Code and figures for 'DTAT should supersede MTD' v1

David C. Norris david@precisionmethods.guru

March 2023

The following code reproduces all analyses and figures presented in article 'Dose Titration Algorithm Tuning (DTAT) should supersede the Maximum Tolerated Dose (MTD) concept in oncology dose-finding trials' (v1) submitted to *F1000Research*. Several analyses and figures excluded from the article are included here; numbering of figures does not necessarily correspond to that in the article.

Pharmacokinetic model

The simulations presented in the article are based on a notional cytotoxic drug, modeled after docetaxel. The pharmacokinetics are supposed to follow a standard 2-compartment pharmacokinetic model with *c*entral and *p*eripheral compartments having volumes $\mathrm{V_c}$ and $\mathrm{V_p}$, respectively, and drug concentrations $C_c(t)$ and $C_p(t)$. Denoting the intercompartmental and peripheral-compartment clearances Q and CL, respectively, and (time-dependent) cumulative dose by $D(t)$, this model is a system of two ordinary differential equations (ODEs):

$$\dot{C_c} = \frac{\dot{D}}{\mathrm{V_c}} - \frac{\mathrm{CL}}{\mathrm{V_c}} C_c - \frac{\mathrm{Q}}{\mathrm{V_c}}(C_c - C_p) \tag{1}$$

$$\dot{C_p} = \frac{\mathrm{Q}}{\mathrm{V_p}}(C_c - C_p). \tag{2}$$

Myelosuppression model

Chemotherapy-induced neutropenia (CIN) is supposed to occur according to the semimechanistic model of Friberg *et al.* (2002). The model may be expressed by the following system of ODEs:

$$\dot{Prol} = k_{tr} \cdot Prol \cdot (1 - E_{drug}) \left(\frac{Circ_0}{Circ} \right)^{\gamma} - k_{tr} \cdot Prol \tag{3}$$

$$\dot{Tx_1} = k_{tr}\,(Prol - Tx_1) \tag{4}$$

$$\dot{Tx_2} = k_{tr}\,(Tx_1 - Tx_2) \tag{5}$$

$$\dot{Tx_3} = k_{tr}\,(Tx_2 - Tx_3) \tag{6}$$

$$\dot{Circ} = k_{tr}\,(Tx_3 - Circ) \tag{7}$$

Simulated population (inter-individual heterogeneity)

A population of 25 individuals is simulated, using population pharmacokinetic parameters reported for docetaxel in Onoue *et al.* (2016) and parameters estimated in Friberg *et al.* (2002) for their myelosuppression model.

```
N <- 25
dtx.mm <- 0.808 # molar mass (mg/µM) of docetaxel
pop <- data.frame(id=1:N
                  # From Friberg et al 2002 (Table 4, row 1), taking sdlog ~= CV
                  ,Circ0=rlnorm(N, meanlog=log(5050), sdlog=0.42) # units=cells/mm^3
                  ,MTT=rlnorm(N, meanlog=log(89.3), sdlog=0.16)  # mean transit time
                  ,gamma=rlnorm(N, meanlog=log(0.163), sdlog=0.039) # feedback factor
                  ,Emax=rlnorm(N, meanlog=log(83.9), sdlog=0.33)
                  ,EC50=rlnorm(N, meanlog=log(7.17*dtx.mm), sdlog=0.50)
                  # PK params from 2-compartment docetaxel model of Onoue et al (2016)
                  ,CL=rlnorm(N, meanlog=log(32.6), sdlog=0.295)
                  ,Q =rlnorm(N, meanlog=log(5.34), sdlog=0.551)
                  ,Vc=rlnorm(N, meanlog=log(5.77), sdlog=0.1) # Onoue gives no CV% for V1
                  ,Vp=rlnorm(N, meanlog=log(11.0), sdlog=0.598) # Called 'V2' in Onoue
)

pop <- upData(pop
              ,kTR = 4/MTT
              ,units = c(Circ0="cells/mm^3"
                         ,MTT="hours"
                         ,kTR="1/hour"
                         ,CL="L/h"
                         ,Q="L/h"
                         ,Vc="L"
                         ,Vp="L"
                         )
              ,print=FALSE
              )

latex(describe(pop), file="")
```

pop 11 Variables 25 Observations

id

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	13	13	8.667	2.2	3.4	7.0	13.0	19.0	22.6	23.8

lowest : 1 2 3 4 5, highest: 21 22 23 24 25

Circ0 [cells/mm^3]

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	5063	4823	2079	3442	3469	3766	4485	5810	7768	8654

lowest : 1563.57 3439.05 3452.3 3493.87 3664.58, highest: 6484.65 7689.94 7819.93 8862.37 9867.58

MTT [hours]

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	88.45	87.52	17.71	70.25	71.33	76.18	84.97	102.10	109.96	110.87

lowest : 67.959 70.2257 70.3491 72.8133 75.1749, highest: 107.017 108.722 110.787 110.889 126.859

γ:

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25
25	0	25	1	0.1616	0.1617	0.00735	0.1516	0.1539	0.1574
.50	.75	.90	.95						
0.1613	0.1661	0.1689	0.1715						

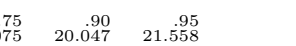
lowest : 0.147769 0.151288 0.152986 0.155289 0.155991, highest: 0.167548 0.168357 0.169222 0.172062 0.173726

Emax

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	90.39	89.18	30.52	52.42	59.63	74.28	89.82	104.67	120.58	129.87

lowest : 39.5723 51.3396 56.7266 63.9952 69.1917, highest: 109.461 112.767 125.794 130.888 162.071

EC50



n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	9.889	9.074	6.893	3.061	3.367	6.120	7.525	13.075	20.047	21.558

lowest : 2.9568 3.00208 3.29845 3.47067 4.77445, highest: 15.055 18.3246 21.1949 21.6482 24.9759

CL [L/h]

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	32.72	33.07	9.214	19.86	19.95	30.34	32.75	37.57	40.88	41.28

lowest : 18.5327 19.8474 19.9259 19.9772 20.9673, highest: 40.0639 40.3992 41.1974 41.2991 52.6234

Q [L/h]

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	6.452	5.824	4.628	2.165	2.673	3.034	5.210	7.784	11.736	15.210

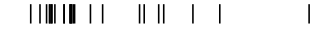
lowest : 1.6063 2.05371 2.61263 2.76334 2.94361, highest: 9.35977 9.83702 13.0015 15.762 18.802

Vc [L]

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	5.803	5.806	0.558	5.078	5.204	5.409	5.788	6.110	6.449	6.503

lowest : 4.83721 5.06056 5.14713 5.28877 5.38205, highest: 6.39435 6.44294 6.45242 6.5153 6.55067

Vp [L]



n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25	.50	.75	.90	.95
25	0	25	1	18.42	15.38	15.87	5.193	6.689	7.654	12.179	28.020	37.721	45.428

lowest : 2.72992 4.88309 6.43276 7.07247 7.21422, highest: 32.4718 33.65 40.4347 46.6759 67.4465

kTR [1/hour]

n	missing	distinct	Info	Mean	pMedian	Gmd	.05	.10	.25
25	0	25	1	0.0465	0.04683	0.008853	0.03608	0.03638	0.03918
.50	.75	.90	.95						
0.04707	0.05251	0.05609	0.05694						

lowest : 0.0315311 0.0360722 0.0361055 0.0367909 0.0373771
highest: 0.0532092 0.054935 0.0568593 0.0569592 0.058859

PK/PD simulation model

To simulate the pharmacokinetics and (myelosuppressive) pharmacodynamics of our notional cytotoxic drug, we define a **pomp** model as follows:

```
pkpd.skel <- "
  double c2p = Q*( Cc - Cp ); // central-to-peripheral flux
  DCc = (dose/duration)*(t < duration ? 1.0 : 0.0)/Vc - (CL/Vc)*Cc - c2p/Vc;
  DCp = c2p/Vp;
  // Myelosuppression model (Emax model, then dynamics per eqs 3-7 from Friberg et al 2002
  double Edrug = Emax*Cc/(Cc + EC50); // classic 'Emax model'
  DProl = (1-Edrug) * Prol * kTR * pow((Circ0 / Circ), gamma)  -  kTR * Prol;
  DTx_1 = kTR * (Prol - Tx_1);
  DTx_2 = kTR * (Tx_1 - Tx_2);
  DTx_3 = kTR * (Tx_2 - Tx_3);
  DCirc = kTR * (Tx_3 - Circ);
"
pkpd.txform <- "
  T_Circ0 = log(Circ0);
  T_kTR = log(kTR);
  T_Emax = log(Emax);
  T_EC50 = log(EC50);
  T_CL = log(CL);
  T_Q = log(Q);
  T_Vc = log(Vc);
  T_Vp = log(Vp);
  T_sigma = log(sigma);
  T_dose = log(dose);
  T_duration = log(duration);
"
pkpd.txback <- "
  Circ0 = exp(T_Circ0);
  kTR = exp(T_kTR);
  Emax = exp(T_Emax);
  EC50 = exp(T_EC50);
  CL = exp(T_CL);
  Q = exp(T_Q);
  Vc = exp(T_Vc);
  Vp = exp(T_Vp);
  sigma = exp(T_sigma);
  dose = exp(T_dose);
  duration = exp(T_duration);
"

Tmax <- 21*24 # solve for full 21 days of 3-week cycle
df <- data.frame(time=c(seq(0.0, 1.95, 0.05) # q3min for 2h,
                        ,seq(2.0, Tmax, 1.0)) # then hourly until Tmax
                 ,y=NA)
pkpd <- pomp(data = df
             , times="time", t0=0
             , skeleton=vectorfield(Csnippet(pkpd.skel))
             , statenames=c("Cc", "Cp", "Prol", "Tx.1", "Tx.2", "Tx.3", "Circ")
             , paramnames=c("Circ0","kTR","gamma","Emax","EC50","CL","Q","Vc","Vp"
                            ,"sigma","dose","duration"
                            ,"Cc.0","Cp.0","Prol.0","Tx.1.0","Tx.2.0","Tx.3.0","Circ.0")
```

```
            , partrans=parameter_trans(
                toEst = Csnippet(pkpd.txform)
               ,fromEst = Csnippet(pkpd.txback)
              )
)
```

In each subject, we now infuse the same initial dose over 1 hour, and integrate the PK and myelosuppression ODEs to obtain time series for plotting:

```
# initial conditions and output times
dose1 <- 100 # mg
Tinfusion <- 1 # 1-hour infusion
allout <- data.frame() # accumulator for nrow(pop) individual ODE solutions
parms <- function(id, dose, duration){
  parms <- unlist(pop[id,c('Circ0','kTR','gamma','Emax','EC50','CL','Q','Vc','Vp')])
  parms['sigma'] <- 0.05
  parms['dose'] <- dose
  parms['duration'] <- duration
  parms['Cc.0'] <- parms$Cp.0 <- 0.0
  parms[c('Prol.0','Tx.1.0','Tx.2.0','Tx.3.0','Circ.0')] <- parms$Circ0
  parms
}
for (id in 1:nrow(pop)) {
  Circ0 <- pop$Circ0[id]
  out <- trajectory(pkpd,
                    params=c(pop[pop$id==id, -which(names(pop) %in% c('id','MTT'))]
                             , sigma=0.05
                             , dose=dose1
                             , duration=Tinfusion
                             , Cc.0 = 0.0
                             , Cp.0 = 0.0
                             , Prol.0 = Circ0
                             , Tx.1.0 = Circ0
                             , Tx.2.0 = Circ0
                             , Tx.3.0 = Circ0
                             , Circ.0 = Circ0)) |> as.data.frame()
  out <- out[,c('time','Cc','Cp','Prol','Tx.1','Tx.2','Tx.3','Circ')]
  out$id <- paste("id",id,sep="")
  allout <- rbind(allout, out)
}

library(data.table)
```

```
##
## Attaching package: 'data.table'
```

```
## The following object is masked from 'package:pomp':
##
##     melt
```

```
allout <- as.data.table(allout)

## When a function y(x) is sampled around a minimum at equispaced (x-dx, x, x+dx)
## with corresponding values (y-dy1, y, y+dy2) such that dy1 < 0 < dy2 (i.e., with
## the middle value y being lowest value), then a quadratic interpolation yields
```

```
## estimated minimum at (x_, y_) given by:
##   x_ = x - dx/2 * (dy1 + dy2)/(dy2 - dy1)
##   y_ = y - 1/8 [(dy1 + dy2)/(dy2 - dy1)]^2.

for(.id in unique(allout$id)) {
  with(subset(allout, id == .id), {
    nadirIdx <- which.min(Circ)
    Dy1Dy2 <- diff(Circ[nadirIdx + (-1:1)])
    SdyDdy <- sum(Dy1Dy2)/diff(Dy1Dy2)
    allout[id == .id, CircMin := Circ[nadirIdx] - (1/8)*SdyDdy^2]
    allout[id == .id, tNadir := time[nadirIdx] - 0.5*diff(time[nadirIdx+(0:1)])*SdyDdy]
  })
}

allout <- upData(allout
      ,id = ordered(id, levels=paste("id",1:N,sep="")) # Note that we may
      ,units=c(Prol="cells/mm^3" # treat the non-circulating compartments
              ,Tx.1="cells/mm^3" # on a 'circulating-equivalent basis';
              ,Tx.2="cells/mm^3" # thus we attach 'cells/mm^3' units to
              ,Tx.3="cells/mm^3" # these quantities as well.
              ,Circ="cells/mm^3"
              ,Cc="ng/mL"
              ,Cp="ng/mL"
              ,time="hours")
      ,print=FALSE
       )
```

```
cout <- gather(allout, key="Series", value="Concentration"
               , Cc, Cp
               , factor_key = TRUE)

label(cout$Concentration) <- "Drug Concentration"

xYplot(Concentration ~ time | id, group=Series
       , data=cout, subset=time<6
       , layout=c(5,5)
       , type='l', as.table=TRUE
       , label.curves=FALSE
       , par.settings=list(superpose.line=list(lwd=2,col=brewer.pal(4,"PRGn")[c(1,4)]))
       , scales=list(y=list(log=TRUE, lim=c(10^-3,10^1)))
       , auto.key=list(points=FALSE, lines=TRUE, columns=2))
```

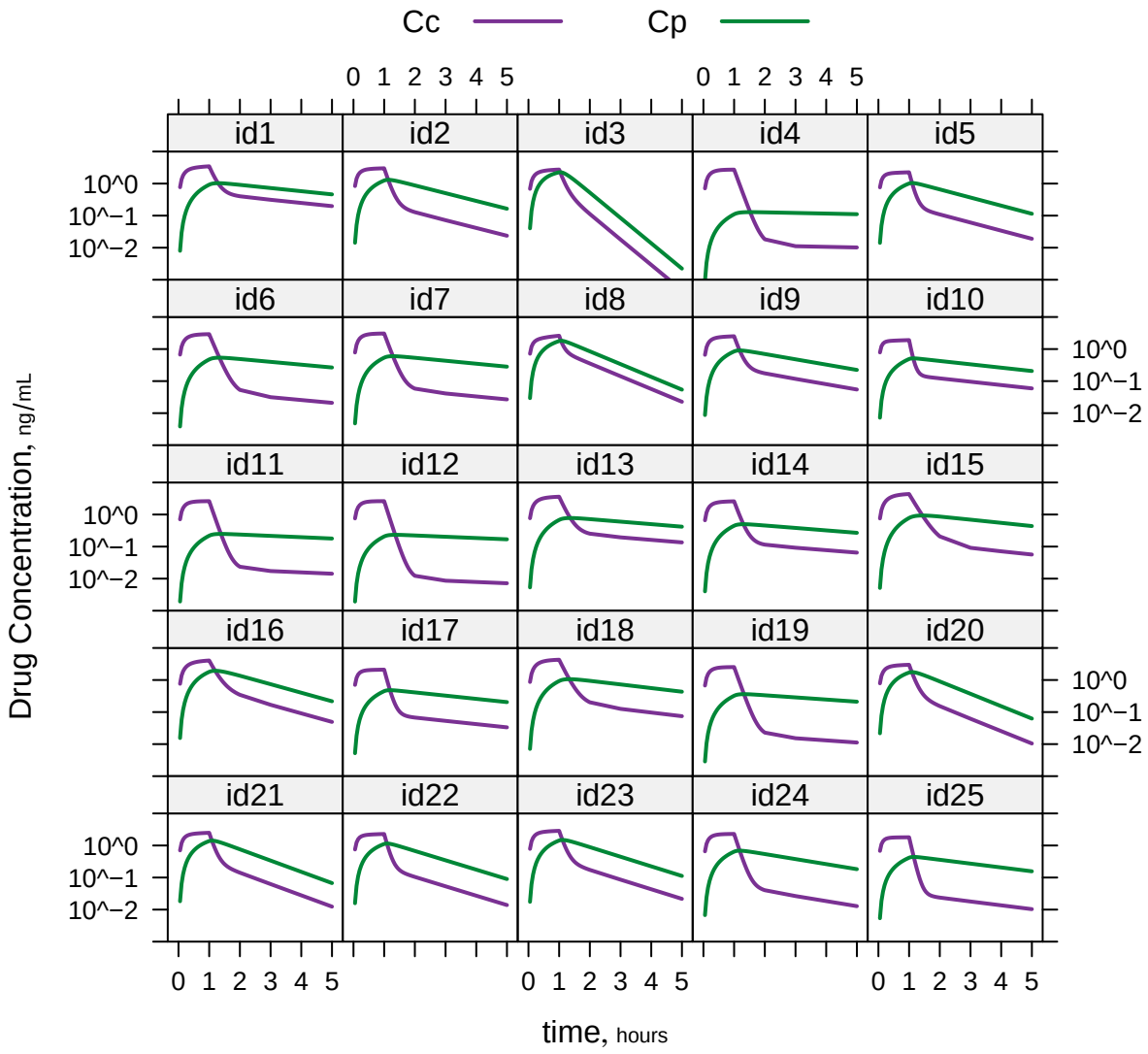


Figure 1: Two-compartment pharmacokinetics of the drug.

```
mout <- gather(allout, key="Series", value="ANC"
               , Prol, Tx.1, Tx.2, Tx.3, Circ
               , factor_key = TRUE)

mout <- upData(mout
               , time = time/24
               , units = c(time="days")
               , print = FALSE)

xYplot(ANC ~ time | id, group=Series
       , data=mout
       , layout=c(5,5)
       , type='l', as.table=TRUE
       , label.curves=FALSE
       , par.settings=list(superpose.line=list(lwd=2,col=brewer.pal(11,"RdYlBu")[c(1,3,4,8,10)]))
       , scales=list(y=list(log=TRUE, lim=c(100,15000), at=c(200, 500, 1000, 2000, 5000, 10000)))
       , auto.key=list(points=FALSE, lines=TRUE, columns=5))
```

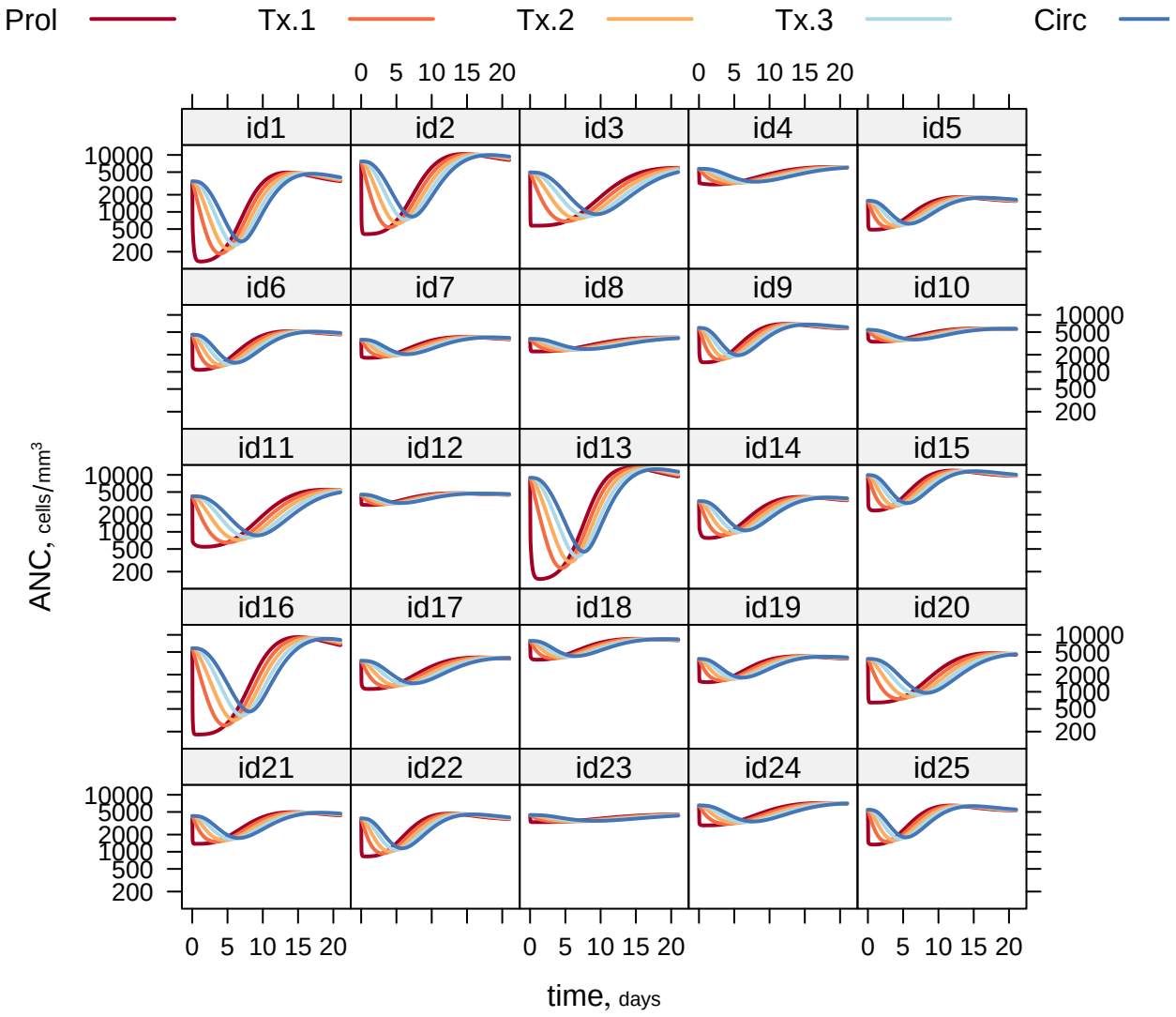


Figure 2: Myelosuppression in the 5-compartment model of Friberg *et al.*, with an infused dose of 100 mg. **Prol:** proliferative compartment; **Tx.n:** transit compartments; **Circ:** circulating cells.

Adaptive dosing based on a simple Newton-Raphson heuristic

The `doChemo` function defined below implements multiple, 3-week courses of chemotherapy with optional adaptive dosing based on the heuristic of Newton-Raphson root-finding. When `adapt.dosing==FALSE`, the infusion doses are stepped from 50 up to 500 mg, in increments that are evenly spaced on the scale of $\sqrt[4]{dose}$.

```
scaled <- function(dose, a=4.0) dose^(1/a)
dose.scale <- list(lim=scaled(c(40,550))
                   , at=scaled(c(50, 100, 250, 500))
                   , lab=c('50','100','250','500')) # TODO: Redo limits & labels when right dosing is f
doChemo <- function(draw.days=NULL, Tcyc=3*7*24, adapt.dosing=c('Newton'), omega=0.75) {
  # Find the ANC nadirs of all 20 IDs, checking ANCs on (integer-vector) draw.days
  # We will accumulate data about each course of treatment into this data frame.
  # TODO: Consider doing away entirely with columns Cc..Circ, since these inits are available in 'out'
  hourly <- which(abs(time(pkpd) - round(time(pkpd))) < .Machine$double.eps^0.5)
  anc.ts <- data.frame() # This will be used to collect an hourly 'Circ' time series
  course <- expand.grid(cycle=1:10, id=1:nrow(pop), Cc=0.0, Cp=0.0
                        , Prol=NA, Tx.1=NA, Tx.2=NA, Tx.3=NA, Circ=NA
                        , dose=NA, ANC.nadir=NA, time.nadir=NA, scaled.dose=NA
                        , ANC.nadir.est=NA, time.nadir.est=NA)
  for (day in draw.days) {
    newcolumn <- paste("ANC", day, sep="_d")
    course[,newcolumn] <- NA
    units(course[,newcolumn]) <- "cells/mm^3"
    label(course[,newcolumn]) <- paste("Day-",day," ANC", sep="")
  }
  course$dose <- seq(scaled, from=50, to=500, length.out=max(course$cycle), digits=0)[course$cycle]
  statevector <- c('Cc','Cp','Prol','Tx.1','Tx.2','Tx.3','Circ')
  course[,statevector[-(1:2)]] <- pop$Circ0[course$id] # Prol(0)=Tx.1(0)=Tx.2(0)=Tx.3(0)=Circ(0):=Circ0
  for (id in 1:nrow(pop)) { # outer loop over IDs permits state cycling
    params <- unlist(pop[id,c('Circ0','gamma','Emax','EC50','CL','Q','Vc','Vp','kTR')])
    params['sigma'] <- 0.05
    params['duration'] <- Tinfusion
    params[c('Cc.0','Cp.0')] <- 0.0
    params[c('Prol.0','Tx.1.0','Tx.2.0','Tx.3.0','Circ.0')] <- params['Circ0']
    for (cycle in 1:max(course$cycle)) {
      idx <- which(course$cycle==cycle & course$id==id)
      if (cycle>1) {
        lag_1 <- which(course$cycle==(cycle-1) & course$id==id)
        if (adapt.dosing=='Newton') { # Override preconfigured dose
          params[paste(statevector,'0',sep='.')] <- traj[nrow(traj),statevector] # recycle end-state
          if (cycle==2) {
            slope <- -2.0
          } else { # cycle >= 3 so we also have lag -2 to look back at
            lag_2 <- which(course$cycle==(cycle-2) & course$id==id)
            dY <- log(course[lag_1,'ANC.nadir'] / course[lag_2,'ANC.nadir'])
            dX <- scaled(course[lag_1,'dose']) - scaled(course[lag_2,'dose'])
            slope <- dY/dX
            if (slope >= 0) slope <- NA # to avoid instability from dY/dX>=0 due to hysteresis
          }
          delta.scaleddose <- ifelse(is.na(slope), 0, log(500 / course[lag_1,'ANC.nadir']) / slope)
          # For safety's sake, we (asymmetrically) apply relaxation factor 'omega' to any proposed dose
          delta.safer <- ifelse(delta.scaleddose > 0
                                , omega*delta.scaleddose
                                , delta.scaleddose)
```

```
        new.scaleddose <- scaled(course[lag_1,'dose']) + delta.safer
        course$dose[idx] <- uniroot(function(y) scaled(y)-new.scaleddose, c(0,100000))$root
      }
    }
    params['dose'] <- course$dose[idx]
    traj <- trajectory(pkpd, params=params) |> as.data.frame()
    to.add <- data.frame(id=rep(id,length(hourly))
                         , time=traj$time[hourly]+(cycle-1)*Tmax
                         , ANC=traj$Circ[hourly])
    anc.ts <- rbind(anc.ts, to.add)
    course[idx,c('ANC.nadir','time.nadir')] <- traj[which.min(traj$Circ),c('Circ','time')]
    course[idx,statevector] <- traj[which.max(traj$time),statevector]
    for (day in draw.days) {
      day.idx <- which(traj$time==day*24)
      course[idx,paste("ANC", day, sep="_d")] <- traj[day.idx,'Circ']
    }
    if (length(draw.days)) {
      # TODO: Here, estimate nadir level and timing based on fitted spline
      ys <- course[idx,paste("ANC", draw.days, sep="_d")]
      fit <- spline(draw.days, log(ys), n=200)
      course[idx,'ANC.nadir.est'] <- exp(min(fit$y))
      course[idx,'time.nadir.est'] <- fit$x[which.min(fit$y)]
    }
  }
}

course <- upData(course[order(course$cycle),]
                 , id = ordered(paste("id",id,sep="")
                       ,levels=paste("id",1:N,sep=""))
                 , time.nadir = time.nadir/24
                 , scaled.dose = scaled(dose)
                 , labels=c(ANC.nadir="ANC nadir"
                            ,time.nadir="Time of ANC nadir"
                            ,dose="Drug Dose"
                            ,scaled.dose="Drug Dose (rescaled)")
                 , units=c(ANC.nadir="cells/mm^3"
                           ,time.nadir="days"
                           ,dose="mg"
                           ,scaled.dose="mg")
                 , print=FALSE
)

anc.ts <- upData(anc.ts
                 , id = ordered(paste("id",id,sep="")
                       ,levels=paste("id",1:N,sep=""))
                 , time = time/(24*7)
                 , labels=c(ANC="ANC")
                 , units=c(ANC="cells/mm^3"
                           ,time="weeks")
                 , print=FALSE
)

list(course=course, anc.ts=anc.ts)
```

```
} # end of function
```

Linearize dynamics

We now demonstrate graphically an approximate linearization of ANC nadir level and timing, under logarithmic transformation of ANC and fourth-root transformation of dose.

```
chemo <- doChemo(draw.days=c(5,6,7,8,11), adapt.dosing=FALSE)
course <- chemo$course
anc.ts <- chemo$anc.ts
```

```
xYplot(ANC.nadir ~ scaled.dose | id, data=course, as.table=TRUE
      , layout=c(5,5)
      , scales=list(y=list(log=TRUE, lim=c(100,10000), at=c(200, 500, 1000, 2000, 5000)),
                    x=dose.scale)
)
```

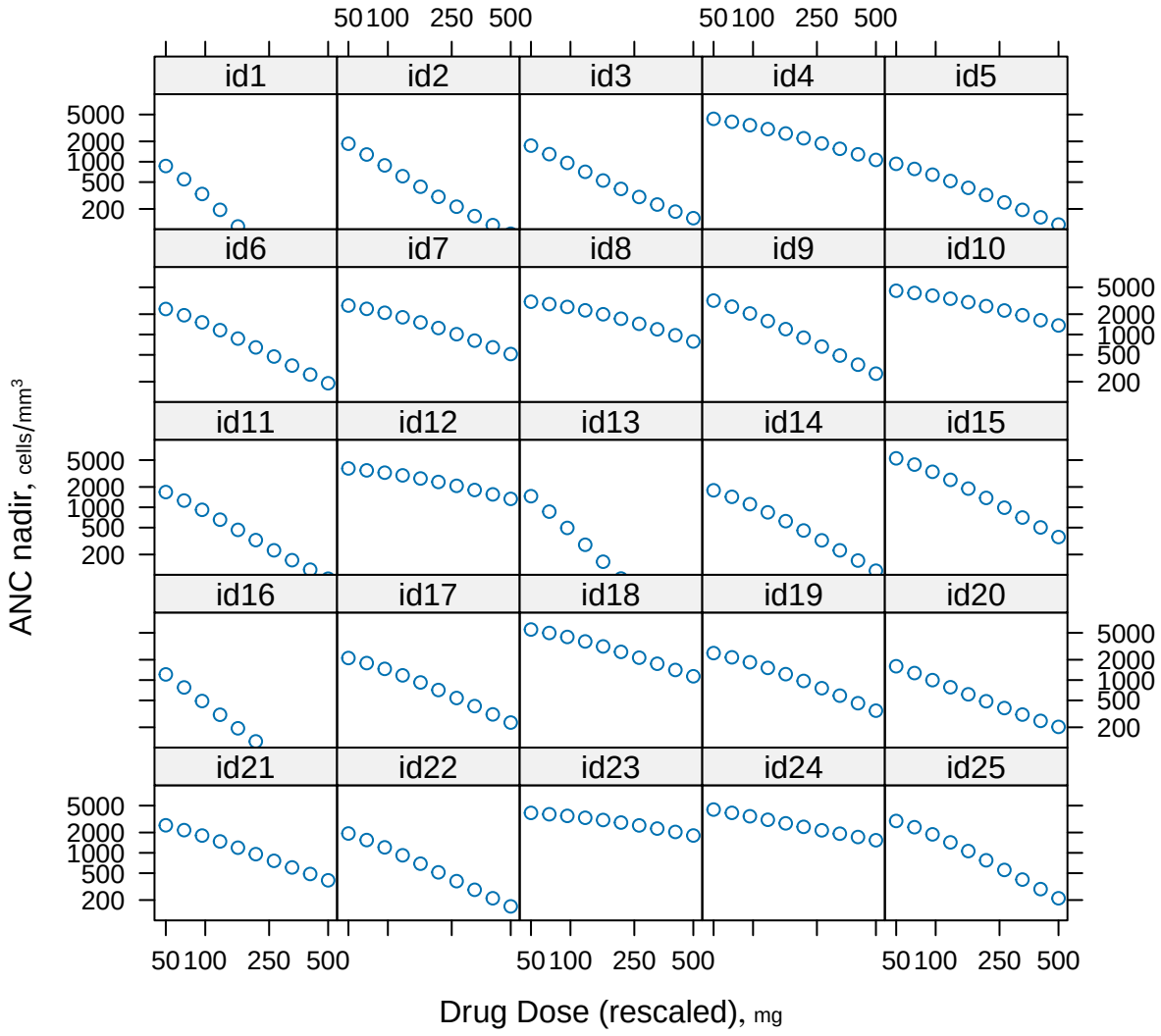


Figure 3: ANC nadir vs cytotoxic dose. Note the dose axis is scaled to achieve near-linearity.

```
slopeForID <- function(x){
  fit <- lm(log(ANC.nadir) ~ scaled.dose, data=subset(course, id==x))
  slope <- fit$coefficients['scaled.dose']
  unname(slope)
}

slope <- sapply(levels(course$id), slopeForID)
densityplot(~slope, plot.points="rug")
```

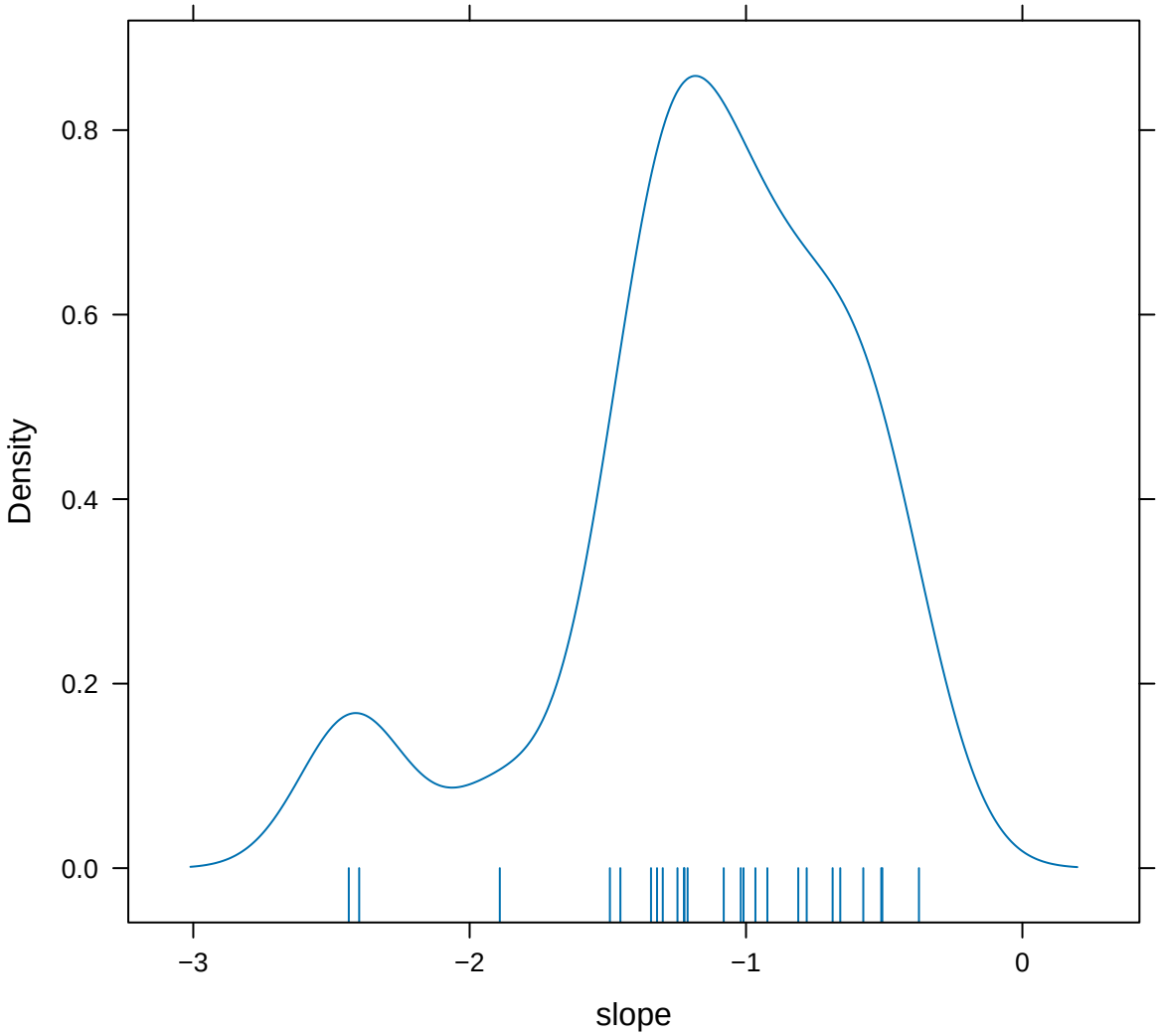


Figure 4: Slopes of $\log(ANC_{nadir})$ vs $\sqrt[4]{dose}$ for 25 simulated study subjects.

```
xYplot(time.nadir ~ scaled.dose | id, data=course, as.table=TRUE
      , layout=c(5,5)
      , scales=list(x=dose.scale)
)
```

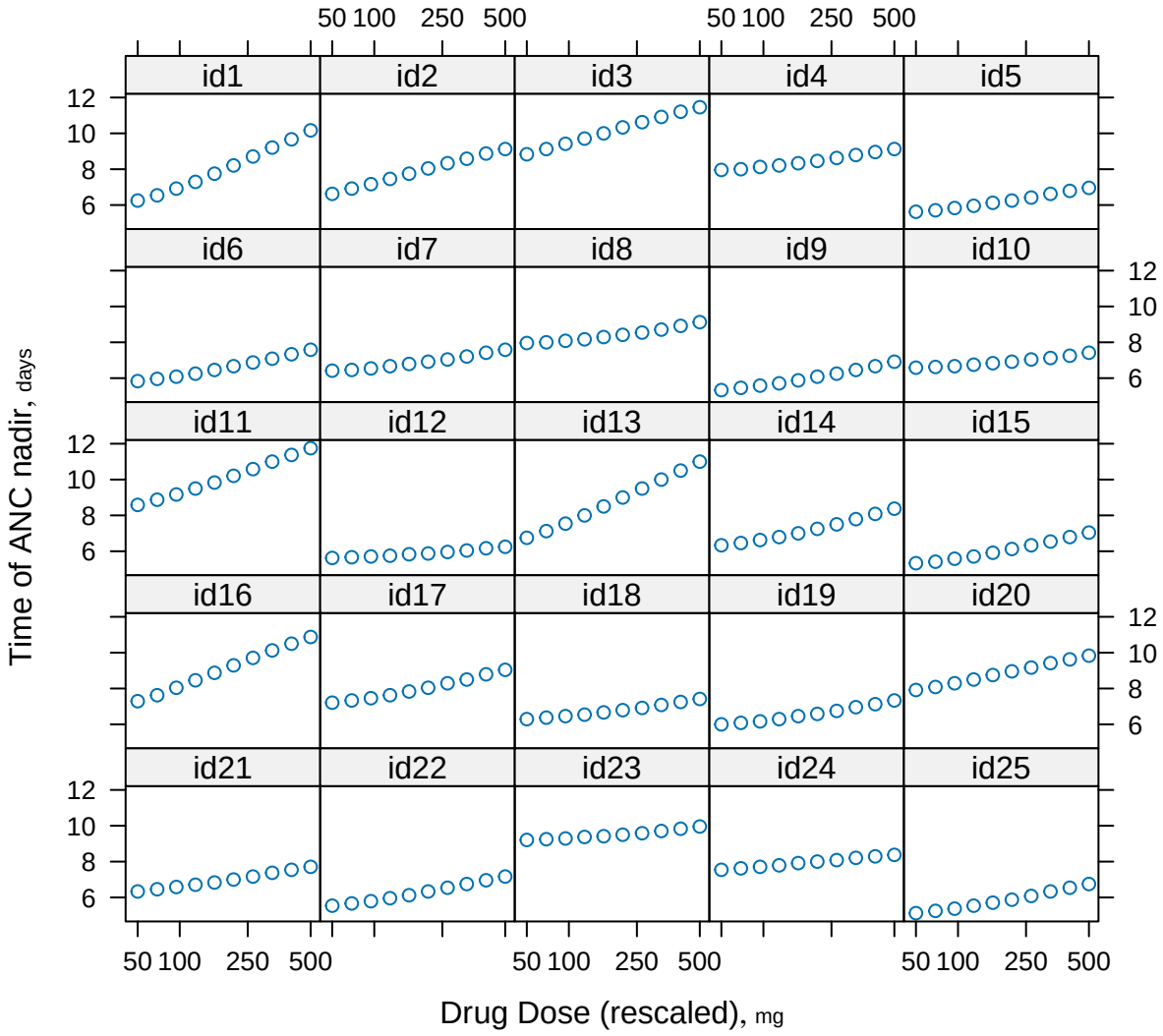


Figure 5: Timing of ANC nadir vs initial cytotoxic dose. Note the dose axis is scaled to achieve near-linearity.

```
##, subset=(dose %in% c(500,1000,1500,2500,4000))
xYplot(ANC.nadir ~ time.nadir | id, group=dose, data=course
      , as.table=TRUE
      , layout=c(5,5)
      , auto.key=list(columns=5, title="Dose (mg)", lines=FALSE)
      , par.settings=list(superpose.symbol=list(col=gray.colors(10, start=0.7, end=0.0)))
      , scales=list(y=list(log=TRUE, lim=c(10,6000), equispaced.log=FALSE))
       )
```

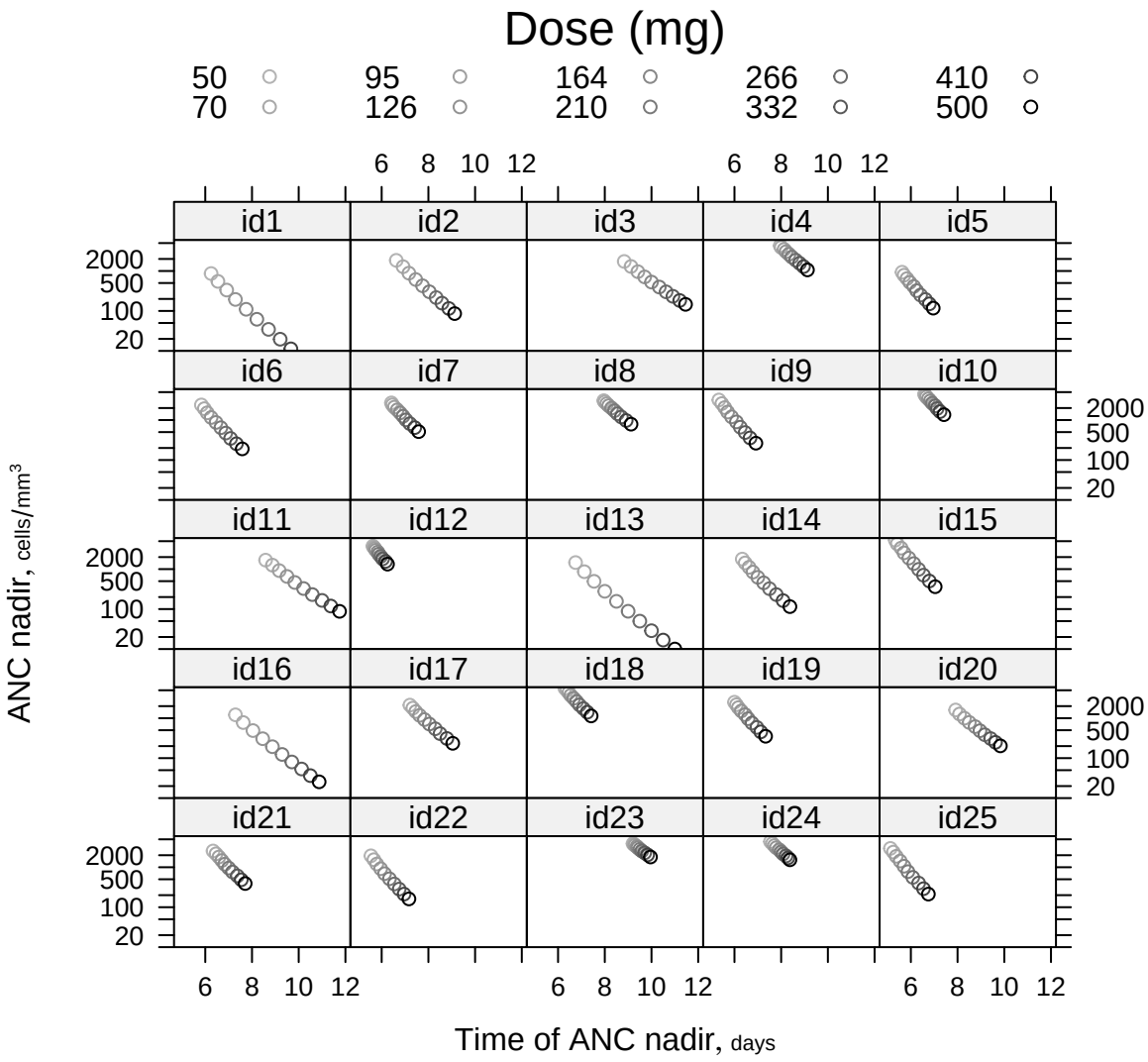


Figure 6: Timing and level of ANC nadir vs initial cytotoxic dose. The doses are equally spaced on a power-law scale that yields a roughly linear parametrization of the nadir coordinate paths.

Would day-7 ANC suffice as a proxy for nadir?

Evidently not; see Figure 7. Thus, CIN monitoring constitutes a nontrivial aspect of any practical implementation of DTAT, and requires explicit modeling in follow-on work. No doubt, the implementation must take the form of an adaptive process designed to balance (a) the burden of multiple blood draws against (b) the need for early warning of an impending severely neutropenic nadir that would indicate prophylactic administration of colony-stimulating factor.

```
xYplot(ANC.nadir ~ ANC_d7, data=course, group=id, type='l', as.table=TRUE
       , aspect=1
       , label.curves=list(method="offset")
       , scales=list(x=list(log=TRUE, lim=c(40, 6000), equispaced.log=FALSE),
                     y=list(log=TRUE, lim=c(40, 6000), equispaced.log=FALSE))
       , par.settings=list(superpose.line=list(col="black"))
)
```

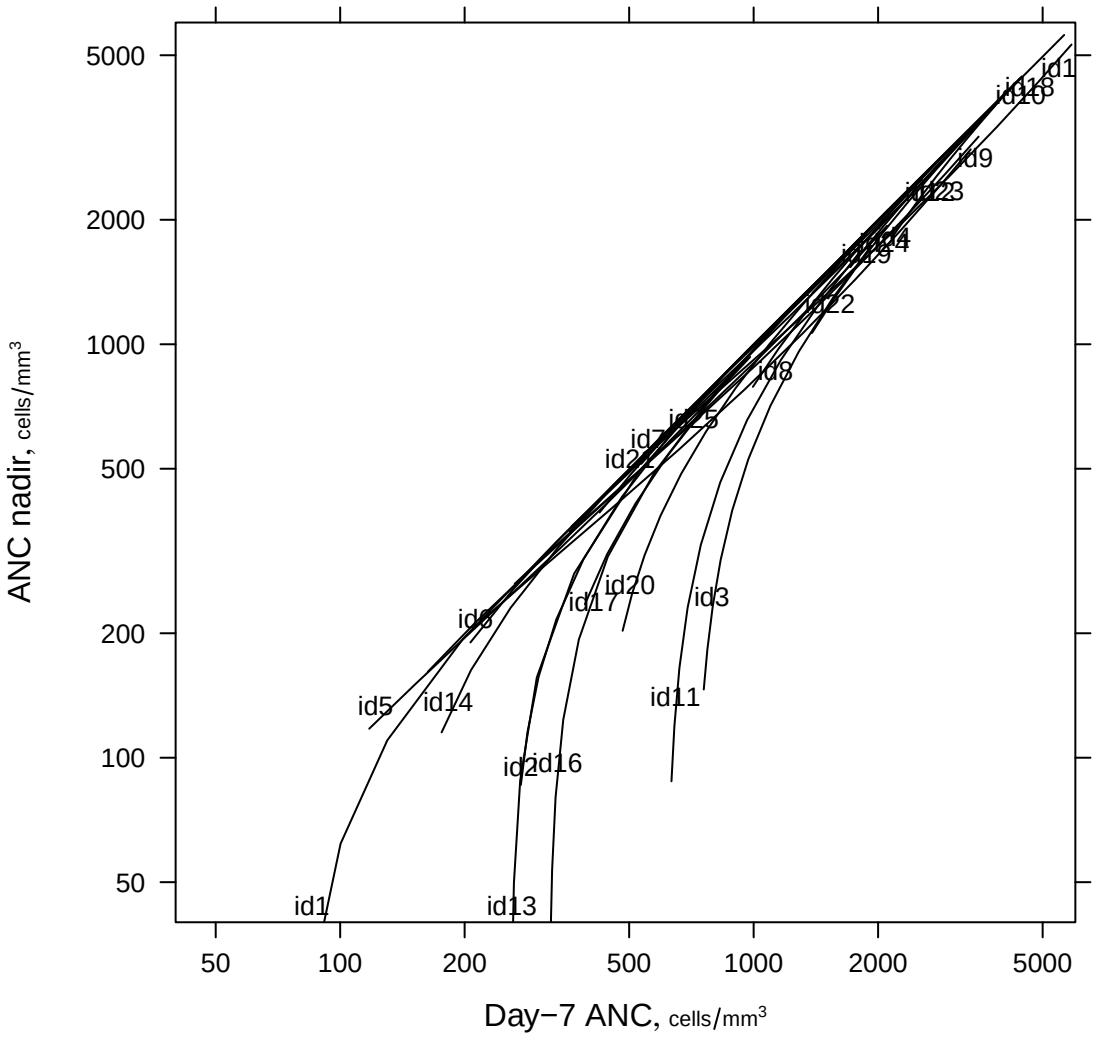


Figure 7: Day-7 ANC does not serve as suitable proxy for ANC nadir across the whole modeled population. Note that for some individuals (e.g., id3, id11, id20), the day-7 ANC may dangerously underestimate the actual nadir.

Simulated dose titration in 25 study subjects

```
chemo <- doChemo(adapt.dosing='Newton')
newton <- chemo$course
new.ts <- chemo$anc.ts
anc.tics <- c(200,500,1500,4000,10000)
right <- xYplot(ANC ~ time | id, data=new.ts
                , as.table=TRUE, type="l"
                , layout=c(5,5)
                , scales=list(y=list(log=TRUE, lim=c(100,1.5e4)
                                     , at=anc.tics
                                     , lab=as.character(anc.tics)),
                              x=list(at=seq(0,30,3)))
                )
newton <- upData(newton
                 , time = 3*(cycle-1)
                 , labels = c(time="Time")
                 , units = c(time="weeks")
                 , print = FALSE)
dose.tics <- c(50, 200, 600, 1500, 3000)
left <- xYplot(scaled.dose ~ time | id, data=newton
               , as.table=TRUE, type='p', pch='+', cex=1.5
               , layout=c(5,5)
               , scales=list(y=list(lim=scaled(c(30,3200))
                                    , at=scaled(dose.tics)
                                    , lab=as.character(dose.tics)),
                             x=list(lim=c(-1,31)
                                    , at=seq(0,30,3)
                                    , lab=c('0','','6','','12','','18','','24','','30')))
               )
update(doubleYScale(left, right, add.ylab2=TRUE)
       , par.settings = simpleTheme(col=brewer.pal(4,"PRGn")[c(4,1)])
       )
```

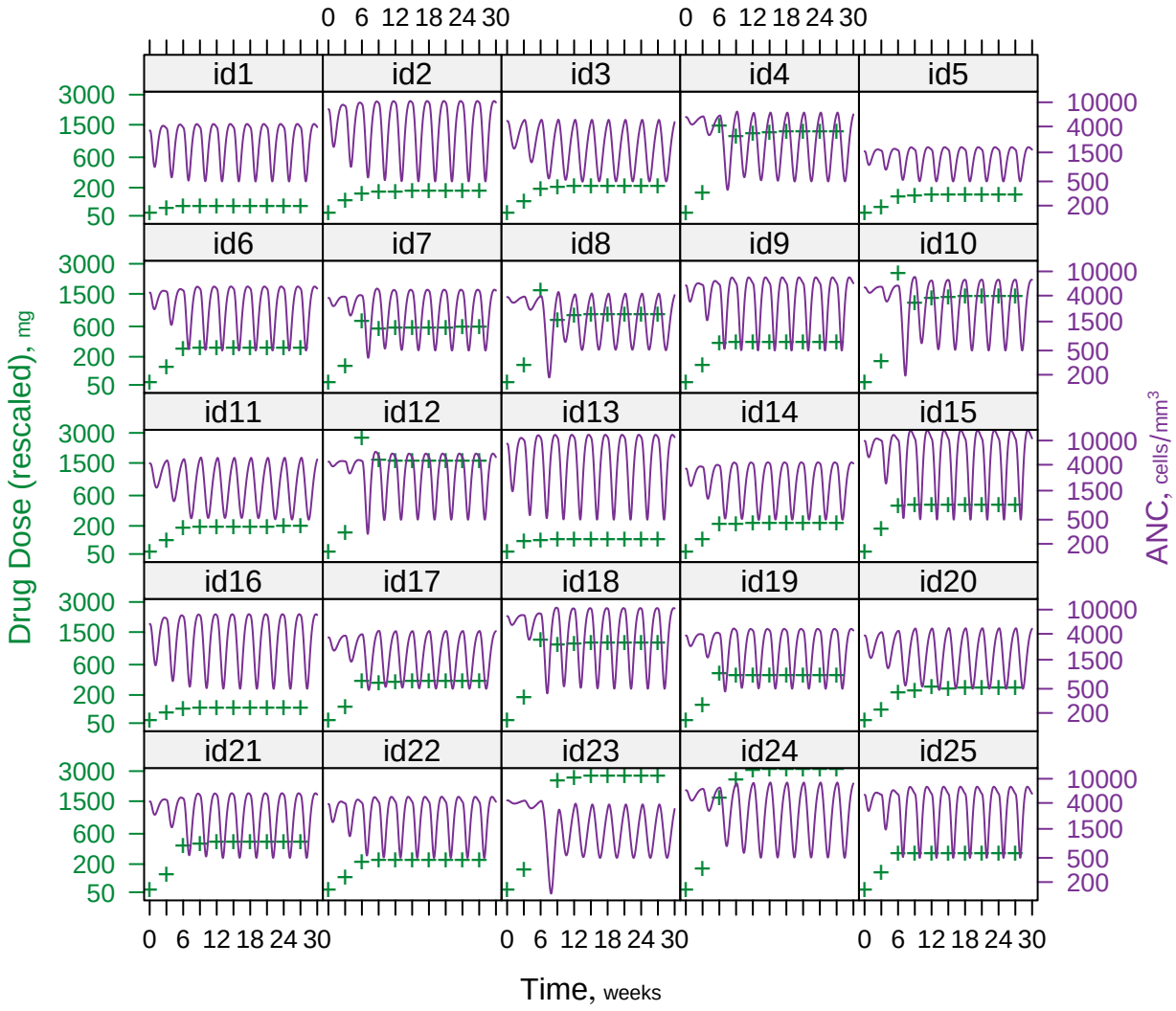


Figure 8: Dose titration by Newton's method, to a goal ANC nadir of 500 cells/mm^3.

SessionInfo

This document was produced using:

```
sessionInfo()
```

```
## R version 4.5.1 (2025-06-13)
## Platform: x86_64-pc-linux-gnu
## Running under: PureOS 11 (Crimson)
##
## Matrix products: default
## BLAS/LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.21.so;  LAPACK version 3.1
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=C               LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: America/New_York
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] data.table_1.17.8   pomp_6.3            RColorBrewer_1.1-3  latticeExtra_0.6-30
##  [5] tidyr_1.3.1         rms_8.0-0           Hmisc_5.2-3         knitr_1.50
##  [9] DTAT_0.3-8          survival_3.8-3      widgetframe_0.3.1   htmlwidgets_1.6.4
## [13] r2d3_0.2.6          lattice_0.22-7
##
## loaded via a namespace (and not attached):
##  [1] gtable_0.3.6       xfun_0.52          bslib_0.9.0        ggplot2_3.5.2
##  [5] vctrs_0.6.5        tools_4.5.1        generics_0.1.4     sandwich_3.1-1
##  [9] tibble_3.3.0       cluster_2.1.8.1    pkgconfig_2.0.3    Matrix_1.7-3
## [13] checkmate_2.3.2    lifecycle_1.0.4    deldir_2.0-4       compiler_4.5.1
## [17] farver_2.1.2       stringr_1.5.1      MatrixModels_0.5-4 tinytex_0.57
## [21] codetools_0.2-20   SparseM_1.84-2     httpuv_1.6.16      quantreg_6.1
## [25] htmltools_0.5.8.1  sass_0.4.10        yaml_2.3.10        htmlTable_2.4.3
## [29] Formula_1.2-5      pillar_1.11.0      later_1.4.2        jquerylib_0.1.4
## [33] MASS_7.3-65        cachem_1.1.0       multcomp_1.4-28    rpart_4.1.24
## [37] nlme_3.1-168       mime_0.13          km.ci_0.5-6        tidyselect_1.2.1
## [41] digest_0.6.37      polspline_1.1.25   mvtnorm_1.3-3      stringi_1.8.7
## [45] dplyr_1.1.4        purrr_1.1.0        splines_4.5.1      fastmap_1.2.0
## [49] grid_4.5.1         colorspace_2.1-1   cli_3.6.5          magrittr_2.0.3
## [53] base64enc_0.1-3    TH.data_1.1-3      withr_3.0.2        foreign_0.8-90
## [57] scales_1.4.0       promises_1.3.3     backports_1.5.0    rmarkdown_2.29
## [61] jpeg_0.1-11        interp_1.1-6       nnet_7.3-20        gridExtra_2.3
## [65] deSolve_1.40       png_0.1-8          zoo_1.8-14         coda_0.19-4.1
## [69] shiny_1.11.1       evaluate_1.0.4     rlang_1.1.6        Rcpp_1.1.0
## [73] xtable_1.8-4       glue_1.8.0         rstudioapi_0.17.1  jsonlite_2.0.0
## [77] R6_2.6.1
```