Intracellular antibiotic persistence

G. Ansmann, April 17, 2026

In biological physics, we often aim to boil down complex biological processes to simple pen-and-paper models that allow us to gain a better overview. In this experiment, we apply this approach to the problem of antibiotic persistence, individual bacteria escaping the effects of an antibiotic, which can lead to recurring infections. We model this with a set of simple differential equations describing the transitions between the normal and persistent state. We will observe this behaviour in populations of the bacterium *Klebsiella pneumoniae* infecting human bladder cells and quantify the transition rates. In the process, we will also apply modern automated robotic and image-analysis techniques for high-throughput experiments that allow for quantifying biological quantities with unusually high precision.

1 Background

A general note on terminology for this experiment: *Cell* always refers to eukaryotic host cells (here: epithelial cells from the human bladder). By contrast, bacterial cells (here: *Klebsiella pneumoniae*) are always referred to as *bacteria*. This is no universal convention, but common usage in this particular subfield.

1.1 Antibiotics and persisters

Antibiotics are drugs that severely harm bacteria, while having little effect on other forms of life, in particular humans. As such they are central to the treatment of bacterial infections. There are two general types of antibiotics:

- ***Bacteriocidal* antibiotics** kill bacteria directly.
- ***Bacteriostatic* antibiotics** stop the growth of bacteria without killing them. These do not just simply delay an infection as they give the immune system time to catch up and then kill the bacteria.

Persisters are individual bacteria who escape bacteriocidal antibiotics by shutting down their metabolism [1]. Roughly speaking, for them a bacteriocidal drug is only bacteriostatic.

Persistence is not triggered by a genetic change like antibiotic resistance. Instead, it is a phenotypic switch: normal bacteria randomly become persisters ("fall asleep") and also persisters randomly turn back to normal ("wake up") without their genome being affected. This effect can be demonstrated on the single-cell level using microfluidic devices (Fig. 1). While it is random whether an individual bacterium turns into a persister (and vice versa), the rates at which this happens can depend on the environmental conditions. One example of how such a switch is realised inside a bacterium is a noise-sensitive genetic toggle switch as explored in the experiment *Genetic toggle switch*.

Persistence is an example of *bet hedging* by stochastic phenotypic switching: While persisters are not beneficial to population growth under normal conditions,

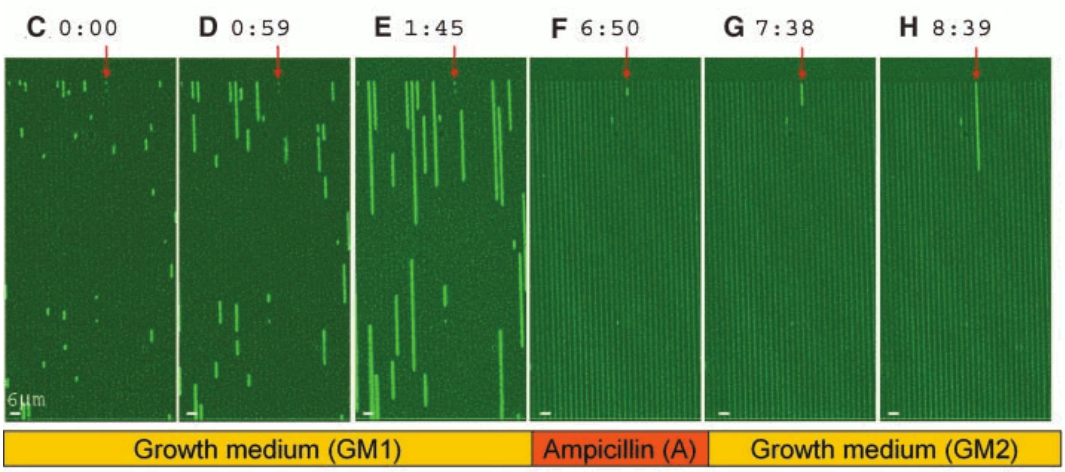


Figure 1: Demonstration of persistence on the single-cell level: Microscopic images of a microfluidic device that traps the bacteria in tiny grooves (columns). Bacteria appear brighter than the background and can only grow in vertical lines when replicating. The red arrow marks a presumed persister well. Ampicillin is an antibiotic. From Ref. 2.

they can make a population survive an antibiotic treatment. Thus, by having a certain fraction of persisters, the population can increase its overall survival chances in unpredictable environments. Such a stochastic switching can be evolutionary favourable to individuals sensing environmental changes and adapting their metabolism as a reaction [3]. The main reasons for this is that sensing comes with a cost which may be too high and that environmental changes affect the bacteria too quickly. For example, an antibiotic may inhibit or kill bacteria before their sensing mechanisms can be active and trigger an adaption. In this context, it is important to consider that most actions and interactions of bacteria are a side-product of growth. For example to become a persister a bacterium must shut down its own metabolism, and any molecules that enact this are only produced as a side product of the replication machinery. If an antibiotic attacks this machinery, it may already be too late for the bacterium, if persistence happens only as a reaction.

1.1.1 Intracellular persisters

In this experiment, we will focus on the bacterium *Klebsiella pneumoniae*, which is responsible for urinary-tract infections (amongst others). It uses the following strategy to escape the immune system and antibiotic treatment:

Klebsiella pneumoniae can enter host cells, namely epithelial bladder cells and multiply within those. This is something most infectious bacteria do not do and even *Klebsiella pneumoniae* only does this with a small rate: Only one in a thousand cells becomes *intracellular.* Intracellular bacteria then turn into persisters with an increased rate, escaping antibiotic treatment. Within a cell, the bacteria are not as accessible to the immune system and can multiply without much hindrance after the antibiotic treatment. Once the host cell bursts, bacteria re-emerge in larger numbers, can overcome the immune response and re-establish the infection.

From a medical point of view, this is of particular interest, as it can lead to recurring infections.

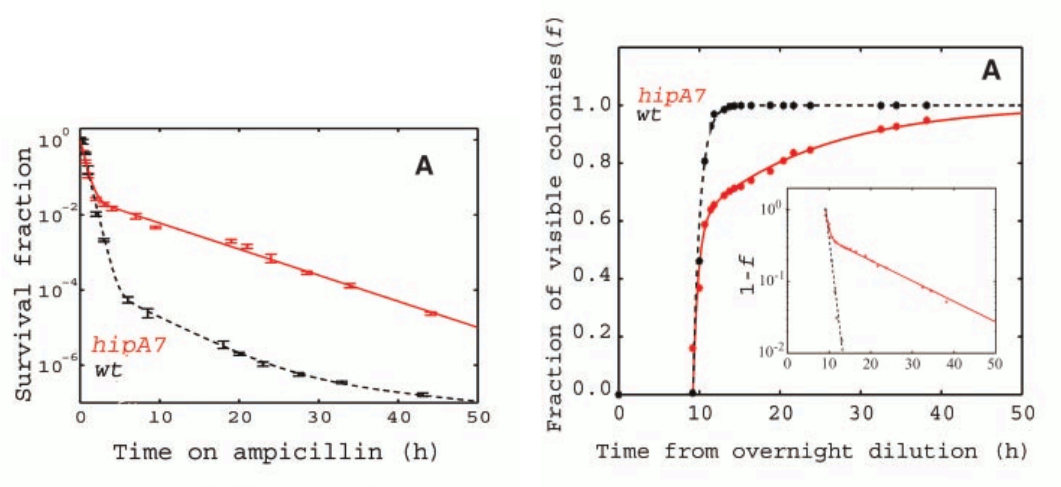
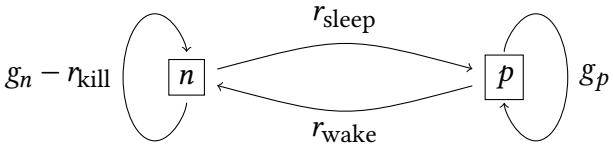


Figure 2: Killing curve (left) and wake-up curve (right) for two strains of Escherichia coli. wt *refers to the wild type,* hipA7 *to a mutant that has an increased ratio of persisters. Ampicillin is an antibiotic. The fraction of visible colonies corresponds to the fraction of woken-up colonies and is explained in Sec. 1.2. From Ref. 2.*

1.1.2 Modelling persisters

In a full model of persistence, we have to cover the following transitions:



Here, n is the number of normal bacteria, p is the number of persisters, g_n and g_p are their respective growth rates, and the rests of the rates are self-explanatory. A typical consequence of these transition rates is that the killing curves and wake-up curves of persisting bacteria are bimodal, i.e., they feature two regions of exponential decay with different rates (see Fig. 2).

For an individual bacterium, the transition into and out of persistence is a random process for all our means and purposes. On this basis, the most detailed model considers individual bacteria and their transitions stochastically (continuous-time discrete-state Markov process). While such a model has the advantage of correctly reflecting the stochastic fluctuations at low numbers of bacteria, it becomes unnecessarily unwieldy for large numbers such as those in this experiment.

An alternative is to discretise time and consider transition probabilities between different times in form of a transition matrix (discrete-time discrete-state Markov process or discrete-time Markov chain). This is done amongst others in Ref. 3.

Instead, we here look at the continuous deterministic limit as a system of ordinary differential equations. This limit describes the average over an ensemble of realisations of the process, which we take as a good approximation of reality for sufficiently high abundances. For our system, we obtain [2]:

$$\begin{aligned} \dot{n} &= g_n n - r_{\text{sleep}}\, n + r_{\text{wake}}\, p - \eta_{\text{kill}}\, n, \\ \dot{p} &= g_p p + r_{\text{sleep}}\, n - r_{\text{wake}}\, p. \end{aligned} \tag{1}$$

In most cases, we can simplify the model considerably, e.g., we can often assume $g_p = 0$ since persisters hardly grow.

1.2 Experimental techniques: growing and measuring cells and bacteria in vitro

In this experiment, you will encounter three different ways to perform microbiological experiments in the lab:

- In a ***culture*** bacteria are suspended in a liquid nutrient medium, in which they can freely grow. At high concentrations, bacteria (like cells and other contents on the micrometer scale) cause the water to be turbid. This can be used to estimate their number by measuring the *optical density* (OD). In this experiment, this is only used as a quick check or rough estimator; for more detail, see the experiment *Exponential growth and bacterial growth laws*.

- **Agar plates** contain nutrient medium embedded in a rather solid gel (agar). Bacteria can grow on the surface, usually forming colonies, i.e., roughly circular mounds consisting mostly of bacteria.

  This can be used to measure concentrations of bacteria by *plating* them: A droplet containing bacteria is spread on the plate. If the concentration of bacteria is sufficiently low, individual bacteria are sufficiently spread apart such that many can become the sole ancestor of a distinguishable macroscopic colony. By counting the colonies, we can estimate the number of *colony-forming units* in the original solution, which in turn is a good estimator for the number of bacteria. (For more detail, see the experiment *Exponential growth and bacterial growth laws* once more.)

  If the individual colonies are sufficiently far apart, they do not interact with each other, e.g., by competing for nutrients or space. While the growth dynamics within a colony is complex as it depends on the access of the bacteria to nutrients and space [4], isolated colonies of a given bacterial strain grow in a very similar speed and fashion.

  However, individual colonies start growing with different lags. These lags are caused by the initial bacterium adapting its metabolism to the new conditions on the plate, but also turning from a persister into a regular bacterium. By comparing the growth progress of different colonies to each other, we can estimate the relative lag times of the bacteria from which they emerged [5].

- The in-vitro cultivation of cells of a multicellular organism is usually called ***cell culture***. The cell lines used for these usually originate from cancer cells, which are characterised by having a mutation that disables the regulation of growth that is normally found in multicellular organisms and which would prevent most in-vitro experiments. Cell cultures are more difficult to maintain than cultures of bacteria (or other microbes) since the cells are highly adapted to the specific conditions within the organism. In many experiments (including this one), the cells only grow in a single layer on the specifically coated bottom of a container containing a suitable medium.

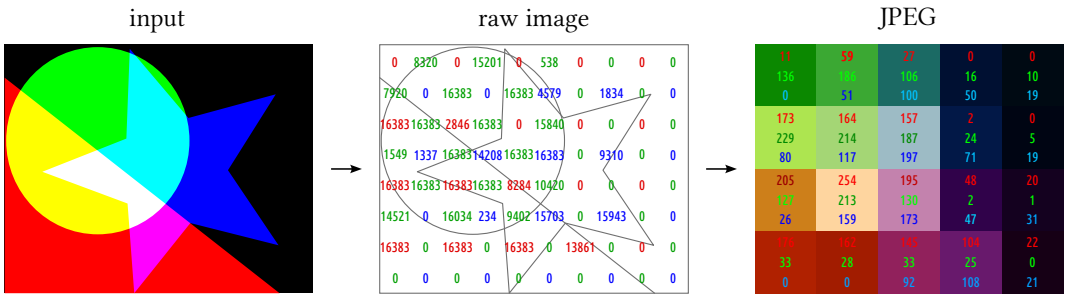


Figure 3: How a camera digitises an image: Left: Actual light shining on the camera. Middle: Grid of colour-specific sensors (subpixels) and 14 bit intensities recorded by them. (Lines are for reference only.) Right: Final pixel image for human consumption storing intensities in 8 bit per colour and pixel.

1.3 Analysis techniques: automated image processing

A major part of the data analysis for this experiment is automatically tracking the growth of bacterial colonies from time-lapse images taken by a robot.

1.3.1 Taking and storing images

The robot can takes images with a consumer-grade digital camera. While it may seem odd to use no dedicated device optimised for this, you have to consider that cameras are the outcome of a considerable amount of industrial development and cheapened by the economy of scale. They therefore out-compete any dedicated device in terms of cost and benefit.

A modern digital camera has a sensor array that can be subdeivided into groups of four colour-sensitive subpixels. In our case one subpixel is sensitive to red, one to blue, and two are sensitive to green. The light intensity recorded by the subpixel sensors gets converted to 14 bit digital information, i.e., numbers between 0 and $2^{14} - 1 = 16383$. This information constitutes the raw image (see Fig. 3, middle). Our camera allows us to store these raw images.

The raw image often gets compiled into a pixel image such as a JPEG, which stores the intensity of the colours red, green, and blue with 8 bit each per pixel (see Fig. 3, right). Depending on the camera, either four subpixels get mapped onto one pixel or onto four pixels, using interpolation. Fig. 3 illustrates the former strategy, but our camera for this experiment uses the latter[1]. During this process, colour adjustments are made and image compression (such as in a JPEG) may further reduce the information (This explains, e.g., why the green component does not drop to zero in the right-most pixels in Fig. 3).

In comparison to raw images, JPEGs (and similar image formats) are optimised for human visual consumption and thus lack details that are not or barely discernible by the human eye, but we can use in an automated image analysis. In this experiment, we can primarily exploit the higher intensity resolution of raw images, while detailed positional information is not relevant to us.

[1]Guess which one is more marketable.

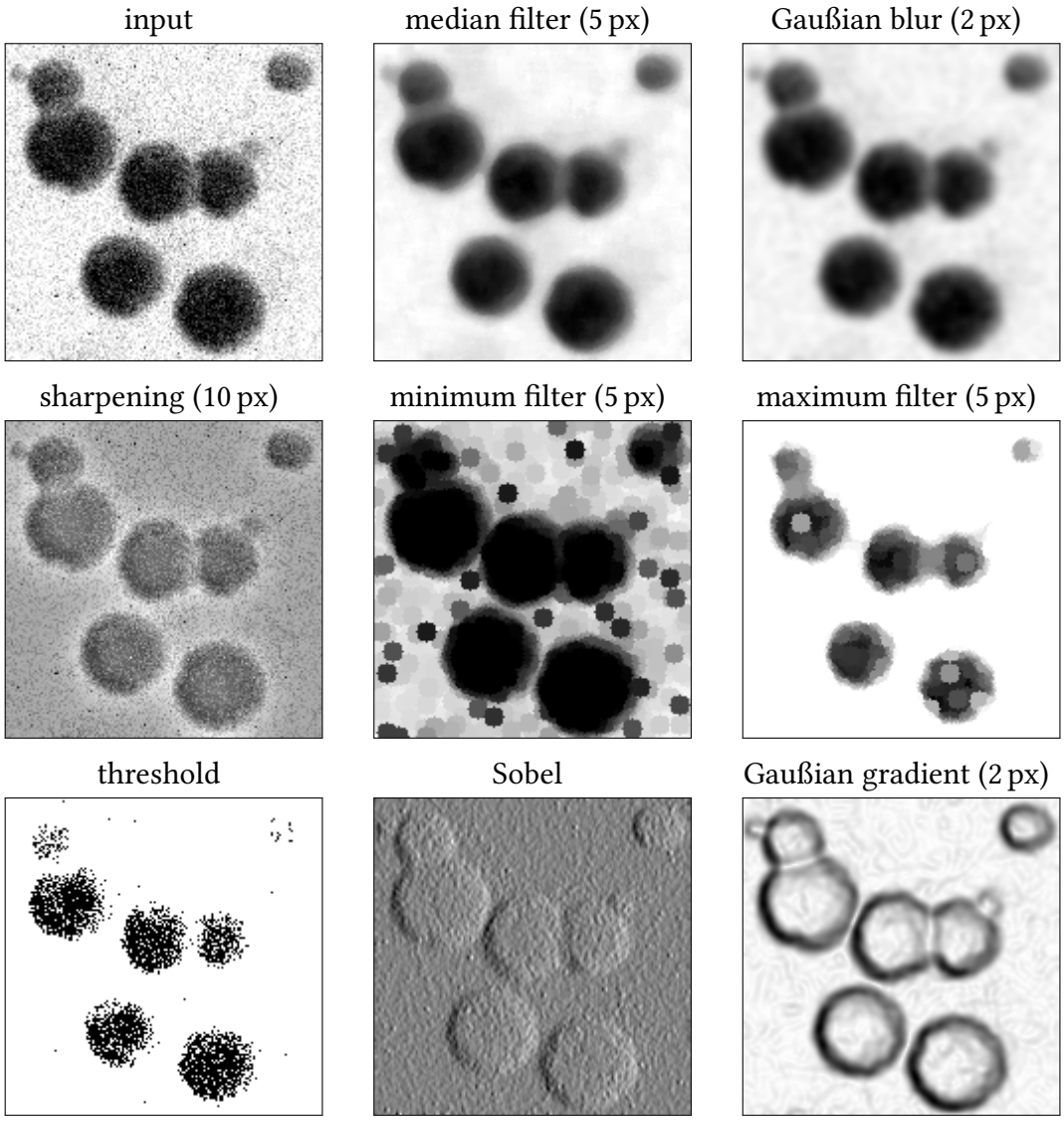


Figure 4: Effects on several filters on a greyscale close-up of some bacterial colonies (top left). The original image is 160 px × 160 px *(and so are all filtered images). If applicable, filters are applied to a circular footprint, and the radius is given in parentheses. (The quality of the input is worse than what you usually get in the experiment.)*

1.3.2 Filters

Broadly speaking an *image filter* replaces the value(s) of a pixel with function of small neighbourhood *(footprint)* around it, often a circle. A simple example is the *median filter,* which replaces each pixel with the median of all pixels in the footprint (Fig. 4, top middle). It smooths the image, removing particularly dark or bright pixels (outliers). In general, filters process an image locally, which is what we need, since bacterial colonies will be local features.

Filters are computationally rather cheap as they are considering a (usually small) footprint in a simple way. Most suitable programming languages feature a library with highly optimised implementation of standard filters. It therefore makes sense to use these as the building blocks of an image-processing pipeline.

Many filters are a convolution of the image – with the intensity being interpreted as a function of the coordinates – with a *kernel* that characterises the filter. For example, to smooth an image, one can use a Gaußian kernel, i.e., two-dimensional normal function (Fig. 4, top right). The support of the kernel corresponds to its footprint and it is computationally beneficial to keep it small. For example, instead of a true Gaußian kernel, we usually use one which is truncated at a radius where its value becomes negligible.

In terms of spacial frequencies, smoothing filters are low-pass filters. For example, an isolated bright pixel corresponds to high spacial frequencies, which a smoothing filter removes. By contrast high-pass filters emphasise small details, which can be perceived as sharpening the image (Fig. 4, middle left).

Some further filters of potential interest for this experiment are:

- The *minimum and maximum filter* set each pixel to the minimum or maximum of the filter's footprint, respectively (Fig. 4, middle center and right).
- *Threshold filters* simply set every pixel above a certain intensity as black and every other pixel as white (Fig. 4, bottom left).
- *Edge-detection* filters aim at finding borders between objects. They usually return some sort of gradient as edges are characterised by a sudden change of colour. The *Sobel filter* employs a simple discrete gradient determined from a footprint of 3 px×3 px only, but is very efficient to compute (Fig. 4, bottom centre). Modern edge-detection filters take into account more pixels and smoothing, e.g., the Gaußian gradient filter (Fig. 4, bottom right).

Many filters are not very useful on their own, but powerful in combination. For example, the minimum and maximum filters in Fig. 4 are sensitive to single dark or bright pixels, respectively, which is usually not a desirable outcome. However, smoothing can eliminate such bad pixels beforehand.

1.3.3 Counting colonies

A plethora of software solutions for colony counting have been published [6]. However, their applicability depends on the quality of images and information to be extracted. For example, distinguishing colonies of different microbes by their colony morphology is considerably more challenging than simply counting colonies. Therefore, it can still be preferable to develop a new analysis pipeline tailored to the data and problem at hand.

In this experiment, we use image analysis to determine the wake-up time of colonies. To this end, we use time-lapse images, which most existing solutions are not made for. Moreover, instead of quantifying characteristics such as the growth rate, we have growth curves which are ideally identical except for a temporal shift (lag), which is what we are interested in. Therefore (and for didactic reasons), you will write a fresh analysis pipeline.

The basic idea (we suggest to use) to detect colonies is similar to the one used by Ref. 7: Colony centres should be sufficiently intense local extrema of a smoothed late image. We can then monitor the intensity in the colony's position over time to determine its growth curve and from that its wake-up rate.

2 Goals and Tasks

The goal of this experiment is to observe and quantify the persister dynamics of *Klebsiella pneumoniae*, specifically:

1. Treat infected human-bladder epithelial cells with the antibiotic ciprofloxacin and record the killing curve by plating lysed host cells for colony forming units to obtain the killing and transition rate.
2. Take time-lapse images of bacterial colonies without and after treatment. Use an automated image analysis to determine the wake-up rates.

3 Experimental Procedure

*What follows is a full procedure that also includes summaries of preparations and similar that **will be done for you**, in particular everything done on Monday, Tuesday, and Friday.*

3.1 Monday: preparations

1. Seed a cell culture of epithelial cells that is going to be infected later and incubate at 37℃.
2. Inoculate a tube of lysogeny broth (a liquid growth medium) with *Klebsiella pneumoniae* and incubate at 37℃.

3.2 Tuesday: infection

1. Prepare PBS medium with bacteria at $OD_{600} \approx 0.2$ to obtain a multiplicity of infection (MOI) of about 100, i.e., 100 bacteria per cell.
2. Apply 50 µℓ to each cell-culture well.
3. Centrifuge to synchronize infection.
4. Incubate at 37℃ for two hours.
5. Wash the cells: Remove the medium, add fresh medium, and remove it again.
6. Add medium with gentamicin and incubate for another two hours. Gentamicin is an antibiotic other than the one we are interested in (ciprofloxacin). It does not enter the cells and serves to kill only extracellular bacteria, which would confound any investigation of intracellular bacteria.
7. Remove the medium, wash the cells, and add medium with gentamicin once more.
8. Incubate for 24 hours post infection.

After this, we ideally have bacteria inside our cells and only there.

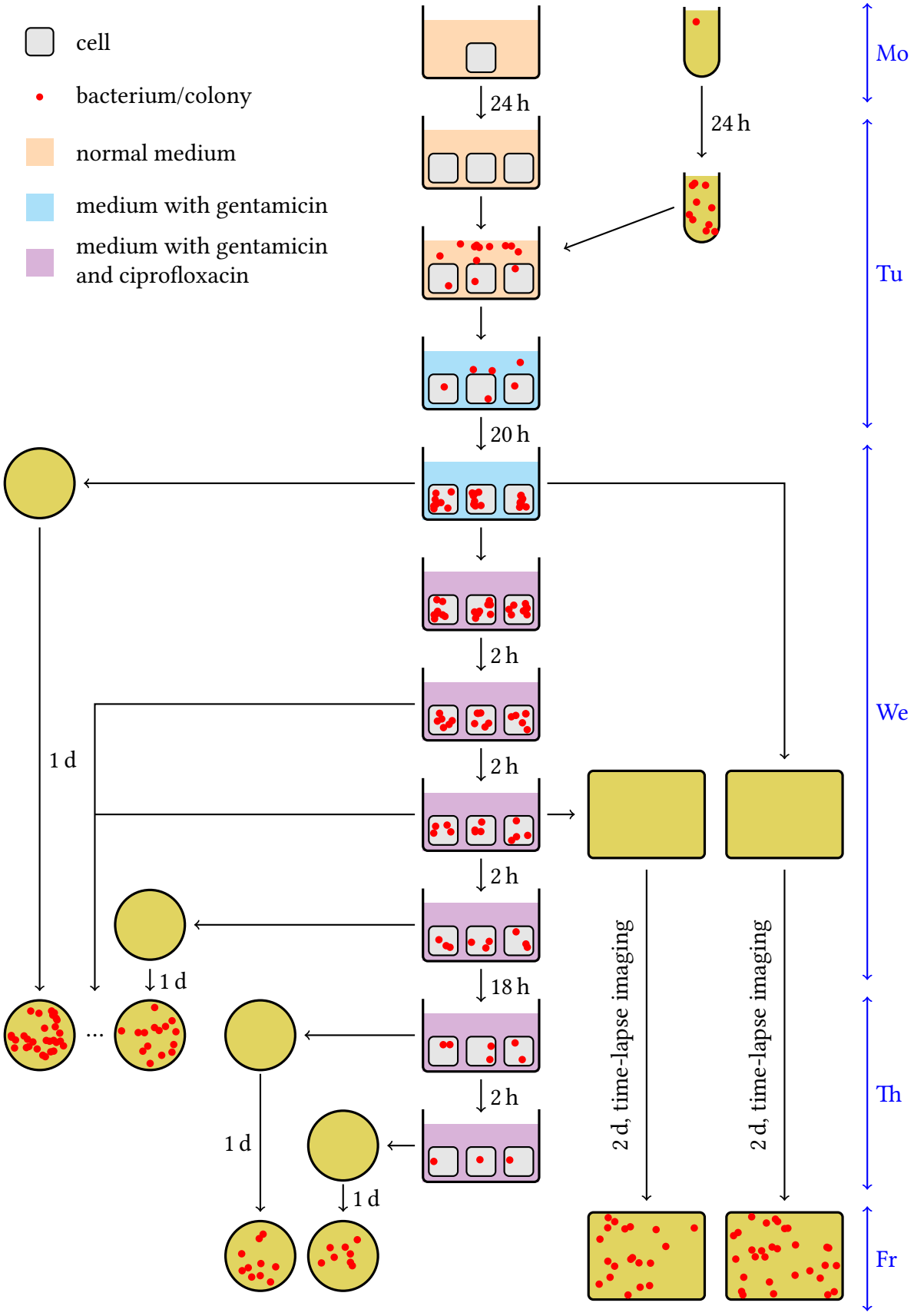


Figure 5: Overview of the experiments

3.3 Wednesday: antibiotic treatment and plating

3.3.1 Antibiotic treatment

We now treat the infected cells with ciprofloxacin for different durations, namely roughly 0 h (control), 2 h, 4 h, 6 h, 24 h, and 26 h. To evaluate the effect of the treatment, we liberate the bacteria by lysing the cells and then plate them onto agar.

1. Prepare medium (RPMI1640 with 10% FBS) with ciprofloxacin at a concentration of 4 µg/µℓ.
2. Aspirate medium from the wells.

At this point, you need to distinguish between the treated samples and the control sample, which gets lysed immediately. For the treatment samples:

3. Add 800 µℓ of medium with ciprofloxacin. Don't directly pipette onto the cells, lest they get washed off. Instead target the wall when pipetting.
4. Incubate at 37℃ for the desired time.

Once a sample is ready for lysis:

5. Write down the time.
6. Wash the samples three times with 1 mℓ of PBS.
7. Add 1 mℓ of 0.1% Triton X 100 to lyse the cells.
8. Incubate for 20 min at 37℃.
9. Scratch the bottom of the well thoroughly with a pipette (to scrape off any adherent cells) and pipette-mix the lysate.
10. Transfer the lysate to a tube for further processing.

3.3.2 Plating the bacteria

- At every time point: Plate out the bacteria on a Petri dish using the spiral dispenser. Incubate the plates at 37℃.
- At 0 h: On omnitrays (larger, rectangular trays), plate out 100 µℓ of bacteria solution as is (1×), in three-fold dilution (⅓×), and ten-fold dilution (1/10×). Use beads to distribute the bacteria. Store these plates at 4℃ until you remove the plates from the following step. This maximises the comparability of the results.
- At 4 h: Plate out the bacteria on omnitrays as above, but **additionally** in three-fold concentration (3×), and ten-fold concentration (10×). To concentrate the bacteria, centrifuge them to the bottom, carefully pipette out the appropriate amount of medium, and then vortex the result to mix it again.

3.3.3 Time-lapse imaging

After the 6 h step:

1. Transport the omnitray plates to the Centre of Molecular Biosciences (CoMB).
2. Start a time-lapse imaging series using the Biomatrix BM3-BC robot, taking one image of each plate every twenty minutes. (The robot is located in a room that is kept at 30℃.)

The robot can store raw images or JPGs. As elaborated above raw images contain more information, but they are slightly more difficult to handle and larger in file size. Due to parallel experiments, you may be forced to use either format.

3.4 Thursday: last time points and first results

Initiate the last measurements at roughly 24 h and 26 h by plating as described above.

In between, evaluate the Petri dishes you plated out on Wednesday to deduce the concentration of bacteria in the solution you plated out.

Afterwards, go to the CoMB to obtain the first time-lapse imaging data and start the setting up the image analysis under guidance (see below).

3.5 Friday: more results

Evaluate the Petri dishes you plated out on Thursday and obtain the complete data from the time-lapse imaging.

4 Data Analysis

Generally, use logarithmic scales where appropriate.

4.1 The killing curve

1. For each time point, estimate the fraction of surviving bacteria and plot it with respect to time.

2. Adopt Eq. 1 assuming:

   - There are no persisters at the beginning of the treatment.
   - Bacteria (and persisters) do not grow.

   To obtain a time evolution for this model for given parameters, you can solve it analytically or numerically – your choice.

3. Fit this model to your killing curve and determine r_{kill} and r_{sleep}. Compare fitting in the linear and logarithmic domain. Which one is better and why?

4. Can you find another set of assumptions that explains the data comparably well?

4.2 Wake-up rates

For this part, you will analyse the time-lapse images. In general, try and explore every part of the process with a some representative pictures or plates until you can write a satisfactory general function.

You can use any programming language you want, but we strongly recommend that you make use of an image-processing library, instead of writing fundamental functions yourself. In the following, we will hint at useful existing functions in Python, but analogues in other reasonable programming languages should be straightforward to find.

Document all steps of the process with exemplary images when applicable. In general, we strongly recommend that you frequently visualise your intermediate results to understand what is happening; have a handy function for this.

Finally, we only give you the minimal procedure required to obtain decent results. Almost any part of the process has clear room for improvement and you are welcome to think about and use more sophisticated procedures.

1. Extract the timestamps from the file names (e.g., using `datetime.strptime`).

2. Load the images. You can use `matplotlib.pyplot.imread` for JPGs and `rawpy` for raw images.

   Be aware that your memory might not be able to hold all the images. This should not be a big problem as you need to have at most two images loaded simultaneously for the following. A big part of the following analysis is best applied image by image instead of step by step.

3. Select an appropriate image as a background. Usually this is the first image, but carefully check that it's not unsuitable due to condensation or similar.

4. Subtract the background from all subsequent images of the same plate. To improve your subtraction result, it may be worthwhile to apply a slight blur beforehand. From now on, you can work with intensities, i.e., use a single value per pixel (some norm of the distance) instead of three or four.

   Raw images have two green subpixels per pixel. You can first average these or treat them as separate colours (although they actually capture the same colour).

   Your images are likely arrays of unsigned integers. That's usually a good thing, but requires you to avoid integer underflows when subtracting (e.g., for 8-bit unsigned integers, $23 - 42 = 237$). One way to handle this is `numpy.subtract` with the right arguments.

5. Crop the images appropriately, such that all border effects are excluded.

6. Determine a radius R you use for analysing individual colonies. The respective area should be smaller than a full grown colony and instead capture an area in which a colony first emerges.

7. Determine colony centres using a late image (where all colonies of interest are clearly visible):

   (a) Smoothen the image (e.g., with `scipy.ndimage.gaussian_filter`) using a kernel width of about ten pixels.

   (b) Determine local maxima, e.g., by checking where the result of the maximum filter (`scipy.ndimage.maximum_filter`) equals the local pixel.

   (c) Do not consider a maximum when a more intense maximum is closer than $2R$.

   (d) Exclude maxima whose intensity is too low.

   (e) Exclude maxima that are closer than R to the border of the image.

8. For each colony (centre), obtain the temporal evolution of the intensity, i.e., within the radius R. You may speed this up by storing the relevant array indices once per maximum.

9. For each colony, determine the wake-up time with an appropriate intensity threshold. You can filter out most falsely detected colonies at this step, as they never wake up or similar.

10. Plot the ECCDF (empirical complementary cumulative distribution function) of the wake-up times, i.e., the fraction of colonies that are asleep at any given time.

11. For each plate, determine the wake-up rates with an appropriate fit. Compare fitting an exponential function in linear space and fitting a linear function in logarithmic space.

12. If all goes well, you should see two regimes for the untreated case: Can you explain these and why you don't see these in the treated case?

13. For each treatment duration (0 h and 4 h), plot the wake-up rates in dependence of the number of detected colonies on the plate. This should give you an insight into the systematic overestimate caused by early risers oppressing late bloomers.

14. Extrapolate the true wake-up rate.

References

[1] N. Q. Balaban, S. Helaine, K. Lewis, et al.: "Definitions and guidelines for research on antibiotic persistence". In: *Nature Reviews Microbiology* 17.7 (2019), pp. 441–448. DOI: 10.1038/s41579-019-0196-3.

[2] N. Q. Balaban, J. Merrin, R. Chait, et al.: "Bacterial persistence as a phenotypic switch". In: *Science* 305.5690 (2004), pp. 1622–1625. DOI: 10.1126/science.1099390.

[3] E. Kussell and S. Leibler: "Phenotypic Diversity, Population Growth, and Information in Fluctuating Environments". In: *Science* 309.5743 (2005), pp. 2075–2078. DOI: 10.1126/science.1114383.

[4] M. R. Warren, H. Sun, Y. Yan, et al.: "Spatiotemporal establishment of dense bacterial colonies growing on hard agar". In: *eLife* 8 (2019), e41093. DOI: 10.7554/eLife.41093.

[5] I. Levin-Reisman, O. Gefen, O. Fridman, et al.: "Automated imaging with ScanLag reveals previously undetectable bacterial growth phenotypes". In: *Nature Methods* 7.9 (2010), pp. 737–739. DOI: 10.1038/nmeth.1485.

[6] J. Zhang, C. Li, M. M. Rahaman, et al.: "A comprehensive review of image analysis methods for microorganism counting: from classical image processing to deep learning approaches". In: *Artificial Intelligence Review* 55 (2022), pp. 2875–2944. DOI: 10.1007/s10462-021-10082-4.

[7] M. L. Clarke, R. L. Burton, A. N. Hill, et al.: "Low-cost, high-throughput, automated counting of bacterial colonies". In: *Cytometry Part A* 77A.8 (2010), pp. 790–797. DOI: 10.1002/cyto.a.20864.

References 2 and 3 are particularly relevant for the exam.