Ecological Origins of Complex Multicellularity: Niche Construction Sustains Morphological Innovation

Matthew Andres Moreno[1,2,3,4,†], Santiago Rodriguez Papa[5,7], Luis Zaman[1,2,4], Emily Dolson[5,6,7], Charles Ofria[5,6,7]

[1]Department of Ecology and Evolutionary Biology [2]Center for the Study of Complex Systems
[3]Michigan Institute for Data and AI in Society [4]University of Michigan, Ann Arbor, USA
[5]Department of Computer Science and Engineering [6]Program in Ecology, Evolution, and Behavior
[7]Michigan State University, East Lansing, USA
[†]morenoma@umich.edu

Abstract

Complex multicellularity exemplifies the surprising constructive power of evolution. While natural history has produced extensive innovation in form and function along some multicell lineages, evolutionary processes underlying this complexity remain opaque. As a general, tractable model, we investigated this issue using digital multicells: assemblages of cell-like computer programs that collectively replicate, mutate, and compete. Among evolutionary outcomes, we observed an extreme outlier in complexity measures with patterned multicellular growth and multi-stage life history. Case study analysis of this lineage revealed biogenic niche construction, which imposed crowding interference on alternate morphologies. Neutralizing biogenic context in replay experiments revealed that this case study lineage, in fact, exerted biotic selection to maintain its own complexity. Such ecological lock-in of complexity appears across replicates, with correlation between niche construction and functional complexity matching or exceeding the effects of adaptation and drift. These findings show how biotic selection can act as a powerful complexity ratchet, and concretely demonstrate a possibly critical role of eco-evolutionary feedback in early evolution of biological multicellularity.

Introduction

Why should complex multicellular organisms exist, when simpler unicellular organisms grow biomass with substantially greater efficiency[1]? Adaptations for somatic division of labor and economies of scale[2–5] alone cannot fully explain complex multicellularity, as they recoup only a portion of metabolic costs. Instead, some argue complex multicellular organisms to reflect diffusion or a random walk from a lower complexity bound[6–11]. In this framing, modest population sizes among multicells act as a ratchet to accumulate near-neutral redundancies and superficialities that, through establishment of mechanistic interdependencies, become locked in to essential functionality[12–15].

An alternate framing of adaptation, however, may complement such explanations: niche construction — where feedback from organisms reshapes environmental context[16–18], through abiotic ecosystem engineering or direct production of biogenic structure. Where advantaged by new environmental context, complex traits may be maintained by selection[19–22]. In the case of multicellularity, combinatorially vast morphological possibility opens a Pandora's box of potential biotic interactions. For instance, it has long been recognized that body size may provide means to resist predation[4,23,24]. Indeed, such effects arise in experiments that introduce known microbial predators to populations of naive prey[25–28]. To date, however, the role of *de novo* niche construction in promoting complexity during early multicell evolution remains to be tested experimentally[29–35].

Leveraging a digital model system to study synthetic instances of multicell evolution *in silico*, we observe spontaneous niche construction effects and test their relationship to evolved complexity. Model digital organisms act analogously to biological cells: replicating, mutating, competing, and thus evolving[36]. Though artificial by nature, such tractable models can nonetheless illuminate principles general to biological life and beyond[37–39]. For instance, niche construction and Evolutionary Transitions in Individuality (ETI) — such as multicellularity — have guided work among computer scientists to realize the "open-ended" creativity of evolution in abiotic systems[40–47].

The next section describes our experimental system, detailing how model digital organisms replicate, form multicells, and mediate interactions between them. Following this, we examine the history of a case study lineage: showing how genotype complexity, phenotype complexity, and morphological novelty can arise in early multicell evolution with much greater detail than previously possible. Competition assays reveal that our case study lineage became fit when abundant — in other words, when shaping its own environmental context. We next test whether niche construction exerts self-acting biotic selection. We find it does, and evolutionary replay experiments further show this self-acting biotic selection to lock in evolved complexity. Finally, we conclude by examining broader associations between niche construction and complexity across many independent populations.

Experimental System

DISHTINY is a software platform for conducting multicell evolution experiments using digital organisms[48]. These digital organisms populate a 2D grid, and reproduce via asexual budding to a chosen adjacent position — supplanting a chosen neighbor.

To introduce multicellular structure, DISHTINY allows parent cells to distinguish budded offspring between growing the parent's group (i.e., agglomeration; Figure 1d) and founding a new group

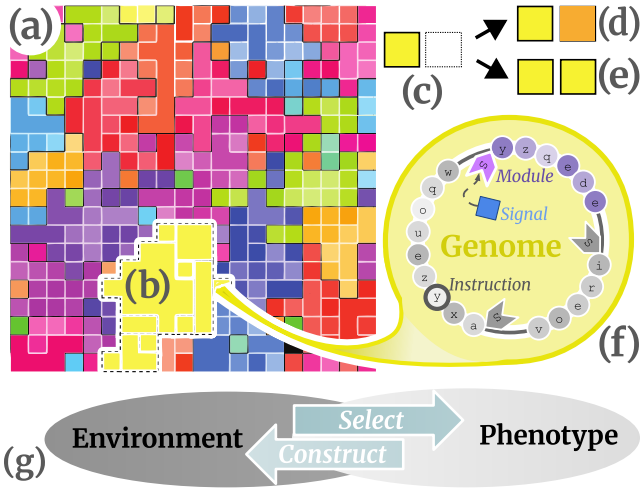


Figure 1: **Digital multicell model.** Cell-like digital organisms populate a toroidal grid (*a*), forming clonal multicell groups (color-coded patches; *b*) that compete to occupy limited space. Cell replication occurs by asexual budding, replacing a chosen neighbor (*c*) — either growing the parent's existing group (*d*) or founding a new group (*e*). Heritable genetic programs analogous to biological genomes control cell behaviors by activating submodules in response to local signals (e.g., neighbor actions, incoming messages, etc.; *f*). Because multicells constitute each other's environmental context, potential arises for biotic selection feedbacks (*g*).

(i.e., propagule generation; Figure 1e).* As such, an implicit single-cell bottleneck arises in group establishment[50]. Cells become eligible to reproduce by passively accruing energy, and we reward group formation by boosting energy inflow. Importantly, though, no extrinsic steering restricts cooperation or conflict among cells within groups. As a practical matter to ensure generational turnover, groups are split into unicellular propagules after a fixed duration.

Neighboring cells interact by competing for space, sharing energy, exchanging messages, and sensing each other's state. Within-group cell-cell interactions allow dynamic development and patterning processes[29,49], while between-group cell-cell interactions build an environment of free-form interface competition — creating feedback between evolved phenotypes and the environment (Figure 1g).

Cell behavior arises from activation of genome subprograms in tagged modules[51]. These modules selectively activate in response to incoming signals[52,53] — triggered by updates to the local environment, cell-cell messages, or expression of other modules (Figure 1f). To facilitate plasticity and differentiation, cells may dynamically regulate their genome modules' signal-matching priorities[54].

Within modules, programs consist of a linear instruction sequence. Available instructions include arithmetic operations, conditional control flow operations (e.g., looping, branching, etc.), input operations to sense local state, and output operations to trigger actions (e.g., resource sharing, reproduction, etc.). The full set of 56 available instruction types is described in supplementary material, alongside details for full simulation logic and parameter settings.

*In earlier work, we found that more case studies of interest arose with two nested levels of group membership, as opposed to a single, unnested level of group membership[49]; we continue this approach here. Cells may thus retain offspring within both groups, expel offspring from both groups, or expel offspring from the innermost group only.

Results

Case-study Lineage

Figures 2a to 2c compare genotype and phenotype complexity across our 40 evolutionary replicates, measured using genetic knockouts and phenotype interference assays. Among replicates, extreme outlying complexity arose in one replicate with mean sustained complexity exceeding 3× IQR in both complexity measures (blue highlight, Figure 2c).

This lineage also exhibited patterned growth and multi-stage life history. Growing from unicellular inoculum (Figure 2d), groups first elongate horizontally to establish an asymmetrical cigar-like morphology (Figure 2e and 2f). Then, in a secondary phase of growth, groups expand in the vertical direction to establish an irregular "box-like" morphology (Figure 2g and 2h).

To investigate how complexity arose and stabilized in this lineage, we investigated its history in detail as a case study — reported next.

Morphology We qualitatively categorized multicell life histories across each of 100 completed passage steps, identifying nine morphologies: labeled "*b*" through "*j*" (Figure 3a, Supplementary Tables 1 and 2, and Supplementary Figure 33).†

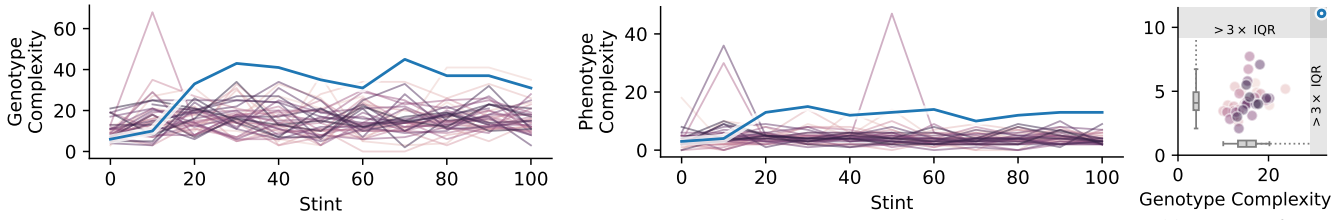
Early sampled genomes mostly grew as unicells and small groups up to 4 cells (morph *b*), although our earliest sample from the case study lineage formed oversized groups through unchecked cell proliferation (morph *c*, stint 2).

The case study lineage's cigar-like patterning first arose at stint 15 (morph *e*). Subsequently, the secondary phase of box-like vertical growth arose at stint 45 (morph *g*). However, around stint 60, cigar-like morph *e* began to be sampled as a reversion from morph *g*. Finally, at stint 100, we first observed morph *j* as shown in Figure 2d to 2h. This morph presented the two-phase growth pattern of morph *g*, except that some groups produced cigar-like daughter propagules during the secondary growth phase.

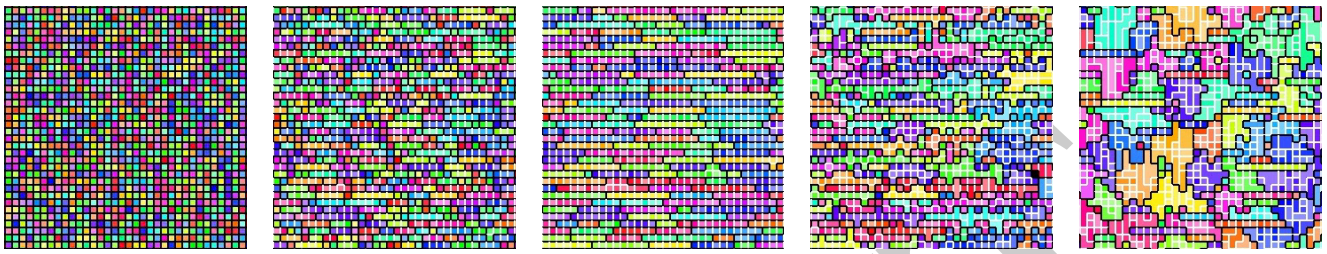
Complexity Next, we assayed genotype and phenotype complexity across lineage history (Figure 3c and 3d, Supplementary Figure 54).

In several instances, complexity increased abruptly with new morphology. For instance, at stint 14, morph *d* (irregular clumps of 10 to 15 cells) coincided with a spike in genotype complexity from 28 to 44 sites. At stint 15, cigar-like morph *e* grew phenotype complexity over threefold from 5 to 17 interactions.

†Descriptions of morphs *d*, *h*, *i*, and *f* — which were sampled rarely — appear in supplementary material. Morph *a* arose on an early independent lineage soon, driven extinct.



(a) genotype complexity, case study (blue) vs. others (b) phenotype complexity, case study (blue) vs. others (c) mean g.c./p.c.



(d) update 8 (e) update 32 (f) update 104 (g) update 840 (h) update 952

Figure 2: **Case study strain.** Compared to other replicates, case study 16005 sustained extreme outlying complexity (upper panels 2a to 2c) and exhibited unique multi-stage patterned growth (lower panels 2d to 2h). Genotype complexity counts genome sites contributing to fitness; phenotype complexity counts distinct cell input/output operations contributing to fitness — both determined via knockout/interference trials. Snapshots show example life history (morph "j," stint 100; Supplementary Table 1). Hue denotes, and black borders divide, multicell groups; white borders reflect inner structure. From initial unicell groups (2d), cells replicate to form horizontal "cigar-like" bodies (2e and 2f). In a delayed second stage of growth, groups expand vertically to produce a "box-like" final morphology (2g and 2h). During this secondary phase of growth, some groups establish cigar-like propagules (2h). Video animation: `https://hopth.ru/gb`.

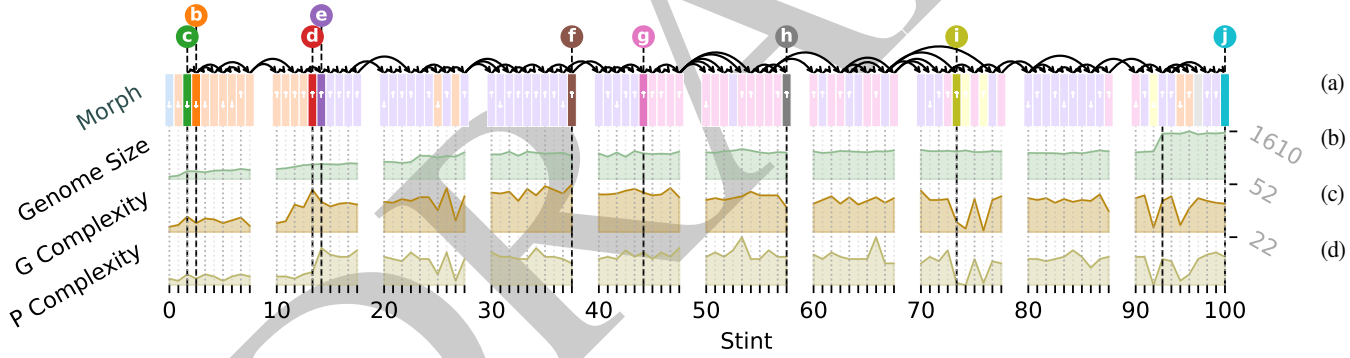


Figure 3: **Complexity, novelty, and adaptation across case study lineage history.** Time shown left-to-right across 100 sequential 3-hour passages. Morph is qualitative classification of multicell life history, with lollipop markers flagging novelties (3a). Black arrows along top trace ancestry, which had few deep branches. Vertical white arrows in morph keys denote fitness gain (up) or loss (down), $p < 0.005$. Genome size gives site count (3b). G complexity is genotype complexity — the number of genome sites deleterious when knocked out (3c). P complexity is phenotype complexity — the number of distinct adaptive cell input/output operations (3d). Supplementary Figures 35 and 39 to 41 report genome module counts and alternate fitness assays.

In contrast to these abrupt innovations, genotype complexity also grew in a more gradual phase from 27 to 55 sites between stints 16 and 39.

Across the case study lineage's history, split regression analysis suggests genotype complexity developed in two distinct regimes (Davies test, $p < 10^{-14}$, adj. $R^2 = 0.43$): increasing until stint 34 (Wald test, $R^2 = 0.59$, $p < 10^{-6}$), and then decreasing (Wald test, $R^2 = 0.24$, $p < 10^{-4}$) (Supplementary Figure 50d).

In the later period, it is possible that genotype complexity simply became harder to detect due to gains in mutational robustness. To test this possibility, we "skeletonized" genomes by removing unexpressed sites then repeated our single-site knockout complexity assay. This "deep" genotype complexity measure remained generally similar, but the trend of decreasing complexity was weakened ($p = 0.06$) — especially from stint 40 onwards ($p = 0.56$) (Supplementary Figures 50e and 50f). Thus, the evolutionary trajectory of genotype complexity appears to be initial growth followed by stabilization.

Adaptation Next, we examined fitness changes along the case study lineage. For this purpose, we competed samples from successive passages to test for fitness gains at each step.

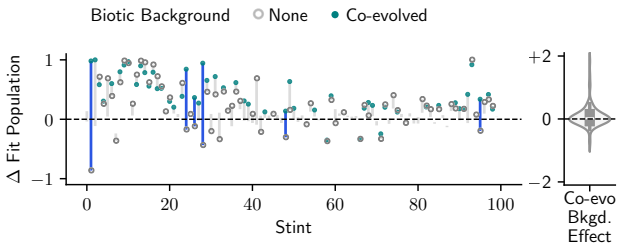
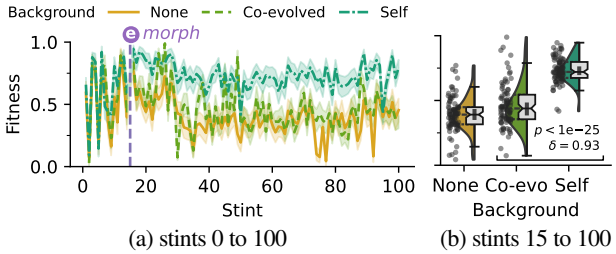


Figure 4: **Biotic context drives fitness trade-offs.** Adaptation during each stint was tested by competing bookend population samples. Competition outcomes are normalized between 1 and −1 (fitness gain vs. loss). Vertical lines span difference with and without co-evolved biotic context (filled vs. hollow markers). Markers omitted where no significant fitness change detected ($\alpha = 0.005$). Blue segments highlight where otherwise harmful traits become beneficial in co-evolved context. Right panel shows distribution of effects on competition outcome when introducing co-evolved context. Prefatory co-evolved context used; further detail in Supplementary Figure 44.



(a) stints 0 to 100 (b) stints 15 to 100

Figure 5: **Patterned growth creates biogenic niche.** Around stint 15, fitness of case study strain in self-context diverges significantly above other contexts. For background type "None," competition is initialized 50%/50% between compared strains (Figure 9a). Otherwise, competition is initialized 25%/25%/50% between compared strains and background (Figure 9b). Shaded bands are bootstrap 95% CI. Mann-Whitney test reported.

Across 100 such assays, 57 sampled genotypes outcompeted their predecessor by a significant margin (Figure 3a, Supplementary Figures 35 and 36). Fitness gains became somewhat more sparse at later time points — a notable exception, though, being a strong fitness gain from 500-site genome expansion at stint 93.

Besides the 57 adaptive steps discussed above, 23 steps were neutral and — surprisingly — 20 were deleterious (Figure 3a). To assess whether these apparent fitness losses could be explained by within-population variation, we repeated assays to compete whole-population samples rather than monoclonal isolates. However, whole-population competition results remained similar: 50 assays indicated significant fitness gain, 34 were neutral, and 16 indicated significant fitness loss (Supplementary Figure 41).

Although rarer than fitness gains ($p < 10^{-4}$; exact Binomial test), nearly one in five lineage intermediates thus appeared maladaptive. While such maladaptation could occur through drift effects (carrying capacity supported only around 500 multicell groups in the case study lineage), it could also arise from trade-offs imposed by exogenous environmental factors[55] — a possibility we investigate next.

Adaptation to Co-evolved Ecological Context

Recall that our experiment maintained long-term co-existence of an additional independent lineage alongside the case study strain. To test whether co-evolutionary interactions with this background strain could account for apparent maladaptations, we repeated earlier competitions to incorporate the co-evolved lineage as biotic context (Figure 9b).

After incorporating the co-evolved lineage into our measure of fitness, whole-population maladaptations decreased from 16 to 3 (Figure 4). Lower assay sensitivity might partially explain this result, as no fitness effect was detected in nearly twice as many cases (23 vs. 53). However, in 6 cases, what had appeared as fitness losses were actually fitness gains present the co-evolved lineage.

Having found co-evolved biotic context to exert at least transient selection on the case study strain, we next sought to more closely assess these ecological interactions — and if they contributed to complexity in the case study strain.

Biotic Context Influences Fitness

To characterize the case study lineage's co-evolutionary interactions, we first set out to test if these interactions were mutualistic or antagonistic.

As a proxy for absolute fitness, we assembled a biotic "measuring stick" at each stint by drawing genome samples from each of our 40 independent replicates. Competitions against members of the pool provided a fitness measure between 0 (all losses) and 1 (all wins).

By this measure, case study lineage fitness was volatile during early evolution, but settled near or below 0.5 from stint 30 onwards (solid line, Figure 5) — meaning it underperformed most other replicates.

To test how biotic context affected fitness, we repeated these "measuring stick" competitions to incorporate the co-evolved lineage (dashed line, Figure 5). Fitness in co-evolved context closely tracked baseline fitness, with strong fitness advantage in several transient episodes and a strong fitness penalty at stint 30.

These results — in conjunction with reciprocal experiments testing how the case study strain influences its co-evolved counterpart (Supplementary Figure 51) — suggest a relationship of transient commensalist interaction, rather than systematic cooperation or antagonism.

For completeness, we additionally tested fitness where the case study lineage served as its own biotic context (dot-dash line, Figure 5). (In effect, these conditions boosted initial concentration of the case study strain from 50% to 75%.) Surprisingly, this self-context produced a strong, sustained fitness benefit from stint 15 onward (Mann-Whitney test, $p < 10^{-25}$; Cliff's delta $\delta = 0.93$; 5b). The strength of this effect — and its origination with cigar-like morph e — led us to consider an alternate complexity driver: self-acting biotic selection driven by the case study strain itself.

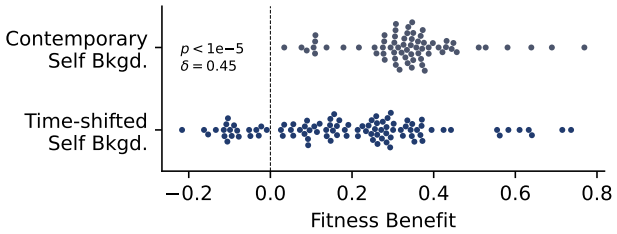


Figure 6: **Adaptive tracking of self-context.** Fitness benefit from exact self-context (top) exceeds benefit from time-shifted self-context (drawn from stints 33, 40, 50, 66, 75, 90, and 99; Mann-Whitney test). Supplementary Figure 55 further details time-shift analysis.

Self-acting Selection in Case Study Strain

Having identified that the case study strain's cigar-like biogenic structure provided a competitive advantage against other strains, we next sought to assess whether this niche construction produced self-acting selection. Such selection, were it to have occurred, could then be tested for connections to the case study strain's complexity.

To detect self-acting selection, we tested whether the case study strain's self-benefit exhibited adaptive tracking over time. For this purpose, we repeated competitions against our "measuring stick" of cross-replicate comparators. However, for these new assays, we used time-shifted self-context. That is, for each stint we performed competitions in the presence of case study lineage samples from fixed timepoints after the emergence of morph e — rather than contemporary self-context, as before.

Compared to time-shifted context, contemporary self-context did indeed provide substantially greater fitness benefit (Mann-Whitney test, $p < 10^{-5}$, Cliff's delta $\delta = 0.45$; Figure 6). Nonetheless, time-shifted backgrounds still provided some benefit in 78 of 98 tested cases. Thus, while the underlying mechanism of niche construction appears to have remained generally stable, traits responsible for exploiting this niche adapted over time to track changes in self-context.

Self-acting Selection Maintains Complexity in Case Study Strain

Seeing evidence that the case study lineage's biogenic structure exerted selection, we next turned to test whether this selection favored complexity. To address this question, we carried out evolutionary replay trials, comparing complexity outcomes with and without self-context.

A key practical challenge presents itself, however: some means is needed to remove — or at least meaningfully abate — self-context. This problem is nontrivial, because any amount of the case study strain inherently introduces some degree of self-context. To overcome this challenge, we mutationally attenuated the case study strain when seeding replay trials. Mutagenic dose (8 mutations) was chosen such that most, but not all, seeded genomes were substantially disrupted. As an additional benefit, applying mutation also enhanced standing variation of seeded populations — improving sensitivity to detect selection on complexity (Figure 7a).

*For legibility, Figure 7c clips an extreme outlier with 142 site genotype complexity observed from self-context replay (Supplementary Figure 47).

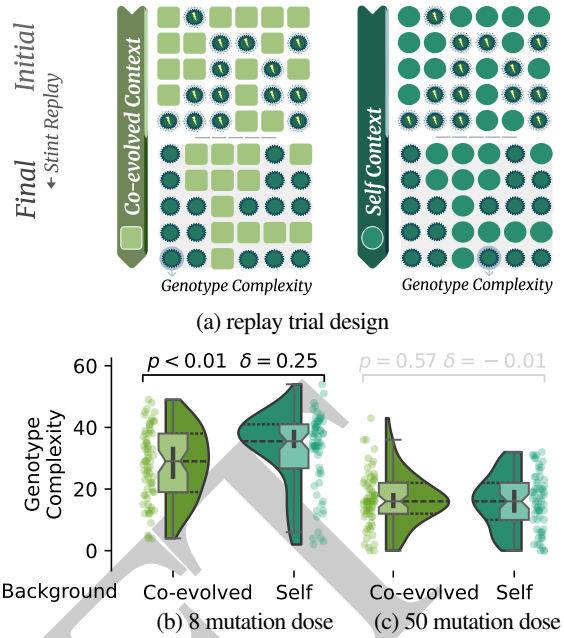
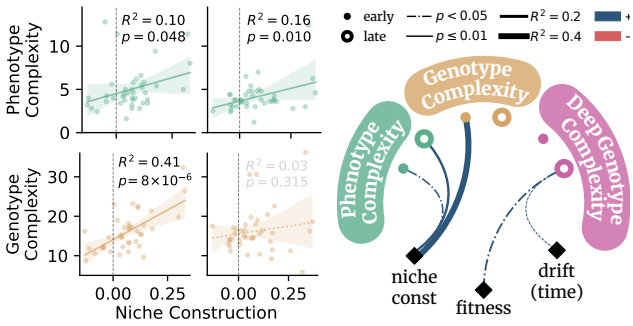


Figure 7: **Constructed niche selects to maintain genotype complexity.** We performed short one-stint replay trials on the case study strain, in either co-evolved or self context. To enhance standing variation and weaken inherent self-context effects, replay genomes were subject to moderate mutagenesis — dosed to maintain some functionally intact genomes (7a). Replays in self context maintained higher genotype complexity than co-evolved context (Mann-Whitney test, 7b). By contrast, more intense mutagenesis did not reproduce this effect, suggesting specificity to preserve already-evolved complexity (Mann-Whitney test, 7c*). Supplementary Figure 46 shows replay results for stints 0 to 34.

Replays were conducted under both a "self context" treatment and a "co-evolved context" treatment, seeded with 50% intact case study lineage genomes or co-evolved genomes. To target the complexity maintenance period identified by split regression analysis, replays were conducted from stint 33 onwards — thus, providing a sample of 67 1-stint replays per treatment. As the endpoint of each replay, we sampled a genome descended from the mutagenized case study inoculum and measured its complexity.

Across replays, self context preserved higher genotype complexity than co-evolved ecological context (Mann-Whitney test, $p < 0.01$, $\delta = 0.25$; Figure 7b) — biotic selection exerted by the case study strain indeed acted to stabilize its own complexity.

As a control treatment, we repeated replay experiments with a higher dose of 50 mutations — sufficient to ensure substantial disruption to seeded genomes. Under these conditions, complexity outcomes were indistinguishable between self and co-evolved context treatments (Figure 7c). Thus, the case study strain's biotic selection appears to act specifically in maintaining its own evolved complexity. In other words, evolved traits acted to "lock



(a) early ($t \leq 50$) (b) late ($t > 50$) (c) drivers of complexity

Figure 8: **Niche construction co-occurs with phenotype and genotype complexity across replicates.** To test generality of the association between niche construction and complexity observed in the case study strain, we tested for correlation across replicates. We screen for niche construction via self advantage, the difference between fitness in self context versus co-evolved context. Across early stints, self adaptation co-occurred with both genotype complexity and phenotype complexity (8a). Association with genotype complexity dissipated in later stints, but association with phenotype complexity strengthened (8b). During the later period, both lineage fitness ($p < 0.03$, $R^2 = 0.12$) and lineage age ($p < 0.05$, $R^2 = 0.02$) correlated positively with deep genotype complexity, which includes genome sites otherwise resilient to single knockouts (8c). Pearson correlation and two-tail Wald tests reported, detailed in Supplementary Figure 56.

in" complexity by advantaging themselves, as opposed to simply cultivating generic conditions conducive to complexity (e.g., weakening selection).

Self Adaptation Drives Functional Complexity

Having found that niche construction helped maintain evolved complexity in the case study lineage, we next sought to assess whether this pattern manifested more broadly across replicates.

To screen for analogous niche construction effects, we measured self adaptation across all 40 replicates — calculated as the difference between "measuring stick" fitness in self versus co-evolved contexts. Other stints also exhibited self-advantage (Supplementary Figure 48) and, across replicates, mean self adaptation correlates positively with mean genotype and phenotype complexity (Wald test; $p = 0.01$ and $p = 5 \times 10^{-3}$; Supplementary Figure 56).

For genotype complexity, correlation with self-adaptation focused in early evolution (stints 10 to 50, Figure 8a; Wald test, $p = 8 \times 10^{-6}$; $R^2 = 0.41$), and dissipated later (stints 60 to 100, Figure 8b; Wald test, $p = 0.3$). By contrast, phenotype complexity — which reflects the richness of evolved cell-cell interactions — correlated significantly with self adaptation across both time periods (Wald tests; $p = 0.05$, $p = 0.01$; $R^2 = 0.1$, $R^2 = 0.16$; Figure 8a and 8b).

In line with both adaptationist and neutralist hypotheses for the evolution of complexity, deep genotype complexity — which includes sites otherwise masked by robustness — correlated with both fitness and elapsed duration during later evolution (panel 8c). Drift effects, in particular, manifested robustly over the longer whole-experiment timespan (Wald test, $p \leq 1 \times 10^{-6}$; $R^2 = 0.06$; Supplementary Figure 56).

Thus, functional complexity appears to be significantly influenced by density-dependent self-adaptation — a signature of biogenic niche construction in our model system. Although fitness-critical genotype complexity only associated with niche construction during early evolution, niche construction more lastingly stabilized phenotype complexity. It is likely that later selection for robustness eroded early accrual of mutationally-sensitive genotype complexity — as seen above in split-regression analysis of the case study strain (Supplementary Figure 50).

Discussion

Novelty and complexity are not inevitable outcomes of Darwinian evolution[56]. Instead, how and why some multicellular lineages evolve novel traits and complex functionality merits explanation. Here, using a digital organism model, we found that complex traits underlying patterned growth can reshape environmental context to favor their own maintenance. That is, complex multicellularity can be stabilized by niche construction effects. Examining a high-complexity lineage with multi-stage patterned growth, we found that collective biogenic structure arising from asymmetric multicell bodies selected to maintain that asymmetry by crowding other morphologies. This pattern arose broadly across replicates, with those exhibiting specific adaptation to self-context tending to exhibit greater functional complexity.

As a model, digital organisms can reflect the dynamic plasticity, rugged fitness landscapes, flexible information processing behavior, and Darwinian evolution characteristic of natural life[36]. Nonetheless, given profound differences between the two, digital findings are only a first step to further tests of generality[57,58] — like any model system. For instance, while the particular literal crowding mechanism seen in our case study is likely substrate-specific, biological multicells may encounter analogous scenarios via alternate means for biotic selection and niche construction.

To remain computationally tractable, our experiments also forgo the vast timescales and population sizes characteristic of biological evolution. Neutral processes, in particular, should exert a cumulative influence on complexity that intensifies over these longer timescales. In future work, high-performance computing hardware — leveraged here only at single-machine scale — may soon make such explorations feasible[59–61].

Despite these limitations, reported experiments open a new window into possible ecological dynamics of early multicellularity — a transition that simultaneously unlocks vast morphological possibilities and vast niche opportunities[62]. While co-evolutionary arms race dynamics are well-recognized drivers of body enlargement and other complex traits[63,64], our experiments reveal an alternate ecological pathway to complexity: adaptive entrenchment through niche construction.

This result suggests that the logic of constructive neutral evolution may extend to the ecosystem level[15]. Rather than lock-in through within-organism epistatic dependencies, this regime's

ratchet mechanism of generative entrenchment would instead act via biotic dependencies among organisms[20]. Notably, such a mechanism combines the active driving force of selection with passive lock-in of arbitrary contingencies, thus blending adaptationist and neutralist hypotheses. Such effects need not only be self-acting — as appeared to occur most readily in our model system — and it is interesting to speculate how such effects may generalize to networks of interacting lineages. Even in cases where niche construction is only transiently stable, scaffolded biotic traits may nonetheless persist by entraining lasting lower-level, within-organism epistatic entrenchment.

Methods

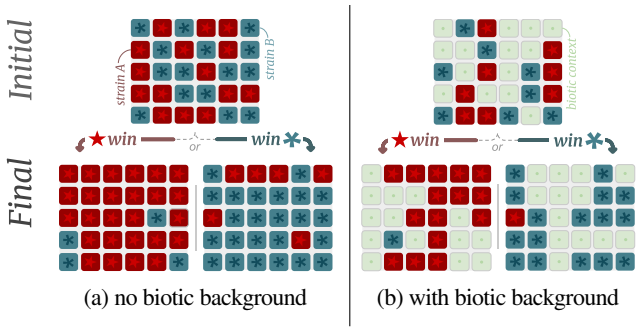
Evolutionary Replicates

Ancestor populations for evolution experiments were randomly generated 100-instruction genomes, typically encoding on the order of 5 signal-response modules. Configured population size of 14,400 cells reliably sufficed that some founding ancestors were viable replicators. When replicating, all offspring were subject to possible mutation: point mutations, single-site insertions and deletions, or multi-site duplications and deletions (Supplementary Section D). Mutations could also introduce new signal-response modules by creating new activation entrypoints.

After initialization, competition for space drove most founding strains extinct. To ensure a heterogeneous biotic environment within each experimental replicate, we imposed a simple diversity balancing scheme: any strain exceeding half of available carrying capacity incurred an escalating energy decay penalty. Thus, replicates converged to maintain at least two founding strains. Among these, we arbitrarily chose one strain for primary study — the "focal" strain; another was sampled as the "background" (or "co-evolved") strain.

For compatibility with our compute cluster's scheduling system, evolutionary replicates were passaged between fresh simulation runtimes in three hour increments, which we refer to as "stints." Each transfer carried over all population genomes, but no other simulation state. As a benefit, this protocol ensured strains remained capable of growth from well-mixed inoculum. Supplementary Figure 14 summarizes our serial transfer and diversity maintenance procedures.

We conducted 100 passages, amounting to just over 12.5 days of compute time per each of 40 replicates. Across replicates, between 5.4M and 9.9M simulation time steps elapsed (mean 7.4M; SD 1.1M). During this time, between 11.7k and 26.9k cell generations elapsed (mean 18.5k; SD 3.4k) and between 1.2k and 3.4k group generations elapsed (mean 2.3k; SD 533). In the replicate producing our high-complexity case study, 7,565,309 simulation time steps elapsed — accreting 11,713 cell generations and 1,325 group generations. To accelerate the experiment, quadrants of simulation space were distributed across four asynchronous hardware threads for parallel evaluation[65]. For this reason, phylogeny analyses relied on parsimony-based reconstruction rather than exact tracking[66].



(a) no biotic background (b) with biotic background

Figure 9: **Fitness assay design.** Fitness measurement relies on competition trials. Panel 9a overviews basic design: compared strains are mixed in equal proportion, simulation runs for a fixed duration, and then end-state competitor abundances are compared. Panel 9b shows "biotic background" design — used to test context-sensitivity of fitness. Here, the population is half seeded with a "bystander" strain, which interacts with competitors but is excluded from end-state comparison of competitors.

Measuring Fitness

Like nature, cell behavior and fate in DISHTINY emerges from stochastic interaction between expressed genetic traits and the local environment. Fitness thus is not readily determined *a priori*[67] and must instead be measured *in situ*. Further complicating matters, dense biotic interaction between groups prevents reliable fitness estimation on the basis of a single cell or group in isolation[68].

Therefore, to assess fitness, we rely on competition assays.* In these competition assays, available space is seeded with an even mix of compared strains. Then, after a 10 minute runtime, we evaluate the abundance of compared strains (Figure 9a).† To assess significance, we repeat this procedure 20 times and apply a two-tailed binomial exact test to competition win/loss counts.

For some analyses, we also used a "biotic background" competition design to integrate environmental context — similar in principle to mesocosm experiments[69]. Biotic background assays were seeded by mixing two competitors with a "bystander" strain in 25%/25%/50% proportion. The bystander strain participates fully in the competition environment, but is excluded from end-state comparison of competitor abundances (Figure 9b). In some cases, we used this biotic background competition design to test frequency-dependent selection effects: prevalence of one competitor strain was updosed by assigning a facsimile as bystander — effectively emulating a replacement series experiment[70].

Measuring Complexity

Defining and quantifying complexity remains a nuanced, unresolved topic[71–79]. Here, we consider complexity as evolved

*Supplementary material also reports doubling-time fitness assays, which measure rate of expansion into empty space (Supplementary Figure 37). However, as our evolutionary trials begin and remain near carrying capacity, we did not find these assays informative of ability to compete for occupied space.

†As a point of reference, in the high-complexity lineage this runtime elapsed between 8,000 update cycles (stint 0) and 2,000 update cycles (stint 100). Further detail in Supplementary Figures 21 to 23.

structure that is functionally adaptive. As a measure of phenotype complexity, we count cell input/output interfaces contributing to fitness. This measure describes the richness of behavior across cell interfaces[80], to capture dynamic interaction essential to complex multicellularity[81–84]. To measure genotype complexity, we count genome sites contributing to fitness[37,85] as a proxy for the richness of control structures directing phenotypic interactions.* For tractability, our implementations of these complexity measures rely on pragmatic, coarse assay designs — discussed next.

We measure phenotype complexity via manipulation interventions across a suite of wildtype-vs-degraded competition trials[86]. Each trial de-contextualized (i.e., spatially shuffled) one of the 105 available readable "sensor" cell inputs or one of the 18 available writable "actuator" cell outputs. Phenotype complexity trials also screened for message-passing functionality by intercepting and reflecting portions of message traffic.

To measure genotype complexity, we used single knockouts to screen functionality of individual genome sites. Because combination knockouts were not tested, this measure is limited in detecting sites with compensatory redundancy or small fitness effects below the competition sensitivity threshold[87]. Some such sites are captured by an extended "deep" genotype complexity measure, which applies knockouts to a "skeletonized" genome stripped of unexpressed sites.

To reduce workload for measuring complexity across many replicates and timepoints, we evaluated single competition outcomes against wildtype-vs-wildtype control trials and pre-filtered non-expressed genome sites. Supplementary materials further detail these procedures, including quality control measures.

Data Availability

Project data is archived under a CC-By Attribution 4.0 International license via the Open Science Framework at `https://osf.io/prq49`[88].

Code Availability

Project code is archived via Zenodo[89], available open source under MIT licenses. Core DISHTINY software is hosted on GitHub at `https://github.com/mmore500/dishtiny`. This project benefited significantly from open-source scientific software and data standards[65,90–101].

Author Contributions

Conceptualization: M.A.M., C.A.O.
Methodology: M.A.M., L.H.Z., E.L.D., C.A.O.
Software: M.A.M., S.R.P.
Writing: M.A.M. (lead), S.R.P. (supporting contributions)
Supervision: C.A.O., L.H.Z., E.L.D.

Competing Interests

The authors declare no competing interests.

*Promisingly, we find moderate correlation between these measures across replicates and, within the case study lineage, also find moderate correlation in evolutionary changes to these measures (Supplementary Figures 52 and 53).

Acknowledgements

Thanks to members of the DEVOLAB, in particular Katherine Perry for help implementing DISHTINY visualizations and Alexander Lalejini for help with SignalGP.

This research was supported in part by NSF grants DEB-1655715 and DBI-0939454 as well as by Michigan State University through the computational resources provided by the Institute for Cyber-Enabled Research (ICER). Thank you to ICER staff, including Nicholas Panchy, for their assistance. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1424871. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

This material is based upon work supported by the Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, a Schmidt Sciences program. M.A.M. completed portions of this work while affiliated with the BEACON Center at Michigan State University.

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research (ASCR), under Award Number DE-SC0025634. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

References

[1] Lynch, M. (2024). The bioenergetic cost of building a metazoan. *Proceedings of the National Academy of Sciences*, 121(46). `https://doi.org/10.1073/pnas.2414742121`

[2] Saunders, P. & Ho, M. (1976). On the increase in complexity in evolution. *Journal of Theoretical Biology*, 63(2), 375–384. `https://doi.org/10.1016/0022-5193(76)90040-0`

[3] Corning, P. A. & Szathmáry, E. (2015). "synergistic selection": A darwinian frame for the evolution of complexity. *Journal of Theoretical Biology*, 371, 45–58. `https://doi.org/10.1016/j.jtbi.2015.02.002`

[4] Niklas, K. J. (2014). The evolutionary-developmental origins of multicellularity. *American Journal of Botany*, 101(1), 6–25. `https://doi.org/10.3732/ajb.1300314`

[5] Bell, G. & Mooers, A. O. (1997). Size and complexity among multicellular organisms. *Biological Journal of the Linnean Society*, 60(3), 345–363. `https://doi.org/10.1111/j.1095-8312.1997.tb01500.x`

[6] Gould, S. J. (1988). Trends as changes in variance: a new slant on progress and directionality in evolution. *Journal of Paleontology*, 62(3), 319–329. https://doi.org/10.1017/s0022336000059126

[7] Gould, S. J. (1990). *Wonderful life*. WW Norton.

[8] Gould, S. (2011). *Full House*. Harvard University Press.

[9] McShea, D. W. & Brandon, R. N. (2010). *Biology's First Law: The Tendency for Diversity and Complexity to Increase in Evolutionary Systems*. University of Chicago Press.

[10] Lehman, J. & Stanley, K. O. (2010). Revising the evolutionary computation abstraction: minimal criteria novelty search. *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, Gecco '10, 103–110. https://doi.org/10.1145/1830483.1830503

[11] Miconi, T. (2008). Evolution and complexity: The double-edged sword. *Artificial Life*, 14(3), 325–344. https://doi.org/10.1162/artl.2008.14.3.14307

[12] Lynch, M. & Conery, J. S. (2003). The origins of genome complexity. *Science*, 302(5649), 1401–1404. https://doi.org/10.1126/science.1089370

[13] Smith, J. M. (1972). *Time in the Evolutionary Process*, 200–205. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-65387-2_14

[14] Stoltzfus, A. (1999). On the possibility of constructive neutral evolution. *Journal of Molecular Evolution*, 49(2), 169–181. https://doi.org/10.1007/pl00006540

[15] Muñoz-Gómez, S. A., Bilolikar, G., Wideman, J. G., & Geiler-Samerotte, K. (2021). Constructive neutral evolution 20 years later. *Journal of Molecular Evolution*, 89(3), 172–182. https://doi.org/10.1007/s00239-021-09996-y

[16] ODLING-SMEE, F. J., LALAND, K. N., & FELDMAN, M. W. (2003). *Niche Construction: The Neglected Process in Evolution (MPB-37)*. Princeton University Press. http://www.jstor.org/stable/j.ctt24hqpd

[17] Odling-Smee, F. J. (1988). Niche-constructing phenotypes. *The Role of Behavior in Evolution*, Bradford Books, 73–132. MIT Press.

[18] Levins, R. & Lewontin, R. (1985). *The Dialectical Biologist*. Harvard University Press. https://www.hup.harvard.edu/books/9780674202832

[19] Taylor, T. (2004). *Niche Construction and the Evolution of Complexity*, 375–380. The MIT Press. https://doi.org/10.7551/mitpress/1429.003.0063

[20] Waddington, C. H. (2008). Paradigm for an evolutionary process. *Biological Theory*, 3(3), 258–266. https://doi.org/10.1162/biot.2008.3.3.258

[21] Laland, K. N., Boogert, N., & Evans, C. (2014). Niche construction, innovation and complexity. *Environmental Innovation and Societal Transitions*, 11, 71–86. https://doi.org/10.1016/j.eist.2013.08.003

[22] Creanza, N., Fogarty, L., & Feldman, M. W. (2016). Cultural niche construction of repertoire size and learning strategies in songbirds. *Evolutionary Ecology*, 30(2), 285–305. https://doi.org/10.1007/s10682-015-9796-1

[23] Stanley, S. M. (1973). An ecological theory for the sudden origin of multicellular life in the late precambrian. *Proceedings of the National Academy of Sciences*, 70(5), 1486–1489. https://doi.org/10.1073/pnas.70.5.1486

[24] Bonner, J. (1988). *The Evolution of Complexity by Means of Natural Selection*. Princeton University Press. https://press.princeton.edu/books/paperback/9780691084947/the-evolution-of-complexity-by-means-of-natural-sele

[25] Boraas, M. E., Seale, D. B., & Boxhorn, J. E. (1998). Phagotrophy by a flagellate selects for colonial prey: A possible origin of multicellularity. *Evolutionary Ecology*, 12(2), 153–164. https://doi.org/10.1023/a:1006527528063

[26] Yoshida, T., Jones, L. E., Ellner, S. P., Fussmann, G. F., & Hairston, N. G. (2003). Rapid evolution drives ecological dynamics in a predator–prey system. *Nature*, 424(6946), 303–306. https://doi.org/10.1038/nature01767

[27] Herron, M. D., Borin, J. M., Boswell, J. C., Walker, J., Chen, I.-C. K., Knox, C. A., Boyd, M., Rosenzweig, F., & Ratcliff, W. C. (2019). De novo origins of multicellularity in response to predation. *Scientific Reports*, 9(1). https://doi.org/10.1038/s41598-019-39558-8

[28] Pentz, J. T., Limberg, T., Beermann, N., & Ratcliff, W. C. (2015). Predator escape: An ecologically realistic scenario for the evolutionary origins of multicellularity. *Evolution: Education and Outreach*, 8(1). https://doi.org/10.1186/s12052-015-0041-8

[29] Goldsby, H. J., Dornhaus, A., Kerr, B., & Ofria, C. (2012). Task-switching costs promote the evolution of division of labor and shifts in individuality. *Proceedings of the National Academy of Sciences*, 109(34), 13686–13691. https://doi.org/10.1073/pnas.1202233109

[30] Joachimczak, M. & Wróbel, B. (2012). Co-evolution of morphology and control of soft-bodied multicellular animats. *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, Gecco '12, 561–568. https://doi.org/10.1145/2330163.2330243

[31] Goldsby, H. J., Knoester, D. B., Ofria, C., & Kerr, B. (2014). The evolutionary origin of somatic cells under the dirty work hypothesis. *PLoS Biology*, 12(5), e1001858. https://doi.org/10.1371/journal.pbio.1001858

[32] Bozdag, G. O., Zamani-Dahaj, S. A., Day, T. C., Kahn, P. C., Burnetti, A. J., Lac, D. T., Tong, K., Conlin, P. L., Balwani, A. H., Dyer, E. L., Yunker, P. J., & Ratcliff, W. C. (2023). De novo evolution of macroscopic multicellularity. *Nature*, 617(7962), 747–754. https://doi.org/10.1038/s41586-023-06052-1

[33] Ratcliff, W. C., Denison, R. F., Borrello, M., & Travisano, M. (2012). Experimental evolution of multicellularity. *Proceedings of the National Academy of Sciences*, 109(5), 1595–1600. https://doi.org/10.1073/pnas.1115323109

[34] Herron, M. D. (2016). Origins of multicellular complexity: Volvox and the volvocine algae. *Molecular Ecology*, 25(6), 1213–1223. https://doi.org/10.1111/mec.13551

[35] Kalambokidis, M. & Travisano, M. (2023). Multispecies interactions shape the transition to multicellularity. *Proceedings of the Royal Society B: Biological Sciences*, 290(2007). https://doi.org/10.1098/rspb.2023.1055

[36] Pennock, R. T. (2007). Models, simulations, instantiations, and evidence: the case of digital evolution. *Journal of Experimental & Theoretical Artificial Intelligence*, 19(1), 29–42. https://doi.org/10.1080/09528130601116113

[37] Adami, C., Ofria, C., & Collier, T. C. (2000). Evolution of biological complexity. *Proceedings of the National Academy of Sciences*, 97(9), 4463–4468. https://doi.org/10.1073/pnas.97.9.4463

[38] Lenski, R. E., Ofria, C., Pennock, R. T., & Adami, C. (2003). The evolutionary origin of complex features. *Nature*, 423(6936), 139–144. https://doi.org/10.1038/nature01568

[39] Zaman, L., Meyer, J. R., Devangam, S., Bryson, D. M., Lenski, R. E., & Ofria, C. (2014). Coevolution drives the emergence of complex traits and promotes evolvability. *PLoS Biology*, 12(12), e1002023. https://doi.org/10.1371/journal.pbio.1002023

[40] Dolson, E. (2019). *On the Constructive Power of Ecology in Open-ended Evolving Systems*. Michigan State University.

[41] Soros, L. & Stanley, K. (2014). Identifying necessary conditions for open-ended evolution through the artificial life world of chromaria. *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, Alife, 793–800. https://doi.org/10.1162/978-0-262-32621-6-ch128

[42] Huizinga, J., Stanley, K. O., & Clune, J. (2018). The emergence of canalization and evolvability in an open-ended, interactive evolutionary system. *Artificial Life*, 24(3), 157–181. https://doi.org/10.1162/artl_a_00263

[43] Lehman, J. & Stanley, K. O. (2012). Beyond open-endedness: Quantifying impressiveness. *Artificial Life Conference Proceedings 12*, 75–82.

[44] Nguyen, A. M., Yosinski, J., & Clune, J. (2015). Innovation engines: Automated creativity and improved stochastic optimization via deep learning. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, Gecco '15, 959–966. https://doi.org/10.1145/2739480.2754703

[45] Stanley, K. O. (2019). Why open-endedness matters. *Artificial Life*, 25(3), 232–235. https://doi.org/10.1162/artl_a_00294

[46] Packard, N., Bedau, M. A., Channon, A., Ikegami, T., Rasmussen, S., Stanley, K. O., & Taylor, T. (2019). An overview of open-ended evolution: Editorial introduction to the open-ended evolution ii special issue. *Artificial Life*, 25(2), 93–103.

[47] Taylor, T., Bedau, M., Channon, A., Ackley, D., Banzhaf, W., Beslon, G., Dolson, E., Froese, T., Hickinbotham, S., Ikegami, T., McMullin, B., Packard, N., Rasmussen, S., Virgo, N., Agmon, E., Clark, E., McGregor, S., Ofria, C., Ropella, G., Spector, L., Stanley, K. O., Stanton, A., Timperley, C., Vostinar, A., & Wiser, M. (2016). Open-ended evolution: Perspectives from the oee workshop in york. *Artificial Life*, 22(3), 408–423. https://doi.org/10.1162/artl_a_00210

[48] Moreno, M. A. & Ofria, C. (2019). Toward open-ended fraternal transitions in individuality. *Artificial Life*, 25(2), 117–133.

[49] Moreno, M. A. & Ofria, C. (2022). Exploring evolved multicellular life histories in a open-ended digital evolution system. *Frontiers in Ecology and Evolution*, 10. https://doi.org/10.3389/fevo.2022.750837

[50] Ratcliff, W. C., Fankhauser, J. D., Rogers, D. W., Greig, D., & Travisano, M. (2015). Origins of multicellular evolvability in snowflake yeast. *Nature Communications*, 6(1). https://doi.org/10.1038/ncomms7102

[51] Lalejini, A. & Ofria, C. (2018). Evolving event-driven programs with signalgp. *Proceedings of the Genetic and Evolutionary Computation Conference*, Gecco '18, 1135–1142. https://doi.org/10.1145/3205455.3205523

[52] Moreno, M. A., Lalejini, A., & Ofria, C. (2023). Matchmaker, matchmaker, make me a match: geometric, variational, and evolutionary implications of criteria for tag affinity. *Genetic Programming and Evolvable Machines*, 24(1), 4. https://doi.org/10.1007/s10710-023-09448-0

[53] Holland, J. (2012). *Signals and Boundaries: Building Blocks for Complex Adaptive Systems*. The MIT Press. MIT Press. https://mitpress.mit.edu/9780262525930/signals-and-boundaries

[54] Lalejini, A., Moreno, M. A., & Ofria, C. (2021). Tag-based regulation of modules in genetic programming improves context-dependent problem solving. *Genetic Programming and Evolvable Machines*, 22(3), 325–355. https://doi.org/10.1007/s10710-021-09406-8

[55] Brady, S. P., Bolnick, D. I., Angert, A. L., Gonzalez, A., Barrett, R. D., Crispo, E., Derry, A. M., Eckert, C. G., Fraser, D. J., Fussmann, G. F., Guichard, F., Lamy, T., McAdam, A. G., Newman, A. E., Paccard, A., Rolshausen, G., Simons, A. M., & Hendry, A. P. (2019). Causes of maladaptation. *Evolutionary Applications*, 12(7), 1229–1242. https://doi.org/10.1111/eva.12844

[56] Stanley, K. O., Lehman, J., & Soros, L. (2017). Open-endedness: The last grand challenge you've never heard of. *Radar / AI & ML*.

[57] Wilke, C. O., Wang, J. L., Ofria, C., Lenski, R. E., & Adami, C. (2001). Evolution of digital organisms at high mutation rates leads to survival of the flattest. *Nature*, 412(6844), 331–333. https://doi.org/10.1038/35085569

[58] Codoñer, F. M., Darós, J.-A., Solé, R. V., & Elena, S. F. (2006). The fittest versus the flattest: Experimental confirmation of the quasispecies effect with subviral pathogens. *PLoS Pathogens*, 2(12), e136. https://doi.org/10.1371/journal.ppat.0020136

[59] Ackley, D. & Small, T. (2014). Indefinitely scalable computing = artificial life engineering. *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems*, Alife, 606–613. https://doi.org/10.7551/978-0-262-32621-6-ch098

[60] Moreno, M. A., Yang, C., Dolson, E., & Zaman, L. (2024). Trackable agent-based evolution models at wafer scale. *The 2024 Conference on Artificial Life*, Alife 2024. https://doi.org/10.1162/isal_a_00830

[61] Heinemann, C. (2007). Artificial life environment. *Informatik-Spektrum*, 31(1), 55–61. https://doi.org/10.1007/s00287-007-0205-1

[62] Carroll, S. B. (2001). Chance and necessity: the evolution of morphological complexity and diversity. *Nature*, 409(6823), 1102–1109. https://doi.org/10.1038/35059227

[63] Tong, K., Bozdag, G. O., & Ratcliff, W. C. (2022). Selective drivers of simple multicellularity. *Current Opinion in Microbiology*, 67, 102141. https://doi.org/10.1016/j.mib.2022.102141

[64] Dawkins, R. & Krebs, J. R. (1979). Arms races between and within species. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 205(1161), 489–511. https://doi.org/10.1098/rspb.1979.0081

[65] Moreno, M. A., Papa, S. R., & Ofria, C. (2021). Conduit: a c++ library for best-effort high performance computing. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Gecco '21, 1795–1800. https://doi.org/10.1145/3449726.3463205

[66] Moreno, M. A., Dolson, E., & Ofria, C. (2022). Hereditary stratigraphy: Genome annotations to enable phylogenetic inference over distributed populations. *The 2022 Conference on Artificial Life*, Alife 2022, 64. https://doi.org/10.1162/isal_a_00550. 64

[67] Franklin, J., LaBar, T., & Adami, C. (2019). Mapping the peaks: Fitness landscapes of the fittest and the flattest. *Artificial Life*, 25(3), 250–262. https://doi.org/10.1162/artl_a_00296

[68] Wadgymar, S. M., Sheth, S., Josephs, E., DeMarche, M., & Anderson, J. (2024). Defining fitness in evolutionary ecology. *International Journal of Plant Sciences*, 185(3), 218–227. https://doi.org/10.1086/729360

[69] Sala, O., Odum, E., Jackson, R., Mooney, H., & Howarth, R. (2000). *Methods in Ecosystem Science*. Springer New York. https://doi.org/10.1007/978-1-4612-1224-9

[70] de Wit, C. T. (1960). On competition. *Verslagen van Landbouwkundige Onderzoekingen*, 66(8), 1–82.

[71] Adami, C. (2002). What is complexity? *BioEssays*, 24(12), 1085–1094. https://doi.org/10.1002/bies.10192

[72] Parrott, L. (2010). Measuring ecological complexity. *Ecological Indicators*, 10(6), 1069–1076. https://doi.org/10.1016/j.ecolind.2010.03.014

[73] Böttcher, T. (2017). From molecules to life: Quantifying the complexity of chemical and biological systems in the universe. *Journal of Molecular Evolution*, 86(1), 1–10. https://doi.org/10.1007/s00239-017-9824-6

[74] Tenaillon, O., Silander, O. K., Uzan, J.-P., & Chao, L. (2007). Quantifying organismal complexity using a population genetic approach. *PLoS ONE*, 2(2), e217. https://doi.org/10.1371/journal.pone.0000217

[75] Bonchev, D. & Buck, G. A. (2005). *Quantitative Measures of Network Complexity*, 191–235. Springer US. https://doi.org/10.1007/0-387-25871-x_5

[76] Hazen, R. M., Griffin, P. L., Carothers, J. M., & Szostak, J. W. (2007). Functional information and the emergence of biocomplexity. *Proceedings of the National Academy of Sciences*, 104(suppl_1), 8574–8581. https://doi.org/10.1073/pnas.0701744104

[77] Marion, Z. H., Fordyce, J. A., & Fitzpatrick, B. M. (2015). Extending the concept of diversity partitioning to characterize phenotypic complexity. *The American Naturalist*, 186(3), 348–361. https://doi.org/10.1086/682369

[78] Rebout, N., Lone, J.-C., De Marco, A., Cozzolino, R., Lemasson, A., & Thierry, B. (2021). Measuring complexity in organisms and organizations. *Royal Society Open Science*, 8(3). https://doi.org/10.1098/rsos.200895

[79] McShea, D. W. (1996). Perspective metazoan complexity and evolution: Is there a trend? *Evolution*, 50(2), 477–492. https://doi.org/10.1111/j.1558-5646.1996.tb03861.x

[80] McShea, D. W. (2000). Functional complexity in organisms: Parts as proxies. *Biology and Philosophy*, 15(5), 641–668. https://doi.org/10.1023/a:1006695908715

[81] Knoll, A. H. (2011). The multiple origins of complex multicellularity. *Annual Review of Earth and Planetary Sciences*, 39(1), 217–239. https://doi.org/10.1146/annurev.earth.031208.100209

[82] Niklas, K. J. & Newman, S. A. (2013). The origins of multicellular organisms. *Evolution & Development*, 15(1), 41–52. https://doi.org/10.1111/ede.12013

[83] Brunet, T. & King, N. (2017). The origin of animal multicellularity and cell differentiation. *Developmental Cell*, 43(2), 124–140. https://doi.org/10.1016/j.devcel.2017.09.016

[84] Bonner, J. T. (2001). *First Signals: The Evolution of Multicellular Development*. Princeton University Press. https://press.princeton.edu/books/paperback/9780691070384/first-signals

[85] Dolson, E. L., Vostinar, A. E., Wiser, M. J., & Ofria, C. (2019). The modes toolbox: Measurements of open-ended dynamics in evolving systems. *Artificial Life*, 25(1), 50–73. https://doi.org/10.1162/artl_a_00280

[86] Sinervo, B. & Basolo, A. L. (1996). *Testing Adaptation Using Phenotypic*, 149. Academic Press.

[87] Moreno, M. A. (2024). Methods to estimate cryptic sequence complexity. *The 2024 Conference on Artificial Life*, Alife 2024. https://doi.org/10.1162/isal_a_00776

[88] Foster, E. D. & Deardorff, A. (2017). Open science framework (osf). *Journal of the Medical Library Association*, 105(2). https://doi.org/10.5195/jmla.2017.88

[89] Moreno, M. A., Rodriguez Papa, S., & Ofria, C. (2025). *Toward open-ended fraternal transitions in individuality*. https://doi.org/10.5281/zenodo.16990564

[90] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., & SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

[91] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser,

W., Abbasi, H., Gohlke, C., & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[92] pandas developers (2020). pandas-dev/pandas: Pandas. *Zenodo*. https://doi.org/10.5281/zenodo.3509134

[93] Wes McKinney (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

[94] Waskom, M. L. (2021). seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60), 3021. https://doi.org/10.21105/joss.03021

[95] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. https://doi.org/10.1109/mcse.2007.55

[96] Moreno, M. A. (2023). *mmore500/teeplot*. https://doi.org/10.5281/zenodo.10440670

[97] Seabold, S. & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. *Proceedings of the 9th Python in Science Conference*, SciPy, 92–96. https://doi.org/10.25080/majora-92bf1922-011

[98] Vostinar, A., Lalejini, A., Ofria, C., Dolson, E., & Moreno, M. A. (2024). Empirical: A scientific software library for research, education, and public engagement. *Journal of Open Source Software*, 9(98), 6617. https://doi.org/10.21105/joss.06617

[99] Moreno, M. A., Rodriguez Papa, S., Lalejini, A., & Ofria, C. (2021). *Signalgp-lite: Event driven genetic programming library for large-scale artificial life applications*. https://doi.org/10.48550/arxiv.2108.00382

[100] Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., & de Hoon, M. J. L. (2009). Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11), 1422–1423. https://doi.org/10.1093/bioinformatics/btp163

[101] Lalejini, A., Dolson, E., Bohm, C., Ferguson, A. J., Parsons, D. P., Rainford, P. F., Richmond, P., & Ofria, C. (2019). Data standards for artificial life software. *The 2019 Conference on Artificial Life*, Alife 2019. https://doi.org/10.1162/isal_a_00213

[102] Ackley, D. H. (2023). A robust programmable replicator for an indefinitely scalable machine. *The 2023 Conference on Artificial Life*, Alife 2023. https://doi.org/10.1162/isal_a_00701

[103] Lauterbach, G. (2021). The path to successful wafer-scale integration: The cerebras story. *IEEE Micro*, 41(6), 52–57. https://doi.org/10.1109/mm.2021.3112025

[104] Downing, K. L. (2015). *Intelligence Emerging: Adaptivity and Search in Evolving Neural Systems*. The MIT Press. https://doi.org/10.7551/mitpress/9898.001.0001

[105] Moreno, M. A., Rodriguez Papa, S., & Dolson, E. (2024). *Analysis of phylogeny tracking algorithms for serial and multiprocess applications*. https://doi.org/10.48550/arXiv.2403.00246

Supplement for "Case Study of Novelty, Complexity, and Adaptation in a Multicellular System" by Matthew Andres Moreno, Santiago Rodriguez Papa, and Charles Ofria in OEE4: The Fourth Workshop on Open-Ended Evolution

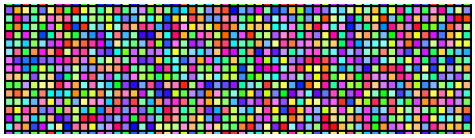
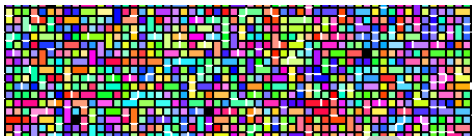
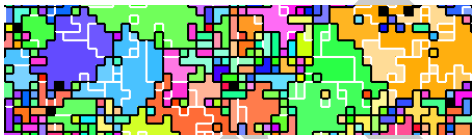
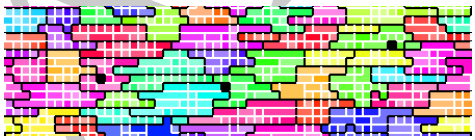
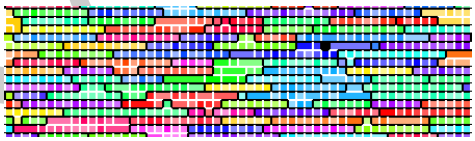
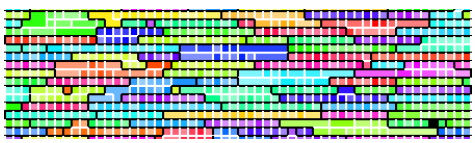
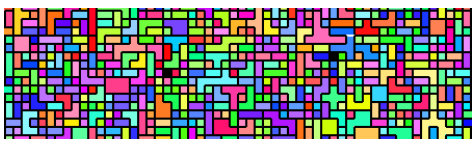
ID	Morphology	Snapshot	Video
a	Individual cells, no multicellular kin groups. Resource use is low—most cells simply hoard resource until their stockpile is beyond sufficient to reproduce. Only a handful of cells intermittently expend resource.		https://hopth.ru/21/b=prq49+s=16005+t=0+v=video+w=specimen
b	Mostly individual cells, with some two-, three-, and four-cell groups evenly spread out. Resource usage occurs in short spurts in one or two adjacent cells.		https://hopth.ru/21/b=prq49+s=16005+t=1+v=video+w=specimen
c	Large multicellular groups dominate, consisting of hundreds of cells. Group growth is unchecked and continues until cells' resource stockpiles are entirely depleted by the excess group size penalty.		https://hopth.ru/21/b=prq49+s=16005+t=2+v=video+w=specimen
d	Clear groups of 10 to 15 cells form. Cell proliferation appears somewhat more active at the periphery of groups compared to the interior.		https://hopth.ru/21/b=prq49+s=16005+t=14+v=video+w=specimen
e	Groups are visibly elongated along the horizontal axis. After initial development, some gradual, irregular growth occurs along the vertical axis.		https://hopth.ru/21/b=prq49+s=16005+t=15+v=video+w=specimen
f	Groups are horizontally elongated similarly to morphology e, but have a larger consistent vertical thickness of three or four cells.		https://hopth.ru/21/b=prq49+s=16005+t=39+v=video+w=specimen
g	Initial group growth is almost entirely horizontal, with groups usually taking up only one row of cells. However, after an apparent timing cue groups perform a brief bout of aggressive vertical growth.		https://hopth.ru/21/b=prq49+s=16005+t=45+v=video+w=specimen
h	Groups grow horizontally and then proliferate vertically on a timing cue like morph e. However, after that timing cue cell proliferation is incessant with almost no resource retention.		https://hopth.ru/21/b=prq49+s=16005+t=59+v=video+w=specimen
i	Irregular groups of mostly less than ten cells. Incessant proliferation with almost no resource retention leads to rapid group turnover.		https://hopth.ru/21/b=prq49+s=16005+t=74+v=video+w=specimen
j	Groups grow horizontally and then proliferate vertically on a timing cue like morph e. However, several viable horizontal-bar offspring groups form before forced fragmentation.		https://hopth.ru/21/b=prq49+s=16005+t=100+v=video+w=specimen

Table 1: **Qualitative morph phenotype categorizations.** Color coding of morph IDs has no significance beyond guiding the eye in scatter plots where points are labeled by morph. Snapshot visualizes spatial layout of kin groups on toroidal grid at a fixed point in time. Each cell corresponds to a small square tile. Color hue denotes, and black borders divide, outermost kin groups; color saturation denotes, and white borders divide, innermost kin groups.

Table 2: Qualitative morph categorization of representative specimens sampled across evolutionary stints. Video links provide a time-lapse animation of each specimen's morphology when grown in monoculture. The human-readable genome link will download the corresponding genome as an annotated JSON file.

Stint	Morph	Monoculture Video Link	Human-readable Genome Link
0	a	https://hopth.ru/21/b=prq49+s=16005+t=0+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=0+v=text+w=specimen
1	b	https://hopth.ru/21/b=prq49+s=16005+t=1+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=1+v=text+w=specimen
2	c	https://hopth.ru/21/b=prq49+s=16005+t=2+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=2+v=text+w=specimen
3	b	https://hopth.ru/21/b=prq49+s=16005+t=3+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=3+v=text+w=specimen
4	b	https://hopth.ru/21/b=prq49+s=16005+t=4+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=4+v=text+w=specimen
5	b	https://hopth.ru/21/b=prq49+s=16005+t=5+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=5+v=text+w=specimen
6	b	https://hopth.ru/21/b=prq49+s=16005+t=6+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=6+v=text+w=specimen
7	b	https://hopth.ru/21/b=prq49+s=16005+t=7+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=7+v=text+w=specimen
8	b	https://hopth.ru/21/b=prq49+s=16005+t=8+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=8+v=text+w=specimen
9	b	https://hopth.ru/21/b=prq49+s=16005+t=9+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=9+v=text+w=specimen
10	b	https://hopth.ru/21/b=prq49+s=16005+t=10+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=10+v=text+w=specimen
11	b	https://hopth.ru/21/b=prq49+s=16005+t=11+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=11+v=text+w=specimen
12	b	https://hopth.ru/21/b=prq49+s=16005+t=12+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=12+v=text+w=specimen
13	b	https://hopth.ru/21/b=prq49+s=16005+t=13+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=13+v=text+w=specimen
14	d	https://hopth.ru/21/b=prq49+s=16005+t=14+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=14+v=text+w=specimen
15	e	https://hopth.ru/21/b=prq49+s=16005+t=15+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=15+v=text+w=specimen
16	e	https://hopth.ru/21/b=prq49+s=16005+t=16+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=16+v=text+w=specimen
17	e	https://hopth.ru/21/b=prq49+s=16005+t=17+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=17+v=text+w=specimen
18	e	https://hopth.ru/21/b=prq49+s=16005+t=18+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=18+v=text+w=specimen
19	e	https://hopth.ru/21/b=prq49+s=16005+t=19+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=19+v=text+w=specimen
20	e	https://hopth.ru/21/b=prq49+s=16005+t=20+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=20+v=text+w=specimen
21	e	https://hopth.ru/21/b=prq49+s=16005+t=21+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=21+v=text+w=specimen
22	e	https://hopth.ru/21/b=prq49+s=16005+t=22+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=22+v=text+w=specimen
23	e	https://hopth.ru/21/b=prq49+s=16005+t=23+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=23+v=text+w=specimen
24	e	https://hopth.ru/21/b=prq49+s=16005+t=24+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=24+v=text+w=specimen
25	e	https://hopth.ru/21/b=prq49+s=16005+t=25+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=25+v=text+w=specimen
26	b	https://hopth.ru/21/b=prq49+s=16005+t=26+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=26+v=text+w=specimen
27	e	https://hopth.ru/21/b=prq49+s=16005+t=27+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=27+v=text+w=specimen
28	b	https://hopth.ru/21/b=prq49+s=16005+t=28+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=28+v=text+w=specimen
29	e	https://hopth.ru/21/b=prq49+s=16005+t=29+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=29+v=text+w=specimen
30	e	https://hopth.ru/21/b=prq49+s=16005+t=30+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=30+v=text+w=specimen
31	e	https://hopth.ru/21/b=prq49+s=16005+t=31+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=31+v=text+w=specimen
32	e	https://hopth.ru/21/b=prq49+s=16005+t=32+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=32+v=text+w=specimen
33	e	https://hopth.ru/21/b=prq49+s=16005+t=33+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=33+v=text+w=specimen
34	e	https://hopth.ru/21/b=prq49+s=16005+t=34+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=34+v=text+w=specimen
35	e	https://hopth.ru/21/b=prq49+s=16005+t=35+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=35+v=text+w=specimen
36	e	https://hopth.ru/21/b=prq49+s=16005+t=36+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=36+v=text+w=specimen

Table 2 (cont'd)

Stint	Morph	Monoculture Video Link	Human-readable Genome Link
37	e	https://hopth.ru/21/b=prq49+s=16005+t=37+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=37+v=text+w=specimen
38	e	https://hopth.ru/21/b=prq49+s=16005+t=38+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=38+v=text+w=specimen
39	f	https://hopth.ru/21/b=prq49+s=16005+t=39+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=39+v=text+w=specimen
40	e	https://hopth.ru/21/b=prq49+s=16005+t=40+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=40+v=text+w=specimen
41	e	https://hopth.ru/21/b=prq49+s=16005+t=41+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=41+v=text+w=specimen
42	e	https://hopth.ru/21/b=prq49+s=16005+t=42+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=42+v=text+w=specimen
43	e	https://hopth.ru/21/b=prq49+s=16005+t=43+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=43+v=text+w=specimen
44	e	https://hopth.ru/21/b=prq49+s=16005+t=44+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=44+v=text+w=specimen
45	g	https://hopth.ru/21/b=prq49+s=16005+t=45+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=45+v=text+w=specimen
46	g	https://hopth.ru/21/b=prq49+s=16005+t=46+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=46+v=text+w=specimen
47	g	https://hopth.ru/21/b=prq49+s=16005+t=47+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=47+v=text+w=specimen
48	g	https://hopth.ru/21/b=prq49+s=16005+t=48+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=48+v=text+w=specimen
49	g	https://hopth.ru/21/b=prq49+s=16005+t=49+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=49+v=text+w=specimen
50	g	https://hopth.ru/21/b=prq49+s=16005+t=50+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=50+v=text+w=specimen
51	g	https://hopth.ru/21/b=prq49+s=16005+t=51+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=51+v=text+w=specimen
52	g	https://hopth.ru/21/b=prq49+s=16005+t=52+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=52+v=text+w=specimen
53	e	https://hopth.ru/21/b=prq49+s=16005+t=53+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=53+v=text+w=specimen
54	g	https://hopth.ru/21/b=prq49+s=16005+t=54+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=54+v=text+w=specimen
55	g	https://hopth.ru/21/b=prq49+s=16005+t=55+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=55+v=text+w=specimen
56	g	https://hopth.ru/21/b=prq49+s=16005+t=56+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=56+v=text+w=specimen
57	g	https://hopth.ru/21/b=prq49+s=16005+t=57+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=57+v=text+w=specimen
58	g	https://hopth.ru/21/b=prq49+s=16005+t=58+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=58+v=text+w=specimen
59	h	https://hopth.ru/21/b=prq49+s=16005+t=59+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=59+v=text+w=specimen
60	g	https://hopth.ru/21/b=prq49+s=16005+t=60+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=60+v=text+w=specimen
61	e	https://hopth.ru/21/b=prq49+s=16005+t=61+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=61+v=text+w=specimen
62	g	https://hopth.ru/21/b=prq49+s=16005+t=62+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=62+v=text+w=specimen
63	g	https://hopth.ru/21/b=prq49+s=16005+t=63+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=63+v=text+w=specimen
64	e	https://hopth.ru/21/b=prq49+s=16005+t=64+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=64+v=text+w=specimen
65	e	https://hopth.ru/21/b=prq49+s=16005+t=65+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=65+v=text+w=specimen
66	g	https://hopth.ru/21/b=prq49+s=16005+t=66+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=66+v=text+w=specimen
67	g	https://hopth.ru/21/b=prq49+s=16005+t=67+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=67+v=text+w=specimen
68	e	https://hopth.ru/21/b=prq49+s=16005+t=68+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=68+v=text+w=specimen
69	g	https://hopth.ru/21/b=prq49+s=16005+t=69+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=69+v=text+w=specimen
70	e	https://hopth.ru/21/b=prq49+s=16005+t=70+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=70+v=text+w=specimen
71	e	https://hopth.ru/21/b=prq49+s=16005+t=71+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=71+v=text+w=specimen
72	e	https://hopth.ru/21/b=prq49+s=16005+t=72+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=72+v=text+w=specimen
73	g	https://hopth.ru/21/b=prq49+s=16005+t=73+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=73+v=text+w=specimen
74	i	https://hopth.ru/21/b=prq49+s=16005+t=74+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=74+v=text+w=specimen

Table 2 (cont'd)

Stint	Morph	Monoculture Video Link	Human-readable Genome Link
75	i	https://hopth.ru/21/b=prq49+s=16005+t=75+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=75+v=text+w=specimen
76	g	https://hopth.ru/21/b=prq49+s=16005+t=76+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=76+v=text+w=specimen
77	i	https://hopth.ru/21/b=prq49+s=16005+t=77+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=77+v=text+w=specimen
78	e	https://hopth.ru/21/b=prq49+s=16005+t=78+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=78+v=text+w=specimen
79	g	https://hopth.ru/21/b=prq49+s=16005+t=79+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=79+v=text+w=specimen
80	e	https://hopth.ru/21/b=prq49+s=16005+t=80+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=80+v=text+w=specimen
81	e	https://hopth.ru/21/b=prq49+s=16005+t=81+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=81+v=text+w=specimen
82	e	https://hopth.ru/21/b=prq49+s=16005+t=82+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=82+v=text+w=specimen
83	e	https://hopth.ru/21/b=prq49+s=16005+t=83+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=83+v=text+w=specimen
84	e	https://hopth.ru/21/b=prq49+s=16005+t=84+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=84+v=text+w=specimen
85	e	https://hopth.ru/21/b=prq49+s=16005+t=85+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=85+v=text+w=specimen
86	e	https://hopth.ru/21/b=prq49+s=16005+t=86+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=86+v=text+w=specimen
87	e	https://hopth.ru/21/b=prq49+s=16005+t=87+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=87+v=text+w=specimen
88	e	https://hopth.ru/21/b=prq49+s=16005+t=88+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=88+v=text+w=specimen
89	g	https://hopth.ru/21/b=prq49+s=16005+t=89+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=89+v=text+w=specimen
90	g	https://hopth.ru/21/b=prq49+s=16005+t=90+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=90+v=text+w=specimen
91	e	https://hopth.ru/21/b=prq49+s=16005+t=91+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=91+v=text+w=specimen
92	i	https://hopth.ru/21/b=prq49+s=16005+t=92+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=92+v=text+w=specimen
93	e	https://hopth.ru/21/b=prq49+s=16005+t=93+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=93+v=text+w=specimen
94	e	https://hopth.ru/21/b=prq49+s=16005+t=94+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=94+v=text+w=specimen
95	b	https://hopth.ru/21/b=prq49+s=16005+t=95+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=95+v=text+w=specimen
96	b	https://hopth.ru/21/b=prq49+s=16005+t=96+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=96+v=text+w=specimen
97	h	https://hopth.ru/21/b=prq49+s=16005+t=97+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=97+v=text+w=specimen
98	e	https://hopth.ru/21/b=prq49+s=16005+t=98+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=98+v=text+w=specimen
99	e	https://hopth.ru/21/b=prq49+s=16005+t=99+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=99+v=text+w=specimen
100	j	https://hopth.ru/21/b=prq49+s=16005+t=100+v=video+w=specimen	https://hopth.ru/21/b=prq49+s=16005+t=100+v=text+w=specimen

In the following sections, L refers to the number of hierarchical kin group levels defined for the simulation. In this work, we use $L = 2$.

A Methods Supplement

A.1 Simulation

The DISHTINY simulation environment tracks cells occupying tiles on a toroidal grid (size 120×120 by default). Cells collect a uniform inflow of continuous-valued resource. This resource can be spent in increments of 1.0 to attempt asexual reproduction into any of a cell's four adjacent cells.* A cell can only be replaced if it commands less than 1.0 resource. If a cell rebuffs a reproduction attempt, its resource stockpile decrements by 1.0 down to a minimum of 0.0.

To facilitate the formation of coherent multicellular groups, the DISHTINY framework provides a mechanism for cells to form groups and detect group membership[48] . Groups arise through cellular reproduction. When a cell proliferates, it may choose to initiate its offspring as a member of its kin group, thereby growing it, or induce the offspring to found a new kin group. This process is similar to the growth of biological multicellular tissues, where cell offspring can be retained as members of the tissue or permanently expelled.

We incentivize group formation by providing an additional resource inflow bonus based on group size. Per-cell resource collection rate increases linearly with group size up to a cap of 12 members. Beyond 12 members, the decay rate of cells' resource stockpiles begins to increase exponentially. These mechanisms select for medium-sized groups; in particular, the harsh penalization of oversized groups prevents any single group from consuming the entire population. Undersized groups, on the other hand, do not receive resource bonuses. Groups that are too large receive a penalty. To ensure group turnover, we force groups to fragment into unicells after 8,192 (2^{13}) updates.

In previous work , we established that this framework can select for traits characteristic of multicellularity, such as cooperation, coordination, and reproductive division of labor[49] . We also found that more case studies of interest arose when two nested levels of group membership were tracked as opposed to a single, unnested level of group membership[49] . With nested group membership, group growth still occurs by cellular reproduction. Cells are given the choice to retain offspring within both groups, to expel offspring from both groups, or to expel offspring from the innermost group only. In this work, we allow for nested kin groups.

In addition to controlling reproduction behavior, evolving genomes can also share resources with adjacent cells, perform apoptosis (recovering a small amount of resource that may be shared with neighboring cells), and pass arbitrary messages to neighboring cells. Cell behaviors are controlled by event-driven genetic programs in which linear GP modules are activated in response to cues from the environment or neighboring agents; signals are handled in quasi-parallel on up to 32 virtual cores (Figure 1g)[51]. Each cell contains four independent virtual CPUs, all of which execute the same genetic program (Figure 1e). Each CPU manages interactions with a single neighboring cell. We refer to a CPU managing interactions with a particular neighbor as a "cardinal processor" (in analogy to "cardinal directions"). These CPUs may communicate via intracellular message passing. To support plasticity and differentiation, each CPU may also dynamically regulate tag-matching priorities of program modules[54].

Full details on the instruction set and event library used, as well as simulation logic and parameter settings, appear in supplementary material.

*All reproduction in this model was asexual. While sexual recombination can facilitate evolution of complexity by combining adaptive traits and purging deleterious mutations, these mechanisms are orthogonal to our focus on dynamics of emergent biotic selection. However, given the prevalence of sexual recombination among complex multicellular species, the interplay between sex and biotic selection merits dedicated treatment in further research.

A.2 Evolution

We performed evolution in three-hour windows for compatibility with our compute cluster's scheduling system. We refer to these windows as "stints." We randomly generated 100-instruction genomes at the outset of the initial stint, stint 0. At the end of each three-hour window, the system harvested and stored genomes in a population file. We then seeded subsequent stints with the previous stint's population. No simulation state besides genome content was preserved between stints. In addition to simplifying implementation concerns, re-seeding each stint ensured that strains retained the capability to grow from a well-mixed inoculum. This facilitated later competition experiments between strains. Figure 14 summarizes this procedure.

In order to ensure heterogeneity of biotic environmental factors experienced by evolving cells, we imposed a diversity maintenance scheme. In this scheme, descendants of a single progenitor cell from stint 0 that proliferated to constitute more than half of the population were penalized with resource loss. The severity of the penalty increased with increasing prevalence beyond half of the population. Thus, we ensured that descendants from at least two distinct stint 0 progenitors remained over the course of the simulation. We arbitrarily chose a strain for primary study — we refer to this strain as the "focal" strain and others as "background" strains. In our case study, there was only one background strain in addition to this focal strain.

In our screen for case studies, we evolved 40 independent populations for 101 stints. We selected population 16005 from among these 40 to profile as a case study due to its distinct asymmetrical group morphology.

At the conclusion of each stint, we selected the most abundant genome within the population as a representative specimen. We performed a suite of follow-up analyses on each representative specimen to characterize aspects of complexity, detailed in the following subsections. To ensure that specimens were consistently sampled from descendants of the same stint 0 progenitor, we only considered genomes with the lowest available stint 0 progenitor ID.

A.3 Phenotype-neutral Nopout

After harvesting representative specimens from each stint, we filtered out genome instructions that had no impact on the simulation.

To accomplish this, we performed sequential single-site "nopouts" where individual genome instructions were disabled by replacing them with a `Nop` instruction.† We reverted nopouts that altered a strain's phenotype and kept those that did not. To determine whether phenotypic alteration occurred, we seeded an independent, mutation-disabled simulation with the strain in question and ran it side-by-side with an independent, mutation-disabled simulation of the wildtype strain. If any divergence in resource concentration was detected between the two strains within a 2,048-update window, the single-site nopout was reverted. We continued this process until no single-site nopouts were possible without altering the genome's phenotype. To speed up evaluation, we performed step-by-step, side-by-side comparisons using a smaller toroidal grid size of just 100 tiles.

This process left us with a "Phenotype-neutral Nopout" variant of the wildtype genome where all remaining instructions contributed to the phenotype.

However, in further analyses we discovered that 21 phenotype-neutral nopouts from our case study were *not* actually neutral — competition experiments revealed they were significantly less fit than the wildtype strain. This might be due to insufficient spatial or temporal scope to observe the expression of particular genome sites in our test for phenotypic divergence.

†This `Nop` instruction was chosen to perform the same number of random number generator touches as the original instruction to control for arbitrary effects of advancing the generator.

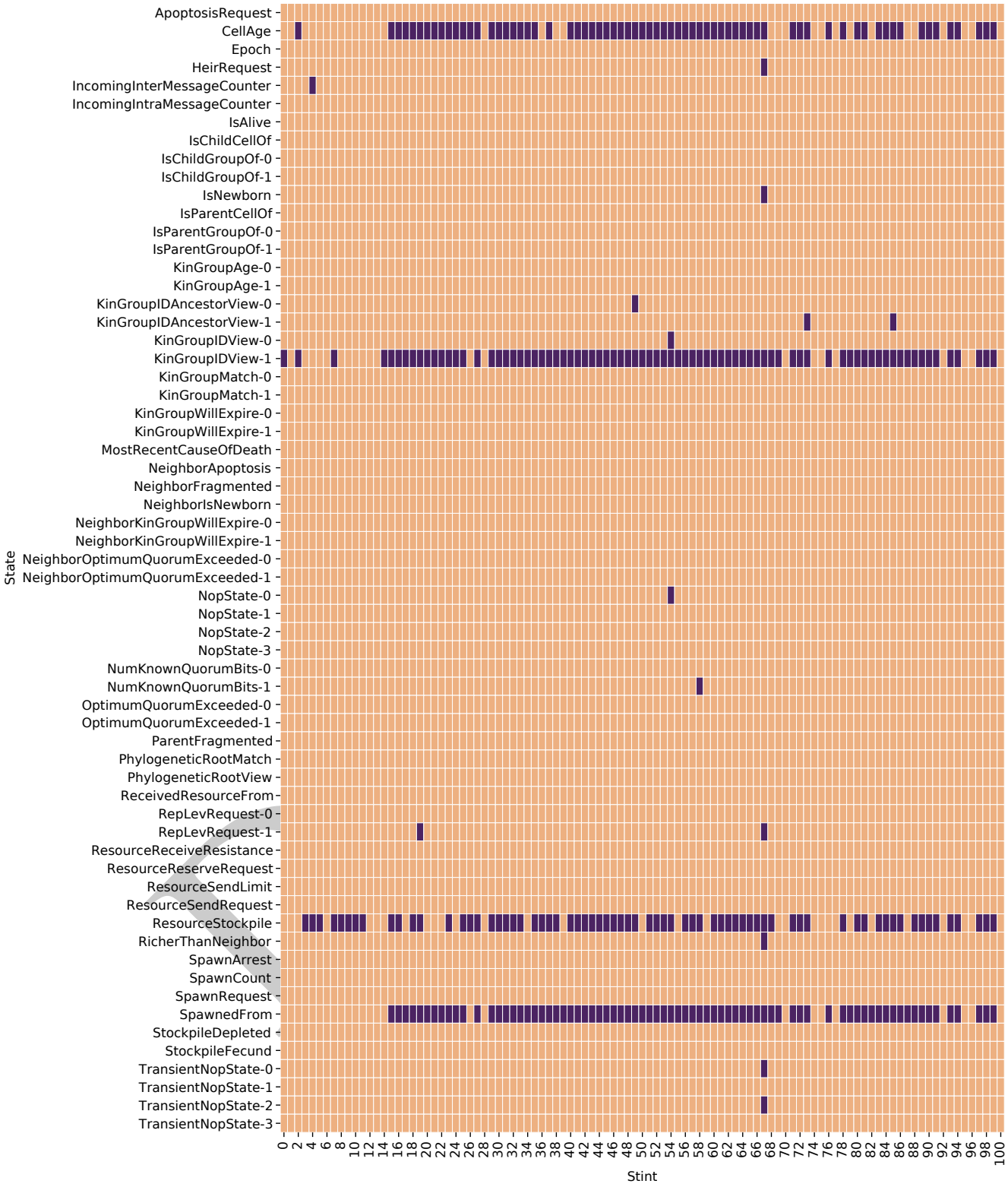


Figure 10: Fitness effect of extrospective states (read-only state information of neighboring cells) for focal strains between stint 0 and 100. Peach color indicates no fitness effect. Burgundy indicates a significant fitness effect. Supplementary material provides a description for each state.

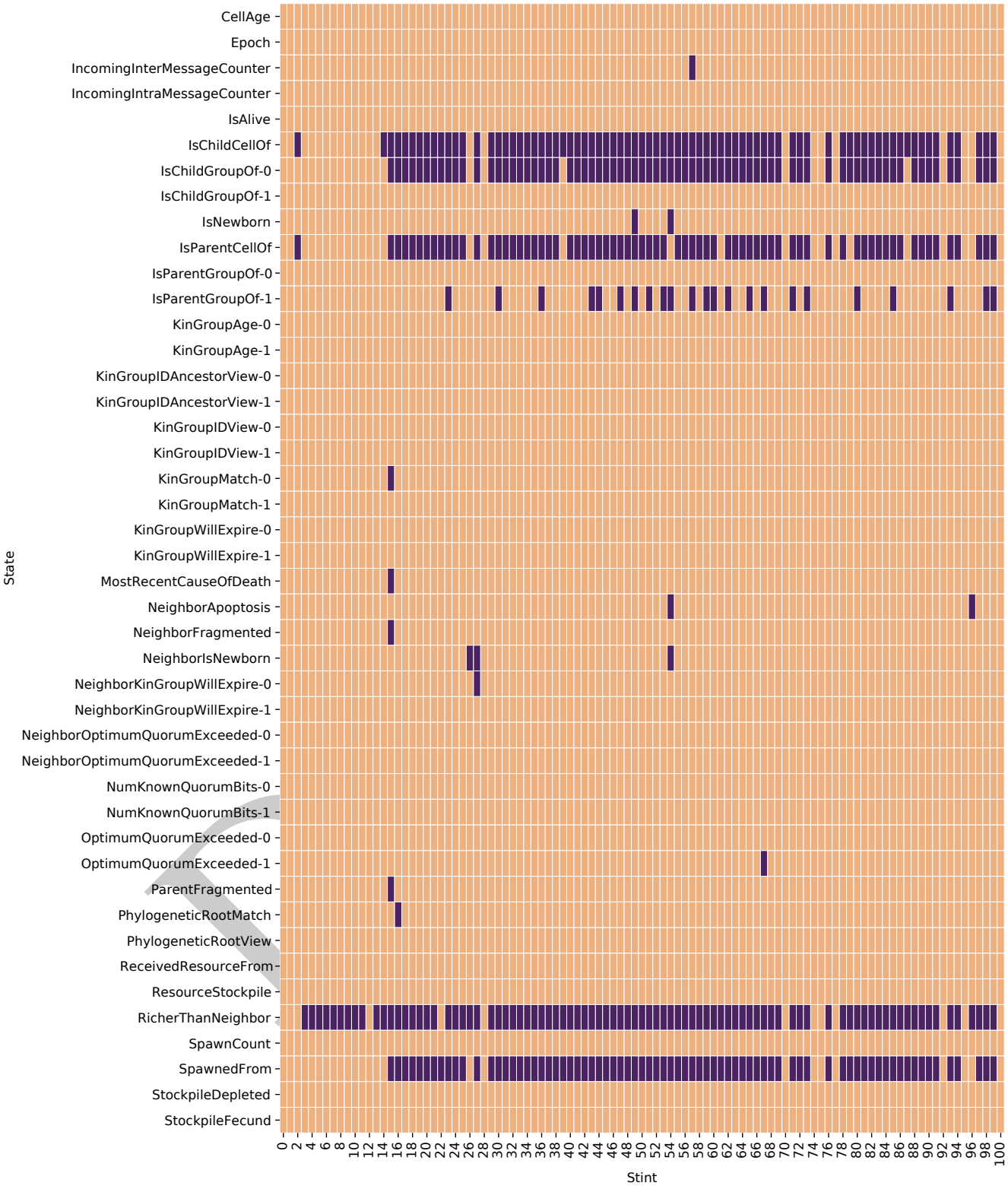


Figure 11: Fitness effect of introspective states (read-only state information of own cell) for focal strains between stint 0 and 100. Peach color indicates no fitness effect. Burgundy indicates a significant fitness effect. Supplementary material provides a description for each state.

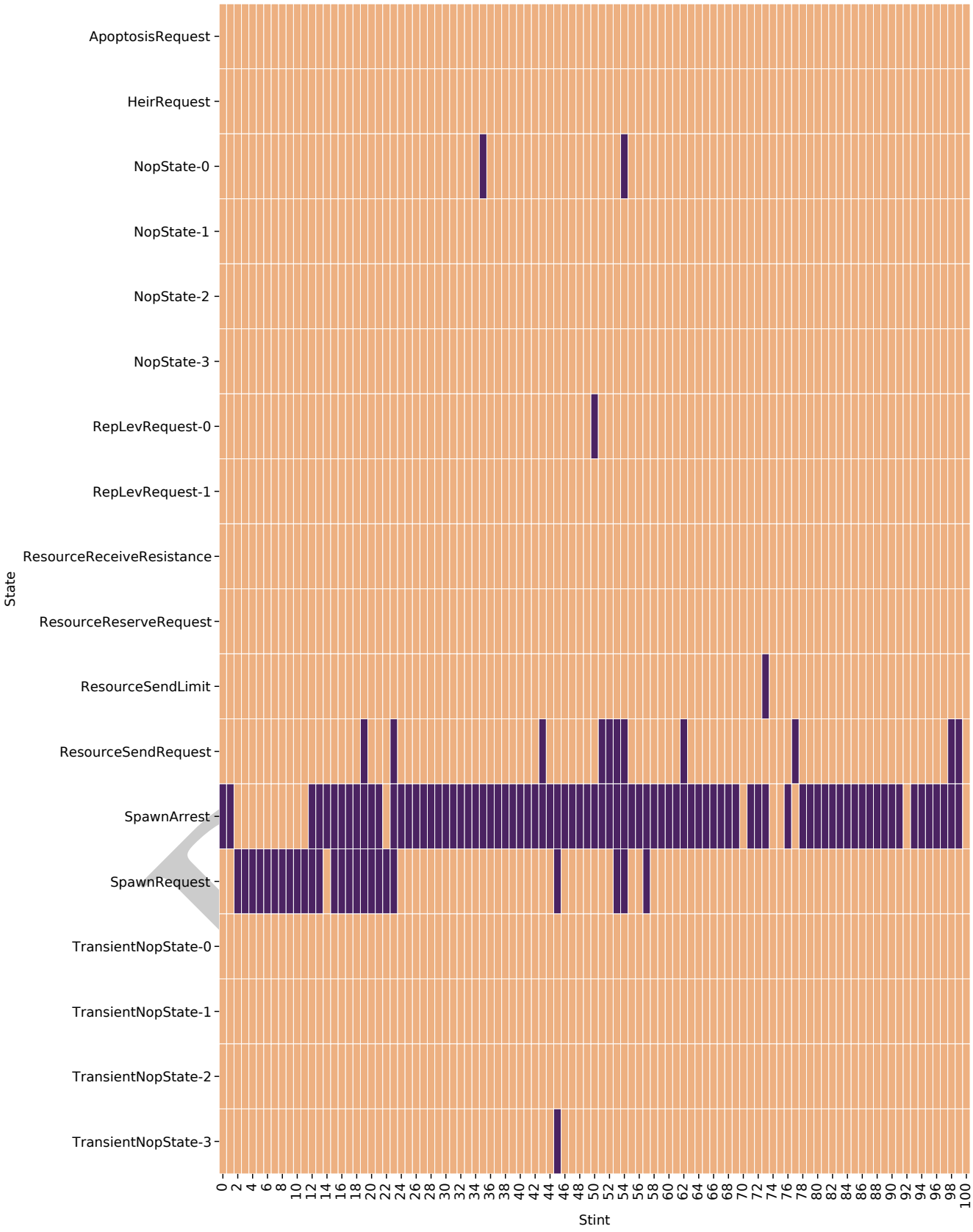


Figure 12: Fitness effect of writable states for focal strains between stint 0 and 100. Peach color indicates no fitness effect. Burgundy indicates a significant fitness effect. Supplementary material provides a description for each state.

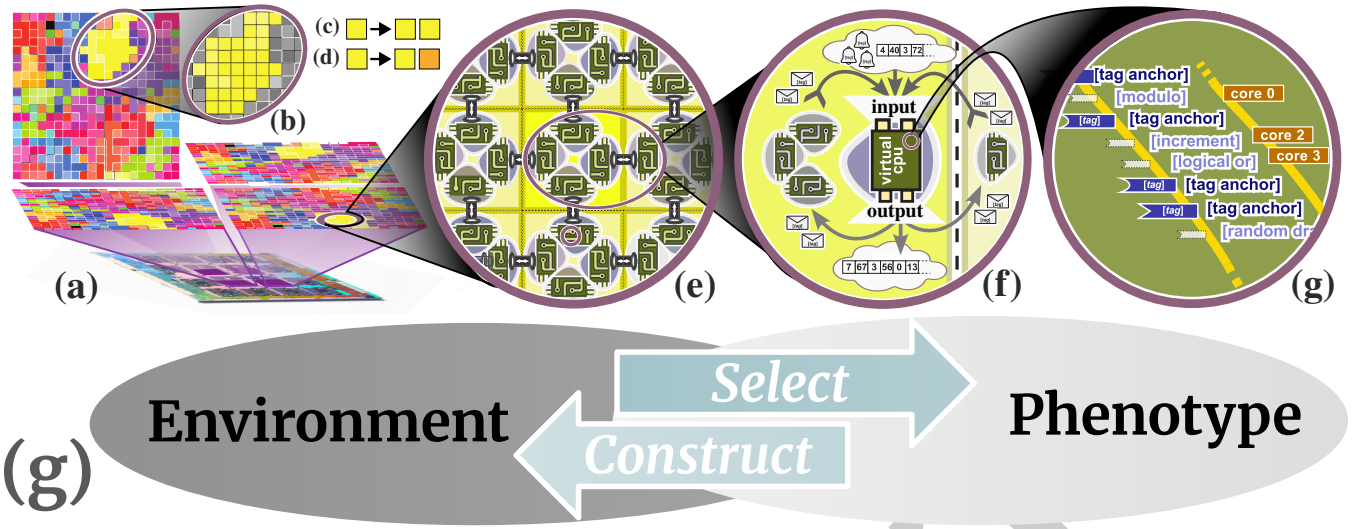


Figure 13: **Overview of digital multicell model.** Model comprises a toroidal grid of 14,400 cells (120×120) occupied by a population of replicating computer programs. To accelerate the feasible evolutionary duration of experiments, population subgrids are evaluated across available hardware threads and processes — in this work, 4 threads (panel a). At subgrid boundaries where inter-cell interactions reach between different threads or processes, communication between cells is handled on a best-effort basis via the underlying Conduit library. On the population grid, cells may form local multicell groups — shown as color-coded patches (panel b). Cells replicate by copying program content into a chosen neighbor cell, and may choose to grow their existing multicell group (c) — or splinter their offspring off to found a new group (d). Within each cell, behavior is directed by a collection of four virtual CPUs ("cardinal" processors), each independently managing the cell's interactions with N, S, E, or W neighbor cell (panel e). Each cardinal processor can interact with other cardinal processors in its own cell via message passing. A cardinal processor can also exchange messages with its counterpart in the abutting neighbor cell, allowing arbitrary inter-cell communication. In addition to message-passing I/O, cardinal processors accept input to sense local simulation state and create output to perform cell actions (panel f). Sensory input occurs via a combination of special input registers and special messages triggered by simulation conditions ("events," depicted as bells in panel f). Behavior outputs are dispatched via special writable registers (e.g., for resource sharing, replication, apoptosis, etc.). At implementation level, replicator program code is evaluated via an event-driven execution model. That is, program submodules trigger in response to stimuli — determined by tag-matching mechanisms (panel g). In addition to external stimuli (i.e., inter-cellular messages, intra-cellular messages, and simulation events), program code can also trigger arbitrary self-stimuli. Virtual hardware supports concurrent interaction handling via independent pseudo-cores, as well as dynamic plasticity through regulation of tag-matching affinities within each cardinal processor. In contrast to deme-based digital models where multicell life histories are evaluated in isolation [29,31] (panel h), instantiating multicells within a unified spatial realm with dense biotic interaction introduces potential for rich feedbacks on selection for life history traits as multicells constitute their own environmental context (panel i). This approach creates opportunity to study the relationship between eco-evolutionary feedbacks and the origins of biotic complexity, novelty, and adaptation (panel j).

A.4 Estimating Genotype Complexity

Next, we sought to detect genome instructions that contributed to strain fitness.

For each remaining op instruction in the Phenotype-neutral Nopout variant, we took the wildtype strain and applied a nopout at the corresponding site. We then competed this variant against the wildtype strain. Evaluating only remaining op instructions in the Phenotype-neutral Nopout variant allowed us to decrease the number of fitness competitions we had to perform.

We initialized fitness competitions by seeding a population half-and-half with two strains. We ran these competitions for 10 minutes (about 4,200 updates) on a 60×60 toroidal grid, after which we assessed the relative abundances of descendants of both seeded strains.

To determine whether fitness differed significantly between a wildtype and variant strain, we compared the relative abundance of the strains observed at the end of competitions against outcomes from 20 control wildtype-vs-wildtype competitions. We fit a T-distribution to the abundance outcomes observed under the control wildtype-vs-wildtype competitions and deemed outcomes that fell outside the central 98% probability density of that distribution a significant difference in fitness. This allowed us to screen for fitness effects of single-site nopouts while only performing a single competition per site.

This process left us with a "Fitness-noncritical Nopout" variant of the wildtype genome where all remaining instructions contributed to the phenotype. We called the number of remaining instructions its "genotype complexity." This metric mirrors the MODES complexity metric described by Dolson et al. [85] and the approximation of sequence complexity advanced by Adami et al. [37].

A.5 Estimating State Phenotype Complexity

In addition to estimating the number of genome sites that contribute to fitness, we measured the number of different environmental cues and the number of different output mechanisms that cells adaptively incorporated into behavior.

One possible way to take this measure would be to disable event cues, sensor instructions, and output registers one by one and test for changes in fitness. However, this approach would fail to distinguish context-dependent input/output from merely contingent input/output. For example, a cell might arbitrarily depend on a sensor being set at a certain frequency, but not make use of underlying information the sensor provides.

To isolate context-dependent input/output state interactions, we tested the fitness effect of swapping particular input/output states between CPUs rather than completely disabling them. That is, for example, CPU b would be forced to perform the output generated by CPU a or CPU b would be shown the input meant for CPU a. We performed this manipulation on half the population in a fitness competition for each individual component of the simulation's introspective state (44 sensor states relating to the

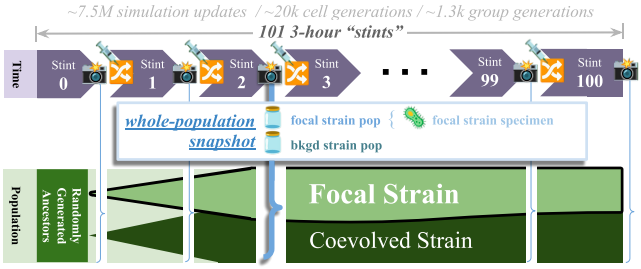


Figure 14: **Evolutionary replicate design.** Experiment was initialized with 40 founder populations of randomly generated genomes. Among founder lineages, an arbitrary clade was designated as the "focal strain" subject of the case study. To ensure genomes maintained viability as simulation seeds (e.g., for competition and monoculture trials) and provide checkpoints in the case of cluster downtime, population was subjected to serial passage at 3-hour intervals — referred to as "stints." Between stints, a whole-population snapshot of genome content was recorded and a sample genome was collected from the focal strain ("focal specimen"). To continue evolution in the following stint, a fresh simulation instance was initialized by shuffled injection of snapshot genomes. Note that, due to early extinctions, focal designation switched to a new strain at stint 2 in case study replicate, which was maintained onwards. A resource-balancing diversity-maintenance procedure was used to curtail growth of any founder lineage beyond population majority, hence ensuring long-term coexistence of at least one independent "background" strain as ecological context.

status of a CPU's own cell), extrospective state (61 sensor states relating to the status of a neighboring cell), and writable state (18 output states, 10 of which control cell behavior and 8 of which act as global memory for the CPU).* We deemed a state as fitness-critical if this manipulation resulted in decreased fitness at significance $p < 0.01$ using a T-test parameterized by 20 control wildtype vs wildtype competitions.

We describe the number of states that cells interact with to contribute to fitness as "State Phenotype Complexity."

A.6 Estimating Messaging Phenotype Complexity

In addition to estimating the number of input/output states cells use to interact with the environment, we also estimated the number of distinct intracellular messages cardinal processors within a cell use to coordinate, as well as the intercellular messages that cells use to coordinate. As with state phenotype complexity, distinguishing context-dependent behavior from contingent behavior is critical to attaining a meaningful measurement. For example, a cardinal processor might depend on always receiving an intercellular message from a neighbor or an intracellular message from another cardinal processor. Although meaningless, if that message were blocked, fitness would decrease. So, instead of simply discarding messages to test for a fitness effect, we reroute messages back to the sending cardinal processor instead of their intended recipient. We deemed messages as fitness-critical if this manipulation resulted in decreased fitness at significance $p < 0.01$ using a T-test parameterized by 20 control wildtype vs wildtype competitions.

We refer to the number of distinct messages that cells send to contribute to fitness as "Messaging Phenotype Complexity."

We refer to the sum of State Phenotype Complexity, Intra-messaging Phenotype Complexity, and Inter-messaging Phenotype Complexity as "Phenotype Complexity."

*A full description of each piece of introspective, extrospective, and writable state is listed in supplementary material.

A.7 Estimating Adaptation

In order to assess ongoing changes in fitness, we performed fitness competitions between the representative focal strain specimen sampled at each stint and the focal strain population from the preceding stint.† Using the population from the preceding stint as the competitive baseline (rather than the representative specimen) ensured more focused, consistent measurement of the fitness properties of the specimen at the current stint (e.g., preventing skewed results from a sampled "dud" at the preceding stint). Figure 9 overviews fitness competition design.

We performed 20 independent replicates of each competition. Competing strains were well-mixed within the full-sized toroidal grid at the outset of each competition, which lasted for 10 minutes of wall time. This was sufficient to simulate about 8,000 updates at stint 0 and 2,000 updates at stint 100 (Supplementary Figures 21, 23, and 22). We determined that a gain of fitness had occurred if the current stint specimen constituted a population majority at the conclusion of more than 17 of those competitions, corresponding to a significance level of $p < 0.005$ under the two-tailed binomial null hypothesis. Likewise, we deemed winning fewer than 3 competitions a significant fitness loss.

A.8 Implementation

We employed multithreading to speed up execution. We split the simulation into four 60×60 subgrids. Each subgrid executed asynchronously, using the Conduit C++ Library to orchestrate best-effort, real-time interactions between simulation elements on different threads[65]. This approach is in part informed by Ackley's principles of indefinite scalability, which highlight fault tolerance, soft synchronization, decentralization, and spatial locality as key engineering principles for large-scale *in silico* artificial life systems[59,102]. In particular, the localized grid topology and best-effort communication model of DISHTINY are designed with an eye towards compatibility with the dataflow architecture of next-generation AI/ML hardware accelerator devices, which field up to hundreds of thousands of independent processor cores but impose restrictions on communication locality and memory availability[60,103]. In other work benchmarking the system, we have demonstrated that this approach improves scalability. The simulation scales to 4 threads with 80% efficiency, up to 64 threads with 40% efficiency and up to 64 nodes with 80% efficiency[65].

Over the 101 three-hour evolutionary stints performed to evolve the case study, 7,565,309 simulation updates elapsed. This translates to 74,904 updates elapsed per stint or about 6.9 updates per second. However, the update processing rate was not uniform across stints: the simulation slowed about 77% as stints progressed. Supplementary Figure 20 shows elapsed updates for each stint. During stint 0, 176,816 updates elapsed (about 16.3 updates per second). During stint 100, only 41,920 updates elapsed (about 3.8 updates per second).

Although working asynchronously, threads processed similar numbers of updates during each stint. The mean standard deviation of update-processing rate between threads was 2%. The mean difference of the update-processing rate between the fastest and slowest threads was 5%. The maximum value of these statistics observed during a stint was 9% and 20%, respectively, at stint 44. Supplementary Figure 20b shows the distribution of elapsed updates across threads for each stint evolved during the case study.

B Virtual CPU

Each cardinal processor hosts a signalgp-lite virtual CPU[51,99]. Each CPU can host up to 16 active virtual cores. If additional cores are required after all 16 available are in use, the oldest active core is killed and replaced. Each virtual core contains 8 virtual `float` registers.

†Recall from Section A.2 that, due to a diversity maintenance procedure, two completely independent strains coexisted over the course of the experiment — the "focal" strain selected for analysis and a "background" strain.

Cores execute round-robin in quasi-parallel, with up to 8 instructions being executed on a single core before execution shifts to the next active core.

Like SignalGP, the signalgp-lite system uses tag-matching to determine which modules to activate in response to incoming signals from the environment, from other agents (i.e., messages), and from internal events (i.e., execution of `call` and `fork` instructions).

In the DISHTINY simulation, each CPU hosts two independent module-lookup data structures. The first module-lookup data structure is used to activate modules in response to internally-generated signals, messages from other cardinal processors within the same cell, and environmental events; this module-lookup data structure contains *all* modules within the genetic program. The second module-lookup data structure is used to activate modules in response to messages from other cells; this module-lookup data structure contains only modules with bitstring tags that end in 0. (Hence, the subset of modules with bitstring tags that end in 1 *cannot* be activated by messages from other cells, so that sensitive functionality like resource sharing and apoptosis can be protected from potentially malicious exploitation.)

We also use a tag-matching system to route `jump` instructions executed within a module. When a module is loaded, all `local anchor` instructions are registered within a tag-matching data structure. The local `jump` instruction routes to the best-matching local anchor. If no matching local anchor is available, then no jump is performed and execution continues as if a `nop` instruction had elapsed.

C Tag Matching

We use 64-bit bitstring tags to label modules and jump destinations. We use a variant of Downing's streak metric to compute tag matches[52,104]. We deterministically select the single best-matching result for a tag lookup. If there is a tie, an arbitrary result is selected. For module-lookup, the best-matching tag must have a match quality at the 80th or better percentile among match qualities of pairs of randomly generated tags. Otherwise, no module is activated. For jump-lookup, the best match must be at the 50th or better percentile. Otherwise, no jump is performed.

D Program Generation and Mutation

Initial populations were seeded with programs consisting of 128 randomly generated instructions. Program length was capped at 4096 instructions.

Mutation was applied to one in 10 reproductions where any kin group commonality was maintained and to all reproductions where it was not. If mutation occurred, bits in the binary representation of the genome were flipped with 0.02% probability. If mutation occurred, sequence mutations were also introduced into the program at a per-site rate of 0.1%. Half of sequence mutations were deletion events, with a number of sites deleted drawn uniformly between 0 and 8. Half of sequence mutations were insertion events, with a number of sites inserted drawn uniformly between 0 and 8. When sites were inserted, half of the time randomly-generated instructions were added and half of the time the preceding sequence of instructions was duplicated. With 0.1% probability a sequence mutation took on severe intensity, meaning that the number of sites inserted or deleted was drawn uniformly between 0 and program size rather than between 0 and 8.

E Cooperative Resource Collection

In order to ensure kin group structure had functional ramifications, we based part of cell resource collection on the number of contiguous kin group members. To do this, we needed an efficient distributed method to approximate kin group size.

Each cell held a 64-bit bitstring with one chosen bit fixed. We estimated group size by counting the number of distinct set bits (out of 64 available slots) that were contained within a kin group. We refer to this count of distinct set bits as a group's "quorum count."

At every update within each tile, the simulation system broadcasts all bits that were known to be set within that tile's kin group. This broadcast was only sent to neighboring tiles that were part of the same kin group as the broadcasting tile. Each tile tracked which neighbor it learned of each set bit from so that when tiles left the kin group, their set bits could be forgotten from the bits known to be set within the kin group.

This scheme was replicated independently for each kin group level simulated. For the lowest-level kin group, a different fixed bit was chosen independently for each tile. Thus, the quorum count for these lowest-level kin groups was a function of the number of cells contained. For the highest-level kin group, each tile's fixed bit was chosen as a deterministic function of its lowest-level kin group ID. Thus, the quorum count for these highest-level kin groups was a function of the number of lowest-level kin groups contained.

To incentivize kin group formation and maintenance, we gave each cell a 0.02 resource bonus every four updates for each non-self quorum count. This bonus saturated at the simulation-defined target quorum count. For both the lowest- and highest-level kin groups we used a target quorum count of 12. The source code controlling cooperative resource collection can be found at `https://github.com/mmore500/dishtiny/blob/prq49/include/dish2/services/CollectiveHarvestingService.hpp`.

In addition to this cooperative resource collection mechanism, cells enjoyed a continuous resource inflow of 0.02 units per update. The source code controlling base resource inflow can be found at `https://github.com/mmore500/dishtiny/blob/master/include/dish2/services/ResourceHarvestingService.hpp`.

To penalize groups that expanded beyond the simulation-defined target quorum count, we decayed held resource by a multiplicative factor of 0.9995^{2^n}, where n is the excess quorum count beyond the simulation-defined target. The source code controlling cooperative resource collection can be found at `https://github.com/mmore500/dishtiny/blob/prq49/include/dish2/services/CollectiveResourceDecayService.hpp`.

In addition to any decay due to group size, held resource decayed at a rate of 0.05% per update. Received resource was decayed by 0.099975% upon receipt.

F Evolutionary History of High-complexity Case Study

Due to the parallel, asynchronous nature of the experimental framework, we did not perform perfect phylogeny tracking[105].

Instead, we opted to track the total number of ancestors seeded into stint 0 with extant descendants. At the end of stints 0 and 1, three distinct original phylogenetic roots were present in the population. From stint 2 onward, only two distinct original phylogenetic roots were present.

We performed follow-up analyses on specimens sampled from the lowest original phylogenetic root ID present in the population.* For the first two stints, the focal strain was root ID 2,378. During stint 2, original phylogenetic root 2,378 went extinct. So, all further follow-up analyses were sampled from descendants of ancestor 12,634.

We also tracked the number of genomes reconstituted at the outset of each stint that had extant descendants at the end of that stint. This count

*This approach was designed to choose an arbitrary strain as focal. Barring extinction, that same strain will then be identified as focal consistently across subsequent stints. Phylogenetic root ID had no functional consequences; it is simply an arbitrary basis for focal strain selection.

grows from approximately 10 around stint 15 to upwards of 30 around stint 40 (Supplementary Figure 15a). Among descendants of the lowest original phylogenetic root, the number of independent lineages spanning a stint also increases from around 5 to around 15 (Supplementary Figure 15b). This decrease in phylogenetic consolidation on a stint-by-stint basis correlates with the waning number of simulation updates performed per stint (Supplementary Figures 15c and 15d). More complete phylogenetic data will be necessary in future experiments to address questions about the possibility of long-term stable coexistence beyond the two strains supported under the explicit diversity maintenance scheme.

On the specimen from stint 100 used in the final case study, an evolutionary history of 20,212 cell generations had elapsed. Of these cellular reproductions, 11,713 (58%) had full kin group commonality, 7,174 had partial kin group commonality (35%), and 1,325 had no kin group commonality (7%). On this specimen, 1,672 mutation events had elapsed. During these events, 7,240 insertion-deletion alterations had occurred and 26,153 point mutations had occurred. This strain experienced a selection pressure of 18% over its evolutionary history, meaning that only 82% of the mutations that would be expected given the number of cellular reproductions that had elapsed were present.

In order to characterize the evolutionary history of the experiment in greater detail, we performed a parsimony-based phylogenetic reconstruction on the sampled representative specimens from each stint, shown in Supplementary Figure 33. We used genomes' fixed-length blocks of 35 64-bit tags that mediate environmental interactions as the basis for this reconstruction. These tag blocks underwent bitwise mutation over the course of the experiment.* Supplementary Figure 16 shows Hamming distance between all pairs of tag blocks. We additionally tried several other tree inference methods, discussed in supplementary material; however, these yielded lower-quality reconstructions.

Although the phylogeny of stint representatives includes many instances that do not constitute a strict sequential lineage (i.e., each stint's representative descending directly from the preceding stint's representative), we did not observe evidence of long-term coexistence of clades over more than ten stints.

F.1 Fitness Complexity

Figure 54b plots genotype complexity of specimens drawn from across the case study's evolutionary history.

Genotype complexity reaches more than 20 under morph b, jumps to more than 40 under morph d, and drops to slightly more than 30 for morph e. Genotype complexity reaches a peak of 48 sites around stint 39, then levels out and decreases.

In two independent sublineages — both going extinct — genotype and phenotype complexity simultaneously and abruptly crashed near zero. Both were morph i: unrestricted cell proliferation with small irregular groups. No complexity crash, however, appears in assays configured with biotic background — suggesting a contingent dependency to this background may have developed (Supplementary Figure 49).

Phylogenetic analysis (Supplementary Figure 33) suggests independent origins of the genotype fitness complexity in morph d and morph e — the morph d specimen from stint 14 is more closely related to the morph b specimen from stint 13 than to the morph e specimen from stint 15. Likewise, specimens of lower complexity morphs i and b that appear past stint 70 appear to have independent evolutionary origins.

F.2 Phenotype Complexity

Supplementary Figure 38 summarizes phenotype complexity, as well as its constituent components, for specimens drawn from across the case study's evolutionary history.

*In future experiments, we plan to incorporate new methodology for "hereditary stratigraph" genome annotations expressly designed to facilitate phylogenetic reconstruction[66] .

Notably, cardinal processor phenotype complexity more than doubles from 6 interactions to 17 interactions coincident with the emergence of morph e (Supplementary Figure 38a). This is due to simultaneous increases in extrospective state sensing (2 to 9 states; Supplementary Figure 38f), introspective state sensing (1 to 4 states; Supplementary Figure 38e), and writable state usage (1 to 2 states; Supplementary Figure 38b).

The emergence of morph g coincided with an increase in writable state phenotype complexity from 1 to 3, as shown in Supplementary Figure 38b. However, morph g was not associated with other changes in other aspects of cardinal processor phenotype complexity. The greatest observed cardinal processor phenotype complexity was 22 interactions at stints 54 and 67.

F.3 Genome Size

Supplementary Figure 39 shows evolutionary trajectories of three genome size metrics in sampled focal strain specimens. Instruction count and module count increased from 100 and 5 to around 800 and 30, respectively, between stints 0 and 40. Within this period at stint 24, instruction count jumped from around 600 to more than 800 and module count jumped by about 5. This was coincident with detection in our adaptation assays of population-level sign-change mediation of adaptation by the background strain (Supplementary Figure 35). In sampled specimen fitness assays at stint 24, we detected significant increases in fitness in the presence of the background strain but no significant change in fitness in its absence.

Between stints 40 and 90, module count gradually increased to around 40 while instruction count remained stable. Then, at stint 93, instruction count jumped to around 1,500 and module count jumped to around 60. This was coincident with the strong fitness differentials observed at stint 93 (Supplementary Figure 36).

To better understand the functional effects of changes in genome size, we additionally measured the number of instructions that affected agent phenotype, shown in Figure 39c. This measure can be considered akin to a count of "active" sites.

Active site count varied greatly stint to stint. The median value increased from nearly 0 to around 200 sites between stints 0 and 40. Between stints 40 and 90, we observed active site values ranging from less than 100 to more than 500. Morph g specimens appear to show particularly great variance in active site count. The first observed morph g specimen at stint 45 exhibited relatively low active site count of around 100 active sites. The highest active site count values of around 700 were measured from three specimens of morphs e and g in the last ten stints.

G Additional Control Trials for Biotic Background Experiments

To better characterize the mechanism behind fitness effects caused by the background strain, we performed additional specimen adaptation assays under biotic background conditions with diversity maintenance disabled. This analysis allowed us to test whether action of the diversity maintenance mechanism, rather than direct interactions between the focal and background strains, caused the observed fitness effects. Supplementary Figure 34 compares adaptation assay outcomes with and without diversity maintenance under both the prefatory and contemporary biotic background conditions. Outcomes were generally similar, and we observed only one sign-change difference: one specimen outcome was beneficial under prefatory biotic background conditions without diversity maintenance, but deleterious with diversity maintenance. Further, without diversity maintenance, we still observed four outcomes that were advantageous under biotic conditions but deleterious under abiotic conditions (Supplementary Figure 32b). So, biotic selective effects cannot be explained as an artifact of the diversity maintenance scheme.

We also conducted specimen adaptation assays with diversity maintenance disabled under the control focal strain biotic background. In these

experiments, we again found no evidence for impact from the diversity maintenance scheme on results (Supplementary Figure 28).

H Simulation Details

H.1 Events

This section enumerates simulation-managed events that were dispatched on virtual CPUs. In addition to a program, each genome contained an array of 64-bit tags — one for each event. When an event's criteria was met in the simulation, the genome's corresponding tag was used to dispatch a module in the program and launch a core executing that module.

All events are also exposed to the cell as a corresponding input sensor. The state of the event (0 for false, 1 for true) is stored in the sensor prior to virtual CPU execution. In fact, events are triggered based on the reading of the sensor register (not by re-reading the underlying simulation state). This means that experimental perturbations that perturb sensor input also disrupted event-handling, allowing the state interface complexity metric to measure both event-driven and sensor-based behaviors.

The source code controlling events can be found at `https://github.com/mmore500/dishtiny/tree/prq49/include/dish2/events` and `https://github.com/mmore500/dishtiny/blob/prq49/include/dish2/services/InterpretedIntrospectiveStateRefreshService.hpp`.

Always This event is always dispatched.

Is Child Cell Of Is this cell a daughter cell of the cardinal processor's neighbor? Triggered if this cell was spawned from the cardinal processor's neighbor and its cell is younger than the neighbor.

Is Child Group Of (0 thru $L-1$) Is this cell's kin group a daughter group of the cardinal processor's neighbor cell's kin group? Triggered if a cell's kin group ancestor ID(s) are equal to the cardinal processor's neighbor's current kin group ID(s).

Is Newborn This event is dispatched once when a cell is first born. Triggered if cell age is less than frequency at which events are launched.

Is Parent Cell Of Is this cardinal processor's cell the parent cell of the cardinal processor's neighbor? Triggered if neighbor was spawned from cell and cell age is greater than neighbor age.

Kin Group Match (0 thru $L-1$) Is this cell part of the same kin group as the cardinal processor's neighbor? Triggered if a cell's kin group ID(s) are equal to the cardinal processor's neighbors' current kin group ID(s).

Kin Group Mismatch (0 thru $L-1$) Is this cell part of a different kin group from the cardinal processor's neighbor cell? Triggered if a cell's kin group ID(s) are not equal to the cardinal processor's neighbors' current kin group ID(s).

Kin Group will Expire (0 thru $L-1$) Triggered if kin group age is greater than 80% of the kin group expiration duration. (Depending on experiment configuration, the group may be force-fragmented after expiration.)

Kin Group will not Expire (0 thru $L-1$) Triggered if kin group age is less than or equal to than 80% of the kin group expiration duration.

Neighbor Apoptosis Triggered if the most recent cell death in the cardinal processor's neighbor tile was apoptosis.

Neighbor Fragmented Triggered if the most recent cell death in the cardinal processor's neighbor tile was fragmentation.

Neighbor Is Alive Triggered if a cardinal processor's neighbor tile is occupied by a live cell.

Neighbor Is Newborn Triggered once for each time a newborn spawns into the cardinal processor's neighboring tile. Triggered if the cardinal processor's neighbor's age is less than the frequency at which events are launched.

Neighbor Is Not Alive Triggered if the cardinal processor's neighboring tile is not occupied.

Neighbor Kin Group Will Expire (0 thru $L-1$) Triggered if the cardinal processor's cell neighbor's kin group age is less than or equal to 80% of the kin group expiration duration.

Neighbor Optimum Quorum Exceeded Triggered if the cardinal processor's cell neighbor's number of known quorum bits exceed the target quorum count.

Optimum Quorum Exceeded (0 thru $L-1$) Triggered if the cell's number of known quorum bits exceed the target quorum count.

Optimum Quorum Not Exceeded (0 thru $L-1$) Triggered if the cell's number of known quorum bits is less than or equal to than the target quorum count.

Parent Fragmented Triggered if a cell's parent died from fragmentation. That is, if the last cause of death on the current tile was fragmentation.

Phylogenetic Root Match Does this cell descend from the same originally-generated genome as its neighbor? Triggered if a cardinal processor's cell's root ID is equal to that cardinal processor's neighbor cell's root ID.

Phylogenetic Root Mismatch Does this cell and its neighbor descend from a different originally-generated genomes? Triggered if a cardinal processor's cell's root ID is not equal to that cardinal processor's neighbor cell's Root ID.

Poorer Than Neighbor Does this cell have less resource stockpiled than its neighbor? Triggered if a cardinal processor's cell has less resource than that cardinal processor's neighbor cell.

Received Resource From Triggered if a cardinal processor's cell has received resource from that cardinal processor's neighbor cell.

Richer Than Neighbor Does this cell have more resource stockpiled than its neighbor? Triggered if a cardinal processor's cell has more resource in its stockpile than that cardinal processor's neighbor cell.

Stockpile Depleted Is this cell's stockpile empty? Triggered if a cell's stockpile is less than twice the base harvest rate.

Stockpile Fecund Does this cell have enough stockpiled resource to fund cellular reproduction? Triggered if a cell's stockpile is greater than 1.0.

H.2 Operations

This section overviews the operation library made available to evolving signalgp-lite genetic programs within the simulation.

Within the program section of each genome, each instruction contained

- an op code, specifying which operation should be performed;
- a 64-bit bitstring, used as a tag for operations that required tag-matching or as data for some configurable operations; and

- three integer arguments, specifying which registers the operation should apply to (many operations do not use all arguments).

In the operation descriptions below, we refer to register access via to the nth argument as `reg[arg_n]`. Each core has its own eight `float` registers. All core registers are zeroed out at core launch.

In order to prevent bread-and-butter operations like global anchors, local anchors, and terminals from being swamped out by large instruction set size, we manually defined increased "prevalences" for some instructions. This prevalence increased the probability of the operation being selected under mutations and initial program generation. Prevalence works like increasing the number of identical copies of the operation included in the operation library. We provide the prevalence of each operation below.

See `https://github.com/mmore500/dishtiny/tree/prq49/include/dish2/operations` for the source code of DISHTINY-specific operations and `https://github.com/mmore500/signalgp-lite/tree/b6c437f44136651aa6f4051d84bc62a86c2afbbe/include/sgpl/operations` for the source code of generic operations.

Refer to Section B for details on the virtual CPU running these instructions.

Fork If

Prevalence	1
Num Args	1

If `reg[arg_0]` is nonzero, registers a request to activate a new core with the module best-matching the current instruction's tag. These fork requests are only handled when the current core terminates. Each core may only register 3 fork requests.

Nop, 0 RNG Touches

Prevalence	1
Num Args	0

Performs no operation for one virtual CPU cycle.

Nop, 1 RNG Touches

Prevalence	1
Num Args	0

Performs no operation for one virtual CPU cycle, and advances the RNG engine once. (Important to nop-out operations that perform one RNG touch without causing side effects.)

Nop, 2 RNG Touches

Prevalence	1
Num Args	1

Performs no operation for one virtual CPU cycle, and advances the RNG engine twice. (Important to nop-out operations that perform two RNG touches without causing side effects.)

Terminate If

Prevalence	1
Num Args	1

Terminates current core if `reg[arg_0]` is nonzero.

Add

Prevalence	1
Num Args	3

Adds `reg[arg_1]` to `reg[arg_2]` and stores the result in `reg[arg_0]`.

Divide

Prevalence	1
Num Args	3

Divides `reg[arg_1]` by `reg[arg_2]` and stores the result in `reg[arg_0]`. Division by zero can result in an `Inf` or `NaN` value.

Modulo

Prevalence	1
Num Args	3

Calculates the modulus of `reg[arg_1]` by `reg[arg_2]` and stores the result in `reg[arg_0]`. Mod by zero can result in a `NaN` value.

Multiply

Prevalence	1
Num Args	3

Multiplies `reg[arg_1]` by `reg[arg_2]` and stores the result in `reg[arg_0]`.

Subtract

Prevalence	1
Num Args	3

Subtracts `reg[arg_2]` from `reg[arg_1]` and stores the result in `reg[arg_0]`.

Bitwise And

Prevalence	1
Num Args	3

Performs a bitwise AND of `reg[arg_1]` and `reg[arg_2]` then stores the result in `reg[arg_0]`.

Bitwise Not

Prevalence	1
Num Args	2

Computes the bitwise NOT of `reg[arg_1]` and stores the result in `reg[arg_0]`.

Bitwise Or

Prevalence	1
Num Args	3

Performs a bitwise OR of `reg[arg_1]` and `reg[arg_2]` then stores the result in `reg[arg_0]`.

Bitwise Shift

Prevalence	1
Num Args	3

Shifts the bits of `reg[arg_1]` by `reg[arg_2]` positions. (If `reg[arg_2]` is negative, this is a right shift. Otherwise it is a left shift.) Stores the result in `reg[arg_0]`.

Bitwise Xor

Prevalence	1
Num Args	3

Performs a bitwise XOR of `reg[arg_1]` and `reg[arg_2]` then stores the result in `reg[arg_0]`.

Count Ones

Prevalence	1
Num Args	2

Counts the number of bits set in `reg[arg_1]` and stores the result in `reg[arg_0]`.

Random Fill

Prevalence	1
Num Args	1

Fills register pointed to by `reg[arg_0]` with random bits chosen from a uniform distribution.

Equal

Prevalence	1
Num Args	3

Checks whether `reg[arg_1]` is equal to `reg[arg_2]` and stores the result in `reg[arg_0]`.

Greater Than

Prevalence	1
Num Args	3

Checks whether `reg[arg_1]` is greater than `reg[arg_2]` and stores the result in `reg[arg_0]`.

Less Than

Prevalence	1
Num Args	3

Checks whether `reg[arg_1]` is less than `reg[arg_2]` and stores the result in `reg[arg_0]`.

Logical And

Prevalence	1
Num Args	3

Performs a logical AND of `reg[arg_1]` and `reg[arg_2]`, storing the result in `reg[arg_0]`.

Logical Or

Prevalence	1
Num Args	3

Performs a logical OR of `reg[arg_1]` and `reg[arg_2]`, storing the result in `reg[arg_0]`.

Not Equal

Prevalence	1
Num Args	3

Checks whether `reg[arg_1]` is not equal to `reg[arg_2]` and stores the result in `reg[arg_0]`.

Global Anchor

Prevalence	15
Num Args	0

Marks a module-begin position. Based on tag-lookup, new cores or global jump instructions may set the program counter to this instruction's program position.

This instruction can also mark a module-end position — executing this instruction can terminate the executing core. If no local anchor instruction is present between the current global anchor instruction and the preceding global anchor instruction, this operation will not terminate the executing core. (This way, several global anchors may lead into the same module.)

However, if a local anchor instruction is present between the current global anchor instruction and the preceding global anchor instruction, this operation will terminate the executing core. Local jump instructions will only consider local anchors between the preceding global anchor and the subsequent global anchor instruction.

Global Jump If

Prevalence	1
Num Args	2

Jumps the current core to a global anchor that matches the instruction tag if `reg[arg_0]` is nonzero. If `reg[arg_1]` is nonzero, resets registers.

Global Jump If Not

Prevalence	1
Num Args	2

Jumps the current core to a global anchor that matches the instruction tag if `reg[arg_0]` is nonzero. If `reg[arg_1]` is zero, resets registers.

Protected Regulator Adjust

Prevalence	1
Num Args	1

Adjusts the regulator value of global jump table tags matching this instruction's tag by the amount `reg[arg_0]`.

This regulator value affects the outcome of tag lookup for internal events and signals from the environment. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Protected Regulator Decay

Prevalence	1
Num Args	1

Ages the regulator decay countdown of global jump table tags matching this instruction's tag by the amount `reg[arg_0]`. If `reg[arg_0]` is negative, this can forestall decay.

This decay countdown affects the outcome of tag lookup for internal events, and signals from the environment. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Protected Regulator Get

Prevalence	1
Num Args	1

Gets the regulator value of the global jump table tag that best matches this instruction's tag. Stores the value in `reg[arg_0]`.

If no tag matches, a no-op is performed.

The regulator value gotten controls internal events and signals from the environment. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Protected Regulator Set

Prevalence	1
Num Args	1

Sets the regulator value of global jump table tags matching this instruction's tag to `reg[arg_0]`.

This regulator value affects the outcome of tag lookup for internal events and signals from the environment. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Local Anchor

Prevalence	20
Num Args	0

Marks a program location local jump instructions may route to. This program location is tagged with the instruction's tag.

As described in Section H.2, this operation also plays a role in determining whether global anchor instructions close a module.

Local Jump If

Prevalence	1
Num Args	1

Jumps to a local anchor that matches the instruction tag if `reg[arg_0]` is nonzero.

Local Jump If Not

Prevalence	1
Num Args	1

Jumps to a local anchor that matches the instruction tag if `reg[arg_0]` is zero.

Local Regulator Adjust

Prevalence	1
Num Args	1

Adjusts the regulator value of local jump table tags matching this instruction's tag by the amount `reg[arg_0]`.

Local Regulator Decay

Prevalence	1
Num Args	1

Ages the regulator decay countdown of local jump table tags matching this instruction's tag by the amount `reg[arg_0]`. If `reg[arg_0]` is negative, this can forestall decay.

Local Regulator Get

Prevalence	1
Num Args	1

Gets the regulator value of the local jump table tag that best matches this instruction's tag. Stores the value in `reg[arg_0]`.

If no tag matches, a no-op is performed.

Local Regulator Set

Prevalence	1
Num Args	1

Sets the regulator value of global jump table tags matching this instruction's tag to `reg[arg_0]`.

Decrement

Prevalence	1
Num Args	1

Takes `reg[arg_0]`, decrements it by one, and stores the result in `reg[arg_0]`.

Increment

Prevalence	1
Num Args	1

Takes `reg[arg_0]`, increments it by one, and stores the result in `reg[arg_0]`.

Negate

Prevalence	1
Num Args	1

Negates `reg[arg_0]` and stores the result in `reg[arg_0]`.

Not

Prevalence	1
Num Args	1

Performs a logical not on `reg[arg_0]` and stores the result in `reg[arg_0]`.

Random Bool

Prevalence	1
Num Args	1

Stores `1.0f` to `reg[arg_0]` with probability determined by this instruction's tag. Otherwise, stores `0.0f` to `reg[arg_0]`.

Random Draw

Prevalence	1
Num Args	1

Stores a randomly drawn float value to `reg[arg_0]`.

Terminal

Prevalence	50
Num Args	1

Stores a genetically-encoded value to `reg[arg_0]`. This value is determined deterministically using the instruction's tag.

Exposed Regulator Adjust

Prevalence	1
Num Args	1

Adjusts the regulator value of global jump table tags matching this instruction's tag by the amount `reg[arg_0]`.

This regulator value affects the outcome of tag lookup for messages from neighbor cells. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Exposed Regulator Decay

Prevalence	1
Num Args	1

Ages the regulator decay countdown of global jump table tags matching this instruction's tag by the amount `reg[arg_0]`. If `reg[arg_0]` is negative, this can forestall decay.

This decay countdown affects the outcome of tag lookup for messages from neighbor cells. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Exposed Regulator Get

Prevalence	1
Num Args	1

Gets the regulator value of the global jump table tag that best matches this instruction's tag. Stores the value in `arg[0]`.

If no tag matches, a no-op is performed.

The regulator value gotten controls messages from other cells. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Exposed Regulator Set

Prevalence	1
Num Args	1

Sets the regulator value of global jump table tags matching this instruction's tag to `reg[arg_0]`.

This regulator value affects the outcome of tag lookup for messages from other cells. (Note, as described in B, that independent tag lookup tables handle activating genome modules across different contexts.)

Add to Own State

Prevalence	5
Num Args	1

Adds `reg[arg_0]` to the current value in a target writable state then stores the sum back in to that target writable state.

To determine the target writable state, interprets the first 32 bits of the instruction tag as an unsigned integer then calculates the remainder of integer division by the number of writable states.

Broadcast Intra Message If

Prevalence	1
Num Args	1

If `reg[arg_0]` is nonzero, generates a message tagged with the instruction's tag that contains the core's current register state. Broadcasts this message to every other cardinal processors within the cell.

Multiply Own State

Prevalence	5
Num Args	1

Multiplies `reg[arg_0]` by the current value in a target writable state then stores the result back in to that target writable state.

To determine the target writable state, interprets the first 32 bits of the instruction tag as an unsigned integer then calculates the remainder of integer division by the number of writable states.

Read Neighbor State

Prevalence	10
Num Args	1

Reads a target readable state from the neighboring cell and stores it into `reg[arg_0]`.

To determine the target readable state, interprets the first 32 bits of the instruction tag as an unsigned integer then calculates the remainder of integer division by the number of readable states.

Read Own State

Prevalence	20
Num Args	1

Reads a target readable state and stores it into `reg[arg_0]`.

To determine the target readable state, interprets the first 32 bits of the instruction tag as an unsigned integer then calculates the remainder of integer division by the number of readable states.

Send Inter Message If

Prevalence	5
Num Args	1

If `reg[arg_0]` is nonzero, generates a message tagged with the instruction's tag that contains the core's current register state. Sends this message to the neighboring cell.

Send Intra Message If

Prevalence	5
Num Args	1

If `reg[arg_0]` is nonzero, generates a message tagged with the instruction's tag that contains the core's current register state. Sends this message to a target cardinal processor within the cell.

To determine the target cardinal processor, sums instruction arguments then calculates the remainder of integer division by the number of other same-cell cardinal processors.

Write Own State If

Prevalence	5
Num Args	2

If `reg[arg_1]` is nonzero, stores `reg[arg_0]` into a target writable state.

To determine the target writable state, interprets the first 32 bits of the instruction tag as an unsigned integer then calculates the remainder of integer division by the number of writable states.

H.3 Introspective State

Introspective state refers to the collection of simulation-generated sensor values that evolving programs running within each cardinal processor can access via read-only operations. Each cardinal processor has an independent copy of each piece of introspective state state. (However, some introspective states representing cell state are set to identical values across cardinal processors within the same cell.)

Each cardinal processor's introspective state regularly copied and dispatched to that cardinal processor's neighbor cell, where it serves as read-only extrospective state.

Introspective state is organized into two categories:

1. raw introspective state, and
2. interpreted introspective state.

Raw introspective state directly exposes aspects of simulation state. Interpreted introspective state is filled with truthy values that are interpreted as booleans to dispatch environmentally-managed events.

See `https://github.com/mmore500/dishtiny/tree/prq49/include/dish2/peripheral/readable_state/introspective_state` for source code implementing introspective state.

Is Child Cell Of

Type	`char` (w/ boolean semantics)
Category	interpreted

Did this cell spawn from this cardinal processor's neighbor cell?

Is Child Group Of (0 thru $L-1$)

Type	`char` (w/ boolean semantics)
Category	interpreted

Does this cell's kin group ID descend directly from the neighbor's kin group ID?

Is Newborn

Type	`char` (w/ boolean semantics)
Category	interpreted

Is this cell's age less than `EVENT_LAUNCHING_SERVICE_FREQUENCY`?

Is Parent Cell Of

Type	`char` (w/ boolean semantics)
Category	interpreted

Did this cardinal processor's neighbor cell spawn from this cell? That is, was neighbor was spawned from this cell and is this cell older than neighbor?

Is Parent Group Of

Type	`char` (w/ boolean semantics)
Category	interpreted

Did this cell's kin group descend directly from the cardinal processor's neighbor cell's kin group? That is, is cell's kin group ancestor ID(s) equal to the cardinal processor's neighbor's current kin group ID(s).

Kin Group Match (0 thru $L-1$)

Type	`char` (w/ boolean semantics)
Category	interpreted

Does this cell's kin group ID match the neighbor's kin group ID?

Kin Group will Expire (0 thru $L-1$)

Type	`char` (w/ boolean semantics)
Category	interpreted

Is this cell's kin group age greater than 80% of this level's `GROUP_EXPIRATION_DURATIONS`?

Neighbor Apoptosis Was the neighbor tile's most recent death apoptosis?

Type	`char` (w/ boolean semantics)
Category	interpreted

Neighbor Fragmented Was group fragmentation the most recent cause of death in the cardinal processor's neighbor cell?

Type	`char` (w/ boolean semantics)
Category	interpreted

Neighbor Is Newborn

Type	`char` (w/ boolean semantics)
Category	interpreted

Is the neighbor's cell age less than `EVENT_LAUNCHING_SERVICE_FREQUENCY`?

Neighbor Kin Group Will Expire (0 thru $L-1$) interpreted

Type	`char` (w/ boolean semantics)
Category	interpreted

Is this cell's kin group age greater than 80% of the this level's `GROUP_EXPIRATION_DURATIONS`?

Neighbor Optimum Quorum Exceeded (0 thru $L-1$)

Type	`char` (w/ boolean semantics)
Category	interpreted

Is this cardinal processor's neighbor cell's kin group quorum count more than the simulation-defined target count `OPTIMAL_QUORUM_COUNT`?

Optimum Quorum Exceeded (0 thru $L-1$)

Type	`char` (w/ boolean semantics)
Category	interpreted

Is this cell's kin group quorum count more than the simulation-defined target count `OPTIMAL_QUORUM_COUNT`?

Parent Fragmented

Type	`char` (w/ boolean semantics)
Category	interpreted

Did the cell's parent die from fragmentation? That is, was the last cause of death on the current tile was fragmentation?

Phylogenetic Root Match

Type	`char` (w/ boolean semantics)
Category	interpreted

Does this cell's root ID equal the cardinal processor's cell neighbor's root ID? (This means they originate from the same seed ancestor.)

Richer Than Neighbor

Type	`char` (w/ boolean semantics)
Category	interpreted

Does this cell's stockpile more resource than the cardinal processor's cell neighbor?

Stockpile Depleted

Type	`char` (w/ boolean semantics)
Category	interpreted

Is this cell's stockpile less than twice the base harvest rate?

Stockpile Fecund

Type	`char` (w/ boolean semantics)
Category	interpreted

Does this cell have enough resource stockpiled to fund spawning an offspring cell?

Cell Age

Type	`size_t`
Category	raw

Number CellAgeService calls elapsed since cell was born.

Epoch

Type	`size_t`
Category	raw

Updates elapsed since start of simulation.

Incoming Inter Message Counter

Type	`size_t`
Category	raw

Counter tracking incoming messages from cardinal processor's neighbor cell. Intermittently reset to zero.

Incoming Intra Message Counter

Type	`size_t`
Category	raw

Counter of incoming messages from other cardinal processors within the cell. Intermittently reset to zero.

Is Alive

Type	`char` (w/ boolean semantics)
Category	raw

Whether the cell is alive. Although trivial as introspective state, this state is useful for neighbor cell's extrospective state.

Kin Group Age (0 thru $L-1$)

Type	`size_t`
Category	raw

Number of epochs elapsed since kin group formation.

Kin Group ID Ancestor View (0 thru $L-1$)

Type	`size_t`
Category	raw

Kin group ID from which cell's kin group ID is descended.

Kin Group ID View (0 thru $L-1$)

Type	`size_t`
Category	raw

Kin group ID of this cell.

Most Recent Cause of Death raw

Type	`char`
Category	raw

What was this the most recent cause of death on this tile? Encoded using the `CauseOfDeath` enum.

Num Known Quorum Bits (0 thru $L-1$)

Type	`size_t`
Category	raw

What is this cell's known quorum count? (How many unique quorum bits collected from kin group members are known?)

Phylogenetic Root View

Type	`size_t`
Category	raw

What is this cell's phylogenetic root ID?

(Which initially-generated ancestor is this cell descended from?)

Received Resource From

Type	`float`
Category	raw

How much resource is being received from the cardinal processor's cell neighbor?

Resource Stockpile

Type	`float`
Category	raw

Amount of resource this cell has.

Spawn Count

Type	`float`
Category	raw

Number of offspring generated from this cell and sent to the cardinal processor's neighbor tile. Includes offspring that do not successfully take into the neighbor tile or have not survived.

Spawned From

Type	`char` (w/ boolean semantics)
Category	raw

Did this cell spawn from this cardinal processor's neighbor cell?

H.4 Writable State

Writable state refers to the collection of output values that evolving programs running within each cardinal processor can write to and read from. Some of these outputs enable interaction with the simulation (i.e., control phenotypic characteristics). Each cardinal processor has an independent copy of each piece of writable state state.

See `https://github.com/mmore500/dishtiny/tree/prq49/include/dish2/peripheral/readable_state/writable_state` for source code implementing writable state.

Nop State ($4\times$) | Type | `float`
Writing to this state has no external effect. It can be used as global memory shared between cores.

Transient Nop State ($4\times$) | Type | `float`
Writing to this state has no external effect. It is cleared regularly by the decay to baseline service. It can be used as temporary global memory shared between cores.

Apoptosis Request | Type | `char`
Writing a nonzero value to this state causes cell death.

Heir Request | Type | `char`
If this state is set when cell death occurs, the cardinal processor's neighbor cell will inherit leftover resource from the cell's stockpile.

RepLev Request (0 thru L) | Type | `char`
Controls kin group inheritance for daughter cells spawned to this cardinal processor's neighbor tile.

If no copies of this state are set at cell spawn, the daughter cell will have no common kin group IDs. If one copy of this state is set at cell spawn, the daughter cell will have one common kin group ID. If L copies of this state are set at cell spawn, the daughter cell will have L common kin group IDs.

Resource Receive Resistance | Type | `float`
Setting this state reduces the amount of resource received from the cardinal processor's neighbor cell.

Resource Reserve Request | Type | `float`
Setting this state prevents that amount of stockpiled resource from being drawn from to be sent to the cardinal processor's neighbor cell.

Resource Send Limit | Type | `float`
Setting this state caps the amount of resource that this cell can send to the cardinal processor's neighbor cell per update.

Resource Send Request | Type | `float`
Setting this state initiates resource sharing to the cardinal processor's neighbor cell. The value stored controls the amount of resource shared.

Spawn Arrest | Type | `char`
Setting this state prevents this cell from spawning offspring into this cardinal processor's neighbor tile, even if sufficient resource is available.

Spawn Request | Type | `char`
Setting this state attempts to initiate spawning offspring into this cardinal processor's neighbor tile.

H.5 Cellular Simulation Services

Simulation logic is applied to each cell through a collection of distinct functors, referred to as services.

All services specified to run on a particular update are applied in sequence to a single cell. (Some services run only every nth update.) Then, to another randomly-chosen cell in a `thread_local` population, and another until the entire population has been updated.

See `https://github.com/mmore500/dishtiny/tree/prq49/include/dish2/services` for source code implementing these services.

Decay to Baseline Service | Frequency | every 32 update(s)
Decays a cell's global regulators, resets its controller-mapped peripheral states, and resets its transient NOP states.

Running Log Purge Service | Frequency | every 64 update(s)
Purges a cell's running logs. (Only affects data collection, not simulation logic.)

Controller Mapped State Noise Service | Frequency | every 8 update(s)
Given a non-zero controller-mapped state defect rate, picks a random number n from a Poisson distribution parameterized by `CONTROLLER_MAPPED_STATE_DEFECT_RATE`. Then, it introduces n defects to a cell's writable state. Half of these defects zero out the state and half randomize it.

Interpreted Introspective State Refresh Service | Frequency | every 1 update(s)
Refreshes the interpreted introspective state of a cell.

Extrospective State Exchange Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of extrospective state. (Not part of core simulation logic.)

Extrospective State Rotate Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of extrospective state. (Not part of core simulation logic.)

Introspective State Exchange Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of introspective state. (Not part of core simulation logic.)

Introspective State Rotate Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of introspective state. (Not part of core simulation logic.)

CPU Execution Service | Frequency | every 1 update(s)
Executes a cell's genome on its cardinals processors for `HARDWARE_EXECUTION_CYCLES` virtual cycles. The order of cardinal evaluation is randomized. This is repeated `HARDWARE_EXECUTION_ROUNDS` times.

Event Launching Service | Frequency | every 8 update(s)
Dispatches environmentally-managed events for each cardinal processor.

Introspective State Rotate Restore Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of introspective state. (Not part of core simulation logic.)

Introspective State Exchange Restore Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of introspective state. (Not part of core simulation logic.)

Extrospective State Rotate Restore Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of extrospective state. (Not part of core simulation logic.)

Extrospective State Exchange Restore Service | Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of extrospective state. (Not part of core simulation logic.)

Writable State Exchange Service | Frequency | every 1 update(s)

Used for experimental manipulations testing the fitness effect of writable state. (Not part of core simulation logic.)

Writable State Rotate Service
Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of writable state. (Not part of core simulation logic.)

Birth Setup Service Frequency | every 16 update(s)
Births a new cell into the current cell.

This occurs by first iterating through the cell's cardinal processors' birth request inputs in random order. While the cell's resource stockpile is greater than the `SPAWN_DEFENSE_COST`, the requests are ignored and the stockpile depleted by that cost. The first request that cannot be defended against is then acted upon. The current cell's death routine is called, the old genome is replaced by the incoming genome, and the cell's make-alive routine is called.

Cell Age Service Frequency | every 1 update(s)
Advances the cell age introspective state and refreshes kin group age introspective state.

Collective Harvesting Service
Frequency | every 4 update(s)
Calculates the total amount of resource collectively harvested to this cell by the cell's kin group. This amount increases with quorum count and saturates at `OPTIMAL_QUORUM_COUNT`. Adds the harvested amount to the cell's resource stockpile.

Collective Resource Decay Service
Frequency | every 1 update(s)
If the cell's quorum count exceeds `OPTIMAL_QUORUM_COUNT`, applies multiplicative decay to the cell's resource stockpile. This effect strengthens exponentially with excess cell quorum count.

Conduit Flush Service Frequency | every 16 update(s)
Flushes each cardinal processors' inter-process and inter-thread output conduits.

Inter Message Launching Service
Frequency | every 8 update(s)
Launches new virtual cores to process incoming inter-cell messages.

Inter Message Purging Service
Frequency | every 8 update(s)
Purges excess incoming inter-cell messages that couldn't be handled due to virtual core availability.

Intra Message Launching Service
Frequency | every 8 update(s)
Launches new virtual cores to process incoming messages from same-cell cardinal processors.

Message Counter Clear Service
Frequency | every 16 update(s)
Intermittently resets introspective message count state.

Quorum Service Frequency | every 1 update(s)
Performs distributed estimation of kin group size by simulation.

Each cell has a single randomly-chosen index set within a fixed-length bitstring. (Depending in parameter settings, some cells may have index set — all positions within the bitstring are zeroed out.)

Broadcasts bits known to be set are to all neighbor cells within the same kin group. Incoming bitstrings from neighbors are ORed with known bits.

The original neighbor each non-self bit was first learned from is recorded alongside that bit. If that neighbor no longer broadcasts that bit, it is erased from the cell's known bits.

Updates latest quorum count into introspective state.

This scheme is replicated independently for each kin group level simulated.

Resource Decay Service Frequency | every 1 update(s)
Decays cell resource stockpile multiplicatively by `RESOURCE_DECAY` constant.

Resource Harvesting Service
Frequency | every 1 update(s)
Adds a constant amount to cell's resource stockpile.

Resource Receiving Service Frequency | every 4 update(s)
Calculates total amount of resource received across every cardinal processor, and then adds that total to resource stockpile.

If the cell is not alive, it instead refunds all received resources back to each sending cell.

Resource Sending Service Frequency | every 1 update(s)
Based on writable state within each cardinal processor, calculates and dispatches resource that should be shared to each neighbor cell.

Spawn Sending Service Frequency | every 16 update(s)
If available resource is greater than or equal to 1.0, iterates randomly through every cardinal processor to determine whether it requested to spawn and has not arrested spawning. Then, one of these requests is dispatched at random and stockpile is decreased by one.

State Input Jump Service Frequency | every 8 update(s)
Pulls a fresh copy of each neighboring cardinal processor's current readable state.

State Output Put Service Frequency | every 8 update(s)
Dispatches a copy of each cardinal processor's current readable state to corresponding neighbor cells.

Epoch Advance Service Frequency | every 8 update(s)
The cell's current-known epoch count is advanced by one then set to the maximum of the cell's current-known epoch count and neighbor cells' current-known epoch count.

Writable State Rotate Restore Service
Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of writable state. (Not part of core simulation logic.)

Writable State Exchange Restore Service
Frequency | every 1 update(s)
Used for experimental manipulations testing the fitness effect of writable state. (Not part of core simulation logic.)

Group Expiration Service Frequency | every 64 update(s)
As group age exceeds `GROUP_EXPIRATION_DURATIONS`, with increasing probability fragments cell from its kin group. This process kills the cell and replaces it in place with a daughter without kin ID commonality.

Apoptosis Service Frequency | every 16 update(s)
If any cardinal processors have requested apoptosis, do death routine on the cell.

H.6 Threadlocal Simulation Services

Actions that are performed on each `thread_local` population.

See `https://github.com/mmore500/dishtiny/tree/prq49/include/dish2/services_threadlocal` for source code implementing these services.

Cell Update Service | Frequency | every 1 update(s)

Performs each cell's simulation services, iterating over cells in randomized order.

Diversity Maintenance Service | Frequency | every 8 update(s)

Prevents any one originally-generated ancestor from sweeping the population, preserving deep phylogenetic diversity.

Counts cells that descend from each originally-seeded ancestor. If more than `DIVERSITY_MAINTENANCE_PREVALENCE` of cells descend from a single seeded ancestor, decay their resource stockpiles. The magnitude of this effect increases with excess prevalence.

Stint Diversity Maintenance Service | Frequency | every n/a update(s)

Prevents any one seeded or reconstituted stint-originating ancestor from sweeping the population, preserving phylogenetic diversity within a single stint.

Counts cells that descend from each seeded or reconstituted stint-originating ancestor. If more than `STINT_DIVERSITY_MAINTENANCE_PREVALENCE` of cells descend from a single seeded or reconstituted ancestor, decay their resource stockpiles. The magnitude of this effect increases with excess prevalence.

H.7 Runtime Parameters

This section enumerates simulation parameters and provides default settings that were used.

See `https://github.com/mmore500/dishtiny/blob/prq49/include/dish2/config/ConfigBase.hpp` for source code defining run time parameters.

Some parameter settings were overridden in some assays. See `https://github.com/mmore500/dishtiny/tree/prq49/configpacks/bucket=prq49+diversity=0.50_series+mut_freq=1.00+mut_sever=1.00` for configuration files used in each assay and `https://github.com/mmore500/dishtiny/tree/prq49/slurm` for runscripts used in each assay.

EXECUTION

N_THREADS

Type	size_t
Default	4

How many threads should we run with?

RUN_UPDATES

Type	bool
Default	false

Should we run evolution or skip directly to post-processing and data collection?

RUN_UPDATES

Type	size_t
Default	0

How many updates should we run the experiment for?

RUN_SECONDS

Type	double
Default	0

How many updates should we run the experiment for?

MAIN_TIMEOUT_SECONDS

Type	double
Default	10800

After how many seconds should we time out and fail with an error?

END_SNAPSHOT_TIMEOUT_SECONDS

Type	double
Default	1200

After how many seconds should the end snapshot timeout?

LOG_FREQ

Type	double
Default	20

How many seconds should pass between logging progress?

ASYNCHRONOUS

Type	size_t
Default	3

Should updates occur synchronously across threads and processes?

SYNC_FREQ_MILLISECONDS

Type	size_t
Default	100

How often updates occur synchronously across threads and processes for async mode 1?

RNG_PRESEED

Type	utin64_t
Default	std::numeric_limits<uint64_t>::max()

Optionally override the calculated rng preseed.

THROW_ON_EXTINCTION

Type	bool
Default	true

Should we throw an exception if populations go extinct?

EXPERIMENT

RUN_SLUG

Type	std::string
Default	``default''

Run-identifying slug.

PHENOTYPIC_DIVERGENCE_N_UPDATES

Type	size_t
Default	2048

How many updates should we run phenotypic divergence experiments for? If phenotypic divergence is not detected within this many updates, we consider two strains to be phenotypically identical.

PHENOTYPIC_DIVERGENCE_N_CELLS

Type	size_t
Default	100

How many cells should we simulate while testing for phenotypic divergence?

STINT

Type	utin64_t
Default	std::numeric_limits<uint64_t>::max()

How many evolutionary stints have elapsed?

SERIES

Type	utin64_t
Default	std::numeric_limits<uint64_t>::max()

Which evolutionary series are we running?

REPLICATE

Type	std::string
Default	``''

What replicate are we running?

TREATMENT

Type	std::string
Default	``none''

What experimental treatment has been applied?

SEED_FILL_FRACTION

Type	double
Default	1.0

If we are seeding the population, what fraction of available slots should we fill?

GENESIS

Type	std::string
Default	"generate"

How should we initialize the population? Can be "generate" to randomly generate a new population, "reconstitute" to load a population from file, "monoculture" to load a single genome from file, or "inoculate" to load genomes annotated with root ID keyname attributes from file.

DEMOGRAPHICS

N_CELLS

Type	size_t
Default	10000

How many cells should be simulated?

WEAK_SCALING

Type	bool
Default	false

Should number of total cells be multiplied by the total number of threads (num procs times threads per proc)?

N_DIMS

Type	size_t
Default	DISH2_NLEV

What dimensionality should the toroidal mesh have?

GROUP_EXPIRATION_DURATIONS

Type	internal::nlev_size_t_t
Default	internal::nlev_size_t_t{ 256, 1024 }

After how many epochs should groups stop collecting resource?

CELL_AGE_DURATION

Type	size_t
Default	1024

After how many epochs should cells die?

RESOURCE

MIN_START_RESOURCE

Type	float
Default	0.8

How much resource should a cell start with?

MAX_START_RESOURCE

Type	float
Default	0.9

How much resource should a cell start with?

RESOURCE_DECAY

Type	float
Default	0.995

How much resource should remain each update?

APOP_RECOVERY_FRAC

Type	float
Default	0.8

What fraction of REP_THRESH is recovered to heirs after apoptosis?

SPAWN_DEFENSE_COST

Type	float
Default	1.1

What is the cost of repelling an incoming spawn?

HARVEST

BASE_HARVEST_RATE

Type	float
Default	0.02

How much resource should cells accrue per update?

COLLECTIVE_HARVEST_RATE

Type	internal::nlev_float_t
Default	internal::nlev_float_t{0.25}

How much resource should cells accrue per update?

OPTIMAL_QUORUM_COUNT

Type	internal::nlev_float_t
Default	internal::nlev_size_t{12}

What group size does collective harvest work most effectively at?

QUORUM

P_SET_QUORUM_BIT

Type	internal::nlev_float_t
Default	internal::nlev_float_t{1.0}

What fraction of cells should have a quorum bit set?

GENOME

PROGRAM_START_SIZE

Type	size_t
Default	128

How many instructions should initial programs be?

PROGRAM_MAX_SIZE

Type	size_t
Default	4096

How many instructions should programs be capped at?

MUTATION_RATE

Type	internal::nlev_float_t
Default	internal::nreplev_float_t{0.1}

For each replev, what fraction of cells should be mutated at all?

POINT_MUTATION_RATE

Type	float
Default	0.0002

What fraction of bits should be scrambled?

SEQUENCE_DEFECT_RATE

Type	float
Default	0.001

How often should sloppy copy defect occur?

MINOR_SEQUENCE_MUTATION_BOUND

Type	size_t
Default	8

For minor sequence mutations, at most how many instructions should be inserted or deleted?

SEVERE_SEQUENCE_MUTATION_RATE

Type	float
Default	0.001

With what probability should sequence mutation be severe?

HARDWARE

HARDWARE_EXECUTION_ROUNDS

Type	size_t
Default	1

How many hardware evaluation rounds to run per update?

HARDWARE_EXECUTION_CYCLES

Type	size_t
Default	16

How many hardware cycles to run per round?

CONTROLLER_MAPPED_STATE_DEFECT_RATE

Type	float
Default	0.0005

At what rate should bits should be flipped in writable memory?"

SERVICES

APOPTOSIS_SERVICE_FREQUENCY

Type	size_t
Default	16

Run service every ?? updates. Must be a power of 2.

BIRTH_SETUP_SERVICE_FREQUENCY

Type	size_t
Default	16

Run service every ?? updates. Must be a power of 2.

CONDUIT_FLUSH_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

COLLECTIVE_HARVESTING_SERVICE_FREQUENCY

Type	size_t
Default	16

Run service every ?? updates. Must be a power of 2.

CPU_EXECUTION_SERVICE_FREQUENCY

Type	size_t
Default	4

Run service every ?? updates. Must be a power of 2.

GROUP_EXPIRATION_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

RUNNING_LOG_PURGE_SERVICE_FREQUENCY

Type	size_t
Default	64

Run service every ?? updates. Must be a power of 2.

DIVERSITY_MAINTENANCE_SERVICE_FREQUENCY

Type	size_t
Default	64

Run service every ?? updates. Must be a power of 2.

DIVERSITY_MAINTENANCE_PREVALENCE

Type	double
Default	0.25

If an originally-seeded ancestor's descendants constitute more than this fraction of the population, decay their resource stockpiles.

STINT_DIVERSITY_MAINTENANCE_SERVICE_FREQUENCY

Type	size_t
Default	0

Run service every ?? updates. Must be a power of 2.

STINT_DIVERSITY_MAINTENANCE_PREVALENCE

Type	double
Default	0.25

If a seeded or reconstituted stint-originating ancestor's descendants constitute more than this fraction of the population, decay their resource stockpiles.

DECAY_TO_BASELINE_SERVICE_FREQUENCY

Type	size_t
Default	32

Run service every ?? updates. Must be a power of 2.

EPOCH_ADVANCE_SERVICE_FREQUENCY

Type	size_t
Default	8

Run service every ?? updates. Must be a power of 2.

EVENT_LAUNCHING_SERVICE_FREQUENCY

Type	size_t
Default	8

Run service every ?? updates. Must be a power of 2. Must be ¿ 1.

INTERMITTENT_CPU_RESET_SERVICE_FREQUENCY

Type	size_t
Default	64

Run service every ?? updates. Must be a power of 2.

INTERMITTENT_STATE_PERTURB_SERVICES_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

INTER_MESSAGE_COUNTER_CLEAR_SERVICE_FREQUENCY

Type	size_t
Default	16

Run service every ?? updates. Must be a power of 2.

INTER_MESSAGE_LAUNCHING_SERVICE_FREQUENCY

Type	size_t
Default	8

Run service every ?? updates. Must be a power of 2.

INTRA_MESSAGE_LAUNCHING_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

STATE_OUTPUT_PUT_SERVICE_FREQUENCY

Type	size_t
Default	8

Run service every ?? updates. Must be a power of 2.

PUSH_SERVICE_FREQUENCY

Type	size_t
Default	16

Run service every ?? updates. Must be a power of 2.

QUORUM_CAP_SERVICE_FREQUENCY

Type	size_t
Default	16

Run service every ?? updates. Must be a power of 2.

QUORUM_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

RESOURCE_DECAY_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

RESOURCE_HARVESTING_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

RESOURCE_INPUT_JUMP_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

RESOURCE_RECEIVING_SERVICE_FREQUENCY

Type	size_t
Default	4

Run service every ?? updates. Must be a power of 2.

RESOURCE_SENDING_SERVICE_FREQUENCY

Type	size_t
Default	1

Run service every ?? updates. Must be a power of 2.

SPAWN_SENDING_SERVICE_FREQUENCY

Type	size_t
Default	16

Run service every ?? updates. Must be a power of 2.

STATE_INPUT_JUMP_SERVICE_FREQUENCY

Type	size_t
Default	8

Run service every ?? updates. Must be a power of 2.

CONTROLLER_MAPPED_STATE_NOISE_SERVICE_FREQUENCY

Type	size_t
Default	8

Run service every ?? updates. Must be a power of 2.

DATA

PHENOTYPE_EQUIVALENT_NOPOUT

Type	bool
Default	false

Should we make and record a phenotype equivalent nopout strain at the end of the run? Must also enable ARTIFACTS_DUMP.

BATTLESHIP_PHENOTYPE_EQUIVALENT_NOPOUT

Type	bool
Default	false

Should we make and record a phenotype equivalent nopout strain at the end of the run? Must also enable ARTIFACTS_DUMP.

JENGA_PHENOTYPE_EQUIVALENT_NOPOUT

Type	bool
Default	false

Should we make and record a phenotype equivalent nopout strain at the end of the run? Must also enable ARTIFACTS_DUMP.

JENGA_NOP_OUT_SAVE_PROGRESS_AND_QUIT_SECONDS

Type	size_t
Default	10800

After how many seconds should we save nop-out progress and quit?

TEST_INTERROOT_PHENOTYPE_DIFFERENTIATION

Type	bool
Default	false

Should we test for phenotype differentiation between roots?

ALL_DRAWINGS_WRITE

Type	bool
Default	false

Should we generate and record drawings of the final state of the simulation? Must also enable DATA_DUMP.

DATA_DUMP

Type	bool
Default	false

Should we record data on the final state of the simulation?

RUNNINGLOGS_DUMP

Type	bool
Default	false

Should we dump running logs at the end of the simulation? Must also enable DATA_DUMP.

CENSUS_WRITE

Type	bool
Default	false

Should we write the cell census at the end of the simulation? Must also enable DATA_DUMP.

ARTIFACTS_DUMP

Type	bool
Default	false

Should we record data on the final state of the simulation?

BENCHMARKING_DUMP

Type	bool
Default	false

Should we record data for benchmarking the simulation?

ROOT_ABUNDANCES_FREQ

Type	size_t
Default	0

How many updates should elapse between recording phylogenetic root abundances? If 0, never record phylogenetic root abundances. Must be power of two.

ABORT_IF_COALESCENT_FREQ

Type	size_t
Default	0

How many updates should elapse between checking for coalescence? If 0, never check for coalescence. Must be power of two.

ABORT_IF_EXTINCT_FREQ

Type	size_t
Default	0

How many updates should elapse between checking for coalescence? If 0, never check for coalescence. Must be power of two.

ABORT_AT_LIVE_CELL_FRACTION

Type	double
Default	0.0

Should we terminate once a live cell fraction is reached? If 0, will not terminate.

REGULATION_VIZ_CLAMP

Type	double
Default	10.0

What bounds should we clamp regulation values into before running PCA visualization?

RUNNING_LOG_DURATION

Type	size_t
Default	4

How many purge epochs should we keep events in the running log?

SELECTED_DRAWINGS_FREQ

Type	size_t
Default	0

How many updates should elapse between outputting snapshot images?

DRAWING_WIDTH_PX

Type	double
Default	500.0

What should the width of the drawings be, in pixels?

DRAWING_HEIGHT_PX

Type	double
Default	500.0

What should the height of the drawings be, in pixels?

SELECTED_DRAWINGS

Type	std::string
Default	`\\\`

What drawings should be drawn? Provide slugified drawer names separated by colons.

ANIMATE_FRAMES

Type	bool
Default	false

Should we stitch the output images into a video? Only valid if DRAWING_FREQ is not 0.

VIDEO_FPS

Type	size_t
Default	16

How many frames per second should the video be?

VIDEO_MAX_FRAMES

Type	size_t
Default	960

At most how many frames should output video include?

H.8 Compile Time Parameters

This section enumerates simulation parameters and provides default settings that were used.

See `https://github.com/mmore500/dishtiny/blob/prq49/include/dish2/spec/Spec_prq49.hpp` for source code defining compile time parameters.

NLEV

Type	size_t
Value	2

How many hierarchical kin group levels should be simulated?

AMT_NOP_MEMORY

Type	size_t
Value	4

How many nop and transient nop states should exist in the peripheral?

STATE_EXCHANGE_CHAIN_LENGTH

Type	size_t
Value	128

How many callees should we displace state by in state exchange experiments?

sgpl_spec_t::num_cores

Type	size_t
Value	32

How many virtual cores should each cardinal processor's virtual CPU be able to support?

sgpl_spec_t::num_fork_requests

Type	size_t
Value	3

How many fork requests can a virtual core make at most?

sgpl_spec_t::num_registers

Type	size_t
Value	8

How many registers should each virtual core contain?

sgpl_spec_t::switch_steps

Type	size_t
Value	8

Maximum num steps executed on one core before next core is executed.

sgpl_spec_t::global_matching_t

Type	typedef
Value	`emp::MatchDepository< // program index type unsigned short, // match metric emp::OptimizedApproxDualStreakMetric<64>, // match selector emp::statics::RankedSelector< // match threshold std::ratio<1, 5> >, // regulator emp::PlusCountdownRegulator< std::deci, // Slope std::ratio<1,4>, // MaxUpreg std::deci, // ClampLeeway 2 // CountdownStart >, true, // raw caching 8 // regulated caching >`

What matching datastructure implementation should we use for global jump tables?

sgpl_spec_t::local_matching_t

Type	typedef

Value	emp::MatchDepository<
	// program index type
	unsigned short,
	// match metric
	emp::OptimizedApproxDualStreakMetric<64>,
	// match selector
	emp::statics::RankedSelector<
	// match threshold
	std::ratio<1, 2>
	>,
	// regulator
	emp::PlusCountdownRegulator<
	std::deci, // Slope
	std::ratio<1,4>, // MaxUpreg
	std::deci, // ClampLeeway
	2 // CountdownStart
	>,
	false, // raw caching
	0 // regulated caching
	>

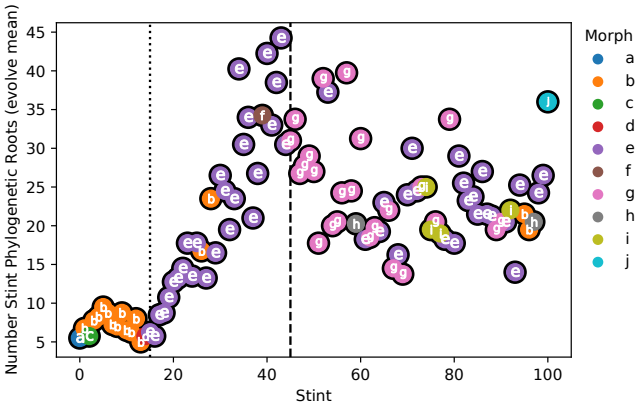
What matching datastructure implementation should we use for local jump tables?

sgpl_spec_t::tag_width

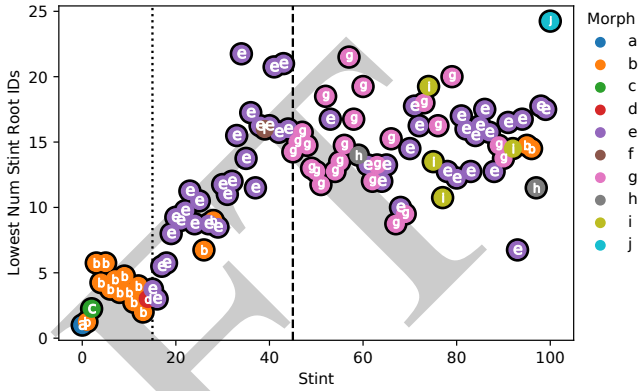
Type	size_t
Value	64

Tag width in bits.

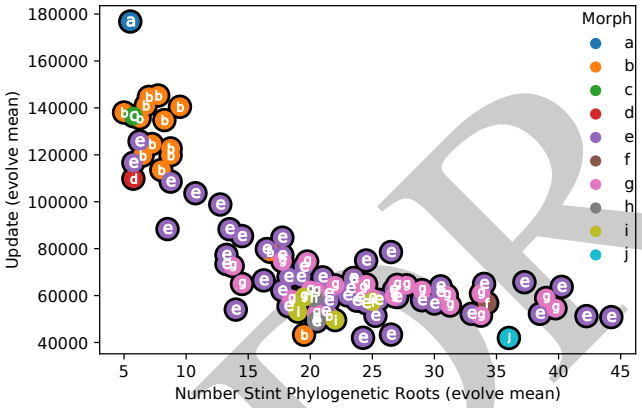
I Supplementary Figures



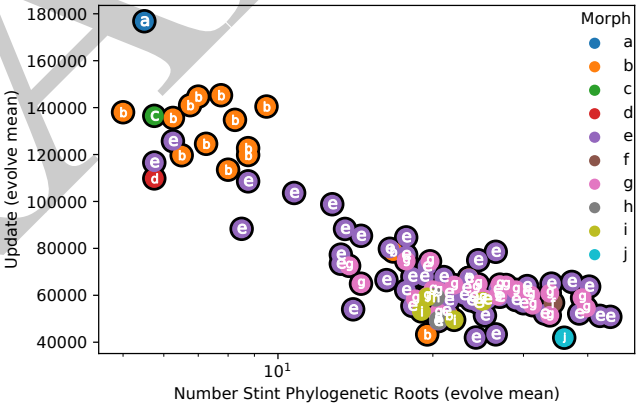
(a) Number of genomes from the beginning of a stint with extant offspring at the end of that stint.



(b) Number of genomes with the lowest surviving original phylogenetic root ID from the beginning of a stint with extant offspring at the end of that stint.



(c) Relationship between the number of simulation updates elapsed in a stint and the number of genomes seeded into a stint with extant descendants at the end of that stint.



(d) Relationship between the number of simulation updates elapsed in a stint and the number of genomes seeded into a stint with extant descendants at the end of that stint, log axis.

Figure 15: **Phylogenetic statistics.** Color coding and letters correspond to qualitative morph codes described in Table 1. Dotted vertical line denotes emergence of morph e. Dashed vertical line denotes emergence of morph g.

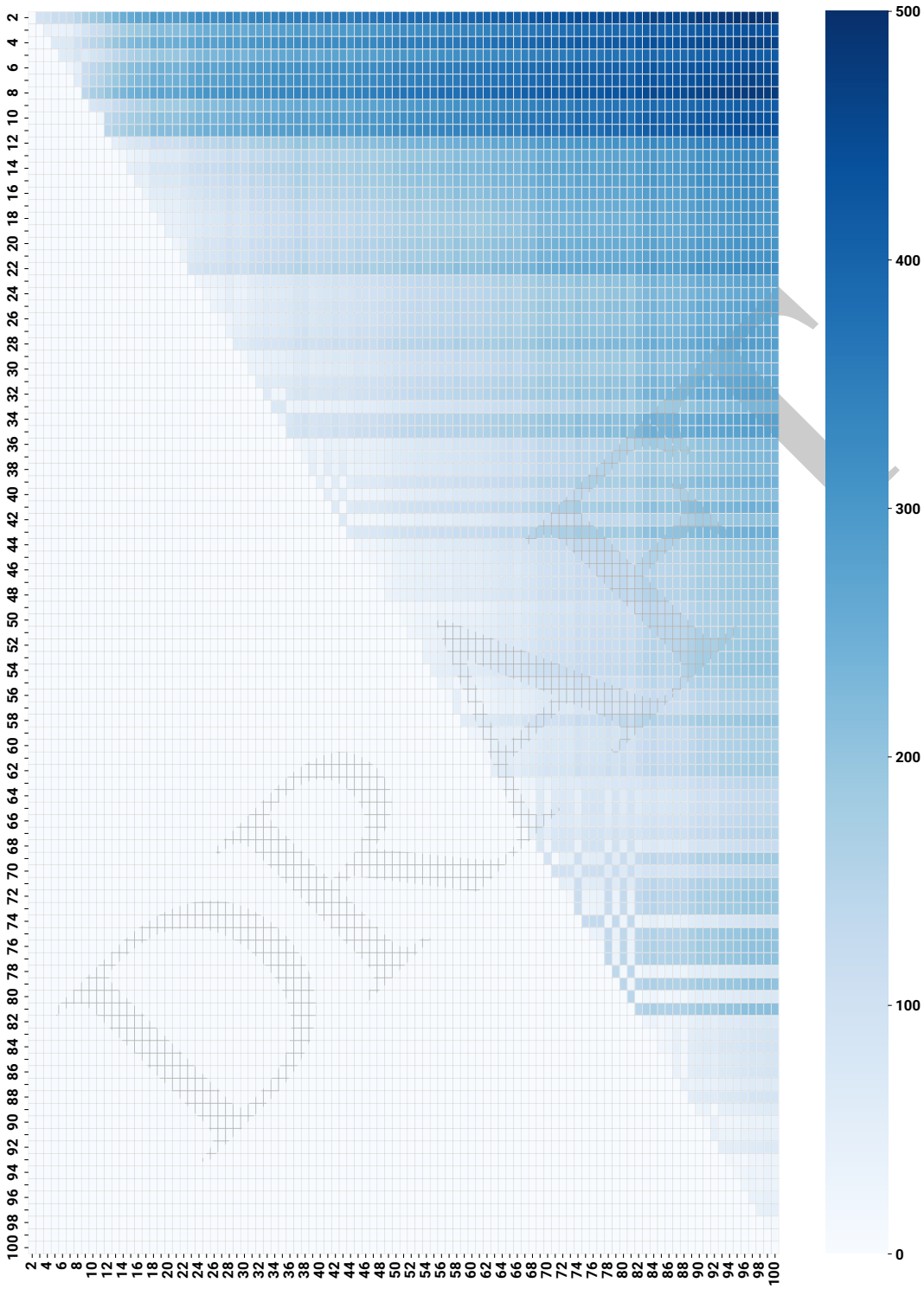
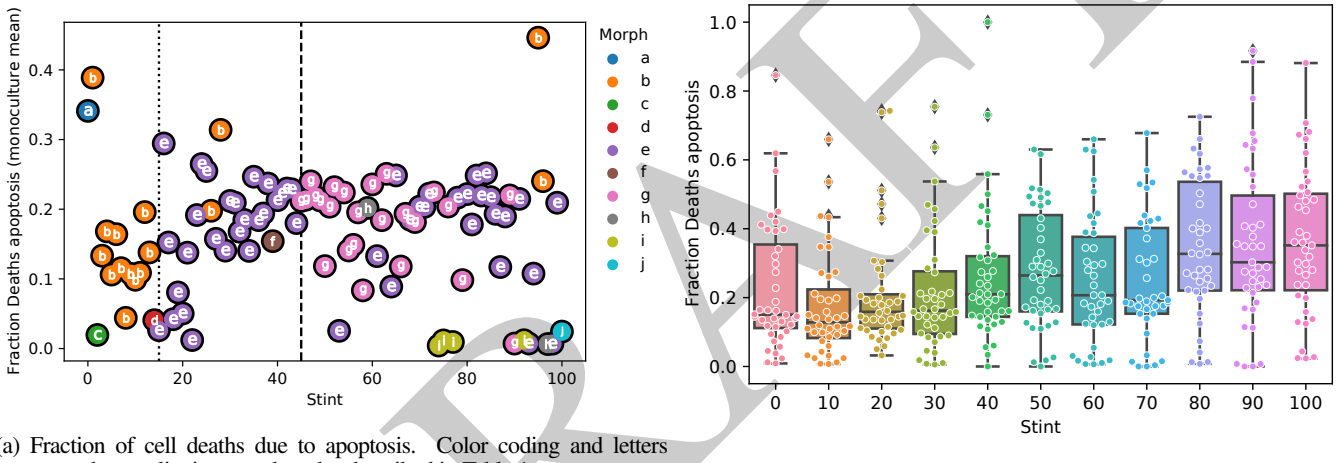


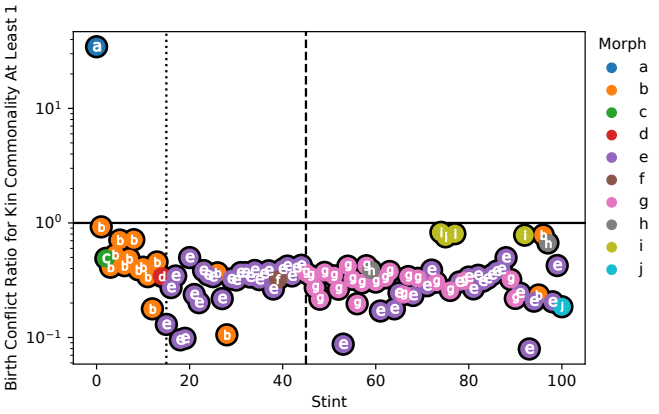
Figure 16: Point mutation distance between tag blocks from sampled focal strain representatives across stints. These distances were used to reconstruct phylogenetic relationship of sampled focal strain representatives. Light white indicates high similarity between tag blocks and dark blue indicates low similarity. Representatives from stints 0 and 1, which share no common ancestry with representatives from other stints, are excluded. Only upper triangular data plotted.



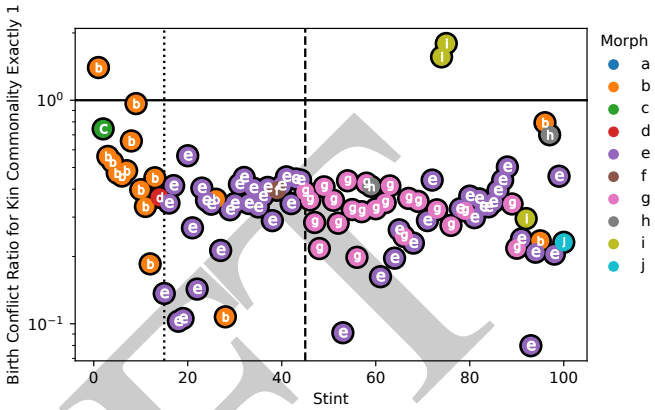
(a) Fraction of cell deaths due to apoptosis. Color coding and letters correspond to qualitative morph codes described in Table 1.

(b) Distribution of apoptosis rates across evolutionary replicates.

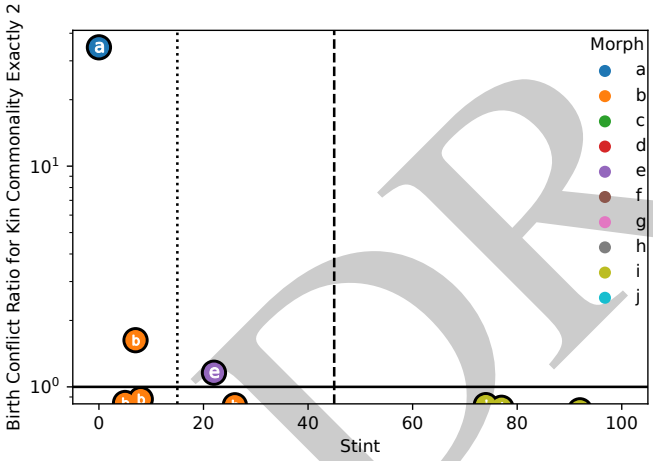
Figure 17: **Apoptosis rates.** In case study strain and across evolutionary replicates.



(a) Frequency at which cell proliferation replaces neighbors with any kin group commonality to the parent, normalized for the relative frequency of neighbors with kin group commonality to the parent. Values below 1.0 (horizontal bar) indicate preference to displace neighbors with no kin group commonality.

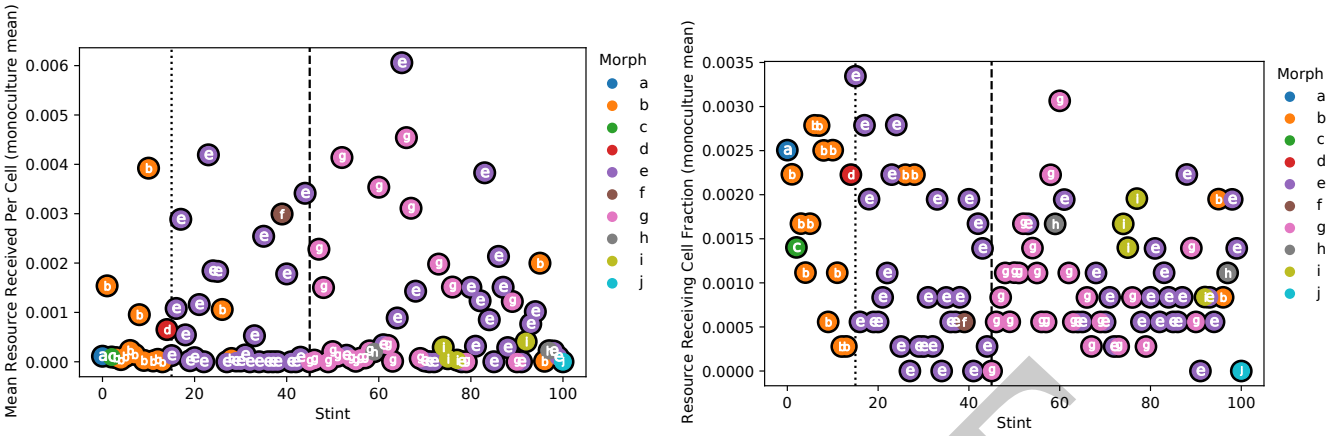


(b) Frequency at which cell proliferation replaces neighbors with only outer kin group commonality to the parent, normalized for the relative frequency of neighbors with kin group commonality to the parent. Values below 1.0 (horizontal bar) indicate preference to displace neighbors with no kin group commonality.



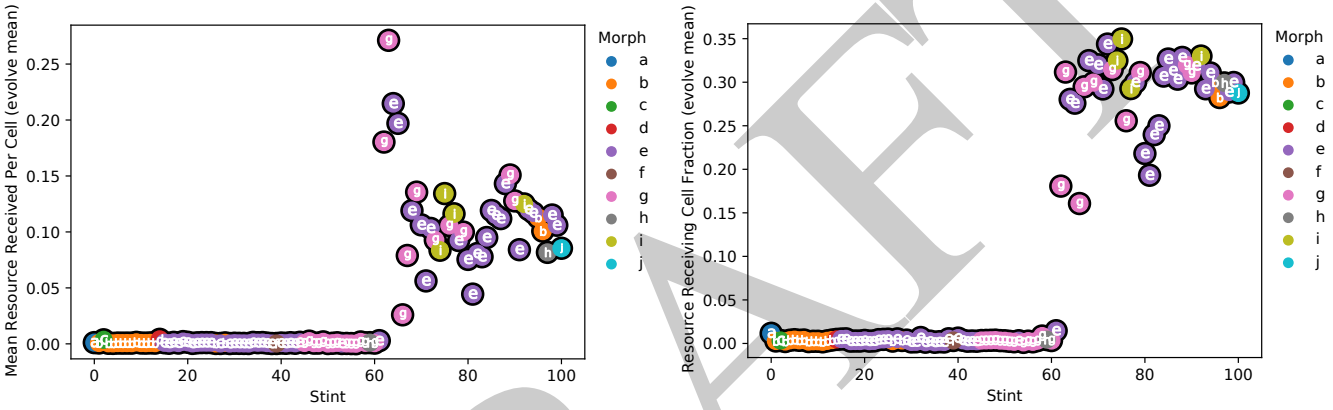
(c) Frequency at which cell proliferation replaces neighbors with full kin group commonality to the parent, normalized for the relative frequency of neighbors with kin group commonality to the parent. Values below 1.0 (horizontal bar) indicate preference to displace neighbors without full kin group commonality.

Figure 18: **Kin conflict rates.** Color coding and letters correspond to qualitative morph codes described in Table 1.



(a) Fraction of cells receiving shared resource at end of evolutionary stints, including not just the focal lineage.

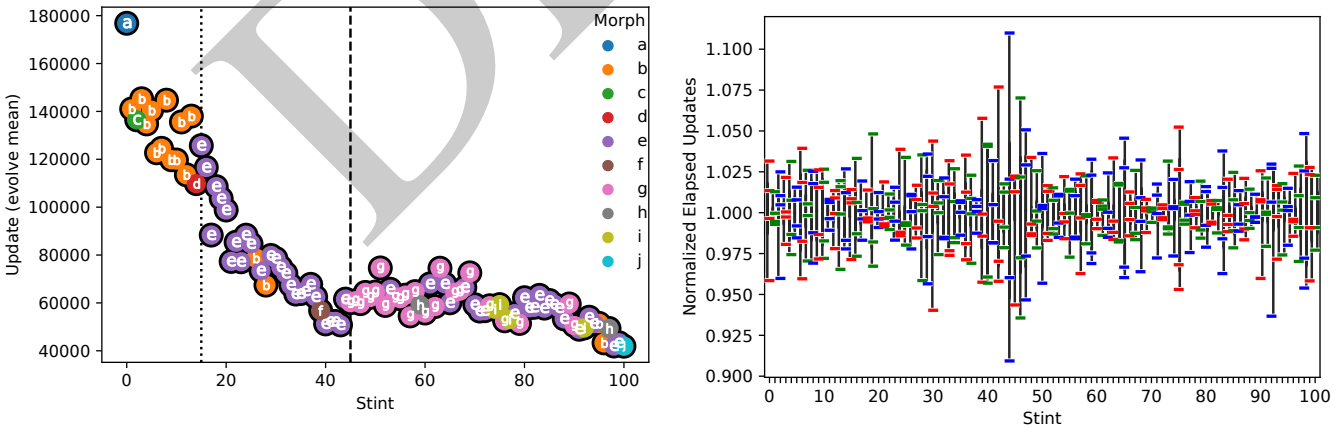
(b) Fraction of cells receiving shared resource in monocultures of focal lineage.



(c) Mean amount of resource shared per cell-update at end of evolutionary stints, including not just the focal lineage.

(d) Mean amount of resource shared per cell-update in monocultures of focal lineage.

Figure 19: **Resource-sharing phenotypic traits.** Color coding and letters correspond to qualitative morph codes described in Table 1. Dotted vertical line denotes emergence of morph e. Dashed vertical line denotes emergence of morph g.



(a) Number of simulation updates elapsed per three-hour evolutionary stint. Color coding and letters correspond to qualitative morph codes described in Table 1.

(b) Distribution of real-time simulation rate of concurrent threads. Colors provided only to distinguish neighboring data.

Figure 20: **Real-time simulation performance.**

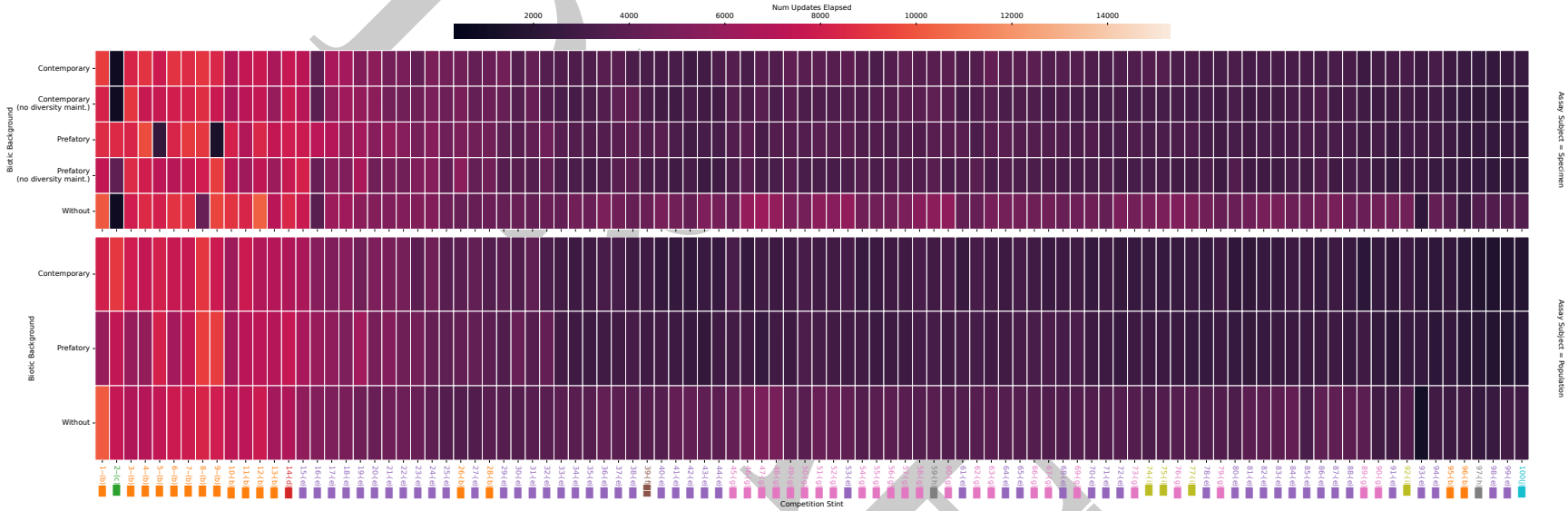


Figure 21: **Number updates elapsed during fixed-duration adaptation assay competitions.** For sampled representative specimen (top) and population-level adaptation (bottom). See Figure 31 for explanation of competition biotic backgrounds. See Supplementary Figure 23 for confidence interval estimates of mean updates elapsed during competition experiments and Supplementary Figure 22 for distributions of updates elapsed during competition experiments.

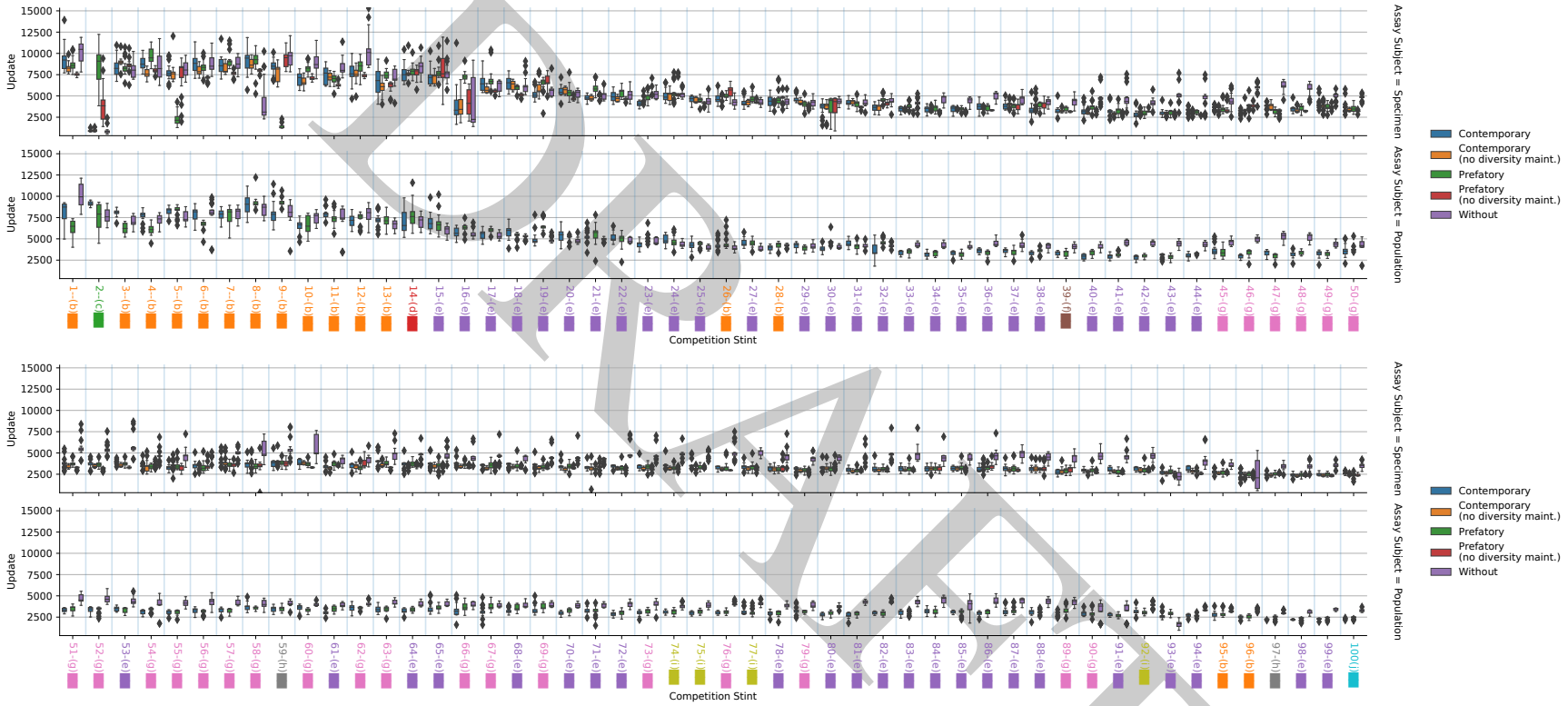


Figure 22: **Number updates elapsed during fixed-duration adaptation assay competitions.** For sampled representative specimen (upper panels) and population-level adaptation (lower panels). Figure is split into two rows due to layout considerations. See Figure 31 for explanation of competition biotic backgrounds.

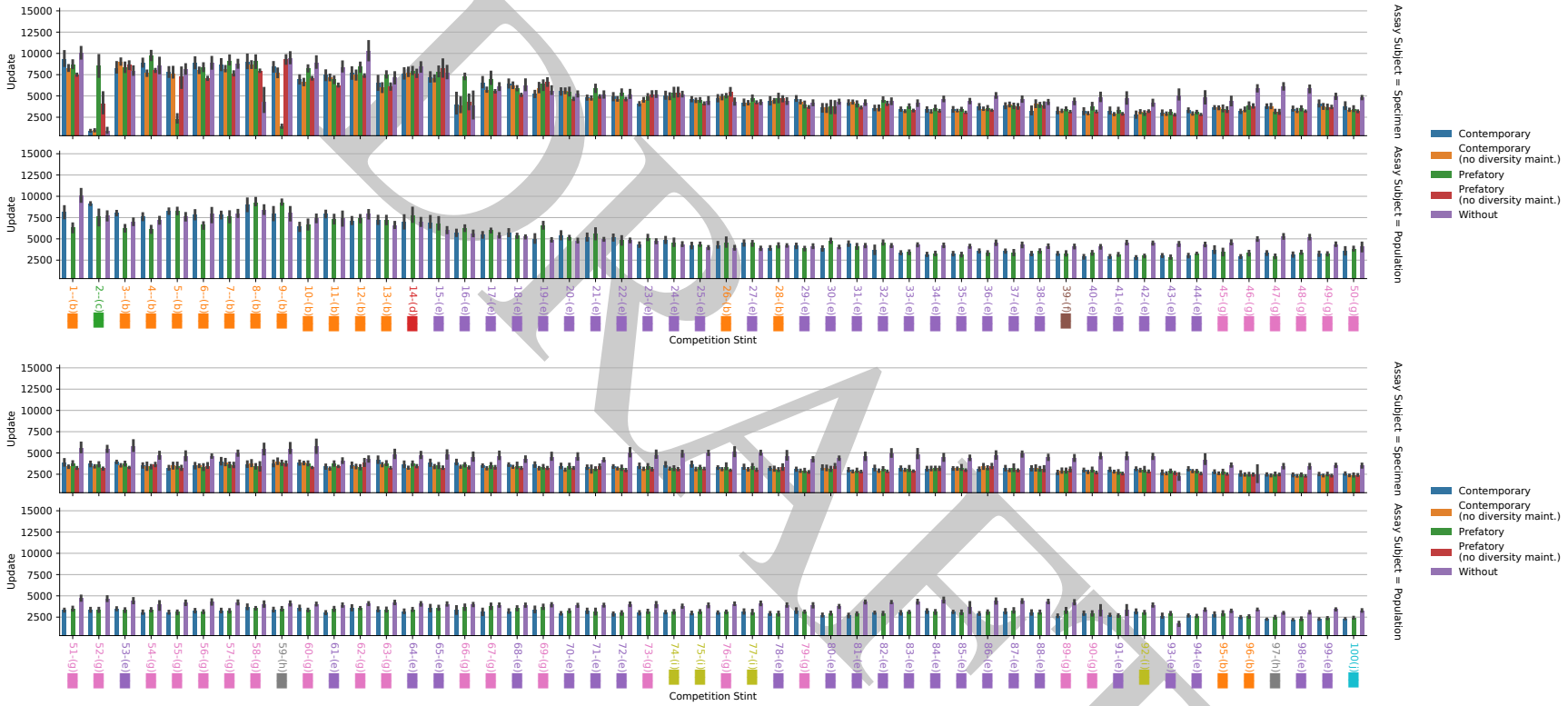


Figure 23: **Number updates elapsed during fixed-duration adaptation assay competitions.** For sampled representative specimen (upper panels) and population-level adaptation (lower panels). Error bars are bootstrapped 95% confidence intervals. Figure is split into two rows due to layout considerations. See Figure 31 for explanation of competition biotic backgrounds.

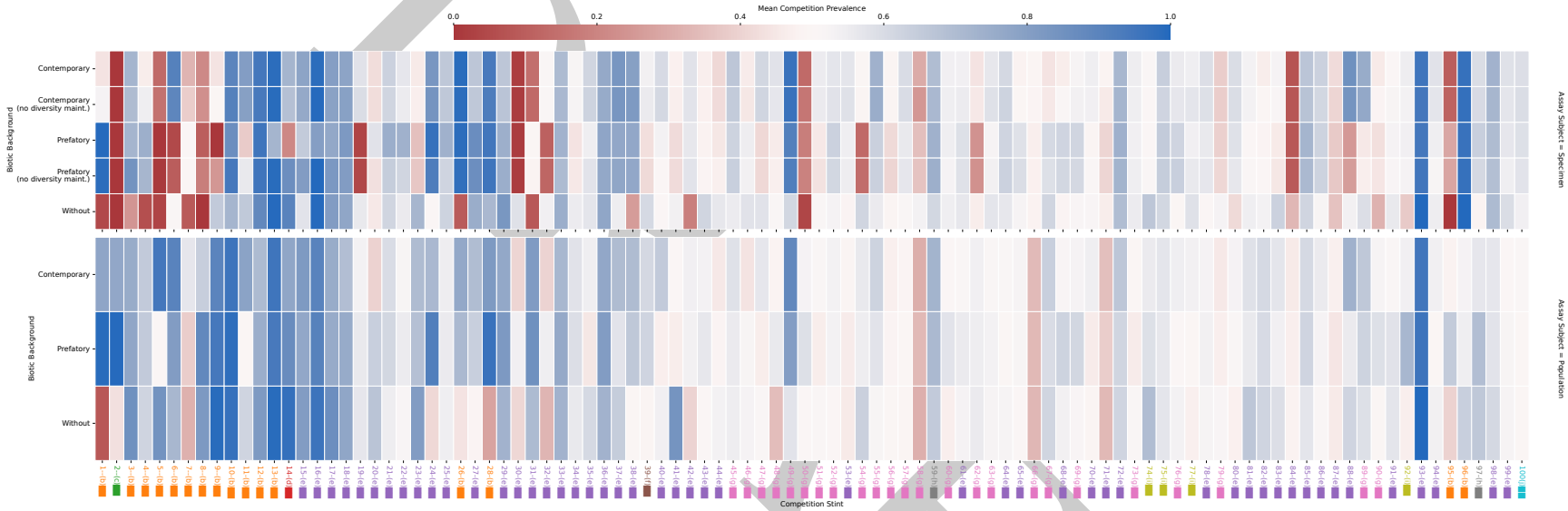


Figure 24: **Mean end-state population composition of competition experiments.** Half (0.5) population composition corresponds to a neutral result, color mapped to white. Blue indicates fitness gain compared to the previous stint and red indicates fitness loss. Color coding and parentheticals of stint labels correspond to qualitative morph codes described in Table 1. Upper panel shows results for sampled focal strain genome, lower panel shows results for entire focal strain population. See Figure 31 for explanation of competition biotic backgrounds. See Figure 25 for boxplot depiction of prevalence outcomes and Figure 26 for bootstrapped confidence intervals on mean prevalence outcomes.

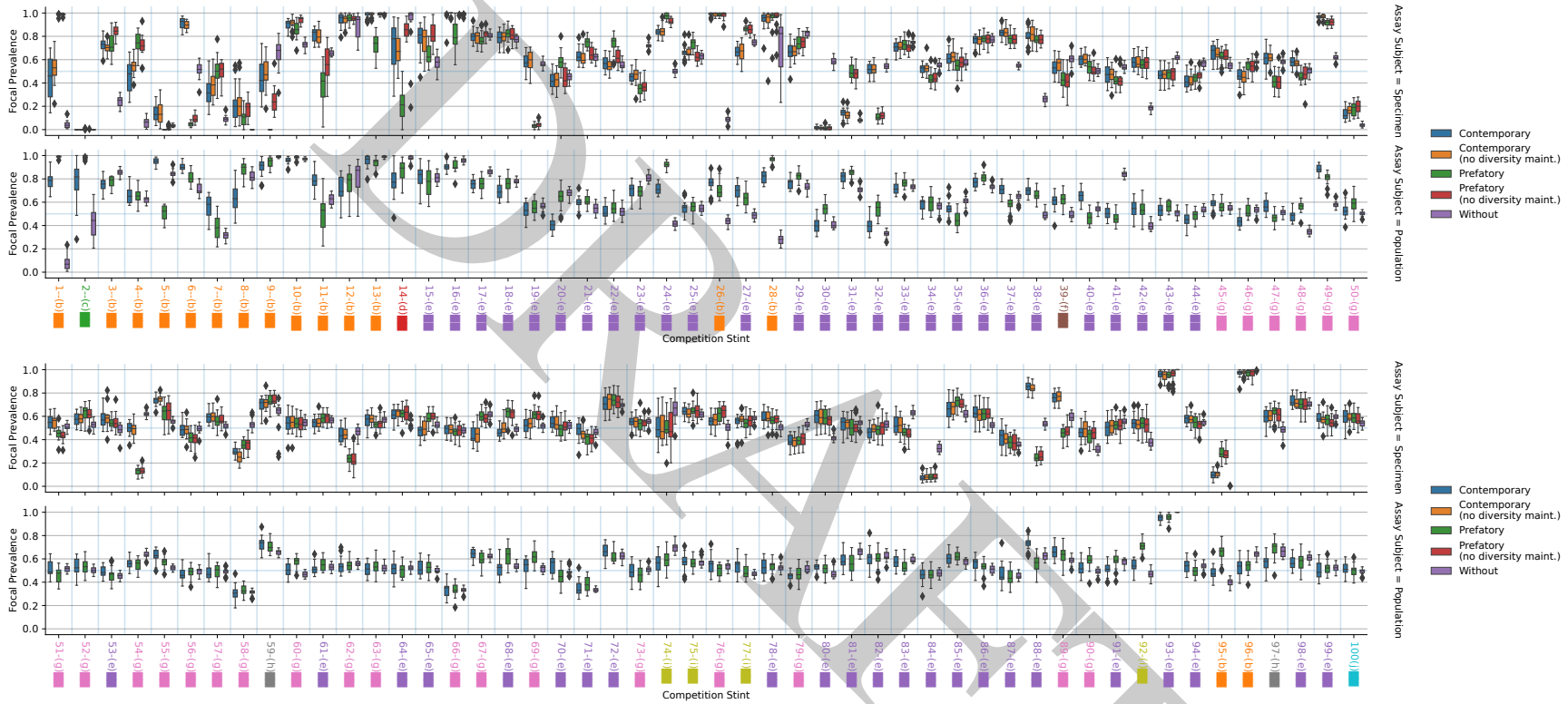


Figure 25: **End-state population composition of competition experiments.** Half (0.5) population composition corresponds to a neutral result. Zero population composition corresponds to extreme fitness loss compared to the previous stint. Population composition of 1.0 corresponds to extreme fitness gain compared to the previous stint. Color coding and parentheticals of stint labels correspond to qualitative morph codes described in Table 1. Upper panels show results for sampled focal strain genome, lower panels show results for entire focal strain population. Figure is split into two rows due to layout considerations. See Figure 31 for explanation of competition biotic backgrounds. See Figure 26 for bootstrapped confidence intervals on mean prevalence outcomes.

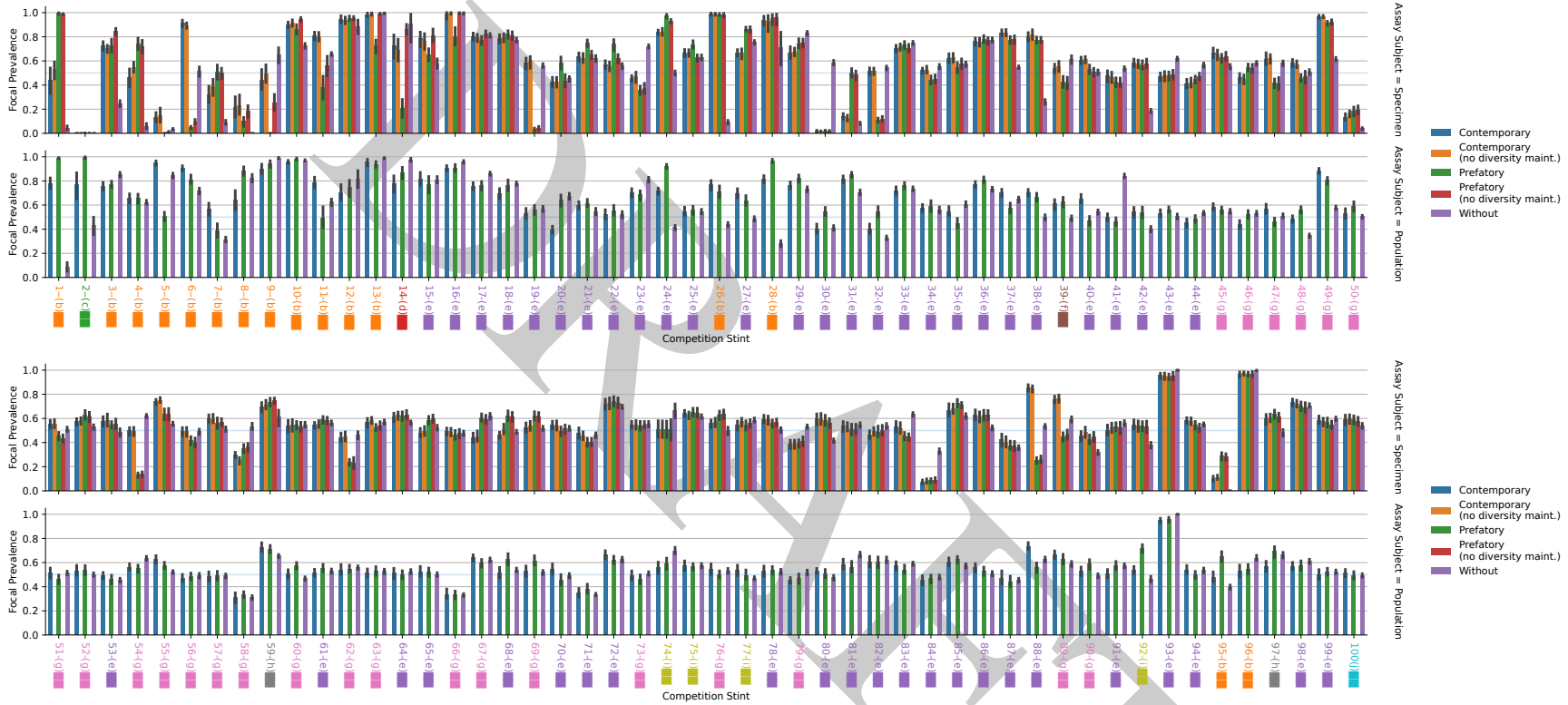
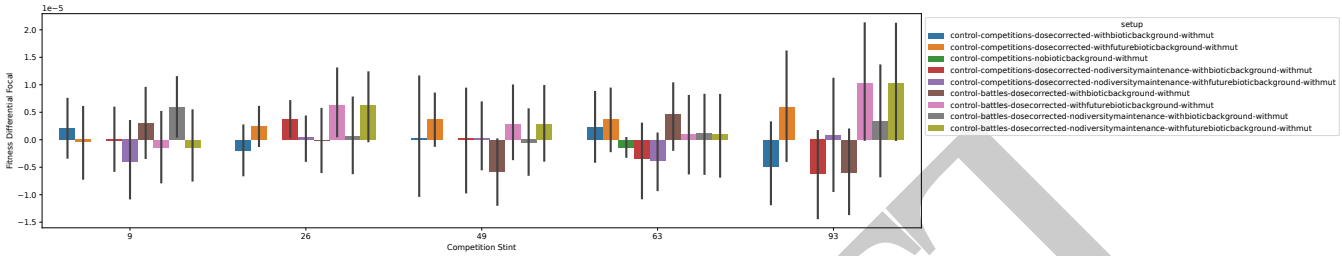
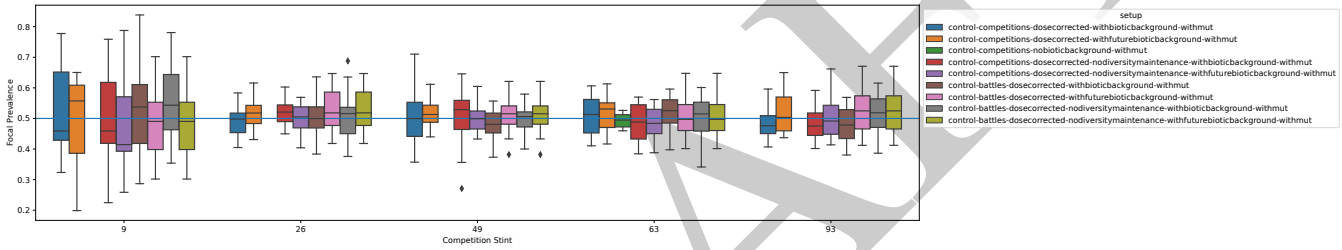


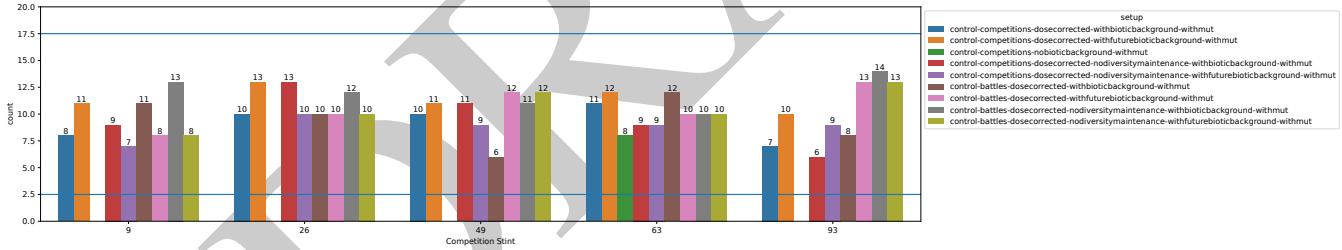
Figure 26: **End-state population composition of competition experiments.** Half (0.5) population composition corresponds to a neutral result. Zero population composition corresponds to extreme fitness loss compared to the previous stint. Population composition of 1.0 corresponds to extreme fitness gain compared to the previous stint. Error bars are bootstrapped 95% confidence intervals. Color coding and parentheticals of stint labels correspond to qualitative morph codes described in Table 1. Upper panels show results for sampled focal strain genome, lower panels show results for entire focal strain population. Figure is split into two rows due to layout considerations. See Figure 31 for explanation of competition biotic backgrounds. See Figure 25 for boxplot depiction of prevalence outcomes.



(a) Calculated fitness differential between competing strains based on population composition at the end of competition experiments. Zero is neutral. Error bars are 95% confidence intervals.



(b) Fractional composition of focal population at the end of competition experiments. A neutral outcome corresponds to even (0.5) composition, annotated with a horizontal line.



(c) Number competitions out of 20 won by first strain. Ten competitions won corresponds to a perfectly neutral outcome. Eighteen and more or two or less competitions won were considered to indicate a significant fitness difference between strains. These thresholds for significance annotated with horizontal lines.

Figure 27: **Control adaptation experiments for selected stints.** Control experiments were performed by competing two identical genomes or populations against each other with the contemporary biotic background, with the prefatory biotic background, or with no biotic background. See Figure 31 for summary of adaptation experiment design.

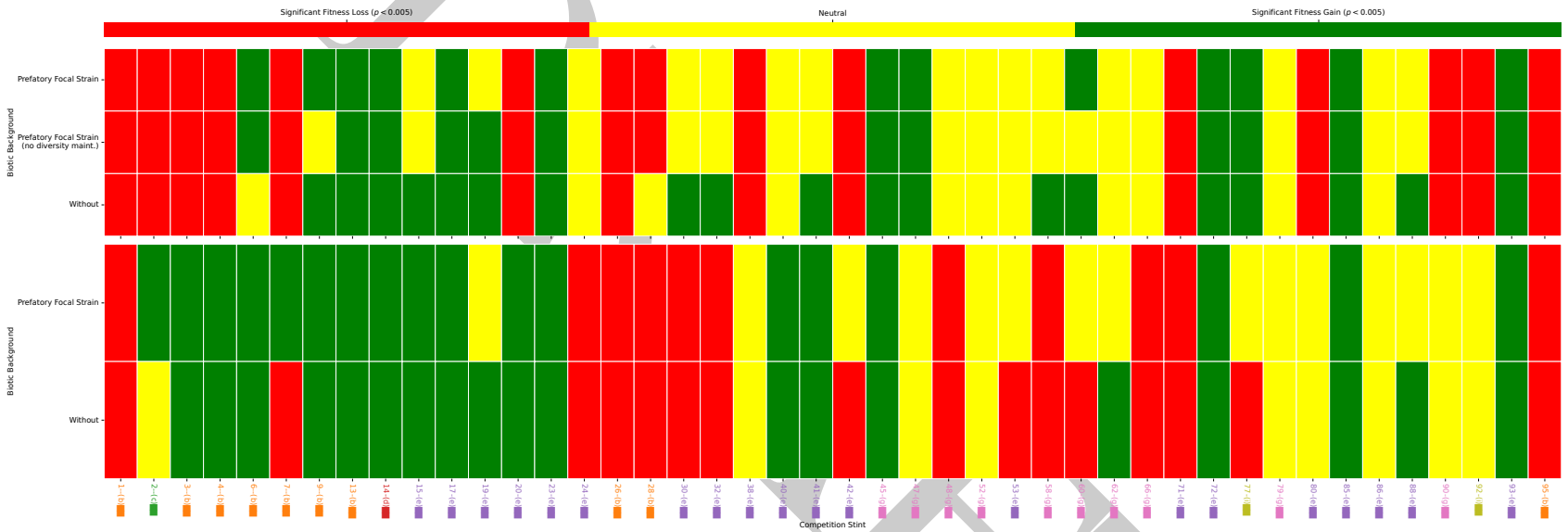
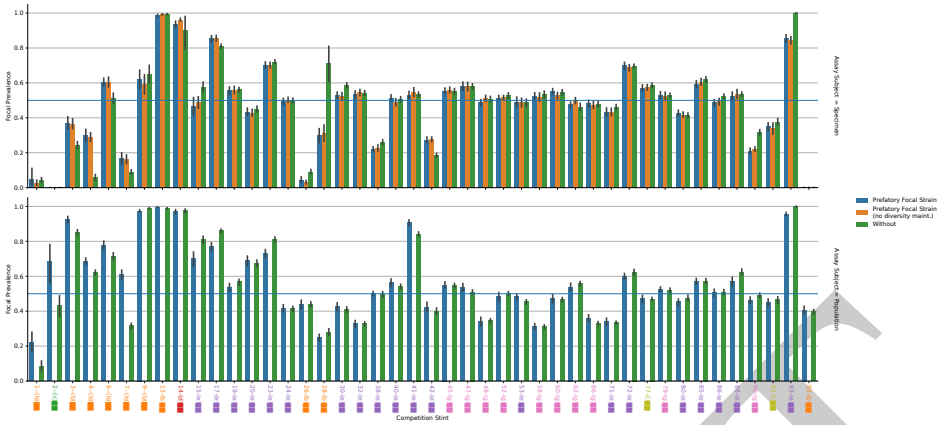
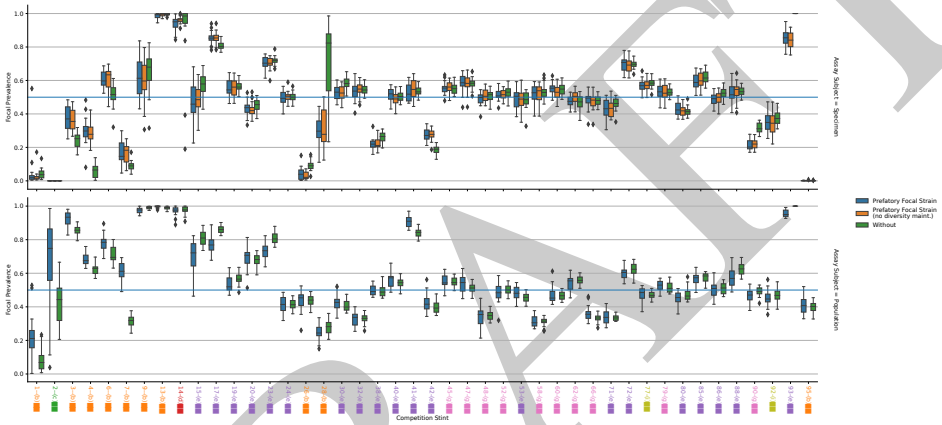


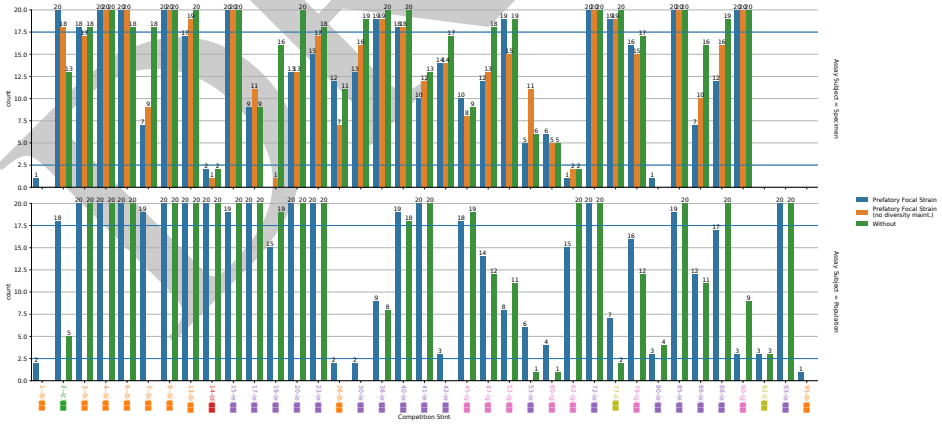
Figure 28: **Adaptation assay outcomes.** Summary of biotic background control adaptation assay outcomes for sampled representative specimen (top) and population-level adaptation (bottom). Color coding and parentheticals of stint labels correspond to qualitative morph codes described in Table 1. See Figure 31 for explanation of competition biotic backgrounds.



(a) Fractional composition of focal population at the end of competition experiments. Zero is neutral. Error bars are 95% confidence intervals.



(b) Fractional composition of focal population at the end of competition experiments. A neutral outcome corresponds to even (0.5) composition, annotated with a horizontal line.



(c) Number competitions out of 20 won by first strain. Ten competitions won corresponds to a perfectly neutral outcome. Eighteen and more or two or less competitions won were considered to indicate a significant fitness difference between strains. These thresholds for significance annotated with horizontal lines.

Figure 29: **Biotic background control adaptation experiments for selected stints.** Biotic background control experiments were performed by substituting the baseline competitor for the biotic background. See Figure 31 for summary of adaptation experiment design.

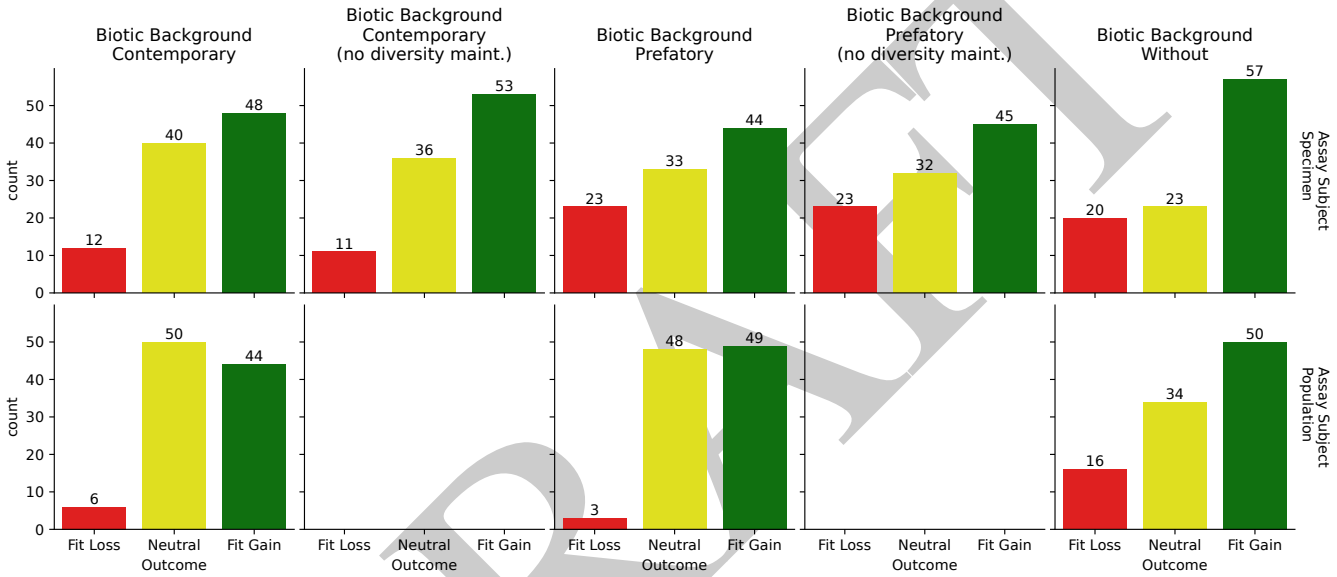


Figure 30: **Distribution of adaptation assay outcomes.** For each adaptation assay, three outcomes were possible: significant fitness gain, significant fitness loss, or no significant fitness change ("neutral"). Significance cutoff $p < 0.005$ was used. A fitness loss — color-coded red — corresponds to winning 2 or fewer competitions out of 20 against the preceding stint's focal strain population. A fitness gain — color-coded green — corresponds to winning 18 or more competitions out of 20 against the preceding stint's focal strain population. Neutral fitness outcomes are color-coded yellow. Outcome counts are accumulated over experiments from stint 1 through stint 100. Upper row shows results for sampled focal strain genome, lower row shows results for entire focal strain population. See Figure 31 for explanation of competition biotic backgrounds. See Figure 32 for joint distributions of fitness outcomes across biotic backgrounds.

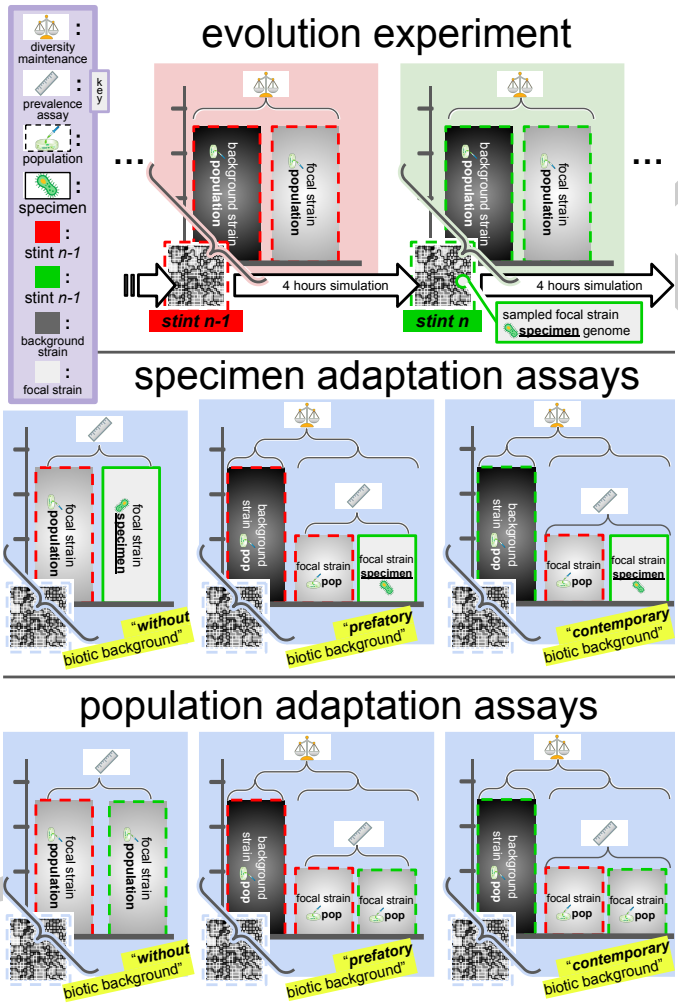
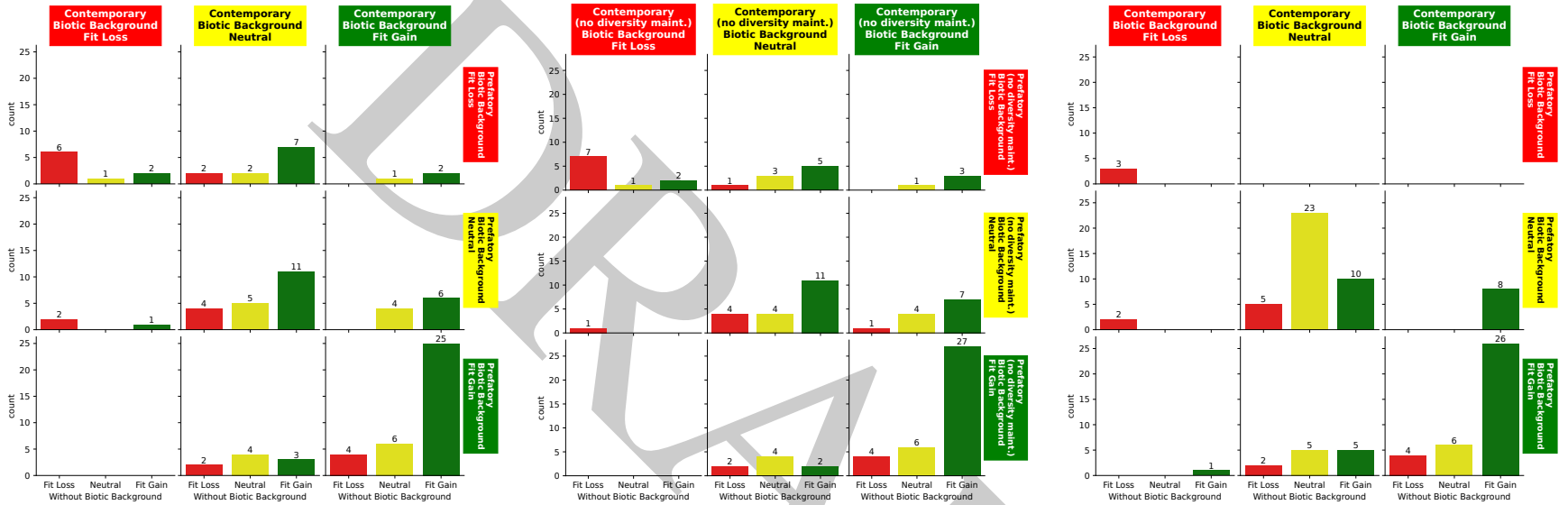


Figure 31: **Adaptation assay design.** Top panel shows progress of original evolutionary experiment over one stint. A diversity maintenance procedure was used to ensure long-term coexistence of at least two strains over the course of the experiment by penalizing any strain that occupied more than half of thread-local population space. A "focal strain" was arbitrarily chosen for study; we refer to the other strain as the "background strain." Adaptation assays in lower panels measure fitness change over the course of that stint through competition against the population from the preceding stint. The middle panel shows measurement of adaptation of the representative specimen that was sampled for analysis at each stint. The bottom panel shows measurement of the adaptation of the entire focal strain population at each stint. Competitors were mixed in even proportion into the environment. Bar heights represent initial relative proportions of assay participants at the beginning of the competition. Adaptation was measured by measuring change in population composition over a 10 minute competition window. We call this measurement of population composition change a "prevalence assay". Competition experiments were performed absent the background strain, with the background strain population from the preceding stint, or with the background strain population from the current stint — shown separately in each panel.



(a) Joint distribution of adaptation assay on representative specimen from focal strain over biotic backgrounds, with diversity maintenance during competition.

(b) Joint distribution of adaptation assay on representative specimen from focal strain over biotic backgrounds, with diversity maintenance disabled during competition.

(c) Joint distribution of adaptation assay on focal strain population over biotic backgrounds, with diversity maintenance during competition.

Figure 32: **Effect of biotic background on adaptation assay outcomes.** Joint distribution of adaptation assay outcomes across biotic backgrounds. For each adaptation assay, three outcomes were possible: significant fitness gain, significant fitness loss, or no significant fitness change ("neutral"). Significance cutoff $p < 0.005$ was used. A fitness loss — color-coded red — corresponds to winning 2 or fewer competitions out of 20 against the preceding stint's focal strain population. A fitness gain — color-coded green — corresponds to winning 18 or more competitions out of 20 against the preceding stint's focal strain population. Neutral fitness outcomes are color-coded yellow. Outcome counts are accumulated over experiments from stint 1 through stint 100. Counts in each subfigure therefore sum to 100. Column position in facet grid indicates outcome with contemporary biotic background, row position indicates outcome with prefatory biotic background, and bar color and x position indicates outcome without biotic background. See Figure 31 for explanation of competition biotic backgrounds. See Figure 34 for detail on joint distribution of outcomes with and without diversity maintenance, which were mostly identical.

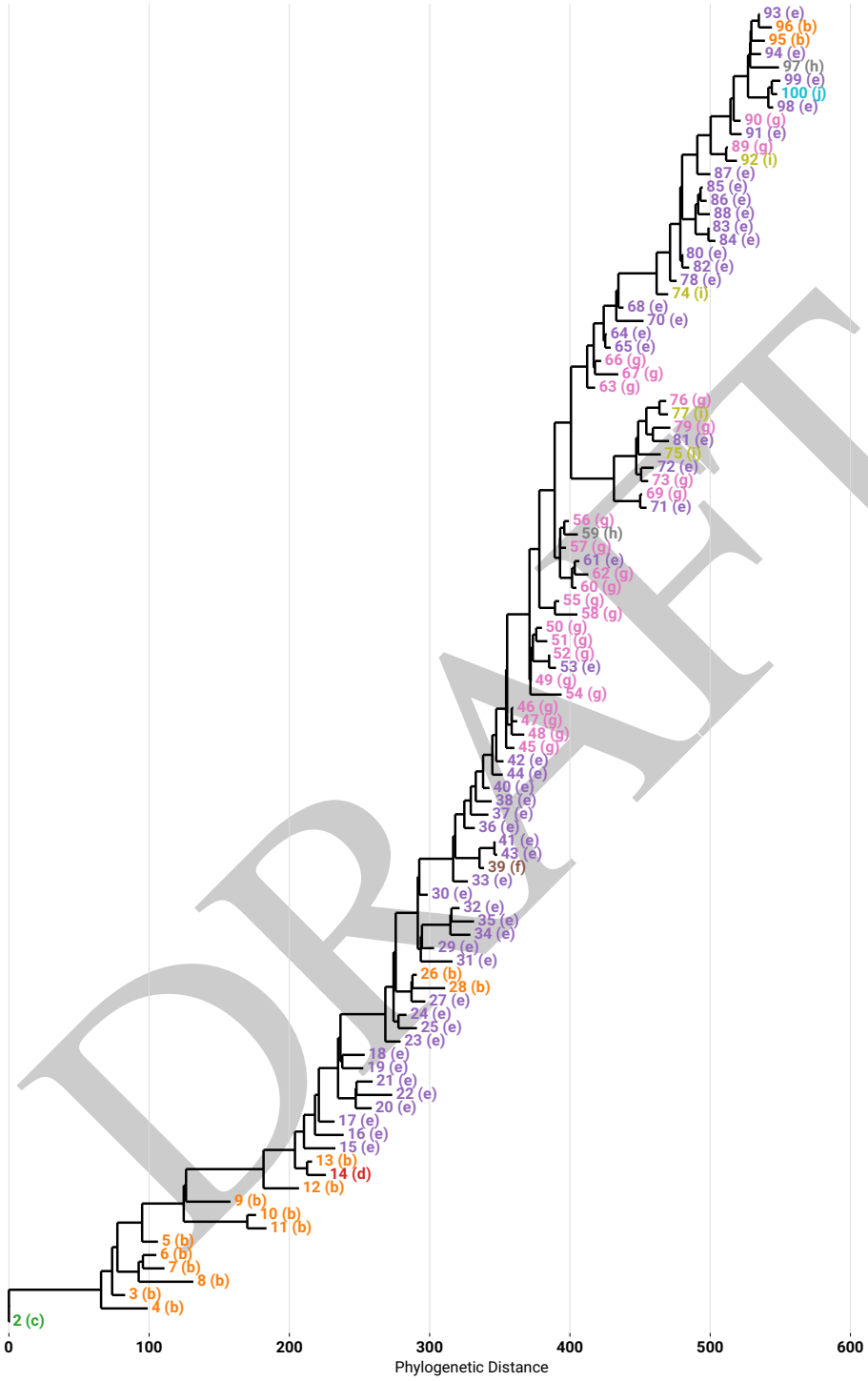
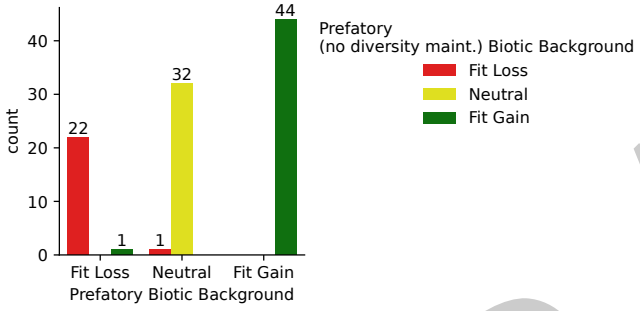
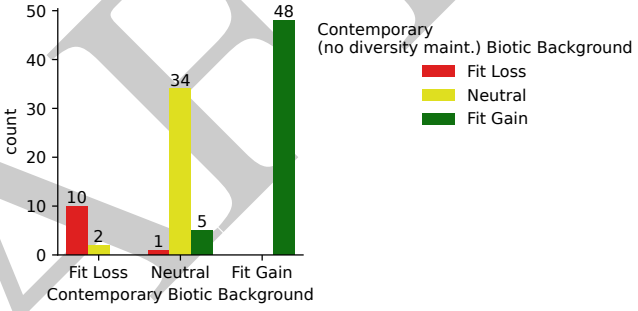


Figure 33: **Estimated phylogeny of sampled focal strain representatives.** Phylogeny constructed using parsimony algorithm based on genome tag bitstring content [100]. Each leaf node corresponds to a sampled representative. Representatives from stints 0 and 1, which share no common ancestry with representatives from other stints, are excluded. Numbers refer to stint that each representative was sampled from. Color coding and parentheticals of stint labels correspond to qualitative morph codes described in Table 1.



(a) prefatory biotic background outcomes with and without diversity maintenance



(b) contemporary biotic background outcomes with and without diversity maintenance

Figure 34: **Effect of diversity maintenance on adaptation assays with biotic background.** Joint distribution of competition experiments performed under biotic background conditions with diversity maintenance enabled and disabled. Color coding denotes outcome without diversity maintenance and x position denotes outcome with diversity maintenance. Note that both plots above show distributions for adaptation assays on representative specimens. Competition experiments without diversity maintenance were not performed for population-level adaptation. See Figure 31 for explanation of competition biotic backgrounds.

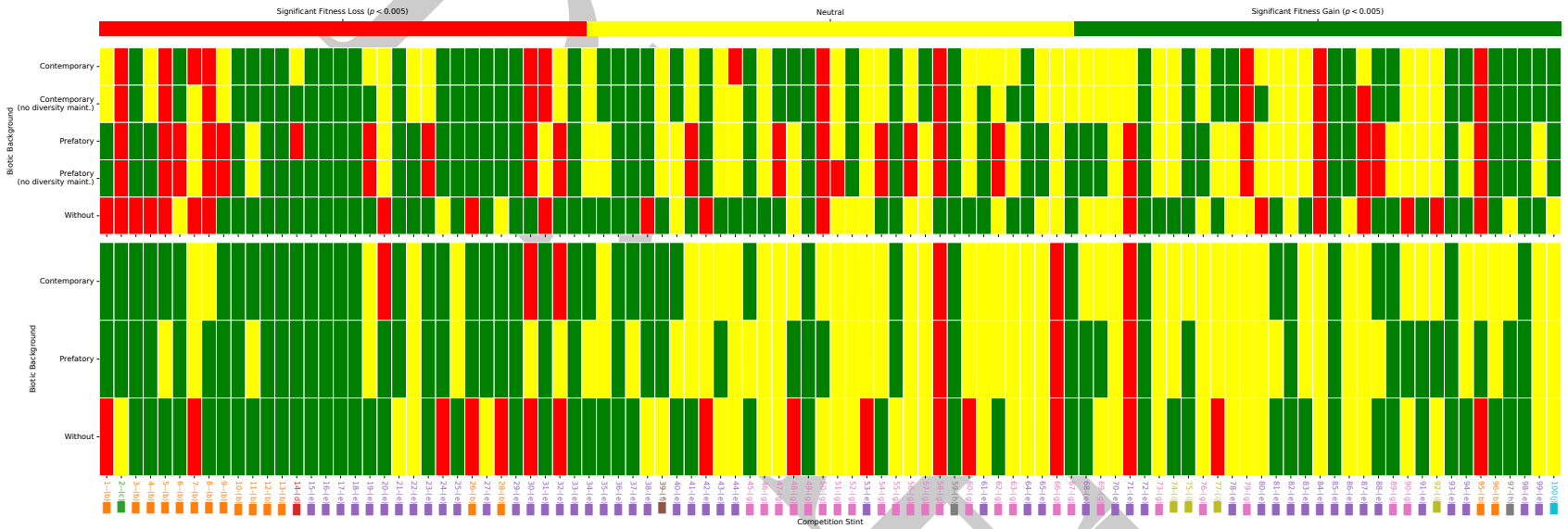


Figure 35: **Adaptation assay outcomes.** Summary of adaptation assay outcomes for sampled representative specimen (top) and population-level adaptation (bottom). Color coding and parentheticals of stint labels correspond to qualitative morph codes described in Table 1. See Figure 31 for explanation of competition biotic backgrounds.

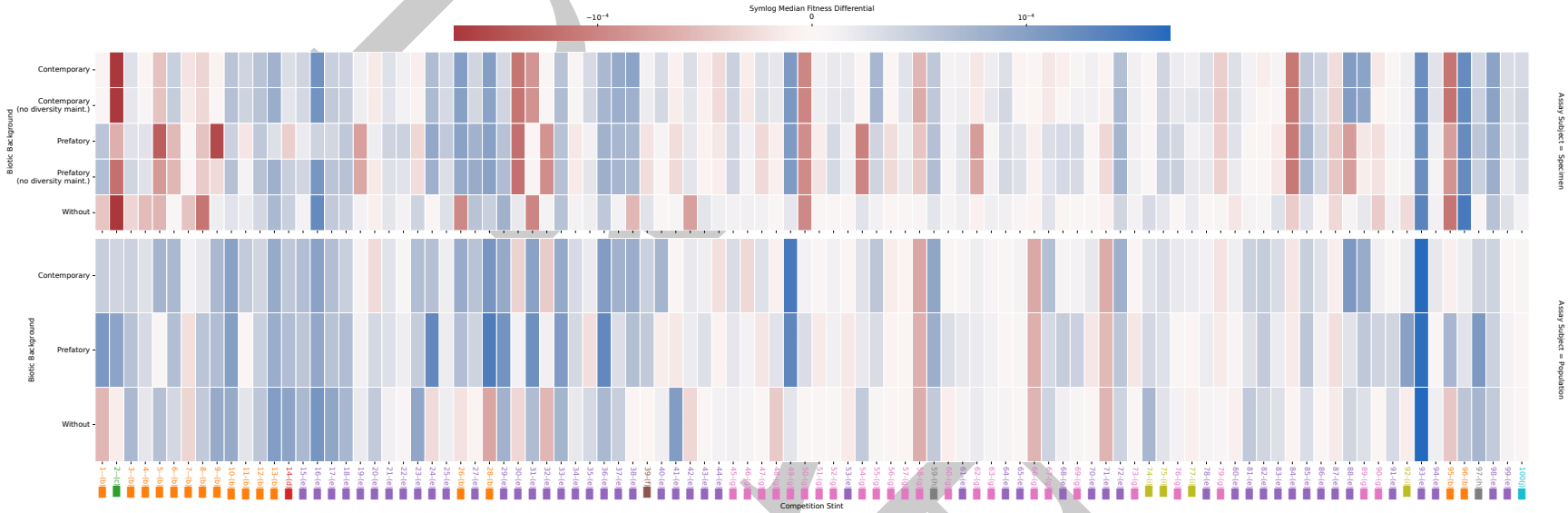


Figure 36: **Adaptation assay effect strengths.** Median calculated fitness differential outcomes of competition experiments. Zero fitness differential corresponds to a neutral result, color mapped to white. Blue indicates positive fitness differential (fitness gain) compared to the previous stint and red indicates negative fitness differential (fitness loss). Color coding and parentheticals of stint labels correspond to qualitative morph codes described in Table 1. Note that color intensity is plotted on a symlog scale due to distribution of fitness differentials over multiple orders of magnitude. Upper panels shows results for sampled focal strain genome, lower panel shows results for entire focal strain population. See Figure 31 for explanation of competition biotic backgrounds.

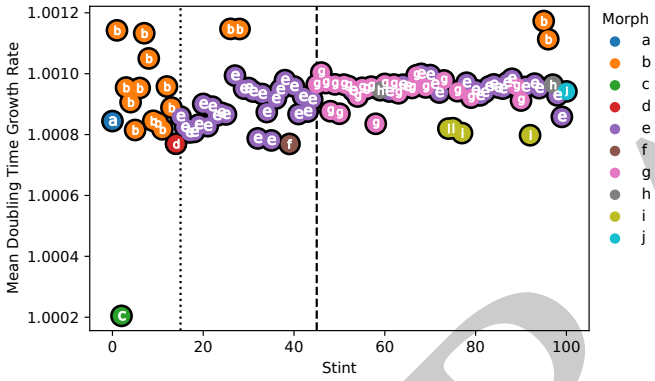
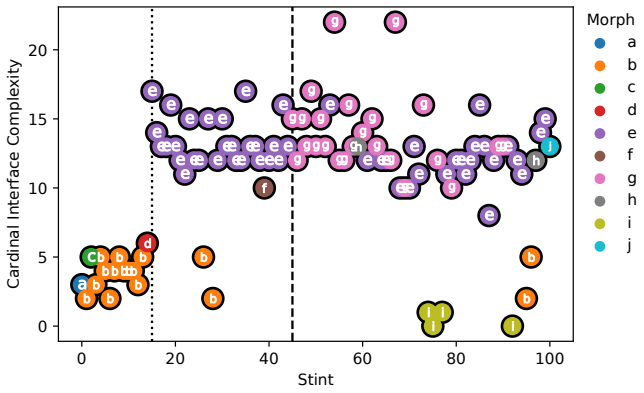
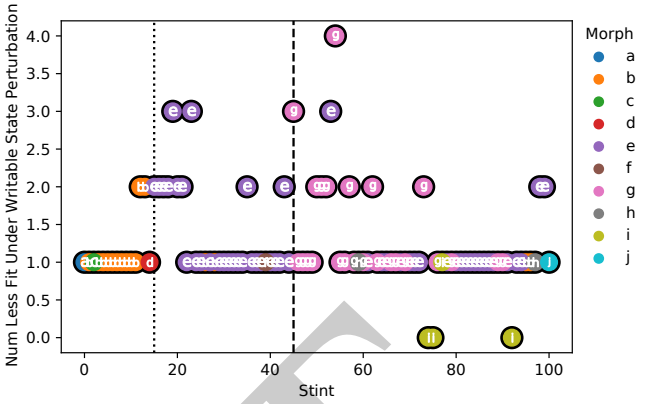


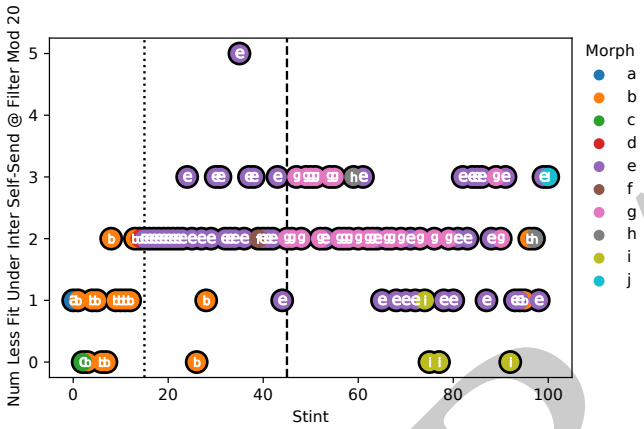
Figure 37: **Growth rate estimates.** Estimates made from doubling time experiments, measuring time for a monoculture to grow from 0.25 maximum population size to 0.5 maximum population size.



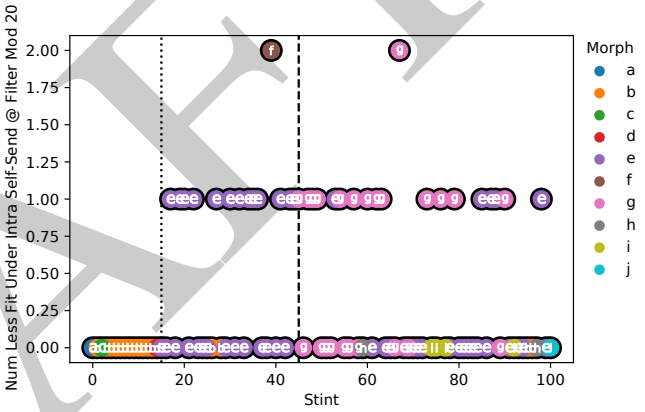
(a) Cardinal processor interface complexity, the total number of distinct interactions between a virtual CPU controlling cell behavior and its surroundings that contribute to fitness. (Sum of Figures 38b to 38f.)



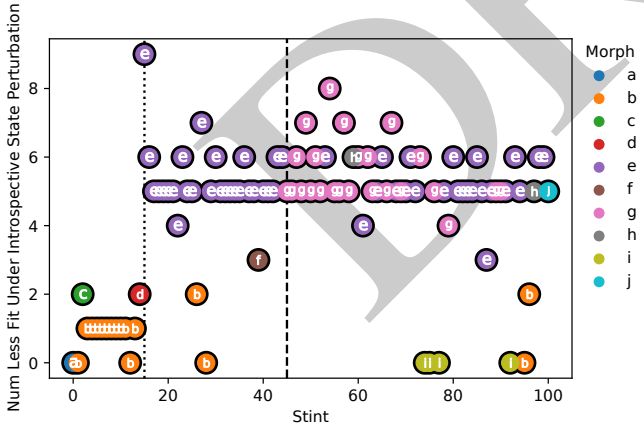
(b) Writable state interface complexity, the number of output states that contribute to fitness. See Supplementary Figure 12 for detail on the writable states that contribute to fitness.



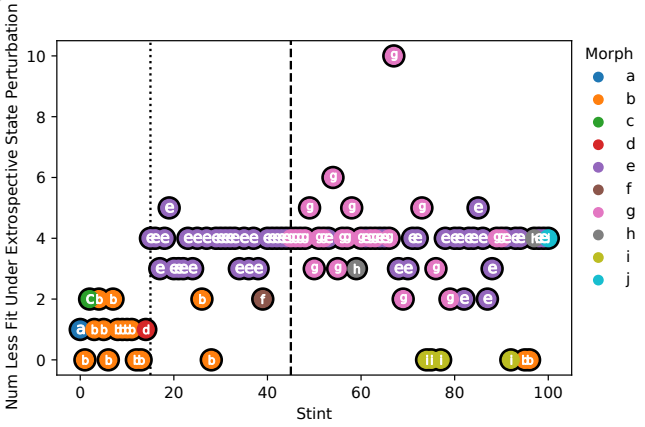
(c) Intermessage interface complexity, the number of distinct inter-cell messages that contribute to fitness.



(d) Intramessage interface complexity, the number of distinct intra-cell messages that contribute to fitness.

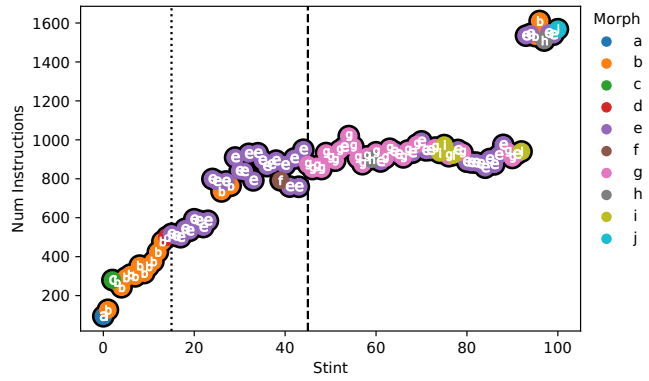


(e) Introspective interface complexity, the number of states viewed within the cell itself that contribute to fitness. See Supplementary Figure 11 for detail on the introspective states that contribute to fitness.

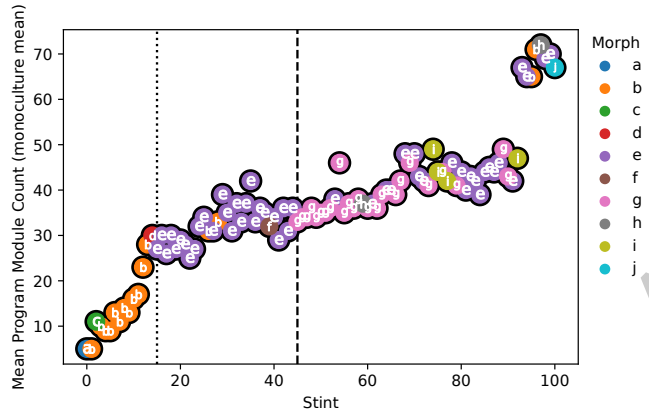


(f) Extrospective interface complexity, the number of states viewed in neighboring cells that contribute to fitness. See Supplementary Figure 10 for detail on the extrospective states that contribute to fitness.

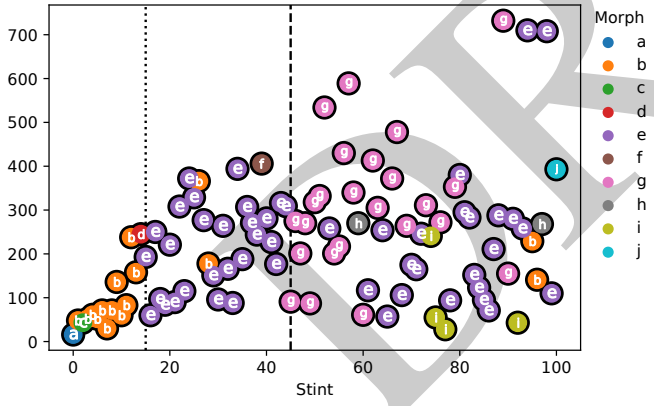
Figure 38: **Interface complexity estimates.** Color coding and letters correspond to qualitative morph codes described in Table 1. Dotted vertical line denotes emergence of morph e. Dashed vertical line denotes emergence of morph g.



(a) instruction count



(b) module count



(c) active site count complexity

Figure 39: **Genome content measures.** Instruction count is the total number of instructions present in the genome. Module count is the number of tagged linear GP modules available for activation by signals from the environment, from other agents, or from within an agent. Active site count is the number of genome sites that contribute to phenotype, measured as number sites remaining after phenotype-neutral nopout (Section A.3). This measure gives a sense of the number of "active" instructions that influence agents' behavior. Color coding and letters correspond to qualitative morph codes described in Table 1. Dotted vertical line denotes emergence of morph e. Dashed vertical line denotes emergence of morph g.

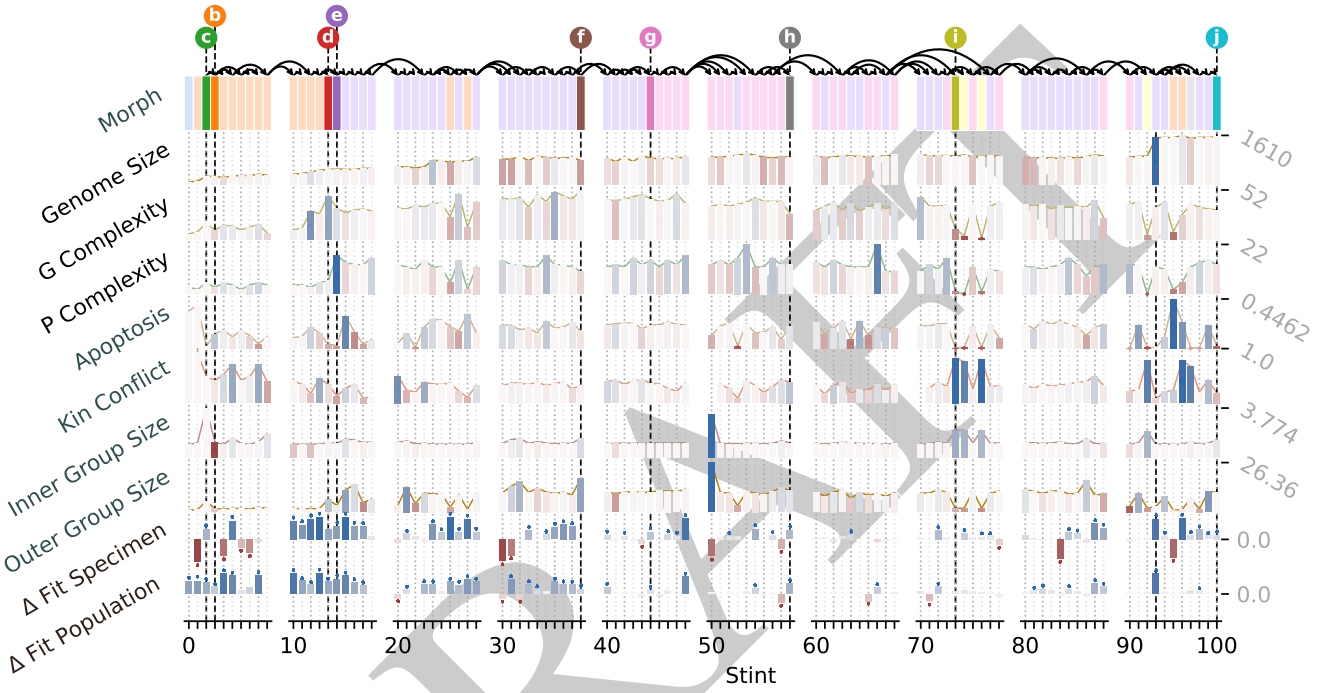


Figure 40: **Overview of complexity, novelty, and adaptation (contemporary biotic background) across evolutionary history.** Columns correspond to 100 intermediate genome specimens sampled at sequential checkpoints across evolutionary history ("stints"), arranged left to right. Top row, "Morph," reports qualitative categorization of multicell structure for sampled genome specimens. Arrows indicate estimated ancestry among genome specimens. Middle rows trace evolution of individual traits across sampled genome specimens. G complexity is genotype complexity — the number of genome sites with detectable fitness effects in knockout experiments. P complexity is phenotype complexity — the number of distinct cell input/output interfaces that cause fitness loss when disrupted. Group size, apoptosis, and kin conflict were measured in a monoculture trial. Bar color-coding indicates magnitude of change relative to ancestor (dark blue as strong positive; dark red as strong negative). Bottom two rows, "Δ Fit Specimen" and "Δ Fit Population" report fitness competitions between current checkpoint and preceding checkpoint. In specimen assays, sampled genome from current checkpoint was competed against a whole-population sample from the preceding checkpoint. In population assays, whole-population sample from current checkpoint was competed against whole-population sample from the preceding checkpoint. Positive results indicate fitness gain; negative results indicate fitness loss; asterisks indicate significance ($p < 0.005$). Fitness assays with contemporary biotic background shown; Supplementary Figure 41 shows corresponding fitness results without biotic background.

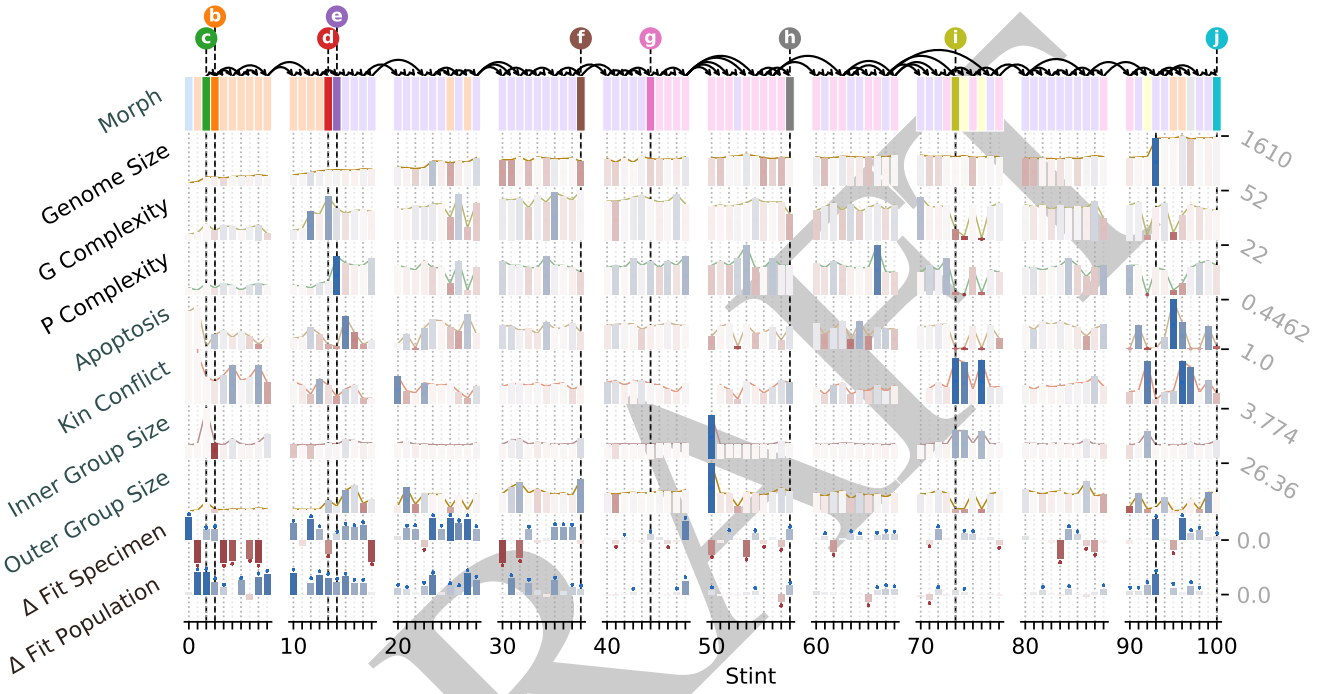


Figure 41: **Overview of complexity, novelty, and adaptation across evolutionary history of case study strain.** Columns correspond to 100 intermediate genome specimens sampled at sequential checkpoints across evolutionary history ("stints"), arranged left to right. Top row, "Morph," reports qualitative categorization of multicell structure for sampled genome specimens. Arrows indicate estimated ancestry among genome specimens. Middle rows trace evolution of individual traits across sampled genome specimens. G complexity is genotype complexity — the number of genome sites with detectable fitness effects in knockout experiments. P complexity is phenotype complexity — the number of distinct cell input/output interfaces that cause fitness loss when disrupted. Group size, apoptosis, and kin conflict were measured in a monoculture trial. Bar color-coding indicates magnitude of change relative to ancestor (dark blue as strong positive; dark red as strong negative). Bottom two rows, "Δ Fit Specimen" and "Δ Fit Population" report fitness competitions between current checkpoint and preceding checkpoint. In specimen assays, sampled genome from current checkpoint was competed against a whole-population sample from the preceding checkpoint. In population assays, whole-population sample from current checkpoint was competed against whole-population sample from the preceding checkpoint. Positive results indicate fitness gain; negative results indicate fitness loss; asterisks indicate significance ($p < 0.005$). Fitness assays with prefatory biotic background shown; Supplementary Figure 40 shows corresponding fitness results with contemporary biotic background.

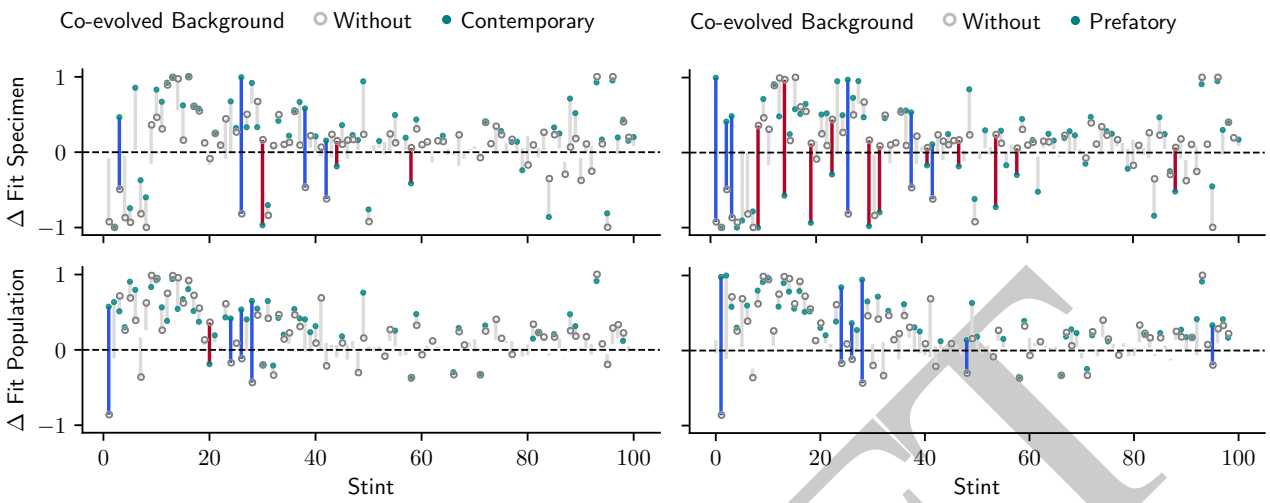


Figure 42: **Fitness trade-offs driven by ecological context.** Plots report influence on focal strain competition assays attributable to presence versus absence of background strain. Experiments competed focal strain samples (specimen or population from stint $n+1$) against baseline focal strain population from stint n. Outcomes recorded between 1 (fitness gain; stint n baseline driven extinct) and -1 (fitness loss; stint $n+1$ driven extinct), with near-zero values indicating comparable fitness. Vertical line segments mark discrepancy between abiotic conditions ("without," hollow markers) and biotic conditions (filled markers). Markers omitted where no significant fitness change detected between stint $n+1$ and stint n baseline ($\alpha = 0.005$). Segment color highlights sign-change effect: harmful trait becomes beneficial under ecological context (blue) or vice versa (red). Subplots differ in focal strain sample type (rows) and source of ecological context (columns). Under prefatory context, background strain from stint n; under contemporary context, stint $n+1$. To account for possible confounding effects from resource-balancing diversity maintenance mechanism used with background strain, specimen assays were replicated with this mechanism disabled (Supplementary Figure 45); results were generally similar (Supplementary Figure 43).

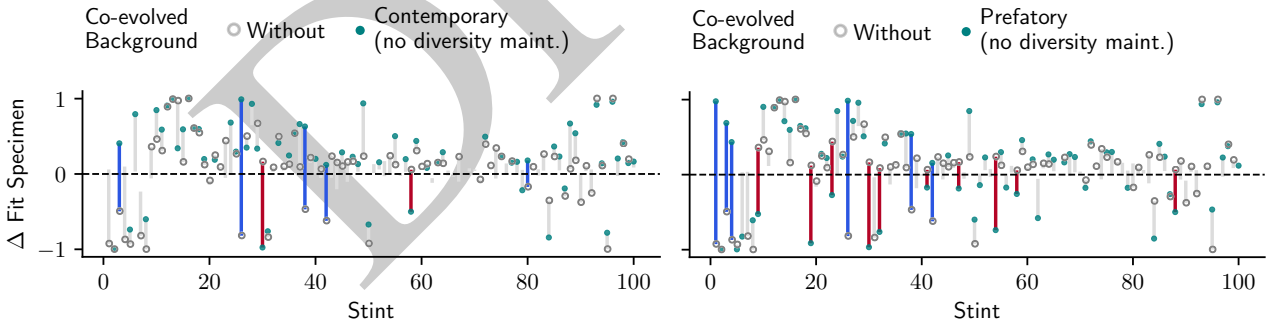


Figure 43: **Fitness trade-offs driven by ecological context, no diversity maintenance control.** Plots report influence on focal strain competition assays attributable to presence versus absence of background strain. Experiments competed focal strain samples (specimen from stint $n+1$) against baseline focal strain population from stint n. Outcomes recorded between 1 (fitness gain; stint n baseline driven extinct) and -1 (fitness loss; stint $n+1$ driven extinct), with near-zero values indicating comparable fitness. Vertical line segments mark discrepancy between abiotic conditions ("without," hollow markers) and biotic conditions (filled markers). Markers omitted where no significant fitness change detected between stint $n+1$ and stint n baseline ($\alpha = 0.005$). Segment color highlights sign-change effect: harmful trait becomes beneficial under ecological context (blue) or vice versa (red). Subplots differ in source of ecological context (columns). Under prefatory context, background strain from stint n; under contemporary context, stint $n+1$.

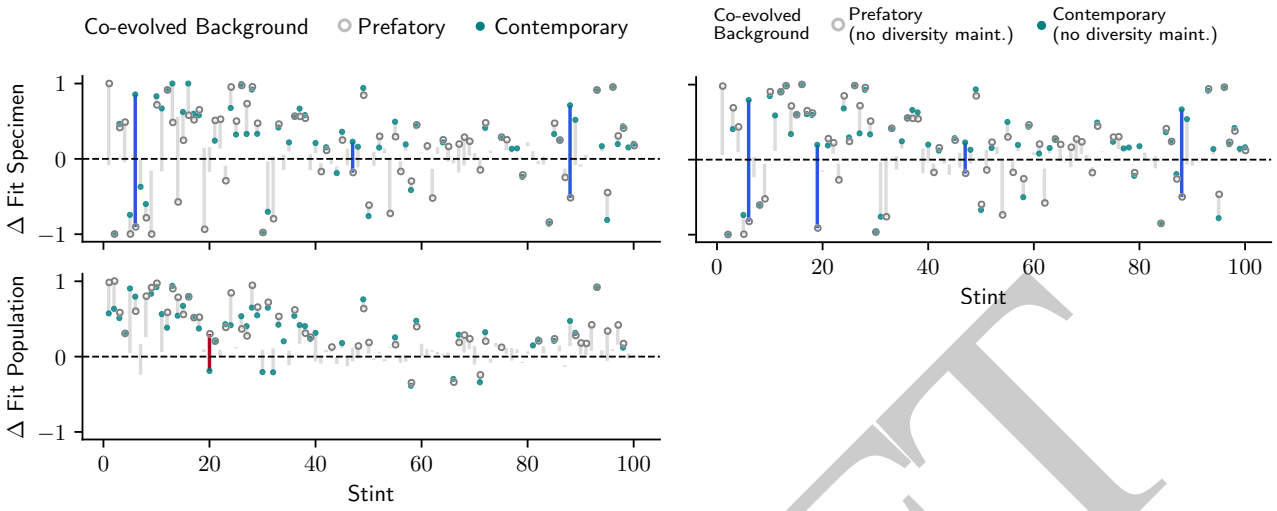


Figure 44: **Fitness trade-offs between prefatory and contemporary ecological context.** Plots report influence on focal strain competition assays attributable to source of background strain used for ecological context. Experiments competed focal strain samples (specimen or population from stint $n+1$) against baseline focal strain population from stint n. Outcomes recorded between 1 (fitness gain; stint n baseline driven extinct) and -1 (fitness loss; stint $n+1$ driven extinct), with near-zero values indicating comparable fitness. Under prefatory context, background strain from stint n; under contemporary context, stint $n+1$. Vertical line segments mark discrepancy between conditions with prefatory background strain (i.e., from stint n; hollow markers) and contemporary background strain (i.e., from stint $n+1$; filled markers). Markers omitted where no significant fitness change detected between stint $n+1$ and stint n baseline ($\alpha = 0.005$). Segment color highlights sign-change effect: harmful trait becomes beneficial with contemporary strain (blue) or vice versa (red). Subplots differ in focal strain sample type (rows) and whether background-strain balancing diversity maintenance mechanism was enabled (columns). Note that no assays with background strain diversity maintenance mechanism disabled were conducted for population-subject trials, owing to the lack of effect seen in specimen-subject trials (Supplementary Figure 45).

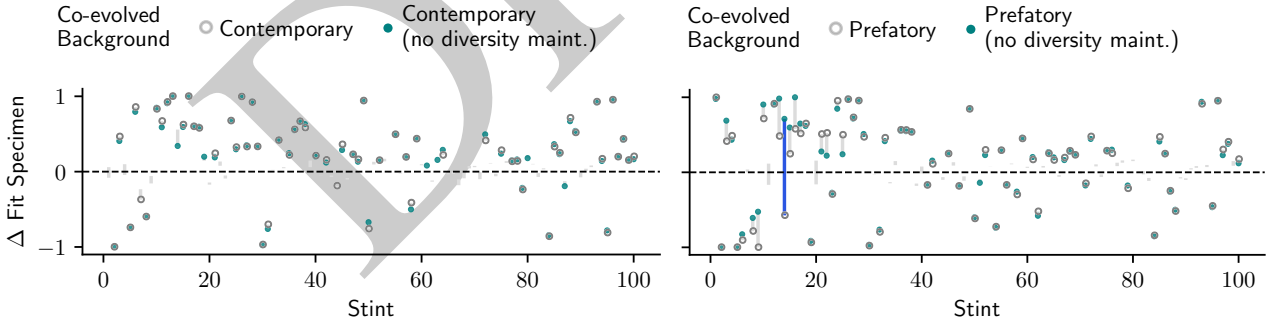


Figure 45: **Screen for fitness trade-offs driven by diversity maintenance scheme.** Plots report influence on focal strain competition assays attributable to diversity maintenance mechanism. Experiments competed focal strain samples (specimen from stint $n+1$) against baseline focal strain population from stint n. Outcomes recorded between 1 (fitness gain; stint n baseline driven extinct) and -1 (fitness loss; stint $n+1$ driven extinct), with near-zero values indicating comparable fitness. Vertical line segments mark discrepancy between conditions with diversity maintenance (hollow markers) and without diversity maintenance (filled markers). Markers omitted where no significant fitness change detected between stint $n+1$ and stint n baseline ($\alpha = 0.005$). Segment color highlights sign-change effect: harmful trait becomes beneficial without diversity maintenance mechanism (blue) or vice versa (red). Subplots differ in source of ecological context (columns). Under prefatory context, background strain from stint n; under contemporary context, stint $n+1$.

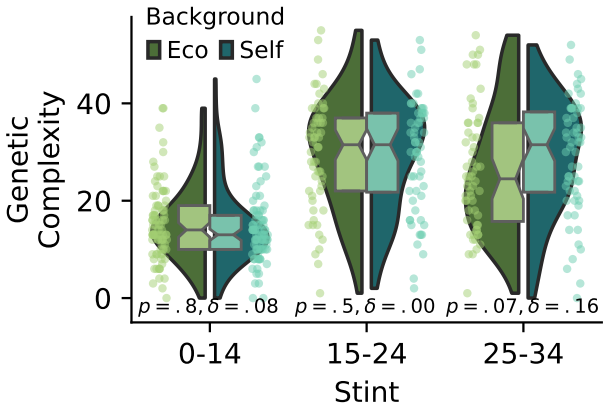


Figure 46: **Self context effect on genotype complexity maintenance in early case study strain lineage history.** Replay experiments for stints 1-34 were conducted with design as shown in Figure 7a, but sample size increased from 1 to 6 independent replay replicates per stint. Binning chosen to reflect emergence of morph e at stint 15, and provide comparable sample sizes to analysis of later timepoints in Figure 7. Similar complexity outcomes arose between ecological context and self context conditions, although marginally higher complexity outcomes were observed under self context for replays of stints 25-34 (Mann-Whitney U test/Cliff's δ).

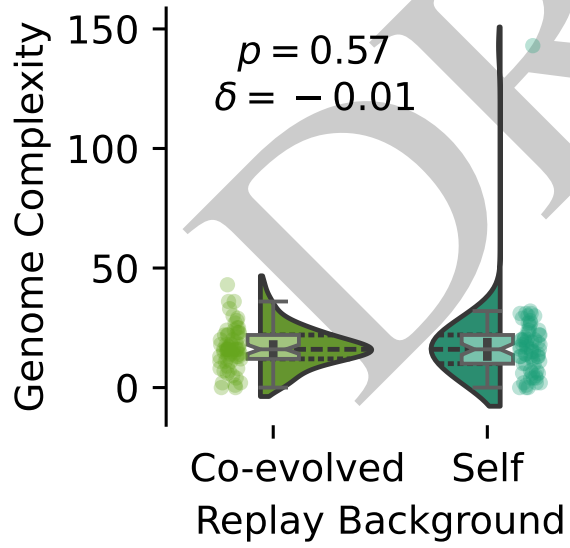


Figure 47: **Alternate rendering of Figure 7c.** Includes 142-site genotype complexity outlier from self-background treatment. Dose size is 50 mutations.

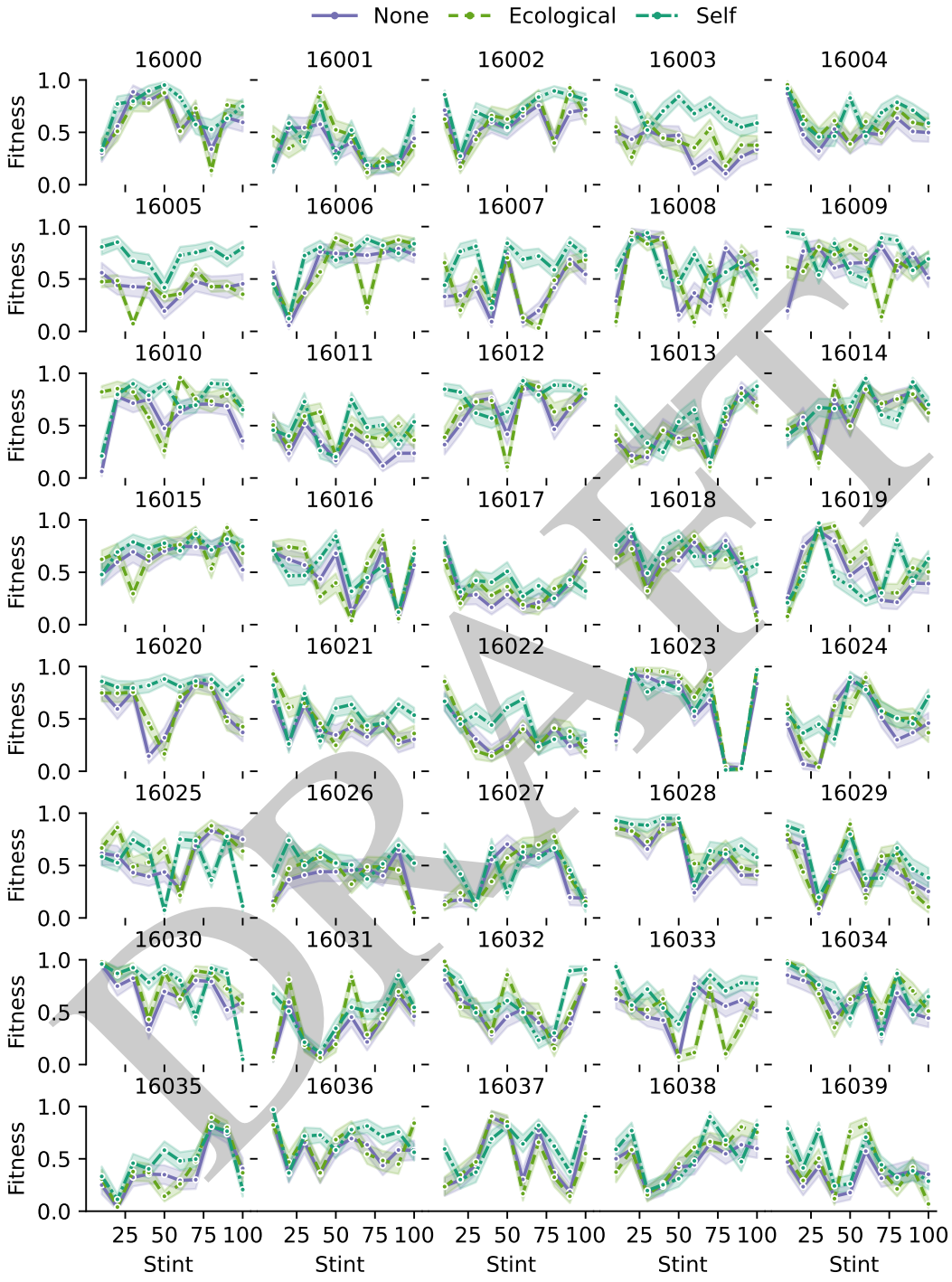
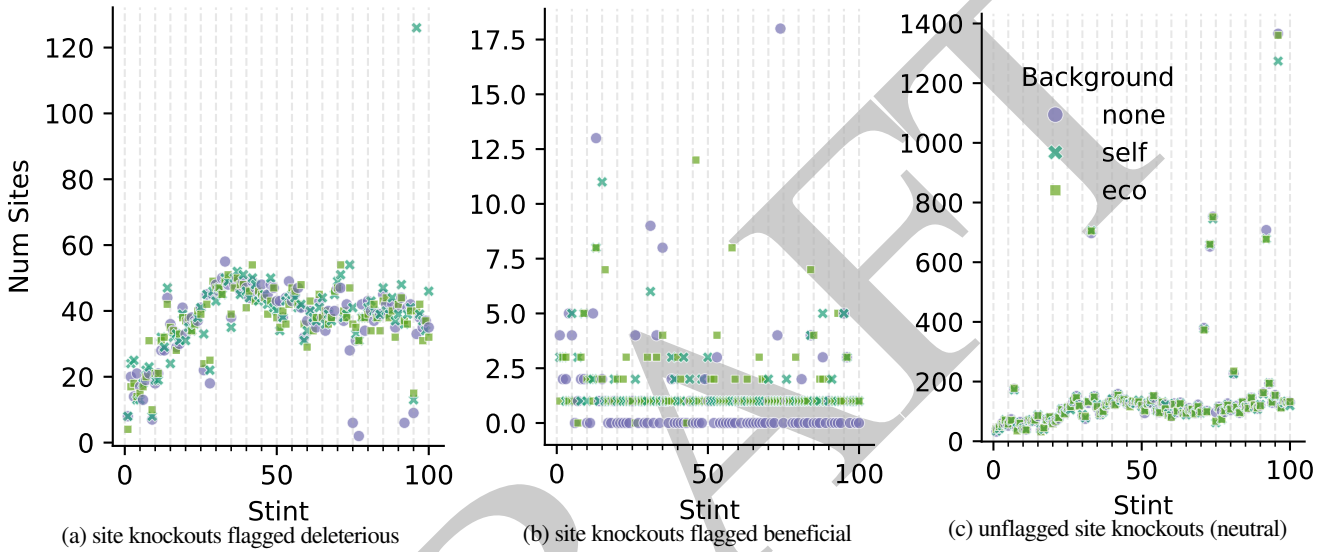
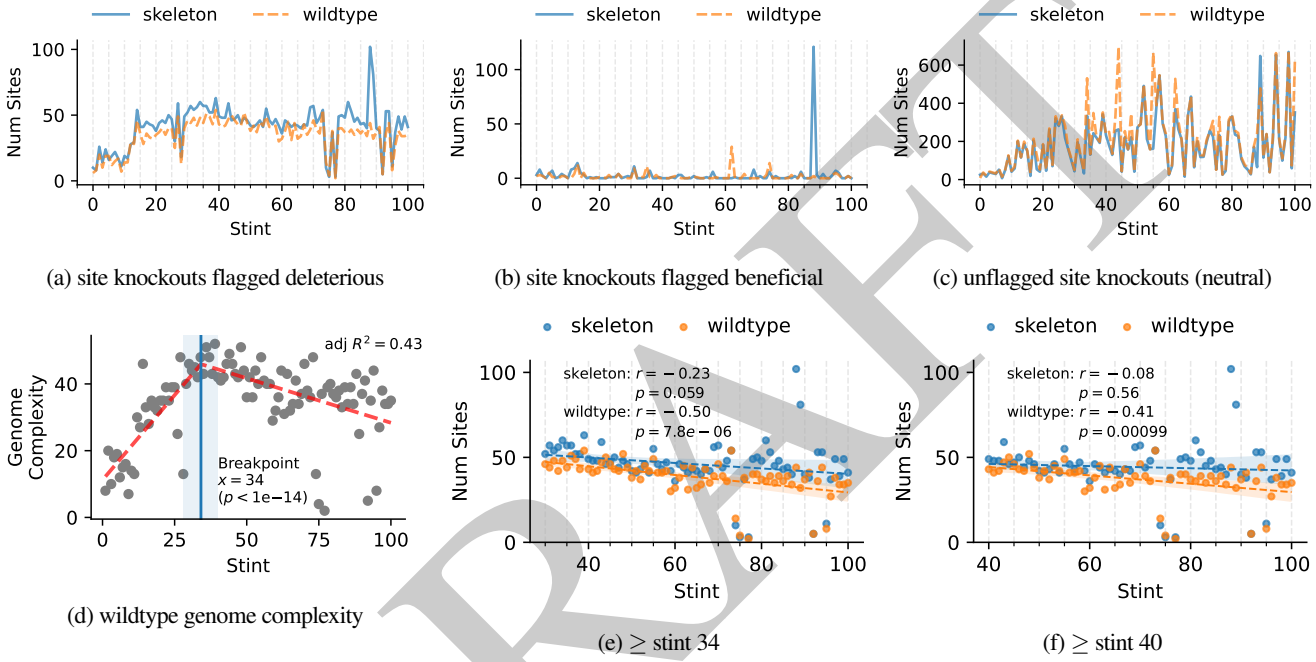


Figure 48: **Relative fitness trajectories across independent evolutionary replicates.** Reported fitness is measured for specimen samples as mean end-state prevalence outcome across "tournament" of competition experiments against contemporary specimen from other independent evolutionary replicates. For background type "none," competition is initialized 50/50 between compared strains (Figure 9a). Otherwise, competition is initialized 25/25 between compared strains, with remaining half initialized as a population sample from either (1) the assayed specimen's co-evolved background strain ("ecological") or (2) the assayed specimen's own strain population (Figure 9b).



(a) site knockouts flagged deleterious (b) site knockouts flagged beneficial (c) unflagged site knockouts (neutral)

Figure 49: **Genotype complexity measures incorporating background context.** After a preprocessing step to exclude sites not found to produce observable phenotypic effects when nopped out, end-state prevalence outcomes from competition trials between wildtype and site-knockout strains are screened against prevalence outcome distributions from 20 wildtype-versus-wildtype trials to flag those with apparent knockout fitness effects. Main text reports competition trials performed with no background, meaning competitions were initialized 50/50 between compared strains (Figure 9a). We also tested alternate designs, where competition is initialized 25/25 between compared strains, with remaining half initialized as a population sample from either (1) the assayed strain's co-evolved background strain population ("ecological" context) or (2) the assayed strain's own population (self context) (Figure 9b). Fitness complexity (knockouts flagged deleterious) was generally comparable across assay designs (panel 49a), although several substantial exceptions are notable. Specimens of morph i at stints 74, 75, and 92 from appearing as very low genotype complexity under the no-background assay are measured as higher complexity in assay designs incorporating background context (both ecological or self). At stint 96, a high-complexity outlier is measured as 126 deleterious single-site knockouts under self-context assay design. Panel 49b gives site knockouts flagged advantageous, which are generally low (matching expectation) and roughly consistent across assay designs. For completeness, panel 49c gives site knockouts considered as producing observable phenotypic effects but flagged neither deleterious nor advantageous.



(a) site knockouts flagged deleterious

(b) site knockouts flagged beneficial

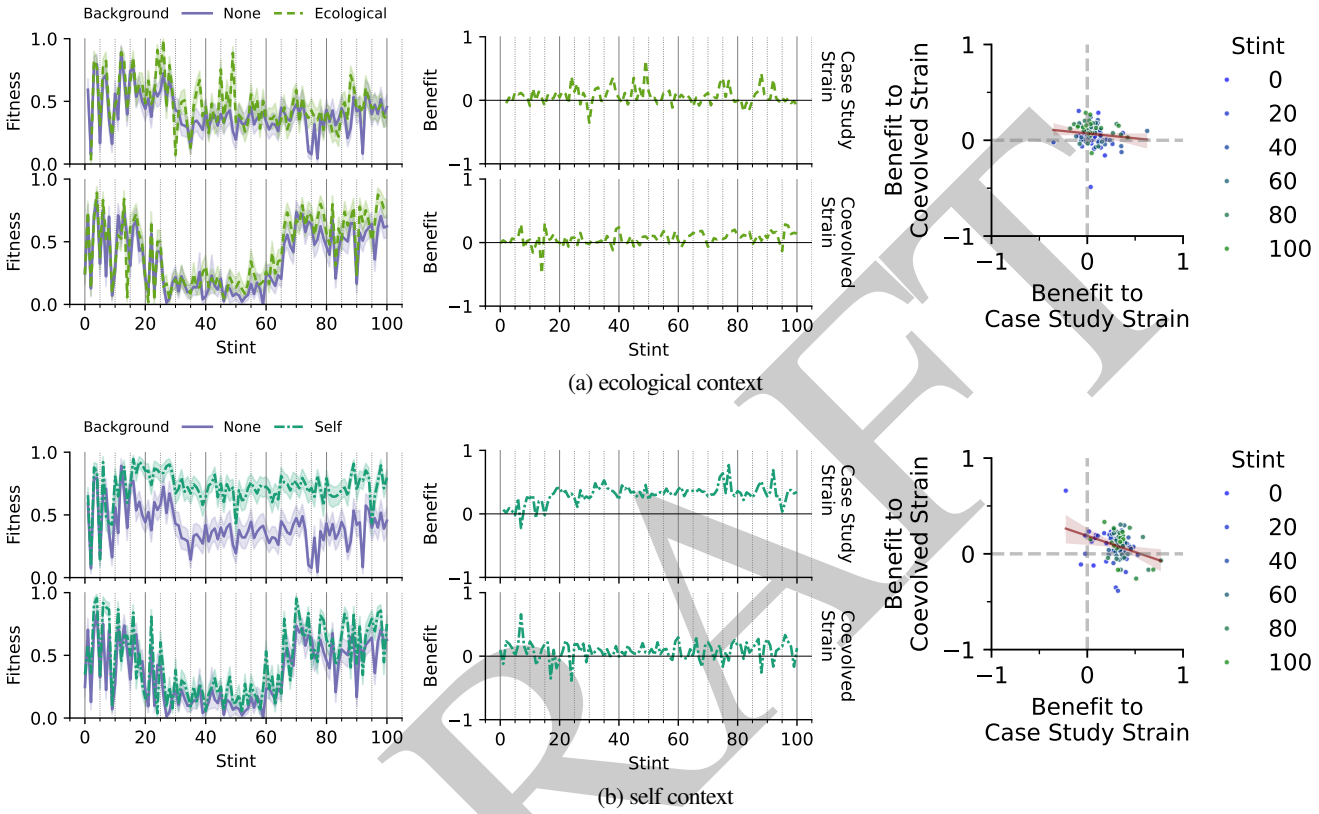
(c) unflagged site knockouts (neutral)

(d) wildtype genome complexity

(e) $\geq$ stint 34

(f) $\geq$ stint 40

Figure 50: **Deep (cryptic) genotype complexity in case study strain.** To assess genetic functionality not detectable by single-site knockout (e.g., due to redundancy), knockout targets were additionally assayed on "skeletonized" (rather than wildtype) genome variants. To create these variants, instructions were sequentially replaced with NOP placeholders until no more could be silenced without causing detectable phenotypic change. Fitness competitions were then conducted between skeletonized genomes and variants with all possible additional single-site knockouts. Panel 50a compares site knockout counts flagged deleterious for case study strain against skeletonized and wildtype backgrounds. For completeness, panels 50b and 50c provide counts of knockouts flagged advantageous and those flagged neither advantageous nor deleterious, respectively. Regression analysis suggests that around stint 34 case study strain transitioned from early-evolution regimen of increasing complexity to a late-evolution regimen of decreasing complexity (panel 50d, shaded area indicates 95% CI). Applying Pearson regression analysis to the skeletonized "deep" genotype complexity measure (i.e., including possible cryptic genotype complexity), the trend in decreasing complexity weakens — with complexity remaining particularly more stable from stint 40 onwards (panel 50f).



(a) ecological context

(b) self context

Figure 51: **Coincident fitness effects of case study strain and co-evolved strain.** Fitness is measured as mean end-state prevalence outcome across "tournament" of specimen competition experiments between subject strain and strains from other independent evolutionary replicates. For background type "none," competition is initialized 50/50 between compared strains (Figure 9a). Panel 51a (top) compares reciprocal fitness effects of the case study strain and its coevolved strain (Figure 14) as each other's ecological context. To measure fitness in ecological context, competition is initialized 25/25 between compared strains, with remaining half initialized as a population sample from the subject strain's coevolved background strain (Figure 9b). In tournament fitness competitions against foreign strains from independent replicates, the case study strain and its coevolved strain both tend to benefit from familiar context (exact Binomial tests; $k=70$, $n=99$, $p=5\times10^{-5}$; $k=73$, $n=98$, $p=1\times10^{-6}$). Although in several instances one strain fares worse in the other's presence, only a marginal weak negative correlation exists between their reciprocal fitness effects (Pearson correlation, $r=-0.13$, $p=0.22$) — suggesting strong adversarial dynamics are not at play. For completeness, panel 51b (bottom) compares contemporary fitness effects of the case study strain and its coevolved strain (Figure 14) as their own biotic context. To measure fitness in self context, competition is initialized 25/25 between compared strains, with remaining half initialized as a population sample from the subject strain itself (Figure 9b). Although case study strain exhibits higher sustained self-benefit, both strains tend to benefit from self context (exact binomial tests; $k=96$, $n=99$, $p=6\times10^{-14}$; $k=74$, $n=98$, $p=4\times10^{-7}$). Interestingly, a weak negative correlation arises between the subject and coevolved strain's self-benefit (Pearson correlation, $r=-0.33$, $p=1\times10^{-4}$). One possible explanation is as a statistical artifact, given self-correlation in underlying time series data.

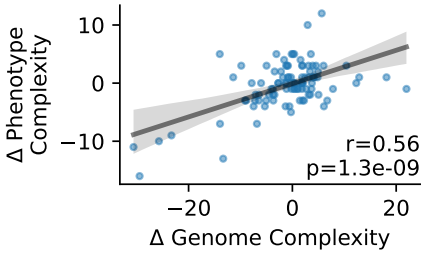


Figure 52: **Correlation between changes in genotype and phenotype complexity in case study strain.** Specimen complexity changes are relative to estimated ancestral specimen. Significance and strength reported are via Spearman's rank correlation. Value $r = 0.56$ corresponds to moderate correlation.

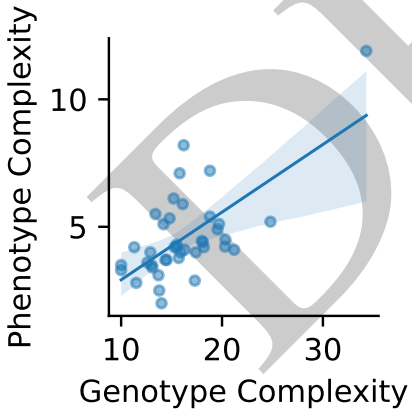
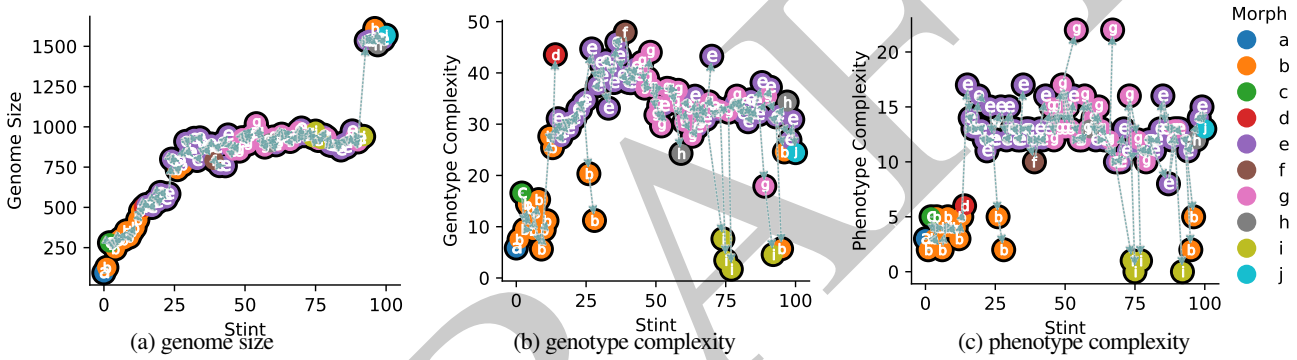
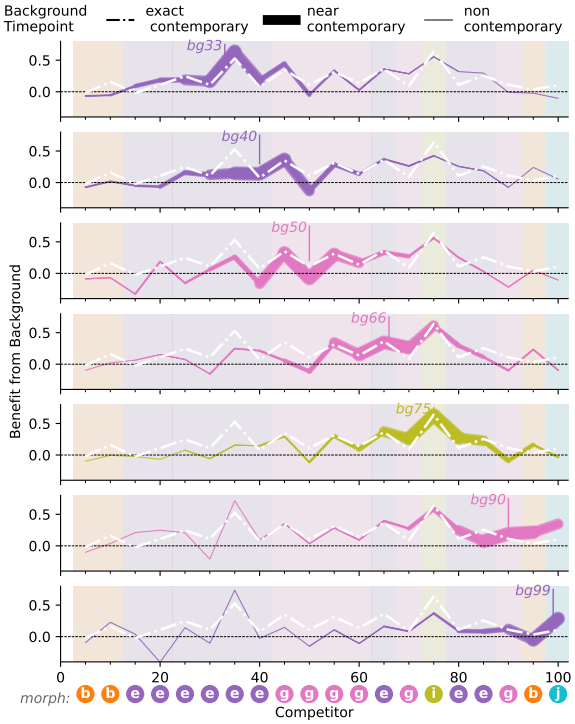


Figure 53: **Significant correlation between genotype and phenotype complexity across replicates.** Mean values per replicate, measured at 10-stint intervals ($p < 5 \times 10^{-6}$, $R^2 = 0.42$).

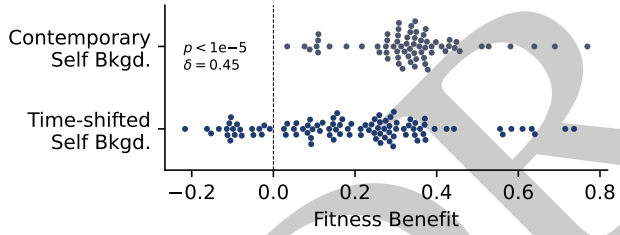


(a) genome size (b) genotype complexity (c) phenotype complexity

Figure 54: **Genome content and complexity.** Genome size (panel 54a) is the total number of genome sites. Genotype complexity (panel 54b) is the count of genome sites with detectable fitness loss in knockout trials. Phenotype complexity (panel 54c) is a count of distinct input/output interfaces between cell and environment/neighbors with detectable fitness loss in knockout trials. Color coding corresponds to morphology categorization (Table 2). Measures are with respect to sampled focal strain specimen at each stint. Arrows indicate estimated phylogenetic relationships.



(a) self-context fitness effect of fixed-timepoint backgrounds



(b) timeshift influence on self-context fitness effect

Figure 55: **Timeshift analysis of self-context fitness effect in case study strain.** To assess the evolutionary stability of self-context fitness benefits observed in the case study strain (Figure 5), additional fitness assays were conducted with focal strain population samples drawn from stints 33, 40, 50, 66, 75, 90, and 99 as background context (panel 55a). Fitness is measured as mean end-state prevalence outcome across "tournament" of specimen competition trials between case study strain and focal strain from other independent evolutionary replicates. Competition is initialized 25/25 between compared strains, with remaining half initialized as a population sample from the case study strain itself (Figure 9b). Color coding indicates morph category of assayed competitor (background fill) and self-context background (line). Line thickness denotes proximity to self-context background timepoint. Dashed white overlay shows fitness effect of exact-contemporary self-context background. Compared against time-shifted effects, exact-contemporary background contexts produced greater fitness benefit (Mann-Whitney U test, panel 55b).

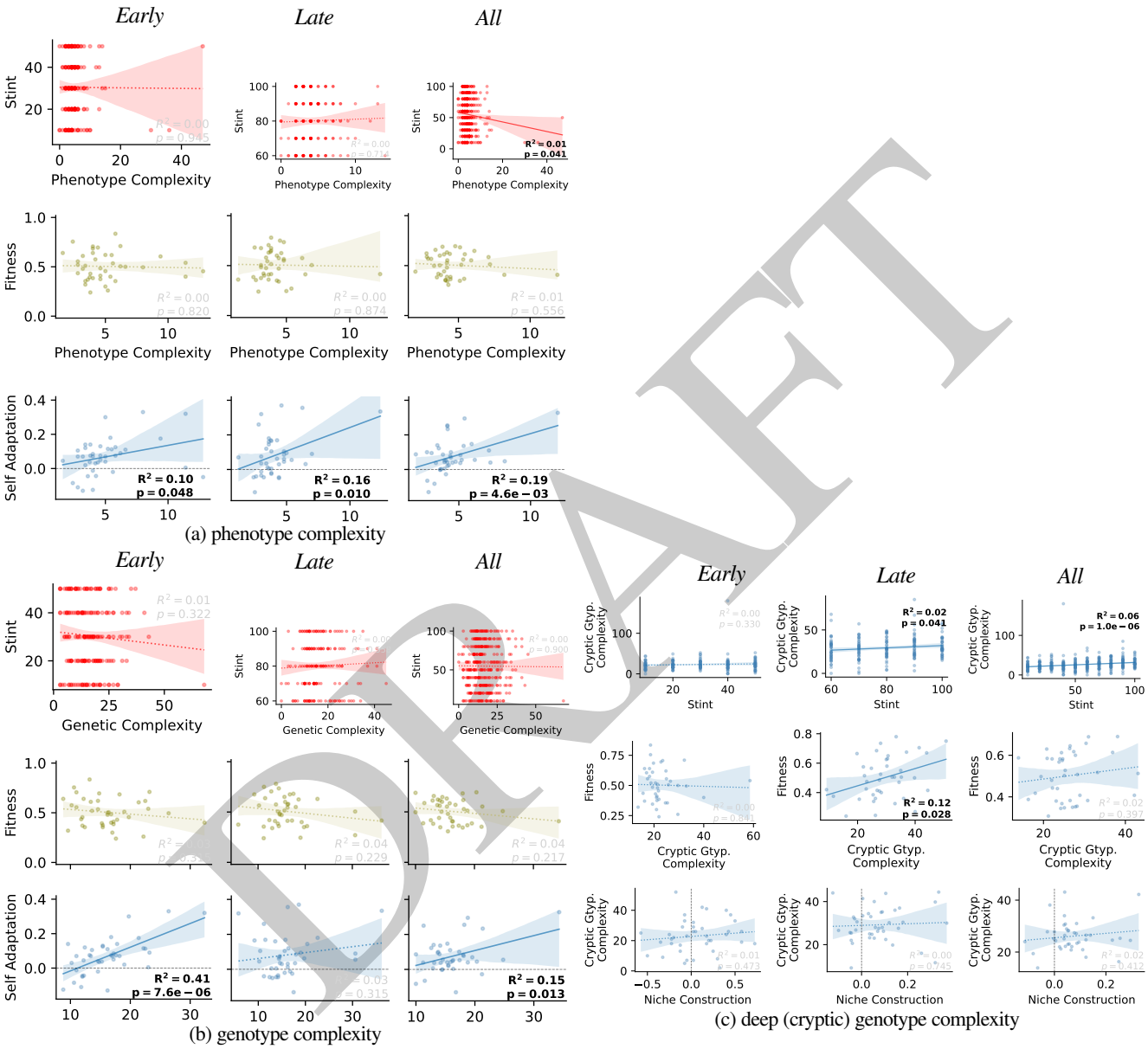


Figure 56: **Co-occurrence of drift (time), niche construction (self adaptation), and fitness with complexity across replicates.** Pearson correlation and two-tail Wald tests reported.