AZRsim Documentation

AstraZeneca

Contents

User Guide

The user guide documents the functionality for the AZRsim R package which can be used to simulate ordinary differential equations in C and NONMEM.

In order to simulate the model in the **AZRsim** package we have to,

1. Create an appropriate text file representation of the model.
2. Compile the model with the `create_model()` function.
3. Simulate the compiled model with the `simulate()` function.

Chapter 1

Installation

The following does not apply to those who do not have access to the package while it is being maintained on GitHub as a private repository.

The **AZRsim** package can be installed from the AZRQCP repository as follows:

```
devtools::install_github("AZRQCP/AZRsim")
```

Chapter 2

Model Specification

In order to simulate the time varying variables of a system of ordinary differential equations you need to provide a specific representation of your mathematical model in a text file (a file with extension `.txt`). Every text file representation of a model must contain the following blocks presented exactly as defined below. Notice that each block name is preceded with ten *s. It is possible to comment out a line of code using `%`.

```
********** MODEL NAME

********** MODEL NOTES

********** MODEL STATES

********** MODEL PARAMETERS

********** MODEL VARIABLES

********** MODEL REACTIONS

********** MODEL FUNCTIONS

********** MODEL EVENTS
```

The remainder of this section deals with what can be declared within each block.

2.1 Model Name and Model Notes

`MODEL NAME` and `MODEL NOTES` are descriptive blocks that allow the user to specify a model name and any notes about the model. The name of the model will be presented when an `azrmod` object is printed to the console.

Below are the model name and model notes blocks for the simple harmonic oscillator example in the **AZRsim** package (`sho.txt`).

```
********** MODEL NAME

Harmonic Oscillator

********** MODEL NOTES
```

```
Harmonic Oscillator ODE example.
```

This meta data can be extracted from an `azrmod` object.

2.2 Model States

The `MODEL STATES` block is where the user describes the system of differential equations followed by the initial conditions for each variable. The left hand side of each differential equation *must* be declared as `d/dt(x)` where `x` is a variable name chosen by the user.

The initial conditions *must* be defined, and they *must* be defined after the system of differential equations is defined. If variable `x` in the preceding paragraph had an initial condition value of 3 then we can declare `x(0) = 3` after defining the system of differential equations.

2.3 Model Parameters

The `MODEL PARAMETERS` block is where the user defines the values of the parameters in the system of differential equations defined in `MODEL STATES`.

2.4 Model Variables

The `MODEL VARIABLES` block provides a way to reparametrize parts of the model. Variables for intermediate calculations should be defined under `MODEL VARIABLES`.

```
********** MODEL STATES

d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0

********** MODEL PARAMETERS

k1 = 0.5

********** MODEL VARIABLES

R = k1*A
```

2.5 Model Reactions

The `MODEL REACTIONS` block provides a way to reparametrize parts of the model. Reaction rates should be defined in the `MODEL REACTIONS` block.

```
********** MODEL STATES

d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
```

```
********** MODEL PARAMETERS

k1 = 0.5

********** MODEL REACTIONS

R = k1*A
```

2.6 Model Functions

The `MODEL FUNCTIONS` block can be used to define reoccuring calculations.

As an example, the `sho_func.txt` example model in the **AZRsim** package applies the function $f(x) = \frac{x}{100}$ to the variable y_2 in $\frac{d}{dt}y_2$ equation in the system.

```
********** MODEL NAME

Simple Harmonic Oscillator

********** MODEL NOTES

Simple Harmonic Oscillator ODE example.

********** MODEL STATES

% ODE system
d/dt(y1) = y2
d/dt(y2) = - y1 - theta * f(y2)

% initial conditions
y1(0) = 1
y2(0) = 0

********** MODEL PARAMETERS

theta = 0.15

********** MODEL VARIABLES
********** MODEL REACTIONS
********** MODEL FUNCTIONS

f(x) = x * 0.01

********** MODEL EVENTS
```

2.7 Model Events

The `MODEL EVENTS` block can be used to define discrete state events. Note that the declaration of events is not vectorized.

There are essentially two parts to defining a discrete event in the model:

1. When does the event take place?
2. What is the event?

Below is an example of a event block.

```
********* MODEL EVENTS

event1 = eq(time, 1), y, y+40
event2 = eq(time, 2), y, y+40
```

The term `eq(time, 1)` pertains to (1). This says that the event occurs when the time step is strictly equal to 1. The term `y, y+40` pertains to (2). This is the event that is to take place. In this case the event involves variable `y` and we want to add 40 units of the drug this variable, `y+40`. (Note that events cannot take place at time 0.) In summary, the above block defines adding 40 to the value of `y` at time steps 1 and 2.

We can issue multiple events. The event defined below specifies resetting the value of `x` to 1 and `y` to 0 when `x` is equal to 0.1.

```
********* MODEL EVENTS

event = eq(x, 0.1), x, 1, y, 0
```

Below is a summary of the options that can be used to declare when an event takes place.

- `eq(a, 0)` implies variable a is strictly equal to 0.
- `lq(a, 0)` implies variable a is less than or equal to 0.
- `gq(a, 0)` implies variable a is greater than or equal to 0.
- `lt(a, 0)` implies variable a is strictly less than 0.
- `gt(a, 0)` implies variable a is strictly greater than 0.

Chapter 3

Modeling Functions

This section specifies the user-facing functions required to simulate the system of ordinary differential equations.

3.1 create_model

The `create_model` function is used to create the C code based on the text file representation of the model. This function returns and object of class `azrmod`. Once you create a model you can print a summary of the model in R. Below is an example of printing the simple harmonic oscillator, which is part of the **AZRsim** package.

```
library(AZRsim)
```

```
sho <- create_model(system.file("examples/sho.txt", package="AZRsim"))
sho
```

```
##  AZRmodel
##  ========
##  Name:                    Harmonic Oscillator
##  Number States:           2
##  Number Parameters:       1
##  Number Variables:        0
##  Number Reactions:        0
##  Number Functions:        0
```

In this model, the system of differential equations involves two states (y_1 and y_2) and one parameter (θ).

The `azrmod` object is a list and we can extract the meta data that was declared in the `Model Name` and `Model Notes` block by using `$` to access the elements within the list. Continuing the above example, below is the code that allows the user print the meta data associated with the `sho.txt` example model to the R console.

```
sho$name
```

```
## [1] "Harmonic Oscillator"
```

```
sho$notes
```

```
## [1] "Harmonic Oscillator ODE example."
```

3.2 simulate

Once you have created an `azrmod` object with `create_model` you can use the `simulate` generic to simulate the state variables over a specified number of time steps. Note that the `simulate` function requires an object of class `azrmod`, and will return an object of class `azrsim` and `data.frame`.

In the simple harmonic oscillator example above we have two states, so calling `simulate(sho, seq(1, 100, by=0.1))` will provide a simulation of these two states over the time sequence 1 to 100 at 0.1 intervals.

```
sho_sim <- simulate(sho, seq(1, 100, by=0.1))
head(sho_sim)
```

```
##   TIME        y1          y2
## 1  1.0 1.0000000  0.00000000
## 2  1.1 0.9950290 -0.09908659
## 3  1.2 0.9802654 -0.19571796
## 4  1.3 0.9560004 -0.28896987
## 5  1.4 0.9226146 -0.37796711
## 6  1.5 0.8805758 -0.46189098
```

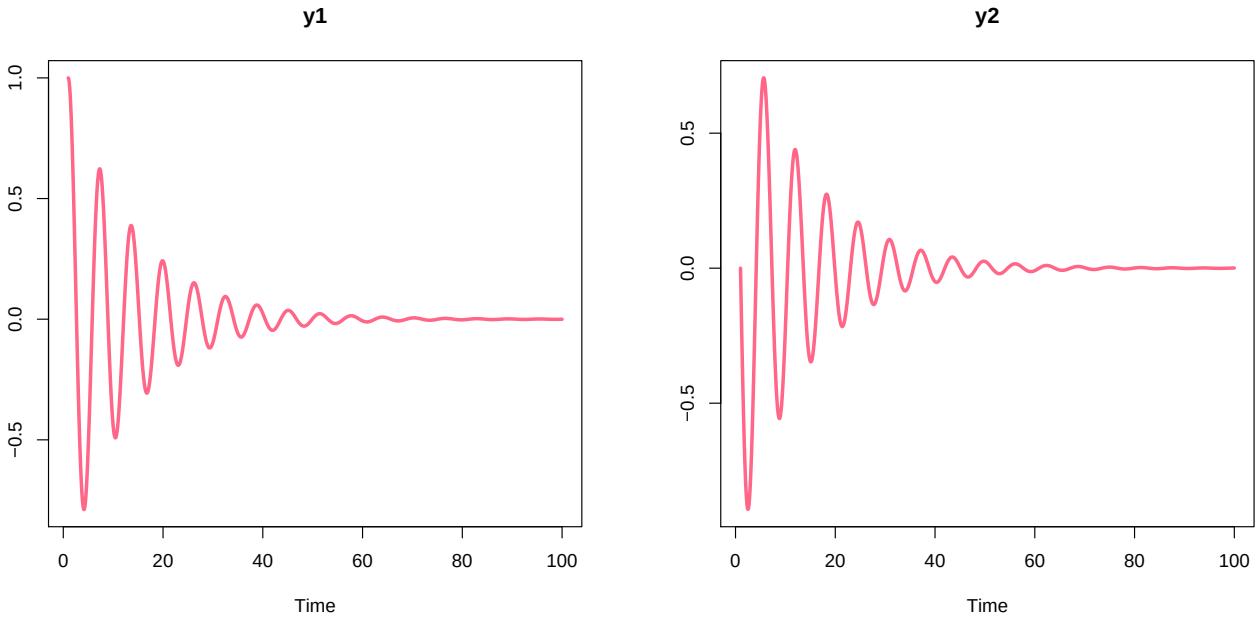
```
dim(sho_sim)
```

```
## [1] 991   3
```

3.3 plot

The `plot` generic can be called on an `azrsim` object and, by default, will construct a lattice plot of the first 9 parameters over time.

```
plot(sho_sim, col = "#FF6688", lwd = 3)
```



3.4 shiny_plot

The `shiny_plot` function must be called on an `azrsim` object and will launch a shiny app that will allow the user to interactively include/exclude any state variable estimated.

Chapter 4

Dosing

Dosing can be handled either in the `Model Events` block or by providing a data frame that describes a dosing table that can be called into the `simulate` function as the argument `dosingTable`. Both approaches are discussed below using, as an example, the following model of a simple one compartment model.

$$\frac{d}{dt}y = -\theta \cdot y$$

4.1 Dosing in the Model Events Block

The model below defines dosing within the `Model Events` block for a simple one compartment model.

```
********** MODEL NAME

Simple One Compartment Dosing Model

********** MODEL NOTES

A simple one compartment dosing model where
ten equally spaced doses are given to the subject.

********** MODEL STATES

d/dt(y) = -theta * y

y(0) = 0

********** MODEL PARAMETERS

theta = 0.6

********** MODEL VARIABLES
********** MODEL REACTIONS
********** MODEL FUNCTIONS
********** MODEL EVENTS

% adding 40 units at a specified time step
event1 = eq(time, 1), y, y+40
```

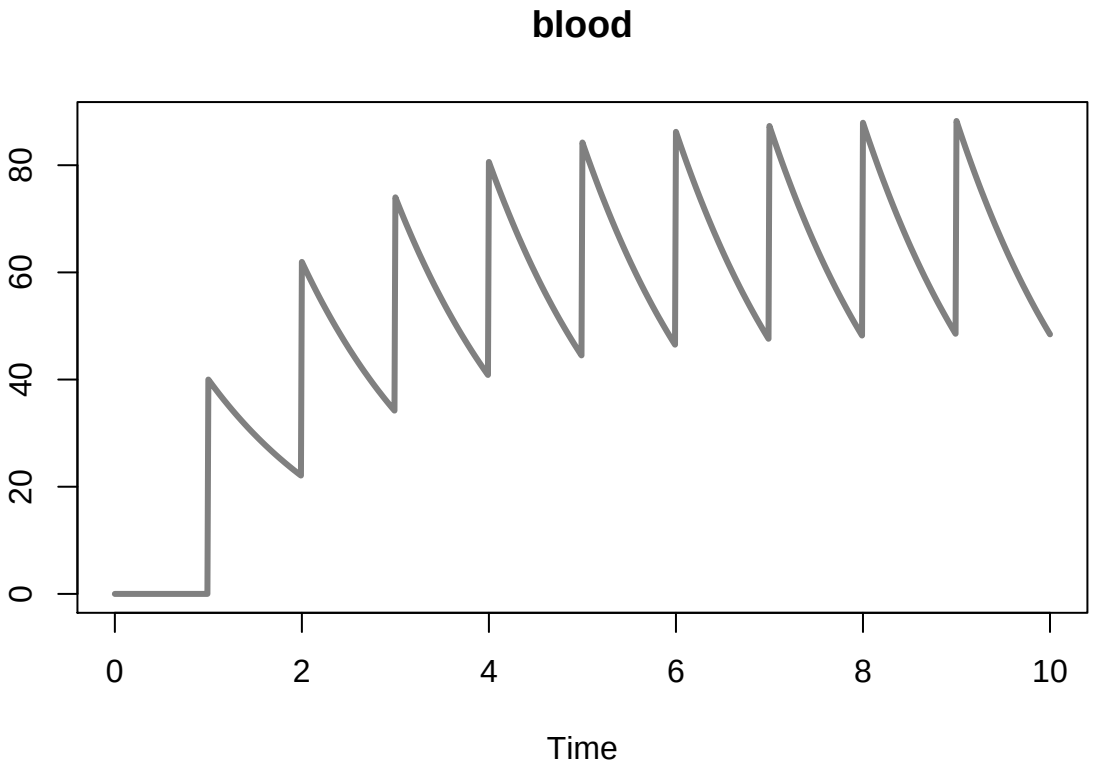
```
event2 = eq(time, 2), y, y+40
event3 = eq(time, 3), y, y+40
event4 = eq(time, 4), y, y+40
event5 = eq(time, 5), y, y+40
event6 = eq(time, 6), y, y+40
event7 = eq(time, 7), y, y+40
event8 = eq(time, 8), y, y+40
event9 = eq(time, 9), y, y+40
```

Below we compile the model, simulate, and plot the variable.

```
one_cpt <- create_model(system.file("examples/one_cpt.txt", package="AZRsim"))
```

```
## Warning in check_azrmod(model): check_azrmod: AZRmodel contains state names
## with a single character name. Try to avoid that if you plan to use NONMEM
## or MONOLIX
```

```
one_cpt_simulation <- simulate(one_cpt, simtime = 10)
plot(one_cpt_simulation, col = "#808080", lwd = 3, plot_names = "blood")
```



4.2 Dosing with a Dosing Table

The model below defines an analogous dosing regimen using the `INPUT*` variable in the system of differential equations for the same model. Note that `INPUT*` must be applied to the relevant variables in the `Model States` block. In this context a dose (the units of which will be defined in the dosing table) is added to the variable `y` at specific time intervals (also defined in the dosing table).

```
********** MODEL NAME

Simple One Compartment Dosing Model
```

```
********** MODEL NOTES

A simple one compartment dosing model where
the doses must be defined as a data frame in
the 'dosingTable' argument of the simulate
function.

********** MODEL STATES

d/dt(y) = -theta * y + INPUT1

y(0) = 0

********** MODEL PARAMETERS

theta = 0.6

********** MODEL VARIABLES
********** MODEL REACTIONS
********** MODEL FUNCTIONS
********** MODEL EVENTS
```

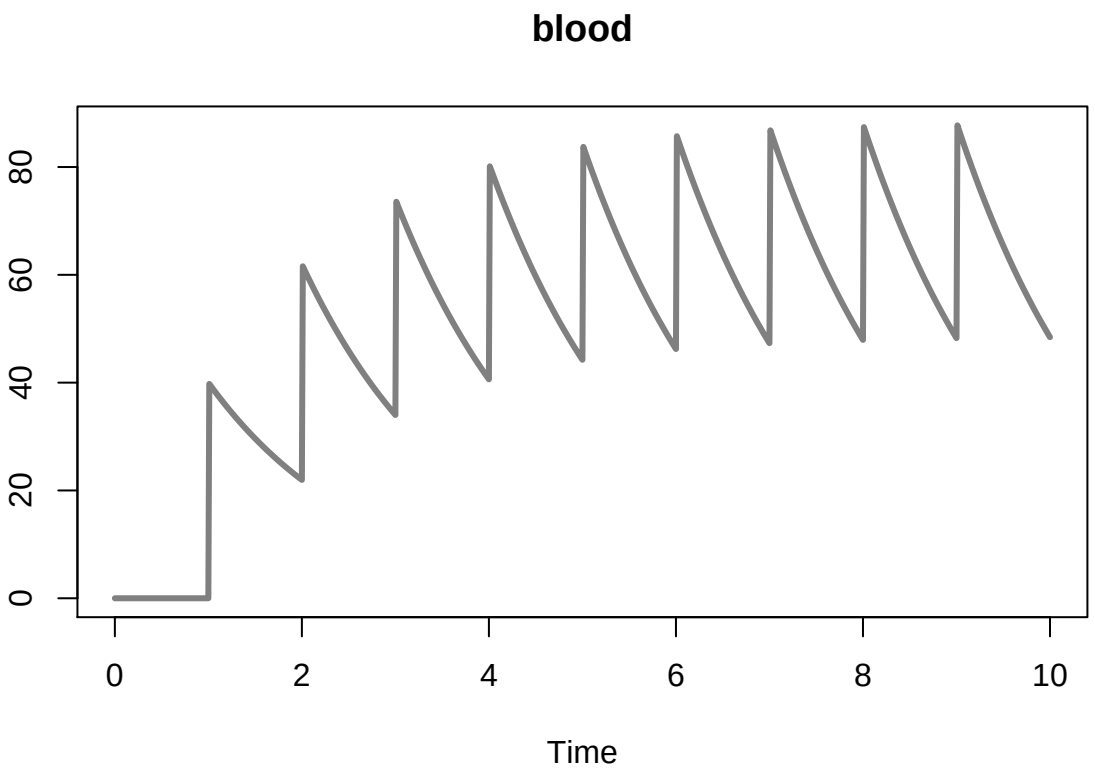
Below we construct a dosing table, compile the model, simulate, and plot the variable. The dosing table must be a data frame with the following columns,

- `TIME` - the time at which the does occurs.
- `DOSE` - the units of the dose.
- `DURATION` - how long the dose is administered.
- `INPUT` - which input variable the dose is associated with.
- `LAGTIME` - the lag time associated with each dose.

```
one_cpt_dt <- create_model(system.file("examples/one_cpt_dt.txt", package="AZRsim"))
```

```
## Warning in check_azrmod(model): check_azrmod: AZRmodel contains state names
## with a single character name. Try to avoid that if you plan to use NONMEM
## or MONOLIX
```

```
dt <- data.frame("TIME" = seq(1,9, by = 1),
                 "DOSE" = 40,
                 "DURATION" = 0,
                 "INPUT" = 1,
                 "LAGTIME" = 0,
                 stringsAsFactors = FALSE)
one_cpt_dt_simulation <- simulate(one_cpt_dt, 10, dosingTable = dt)
plot(one_cpt_dt_simulation, col = "#808080", lwd = 3, pars = "y", plot_names = "blood")
```



4.3 Outputs

It is also possible to defined outputs from using `OUTPUT*` in the model. An example is provided below where the output is half of the state variable.

```
********** MODEL NAME

Simple One Compartment Dosing Model

********** MODEL NOTES

A simple one compartment dosing model where
the doses must be defined as a data frame in
the 'dosingTable' argument of the simulate
function.

********** MODEL STATES

d/dt(y) = -theta * y + INPUT1

y(0) = 0

********** MODEL PARAMETERS

theta = 0.6

********** MODEL VARIABLES
```

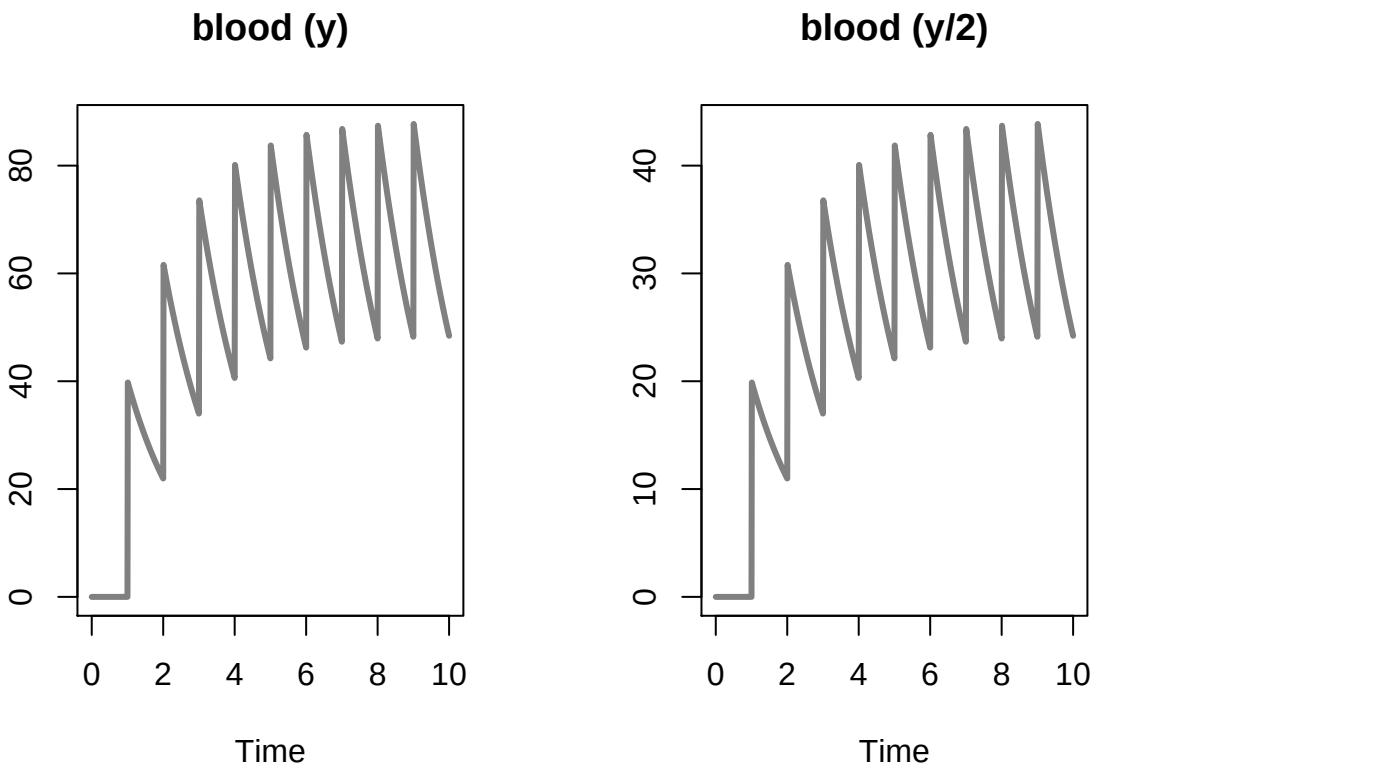
```
y_new = y/2
OUTPUT1 = y_new

********** MODEL REACTIONS
********** MODEL FUNCTIONS
********** MODEL EVENTS
```

Below we compile the model, simulate the state variables, and plot the results.

```
one_cpt_out <- create_model(system.file("examples/one_cpt_output.txt", package="AZRsim"))
```

```
## Warning in check_azrmod(model): check_azrmod: AZRmodel contains state names
## with a single character name. Try to avoid that if you plan to use NONMEM
## or MONOLIX
```

```
dt <- data.frame("TIME" = seq(1,9, by = 1),
                 "DOSE" = 40,
                 "DURATION" = 0,
                 "INPUT" = 1,
                 "LAGTIME" = 0,
                 stringsAsFactors = FALSE)
one_cpt_out_simulation <- simulate(one_cpt_out, 10, dosingTable = dt, output = c("y", "OUTPUT1"))
plot(one_cpt_out_simulation, col = "#808080", lwd = 3, plot_names = c("blood (y)", "blood (y/2)"))
```



Chapter 5

Examples

5.1 Simple Harmonic Oscillator

The equations that govern a harmonic oscillator are the following,

$$\frac{d}{dt}y_1 = y_2$$
$$\frac{d}{dt}y_2 = -y_1 - \theta \cdot y_2$$

where y_1, y_2 are state variables and θ is a parameter in this model. The text file representation of the model is provided below.

```
********** MODEL NAME

Harmonic Oscillator

********** MODEL NOTES

Harmonic Oscillator ODE example.

********** MODEL STATES

d/dt(y1) = y2
d/dt(y2) = - y1 - theta * y2

y1(0) = 1
y2(0) = 0

********** MODEL PARAMETERS

theta = 0.15

********** MODEL VARIABLES
********** MODEL REACTIONS
********** MODEL FUNCTIONS
********** MODEL EVENTS
```

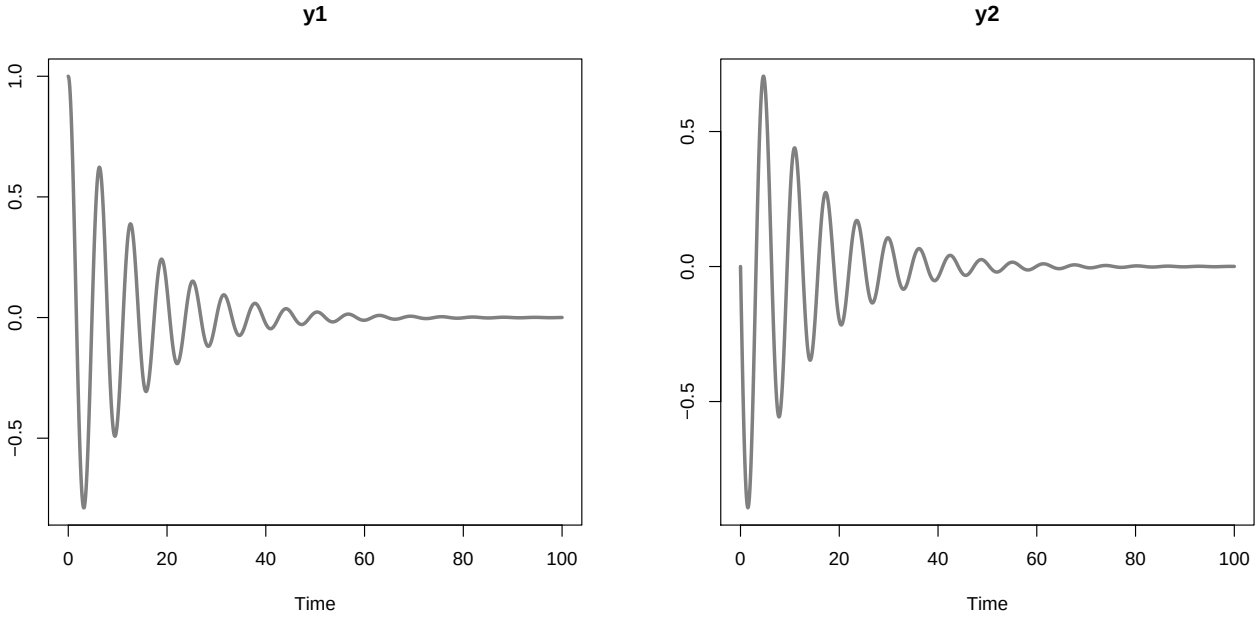
Compiling and simulating the model.

```
sho_model <- create_model(system.file("examples/sho.txt", package="AZRsim"))
sho_sim <- simulate(sho, seq(0, 100, by=0.1))
head(sho_sim)
```

```
##   TIME        y1          y2
## 1  0.0 1.0000000  0.00000000
## 2  0.1 0.9950291 -0.09908663
## 3  0.2 0.9802655 -0.19571802
## 4  0.3 0.9560005 -0.28896993
## 5  0.4 0.9226147 -0.37796717
## 6  0.5 0.8805759 -0.46189106
```

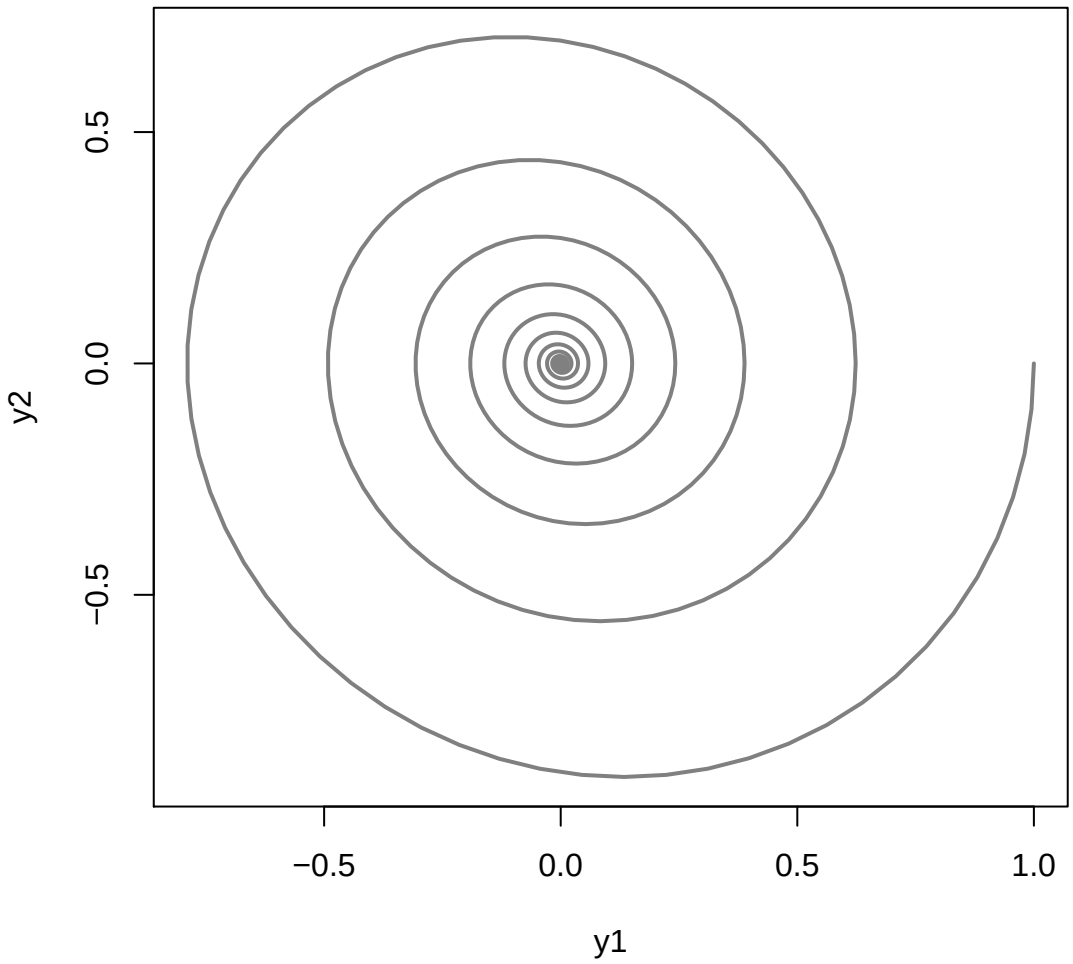
Plotting variables as a function of time.

```
plot(sho_sim, col = "#808080", lwd = 3)
```



Plotting the variables as a function of each other.

```
plot(sho_sim[,"y1"], sho_sim[,"y2"], type = "l", col = "#808080", lwd = 2, xlab = "y1", ylab = "y2")
```



5.2 Two Compartment Model

Here we consider a simple two compartment model where, for example, we model the concentration of an oral drug in the intestines and in the blood. The ODE system takes the following form,

$$\frac{d}{dt}y_1 = -a \cdot y_1 + u_t$$
$$\frac{d}{dt}y_2 = a \cdot y_1 - b \cdot y_2$$

where y_1 denotes the concetration of the drug the intenstine, y_2 denotes the concentration of the drug in the blood, a and b are parameters, and u_t denotes the uptake of the drug at time period t.

The text file representation of the model above provided below.

```
********** MODEL NAME

Simple Two Compartment Dosing Model

********** MODEL NOTES
```

```
A simple two compartment dosing model where
ten equally spaced doses are given to the subject.

********** MODEL STATES

d/dt(y1) = -a * y1
d/dt(y2) = a * y1 - b * y2

y1(0) = 0
y2(0) = 0

********** MODEL PARAMETERS

a = 6
b = 0.6

********** MODEL VARIABLES
********** MODEL REACTIONS
********** MODEL FUNCTIONS
********** MODEL EVENTS

% adding 40 units at a specified time step
event1 = eq(time, 1), y1, y1+2
event2 = eq(time, 2), y1, y1+2
event3 = eq(time, 3), y1, y1+2
event4 = eq(time, 4), y1, y1+2
event5 = eq(time, 5), y1, y1+2
event6 = eq(time, 6), y1, y1+2
event7 = eq(time, 7), y1, y1+2
event8 = eq(time, 8), y1, y1+2
event9 = eq(time, 9), y1, y1+2
```

Below the model is compiled and simulated and the 10 initial and final values are presented.

```
two_cpt <- create_model(system.file("examples/two_cpt.txt", package="AZRsim"))
```

```
## Warning in check_azrmod(model): check_azrmod: AZRmodel contains parameter
## names with a single character name. Try to avoid that if you plan to use
## NONMEM or MONOLIX
```

```
two_cpt_simulation <- simulate(two_cpt, simtime = 10)
head(two_cpt_simulation, n = 10)
```

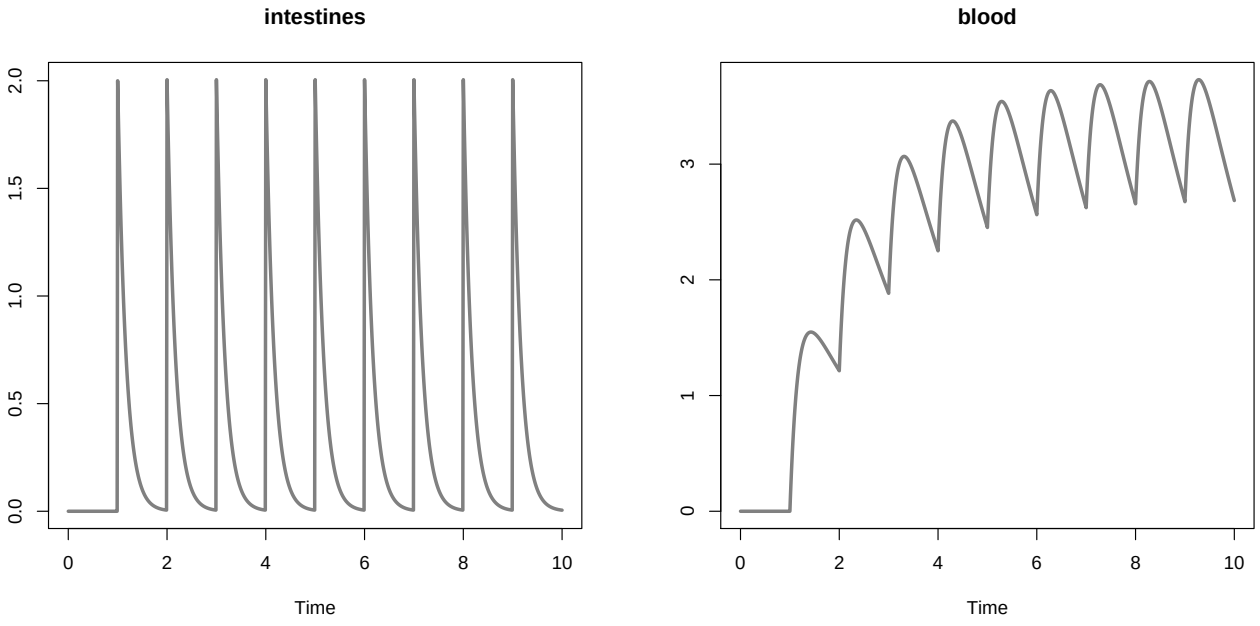
```
##    TIME y1 y2
## 1  0.00  0  0
## 2  0.01  0  0
## 3  0.02  0  0
## 4  0.03  0  0
## 5  0.04  0  0
## 6  0.05  0  0
## 7  0.06  0  0
## 8  0.07  0  0
## 9  0.08  0  0
## 10 0.09  0  0
```

```
tail(two_cpt_simulation, n = 10)
```

```
##       TIME          y1       y2
## 992   9.91 0.008530545 2.830656
## 993   9.92 0.008033781 2.814218
## 994   9.93 0.007565934 2.797850
## 995   9.94 0.007125330 2.781552
## 996   9.95 0.006710385 2.765326
## 997   9.96 0.006319605 2.749174
## 998   9.97 0.005951668 2.733095
## 999   9.98 0.005605222 2.717091
## 1000  9.99 0.005279009 2.701162
## 1001 10.00 0.004971828 2.685310
```

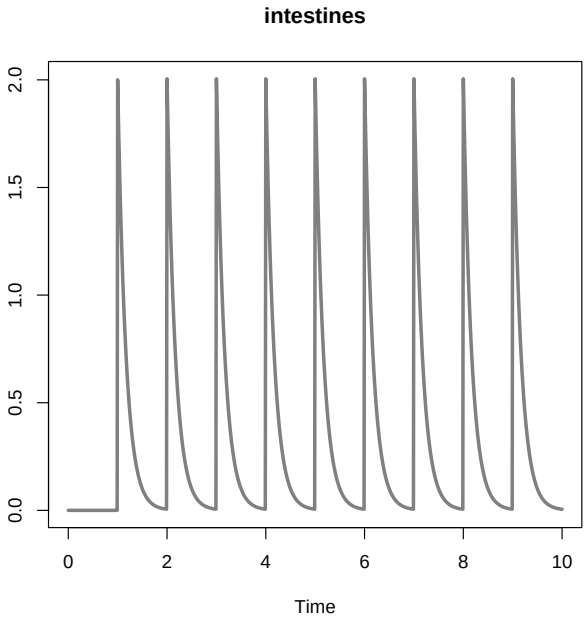
Using the `plot` generic we can plot the simulation to visualize the concentration of the drug in both compartments over time.

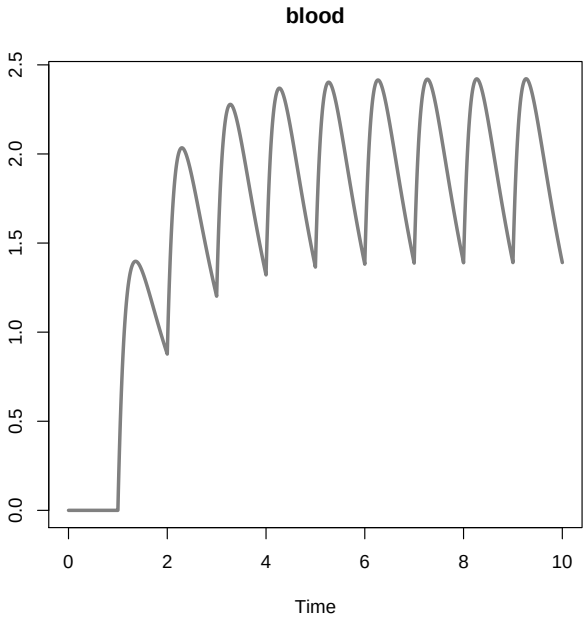
```
plot(two_cpt_simulation, col = "#808080", lwd = 3, plot_names = c("intestines", "blood"))
```



Below we simulate the model with a different parameter value for b.

```
two_cpt_simulation2 <- simulate(two_cpt, simtime = 10, parameters = c("a" = 6, "b" = 1))
plot(two_cpt_simulation2, col = "#808080", lwd = 3, plot_names = c("intestines", "blood"))
```


intestines
0.0
0.5
1.0
1.5
2.0
0
2
4
6
8
10
Time


blood
0.0
0.5
1.0
1.5
2.0
2.5
0
2
4
6
8
10
Time