Supplemental Online Material

### Supplemental Methods

#### Double barcode design

##### Overview

The DNA barcoding system developed in Levy et al. [-@Levy15a] allows for the use of two distinct DNA barcodes – one in the landing pad itself, introduced via homologous recombination, and a second, which is introduced via *Cre-lox* mediated recombination. In Levy et al. [-@Levy15a], only a single sequence was used for the landing pad barcode, but here we used the landing pad barcode as a low complexity barcode to encode the experimental condition under which a population was evolved, while we used the other barcode (as in Levy et al. [-@Levy15a]) for lineage tracking itself. Figure S1 details the stepwise approach by which the barcoded populations were generated.

##### Ancestral strain construction

To generate the ancestral strain for barcoding, we first constructed strain HR206, which contains the Magic Marker (Tong et al. 2004) by **XXX**. *HR206* was then crossed strain *SHA321* [@Jaffe17a], which carries the pre-landing pad, *Gal-Cre-NatMX*, at the *YBR209W* locus [@Levy15a]. spores derived from this cross were grown on Nourseothricin (Nat), to select for the pre-landing pad locus, and then one of these was backcrossed to FY3 (Winston et al., 1995). This process was repeated 5 times, each time selecting for spores containing the landing pad. A single spore derived from the final backcross was used as the ancestor for generating haploid barcoded libraries; one more mating with FY3 was performed to obtain the diploid ancestor.

We next introduced the barcoded landing pad into the pre-landing pad locus by homologous recombination with NatMX (Figure S1). The landing pad contains a *lox66* site, a DNA barcode (which we refer to as BC1), an artificial intron, the 3’ half of *URA3*, and *HygMX*. We PCR amplified the fragment of interest from the plasmid library L001 (~75,000 barcodes), described in Jaffe et al. [-@Jaffe17a] and transformed to obtain a library of landing pad strains containing a BC1 barcode. The transformation protocol is as described in detail in Levy et al [-@Levy15a].

##### Construction of high diversity libraries

We selected 101 barcoded landing pad haploid strains and 117 barcoded landing pad diploid strains, listed in Table S2 with their associated environment for evolution. Each individual barcoded landing pad was then transformed using the plasmid library pBAR3-L1 (~500,000 barcodes) described in Levy et al. [-@Levy15a]. This plasmid carries *lox71*, a DNA barcode (referred to as BC2), an artificial intron, the 5’ half of *URA3*, and *HygMX*. Transformants were selected onto SC +Gal –Ura, to allow expression of the Cre recombinase, which is under the *GAL1* promoter. The recombination between *lox66* and *lox71* is irreversible and brings the two barcodes in close proximity to form an intron within the complete and functional *URA3* gene. For each landing pad strain, we generated transformants. The plates were scraped, and transformants from each plate were stored separately in glycerol at -80°C.

To estimate barcode diversity in each of the single libraries, we performed high throughput sequencing of the barcode region following DNA extraction (described in detail below). Using this sequence data, we quantified the distribution of frequencies of all unique barcodes present in each single library. We then pooled non-overlapping sets of these single libraries in order to obtain 12 pools (per ploidy) estimated to contain ~500,000 unique lineages (combination of BC1 and BC2). The number of landing pad (BC1) strains per pool ranged from 4 to 16. The minimum Hamming distance between landing pad barcodes within any pool is 6 for the diploid pools and 7 for the haploid pools.

**Table S1.** Primers and details on DBC construction, primers, etc.

WIP

**Table S2.** with statistics on library size, diversity, from ancestral pools.

WIP

#### Experimental evolution protocols

##### Growth conditions and transfer regime

Each starting pool was evolved by serial batch culture in 500 mL baffled flasks in various conditions described below (Table S3). Unless otherwise specified, vegetative propagation was conducted in in 100 mL of each media at 30°C with shaking at 225 rpm.

Base medium for most conditions was SC complete (YNB+ Nitrogen- 1501-500, SC-1300-500, Sunrise Science Products) supplemented with 2% Glucose (BD-Difco-215510). The evolution experiments were started with a pre-culture of each pool grown in 100 mL of SC-2% Glucose at 30°C overnight. This pre-culture was used to inoculate both replicate evolutions with cells (~400 µL).

**Table S3.** Media recipe per source environment.

WIP

Serial transfers were performed by transferring cells (~ 400 µL) into fresh media at the designated transfer time (24 hours for most environments; Table S3). We generated archival glycerol stocks (3 1 mL culture at 16.6% final glycerol concentration), and pelleted, washed, and concentrated the remaining culture volume (90 mL) for later DNA extraction and barcode sequencing (described below).

#### Bulk Fitness Assay

##### Assembling clone pools

We generated pools of adapted genotypes by isolating individual yeast clones from each of our evolution environments. To do so, we first determined appropriate time point(s) from which to sample in order to gather a large number of unique lineages carrying distinct beneficial mutations.

When isolating lineages from time points throughout the evolutions, we were faced with trying to balance requiring that clones be adapted with the necessity of sampling as many *unique* adapted lineages at once. We first measured population mean fitness using bulk population fitness assays against a fluorescently labeled reference strain (describe below) to determine time points with high mean fitness. For many environments, sampling clones from time-point(s) shortly after mean fitness began to consistently increase yielded sufficiently high numbers of lineages that were adapted and had unique barcodes.

However, several evolution environments showing positive mean fitness were dominated by a very small number of lineages, such that few unique barcodes were recovered after clone isolation. To solve this problem, we took advantage of low-coverage barcode sequencing of various time points from the initial evolutions. Using a simple heuristic to define whether barcodes were adapted, we calculated the time point which would yield the largest number of unique and adapted barcodes in a sample of clones per time point.

To isolate clones from the selected time points, we inoculated a ~10 µL aliquot of frozen glycerol stocks of the target sample into 5 mL of YPD and grew this pre-culture overnight. This culture was suspended at cells per mL in PBS before clone isolation into 96-well plates by way of single cell sorting.

##### Bulk competition protocols

WIP: Parris

#### High-throughput barcode sequencing

##### DNA extraction

From a dry cell pellet stored at -20°, DNA was extracted using the MasterPure™ Yeast DNA Purification Kit (Epicentre MPY80200) with slight modifications compared to the manufacturer’s guidelines: the lysis step was performed for one hour in lysis buffer, supplemented with RNAse at 1.66 µg/µL. Prior to resuspension with sterile water, each DNA pellet was washed twice with 70% Ethanol to remove remaining contaminants.

Following quantification via QuBit™, DNA was resuspended to a final concentration of 50 ng/µL.

##### PCR Amplification

Amplification of the barcode region was performed as previously described [@Levy15a; @Venkataram16a], with the following amendments.

**Table S4**. Primers used for library construction

WIP

**Table S5.** Time points and replicates used for fitness inference from all BFA assays.

WIP

#### Fitness inference

##### Demultiplexing Illumina reads and counting barcodes

We first divide reads into individual libraries based on inline indices following the unique molecular index on the paired-end reads. We discard reads if the average quality score of the barcode regions is less than 30. The unique molecular index (UMI) for a set of paired reads is the first 8 bases of read 1 plus the first 8 bases of read 2. For each library, we discard reads with duplicate UMIs. Finally, we extract barcodes by searching the barcode region, plus 10 bases on either side, using the regular expression below. We discard reads that do not match this regular expression

('\D\*?(GTACC|GGACC|GGTCC|G.TACC|GG.ACC|GGT.CC|GGTA.C|GGTAC.)(\D{24,28})(.TAACT|A.AACT|AT.ACT|ATA.CT|ATAA.T|ATAAC|AAACT|ATACT|ATAAT)\D\*')

##### Clustering / error correcting

Because of PCR and sequencing errors, many of the barcodes counted in the previous step are simply errors and should not be counted as “true” barcodes. To error correct the set of barcodes for each experiment, we cluster errors to true barcodes based on the edit distance between barcodes. We use the following algorithm to cluster the environment barcodes and diverse barcodes separately. The algorithm takes advantage of the fact that all errors should be connected by single insertions, deletions, and substitutions by using deletion neighborhoods to speed up the error-correction process. The algorithm uses as input a list of barcodes and total counts (reads) corresponding to each barcode. The steps are listed below:

* Make deletion neighborhoods
  + For each barcode, create the set of single-base deletions at each position
  + Connect barcodes with overlapping single-base deletion sets *Note: this overlap indicates that the two barcodes are separated by one “edit”: a substitution, insertion, or deletion*
  + Within each neighborhood, define peaks by these criteria:
    - Barcode does not contain an uncalled base (“N”)
    - Barcode has no single-edit neighbors with more total counts
    - Barcode has more than 10 total counts
    - Barcode is more than 3 edits away from any peak with more total counts
  + Within each neighborhood, error correct non-peak barcodes:
    - Check the Levenshtein (edit) distance between each non-peak barcode and each peak barcode. If the edit distance is less than or equal to 3, the barcode error-corrects to the peak barcode. If a barcode is within 3 edits of more than one peak, it corrects to the barcode with higher total counts *Note: This step uses the python Levenshtein module* (https://pypi.python.org/pypi/python-Levenshtein/0.12.0#license)

Once we have error-corrected both the environment and diverse barcodes, we define the combinations of “true” environment and diverse barcodes, which we call “centroids.” We discard combinations that have less than or equal to 10 total counts and add the counts from error barcodes to the appropriate centroids to produce a final list of barcodes and counts.

##### Final steps

Because the bulk fitness assays were all sequenced over multiple lanes, we perform the above steps for each lane separately and then combine count files based on full barcode (environment barcode + diverse barcode) sequences. We discard any full barcodes that are not found in all lanes, which removes any lane-specific sequencing contamination. Chimeric barcode pairs, which arise from sequencing or PCR issues, occur at a low rate in our reads. We identify chimeras by finding barcodes that share a diverse barcode (but not an environment barcode) with another barcode that has at least 100X more reads. These barcodes are removed from the dataset.

### Whole-genome clone sequencing

#### Clone selection

WIP: Parris/Milo

**Table S7.** Adapted clones per environment inferred from BFAs, with number with WGS sequencing data.

WIP

#### Library preparation

WIP: Lucas/Parris

#### Variant calling pipeline

**Table S8.** Number of mutations discovered per clone per source and ploidy.

WIP

The pipeline for variants calling has been detailed in Li, et al., 2019 (Nature Eco&Evo). Briefly, Sentieon Genomic Tools Version 201711.02 (REF: doi:10.1101/115717) were used for SNP, small indel and structural variants calling with *S. cerevisiae* S288C reference genome R64-1-1 (https://downloads.yeastgenome.org/sequence/S288C\_reference/genome\_releases/). The source code for variants calling and annotation (section 7.2-7.6) can be found at https://github.com/liyuping927/DNAscope-variants-calling.

##### FASTQ processing

For each sample, we received two .fastq files, one for each read of the paired end sequencing (‘fastqR1’ and ‘fastqR2’). Using cutadapt v.1.16, we trimmed the first 10 bp of each read (-u 10), low-quality ends (-q 30) and any adapter sequences (-a). After trimming, sequences with a length shorter than 12 bp (--minimum-length 12) were discarded.

First we trimmed the forward read, writing output to temporary files (note, commands are a single line):

cutadapt --minimum-length 12 -q 30 -u 10 -a CTGTCTCTTATACACATCTCCGAGCCCACGAGAC -o tmp.1.fastq.gz -p tmp.2.fastq.gz fastqR1 fastqR2

Next we trimmed the reverse read, using the temporary files as input:

cutadapt --minimum-length 12 -q 30 -u 10 -a CTGTCTCTTATACACATCTGACGCTGCCGACGA -o trimmedR2.fastq.gz -p trimmedR1.fastq.gz tmp.2.fastq.gz tmp.1.fastq.gz

We then mapped reads using bwa (citation) to *S. cerevisiae* S288C reference genome R64-1-1 (<https://downloads.yeastgenome.org/sequence/S288C_reference/genome_releases/>) and sorted using Sentieon Genomic Tools v.6:

bwa mem -M -R readGroupInfo -K 10000000 ReferenceGenome trimmedR1.fastq.gz trimmedR2.fastq.gz) | sentieon util sort -o SORTED\_BAM --sam2bam -i

Duplicates were removed using the sorted .bam file. The first command collected read information, and the second command performed the de-duping:

sentieon driver -i SORTED\_BAM --algo LocusCollector --fun score\_info SCORE\_TXT  
sentieon driver -i SORTED\_BAM --algo Dedup -- rmdup --score\_info SCORE\_TXT --metrics DEDUP\_METRIC\_TXT DEDUP\_BAM

Local realignment around indels was performed using the deduped .bam file:

sentieon driver -r ReferenceGenome -i DEDUP\_BAM -- algo Realigner REALIGNED\_BAM

Lastly, base quality score re-calibration was performed using the realigned .bam file. The first command calculated the required modification of the quality scores assigned to individual read bases of the sequence read data. The second command applied the re-calibration to calculate the post calibration data table:

sentieon driver -r ReferenceGenome -i REALIGNED\_BAM --algo QualCal RECAL\_DATA.TABLE  
sentieon driver -r ReferenceGenome -i REALIGNED\_BAM -q RECAL\_DATA.TABLE --algo QualCal RECAL\_DATA.TABLE.POST

##### SNP and small indel variant calling

SNP and small indels variants were called by the DNAscope algorithm (Sentieon Genomic Software) using the realigned .bam file and the output table of the base quality score recalibration (RECAL\_DATA.TABLE). The parameter ploidy is assigned as 1 for haploids and as 2 for diploids.

sentieon driver -r ReferenceGenome -i REALIGNED\_BAM -q RECAL\_DATA.TABLE --algo DNAscope --ploidy [1|2] VARIANT\_VCF

##### Structural variant calling

The first command enabled the DNAscope algorithm to detect the break-end variant type (BND). The parameter ploidy was assigned as 1 for haploids and as 2 for diploids. The second command processed the temporary .vcffile using the SVSolver algorithm and output structural variants to a .vcf file.

sentieon driver -r ReferenceGenome -i REALIGNED\_BAM -q RECAL\_DATA.TABLE --algo DNAscope --var\_type bnd --ploidy [1|2] TMP\_VARIANT\_VCF

Then process the .vcf using the SVSolver algorithm with the following command:

sentieon driver -r ReferenceGenome --algo SVSolver -v TMP\_VARIANT\_VCF STRUCTURAL\_VARIANT\_VCF

##### Variant annotation

Here the .vcf file from SNP and small indel variants calling (VARIANT\_VCF) is used as an example. The same commands were used to annotate structural variants.

We then used snpEff7 (<http://snpeff.sourceforge.net/download.html>) to annotate a .vcffile and output the annotated .vcffile, named Ann.vcf.

java -Xmx2g -jar snpEff -c snpEff\_config -v R64-1-1.75 -class VARIANT\_VCF > Ann.vcf -s snpEff\_summary.html

For variants in coding regions, SNPSift was used to extract the first annotation of each variant, which is the annotation with the largest effect. Output the extracted annotation as a .vcf file, named Final\_Ann.vcf:

java -jar SnpSift extractFields Ann.vcf CHROM POS ID REF ALT QUAL FILTER EFF[0].EFFECT EFF[0].GENE: EFF[0].IMPACT: EFF[0].FUNCLASS: EFF[0].CODON: EFF[0].AA ANN[0].BIOTYPE: GEN[0].GT GEN[0].AD GEN[0].DP GEN[0].GQ GEN[0].PL > Final\_Ann.vcf

For variants in non-coding regions, the nearest gene of each variant was extracted. Thus, the non-coding variants were annotated as either the upstream or downstream of the nearest genes.

##### Filtering SNPs, small indels, and structural variants

First, mitochondrial variants were discarded. Second, any variants in genes *FLO1* and *FLO9* were filtered out due to poor alignment in both genomic regions. Third, diploids with an average coverage lower than 15 and haploids with an average coverage lower than 10 were discarded. Fourth, background variants were removed. If a variant is present in >~12% of clones isolated from the same evolutionary condition, this variant is considered as a background variant and discarded. Fifth, variants with a quality score smaller than 150 were filtered out. Note that if a variant was present in multiple clones, the alignment of this variant was manually checked regardless of its quality score and a decision was made based on all clones carrying this variant. Thus, a variant with a quality score <150 may not be filtered out if the same variant contained in other clones was proven to be authentic. Similarly, a variant with a quality score >150 may be filtered out if the same variant was proven to be bogus in other clones.

Furthermore, all variants were further verified by manually checking .bam files after alignment. By doing this, variants within repetitive regions and regions with a poor alignment were filtered out. More importantly, due to mishandling, sequencing data used here is contaminated by other yeast species/isolates at a low frequency (0-30%). While this low frequency contamination does not pose a big problem for variants calling in haploids, it leads to excessive miscalling of heterozygous variants in diploids. These miscalled heterozygous variants are caused by genetic variations between S288C (the *S. cerevisiae* strain used in this study) and the contaminated yeast source and often appear in “patch” (multiple variants within a small region, e.g. 4 variants within 100 bp), which is statistically impossible considering the short-period evolutionary time. To remove false variants caused by contamination, we manually checked alignment through .bamfiles and removed heterozygous variants appearing in patch. In addition, the ratio of ref:alt is used to assist the removal of false variants. Variants with Ref:Alt >3:1 are very likely to be a result of low-frequency contamination. Lastly, ambiguous variants were checked using blast. Variants that are not present in other yeast species/isolates were considered as *de novo* mutations arising during the course of evolution.

##### Identifying large rearrangements and aneuploidies

WIP: Lucas

### Statistical analyses of fitness and pleiotropy

WIP Parris

#### Statistical analysis of fitness patterns

##### Identifying adapted lineages

##### Identifying auto-diploidized lineages

##### Clustering and variance partitioning

Table S8. Number of lineages in each of clusters defines for each source environment, as well as robustness statistics to groupings.

WIP

#### Linking phenotypes with genotypes

##### Mutual information

##### Hierarchical gene–pathway analysis

### References