starts with the experimental tools employed. *Cortex*, 113:352–354, April 2019. ISSN 0010-9452. https://doi.org/10.1016/j.cortex.2018.11.034. 3, 4.3.1

[164] Peter Johnson-Lenz and Trudy Johnson-Lenz. Post-mechanistic groupware primitives: Rhythms, boundaries and containers. *International Journal of Man-Machine Studies*, 34(3):395–417, February 1991. ISSN 0020-7373. https://doi.org/10.1016/0020-7373(91)90027-5. 3, 1

[165] Peter Johnson-Lenz and Trudy Johnson-Lenz. Groupware: Coining and defining it. *ACM SIGGROUP Bulletin*, 19(2): 34, August 1998. ISSN 2372-7403, 2372-739X. https://doi.org/10.1145/290575.290585. 3

[166] STEPHEN R. BARLEY and BETH A. BECHKY. In the Backrooms of Science: The Work of Technicians in Science Labs. *Work and Occupations*, 21(1):85–126, February 1994. ISSN 0730-8884. https://doi.org/10.1177/073088849402100104. 3, 10.1

[167] Thomas Akam, Andy Lustig, James M Rowland, Sampath KT Kapanaiah, Joan Esteve-Agraz, Mariangela Panniello, Cristina Márquez, Michael M Kohl, Dennis Kätzel, Rui M Costa, and Mark E Walton. Open-source, Python-based, hardware and software for controlling behavioural neuroscience experiments. *eLife*, 11:e67846, January 2022. ISSN 2050-084X. https://doi.org/10.7554/eLife.67846. 3.1.1

[168] Gonçalo Lopes, Niccolò Bonacchi, João Frazão, Joana P. Neto, Bassam V. Atallah, Sofia Soares, Luís Moreira, Sara Matias, Pavel M. Itskov, Patrícia A. Correia, Roberto E. Medina, Lorenza Calcaterra, Elena Dreosti, Joseph J. Paton, and Adam R. Kampff. Bonsai: An event-based framework for processing and controlling data streams. *Frontiers in Neuroinformatics*, 9, 2015. ISSN 1662-5196. 5, 7.3

[169] Florian Krause and Oliver Lindemann. Expyriment: A Python library for cognitive and neuroscientific experiments. *Behavior Research Methods*, 46(2):416–428, June 2014. ISSN 1554-3528. https://doi.org/10.3758/s13428-013-0390-6. 5

[170] Jonathan Peirce, Jeremy R. Gray, Sol Simpson, Michael MacAskill, Richard Höchenberger, Hiroyuki Sogo, and Erik Kastman. PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, 51(1):195–203, February 2019. ISSN 1554-3528. https://doi.org/10.3758/s13428-018-01193-y. 5, 5.6

[171] Sebastiaan Mathôt, Daniel Schreij, and Jan Theeuwes. OpenSesame: An open-source, graphical experiment builder for the social sciences. *Behavior Research Methods*, 44(2):314–324, June 2012. ISSN 1554-3528. https://doi.org/10.3758/s13428-011-0168-7. 5

[172] Xinfeng Chen and Haohong Li. ArControl: An Arduino-Based Comprehensive Behavioral Platform with Real-Time Performance. *Frontiers in Behavioral Neuroscience*, 11, 2017. ISSN 1662-5153. https://doi.org/10.3389/fnbeh.2017.00244. 5

[173] S. van der Walt, S. C. Colbert, and G. Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering*, 13(2):22–30, March 2011. ISSN 1521-9615. https://doi.org/10.1109/MCSE.2011.37. 6, 6.1

[174] Eric Jones, Travis Oliphant, and Pearu Peterson. SciPy: Open Source Scientific Tools for Python. http://www.scipy.org/, 2001. 6

[175] Sandeep Robert Datta, David J. Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational Neuroethology: A Call to Action. *Neuron*, 104(1):11–24, October 2019. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2019.09.038. 3.2, 5.7.1

[176] Sanworks, LLC. 8 reasons to use Bpod's new HiFi module. https://sanworks.io/news/viewArticle.php?articleID=HiFi01, July 2021. 3.2

[177] The International Brain Laboratory, Valeria Aguillon-Rodriguez, Dora E. Angelaki, Hannah M. Bayer, Niccolò Bonacchi, Matteo Carandini, Fanny Cazettes, Gaelle A. Chapuis, Anne K. Churchland, Yang Dan, Eric E. J. Dewitt, Mayo Faulkner, Hamish Forrest, Laura M. Haetzel, Michael Hausser, Sonja B. Hofer, Fei Hu, Anup Khanal, Christopher S. Krasniak, Inês Laranjeira, Zachary F. Mainen, Guido T. Meijer, Nathaniel J. Miska, Thomas D. Mrsic-Flogel, Masayoshi Murakami, Jean-Paul Noel, Alejandro Pan-Vazquez, Cyrille Rossant, Joshua I. Sanders, Karolina Z. Socha, Rebecca Terry, Anne E. Urai, Hernando M. Vergara, Miles J. Wells, Christian J. Wilson, Ilana B. Witten, Lauren E. Wool, and Anthony Zador. Standardized and reproducible measurement of decision-making in mice, October 2020. 3.2

[178] Tim Anderson. Guido van Rossum aiming to make CPython 2x faster in 3.11. https://www.theregister.com/2021/05/13/guido_van_rossum_cpython_3_11/, May 2021. 2

[179] Guido van Rossum. Glue It All Together With Python. https://www.python.org/doc/essays/omg-darpa-mcc-position/, January 1998. 4.1.2

[180] Gary A Kane, Gonçalo Lopes, Jonny L Saunders, Alexander Mathis, and Mackenzie W Mathis. Real-time, low-latency closed-loop feedback using markerless posture tracking. *eLife*, 9:e61909, December 2020. ISSN 2050-084X. https://doi.org/10.7554/eLife.61909. 4.2.1, 5.5, 5.5, 5.6, 15

[181] Pieter Hintjens. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, Beijing, 1 edition edition, March 2013. ISBN 978-1-4493-3406-2. 4.2.4, 9

[182] Oliver Rübel, Andrew Tritt, Benjamin Dichter, Thomas Braun, Nicholas Cain, Nathan Clack, Thomas J. Davidson, Max Dougherty, Jean-Christophe Fillion-Robin, Nile Graddis, Michael Grauer, Justin T. Kiggins, Lawrence Niu, Doruk Ozturk, William Schroeder, Ivan Soltesz, Friedrich T. Sommer, Karel Svoboda, Ng Lydia, Loren M. Frank, and Kristofer Bouchard. NWB:N 2.0: An Accessible Data Standard for Neurophysiology. *bioRxiv*, January 2019. https://doi.org/10.1101/523035. 4.3.2, 7.4

[183] David A. W. Soergel. Rampant software errors may undermine scientific results. *F1000Research*, 3, July 2015. ISSN 2046-1402. https://doi.org/10.12688/f1000research.5930.2. 4.3.3, 11.3.1

[184] Anders Eklund, Thomas E. Nichols, and Hans Knutsson. Cluster failure: Why fMRI inferences for spatial extent have inflated false-positive rates. *Proceedings of the National Academy of Sciences*, 113(28):7900–7905, July 2016. ISSN 0027-8424, 1091-6490. https://doi.org/10.1073/pnas.1602413113. 4.3.3, 11.3.1

[185] Jayanti Bhandari Neupane, Ram P. Neupane, Yuheng Luo, Wesley Y. Yoshida, Rui Sun, and Philip G. Williams. Characterization of Leptazolines A–D, Polar Oxazolines from the Cyanobacterium Leptolyngbya sp., Reveals a Glitch with the "Willoughby–Hoye" Scripts for Calculating NMR Chemical Shifts. *Organic Letters*, 21(20):8449–8453, October 2019. ISSN 1523-7060. https://doi.org/10.1021/acs.orglett.9b03216. 4.3.3, 11.3.1

[186] Greg Miller. A Scientist's Nightmare: Software Problem Leads to Five Retractions. *Science*, 314(5807):1856–1857, December 2006. ISSN 0036-8075, 1095-9203. https://doi.org/10.1126/science.314.5807.1856. 4.3.3, 11.3.1

[187] Yarden Katz and Ulrich Bernhard Matter. On the Biomedical Elite: Inequality and Stasis in Scientific Knowledge Production. *Berkman Klein Center for Internet & Society Research*, July 2017. 4.3.4

[188] Jeremy Ashkenas, Haeyoun Park, and Adam Pearce. Even With Affirmative Action, Blacks and Hispanics Are More Underrepresented at Top Colleges Than 35 Years Ago. *The New York Times*, August 2017. ISSN 0362-4331. 4.3.4

[189] Aaron Clauset, Samuel Arbesman, and Daniel B. Larremore. Systematic inequality and hierarchy in faculty hiring networks. *Science Advances*, 1(1):e1400005, February 2015. ISSN 2375-2548. https://doi.org/10.1126/sciadv.1400005. 4.3.4

[190] J. M. Pearce, J. C. Molloy, S. Kuznetsov, and S. Dosemagen. Expanding Equitable Access to Experimental Research and STEM Education by Supporting Open Source Hardware Development, January 2019. 4.3.4

[191] Emmeke Aarts, Matthijs Verhage, Jesse V. Veenvliet, Conor V. Dolan, and Sophie van der Sluis. A solution to dependency: Using multilevel analysis to accommodate nested data. *Nature Neuroscience*, 17(4):491–496, April 2014. ISSN 1546-1726. https://doi.org/10.1038/nn.3648. 4.8, 4.3.4

[192] Patrick E. Shrout and Joseph L. Rodgers. Psychology, Science, and Knowledge Construction: Broadening Perspectives from the Replication Crisis. *Annual Review of Psychology*, 69(1):487–510, 2018. https://doi.org/10.1146/annurev-psych-122216-011845. 4.3.4

[193] Katherine S. Button, John P. A. Ioannidis, Claire Mokrysz, Brian A. Nosek, Jonathan Flint, Emma S. J. Robinson, and Marcus R. Munafò. Power failure: Why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, 14(5):365–376, May 2013. ISSN 1471-0048. https://doi.org/10.1038/nrn3475. 4.3.4

[194] Wes McKinney. Pandas: A foundational Python library for data analysis and statistics. *Python for high performance and scientific computing*, 14(9):1–9, 2011. 5.2, 6.1

[195] Dexter C. Kozen. Limitations of Finite Automata. In Dexter C. Kozen, editor, *Automata and Computability*, Undergraduate Texts in Computer Science, pages 67–71. Springer New York, New York, NY, 1997. ISBN 978-1-4612-1844-9. https://doi.org/10.1007/978-1-4612-1844-9_12. 5.3.3

[196] Ji Hyun Bak, Jung Yoon Choi, Athena Akrami, Ilana Witten, and Jonathan W Pillow. Adaptive optimal training of animal behavior. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1947–1955. Curran Associates, Inc., 2016. 5.3.3

[197] Aaron Swartz. Aaron Swartz's A Programmable Web: An Unfinished Work. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 3(2):1–64, February 2013. ISSN 2160-4711, 2160-472X. https://doi.org/10.2200/S00481ED1V01Y201302WBE005. 4, 10.2.4

[198] Fatemeh Abyarjoo, Armando Barreto, Jonathan Cofino, and Francisco R. Ortega. Implementing a Sensor Fusion Algorithm for 3D Orientation Detection with Inertial/Magnetic Sensors. *Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering*, pages 305–310, 2015. https://doi.org/10.1007/978-3-319-06773-5_41. 5.5

[199] Photis Patonis, Petros Patias, Ilias N. Tziavos, Dimitrios Rossikopoulos, and Konstantinos G. Margaritis. A Fusion Method for Combining Low-Cost IMU/Magnetometer Outputs for Use in Applications on Mobile Devices. *Sensors (Basel, Switzerland)*, 18(8), August 2018. ISSN 1424-8220. https://doi.org/10.3390/s18082616. 5.5

[200] Leland Wilkinson. The Grammar of Graphics. In James E. Gentle, Wolfgang Karl Härdle, and Yuichi Mori, editors, *Handbook of Computational Statistics: Concepts and Methods*, Springer Handbooks of Computational Statistics, pages 375–414. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-21551-3. 5.9.1

[201] Hadley Wickham. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, June 2016. ISBN 978-3-319-24277-4. 6.1

[202] Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. Dplyr: A Grammar of Data Manipulation, 2022. 6.1

[203] Lionel Henry and Hadley Wickham. Purrr: Functional Programming Tools, 2020. 6.1

[204] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. ISSN 1476-4687. https://doi.org/10.1038/s41586-020-2649-2. 6.1

[205] Eduardo Soares, Pedro Brandão, and Rui Prior. Analysis of Timekeeping in Experimentation. In *2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP)*, pages 1–6, July 2020. https://doi.org/10.1109/CSNDSP49049.2020.9249632. 6.3, 3

[206] Joshua H. Siegle, Aarón Cuevas López, Yogi A. Patel, Kirill Abramov, Shay Ohayon, and Jakob Voigts. Open Ephys: An open-source, plugin-based platform for multichannel electrophysiology. *Journal of Neural Engineering*, 14(4):045003, June 2017. ISSN 1741-2552. https://doi.org/10.1088/1741-2552/aa5eea. 7.3, 12.1.1

[207] Daniel Aharoni and Tycho M. Hoogland. Circuit Investigations With Open-Source Miniaturized Microscopes: Past, Present and Future. *Frontiers in Cellular Neuroscience*, 13, 2019. ISSN 1662-5102. 7.3

[208] Daniel Aharoni, Baljit S. Khakh, Alcino J. Silva, and Peyman Golshani. All the light that we can see: A new era in miniaturized microscopy. *Nature Methods*, 16(1):11–13, January 2019. ISSN 1548-7105. https://doi.org/10.1038/s41592-018-0266-x. 7.3, 12.1.1

[209] Dimitri Yatsenko, Edgar Y. Walker, and Andreas S. Tolias. DataJoint: A Simpler Relational Data Model, July 2018. 7.4

[210] Geoffrey C. Bowker, Karen Baker, Florence Millerand, and David Ribes. Toward Information Infrastructure Studies: Ways of Knowing in a Networked Environment. In Jeremy Hunsinger, Lisbeth Klastrup, and Matthew Allen, editors, *International Handbook of Internet Research*, pages 97–117. Springer Netherlands, Dordrecht, 2010. ISBN 978-1-4020-9789-8. https://doi.org/10.1007/978-1-4020-9789-8_5. III, 10.1

[211] Susan Leigh Star and Karen Ruhleder. Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces. *Information Systems Research*, 7(1):111–134, March 1996. ISSN 1047-7047. https://doi.org/10.1287/isre.7.1.111. 9

[212] Michican Civil Rights Commission. The Flint Water Crisis: Systemic Racism Through the Lens of Flint. Technical report, Michican Civil Rights Commission, February 2017. 9

[213] Philip Mirowski. The future(s) of open science. *Social Studies of Science*, 48(2):171–203, April 2018. ISSN 0306-3127. https://doi.org/10.1177/0306312718772086. 9, 10.2.6

[214] Jefferson Pooley. Surveillance Publishing. Article; https://web.archive.org/web/20211123130445/https://osf.io/preprints/socarxiv/j6ung/ SocArXiv, November 2021. 9, 10.2.6, 12.4.1

[215] Shoshana Zuboff. Big other: Surveillance capitalism and the prospects of an information civilization. *Journal of Information Technology*, 30(1):75–89, March 2015. ISSN 1466-4437. https://doi.org/10.1057/jit.2015.5. 9

[216] McKenzie Wark. *Capital Is Dead: Is This Something Worse?* Verso Books, February 2021. ISBN 978-1-78873-533-9. 9, 11.2.5

[217] Richard J. Roberts, Harold E. Varmus, Michael Ashburner, Patrick O. Brown, Michael B. Eisen, Chaitan Khosla, Marc Kirschner, Roel Nusse, Matthew Scott, and Barbara Wold. Building A "GenBank" of the Published Literature. *Science*, 291(5512):2318–2319, March 2001. https://doi.org/10.1126/science.1060273. 9, 11.1.1

[218] Harold Varmus. *The Art and Politics of Science*. W.W. Norton & Company, New York, 2009. ISBN 978-0-393-06128-4. 9, 11.1.1

[219] Rob Kling, Lisa B. Spector, and Joanna Fortuna. The real stakes of virtual publishing: The transformation of E-Biomed into PubMed central. *Journal of the American Society for Information Science and Technology*, 55(2):127–148, 2004. ISSN 1532-2890. https://doi.org/10.1002/asi.10352. 9

[220] Barry P. Markovitz. Biomedicine's Electronic Publishing Paradigm Shift. *Journal of the American Medical Informatics Association : JAMIA*, 7(3):222–229, 2000. ISSN 1067-5027. 9

[221] Lorenzo Franceschi-Bicchierai. Academic Journal Claims it Fingerprints PDFs for 'Ransomware,' Not Surveillance. *Vice*, January 2022. 9

[222] Björn Brembs. Algorithmic employment decisions in academia?, September 2021. 9, 10.2.6, 12.4.1

[223] Apple. watchOS 9 delivers new ways to stay connected, active, and healthy. https://www.apple.com/newsroom/2022/06/watchos-9-delivers-new-ways-to-stay-connected-active-and-healthy/, June 2022. 9, 11.2.4

[224] Joseph Douress, Peter Previte, Jay Butchko, Bernard Kennedy, Christopher Stagg, Frank Grippo, and Eric Lee. Professional matching service, August 2007. 9

[225] Sam Biddle. ICE Searched LexisNexis Database Over 1 Million Times in Just Seven Months. *The Intercept*, June 2022. 9

[226] Sam Biddle. LexisNexis to Provide Giant Database of Personal Information to ICE, April 2021. 9, 10.2.6

[227] Sarah Lamdan. Defund the Police, and Defund Big Data Policing, Too. https://www.jurist.org/commentary/2020/06/sarah-lamdan-data-policing/, June 2020. 9

[228] Sarah Lamdan. Librarianship at the Crossroads of ICE Surveillance. *In the Library with the Lead Pipe*, November 2019. 9, 11.4.3

[229] Sarah Myers West. Data Capitalism: Redefining the Logics of Surveillance and Privacy. *Business & Society*, 58(1):20–41, January 2019. ISSN 0007-6503. https://doi.org/10.1177/0007650317718185. 9

[230] McKenzie Wark. *A Hacker Manifesto*. Harvard University Press, October 2004. ISBN 978-0-674-01543-2. 9

[231] Tim Berners-Lee. Socially aware cloud storage - Design Issues. https://www.w3.org/DesignIssues/CloudStorage.html, August 2009. 9, 11.2.5

[232] Tim Berners-Lee. Web Services overview - Design Issues. https://www.w3.org/DesignIssues/WebServices.html, August 2009. 9

[233] Björn Brembs, Philippe Huneman, Felix Schönbrodt, Gustav Nilsonne, Toma Susi, Renke Siems, Pandelis Perakakis, Varvara Trachana, Lai Ma, and Sara Rodriguez-Cuadrado. Replacing academic journals. September 2021. https://doi.org/10.5281/zenodo.5526635. 9, 10.2.6, 11.4

[234] Charleeze Ponzi. Is science a pyramid scheme? The correlation between an author's position in the academic hierarchy and her scientific output per year, January 2020. 9

[235] Matthew J. Bietz, Toni Ferro, and Charlotte P. Lee. Sustaining the development of cyberinfrastructure: An organization adapting to change. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 901–910, New York, NY, USA, February 2012. Association for Computing Machinery. ISBN 978-1-4503-1086-4. https://doi.org/10.1145/2145204.2145339. 9

[236] Lauren E. Wool and The International Brain Laboratory. Knowledge across networks: How to build a global neuroscience collaboration. July 2020. https://doi.org/10.1016/j.conb.2020.10.020. 10.1, 10.3.3

[237] James Howison and James D. Herbsleb. Incentives and integration in scientific software production. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 459–470, New York, NY, USA, February 2013. Association for Computing Machinery. ISBN 978-1-4503-1331-5. https://doi.org/10.1145/2441776.2441828. 10.2.1

[238] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281–1289, September 2018. ISSN 1546-1726. https://doi.org/10.1038/s41593-018-0209-y. 10.2.1, 11.3.1

[239] Jeffrey C. Carver, Nic Weber, Karthik Ram, Sandra Gesing, and Daniel S. Katz. A survey of the state of the practice for research software in the United States. *PeerJ Computer Science*, 8:e963, May 2022. ISSN 2376-5992. https://doi.org/10.7717/peerj-cs.963. 10.2.1

[240] Serghei Mangul, Lana S. Martin, Eleazar Eskin, and Ran Blekhman. Improving the usability and archival stability of bioinformatics software. *Genome Biology*, 20(1):47, February 2019. ISSN 1474-760X. https://doi.org/10.1186/s13059-019-1649-8. 10.2.1

[241] Sudhir Kumar and Joel Dudley. Bioinformatics software for biologists in the genomics era. *Bioinformatics*, 23(14):1713–1717, July 2007. ISSN 1367-4803. https://doi.org/10.1093/bioinformatics/btm239. 10.2.1

[242] James Howison and Julia Bullard. Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. *Journal of the Association for Information Science and Technology*, 67(9):2137–2155, 2016. ISSN 2330-1643. https://doi.org/10.1002/asi.23538. 10.2.1

[243] NIH Strategic Plan for Data Science. Technical report, National Institutes of Health, June 2018. 10.2.2, 10.2.6, 11.2.4

[244] David Ribes and Thomas Finholt. The Long Now of Technology Infrastructure: Articulating Tensions in Development. *Journal of the Association for Information Systems*, 10(5):375–398, May 2009. ISSN 15369323. https://doi.org/10.17705/1jais.00199. 10.2.3

[245] Stephen Altschul, Barry Demchak, Richard Durbin, Robert Gentleman, Martin Krzywinski, Heng Li, Anton Nekrutenko, James Robinson, Wayne Rasband, James Taylor, and Cole Trapnell. The anatomy of successful computational biology software. *Nature Biotechnology*, 31(10):894–897, October 2013. ISSN 1546-1696. https://doi.org/10.1038/nbt.2721. 10.2.3

[246] Sean B. Palmer. Ditching the Semantic Web? http://inamidst.com/whits/2008/ditching, March 2008. 10.2.3, 11.2.5

[247] Lindsay Poirier. A Turn for the Scruffy: An Ethnographic Study of Semantic Web Architecture. In *Proceedings of the 2017 ACM on Web Science Conference*, WebSci '17, pages 359–367, New York, NY, USA, June 2017. Association for Computing Machinery. ISBN 978-1-4503-4896-6. https://doi.org/10.1145/3091478.3091505. 10.2.4, 11.2.4

[248] Michael A. Hiltzik. Taming the Wild, Wild Web. https://web.archive.org/web/20010801142640/https://www.latimes.com/business/la-072601netarch.story, July 2001. 10.2.4

[249] Rebekah Larsen. The Political Nature of TCP/IP. page 56, 2012. 10.2.4, 11.1.1

[250] Iain. Freebase is dead, long live Freebase, May 2019. 4

[251] M.D. McIlroy, E.N. Pinson, and B.A. Tague. UNIX Time-Sharing System: Forward. *The Bell System Technical Journal*, 57(6), July 1978. 10.2.5

[252] Elsevier and Seven Bridges receive NIH Data Commons grant for biomedical data analysis. https://www.elsevier.com/about/press-releases/archive/science-and-technology/elsevier-and-seven-bridges-receive-nih-data-commons-grant-for-biomedical-data-analysis, November 2017. 10.2.6

[253] In re: Google Digital Advertising Antitrust Litigation - Amended Complaint #152, October 2021. 8

[254] Ian MacInnes. Compatibility standards and monopoly incentives: The impact of service-based software licensing. *International Journal of Services and Standards*, 1(3):255–270, January 2005. ISSN 1740-8849. https://doi.org/10.1504/IJSS.2005.005799. 10.2.6

[255] RELX Annual Report 2020. page 196, 2020. 10.2.6, 11.4

[256] Criticism of Amazon. *Wikipedia*, September 2021. 10.2.6

[257] Elsevier. Topic Prominence in Science - Scival | Elsevier Solutions. https://www.elsevier.com/solutions/scival/features/topic-prominence-in-science. 10.2.6

[258] Stephen Buranyi. Is the staggeringly profitable business of scientific publishing bad for science? *The Guardian*, June 2017. ISSN 0261-3077. 10.2.6

[259] R. Todd Reilly. NIH STRIDES Initiative, January 2021. 10.2.6

[260] STRIDES Initiative Success Story: University of Michigan TOPMed | Data Science at NIH. https://web.archive.org/web/20210324024612/https://datascience.nih.gov/strides-initiative-success-story-university-michigan-topmed, October 2020. 9

[261] Richard Lawler. An Amazon server outage caused problems for Alexa, Ring, Disney Plus, and deliveries. https://www.theverge.com/2021/12/7/22822332/amazon-server-aws-down-disney-plus-ring-outage, December 2021. 10

[262] Lee Hutchinson. Amazon Web Services outage once again shows reality behind "the cloud". https://arstechnica.com/information-technology/2012/10/amazon-web-services-outage-once-again-shows-reality-behind-the-cloud/, October 2012. 10

[263] Ed S. Lein, Michael J. Hawrylycz, Nancy Ao, Mikael Ayres, Amy Bensinger, Amy Bernard, Andrew F. Boe, Mark S. Boguski, Kevin S. Brockway, Emi J. Byrnes, Lin Chen, Li Chen, Tsuey-Ming Chen, Mei Chi Chin, Jimmy Chong, Brian E. Crook, Aneta Czaplinska, Chinh N. Dang, Suvro Datta, Nick R. Dee, Aimee L. Desaki, Tsega Desta, Ellen Diep, Tim A. Dolbeare, Matthew J. Donelan, Hong-Wei Dong, Jennifer G. Dougherty, Ben J. Duncan, Amanda J. Ebbert, Gregor Eichele, Lili K. Estin, Casey Faber, Benjamin A. Facer, Rick Fields, Shanna R. Fischer, Tim P. Fliss, Cliff Frensley, Sabrina N. Gates, Katie J. Glattfelder, Kevin R. Halverson, Matthew R. Hart, John G. Hohmann, Maureen P. Howell, Darren P. Jeung, Rebecca A. Johnson, Patrick T. Karr, Reena Kawal, Jolene M. Kidney, Rachel H. Knapik, Chihchau L. Kuan, James H. Lake, Annabel R. Laramee, Kirk D. Larsen, Christopher Lau, Tracy A. Lemon, Agnes J. Liang, Ying Liu, Lon T. Luong, Jesse Michaels, Judith J. Morgan, Rebecca J. Morgan, Marty T. Mortrud, Nerick F. Mosqueda, Lydia L. Ng, Randy Ng, Geralyn J. Orta, Caroline C. Overly, Tu H. Pak, Sheana E. Parry, Sayan D. Pathak, Owen C. Pearson, Ralph B. Puchalski, Zackery L. Riley, Hannah R. Rockett, Stephen A. Rowland, Joshua J. Royall, Marcos J. Ruiz, Nadia R. Sarno, Katherine Schaffnit, Nadiya V. Shapovalova, Taz Sivisay, Clifford R. Slaughterbeck, Simon C. Smith, Kimberly A. Smith, Bryan I. Smith, Andy J. Sodt, Nick N. Stewart, Kenda-Ruth Stumpf, Susan M. Sunkin, Madhavi Sutram, Angelene Tam, Carey D. Teemer, Christina Thaller, Carol L. Thompson, Lee R. Varnam, Axel Visel, Ray M. Whitlock, Paul E. Wohnoutka, Crissa K. Wolkey, Victoria Y. Wong, Matthew Wood, Murat B. Yaylaoglu, Rob C. Young, Brian L. Youngstrom, Xu Feng Yuan, Bin Zhang, Theresa A. Zwingman, and Allan R. Jones. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, 445(7124):168–176, January 2007. ISSN 1476-4687. https://doi.org/10.1038/nature05453. 10.3.2

[264] Sten Grillner, Nancy Ip, Christof Koch, Walter Koroshetz, Hideyuki Okano, Miri Polachek, Mu-ming Poo, and Terrence J. Sejnowski. Worldwide initiatives to advance brain research. *Nature Neuroscience*, 19(9):1118–1122, September 2016. ISSN 1546-1726. https://doi.org/10.1038/nn.4371. 10.3.2

[265] Christof Koch and Allan Jones. Big Science, Team Science, and Open Science for Neuroscience. *Neuron*, 92(3):612–616, November 2016. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2016.10.019. 10.3.2

[266] Zachary F. Mainen, Michael Häusser, and Alexandre Pouget. A better way to crack the brain. *Nature News*, 539(7628):159, November 2016. https://doi.org/10.1038/539159a. 10.3.3, 11

[267] Larry F. Abbott, Dora E. Angelaki, Matteo Carandini, Anne K. Churchland, Yang Dan, Peter Dayan, Sophie Deneve, Ila Fiete, Surya Ganguli, Kenneth D. Harris, Michael Häusser, Sonja Hofer, Peter E. Latham, Zachary F. Mainen, Thomas Mrsic-Flogel, Liam Paninski, Jonathan W. Pillow, Alexandre Pouget, Karel Svoboda, Ilana B. Witten, and Anthony M. Zador. An International Laboratory for Systems and Computational Neuroscience. *Neuron*, 96(6):1213–1218, December 2017. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2017.12.013. 10.3.3

[268] The International Brain Laboratory, Valeria Aguillon-Rodriguez, Dora E. Angelaki, Hannah M. Bayer, Niccolò Bonacchi, Matteo Carandini, Fanny Cazettes, Gaelle A. Chapuis, Anne K. Churchland, Yang Dan, Eric E. J. Dewitt, Mayo Faulkner, Hamish Forrest, Laura M. Haetzel, Michael Hausser, Sonja B. Hofer, Fei Hu, Anup Khanal, Christopher S. Krasniak, Inês Laranjeira, Zachary F. Mainen, Guido T. Meijer, Nathaniel J. Miska, Thomas D. Mrsic-Flogel, Masayoshi Murakami, Jean-Paul Noel, Alejandro Pan-Vazquez, Cyrille Rossant, Joshua I. Sanders, Karolina Z. Socha, Rebecca Terry, Anne E. Urai, Hernando M. Vergara, Miles J. Wells, Christian J. Wilson, Ilana B. Witten, Lauren E. Wool, and Anthony Zador. Standardized and reproducible measurement of decision-making in mice. *bioRxiv*, page 2020.01.17.909838, October 2020. https://doi.org/10.1101/2020.01.17.909838. 10.3.3

[269] The International Brain Laboratory, Niccolò Bonacchi, Gaelle Chapuis, Anne Churchland, Kenneth D. Harris, Max Hunter, Cyrille Rossant, Maho Sasaki, Shan Shen, Nicholas A. Steinmetz, Edgar Y. Walker, Olivier Winter, and Miles Wells. Data architecture for a large-scale neuroscience collaboration. *bioRxiv*, page 827873, February 2020. https://doi.org/10.1101/827873. 10.3.3

[270] Gonçalo Lopes, Niccolò Bonacchi, João Frazão, Joana P. Neto, Bassam V. Atallah, Sofia Soares, Luís Moreira, Sara Matias, Pavel M. Itskov, Patrícia A. Correia, Roberto E. Medina, Lorenza Calcaterra, Elena Dreosti, Joseph J. Paton, and Adam R. Kampff. Bonsai: An event-based framework for processing and controlling data streams. *Frontiers in Neuroinformatics*, 9, 2015. ISSN 1662-5196. https://doi.org/10.3389/fninf.2015.00007. 10.3.3

[271] Aral Balkan and Laura Kalbag. Small Technology Foundation - About. https://small-tech.org/about/. 11

[272] Jae Kaplan and Colin Bayer. Part 1: Antisoftware action. https://antisoftware.club/manifesto/2020/03/16/part-1.html, March 2020. 11

[273] Andrei Vlad Sambra, Essam Mansour, Sandro Hawke, Maged Zereba, Nicola Greco, Abdurrahman Ghanem, Dmitri Zagidulin, Ashraf Aboulnaga, and Tim Berners-Lee. Solid: A Platform for Decentralized Social Applications Based on Linked Data. *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.*, page 16, 2016. 11, 11.2.2

[274] Rfc5321 - Simple Mail Transfer Protocol. https://datatracker.ietf.org/doc/html/rfc5321. 11.1.1

[275] Christine Lemmer-Webber. On standards divisions and collaboration (or: Why can't the decentralized social web people just get along?), January 2018. 11.1.1, 11.4.2

[276] Bill Rosenblatt. The Digital Object Identifier: Solving the Dilemma of Copyright Protection Online. *Journal of Electronic Publishing*, 3(2), December 1997. ISSN 1080-2711. https://doi.org/10.3998/3336451.0003.204. 11.1.1

[277] CrossRef. The Formation of CrossRef: A Short History, 2009. 3, 11.1.1

[278] ISO. ISO 26324:2012 - The Digital Object Identifier System, 2012. 11.1.1, 12.1.1

[279] D. Clark. The design philosophy of the DARPA internet protocols. In *Symposium Proceedings on Communications Architectures and Protocols*, SIGCOMM '88, pages 106–114, New York, NY, USA, August 1988. Association for Computing Machinery. ISBN 978-0-89791-279-2. https://doi.org/10.1145/52324.52336. 11.1.2, 11.1.4

[280] Tim Berners-Lee. Principles of Design. https://www.w3.org/DesignIssues/Principles.html, 1998. 11.1.3

[281] Dennis Schubert. ActivityPub - Final thoughts, one year later., January 2019. 11.1.3, 11.4.2

[282] Christopher S Yoo. Protocol Layering and Internet Policy. *University of Pennsylvania Law Review*, 161:66, May 2013. 11.1.3

[283] Brian E. Carpenter. RFC 1958 - Architectural Principles of the Internet. https://tools.ietf.org/html/rfc1958, June 1996. 11.1.3, 11.1.6

[284] Jonathan Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1): 92–105, January 1994. ISSN 0001-0782. https://doi.org/10.1145/175222.175230. 11.1.5

[285] Dave Randall, Rob Procter, Yuwei Lin, Meik Poschen, Wes Sharrock, and Robert Stevens. Distributed ontology building as practical work. *International Journal of Human-Computer Studies*, 69(4):220–233, April 2011. ISSN 1071-5819. https://doi.org/10.1016/j.ijhcs.2010.12.011. 11.1.5

[286] John Markoff. Tomorrow, the World Wide Web!;Microsoft, the PC King, Wants to Reign Over the Internet. *The New York Times*, July 1996. ISSN 0362-4331. 11.1.6

[287] Archive Team. Scientific Data formats - Just Solve the File Format Problem. http://justsolve.archiveteam.org/wiki/Scientific_Data_formats. 11.2.1

[288] Oliver Rübel, Andrew Tritt, Benjamin Dichter, Thomas Braun, Nicholas Cain, Nathan Clack, Thomas J. Davidson, Max Dougherty, Jean-Christophe Fillion-Robin, Nile Graddis, Michael Grauer, Justin T. Kiggins, Lawrence Niu, Doruk Ozturk, William Schroeder, Ivan Soltesz, Friedrich T. Sommer, Karel Svoboda, Ng Lydia, Loren M. Frank, and Kristofer Bouchard. NWB:N 2.0: An Accessible Data Standard for Neurophysiology. *bioRxiv*, page 523035, January 2019. https://doi.org/10.1101/523035. 11.2.1

[289] Oliver Rübel, Andrew Tritt, Ryan Ly, Benjamin K. Dichter, Satrajit Ghosh, Lawrence Niu, Ivan Soltesz, Karel Svoboda, Loren Frank, and Kristofer E. Bouchard. The Neurodata Without Borders ecosystem for neurophysiological data science, March 2021. 11.2.1

[290] Joseph Paul Cohen and Henry Z. Lo. Academic Torrents: A Community-Maintained Distributed Repository. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, XSEDE '14, pages 1–2, New York, NY, USA, July 2014. Association for Computing Machinery. ISBN 978-1-4503-2893-7. https://doi.org/10.1145/2616498.2616528. 11.2.2, 11.2.3

[291] Henry Z. Lo and Joseph Paul Cohen. Academic Torrents: Scalable Data Distribution. March 2016. https://doi.org/10.48550/arXiv.1603.04395. 11.2.2

[292] Morgan G. I. Langille and Jonathan A. Eisen. BioTorrents: A File Sharing Service for Scientific Data. *PLoS ONE*, 5(4), April 2010. ISSN 1932-6203. https://doi.org/10.1371/journal.pone.0010071. 11.2.2, 11.2.3

[293] Daniel S Himmelstein, Ariel Rodriguez Romero, Jacob G Levernier, Thomas Anthony Munro, Stephen Reid McLaughlin, Bastian Greshake Tzovaras, and Casey S Greene. Sci-Hub provides access to nearly all scholarly literature. *eLife*, 7, March 2018. ISSN 2050-084X. https://doi.org/10.7554/eLife.32822. 11.2.2

[294] Sci-Hub. Torrent Health Tracker. https://web.archive.org/web/20220609161833/https://phillm.net/torrent-health-frontend/stats-filtered-table.php?propname%5B%5D=seeders&comp%5B%5D=%3C&value%5B%5D=12&propname%5B%5D=type&co June 2022. 11.2.2

[295] bookwarrior. Library Genesis: Decentralized Library. https://web.archive.org/web/20220601201202/https://libgen.fun/dweb.html, August 2021. 11.2.2

[296] Xuemin Shen, Heather Yu, John Buford, and Mursalin Akon. *Handbook of Peer-to-Peer Networking*. Springer Science & Business Media, March 2010. ISBN 978-0-387-09751-0. 12

[297] Bram Cohen. The BitTorrent Protocol Specification, February 2017. 11.2.2

[298] Ernesto Van der Sar. The Pirate Bay: Five Years After The Raid, May 2011. 11.2.2

[299] Janko Roettgers. The Pirate Bay: Distributing the World's Entertainment for $3,000 a Month. https://gigaom.com/2009/07/19/the-pirate-bay-distributing-the-worlds-entertainment-for-3000-a-month/, July 2009. 11.2.2

[300] The Pirate Bay - Archiveteam. https://wiki.archiveteam.org/index.php?title=The_Pirate_Bay&oldid=45467, September 2020. 11.2.2

[301] Jeffrey Spies. Data Integrity for Librarians, Archivists, and Criminals: What We Can Steal from Bitcoin, BitTorrent, and Usenet, March 2017. 11.2.2

[302] Eddie Kim. After 15 Years, the Pirate Bay Still Can't Be Killed. https://melmagazine.com/en-us/story/after-15-years-the-pirate-bay-still-cant-be-killed, May 2019. 11.2.2

[303] Ernesto Van der Sar. The Open Bay: Now Anyone Can Run A Pirate Bay 'Copy', December 2014. 11.2.2

[304] Ernesto Van der Sar. What.cd is Dead, But The Torrent Hydra Lives on, December 2016. 11.2.2

[305] Jason Scott. Geocities Torrent Update, December 2010. 11.2.2

[306] Dario Rossi, Guilhem Pujol, Xiao Wang, and Fabien Mathieu. Peeking through the BitTorrent Seedbox Hosting Ecosystem. In Alberto Dainotti, Anirban Mahanti, and Steve Uhlig, editors, *Traffic Monitoring and Analysis*, Lecture Notes in Computer Science, pages 115–126, Berlin, Heidelberg, 2014. Springer. ISBN 978-3-642-54999-1. https://doi.org/10.1007/978-3-642-54999-1_10. 11.2.2

[307] John Hoffman and DeHackEd. HTTP-Based Seeding Specification. http://www.bittornado.com/docs/webseed-spec.txt. 11.2.2

[308] Brewster Kahle. Over 1,000,000 Torrents of Downloadable Books, Music, and Movies, August 2012. 11.2.2

[309] G. Kreitz and F. Niemela. Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming. In *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10, Delft, Netherlands, August 2010. IEEE. ISBN 978-1-4244-7140-9. https://doi.org/10.1109/P2P.2010.5569963. 11.2.2

[310] Andrey Andreev, Tom Morrell, Kristin Briney, Sandra Gesing, and Uri Manor. Biologists need modern data infrastructure on campus. *arXiv:2108.07631 [q-bio]*, August 2021. 11.2.2, 12.1.2

[311] Adam S. Charles, Benjamin Falk, Nicholas Turner, Talmo D. Pereira, Daniel Tward, Benjamin D. Pedigo, Jaewon Chung, Randal Burns, Satrajit S. Ghosh, Justus M. Kebschull, William Silversmith, and Joshua T. Vogelstein. Toward Community-Driven Big Open Brain Science: Open Big Data and Tools for Structure, Function, and Genetics. *Annual Review of Neuroscience*, 43:441–464, July 2020. ISSN 1545-4126. https://doi.org/10.1146/annurev-neuro-100119-110036. 11.2.2

[312] Bram Cohen. BEP 52: The BitTorrent Protocol Specification v2. https://www.bittorrent.org/beps/bep_0052.html, August 2017. 15

[313] Juan Benet. IPFS - Content Addressed, Versioned, P2P File System. *arXiv:1407.3561 [cs]*, July 2014. 11.2.2, 11.4.2

[314] Maxwell Ogden. Dat - Distributed Dataset Synchronization And Versioning. Preprint, Open Science Framework, January 2017. 11.2.2

[315] Constantinos Patsakis and Fran Casino. Hydras and IPFS: A Decentralised Playground for Malware. *International Journal of Information Security*, 18(6):787–799, December 2019. ISSN 1615-5262, 1615-5270. https://doi.org/10.1007/s10207-019-00443-0. 11.2.2

[316] C. Zhang, P. Dhungel, D. Wu, and K. W. Ross. Unraveling the BitTorrent Ecosystem. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1164–1177, July 2011. ISSN 1558-2183. https://doi.org/10.1109/TPDS.2010.123. 11.2.2, 11.2.3

[317] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In Hannes Federrath, editor, *Designing Privacy Enhancing Technologies: International Workshop on Design Issues in Anonymity and Unobservability Berkeley, CA, USA, July 25–26, 2000 Proceedings*, Lecture Notes in Computer Science, pages 46–66. Springer, Berlin, Heidelberg, 2001. ISBN 978-3-540-44702-3. https://doi.org/10.1007/3-540-44702-4_4. 11.2.2

[318] Sarven Capadisli, Tim Berners-Lee, Ruben Verborgh, Kjetil Kjernsmo, Justin Bingham, and Dmitri Zagidulin. Solid Protocol. https://solidproject.org/TR/protocol, December 2020. 11.2.2

[319] Solid - P2P Foundation. https://wiki.p2pfoundation.net/Solid. 11.2.2

[320] Jonathan Robert Basamanowicz. *Release Groups and Digital Copyright Piracy*. Thesis, Arts & Social Sciences: School of Criminology, May 2011. 11.2.3

[321] Sameer Hinduja. Deindividuation and Internet Software Piracy. *CyberPsychology & Behavior*, 11(4):391–398, August 2008. ISSN 1094-9313. https://doi.org/10.1089/cpb.2007.0048. 11.2.3

[322] Martin Paul Eve. *Warez: The Infrastructure and Aesthetics of Piracy*. punctum books, first edition, December 2021. ISBN 978-1-68571-037-8. https://doi.org/10.53288/0339.1.00. 11.2.3

[323] Ian Dunham. *What.CD: A Legacy of Sharing*. PhD thesis, Rutgers University - School of Graduate Studies, 2018. 17, 11.2.3

[324] Jody Rosen. The Day the Music Burned. *The New York Times*, June 2019. ISSN 0362-4331. 11.2.3

[325] Nikhil Sonnad. A eulogy for What.cd, the greatest music collection in the history of the world—until it vanished. https://qz.com/840661/what-cd-is-gone-a-eulogy-for-the-greatest-music-collection-in-the-world/, November 2016. 11.2.3

[326] M Meulpolder, L D'Acunto, M Capota, M Wojciechowski, J A Pouwelse, D H J Epema, and H J Sips. Public and private BitTorrent communities: A measurement study. page 5. 11.2.3

[327] Adele Lu Jia, Xiaowei Chen, Xiaowen Chu, Johan A. Pouwelse, and Dick H. J. Epema. How to Survive and Thrive in a Private BitTorrent Community. In Davide Frey, Michel Raynal, Saswati Sarkar, Rudrapatna K. Shyamasundar, and Prasun Sinha, editors, *Distributed Computing and Networking*, Lecture Notes in Computer Science, pages 270–284, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-35668-1. https://doi.org/10.1007/978-3-642-35668-1_19. 11.2.3

[328] Z. Liu, P. Dhungel, D. Wu, C. Zhang, and K. W. Ross. Understanding and Improving Ratio Incentives in Private Communities. In *2010 IEEE 30th International Conference on Distributed Computing Systems*, pages 610–621, June 2010. https://doi.org/10.1109/ICDCS.2010.90. 11.2.3

[329] Ian A. Kash, John K. Lai, Haoqi Zhang, and Aviv Zohar. Economics of BitTorrent communities. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 221–230, New York, NY, USA, April 2012. Association for Computing Machinery. ISBN 978-1-4503-1229-5. https://doi.org/10.1145/2187836.2187867. 11.2.3

[330] X. Chen, X. Chu, and Z. Li. Improving Sustainability of Private P2P Communities. In *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6, July 2011. https://doi.org/10.1109/ICCCN.2011.6005944. 11.2.3

[331] Benedikt Fecher, Sascha Friesike, Marcel Hebing, and Stephanie Linek. A reputation economy: How individual reward considerations trump systemic arguments for open access to data. *Palgrave Communications*, 3(1):1–10, June 2017. ISSN 2055-1045. https://doi.org/10.1057/palcomms.2017.51. 11.2.3

[332] Jordan Bross. *Community, Collaboration and Contribution: Evaluating a BitTorrent Tracker as a Digital Library.* M.S. in Library Science, UNC Chapel Hill, December 2013. 11.2.3

[333] Michel Foucault. *The Order of Things*. Routledge, London, October 2001. ISBN 978-0-203-99664-5. https://doi.org/10.4324/9780203996645. 11.2.4

[334] Tim Berners-Lee. Linked Data. https://www.w3.org/DesignIssues/LinkedData.html, July 2006. 11.2.4

[335] Eun Seo Jo and Timnit Gebru. Lessons from archives: Strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, pages 306–316, New York, NY, USA, January 2020. Association for Computing Machinery. ISBN 978-1-4503-6936-7. https://doi.org/10.1145/3351095.3372829. 11.2.4

[336] Andrew D. Selbst, Danah Boyd, Sorelle A. Friedler, Suresh Venkatasubramanian, and Janet Vertesi. Fairness and Abstraction in Sociotechnical Systems. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 59–68, New York, NY, USA, January 2019. Association for Computing Machinery. ISBN 978-1-4503-6125-5. https://doi.org/10.1145/3287560.3287598. 11.2.4

[337] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, December 2021. ISSN 0001-0782, 1557-7317. https://doi.org/10.1145/3458723. 11.2.4

[338] Geoffrey C. Bowker and Susan Leigh Star. *Sorting Things Out: Classification and Its Consequences*. Inside Technology. MIT Press, Cambridge, MA, USA, September 1999. ISBN 978-0-262-02461-7. 11.2.4

[339] Matthew J. Bietz and Charlotte P. Lee. Collaboration in Metagenomics: Sequence Databases and the Organization of Scientific Work. In Ina Wagner, Hilda Tellioğlu, Ellen Balka, Carla Simone, and Luigina Ciolfi, editors, *ECSCW 2009*, pages 243–262, London, 2009. Springer. ISBN 978-1-84882-854-4. https://doi.org/10.1007/978-1-84882-854-4_15. 11.2.4

[340] Werner Ceusters and Barry Smith. Foundations for a realist ontology of mental disease. *Journal of Biomedical Semantics*, 1(1):10, December 2010. ISSN 2041-1480. https://doi.org/10.1186/2041-1480-1-10. 11.2.4

[341] The Biomedical Data Translator Consortium. Toward A Universal Biomedical Data Translator. *Clinical and Translational Science*, 12(2):86–90, 2019. ISSN 1752-8062. https://doi.org/10.1111/cts.12591. 11.2.4

[342] Richard Bruskiewich, Deepak, Sierra Moxon, Chris Mungall, Harold Solbrig, cbizon, Matthew Brush, Kent Shefchek, Lance Hannestad, YaphetKG, Nomi Harris, bbopjenkins, diatomsRcool, Patrick Wang, Jim Balhoff, Kevin Schaper, JIWEN XIN, Phil Owen, Gregory Stupp, JervenBolleman, The Gitter Badger, Vincent Emonet, and vdancik. Biolink/biolink-model: 2.2.5. Zenodo, September 2021. 11.2.4

[343] Deepak R. Unni, Sierra A. T. Moxon, Michael Bada, Matthew Brush, Richard Bruskiewich, J. Harry Caufield, Paul A. Clemons, Vlado Dancik, Michel Dumontier, Karamarie Fecho, Gustavo Glusman, Jennifer J. Hadlock, Nomi L. Harris, Arpita Joshi, Tim Putman, Guangrong Qin, Stephen A. Ramsey, Kent A. Shefchek, Harold Solbrig, Karthik Soman, Anne E. Thessen, Melissa A. Haendel, Chris Bizon, Christopher J. Mungall, and The Biomedical Data Translator Consortium. Biolink Model: A universal schema for knowledge graphs in clinical, biomedical, and translational science. *Clinical and Translational Science*, n/a(n/a), May 2022. ISSN 1752-8062. https://doi.org/10.1111/cts.13302. 11.2.4

[344] Karamarie Fecho, Anne E. Thessen, Sergio E. Baranzini, Chris Bizon, Jennifer J. Hadlock, Sui Huang, Ryan T. Roper, Noel Southall, Casey Ta, Paul B. Watkins, Mark D. Williams, Hao Xu, William Byrd, Vlado Dančík, Marc P. Duby, Michel Dumontier, Gustavo Glusman, Nomi L. Harris, Eugene W. Hinderer, Greg Hyde, Adam Johs, Andrew I. Su, Guangrong Qin, Qian Zhu, and The Biomedical Data Translator Consortium. Progress toward a universal biomedical data translator. *Clinical and Translational Science*, n/a(n/a), May 2022. ISSN 1752-8062. https://doi.org/10.1111/cts.13301. 11.2.4

[345] Renaissance Computing Institute (RENCI). Biomedical Data Translator Platform moves to the next phase, March 2022. 11.2.4, 30

[346] Ruth Hailu. NIH-funded project aims to build a 'Google' for biomedical data, July 2019. 11.2.4

[347] Renaissance Computing Institute (RENCI). Use cases show Translator's potential to expedite clinical research, March 2022. 11.2.4

[348] Prateek Goel, Adam J Johs, Manil Shrestha, and Rosina O Weber. Explanation Container in Case-Based Biomedical Question-Answering. page 10, September 2021. 11.2.4

[349] Thomas Grote and Philipp Berens. On the ethics of algorithmic decision-making in healthcare. *Journal of Medical Ethics*, 46(3):205–211, March 2020. ISSN 0306-6800, 1473-4257. https://doi.org/10.1136/medethics-2019-105586. 11.2.4

[350] Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, October 2019. https://doi.org/10.1126/science.aax2342. 11.2.4

[351] Trishan Panch, Heather Mattie, and Rifat Atun. Artificial intelligence and algorithmic bias: Implications for health systems. *Journal of Global Health*, 9(2):020318, November 2019. ISSN 2047-2978. https://doi.org/10.7189/jogh.09.020318. 11.2.4

[352] Trishan Panch, Heather Mattie, and Leo Anthony Celi. The "inconvenient truth" about AI in healthcare. *npj Digital Medicine*, 2(1):1–3, August 2019. ISSN 2398-6352. https://doi.org/10.1038/s41746-019-0155-4. 11.2.4

[353] Kenneth Morton, Patrick Wang, Chris Bizon, Steven Cox, James Balhoff, Yaphet Kebede, Karamarie Fecho, and Alexander Tropsha. ROBOKOP: An abstraction layer and user interface for knowledge graphs to support question answering. *Bioinformatics*, 35(24):5382–5384, December 2019. ISSN 1367-4803. https://doi.org/10.1093/bioinformatics/btz604. 11.2.4

[354] Johan Ordish, Hannah Murfet, and Alison Hall. Algorithms as Medical Devices. Technical report, PHG Foundation, 2019. 11.2.4

[355] Soleil Shah El-Sayed, Abdul. Medical Algorithms Need Better Regulation. https://www.scientificamerican.com/article/the-fda-should-better-regulate-medical-algorithms/, October 2021. 11.2.4

[356] A Ram, Clair A Kronk, Jacob R Eleazer, Joseph L Goulet, Cynthia A Brandt, and Karen H Wang. Transphobia, encoded: An examination of trans-specific terminology in SNOMED CT and ICD-10-CM. *Journal of the American Medical Informatics Association*, (ocab200), September 2021. ISSN 1527-974X. https://doi.org/10.1093/jamia/ocab200. 11.2.4

[357] Wanvisa Udomsinprasert, Noppadol Chanhom, Supharat Suvichapanich, Sukanya Wattanapokayakit, Surakameth Mahasirimongkol, Wasun Chantratita, and Jiraphun Jittikoon. Leukocyte telomere length as a diagnostic biomarker for anti-tuberculosis drug-induced liver injury. *Scientific Reports*, 10(1):5628, March 2020. ISSN 2045-2322. https://doi.org/10.1038/s41598-020-62635-2. 11.2.4

[358] RePORT ⟩ RePORTER "Biomedical Data Translator". https://reporter.nih.gov/search/kDJ97zGUFEaIBIltUmyd_Q/projects?sort_field= October 2021. 11.2.4

[359] Krishnan Subramanian. Google Buys Freebase - This is Huge, July 2010. 11.2.4

[360] Introducing the Knowledge Graph: Things, not strings. https://blog.google/products/search/introducing-knowledge-graph-things-not/, May 2012. 11.2.4

[361] Danny Sullivan. FAQ: All About The New Google "Hummingbird" Algorithm. https://searchengineland.com/google-hummingbird-172816, September 2013. 11.2.4

[362] Translator Consortium. Clinical Data Services Provider, April 2020. 28

[363] Krishna Bharat, Stephen Lawrence, and Mehran Sahami. Generating user information for use in targeted advertising, June 2005. 29, 11.2.4

[364] Smith v. Facebook, Inc., No. 17-16206 (9th Cir. Dec. 6, 2018). https://casetext.com/case/smith-v-facebook-inc-2, December 2018. 11.2.4

[365] Susan Krashinsky. Google broke Canada's privacy laws with targeted health ads, watchdog says. *The Globe and Mail*, January 2014. 11.2.4

[366] Marc Bourreau, Cristina Caffarra, Zhijun Chen, Chongwoo Choe, Gregory S Crawford, Tomaso Duso, Christos Genakos, Paul Heidhues, Martin Peitz, Thomas Rønde, Monika Schnitzer, Nicolas Schutz, Michelle Sovinsky, Giancarlo Spagnolo, Otto Toivanen, Tommaso Valletti, and Thibaud Vergé. Google/Fitbit will monetise health data and harm consumers. (107):13, 2020. 11.2.4

[367] Lauren Bridges. Amazon's Ring is the largest civilian surveillance network the US has ever seen. *The Guardian*, May 2021. ISSN 0261-3077. 11.2.4

[368] AWS announces AWS Healthcare Accelerator for startups in the public sector. https://aws.amazon.com/blogs/publicsector/aws-announces-healthcare-accelerator-program-startups-public-sector/, June 2021. 11.2.4

[369] Rachel Lerman. Amazon built its own health-care service for employees. Now it's selling it to other companies. *Washington Post*, March 2021. ISSN 0190-8286. 11.2.4

[370] Corey Quinn. You Can't Trust Amazon When It Feels Threatened. https://www.lastweekinaws.com/blog/you-cant-trust-amazon-when-it-feels-threatened/, March 2021. 11.2.4

[371] Elsevier. 360° advertising solutions | Advertising | Advertisers. https://www.elsevier.com/advertising-reprints-supplements/advertising, . 11.2.4, 11.4, 12.4.1

[372] RELX. 2021 Annual Report. Annual Report, RELX, 2021. 11.2.4

[373] The Biomedical Data Translator Consortium. The Biomedical Data Translator Program: Conception, Culture, and Community. *Clinical and Translational Science*, 12(2):91–94, 2019. ISSN 1752-8062. https://doi.org/10.1111/cts.12592. 11.2.4, 30

[374] Tarek Shamma. Translation and colonialism. In *The Routledge Handbook of Translation and Culture*, pages 279–295. Routledge, 2018. ISBN 978-1-315-67089-8. 11.2.4

[375] Tim Berners-Lee, James HENDLER, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001. ISSN 0036-8733. 11.2.5

[376] Michael Hanke, Franco Pestilli, Adina S. Wagner, Christopher J. Markiewicz, Jean-Baptiste Poline, and Yaroslav O. Halchenko. In defense of decentralized research data management. *Neuroforum*, 27(1):17–25, February 2021. ISSN 1868-856X. https://doi.org/10.1515/nf-2020-0037. 31, 34, 11.2.5

[377] Graham Klyne, David Wood, and Markus Lanthaler. RDF 1.1 Concepts and Abstract Syntax. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/, February 2014. 11.2.5

[378] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. RDF 1.1 Turtle, February 2014. 11.2.5

[379] w3c. Ontologies - W3C. https://www.w3.org/standards/semanticweb/ontology. 11.2.5

[380] Dan Brickley and R.V. Guha. RDF Schema 1.1, February 2014. 11.2.5

[381] Tim Berners-Lee. Relational Databases and the Semantic Web (in Design Issues). https://www.w3.org/DesignIssues/RDB-RDF.html, August 2009. 11.2.5

[382] Tim Berners-Lee. What the Semantic Web can Represent. https://www.w3.org/DesignIssues/RDFnot.html, September 1998. 11.2.5

[383] Dan Brickley and Alistair Miles. SKOS Core Guide. https://www.w3.org/TR/swbp-skos-core-guide/, November 2005. 11.2.5

[384] Alistair Miles. Quick Guide to Publishing a Thesaurus on the Semantic Web. https://www.w3.org/TR/swbp-thesaurus-pubguide/, May 2005. 11.2.5

[385] Gavia Libraria. Stoning Goliath, June 2022. 11.2.5, 11.4.3

[386] Steve Speicher, John Arwe, and Malhotra. Linked Data Platform 1.0, February 2015. 11.2.5, 11.2.5

[387] Dennis Heimbigner and Dennis McLeod. A federated architecture for information management. *ACM Transactions on Information Systems*, 3(3):253–278, July 1985. ISSN 1046-8188. https://doi.org/10.1145/4229.4233. 11.2.5

[388] Witold Litwin, Leo Mark, and Nick Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22(3):267–293, September 1990. ISSN 0360-0300. https://doi.org/10.1145/96602.96608. 11.2.5

[389] Vipul Kashyap and Amit Sheth. Semantic and schematic similarities between database objects:a context-based approach. *The VLDB Journal*, 5(4):276–304, December 1996. ISSN 0949-877X. https://doi.org/10.1007/s007780050029. 11.2.5

[390] Richard Hull. Managing semantic heterogeneity in databases: A theoretical prospective. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '97, pages 51–61, New York, NY, USA, May 1997. Association for Computing Machinery. ISBN 978-0-89791-910-4. https://doi.org/10.1145/263661.263668. 11.2.5

[391] Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser, and Herbert Weber. Federated Information Systems: Concepts, Terminology and Architectures. page 40, 1999. 11.2.5, 11.2.5

[392] Marija Djokic-Petrovic, Vladimir Cvjetkovic, Jeremy Yang, Marko Zivanovic, and David J. Wild. PIBAS FedSPARQL: A web-based platform for integration and exploration of bioinformatics datasets. *Journal of Biomedical Semantics*, 8(1): 42, September 2017. ISSN 2041-1480. https://doi.org/10.1186/s13326-017-0151-z. 11.2.5, 11.2.5

[393] Ali Hasnain, Qaiser Mehmood, Syeda Sana e Zainab, Muhammad Saleem, Claude Warren, Durre Zehra, Stefan Decker, and Dietrich Rebholz-Schuhmann. BioFed: Federated query processing over life sciences linked open data. *Journal of Biomedical Semantics*, 8(1):13, March 2017. ISSN 2041-1480. https://doi.org/10.1186/s13326-017-0118-0. 11.2.5, 11.2.5

[394] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, September 1990. ISSN 0360-0300. https://doi.org/10.1145/96602.96604. 11.2.5

[395] Angela Bonifati, Panos K. Chrysanthis, Aris M. Ouksel, and Kai-Uwe Sattler. Distributed databases and peer-to-peer databases: Past and present. *ACM SIGMOD Record*, 37(1):5–11, March 2008. ISSN 0163-5808. https://doi.org/10.1145/1374780.1374781. 11.2.5

[396] Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reed, Orie Steele, and Christopher Allen. Decentralized Identifiers (DIDs) v1.0. https://w3c.github.io/did-core/, August 2021. 11.2.5

[397] Meatball Wiki: PersonalCategories. http://meatballwiki.org/wiki/PersonalCategories, . 11.2.5

[398] Giuseppe Pirrò, Domenico Talia, and Paolo Trunfio. A DHT-based semantic overlay network for service discovery. *Future Generation Computer Systems*, 28(4):689–707, April 2012. ISSN 0167-739X. https://doi.org/10.1016/j.future.2011.11.007. 11.2.5

[399] Christine Webber, Jessica Tallon, Erin Shepherd, Amy Guy, and Evan Prodromou. ActivityPub. W3C recommendation, W3C, January 2018. 11.2.5, 11.4.2

[400] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, Pierre-Antoine Champin, and Niklas Lindström. JSON-LD 1.1 - A JSON-based Serialization for Linked Data. https://www.w3.org/TR/json-ld/, July 2020. 11.2.5

[401] James M Snell and Evan Prodromou. Activity Streams 2.0. https://www.w3.org/TR/activitystreams-core/, May 2017. 11.2.5

[402] SPARQL 1.1 Federated Query. https://www.w3.org/TR/sparql11-federated-query/, March 2013. 11.2.5

[403] Ana Claudia Sima, Tarcisio Mendes de Farias, Erich Zbinden, Maria Anisimova, Manuel Gil, Heinz Stockinger, Kurt Stockinger, Marc Robinson-Rechavi, and Christophe Dessimoz. Enabling semantic queries across federated bioinformatics databases. *Database*, 2019(baz106), January 2019. ISSN 1758-0463. https://doi.org/10.1093/database/baz106. 11.2.5

[404] Yaroslav O. Halchenko, Kyle Meyer, Benjamin Poldrack, Debanjum Singh Solanky, Adina S. Wagner, Jason Gors, Dave MacFarlane, Dorian Pustina, Vanessa Sochat, Satrajit S. Ghosh, Christian Mönch, Christopher J. Markiewicz, Laura Waite, Ilya Shlyakhter, Alejandro de la Vega, Soichi Hayashi, Christian Olaf Häusler, Jean-Baptiste Poline, Tobias Kadelka, Kusti Skytén, Dorota Jarecka, David Kennedy, Ted Strauss, Matt Cieslak, Peter Vavra, Horea-Ioan Ioanas, Robin Schneider, Mika Pflüger, James V. Haxby, Simon B. Eickhoff, and Michael Hanke. DataLad: Distributed system for joint management of code, data, and their relationship. *Journal of Open Source Software*, 6(63):3262, July 2021. ISSN 2475-9066. https://doi.org/10.21105/joss.03262. 34, 12.1.1

[405] Jonny L. Saunders and Michael Wehr. Autopilot: Automating behavioral experiments with lots of Raspberry Pis. *bioRxiv*, page 807693, October 2019. https://doi.org/10.1101/807693. 11.3, 11.3.2

[406] Jeffrey Spies. A Workflow-Centric Approach to Increasing Reproducibility and Data Integrity. August 2017. 11.3

[407] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. ISSN 1476-4687. https://doi.org/10.1038/s41586-020-2649-2. 11.3.1

[408] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C. J. Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, and Paul van Mulbregt. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3):261–272, March 2020. ISSN 1548-7105. https://doi.org/10.1038/s41592-019-0686-2. 11.3.1

[409] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research*, 12(null):2825–2830, November 2011. ISSN 1532-4435. 11.3.1

[410] Alexander B. Wiltschko, Tatsuya Tsukahara, Ayman Zeine, Rockwell Anyoha, Winthrop F. Gillis, Jeffrey E. Markowitz, Ralph E. Peterson, Jesse Katon, Matthew J. Johnson, and Sandeep Robert Datta. Revealing the structure of pharmacobehavioral space through motion sequencing. *Nature Neuroscience*, 23(11):1433–1443, November 2020. ISSN 1546-1726. https://doi.org/10.1038/s41593-020-00706-3. 11.3.1

[411] Kevin R. Coffey, Russell G. Marx, and John F. Neumaier. DeepSqueak: A deep learning-based system for detection and analysis of ultrasonic vocalizations. *Neuropsychopharmacology*, 44(5):859–868, April 2019. ISSN 1740-634X. https://doi.org/10.1038/s41386-018-0303-6. 11.3.1

[412] Dimitri Yatsenko, Edgar Y. Walker, and Andreas S. Tolias. DataJoint: A Simpler Relational Data Model. *arXiv:1807.11104 [cs]*, July 2018. 11.3.1

[413] Dimitri Yatsenko, Thinh Nguyen, Shan Shen, Kabilar Gunalan, Christopher A. Turner, Raphael Guzman, Maho Sasaki, Daniel Sitonic, Jacob Reimer, Edgar Y. Walker, and Andreas S. Tolias. DataJoint Elements: Data Workflows for Neurophysiology. *bioRxiv*, page 2021.03.30.437358, March 2021. https://doi.org/10.1101/2021.03.30.437358. 11.3.1, 12.1.1

[414] Marius Pachitariu, Nicholas Steinmetz, Shabnam Kadir, Matteo Carandini, and Harris Kenneth D. Kilosort: Realtime spike-sorting for extracellular electrophysiology with hundreds of channels. Article; https://web.archive.org/web/20211015215729/https://www.biorxiv.org/content/10.1101/061481v1, Cold Spring Harbor Laboratory, June 2016. 11.3.1

[415] Todd Gamblin, Matthew LeGendre, Michael R. Collette, Gregory L. Lee, Adam Moody, Bronis R. de Supinski, and Scott Futral. The spack package manager: Bringing order to HPC software chaos. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, New York, NY, USA, November 2015. Association for Computing Machinery. ISBN 978-1-4503-3723-6. https://doi.org/10.1145/2807591.2807623. 11.3.1

[416] Eelco Dolstra, Merijn de Jonge, and Eelco Visser. Nix: A Safe and Policy-Free System for Software Deployment. In *Proceedings of the 18th USENIX Conference on System Administration*, LISA '04, pages 79–92, USA, November 2004. USENIX Association. 11.3.1

[417] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: Yet another workflow language. *Information Systems*, 30(4): 245–275, June 2005. ISSN 0306-4379. https://doi.org/10.1016/j.is.2004.02.002. 11.3.1

[418] Ax Sharma. Python packages upload your AWS keys, env vars, secrets to the web. https://blog.sonatype.com/python-packages-upload-your-aws-keys-env-vars-secrets-to-web, June 2022. 11.3.1

[419] Sara Steegen, Francis Tuerlinckx, Andrew Gelman, and Wolf Vanpaemel. Increasing Transparency Through a Multiverse Analysis. *Perspectives on Psychological Science*, 11(5):702–712, September 2016. ISSN 1745-6916. https://doi.org/10.1177/1745691616658637. 11.3.1

[420] Stefan M. Larson, Christopher D. Snow, Michael Shirts, and Vijay S. Pande. Folding@Home and Genome@Home: Using distributed computing to tackle previously intractable problems in computational biology, January 2009. 11.3.1

[421] Adam L. Beberg, Daniel L. Ensign, Guha Jayachandran, Siraj Khaliq, and Vijay S. Pande. Folding@home: Lessons from eight years of volunteer distributed computing. In *2009 IEEE International Symposium on Parallel & Distributed Processing*, pages 1–8, May 2009. https://doi.org/10.1109/IPDPS.2009.5160922. 11.3.1

[422] Larry Smarr, Camille Crittenden, Thomas DeFanti, John Graham, Dmitry Mishin, Richard Moore, Philip Papadopoulos, and Frank Würthwein. The pacific research platform: Making high-speed networking a reality for the scientist. *Proceedings of the practice and experience on advanced research computing*, pages 1–8, 2018. https://doi.org/10.1145/3219104.3219108. 11.3.1

[423] Michael Wulf. *BEADL XML Documentation V 0.1*. July 2020. 11.3.2

[424] NWB Behavioral Task WG. NWB Behavioral Task WG, April 2020. 11.3.2

[425] Jonny L. Saunders, Lucas A. Ott, and Michael Wehr. AUTOPILOT: Automating experiments with lots of raspberry pis. *bioRxiv : the preprint server for biology*, 2022. https://doi.org/10.1101/807693v2. 11.3.2

[426] Gonçalo Lopes, Niccolò Bonacchi, João Frazão, Joana P. Neto, Bassam V. Atallah, Sofia Soares, Luís Moreira, Sara Matias, Pavel M. Itskov, Patrícia A. Correia, Roberto E. Medina, Lorenza Calcaterra, Elena Dreosti, Joseph J. Paton, and Adam R. Kampff. Bonsai: An event-based framework for processing and controlling data streams. *Frontiers in Neuroinformatics*, 9:7, 2015. ISSN 1662-5196. https://doi.org/10.3389/fninf.2015.00007. 11.3.2

[427] Gonçalo Lopes and Patricia Monteiro. New Open-Source Tools: Using Bonsai for Behavioral Tracking and Closed-Loop Experiments. *Frontiers in Behavioral Neuroscience*, 15:53, 2021. ISSN 1662-5153. https://doi.org/10.3389/fnbeh.2021.647640. 11.3.2

[428] Theodore D. Sterling. Publication Decisions and Their Possible Effects on Inferences Drawn from Tests of Significance–Or Vice Versa. *Journal of the American Statistical Association*, 54(285):30–34, 1959. ISSN 0162-1459. https://doi.org/10.2307/2282137. 11.3.2

[429] Annie Franco, Neil Malhotra, and Gabor Simonovits. Publication bias in the social sciences: Unlocking the file drawer. *Science*, 345(6203):1502–1505, September 2014. ISSN 0036-8075, 1095-9203. https://doi.org/10.1126/science.1255484. 11.3.2

[430] Eve Marder. Maintaining the joy of discovery. *eLife*, 11:e80711, June 2022. ISSN 2050-084X. https://doi.org/10.7554/eLife.80711. 38

[431] RELX. 2020 Results presentation to Investors - Transcript, February 2021. 11.4

[432] RELX Annual Report 2019, 2019. 11.4

[433] Björn Brembs. Prestigious Science Journals Struggle to Reach Even Average Reliability. *Frontiers in Human Neuroscience*, 12, 2018. ISSN 1662-5161. 11.4

[434] Freya De Keyzer, Nathalie Dens, and Patrick De Pelsmacker. The processing of native advertising compared to banner advertising: An eye-tracking experiment. *Electronic Commerce Research*, November 2021. ISSN 1572-9362. https://doi.org/10.1007/s10660-021-09523-7. 41

[435] Iordanis Koutsopoulos and Panagiotis Spentzouris. Native Advertisement Selection and Allocation in Social Media Post Feeds. In Paolo Frasconi, Niels Landwehr, Giuseppe Manco, and Jilles Vreeken, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 588–603, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46128-1. https://doi.org/10.1007/978-3-319-46128-1_37. 41

[436] Springer Nature. Branded Content. https://partnerships.nature.com/product/branded-content-native-advertising/. 11.4

[437] Elsevier. Drug design optimization. https://www.elsevier.com/solutions/professional-services/drug-design-optimization, . 11.4

[438] Elsevier. Topic Prominence in Science - Scival | Elsevier Solutions. https://www.elsevier.com/solutions/scival/features/topic-prominence-in-science. 11.4

[439] Cody Hanson. User Tracking on Academic Publisher Platforms. In *The Coalition for Networked Information*, St. Louis, MO, April 2019. 11.4

[440] Research Foundation of Korea and Elsevier. South Korea: A technological powerhouse strengthening its research and innovation footprint, 2020. 11.4

[441] Somi Seong, Steven W. Popper, Charles A. Goldman, David K. Evans, and Clifford A. Grammich. Brain Korea 21 Phase II: A New Evaluation Model. Technical report, RAND Corporation, March 2008. 11.4

[442] Elsevier. Case Study: National Research Foundation of Korea. https://www.elsevier.com/__data/assets/pdf_file/0007/927583/ACAD_RI 2019. 11.4

[443] Elsevier Korea. SciVal □□□ □□ Workflow □□. https://www.elsevier.com/__data/assets/pdf_file/0010/1179244/01.-SciVal-Workflow202112thSciVal-User-Group-Meeting.pdf, 2021. 11.4

[444] James Heathers. The Real Scandal About Ivermectin. https://www.theatlantic.com/science/archive/2021/10/ivermectin-research-problems/620473/, October 2021. 11.4

[445] Helen Shen. Meet this super-spotter of duplicated images in science papers. *Nature*, 581(7807):132–136, May 2020. https://doi.org/10.1038/d41586-020-01363-z. 11.4

[446] Elisabeth M. Bik, Arturo Casadevall, and Ferric C. Fang. The Prevalence of Inappropriate Image Duplication in Biomedical Research Publications. *mBio*, 7(3):e00809–16, June 2016. https://doi.org/10.1128/mBio.00809-16. 11.4

[447] Jevin D. West and Carl T. Bergstrom. Misinformation in and about science. *Proceedings of the National Academy of Sciences*, 118(15):e1912444117, April 2021. https://doi.org/10.1073/pnas.1912444117. 11.4

[448] Brian Kennedy, Alec Tyson, and Cary Funk. Americans' Trust in Scientists, Other Groups Declines. February 2022. 11.4

[449] Nicole M. Krause, Dietram A. Scheufele, Isabelle Freiling, and Dominique Brossard. The Trust Fallacy: Scientists' search for public pathologies is unhealthy, unhelpful, and ultimately unscientific. *American Scientist*, 109(4):226–232, July 2021. ISSN 00030996. 11.4

[450] Johan S. G. Chu and James A. Evans. Slowed canonical progress in large fields of science. *Proceedings of the National Academy of Sciences*, 118(41), October 2021. ISSN 0027-8424, 1091-6490. https://doi.org/10.1073/pnas.2021636118. 11.4

[451] Dave Ellenwood. "Information Has Value": The Political Economy of Information Capitalism – In the Library with the Lead Pipe. *In The Library With The Lead Pipe*, August 2020. 11.4

[452] Samantha R. White, Linda M. Amarante, Alexxai V. Kravitz, and Mark Laubach. The Future Is Open: Open-Source Tools for Behavioral Neuroscience Research. *eNeuro*, 6(4):ENEURO.0223–19.2019, August 2019. ISSN 2373-2822. https://doi.org/10.1523/ENEURO.0223-19.2019. 11.4

[453] Dan Goodman. The current system of journals and peer review is not serving science. I have therefore resigned from all editorial roles and will no longer do pre-publication peer review. I explain why in this article and in the thread below. Please consider joining me. http://neural-reckoning.org/reviewing.html, May 2022. 43

[454] Melinda Baldwin. Scientific Autonomy, Public Accountability, and the Rise of "Peer Review" in the Cold War United States. *Isis*, 109(3):538–558, September 2018. ISSN 0021-1753. https://doi.org/10.1086/700070. 11.4

[455] Swartz. Making More Wikipedias (Aaron Swartz's Raw Thought). http://www.aaronsw.com/weblog/morewikipedias, September 2006. 11.4.1

[456] Bo Leuf and Ward Cunningham. *The Wiki Way : Quick Collaboration on the Web*. Boston : Addison-Wesley, 2001. ISBN 978-0-201-71499-9. 11.4.1

[457] Benjamin Mako Hill and Aaron Shaw. Wikipedia and the End of Open Collaboration? *Wikipedia @ 20*, page 12, 2019. 11.4.1

[458] Aaron Swartz. Who Writes Wikipedia? (Aaron Swartz's Raw Thought). http://www.aaronsw.com/weblog/whowriteswikipedia, September 2006. 11.4.1

[459] Aaron Halfaker, R. Stuart Geiger, Jonathan T. Morgan, and John Riedl. The Rise and Decline of an Open Collaboration System: How Wikipedia's Reaction to Popularity Is Causing Its Decline. *American Behavioral Scientist*, 57(5):664–688, May 2013. ISSN 0002-7642, 1552-3381. https://doi.org/10.1177/0002764212469365. 11.4.1

[460] C2wiki - Wiki History. http://wiki.c2.com/?WikiHistory. 11.4.1

[461] beka valentine. C2wiki is an exercise in dialogical methods. of laying bare the fact that knowledge and ideas are not some truth delivered from On High, but rather a social process, a conversation, a dialectic, between various views and interests, October 2021. 11.4.1

[462] Wikipedia:Flow. *Wikipedia*, June 2021. 11.4.1

[463] Jodi Schneider, Alexandre Passant, and John G. Breslin. Understanding and improving Wikipedia article discussion spaces: 2011 ACM Symposium. pages 808–813, 2011. https://doi.org/10.1145/1982185.1982358. 11.4.1

[464] Fernanda B. Viegas, Martin Wattenberg, Jesse Kriss, and Frank van Ham. Talk Before You Type: Coordination in Wikipedia. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pages 78–78, Waikoloa, HI, January 2007. IEEE. ISBN 978-0-7695-2755-0. https://doi.org/10.1109/HICSS.2007.511. 11.4.1

[465] Robert E. Cummings. "WhatWas a Wiki, and Why Do I Care? A Short and Usable History of Wikis" - Wildwiki. https://web.archive.org/web/20090921042301/http://www.wildwiki.net/mediawiki/index.php?title=%E2%80%9CWhatWas_a_Wiki%2 September 2009. 50

[466] Meatball Wiki: RightToLeave. http://meatballwiki.org/wiki/RightToLeave, . 11.4.1

[467] Terry Hancock. OpenOffice.org is Dead, Long Live LibreOffice – or, The Freedom to Fork. http://freesoftwaremagazine.com/articles/openoffice_org_dead_long_live_libreoffice/, October 2010. 11.4.1

[468] Meatball Wiki: EnlargeSpace. http://meatballwiki.org/wiki/EnlargeSpace, . 11.4.1

[469] Meatball Wiki: ForkingOfOnlineCommunities. http://meatballwiki.org/wiki/ForkingOfOnlineCommunities, . 11.4.1

[470] Nathaniel Tkacz. The Spanish Fork: Wikipedia's ad-fuelled mutiny. *Wired UK*, January 2011. ISSN 1357-0978. 11.4.1

[471] Nathaniel Tkacz. *Wikipedia and the Politics of Openness*. University of Chicago Press, December 2014. ISBN 978-0-226-19244-4. https://doi.org/10.7208/9780226192444. 11.4.1

[472] Naomi Klein. Were the DC and Seattle Protests Unfocused?, July 2001. 11.4.1

[473] Murray Bookchin. A Note on Affinity Groups. page 2, 1969. 11.4.1

[474] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, Carol Willing, and Jupyter Development Team. Jupyter Notebooks – a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, 2016. https://doi.org/10.3233/978-1-61499-649-1-87. 11.4.2, 56

[475] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(03):90–95, May 2007. ISSN 1558-366X. https://doi.org/10.1109/MCSE.2007.55. 11.4.2

[476] Silvio Peroni and David Shotton. FaBiO and CiTO: Ontologies for describing bibliographic resources and citations. *Journal of Web Semantics*, 17:33–43, December 2012. ISSN 1570-8268. https://doi.org/10.1016/j.websem.2012.08.001. 11.4.2

[477] Pål Sørgaard, Tone Irene Sandahl, and Fredrik Ljungberg. Use of paragraph styles in word processing: A stepping stone for CSCW? page 10, 1996. https://doi.org/10.1.1.55.7928. 11.4.2

[478] Donald Ervin Knuth. *The TeXbook*. Number A in Computers & Typesetting. Addison-Wesley, Reading, Mass, 1986. ISBN 978-0-201-13447-6 978-0-201-13448-3. 11.4.2

[479] Elizabeth DuPre, Chris Holdgraf, Agah Karakuzu, Loïc Tetrel, Pierre Bellec, Nikola Stikov, and Jean-Baptiste Poline. Beyond advertising: New infrastructures for publishing integrated research objects. *PLOS Computational Biology*, 18(1): e1009651, January 2022. ISSN 1553-7358. https://doi.org/10.1371/journal.pcbi.1009651. 11.4.2, 11.4.3

[480] James Forrester. Inventing as we go: Building a visual editor for MediaWiki, December 2012. 11.4.2

[481] Sarven Capadisli, Amy Guy, Ruben Verborgh, Christoph Lange, Auer Sören, and Tim Berners-Lee. Decentralised Authoring, Annotations and Notifications for a Read-Write Web with dokieli. https://csarven.ca/dokieli-rww, February 2017. 11.4.2

[482] Sarven Capadisli. Linked Research on the Decentralised Web. https://csarven.ca/linked-research-decentralised-web, July 2019. 11.4.2

[483] Souti Chattopadhyay, Ishita Prasad, Austin Z. Henley, Anita Sarma, and Titus Barik. What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–12. Association for Computing Machinery, New York, NY, USA, April 2020. ISBN 978-1-4503-6708-0. 11.4.2

[484] Adam Rule, Amanda Birmingham, Cristal Zuniga, Ilkay Altintas, Shih-Cheng Huang, Rob Knight, Niema Moshiri, Mai H. Nguyen, Sara Brin Rosenthal, Fernando Pérez, and Peter W. Rose. Ten simple rules for writing and sharing computational analyses in Jupyter Notebooks. *PLOS Computational Biology*, 15(7):e1007007, July 2019. ISSN 1553-7358. https://doi.org/10.1371/journal.pcbi.1007007. 11.4.2

[485] Morgan F. Wofford, Bernadette M. Boscoe, Christine L. Borgman, Irene V. Pasquetto, and Milena S. Golshan. Jupyter Notebooks as Discovery Mechanisms for Open Science: Citation Practices in the Astronomy Community. *Computing in Science & Engineering*, 22(1):5–15, January 2020. ISSN 1558-366X. https://doi.org/10.1109/MCSE.2019.2932067. 11.4.2

[486] kaniini. ActivityPub: The present state, or why saving the 'worse is better' virus is both possible and important, January 2019. 11.4.2

[487] Christine Webber and Manu Sporny. ActivityPub: From Decentralized to Distributed Social Networks. page 15, November 2017. 11.4.2

[488] Laura M. Jiménez and Jill M. Hermann-Wilmarth. Borderlanding academic researchers: A range of whisper networks. *Journal of Lesbian Studies*, 24(4):327–331, 2020. https://doi.org/10.1080/10894160.2019.1678331. 11.4.2

[489] Cory Doctorow. Adversarial Interoperability: Reviving an Elegant Weapon From a More Civilized Age to Slay Today's Monopolies. https://www.eff.org/deeplinks/2019/06/adversarial-interoperability-reviving-elegant-weapon-more-civilized-age-slay, June 2019. 11.4.2

[490] Cory Doctorow. Adversarial Interoperability. https://www.eff.org/deeplinks/2019/10/adversarial-interoperability, October 2019. 11.4.2

[491] H. J. Jackson. *Marginalia: Readers Writing in Books*. Yale University Press, January 2001. ISBN 978-0-300-09720-7. 11.4.2

[492] csillag. Fuzzy Anchoring, April 2013. 11.4.2

[493] Stevan Harnad. Sky-Writing. https://web.archive.org/web/20220315173536/https://www.southampton.ac.uk/~harnad/skywriting.html, 1987. 11.4.2

[494] Stevan Harnad. Scholarly Skywriting and the Prepublication Continuum of Scientific Inquiry. *Psychological Science*, 1 (6):342–344, November 1990. ISSN 0956-7976. https://doi.org/10.1111/j.1467-9280.1990.tb00234.x. 11.4.2

[495] Michael Ellis. Books on translation wanted, November 1986. 11.4.2

[496] eLife partners with Hypothes.is to advance open scholarly annotation. https://elifesciences.org/for-the-press/7e7220f6/elife-partners-with-hypothes-is-to-advance-open-scholarly-annotation, September 2016. 11.4.2

[497] dwhly. bioRxiv Selects Hypothesis to Enable Annotation on Preprints, September 2017. 11.4.2

[498] heatherstaines. Preprint Services Gather to Explore an Annotated Future, February 2018. 11.4.2

[499] nateangell. Announcing TRiP: Transparent Review in Preprints, Powered by Hypothesis Annotation, September 2019. 11.4.2

[500] Eduardo Ivanec, Dan Whaley, Daniel Doyon, Ward Cunningham, Bastien Guerry, Oliver Sauter, Conor White Sullivan, and Junyu Zhan. The Future of Note Taking (FoNT). In *I Annotate 2021*, June 2021. 11.4.2

[501] Ivo Velitchkov and George Anadiotis, editors. *Personal Knowledge Graphs*. Exapt Press. 11.4.2

[502] Arthur Boston. Need to Know: The Information-Seeking Behavior of Doja Cat. *twitter.com*, June 2022. 11.4.2

[503] Krisztian Balog and Tom Kenter. Personal Knowledge Graphs: A Research Agenda. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 217–220, Santa Clara CA USA, September 2019. ACM. ISBN 978-1-4503-6881-0. https://doi.org/10.1145/3341981.3344241. 11.4.2

[504] Protocol Labs. IPLD Docs. https://ipld.io/docs/, 2021. 11.4.2

[505] Scott Chacon and Ben Straub. *Pro Git*. Apress, 2.1.346-2-g258c9a1 edition, 22-06-20. 11.4.2

[506] Daniel Aleksandersen. Four P2P distribution tools for Git repositories compared. https://www.ctrl.blog/entry/git-p2p-compared.html, May 2020. 61

[507] Chris Hartgerink. Verified, Shared, Modular, and Provenance Based Research Communication with the Dat Protocol. *Publications*, 7(2):40, June 2019. ISSN 2304-6775. https://doi.org/10.3390/publications7020040. 61

[508] Sarven Capadisli and Amy Guy. Linked Data Notifications. https://www.w3.org/TR/ldn/, May 2017. 11.4.2

[509] Brian Butler, Elisabeth Joyce, and Jacqueline Pike. Don't look now, but we've created a bureaucracy: The nature and roles of policies and rules in wikipedia. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1101–1110, New York, NY, USA, April 2008. Association for Computing Machinery. ISBN 978-1-60558-011-1. https://doi.org/10.1145/1357054.1357227. 11.4.2

[510] Wikipedia. Wikipedia:Barnstars. *Wikipedia*, July 2022. 11.4.2

[511] Orlando de Lange, Kellie Dunn, and Nadya Peek. "Short on time and big on ideas": Perspectives from Lab Members on DIYBio Work in Community Biolabs, April 2022. 11.4.3

[512] Alexander M. Strasak, Qamruz Zaman, Karl P. Pfeiffer, Georg Göbel, and Hanno Ulmer. Statistical errors in medical research - a review of common pitfalls. *Swiss Medical Weekly*, 137(3-4):44–49, January 2007. ISSN 1424-7860. https://doi.org/10.4414/smw.2007.11587. 11.4.3

[513] Andrew W. Brown, Kathryn A. Kaiser, and David B. Allison. Issues with data and analyses: Errors, underlying themes, and potential solutions. *Proceedings of the National Academy of Sciences*, 115(11):2563–2570, March 2018. https://doi.org/10.1073/pnas.1708279115. 11.4.3

[514] Jeffrey T. Leek and Roger D. Peng. Statistics: P values are just the tip of the iceberg. *Nature*, 520(7549):612–612, April 2015. ISSN 1476-4687. https://doi.org/10.1038/520612a. 11.4.3

[515] James Heathers. The 450 Movement, September 2020. 11.4.3

[516] Katherine Quinn and Jo Bates. Resisting neoliberalism: The challenge of activist librarianship in English Higher Education. *Journal of Documentation*, 73(2):317–335, January 2017. ISSN 0022-0418. https://doi.org/10.1108/JD-06-2016-0076. 11.4.3

[517] Paul E. Meehl. Theoretical risks and tabular asterisks: Sir Karl, Sir Ronald, and the slow progress of soft psychology. *Journal of Consulting and Clinical Psychology*, 46(4):806–834, 1978. ISSN 1939-2117(Electronic),0022-006X(Print). https://doi.org/10.1037/0022-006X.46.4.806. 11.4.3

[518] Nancy Cartwright. *How the Laws of Physics Lie*. Oxford University Press, Oxford, 1983. ISBN 978-0-19-824704-3. 11.4.3

[519] Iris van Rooij and Giosuè Baggio. Theory Before the Test: How to Build High-Verisimilitude Explanatory Theories in Psychological Science. *Perspectives on Psychological Science*, page 1745691620970604, January 2021. ISSN 1745-6916. https://doi.org/10.1177/1745691620970604. 11.4.3

[520] Olivia Guest and Andrea E. Martin. How Computational Modeling Can Force Theory Building in Psychological Science. *Perspectives on Psychological Science*, page 1745691620970585, January 2021. ISSN 1745-6916. https://doi.org/10.1177/1745691620970585. 11.4.3

[521] Harold Varmus. Of oncogenes and open science: An interview with Harold Varmus. *Disease Models & Mechanisms*, 12 (3):dmm038919, March 2019. ISSN 1754-8403. https://doi.org/10.1242/dmm.038919. 11.4.4

[522] Michael Eisen. The reason we are (once again) having a fight about whether the producers of publicly available/published data should be authors on any work using said data is that we have a completely dysfunctional system for crediting the generation of useful data., November 2021. 11.4.4

[523] Michael Eisen. The same is true for people who generate useful reagents, resources and software., November 2021. 11.4.4

[524] Michael Eisen. And like everything, the real answer lies on how we assess candidates for jobs, grants, etc... So long as people treat authorship as the most/only valuable currency, this debate will fester. But it's in our power to change it., November 2021. 11.4.4

[525] Jaime A. Teixeira da Silva and Judit Dobránszki. Multiple versions of the h-index: Cautionary use for formal academic purposes. *Scientometrics*, 115(2):1107–1113, May 2018. ISSN 1588-2861. https://doi.org/10.1007/s11192-018-2680-3. 11.4.4

[526] Rodrigo Costas and Thomas Franssen. Reflections around 'the cautionary use' of the h-index: Response to Teixeira da Silva and Dobránszki. *Scientometrics*, 115(2):1125–1130, May 2018. ISSN 1588-2861. https://doi.org/10.1007/s11192-018-2683-0. 11.4.4

[527] Crossref. January 2021 Public Data File from Crossref. *Academic Torrents*, January 2021. https://doi.org/10.13003/GU3DQMJVG4. 11.4.4

[528] RENT STRIKE. Work! (Future Perfect), October 2021. 12.1

[529] Sam Biddle. Facebook Won't Say If It Will Use Your Brain Activity for Advertisements. https://theintercept.com/2017/05/22/facebook-wont-say-if-theyll-use-your-brain-activity-for-advertisements/, May 2017. 12.1.1

[530] Már Másson Maack. 'YouTube recommendations are toxic,' says dev who worked on the algorithm. https://thenextweb.com/news/youtube-recommendations-toxic-algorithm-google-ai, June 2019. 12.1.1

[531] Titipat Achakulvisut, Tulakan Ruangrong, Patrick Mineault, Tim P. Vogels, Megan A. K. Peters, Panayiota Poirazi, Christopher Rozell, Brad Wyble, Dan F. M. Goodman, and Konrad Paul Kording. Towards Democratizing and Automating Online Conferences: Lessons from the Neuromatch Conferences. *Trends in Cognitive Sciences*, 25(4):265–268, April 2021. ISSN 1364-6613. https://doi.org/10.1016/j.tics.2021.01.007. 12.1.1

[532] Konrad Paul Kording. For love of neuroscience: The Neuromatch movement. *Neuron*, 109(19):3034–3035, October 2021. ISSN 0896-6273. https://doi.org/10.1016/j.neuron.2021.07.021. 12.1.1

[533] Tara van Viegen, Athena Akrami, Kathryn Bonnen, Eric DeWitt, Alexandre Hyafil, Helena Ledmyr, Grace W. Lindsay, Patrick Mineault, John D. Murray, Xaq Pitkow, Aina Puce, Madineh Sedigh-Sarvestani, Carsen Stringer, Titipat Achakulvisut, Elnaz Alikarami, Melvin Selim Atay, Eleanor Batty, Jeffrey C. Erlich, Byron V. Galbraith, Yueqi Guo, Ashley L. Juavinett, Matthew R. Krause, Songting Li, Marius Pachitariu, Elizabeth Straley, Davide Valeriani, Emma Vaughan, Maryam Vaziri-Pashkam, Michael L. Waskom, Gunnar Blohm, Konrad Kording, Paul Schrater, Brad Wyble, Sean Escola, and Megan A. K. Peters. Neuromatch Academy: Teaching Computational Neuroscience with Global Accessibility. *Trends in Cognitive Sciences*, 25(7):535–538, July 2021. ISSN 1364-6613. https://doi.org/10.1016/j.tics.2021.03.018. 12.1.1

[534] Colin Wood. Universities are adopting cloud services at an accelerating pace. https://edscoop.com/universities-cloud-services-tambellini-group/, April 2021. 12.1.2

[535] Ariadne Conill. What would ActivityPub look like with capability-based security, anyway?, January 2019. 12.2

[536] Meatball Wiki. SoftSecurity. http://meatballwiki.org/wiki/?SoftSecurity. 12.2

[537] Ernesto Van der Sar. Comcast Throttles BitTorrent Traffic, Seeding Impossible * TorrentFreak, August 2007. 12.2

[538] Samuel Moore. Why open science is primarily a labour issue, June 2022. 12.3

[539] Ashley Wong. Student Workers at Columbia End 10-Week Strike After Reaching a Deal. *The New York Times*, January 2022. ISSN 0362-4331. 12.3

[540] Cara J. Chang and Meimei Xu. Harvard Graduate Student Union to Begin Strike at 6 a.m. *The Harvard Crimson*, October 2021. 12.3

[541] Katrina Brooker. "I Was Devastated": The Man Who Created the World Wide Web Has Some Regrets. https://www.vanityfair.com/news/2018/07/the-man-who-created-the-world-wide-web-has-some-regrets, July 2018. 12.3

[542] Leonard Kleinrock. An early history of the internet [History of Communications]. *IEEE Communications Magazine*, 48(8):26–36, August 2010. ISSN 1558-1896. https://doi.org/10.1109/MCOM.2010.5534584. 12.3

[543] Frances Sari. Guest Post: Technology, Law, and Education: A Three-Pronged Approach to Fight Digital Piracy. https://scholarlykitchen.sspnet.org/2018/04/24/guest-post-technology-law-education-three-pronged-approach-fight-digital-piracy/, April 2018. 12.4.1

[544] Björn Brembs. Is the SNSI the new PRISM?, October 2020. 12.4.1

[545] NISO. NISO RP-27-2019, Recommended Practices for Improved Access to Institutionally-Provided Information Resources: Results from the Resource Access in the 21st Century (RA21) Project | NISO website, June 2019. 12.4.1

[546] SNSI. Cybersecurity Landscape - Protecting the Scholarly Infrastructure, October 2020. 12.4.1

[547] SeamlessAccess in Action - SeamlessAccess. https://seamlessaccess.org/work/. 12.4.1

[548] Life Sciences Professional Services. Emerging trends for pancreatitis in the scientific literature, September 2021. 12.4.1

[549] Elsevier. Topic Prominence in Science - Scival | Elsevier Solutions. https://www.elsevier.com/solutions/scival/features/topic-prominence-in-science. 12.4.1