Estimating lymphocyte turnover by deuterium labeling

This practical describes a few approaches to estimate the life spans of lymphocytes from deuterium data. This text is based upon the Appendix of the paper of Vrisekoop *et al.* [9] and the review of De Boer and Perelson [3]. You will learn

- how to perform non-linear parameter estimation by fitting fairly complicated data
- that T cell memory is maintained by short lived cells, and that naive T cells are extremely long-lived,
- that the selection of the most appropriate model is of utmost importance because different models give different estimates,
- and hence that quantification of cellular population dynamics is far from trivial.

Volunteers and patients can drink heavy water (^2H_2O) for weeks. During the labeling period deuterium is build into the newly synthesized DNA strands of cells that divide. Labeled DNA strands will subsequently disappear by cell death. We here fit data of five volunteers whom have been drinking a glass of 4% deuterated water for nine weeks. Because the fraction of heavy water in body water is similar to that in urine, one typically measures the urine enrichment, and models this by the differential equation $\mathrm{d}u/\mathrm{d}t = f - \delta u$, where f represents the fraction of ^2H_2O in the drinking water, and δ is the turnover rate of body water per day. During the de-labeling phase, starting at $\tau = 63$ days, we simply set $f = 0$. The initial condition, $u(0) = \beta$, is set by the initial deuterium boost that is given at day zero to rapidly approach the steady state $u(\infty) = f/\delta$. This simple ODE has the piece wise solution

$$u(t) = \begin{cases} f(1 - \mathrm{e}^{-\delta t}) + \beta \mathrm{e}^{-\delta t} \ , & \text{if } t \leq \tau \ , \\ \left[f(1 - \mathrm{e}^{-\delta \tau}) + \beta \mathrm{e}^{-\delta \tau} \right] \mathrm{e}^{-\delta (t-\tau)}, & \text{otherwise} \ , \end{cases} \tag{1}$$

which is provided by the function `urine()` in the accompanying R documents.

A general model for a population of leucocytes that is produced at rate s (for source), is maintaining itself by cell division at rate p (for proliferation), and with cells that die at rate d is

$$\frac{\mathrm{d}T}{\mathrm{d}t} = s + (p - d)T \quad \text{with steady state} \quad \bar{T} = \frac{s}{d - p} \ , \tag{2}$$

where T stands for total cell numbers in some volume. Without loss of generality this steady state can be scaled to $\bar{T} = 1$, giving that $s = d - p$. At steady state conditions the model therefore has two parameters. From this "master" equation one derives a model for the fraction of labeled strands, $L(t)$,

$$\frac{\mathrm{d}L}{\mathrm{d}t} = \gamma(t)s + \gamma(t)pL + \gamma(t)pU - dL \ , \tag{3}$$

where $\gamma(t)$ is the rate at which newly synthesized DNA in a dividing cell is incorporating deuterium (which should be proportional to the deuterium concentration in the serum $u(t)$, see below). Note that in cells dividing during the labeling phase both unlabeled and labeled DNA strands are copied into labeled DNA strands, and that labeled strands disappear by cell death when $\gamma(t) = 0$. Because $L(t) + U(t) = 1$ this simplifies into

$$\frac{\mathrm{d}L}{\mathrm{d}t} = \gamma(t)(s + p) - dL \quad \text{or, at steady state,} \quad \frac{\mathrm{d}L}{\mathrm{d}t} = \gamma(t)d - dL \ . \tag{4}$$

The latter is provided by the function `model()` in the accompanying R documents. Fortuitously, this model can be used both for populations maintained entirely by the source, i.e., $\mathrm{d}T/\mathrm{d}t = s - dT$ as its steady state $\bar{T} = 1 = s/d$ implies $s = d$, and for populations maintained entirely by self renewal, i.e., $\mathrm{d}T/\mathrm{d}t = (p - d)T$ as its steady state implies $p = d$ [3].

Since deuterium can be incorporated at several positions of the adenosine moiety that is analyzed by the gas chromatography mass spectrometry (GC-MS), the enrichment of this moiety is expected to exceed that of the body water. One therefore expects an amplification factor, c, that is estimated from

the estimated plateau enrichment of a fast population of cells (here granulocytes). This is achieved by fitting Eq. (1) to the urine data, substituting $\gamma(t) = cu(t)$ into Eq. (4), and fitting that to the granulocyte data of Vrisekoop *et al.* [9]. This is achieved in the script `granul.R`.

A popular model to describe deuterium labeling data was proposed by Asquith and her co-workers [1]. They argue that in kinetically heterogeneous populations the label should accumulate faster in sub-populations that are turning over more rapidly. The initial up-slope during the labeling phase should reflect the average turnover rate, p, and the labeled cells should die at a rate d (typically called d^*) that is faster than p. Hence they write Eq. (4) as

$$\frac{\mathrm{d}L}{\mathrm{d}t} = \gamma(t)p - dL \ , \tag{5}$$

which is provided by the function `modelp()` in the `tcells.R` script. According to this model the population will not approach the same enrichment of the fast population because this model has an asymptote $L(\infty) = \gamma(\infty)p/d < \gamma(\infty)$ (as $p < d$). To repair this the death rate of labeled cells should decrease over time and ultimately approach the average turnover rate p [3].

A more mechanistic model to describe kinetically heterogeneous populations was proposed by Ganusov *et al.* [4], and basically generalizes Eq. (4) into the sum of several sub-populations with different turnover rates d_i, and where $L = \sum_i^n \alpha_i L_i$. Here α_i is the fractional size of sub-population i, and L_i obeys the very similar

$$\frac{\mathrm{d}L_i}{\mathrm{d}t} = \gamma(t)d_i - d_i L_i \ . \tag{6}$$

The basic procedure to fit this model to deuterium data is to start with one sub-population, $n = 1$, and increase the number of compartments, n, until the best fit is obtained [4, 10]. (What is best should be defined statistically by the F-test, as the quality of the fit will always improve when the number of parameters is increased). A two compartment version of this model is provided by the function `model2()` in the `tcells.R` script. The average turnover rate of this model is defined as $\bar{d} = \sum_i^n \alpha_i d_i$, and $\bar{d}$ is fortunately much more identifiable than the estimates of the individual α_i and d_i parameters [2, 4, 10]. The average life span of the cells in the entire population is defined as $1/\bar{d}$. Finally, note that this is a nested set of models as the one and two parameter model can be defined by Eq. (6) by setting $\alpha_1 = d_2 = 0$ or by setting $d_2 = 0$.

Running with `grind.R`.

Today you will work with an R-script called `grind.R` that is a wrapper around the R-packages `deSolve`, `FME` and `rootSolve` developed by Karline Soetaert and colleagues [5–8]. These packages allow one to solve differential equations, find their steady state, and perform nonlinear parameter estimation. Today you only need three of `grind.R`'s easy-to-use functions:

- `run()` integrates a model numerically and provides a time plot or a trajectory in the phase plane,
- `fit()` fits a model to data by estimating its parameters, and depicts the result in a timeplot.
- `timePlot()` plots a data frame.

The `run()` function calls `ode()` from the `deSolve` library, and the `fit()` function calls `modFit` from the `FME` library. For instance, typing `?ode`, provides help on the `ode()` function. The full manual of `grind.R` is available on the website `http://tbb.bio.uu.nl/rdb/practicals/grindR/`.

We will work in the RStudio environment. If you work on your own laptop you will need to install the Soetaert libraries by using `Install Packages` in the `Tools` menu of RStudio. (Experts may prefer to type `install.packages(c("deSolve","FME", "rootSolve")` in the R-console). All documents can be found on the webpage `http://tbb.bio.uu.nl/rdb/practicals/deuterium`. Download the R-scripts `grind.R`, `granul.R`, and `tcells.R`, and store them in a local directory, and open them via the `File` menu. Next download the data file `Vrisekoop_pnas08.csv`, and store it in the same folder. Set the working directory of RStudio to the folder where your R-codes and data are stored

(`Set working directory` in the `Session` menu of RStudio). Files will then be opened and saved in that directory.

First "Source" the `grind.R` file (button in right hand top corner) to define the functions, then "Run" the first part of the `granul.R` script to define the models and a few convenient functions (button in right hand top corner). It is useful to first highlight everything up to the line `Here the session starts`, and "Run" that highlighted section, and subsequently run the script line-by-line. Do read this first part to know what we predefined for you (this R-code is fairly readable). In the R-console one can type and call any function in R. (Use "Run" or "Control Enter" to execute lines from the code panel into the console). Subsequently proceed through the `granul.R` file by selecting line and `Run` that using `Control Enter`.

Estimating the amplification factor

The main mission of this practical is to fit the data of the Vrisekoop *et al.* [9] data with the various models explained above to study how the estimated turnover rates, or life spans, of the various populations of T lymphocytes depend on the choice of the model. Fit the memory T cell data to the models with one (`model()`), two (`modelp()`) and three parameters (`model2()`), and study which model best describes the data, and how the estimated death rates depends on the choice of the model. This can be done for the $CD4^+$ and $CD8^+$ memory T cell data of all five volunteers (i.e., $2 \times 3 \times 5$ fits). When you finish early, try the same for the naive T cell data (using `model()` and `modelp()` only). Make sure you understand what happens when you execute a line of the R-scripts, and provide answers to the following questions:

1. What is the turnover rate of body water per day in volunteer one? How confident are you of this estimate? Note that you can add `bootstrap=100` to the call of `myfit` to obtain confidence intervals (but this takes long). What is his/her expected life span of neutrophils? Does this make sense? How confident are you of the estimate of the amplification factor c? Do you think the neutrophil population is fast enough to accurately estimate the amplification factor c?
2. To obtain the most reliable estimate of the amplification factor, we estimate one c and one granulocyte death rate, d, from all five patients, while giving each of them their own kinetics for the heavy water. Is this general c very different from the individual amplification factors? If you are on a fast computer, try to get a confidence interval for c, because this is the parameter required in the next session.

Estimating T cell turnover.

Copy your estimate of c into the R-script `tcells.R`. Again `Run` the first part of the script until the line `Here the session starts`:

1. Fit the memory T cell data of every volunteer to each of the three models.
2. What is the expected lifespan of memory T cells? How do these estimates depend on the choice of the model?
3. Which model provides the best description of the data? Use the `ftest()` function.
4. How confident are you of these estimates (perform bootstrapping)?
5. How can we have life long T cell memory to many pathogens?
6. Which model seems most appropriate for describing the $CD4^+$ naive T cells? What is your best estimate for the expected life span on $CD4^+$ naive T cells?

June 14, 2017, Rob J. de Boer

References

[1] **Asquith, B., Debacq, C., Macallan, D. C., Willems, L., and Bangham, C. R.**, 2002. Lymphocyte kinetics: the interpretation of labelling data. Trends Immunol. **23**:596–601.

[2] **De Boer, R. J., Mohri, H., Ho, D. D., and Perelson, A. S.**, 2003. Turnover rates of B cells, T cells, and NK cells in simian immunodeficiency virus-infected and uninfected rhesus macaques. J. Immunol.

170:2479–2487.

[3] **De Boer, R. J. and Perelson, A. S.**, 2013. Quantifying T lymphocyte turnover. J. theor. Biol. **327**:45–87.

[4] **Ganusov, V. V., Borghans, J. A., and De Boer, R. J.**, 2010. Explicit kinetic heterogeneity: mathematical models for interpretation of deuterium labeling of heterogeneous cell populations. PLoS Comput. Biol. **6**:e1000666.

[5] **Soetaert, K.**, 2009. rootSolve: Nonlinear root finding, equilibrium and steady-state analysis of ordinary differential equations. R package 1.6.

[6] **Soetaert, K. and Herman, P. M.**, 2009. A Practical Guide to Ecological Modelling. Using R as a Simulation Platform. Springer. ISBN 978-1-4020-8623-6.

[7] **Soetaert, K. and Petzoldt, T.**, 2010. Inverse modelling, sensitivity and monte carlo analysis in R using package FME. Journal of Statistical Software **33**:1–28.

[8] **Soetaert, K., Petzoldt, T., and Setzer, R. W.**, 2010. Solving differential equations in R: Package desolve. Journal of Statistical Software **33**:1–25.

[9] **Vrisekoop, N., Den Braber, I., De Boer, A. B., Ruiter, A. F., Ackermans, M. T., Van der Crabben, S. N., Schrijver, E. H., Spierenburg, G., Sauerwein, H. P., Hazenberg, M. D., De Boer, R. J., Miedema, F., Borghans, J. A., and Tesselaar, K.**, 2008. Sparse production but preferential incorporation of recently produced naive T cells in the human peripheral pool. Proc. Natl. Acad. Sci. U.S.A. **105**:6115–6120.

[10] **Westera, L., Drylewicz, J., Den Braber, I., Mugwagwa, T., Van der Maas, I., Kwast, L., Volman, T., Van de Weg-Schrijver, E. H., Bartha, I., Spierenburg, G., Gaiser, K., Ackermans, M. T., Asquith, B., De Boer, R. J., Tesselaar, K., and Borghans, J. A.**, 2013. Closing the gap between T-cell life span estimates from stable isotope-labeling studies in mice and humans. Blood **122**:2205–2212.

granul.R

```
urine <- function(t, state, parms) {  # state is a dummy variable
  with(as.list(parms), {
    U <- ifelse(t < tau,
      f*(1-exp(-delta*t))+beta*exp(-delta*t),
      (f*(1-exp(-delta*tau))+beta*exp(-delta*tau))*exp(-delta*(t-tau)))
    return(U)
  })
}

model <- function (t, state, parms) {
  with(as.list(c(state,parms)), {
    dL <- d*c*urine(t,NULL,parms) - d*L
    return(list(dL))
  })
}

asinsqrt <- function(x) return(asin(sqrt(x)))

myfit <- function(data, model, who, ...)
  return(fit(datas=data,odes=model,who=who,fun=asinsqrt,lower=0,...))

select <- function(i) {
  patient <- subset(rawdata,rawdata$patient == i)
  patient <- patient[(patient$time < 200),]   # Use early time points only
  return(na.omit(patient[2:4]))
}

opar <- par();par(mar=c(2.6,2.6,1.6,0.2),mgp=c(1.5,0.5,0))

# Here the session starts:
rawdata <- read.csv("Vrisekoop_pnas08.csv")
for (i in seq(4,8))                          # Set negative values to zero
  rawdata[,i] <- sapply(rawdata[,i],max,0)
data <- lapply(seq(5),select)                # Urine & granulos from 5 patients

# Initial parameters
p <- c(beta=0.01,f=0.018,delta=0.05,tau=63,c=2,d=0.1)
tweak <- "nsol<-cbind(nsol,urine(times,NULL,parms));names(nsol)[2:3]<-c(\"G\",\"U\")"
```

```
s <- c(L=0)                    # Initial state
timePlot(data[[1]])            # Plot the first data set
run(200,tweak=tweak,add=T)     # add the solution of the model

# Fit the urine and granulo data together: Estimate one c and d
who <- c("c","d","beta","f","delta")              # select free parameters
fit1 <- myfit(data[[1]],model,who,tweak=tweak)    # Fit one data set
differ <- c("beta","f","delta")                   # subset that differs between data sets
par(mfrow=c(3,2))
fall <- myfit(data,model,who,tweak=tweak,differ=differ,main=seq(5),legend=F)
par(mfrow=c(1,1))
pall <- fall$par
print("Shared C & Granulo turnover:")
pall[1:2]
```

tcells.R

```
urine <- function(t, state, parms) {  # state is a dummy variable
  with(as.list(parms), {
    U <- ifelse(t < tau,
      f*(1-exp(-delta*t))+beta*exp(-delta*t),
      (f*(1-exp(-delta*tau))+beta*exp(-delta*tau))*exp(-delta*(t-tau)))
    return(U)
  })
}

model <- function (time, state, parms) {
  with(as.list(c(state,parms)), {
    dL <- d*c*urine(time,NULL,parms) - d*L
    return(list(dL))
  })
}

modelp <- function (time, state, parms) {
  with(as.list(c(state,parms)), {
    dL <- p*c*urine(time,NULL,parms) - d*L
    return(list(dL))
  })
}

model2 <- function (t, state, parms) {
  with(as.list(c(state,parms)), {
    dL1 <- d1*c*urine(t,NULL,parms) - d1*L1
    dL2 <- d2*c*urine(t,NULL,parms) - d2*L2
    return(list(c(dL1,dL2)))
  })
}

asinsqrt <- function(x) return(asin(sqrt(x)))

ftest=function(ssr1,p1,ssr2,p2,n) {
  if (p2 > p1) {
    df1 <- p2-p1
    df2 <- n-p2
    f <- ((ssr1-ssr2)/df1)/(ssr2/df2)
    cat("F[",df1,",",df2,"] =", f, ": P = ",1-pf(f,df1,df2),"\n")
  } else
    ftest(ssr2,p2,ssr1,p1,n)
}

myfit <- function(data, model, who, ...)
  return(fit(datas=data,odes=model,who=who,fun=asinsqrt,lower=0,upper=1,arrest="tau"
      ,...))

opar <- par();par(mar=c(2.6,2.6,1.6,0.2),mgp=c(1.5,0.5,0))
```

```
# Here the session starts:
rawdata <- read.csv("Vrisekoop_pnas08.csv")
for (i in seq(4,8)) rawdata[,i] <- sapply(rawdata[,i],max,0)

id <- 3
patient <- subset(rawdata,rawdata$patient == id)
udata <- na.omit(patient[2:3])                          # Urine data
tdata <- na.omit(cbind(patient[2],patient["M4"]))       # Select Memory CD4
names(udata)[2] <- "L"; names(tdata)[2] <- "L"          # Rename columns 2

par(mfrow=c(2,2))
s <- c(L=0)
p <- c(beta=0.01,f=0.012,delta=0.05,tau=63)
p["beta"] <- udata[1,2]                 # Get guess for beta from data
who <- c("beta","f","delta")
fu <- myfit(udata,urine,who,solution=T,main="urine")

p <- c(fu$par,tau=63,c=4.37,d=0.05)
s <- c(L=0)                             # Initial state
who <- c("d")                           # Select free parameters
f0 <- myfit(tdata,model,who,main=id)
print(c(f0$par["d"],LifeSpan=1/as.numeric(f0$par["d"])))

p <- c(fu$par,tau=63,c=4.37,d=0.05,p=0.01)
who <- c("d","p")
fp <- myfit(tdata,modelp,who,main=id)
print(c(fp$par["p"],LifeSpan=1/as.numeric(fp$par["p"])))

ftest(f0$ssr,length(f0$par),fp$ssr,length(fp$par),nrow(tdata))

s <- c(L1=0,L2=0)
p <- c(fu$par,tau=63,c=4.37,alpha=0.5,d1=0.05,d2=0.001)
who <- c("alpha","d1","d2")
tweak <- "nsol<-cbind(nsol,parms[\"alpha\"]*nsol[,2]+(1-parms[\"alpha\"])*nsol[,3]);
    names(nsol)[4]<-\"L\""
f2 <- myfit(tdata,model2,who,tweak=tweak,main=id)
print(c("Average death rate:",with(as.list(f2$par),alpha*d1+(1-alpha)*d2)))

ftest(f0$ssr,length(f0$par),f2$ssr,length(f2$par),nrow(tdata))
par(mfrow=c(1,1))
```