Contents

Errors arising from spatial inhomogeneity of cells on slides.

Our measurements were performed on large coverslips (24X60 mm) that are approximately 50% confluent with cells (Figure 1).

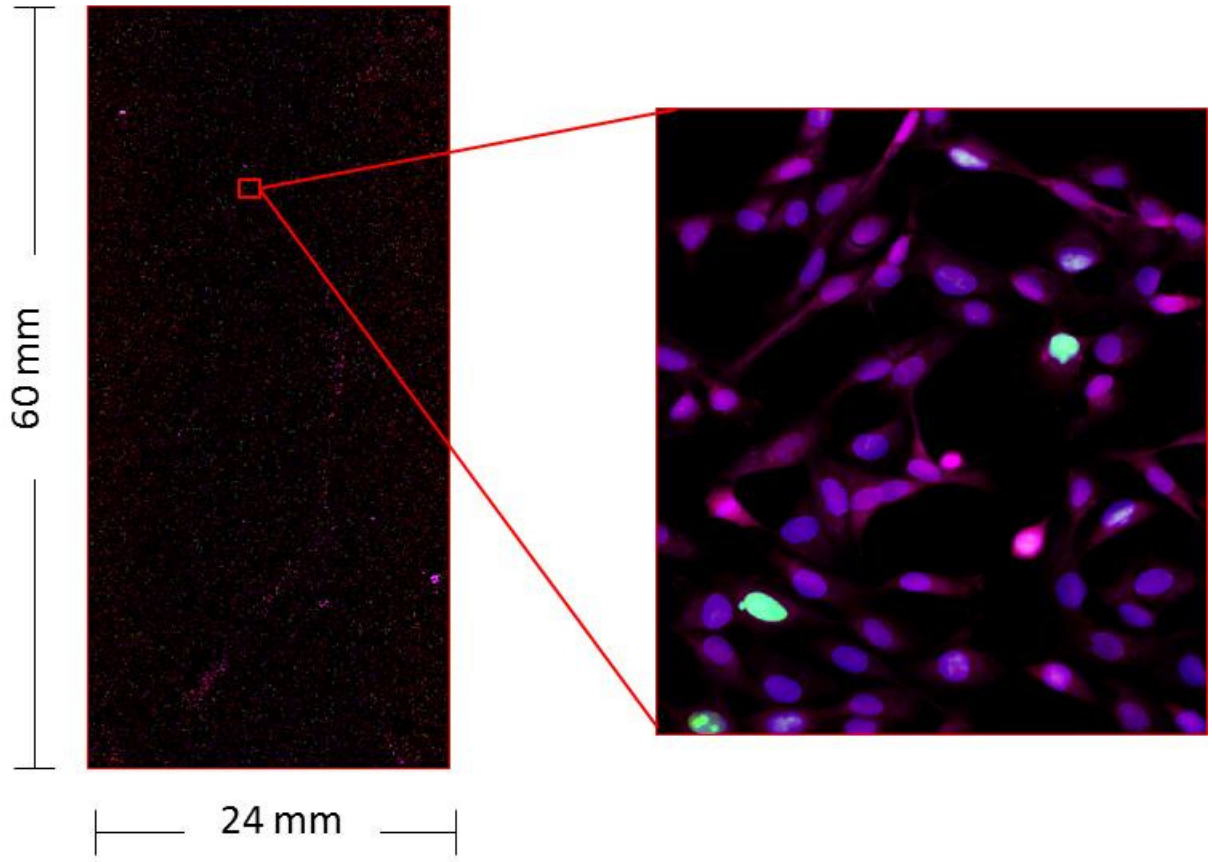


Figure 1: HeLa cells fixed on a 24 X 60mm coverslip. Cells are routinely imaged for four different fluorophores, DAPI (DNA), mAZ-hGem (Geminin degron), Alexa-647 SE (protein mass) and a fourth fluorophore related to signaling (e.g. cMyc).

An assumption of our experimental system is that measured properties of a cell are not influenced by its spatial coordinates on the coverslips. Otherwise, the coordinates of a cell would constitute a confounding factor that may influence statistics. Such a possibility of spatial clustering of cells with similar properties could, for example, arise if certain regions on the coverslip are highly dense (confluent) to an extent that affects cell cycle (e.g. contact inhibition). To test for local spatial inhomogeneities in our measurements we systematically plotted, for each of the measured coverslips, the coordinates of cells and used a color scheme to show the basic measured properties (cell size, DNA, and Geminin) (Figure 2). This analysis demonstrated that while in the majority of coverslips cells are homogenously distributed (Figure 2, A-C), some coverslips contained significant spatial in homogeneities (Figure 2, D-F). To

quantitatively estimate the extent of spatial homogeneity we plotted the average cell size, DNA and Geminin levels as a function of both dimensions (Figure 2, G).

To avoid artifacts arising from spatial inhomogeneities, all coverslips with local clustering of cells with similar properties (such as shown in Figure 2 D-F) were excluded from further analyses.

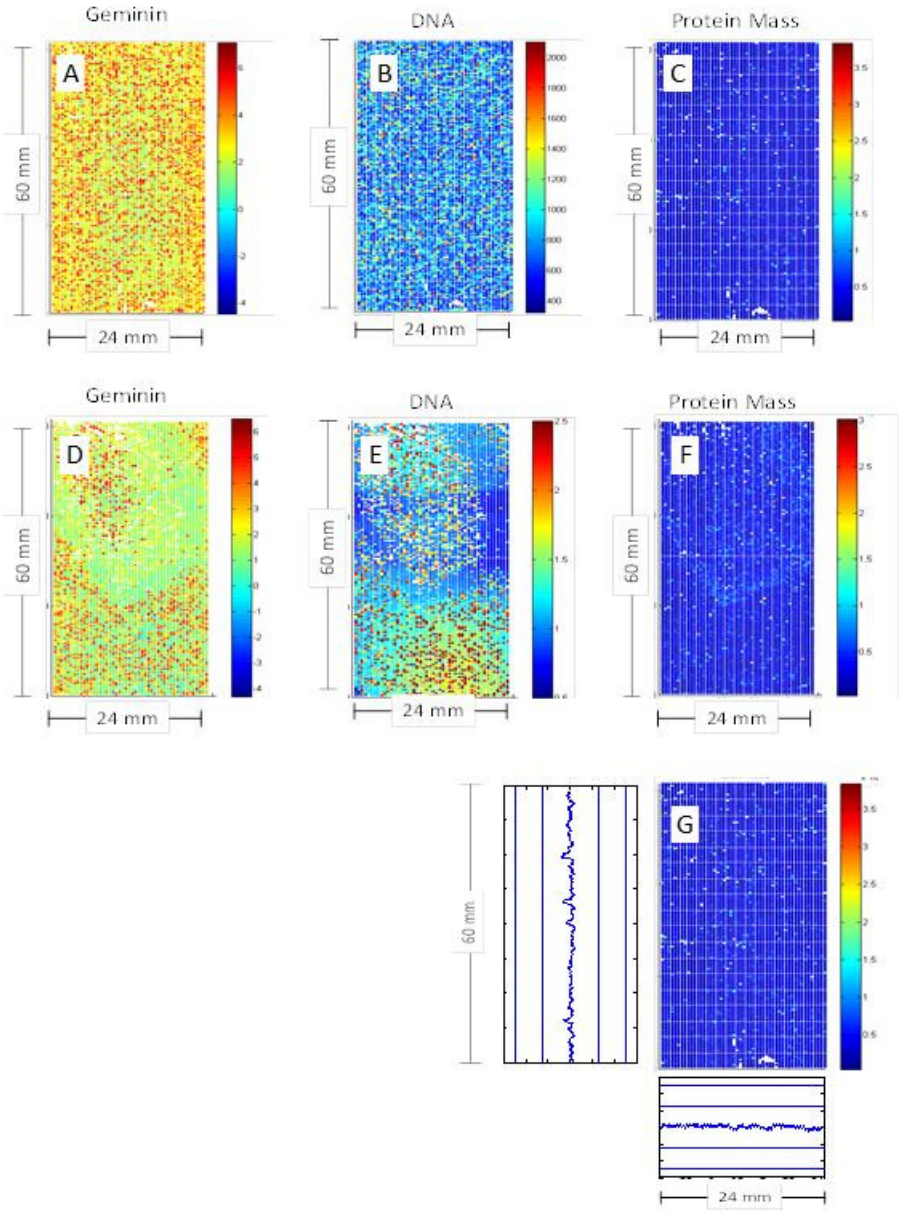


Figure 2: Spatial distribution of cells in coverslips. The location of each cell (HeLa) on 3 example coverslips is plotted and color coded for the cells' level of Geminin, DNA and size. Slide 1 (A-C) exemplifies a homogeneous distribution of cells with respect to all measured properties. Slide 2 (D-F) exemplifies nonhomogenous

distributions: one can see a region of cells that have likely exited cell cycle, having low levels of both Geminin and DNA. Such slides were excluded from analyses in our study. To quantify the extent of spatial homogeneity on a slide we plotted the level of DNA, Geminin and cell size averaged for the horizontal and vertical coordinates (G)

Testing the ergodic assumption, variability between slides and reproducibility of the measured distributions.

A criterion for application of ERA is that the distribution of any of the measured variables does not change with time. We tested this criterion.

To test the first, multiple coverslips were placed on a single 15cm tissue culture dish and seeded with cells. After letting cells settle for 24 hours, we collected coverslips and at different times where time t=0 hrs was defined as 24 hours after plating. Each collected coverslip was fixed and imaged for cell cycle stage based on DAPI (DNA), mAG-hGem (Geminin) and Alexa647-SE (protein mass). Figure 3 shows a quantile-quantile plot comparing distributions of Geminin and DNA between each time point to time t=0 (24 hours after plating). Results are shown on HeLa cells but similar measurements were performed on all four cell lines obtaining qualitatively identical results. Figure 7 shows that trends calculated by ERA from coverslips collected at different time points remain unchanged.

Distributions from coverslips collected at different time points from a single proliferating population.

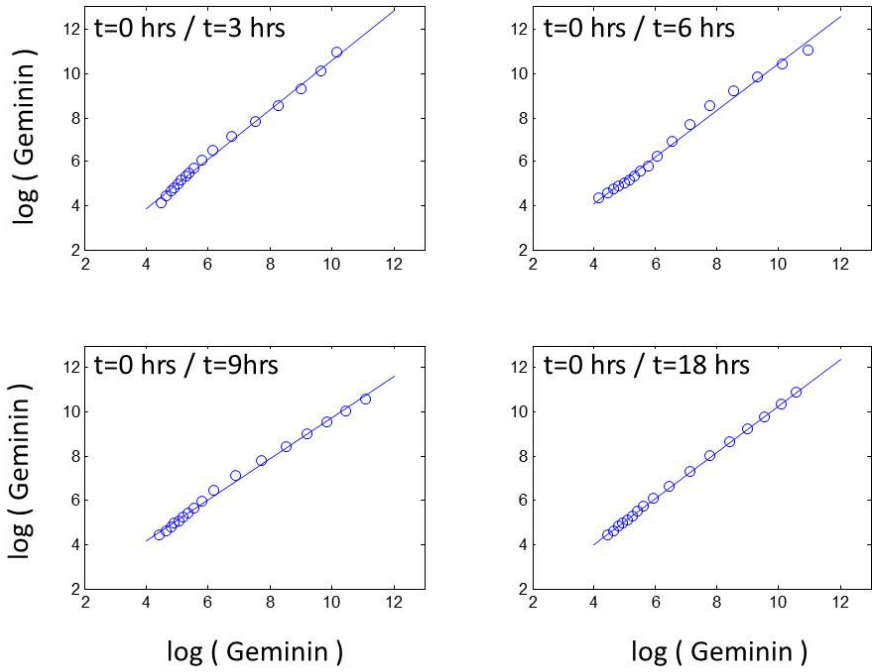


Figure 3: A quantile/quantile (Q-Q) plot comparing distributions of Geminin from slides collected from the same proliferating culture at different times. The coverslip for t=0 hrs was collected 24 hours after cells were plated; the coverslip for t=3 was collected 27 hours after plating, and so on. Each of the boxed plots compares the results for a different time point to the results for t=0.

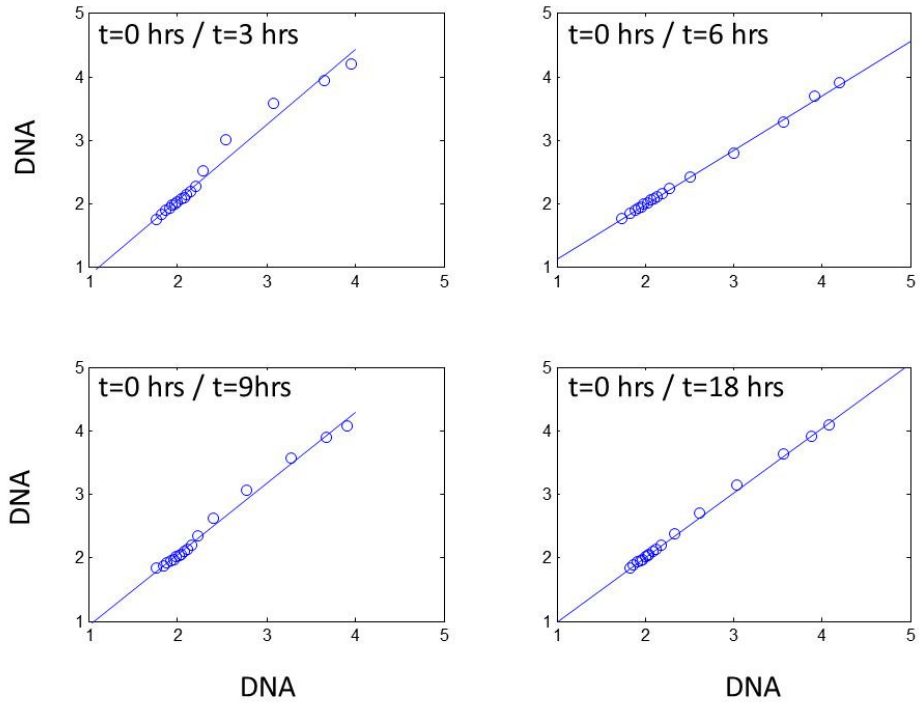


Figure 4: Comparing distributions of DNA levels taken from a single population at different times as in Figure 3.

Distributions from coverslips collected from experiment repetitions.

Variation in experimental conditions could theoretically alter the distributions in cell cycle stage and cell size. To test this we asked: to what extent do measured distributions of cell size and cell cycle stage vary in experiment repetitions? To test this we performed pair-wise comparisons of distributions of Geminin and DNA for all coverslips used in analyses in our study (Figure 5). As above, we used quantile-quantile plots as means of comparison. The main advantage of this method is that it is uninfluenced by scaling that occurs, for example, when microscope lamp is brighter or dimmer. Due to the large sample size used in our study, methods like the Kolmogorov-Smirnoff test are inadequate, as differences on the order of a few percent (which are typical for biological measurements) would be identified as statistically significant, despite having no biological significance. As a more biologically relevant test for reproducibility, we also compared whether trends calculated by ERA from measurements of experiment repetitions and from a time course experiment (figures 6 and 7) were similar.

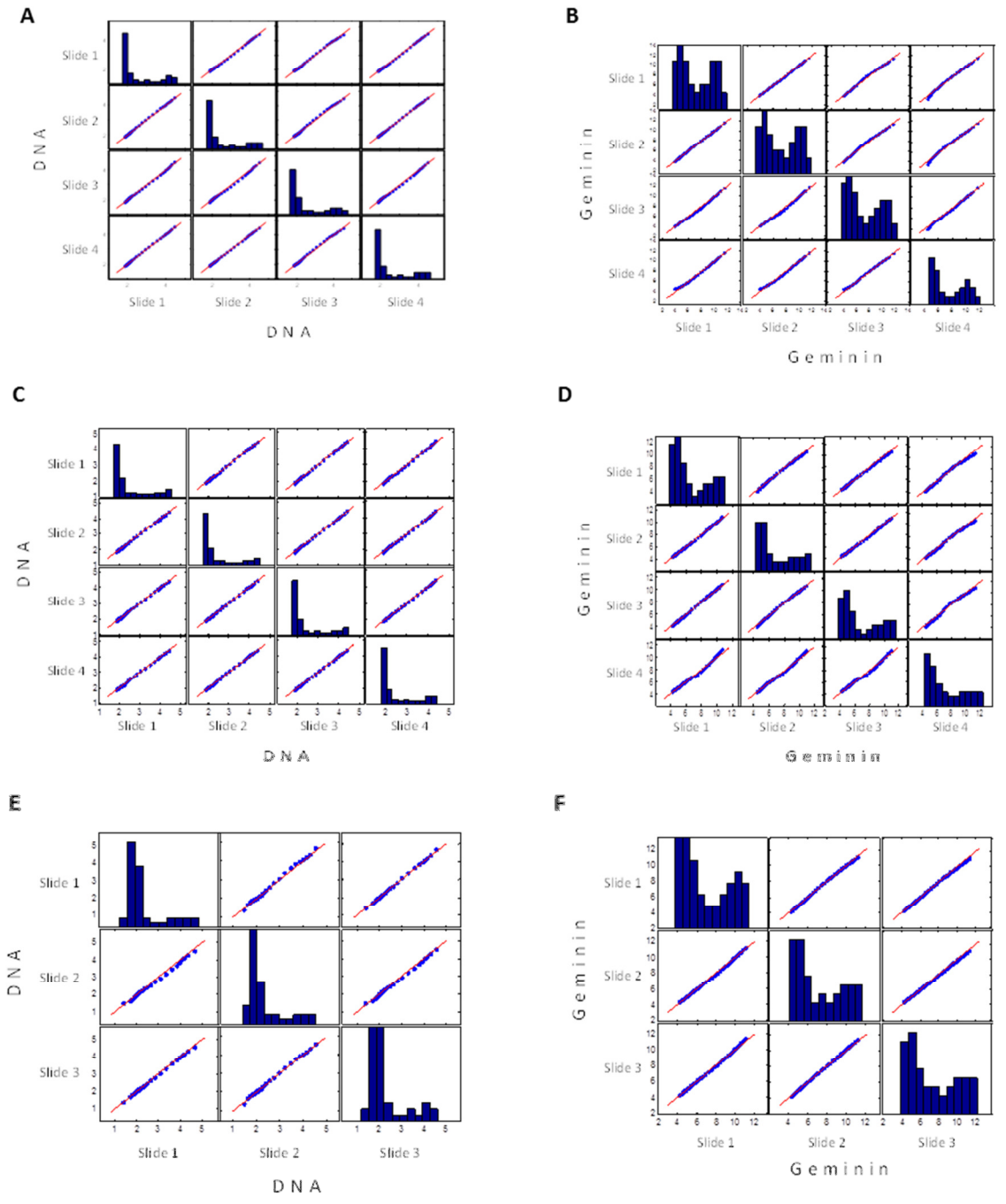
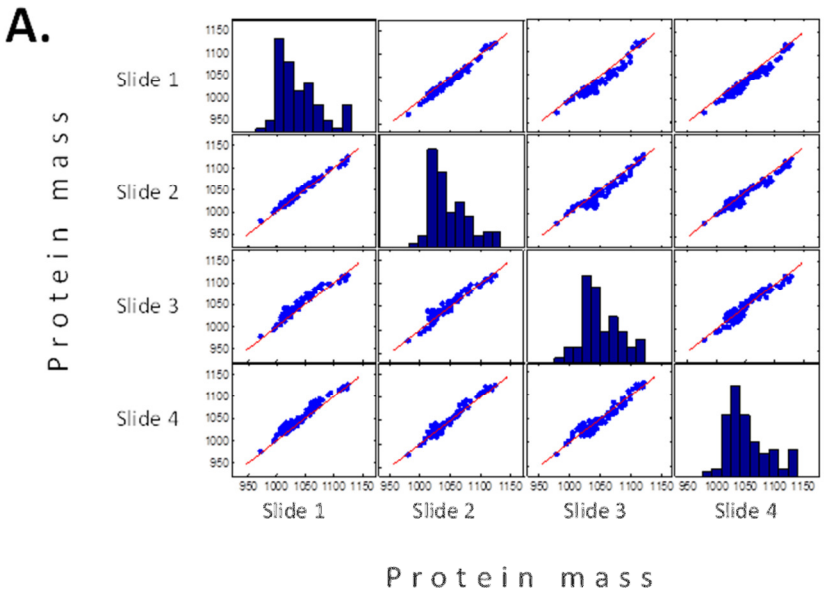
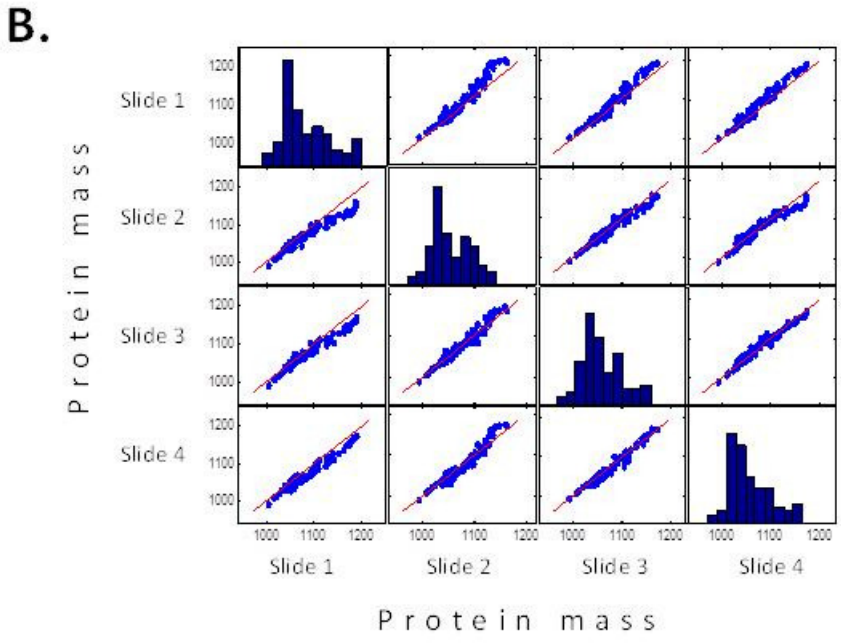


Figure 5: Quantile-Quantile pairwise comparisons of the distributions of DNA and Geminin between all coverslips used in our study. Comparisons are separated based on cell lines (**A-B**, RPE1; **C-D**, HT1080; **E-F**, HeLa). Diagonal plots show the marginal distributions of DNA and Geminin in each coverslip.

Testing the ergodic assumption on distributions of cell size and the reproducibility of the protein mass measurements.

To test reproducibility, we asked whether the trend of protein mass as a function of the cell cycle trajectory, ℓ, is similar in different experiments. To test this we calculated, for each of the coverslips used in the study, the distribution of protein mass along the cell cycle trajectory, ℓ. We then plotted the pairwise comparison of this distribution for all pairs of coverslips in HT1080 cells (Figure 6, **A**), RPE1 cells (Figure 6, **B**) and HeLa cells (Figure 6, **C**).





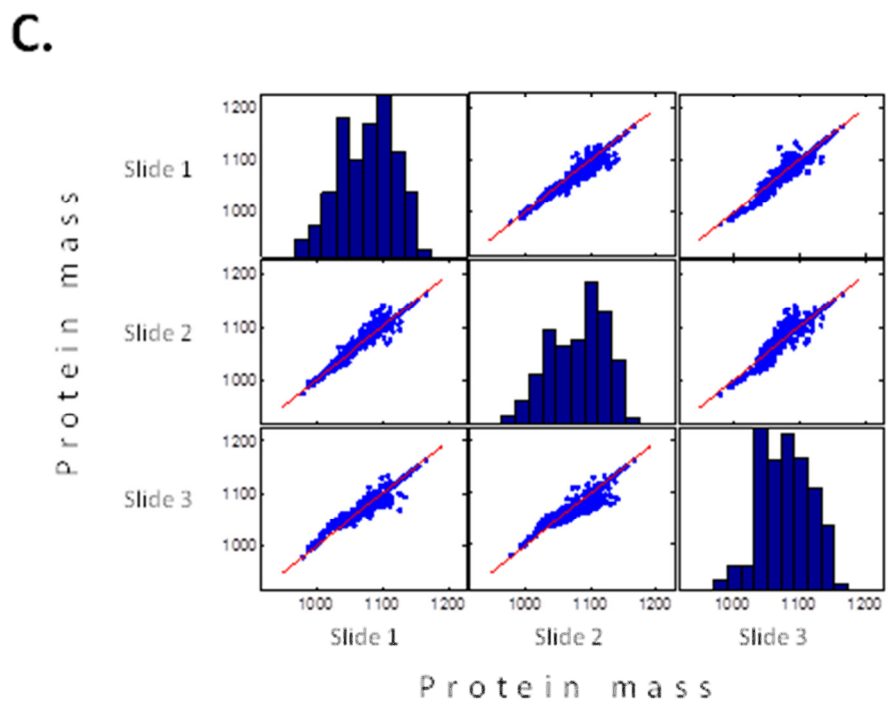
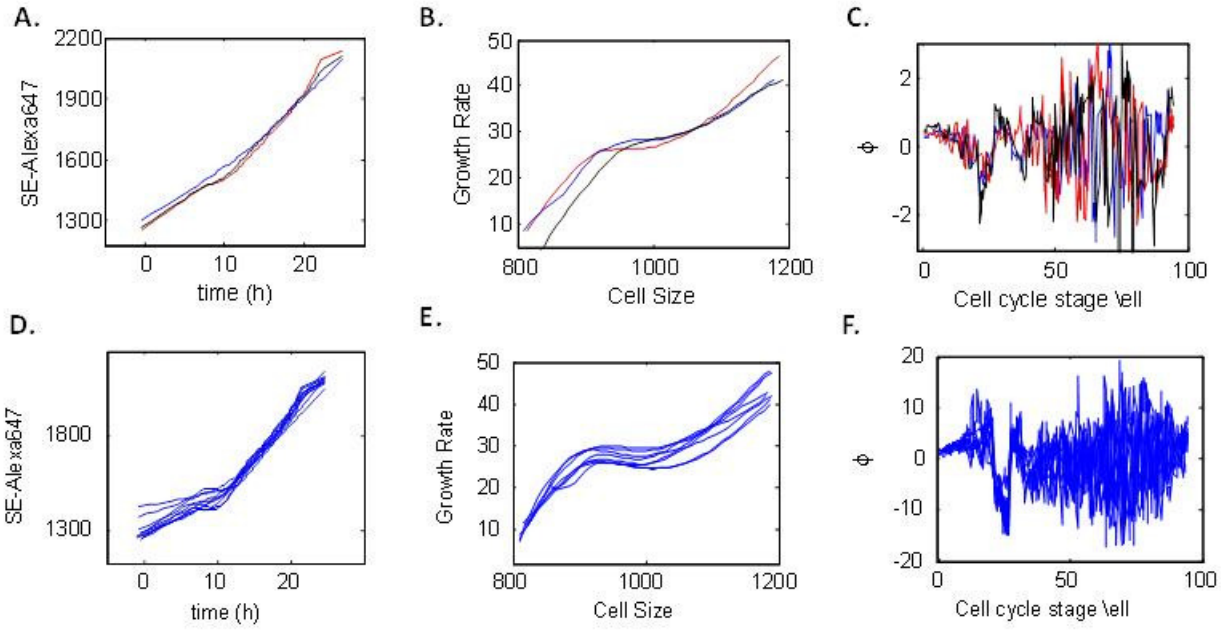
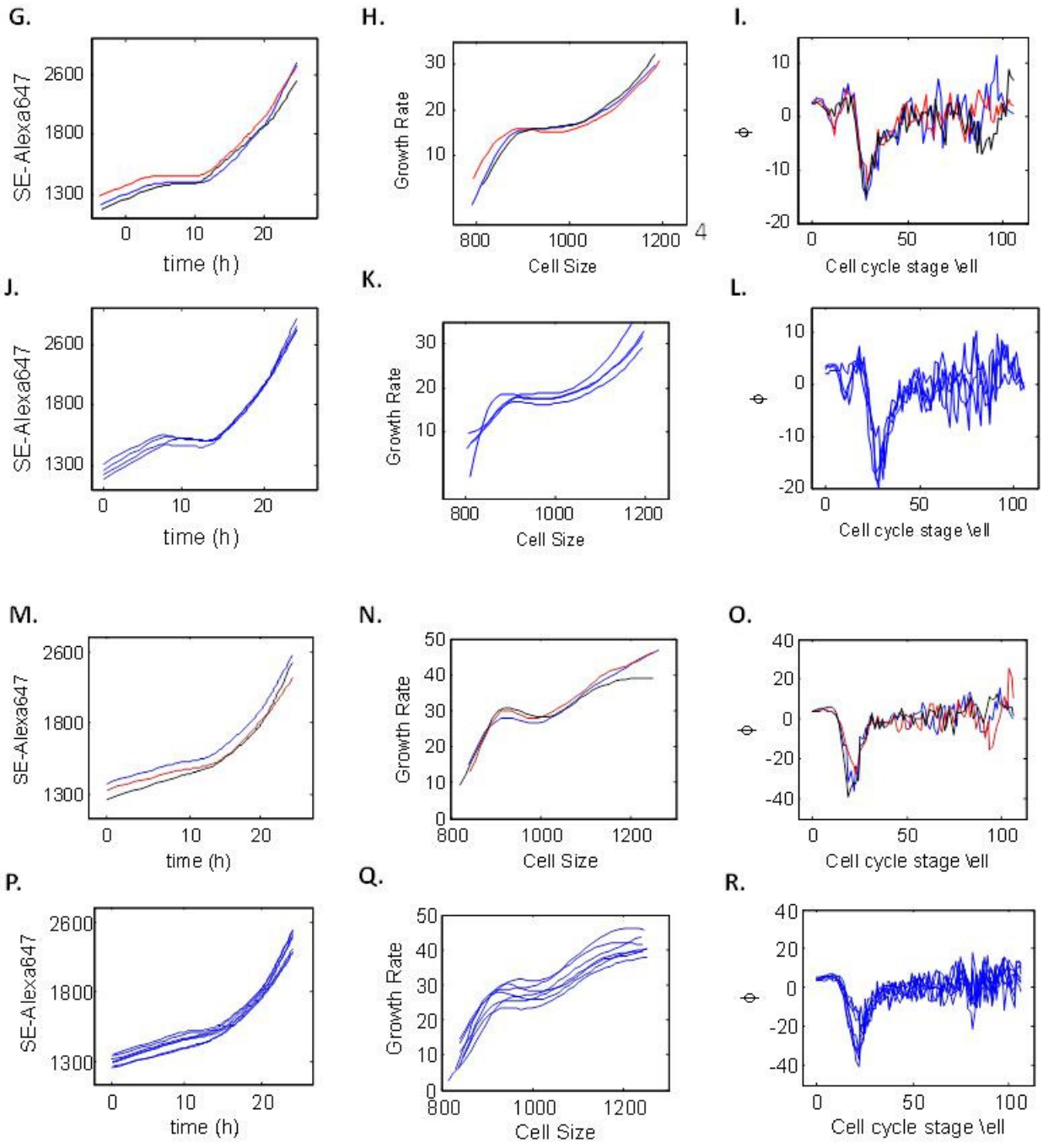


Figure 6: Comparison of the protein mass as a function of the cell cycle trajectory, ℓ, obtained from different experiments in HT1080 cells (**A**), RPE1 cells (**B**) and HeLa cells (**C**).




G.
SE-Alexa647
2600
1800
1300
0
10
20
time (h)
H.
Growth Rate
30
20
10
800
1000
1200
Cell Size
I.
ϕ
10
0
-10
-20
0
50
100
Cell cycle stage \ell
J.
SE-Alexa647
2600
1800
1300
0
10
20
time (h)
K.
Growth Rate
30
20
10
800
1000
1200
Cell Size
L.
ϕ
10
0
-10
-20
0
50
100
Cell cycle stage \ell
M.
SE-Alexa647
2600
1800
1300
0
10
20
time (h)
N.
Growth Rate
50
40
30
20
10
0
800
1000
1200
Cell Size
O.
ϕ
40
20
0
-20
-40
0
50
100
Cell cycle stage \ell
P.
SE-Alexa647
2600
1800
1300
0
10
20
time (h)
Q.
Growth Rate
50
40
30
20
10
0
800
1000
1200
Cell Size
R.
ϕ
40
20
0
-20
-40
0
50
100
Cell cycle stage \ell

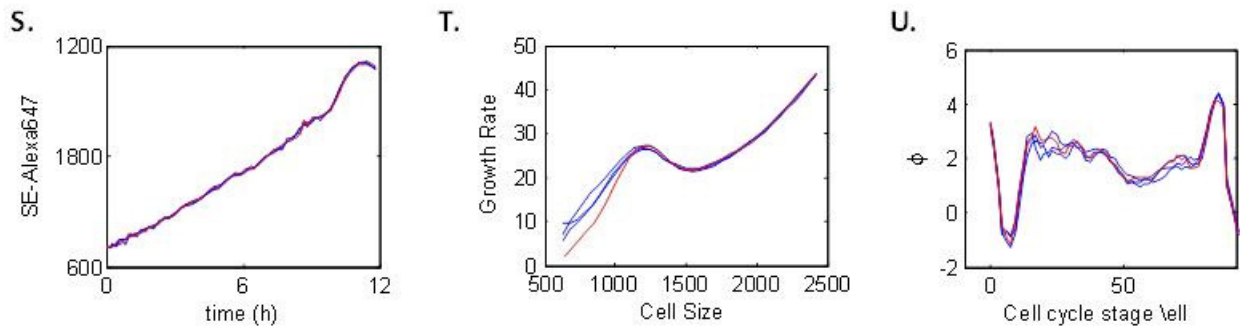


Figure 7: Growth curves of cells collected from a single proliferating population at 24 (black), 30 (red) and 36 (blue) hours after plating are compared for HeLa cells (A-C), HT1080 (G-I), RPE1 (M-O) and L1210 (S-U). Also shown are comparisons of growth curves from 8 different independent starting batches (experiment repetitions) of HeLa cells (D-F), 4 different starting batches of HT1080 (J-L) and 7 different starting bathes of RPE1 cells (P-R). Curves were calculated by ERA and procedures from the main article text to describe cell size as a function of time (first column), growth rate as a function of cell size (second column) and the feedback spectra (third column). Each curve represents data calculated from a single coverslip.

Parameterization of cell cycle stage - calculation of ℓ

To obtain a unique solution for the ERA equation (Eq. 2, main text) we reduced the dimensionality of our DNA/Geminin measurement into a single variable, ℓ, which represents a continuous measure of cell cycle stage (Figure 8). The following description and MATLAB code provides a simple method of doing this; for more detail about another approach that was also used, see Supplementary material #1.

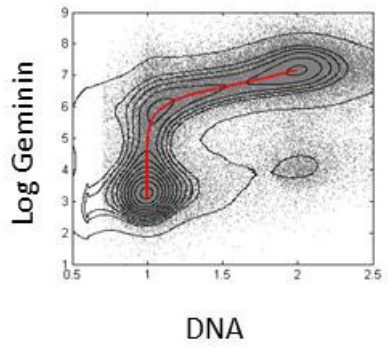


Figure 8: Parameterization of cell cycle. A scatter plot of DNA vs Geminin in HeLa cells together with the probability density function (black contour lines) calculated by the Parzen method [1,2] with a Gaussian kernel. Also shown (red) is the curve passing through the density ridge. This curve represents the path of an "average" cell and is used to parameterize cell cycle stage and reduce the 2D DNA/Geminin representation into a 1D curve. We use the notation, ℓ to describe a cells' position on the red curve.

To calculate the curve ℓ, we applied the following recursive search algorithm: The first point on ℓ, ℓ_1, is arbitrarily chosen as the global density maximum at G_1 (Figure 9). This choice is convenient as this maxima is simple to identify. To find the second point, ℓ_2 we calculate the probability density on the perimeter of a circle with radius R centered on ℓ_1 (Figure 9). The maximum of this probability density is used to specify the point ℓ_2. This procedure is then repeated recursively to identify any point ℓ_i from ℓ_{i-1}. In other words, to identify the position of any point ℓ_i, we draw a circle around the previous point, ℓ_{i-1} and ask which point on that circle is associated with the largest cell count.

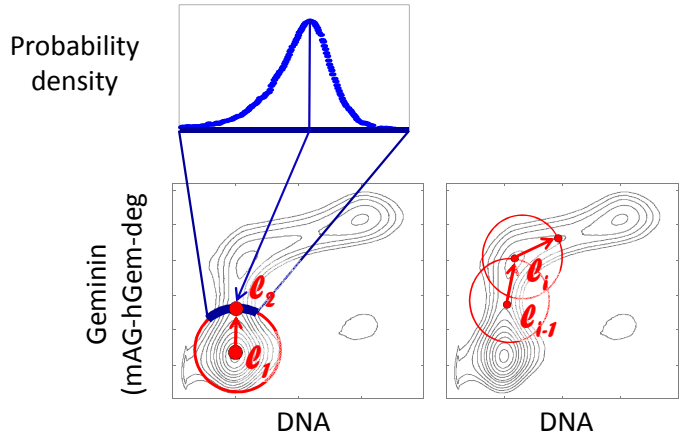


Figure 9: the algorithm for calculating the cell cycle trajectory, ℓ (see text)

Once the curve, ℓ, has been calculated, we use a standardized Euclidian distance to associate each single cell with a discrete point on ℓ (Figure 10). Initially, the cell cycle stage of each single cell is specified by a 2D coordinate system, e.g. the cells' joint levels of Geminin (mAG-hGem-deg) and DNA (DAPI). Based on a standardized Euclidean distance measure, we ask for each cell which is the point on ℓ to which it is closest. The endpoint of this calculation is that the cell cycle stage of each cell is specified with a single 1D variable, ℓ.

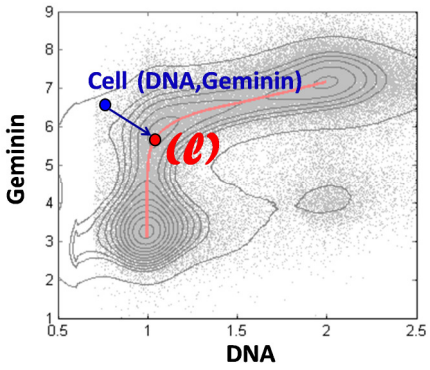


Figure 10: Each cell is associated with the point on ℓ to which it is closest. Distance is standardized Euclidean distance.

Matlab code for generating a loop:

```
function [ell]=GenerateLoop(DNA,Gem)
figure(1)
minDNA=0.7;
maxDNA=2.5;
minGem=3;
maxGem=13;
N=101;
dna_axis=linspace(minDNA,maxDNA,N);
gem_axis=linspace(minGem,maxGem,N);
[discDATA ] = bin_raw([DNA' Gem'],{[dna_axis] [gem_axis]});
f=imfilter(discDATA,fspecial('average',5));
LengthScale=round(0.1*sqrt(prod(size(f))));
imagesc(f)
title('click starting pos and press enter')
[x0, y0] = getpts;
imagesc(f)
title('click ending pos and press enter')
[xe, ye] = getpts;
clf
imagesc(f)
Z=zeros(size(f));
Z(round(x0),round(y0))=1;
Z=imdilate(Z,strel('disk',LengthScale));
I=find(Z>0);
[srt,srti]=sort(f(I),'descend');
[y0,x0] = ind2sub(size(f),I(srti(1)));
[DNA0,Gem0] = SwitchCoords(y0,x0,N,minDNA,maxDNA,minGem,maxGem);
Z=zeros(size(f));
Z(round(xe),round(ye))=1;
Z=imdilate(Z,strel('disk',LengthScale));
I=find(Z>0);
[srt,srti]=sort(f(I),'descend');
[ye,xe] = ind2sub(size(f),I(srti(1)));
[DNAe,Geme] = SwitchCoords(ye,xe,N,minDNA,maxDNA,minGem,maxGem);
hold on
StepSize=1;
[directions,RHO,THETA]=FindAngle(f,[x0 y0],0,1);
[dx,dy]=pol2cart(pi/180*(directions-180),0);
plot(x0+dx,y0+dy,'wo')
ell=[DNA0 Gem0];
[dx,dy]=pol2cart(pi/180*(directions-180),StepSize);
x=x0+dx;
y=y0+dy;
[dnai,gemi] = SwitchCoords(y,x,N,minDNA,maxDNA,minGem,maxGem);
```

```
ell=[ell ; [dnai gemi]];
plot(x,y,'ko')
D=100;
counter=0;
while D>StepSize
    counter=counter+1;
    Previous_direction=directions;
    [directions]=FindAngle(f,[x y],Previous_direction-180,counter);
    [dx,dy]=pol2cart(pi/180*(directions-180),StepSize);
    figure(1)
    plot(x+dx,y+dy,'k.')
    text(x+dx,y+dy+3,num2str(counter),'fontsize',5)
    disp(counter)
    pause(0.05)
    x=x+dx;
    y=y+dy;
    [dnai,gemi] = SwitchCoords(y,x,N,minDNA,maxDNA,minGem,maxGem);

    ell=[ell ; [dnai gemi]];
    D=sqrt((x-xe)^2+(y-ye)^2);

end
[tt,rr]=cart2pol(ell(:,1),ell(:,2));
[l1,l2]=pol2cart(smooth(tt,10,'loess'),rr);
ell=[l1 l2];
    function [directions,RHO,THETA]=FindAngle(f,pos,ExcludedAngle,Ell_i)
        LengthScale=0.1*sqrt(prod(size(f)));
        [X,Y] = meshgrid(1:size(f,1),1:size(f,2));
        [THETA,RHO] = cart2pol(X-pos(1),Y-pos(2));
        THETA=ceil(180*THETA/pi)+180;
        RHO=ceil(RHO);
        for rho=1:max(RHO(:))
            for theta=1:360
                if theta>=351
                    I=find(RHO>rho & RHO<rho+10 & (THETA<theta+10-360 | THETA>theta));
                else
                    I=find(RHO>rho & RHO<rho+10 & THETA>theta & THETA<theta+10);
                end
                M(rho,theta)=mean(f(I));
            end
        end
        IntensityProfile=smooth(nansum(M(1:LengthScale,:)),15);
        IntensityProfile=IntensityProfile(IntensityProfile>0);

        [mn,mni]=min(IntensityProfile);
        ShiftAxis=[mni:length(IntensityProfile) 1:mni];
        ShiftedIntensityProfile=IntensityProfile(ShiftAxis);

        Xvals=1:length(ShiftedIntensityProfile);
        [pks,pksi]=findpeaks(smooth(ShiftedIntensityProfile,10),...
            'MINPEAKDISTANCE',20,'MINPEAKHEIGHT',2*mean(abs(diff(ShiftedIntensityProfile))));
        [srt,srti]=sort(pks,'descend');
        EstimatedDir=ShiftAxis(pksi(srti(1:2)));

        [mx,mxi]=max(min([360-abs( EstimatedDir-ExcludedAngle) ; abs(EstimatedDir-
ExcludedAngle)]));

        EstimatedDir=EstimatedDir(mxi);
        directions=EstimatedDir;
    end

    function [DNAi,Gemi] = SwitchCoords(x,y,N,minDNA,maxDNA,minGem,maxGem)
        DNAi=minDNA*(N-x)/(N-1)+maxDNA*(x-1)/(N-1);
        Gemi=minGem*(N-y)/(N-1)+maxGem*(y-1)/(N-1);
    end
end
```

Calculation and error analysis of the distribution, $f(\ell)$, of cells in cell cycle.

Definition and calculation of f and F

We define $f(\ell)$ as the fraction of cells associated to points on ℓ.

$$f(\ell_i) = \frac{number\ of\ cells\ at\ \ell = \ell_i}{total\ number\ of\ cells} \qquad \text{Eq. S1}$$

Further, let $F(\ell)$ be the cumulative probability distribution describing the frequency of cells that are either at cell cycle stage ℓ_i or at earlier cell cycle stages:

$$F(\ell) = \int_{G_1}^{\ell} f(\phi) d\phi$$

$$F(\ell_n) = \sum_{i=1}^{n} f(\ell_i) \qquad \text{Eq. S2}$$

Errors in f and F

Errors in f and F result from two factors: (1) errors in assigning a correct value of ℓ to individual cells, and (2) sampling errors.

Notation

We will use the notation δX to describe the errors in a variable, X.

Errors in f and F that result from errors in localizing cells to ℓ.

A prerequisite for calculation of the proportion of cells, $f(\ell)$, at each value of ℓ is the correct assignment of a specific cell cycle position, ℓ_i, for each cell. To calculate the extent to which errors propagate from this assignment to the probability functions, $f(\ell)$ and $F(\ell)$, we used a randomization method akin to bootstrapping. Specifically, each cell was shifted in cell cycle

assignment from its calculated cell cycle stage ℓ_i to a nearby cell cycle stage, ℓ_{i+r}, where r is a random variable, $r \sim N(\mu,\sigma)$ with $\mu = 0$ and $\sigma = \frac{L}{6}$ where L is the length of ℓ. By doing this, cells are reshuffled randomly to nearby cell cycle positions. After each reshuffling, we recalculated the function f. This procedure was repeated numerous times to generate a distribution of values for f and F. We use the standard deviations of f and F that result from this sampling procedure as measures of the errors, δf_ℓ and δF_ℓ associated with mis-localizing cells on ℓ.

Sampling errors in f and F

An estimate for the sampling errors, δf_s and δF_s, were calculated using standard approaches described in [1,2] and are given by:

$$\delta F_s = \sqrt{\frac{F(1-F)}{N}} \qquad \text{Eq. S3}$$

$$\delta f_s = \sqrt{\frac{f(1-f)}{N}} \qquad \text{Eq. S4}$$

where $N = N_t F$, $n = N_t f$ and N_t is the total number of cells in the population.

Total error in f and F

The total errors in f and F are the sums of the errors resulting sampling statistics and errors resulting from mistakes in the cell cycle parameterization, ℓ and are given by

$$\delta f = \delta f_s + \delta f_\ell \qquad \text{Eq. S5}$$

$$\delta F = \delta F_s + \delta F_\ell \qquad \text{Eq. S6}$$

To express the fact that δf_ℓ is an error that is propagated from ℓ we rewrite that term as:

$$\delta F_\ell \equiv \left|\frac{\partial F}{\partial \ell}\right| \delta \ell$$

$$\delta f_\ell \equiv \left|\frac{\partial f}{\partial \ell}\right| \delta \ell$$

Resulting with

$$\delta f = \delta f_{int} + \left|\frac{\partial f}{\partial \ell}\right| \delta \ell \qquad \text{Eq. S7}$$

$$\delta F = \delta F_{int} + \left|\frac{\partial F}{\partial \ell}\right| \delta \ell \qquad \text{Eq. S8}$$

Figure 11 shows the error in F and f in RPE1 cells calculated using Eq. S7 and Eq. S8.

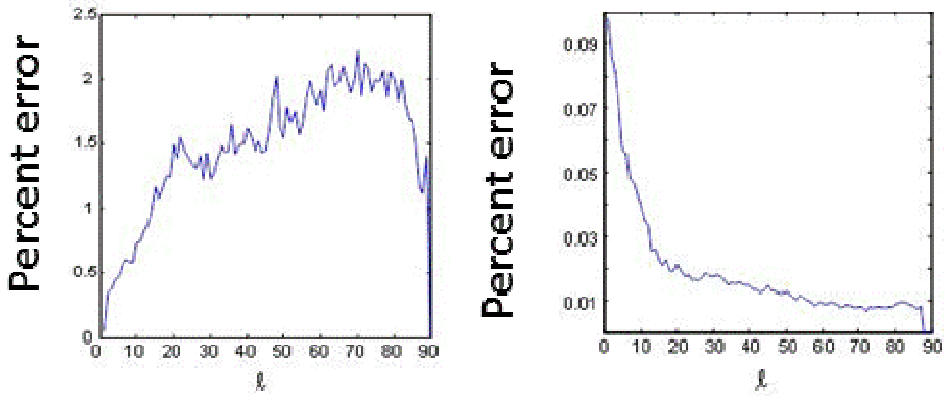


Figure 11: The error in the probability density function, f, and in the cumulative probability function, F, from data collected for RPE1 cells. Note that due to the large cell counts, errors fall below 2% in both cases. Also, note that the cumulative distribution function has significantly lower errors, again due to increased cell count compared to the non-cumulative distribution.

Calculation and errors in the rate of cell cycle progression, ω

Eq. 2 in the main article text describes ERA expressed in a single dimension (ℓ) in integral form. It can be derived from the more general Eq. 1 in the main article text. Here we will take a simpler route and derive Eq. 2 directly in its 1-dimensional integral form using very simple and intuitive formalism.

A simplified derivation of the ERA equation.

Figure 12 shows how cells distribute along the cell cycle axis, ℓ, in RPE1 cells.

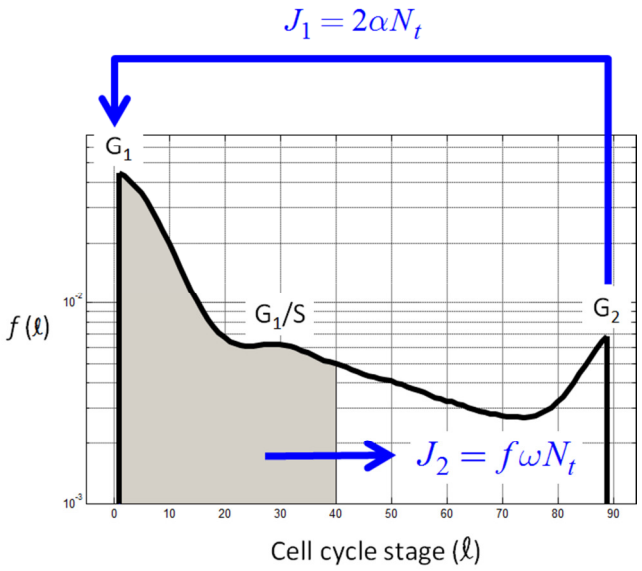


Figure 12: Cell growth and cell division balance to shape the distribution of cells along ℓ. The distribution of cell progression is shown as a function of the cell cycle progression axis, ℓ. A state ($\ell = 40$) is arbitrarily chosen (border of gray region) to illustrate explanations in the text. The flux of cell growth and cell division are shown (blue arrows)

In steady state $f(\ell)$ does not change with time. This time invariance results from a balance above between two fluxes: the flux, J_1, of newborn cells entering G_1, and the flux, J_2, of cells exiting to a later cell cycle stage. For example, cells continuously exit G_1 (Flux: J_2). Nevertheless, the proportion of cells in G_1 is held constant because the flux of cells leaving G_1 to later cell cycle stages is balanced by a continuous flux of newborn cells that enter G_1 from M-phase. In fact, at steady state, the incoming flux of newborn cells entering G_1 is larger than the flux of

cells leaving G_1 into S-phase. To understand this one must recall that while the proportion of cells in G_1 (or any other cell cycle stage) is constant with time, the number of cells increases exponentially. This latter point is important as it forms the basis for the balance equation that will be used to derive ERA.

In steady state the number of cells in G_1 is given by:

$$N_{G_1} = N_t f(G_1) \qquad \text{Eq. S9}$$

where N_t is the total number of cells in the population and $f(G_1)$ is the proportion of cells in G_1. Since

$$N_t = N_0 e^{\alpha t} \qquad \text{Eq. S10}$$

and since at steady state $f(G_1)$ does not depend on time, the number of cells by which the G_1 subpopulation increases per unit time is given by:

$$\frac{d}{dt}\left[N_{G_1}\right] = \alpha N_t f(G_1) \qquad \text{Eq. S11}$$

A generalization of this, illustrated by Figure 12, is that the increase in cell count within any cell cycle interval $\ell \le \ell_0$ (gray region in Figure 12) is given by

$$\frac{d}{dt}\left[N_{\ell \le \ell_0}\right] = \alpha N_t F(\ell) \qquad \text{Eq. S12}$$

To derive ERA consider the subpopulation of cells that are at or before cell cycle stage, ℓ_0 (i.e. $\ell \le \ell_0$). This subpopulation increases by $\alpha N_t F(\ell_0)$ cells per unit time (Eq. S13). This net increase is the consequence of a balance between fluxes as described above:

$$J_1 = 2\alpha N_t \qquad \text{Eq. S14}$$

$$J_2 = \omega f N_t \qquad \text{Eq. S15}$$

The factor "2" in flux J_1 arises from the fact that every cell division event results in two newborns being added to the interval $\ell \le \ell_0$

Combining Eq. S12 with Eq. S14 and Eq. S15 gives:

$$\alpha N_t F = 2\alpha N_t - \omega f N_t \qquad \text{Eq. S16}$$

Upon rearrangement, Eq. S17 yields:

$$\omega = \alpha \frac{2-F}{f} \qquad \text{Eq. S18}$$

where α is the exponential constant of population expansion (proliferation) and is obtained from plotting the number of cells vs. time and fitting to exponential kinetics. α can be interpreted as the fraction of cells dividing per unit time and is given in units of $^{1}/_{time}$.

Matlab script for calculating the rate of cell cycle progression.

```
function [w]=ERA(f,F,alpha)
% the ERA transform
% F - the cumulative probability density as a function of Ell.
% Thus, F(1)=0 and F(end)=1. F(i) is the proportion of cells
% with Ell<=i.
% f - the probability density as a function of Ell. f(i) is the
% proportion of cells at cell cycle stage "Ell==i".
% alpha - the proportion of cells dividing per unit time.
% v - the rate of cell cycle progression as a function of Ell
w=alpha*(2-F)./f;
```

Error analysis of the rate of cell cycle progression.

Since the rate, ω, of cell cycle progression is calculated exclusively from *f* (Eq. S1) and *F* (Eq. S2) (α is a scaling factor and does not affect the shape of the resulting curves), errors in ω are propagated solely from errors in the functions *f* and *F*. Applying the method of *propagation of errors*[3], the total error in ω is therefore given by:

$$\delta\omega = \left|\frac{d\omega}{df}\right| \delta f + \left|\frac{d\omega}{dF}\right| \delta F \qquad \text{Eq. S19}$$

Errors in the parameterization of cell cycle (ℓ) are propagated into $\delta\omega$ from Eq. S7 and Eq. S8. Introducing Eq. S18 into Eq. S19 results in:

$$\delta\omega = \alpha\left|\frac{\partial}{\partial f}\left(\frac{2-F}{f}\right)\right|\delta f + \alpha\left|\frac{\partial}{\partial F}\left(\frac{2-F}{f}\right)\right|\delta F$$

$$= \alpha\frac{2-F}{f^2}\delta f + \frac{\alpha}{f}\delta F$$

$$\delta\omega = \alpha\frac{2-F}{f^2}\delta f + \frac{\alpha}{f}\delta F \qquad \text{Eq. S20}$$

Eq. S20 can be solved by using values for δf and δF (shown in Figure 11) calculated above.

Assumptions of the ERA calculation

One: the appropriateness of the labeled coordinate system.

To perform ERA, single cells are co-labeled for two types of targets differing in their roles in the ERA calculation. First, one must label the target or targets that are under investigation. In the case of the present study this target was the total cellular protein mass. The second type of labeled targets function not as a focus of research but as an "axis of representation", i.e. "a coordinate system". In our case, these targets were the Geminin degron (mAG-hGem) and DNA (DAPI). The function of the measured levels of DNA and Geminin in this study was to form a coordinate system quantifying progression through cell cycle. For accurate ERA, one must assume adequate knowledge of the behavior of the targets that are selected to function as a coordinate system. For example, in the case of the present study, interpretation of the data was based on prior knowledge that the levels of neither DNA nor mAG-hGem decrease until the M-phase stage of cell cycle. In general, the quality of ERA largely depends on the appropriateness the chosen coordinate system.

Test of validity: This assumption requires some understanding of the biological system that is studied. In our case, the cell cycle dependency of Geminin and DNA have been sufficiently

characterized to be used as cell cycle markers. We know, for example, that neither DNA levels nor Geminin levels go down until mitosis. To establish this for the mAG-hGem, we recorded movies by time lapse microscopy of Geminin accumulation during cell cycle in single cells (Fig. 3, main text).

Two: average cell dynamics match averaged cell dynamics

In cases where ERA is implemented by reduction in dimensionality, as done in the current study, we assume that the individual cell dynamics are closely approximated by the averaged cell dynamics determined by the calculated parameterized trajectory. An example where this condition is not satisfied is when the measured protein levels in individual cells oscillate with time but the population average remains roughly constant (for example, p53, NF-κB).

Test of validity: This assumption is not mathematical but biological. It assumes prior knowledge about the system studied. For example, in our case we assume that Geminin levels do not oscillate.

Three: the weak ergodic assumption

We assume that the distributions of all variables in which we are interested are not dependent on time. For example, the distribution of cell size as a function of the reaction coordinate, ℓ, does not vary with time.

Test of validity: Distributions of the levels of labeled targets can be measured at different times after plating to establish time-invariance as in Figure 4 and figure 7 in this supplementary file.

Four: the strong ergodic assumption.

We assume a homogeneity in the population in the sense that the dynamics of each of our measured (labeled) targets (e.g. Geminin, protein mass, etc.), averaged over the parental lineage of any single cell would be equal (or sufficiently close to) the average dynamics of all cells in a population at a signal time point. This criterion would not hold, for example, if there are inheritable differences (genetic or non-genetic) that affect the dynamic behavior of these targets in the population.

Test of validity: This assumption requires that the population is homogenous and that the averaged calculated dynamics represent a single population. The assumption can be tested by comparing calculated dynamics to actual measured dynamics, as we have done in Fig. 3 in the main article text.

Five: reliability of the measurements.

As with every experimental method, biases and artifacts in measurement could lead to false interpretations.

Test of validity: In the case of our study, cell size measurements were justified by comparison with size measurements collected by QPM, an alternative measurement method (fig 3E, main text). Dynamics of Geminin were characterized by time lapse microscopy (Fig. 3, main text).

A note on averaged rates.

Suppose that the population is composed of multiple subpopulations that progress along cell cycle at different rates (but the same population exponential growth rate, in keeping with the weak ergodic assumption). The calculated value of ω at a given point along cell cycle measures the arithmetic average of the velocity of all cells at that point along cell cycle; however this would not be the arithmetic average of the two averaged velocities of cells from each of the two subpopulations. ω would lie between those two velocities and would be approximately their harmonic mean, in the same way that in the one-dimensional case the frequency f is approximately the reciprocal of velocity.

Calculation and error analysis of the time axis, t.

The ERA transform – transforming the parameterized cell cycle curve into a time axis.

To transform the parameterized cell cycle axis, ℓ into a time axis, *t*, we use Eq. 3 from the main manuscript (Eq. S21 and Eq. S23 in this document).

$$t = \int_{G_1}^{x} \frac{1}{\omega(\ell)} d\ell \qquad \text{Eq. S21}$$

Substituting Eq. S18 into Eq. S21 we obtain

$$t = \frac{1}{\alpha}\int \frac{f}{2-F} d\ell \ . \qquad \text{Eq. S22}$$

Solving the integral we get

$$t = \frac{1}{\alpha}\ln\left(\frac{2}{2-F}\right) . \qquad \text{Eq. S23}$$

where α and F are as defined above. The transformation given in Eq. S23 is shown in Fig. 1B from the main text. The equation relates any point on ℓ to a time from G_1 ($t = 0$ represents G_1).

Error analysis of the time axis.

From Eq. S23 it is clear that the time axis, *t*, is calculated exclusively from the cumulative probability function, F, which describes the probability of cells to be at or before cell cycle stage ℓ. Therefore, the error in *t* is propagated exclusively from errors in *F* by:

$$\delta t = \left|\frac{dt}{dF}\right| \delta F \quad . \qquad \text{Eq. S24}$$

Introducing Eq. S23 into Eq. S24 we obtain

$$\delta t = \frac{1}{2-F}\delta F \qquad \text{Eq. S25}$$

A point about Eq. S25 is that since the error in the calculated time axis, *t*, depends only on the error in *F* and since the error in *F* is very small (see Figure 11, cumulative functions have lower error), there is a very small error associated with *t*.

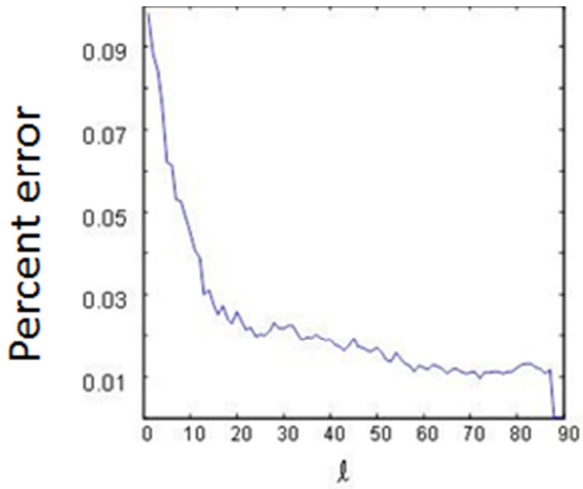


Figure 13: The percent error in the calculated time axis from a measurement on a fixed population of HeLa cells.

Calculating feedbacks: a simplified derivation

Eq. 4 in the main text describes the rate of cell size increase as a function of cell size for any position on ℓ (i.e. any stage in cell cycle). For a simple derivation of Eq. 4, consider the DNA/Geminin phase space (Figure 14). We will derive the calculation based on the assumption that progression along the DNA/Geminin axis is not a function of cell size, though results from the calculation are interpretable more generally.

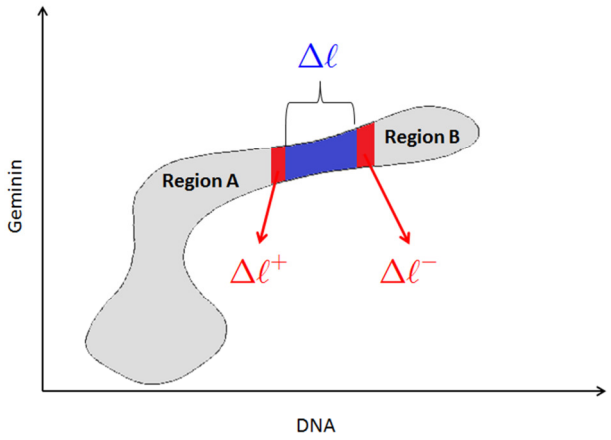


Figure 14: A cartoon depicting the joint probability distribution of DNA and Geminin. A region is marked for calculation of the feedback analysis as described in the text.

Consider Figure 14. We want to calculate the growth rate dependency for cells at the cell cycle stage $\Delta\ell$ (blue region in the figure). To do this we will separate cell cycle into three regions; A, $\Delta\ell$ and B. For these three regions we will, at first, consider two fluxes:

1) The number of cells per unit time transitioning from region A to region $\Delta\ell$: $J_{A\to\Delta\ell}$

2) The number of cells per unit time transitioning from region $\Delta\ell$ to region B: $J_{\Delta\ell\to B}$

Calculation of flux 1: $J_{A\to\Delta\ell}$

From Eq. S12, the rate of increase in cell count in region A is given by

$$\frac{d}{dt}\left[\lambda_A N_t e^{\alpha t}\right] = \alpha\lambda_A N_t$$

where λ_A is the fraction of cells in Region A and α is the fraction of cells dividing per unit time. In contrast, the number of newborn cells entering region A per unit time is $2\alpha N_t$. So, the number of cells per unit time leaving region A is:

$$J_{A\to\Delta\ell} = 2\alpha N_t - \alpha N_t \lambda_A$$

Note that $J_{A\to\Delta\ell}$ is a number and not a function of ℓ

Calculation of flux 2: $J_{\Delta\ell\to B}$

Similar to the calculation of flux 1, we have

$$J_{\Delta\ell\to B} = J_{A\to\Delta\ell} - \alpha N_t \lambda_B$$

where λ_B is the fraction of cells in the region $\Delta\ell$. After substituting, the above Eq. becomes:

$$J_{\Delta\ell\to B} = 2\alpha N_t - \alpha N_t \lambda_A - \alpha N_t \lambda_B$$

The size dependent flux

The total number of cells transitioning from region A to region $\Delta\ell$ per unit time is $J_{A\to\Delta\ell}$. The proportion of cells transitioning from A to $\Delta\ell$ that have a size smaller than or equal to s_0 is $F(s_0 \mid \Delta\ell^+)$, where $F(s_0 \mid \Delta\ell^+)$ is the cumulative probability distribution of cell size at the entrance to the interval, $\Delta\ell$ (marked red in Figure 14).

Thus, the number of cells with $s < s_0$ that enter region $\Delta\ell$ per unit time is:

$$J_{A\to\Delta\ell} F(s_0 \mid \Delta\ell^+)$$

$$= \left(2\alpha N_t - \alpha N_t \lambda_A\right) F(s_0 \mid \Delta\ell^+)$$

$$= \alpha N_t \left(2 - \lambda_A\right) F(s_0 \mid \Delta\ell^+) \qquad \text{Eq. S26}$$

Similarly, the number of cells with $s < s_0$ that exit region $\Delta\ell$ per unit time is:

$$J_{\Delta\ell\to B} F(s_0 \mid \Delta\ell^-)$$

$$= \left(2\alpha N_t - \alpha N_t \lambda_A - \alpha N_t \lambda_B\right) F(s_0 \mid \Delta\ell^-)$$

$$= \alpha N_t \left(2 - \lambda_A - \lambda_B\right) F(s_0 \mid \Delta\ell^-) \qquad \text{Eq. S27}$$

Cell growth

From Eq. S26, the number of cells with size $S<S_0$ that enter region $\Delta\ell$ per unit time is (Figure 15):

$$InFlux = \alpha N_t \left(2 - \lambda_A\right) F(s_0 \mid \Delta\ell^+) \qquad \text{Eq. S28}$$

From Eq. S27, the number of cells with size $S<S_0$ that exit region $\Delta\ell$ per unit time is:

$$OutFlux = \alpha N_t \left(2 - \lambda_A - \lambda_B\right) F(s_0 \mid \Delta\ell^-) \qquad \text{Eq. S29}$$

The net accumulation of cells with size $S<S_0$ in region $\Delta\ell$ is $\alpha N_t P(s < s_0, \Delta\ell)$, where $P(s < s_0, \Delta\ell)$ is the proportion of cells in region $\Delta\ell$ with size smaller than s_0. By the conditional probability rule:

$$P(s < s_0, \Delta\ell) = F(s_0 \mid \Delta\ell)\lambda_B \ ,$$

since λ_B is the fraction of cells in the region $\Delta\ell$.

Thus,

$$NetAccumulation = \alpha N_t F(s_0 \mid \Delta\ell)\lambda_B \qquad \text{Eq. S30}$$

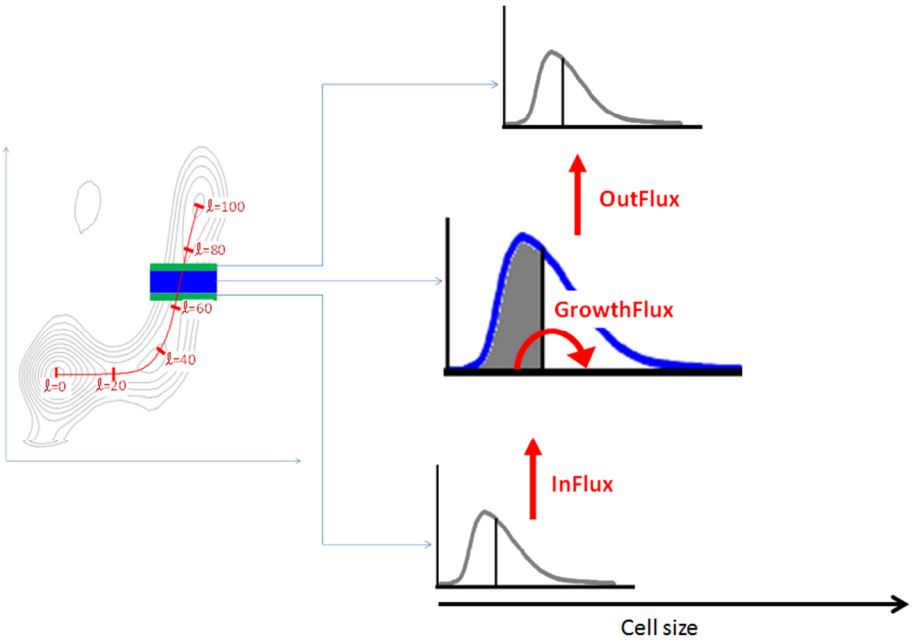


Figure 15: Feedback calculation. The size distribution of cells at any interval, $\Delta\ell$, in cell cycle is the outcome of a balance between three fluxes: the flux (number of cells per unit time) of small cells entering $\Delta\ell$ from a previous cell cycle stage, the flux of cells leaving $\Delta\ell$ to a later cell cycle stage, and the flux of cells growing out of any size bin.

From equations Eq. S28 to Eq. S30 we can formulate a balance equation (Figure 15):

$$NetAccumulation = InFlux - OutFlux - GrowthFlux$$

Substituting Eqs. S24-S26 and rearranging we get:

$$v = \alpha \frac{(2-\lambda_A)F(s_0 \mid \Delta\ell^+) - (2-\lambda_A-\lambda_B)F(s_0 \mid \Delta\ell^-) - F(s_0 \mid \Delta\ell)\lambda_B}{f(s_0 \mid \Delta\ell)\lambda_B} \qquad \text{Eq. S31}$$

To calculate the feedback spectra, $\phi(\ell)$, from Eq. S31 we used least squares to calculate the slope of v vs. s for every value of ℓ (Figure 16).

$$\phi(\ell) = slope(v, s)$$

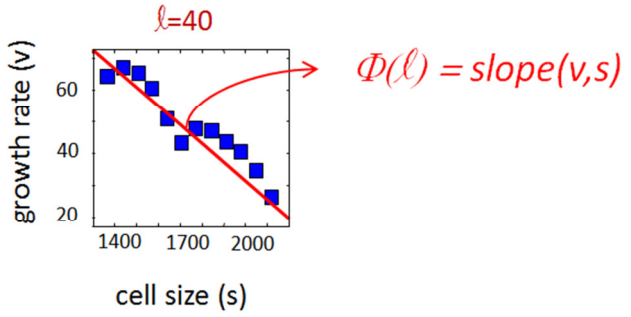


Figure 16: Calculation of feedback spectra from Eq. S31. Eq. S31 yields growth rate as a function of cell size for every interval on the cell cycle axis, ℓ. From each such plot we estimate the best fitting linear approximation using least squares. The slope of growth rate vs. cell size (resulting from the least square fit) is set as $\phi(\ell)$.

Error analysis of the feedback calculation

To estimate the uncertainty in the measurements of $\phi(\ell)$, we used resampling, as direct calculations using error propagation are infeasible.

We used 800 iterations of bootstrapping without replacement, making the following calculations at each iteration i:

1) Randomly sample 75% of the observed cells (20% for the substantially larger L1210 dataset)
2) Calculate the 2-dimensional probability distribution from this sample
3) Calculate the loop ℓ_i and loop assignment based on this probability distribution
4) Calculate the rate of progression through ℓ_i, and derive the time axis t_i for this loop
5) Calculate the size-dependent growth rate $v(s, \ell_i)$ and the slope $\phi_i(\ell_i)$
6) Plot ϕ_i as a function of time t_i along the loop ℓ_i.

Given the 800 slope functions ϕ_i, we interpolated them to a common set of 50 time points, and then took the mean and standard deviation of the 800 values of ϕ at each time point.

We then used the time axis t to convert these time points back to locations on the loop ℓ determined from the full data set. This give us, at 50 points along ℓ, the mean and standard deviation of the bootstrap calculation of $\phi(\ell)$.

Confidence intervals for $\phi(\ell)$ calculated with the above approach are shown in Fig. 6 in the main article text. The horizontal axis represents the loop ℓ as used with the full data set, and vertical axis represents growth rate vs. cell size. The line in red represents the function ϕ of slopes as computed with the full data set. The shaded area represents the interval one standard deviation on either side of the mean of the bootstrap calculation of $\phi(\ell)$. For the HeLa cells, the variation was much higher; in that case we plot 1.645 standard deviations on either side of the mean to show that with 90% confidence, the slope does become negative at the G1/S transition.

Some technical notes:

1) sampling without replacement was used because the kernel density estimator used in step 5, based on [kde [4]], chooses an extremely small smoothing parameter when duplicated data is present.
2) Switching from ℓ_i to t_i to align the functions ϕ_i, and then switching back to ℓ for the plot in the figure, will only serve to increase the width of the error bars.

Matlab script for feedback calculation:

```
function [Sl,Rsqare,EllAxis]=NegativeFeedBackCalculator(Ell,P,ResP,W,w)

% Calculate the negative feedback plot along the trajectory Ell (Fig. 6 in manuscript).
% input arguments:
%
% P -       a vector with values of protein mass (cell size) for each individual
%           cell. The length of P is equal to the number of cells measured.
% Ell -     the cell cycle parametrization. Ell is a vector with
%           length(Ell)==length(P). Each cell-size P(i) is associated with a cell
%           cycle position, Ell(i).
% ResP -    the resolution of the calculation (number of bins of cell size).
%           Larger values for ResP would correspond to better resolutiion and worse
%           acuracy (more noise).
% W -       the width of the intervals in which the negative feedback is
%           calculated
% w -       width of the bounding intervals (see text)

% output arguments:
% Sl -      the slope of growth rate vs cell size as a function of Ell. The
%           length of Sl is the same as the length of Ell
% Rsqare -  quality of the linear estimate
% EllAxis - The value of Ell that corresponds to each value of Sl


sc=linspace(prctile(P,5),prctile(P,95),ResP);
windowSize=W;
BoundingSize=w;
L=0;
counter=0;

for i=BoundingSize+1:1:310
    counter=counter+1;
    Ia=find(Ell<i);
    IaExit=find(Ell>i-BoundingSize & Ell<i);
    Ib=find(Ell>=i & Ell<i+windowSize);
    IbExit=find(Ell>=i+windowSize & Ell<=i+windowSize+BoundingSize);
    SaExit=(P(IaExit));
    Sb=(P(Ib));
    SbExit=(P(IbExit));
    PhiA=length(Ia)/length(Ell);
    PhiB=length(Sb)/length(Ell);
    [v,fb,faExit,fbExit]=RateCalculator3(Sb,SaExit,SbExit,PhiA,PhiB,sc);
    pcr=prctile(Sb,[10 90]);
    ii=find(sc>pcr(1) & sc<pcr(2));
    [fresult,gof]=fit(sc(ii)',v(ii)','poly1','weights',fb(ii),'robust','on');
    Sl(counter)=fresult.p1;
    Rsqare(counter)=gof.rsquare;
    Interc(counter)=fresult.p2;
    mV(counter)=(2-length(Ia)/length(Ell))/length(Ib);
    EllAxis(counter)=mean(Ell(Ib));
    L=L+length(ii);
    disp(i)
end
```

Calculation and error analysis of growth rate vs cell size.

Figs 4 in the main text shows the averaged dependency of growth rate on cell size for cells from mid G_1 to G_2. This dependency was calculated by Eq. S31 (Eq. 4 in main text) with $\Delta\ell$ being from late G1 (APC inactivation) to G2. In the following we provide an analysis of errors associated with the calculation.

Calculation of growth rate vs. cell size

The equation providing growth rate as a function of cell size is:

$$v(s \mid \ell_{(a,b)}) = \alpha \frac{(2-\lambda_a) F(s \mid \ell_b) - (2-\lambda_a-\lambda_b) F(s \mid \ell_b) - \lambda_b F(s \mid \ell_b)}{\lambda_b f(s \mid \ell_{(a,b)})} \qquad \text{Eq. S32}$$

Where λ_a is the fraction of cells with $\ell < \ell_a$ and $F(s \mid \ell_a)$ is cumulative size distribution of cells with cell cycle stage $\ell = \ell_a$ (the fraction of cells at $\ell = \ell_a$ with size that is equal to or smaller than *s*). $f(s \mid \ell_{(a,b)})$ is the probability distribution of cell size for cells in the cell cycle stage interval $\ell_a < \ell < \ell_b$. $v(s \mid \ell_{(a,b)})$ is the average growth rate as a function of cell size for cells in the cell cycle stage interval $\ell_a < \ell < \ell_b$ (Figure 17). Eq. S32 is a function of three cumulative distribution functions, $F(s \mid \ell_b)$, $F(s \mid \ell_b)$, $F(s \mid \ell_b)$ and the probability density function, $f(s \mid \ell_b)$. All density functions were calculated from single cell size measurements using the Parzen kernel density estimation method[5]. In Matlab, this method is implemented by the function *ksdensity*.

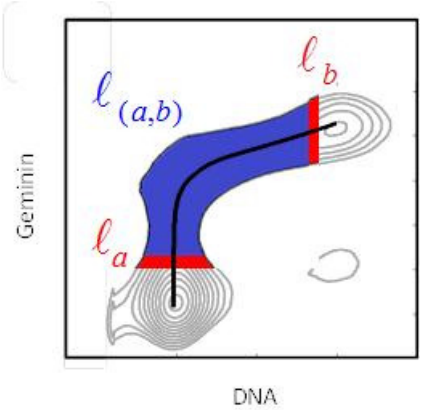


Figure 17: Calculation of growth rate vs cell size for the cell cycle interval between early G_1 to G_2. Calculation was performed with the same strategy employed for the calculation of feedbacks.

Error analysis of growth rate vs. cell size.

Errors in the cumulative probability density functions were calculated by Eq. S8. Confidence intervals for the (non-cumulative) probability density function, $f\left(s \mid \ell_b\right)$ were calculated based on[5]:

$$f_{min} = (\sqrt{f} - se)^2 \qquad \text{Eq. S33}$$

$$f_{\max} = (\sqrt{f} + se)^2 \qquad \text{Eq. S34}$$

Where

$$se = \frac{1}{\sqrt[2]{\pi}\sqrt{2hN}}$$

N is the sample size and h is a width parameter which is estimated from data as described in [5].

Because F is a cumulative we have (see Figure 11):

$$\delta F\left(s \mid \ell_b\right) \ll \delta f\left(s \mid \ell_{(a,b)}\right)$$

$$\delta F\left(s \mid \ell_b\right) \ll \delta f\left(s \mid \ell_a\right)$$

$$\delta F\left(s \mid \ell_b\right) \ll \delta f\left(s \mid \ell_b\right)$$

And errors in the estimation of the cumulative probability density functions are negligible compared to errors on the non-cumulative probabilities. Based on this, the error in the growth rate vs. cell size could be estimated from Eq. S32 by:

$$\delta v = \left|\frac{\partial v}{\partial f}\right| \delta f$$

$$= c \frac{1}{f\left(s \mid \ell_{(a,b)}\right)^2} \delta f\left(s \mid \ell_{(a,b)}\right) \qquad \text{Eq. S35}$$

Where

$$c = \alpha \frac{\left(2-\lambda_a\right) F\left(s \mid \ell_b\right) - \left(2-\lambda_a-\lambda_b\right) F\left(s \mid \ell_b\right) - \lambda_b F\left(s \mid \ell_b\right)}{\lambda_b}$$

Figure 18 shows the result of implementing the above expression to calculate confidence intervals for the growth rate vs cell size calculation:

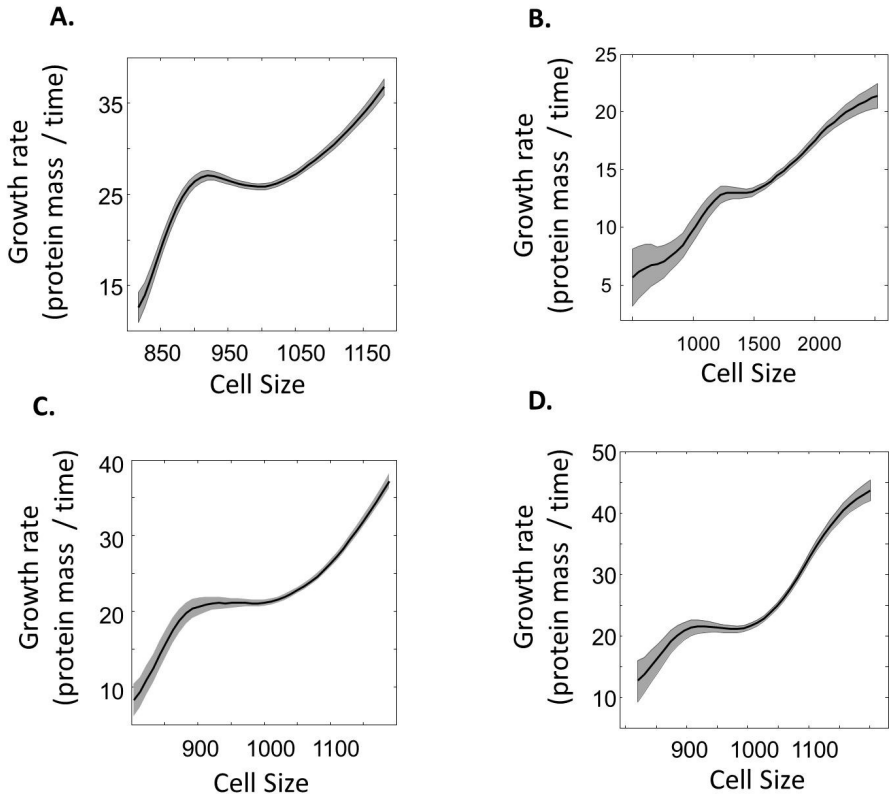


Figure 18: Growth rate vs. size and confidence intervals for HeLa (A), L1210 (B), HT1080 (C) and RPE1 (D) cells.

Calculation and error analysis of growth rate vs time.

To calculate growth rate vs time we calculated average cell size per position on the parametrized cell cycle trajectory, ℓ. We then converted ℓ into time using Eq. S21 and computed the derivative. As typical with numerical calculation of derivatives, the method is highly affected by noise in the data. To overcome this and to provide estimates for the errors in growth rate we applied linear fitting with a smoothing window. We chose a smoothing window with width $\Delta\ell = W$, where W was chosen as approximately one tenth the length of ℓ. Following that, for each point, ℓ_i, we used least square fitting to calculate the slope of cell size vs time for cells confined to an interval of width W centered on ℓ_i. Errors for the calculated slope were computed based on standard error in least square statistics[1,2], thus:

$$\delta Gr = s\sqrt{\frac{1}{W} + \frac{\overline{x}^2}{ss_{xx}}}$$

Where[1,2] :

$$ss_{xx} = \sum_{i=k}^{k+W} (x_i - \overline{x})^2$$

$$ss_{yy} = \sum_{i=k}^{k+W} (y_i - \overline{y})^2$$

$$ss_{xy} = \sum_{i=k}^{k+W} (x_i - \overline{x})(y_i - \overline{y})$$

$$s = \sqrt{\frac{ss_{yy} - \frac{ss_{xy}^2}{ss_{xx}}}{W-1}}$$

To test the effect of errors in cell cycle parameterization and sampling statistics on calculations of growth rate vs time and further characterize the confidence intervals we employed a bootstrapping resampling procedure. Results of this calculation are shown in Figure 19.

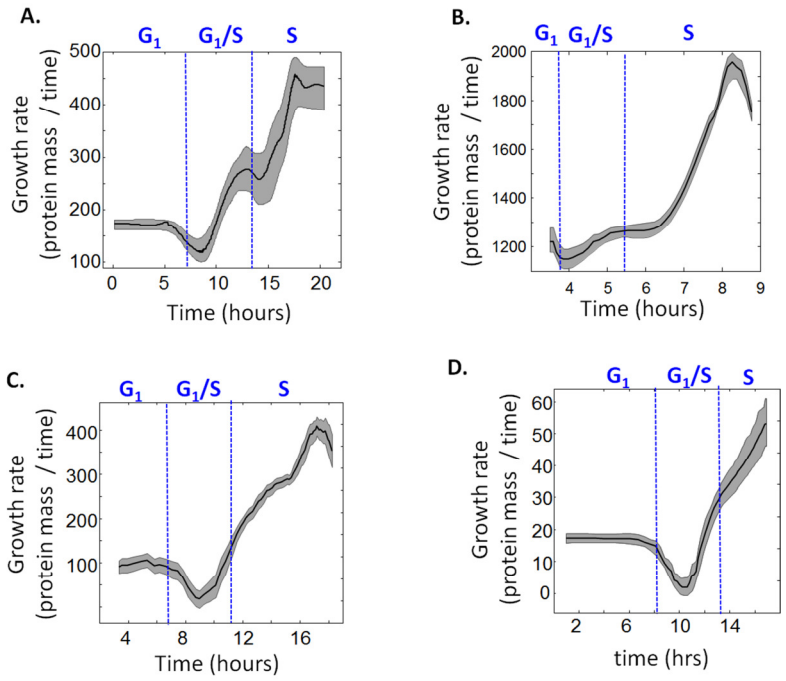
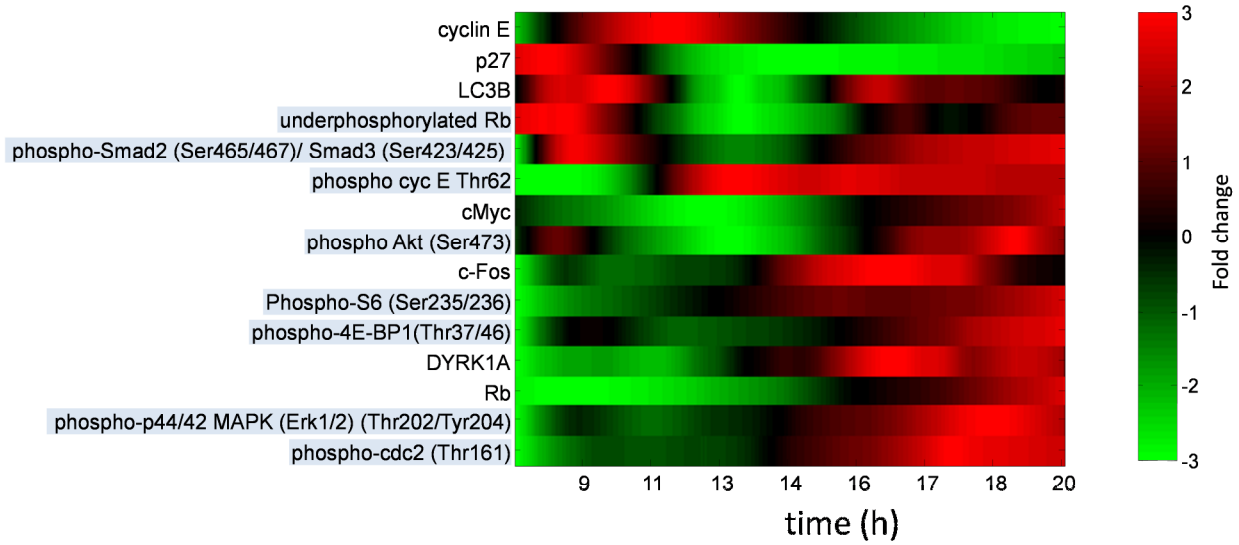


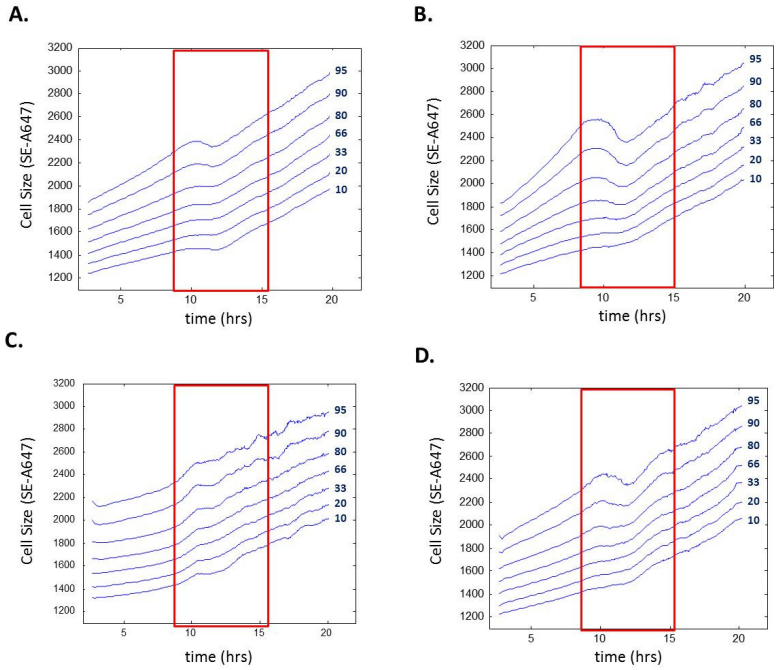
Figure 19: Legend: Dynamics of growth as a function of cell cycle stage and as a function of cell size. The panels show the results of ERA calculations for growth rate as a function of time (A-D) and growth rate as a function of cell protein mass (E-H) from single cell data on HeLa cells (A, E), L1210 mouse lymphocytes (B, F) human fibrosarcoma, HT1080 (C, G) and the non-transformed immortalized human retina epithelium RPE1 (D,H).

Applying ERA to signal transduction



Legend: Dynamics of protein and phospho-protein levels calculated by ERA. To calculate the time dependency of the antibody labels, immunofluorescence measurements were multiplexed with measurements of mAG-hGem, DAPI and SE-A647. By associating every cell with a specific point on the inferred cell cycle stage axis, ℓ, we obtained average antibody signal intensities as a function of ℓ. We then used Eq. 3 (Box 2) to transform ℓ into the time axis plotted here. Antibodies used are: Phospho-S6 Ribosomal Protein (Ser235/236) (Cell Signaling, #4858), phospho-Akt Ser473 (Cell Signaling, #4060), Phospho-p44/42 MAPK (Erk1/2) (Thr202/Tyr204) (Cell Signaling #4370), cMyc (Cell Signaling, #5605), Phospho-cdc2 (Thr161) (Cell Signaling, #9114), Rb (Cell Signaling, #9309M), DYRK1A (Cell Signaling, #2771), c-Fos (Cell Signaling, #2250), Phospho-4E-BP1(Thr37/46) (Cell Signaling, #2855), p27 (AbCam, ab32034), underphosphorylated Rb (Enzo Life Sciences, MAb549), cyclin E (Santa Cruz, sc-247), phospho cyclin E Thr62 (Cell Signaling, #4136), Phospho-Smad2 (Ser465/467)/ Smad3 (Ser423/425) (Cell Signaling, #9510), LC3B (D11) (Cell Signaling, #3868)

Further analysis of drug response measurements.



Legend: The contribution of protein synthesis and protein degradation to cell size control. The panels show the effects of various drugs on cell growth for different percentiles of the cell size distributions (labeled as the 10^{th}, 20^{th}, 33^{rd}, 66^{th}, 80^{th}, 90^{th} and 95^{th} percentiles) (A) untreated HeLa cells; (B) cells treated for one hour pulse of the translation inhibitor, cycloheximide; (C), cells treated for one hour with the proteasome inhibitor, MG132; and (D) cells treated for 1 hour pulse with the mTOR inhibitor, rapamycin. Further statistical analysis and confidence intervals for this plot are provided in supplementary file 2.

Comparison to results from previous publication

In a previous publication we have used the Collins-Richmond equation [6] to calculate growth rate as a function of cell size for L1210 cells. Unlike the use of ERA, the Collins Richmond method fails to capture the association and dependency of growth rate on other variables such as cell cycle. Below are results of the Collins-Richmond calculation from three different studies, performed on different cell lines and relying on different experimental methods for cell size measurements. First is an early calculation by Anderson, published in 1969 [7] and based on Coulter Counter measurements of cell volume. A drawback of that study is that is that the calculation was based on an untested assumption regarding the size distribution of newborn cells. The second is a more recent study that we have published in 2009 [8], also based on Coulter Counter measurements that relies on less assumptions. Last is a Collins-Richmond calculation from data collected in the present study, with succinimidyl ester measurements of total cellular protein mass as a proxy for cell size. Strikingly, the growth rate curves resulting from these three studies are nearly identical. This comparison shows that the growth rate dependency on cell size, described in [8] and [7], is consistent across different cell lines and different measurement techniques and is reproducible by different labs. It further justifies our interpretation of the succinimidyl ester based measurement as a proxy for cell size. Nevertheless, considering this, one might ask how these dynamics shown in Fig. 20 are consistent with those presented in present research. Specifically, figure 4E-F may not appear to be consistent with [8] (Fig. 20B, below). To address this question, we calculated the growth rate dependency on size for separate stages of cell cycle (Figure 21).

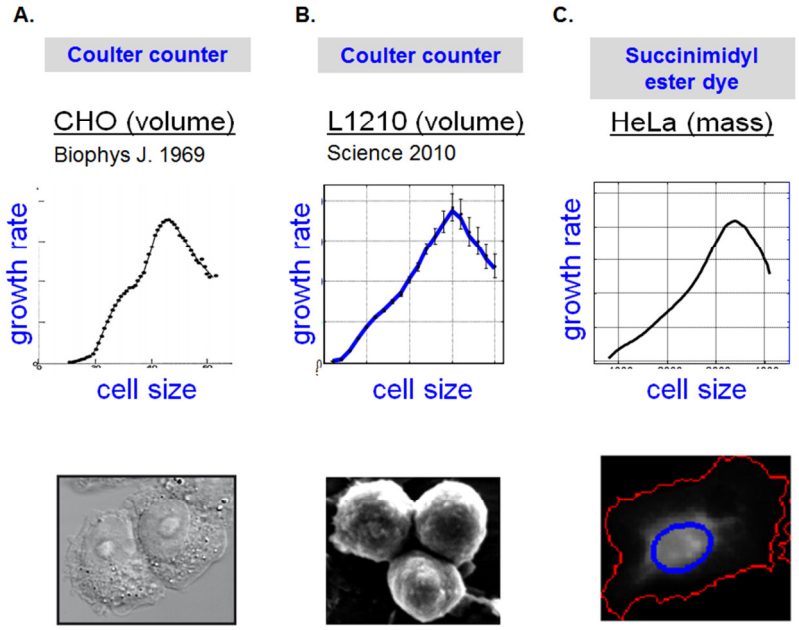


Fig. 20. Growth rate vs. cell size calculated using the Collins Richmond equation. Plots were taken from three independent studies: (A) [7], (B) [8] and the current present study (C). In the first two, size measurement was cell

volume measured by Coulter Counter. In the present study (C), size was measured with succinicimidyl ester reacted with fixed and permeabilized cells.

The results show that different features of the plots in the current study are cell cycle stage specific. For example, the rapid increase in growth rate for the small cells is a result of early G_1; the slight reduction in growth rate following that rapid growth phase is a result of late G_1 and so forth. Further, the results of the growth rate vs. cell cycle that are shown in Fig. 20 and calculated for the whole population (undissected by cell cycle stage) is strongly biased by the G_1 and G_2 subpopulations as these are the most occupied with cell count. This demonstrates that the differences between Fig.4 from the main text and results published in [8] can be accounted for by the dependency of growth rate on cell cycle stage. In early G1 growth rate is dependent on cell size. Since early G_1, i.e. before APC inactivation, constitutes the largest cell cycle phase, its trend dominates the trend calculated for the whole population. The reduction in growth rate seen for very large cells is a consequence of the fact that the fast growing cells divide earlier [9]

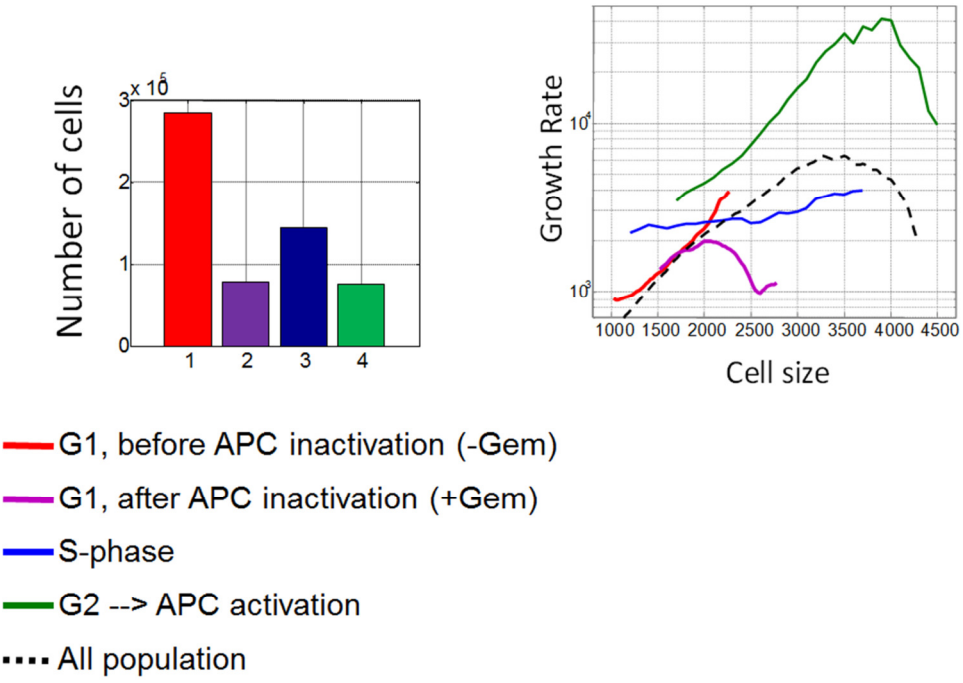


Figure 21: Growth rate vs. cell size calculated using the Collins Richmond equation for different stages of the cell cycle. Also shown is the number of cells in each cell cycle stage.

Testing for non-proliferating subpopulations.

The existence of a non-cycling sub-G1 population would violate the assumptions of ERA. To test whether such a subpopulation exists we pulsed cells for 20 minutes with EdU and tested for EdU retaining cells in G1 33 hours after the pulse (Fig. 22).

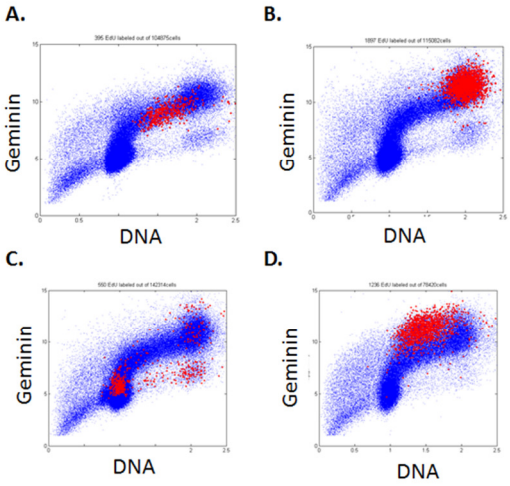


Fig. 22: EdU time course. Cells were pulsed with EdU for 20 min and then fixed 0 hrs (A), 6 hrs (B), 15 hrs (C) and 33 hrs (D) after the 20 min EdU exposure. EdU positive cells are labeled red to distinguish them from the total population (blue). The plot demonstrates that EdU positive cells are first localized to S phase (A). These cells then transition into G2 (B), G1 (C) and back to S phase (D). Note that upon return to S-phase (D) no EdU positive cells are observed in G1.

Confidence intervals for drug response curves

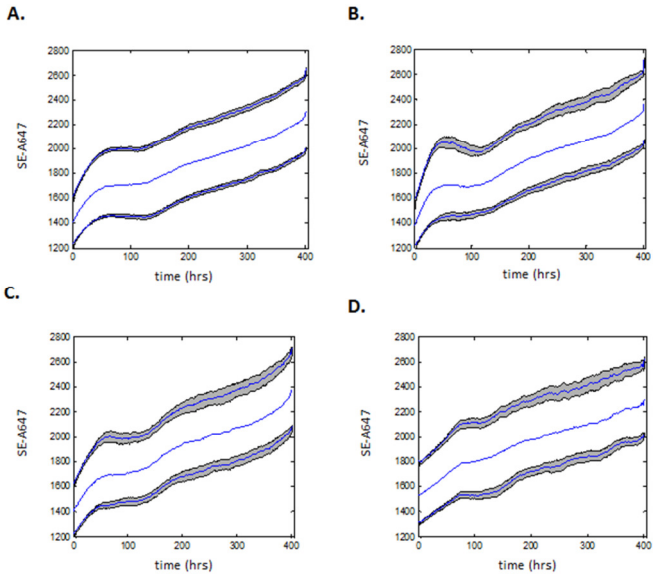


Figure 23: the 90th, 50th and 10th percentiles of cell size as a function of the parameterized cell cycle for untreated cells (A), cells treated with a one hour pulse of cycloheximide (B), cells treated with a 1 hour pulse of rapamycin (C) and cells treated with a 1 hour pulse of MG132 (D). Confidence intervals were calculated by bootstrapping.

Computing the vector field without dimensional reduction.

Eq. 1 in the main article text (the ERA transform) aims to describe dynamics of intracellular events from static measurements performed on a large population of single cells. A limitation of the approach is that a single explicit solution to Eq. 1 can only be obtained if Eq. 1 is expressed in a single dimension. In the described research we solved this problem by reducing the 2 dimensional DNA/Geminin phase space into a single dimension, ℓ, describing cell cycle progression. In the general case, it is interesting to contemplate whether there are approximation methods other than dimensionality reduction that could provide solutions to ERA.

In higher dimensions a solution of Eq. 1 is a vector field describing the dynamics of each of the measured variables. For example, a solution of Eq. 1 for the 2 dimensional DNA/Geminin phase space is a vector field $v \equiv \left(\frac{dDNA}{dt}, \frac{dGeminin}{dt} \right)$

Here we show that although Eq. 1 in the main article text limits but does not completely determine the vector field, v, under mild assumptions we can actually solve for v directly.

If we assume that v is a gradient field, that is, $v = \nabla u$ for some scalar-valued function u on phase space, then Eq. 1 becomes

$$-\nabla \cdot (f \nabla u) + B = \alpha f \quad \text{(S1)}$$

this is an elliptic differential equation in the unknown function u and there exist numerical solvers for such problems. There are some complications, though: since f is very close to 0 over much of phase space (configurations never achieved by cells), the resulting linear system of equations is singular. We solved equation (S1) by imposing a Dirichlet condition that u vanishes whenever f is sufficiently small.

We illustrate this method with the RPE1 cell distribution described in the main text. We first removed from the dataset those cells in late G2, where Geminin levels have already fallen. We then estimated the function $B = 2 * \alpha f_{nb} - \alpha f_{mit}$ accounting for cell birth – f_{nb} representing the actual distribution of newborn cell states, and f_{mit} now representing the distribution of cell states immediately prior to APC activation (and progression to late G2). We solved equation (S1) with the Dirichlet boundary conditions using MATLAB's assempde function. The resulting vector field $v = \nabla u$ is shown in Fig. S1 (A).

To test the quality of this approach, we produced simulated time profiles of Geminin and DNA. The advantage of simulated data is that we know the real parameters which we are attempting to

characterize. The simulation of cells' progression through the Geminin-DNA phase space was based on a differential equation system

$$\frac{dx}{dt} = f_1(x, y)$$

$$\frac{dy}{dt} = f_2(x, y)$$

described by the vector field $v_{true}(x, y) = (f_1, f_2)$ shown below (B), together with division events. After producing a steady-state distribution f (from a population of 5x10^6 "cells") together with the distributions f_{nb} and f_{mit} from this system, we again used MATLAB's assempde to solve (S1) with the Dirichlet condition. The resulting vector field $v = \nabla u$ is shown (fig S1 C).

This method is not as accurate as the reduction to one dimension. The assumption that v is a gradient field has no strong biological justification. Additional assumptions limiting the size and direction of v based on biological understanding should be imposed: for example it is unlikely that cells will increase DNA before APC inactivation and the consequent rise in geminin, and hence the vector field v should not point upwards at that point in the Geminin-DNA phase space. The biologically-motivated assumption we made in the paper was that cells in a given state (as determined from the one-dimensional loop) would have very similar values of v.

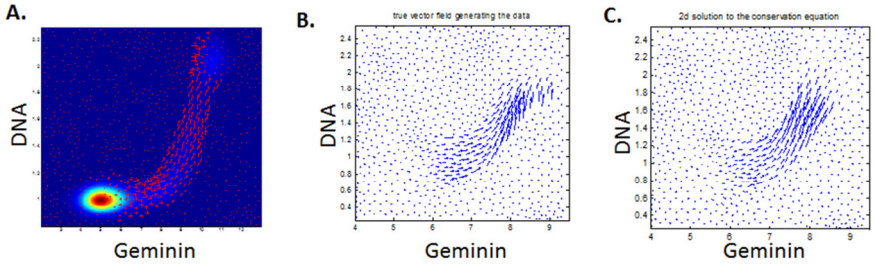


Figure: 2-dimentional ERA calculation.

Supplementary Materials and methods

Culturing procedures: Cells were cultured in Dulbecco's Modified Eagle Medium (Cellgro; DMEM 10-013-CV) with 10% FBS (Cellgro; 35-010-CV) and 1% antibiotic-antimycotic solution (Cellgro; 30–004-CI). Media for cells expressing the mAG-hGem fucci reporter system was further supplemented with 3ug/ml blasticidin (Invivogen; ant-bl-5b) to maintain selection.

Plating cells on coverslips: Cells were plated and fixed on 24X60mm coverslips, No. 1.5 (VWR; 48393 252). Prior to plating, coverslips were sterilized by 20 minute incubation in 70% ethanol at room temperature and then dried in sterile conditions. Cells were typically plated at 10^5 cells per ml into 15cm dishes that were pre-prepared with sterile coverslips as described above. Cells were prepared for

experiment and fixed approximately 48 hours after plating. To avoid artifacts of the freeze/thaw procedure, cells were cultured for at least a week before being plated on coverslips. To dissociate cells from culture plates we incubated the culture in 0.05% trypsin (Cellgro; 25-051-CI) for 5 minutes at 37°C.

Fixing and staining cells. To fix cells, coverslips were removed from culture plates and immediately submersed and incubated for 10 minutes in 4% paraformaldehyde (Alfa Aesar; 30525-89-4) at room temperature. Following the paraformaldehyde fix, cells were washed with PBS and permeabilized by incubating coverslips for 5 minutes in dry methanol at -20°C.

Immunofluorescence and cell size measurement: Cells grown on glass coverslips were fixed in 4% para-formaldehyde for 10 minutes and permeabilized in cold methanol (−20°C) for 5 min. Immunofluorescence protein detection was performed by incubating fixed, permeabilized cells with primary antibody overnight at 4°C and then treating with a fluorescent secondary antibody for 1 hour. To label protein mass, fixed, permeabilized samples were incubated with 0.04 ug/ml succinimidyl ester linked alexa dyes diluted in DMSO (Alexa Fluor 647 carboxylic acid, succinimidyl ester, Invitrogen, A-20106). Following labeling procedures, cells were mounted on glass slides in ProLong® Gold antifade (Life technologies, P36930).

Imaging cells. Slides prepared as described above were imaged with a Nikon Ti Inverted Fluorescence Microscope w/ Perfect Focus controlled by the software, Nikon Elements. We used the scan-slide function to image the full area of the slide at 20X magnification. This resulted in approximately 5000-8000 images per slide, producing data on a total of about 100,000 cells. For larger cell counts, data from multiple slides was concatenated.

Antibodies: Phospho-S6 Ribosomal Protein (Ser235/236) (Cell Signaling, #4858), phospho-Akt Ser473 (Cell Signaling, #4060), Phospho-p44/42 MAPK (Erk1/2) (Thr202/Tyr204) (Cell Signaling #4370), cMyc (Cell Signaling, #5605), Phospho-cdc2 (Thr161) (Cell Signaling, #9114), Rb (Cell Signaling, #9309M), DYRK1A (Cell Signaling, #2771), c-Fos (Cell Signaling, #2250), Phospho-4E-BP1(Thr37/46) (Cell Signaling, #2855), p27 (AbCam, ab32034), underphosphorylated Rb (Enzo Life Sciences, MAb549), cyclin E (Santa Cruz, sc-247), phospho cyclin E Thr62 (Cell Signaling, #4136), Phospho-Smad2 (Ser465/467)/ Smad3 (Ser423/425) (Cell Signaling, #9510), LC3B (D11) (Cell Signaling, #3868)

Image Analysis: Image analysis was performed with custom written software (EnsembleThresher). The algorithm identifies cell boundaries by two complementary approaches. (i) Cells were separated from background by thresholding a Top-Hat transform of the original image. Top-Hat transformation was used to remove trends that are spatially wider than cell diameters. (ii) Boundaries between adjacent, touching cells were identified by seed based watershedding. Seeds were calculated as the regional maxima of the Gaussian smoothed image.

Summary of ERA procedure: To calculate dynamics from the samples of fixed cells the following steps were performed:

i) Cells were fixed on large coverslips and labeled for targets of interest (e.g. protein mass, cell cycle position, etc).

ii) Coverslips were imaged using a microscope equipped with an automated stage and a "scan-slide" algorithm (Nikon Ti Inverted Fluorescence Microscope w/ Perfect Focus controlled by Nikon Elements) to image the full area of the slide (100,000 cells per slide).

iii) Images were processed to identify single cell and nuclei boundaries and collect fluorescence intensity per cell, per label.

iv) The probability density was calculated for the phase space defined by the labeled targets (2D for geminin/DNA). Probability density was calculated with the Parzen kernel density procedure if cell count is low and with a ND histogram if cell count was high)

v) A parameterization of cell cycle was calculated as described in supplementary file 2 and supplementary file 1.

vi) The number of cells was calculated as a function of the parameterized cell cycle path, $f(\ell)$.

vii) $f(\ell)$ was used to calculate the dynamics of cell cycle progression using Eq. 2 (Box 2; Matlab script provided in supplementary file 2).

EnsembleThresher: a matlab code for image processing

For the described research we have wrote EnsembleThresher, a Matlab based image processing software. Below we provide an explanation of the input into EnsembleThresher followed by the source code for EnsembleThresher is provided below.

The input to EnsembleThresher is an excel sheet describing the location of image files to be processed and additional information. Below is an example input excel file. The colums of the file relate to following data:
Index: a running index of imaged slides
Times: if time course was performed, this describes the time point
Software: the software with which images were collected. EnsemleThresher recognized two software, Metamorph and Nikon Elements
Input Directory: the directory where images are stored
Output Dir: directory into which processed images will be saved
Flat: location of the flatfield image (an image of a slide that has uniform fluorescence)
Dark: an image with camera noise
C1 to C5: the names of the imaged channels
Given name: any text that will be stored in the final output variable.

General Info		Source Data: xxx																
Combine_0		index	times	cell line	software	input directory	output directory	flat	dark	Slide Names	rows	columns	C1	C2	C3	C4	C5	SpottedChannal	Given Name
outputNa	Miriam_5_24_12_Rpe1_drugs_large_4	1	1	HeLa	NewElemen	\\research.files.med.h	C:\Users\r\\\researcl	\\researcl	tile			DAPI [nucleus]	Gem	gAkt	prot [prot]				150 uM bpV(phen)
Saturatior	4096	2	2	HeLa	NewElemen	\\research.files.med.h	C:\Users\r\\\researcl	\\researcl	tile			DAPI [nucleus]	Gem	gAkt	prot [prot]				150 uM bpV(phen)

```
function [ETopts]=EnsembleThresherJuly2011(FileName,sheet,varargin)
%% CHECK LABELS
% dbg  - option A:
% (FileName,sheet,dbg,ETopts) - will start from place last recorded by ETopts.
% (FileName,sheet,[t r s]) - will segment only that file
% ... 'sheet1','breakpoint',n) - will stop timepoint after n images
% ... 'sheet1','dbg',dbg):
%     if dbg is a number - do only that slide # (single row from the excell
%     file.
%     if dbg is a vector of 3 - specifies time, row column
%     if dbg is a struct - continue based on ETopts (dbg=ETopts)
%     if dbg is a cell with the structure dbg={['NewElementsID'] 3177 ETopts}
%        then 3177 is the running index id
addpath('ETRuns\')
global DBG % for debugging
DBG=0;
warning off
ETopts.date=date;
[ETopts]=XLS_interface(FileName,sheet);
NumId=ETopts.XLS.NumId';
[ETopts]=num_of_channals(ETopts);

%% input varargin arguments
arg=1;
ETopts.outputpic{1}=0;
while (arg<=length(varargin))
    switch varargin{arg}
        case 'outputpic'
            arg=arg+1;
            ETopts.outputpic=varargin(arg);
        case 'breakpoint'                   %
            ETopts.BreakPoint=varargin{arg+1};   %
            arg=arg+1;
        case 'dbg'                  %
            dbg=varargin{arg+1};   %
            arg=arg+1;
            if isstruct(dbg)
                ETopts_old=ETopts;
                ETopts=dbg;
                ETopts.XLS=ETopts_old.XLS;
                ETopts.timepoint=ETopts.timepoint;
                NumId=ETopts.timepoint:NumId(end);
                DBG='ETopts start'; % code to start from here.
            elseif iscell(dbg)
                if strcmpi(dbg{1},'NewElementsID')
                    DBG=dbg;
                    ETopts_old=ETopts;
                    ETopts=dbg{3};
                    ETopts.XLS=ETopts_old.XLS;
                    ETopts.timepoint=ETopts.timepoint;
                    NumId=ETopts.timepoint:NumId(end);
                end
            elseif length(dbg)==3
                NumId=dbg(1);
                DBG=dbg;
            elseif length(dbg)==1
                NumId=dbg;
            end

        otherwise
            fprintf('Error In Input !  check yourself...\n');
            return;
    end
    arg=arg+1;
end
```

```
%% loop over timepoints
for t=NumId
    ETopts.NumOfProcessedIms=0;
    if ~strcmpi(DBG,'ETopts start')
        ETopts.timepoint=t;
        ETopts.r=1;
        ETopts.c=1;
    end
    % plotting the waitbar
    TotImgs=ETopts.XLS.NumId(end);
    % end of waitbar
    if strcmpi(ETopts.XLS.software(t),'newelements') || ...
            strcmpi(ETopts.XLS.software(t),'newelements4slides')
        [ETopts]=CalculateSingleTimePoint_newelements(ETopts,FileName,sheet);
    else
        [ETopts]=CalculateSingleTimePoint(ETopts,FileName,sheet);
    end
    if length(DBG)==3
        return
    end
    eval(['save ''DATASET_' ETopts.XLS.outputName  ''' ETopts'])
end
[ETopts]=ETAddFieldsv4(ETopts);
eval(['save ''DATASET_' ETopts.XLS.outputName  ''' ETopts'])


function [ETopts]=CalculateSingleTimePoint_newelements(ETopts,FileName,sheet)
global DBG
StartPoint=1;
t=ETopts.timepoint;

% EXPLENATION: CollectFileIndices_4slide or CollectFileIndices are
% functions that, based on the directory of the original images, retrieves
% the indices of all images (the sufix numbers that identify te images. For
% example, if the image name is "tile_s1_c0001_r0002_488" we want to
% collect from the dir all sufixes of "c" and of "r". These suffixes will
% be stord in the variables spX and spY that result from the functions
% below

if strcmpi(ETopts.XLS.software(t),'newelements4slides')==1
    [ETopts,spX,spY]=CollectFileIndices_4slide(ETopts,t);
else
    [ETopts,spX,spY]=CollectFileIndices(ETopts,t);
end
if exist('spX')~=1
    disp('problem !!!!!!!!!!')
    keyboard
end


ETopts.DATA{t}.NumOfCells=0;
mkdir(ETopts.XLS.OutDir{t})
% COLLECTION FILE COORDINATES
% i.e. an association between the linear index of the file and the
% position of the image in the imaged grid.
if strcmpi(ETopts.XLS.software(t),'metamorph')
    [ETopts]=CollectImageIndices(ETopts);
end
% LOOPING ALL POSITIONS IN THE GRID
c=1;r=1;
counter=0;

% DBG option - single entery with [t r c]
if length(DBG)==3
    r=DBG(2);c=DBG(3);
end

% DBG option - entery with ETopts from crashed run
if iscell(DBG)
    if strcmpi(DBG{1},'NewElementsID')
        StartPoint=DBG{2};
    end
elseif strcmpi(DBG,'ETopts start')
    if isfield(ETopts,'r')
        r=ETopts.r;
        c=ETopts.c;
        StartPoint=ETopts.counter;
    end
    DBG='';
end

ETopts.XLS.Last_image(t)=length(spX);
OldTime=clock;
for ImageNumber=StartPoint:length(spX); % StartPoint is 1 unless debugging
    if isfield(ETopts,'BreakPoint')
        if ImageNumber>ETopts.BreakPoint
            break
        end
    end
    ETopts.counter=ImageNumber;
    [Organelles,CombinedImage,ETopts]=LoadImages([spX(ImageNumber) spY(ImageNumber)],ETopts);
    disp(['image number (NewElements): ' num2str(ImageNumber)])
```

```
if ~isfield(Organelles,'nucleus')
    im_nn=Organelles.noname;
    [bw3,x,y]=FindFakeNucs(Organelles.noname,0.3,1,10);
    bw3=imdilate(bw3>0,strel('disk',5));
    Pic=imoverlay(NormalizeImage(im_nn,[0 prctile(im_nn(:),99.5)]),bw3,[1 0 0]);
    Nuclii_p=bw3;
    Organelles.nucleus=Nuclii_p;
else

    if strcmpi(ETopts.XLS.cell_line{t},'hela')
        [Nuclii_p,Pic,ETopts]=SegmentingNucliiv2(Organelles.nucleus,Organelles.nucleus,ETopts);
    elseif strcmpi(ETopts.XLS.cell_line{t},'sknas')
        [Nuclii_p,Pic,ETopts]=SknasNuclii(Organelles.nucleus,Organelles.nucleus,ETopts);
    elseif strcmpi(ETopts.XLS.cell_line{t},'ES cells')
        [Nuclii_p,Pic,ETopts]=SegmentingNucliiv2(Organelles.nucleus,Organelles.nucleus,ETopts);
    else
        clc
        disp('Unknown Cell line')
        return
    end
end
if sum(Nuclii_p(:))==0
    continue
end
counter=counter+1;
Regions.Nuclii=Nuclii_p;
[A,L,L2]=Area_old(Regions.Nuclii,0);

IMt=CombinedImage{1};
for CountChannals=2:length(CombinedImage)
    IMt=IMt+CombinedImage{CountChannals};
end

if isfield(Organelles,'focci')
    IMt=IMt+imopen(Organelles.focci,strel('disk',3));
end

if isfield(Organelles,'prot')
    IMt=Organelles.prot;
end

if strcmpi(ETopts.XLS.cell_line{t},'hela')
    [Cells,Cells2,ETopts]=ET_Lite_Watershed(IMt,Nuclii_p,ETopts);
elseif strcmpi(ETopts.XLS.cell_line{t},'sknas')
    [Cells,ETopts]=SegmentSknasCells(IMt,Nuclii_p,ETopts);
elseif strcmpi(ETopts.XLS.cell_line{t},'ES cells')
    [Cells,Cells2,ETopts]=ET_Lite_Watershed(IMt,Nuclii_p,ETopts);
else
    clc
    disp('Unknown Cell line')
    return
end
if isfield(Organelles,'pericentrin')
    [D,Pericentrin_Regions]=SegmentPericentrin(Organelles.pericentrin,Cells);
    Regions.Pericentrin.loc=Pericentrin_Regions;
    Regions.Pericentrin.D=D;
end
if isfield(Organelles,'cilia')
    [cilia]=SegmentCilia(Organelles.cilia,Cells);
    Regions.cilia=cilia;
end


Regions.Cells=Cells;
[Regions,IOL1,IOL2]=CleaningRegions(Regions);

if isfield(Organelles,'focci')
    Regions.focci=Organelles.focci-imopen(Organelles.focci,strel('disk',4))>0.3;
    Regions.focci=Regions.focci&Regions.Cells;
end

if ETopts.XLS.Combine_Images==1
    [C,N]=CropSegmentedImage(ETopts,Cells,Nuclii);
    [ETopts,Cmat,Nmat]=buildSegmentedImage(ETopts,C,N,c,r);
end
if mod(ImageNumber,10)==0
        JustForPic=find(Cells>0);
        JFP_thr=prctile(CombinedImage{2}(JustForPic),99);
        IOL22=imdilate(IOL2,strel('disk',1));
        im_pic=imoverlay(imoverlay(NormalizeImage(CombinedImage{2}),IOL1,[1 0 0]),IOL22,[0 0 1]);
        if isfield(Organelles,'focci')
            IOL22=Regions.focci;
            im_pic=imoverlay(imoverlay(NormalizeImage(Organelles.focci,[0 2]),IOL1,[1 0 0]),IOL22,[0 0 1]);
        end

            figure(1)
            imshow(im_pic)
            TotImgs=ETopts.XLS.Last_image(t);


end

if isfield(Organelles,'pericentrin')
    Pic=imoverlay(NormalizeImage(Organelles.pericentrin),Pericentrin_Regions>0,[1 0 0]);
```

```
        if mod(ImageNumber,10)==0
        end
    end
    if isfield(Organelles,'cilia')
        Pic=imoverlay(NormalizeImage(Organelles.cilia,[0 0.5]),cilia>0,[1 0 0]);
        Pic3=imoverlay(Pic,bwperim(Cells),[0 0 1]);
        if mod(ImageNumber,10)==0
        end
    end

    % DISPLAYING THE PHASE IMAGE
    if isfield(Organelles,'phase')
        if mod(ImageNumber,10)==0
        end
    end


    [ETopts,Regions]=ExtractData(CombinedImage,Regions,Organelles,spX(ImageNumber),spY(ImageNumber),ETopts);
    if length(DBG)==3
        return
    end
    CURRENT_TIME=clock;
    if CURRENT_TIME(4)~=OldTime(4)
        eval(['save ''DATASET_' ETopts.XLS.outputName  ''' ETopts'])
    end
    OldTime=CURRENT_TIME;
end
eval(['save ''DATASET_' ETopts.XLS.outputName  ''' ETopts'])


function [IM_n,Organelles,ETopts]=Load_Normalize_All_Phases(ETopts,ImageNumber)
t=ETopts.timepoint;
SaturationLevel=ETopts.XLS.SaturationLevel;
ImgNum=ImageNumber;
% loading images, normalizing images and placing them into a structre
counter=0;
counter2=0;
for c=1:NumOfChannals(ETopts,t)
    txt=ETopts.XLS.channal_id{t,c};
    if ~isempty(txt)
        counter2=counter2+1;
        if strcmpi(ETopts.XLS.software{t},'elements')
            [ETopts]=NumOfDigits(ETopts);
            ImageNum_str = num2str(ImgNum,['%05.' num2str(ETopts.NumberOfDigits) 'd']);
            disp([ETopts.XLS.InDir{t} '\' ETopts.XLS.SlideNames{t} ImageNum_str 'c' num2str(counter2) '.tif'])
            im=double(imread([ETopts.XLS.InDir{t} '\' ETopts.XLS.SlideNames{t} ImageNum_str 'c' num2str(counter2) '.tif']));
            if size(im,1)~=512
                im=imresize(im,[512 672]);
            end
        elseif strcmpi(ETopts.XLS.software{t},'metamorph')
            FileName=[ETopts.XLS.InDir{t} '\' ETopts.XLS.SlideNames{t} '_w' num2str(counter2) '_s' num2str(ImgNum)  '_t1.TIF'];
            disp(FileName)
            im=double(imread(FileName));
            if size(im,1)~=512
                im=imresize(im,[512 672]);
            end
        elseif strcmpi(ETopts.XLS.software{t},'newelements')
            [ETopts]=NumOfDigits_newelements(ETopts);
            ImageNum_str1 = num2str(ImgNum(1),['%05.' num2str(ETopts.NumberOfDigits) 'd']);
            ImageNum_str2 = num2str(ImgNum(2),['%05.' num2str(ETopts.NumberOfDigits) 'd']);
            FileName=[ETopts.XLS.InDir{t} '\' ETopts.XLS.SlideNames{t} '_x' ImageNum_str1  '_y' ImageNum_str2 '.tif'];
            ETopts.filename=ImgNum;
            disp([FileName ' ' num2str(ImageNumber)])
            im=double(imread(FileName,counter2));
            if size(im,1)~=512
                im=imresize(im,[512 672]);
            end
        elseif strcmpi(ETopts.XLS.software{t},'newelements4slides')
            [ETopts]=NumOfDigits_newelements4slides(ETopts);
            ImageNum_str1 = num2str(ImgNum(1),['%05.' num2str(ETopts.NumberOfDigits) 'd']);
            ImageNum_str2 = num2str(ImgNum(2),['%05.' num2str(ETopts.NumberOfDigits) 'd']);
            FileName=[ETopts.XLS.InDir{t} '\' ETopts.XLS.SlideNames{t} '_s' num2str(t) '_c' ImageNum_str1  '_r' ImageNum_str2 '_'
ETopts.ChannalSuffix{counter2} '.tif'];
            ETopts.filename=ImgNum;
            disp([FileName ' ' num2str(ImageNumber)])
            im=double(imread(FileName));
            if size(im,1)~=512
                im=imresize(im,[512 672]);
            end
            ETopts.PrcThr(c)=prctile(im(:),97); % an estimate for the total brightness of the picture
        end

        if ~isempty(ETopts.XLS.flat{t})
            flat=double(imread(ETopts.XLS.flat{t}));
            if size(flat,1)~=512
                flat=imresize(flat,[512 672]);
            end

        end
        if ~isempty(ETopts.XLS.dark{t})
            dark=double(imread(ETopts.XLS.dark{t}));
            if size(dark,1)~=512
                dark=imresize(dark,[512 672]);
            end
        else
```

```
            dark=zeros(size(im));
        end
        if c==1
            ETopts.SaturatedPixals=false(size(im));
        end


        ETopts.SaturatedPixals(find(im==SaturationLevel))=true;
        if ~isempty(ETopts.XLS.flat) && ~strcmp(ETopts.XLS.Region{t,c},'phase')
            flat=imresize(flat,size(im));
            dark=imresize(double(dark),size(im));

            im=(im-dark)./(flat-dark);
        end
        if ~strcmp(ETopts.XLS.Region{t,c},'phase')
            if strcmpi(ETopts.XLS.cell_line{t},'hela')
                if size(im,1)==1024 % no binning
                    BG=imopen(imclose(im,strel('ball',1,0,0)),strel('square',70));
                else
                    BG=imopen(imclose(im,strel('ball',3,0,0)),strel('square',100));
                end
            elseif strcmpi(ETopts.XLS.cell_line{t},'sknas')
                BG=imopen(imclose(im,strel('ball',1,0,0)),strel('square',300));
            elseif strcmpi(ETopts.XLS.cell_line{t},'ES cells')
                BG=imopen(imclose(im,strel('ball',1,0,0)),strel('square',150));
            else
                clc
                disp('Unknown Cell line')
                return
            end
        end

        %% Classification (fluorescent channals or Organelles) ?
        if isempty(ETopts.XLS.Region{t,c}) & NumOfChannals(ETopts,t)==1
            counter=counter+1;
            IM_n{counter}=im-BG;
            Organelles.noname=im;
        elseif isempty(ETopts.XLS.Region{t,c})
            counter=counter+1;
            IM_n{counter}=im-BG;
        elseif strcmpi('nucleus',ETopts.XLS.Region{t,c})
            Organelles.nucleus=im-BG;
        elseif strcmpi('prot',ETopts.XLS.Region{t,c})
            Organelles.prot=im-BG;
            counter=counter+1;
            IM_n{counter}=im-BG;
        elseif strcmpi('pericentrin',ETopts.XLS.Region{t,c})
            Organelles.pericentrin=im;
        elseif strcmpi('cilia',ETopts.XLS.Region{t,c})
            Organelles.cilia=im;
        elseif strcmpi('focci',ETopts.XLS.Region{t,c})
            Organelles.focci=im;
        elseif strcmpi('phase',ETopts.XLS.Region{t,c})
            Organelles.phase=im;
        end
    end
end
Vr(1)=length(find(Organelles.nucleus>0.04));
Vr(2)=length(find(IM_n{1}>0.04));
Vr(3)=length(find(IM_n{2}>0.04));
[mx,mxi]=max(Vr);
if mxi~=ProtChannal(ETopts,t)+1 && ...
        ~(sum(Organelles.nucleus(:)>0.04)<5000 || sum(Organelles.prot(:)>0.04)<5000) && ...
     max(Vr)/min(Vr)>1.6
    disp('1')
end
function [Organelles,CombinedImage,ETopts]=LoadImages(ImageNumber,ETopts)
t=ETopts.timepoint;
[IM_n,Organelles,ETopts]=Load_Normalize_All_Phases(ETopts,ImageNumber);
CombinedImage=IM_n;
ETopts.ImageSize.H=size(IM_n{1},1);
ETopts.ImageSize.L=size(IM_n{1},2);


function [ETopts,Regions]=ExtractData(IM,Regions,Organelles,r,c,ETopts)

t=ETopts.timepoint;
L=bwlabel(Regions.Cells);
Lregions=Regions;
Nt=ETopts.DATA{t}.NumOfCells;


Lregions.Nuclii=double(Lregions.Nuclii).*L;
Lregions.Cells=double(Lregions.Cells).*L;
if isfield(Lregions,'Pericentrin')
    Lregions.Pericentrin.loc=double(Lregions.Pericentrin.loc).*L;
end
if isfield(Lregions,'focci')
    Lregions.focci=double(Lregions.focci).*L;
end

NumChannals=ETopts.XLS.num_channals(t);

STATS_cells = regionprops(L,'BoundingBox','Solidity','Perimeter','MajorAxisLength','MinorAxisLength'
,'Eccentricity','Orientation','PixelIdxList');
```

```
STATS_nuclii =
regionprops(Lregions.Nuclii,'BoundingBox','Centroid','Solidity','Perimeter','MajorAxisLength','MinorAxisLength','Eccentricity','O
rientation','PixelIdxList','area' );
% Lnuc(find(Lnuc>0&(~(L>0))))=0;

for i=1:max(L(:))
    index=ETopts.DATA{t}.NumOfCells+i;

    % BOUNDING REGION OF CELL
    x1=STATS_cells(i).BoundingBox(1);
    y1=STATS_cells(i).BoundingBox(2);
    x2=STATS_cells(i).BoundingBox(1)+STATS_cells(i).BoundingBox(3);
    y2=STATS_cells(i).BoundingBox(2)+STATS_cells(i).BoundingBox(4);

    % IMAGE DETAILS (Row, Column, time)
    Height=ETopts.ImageSize.H;
    Length=ETopts.ImageSize.L;
    R(1,1:2)=[r+double(y1>Height) c+double(x1>Length)];   %1
    R(2,1:2)=[r+double(y2>Height) c+double(x1>Length)];  %2
    R(3,1:2)=[r+double(y1>Height) c+double(x2>Length)];  %3
    R(4,1:2)=[r+double(y2>Height) c+double(x2>Length)]; %4
    R=unique(R,'rows');
    Z=zeros(4,2);
    Z(1:size(R,1),:)=R;
    for Nimgs=1:4
        ETopts.DATA{t}.R(index,Nimgs)=Z(Nimgs,1);
        ETopts.DATA{t}.C(index,Nimgs)=Z(Nimgs,2);
    end
    ETopts.DATA{t}.t(index)=ETopts.timepoint;

    %% bounding box
    ETopts.DATA{t}.BoundingBox(index,1:4)=STATS_cells(i).BoundingBox;
    ETopts.DATA{t}.Perimeter(index)=STATS_nuclii(i).Perimeter;
    ETopts.DATA{t}.MajorAxisLength(index)=STATS_nuclii(i).MajorAxisLength;
    ETopts.DATA{t}.MinorAxisLength(index)=STATS_nuclii(i).MinorAxisLength;
    ETopts.DATA{t}.Neighbours{index}=FindNeighbourCells(L,i)+ETopts.DATA{t}.NumOfCells;
    % CENTROID
    ETopts.DATA{t}.RelativeCenter(index,2)=STATS_nuclii(i).Centroid(2);
    ETopts.DATA{t}.RelativeCenter(index,1)=STATS_nuclii(i).Centroid(1);
    ETopts.DATA{t}.AbsoluteCenter(index,2)=ETopts.DATA{t}.RelativeCenter(index,2)+(ETopts.filename(2)-1)*512;
    ETopts.DATA{t}.AbsoluteCenter(index,1)=ETopts.DATA{t}.RelativeCenter(index,1)+(ETopts.filename(1)-1)*672;


    % RUNNING INDEX
    ETopts.DATA{t}.ID(index)=index;
    ETopts.DATA{t}.NucSolidity(index)=STATS_nuclii(i).Solidity;
    ETopts.DATA{t}.CellSolidity(index)=STATS_cells(i).Solidity;
    ETopts.DATA{t}.CellPerimeter(index)=STATS_cells(i).Perimeter;
    ETopts.DATA{t}.CellMajorAxis(index)=STATS_cells(i).MajorAxisLength;
    ETopts.DATA{t}.CellMinorAxis(index)=STATS_cells(i).MinorAxisLength;
    ETopts.DATA{t}.CellEccentricity(index)=STATS_cells(i).Eccentricity;
    ETopts.DATA{t}.CellOrientation(index)=STATS_cells(i).Orientation;
    ETopts.DATA{t}.NucEccentricity(index)=STATS_nuclii(i).Eccentricity;
    ETopts.DATA{t}.NucOrientation(index)=STATS_nuclii(i).Orientation;
    %% is nucl;ii out of focus? alg 1
    % slope of the dist (should be large negate values if out of focuis
    Nn=hist(Organelles.nucleus(STATS_nuclii(i).PixelIdxList));
    pft=polyfit(1:length(Nn),Nn,1);
    ETopts.DATA{t}.ImageFocus(index)=pft(1);
    %% is nucleus out of focus? alg 2
    % slope of image intensity decrease is fit to gauss. Shold be close to 1 (r square)
    %     imlittle=imcrop(Organelles.nucleus,STATS_nuclii(i).BoundingBox);
    %     ctr=round(size(imlittle)/2-1);
    %     Line=ctr(1):(2*ctr(1)-1);
    %     vals=imlittle(Line,ctr(2));
    %     disp([r c i])
    %     [fresult,gof]=fit((1:length(vals))',vals,'gauss1');
    %     if isempty(gof)
    %         OutOfFocusScore=0;
    %     else
    %         OutOfFocusScore=gof.adjrsquare;
    %     end
    %
    %     ETopts.DATA{t}.ImageFocus2(index)=OutOfFocusScore;
end

%% nuclii
for ch=1:length(IM)   % fluorescent channals
    stats = regionprops(Lregions.Nuclii,IM{ch},'MeanIntensity','area'  );
    for ii=1:length(stats)
        index=ETopts.DATA{t}.NumOfCells+ii;
        ETopts.DATA{t}.Nuclii.Area{ch}(index)=stats(ii).Area;
        ETopts.DATA{t}.Nuclii.MeanIntensity{ch}(index)=stats(ii).MeanIntensity;
        ETopts.DATA{t}.Nuclii.Intensity{ch}(index)=stats(ii).MeanIntensity*stats(ii).Area;
        ETopts.DATA{t}.FileNames(index,1:2)=ETopts.filename;
    end
end

%% cells
for ch=1:length(IM)   % fluorescent channals
    stats = regionprops(Lregions.Cells,IM{ch},'MeanIntensity','area' );
    for ii=1:length(stats)
        index=ETopts.DATA{t}.NumOfCells+ii;
        ETopts.DATA{t}.Cells.Area{ch}(index)=stats(ii).Area;
        ETopts.DATA{t}.Cells.MeanIntensity{ch}(index)=stats(ii).MeanIntensity;
```

```
        ETopts.DATA{t}.Cells.Intensity{ch}(index)=stats(ii).MeanIntensity*stats(ii).Area;
        if length(ETopts.XLS.SpottedChannal)>=t
            if ETopts.XLS.SpottedChannal(t)-1==ch
                granulii=IM{ch}-imopen(IM{ch},strel('disk',5,0));
                granulii(granulii./IM{ch}>0.9)=0;
                ETopts.DATA{t}.Cells.granulii{ch}(index)=sum(granulii(L==ii));
            end
        end

    end
end

%% centrosomes
if isfield(Lregions,'Pericentrin')
    for ch=1:length(IM)   % fluorescent channals
        stats = regionprops(Lregions.Pericentrin.loc,imfilter(IM{ch},fspecial('gaussian',10,2)),'MeanIntensity' );
        for ii=1:length(stats)
            index=ETopts.DATA{t}.NumOfCells+ii;
            [xx,yy]=find(Lregions.Pericentrin.loc==ii);
            if length(xx)==2
                ETopts.DATA{t}.Pericentrin.distance{ch}(index)=sqrt(diff(xx)^2+diff(yy)^2);
            else
                ETopts.DATA{t}.Pericentrin.distance{ch}(index)=0;
            end
            ETopts.DATA{t}.Pericentrin.MeanIntensity{ch}(index)=stats(ii).MeanIntensity;
        end
    end
end

%% focci
if isfield(Lregions,'focci')
    for ii=1:length(stats)
        index=ETopts.DATA{t}.NumOfCells+ii;
        ETopts.DATA{t}.focci.number_of(index)=length(find(Lregions.focci==ii));
        ETopts.DATA{t}.focci.brightness(index)=mean(Organelles.focci(find(Lregions.focci==ii)));
    end
end

%% cilia
if isfield(Lregions,'cilia')
    % initializing
    Zcilia=zeros(max(L(:)),1);
    Dcilia=Zcilia;
    DArea=Zcilia;
    DEccentricity=Zcilia;

    % identifying cells with cilia
    cilia=Lregions.cilia>0;
    Cells=Lregions.Cells>0;
    cilia=cilia&Cells;
    CiliatedCells=imreconstruct(cilia,Cells);
    CiliatedCells=CiliatedCells.*Lregions.Cells;
    U=unique(CiliatedCells)';
    U=U(2:end);

    cilia=(cilia>0).*CiliatedCells;
    stats = regionprops(cilia,'Eccentricity','area' );

    % entering data
    for ii=U
        Zcilia(ii)=1;
        DEccentricity(ii)=stats(ii).Eccentricity;
        DArea(ii)=stats(ii).Area;
    end
    if isfield(ETopts.DATA{t},'cilia')
        ETopts.DATA{t}.cilia.exists=[ETopts.DATA{t}.cilia.exists ; Zcilia];
        ETopts.DATA{t}.cilia.Eccentricity=[ETopts.DATA{t}.cilia.Eccentricity ; DEccentricity];
        ETopts.DATA{t}.cilia.Area=[ETopts.DATA{t}.cilia.Area ; DArea];
    else
        ETopts.DATA{t}.cilia.exists=Zcilia;
        ETopts.DATA{t}.cilia.Eccentricity=DEccentricity;
        ETopts.DATA{t}.cilia.Area=DArea;
    end
end


% Collecting intensity of nuclii
stats = regionprops(Lregions.Nuclii,Organelles.nucleus,'MeanIntensity','area' );
% Normalizing Nuclii data
if ~isempty(stats)
    for ii=1:length(stats)
        NucData(ii)=stats(ii).Area*stats(ii).MeanIntensity;
    end

%     [f,s]=ksdensity(NucData);
%     [mx,mxi]=max(f);
%     NucData=NucData/s(mxi);

    ETopts.DATA{t}.DNA(Nt+1:Nt+length(NucData))=NucData;
end

% Updating number of cells
if isfield(ETopts.DATA{t},'Cells')
    ETopts.DATA{t}.NumOfCells=length(ETopts.DATA{t}.Cells.Area{1});
else
```

```
    return
end

%% recording pictures of individual cells
ETopts.NumOfProcessedIms=ETopts.NumOfProcessedIms+1;
if ETopts.outputpic{1}>0 && ETopts.outputpic{1}>ETopts.NumOfPics && ETopts.NumOfProcessedIms>5 && mod(ETopts.counter,1)==0
    [N,X]=hist(ETopts.DATA{t}.DNA,100);
    [mx,mxi]=max(N);
    X0=X(mxi);
    dna=ETopts.DATA{t}.DNA/X0;
    ETopts.NumOfPics=ETopts.NumOfPics+1;
    for ipics=1:length(stats)
        Szpc=ETopts.DATA{t}.Cells.Intensity{2}(end-ipics);
        Gempc=log(ETopts.DATA{t}.Nuclii.Intensity{1}(end-ipics));
        CellPic=NormalizeImage(imcrop(Organelles.prot,ETopts.DATA{t}.BoundingBox(end-ipics,:)));
        Dpic=dna(end-ipics);
        imwrite(CellPic,[ETopts.XLS.OutDir{t} '\Size_' num2str(Szpc) 'Gem_' num2str(Gempc) 'DNA_' num2str(Dpic) '.bmp'],'bmp')
    end
end


function [C,N]=CropSegmentedImage(ETopts,Cells,Nuclii)
t=ETopts.timepoint;
C{1,1}=Cells(1:ETopts.ImageSize.H,1:ETopts.ImageSize.L);
C{1,2}=Cells(1:ETopts.ImageSize.H,1+ETopts.ImageSize.L:end);
C{2,1}=Cells(1+ETopts.ImageSize.H:end,1:ETopts.ImageSize.L);
C{2,2}=Cells(1+ETopts.ImageSize.H:end,1+ETopts.ImageSize.L:end);

N{1,1}=Nuclii(1:ETopts.ImageSize.H,1:ETopts.ImageSize.L);
N{1,2}=Nuclii(1:ETopts.ImageSize.H,1+ETopts.ImageSize.L:end);
N{2,1}=Nuclii(1+ETopts.ImageSize.H:end,1:ETopts.ImageSize.L);
N{2,2}=Nuclii(1+ETopts.ImageSize.H:end,1+ETopts.ImageSize.L:end);

function [yn]=iseven(n)
yn=(-1)^n==1;

function [ETopts,Cmat,Nmat]=buildSegmentedImage(ETopts,C,N,c,r)
t=ETopts.timepoint;
if r==1 && c==1
    Cmat=cell(ETopts.XLS.rows(t),ETopts.XLS.columns(t));
    Nmat=cell(ETopts.XLS.rows(t),ETopts.XLS.columns(t));
else
    Cmat=ETopts.Cmat;
    Nmat=ETopts.Nmat;
end

for RR=0:1
    for CC=0:1
        if isempty(Cmat{RR+r,CC+c})
            Cmat{RR+r,CC+c}=logical(C{RR+1,CC+1});
            Nmat{RR+r,CC+c}=logical(N{RR+1,CC+1});
        else
            Cmat{RR+r,CC+c}=logical(C{RR+1,CC+1})|Cmat{RR+r,CC+c};
            Nmat{RR+r,CC+c}=logical(N{RR+1,CC+1})|Nmat{RR+r,CC+c};
        end
    end
end
ETopts.Cmat=Cmat;
ETopts.Nmat=Nmat;

function [IMt]=SumImages(IM_n)
IMt=IM_n{1};
for i=2:length(IM_n)
    IMt=IMt+IM_n{i};
end


function [D,Locations]=SegmentPericentrin(IM,Cells)
% D - D(i) if the distance between the centrosomes in cell i.
IM2=imopen(IM,strel('square',5));
L=bwlabel(imclearborder(Cells));
BW=imregionalmax(IM);

[x,y]=find(BW);
I=sub2ind(size(IM),x,y);
Iv=IM(I);
Ivb=IM2(I);
IL=L(I);
Iloc=zeros(size(IL));
D=zeros(max(L(:)),1);
for index=1:max(L(:))

    i=find(IL==index);
    [mx,imx]=max(Iv(i));
    i0=i(imx);

    i2=find(Iv>Iv(i0)*0.5 & IL==index);
    i_thr=(Iv(i2)-Ivb(i2))./Ivb(i2)>0.5;
    i2=i2(find(i_thr));
    if length(i2)>2
        [srt,isrt]=sort(Iv(i2),'descend');
        i2=i2(isrt(1:2));
        D(index)=sqrt(diff(x(i2))^2+diff(y(i2))^2);
    end
    Iloc(i2)=1;
```

```
end

Iremain=find(Iloc);
x=x(Iremain);
y=y(Iremain);
I=sub2ind(size(IM),x,y);
Z=zeros(size(IM));
Z(I)=1;

% imshow(IM,[])
% hold on
% plot(y,x,'.r')
%

Locations=Z;


function [Regions,IOL1,IOL2]=CleaningRegions(Regions)

Regions.Cells=imclearborder(Regions.Cells);
%% removing regions that are not in cells


Regions.Nuclii=Regions.Nuclii&Regions.Cells;


%% removing cells that don't have all regions.

Regions.Cells=imreconstruct(Regions.Nuclii>0,Regions.Cells>0);
Regions.Nuclii=Regions.Nuclii&Regions.Cells;

%%
Regions.Cells=bwareaopen(Regions.Cells,50);
% Regions.Cells=imopen(Regions.Cells,strel('disk',2));
Regions.Nuclii=bwareaopen(Regions.Nuclii,50);
Regions.Nuclii=imopen(Regions.Nuclii,strel('disk',2));
%% removing large nuclii
Regions.Nucli=Regions.Nuclii&~bwareaopen(Regions.Nuclii,2000);

%% removing cells without nucs
J=Regions.Nucli&Regions.Cells;
Regions.Cells=imreconstruct(J,Regions.Cells);
Regions.Nuclii=imreconstruct(J,Regions.Nuclii);
%% creating picturs
IOL1=imdilate(bwperim(Regions.Cells),1);
IOL2=imdilate(bwperim(Regions.Nuclii),1);

function [ETopts]=ETAddFieldsv4(ETopts)
% [ETopts]=CorrectionCoeficients(ETopts);
for i=1:length(ETopts.DATA)
    if ~isempty(ETopts.DATA{i})
        if ~isfield(ETopts.DATA{i},'Cells')
            continue
        end
        [ETopts]=addfields(ETopts,i);
    end
end

function [ETopts]=addfields(ETopts,t)

for i=1:length(ETopts.DATA{t}.Cells.MeanIntensity)
    [ff,ss]=ksdensity(ETopts.DATA{t}.DNA);
    [mx,mxi]=max(ff);
    StepSize=ss(mxi)/100;
    [ff,ss]=ksdensity(ETopts.DATA{t}.DNA,0:StepSize:400*StepSize);
    [mx,mxi]=max(ff);
    ETopts.DATA{t}.DNA=ETopts.DATA{t}.DNA/ss(mxi);

    ETopts.DATA{t}.cytplsmInt{i}=ETopts.DATA{t}.Cells.Intensity{i}-ETopts.DATA{t}.Nuclii.Intensity{i};
    ETopts.DATA{t}.cytplsmArea{i}=ETopts.DATA{t}.Cells.Area{i}-ETopts.DATA{t}.Nuclii.Area{i};
    ETopts.DATA{t}.cytNormInt{i}=ETopts.DATA{t}.cytplsmInt{i}./ETopts.DATA{t}.cytplsmArea{i};

    ETopts.DATA{t}.Nuclearization{i}=ETopts.DATA{t}.Nuclii.Intensity{i}./ETopts.DATA{t}.Cells.Intensity{i};
    Norm=ETopts.DATA{t}.Nuclii.Area{i}./ETopts.DATA{t}.Cells.Area{i};
    ETopts.DATA{t}.NormNuclearization{i}=ETopts.DATA{t}.Nuclearization{i}./Norm;
    ETopts.DATA{t}.NC_conc_ratio{i}=ETopts.DATA{t}.Nuclii.MeanIntensity{i}./ETopts.DATA{t}.Cells.MeanIntensity{i};
    ETopts.DATA{t}.NC_size_ratio=Norm;
end
% ETopts.DATA{t}.p65=ETopts.DATA{t}.Nucleus{p65}-ETopts.DATA{t}.Cell{p65};

function [ETopts]=num_of_channals(ETopts)

for t=1:ETopts.XLS.NumId(end)
    N=0;
    for c=1:5
        if ~isempty(ETopts.XLS.channal_id{t,c});
            N=N+1;
        end
    end
    ETopts.XLS.num_channals(t)=N;
end
function [ETopts]=find_tile_coordinates(ETopts)
t=ETopts.timepoint;
if strcmpi(ETopts.XLS.software(t),'elements')
    if ~isempty(ETopts.XLS.Last_image(t))
```

```
        NumberOfImages=ETopts.XLS.Last_image(t);
        ETopts.TileCoordinates=(1:NumberOfImages)';
    else
        NumberOfImages=ETopts.XLS.columns(t)*ETopts.XLS.rows(t);
        TileCoordinates=reshape(1:NumberOfImages,ETopts.XLS.columns(t),ETopts.XLS.rows(t))';
        TileCoordinates(2:2:end,:)=fliplr(TileCoordinates(2:2:end,:));
        ETopts.TileCoordinates=TileCoordinates;
    end
elseif strcmpi(ETopts.XLS.software(t),'metamorph')

    [TileCoordinates]=find_tile_coordinates_MM(ETopts);
    ETopts.TileCoordinates=TileCoordinates;
    ETopts.XLS.columns(t)=size(TileCoordinates,2);
    ETopts.XLS.rows(t)=size(TileCoordinates,1);
end


function [TileCoordinates,WaveLengths]=find_tile_coordinates_MM(ETopts)
% find the tile coordinates for metamorph generated data
t=ETopts.timepoint;

Directory=ETopts.XLS.InDir{t};
SlideID=ETopts.XLS.SlideNames{t};


% reads in the file of xxx.scan
% this file provieds positions of the images on the slide
addpath(Directory)
fid = fopen([Directory '\' SlideID '.scan']);
C = textscan(fid,'%s','delimiter','\n');
fclose(fid);


counter=0;
WLc=0;
for i=1:length(C{1})
    txt=C{1}{i};
    [mat] = regexp(txt, 'Stage(\d+).*Row(\d+).*Col(\d+)', 'tokens');
    if ~isempty(mat)
        counter=counter+1;
        s(counter)=str2num(mat{1}{1});
        Row(counter)=str2num(mat{1}{2});
        Col(counter)=str2num(mat{1}{3});
    else
        [WL] = regexp(txt, '"WaveName(\d)", "Ran (\w+)"', 'tokens');
        if ~isempty(WL)
            WLc=WLc+1;
            WaveLengths{WLc}=[WL{1}{1} ' - '  WL{1}{2}];
        end
    end

end

TileCoordinates=zeros(max(Row),max(Col));
for i=1:(max(Row)+1)
    for j=1:(max(Col)+1)
        TileCoordinates(i,j)=s(find(Row==i-1 & Col==j-1));
    end
end


function [NOC]=NumOfChannals(ETopts,t)
NOC=1;
while NOC<=5 && ~isempty(ETopts.XLS.channal_id{t,NOC})
    NOC=NOC+1;
end
NOC=NOC-1;

function [ETopts]=NumOfDigits(ETopts)
t=ETopts.timepoint;
if ~isfield(ETopts,'LS')
    LS=ls([ETopts.XLS.InDir{t} '\*' ]);
    ETopts.LS=LS;
end
LS=ETopts.LS;
I=strmatch([ETopts.XLS.SlideNames{t}],LS);
[st,en]=regexp(LS(I(1),:),[ETopts.XLS.SlideNames{t} '(\d+)c']);
ETopts.NumberOfDigits=en-length(ETopts.XLS.SlideNames{t})-1;

function [ETopts]=NumOfDigits_newelements(ETopts)
t=ETopts.timepoint;
if ~isfield(ETopts,'LS')
    LS=ls([ETopts.XLS.InDir{t} '\*' ]);
    ETopts.LS=LS;
end
LS=ETopts.LS;
I=strmatch([ETopts.XLS.SlideNames{t}],LS);
[st,en]=regexp(LS(I(1),:),[ETopts.XLS.SlideNames{t} '_x(\d+)_y']);
ETopts.NumberOfDigits=en-length(ETopts.XLS.SlideNames{t})-4;

function [ETopts]=NumOfDigits_newelements4slides(ETopts)
t=ETopts.timepoint;
if ~isfield(ETopts,'LS')
    LS=ls([ETopts.XLS.InDir{t} '\*' ]);
    ETopts.LS=LS;
```

```
end
LS=ETopts.LS;
I=strmatch([ETopts.XLS.SlideNames{t}],LS);
[st,en]=regexp(LS(I(1),:),['c(\d+)_r']);
ETopts.NumberOfDigits=en-st-2;

function [ETopts,spX,spY]=CollectFileIndices_4slide(ETopts,t)
if ~isfield(ETopts,'LS4slide')
    LS=ls([ETopts.XLS.InDir{t} ]);
    ETopts.LS=LS;
else
    LS=ETopts.LS4slide;
end
ETopts.LS4slide=LS;
counter=0;
ETopts.NumOfPics=0;
NumChannals=0;
for i=1:size(LS,1)
    tk=regexp(LS(i,:),['^' ETopts.XLS.SlideNames{t} '_s(\d+)_c(\d+)_r(\d+)_(\w+).tif'],'tokens');
    if ~isempty(tk) && str2num(tk{1}{1})==t
        counter=counter+1;
        spX(counter)=str2num(tk{1}{2});
        spY(counter)=str2num(tk{1}{3});
        NumChannals=NumChannals+1;
        if NumChannals==1
            ETopts.ChannalSuffix{1}=tk{1}{4};
        elseif strcmpi(tk{1}{4},ETopts.ChannalSuffix{1})==0 && NumChannals<=4
            ETopts.ChannalSuffix{NumChannals}=tk{1}{4};

        end
    end
end
spXspY=[spX' spY'];
spXspY=unique(spXspY,'rows');
spX=spXspY(:,1)';
spY=spXspY(:,2)';

function [ETopts,spX,spY]=CollectFileIndices(ETopts,t)
LS=ls([ETopts.XLS.InDir{t} ]);
ETopts.LS=LS;
counter=0;
ETopts.NumOfPics=0;
for i=1:size(LS,1)
    tk=regexp(LS(i,:),['^' ETopts.XLS.SlideNames{t} '_x(\d+)_y(\d+).tif'],'tokens');
    if ~isempty(tk)
        counter=counter+1;
        spX(counter)=str2num(tk{1}{1});
        spY(counter)=str2num(tk{1}{2});
    end
end
```

1 Wasserman, L. *All of statistics : a concise course in statistical inference*. (Springer, 2004).
2 Wasserman, L. *All of nonparametric statistics*. (Springer, 2006).
3 Taylor, J. R. *An introduction to error analysis : the study of uncertainties in physical measurements*. 2nd edn, (University Science Books, 1997).
4 Botev, Z. I., Grotowski, J. F. & Kroese, D. P. Kernel density estimation via diffusion *Annals of Statistics* **38**, 2916-2957 (2010).
5 Bowman, A. W. & Azzalini, A. *Applied smoothing techniques for data analysis : the kernel approach with S-Plus illustrations*. (Clarendon Press ;

Oxford University Press, 1997).
6 Collins, J. F. & Richmond, M. H. Rate of growth of Bacillus cereus between divisions. *Journal of general microbiology* **28**, 15-33 (1962).
7 Anderson, E. C., Bell, G. I., Petersen, D. F. & Tobey, R. A. Cell growth and division. IV. Determination of volume growth rate and division probability. *Biophys J* **9**, 246-263, doi:10.1016/S0006-3495(69)86383-6 (1969).
8 Tzur, A., Kafri, R., LeBleu, V. S., Lahav, G. & Kirschner, M. W. Cell growth and size homeostasis in proliferating animal cells. *Science* **325**, 167-171, doi:10.1126/science.1174294 (2009).
9 Son, S. *et al.* Direct observation of mammalian cell growth and size regulation. *Nat Methods* **9**, 910-912, doi:10.1038/nmeth.2133 (2012).