Assignment week 3

Detailed analysis of the glucocorticoid receptor dynamica model

Student: Vincent Talen

Student number: 389015

Class: BFV2

Study: Bio-Informatics

Institute: Institute for Life Science & Technology

Teacher: Tsjerk Wassenaar

Date: 2022-05-27

Contents

1 Assignment 1: Assessing Model Validity

To assess the validity of the model, the experimental data will be compared to the simulation data from previous week. The comparison will be done by plotting the experimental data and the simulation data in the same graph.

1.1 Loading in experiment data

```r
# Load in data from file
data <- read.csv("MPL.csv", na.strings = "NA")
# Rename 'mRNA' and 'Free_receptor' columns to use the same name scheme
names(data)[4:5] <- c("Rm", "R")
```

The actual experiment data contains multiple values per time, if a plot is made with those raw values the plot below will result.

```r
# Create plot with raw values to demonstrate the need of using medians
ggplot(data, mapping = aes(x = time, y = R, width = 2)) + geom_line() + geom_point()
```

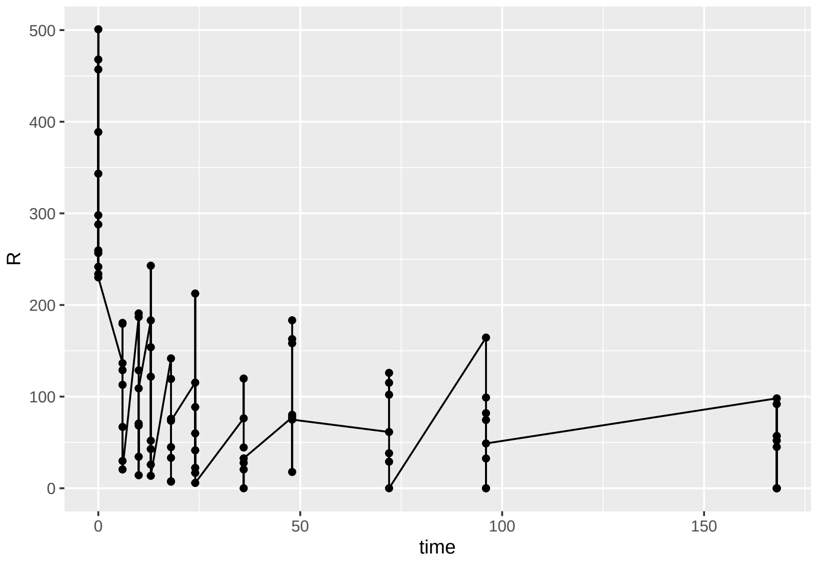


Figure 1: Demonstration of plotting raw data

As can be seen the line goes through each single point and thus is the line not that useful. So to be able to get a meaningful plot the medians are used to create a useful line of the medians that can then be used to compare the values.

```r
# Create new data frame containing the medians per time
data.medians <- aggregate(data[,c("MPL_conc","Rm","R")],
                          list(data$dose,data$time), median, na.rm = T)
names(data.medians)[1:2] <- c("dose","time")
```

1.2 Implementing GRD model

Table 1: Initial Values for MPL

Parameter	Value	Unit	Explanation
Rm	4.74	$fmol/g\ liver$	concentration of receptor mRNA
R	267	$fmol/mg\ protein$	concentration of free receptor in cytosol
DR	0	$fmol/mg\ protein$	concentration of receptor complex in cytosol
DR_N	0	$fmol/mg\ protein$	concentration of receptor complex in nucleus

Table 2: Parameter Values for MPL

Parameter	Value	Unit	Explanation
k.s_Rm	2.9	$fmol/g\ liver/h$	zero-order rate constant of receptor mRNA synthesis
k.d_Rm	0.612	-	first-order rate constant receptor mRNA degradation
IC.50_Rm	26.2	$fmol/mg\ protein$	concentration of DR_N where receptor mRNA synthesis drops to 50% of base value
k.on	0.00329	$L/nmol/h$	second-order rate constant of receptor complex formation
k.T	0.63	$1/h$	first-order rate constant of translocation of receptor complex to nucleus
k.re	0.57	$1/h$	first-order rate constant of receptor 'recovery' from nucleus to cytosol
R.f	0.49	-	fraction of receptor being recycled from complexes
k.s_R	3.22	-	first-order rate constant of receptor synthesis
k.d_R	0.0572	$1/h$	first-order rate constant of receptor degradation
D	-	$nmol/L$	plasma concentration of corticosteroid

```r
# Create function of basic GRD model
modelBasic <- function(time, state, parms) {
  ## Unpack the current state and the parameters for instant access
  with(as.list(c(state, parms)), {
    ## Calculate delta for each equation and return them in a list
    delta.Rm <- k.s_Rm * ( 1 - DR_N / ( IC.50_Rm + DR_N )) - k.d_Rm * Rm
    delta.R <- k.s_R * Rm + R.f * k.re * DR_N - k.on * D * R - k.d_R * R
    delta.DR <- k.on * D * R - k.T * DR
    delta.DR_N <- k.T * DR - k.re * DR_N
    return(list(c(delta.Rm, delta.R, delta.DR, delta.DR_N)))
    })
}

# Zero state values and time frame
basic.zero.state <- c(Rm = 4.74, R = 267, DR = 0, DR_N = 0)
basic.times <- seq(0, 168, by = 1)

# Create function that converts concentration of MPL from ng/mL to nmol/L
calculateD <- function(ng.ml.concentration) {return(ng.ml.concentration * 1000 / 374.471)}
# Define parameters determined for methylprednisolone (MPL)
basic.parameters <- c(k.s_Rm = 2.9, k.d_Rm = 0.612, IC.50_Rm = 26.2,
                      k.on = 0.00329, k.T = 0.63, k.re = 0.57, R.f = 0.49,
                      k.s_R = 3.22, k.d_R = 0.0572, D = calculateD(20))
```

1.3 Create functions to make plots for doses

Everything is coded into functions so they can be used dynamically.
The main function `createPlotsForDose` collects the data for a dose and creates the plots with it, it also uses another function called `addPlotDataAndStyle` which places the data into the plots and styles it.

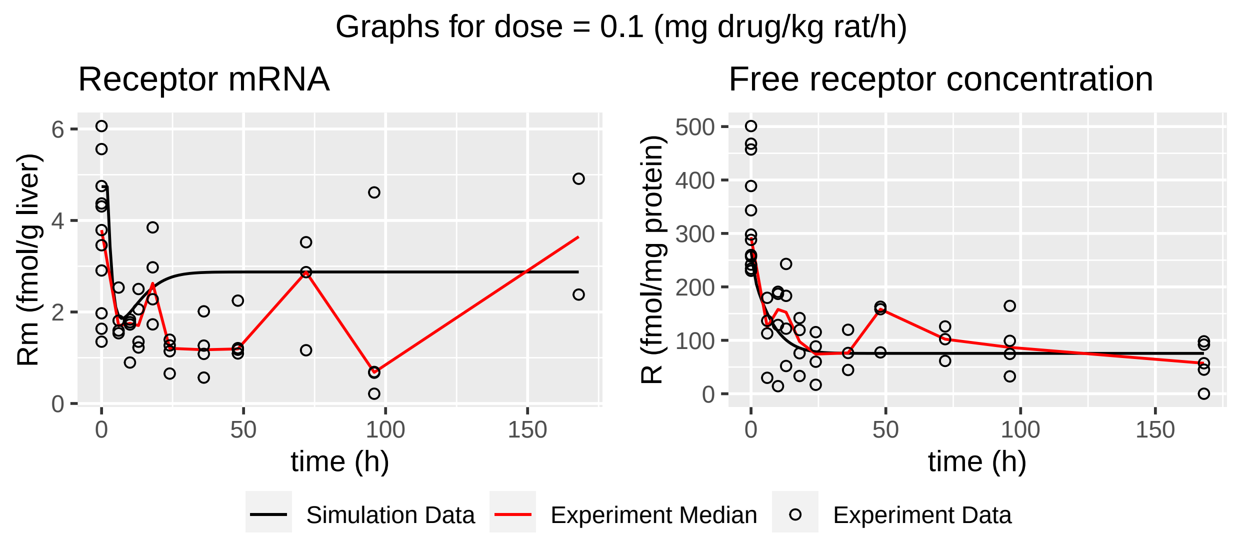
```
# Function to create plots of R and Rm for a dose
createPlotsForDose <- function(current.dose) {
  ## Change to correct D parameter for current dose
  basic.parameters$D <- calculateD(median(data$MPL_conc[data$dose == current.dose],
                                          na.rm = T))

  ## Get subset of experiment data for current dose
  cur.data.points <- subset(data, dose == 0.0 | dose == current.dose)
  ## Get subset of experiment median data for current dose
  cur.data.medians <- subset(data.medians, dose == 0.0 | dose == current.dose)
  ## Perform ODE function with the model to get simulation data
  cur.data.simulated <- as.data.frame(ode(times = basic.times, y = basic.zero.state,
                                          parms = basic.parameters, func = modelBasic,
                                          method = "euler"))

  ## Create plot for receptor mRNA
  plot.Rm <- ggplot(cur.data.simulated, aes(x = time, y = Rm)) +
    labs(title = "Receptor mRNA", y = "Rm (fmol/g liver)", x = "time (h)")
  plot.Rm <- addPlotDataAndStyle(plot.Rm, cur.data.medians, cur.data.points)

  ## Create plot for free receptor concentration
  plot.R <- ggplot(cur.data.simulated, aes(x = time, y = R)) +
    labs(title = "Free receptor concentration", y = "R (fmol/mg protein)", x = "time (h)")
  plot.R <- addPlotDataAndStyle(plot.R, cur.data.medians, cur.data.points)

  ## Print the plots in an arranged grid with the legend at the bottom
  grid.arrange(plot.Rm, plot.R, my.legend, ncol = 2, nrow = 2,
               layout_matrix = rbind(c(1,2), c(3,3)),
               widths = c(2.7, 2.7), heights = c(2.5, 0.2),
               top = sprintf("Graphs for dose = %s (mg drug/kg rat/h)", current.dose))
}
```

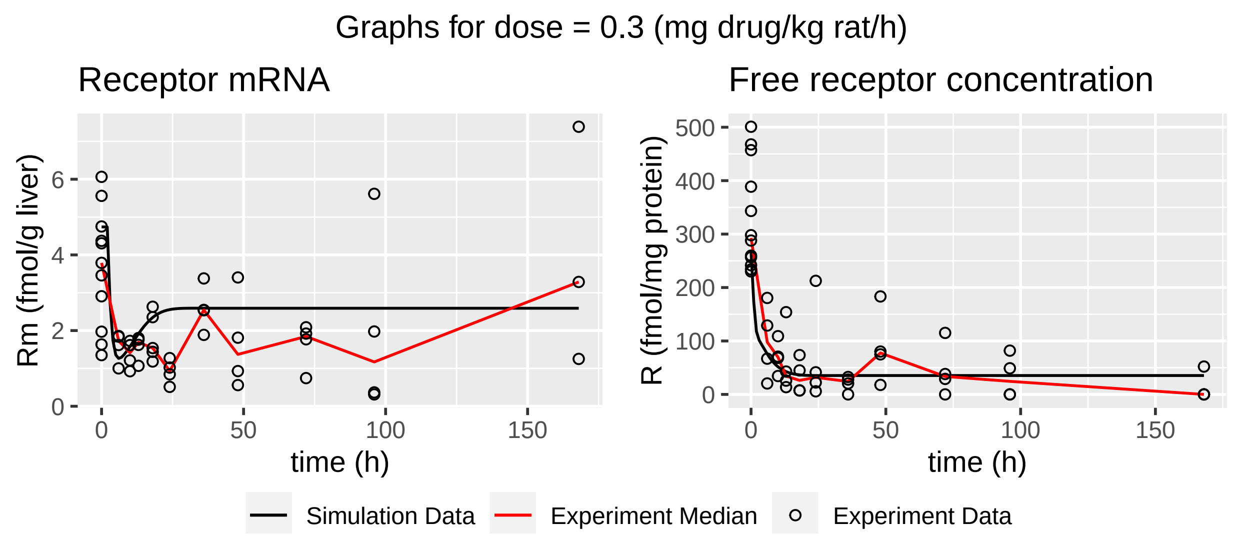
In the addPlotDataAndStyle function the lines and points are added and after which a legend is made. The legend then gets extracted using the getLegend function, put into it's own variable that can be used outside this function scope and then the legend is removed from the plot so it can be shown as a collective legend below both plots.

```r
# Function to add the data and styles to the plots
addPlotDataAndStyle <- function(my.plot, cur.data.medians, cur.data.points) {
  my.plot <- my.plot +
    ## Add the lines and data points
    geom_line(aes(color = "Simulation Data")) +
    geom_line(aes(color = "Experiment Median"), data = cur.data.medians) +
    geom_point(aes(color = "Experiment Data"), data = cur.data.points, shape = 1) +
    theme(
      legend.position = "bottom",
      plot.margin = margin(0.25, 0.25, 0.25, 0.25, "cm")) +
    ## Use scale_color_manual to color the data and also set their names in the legend
    scale_color_manual("",
      values = c("black","red","black"),
      limits = c("Simulation Data", "Experiment Median", "Experiment Data")) +
    ## Correct the displayed line types in the legend
    guides(color = guide_legend(
      override.aes = list(linetype = c(1, 1, NA), shape = c(NA, NA, 1))))

  ## Extract legend into it's own variable and remove from plot
  my.legend <<- getLegend(my.plot)
  my.plot <- my.plot + theme(legend.position="none")
  return(my.plot)
}

# Function to extract the legend from a plot
getLegend <- function(myggplot) {
  tmp <- ggplot_gtable(ggplot_build(myggplot))
  leg <- which(sapply(tmp$grobs, function(x) x$name) == "guide-box")
  legend <- tmp$grobs[[leg]]
  return(legend)
}
```

Finally, after making all the functions, gather the unique doses (except 0.0) and use lapply to create the plots for each dose.

```r
# Get all unique doses present in the data except 0.0
all.doses <- unique(data$dose)
all.doses <- all.doses[!all.doses %in% (0.0)]

# Create plots for each dose by using lapply() on createPlotsForDose()
invisible(lapply(all.doses, createPlotsForDose))
```





1.4 Comparing the resulting plots

When looking the resulting plots it can be seen that the receptor mRNA is not entirely as expected like the model but the free receptor concentration in the cytosol is close to the expectations. Around 50 hours a spike can be seen in the free receptor concentration that is not expected but shortly after it decreases rather than reaching a steady state after around 25 hours like the model. The amount of receptor mRNA varies a lot more than expected as time passes, in the model a steady state is expected after around 25 hours too but the experiment is completely different and the amount of receptor mRNA keeps changing. At first it does decrease like the model but then has a little spike and after some time an even bigger spike that dips really low. After 100 hours the receptor mRNA amount keeps climbing to above the expected steady state after being below it for almost the entire time.

The reason for the different results could be that the medicine is not as effective as thought and that it thus does not really cause the receptor mRNA to reach a stable state over time. With the lower dose more variation can be seen in the amount of receptor mRNA present than in the higher dose, which is a little bit more stable. Overall it seems that the effect is highest when first administered and the effectiveness declines because of the big decrease, the high variations and the steady increase at the end.

2 Assignment 2: Simulating Multiple Scenario's

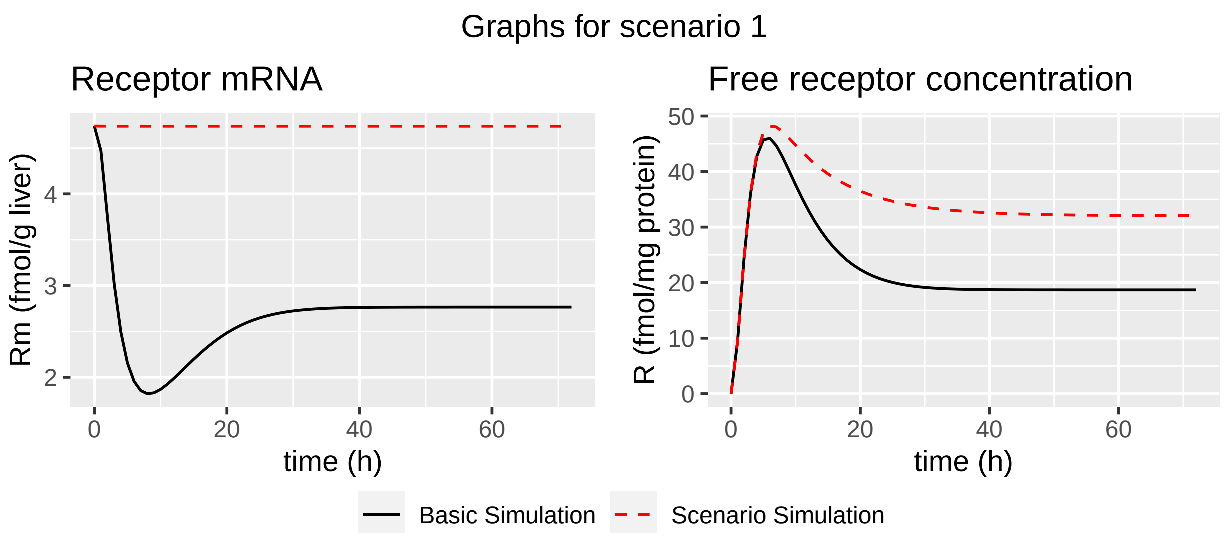
To check the effect and importance of parameter values, simulations of multiple scenario's will be done and then compared to the basic scenario of the given determined values from assignment week 2.

```r
# Set time frame for all scenarios, plots will be limited to scenario length
scenarios.times <- seq(0, 300, by = 1)
# Get the basic simulation data that will be used for all scenario's
basic.simulation <- as.data.frame(ode(times = scenarios.times, y = basic.zero.state,
                                      parms = basic.parameters, func = modelBasic))
```

2.1 Activated receptor complex concentration without auto-regulation of glucocorticoid receptor

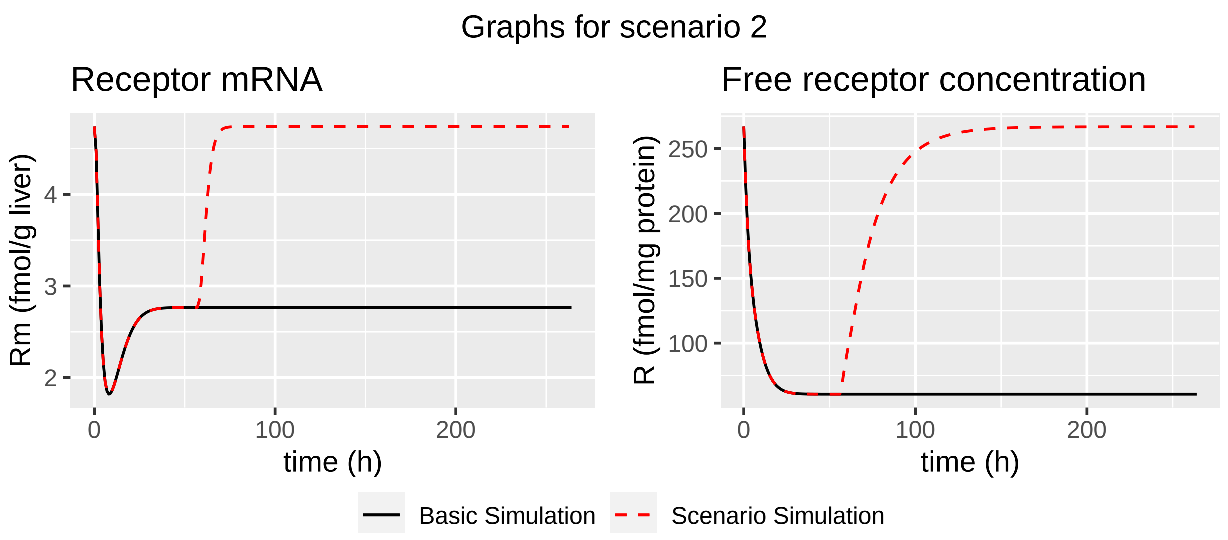
Auto-regulation of glucocorticoid receptor is caused by the activated receptor complexes in the nucleus (`DR(N)`). It is expressed into the parameter `IC.50_Rm` which is the concentration of DR(N) where receptor mRNA synthesis drops to 50% of it's base value. So if DR(N), which needs drugs to form, does not auto-regulate the synthesis of the receptor mRNA then IC.50_Rm can be removed from the equation. After removing `IC.50_Rm` from the equation `1 - DR_N / DR_N` remains, whose outcome is always `1-1 = 0`, so this entire part of the equation can be removed. Without the auto-regulation from DR(N) there is no effect of the drug on receptor mRNA synthesis anymore.

```r
# Create new model where there is no receptor mRNA auto-regulation
modelScen1 <- function(time, state, parms) {
  ## Unpack the current state and the parameters for instant access
  with(as.list(c(state, parms)), {
    ## Calculate delta for each equation and return them in a list
    delta.Rm <- k.s_Rm - k.d_Rm * Rm
    delta.R <- k.s_R * Rm + R.f * k.re * DR_N - k.on * D * R - k.d_R * R
    delta.DR <- k.on * D * R - k.T * DR
    delta.DR_N <- k.T * DR - k.re * DR_N
    return(list(c(delta.Rm, delta.R, delta.DR, delta.DR_N)))
    })
}

# Run the simulation for the no mRNA synthesis auto-regulation scenario
scen1.simulation <- as.data.frame(ode(times = scenarios.times, y = basic.zero.state,
                                      parms = basic.parameters, func = modelScen1))

# Create Rm plot for scenario
scen1.plot.Rm <- ggplot(basic.simulation, aes(x = time, y = Rm)) +
  labs(title = "Receptor mRNA",
       y = "Rm (fmol/g liver)", x = "time (h)") +
  xlim(NA, 72) +
  ## Add the lines and data points
  geom_line(aes(color = "Basic Simulation")) +
  geom_line(aes(color = "Scenario Simulation"),
            data = scen1.simulation, linetype = "dashed") +
  theme(legend.position = "bottom", plot.margin = margin(0.25, 0.25, 0.25, 0.25, "cm")) +
  ## Use scale_color_manual to color the data and also set their names in the legend
  scale_color_manual("", values = c("black","red"),
                     limits = c("Basic Simulation", "Scenario Simulation")) +
  ## Correct the displayed line types in the legend
  guides(color = guide_legend(override.aes = list(linetype = c(1, 2)))) +
  theme(legend.position="none")

# Create DR_N plot for scenario
scen1.plot.DR_N <- ggplot(basic.simulation, aes(x = time, y = DR_N)) +
  labs(title = "Free receptor concentration",
       y = "R (fmol/mg protein)", x = "time (h)") +
  xlim(NA, 72) +
  ## Add the lines and data points
  geom_line(aes(color = "Basic Simulation")) +
  geom_line(aes(color = "Scenario Simulation"),
            data = scen1.simulation, linetype = "dashed") +
  theme(legend.position = "bottom", plot.margin = margin(0.25, 0.25, 0.25, 0.25, "cm")) +
  ## Use scale_color_manual to color the data and also set their names in the legend
  scale_color_manual("", values = c("black","red"),
                     limits = c("Basic Simulation", "Scenario Simulation")) +
  ## Correct the displayed line types in the legend
  guides(color = guide_legend(override.aes = list(linetype = c(1, 2))))

# Extract legend
scen1.legend <<- getLegend(scen1.plot.DR_N)
scen1.plot.DR_N <- scen1.plot.DR_N + theme(legend.position="none")

## Print the plots in an arranged grid with the legend at the bottom
grid.arrange(scen1.plot.Rm, scen1.plot.DR_N, scen1.legend, ncol = 2, nrow = 2,
             layout_matrix = rbind(c(1,2), c(3,3)),
             widths = c(2.7, 2.7), heights = c(2.5, 0.2),
             top = "Graphs for scenario 1")
```



2.2 Receptor and mRNA concentrations when stopping drug treatment at steady state

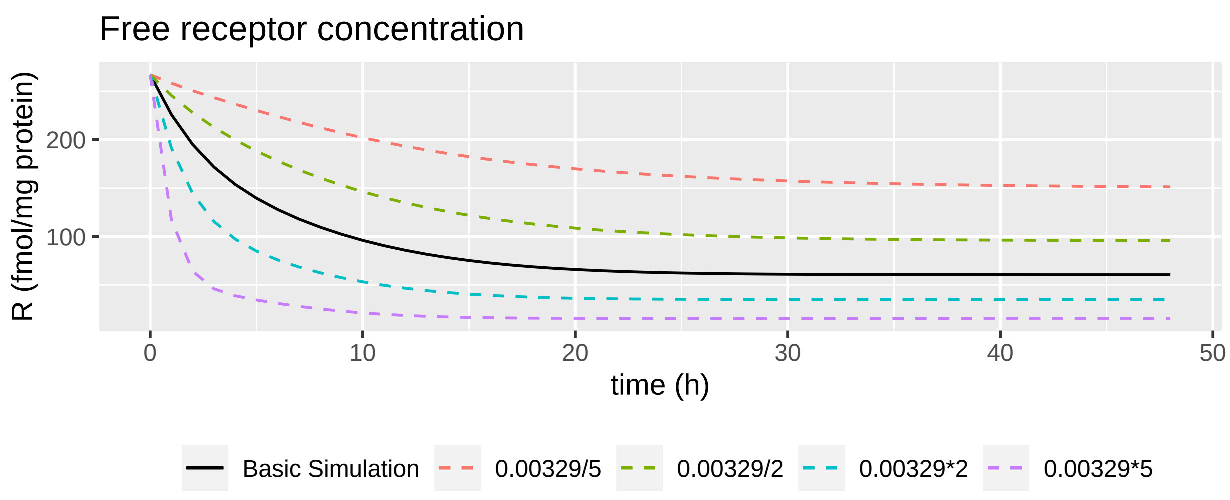
```r
# Zero state values now including D
scen2.zero.state <- c(Rm = 4.74, R = 267, DR = 0, DR_N = 0, D = calculateD(20))
# Define parameters determined for methylprednisolone (MPL) without D
scen2.parameters <- c(k.s_Rm = 2.9, k.d_Rm = 0.612, IC.50_Rm = 26.2, k.on = 0.00329,
                      k.T = 0.63, k.re = 0.57, R.f = 0.49, k.s_R = 3.22, k.d_R = 0.0572)

# Create new model that includes D in state deltas
modelScen2 <- function(time, state, parms) {
  ## Unpack the current state and the parameters for instant access
  with(as.list(c(state, parms)), {
    ## Calculate delta for each equation and return them in a list
    delta.Rm <- k.s_Rm * ( 1 - DR_N / ( IC.50_Rm + DR_N )) - k.d_Rm * Rm
    delta.R <- k.s_R * Rm + R.f * k.re * DR_N - k.on * D * R - k.d_R * R
    delta.DR <- k.on * D * R - k.T * DR
    delta.DR_N <- k.T * DR - k.re * DR_N
    delta.D <- 0
    return(list(c(delta.Rm, delta.R, delta.DR, delta.DR_N, delta.D)))
    })
}

# Root function: root.normal activates event and root.terminate terminates simulation
scen2.root <- function(time, state, parms) {
  # Normal root: Use model to get delta's return total delta - 0.0001
  delta.list <- unlist(modelScen2(time, state, parms))
  root.normal <- sum(abs(delta.list)) - 1e-4
  # Terminator root: if sum of normal root and D state are 0
  root.terminate <- root.normal + state["D"]
  return(c(root.normal, root.terminate))
}

# The event (D <- 0) that happens when root[1] (first steady state) is triggered
scen2.event <- function(time, state, parms) {
  state["D"] <- 0
  return(state)
}

# Run the simulation for the steady state drug stop scenario
scen2.out <- ode(func = modelScen2,
  times = scenarios.times, y = scen2.zero.state, parms = scen2.parameters,
  events = list(func = scen2.event, root = TRUE, terminalroot = 2), rootfun = scen2.root)
scen2.simulation <- as.data.frame(scen2.out)

# Create Rm plot for scenario
scen2.plot.Rm <- ggplot(basic.simulation, aes(x = time, y = Rm)) +
  labs(title = "Receptor mRNA",
       y = "Rm (fmol/g liver)", x = "time (h)") +
  xlim(NA, attributes(scen2.out)$dim[1]) +
  ## Add the lines and data points
  geom_line(aes(color = "Basic Simulation")) +
  geom_line(aes(color = "Scenario Simulation"),
            data = scen2.simulation, linetype = "dashed") +
  theme(legend.position = "bottom", plot.margin = margin(0.25, 0.25, 0.25, 0.25, "cm")) +
  ## Use scale_color_manual to color the data and also set their names in the legend
  scale_color_manual("", values = c("black","red"),
                     limits = c("Basic Simulation", "Scenario Simulation")) +
  ## Correct the displayed line types in the legend
  guides(color = guide_legend(override.aes = list(linetype = c(1, 2)))) +
  theme(legend.position="none")

# Create R plot for scenario
scen2.plot.R <- ggplot(basic.simulation, aes(x = time, y = R)) +
  labs(title = "Free receptor concentration",
       y = "R (fmol/mg protein)", x = "time (h)") +
  xlim(NA, attributes(scen2.out)$dim[1]) +
  ## Add the lines and data points
  geom_line(aes(color = "Basic Simulation")) +
  geom_line(aes(color = "Scenario Simulation"),
            data = scen2.simulation, linetype = "dashed") +
  theme(legend.position = "bottom", plot.margin = margin(0.25, 0.25, 0.25, 0.25, "cm")) +
  ## Use scale_color_manual to color the data and also set their names in the legend
  scale_color_manual("", values = c("black","red"),
                     limits = c("Basic Simulation", "Scenario Simulation")) +
  ## Correct the displayed line types in the legend
  guides(color = guide_legend(override.aes = list(linetype = c(1, 2))))

# Extract legend
scen2.legend <<- getLegend(scen2.plot.R)
scen2.plot.R <- scen2.plot.R + theme(legend.position="none")

## Print the plots in an arranged grid with the legend at the bottom
grid.arrange(scen2.plot.Rm, scen2.plot.R, scen2.legend, ncol = 2, nrow = 2,
             layout_matrix = rbind(c(1,2), c(3,3)),
             widths = c(2.7, 2.7), heights = c(2.5, 0.2),
             top = "Graphs for scenario 2")
```



2.3 Effect of different k.on and k.re values on receptor and mRNA dynamics

```
new.values1 <- list(c(k.on = "0.00329/5"),
                   c(k.on = "0.00329/2"),
                   c(k.on = "0.00329*2"),
                   c(k.on = "0.00329*5"))

new.values2 <- list(c(k.re = "0.57/5"),
                    c(k.re = "0.57/2"),
                    c(k.re = "0.57*2"),
                    c(k.re = "0.57*5"))

# Function
getDataSimulations <- function (new.values.string) {
  ## Define new parameters variable so a more fitting name can be used
  cur.scen.parameters <- basic.parameters

  ## Turn the new values from strings of expressions to numeric values
  new.values.num <- sapply(new.values.string, function(x) eval(parse(text=x)))
  ## Replace the parameters with the new values
  cur.scen.parameters[names(new.values.num)] <- new.values.num

  ## Run the simulation
  cur.scen.out <- ode(func = modelBasic, times = basic.times, y = basic.zero.state, parms = cur.scen.pa
  cur.scen.simulation <- as.data.frame(cur.scen.out)
  return(cur.scen.simulation)
}
```

```
mainFunction <- function(new.values) {
  ## Get simulation names and data for each altered parameter scenario
  simulations.names <- lapply(new.values, function(x) paste(unname(x), collapse=", "))
  simulations.data <- lapply(new.values, getDataSimulations)

  ## Create R plot for scenario
  plotje <- ggplot(basic.simulation, aes(x = time, y = R)) +
    labs(title = "Free receptor concentration",
         y = "R (fmol/mg protein)", x = "time (h)") +
    theme(legend.position = "bottom", plot.margin = margin(0.25, 0.25, 0.25, 0.25, "cm")) +
    addGeomLines(simulations.names, simulations.data) +
    xlim(NA, 48)
  print(plotje)

  ## Extract legend
  # scen3.legend <<- getLegend(plotje)
  # plotje <- plotje + theme(legend.position="none")
}

addGeomLines <- function(simulations.names, simulations.data) {
  ## Add lines of the simulations
  x1 <- geom_line(aes(color = "Basic Simulation"))
  xx <- mapply(
    function(sim.name, sim.data)
      geom_line(aes(color = sim.name), data = sim.data, linetype = "dashed"),
    simulations.names, simulations.data)

  ## Use scale_color_manual to color the data and also set their names in the legend
  extra.colors <- hue_pal()(length(simulations.names))
  y <- scale_color_manual("", values = c("black", extra.colors), limits = c("Basic Simulation", simulat
  ## Correct the displayed line types in the legend
  z <- guides(color = guide_legend(override.aes = list(linetype = c(1, rep(2, length(simulations.names)
  return(list(x1, unlist(xx), y, z))
}

mainFunction(new.values1)
```



2.4 Effects of completely blocking receptor synthesis

2.5 System dynamics with increased or decreased baseline mRNA production

```
new.values3 <- list(c(k.s_Rm = "2.9/5", k.d_Rm = "2.9/5/4.74"),
                    c(k.s_Rm = "2.9/2", k.d_Rm = "2.9/2/4.74"),
                    c(k.s_Rm = "2.9*2", k.d_Rm = "2.9*2/4.74"),
                    c(k.s_Rm = "2.9*5", k.d_Rm = "2.9*5/4.74"))
```

References

[1] Barnes, P.J. (2011), *Glucocorticosteroids: current and future directions*, British Journal of Pharmacology, 163: 29-43, https://doi.org/10.1111/j.1476-5381.2010.01199.x (accessed May 11, 2022).