User's Guide for the PKPDmodels Package

Douglas Bates
Dept. of Statistics, University of Wisconsin - Madison, U.S.A.
and
France Mentré
UMR738, INSERM, University Paris Diderot

January 26, 2012

Abstract

The **PKPDmodels** package provides a function, `PKPDmodel`, that is used to generate model functions for 1, 2, or 3 compartment pharmacokinetic models with linear elimination kinetics. The form of administration can be intravenous bolus, infusion or oral administration with first-order absorption. The model can be for single dose, multiple doses or steady-state conditions.

1 Introduction

A pharmacokinetic model describes the concentration of an analyte in the central compartment of a compartment model as a function of time and other covariates such as dose. The `LinPKmodel` function generates model functions for pharmacokinetic models with linear elimination kinetics. The mode of administration can be intravenous bolus, continuous infusion or oral with first-order absorption kinetics. The dosage can be a single dose, multiple doses or the steady-state condition after multiple doses.

The pharmacokinetic parameters can be expressed as an elimination rate constant, k (units of 1/time), the clearance rate, Cl (units of volume per unit time) or volume of distribution, V (units of volume). Only two of these three

parameters are specified because of the relationship

$$k = \frac{Cl}{V}$$

Some types of models include other parameters such as the absorption rate constant, k_a, for oral doses with first-order absorption.

By default the function that is returned is a byte-compiled function that evaluates both the expected values of the concentration in the central compartment and the gradient with respect to the parameters. There is an option to create a function that will evaluate the expected value of the response and the gradient and the Hessian. Although a default set of parameters is provided, these can be modified by a set of parameter transformations. For example, the default parameterization for a one-compartment model uses the elimination rate constant, k, and the clearance, Cl but the model can be expressed in terms of the volume of distribution, V, and the clearance, Cl, by including the transformation `k ~ V/Cl`. For population pharmacokinetics it is often helpful to transform from parameters such as Cl and V to their logarithms.

2 A simple example

A PK model with linear elimination for a single-dose bolus injection can be expressed in terms of the elimination rate constant, k, and the volume of distribution, V, as

$$C(t) = \frac{D \exp(-k\,t)}{V}$$

where $C(t)$ is the concentration in the central compartment at time t (since administration of the dose) and D is the dose. The corresponding formula in R is

```
> PK1expr("bolus", "sd")

~dose * exp(-k * t)/V
<environment: 0x01dc0e54>
```

If we wish to transform to another set of parameters, we specify the transformations as a list of formulas. For example, to express the model in terms of the volume of distribution, V, and the clearance, Cl,

```
> PK1expr("bolus", "sd", list(k ~ Cl/V))
```

```
~dose * exp(-(Cl/V) * t)/V
<environment: 0x0272c6bc>
```

or, to express the model in terms of $\log(V)$ and $\log(Cl)$,

```
> PK1expr("bolus", "sd", list(k ~ Cl/V, Cl ~ exp(lCl), V ~ exp(lV)))

~dose * exp(-(exp(lCl)/exp(lV)) * t)/exp(lV)
<environment: 0x01e8e96c>
```

Note that a substitution formula has the parameter name on the left and the expression to be substituted for that parameter on the right. When there are multiple substitution formulas they are evaluated left to right.

The function `PK1expr` exists solely so that the user can verify that the formula being used is what they expect. In typical usage we create the model function directly.

```
> (bolus1cptSdVk <- PK1cmpt("bolus", "sd"))

function (dose, t, k, V)
{
    .expr3 <- exp(-k * t)
    .expr4 <- dose * .expr3
    .value <- .expr4/V
    .grad <- array(0, c(length(.value), 2L), list(NULL, c("k",
        "V")))
    .grad[, "k"] <- -(dose * (.expr3 * t)/V)
    .grad[, "V"] <- -(.expr4/V^2)
    attr(.value, "gradient") <- .grad
    .value
}
<bytecode: 0x025ba02c>
```

When this function is evaluated at given values of `dose`, `time` and the pharmacokinetic parameters, it returns the function evaluations with an attribute called `"gradient"`.

```
> tvals <- 0:10
> bolus1cptSdVk(dose=1, t=tvals, k=0.3, V=1)

 [1] 1.000000 0.740818 0.548812 0.406570 0.301194 0.223130 0.165299
 [8] 0.122456 0.090718 0.067206 0.049787
attr(,"gradient")
             k         V
 [1,]  0.00000 -1.000000
 [2,] -0.74082 -0.740818
```

```
 [3,] -1.09762 -0.548812
 [4,] -1.21971 -0.406570
 [5,] -1.20478 -0.301194
 [6,] -1.11565 -0.223130
 [7,] -0.99179 -0.165299
 [8,] -0.85719 -0.122456
 [9,] -0.72574 -0.090718
[10,] -0.60485 -0.067206
[11,] -0.49787 -0.049787
```

When fitting pharmacokinetic models, especially population pharmacokinetic models, it is helpful to have this analytic gradient evaluation rather than calculating numerical gradients. Because common sub-expressions in the gradient columns and the model function are evaluated once only and because the resulting function consists of rather simple byte-compiled expressions the evaluation of the model function and gradient is essentially as fast as the evaluation of the model function itself.

The evaluation of common sub-expressions is more obvious when several parameter transformations are applied

```
> (bolus1cptSdlV1Cl <-
+  PK1cmpt("bolus", "sd", list(k ~ Cl/V, Cl ~ exp(lCl), V ~ exp(lV))))
function (dose, t, lCl, lV)
{
    .expr1 <- exp(lCl)
    .expr2 <- exp(lV)
    .expr3 <- .expr1/.expr2
    .expr6 <- exp(-.expr3 * t)
    .expr7 <- dose * .expr6
    .expr15 <- .expr2^2
    .value <- .expr7/.expr2
    .grad <- array(0, c(length(.value), 2L), list(NULL, c("lCl",
        "lV")))
    .grad[, "lCl"] <- -(dose * (.expr6 * (.expr3 * t))/.expr2)
    .grad[, "lV"] <- dose * (.expr6 * (.expr1 * .expr2/.expr15 *
        t))/.expr2 - .expr7 * .expr2/.expr15
    attr(.value, "gradient") <- .grad
    .value
}
<bytecode: 0x01e22834>
```

where we can see that the clearance, $Cl = \exp(lCl)$ (`.expr1`), the volume of distribution, $V = \exp(lV)$ (`.expr2`), and the elimination rate constant, $k = Cl/V$ (`.expr3`) are each evaluated once only.

Expressing the parameters in this formulation we obtain the same function evaluation as before

```
> bolus1cptSdlVlCl(dose=1, t=tvals, lCl=log(0.3), lV=0)
```

```
 [1] 1.000000 0.740818 0.548812 0.406570 0.301194 0.223130 0.165299
 [8] 0.122456 0.090718 0.067206 0.049787
attr(,"gradient")
           lCl        lV
 [1,]  0.00000 -1.000000
 [2,] -0.22225 -0.518573
 [3,] -0.32929 -0.219525
 [4,] -0.36591 -0.040657
 [5,] -0.36143  0.060239
 [6,] -0.33470  0.111565
 [7,] -0.29754  0.132239
 [8,] -0.25716  0.134702
 [9,] -0.21772  0.127005
[10,] -0.18145  0.114249
[11,] -0.14936  0.099574
```

Naturally the gradient has changed because we are calculating the derivatives with respect to a different set of parameters.

3 Single oral dose

In phase 1 and phase 2 clinical trials population pharmacokinetics are often assessed by sampling at many closely-spaced time points in a small number of subjects after a single oral dose. The `Theoph` data set in the **datasets** package comes from such a study of the drug theophylline which is used in the treatment of asthma.

```
'data.frame':        132 obs. of  5 variables:
 $ Subject: Ord.factor w/ 12 levels "6"<"7"<"8"<"11"<..: 11 11 11 1..
 $ Wt     : num  79.6 79.6 79.6 79.6 79.6 79.6 79.6 79.6 79.6 79.6 ..
 $ Dose   : num  4.02 4.02 4.02 4.02 4.02 4.02 4.02 4.02 4.02 4.02 ..
 $ Time   : num  0 0.25 0.57 1.12 2.02 ...
 $ conc   : num  0.74 2.84 6.57 10.5 9.66 8.58 8.36 7.47 6.89 5.94 ..
```

These data are shown in Figure 1.

A model function for a single oral dose with first-order absorption using the parameters $\log(k_a)$, $\log(V)$ and $\log(Cl)$ is

```
> (oral1cptSdlkalVlCl <-
+  PK1cmpt("oral", "sd", list(ka ~ exp(lka), k ~ exp(lCl)/V, V ~ exp(lV))))
```

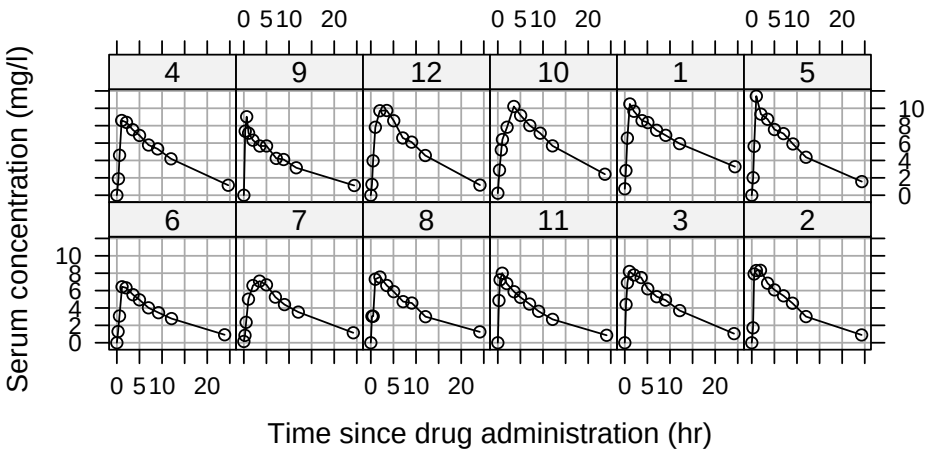

Figure 1: Concentration of theophylline versus time for 12 subjects following a single oral dose of the drug. The panels have been ordered (left to right starting at the bottom row) by increasing maximum concentration.

```
function (dose, t, lV, lka, lCl)
{
    .expr1 <- exp(lV)
    .expr2 <- dose/.expr1
    .expr3 <- exp(lka)
    .expr4 <- exp(lCl)
    .expr5 <- .expr4/.expr1
    .expr6 <- .expr3 - .expr5
    .expr7 <- .expr3/.expr6
    .expr8 <- .expr2 * .expr7
    .expr11 <- exp(-.expr5 * t)
    .expr14 <- exp(-.expr3 * t)
    .expr15 <- .expr11 - .expr14
    .expr18 <- .expr1^2
    .expr19 <- .expr4 * .expr1/.expr18
    .expr24 <- .expr6^2
    .value <- .expr8 * .expr15
    .grad <- array(0, c(length(.value), 3L), list(NULL, c("lV",
        "lka", "lCl")))
    .grad[, "lV"] <- .expr8 * (.expr11 * (.expr19 * t)) - (.expr2 *
        (.expr3 * .expr19/.expr24) + dose * .expr1/.expr18 *
        .expr7) * .expr15
    .grad[, "lka"] <- .expr2 * (.expr7 - .expr3 * .expr3/.expr24) *
```

```
        .expr15 + .expr8 * (.expr14 * (.expr3 * t))
    .grad[, "lCl"] <- .expr2 * (.expr3 * .expr5/.expr24) * .expr15 -
        .expr8 * (.expr11 * (.expr5 * t))
    attr(.value, "gradient") <- .grad
    .value
}
<bytecode: 0x02dea830>
```

corresponding to the formula

```
> PK1expr("oral", "sd", list(ka ~ exp(lka), k ~ exp(lCl)/V, V ~ exp(lV)))

~(dose/exp(lV)) * (exp(lka)/(exp(lka) - exp(lCl)/exp(lV))) *
    (exp(-(exp(lCl)/exp(lV)) * t) - exp(-exp(lka) * t))
<environment: 0x02e24dc0>
```

Initial values for the parameters are taken to be $\log(V)_0 = -1$, $\log(k_a)_0 = 0.5$ and $\log(Cl)_0 = -4$ producing a fitted model

```
> summary(fm1 <- nls(conc ~ oral1cptSdlkalVlCl(Dose, Time, lV, lka, lCl),
+                    Theoph, start=c(lV=-1, lka=0.5, lCl=-4),
+                    subset=Subject==1), corr=TRUE)

Formula: conc ~ oral1cptSdlkalVlCl(Dose, Time, lV, lka, lCl)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
lV   -0.9962     0.0602  -16.54  1.8e-07
lka   0.5752     0.1728    3.33     0.01
lCl  -3.9159     0.1273  -30.77  1.4e-09

Residual standard error: 0.732 on 8 degrees of freedom

Correlation of Parameter Estimates:
    lV    lka
lka  0.68
lCl -0.61 -0.43

Number of iterations to convergence: 9
Achieved convergence tolerance: 4.68e-06
```