Supplementary information

Supplementary Methods

<u>Multi-type branching process equations</u>

Taking into account the assumptions described in the *Quick guide* from the manuscript, the **expected number of cells** at any time after treatment initiation was derived in previous publications (see Chakrabarti and Michor 2017 and its supplementary material) and is given by:

$$\langle n_i(t)\rangle = N_0 e^{-\int_0^t [b_0(s)\cdot(1-\sum_z u_z)-d_0(s)]\,ds} \qquad for\ i=0$$
$$= \frac{N_i + \int_0^t \left[e^{-\int_0^s [b_i(\tau)-d_i(\tau)]d\tau}\cdot u_i \cdot b_0(s)\cdot N_0 \cdot e^{\int_0^s [b_0(\tau)\cdot(1-\sum_z u_z)-d_0(\tau)]d\tau}\right] ds}{e^{-\int_0^t [b_i(s)-d_i(s)]ds}} \qquad for\ i>0 \qquad \text{(S1)}$$

where $i=0$ denotes the original sensitive cell type and $i=1,\dots,N$ denotes the N different resistant cell types. Birth, death and mutation rates are denoted by b_i , d_i and u_i respectively and the initial number of sensitive and resistant cells is represented by N_0 and N_i. Since the mutation rate constant is always much less than 1 ($u_i \ll 1$), the sum over the mutation rates of all N-resistant types $\sum_{z=1}^{N} u_z$ can be removed from the equation.

To model cross resistance to drugs, each of the resistant cell types are allowed to gain additional mutations and become resistant to more than one compound. A resistant type *i* cell can thus further mutate to form a type *ia* cell with probability u_{ia}. The first order birth and death rates of the *ia* cell are given by b_{ia} and d_{ia}. Thus, the time evolution equations of the expected number of cells in each clone are given by:

$$\langle n_i(t)\rangle = N_0 e^{-\int_0^t [b_0(s)\cdot(1-\sum_z u_z)-d_0(s)]\,ds} \qquad for\ i=0$$
$$= \frac{N_i + \int_0^t \left[e^{-\int_0^s [b_i(\tau)(1-\sum_{cr} u_{cr})-d_i(\tau)]d\tau}\cdot u_i \cdot b_0(s)\cdot N_0 \cdot e^{\int_0^s [b_0(\tau)\cdot(1-\sum_z u_z)-d_0(\tau)]d\tau}\right] ds}{e^{-\int_0^t [b_i(s)(1-\sum_{cr} u_{cr})-d_i(s)]ds}} \qquad for\ i>0 \qquad \text{(S2)}$$

$$\langle n_{ia}(t)\rangle = \frac{N_{ia} + \int_0^t \left[e^{-\int_0^s [b_{ia}(\tau)-d_{ia}(\tau)]d\tau}\cdot u_{ia}\cdot b_i(s)\cdot n_i(s)\right] ds}{e^{-\int_0^t [b_{ia}(s)-d_{ia}(s)]ds}} \qquad for\ i>0 \qquad \text{(S3)}$$

Probability of developing resistance

The probability that there exists at least one resistant cell of any type ($i=1,\dots,N$) at time T after treatment initiation was also derived in previous works (Foo and Michor 2010) and is given by:

$$P_R(T) = 1 - e^{[-\int_0^T \left(\sum_{i=1}^{N} N_0 \cdot e^{\int_0^t [b_0(\tau)(1-\sum_z u_z)-d_0(\tau)]d\tau} \cdot b_0(t)\cdot u_i \cdot (1-P_{ext,i}(t,T))\right) dt]} \qquad \text{(S4)}$$

$$\text{where} \quad P_{ext}(t,T) \equiv \frac{\int_0^{T-t} d_i(\tau+t)\cdot\left(e^{\int_0^{\tau} d_i(\eta+t)-b_i(\eta+t)d\eta}\right)d\tau}{1+\int_0^{T-t} d_i(\tau+t)\cdot\left(e^{\int_0^{\tau} d_i(\eta+t)-b_i(\eta+t)d\eta}\right)d\tau}$$

The expressions above cannot be solved analytically for time-dependent growth and death rates, and therefore has to be solved numerically.

Net growth, birth and death rate estimation from data

In order to estimate the **net growth rate** of the cells, an exponential growth model based in the following ordinary differential equation (ODE) is defined:

$$\frac{dN}{dt} = \lambda_j \cdot N \tag{S5}$$

where N is the number of viable cells, λ_j is the first order net growth rate parameter and *j* represents the different drug concentrations analyzed in the experiment. This model implies that each concentration has its own net growth rate associated. The analytical solution of this equation is:

$$N(t) = N_0 \cdot e^{\lambda_j \cdot t} \tag{S6}$$

where N_0 is the number of viable cells at time 0.

In order to estimate the **death rate** from apoptosis assay data where the number of death cells is counted over time and different drug concentrations, the following ODE was defined:

$$\frac{dN_d}{dt} = d_j \cdot N \tag{S7}$$

where N_d is the total number of death cells and d_j is the first order death rate parameter for the different drug concentrations tested. As the equation for N has been already defined, the following analytical solution for N_d can be obtained:

$$N_d(t) = \int_0^t \left(d_j \cdot N_0 \cdot e^{\lambda_j \cdot t}\right) dt = \frac{N_0 \cdot d_j}{\lambda_j} \cdot \left(e^{\lambda_j \cdot t} - 1\right) \tag{S8}$$

As the value of λ_j is already known from the previous exercise and the values for N_0 and N_d over time and different drug concentrations can be obtained from the dataset, the different d_j parameters can be easily obtained using a least square minimization. Finally, as the net growth rate parameter λ_j is the difference between cell proliferation and death, the birth rates b_j for the different drug concentrations can be obtained from the values of λ_j and d_j as $b_j = \lambda_j + d_j$.

Non-linear concentration-response curves

Many drug concentration-effect relationships are described by nonlinear sigmoid models, which are characterized by a sigmoidal or "S" shape. The four-parameter logistic equation (or Hill equation) is one of the most common approaches to describe single-agent concentration-response curves of this type (Ritz et al. 2015). In this work, the responses being analyzed are the birth and death rates of sensitive and resistant cell lines. This model is defined by:

$$E(C) = E_{max} + \frac{E_0 - \mathrm{E}_{max}}{1 + (\frac{C}{EC_{50}})^h} \tag{S9}$$

where E is the first order rate constant obtained at concentration C, E_{max} is the maximum effect of the drug on the birth rate, E_0 is the intrinsic birth rate when no drug is present, EC_{50} is the inflection point of the curve and represents the concentration corresponding to 50% of the maximum effect and h is the shape parameter linked to the steepness of the curve. The five-parameter logistic model is an extension of this equation for fitting asymmetrical data (Ritz et al. 2015) as it adds an asymmetry factor parameter defined as f in the previous equation:

$$E(C) = \mathrm{E}_{max} + \frac{E_0 - E_{max}}{[1 + \left[(\frac{C}{EC_{50}})^h\right]^f} \tag{S10}$$

These models can be generalized to any measure of drug exposure (e.g., dose, plasma concentration, or area under the concentration vs time curve).

Apart from these two examples, ACESO incorporates other nonlinear dose-response models like the three and four-parameter Gompertz model, Weibull models, etc. for parameter estimation in order to characterize the effects of the drugs on the growth and death rates of cells. A simple linear model can also be fitted to the data.

Non-parametric models to estimate the effect of drug combination data

To estimate the regression surface of the *in-vitro* drug combination data, nonparametric fitting methods like Generalized Additive Models (GAMs) (Hastie 2017) or locally weighted scatterplot smoothing (loess) regression (Cleveland et al. 1991) were used. GAMs are a nonparametric extension of generalized linear models (GLMs) (McCullagh & Nelder 1989). Here, a general nonparametric function (e.g. cubic splines) that relates the predicted drug effect values to the drug concentrations is defined (Troconiz et al. 1994). Loess is a particular implementation of local polynomial smoothing which fits simple models to localized subsets of the data. As nonparametric methods, GAMs and loess regression are data-driven rather than model-driven; that is, they allow the data to determine the shape of the response curves. Thus, these models are used to describe the relation between drug concentrations and the growth and death rates of cancer cells without assuming the data must fit some distribution shape. Although these methods are very flexible, they are not biologically interpretable and considerably increase the computation time needed for the simulations. Even so, they are a very powerful exploratory tool which often shows relatively complex relations between a dependent variable and more than one independent variable without being limited by the shapes available in a parametric model.

This approach allowed us to incorporate drug combination data into our evolutionary framework and explore the effect of multidrug dosing schedules in the evolution of cancer cells.

Assessing drug synergy/antagonism

In order to quantify the degree of synergy/antagonism between two compounds, the typical approach is to compare their measured combination effect to a null reference model of no interaction, i.e. the expected response assuming no interaction between the two compounds. If the combination response is greater than what is expected by the reference model, the combination is classified as synergistic, while antagonism is defined when the combination produces less than the expected effect.

There are several well-known conventional approaches that define different null models to assess drug synergy/antagonism. The Loewe Additivity model (Loewe 1953) is one of the most commonly used models to quantify a zero-interactive state for the combination of two drugs. This model is based on the assumption that a drug cannot interact with itself and defines synergy/antagonism as a combined inhibitory effect that is greater/lower than the sum of the individual effects of the drugs. The general equation of this model is:

$$\frac{da}{DA} + \frac{db}{DB} = 1$$

where da and db are the dose (or concentrations) of drug A and B in the combination that produce an effect E_{AB} and DA and DB represent the single doses of drug A and B necessary to reach the same effect E_{AB}. Isobole analysis (Tallarida 2006; Cokol et al. 2011) and the Greco model (Greco et al. 1990) are methods derived from this equation.

Loewe additive model response at any combined concentration is calculated from the sigmoidal fits of the single-agent response curves. In order to obtain the concentrations of each drug given as a single agent that elicits an effect E_{AB}, an inverse hill equation (h^{-1}) with parameters obtained from the individual dose-response curves using a common baseline value (effect when there is no drug concentration) is employed:

$$\frac{da}{h_A{}^{-1}(E_{AB})} + \frac{db}{h_B{}^{-1}(E_{AB})} = 1$$

$$h_A{}^{-1}(E_{AB}) = C_A = EC_{50_A}\left(\frac{E_{max_A} - E_0}{E_{AB} - E_0} - 1\right)^{1/b_A} \quad \text{(and equivalent for } h_B{}^{-1}(E_{AB})\text{)}$$

Classical Loewe additivity model assumes that the drugs in the combination have equal individual baseline and maximum effects. However, this method can be extended to account for different drug maximal responses (Van der Borght et al. 2017). We refer to this extended method as Generalized Loewe additivity model.

Highest Single Agent (HSA), also known as Gaddum's non-interaction model (Gaddum 1985), is another popular model which defines a "independent action" of the drugs when the predicted effect of a combination is that of the one most effective drug alone. The HSA zero interaction model predicts the combined effect E_{AB} for two single agents with effects E_A and E_B as:

$$E_{AB} = \max(E_A, E_B) \quad \text{(or } \min(E_A, E_B) \text{ if the monotherapy curves are decreasing)}$$

where E_A and E_B are measured on the monotherapy dose–response curve of drug A and B respectively. According to this model, any combined effect stronger than the effect of a single drug is called 'synergism' and a weaker effect 'antagonism'.

ACESO is limited to two interacting drugs for now. Calculating the effect of each treatment on the birth or death rate of the different cell population is a very complex task when more than two drugs are taken into account because of the difficulty in discerning the effect of each drug plus the synergistic/antagonistic interaction between them.

Ideal datasets

Cell viability and death in vitro assays are required for both sensitive and resistance cells lines used in the model to determine the growth and death rates for each cell with and without the presence of drugs. One example of viability assay to determine the proliferation rates of various cell types is the MTS assay. Here, the number of viable cells is measured over time which is then used to identify the proliferation rate of an exponentially growing population by performing a linear regression on log-transformed data over time. In order to automatically estimate these rates with ACESO, the ideal cell viability dataset should have the following columns in the text file:

Mandatory columns:

- Time: time point.
- Viable.cells: cell count data.
- CONC: drug concentration

Optional columns:

- Replicate: Technical replicate. If missing the program assumes that there is only 1 replicate.
- Cell.line: Name of the cell line. If missing the function introduces the name 'Cell line 1'.
- Type: numerical column specifying the cell type (0: sensitive cells, 1: resistant cell to drug A, etc.). If missing the program assumes Type=0, that is, a sensitive cell.
- … : other column names where additional information can be saved: drug concentration units, time units, sample id…

We show an example in the following table (with a black box highlighting the mandatory columns):

Time	Viable.cells	CONC	Cell.line	Replicate	Type	...
0	2751	0	PC-9	1	0	
0	2853	0	PC-9	2	0	
0	2902	0	PC-9	3	0	
48	5000	0	PC-9	1	0	
48	4863	0	PC-9	2	0	
48	5126	0	PC-9	3	0	
48	4601	0.5	PC-9	1	0	
48	4488	0.5	PC-9	2	0	
48	4712	0.5	PC-9	3	0	
60	4885	1	PC-9	1	0	
...	...	...	...	...	...	...

In the case where the effects of two drugs are analyzed, an additional column CONC2 need to be added, indicating the concentrations of the second drug:

Time	Viable.cells	CONC	CON2	Cell.line	Replicate	Type	...
0	2751	0	0	PC-9	1	0	
0	2853	0	0	PC-9	2	0	
0	2902	0	0	PC-9	3	0	
48	5000	0	0	PC-9	1	0	
48	4863	0	0	PC-9	2	0	
48	5126	0	0	PC-9	3	0	
48	4601	0.5	0	PC-9	1	0	
48	4488	0.5	0	PC-9	2	0	
48	4712	0.5	0	PC-9	3	0	
48	4885	0	0.5	PC-9	1	0	
48	4702	0	0.5	PC-9	2	0	
48	4992	0	0.5	PC-9	3	0	
...	...	...	...	...	...	...	...

The death rates of the different cell types can also be parameterized using apoptosis assays such as Annexin V/propidium iodide (PI) fluorescence-activated cell sorting (FACS) assays, where cells with positive Annexin V staining are considered to be dead cells. An ideal apoptosis assay dataset should have the following information saved in different columns:

Mandatory columns:

- Time: time point
- Death cell counts/fraction: fraction of dead cells or the total count of dead cells.
- CONC: drug concentration
- CONC2: In the case where the effects of two drugs are analyzed, an additional column need to be added, indicating the concentrations of the second drug.

Optional columns:

- Replicate: Technical replicate. If missing the program assumes that there is only 1 replicate.
- Cell.line: Name of the cell line. If missing the function introduces the name 'Cell line 1'.
- Type: numerical column specifying the cell type (0: sensitive cells, 1: resistant cell to drug A, etc.). If missing the program assumes Type=0, that is, a sensitive cell.
- … : other column names where additional information can be saved: drug concentration units, time units, sample id…

Additional results: optimum dosing strategies for TNBC

Alpelisib and lapatinib combination treatment

We considered combination therapy using neratinib and lapatinib (Fig. S4). We found that lapatinib had almost no effect on type-0 cells and there was no synergy between the two drugs. Therefore, the selection of the best therapy was mainly based on the dosing strategy selected for alpelisib. The effect of lapatinib on type-2 cells was also very small leading less than 10% difference between dosing schedules. For all the drug combinations we investigated, schedules including once-weekly pulses with resting periods and no maintenance dosing resulted in the highest expected number of tumor cells after treatment. Increasing the weekly pulse frequency to twice weekly or three times a week did not improve the results for most drug combinations, but for alpelisib and lapatinib, combining a continuous daily dosing of 300 mg alpelisib with a three-times a week high dose pulse of 3500 mg of lapatinib with four drug holidays (regimen 7) gave slightly better results. This observation may be due to the high lapatinib concentrations necessary to exert a substantial change in type-2 cell birth rate values (Supplementary Figure S1).

Dactolisib and trametinib combination treatment

We then investigated the effects of dactolisib and trametinib combination treatment. When investigating the growth rates of cells in response to these drugs, we found that trametinib was unable to affect large changes of trametinib-sensitive cell growth rates (Supplementary Fig. S5). Nevertheless, due to synergy effects, a more than 30% improvement on growth inhibition was achieved when we selected the most advantageous dosing regimen for type-0 cells (Fig. S5 regimen 8/9). However, this improvement was based on a difference of less than 1000 cells, which is negligible relative to the size of the entire tumor. For dactolisib, which is administered bi-daily (BID), a high-dose pulse of 800 mg BID with a continuous dosing of 200 mg BID probed to be slightly more powerful than the daily dosing of 300 mg BID for dactolisib-sensitive cells (3.4% improvement) but up to 26.2% more effective in the killing of type-0 cells, apart from using less drug amounts. Even thought, when comparing the total number of cells, the different dosing schedules looked similar (<5% difference) because there were predominantly type-2 cells left after one month of treatment period. Therefore, the best predicted schedules were either 8 or 9 based on a 4.8% improvement from the worst schedule (regimen 4). However, the total amounts of drugs used in regimen 4 were much lower compared to the rest of the schedules and, due to the small differences found between them, we hypothesize that this treatment strategy could cause similar tumor shrinkage with less adverse effects for the patients and costs for the hospital.

Dactolisib and lapatinib combination treatment

When investigating the combination of dactolisib and lapatinib, we found that they had antagonistic effects when applied to the BT-20 cell line. Due to this effect, decreasing the dose levels of one of the drugs improved the killing of type-0 cells but not the elimination of resistant cells.

References

[1] Chakrabarti, S., and Michor, F. (2017). Pharmacokinetics and Drug Interactions Determine Optimum Combination Strategies in Computational Models of Cancer Evolution. Cancer Res., 77(14):3908–21.

[2] Cleveland, W. S., Grosse, E., and Shyu, W. M. (1991). Local regression models. chapter 8 of statistical models in S (edited by JM chambers and TJ hastie), 309-376.

[3] Cokol, M., Chua, H. N., Tasan, M., Mutlu, B., Weinstein, Z. B., Suzuki, Y., Nergiz, M. E., Costanzo, M., Baryshnikova, A., Giaever, G., Nislow, C., Myers, C. L., Andrews, B. J., Boone, C., and Roth, F. P. (2011). Systematic exploration of synergistic drug pairs. Mol. Syst. Biol., 7:544.

[4] Foo, J. and Michor, F. (2010). Evolution of Resistance to Anti-Cancer Therapy during General Dosing Schedules. J. Theor. Biol., 263 (2): 179–88.

[5] Gaddum, J. H. (1985). Gaddum's Pharmacology. Oxford University Press, USA.

[6] Greco, W. R., Park, H. S., and Rustum, Y. M. (1990). Application of a new approach for the quantitation of drug synergism to the combination of cis-diamminedichloroplatinum and 1-β-Darabinofuranosylcytosine. Cancer Res.

[7] Hastie, T. J. (2017). Generalized additive models. In Statistical models in S, pages 249-307. Routledge.

[8] McCullagh, P. and Nelder, J. A. (1989). Generalized Linear Models, Second Edition. CRC Press.

[9] Ritz, C., Florent B., Jens C. S., and Gerhard, D. 2015. Dose-Response Analysis Using R. PloS One, 10 (12): e0146021.

[10] Tallarida, R. J. (2006). An overview of drug combination analysis with isobolograms. J. Pharmacol. Exp. Ther., 319(1):1-7.

[11] Troconiz, I. F., Sheiner, L. B., and Verotta, D. (1994). Semiparametric models for antagonistic drug interactions. J. Appl. Physiol., 76(5):2224-2233.

[12] Van der Borght, K., Tourny, A., Bagdziunas, R., Thas, O., Nazarov, M., Turner, H., Verbist, B., and Ceulemans, H. (2017). BIGL: Biochemically intuitive generalized loewe null model for prediction of the expected combined effect compatible with partial agonism and antagonism. Sci. Rep., 7(1):17935.

Supplementary Tables

Supplementary Table S1. Pharmacokinetic models and parameter estimates used for the simulations of the treatment effects.

Drug	Model	Source	Parameters	MTD
Alpelisib	1 compartment model with 1-order absorption	De Buck et al. 2014	CL=11.5 L/h; V=118 L; KA=0.784 h^{-1}; ALAG=0.489h	· 400 mg QD · 300mg QD in combination with other drugs
Neratinib	1 compartment model with 1-order absorption	*PK curve estimator* using data from Keyvanjah et al. 2017	CL=157.18 L/h; V=2402.7 L; KA=0.603 h^{-1}; ALAG=2.349h	240mg QD
Dactolisib	2 compartment model with 1-order absorption	*PK curve estimator* using data from Wise-Draper et al. 2017	CL=178.9 L/h; V=1906 L; KA=0.4 h^{-1}; CL_d=548 L/h; Vp=26457 L	1600mg QD / 30mg BID
Trametinib	2 compartment model with 1-order absorption	Ouellet et al. 2016	CL=4.91 L/h; V=214 L; KA=2.05 h^{-1}; Vp=568 L; CL_d=60 L/h	2 mg QD
Lapatinib	1 compartment model with 1-order absorption	Siegel-Lakhai et al. 2007 Stein et al. 2018	CL=28.9 L/h; V=1000 L; KA=0.95 h^{-1}; ALAG=0.25h	1500 mg QD
Selumetinib	2 compartment model with sequential 0 and 1-order absorption	Patel et al. 2017	KA=3.7 h^{-1}; D_1=0.622nmol/h; CL=13.5 L/h; V=32.6L; Vp=55L; CL_d=8.2L/h; ALAG=0.319h	· 100 mg BID · 75mg BID in combination with other drugs

CL: total drug clearance, V: apparent volume of distribution from the central compartment, KA: first-order absorption rate constant, ALAG: lag time associated with the absorption of the drug, CLd: distribution clearance, Vp: volume of distribution from the peripheral compartment, D_1: duration of the 0-order absorption, MTD: Maximally Tolerated Dose.

References:

· De Buck, et al. 2014. “Population Pharmacokinetics and Pharmacodynamics of BYL719, a Phosphoinositide 3-Kinase Antagonist, in Adult Patients with Advanced Solid Malignancies.” *British Journal of Clinical Pharmacology*.

· Keyvanjah, Kiana, et al. 2017. “Pharmacokinetics of Neratinib during Coadministration with Lansoprazole in Healthy Subjects.” *British Journal of Clinical Pharmacology*.

· Ouellet, et al. 2016. “Population Pharmacokinetics and Exposure–response of Trametinib, a MEK Inhibitor, in Patients with BRAF V600 Mutation-Positive Melanoma.” *Cancer Chemotherapy and Pharmacology*.
Patel, et al. 2017. “Population Pharmacokinetics of Selumetinib and Its Metabolite N-Desmethyl-Selumetinib in Adult Patients With Advanced Solid Tumors and Children With Low-Grade Gliomas.” *CPT: Pharmacometrics & Systems Pharmacology.*

· *Siegel-Lakhai, et al. 2007.* “Phase I Pharmacokinetic Study of the Safety and Tolerability of Lapatinib (GW572016) in Combination with Oxaliplatin/Fluorouracil/Leucovorin (FOLFOX4) in Patients with Solid Tumors.” *Clinical Cancer Research.*

· *Stein, et al. 2018.* “Mathematical Modeling Identifies Optimum Lapatinib Dosing Schedules for the Treatment of Glioblastoma Patients.” *PLoS Computational Biology*.

· Wise-Draper, et al. 2017. “A Phase Ib Study of the Dual PI3K/mTOR Inhibitor Dactolisib (BEZ235) Combined with Everolimus in Patients with Advanced Solid Malignancies.” T*argeted Oncology*.

Supplementary Figures

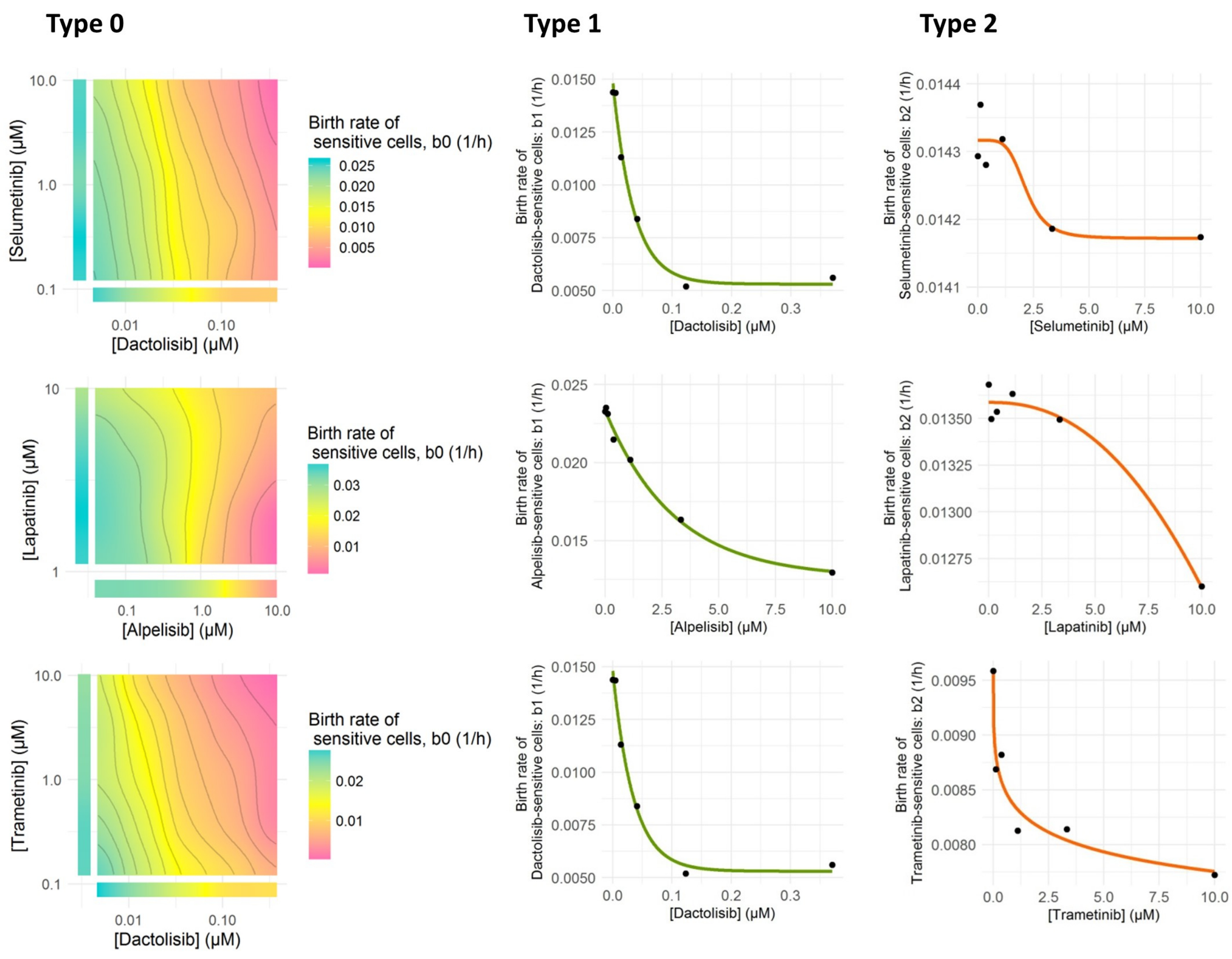


Figure S1. Model fitting results for some of the drug combination studies, where the surface from type-0 (sensitive to both drugs) cell birth rates is characterized with a Generalized Additive Model (GAM) and the birth rates from resistant cell types are described with different non-linear models included in ACESO (exponential, Weibull, log-logistic. . .). The specific model equation and parameter values for each curve are summarized in Table 1.

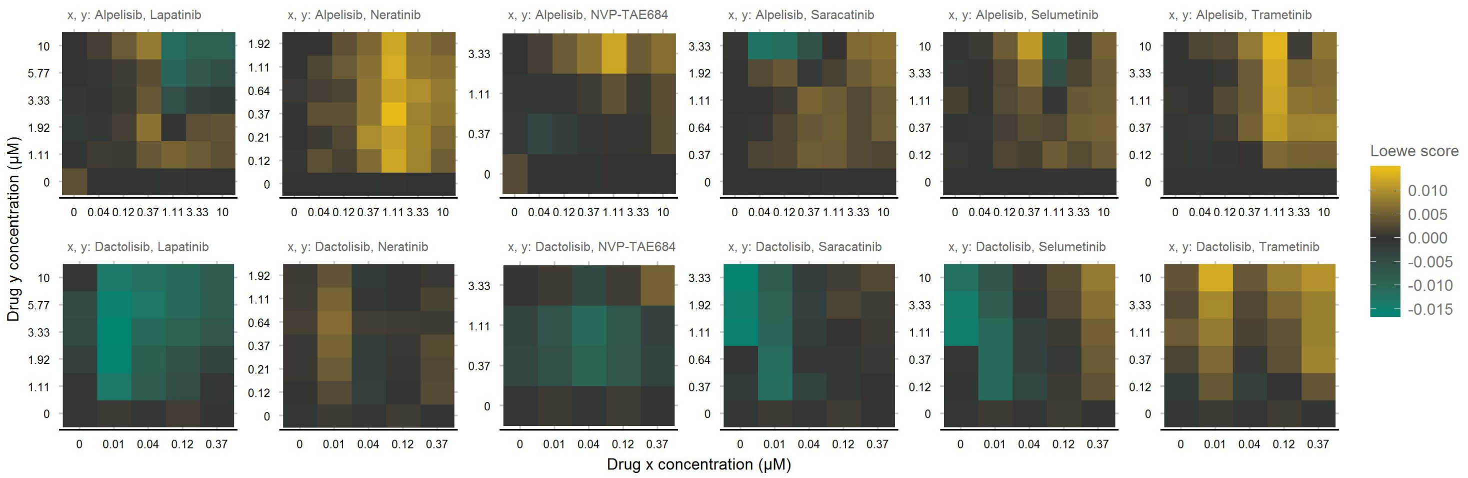


Figure S2. Assessing drug synergy/antagonism. Concentration combination matrices that represent the difference between the measured effect on the birth rates of sensitive cells and the effect obtained under the Loewe additivity model. Yellow indicates synergy, green indicates antagonism and grey means no interaction.

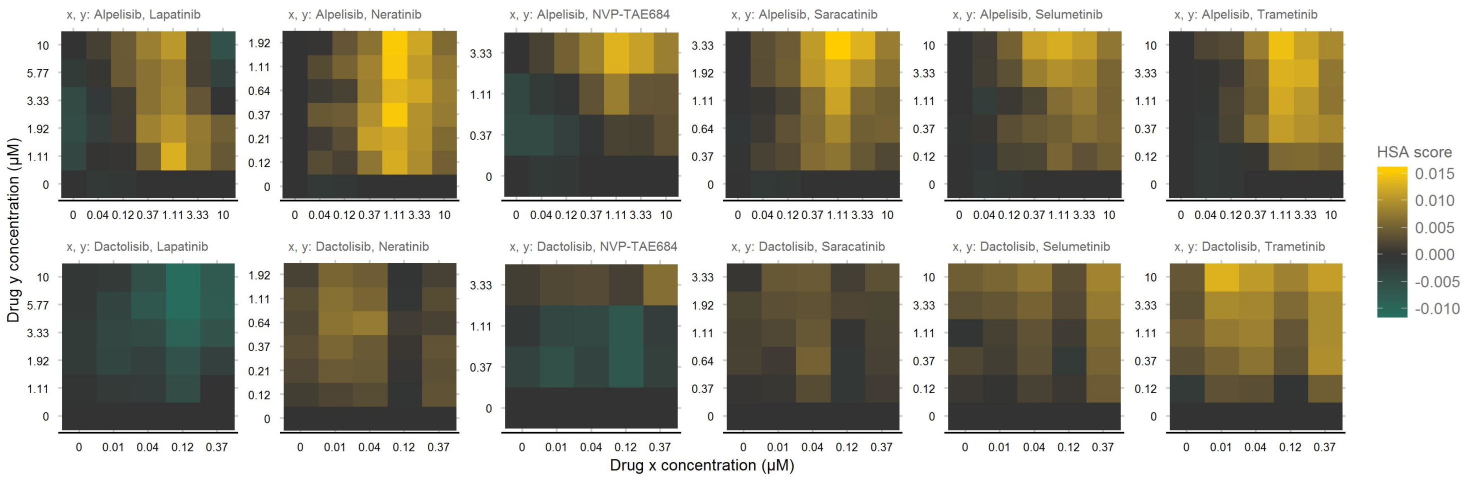


Figure S3. Assessing drug synergy/antagonism. Concentration combination matrices that represent the difference between the measured effect on the birth rates of sensitive cells and the effect obtained under the Highest Single Agent (HSA) model. Yellow indicates synergy, green indicates antagonism and grey means no interaction.

A

Dosing regimen	Alpelisib	Lapatinib	Mode
1	300 mg QD	1500 mg QD	Simultaneous
2	300 mg QD	2250 mg QOD	Simultaneous
3	300 mg QD	1000 mg QOD	Simultaneous
4	300 mg QD	4500 mg QW + 750 mg QD r.m.	Simultaneous
5	600 mg 3 days a week + 4-day holyday	2500mg 4 days a week + 3 day holyday	Alternating
6	400 mg 4 days a week + 3-day holyday	3500mg 3 days a week + 4 day holyday	Alternating
7	300 mg QD	3500mg 3 days a week + 4 day holyday	Simultaneous
8	300 mg QD	2500mg 3 days a week + 4 day holyday	Simultaneous

B

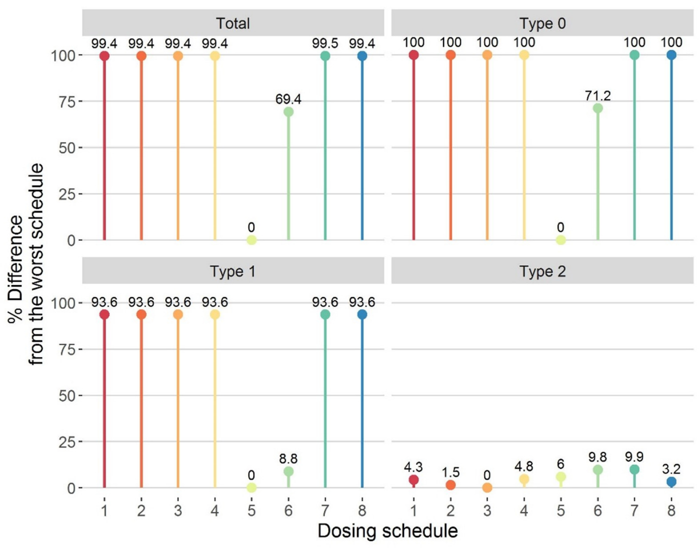


Figure S4. Effects of alpelisib and lapatinib combination on the total number of the different cancer cell types (type 0: sensitive to both drugs, type 1: alpelisib-sensitive cells, type 2: lapatinib-sensitive cells, total: the sum of type 0, type 1 and type 2 cells). A) Different dosing schedules explored (QD: once a day, QW: once a week, QOD: every other day, r.m.: remaining of the week). B) Comparison of the outcomes of the different dosing schedules. The best regimen (least expected number of cancer cells at 30 days) has the highest bar, while the worst regimen has zero height.)

A

Dosing regimen	Dactolisib	Trametinib
1	300 mg BID	2 mg QD
2	200 mg BID	2 mg QD
3	300 mg BID	6 mg QW + 1 mg QD r.m.
4	200 mg BID	6 mg twice a week
5	800 mg QW + 200 mg BID r.m.	2 mg QD
6	600 mg BID QW + 200 mg BID r.m.	2 mg QD
7	1000 mg QW + 200 mg BID r.m.	2 mg QD
8	800 mg BID QW + 200 mg BID r.m.	2 mg QD
9	800 mg BID QW + 200 mg BID r.m.	7 mg QW + 1 mg QD r.m.

B

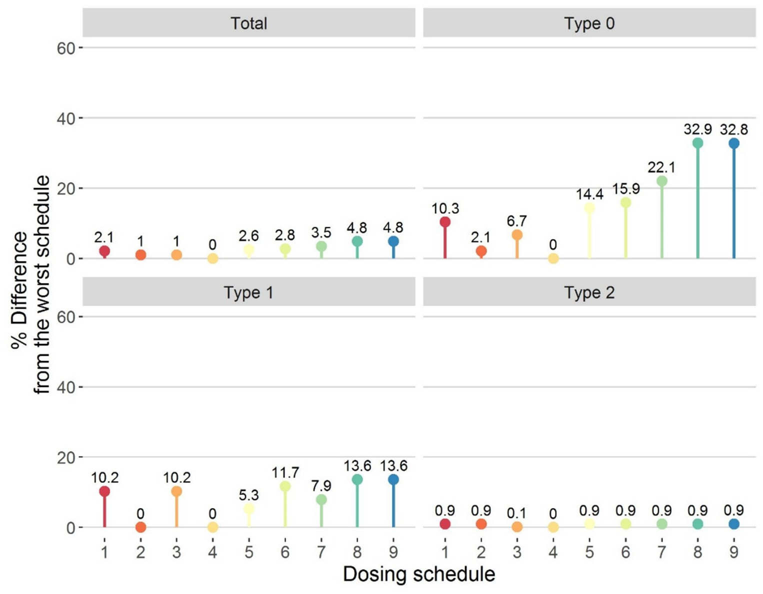
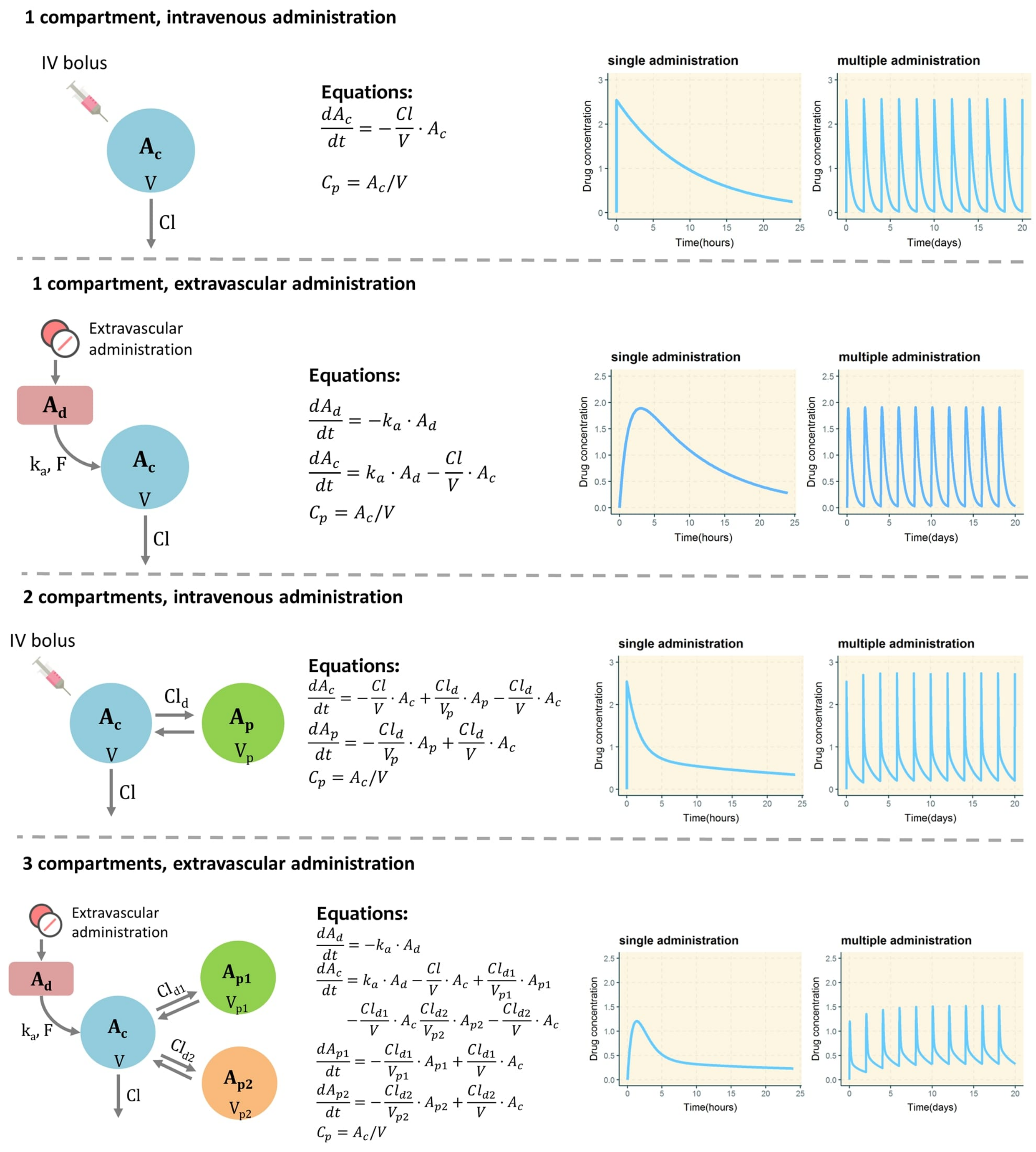


Figure S5. Effects of dactolisib and trametinib combination on the total number of the different cancer cell types (type 0: sensitive to both drugs, type 1: dactolisib-sensitive cells, type 2: trametinib-sensitive cells, total: the sum of type 0, type 1 and type 2 cells). A) Different dosing schedules explored (QD: once a day, QW: once a week, QOD: every other day, r.m.: remaining of the week). B) Comparison of the outcomes of the different dosing schedules. The best regimen (least expected number of cancer cells at 30 days) has the highest bar, while the worst regimen has zero height.)


1 compartment, intravenous administration
IV bolus
Ac
V
Cl
single administration
multiple administration
Drug concentration
Time(hours)
Time(days)
1 compartment, extravascular administration
Extravascular administration
Ad
ka, F
2 compartments, intravenous administration
Cld
Ap
Vp
3 compartments, extravascular administration
Cld1
Cld2
Ap1
Vp1
Ap2
Vp2
Equations:

1 compartment, intravenous administration

$$\frac{dA_c}{dt} = -\frac{Cl}{V}\cdot A_c$$

$$C_p = A_c/V$$

1 compartment, extravascular administration

$$\frac{dA_d}{dt} = -k_a \cdot A_d$$

$$\frac{dA_c}{dt} = k_a \cdot A_d - \frac{Cl}{V}\cdot A_c$$

$$C_p = A_c/V$$

2 compartments, intravenous administration

$$\frac{dA_c}{dt} = -\frac{Cl}{V}\cdot A_c + \frac{Cl_d}{V_p}\cdot A_p - \frac{Cl_d}{V}\cdot A_c$$

$$\frac{dA_p}{dt} = -\frac{Cl_d}{V_p}\cdot A_p + \frac{Cl_d}{V}\cdot A_c$$

$$C_p = A_c/V$$

3 compartments, extravascular administration

$$\frac{dA_d}{dt} = -k_a \cdot A_d$$

$$\frac{dA_c}{dt} = k_a \cdot A_d - \frac{Cl}{V}\cdot A_c + \frac{Cl_{d1}}{V_{p1}}\cdot A_{p1} - \frac{Cl_{d1}}{V}\cdot A_c + \frac{Cl_{d2}}{V_{p2}}\cdot A_{p2} - \frac{Cl_{d2}}{V}\cdot A_c$$

$$\frac{dA_{p1}}{dt} = -\frac{Cl_{d1}}{V_{p1}}\cdot A_{p1} + \frac{Cl_{d1}}{V}\cdot A_c$$

$$\frac{dA_{p2}}{dt} = -\frac{Cl_{d2}}{V_{p2}}\cdot A_{p2} + \frac{Cl_{d2}}{V}\cdot A_c$$

$$C_p = A_c/V$$

Figure S6. Schematic representation and ordinary differential equations of different pharmacokinetic models. A_c, A_d and A_p represents the amount of drug in the central, depot and peripheral compartments respectively; CL is the apparent total clearance; V, V_p, are the apparent volumes of distribution of the central and peripheral compartments respectively; CL_d is the distribution clearance between the central and peripheral compartments; k_a is the first-order absorption rate constant; F is the bioavailability and C_p represents the drug concentration in plasma.

Demo 1: Two type branching process

As a simple example that demonstrates the necessary steps for an analysis of a single schedule, we consider the two-type birth-death process shown in Figure 1. This model is extensively used in investigating the dynamics of tumor cells in response to treatment, where a population initially sensitive to therapy will gain resistance via mutation and expand at a different rate. In this example, we evaluate erlotinib treatment and how erlotinib-sensitive (Type 0 cells, blue in the figure) and erlotinib-resistant (Type 1 cell, green) cell birth and death rates are defined using PC-9 cell viability and apoptosis assay data, how the pharmacokinetics of erlotinib can be defined to evaluate its effect of the growth kinetics of each cell type and finally, how the evolution of these cells over time can be calculated using ACESO package.

ACESO is very flexible when defining functions and allows use of other packages for estimation throughout, but the general workflow for the process is used in the analysis and summarized by the following diagram:

Load libraries:

```
library(ACESO)
#> Warning: replacing previous import 'drc::gaussian' by 'stats::gaussian'
#> when loading 'ACESO'
#> Warning: replacing previous import 'drc::getInitial' by 'stats::getInitial'
#> when loading 'ACESO'
library(mrgsolve)
#> Warning: package 'mrgsolve' was built under R version 3.5.3
#>
#> Attaching package: 'mrgsolve'
#> The following object is masked from 'package:stats':
#>
#>     filter
library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 3.5.3
```

Read cell proliferation file:

Use **read.cellcount.data** function to create a data.frame with the info about cell proliferation under different drug concentrations.

Mandatory columns for in the .csv file:

- Viable.cells: cell count data
- Time: time point
- CONC: drug concentration

Optional columns:

- Replicate: Technical replicate. If missing the program assumes that there is only 1 replicate.
- Cell.line: Name of the cell line. If missing the function introduces the name 'Cell line 1'.
- Type: numerical column specifiying the cell type (0: sensitive cells, 1: resistant cell to drug A, etc.). If missing the program assumes Type=0.

```
growth_data=read.cellcount.data(system.file("extdata", "cell_viability_assay.txt",
                                            package = "ACESO"), sep=";")
head(growth_data)
#>                Sample.ID Cell.line Viable.cells Time Replicate CONC Type
#> 1 pc-9 par dmso 48hrs-1      PC-9         0.38   48         1    0    0
#> 2 pc-9 par dmso 48hrs-2      PC-9         0.39   48         2    0    0
#> 3 pc-9 par dmso 48hrs-3      PC-9         0.33   48         3    0    0
```

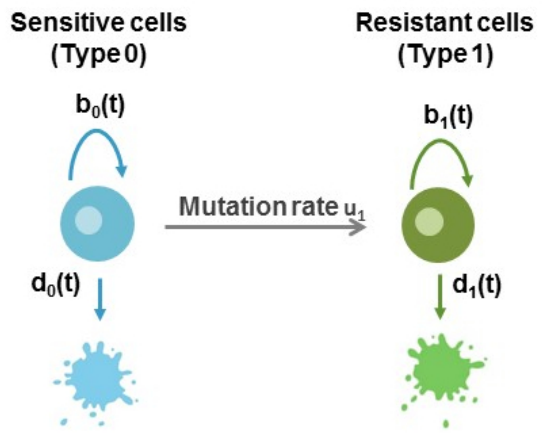


Figure 1: Two type branching process.

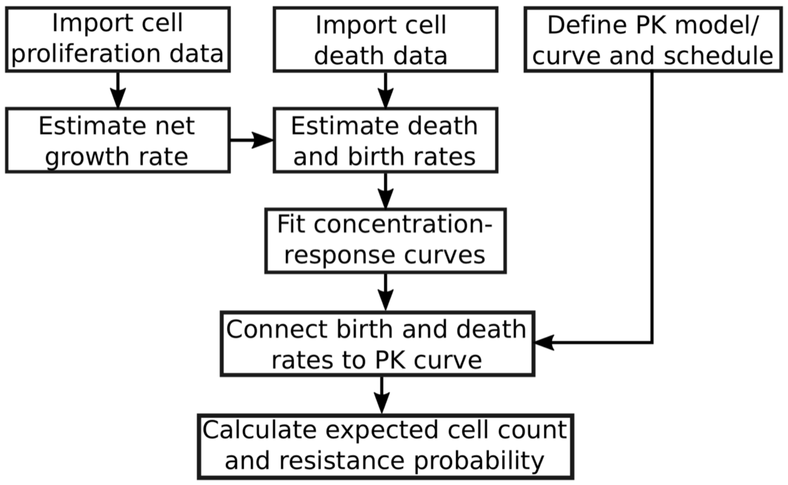

Import cell proliferation data
Import cell death data
Define PK model/ curve and schedule
Estimate net growth rate
Estimate death and birth rates
Fit concentration-response curves
Connect birth and death rates to PK curve
Calculate expected cell count and resistance probability

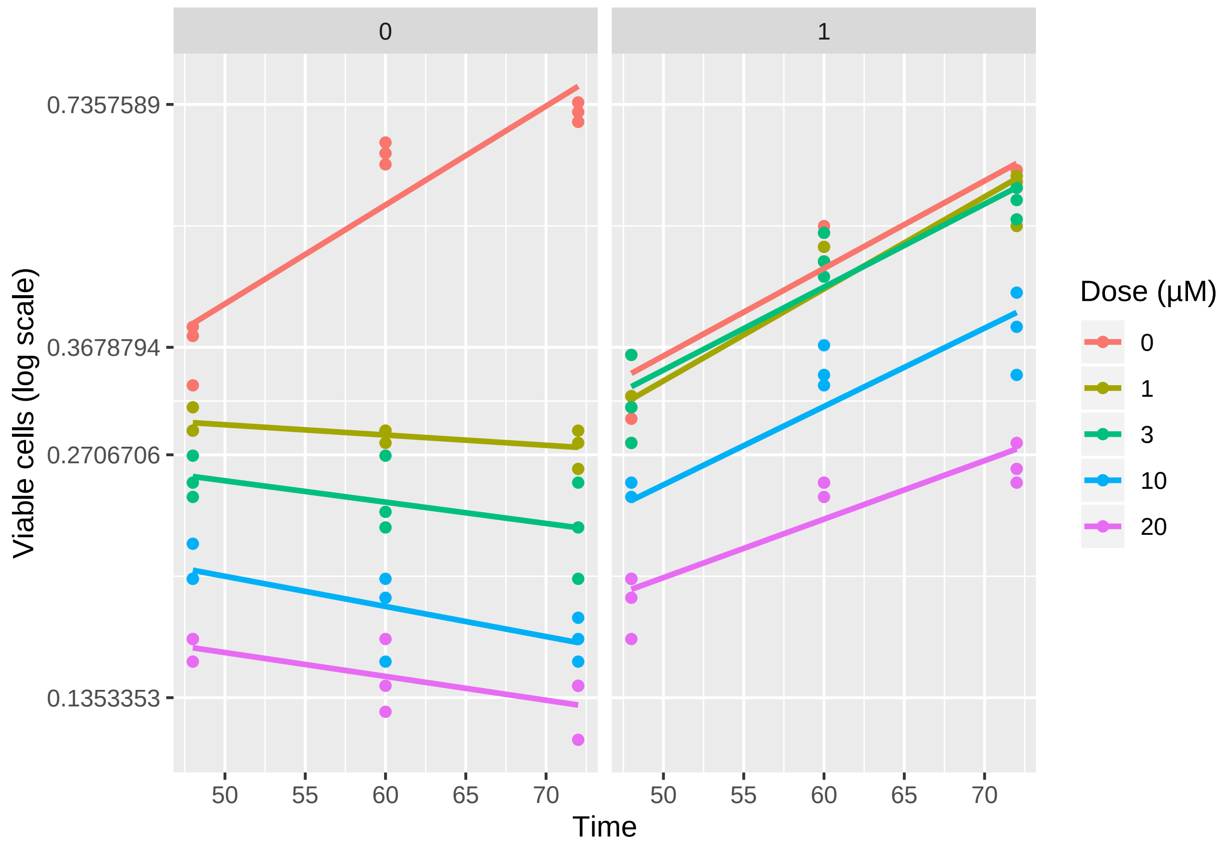
Figure 2: Typical Workflow (single-drug).

```
#> 4   pc-9 par E1 48hrs-1      PC-9         0.29   48         1    1    0
#> 5   pc-9 par E1 48hrs-2      PC-9         0.29   48         2    1    0
#> 6   pc-9 par E1 48hrs-3      PC-9         0.31   48         3    1    0
#>        Type2 Cell_Count_0 Control
#> 1 sensitive        0.062    0.38
#> 2 sensitive        0.066    0.39
#> 3 sensitive        0.065    0.33
#> 4 sensitive        0.062    0.38
#> 5 sensitive        0.066    0.39
#> 6 sensitive        0.065    0.33
```

Two new columns are created:

- Cell_Count_0: cell count at time 0.
- Control: cell count when there is no drug concentration

```r
#Exploraty plot:  viable cells over time
library(ggplot2)
ggplot(data=growth_data,aes(x=Time,y=Viable.cells,col=factor(CONC)))+geom_point()+
  geom_smooth(se=F,method="lm")+facet_wrap(~Type)+
  scale_y_continuous(trans="log")+ylab("Viable cells (log scale)")+
  scale_colour_discrete(name="Dose (µM)")
```



Read cell death file:

Call **read.celldeath.file** to create a data.frame with the info about the total count of dead cells under different drug concentrations.

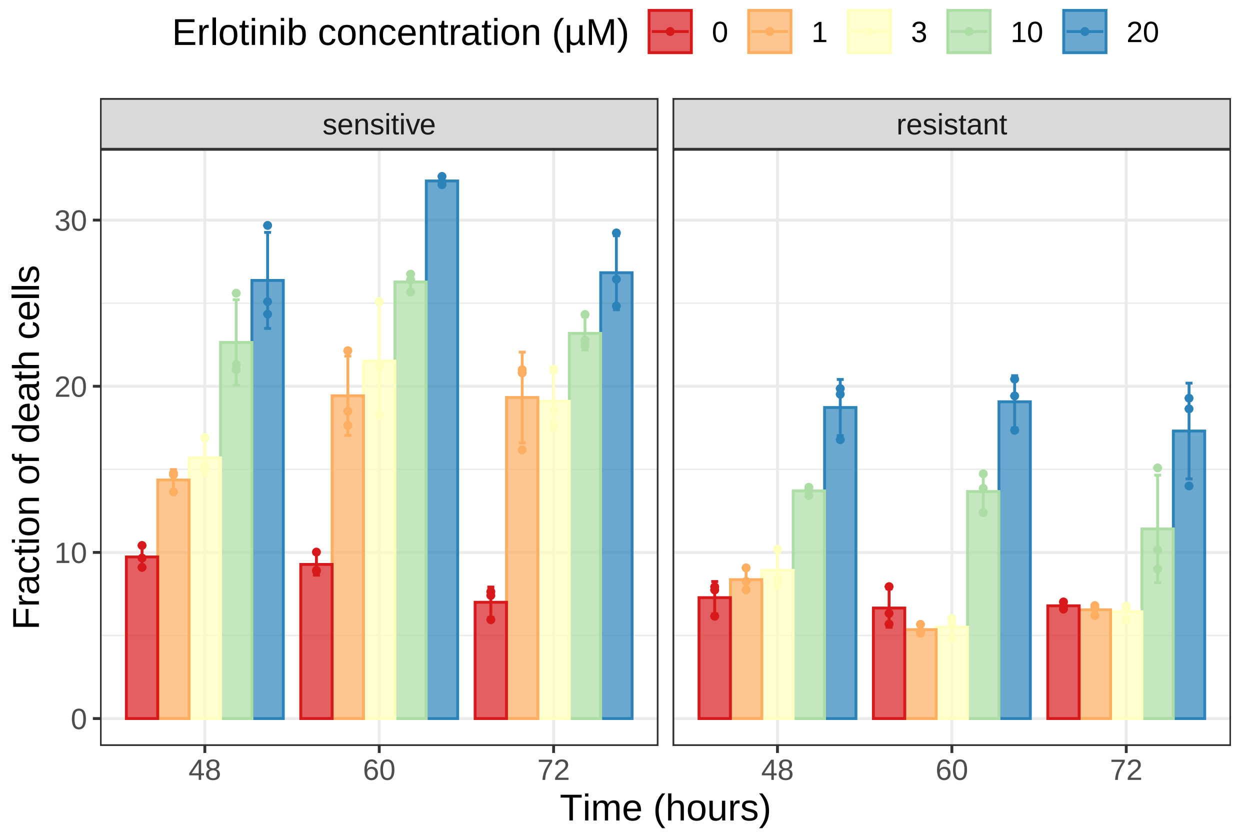
Necessary columns to be in the .csv file:

- Time: time point
- column.name: name of the column where the fraction of dead cells or the total count of dead cells is stored.
- CONC: drug concentration

```
cell_death=read.celldeath.file(system.file("extdata", "apoptosis_assay.txt", package = "ACESO"),
                               column.name="Apoptotic.fraction",sep=";")
head(cell_death)
#>    Cell.line Time Apoptotic.cells Apoptotic.fraction Replicate CONC Type
#> 1:      PC-9    0      0.03047486               9.25         1    0    0
#> 2:      PC-9    0      0.02750485               8.99         2    0    0
#> 3:      PC-9    0      0.02210822               7.71         3    0    0
#> 4:      PC-9    0      0.02186835               9.25         1    1    0
#> 5:      PC-9    0      0.01987907               8.99         2    1    0
#> 6:      PC-9    0      0.02304976               7.71         3    1    0
#>        Type2 Apoptosis_time0
#> 1: sensitive            9.25
#> 2: sensitive            8.99
#> 3: sensitive            7.71
#> 4: sensitive            9.25
#> 5: sensitive            8.99
#> 6: sensitive            7.71
```

Exploratory plot:

```
library(dplyr)
#> Warning: package 'dplyr' was built under R version 3.5.3
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
my_sum <- cell_death %>%
  group_by(CONC,Time,Type2) %>%
  summarise(
    n=n(),
    mean=mean(Apoptotic.fraction),
    sd=sd(Apoptotic.fraction)
  )

ggplot(cell_death[cell_death$Time!=0,])+
  geom_bar(data=my_sum[my_sum$Time!=0,],
           aes(x=as.factor(Time), y=mean,fill=as.factor(CONC),color=as.factor(CONC)),
           stat="identity", alpha=0.7, position=position_dodge(0.9)) +
  geom_jitter(aes(x=as.factor(Time), y=Apoptotic.fraction,color=as.factor(CONC)), size=0.9,
              position=position_dodge(0.9))+
  facet_wrap(~factor(Type2,levels=c("sensitive","resistant")))+
  xlab("Time (hours)")+ylab("Fraction of death cells")+
  geom_errorbar(data=my_sum[my_sum$Time!=0,],
                aes(x=as.factor(Time), ymin=mean-sd, ymax=mean+sd,color=as.factor(CONC)),
                width=.2, position=position_dodge(0.9))+
  theme_bw()+scale_color_brewer(name="Erlotinib concentration (µM)",palette="Spectral")+
  scale_fill_brewer(name="Erlotinib concentration (µM)",palette="Spectral")+
  theme(text = element_text(size=14))+theme(legend.position="top")
```



Calculate net growth rate from cell proliferation data:

In order to calculate the net growth of each cell type, we perform a linear regression on log-transformed viable cells over time as we assume an exponentially growing population of cells. In this exercise, we decided not to use the data corresponding to time = 0 to calculate those rates.

(NOTE: Why not use data at time = 0? Also, I had to multiple cell count by 10 to get the Net_growth given. Did I do something wrong?) :

```
growth_data<-net_growth_rate(growth_data,time0_data = F)
head(growth_data) #See the values in Net_growth column
#>                Sample.ID Cell.line Viable.cells Time Replicate CONC Type
#> 1 pc-9 par dmso 48hrs-1      PC-9         0.38   48         1    0    0
#> 2 pc-9 par dmso 48hrs-2      PC-9         0.39   48         2    0    0
#> 3 pc-9 par dmso 48hrs-3      PC-9         0.33   48         3    0    0
#> 4   pc-9 par E1 48hrs-1      PC-9         0.29   48         1    1    0
#> 5   pc-9 par E1 48hrs-2      PC-9         0.29   48         2    1    0
#> 6   pc-9 par E1 48hrs-3      PC-9         0.31   48         3    1    0
#>       Type2 Cell_Count_0 Control   Net_growth Death_rate Birth_rate
#> 1 sensitive        0.062    0.38  0.028216265         NA         NA
#> 2 sensitive        0.066    0.39  0.028216265         NA         NA
#> 3 sensitive        0.065    0.33  0.028216265         NA         NA
#> 4 sensitive        0.062    0.38 -0.002930305         NA         NA
#> 5 sensitive         0.066     0.39 -0.002930305          NA          NA
#> 6 sensitive         0.065     0.33 -0.002930305          NA          NA
```

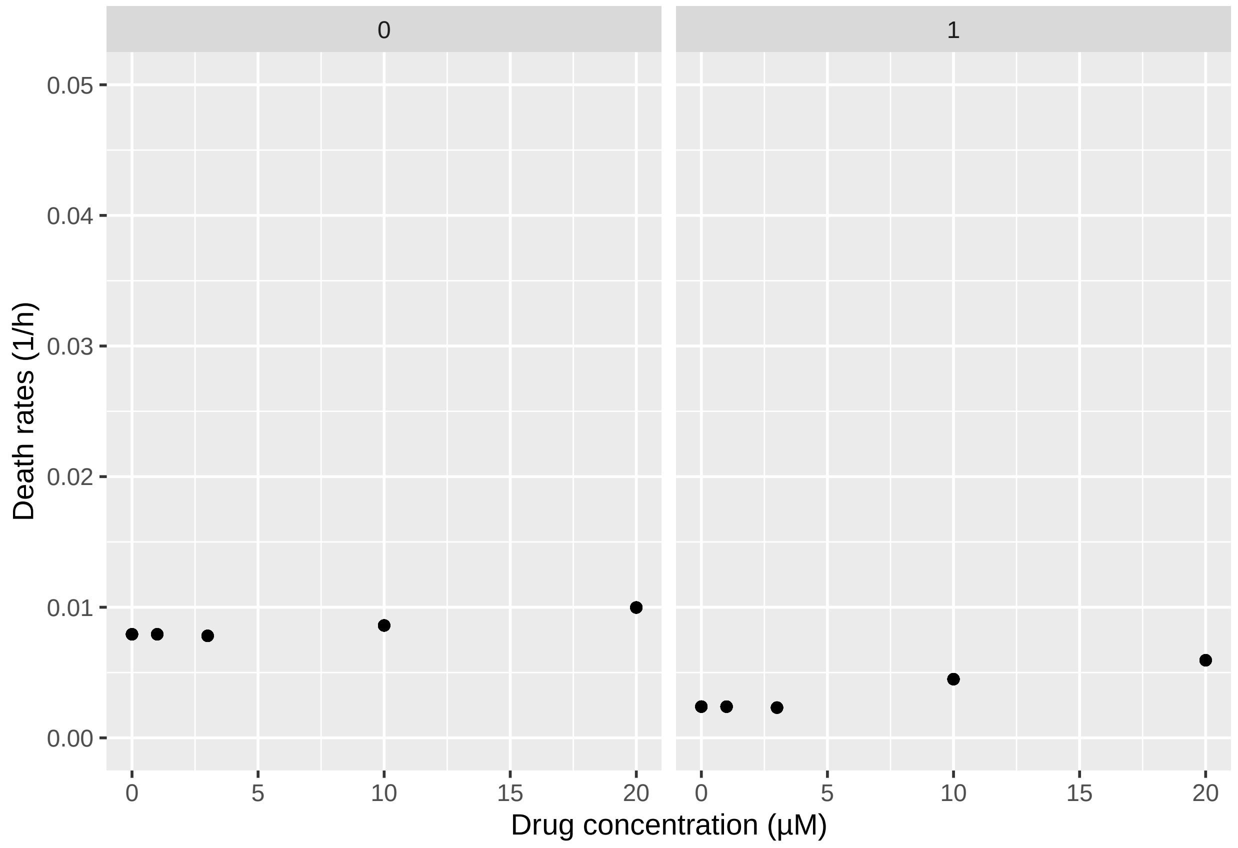
Calculate death rate:

Use the data from cell apoptosis assays and the data where the net growth rate parameters have been calculated to compute the death rate parameter. Call **calculate_death_rate** for this purpose. You must also specify the column name where the apoptotic cell count or the apoptotic cell fraction is stored, and finally specify if this column stores a fraction or the total counts by setting the argument 'Apoptotic.fraction' to TRUE or FALSE.

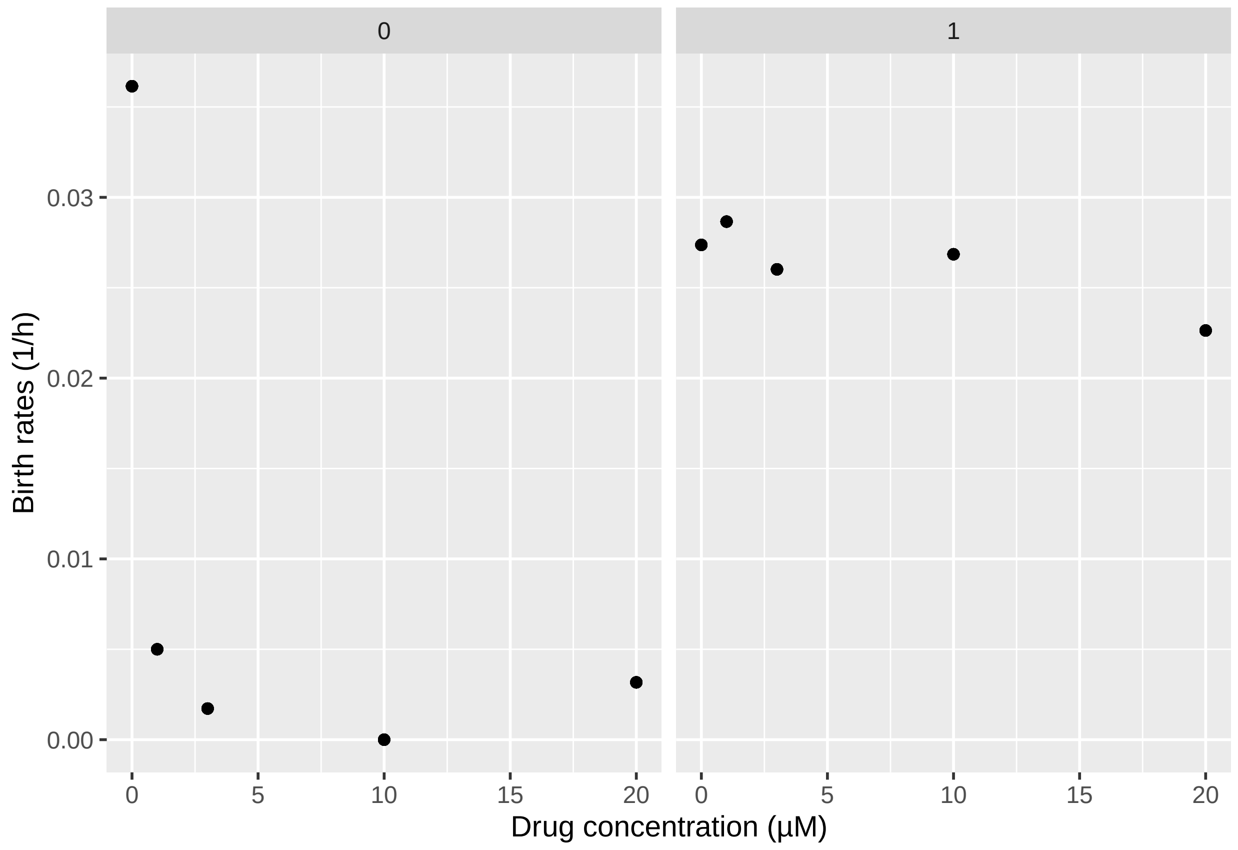
```
all.data=calculate_death_rate(net_growth_data=growth_data,cell_death_data =cell_death,
                              column.name = "Apoptotic.fraction",Apoptotic.fraction = T)
```

Plot the resulting death and birth rates for each cell type:

```
ggplot(all.data,aes(x=CONC,y=Death_rate))+geom_point()+scale_y_continuous(limits = c(0, 0.05))+
  facet_wrap(~Type)+ylab("Death rates (1/h)")+xlab("Drug concentration (µM)")
```



```
ggplot(all.data,aes(x=CONC,y=Birth_rate))+geom_point()+facet_wrap(~Type)+
  ylab("Birth rates (1/h)")+xlab("Drug concentration (µM)")
```

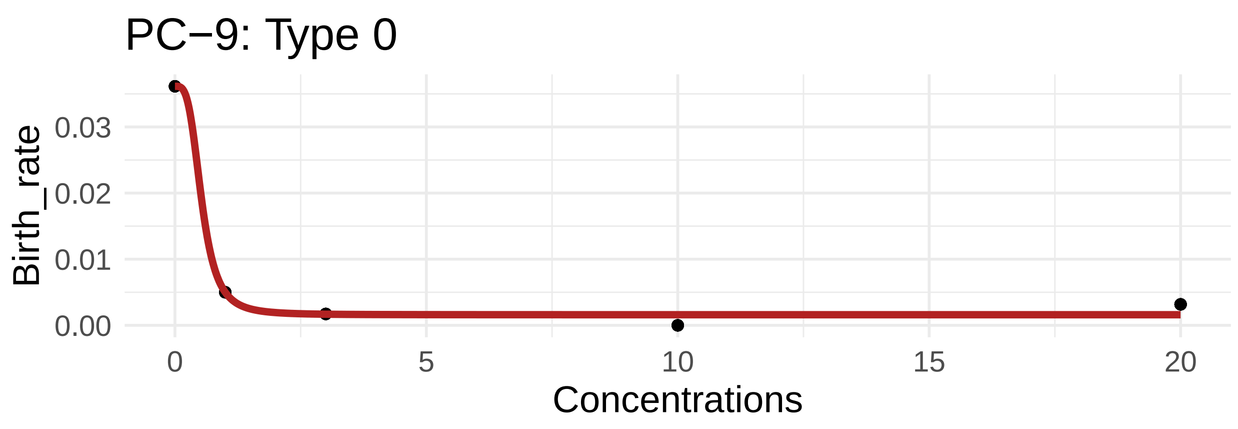


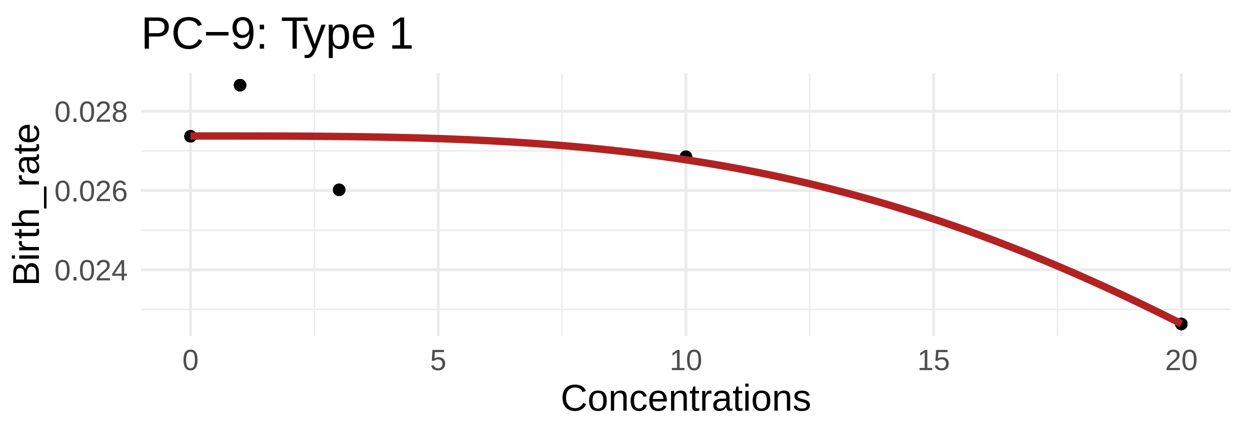
Fit the curves

Call **Multiple.best.singlefit** function to fit the concentration-response curves. In this example we only have data from one cell.line (PC-9), but this function is able to fit multiple cell lines.

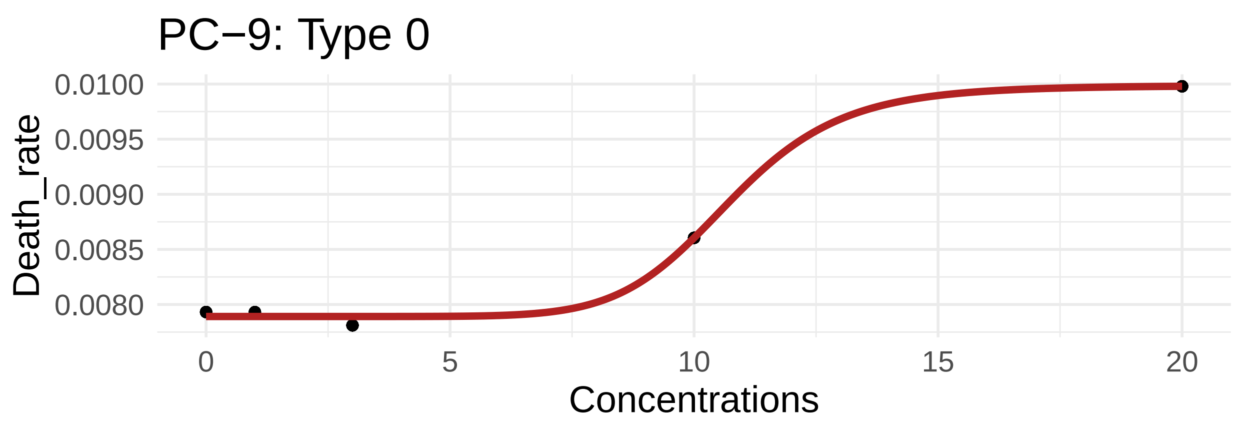
To perform this task easily and faster, we included the flexible and built-in model functions of the drc R package. drc was developed to provide nonlinear model fitting for dose-response analysis and it already includes the most common function to fit this type of curves. To see all the different built-in functions included in the package write: drc::getMeanFunctions(). **Multiple.best.singlefit** tries all the different models of drc library and selects the best one for you based on AIC or BIC (by default AIC is used).

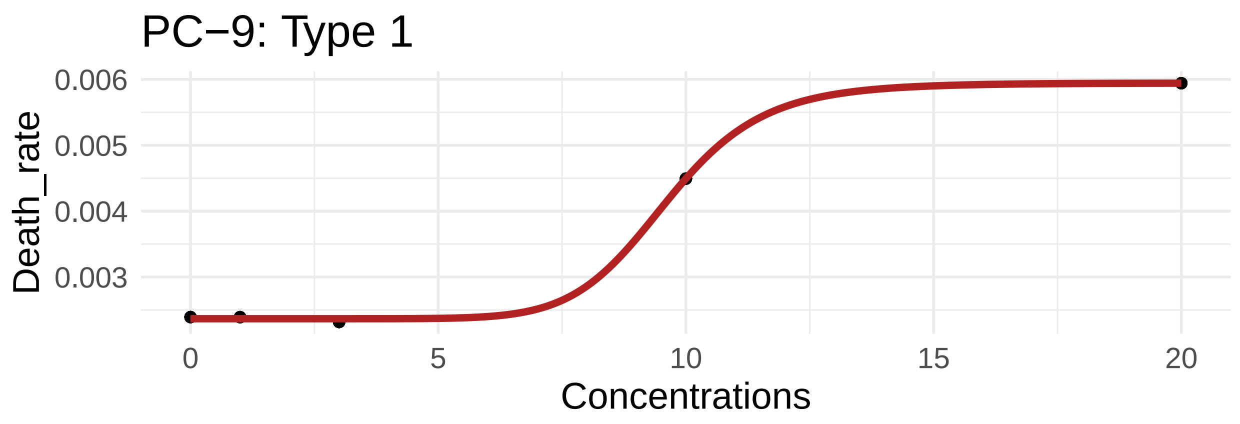
```
#Fit birth rates
BR.fit=Multiple.best.singlefit(all.data,resp="Birth_rate")
```





```
BR.fit
#> [[1]]
#>
#> A 'drc' model.
#>
#> Call:
#> drc::drm(formula = Birth_rate ~ CONC, data = data, fct = LL.4(),     type = "continuous")
#>
#> Coefficients:
#> b:(Intercept)  c:(Intercept)  d:(Intercept)  e:(Intercept)
#>      3.511693       0.001603       0.036148       0.532265
#>
#>
#> [[2]]
#>
#> A 'drc' model.
#>
#> Call:
#> drc::drm(formula = Birth_rate ~ CONC, data = data, fct = LL.3(),     type = "continuous")
#>
#> Coefficients:
#> b:(Intercept)  d:(Intercept)  e:(Intercept)
#>       3.21881        0.02738       32.51748
#Fit death rates
DR.fit=Multiple.best.singlefit(all.data,resp="Death_rate")
```





```
DR.fit
#> [[1]]
#>
#> A 'drc' model.
#>
#> Call:
#> drc::drm(formula = Death_rate ~ CONC, data = data, fct = LL.4(),     type = "continuous")
#>
#> Coefficients:
#> b:(Intercept)  c:(Intercept)  d:(Intercept)  e:(Intercept)
#>     -9.275915       0.007891       0.009986      10.738019
#>
#>
#> [[2]]
#>
#> A 'drc' model.
#>
#> Call:
#> drc::drm(formula = Death_rate ~ CONC, data = data, fct = LL.4(),     type = "continuous")
#>
#> Coefficients:
#> b:(Intercept)  c:(Intercept)  d:(Intercept)  e:(Intercept)
#>     -9.911489       0.002367       0.005946       9.621362
```

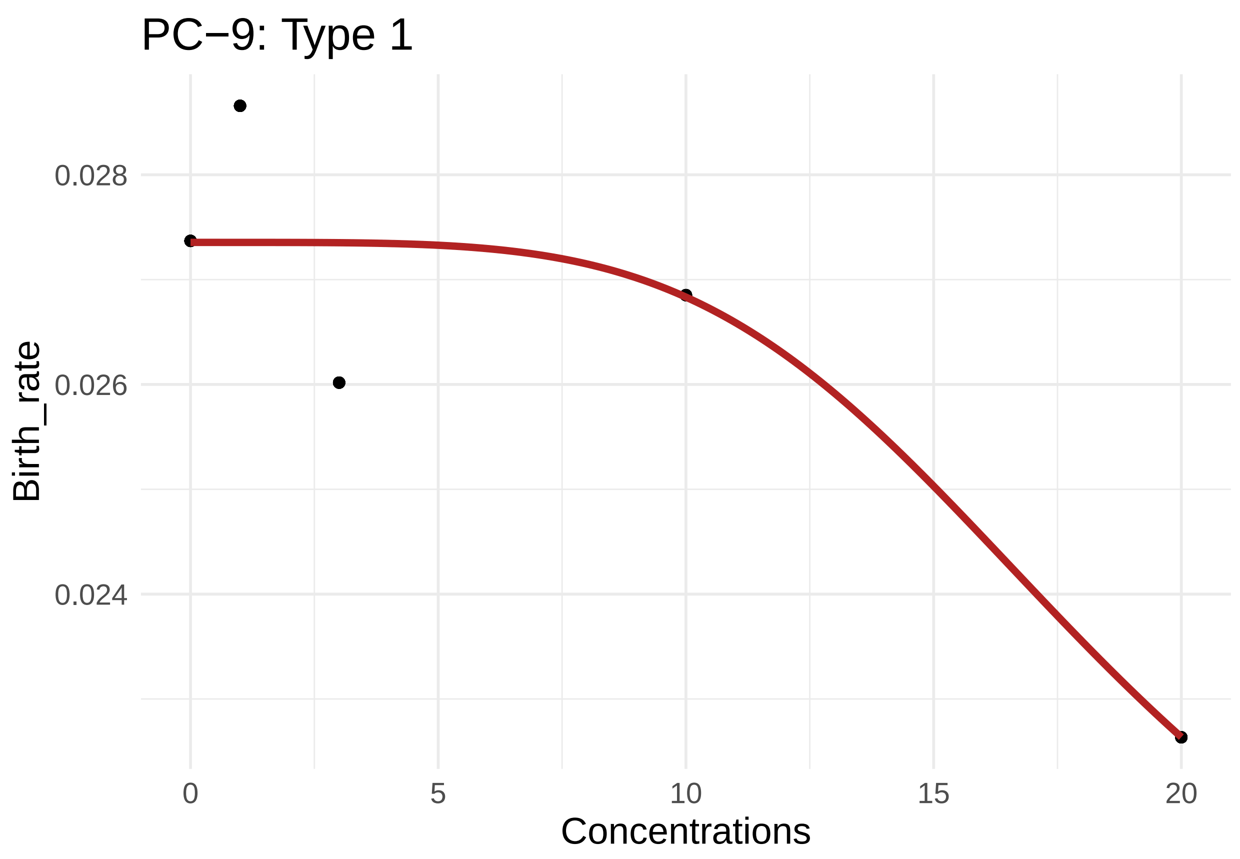
If the user wants to check all the models that has been compared for a particular cell type, **best.singlefit** can be used. Here we are going to check all the models tested for the birth rates of the resistant cell line

(Type=1):

```
best.singlefit(all.data[all.data$Type==1,],resp="Birth_rate",compare=T)
#>          logLik         IC      Res var
#> LL.3  255.27078 -502.54155 7.417511e-07
#> W1.3  255.21138 -502.42276 7.437119e-07
#> L.4   255.49166 -500.98332 7.524197e-07
#> LN.3  254.31487 -500.62974 7.739432e-07
#> W1.4  255.17471 -500.34941 7.630939e-07
#> LL.4  255.15445 -500.30890 7.637812e-07
#> LN.4  255.10483 -500.20966 7.654674e-07
#> L.5   255.35926 -498.71851 7.757820e-07
#> LL.5  255.29987 -498.59974 7.778322e-07
#> L.3   248.80093 -489.60186 9.888696e-07
#> EXD.2 247.58061 -489.16122 1.019705e-06
#> G.3   248.54696 -489.09393 1.000095e-06
#> EXD.3 244.83190 -481.66379 1.179639e-06
#> W2.4  237.84867 -465.69734 1.648181e-06
#> G.4   230.43400 -450.86799 2.291515e-06
#> LL.2   99.72256 -193.44512 7.284960e-04
#> G.2    99.72256 -193.44512 7.284960e-04
#> W1.2  -26.43647   58.87294 1.984019e-01
#Some error messages might appear because not all the models tested are able to fit the data. Ignore th
```

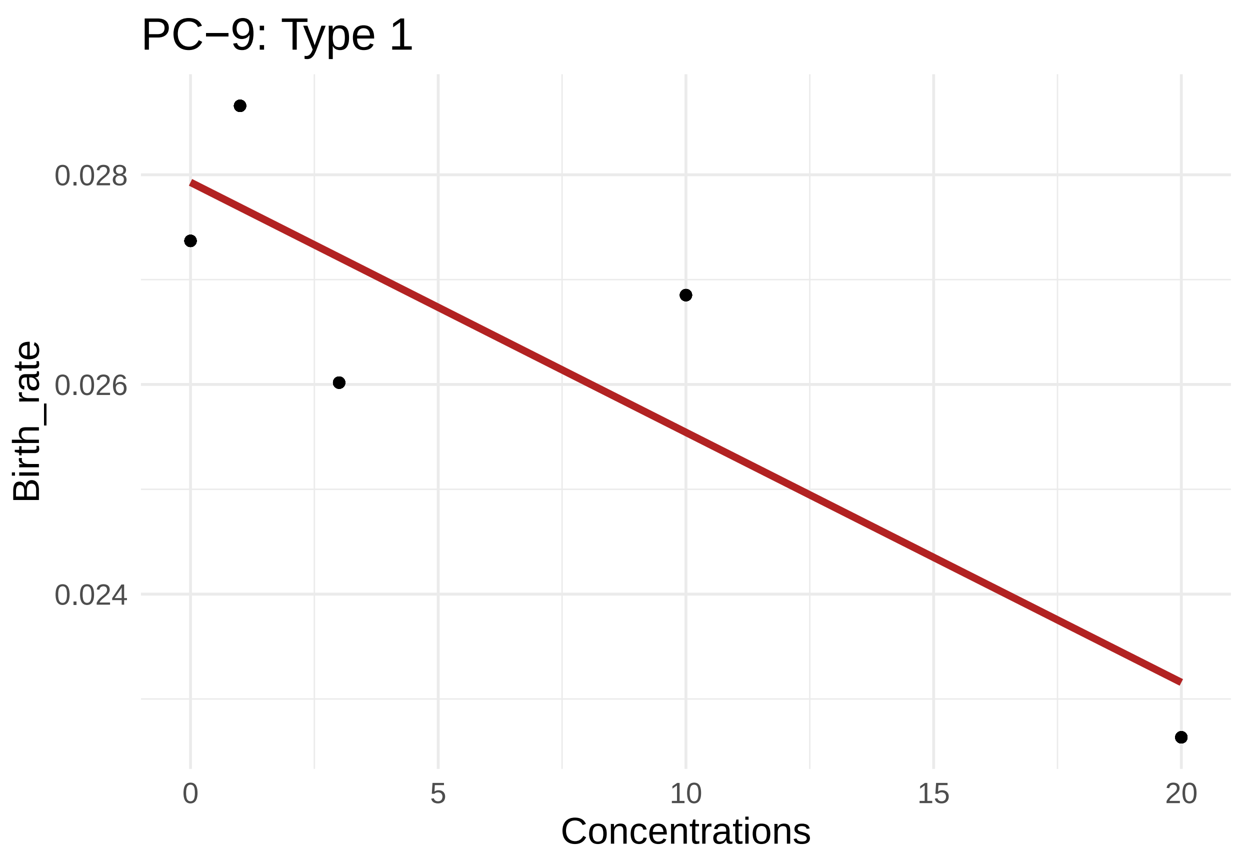
If the user does not want to use the best model selected by **Multiple.best.singlefit** function, **Multiple.singlefit** can be used which allows to fit the data with the model selected by the user in the fct argument. In this example, a 4 parameter log-logistic function (called LL.4) is selected to fit the birth rate vs concentration curve of resistant cells (Type=1):

```
B1.fit=Multiple.singlefit(all.data[all.data$Type==1,],fct=drc::LL.4(),resp="Birth_rate")
#> Warning in sqrt(diag(varMat)): Se han producido NaNs
```



drc package only includes non-linear functions to fit the concentration-response curves. Thus, to select a simple linear function to fit the data, linear.model = T must be specified.

```
B1.fit=Multiple.singlefit(all.data[all.data$Type==1,],resp="Birth_rate",linear.model = T)
```



Pharmacokinetics

Now we need to define the pharmacokinetics (PK) of the drug. To this end, we use the PK model developed by Lu et al. 2016 (https://doi.org/10.1016/j.clpt.2006.04.007) and the mrgsolve package to simulate the chosen model.

We select a 1 compartment model with extravascular administration. Read the model:

```
model_library(list=T)
#> ACESO internal library of PK models:
#> [1] "1cmt_2depot" "1cmt_ev"     "1cmt_iv"     "2cmt_2depot" "2cmt_ev"
#> [6] "2cmt_iv"     "3cmt_ev"     "3cmt_iv"

cmt1_ev <- mread("1cmt_ev", model_library()) %>% Req(CP)
#> Building 1cmt_ev ...
#> done.
see(cmt1_ev)
#>
#> Model file:  1cmt_ev.cpp
#> $PARAM @annotated
#>   TVCL    :  2 : Clearance (volume/time)
#>   TVV     : 20 : Central volume (volume)
#>   TVKA    : 1 : Absorption rate constant (1/time)
#>   F: 1 : bioavailability
#>   ALAG  : 0 : Lag time (time)
#>
#>   $CMT  @annotated
#>   EV   : Extravascular compartment
#>   CENT : Central compartment
#>
#>   $MAIN
#>   double CL = exp(log(TVCL) + ETA_CL);
#>   double V = exp(log(TVV)  + ETA_V);
#>   double KA = exp(log(TVKA)  + ETA_KA);
#>
#> ALAG_EV = ALAG;
#> F_EV = F;
#>
#> $OMEGA @labels ETA_CL ETA_V ETA_KA
#>   0 0 0
#>
#> $GLOBAL
#> #define CP (CENT/V)
#>
#>   $PKMODEL ncmt = 1, depot = TRUE
#>
#>   $CAPTURE @annotated
#>   CP : Plasma concentration (mass/volume)
```

The model has different parameters with default values:

```
param(cmt1_ev)
#>
#>  Model parameters (N=5):
#>  name value . name value
#>  ALAG 0     | TVKA 1
#>  F    1     | TVV  20
#>  TVCL 2     | .    .
```

- TVCL: Typical value for the clearance (volume/time)
- TVV: Typical value for the volumen of distribution
- TVKA: Typical value for the first order absorption rate constant.
- F: Bioavailability
- ALAG: Lag time

Change parameter values to the ones defined in the original publication:

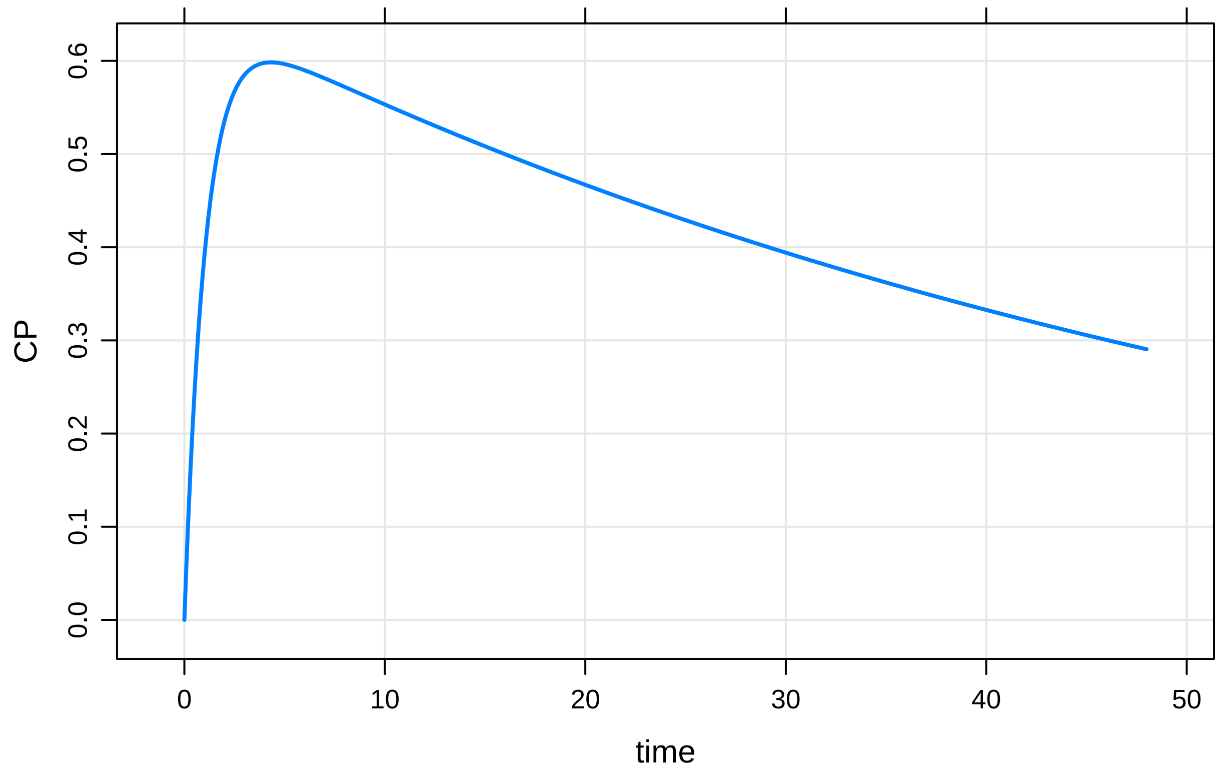
```
newpar <-  list('TVCL' = 3.95, #L/h
          'TVV'  = 233,  #L
          'TVKA' = 0.95) #h-1
```

Define the dosing event: #A dose of 150mg

```
e1 <-  ev(amt = 150, ii = 0, addl = 0, time=0)
# amt: amount
# ii: dosing interval
# addl: additional doses
# time: time when the dose is given.
```

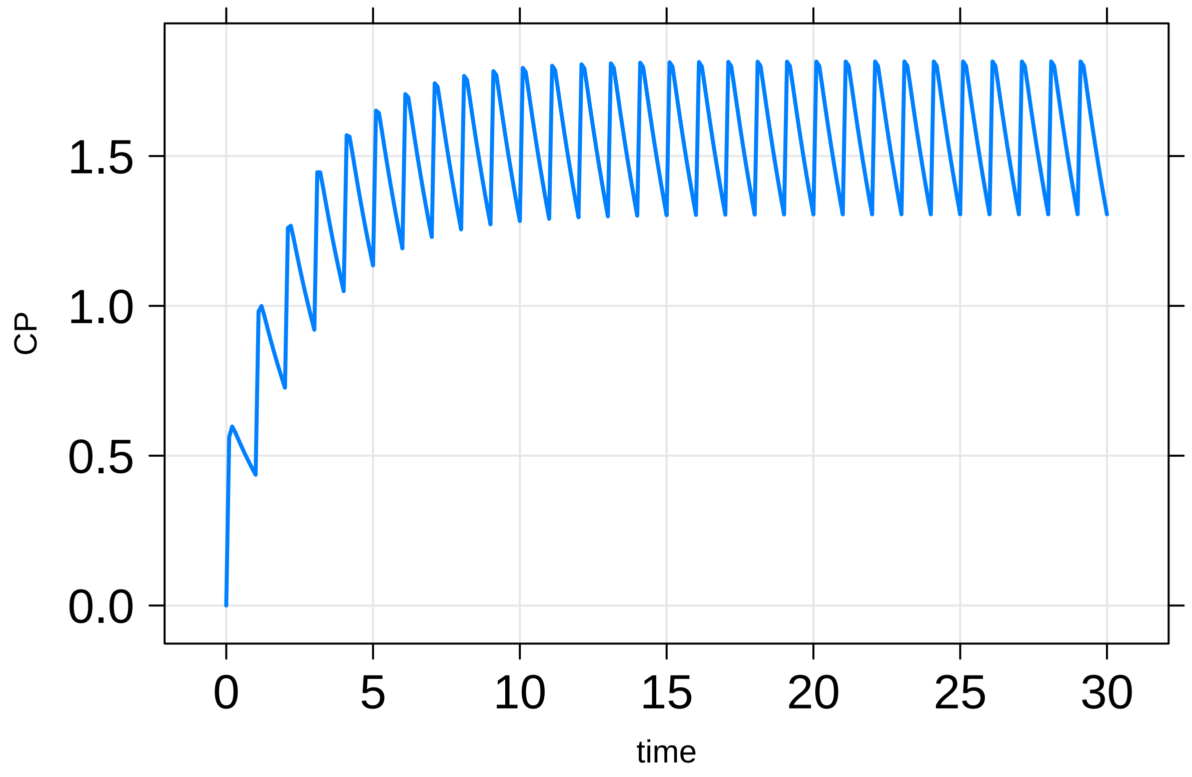
Simulate the model with easy.mrgsim function (simplified version of mrgsim function in mrgsolve package)

```
easy.mrgsim(model=cmt1_ev,dosing_schedule=e1,delta=0.1,tend=48,parameters = newpar) %>% plot
```



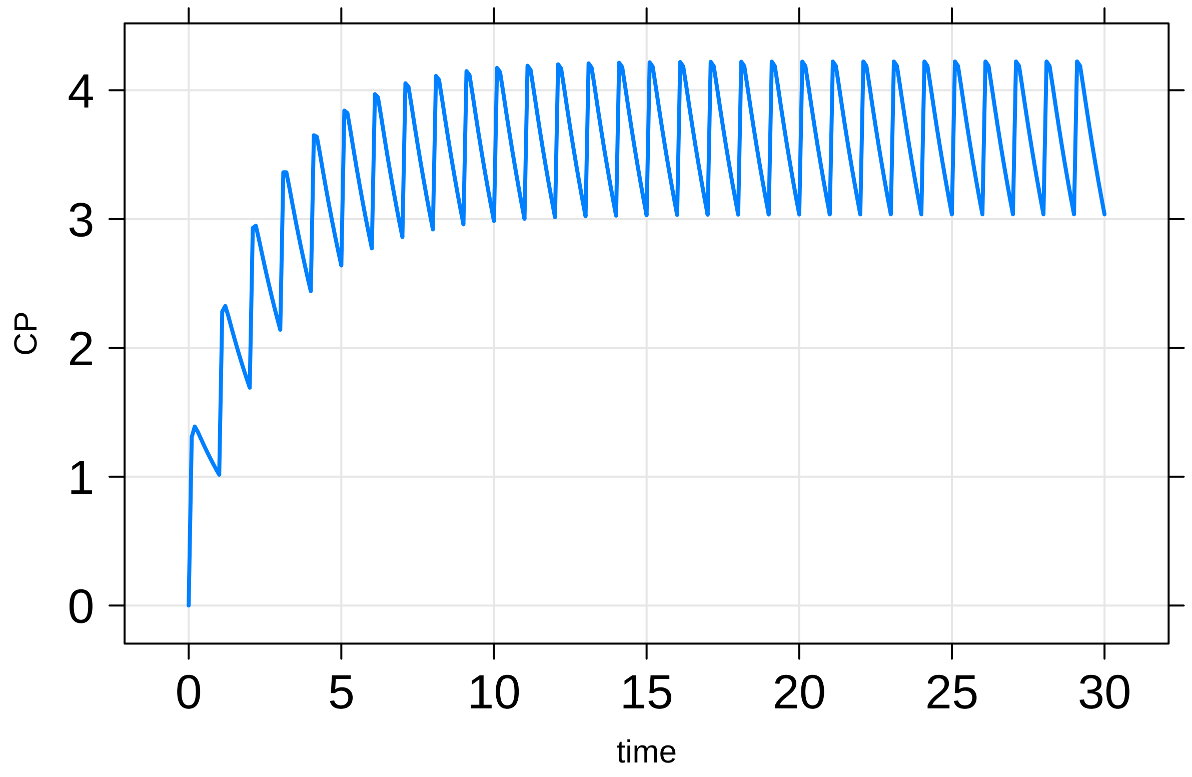
Now we are going to simulate a continuous daily dosing of 150mg for a month. For this we will change the dosing event but also de parameter values to have them in day units instead of hours:

```
e2 <-  ev(amt = 150, ii = 1, addl = 30, time=0) #150 mg every day for 30 days
easy.mrgsim(model=cmt1_ev,dosing_schedule=e2,delta=0.1,tend=30,
            parameters = list(TVCL=3.95*24,TVV=233,TVKA=0.95*24)) %>% plot(scales=list(cex=1.5))
```



To change the concentration units from mg/L to microM use the scale argument:

```
easy.mrgsim(model=cmt1_ev,dosing_schedule=e2,delta=0.1,tend=30,
            parameters = list(TVCL=3.95*24,TVV=233,TVKA=0.95*24),scale=1000/429.9) %>%
plot(scales=list(cex=1.5))
```



To simulate more complex dosing regimens, there is a vignette available in ACESO.

If the user is happy with the simulated pk model, then we need to define a pk function to be used during the simulation:

```
pk=pk.function(model=cmt1_ev,dosing_schedule=e2,tend=30,
               parameters = list(TVCL=3.95*24,TVV=233,TVKA=0.95*24),scale=1000/429.9)
```

If the user doesn't want to use mrgsolve to define the pk model, he/she can define his/her own function.

For example,

```
pk2=function(t, Cp0,k, time_interval,time_first_dose=0){
  n = floor((t-time_first_dose)/time_interval) + 1
  Cp<-Cp0*(1-exp(-k*n*time_interval))*
    exp(-k*((t-time_first_dose)-(n-1)*time_interval))/(1-exp(-k*time_interval))
  return(Cp)
}
```

Additionally, a csv with pharmacokinetic data can be uploaded to fit a curve selecting one of the pk models from the model library. Use **Estimate.PK** function for this purpose (see the help function for an example). For complex pk models or data, we recommend the use of NONMEM, MONOLIX or other parameter estimation software.

Define sensitive and resistant cells

We need to define the sensitive (Type 0) and resistant cells (Type 1) to perfom the simulations.

To define cell types the following arguments are needed:

- N0: initial cell population.

- birth_rate: birth rate function. It can be a numeric value, a user defined function or the result of a model fitting function. If the user provides a numeric value, it is assumed that the birth rate remains constant during the simulation.
- death_rate death rate function. It can be a numeric value, a user defined function or the result of a model fitting function.If the user provides a numeric value, it is assumed that the death rate remains constant during the simulation.
- scale: scaling parameter.
- pk.function: name of the pharmacokinetic function that will affect to the rates of the cells.

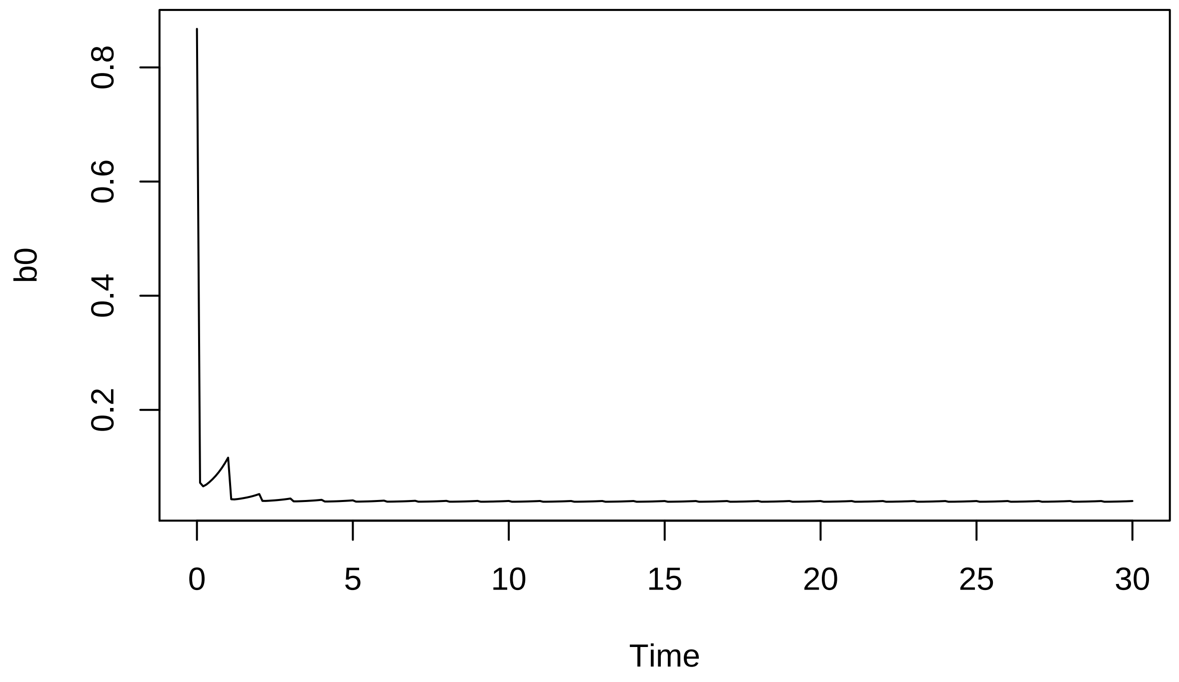
For the resistant cell types, an additional argument is needed: * mutation_rate: Numeric or function specifying the mutation rates for the cell type defined.

We will use the previously fitted functions for the birth rates and constant values for the death rates as the variation of death rate values for the different drug concentrations is very low. We assume the population begins with 10^6 sensitive cells and no resistant cells.

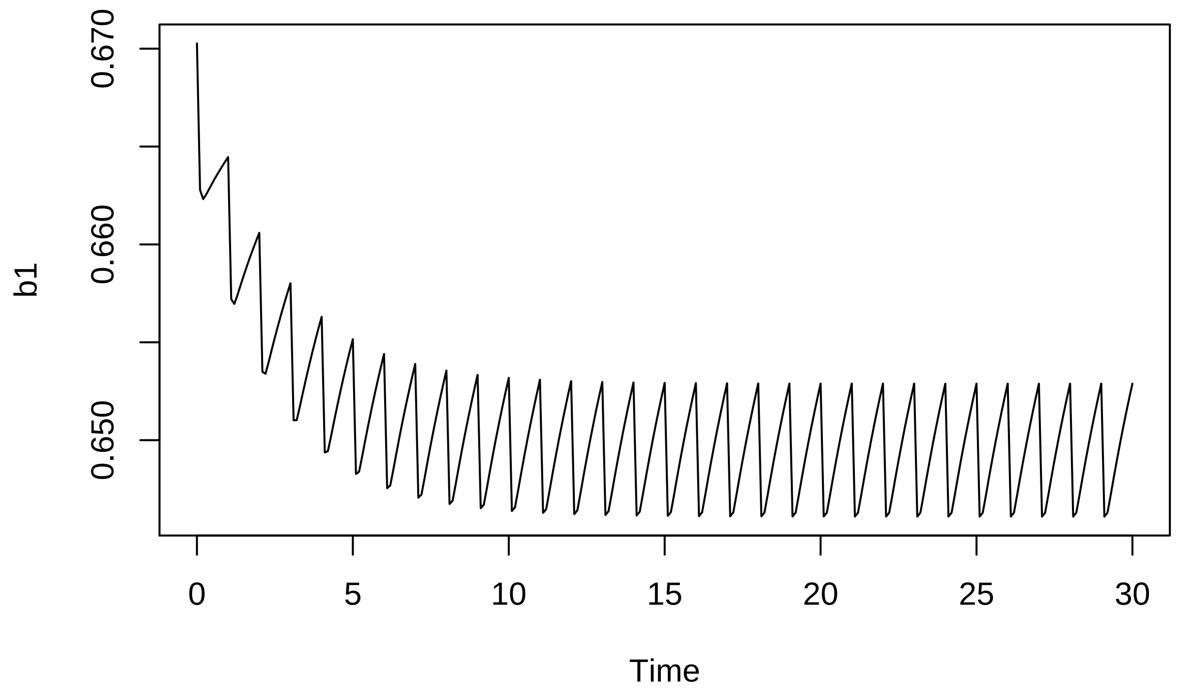
```
Type0 <-define.Type0.cells(N0=10^6,birth_rate = 'BR.fit[[1]]',death_rate= 0.0085,scale=24,
                           pk.function = 'pk')
#The function returns a S4 object with all the information gathered together.
Type0
#> An object of class "Type-0"
#> Slot "N0":
#> [1] 1e+06
#>
#> Slot "b0":
#> function (t, model = BR.fit[[1]], scale = 24)
#> {
#>     CONC1 = pk(t)
#>     BR_predict = dr.function(model, CONC = CONC1)
#>     BR_predict = BR_predict * scale
#>     return(BR_predict)
#> }
#> <environment: 0x0000000013ea4df8>
#>
#> Slot "d0":
#> function (t, scale = 24)
#> {
#>     DR_predict = 0.0085 * scale
#>     return(DR_predict)
#> }
#> <environment: 0x0000000013ea4df8>
#For resistant cells and additional argument is needed: mutation_rate
Type1 <-define.Typei.cells(Ni=0,birth_rate = 'B1.fit[[1]]',death_rate  = 0.0035, mutation_rate=10^-8,
                           scale=24, pk.function = 'pk')
```

To see how the birth rate of the cells change over time due to the pharmacokinetics of the drug:

```
plot(seq(0,30,0.1),Type0@b0(seq(0,30,0.1)),type='l',xlab='Time',ylab='b0')
```



```
plot(seq(0,30,0.1),Type1@bi(seq(0,30,0.1)),type='l',xlab='Time',ylab='b1')
```



Calculate the expected number of sensitive cells over time: In order to calculate the expected number of sensitive cells over time use **EN_type0_cells** function with the following arguments: * t: time (numeric). * type_0: Type-0 S4 object with the information of the sensitive cell population. * ui: mutation rate of the resistant population (numeric). * int.function: integration function to use in R. Options are "integrate" from base R or "pracma"which uses functions from the pracma package. "pracma" is more robust but slower than "integrate".

```
 EN_type0_cells(t=10,type_0=Type0,ui=c(Type1@ui))
#> [1] 207930.3
 EN_type0_cells(t=10,type_0=Type0,ui=c(Type1@ui),int.function="pracma")
#> [1] 207929.9
```

This function gave us the number of sensitive cells at 10 days. To do this for a vector of times:

```
sapply(seq(0,30,1),function(i){ EN_type0_cells(t=i,type_0=Type0,ui=c(Type1@ui))})
#>  [1] 1000000.000  902266.121  771239.069  655917.861  557087.012
#>  [6]  472888.704  401307.921  340511.375  288899.951  245098.237
#> [11]  207930.310  176394.121  149639.894  126940.950  107685.518
#> [16]   91355.035   77493.000   65738.039   55764.549   47305.509
#> [21]   40129.032   34041.142   28877.331   24496.842   20780.813
#> [26]   17628.285   14954.009   12685.643   10761.250    9128.720
#> [31]    7744.195
```

Calculate the number of resistant cells and probability of resistance

In order to calculate the expected number of resistant cells, use **En_resistant_cells_i** function with the following arguments:

- N: the total number of the resistant cell types.

- t: time
- type_0: Type-0 S4 object
- type_i: list with all the Type-i S4 objects
- approximation: logical argument indicating if an approximation of the numerical integration method must be used or not. Default to TRUE for faster computation.

```
En_resistant_cells(N=1,t=10,type_0=Type0,type_i=list(Type1))
```

To do this for a vector of different times:

```
sapply(c(0,2,4,8),function(i){En_resistant_cells(N=1,t=i,type_0=Type0,type_i=list(Type1))})
```

To calculate the probability of resistance use **Prob_resistance** function with the following arguments:

- t: time of production of a resistant cell clone
- type_0: Type-0 S4 object
- type_i: list with all the Type-i S4 objects
- N: number of resistant cell clones

```
#Prob_resistance(t=10,type_0=Type0,type_i=list(Type1),N=1)
#Not executed because it is a very time consuming function:
```

Demo 2: Three type branching process

Let us consider the three-type birth-death process shown in Figure 1. This model is extensively used in investigating the dynamics of tumor cells in response to treatment, where a population initially sensitive to therapy will gain resistance via mutation and expand at a different rate. In this example, the sensitive cell type (type 0, blue) is sensitive to both drugs A and B and proliferates and dies with rate b0 and d0 respectively. Type 0 cells are able to mutate to give two different cell types, type 1 (green) and 2 (orange) which will be resistant to one of the drugs.

Load libraries:

```
library(ACESO)
#> Warning: replacing previous import 'drc::gaussian' by 'stats::gaussian'
#> when loading 'ACESO'
#> Warning: replacing previous import 'drc::getInitial' by 'stats::getInitial'
#> when loading 'ACESO'
library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 3.6.2
```

Data

In the current evaluation cell viability data resulting from the exposure of BT-20 triple-negative breast cancer cell line to different concentration values of two small molecule kinase inhibitors (alpelisib and trametinib) is analyzed. The data was obtained from the HMS LINCS database (https://lincs.hms.harvard.edu/).

```
data(Alpelisib_Trametinib_combination)
head(Alpelisib_Trametinib_combination)
#>   Cell.line Drug.Name CONC CONC.units Drug2.Name  CONC2 CONC2.units
#> 1     BT-20 Alpelisib    0         uM Trametinib 0.0000          uM
#> 2     BT-20 Alpelisib    0         uM Trametinib 0.0000          uM
#> 3     BT-20 Alpelisib    0         uM Trametinib 0.0000          uM
#> 4     BT-20 Alpelisib    0         uM Trametinib 0.1235          uM
#> 5     BT-20 Alpelisib    0         uM Trametinib 0.1235          uM
#> 6     BT-20 Alpelisib    0         uM Trametinib 0.1235          uM
#>   Replicate Cell_Count_0 Viable.cells Control Time Type
#> 1         1         2751         4757    4757   72    0
#> 2         2         2751         4996    4996   72    0
#> 3         3         2751         4650    4650   72    0
#> 4         1         2751         5070    4757   72    0
#> 5         2         2751         4667    4996   72    0
#> 6         3         2751         2609    4650   72    0

DrugA=as.character(Alpelisib_Trametinib_combination$Drug.Name[1])
DrugB=as.character(Alpelisib_Trametinib_combination$Drug2.Name[1])
print(c(DrugA,DrugB))
#> [1] "Alpelisib"  "Trametinib"
```

Exploratory analysis of the data:

```
ggplot(data=Alpelisib_Trametinib_combination,aes(x=(CONC),y=Viable.cells,col=factor(CONC2)))+
  geom_point(size=1.5)+
```

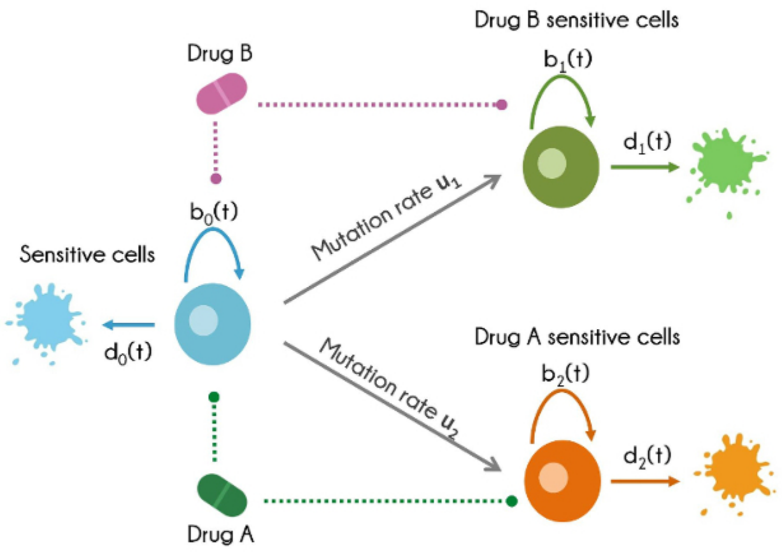
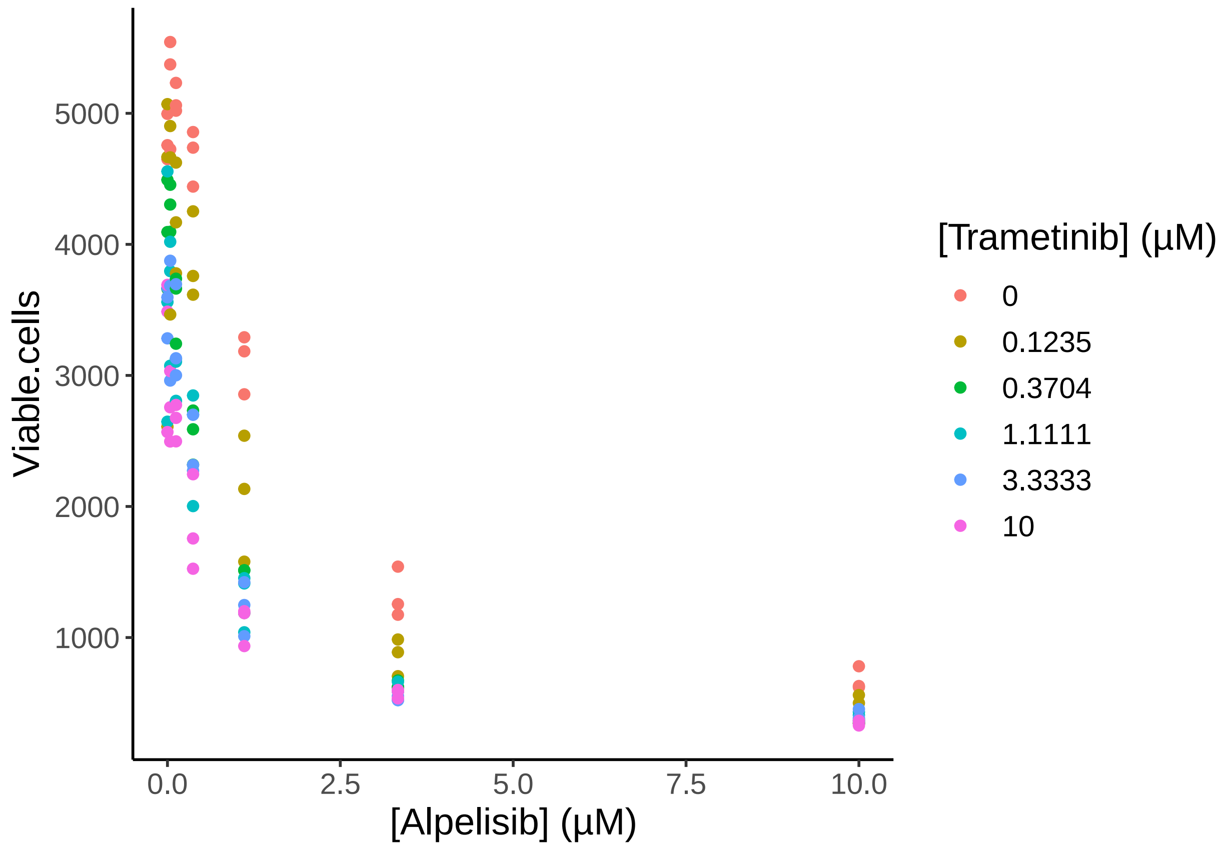


Figure 1: Three type branching process.

```
xlab(paste0("[",DrugA,"] (μM)"))+scale_colour_discrete(name=paste0("[",DrugB,"] (μM)"))+
theme_classic()+theme(text = element_text(size=14))
```



BT-20 cells sensitive to both drugs were defined as type 0 cells, whereas cells resistant to trametinib and alpelisib were defined as type 1 and type 2 cells respectively.

Dynamics of sensitive cells:

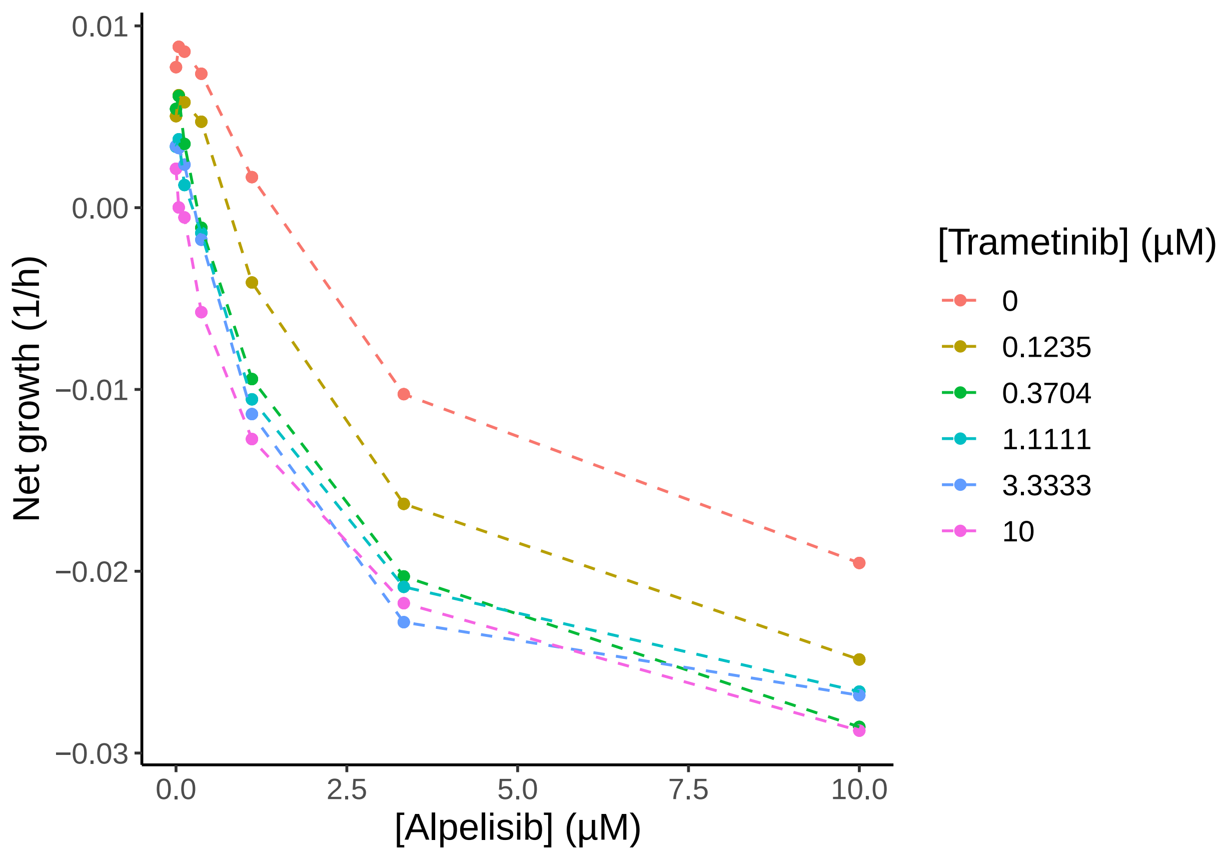
Calculate net growth rate

Calculate the net growth rates of the BT-20 cell type for every concentration of the two drugs. Use the **net_growth_rate** function for this purpose, where an exponential growth function is assumed to calculate those rates.

```
growth_data<-net_growth_rate(Alpelisib_Trametinib_combination)
head(growth_data)
#>   Cell.line Drug.Name CONC CONC.units Drug2.Name  CONC2 CONC2.units
#> 1     BT-20 Alpelisib    0         uM Trametinib 0.0000          uM
#> 2     BT-20 Alpelisib    0         uM Trametinib 0.0000          uM
#> 3     BT-20 Alpelisib    0         uM Trametinib 0.0000          uM
#> 4     BT-20 Alpelisib    0         uM Trametinib 0.1235          uM
#> 5     BT-20 Alpelisib    0         uM Trametinib 0.1235          uM
#> 6     BT-20 Alpelisib    0         uM Trametinib 0.1235          uM
#>   Replicate Cell_Count_0 Viable.cells Control Time Type  Net_growth
#> 1         1         2751         4757    4757   72    0 0.007727910
#> 2         2         2751         4996    4996   72    0 0.007727910
#> 3         3         2751         4650    4650   72    0 0.007727910
#> 4         1         2751         5070    4757   72    0 0.005032087
#> 5         2         2751         4667    4996   72    0 0.005032087
#> 6         3         2751         2609    4650   72    0 0.005032087
#>   Death_rate Birth_rate
#> 1         NA         NA
#> 2         NA         NA
#> 3         NA         NA
#> 4         NA         NA
#> 5         NA         NA
#> 6         NA         NA
```

Plot the results:

```
ggplot(data=growth_data,aes(x=(CONC),y=Net_growth,col=factor(CONC2)))+geom_point()+
  geom_line(linetype="dashed")+
  theme(text = element_text(size=14))+xlab(paste0("[",DrugA,"] (µM)"))+
  scale_colour_discrete(name=paste0("[",DrugB,"] (µM)"))+
  theme_classic()+ylab("Net growth (1/h)")+theme(text = element_text(size=14))
```



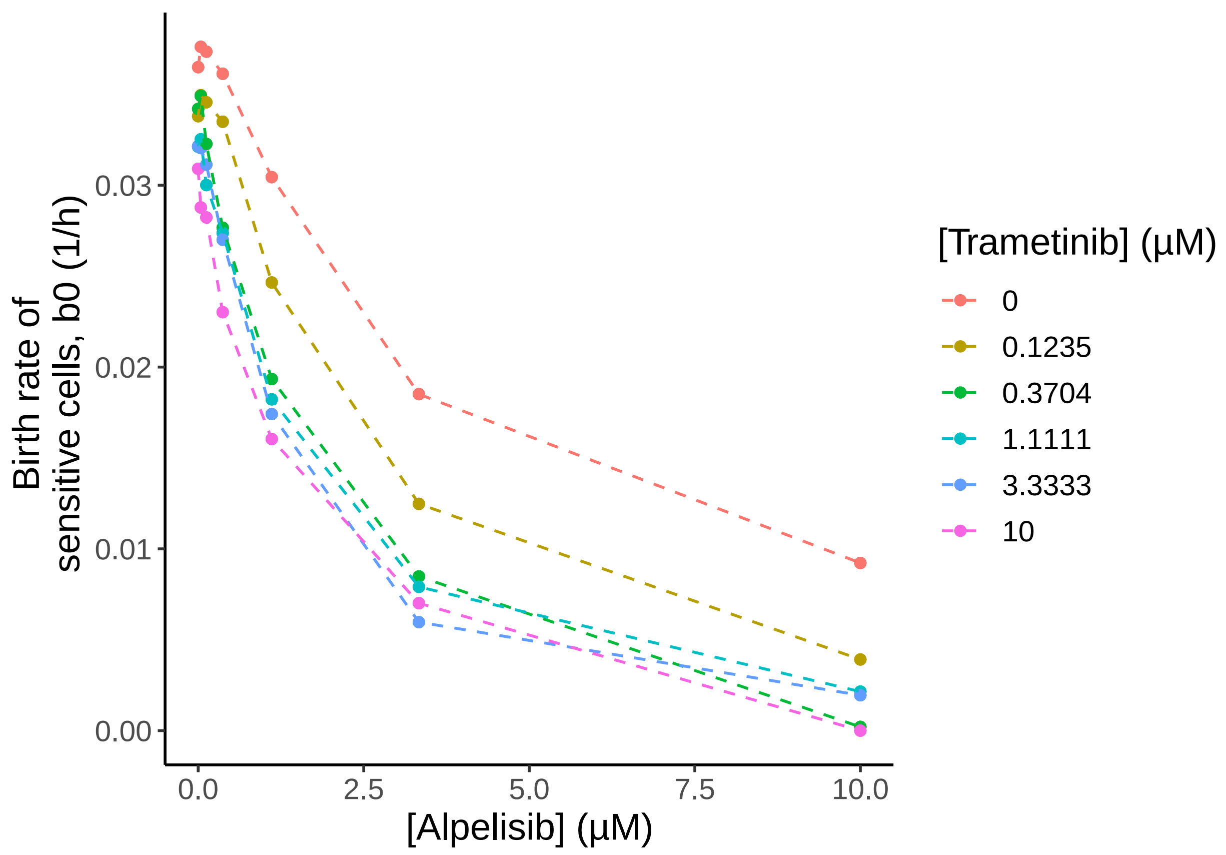
Calculate birth and death rates

As in the LINCS database there is no information regarding apoptosis assays available to estimate the death rate of type 0 cells (d0) and assuming that these targeted therapies do not induce cell death per se, a different d0 value was defined for each combination of the drugs, which was equal to the minimum value needed to ensure that all the birth rates of type 0 cells (b0) were positive regardless of the treatment conditions:

```
d0=vector("list", length = length(unique(growth_data$Cell.line)))
names(d0)=unique(as.character(growth_data$Cell.line))
d0[[1]]=-min(growth_data$Net_growth)
d0
#> $`BT-20`
#> [1] 0.02876796
```

We use the **net_growth_rate** function again, not only to calculate the net growth rate parameter of the cells but also the birth and death rates. To this end, we introduce the newly calculated d0 value:

```
growth_data<-net_growth_rate(growth_data,death_rate = d0)
ggplot(data=growth_data,aes(x=(CONC),y=Birth_rate,col=factor(CONC2)))+
  geom_point()+geom_line(linetype="dashed")+
  theme(text = element_text(size=14))+xlab(paste0("[",DrugA,"] (µM)"))+
  scale_colour_discrete(paste0("[",DrugB,"] (µM)"))+
  theme_classic()+ylab("Birth rate of \n sensitive cells, b0 (1/h)")+
  theme(text = element_text(size=14))
```

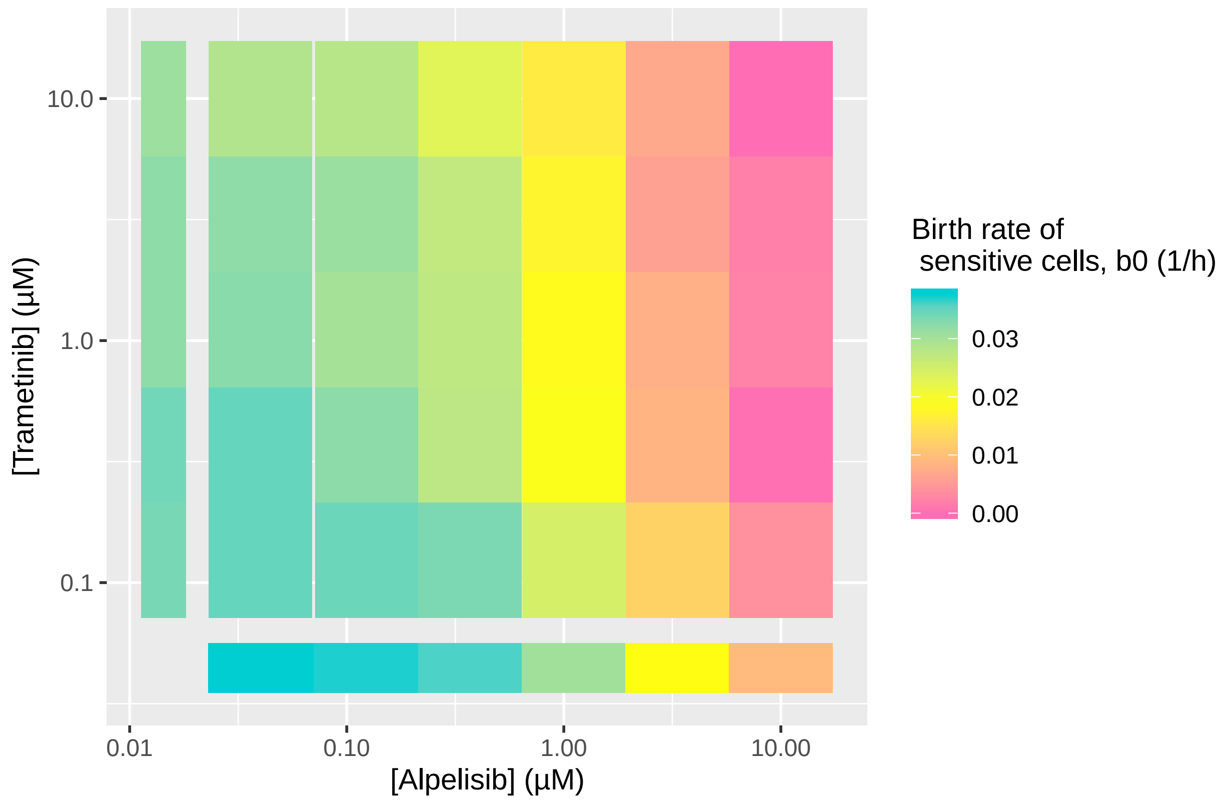


Another plot type to analyze the response surface of the data:

```
#We are going to remove the columns that are not necessary for the analysis of the data in this example
GD=growth_data[,c('Cell.line','CONC','CONC2','Net_growth','Type','Birth_rate','Death_rate')]
GD=unique(GD)
```

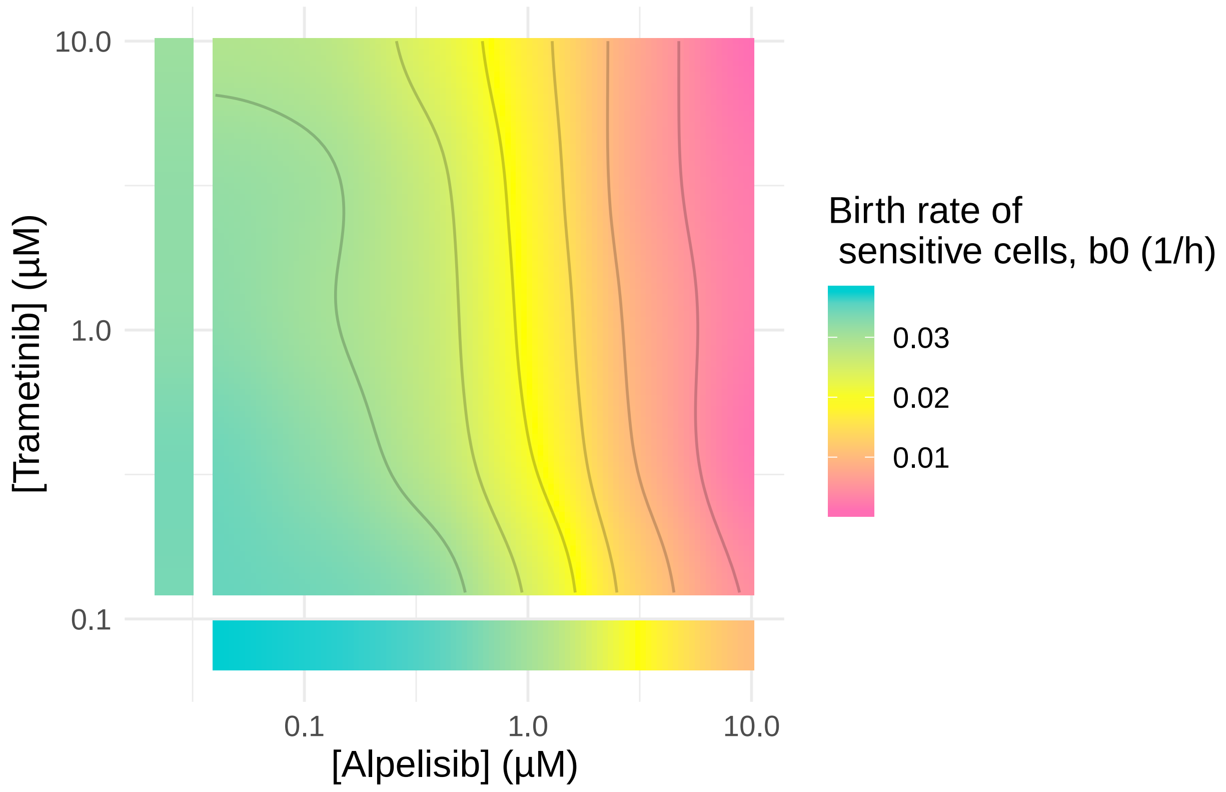
Use **responseMap** function to reorganize the data to easily plot the surface of the birth rate values versus both drug concentrations:

```
rmap <- responseMap(Birth_rate~CONC+CONC2,GD,logscale=T,interpolate=FALSE)
DifferenceSurface.plot(rmap,zcenter=max(growth_data$Birth_rate)/2,
                       xl=paste0("[",DrugA,"] (µM)"),
                       yl=paste0("[",DrugB,"] (µM)"),
                       zl="Birth rate of \n sensitive cells, b0 (1/h)",
                       mid="yellow",low="hotpink1",high="darkturquoise")
```



Use **responseMap** again and **plot.ResponseSurface** to make a contour plot of the data. For this plot a interpolation of the data is needed (see logscale=T in the **responseMap** function):

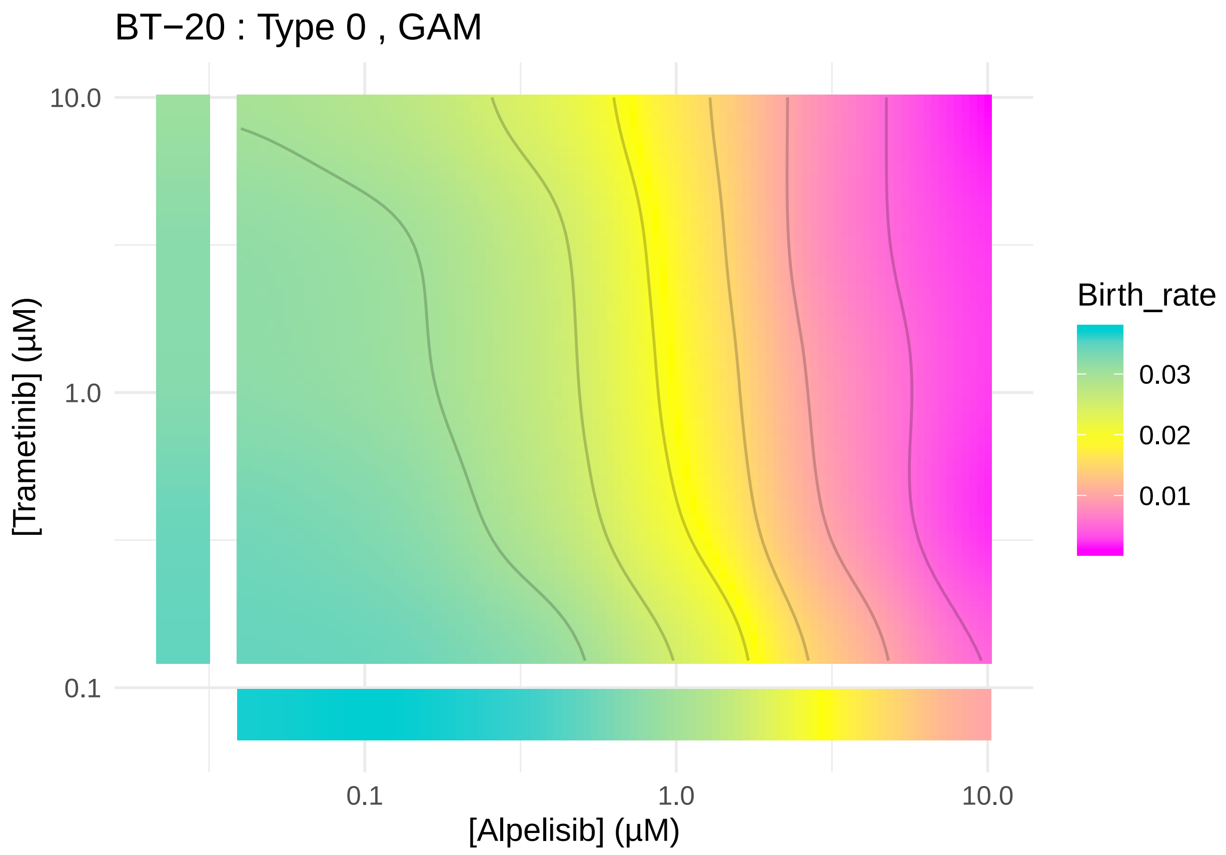
```
rmap <- responseMap(Birth_rate~CONC+CONC2,GD,logscale=T)

ResponseSurface.plot(rmap,xl=paste0("[",DrugA,"] (µM)"),
                     yl=paste0("[",DrugB,"] (µM)"),
                     zl="Birth rate of \n sensitive cells, b0 (1/h)",
                     palette=c("hotpink1","yellow","darkturquoise"))
```



Sensitive cell model fitting

Birth rate parameters were fitted non-parametrically using a Generalized Additive Model (GAM) to generate the predicted surface required for the simulation of the evolutionary process:

```
gam.model=Multiple.resp.surface.fit(GD, title=", GAM",Drug1.name=paste0("[",DrugA,"] (µM)"),
                                    Drug2.name=paste0("[",DrugB,"] (µM)"))
```

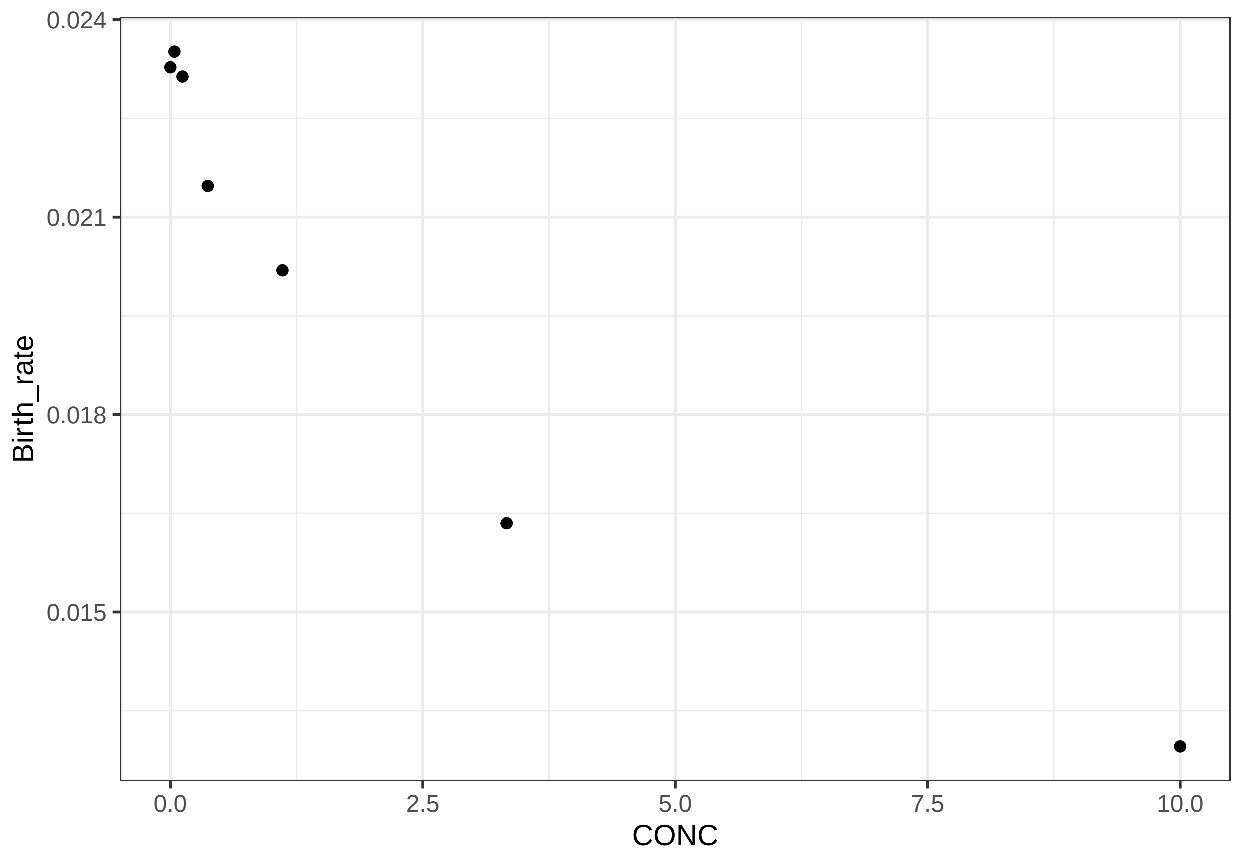


Dynamics of resistant cells:

In the HMS LINCS database there is no information about resistant cell lines for these drugs, even so, we created the data to evaluate the dynamics of Type 1 (resistant to trametinib) and Type 2 (resistant to alpelisib) cells to show a more complete example of the analysis that can be conducted using ACESO.

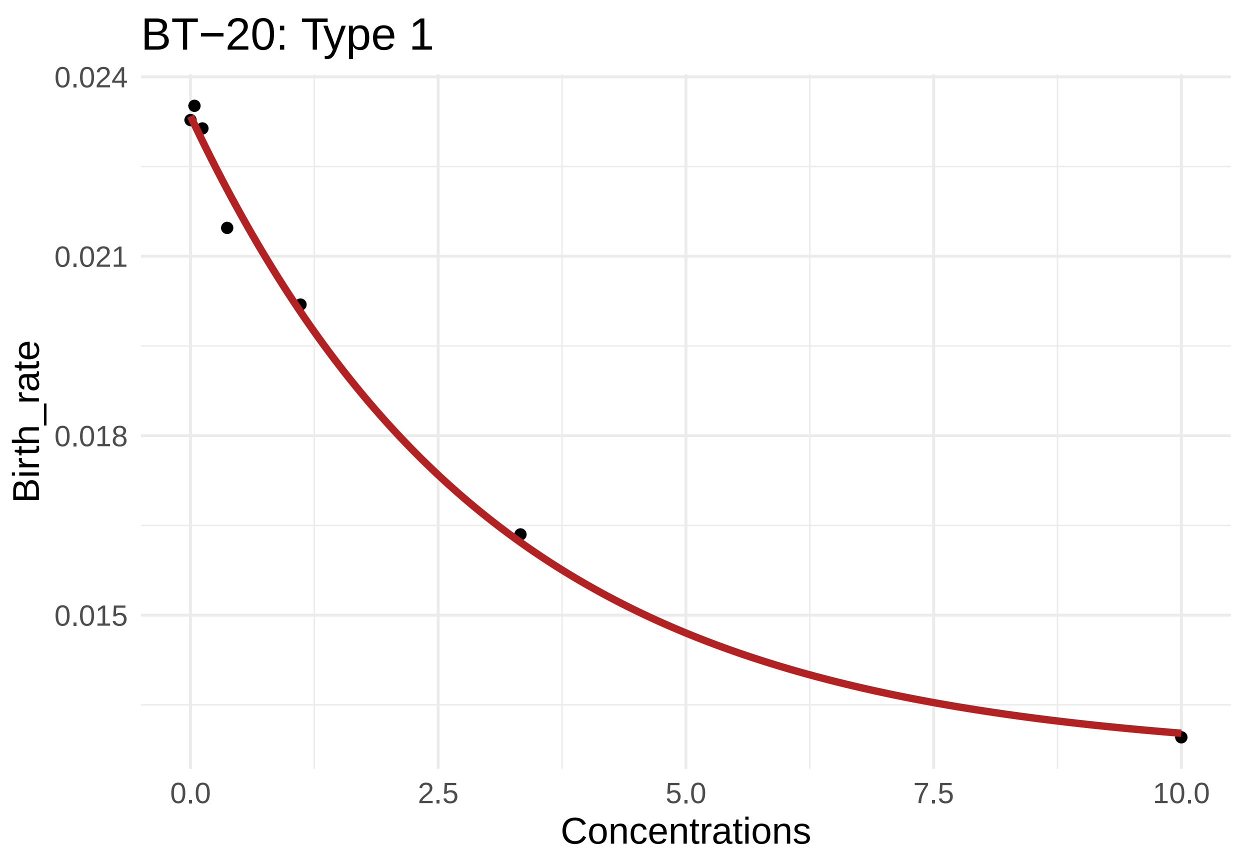
Let's start analyzing the dynamics of alpelisib-sensitive/trametinib-resistant cell type (Type 1).

```
data(Alpelisib_sensitive)
head(Alpelisib_sensitive)
#>   Cell.line Time CONC  Net_growth Type Birth_rate Death_rate
#> 1     BT-20   72 0.00 0.008277157    1 0.02327716      0.015
#> 2     BT-20   72 0.04 0.008515068    1 0.02351507      0.015
#> 3     BT-20   72 0.12 0.008136471    1 0.02313647      0.015
#> 4     BT-20   72 0.37 0.006474282    1 0.02147428      0.015
#> 5     BT-20   72 1.11 0.005192139    1 0.02019214      0.015
#> 6     BT-20   72 3.33 0.001349469    1 0.01634947      0.015
ggplot(data=Alpelisib_sensitive,aes(x=CONC,y=Birth_rate))+geom_point()+theme_bw()
```



Fit the birth rates with the **Multiple.best.singlefit** function:

```
#Some error messages might appear because not all the models tested are able to fit the data. Ignore th
B1=Multiple.best.singlefit(data=Alpelisib_sensitive,resp='Birth_rate')
#> Error in optim(startVec, opfct, hessian = TRUE, method = optMethod, control = list(maxit = maxIt,  :
#>   non-finite finite-difference value [2]
#> Error in optim(startVec, opfct, hessian = TRUE, method = optMethod, control = list(maxit = maxIt,  :
#>   non-finite finite-difference value [2]
```

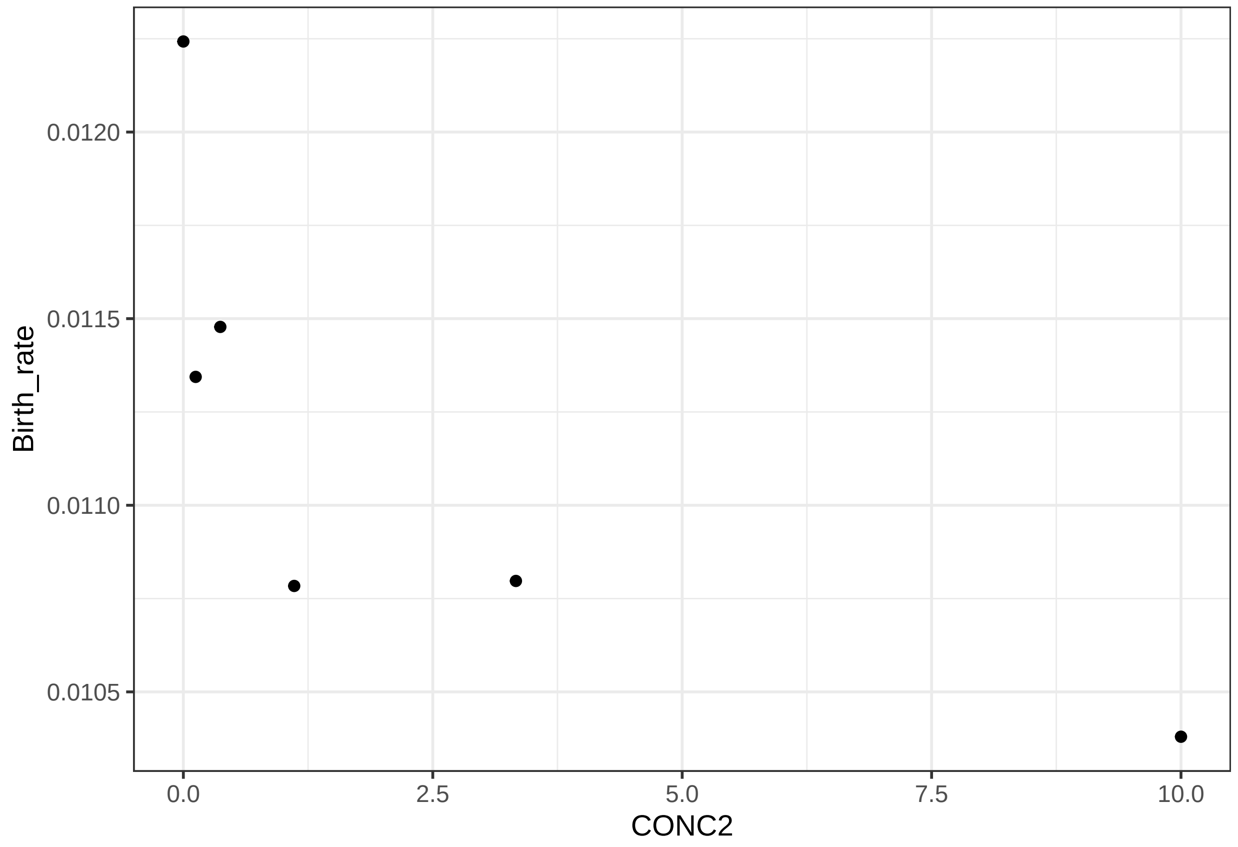


```
B1
#> [[1]]
#>
#> A 'drc' model.
#>
#> Call:
#> drc::drm(formula = Birth_rate ~ CONC, data = data, fct = EXD.3(),     type = "continuous")
#>
#> Coefficients:
#> c:(Intercept)  d:(Intercept)  e:(Intercept)
#>       0.01262        0.02334        3.04795
```

The best function is a expontenial decay model (named EXD.3) with an upper limit of 0.023 1/h, a lower limit of 0.0126 1/h and a steepness of decay of 3.047 µM.

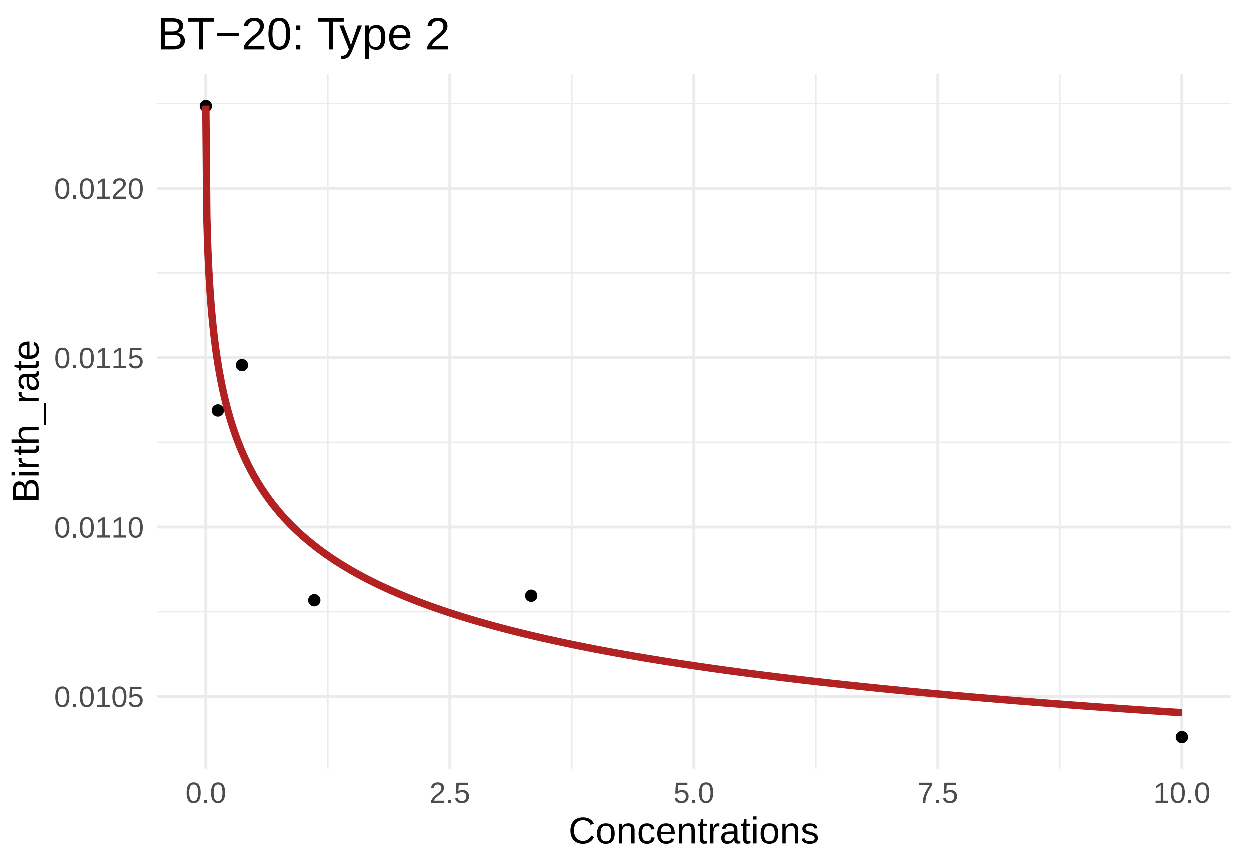
Now, we will repeat this process for the trametinib-sensitive/alpelisib-resistant cell type (Type 2).

```
 data(Trametinib_sensitive)
 head(Trametinib_sensitive)
#>    Cell.line Time   CONC2   Net_growth Type Birth_rate  Death_rate
#> 1      BT-20   72  0.0000 0.0025759700    2 0.01224264 0.009666667
#> 4      BT-20   72  0.1235 0.0016773624    2 0.01134403 0.009666667
#> 7      BT-20   72  0.3704 0.0018112090    2 0.01147788 0.009666667
#> 10     BT-20   72  1.1111 0.0011172200    2 0.01078389 0.009666667
#> 13     BT-20   72  3.3333 0.0011305640    2 0.01079723 0.009666667
#> 16      BT-20   72 10.0000 0.0007132301    2 0.01037990 0.009666667
ggplot(data=Trametinib_sensitive,aes(x=CONC2,y=Birth_rate))+geom_point()+theme_bw()
```



Fit the birth rates with the **Multiple.best.singlefit** function:

```
B2=Multiple.best.singlefit(data=Trametinib_sensitive,resp='Birth_rate',conc = "CONC2")
```



```
B2
#> [[1]]
#> 
#> A 'drc' model.
#> 
#> Call:
#> drc::drm(formula = Birth_rate ~ CONC, data = data, fct = LL.4(),     type = "continuous")
#> 
#> Coefficients:
#> b:(Intercept)  c:(Intercept)  d:(Intercept)  e:(Intercept)  
#>       0.42935        0.00988        0.01224        0.70025
```

If the user wants to check all the models that have been compared for a particupar cell type, **best.singlefit** can be used. With the following code, we check all the models tested for the birth rates of the alpelisib-resistant cell line (Type=2):

```
best.singlefit(Trametinib_sensitive,resp="Birth_rate",conc = "CONC2",compare=T)
#>           logLik        IC      Res var
#> LL.4  44.442283 -78.88457 6.470941e-08
#> W1.4  43.723470 -77.44694 8.222925e-08
#> LL.5  44.667096 -77.33419 1.200750e-07
#> LN.4  43.635385 -77.27077 8.467942e-08
#> EXD.3 41.791340 -75.58268 1.043854e-07
#> W2.4  41.829850 -73.65970 1.545809e-07
#> G.4   41.124185 -72.24837 1.955741e-07
#> L.4   39.324403 -68.64881 3.563334e-07
#> L.5   40.070275 -68.14055 5.557896e-07
#> EXD.2 29.570619 -53.14124 4.600778e-06
#> LN.3  26.409182 -44.81836 1.759685e-05
#> LL.3  26.408345 -44.81669 1.760176e-05
#> L.3   25.076814 -42.15363 2.743563e-05
#> W1.3  23.823770 -39.64754 4.165928e-05
#> G.3   22.513184 -37.02637 6.448199e-05
#> G.2   20.689405 -35.37881 8.882163e-05
#> W1.2  -3.065792  12.13158 2.440258e-01
#> LL.2  -3.065872  12.13174 2.440323e-01
```

Define drug pharmacokinetic model:

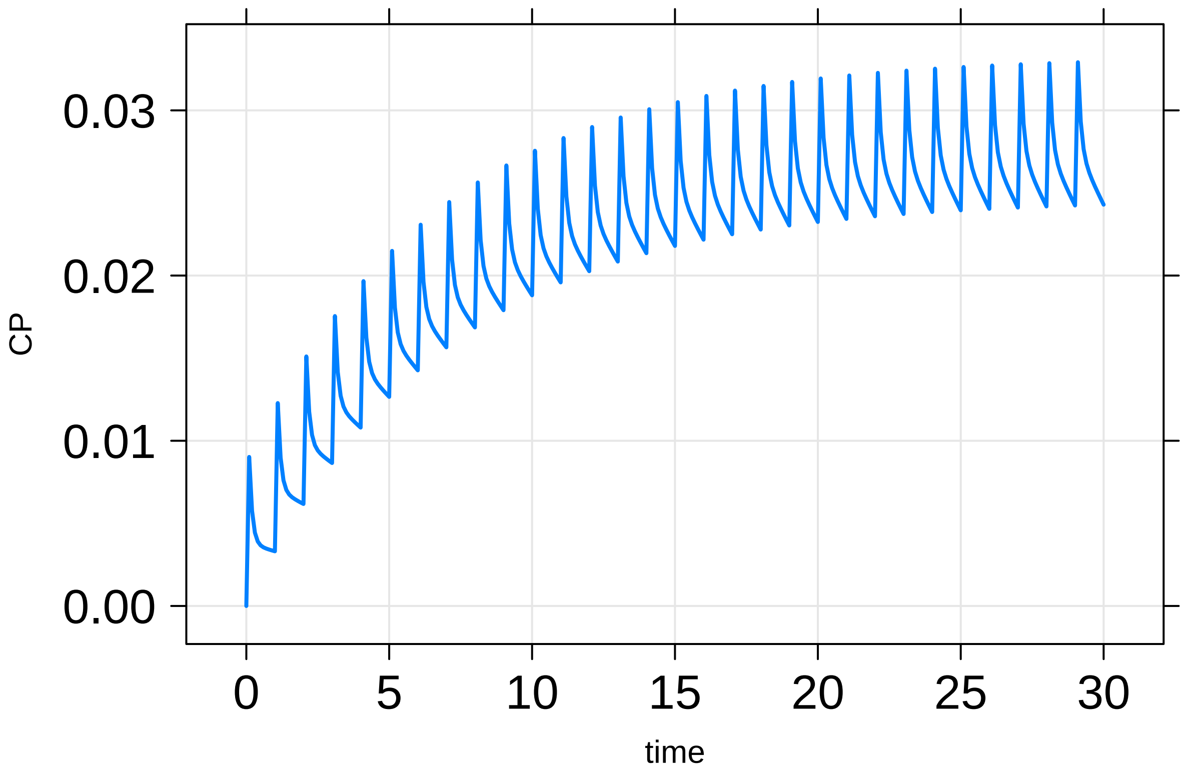
The pharmacokinetic model defined for alpelisib is based on the model from De Buck et al. 2014, whereas for trametinib we used the model defined in Ouellet et al. 2016.

```
library(mrgsolve)
#>
#> Attaching package: 'mrgsolve'
#> The following object is masked from 'package:stats':
#>
#>     filter
#Alpelisib
cmt1_oral<- mread("1cmt_ev", model_library())
#> Building 1cmt_ev ...
#> done.
#Trametinib:
cmt2_oral<- mread("2cmt_ev", model_library())
#> Building 2cmt_ev ... done.
see(cmt2_oral)
#>
#> Model file:  2cmt_ev.cpp
#> $PARAM @annotated
#>   TVCL   :  2 : Clearance (volume/time)
#>   TVV    : 20 : Central volume (volume)
#>   TVKA   : 1 : Absorption rate constant (1/time)
#>   Q    :  2 : Inter-compartmental clearance (volume/time)
#>   Vp   : 10 : Peripheral volume of distribution (volume)
#>   F: 1 : bioavailability
#>   ALAG  : 0 : Lag time (time)
#>
#>   $CMT  @annotated
#>   EV   : Extravascular compartment
#>   CENT : Central compartment
#>   PERIPH : Peripheral compartment
#>
#>   $MAIN
#>   double CL = exp(log(TVCL) + ETA_CL);
#>   double V2 = exp(log(TVV)  + ETA_V);
#>   double KA = exp(log(TVKA)  + ETA_KA);
#>   double V3 = exp(log(Vp)  + ETA_Vp);
```

```
#>
#> ALAG_EV = ALAG;
#> F_EV = F;
#>
#> $OMEGA @labels ETA_CL ETA_V ETA_KA ETA_Vp
#>   0 0 0 0
#>
#> $GLOBAL
#> #define CP (CENT/V2)
#>
#>   $PKMODEL ncmt = 2, depot = TRUE
#>
#>   $CAPTURE @annotated
#>   CP : Plasma concentration (mass/volume)
```

Simulate the drug concentration profile for trametinib when a 2mg dose is given every day for a month:

```
e2 <- ev(amt = 2, time=0, ii=1, addl=30)
easy.mrgsim(model=cmt2_oral,dosing_schedule=e2,delta=0.1,tend=30,parameters = list(TVCL=4.91*24,TVV=214
  plot(scales=list(cex=1.5))
```



Define all the cell types:

Now we need to use **define.Type0.cells** and **define.Typei.cells** functions to define all the cell types that we have in our problem.

To define cell types the following arguments are needed:

- N0: initial cell population.
- birth_rate: birth rate function. It can be a numeric value, a user defined function or the result of a model fitting function. If the user provides a numeric value, it is assumed that the birth rate remains constant during the simulation.
- death_rate death rate function. It can be a numeric value, a user defined function or the result of a model fitting function.If the user provides a numeric value, it is assumed that the death rate remains constant during the simulation.
- scale: scaling parameter.
- pk.function: name of the pharmacokinetic function that will affect to the rates of the cells.

For the resistant cell types, an additional argument is needed:

- mutation_rate: Numeric or function specifying the mutation rates for the cell type defined.

To start with our simulations, we will define a initial number of 10ˆ6 sensitive cells and 1000 resistant cells of each type. The mutation rate is equal to 10ˆ-7.

```
Type0 <-define.Type0.cells(N0=10^6,birth_rate = 'gam.model[[1]]',death_rate= 0.03,scale=24,
                           pk.function = c('pk1','pk2'))

Type1 <-define.Typei.cells(Ni=1000,birth_rate = 'B1[[1]]',death_rate  = 0.015,mutation_rate=10^-7,
                           scale=24,pk.function='pk1')

Type2 <-define.Typei.cells(Ni=1000,birth_rate = 'B2[[1]]',death_rate  = 0.01,mutation_rate=10^-7,
                           scale=24,pk.function='pk2')
```

Calculate the number of sensitive and resistant cells over time:

First define the dosing schedules for each drug:

```
#Alpelisib: 300mg/day
e1 <- ev(amt = 300, ii = 1, addl = 60, time=0)
pk1=pk.function(model=cmt1_oral,dosing_schedule=e1,tend=50,parameters = list(TVCL=11.5*24,TVV=118,TVKA=
#> Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
#> unique 'x' values

#Trametinib: 2mg/day
e2 <- ev(amt = 2, time=0, ii=1, addl=60)
pk2=pk.function(model=cmt2_oral,dosing_schedule=e2,tend=50,parameters = list(TVCL=4.91*24,TVV=214,TVKA=
#> Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
#> unique 'x' values
```

Calculate the number of sensitive cells after a week of treatment (7 days):

```
EN_type0_cells(t=7,type_0=Type0,ui=c(Type1@ui,Type2@ui))
#> [1] 315800.1
```

Remember the meaning of each argument: * t: time. * type_0: Type-0 S4 object with the information of the sensitive cell population. * ui: mutation rates of the resistant population.

```
En_resistant_cells(N=2,t=7,type_0=Type0,type_i=list(Type1,Type2))
#>           t=7
#> [1,] 1631.969
#> [2,] 1376.529
```

- N: the total number of the resistant cell types. 2 in this example.
- t: time
- type_0: Type-0 S4 object
- type_i: list with all the Type-i S4 objects

Demo 3: Complex dosing schedules

We are going to simulate complex dosing schedules using mrgsolve and ACESO packages.

```
#' Load libraries
library(ACESO)
#> Warning: replacing previous import 'drc::gaussian' by 'stats::gaussian'
#> when loading 'ACESO'
#> Warning: replacing previous import 'drc::getInitial' by 'stats::getInitial'
#> when loading 'ACESO'
library(mrgsolve)
#> Warning: package 'mrgsolve' was built under R version 3.5.3
#>
#> Attaching package: 'mrgsolve'
#> The following object is masked from 'package:stats':
#>
#>     filter
#install mrgsolve from github: devtools::install_github("metrumresearchgroup/mrgsolve")
```

Pharmacokinetic model

We are going to simulate a one compartment model with intravenous administration.

Read the model from the model libraries included in ACESO:

```
model_library(list=T)
#> ACESO internal library of PK models:
#> [1] "1cmt_2depot" "1cmt_ev"     "1cmt_iv"     "2cmt_2depot" "2cmt_ev"
#> [6] "2cmt_iv"     "3cmt_ev"     "3cmt_iv"

cmt1_iv <- mread("1cmt_iv", model_library()) %>% Req(CP)
#> Building 1cmt_iv ...
#> done.
#See the code of the model
see(cmt1_iv)
#>
#> Model file:  1cmt_iv.cpp
#> $PARAM @annotated
#>   TVCL   :  1 : Clearance (volume/time)
#>   TVV    : 20 : Central volume (volume)
#>
#>   $CMT  @annotated
#>   CENT : Central compartment
#>
#>   $MAIN
#>   double CL = exp(log(TVCL) + ETA_CL);
#>   double V = exp(log(TVV)  + ETA_V);
#>
#>
#> $OMEGA @labels ETA_CL ETA_V
#>   0 0
#>
#>
#> $GLOBAL
#> #define CP (CENT/V)
#>
#>   $PKMODEL ncmt = 1, depot = FALSE
#>
#>   $CAPTURE @annotated
#>   CP : Plasma concentration (mass/volume)
```

See the parameters of the model with the default values:

```
param(cmt1_iv)
#>
#>  Model parameters (N=2):
#>  name value . name value
#>  TVCL 1     | TVV  20
```

We are going to change the default parameter values for TVCL (typical value for the CLearance) and TVV (Typical value for the Volume of distribution):

```
newpar <-  list('TVCL' = 171.4, #L/h
               'TVV'  = 153.5)
cmt1_iv<-param(cmt1_iv,newpar)

param(cmt1_iv)
#>
#>  Model parameters (N=2):
#>  name value . name value
#>  TVCL 171   | TVV  154
```

Event objects in mrgsolve

First dosing schedule: 150mg dose at time 0, once every day in a week

```
e1 <- ev(amt = 150, ii = 1, addl = 6, time=0)
e1
#> Events:
#>   time cmt amt evid ii addl
#> 1    0   1 150    1  1    6
```

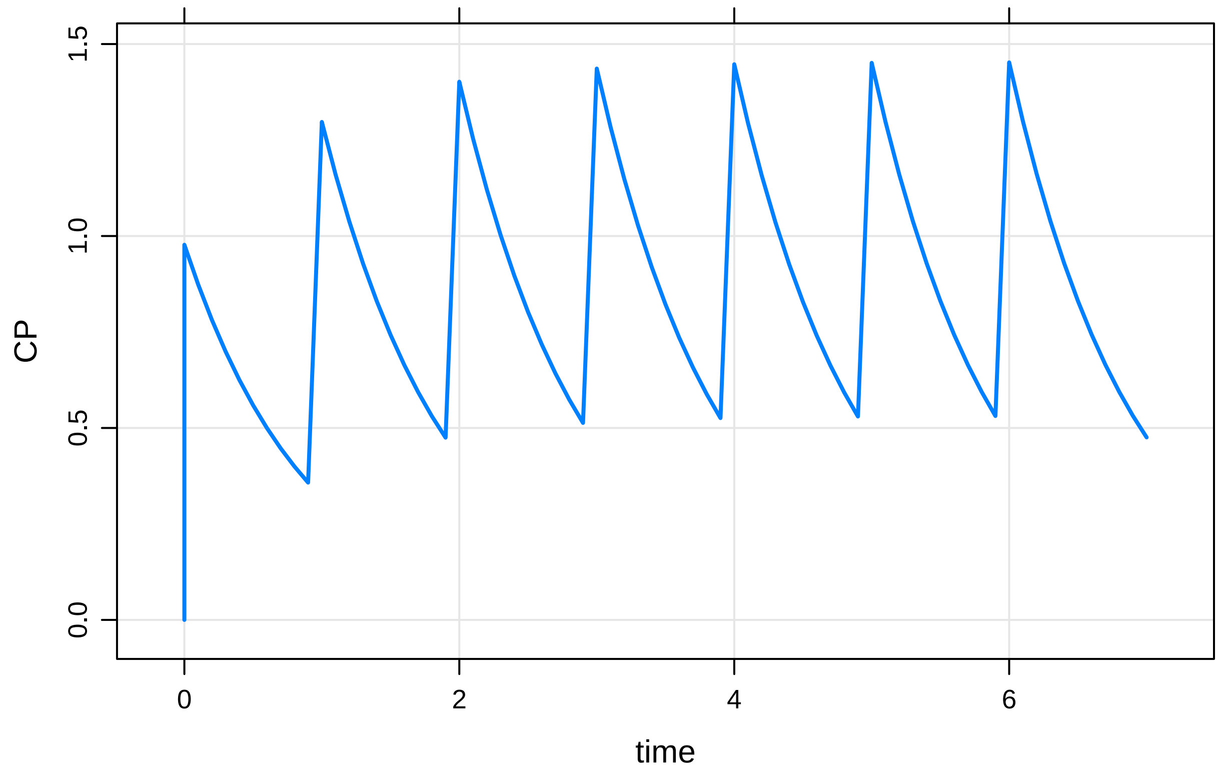
- amt: amount of administered dose
- ii: time interval between doses
- addl: additional doses
- time: time of first dose
- cmt: compartment number (in which compartment a dosing is ocurring)
- evid: type of record (1: defines the record as a dose event)

Simulate

Simulate the model with easy.mrgsim function (extended version of mrgsim function in mrgsolve package)

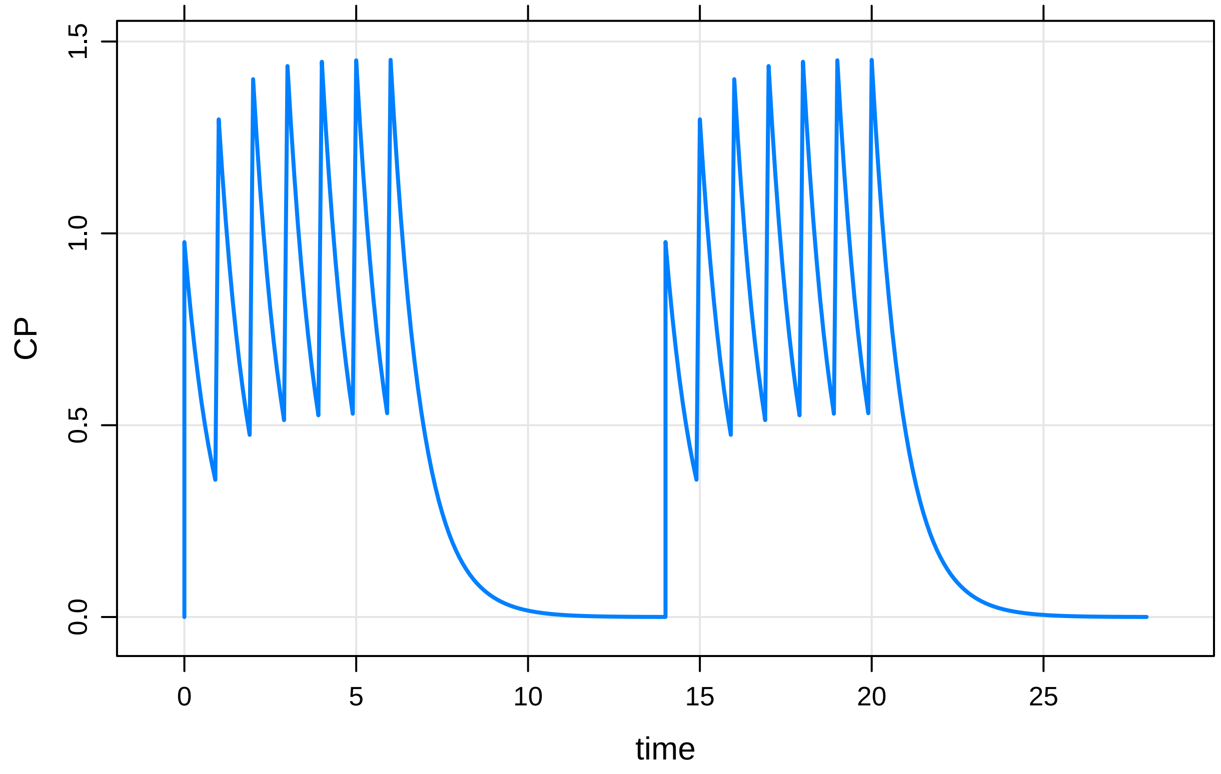
- model: model name
- dosing_schedule: event object where the dosing schedule is specified
- tend: final time
- delta: the increment of time to simulate a sequence from 0 to tend.

```
easy.mrgsim(model=cmt1_iv,dosing_schedule=e1,delta=0.1,tend=7) %>% plot
```



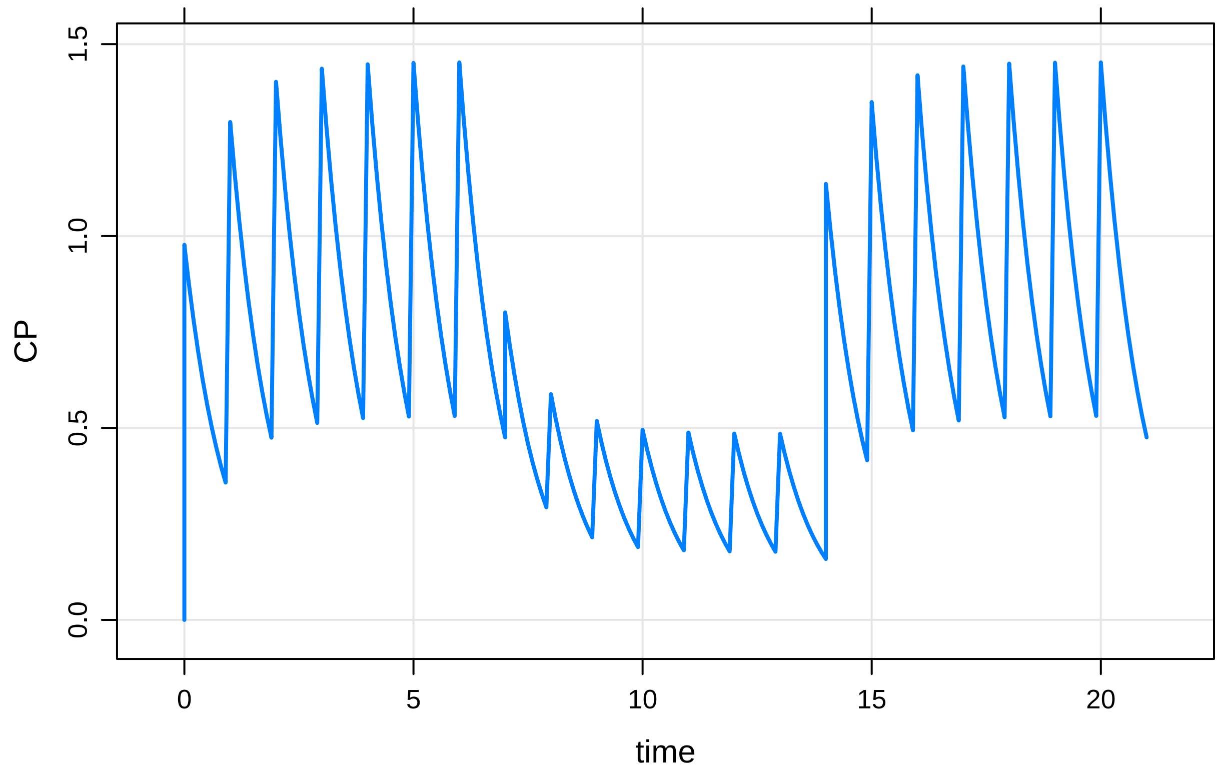
1 week of holiday between each treatment period:

```
e2 <- seq(e1, wait = 7, e1, wait = 7)
e2
#> Events:
#>   time cmt amt evid ii addl
#> 1    0   1 150    1  1    6
#> 2   14   1 150    1  1    6
easy.mrgsim(model=cmt1_iv,dosing_schedule=e2,delta=0.1,tend=7*4) %>% plot
```



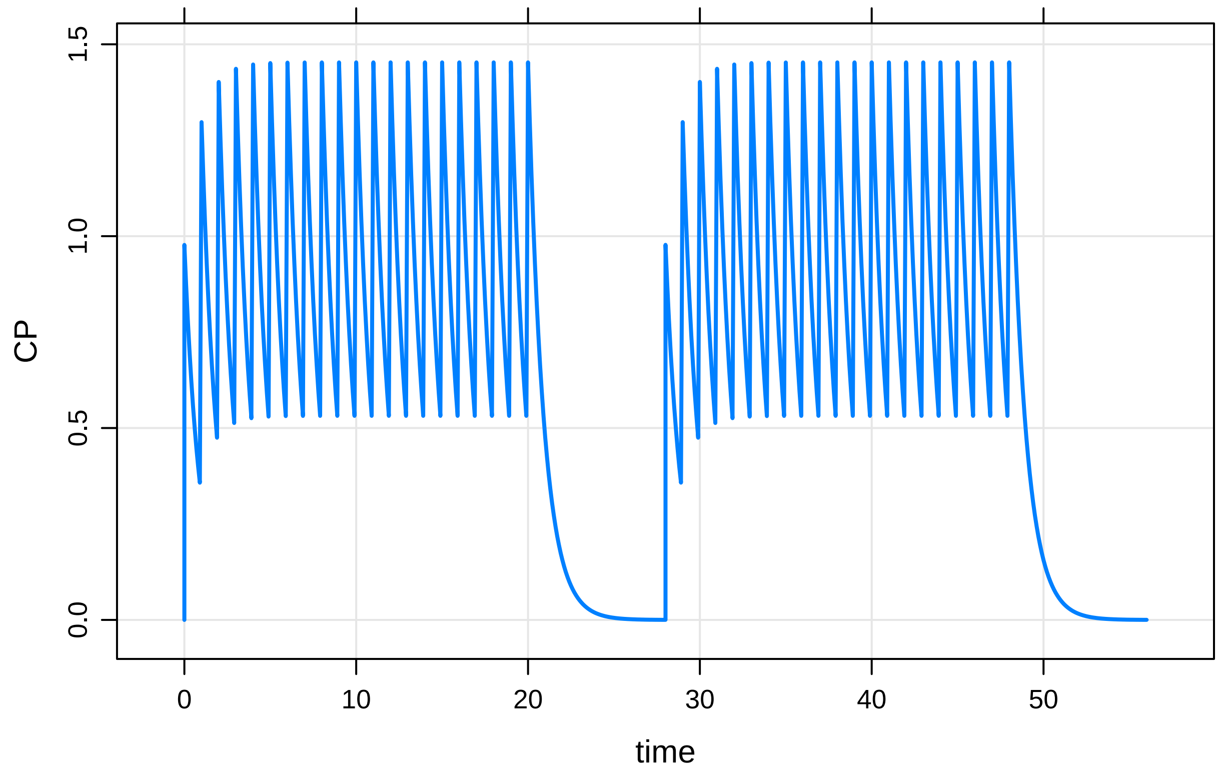
A alternating dosing sequence: 150 mg daily for a week, then 50 mg daily for a week etc.

```
e3 <- ev(amt = 50, ii = 1, addl = 6)
e4 <- seq(e1, e3, e1)
e4
#> Events:
#>   time cmt amt evid ii addl
#> 1    0   1 150    1  1    6
#> 2    7   1  50    1  1    6
#> 3   14   1 150    1  1    6
easy.mrgsim(model=cmt1_iv,dosing_schedule=e4,delta=0.1,tend=7*3) %>% plot
```



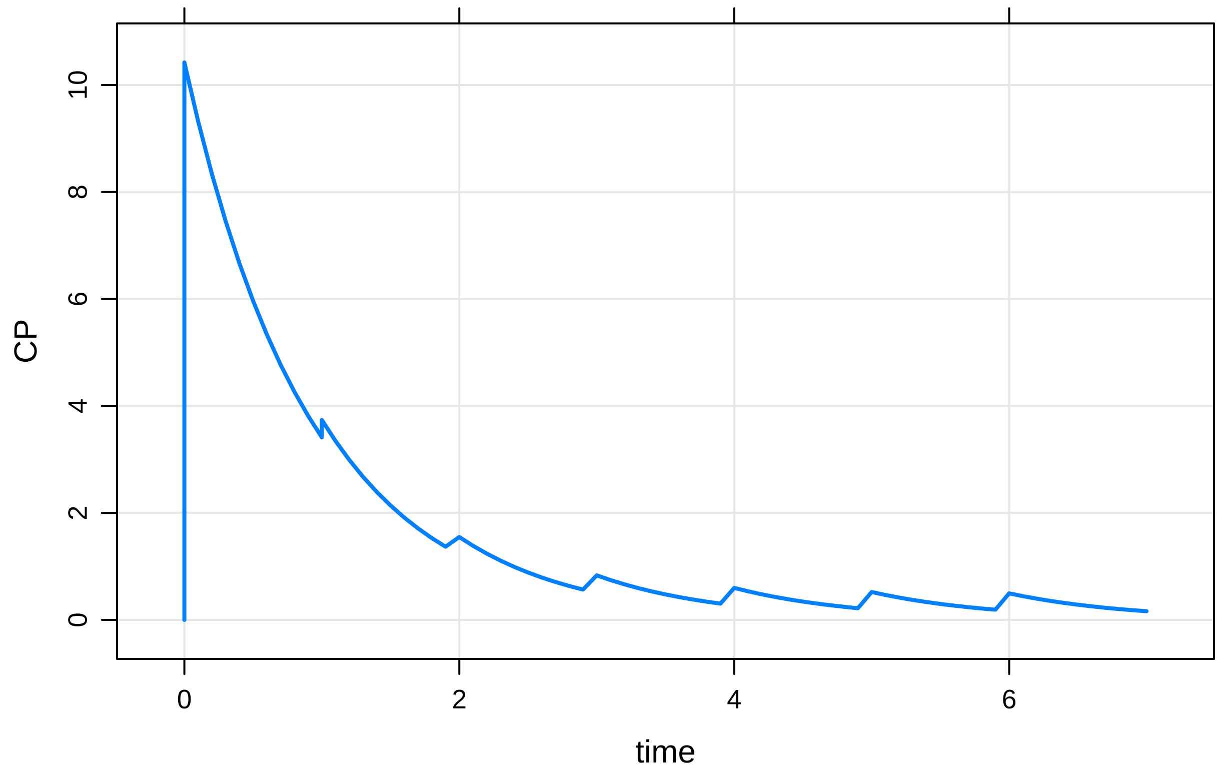
A cycle of 3 weeks on, one week off for two months

```
e5 <- ev(amt = 150, ii = 1, addl = (3*7)-1)
e6 <- seq(e5, wait = 7, e5)
easy.mrgsim(model=cmt1_iv,dosing_schedule=e6,delta=0.1,tend=7*4*2) %>% plot
```



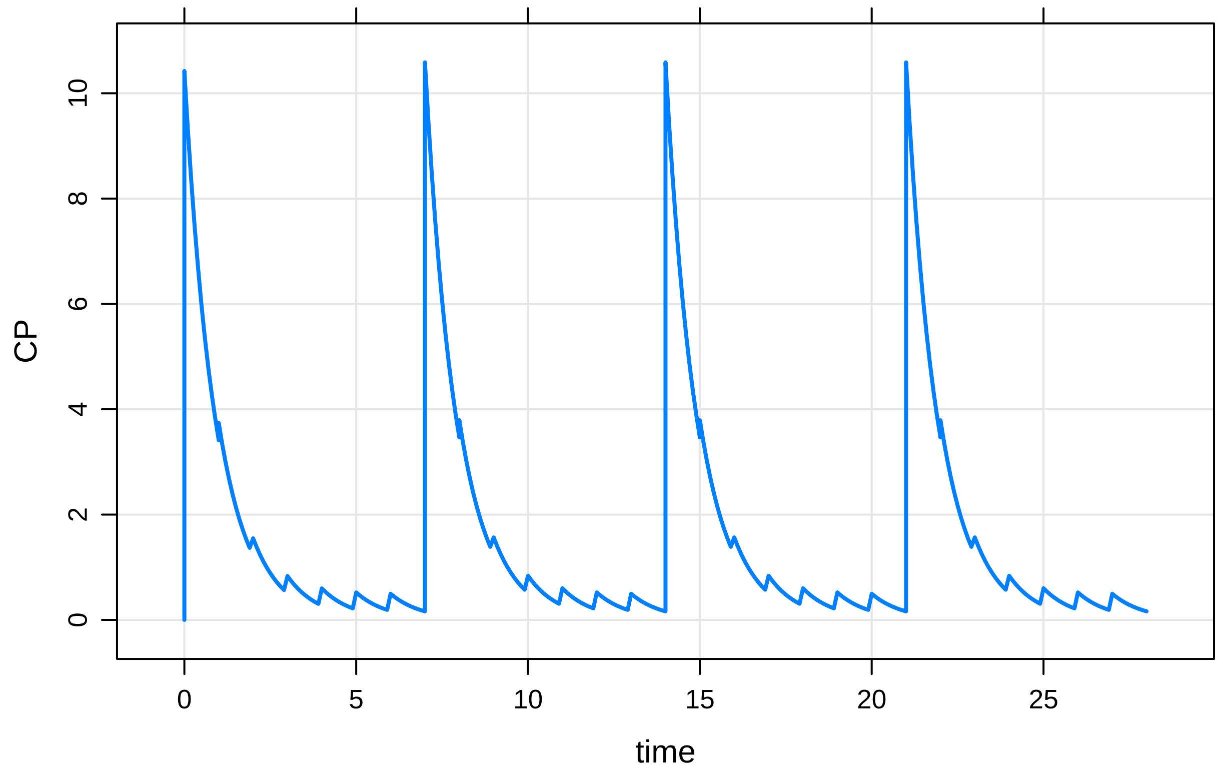
Load + maintenance doses

```
#High dose: 1600mg once every week
load<- ev(amt = 1600, ii = 7, addl = 0, time=0)
#Low dose: 50mg once every day, after the high dose
maintenance<- ev(amt = 50, ii = 1, addl = 5,time=1)
easy.mrgsim(model=cmt1_iv,dosing_schedule=load+maintenance,delta=0.1,tend=7) %>% plot
```



Repeat this for one month period

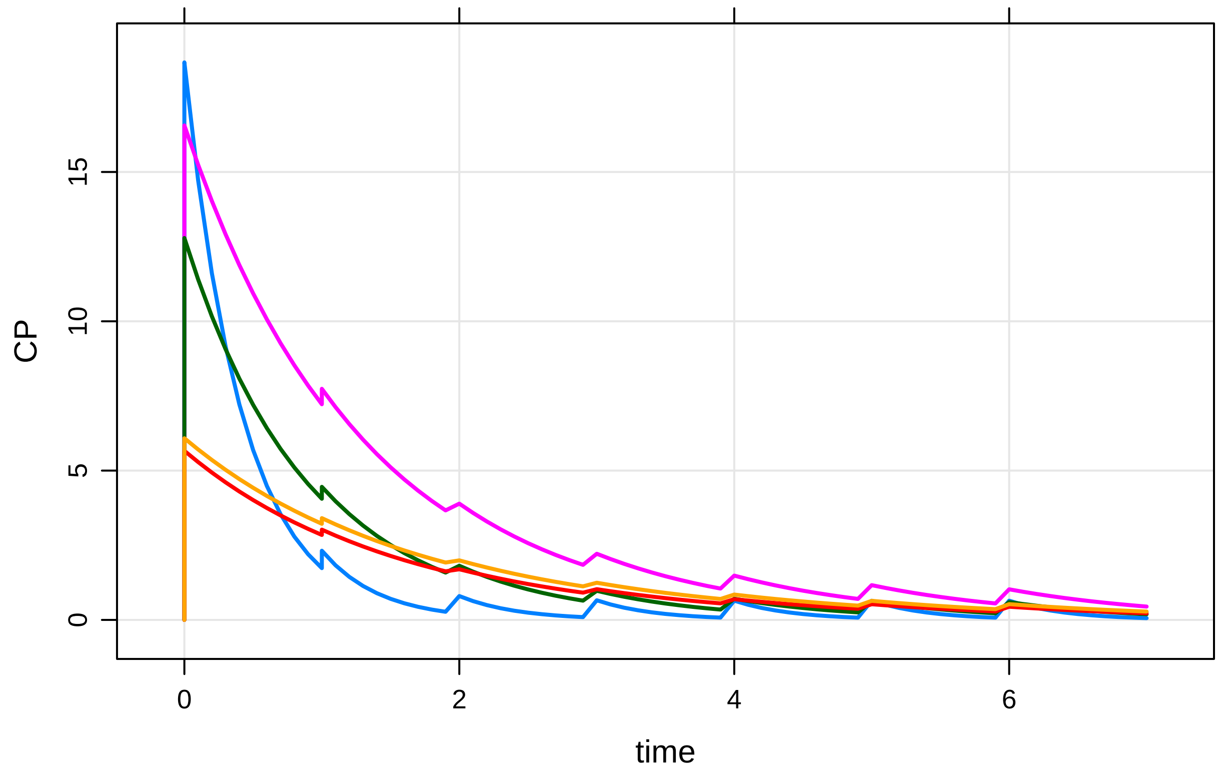
```
load_maint <- ev_rep(load+maintenance, n=4)
head(load_maint)
#>   time cmt  amt evid ii addl ID
#> 1    0   1 1600    1  7    0  1
#> 2    1   1   50    1  1    5  1
#> 3    7   1 1600    1  7    0  1
#> 4    8   1   50    1  1    5  1
#> 5   14   1 1600    1  7    0  1
#> 6   15   1   50    1  1    5  1
easy.mrgsim(model=cmt1_iv,data=load_maint,delta=0.1,tend=7*4) %>% plot
```



Simulate several individuals

To simulate different individuals we need to introduce variability into the pk parameters. To this end, a variance-covariance matrix needs to be defined. Let's simulate 5 individuals, with 0.1 variance for the clearance (CL) and 0.2 variance for the volume of distribution (V).

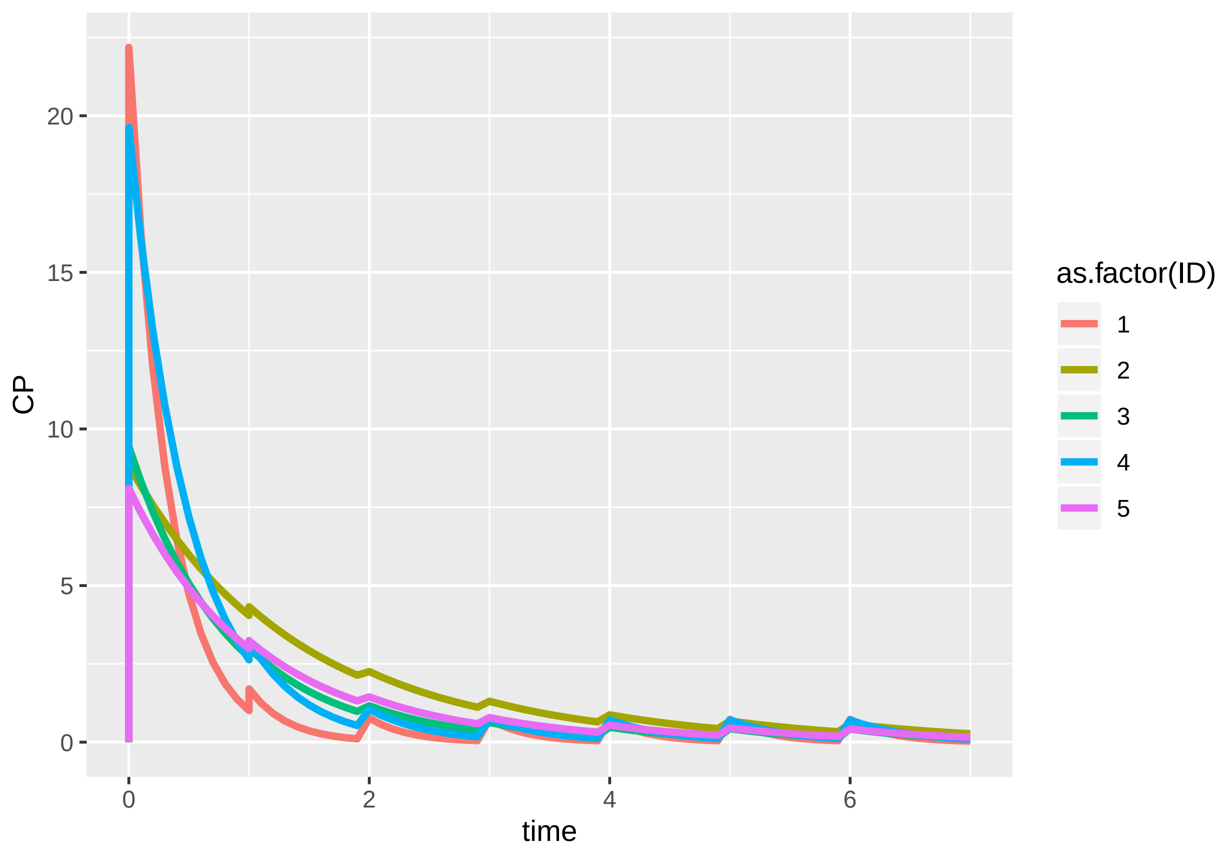
```
Nindividuals=5
easy.mrgsim(model=cmt1_iv,dosing_schedule=load+maintenance,delta=0.1,tend=7,
            variability = dmat(0.1, 0.2),nid=Nindividuals) %>% plot
```



Plot with ggplot2

You prefer plotting with ggplot2? Just save the results as a data.frame

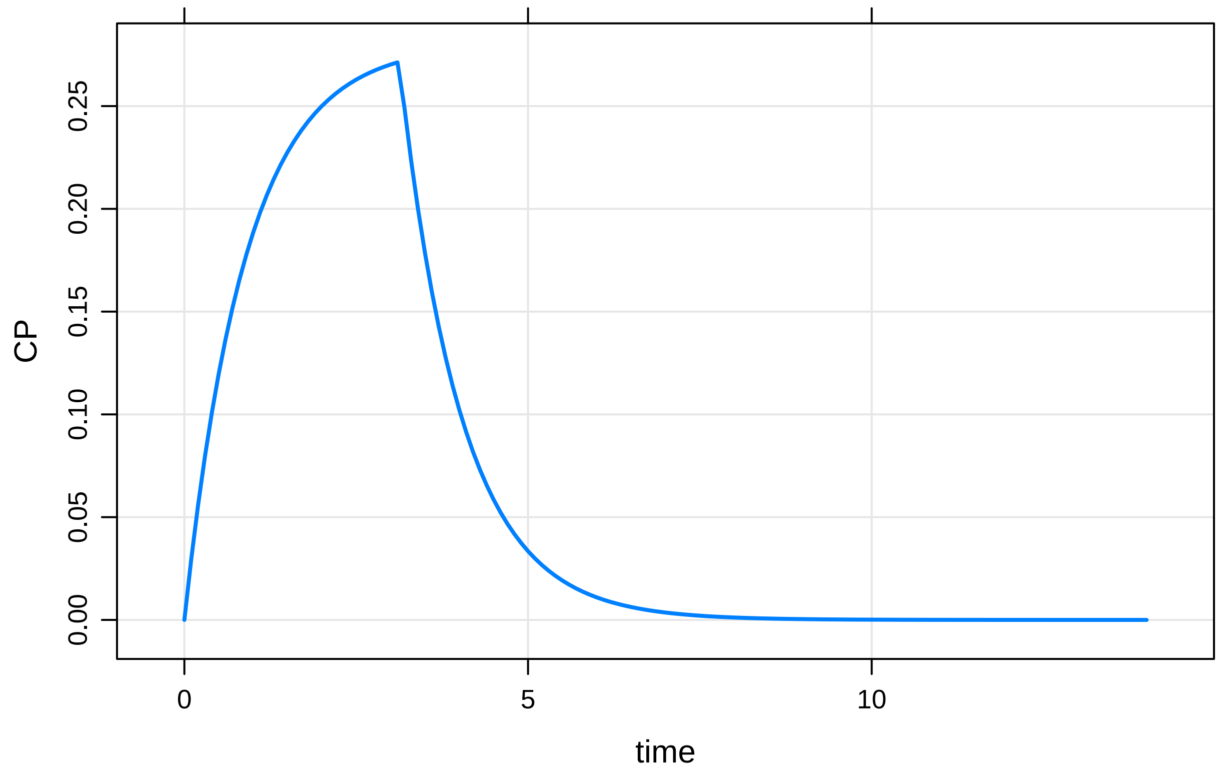
```
pk<-as.data.frame(easy.mrgsim(model=cmt1_iv,dosing_schedule=load+maintenance,delta=0.1,tend=7,
                              variability = dmat(0.1, 0.2),nid=Nindividuals) )
head(pk)
#>   ID time        CP
#> 1  1  0.0  0.000000
#> 2  1  0.0 22.184275
#> 3  1  0.1 16.285083
#> 4  1  0.2 11.954590
#> 5  1  0.3  8.775653
#> 6  1  0.4  6.442051

library(ggplot2)
#> Warning: package 'ggplot2' was built under R version 3.5.3
ggplot(data=pk,aes(x=time,y=CP,col=as.factor(ID)))+geom_line(size=1.3)
```



IV infusion

For drug administration by infusion, or any zero-order input models, the rate of drug administration may be defined using the rate argument.

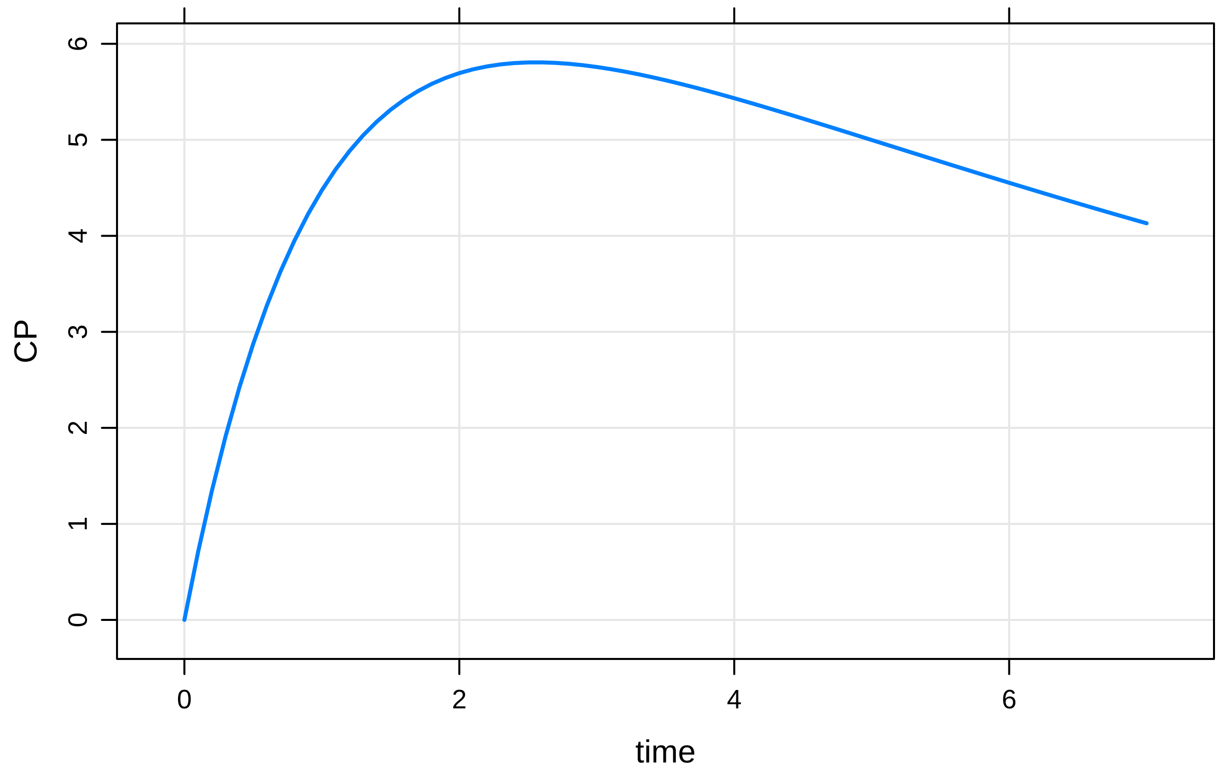
```
inf <- ev(amt = 150, time=0, rate=48)
easy.mrgsim(model=cmt1_iv,dosing_schedule =inf,delta=0.1,tend=7*2) %>% plot
```



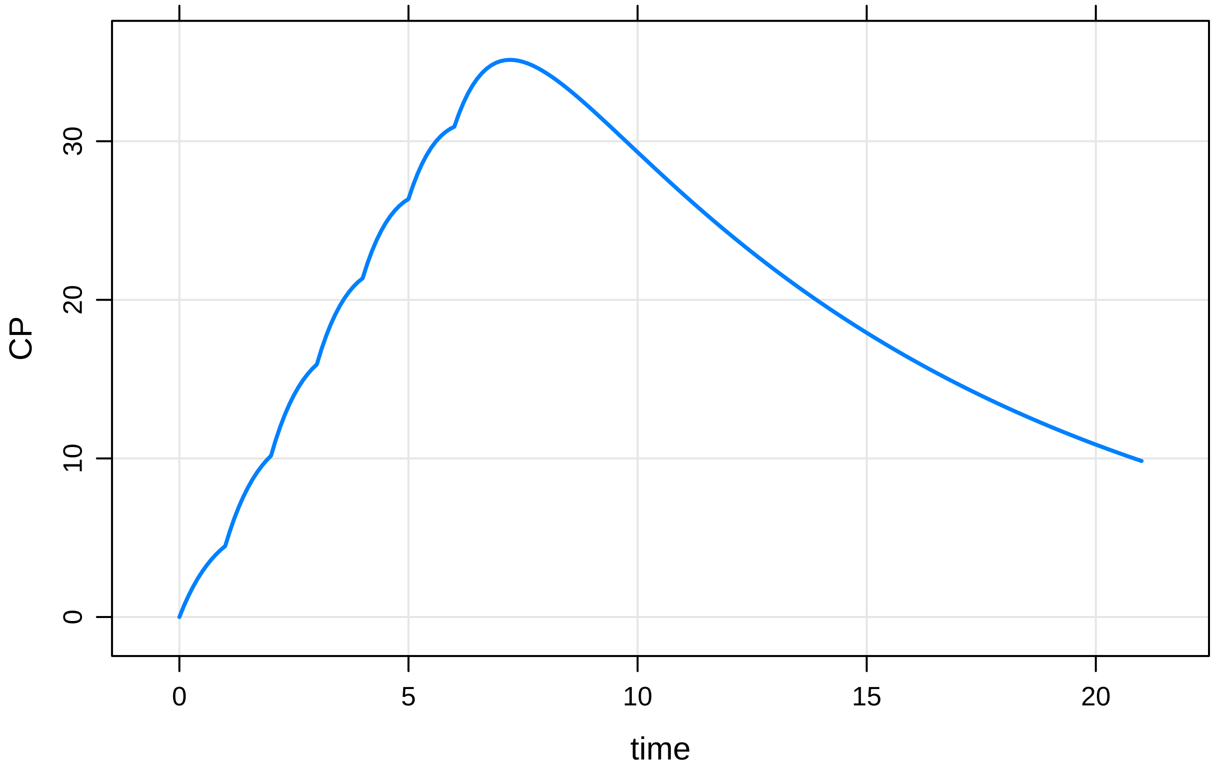
Oral administration

Now we are going to simulate a one compartment model with extravascular administration. We first select a model from the model libraries already included in ACESO instead of writing all the code by ourselves:

```r
model_library(list=T)
#> ACESO internal library of PK models:
#> [1] "1cmt_2depot" "1cmt_ev"     "1cmt_iv"     "2cmt_2depot" "2cmt_ev"
#> [6] "2cmt_iv"     "3cmt_ev"     "3cmt_iv"

cmt1_oral <- mread("1cmt_ev", model_library()) %>% Req(CP)
#> Building 1cmt_ev ...
#> done.

oral1 <- ev(amt = 150, time=0)

easy.mrgsim(model=cmt1_oral,dosing_schedule=oral1,delta=0.1,tend=7) %>% plot
```

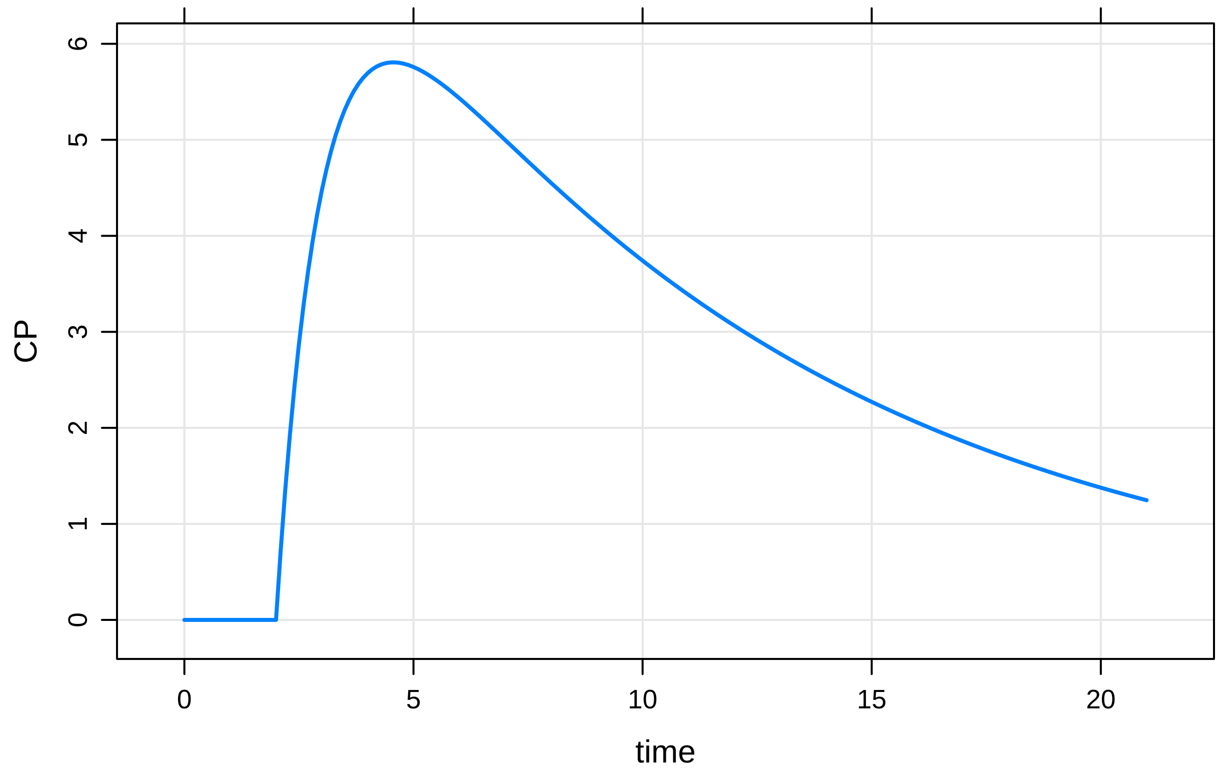


```
oral2 <- ev(amt = 150, ii=1, addl=6, time=0)

easy.mrgsim(model=cmt1_oral,dosing_schedule=oral2,delta=0.1,tend=7*3) %>% plot
```



Introduce a lag time of 2 days

```r
#See the parameters of the model
param(cmt1_oral)
#>
#>  Model parameters (N=5):
#>  name value . name value
#>  ALAG 0     | TVKA 1
#>  F    1     | TVV  20
#>  TVCL 2     | .    .
#The parameter associated with the lag time is ALAG
easy.mrgsim(model=cmt1_oral,dosing_schedule=oral1,delta=0.1,tend=7*3,parameters=list(ALAG=2)) %>% plot
```



CP
time
0
1
2
3
4
5
6
0
5
10
15
20

Demo 4: Synergy assesment

In order to quantify the degree of synergy/antagonism between two compounds, the typical approach is to compare their measured combination effect to a null reference model of no interaction, i.e. the expected response assuming no interaction between the two compounds. If the combination response is greater than what is expected by the reference model, the combination is classified as synergistic, while antagonism is defined when the combination produces less than the expected effect.

There are several well-known conventional approaches that define different null models to assess drug synergy/antagonism. In this example, we will compare the drug-concentration dependent birth rates that can be obtained from the analysis of cell vialibility and apoptosis assay data, to the birth rates obtained under two common no-interaction models: Loewe additivity and Highest Single Agent.

Load libraries:

```
library(ACESO)
#> Warning: replacing previous import 'drc::gaussian' by 'stats::gaussian'
#> when loading 'ACESO'
#> Warning: replacing previous import 'drc::getInitial' by 'stats::getInitial'
#> when loading 'ACESO'
```

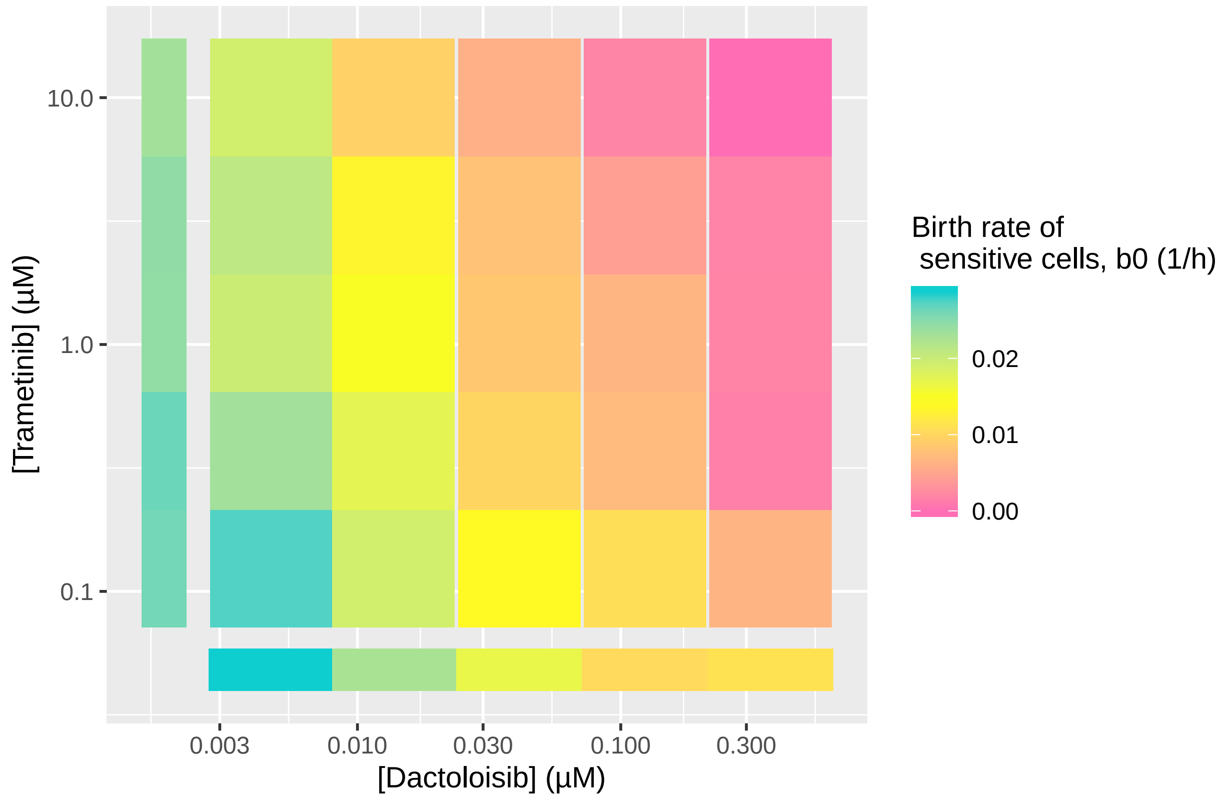
Data

In the current evaluation, cell viability data resulting from the exposure of BT-20 triple-negative breast cancer cell line to the combination of different concentrations of two small molecule kinase inhibitors (dactolisib and trametinib) is analyzed. The data was obtained from the HMS LINCS database (https://lincs.hms.harvard.edu/). The objective of this vignette is to analyze the possible synergism that arises from the combination of the drugs. The birth and death rate have been already calculated using this data and functions from ACESO package.

```
data(Dactolisib_Trametinib_rates)
head(GD)
#>    Cell.line CONC   CONC2  Net_growth Type Birth_rate Death_rate
#> 1      BT-20    0  0.0000 0.007727910    0 0.02872791      0.021
#> 4      BT-20    0  0.1235 0.005032087    0 0.02603209      0.021
#> 7      BT-20    0  0.3704 0.005433627    0 0.02643363      0.021
#> 10     BT-20    0  1.1111 0.003351660    0 0.02435166      0.021
#> 13     BT-20    0  3.3333 0.003391692    0 0.02439169      0.021
#> 16     BT-20    0 10.0000 0.002139690    0 0.02313969      0.021
#>     Drug.Name Drug2.Name
#> 1  Dactolisib Trametinib
#> 4  Dactolisib Trametinib
#> 7  Dactolisib Trametinib
#> 10 Dactolisib Trametinib
#> 13 Dactolisib Trametinib
#> 16 Dactolisib Trametinib

DrugA=as.character(GD$Drug.Name[1])
DrugB=as.character(GD$Drug2.Name[1])
print(c(DrugA,DrugB))
#> [1] "Dactolisib" "Trametinib"
```

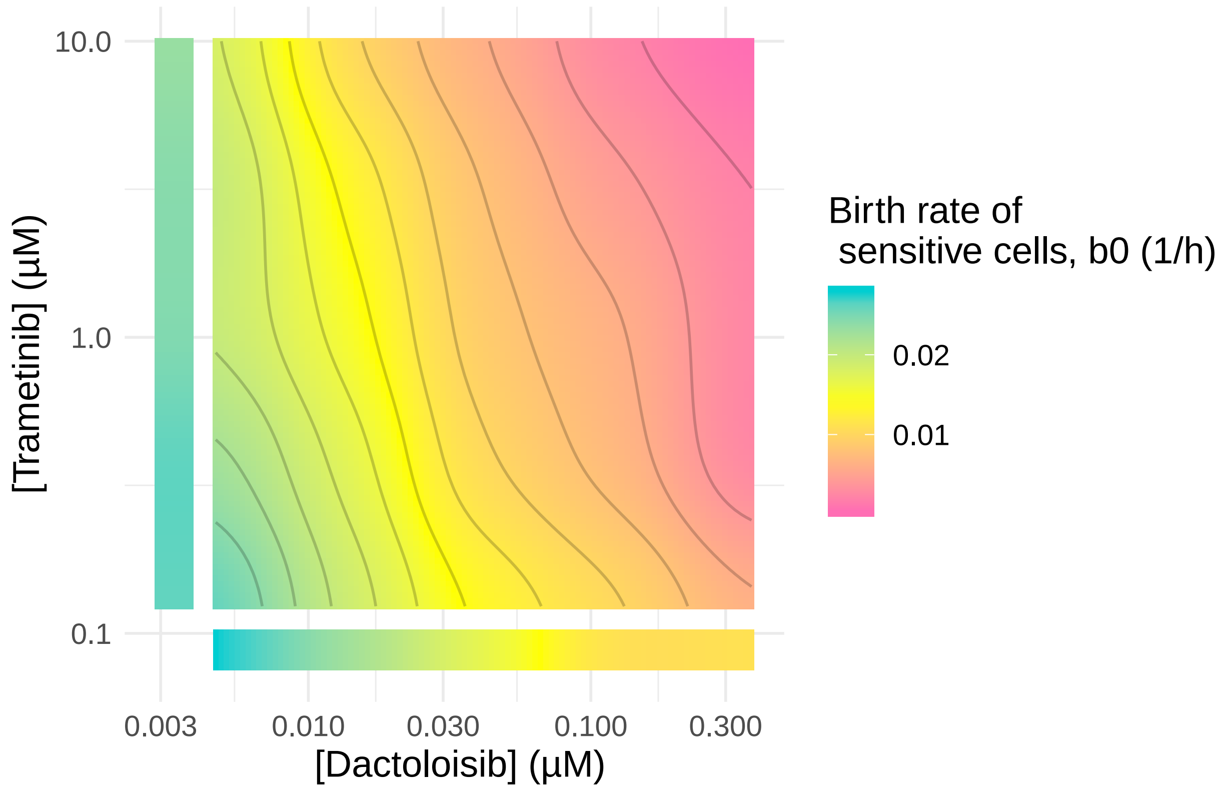
Use **responseMap** function to plot the surface of the birth rate values versus both drug concentrations:

```
rmap <- responseMap(Birth_rate~CONC+CONC2,GD,logscale=T,interpolate=FALSE)
DifferenceSurface.plot(rmap,zcenter=max(GD$Birth_rate)/2,
                       xl=" [Dactoloisib] (µM)",
                       yl="[Trametinib] (µM)",
                       zl="Birth rate of \n sensitive cells, b0 (1/h)",
                       mid="yellow",low="hotpink1",high="darkturquoise")
```



Use **responseMap** again and **plot.ResponseSurface** to make a contour plot of the data. For this plot a interpolation of the data is needed (see logscale=T in the **responseMap** function):

```
rmap <- responseMap(Birth_rate~CONC+CONC2,GD,logscale=T)

ResponseSurface.plot(rmap,xl=" [Dactoloisib] (µM)",
                     yl="[Trametinib] (µM)",
                     zl="Birth rate of \n sensitive cells, b0 (1/h)",
                     palette=c("hotpink1","yellow","darkturquoise"))
```



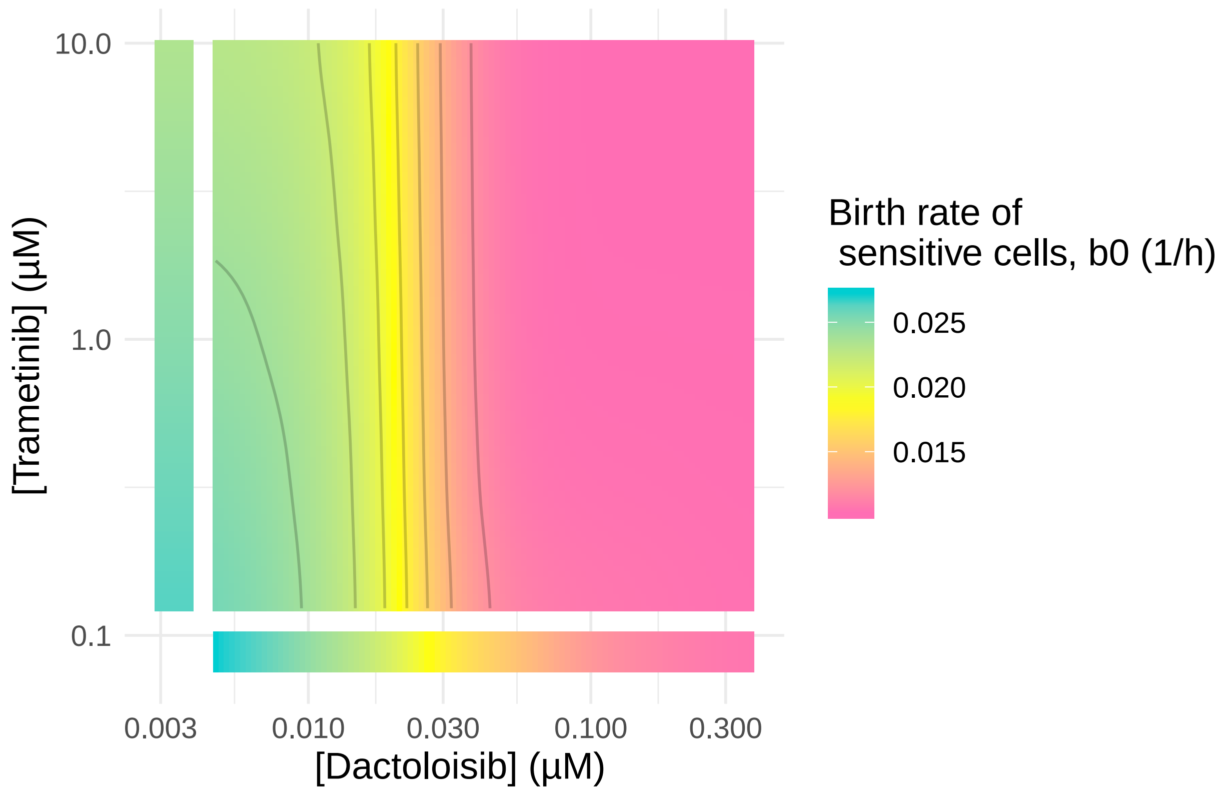
Loewe additivity

Loewe additivity model defines synergy/antagonism as a combined inhibitory effect that is greater/lower than the sum of the individual effects of the drugs. The null reference model in this case is the sum of the individual effect of each drug, which is calculated from the sigmoidal fits of the single-agent response curves. To automatically perform this task the **Loewe** function can be used:

```r
GD=Loewe(data=GD,resp = 'Birth_rate')
```

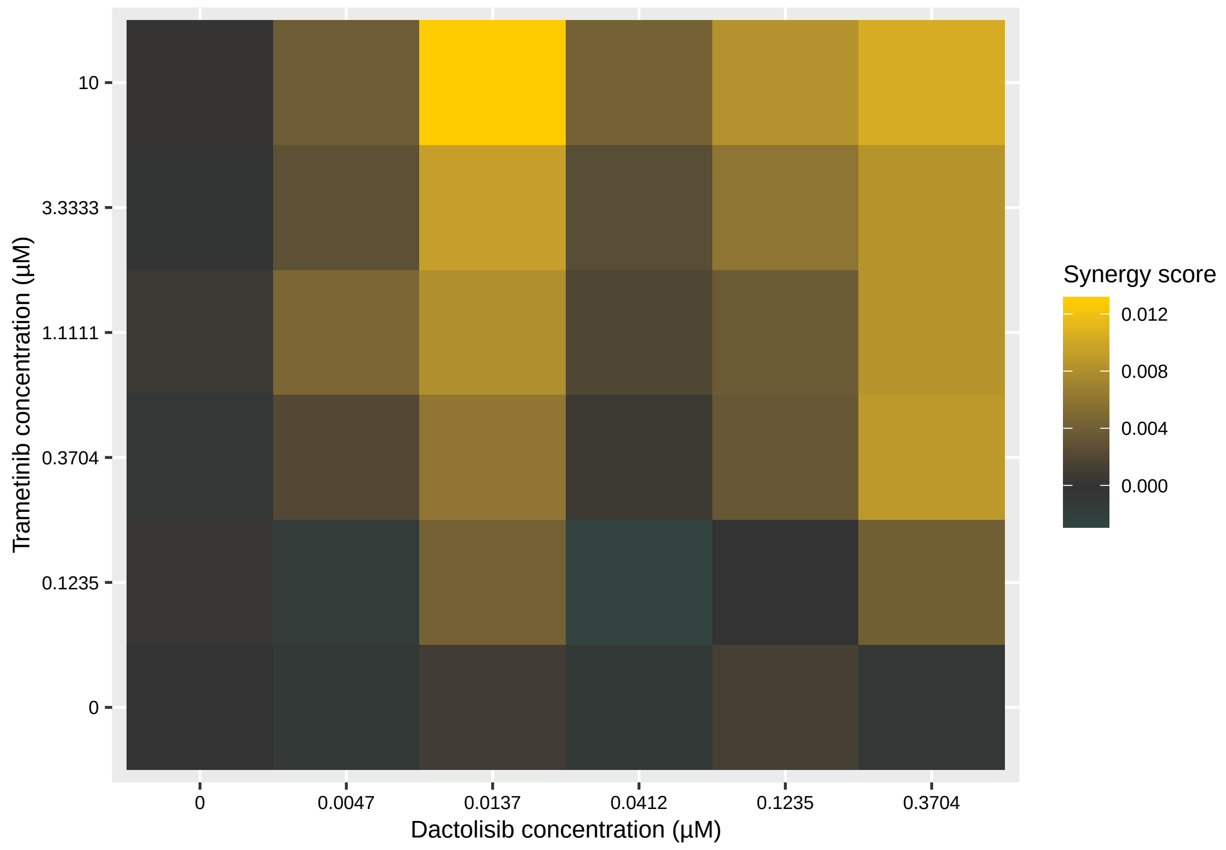
A new column called 'loewe_addivity' is created. Now we plot the response surface of the data using the newly created column to compare it with the previous response surface for the birth rates of sensitive cell population.

```r
rmap_loewe <- responseMap(loewe_additivity~CONC+CONC2,GD,logscale=T)

ResponseSurface.plot(rmap_loewe,xl=" [Dactoloisib] (µM)",
                     yl="[Trametinib] (µM)",
                     zl="Birth rate of \n sensitive cells, b0 (1/h)",
                     palette=c("hotpink1","yellow","darkturquoise"))
```



For a proper comparison of the surfaces, we calculate the difference between them and plot the results in a colored pairwise matrix. The score showed in this matrices reflects the difference between the measurement and the surface obtained under the no interaction models in a way that values less than zero (blue) represent antagonism and values greater that zero (yellow) represent synergism.

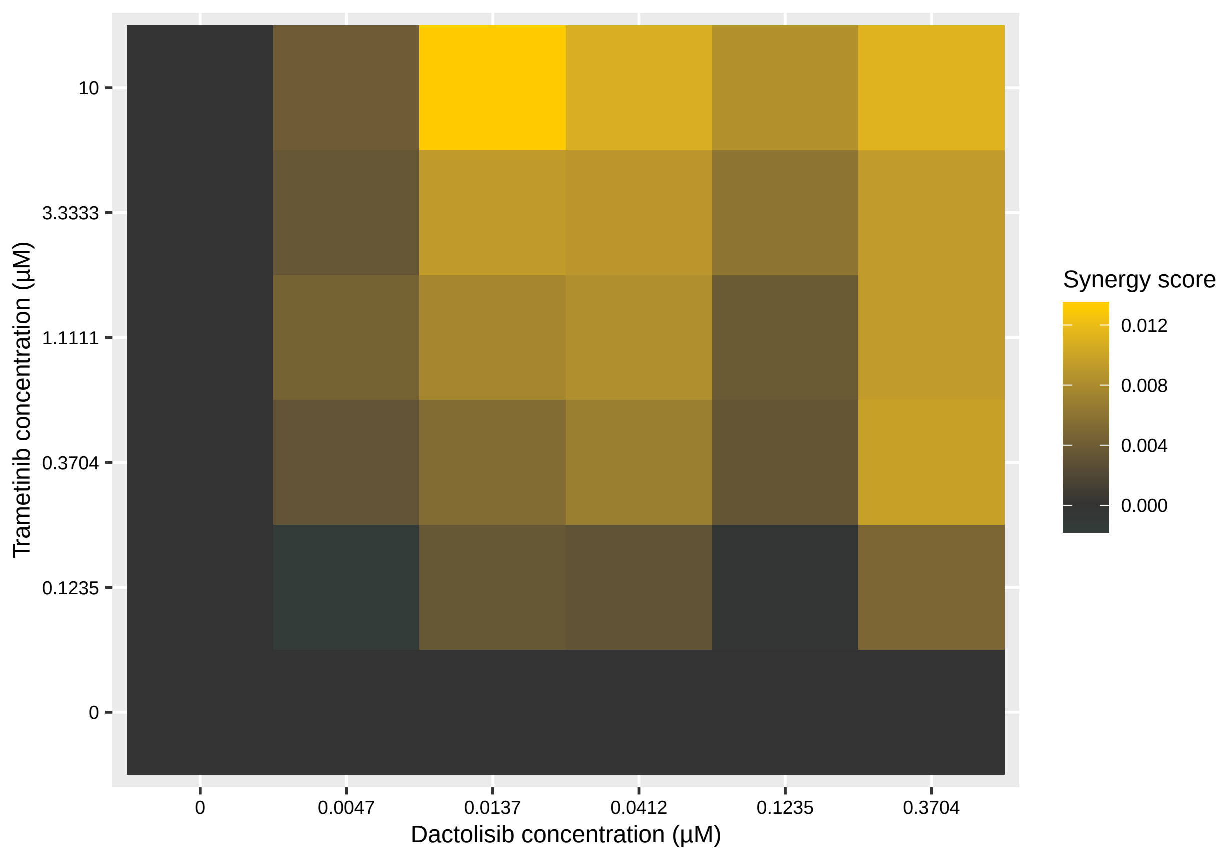
```
GD$diffLoewe=(GD$loewe_additivity-GD$Birth_rate)

p=SynergyMatrix.plot(GD,resp="diffLoewe")
p+ggplot2::labs(x="Dactolisib concentration (µM)", y="Trametinib concentration (µM)")
```



Highest Single Agent (HSA)

Now, we are going to repeat the process using the Highest Single Agent (HSA) model. HSA, also known as Gaddum's non-interaction model, is another popular model which defines a independent action of the drugs when the predicted effect of a combination is that of the one most effective drug alone. To calculate the null surface of the HSA model, use **HSA** function:

```
GD=HSA(GD,resp = 'Birth_rate')

GD$diffHSA=(GD$HSA_response-GD$Birth_rate)

p2=SynergyMatrix.plot(GD,resp="diffHSA")
p2+ggplot2::labs(x="Dactolisib concentration (µM)", y="Trametinib concentration (µM)")
```



Synergy score
0.012
0.008
0.004
0.000
Trametinib concentration (µM)
10
3.3333
1.1111
0.3704
0.1235
0
Dactolisib concentration (µM)
0
0.0047
0.0137
0.0412
0.1235
0.3704