Principles of Modelling and Simulation in Epidemiology

Andreas Karlsson

November 25, 2013

Laboratory exercise 4

Required R-packages

```
require(deSolve)  #For the deterministic solutions (also for initial and environmental stochasticity)
require(stats)  #The stats package has the 'optim'-function
```

8 Input Data

```
rm(list = ls(all = TRUE))
CA_STAT_0_amp = c(0, 0, 0, 0, 1, 4.5, 14, 32, 53, 54, 48, 43, 36.5, 29, 26.5,
    24, 22, 18)
CA_STAT_0 <- approxfun(seq(0, 85, by = 5), CA_STAT_0_amp, rule = 1)

CA_STAT_1_amp <- rep(0, 85)
CA_STAT_1_amp[32:55] = c(21.7, 25.4, 29.2, 34.6, 38.7, 42, 44.7, 49.4, 54.1,
    52.7, 52.3, 46.9, 41.7, 39, 36.8, 32.1, 26.6, 24.5, 22.1, 19.4, 18.1, 20,
    19.5, 19.5)
CA_STAT_1 <- approxfun(1:85, CA_STAT_1_amp, rule = 2)

SCR_STAT_amp <- rep(0, 85)
SCR_STAT_amp[32:55] <- c(10, 12.8, 16.2, 24.7, 37.5, 54.7, 80, 132, 178.6, 206.7,
    212.2, 211.4, 202.3, 183.4, 159.9, 123.7, 95, 78, 61.4, 46.6, 37.1, 31.3,
    29.9, 26.6)
SCR_STAT <- approxfun(1:85, SCR_STAT_amp, rule = 2)

Fraction_alive_amp <- c(1, 0.993, 0.993, 0.992, 0.991, 0.989, 0.987, 0.984,
    0.98, 0.973, 0.963, 0.948, 0.925, 0.889, 0.834, 0.746, 0.601, 0.417)
Fraction_alive <- approxfun(seq(0, 85, by = 5), Fraction_alive_amp, rule = 1)
```

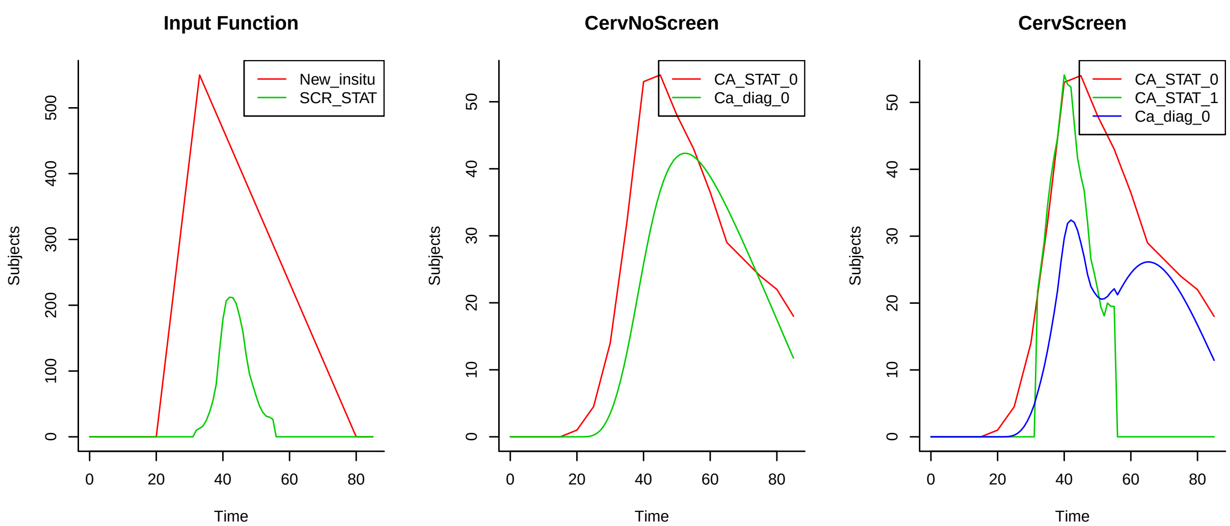
6 Realisation of the Model

```
CervNoScreen <- function(time, state, parameters) {
    with(as.list(c(state, parameters)), {
        D <- T1/(3 * P^(1/3))
        R <- T1/(3 * (1 - P^(1/3)))
        dSITU1_0 <- New_insitu(time) - SITU1_0/R - SITU1_0/D
        dSITU2_0 <- SITU1_0/D - SITU2_0/R - SITU2_0/D
        dSITU3_0 <- SITU2_0/D - SITU3_0/R - SITU3_0/D
        dInvasive_0 <- SITU3_0/D - Invasive_0/T2
        Ca_diag_0 <- (Invasive_0/T2)
        return(list(c(dSITU1_0, dSITU2_0, dSITU3_0, dInvasive_0), Ca_diag_0 = Ca_diag_0))
    })
}
CervScreen <- function(time, state, parameters) {
    with(as.list(c(state, parameters)), {
        D <- T1/(3 * P^(1/3))
        R <- T1/(3 * (1 - P^(1/3)))
        SITU_1 <- SITU1_1 + SITU2_1 + SITU3_1
        dSITU1_1 <- New_insitu(time) - SITU1_1/R - SITU1_1/D - SCR_STAT(time) *
            (SITU1_1/max(SITU_1, 1e-15))
        dSITU2_1 <- SITU1_1/D - SITU2_1/R - SITU2_1/D - SCR_STAT(time) * (SITU2_1/max(SITU_1,
            1e-15))
        dSITU3_1 <- SITU2_1/D - SITU3_1/R - SITU3_1/D - SCR_STAT(time) * (SITU3_1/max(SITU_1,
            1e-15))
        dInvasive_1 <- SITU3_1/D - Invasive_1/T2 - Invasive_1 * SCR_STAT(time)/max(SITU_1,
            1e-15)
        Ca_diag_1 <- Invasive_1/T2
        Ca_diag_scr <- Invasive_1 * SCR_STAT(time)/max(SITU_1, 1e-15)
        Ca_diag_both <- Ca_diag_1 + Ca_diag_scr
        return(list(c(dSITU1_1, dSITU2_1, dSITU3_1, dInvasive_1), Ca_diag_both = Ca_diag_both))
    })
}
callCervModels <- function(parameters, NewInsituLookUpTable = FALSE, make_plots = FALSE) {
    if (NewInsituLookUpTable) {
        New_insitu <<- approxfun(seq(0, 85, by = 2.5), parameters[grep("Y_tri",
            names(parameters))], rule = 2)
    } else {
        New_insitu <<- approxfun(c(0, parameters["a"], parameters["b"], parameters["c"],
            85), c(0, 0, parameters["y"], 0, 0), rule = 2)
    }
    dt <- 1
    times <- seq(0, 85, by = dt)

    Ca_diag_0 <- 0
    init_0 <- c(SITU1_0 = 0, SITU2_0 = 0, SITU3_0 = 0, Invasive_0 = 0)
    out_nscr <- as.data.frame(ode(y = init_0, times = times, func = CervNoScreen,
        parms = parameters))
    out_nscr$time <- NULL

    Ca_diag_1 <- 0
    init_1 <- c(SITU1_1 = 0, SITU2_1 = 0, SITU3_1 = 0, Invasive_1 = 0)
    out_scr <- as.data.frame(ode(y = init_1, times = times, func = CervScreen,
        parms = parameters))
    out_scr$time <- NULL

    V_0 <- sum((out_nscr["Ca_diag_0"] - CA_STAT_0(times))^2)
    V_1 <- sum((out_scr[times >= 31.5 & times <= 55.5, 5] - CA_STAT_1(times[times >=
```

```
        31.5 & times <= 55.5]))^2)
    V <- V_0 + V_1 + ifelse(parameters["P"] <= 0, 1e+07, 0)
    if (make_plots) {
        layout(matrix(1:3, 1, 3, byrow = TRUE))
        matplot(times, cbind(New_insitu(times), SCR_STAT(times)), type = "l",
            xlab = "Time", ylab = "Subjects", main = "Input Function", lwd = 1,
            lty = 1, bty = "l", col = 2:4)
        legend("topright", c("New_insitu", "SCR_STAT"), lty = 1, col = 2:3)
        matplot(times, cbind(CA_STAT_0(times), out_nscr["Ca_diag_0"]), type = "l",
            xlab = "Time", ylab = "Subjects", main = "CervNoScreen", lwd = 1,
            lty = 1, bty = "l", col = 2:3)
        legend("topright", c("CA_STAT_0", "Ca_diag_0"), lty = 1, col = 2:3)
        matplot(times, cbind(CA_STAT_0(times), CA_STAT_1(times), out_scr["Ca_diag_both"]),
            type = "l", xlab = "Time", ylab = "Subjects", main = "CervScreen",
            lwd = 1, lty = 1, bty = "l", col = 2:4)
        legend("topright", c("CA_STAT_0", "CA_STAT_1", "Ca_diag_0"), lty = 1,
            col = 2:4)
        print("Obj. functions:")
        print(round(data.frame(V_0 = V_0, V_1 = V_1, V = V, row.names = NULL)))
    }
    return(V)
}
```

10.2 Results when using the initial values

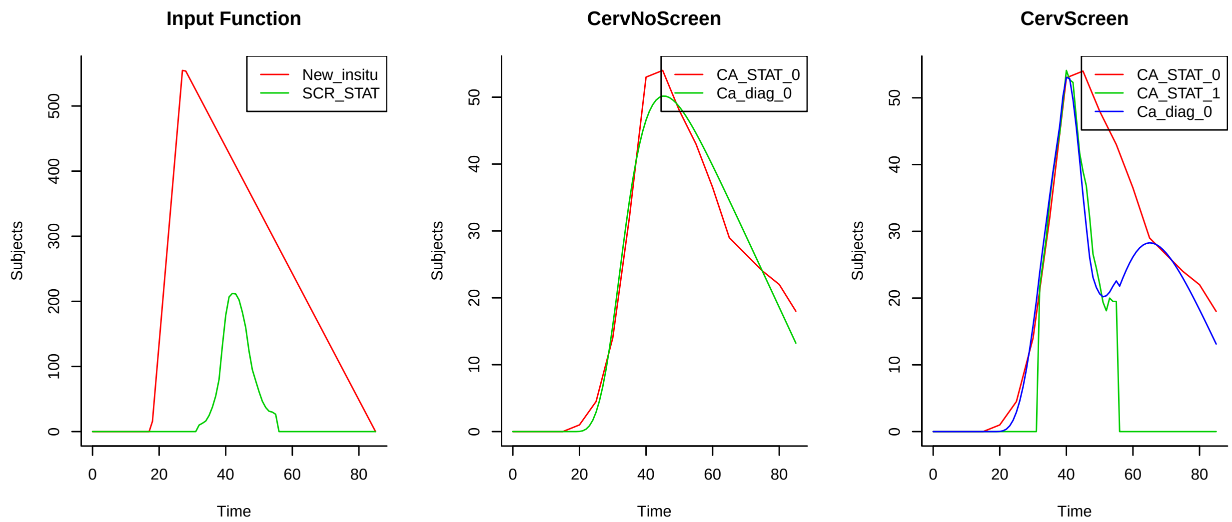
```
parameters <- c(P = 0.1, T1 = 10, T2 = 5, a = 20, b = 33, c = 80, y = 550)
res_value <- callCervModels(parameters, make_plots = TRUE)
```




```
## [1] "Obj. functions:"
##    V_0  V_1     V
## 1 7933 5552 13485
```

10.3 Best fit of P, T1, T2, a, b, c & y

```
opt_res <- optim(parameters, callCervModels, method = "BFGS", control = list(reltol = 0.001))
tmp <- callCervModels(parameters = opt_res$par, make_plots = TRUE)
```




```
## [1] "Obj. functions:"
##   V_0 V_1   V
## 1 585 160 744
```

```
print(opt_res$par)
```

```
##        P       T1       T2        a        b        c        y
##   0.1103   8.0435   4.3058  17.7360  27.1256  85.8930 562.1455
```

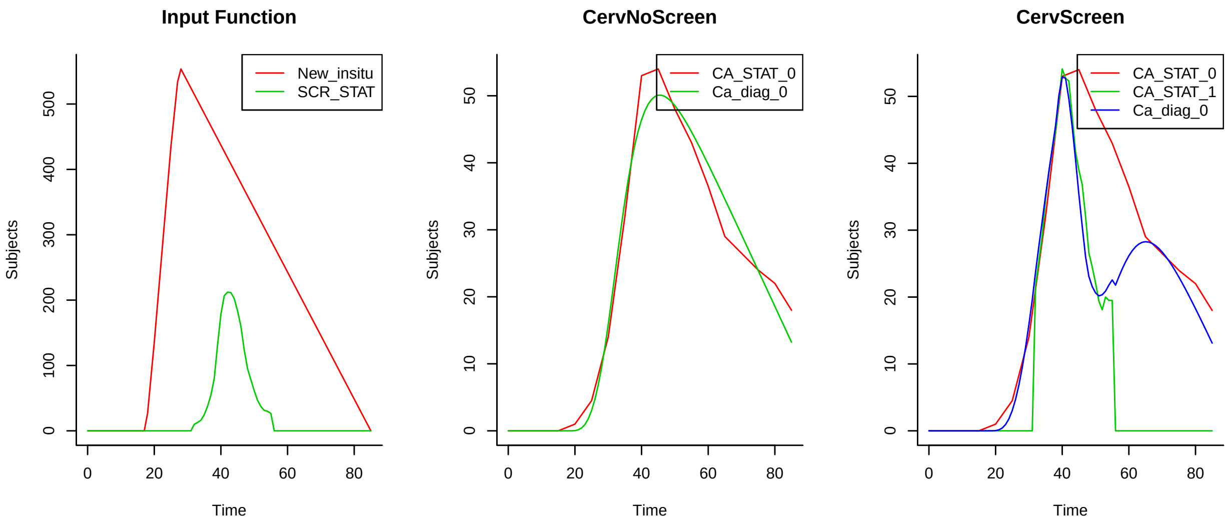
```
print(paste("The obj. function was reduced by:", round((res_value - opt_res$value)/res_value *
    100), "%"))
```

```
## [1] "The obj. function was reduced by: 94 %"
```

10.3 Repeating previous with a lookup table

Got a small difference since the triangle got a flat top

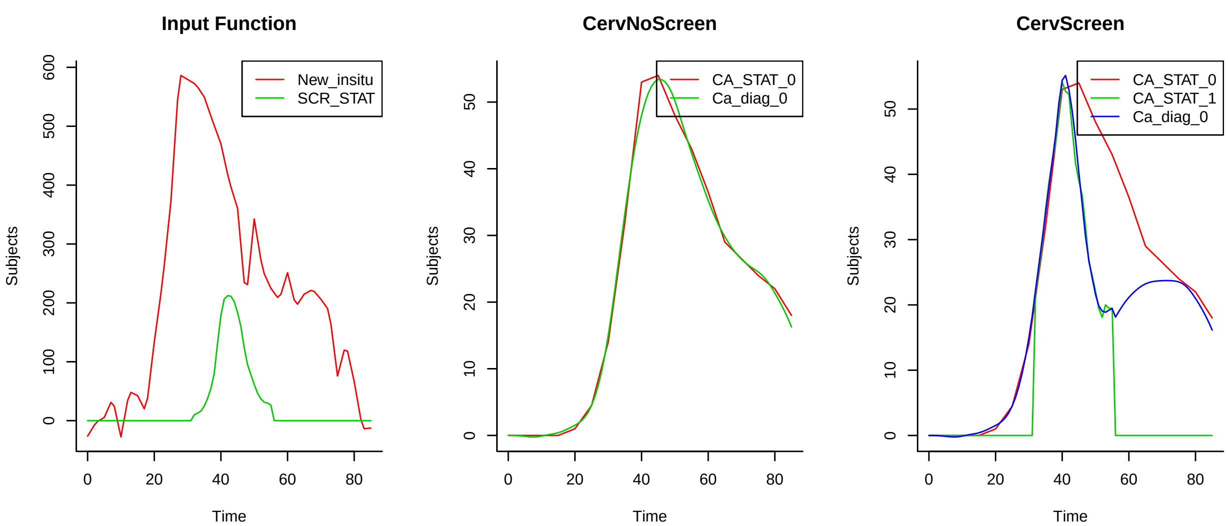
```
triang <- approxfun(c(0, opt_res$par["a"], opt_res$par["b"], opt_res$par["c"],
    85), c(0, 0, opt_res$par["y"], 0, 0), rule = 2)
triang_time <- seq(0, 85, by = 2.5)
parameters_lookup <- c(opt_res$par[c("P", "T1", "T2")], Y_tri = triang(triang_time))
res_triang <- callCervModels(parameters_lookup, NewInsituLookUpTable = TRUE,
    make_plots = TRUE)
```




```
## [1] "Obj. functions:"
##   V_0 V_1   V
## 1 588 161 749
```

10.3 Optimising trianlge table and P, T1 & T2 with Nelder-Mead (Simplex)

```
opt_tri_res_Simplex <- optim(parameters_lookup, callCervModels, method = c("Nelder-Mead"),
    control = list(reltol = 0.001), NewInsituLookUpTable = TRUE)
tmp <- callCervModels(parameters = opt_tri_res_Simplex$par, NewInsituLookUpTable = TRUE,
    make_plots = TRUE)
```




```
## [1] "Obj. functions:"
##   V_0 V_1   V
```

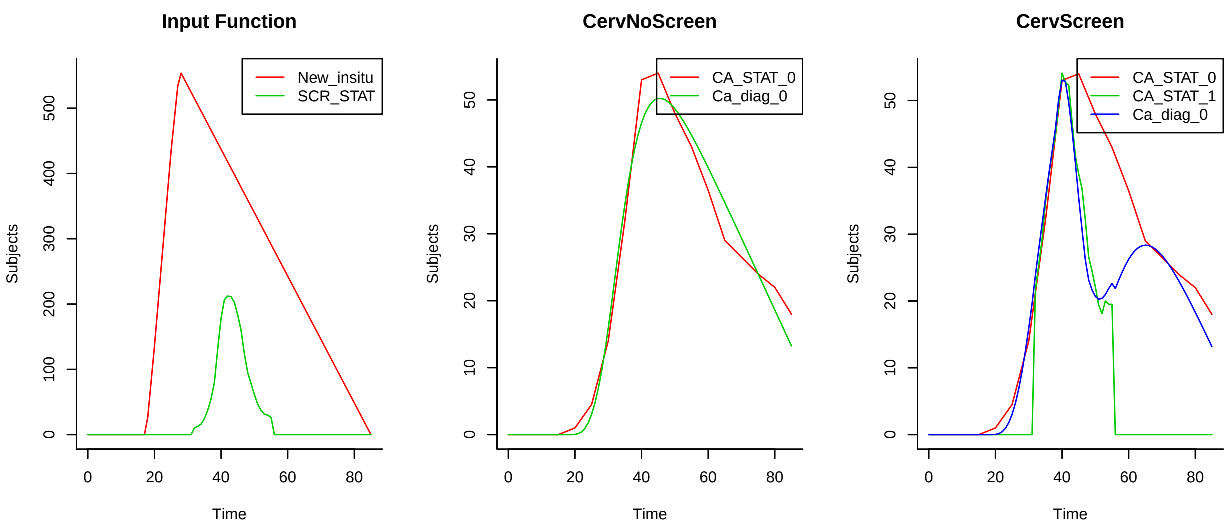
```
## 1 102  43 145
```

```r
print(opt_tri_res_Simplex$par)
```

```
##         P        T1        T2   Y_tri1   Y_tri2   Y_tri3   Y_tri4   Y_tri5
##    0.1097    8.1722    4.3983 -26.3768  -2.9977    5.5069  37.4875 -27.6678
##    Y_tri6    Y_tri7    Y_tri8   Y_tri9  Y_tri10  Y_tri11  Y_tri12  Y_tri13
##   49.4340   42.4304   14.6182 133.4043 236.6920 372.7048 587.9885 579.4474
##   Y_tri14   Y_tri15   Y_tri16  Y_tri17  Y_tri18  Y_tri19  Y_tri20  Y_tri21
## 570.8231 549.6331 508.9976 470.1259 406.1226 360.5462 203.1061 342.4207
##   Y_tri22   Y_tri23   Y_tri24  Y_tri25  Y_tri26  Y_tri27  Y_tri28  Y_tri29
## 255.6821 224.8025 205.3864 251.2069 193.5194 214.8555 222.7734 206.2065
##   Y_tri30   Y_tri31   Y_tri32  Y_tri33  Y_tri34  Y_tri35
## 186.6415  75.8840 130.8789  66.3487 -14.0121 -12.5872
```

10.3 Optimising trianlge table and P, T1 & T2 with BFGS (Quasi-Newton)

```r
opt_tri_res_BFGS <- optim(parameters_lookup, callCervModels, method = c("BFGS"),
    control = list(reltol = 0.001), NewInsituLookUpTable = TRUE)
tmp <- callCervModels(parameters = opt_tri_res_BFGS$par, NewInsituLookUpTable = TRUE,
    make_plots = TRUE)
```




```
## [1] "Obj. functions:"
##   V_0 V_1   V
## 1 591 158 749
```

```r
print(opt_tri_res_BFGS$par)
```

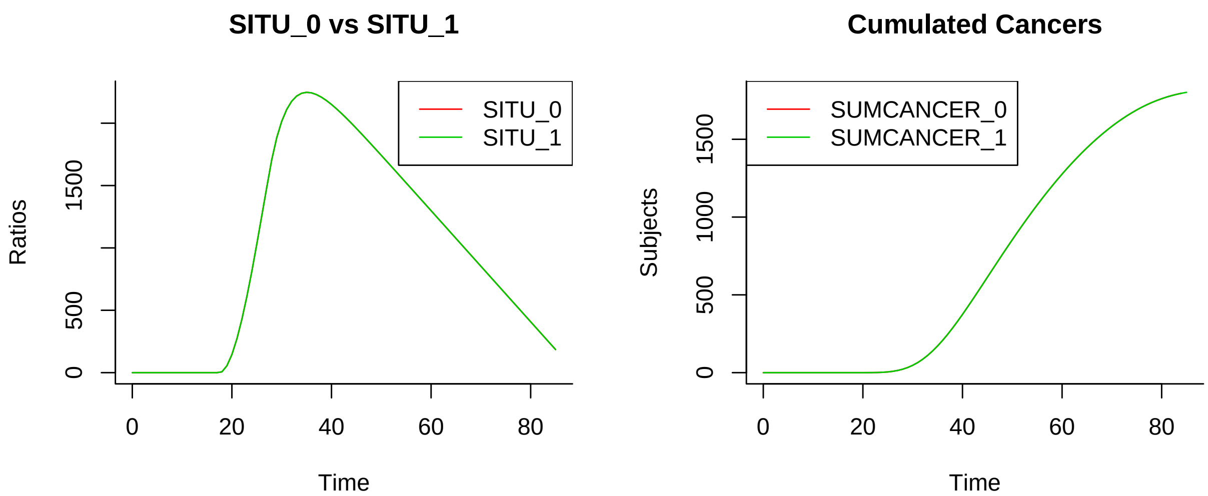
```
##           P         T1         T2     Y_tri1     Y_tri2     Y_tri3
##   1.106e-01  8.043e+00  4.306e+00 -7.840e-07 -2.833e-07 -8.946e-07
##      Y_tri4     Y_tri5     Y_tri6     Y_tri7     Y_tri8     Y_tri9
## -3.144e-08  4.689e-08 -8.129e-08 -5.564e-07 -3.944e-07  1.355e+02
##     Y_tri10    Y_tri11    Y_tri12    Y_tri13    Y_tri14    Y_tri15
##   2.852e+02  4.349e+02  5.585e+02  5.342e+02  5.099e+02  4.857e+02
##     Y_tri16    Y_tri17    Y_tri18    Y_tri19    Y_tri20    Y_tri21
##   4.614e+02  4.371e+02  4.128e+02  3.885e+02  3.642e+02  3.400e+02
##     Y_tri22    Y_tri23    Y_tri24    Y_tri25    Y_tri26    Y_tri27
##   3.157e+02  2.914e+02  2.671e+02  2.428e+02  2.185e+02  1.943e+02
##     Y_tri28    Y_tri29    Y_tri30    Y_tri31    Y_tri32    Y_tri33
##   1.700e+02  1.457e+02  1.214e+02  9.713e+01  7.285e+01  4.857e+01
##     Y_tri34    Y_tri35
##   2.428e+01  3.289e-10
```

12 Optimisation of screening

```
AppliedCervScreen <- function(time, state, parameters) {
    with(as.list(c(state, parameters)), {
        D <- T1/(3 * P^(1/3))
        R <- T1/(3 * (1 - P^(1/3)))
        SITU <- SITU1 + SITU2 + SITU3
        dSITU1 <- New_insitu(time) - SITU1/R - SITU1/D
        dSITU2 <- SITU1/D - SITU2/R - SITU2/D
        dSITU3 <- SITU2/D - SITU3/R - SITU3/D
        dInvasive <- SITU3/D - Invasive/T2
        Ca_diag_both <- Invasive/T2
        dSUMCANCER <- Ca_diag_both * Fraction_alive(time)
        return(list(c(dSITU1, dSITU2, dSITU3, dInvasive, dSUMCANCER), Ca_diag_both = Ca_diag_both,
            SITU = SITU))
    })
}
callAppliedCervModels <- function(ages, opt_parameters, make_plots = FALSE) {
    New_insitu <<- approxfun(seq(0, 85, by = 2.5), opt_parameters[grep("Y_tri",
        names(opt_parameters))], rule = 2)
    dt <- 1
    times <- seq(0, 85, by = dt)

    eventdat <- data.frame(var = c("SITU1", "SITU2", "SITU3", "Invasive"), time = c(rep(ages["Age1"],
        4), rep(ages["Age2"], 4), rep(ages["Age3"], 4)), value = rep(0.25, 12),
        method = rep("mult", 12))

    Ca_diag <- 0
    init <- c(SITU1 = 0, SITU2 = 0, SITU3 = 0, Invasive = 0, SUMCANCER = 0)
    out_appl_nscr <- as.data.frame(ode(y = init, times = times, func = AppliedCervScreen,
        parms = opt_parameters))
    out_appl_scr <- as.data.frame(ode(y = init, times = times, func = AppliedCervScreen,
        parms = opt_parameters, events = list(data = eventdat)))
    if (make_plots) {
        layout(matrix(1:2, 1, 2, byrow = TRUE))
        matplot(times, cbind(out_appl_nscr$SITU, out_appl_scr$SITU), type = "l",
            xlab = "Time", ylab = "Ratios", main = "SITU_0 vs SITU_1", lwd = 1,
            lty = 1, bty = "l", col = 2:4)
        legend("topright", c("SITU_0", "SITU_1"), lty = 1, col = 2:3)
        matplot(times, cbind(out_appl_nscr["SUMCANCER"], out_appl_scr["SUMCANCER"]),
            type = "l", xlab = "Time", ylab = "Subjects", main = "Cumulated Cancers",
            lwd = 1, lty = 1, bty = "l", col = 2:3)
        legend("topleft", c("SUMCANCER_0", "SUMCANCER_1"), lty = 1, col = 2:3)
    }

    REDUCTION <- max(out_appl_nscr["SUMCANCER"], na.rm = T) - max(out_appl_scr["SUMCANCER"],
        na.rm = T)
    return(REDUCTION)
}
# Choosing the optimised parameters from previously
opt_parameters <- opt_tri_res_BFGS$par
```

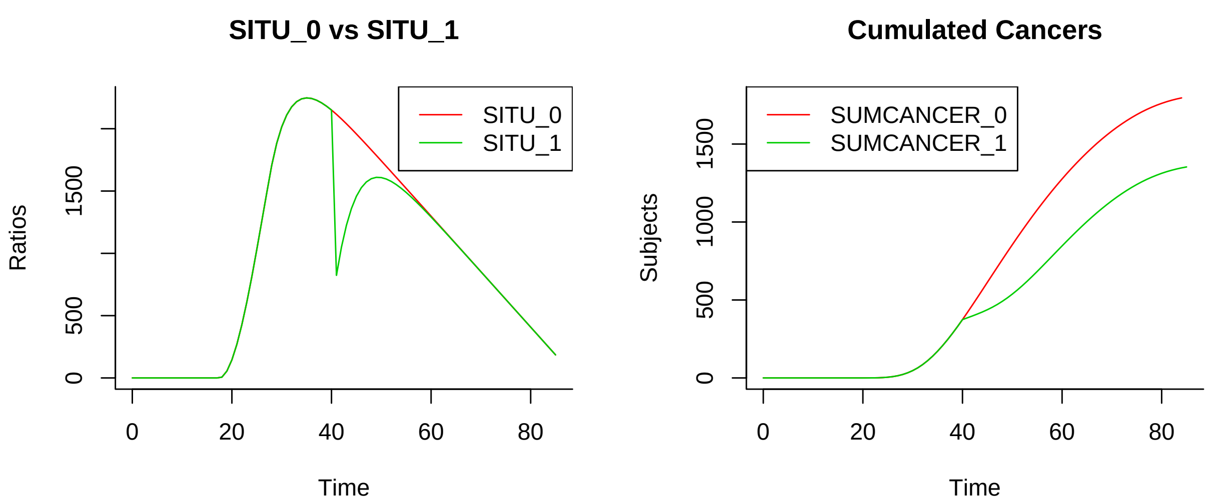
12 Task 1. Comparing results wheen not screening in age-interval

```r
ages <- c(Age1 = 0, Age2 = 0, Age3 = 0)
tmp <- callAppliedCervModels(ages = ages, opt_parameters, make_plots = TRUE)
```



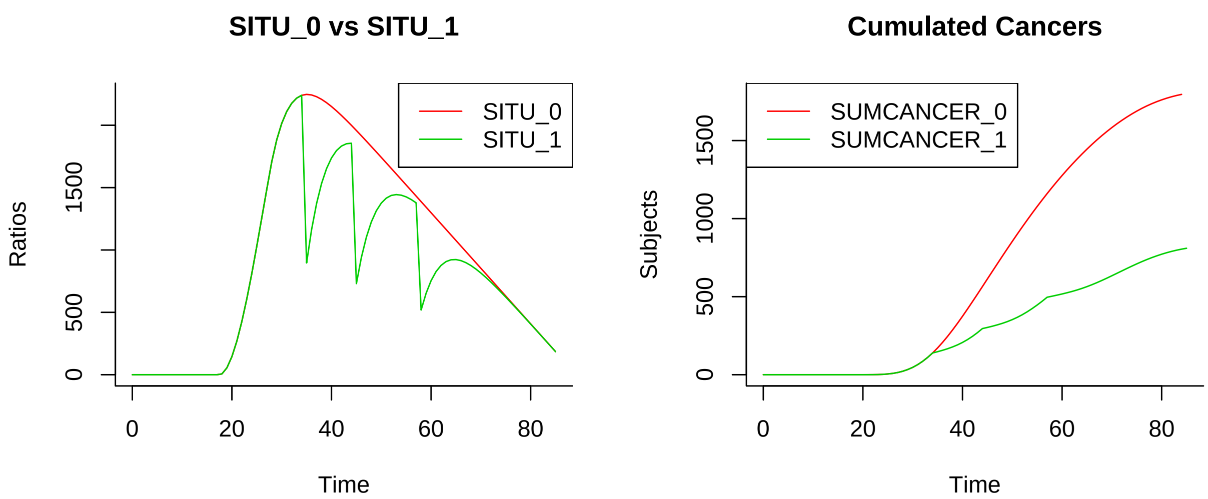
12 Task 2. Comparing results wheen one screening in age-interval

```r
ages <- c(Age1 = 40, Age2 = 0, Age3 = 0)
tmp <- callAppliedCervModels(ages = ages, opt_parameters, make_plots = TRUE)
```



12 Task 3. Comparing results of optimised screening ages

```
lower = 0
upper = 85
ages <- c(Age1 = 35, Age2 = 45, Age3 = 55)
opt_ages_L_BFGS_B <- optim(ages, callAppliedCervModels, method = c("L-BFGS-B"),
    lower = lower, upper = upper, control = list(reltol = 0.001, fnscale = -1),
    opt_parameters = opt_parameters)
tmp <- callAppliedCervModels(ages = round(opt_ages_L_BFGS_B$par), opt_parameters,
    make_plots = TRUE)
```




```
print(opt_ages_L_BFGS_B$par)

##  Age1  Age2  Age3
## 34.14 44.45 56.85
```