High-throughput, high-precision colony phenotyping with *pyphe*

Stephan Kamrad[1,2,3,+], Jürg Bähler[1] and Markus Ralser[2,3,+]

1. University College London, Institute of Healthy Ageing, Department of Genetics, Evolution and Environment, London, United Kingdom
2. The Francis Crick Institute, Molecular Biology of Metabolism Laboratory, London, United Kingdom
3. Charité Universitätsmedizin Berlin, Department of Biochemistry, Berlin, Germany

+ For correspondence: stephan.kamrad@gmail.com and markus.ralser@charite.de

Running title: Colony screens with *pyphe*

Keywords: Screen, Colony, Fitness, Functional genomics, Phenomics, Microbiology, Python software, large-scale phenotyping, Cell viability, Growth curve

Abstract

Colony fitness screens are powerful approaches for functional genomics and genetics. This protocol describes experimental and computational procedures for assaying the fitness of thousands of microbial strains in numerous conditions in parallel. Data analysis is based on *pyphe,* an all-in-one bioinformatics toolbox for scanning, image analysis, data normalisation and interpretation. We describe a standard protocol where endpoint colony areas are used as fitness proxy and two variations on this, one using colony growth curves and one using colony viability staining with phloxine B. Different strategies for experimental design, normalisation techniques and quality control are discussed. Using these approaches, it is possible to collect hundreds of thousands of data points, with low technical noise levels around 5%, in an experiment typically lasting two weeks or less.

1. Introduction

Measuring properties of densely arrayed microbial colonies is a powerful and efficient technique for determining the fitness of large panels of microbial strains. Today, numerous knock-out or over-expression mutant libraries, wild strain collections, segregant panels and synthetic genetic array construction methods are available to underpin systematic, data-driven investigations. Owing to specialised robotics, colony screens are largely automated and can be done at huge scales, e.g. with millions of double knock-out mutants ***(1)***. Colony screens have been employed to measure the fitness of large segregant libraries for the dissection of complex traits ***(2, 3)*** and the characterisation of wild strain libraries ***(4–6)***. All such experiments require scalable computational approaches for data acquisition, data management, batch correction and statistical analysis. To this end, we have recently published a new toolbox named *pyphe* which enables fast, reproducible and flexible pipelines for colony screen analysis (Figure 1) ***(7)***.

One key challenge of colony screens is that the observed colony phenotype is affected by factors other than the strain's fitness. Batch effects, e.g. originating from media preparation or incubation time, make colony sizes not directly comparable between different plates. Even within plates, nutrient, moisture and temperature gradients as well as differences in inoculation mass due to uneven pinning can result in variation between colonies located in different areas of the plate. A well-known example is the "edge effect", where outermost colonies grow larger due to a lower degree of nutrient competition. Normalisation is therefore key for obtaining reliable fitness estimates. Edge effects, and other effects that affect rows and columns evenly, can in some cases be rectified by dividing by the row/column median, and other, regional biases can sometimes be rectified by comparing colonies to their neighbours (e.g. as implemented in SGAtools ***(8)***). Both these normalisation methods only work if the null effect can be reliably estimated, i.e. if the majority of colonies show no growth effect. An alternative approach is to include a grid of control strains on every plate (Figure 2) ***(9)***. Each colony is then compared to essentially a weighted average of the neighbouring controls, resulting in a relative fitness that is standardized within and across plates (as long as the same control strain is used). *Pyphe* implements row/column median normalisation as well as a grid normalisation, giving the user full control over the normalisation strategy. Using grid normalisation, it is possible to achieve technical noise levels of around 5%, enabling the detection of subtle differences in fitness.

Classically, plates are imaged once and colony footprint areas are used as fitness proxy. Recent methods have expanded on this in two areas. ScanLag ***(10)***, Quantitative Fitness Analysis ***(11)***, Colony-live ***(12)*** and Scan-o-matic ***(9)*** use repeated imaging of colonies and extract parameters from the resulting growth curve. Additionally, Colony-live and Scan-o-matic use transmission scanning where the grey-value of a pixel can be used as a proxy for colony thickness. These methods achieve lower noise values than endpoint measurements but require substantially higher investments in scanners (which need to be housed in a temperature-controlled environment), data storage and analysis. We recently found that, at least within of the parameters tested, the growth rate parameters extracted from timeseries correlate tightly with endpoint sizes ***(7)***. This is compatible with a model where colonies grow until the media is exhausted by the ensemble of colonies, giving each colony a fixed time-window for growth. This is an important difference to screens conducted in multi-well plates with liquid media, where each culture is given a fixed amount of resources. *Pyphe* supports the acquisition and analysis of colony growth curves as well as traditional endpoint measurements, allowing the user to choose the method most suitable for their experiment and laboratory setup.

Another variation on classical screens is the use of phloxine B as a dead cell stain ***(13)***. If this is included in the media, the colour of a colony reflects the proportion of dead cells in it ***(7)***. This provides an orthogonal readout that is often uncorrelated to colony sizes after both readouts are standardised to the control strain. As phloxine B in the media generally does not affect cell growth, it can be included in all plates, whereby a second fitness dimension is added to the experiment at little extra cost and effort. *Pyphe* can acquire and analyse colour images and extract quantitative redness scores which then integrate seamlessly into the rest of the analysis pipeline.

We here present a detailed protocol for colony fitness screens, split into an experimental and computational part. The basic protocol describes an endpoint experiment using colony areas as fitness proxies. In our opinion, this should be the default for most investigations and the starting point for new users. Two variations on this are described: How to record and analyse colony growth curves and how to analyse colony viability by phloxine B staining. These, together with the detailed accompanying notes, are meant as a starting point for users to implement their own workflows. By providing a framework of data formats and analysis steps, much of it will be applicable to researchers not using *pyphe* but looking to implement their own computational or experimental procedures. Experimental details will vary between microbial species, assay conditions and laboratory settings. *Pyphe* is set up as an open-source, collaborative software project and we welcome user feedback and contributions for its continuous improvement.

2. Materials

2.1. Experimental

1. Sterile microbial growth media (liquid and with agar)
2. Rectangular agar plate dishes, e.g. Singer PlusPlates
3. Microwave (or autoclave) for melting agar media
4. 50ml serological pipettes and pipette pump
5. Conical flask (e.g. 250ml), shaking incubator

6. Pinning robot and consumables (e.g. Singer RoToR with 96 long RePads, 96 short RePads and 1536 short RePads)
7. Cling film
8. Plate incubator
9. Strain library cryostocks in 96 or 384 format, ideally with unique footprints
10. Cryostock of a suitable control strain

Optional

11. Incubator to house scanners (for timecourse imaging)
12. Phloxine B (for colony viability measurements)
13. Reagents required for testing specific growth conditions of interest (e.g. drugs, chemicals or nutrients)

2.2. Scanning and Data Processing

1. Scanner (Epson V800 or V850 recommended) with a fixture to hold plates into place on the scanner surface (laser cutting guide available on https://github.com/Bahler-Lab/pyphe/tree/master/Documentation/Scan). Please see Figure 3 for more information on how to set up the scanner.
2. A computer with Ubuntu (or another Linux operating system) and Python3 (e.g. Anaconda3) for scanning.
3. Any computer with any operating system with Python3 installed for the rest of the analyses.

3. Methods

3.1. General Method: Endpoints

3.1.1. Experimental design and plating

1. Pick an appropriate control strain for your library. For knock-out, over-expression and similarly constructed libraries this should usually be the library background strain ("wild type"). For collections of wild isolates or segregant libraries, the choice of control strain is more flexible, but it should not fall on either extreme side of the distribution of fitness phenotypes.
2. Plan the arrangement of library and control strains on the final assay plates. Overall, five different types of plates are required throughout the workflow: library plates, grid plates, combined plates, source plates and assay plates (Figure 2). Consider how many replicates are required to answer your biological question (see Note 1). Plan how to rearrange the library plates around the control grid and prepare the layout file of your assay plates (see Notes 2 and 3).
4. Prepare sterile growth media and pour agar plates for waking up the library and for the preparation of the grid plate (see Note 4). We usually use rich media for waking up strains and preparing the grid plate, even if the screen is in the end carried out on minimal media.
5. Wake up the control strain and your library. This protocol is applicable to a wide range of microbial organisms and growth media, so we cannot specify incubation times for plates. The general advice for the preparatory phase is that plates should be incubated until colonies are fully (or almost fully) grown as this makes pinning easier, however, not so long that colonies get stressed due to starvation or desiccation.

6. Prepare grid plates by growing a 40ml liquid culture of the control strain overnight. Then, pour the culture into an empty PlusPlate and, using 96 long RePads, pin from liquid to solid to make the grid plates. Do not use target mixing. Prepare multiple grid plates as required (plus some spares) as the number of times a grid plate can be pinned from is limited to approximately 10.
7. Once grown, re-array grid and library plates as shown in Figure 2. Use 96 short RePads for this. On the Singer RoToR, you can either use the inbuilt 16-to-1 array mode or make your own pinning program in custom mode.
8. To make the source plates, copy the combined plates onto fresh agar plates, using 1536 short pins. This step is important because the combined plates are a patchwork and should therefore be copied again to make consistent, evenly arrayed source plates. Besides, this step can serve as an amplification step for larger screens with many assay plates. Simply prepare several copies of each source plate, approximately one for every 10 assay plates.
9. Prepare your assay plates with drugs, inhibitors or particular nutrient composition as required (Note 4). Make one or two spares per condition in case something goes wrong during pinning.
10. Copy your source plates onto the assay plates using 1536 short pins. Consistent, small inoculums are obtained with low pinning pressure (5-10%). Clearly label each plate with a unique ID (see Note 5). Monitor the quality of pinned plates (see Note 6).
11. Wrap plates in cling film and place in an incubator. Plates should always rest upside-down to prevent condensate dropping onto the colonies. Ideally, there should be no strong air currents or temperature gradients in the incubator which can lead to plates drying out and colonies growing unevenly.
12. Let the colonies grow until they stop or barely grow anymore (usually 2-3 days). Please see ***(7)*** for detailed reasoning for this choice of time point.

3.1.2. Acquisition

1. To install *pyphe*, run the following command in your terminal (see Note 7).

   ```
   $ pip install pyphe
   ```

   ImageMagick needs to be installed, the scanner needs to be accessible with SANE (Scanner Access Now Easy) and the TPU8x10 scanning mode must be available. This is the case by default for recent versions of SANE, which is included in the standard Ubuntu installation.

2. Image your plates once they have grown. Turn on the scanner and run

   ```
   $ pyphe-scan --fixture som3 --postfix POSTFIX --nplates N
   ```

   Where `POSTFIX` is the name of your experiment and `N` the total number of plates you have to scan. The fixture argument specifies how to crop individual plate images from the original scan. Parameters for our fixture model are pre-programmed (use `som3_edge` and `som3` for images containing or not containing the edge respectively), but it is possible to configure your own fixtures in the *pyphe-scan* script. Plates are always placed in the scanner with the agar on the bottom (i.e. not upside-down) and with the A1 colony position in the back-right corner (see Figure 3). Remove the plate lids, lower the scanner top and make sure it is even. Press ‘y’ when prompted to start scanning. Swap plates as instructed. *Pyphe-scan* will create a folder with the current date and the `POSTFIX` in the current directory in which images will be saved.

3. In parallel, prepare a table mapping the scan sequence to the plate ID. Later, expand this to include information about the layout, experimental condition and any additional meta-data (date, batch, comments etc.) you want to keep track of. This will be your experimental design table (EDT, see Note 5).

3.1.3. Quantification

1. Extract colony sizes from images by running

   ```
   $ pyphe-quantify batch --grid auto_1536 --pattern “images/*.jpg”
   ```

   The `batch` parameter indicates the mode in which to run *pyphe-quantify.* In batch mode images are analysed one at a time and colony areas are extracted. The `--grid` option specifies how to assign identified colonies to grid positions. In automatic mode, as used here, the grid is detected based on pixel row and column intensity peaks. There are other ways to manually define grid positions when this is required, please see the documentation. The `--pattern` argument can be used to specify which images should be analysed, in this case, it is assumed that all your images are located in a folder ‘images’ in the current working directory and are in jpg format.
2. Carefully check whether the results are as expected. A QC image is produced for every single analysed image. Investigate if colonies were identified correctly and that colonies are assigned the correct row and column (see Note 8). *Pyphe-quantify* has numerous parameters that can be adjusted, check the documentation for details.

3.1.4. Analysis

1. Edit your EDT table to include a column ‘Data_path’ which contains relative file paths to each image’s data file (these are located in the ‘pyphe_quant’ directory by default). If you have to exclude individual images for QC reasons, simply delete the corresponding lines from the EDT table.
2. Aggregate data across plates and perform grid and subsequent row/column median normalisation by running

   ```
   $ pyphe-analyse --edt EDT.csv --format pyphe-quantify-batch --
   gridnorm standard1536 --rcmedian --qc_plots analyse-qc
   ```

   The `--edt` option specifies the relative path to your experimental design table and should be adjusted if required. The `--format` option describes the type of input data, in this case colony areas produced by *pyphe-quantify*. The `--gridnorm` argument enables grid correction and the following parameter specifies where the grid colonies are located on the plate. "`standard1536`" refers to the recommended position of two 96 grids in the top left and bottom right corners. In some cases, when your grid strain grows differently than your library, it might not be appropriate to use row/column normalisation (see Note 9) after grid correction. In those cases, simply leave out the `--rcmedian` parameter. The `--qc_plots` option tells *pyphe-analyse* to produce a QC report for each plate in the experiment, which will be saved in the specified folder. There are additional options which are explained in the documentation.

3. Once the command has finished, join your layout table onto the produced colony report (in R or python or the programming language of your choice). Alternatively, you can use the `--load-layouts` option to load strain IDs directly into your colony report.
4. Perform quality control suited to your experiment (see Note 10). For each plate, check that the footprints are empty and check the coefficient of variation of the additional grid colonies. Discard plates from the analysis as required.

3.1.5. Interpretation

1. Compute summary statistics and p-values comparing each strain to the control strain (repeated across all conditions).

   ```
   $ pyphe-interpret --ld pyphe-analyse_data_report.csv --
   circularity 0.85 --control CONTROL-STRAIN --grouping_column
   Condition --axis_column Strain
   ```

   The `--ld` option specifies the path to the colony report generated in the previous step. The `--circularity` option triggers a filter that excludes all colonies with a circularity below the specified value. This is useful in order to exclude image analysis artefacts from further analysis. The `--grouping_column` argument specifies a column in the colony report for subsetting your data. In each subset, i.e. for each unique entry in the grouping column, a different set of t-tests will be performed. The `--axis_column` argument specifies the explanatory variable of your t-test. The `--control` option specifies your control strain and this needs to be the string identifier also found in the axis column of your colony report.

2. Compute summary statistics and p-values comparing each condition to the control condition (repeated across all strains). This tests for condition-specific differences in fitness, correcting for basal growth of the strain.

   ```
   $ pyphe-interpret --ld pyphe-analyse_data_report.csv --
   circularity 0.85 --control CONTROL-CONDITION --grouping_column
   Strain --axis_column Condition
   ```

   For example, in a scenario where a slow-growing mutant (relative to the wild-type grid strain) has been measured in a large number of conditions and shows a consistent growth defect (e.g. a fitness of 0.8) in all conditions (including a control condition without stress), the first command would result in a significant p-value for this strain in all conditions as each time it is compared only to the wild type (which has a fitness of 1) in the same condition. This second approach would show no significant difference in fitness as the fitness in each condition is the same as in the control condition.

3. Proceed to analyse your data according to your question. For calling hits, it is usually useful to define an effect size threshold additional to your p-value cut-off.

3.2. Variation 1: Time course analysis

Pyphe implements methods to analyse growth curves of colonies, obtained by automated, regular imaging of plates. Characteristics of the growth curve, usually the maximum slope, are extracted and corrected for plate position effects, similarly as endpoint colony sizes. This method typically achieves lower noise values, but we nonetheless only recommend it for specialised applications as the maximum slopes extracted from growth curves correlates tightly with much more easily obtained endpoints ***(7)***. Please also see Note 11 for a list of other software resources, some dedicated specifically to colony growth curve analysis.

To record and analyse growth curves, prepare your assay plates as described in the **Experimental** section above. Then proceed as follows:

1. Immediately after pinning, place the assay plates into the scanner without lids. Take a note of the plate ID corresponding to each position. Close the lid so that it rests horizontally on the plates. If desired, place the scanner in an incubator or temperature-controlled environment.
2. On the scanner computer, run *pyphe-scan-timecourse*.

   ```
   $ pyphe-scan-timecourse --nscans NSCANS --interval INTERVAL --
   postfix POSTFIX --fixture som3
   ```

   NSCANS is the number of scans in the timecourse, INTERVAL is the time between the end of a scan and the start of the next (we recommend 20 to 30 minutes). After the first scan, check that everything is working as expected and that the first images look ok.
3. After scanning has finished, manually inspect the images. Often, it is necessary to remove some images at the end of the timecourse, e.g. because colonies started touching each other, contaminations or condensation appeared, or the plates dried out.

4. For every plate separately, extract growth curves using *pyphe-quantify* in timecourse mode.

   ```
   $ pyphe-quantify timecourse --grid auto_1536
   ```

   This will produce a table of growth curves, where individual colonies are the columns and timepoints are the rows.
5. For every plate separately, extract growth curve parameters using *pyphe-growthcurves*.

   ```
   $ pyphe-growthcurves --input INPUT --plots
   ```

   Input is the path of the csv file produced by *pyphe-quantify*.
6. Proceed with analysis as above. Run *pyphe-analyse* using the `--format pyphe-growthcurves` option. The Data_path column of your EDT table should contain paths to the files produced by *pyphe-growthcurves*. *Pyphe-analyse* will perform corrections on the maximum slope of the growth curves but will retain and include other growth curve parameters in the colony report.

3.3. Variation 2: Colony viability analysis

Addition of the dead-cell stain phloxine B to the media results in colonies with varying redness which is correlated to the fraction of dead cells in the colony. This can be performed in parallel to the standard workflow, imaging each plate twice: once in greyscale transmission mode (for colony sizes) and once in colour reflective mode (for viability analysis). In our hands, viability analysis with phloxine B works better in 384 format and in conditions which are not too stressful, which both results in larger colonies with a stronger signal. Note that phloxine B staining has only been used in a quantitative screening context in *S. pombe*, so some initial validation will be required when applied to other species.

1. Proceed to prepare your assay plates as above but include 5 mg/L phloxine B in the media. A 1000x aqueous stock can be stored in the fridge protected from light for several weeks.
2. Grow plates as usual, but protected from light (phloxine B is light sensitive)
3. Scan plates with the white background attached to the scanner lid.

```
$ pyphe-scan --mode Color --fixture som3-color --postfix POSTFIX
--nplates N
```

4. Analyse the images using *pyphe-quantify* in redness mode

```
$ pyphe-quantify redness --grid auto_1536 --pattern
"images/*.jpg"
```

5. Proceed with the analysis in *pyphe-analyse*. We recommend using only row/column median correction if possible.

```
$ pyphe-analyse --edt EDT.csv --format pyphe-quantify-redness --
rcmedian --qc_plots analyse-qc
```

3. Notes

1. **Power calculations.** At the beginning of the planning phase of an experiment, consider the resolving power required of it to answer your biological question. This will allow you to design cost- and labour-efficient experiments. In the simplest case, you might only be interested in the binary classification of growth/no growth, e.g. when testing the essentiality of genes in specific contexts. In another setting, the goal might be to characterise growth differences down to a couple of percentage points, e.g. when identifying genetic variants of wild strains which have subtle effects on growth. The number of replicates required to address these two

questions differs drastically. You can design your experiment appropriately by calculating the number of required replicates based on the minimal effect size you are trying to detect, the noise of the method (5-10% is a good, conservative estimate in general) and the desired statistical power (chance of correctly rejecting the null hypothesis). This can be done using the stats.power module of the python statsmodels package or the power.t.test function in R, or other online tools. For example, to have an 80% chance of detecting a 5% difference in growth at 5% noise (standardised effect size = 1), with a p-value cutoff of 0.05, you would require 17 samples in each group (two-sided Student's t-test). You also need to consider that correcting for multiple testing will in the end decrease the statistical power of your test. This effect is of course stronger the larger your strain library is. In the case of the Bonferroni correction (which is not recommended), you can easily calculate this loss of power by dividing your alpha by the number of tests. For preferred FDR methods, this is not straightforward and requires you to make an educated guess.

2. **Plate layouts**. T-tests generally require the samples to be independent. What this means in practice is often not entirely clear, and it will be up to you to decide the details of your experiment design. In general, we would not consider replicates located next to each other on the same plate as independent as they are subject to the same pinning errors as well as local nutrient, moisture and temperature regimes. It is therefore not recommended to generate replicates using 1-to-4 or 1-to-16 multiplexing pinning programmes where a single 96 plate is replicated into 384 or 1536 respectively. Instead, replicates should be obtained by recombining library plates in different combinations into the assay plate, mixing up the extract location and neighbours.

Colonies grow in competition and lack of neighbours usually results in increased growth. This is commonly observed as an edge effect where plates on the borders grow bigger. For the same reason, colony screens are sensitive to empty areas of the plate, effectively creating internal edges. If library plates are only partly filled, we recommend filling empty spots with control strains or random strains. This is less important if empty positions are scattered and more so if entire corners, areas, rows or columns are empty.

Despite generally discouraging empty positions, we strongly recommend leaving one individual, unique position per library plate empty. Such footprints serve two important purposes. Firstly, they are negative control positions which should always be empty, if they are not this indicates a source of contamination. Secondly, they aid identification of specific plates in the case of multiple assay plate layouts. Images will not contain the labelling information written on the side of the plates, so in case of a mix-up, the footprints can provide crucial clues to the identity of the plate.

3. **Layout files.** Preparing the layout of your assay plates is a key task that will require some programming/data processing, at least for larger experiments. This is best done with layouts in long format, i.e. using a table with at least three columns (Row, Column, Strain) where every line of the table describes a single colony position. For every library plate in 96 format, take a note of the position into which it is pinned onto the combined plates. When pinning from 96 to 1536 format, there are 16 of these positions (rows 1-4 and columns 1-4). Let's call the row position p_r and the column position p_c. The row and column position of a colony on the assay plate ($1 \le a_r \le 32$ and $1 \le a_c \le 48$) is then related to the position ($1 \le s_r \le 8$ and $1 \le s_c \le 12$) on the source plate by:

$$a_r = 4*(s_r-1) + p_r$$

$$a_c = 4*(s_c-1) + p_c$$

Using this formula, transform the row and column values for each 96 library plate depending on their target position. Create a layout for the grid plate and include it too. Then, concatenate the tables for all plates and sort by row and column.

4. **Agar plate preparation.** To achieve high data completeness and low technical noise, it is crucial that assay plates are flat, without bubbles and of suitable dryness. For this, ensure the following:

- Always let the media cool down to approximately 60°C before pouring plates. Otherwise, the contraction of the agar during cooling will result in unwanted ripples on the surface. When potentially temperature-sensitive drugs are to be included in the assay plates, add these when the media has cooled down and right before pouring to minimise the exposure to high temperatures.
- Always pour plates on a level and even surface. In our experience, this can be done on a standard lab bench without a sterile environment, as long as the plate lid is immediately replaced after pouring.
- Always add a consistent amount (we use 40ml) of agar medium to each plate. This will result in plates with a consistent height and also avoid other artefacts. Thicker plates mean more nutrients are available to each colony which will change the colony size. Use a serological 50ml pipette and take up 5ml more than required. This will prevent bubbles. If you spot any bubbles, suck them back up with the pipette.
- Plates should be dried for a consistent time (we use 45 minutes without lids) before use. Alternatively, closed plates can be left unwrapped on the bench overnight which in our experience results in plates with a suitable dryness for immediate use. Plates can be stored in the fridge/cold room if not used immediately but will require extra drying before use.

5. **Preparation of the EDT table.** Preparing a correct and complete experimental design table (EDT) is a key requisite for obtaining results with *pyphe-analyse*. All plates in your experiment should have a unique ID and this should be clearly written on the edge of the plate (not on the bottom where it would show up in the scanned images). For example, plate IDs could follow the format date_layout_condition_replicate. While scanning, take a note of which

plate ID corresponds to which image in the scan sequence in table format. This table can then later easily be transformed into the EDT required by *pyphe* by adding extra columns with additional meta-data. The final table must be in csv format and the first column must contain the unique plate IDs. There must also be a column called ‘Data_path’ which points to the image data file produced by *pyphe-quantify*. Any additional meta-data, such as date, condition, library versions, comments, batch information, can be included and will be carried through to the colony report. Please see the Documentation folder in the *pyphe* github repository for an example file. Please note that there is no need for all the data files to be located in the same folder, which is convenient if you have large experiments containing several batches. Generally, file paths should not contain spaces or non-standard characters or those with special meanings in the terminal (%,>,?,/,”,*,&, etc.). Use _ or . or - to separate words.

6. **Quality control during pinning.** Even with precise robots, the transfer of colonies by pinning can be prone to errors which, if unspotted, will result in missing or wrong data. A common problem that occurs when target plates are uneven is that entire areas or corners of the plate have no colony inoculums. Such pinning errors are dangerous as they could result in colonies which are absent for technical reasons to be interpreted as unviable phenotypes. *Pyphe* helps spot these errors by detecting missing control colonies and setting all neighbouring colonies to NA, but this only occurs after the damage is done. We therefore strongly advise to check every plate for missing corners and correct pinning by eye. Have one or two spare plates at hand to repeat transfers that have failed. If the problem persists, it can help to increase the target plate pinning pressure. With the Singer RoToR, it helps to avoid pinning errors if plates are consistently placed and pushed into the same corner in their holder.

7. **Use of command line interface.** To use *pyphe*, you need to be familiar with some standard characteristics of command line programs, especially the concept of the working directory and relative file paths. In the terminal, navigate to the base directory of your experiment and run *pyphe* commands from there. File paths need to be defined in relation to

that directory. Commands in this protocol are marked with $, indicating that this is a line of code to be run in the terminal (the $ is not part of the command). Help for each *pyphe* tool is available by running the tool with the --help parameter (e.g. `$ pyphe-analyse --help`)

8. **Quality control during image analysis.** *Pyphe-analyse* produces a qc image for every image analysed. It is crucial to look at the images, even for huge datasets, to make sure colonies have been correctly identified and correctly matched to their grid positions. Issues with colony detection can usually be remedied by adjusting the threshold parameter (using the `--t` parameter), setting a hard threshold (using the `--hardImageThreshold` parameter) or using local thresholding (activated with the `--localThresh` parameter). The last is especially recommended for images with uneven brightness, e.g. those obtained with the Singer Phenobox. By default, *pyphe-analyse* excludes very small objects. If your colonies are very small, please adjust this exclusion parameter using the `--s` or `--hardSizeThreshold` parameters. For gridding issues, it can be worth to switch from automatic grid placement to manual one. Please see the *pyphe-analyse* documentation.

9. **Normalisation strategies.** *Pyphe-analyse* gives the user several options for normalisation strategies. If neither the `--rcmedian` nor `--gridnorm` options are set, no normalisation is performed and the raw data from the plates is simply aggregated and summarised. For screens without a reference grid, *pyphe* can still be used (with row/column normalisation only or no normalisation). However, lowest noise values are obtained if both options are set. In that case, grid normalisation is performed followed by row/median column normalisation. This second normalisation can correct an artefact of the grid normalisation method which slightly over-corrects colonies next to the edge. This is essentially because colonies just off the edge are compared to colonies on the edge and therefore appear relatively smaller (see Figure 1—figure supplement 2B and Appendix 2 of ***(7)***).

However, row/column median normalisation can and should only be used if the majority of strains in each row and column show no effect (i.e. the null effect can be reliably estimated by taking the median). This is usually the case for library screens where most of the mutants behave like wild-type and there are only a few outlying ‘hits’. For wild strain libraries, this case is harder to argue but it could still work if your strains are arranged randomly. It certainly will not work if your grid strain grows differently to the rest of your library (because only some rows/columns have a lot of replicates of the grid strain). In those cases, performing row/column normalisation after grid normalisation will do more damage than good. *Pyphe-analyse* produces qc plots for every plate analysed and you should inspect these carefully to check that the normalisation is working as expected.

10. **Noise statistics for quality control.** Biological noise, technical noise and experimental errors (incorrect plate preparation, mis-labelling, pinning errors) will impact your data and can result in wrong conclusions if they go undetected. We therefore highly recommend to perform extensive quality control to quantify the unexplained variation and spot experimental errors. The use of negative-control positions (footprints) is key and plates with contaminated footprints should be discarded from the analysis. Furthermore, we include a number of control strains on every assay plate. One easy way to achieve this is to include an additional 96 grid of control strains on each assay plate. These colonies are not used during reference grid correction but are expected to have fitness values of 1. Based on these internal controls, it is possible to calculate two key noise indicators: First, the coefficient of variation (CV), the ratio of the standard deviation of the corrected fitness of these control colonies to their mean is an excellent indicator of the level of noise present in the assay. We usually exclude all plates which exceed a certain CV threshold (e.g. 10%). Secondly, the fraction of unexplained variance (FUV) is the ratio of the variance of the control strains and the variance of all strains. I.e. an FUV of 1 indicates that the spread of values is equally broad in the control as it is across the library, which can indicate that the observed variation across the library strains is purely technical. A

suitable cut-off for exclusion of individual plates and conditions will depend strongly on your library. Certainly, an FUV of greater than 1 would be highly unusual and deserved further investigation. It can also be of value to exclude plate and conditions which show very small uncorrected colony sizes. This would indicate that the stressor included was too strong or the nutrients did not support any growth. In these cases, grid correction can introduce artefacts as small colony sizes are extremely noisy. These QC steps differ greatly between experiments, so they need to be performed manually on the *pyphe-analyse* long data report, removing spurious lines or setting them to NA. Once completed, you should proceed with hit calling using *pyphe-interpret*.

11. **Alternative software solutions.** Scan-o-matic ***(9)*** is a sophisticated platform for scanning, image analysis and spatial normalisation. Scan-o-matic pioneered spatial correction with reference grids and additionally supports calibrating pixel intensities to cell numbers, enabling exact quantification of population sizes. Scan-o-matic uses timecourse imaging and requires a specialised hardware set-up, including pixel calibration strips, modifications to scanners and a local area network.

Gitter ***(14)*** is an R package for determining colony sizes from images. It can work with a range of input image types and has robust algorithms for thresholding and grid assignment. *Pyphe* can work with image quantification data from gitter. Unfortunately, gitter is currently archived by CRAN as it requires outdated packages. For experts, it is possible to install it from the archive but this requires manual installation of several dependencies.

SGAtools ***(8)*** provides tools for spatial normalisation (not including reference grid correction) and statistical analysis of colony screen data. Available as a web service without installation.

Colonyzer ***(15)*** has been developed for the Quantitative Fitness analysis workflow ***(11)***. Colonyzer has been designed to work on colonies which have been stamped on plates with manual replicators, resulting in a spot (i.e. a spread-out inoculum).

IRIS ***(16)*** is an advanced image analysis tool for single or timecourse images, specialising in detecting additional colony morphology features such as bacterial biofilm formation. CellProfiler ***(17)*** is a powerful, multi-purpose image analysis tool with which one can assemble analysis pipelines to count and measure colony sizes as well as potentially other morphological parameters.

Acknowledgements

We would like to thank Natalie Barthel for critical reading of the manuscript. The study was supported by the Francis Crick Institute, which receives its core funding from Cancer Research UK (FC001134), the UK Medical Research Council (FC001134), and the Wellcome Trust (FC001134), and received specific funding from the Wellcome Trust (200829/Z/16/Z) and the ERANET COBIOTEC network “SyCoLim” under grant agreement BMBF-031B0931.

4. References

1. Costanzo M, VanderSluis B, Koch EN, et al (2016) A global genetic interaction network maps a wiring diagram of cellular function. Science 353
2. Bloom JS, Ehrenreich IM, Loo WT, et al (2013), Finding the sources of missing heritability in a yeast cross, http://dx.doi.org/10.1038/nature11867
3. Märtens K, Hallin J, Warringer J, et al (2016) Predicting quantitative traits from genome and phenome with near perfect accuracy. Nat Commun 7:11512
4. Jeffares DC, Rallis C, Rieux A, et al (2015) The genomic and phenotypic diversity of Schizosaccharomyces pombe. Nat Genet 47:235–241
5. Peter J, De Chiara M, Friedrich A, et al (2018) Genome evolution across 1,011 Saccharomyces cerevisiae isolates. Nature 556:339–344

6. Kamrad S, Grossbach J, Rodríguez-López M, et al (2020) Pyruvate kinase variant of fission yeast tunes carbon metabolism, cell regulation, growth and stress resistance. Mol Syst Biol 16:e9270

7. Kamrad S, Rodríguez-López M, Cotobal C, et al (2020), Pyphe, a python toolbox for assessing microbial growth and cell viability in high-throughput colony screens, http://dx.doi.org/10.7554/elife.55160

8. Wagih O, Usaj M, Baryshnikova A, et al (2013), SGAtools: one-stop analysis and visualization of array-based genetic interaction screens, http://dx.doi.org/10.1093/nar/gkt400

9. Zackrisson M, Hallin J, Ottosson L-G, et al (2016) Scan-o-matic: High-Resolution Microbial Phenomics at a Massive Scale. G3 6:3003–3014

10. Levin-Reisman I, Fridman O, and Balaban NQ (2014) ScanLag: high-throughput quantification of colony growth and lag time. J Vis Exp

11. Banks AP, Lawless C, and Lydall DA (2012) A quantitative fitness analysis workflow. J Vis Exp

12. Takeuchi R, Tamura T, Nakayashiki T, et al (2014) Colony-live--a high-throughput method for measuring microbial colony growth kinetics--reveals diverse growth effects of gene knockouts in Escherichia coli. BMC Microbiol 14:171

13. Lie S, Banks P, Lawless C, et al (2018), The contribution of non-essential Schizosaccharomyces pombe genes to fitness in response to altered nutrient supply and target of rapamycin activity, http://dx.doi.org/10.1098/rsob.180015

14. Wagih O and Parts L (2014) gitter: a robust and accurate method for quantification of colony sizes from plate images. G3 4:547–552

15. Lawless C, Wilkinson DJ, Young A, et al (2010) Colonyzer: automated quantification of micro-organism growth characteristics on solid agar. BMC Bioinformatics 11:287

16. Kritikos G, Banzhaf M, Herrera-Dominguez L, et al (2017) A tool named Iris for versatile high-throughput phenotyping in microorganisms. Nat Microbiol 2:17014

17. Lamprecht MR, Sabatini DM, and Carpenter AE (2007) CellProfiler: free, versatile software for automated biological image analysis. Biotechniques 42:71–75

Figure legends

Figure 1: Overview of data analysis workflow.

Figure 2: Assembly of assay plates.

Figure 3: Physical setup of scanners. (A) Epson V800 scanner fitted with a fixture. Note that

the white background insert on the scanner lid has been removed in preparation for transmission scanning (1), the plate positions are clearly numbered to avoid mixing up plates (2), the uppermost area of the scanning bed is free and not covered by the fixture as this area is used for sensor calibration (3), small ‘bumper pads’ of cardboard have been attached next to the hinges to ensure an optimal and consistent distance between the lid and the scanner bed once the lid is closed (4), the fixture contains small notches that fit into pockets of Singer PlusPlates which ensure plates can only be inserted in the correct orientation (5), additional tape has been applied to further stabilise the position of plates within the fixture (6) and that the fixture has been securely attached to the scanner (7). For best results, place plates into the scanners without their lids. **(B)** Scanners are positioned in an incubator for timecourse imaging. Scanners are placed on a custom-built shelf for easy access and fully covered with black fabric. The cover prevents light interference and air currents from the incubator fan hitting the plates directly, which would lead to faster and uneven drying.