

# Package ‘tci’

September 27, 2025

**Title** Target Controlled Infusion (TCI)

**Version** 0.2.1

**Description** Implementation of target-controlled infusion algorithms for compartmental pharmacokinetic and pharmacokinetic-pharmacodynamic models. Jacobs (1990) <[doi:10.1109/10.43622](https://doi.org/10.1109/10.43622)>; Marsh et al. (1991) <[doi:10.1093/bja/67.1.41](https://doi.org/10.1093/bja/67.1.41)>; Shafer and Gregg (1993) <[doi:10.1001/199805000-00006](https://doi.org/10.1001/199805000-00006)>; Abuhelwa, Foster, and Upton (2015) <[doi:10.1016/j.vascn.2015.03.004](https://doi.org/10.1016/j.vascn.2015.03.004)>; Eleveld et al. (2018) <[doi:10.1016/j.bja.2018.01.018](https://doi.org/10.1016/j.bja.2018.01.018)>.

**Depends** R (>= 4.2.0)

**License** GPL-2

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/jarrettrt/tci>

**BugReports** <https://github.com/jarrettrt/tci/issues>

**Imports** ggplot2, stats, utils, mvtnorm, reshape, Rcpp, knitr

**LinkingTo** Rcpp, RcppArmadillo

**SystemRequirements** C++17

**Suggests** testthat (>= 2.1.0), rmarkdown, mrgsolve, gridExtra, reshape2, truncnorm, xtable

**RoxygenNote** 7.1.1

**NeedsCompilation** yes

**VignetteBuilder** knitr

**Author** Ryan Jarrett [aut, cre]

**Maintainer** Ryan Jarrett <[ryantjarrett@gmail.com](mailto:ryantjarrett@gmail.com)>

**Repository** CRAN

**Date/Publication** 2025-09-27 20:20:02 UTC

## Contents

|                              |    |
|------------------------------|----|
| apply_tci . . . . .          | 3  |
| assign_pars . . . . .        | 4  |
| bayes_update . . . . .       | 5  |
| clc . . . . .                | 6  |
| eleveld_pd . . . . .         | 8  |
| eleveld_pk . . . . .         | 9  |
| elvdlpars . . . . .          | 10 |
| emax . . . . .               | 11 |
| emax_eleveld . . . . .       | 11 |
| emax_inv . . . . .           | 12 |
| emax_inv_eleveld . . . . .   | 12 |
| emax_inv_remi . . . . .      | 13 |
| emax_remi . . . . .          | 13 |
| format_pars . . . . .        | 14 |
| infer_pkfn . . . . .         | 15 |
| inf_manual . . . . .         | 15 |
| inf_tci . . . . .            | 16 |
| init_pkmod . . . . .         | 17 |
| init_poppkmod . . . . .      | 19 |
| list_parnms . . . . .        | 20 |
| list_pkmods . . . . .        | 20 |
| log_likelihood . . . . .     | 21 |
| log_posterior_neg . . . . .  | 22 |
| log_prior . . . . .          | 23 |
| olc . . . . .                | 24 |
| pkmod . . . . .              | 25 |
| pkmod1cpt . . . . .          | 26 |
| pkmod2cpt . . . . .          | 27 |
| pkmod3cpt . . . . .          | 28 |
| pkmod3cptm . . . . .         | 28 |
| pkmod_eleveld_ppf . . . . .  | 29 |
| pkmod_eleveld_remi . . . . . | 30 |
| pkmod_kim . . . . .          | 31 |
| pkmod_marsh . . . . .        | 32 |
| pkmod_minto . . . . .        | 32 |
| pkmod_schnider . . . . .     | 33 |
| plot.sim_tci . . . . .       | 34 |
| poppkmod . . . . .           | 35 |
| predict.pkmod . . . . .      | 37 |
| predict.poppkmod . . . . .   | 38 |
| print.pkmod . . . . .        | 39 |
| print.poppkmod . . . . .     | 39 |
| print.sim_tci . . . . .      | 40 |
| sample_iiv . . . . .         | 41 |
| sample_pkmod . . . . .       | 42 |
| simulate.pkmod . . . . .     | 42 |

|                             |    |
|-----------------------------|----|
| simulate.poppkmod . . . . . | 43 |
| simulate_clc . . . . .      | 44 |
| simulate_olc . . . . .      | 46 |
| simulate_tci . . . . .      | 47 |
| tci_documentation . . . . . | 49 |
| tci_effect . . . . .        | 50 |
| tci_effect_only . . . . .   | 51 |
| tci_plasma . . . . .        | 52 |
| update.pkmod . . . . .      | 52 |
| validate_pkmod . . . . .    | 53 |
| validate_poppkmod . . . . . | 54 |

|              |           |
|--------------|-----------|
| <b>Index</b> | <b>55</b> |
|--------------|-----------|

---

|           |                                                  |
|-----------|--------------------------------------------------|
| apply_tci | <i>Apply a TCI algorithm to a 'pkmod' object</i> |
|-----------|--------------------------------------------------|

---

## Description

Apply a TCI algorithm to a set of targets and a 'pkmod' object to calculate infusion rates.

## Usage

```

apply_tci(
  pkmod,
  target_vals,
  target_tms,
  type = c("plasma", "effect"),
  dtm = NULL,
  custom_alg = NULL,
  inittm = 0,
  ignore_pd = FALSE,
  ...
)
```

## Arguments

|             |                                                                                                                          |
|-------------|--------------------------------------------------------------------------------------------------------------------------|
| pkmod       | 'pkmod' object created by 'pkmod()'.                                                                                     |
| target_vals | A vector of numeric values indicating PK or PD targets for TCI algorithm.                                                |
| target_tms  | A vector of numeric values indicating times at which the TCI algorithm should begin targeting each value.                |
| type        | Type of TCI algorithm to be used. Options are plasma- or effect-site targeting.                                          |
| dtm         | TCI update frequency. Defaults to 1/6, corresponding to 10-second intervals if model parameters are in terms of minutes. |

|            |                                                                                                                                                                                                                                                                                                                                       |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| custom_alg | Custom TCI algorithm to be used instead of default plasma- or effect-site targeting algorithms. The algorithm should be a function that takes minimum arguments 'Ct', 'pkmod', and 'dtm' and returns a single infusion rate. See 'tci_plasma' or 'tci_effect' for examples and vignette on custom models/algorithms for more details. |
| inittm     | Initial time to start TCI algorithm. Cannot be greater than the minimum value of 'target_tms'.                                                                                                                                                                                                                                        |
| ignore_pd  | Logical. Should the PD component of the pkmod object (if present) be ignored. By default, predict.tciinf will assume that 'value' refers to PD targets if a PD model is specified.                                                                                                                                                    |
| ...        | Arguments passed to TCI algorithm                                                                                                                                                                                                                                                                                                     |

Examples

```
# 3-compartment model with effect-site
my_mod <- pkmod(pars_pk = c(v1 = 8.995, v2 = 17.297, v3 = 120.963, cl = 1.382,
q2 = 0.919, q3 = 0.609, ke0 = 1.289))
# plasma targeting
apply_tci(my_mod, target_vals = c(2,3,4,4), target_tms = c(0,2,3,10), "plasma")
# effect-site targeting
apply_tci(my_mod, target_vals = c(2,3,4,4), target_tms = c(0,2,3,10), "effect")
# incorporate PD model
my_mod_pd <- update(my_mod, pars_pd = c(c50 = 2.8, gamma = 1.47, e0 = 93, emx = 93),
pdfn = emax, pdinv = emax_inv)
apply_tci(my_mod_pd, target_vals = c(70,60,50,50), target_tms = c(0,2,3,10), "effect")
```

---

|             |                                                                                            |
|-------------|--------------------------------------------------------------------------------------------|
| assign_pars | <i>Set default PK parameter values Set default PK parameter values for a pkmod object.</i> |
|-------------|--------------------------------------------------------------------------------------------|

---

Description

Set default PK parameter values Set default PK parameter values for a pkmod object.

Usage

```
assign_pars(pkmod, pars)
```

Arguments

|       |                                                    |
|-------|----------------------------------------------------|
| pkmod | pkmod object                                       |
| pars  | PK parameters to assign as default values of pkmod |

Value

pkmod object

bayes\_update

*Update PK-PD model parameters using observed data values***Description**

Function will update parameters of 'pkmod' object based on available data using a Laplace approximation to the posterior. Parameters and "Omega" values from 'pkmod' are used as prior point estimates and variance terms, respectively. Parameters fixed within the Omega matrix are assumed to be fixed and are not updated.

**Usage**

```
bayes_update(lpars, pkmod, inf, tms, obs, update_init = FALSE, ...)
```

**Arguments**

|             |                                                                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpars       | Logged parameter values. Can be a subset of the full set of PK or PK-PD parameter values.                                                                                               |
| pkmod       | 'pkmod' object. Mean values are a subset of log(pars_pk), log(pars_pd), log(sigma_add), log(sigma_mult). PK-PD parameter values not specified in 'lpars' will be inferred from 'pkmod'. |
| inf         | Infusion schedule                                                                                                                                                                       |
| tms         | Times associated with observations                                                                                                                                                      |
| obs         | Observed values (concentrations or PD response values)                                                                                                                                  |
| update_init | Logical. Should initial values be updated in addition to parameter values?                                                                                                              |
| ...         | Arguments passed to update.pkmod                                                                                                                                                        |

**Value**

'pkmod' object with PK/PK-PD/sigma parameters updated based on minimizing negative logged posterior.

**Examples**

```
# evaluate negative log posterior at a new set of parameters
lpars = log(c(cl=11,q2=3,q3=25,v=20,v2=40,v3=80,ke0=1.15,sigma_add=0.3))
prior_vcov <- matrix(diag(c(0.265,0.346,0.209,0.610,0.565,0.597,0.702,0.463)),
8,8, dimnames = list(NULL,names(lpars)))
my_mod <- pkmod(pars_pk = c(cl = 10, q2 = 2, q3 =20, v = 15, v2 = 30, v3 = 50, ke0 = 1.2),
sigma_add = 0.2, log_response = TRUE, Omega = prior_vcov)
inf <- inf_manual(inf_tms = 0, inf_rate = 80, duration = 2)
tms <- c(1,2,4,8,12)
obs <- simulate(my_mod, inf = inf, tms = tms)
bayes_update(lpars, my_mod, inf, tms, obs)

# evaluate log-prior for subset of parameters (remove volume parameters)
```

```

lpars_sub = log(c(cl=11,q2=3,q3=25,ke0=1.15,sigma_add=0.15))
my_mod <- update(my_mod, Omega = matrix(diag(c(0.265,0.346,0.209,0.702,0.463)),5,5,
  dimnames = list(NULL,names(lpars_sub))))
bayes_update(lpars_sub, my_mod, inf, tms, obs)

# add a pd response and replace multiplicative error with additive error
my_mod_pd <- update(my_mod, pars_pd = c(c50 = 2.8, gamma = 1.47, e0 = 93, emx = 93),
pdfn = emax, pdinv = emax_inv, ecmt = 4, sigma_mult = 0, sigma_add = 4)
# simulate observations
obs_pd <- simulate(my_mod_pd, inf = inf, tms = seq(0,12,0.5))
# evaluate likelihood at new parameters
lpars_pd_eval = log(c(cl=11,q3=25,v=15,ke0=1.15,sigma_add=4,c50=5,gamma=1))
prior_vcov_pd <- matrix(diag(c(0.265,0.209,0.610,0.702,0.230,0.242,0.1)),7,7,
dimnames = list(NULL,names(lpars_pd_eval)))
my_mod_pd <- update(my_mod_pd, Omega = prior_vcov_pd)
bayes_update(lpars_pd_eval, my_mod_pd, inf, tms, obs_pd)

```

---

 clc

---

*Simulate closed-loop control*


---

## Description

Simulate closed-loop control using Bayesian updates. Infusion rates are calculated using ‘pkmod\_prior’ to reach ‘target\_vals’ at ‘target\_tms’. Data values are simulated using ‘pkmod\_true’ at ‘obs\_tms’. ‘pkmod\_prior’ and ‘pkmod\_true’ do not need to have the same structure. Model parameters are updated at each update time using all available simulated observations. Processing delays can be added through the ‘delay’ argument, such that observations aren’t made available to the update mechanism until ‘update\_tms >= obs\_tms + delay’.

## Usage

```

clc(
  pkmod_prior,
  pkmod_true,
  target_vals,
  target_tms,
  obs_tms,
  update_tms,
  type = c("effect", "plasma"),
  custom_alg = NULL,
  resp_bounds = NULL,
  delay = 0,
  seed = NULL
)

```

## Arguments

|             |                                                                                                                                                                                      |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkmod_prior | ‘pkmod’ object describing a PK/PK-PD model that is used to calculate TCI infusion rates and is updated as data are simulated and incorporated. Must have an associated Omega matrix. |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

|             |                                                                                                                                                                                                                                                                                                                                                      |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkmod_true  | 'pkmod' object describing the patient's "true" response. This model will be used to simulate observations.                                                                                                                                                                                                                                           |
| target_vals | A vector of numeric values indicating PK or PD targets for TCI algorithm.                                                                                                                                                                                                                                                                            |
| target_tms  | A vector of numeric values indicating times at which the TCI algorithm should begin targeting each value.                                                                                                                                                                                                                                            |
| obs_tms     | Times at which data values should be simulated from 'pkmod_true'.                                                                                                                                                                                                                                                                                    |
| update_tms  | Times at which 'pkmod_prior' should be updated using all available simulated observations.                                                                                                                                                                                                                                                           |
| type        | Type of TCI algorithm to be used. Options are "plasma" and "effect". Defaults to "effect". Will be overwritten if 'custom_alg' is non-null.                                                                                                                                                                                                          |
| custom_alg  | Custom TCI algorithm to overwrite default plasma- or effect-site targeting.                                                                                                                                                                                                                                                                          |
| resp_bounds | Optional vector of two values indicating minimum and maximum values possible for the response.                                                                                                                                                                                                                                                       |
| delay       | Optional numeric value indicating a temporal delay between when observations are simulated and when they should be made available for updating 'pkmod_prior'. For example, a delay should be set to account for a processing time delay in Bispectral Index measurements or the time required to measure drug concentrations from collected samples. |
| seed        | An integer used to initialize the random number generator.                                                                                                                                                                                                                                                                                           |

## Examples

```
prior_vcov <- matrix(diag(c(0.265,0.346,0.209,0.610,0.565,0.597,0.702,0.463)),
8,8, dimnames = list(NULL,c('c1','q2','q3','v','v2','v3','ke0','sigma_add')))
pkmod_prior <- pkmod(pars_pk = c(c1 = 10, q2 = 2, q3 = 20, v = 15, v2 = 30, v3 = 50, ke0 = 1.2),
sigma_add = 0.2, log_response = TRUE, Omega = prior_vcov)
pkmod_true <- pkmod(pars_pk = c(c1 = 16, q2 = 4, q3 = 10, v = 20, v2 = 20, v3 = 80, ke0 = 0.8),
sigma_add = 0.1, log_response = TRUE)
target_vals <- c(2,3,4,3,3)
target_tms <- c(0,5,10,36,60)
obs_tms <- c(1,2,4,8,12,16,24,36,48)
update_tms <- c(5,15,25,40,50)
sim <- clc(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms, update_tms, seed = 200)
len <- 500
tms <- seq(0,60,length.out = len)
# true, prior, posterior plasma concentrations
df <- data.frame(time = rep(tms,3),
value = c(predict(pkmod_true, sim$inf,tms)[,1],
predict(pkmod_prior, sim$inf,tms)[,1],
predict(sim$pkmod_post, sim$inf, tms)[,1]),
type = c(rep("true",len),rep("prior",len),rep("posterior",len)))
library(ggplot2)
ggplot(df, aes(x = time, y = value, color = type)) +
  geom_line() +
  geom_point(data = sim$obs, aes(x = time, y = obs), inherit.aes = FALSE) +
  geom_step(data = data.frame(time = target_tms, value = target_vals),
aes(x = time, y = value), inherit.aes = FALSE)
```

```

# PK-PD example with observation delay (30 sec)
prior_vcov <- matrix(diag(c(0.265,0.346,0.209,0.702,0.242,0.230)),6,6,
dimnames = list(NULL,c('cl','q2','q3','ke0','c50','sigma_add')))
pkpdm_prior <- update(pkmod_prior, pars_pd = c(c50 = 2.8, gamma = 1.47, e0 = 93, emx = 93),
pdfn = emax, pdinv = emax_inv, sigma_add = 4, log_response = FALSE, Omega = prior_vcov)
pkpdm_true <- update(pkmod_true, pars_pd = c(c50 = 3.4, gamma = 1.47, e0 = 93, emx = 93),
pdfn = emax, pdinv = emax_inv, sigma_add = 3, log_response = FALSE)
target_vals <- c(75,60,50,50)
target_tms <- c(0,3,6,10)
obs_tms <- seq(1/6,10,1/6)
update_tms <- seq(1,10,0.5)
## Not run:
sim_pkpd <- clc(pkpdm_prior, pkpdm_true, target_vals, target_tms, obs_tms,
update_tms, seed = 201, delay = 0.5)
# plot results
tms <- seq(0,10,length.out = len)
df <- data.frame(time = rep(tms,3),
                 value = c(predict(pkpdm_true, sim_pkpd$inf,tms)[,"pdresp"],
predict(pkpdm_prior, sim_pkpd$inf,tms)[,"pdresp"],
predict(sim_pkpd$pkmod_post, sim_pkpd$inf, tms)[,"pdresp"]),
                 type = c(rep("true",len),rep("prior",len),rep("posterior",len)))
library(ggplot2)
ggplot(df, aes(x = time, y = value, color = type)) +
  geom_line() +
  geom_point(data = sim_pkpd$obs, aes(x = time, y = obs), inherit.aes = FALSE) +
  labs(x = "Hours", y = "Bispectral Index") + theme_bw() +
  geom_vline(xintercept = update_tms, linetype = "dotted", alpha = 0.6) +
  geom_step(data = data.frame(time = target_tms, value = target_vals),
aes(x = time, y = value), inherit.aes = FALSE)

## End(Not run)

```

---

eleveld\_pd

*Eleveld et al. pharmacodynamic data*


---

## Description

Empirical Bayes (EB) estimates of PD parameters made by the Eleveld et al (2018) PK-PD model. EB estimates were calculated using the PK-PD datasets and NONMEM files provided by Eleveled et al. (2018). The original datasets were obtained through the Open TCI Initiative website ([opentci.org](http://opentci.org)) and based on contributions from a number of researchers who made their datasets publically available.

## Usage

```
data(eleveld_pd)
```



**Format**

A data frame with 122 rows and 15 variables:

**ID** Patient ID

**E50** EB estimate of effect-site concentration required to achieve 50 percent response

**KE0** EB estimate of elimination rate from effect-site compartment

**EMAX** EB estimate of baseline bispectral index (BIS) with no drug administered

**GAM** EB estimate of Hill parameter when the effect-site concentration is less than E50

**GAM1** EB estimate of Hill parameter when the effect-site concentration is greater than E50

**RES0** EB estimate of residual error term

**ALAG1** Estimated time lag in BIS measurements due to patient age (fixed-effects only)

**AGE** Patient's age (years)

**WGT** Patient's weight (kg)

**HGT** Patient's height (cm)

**M1F2** Patient's sex: male = 1, female = 2

**A1V2** Sampling site: arterial sampling = 1, venous sampling = 2

**PMA** Patient's post-menstrual age. Assumed to be age + 40 weeks if not provided

**TECH** Presence of concomitant anaesthetic techniques (Local anesthetic = 1, Opioids = 2)

**References**

Eleved et al. (2018) British Journal of Anesthesia Vol. 120, 5:942-959. \doi:10.1016/j.bja.2018.01.018

---

eleved\_pk

*Eleved et al. pharmacokinetic data*

---

**Description**

Empirical Bayes (EB) estimates of PK parameters for the Eleved et al. (2018) PK-PD model. EB estimates were calculated using the PK-PD datasets and NONMEM files provided by Eleved et al. (2018). The original datasets were obtained through the Open TCI Initiative website (opentci.org) and based on contributions from a number of researchers who made their datasets publically available.

**Usage**

data(eleved\_pk)

**Format**

A data frame with 1033 rows and 16 variables:

**ID** Patient ID

**V1** EB estimate of first compartment volume

**V2** EB estimate of second compartment volume

**V3** EB estimate of third compartment volume

**CL** EB estimate of clearance for the first compartment

**Q2** EB estimate of inter-compartmental clearance for second compartment

**Q3** EB estimate of inter-compartmental clearance for third compartment

**AGE** Patient's age (years)

**WGT** Patient's weight (kg)

**HGT** Patient's height (cm)

**M1F2** Patient's sex: male = 1, female = 2

**PMA** Patient's post-menstrual age. Assumed to be age + 40 weeks if not provided

**TECH** Presence of concomitant anaesthetic techniques (Local anesthetic = 1, Opioids = 2)

**BMI** Patient's BMI

**FFM** Patient's fat-free mass (FFM)

**A1V2** Sampling site: arterial sampling = 1, venous sampling = 2

**References**

Elefeld et al. (2018) British Journal of Anesthesia Vol. 120, 5:942-959. \doi:10.1016/j.bja.2018.01.018

---

elvdipars

*Get logged parameters updated in Elefeld model*

---

**Description**

Extract the logged parameter values to be updated within the Elefeld model from a data frame of patient PK-PD values.

**Usage**

```
elvdipars(x, pd = TRUE)
```

**Arguments**

|           |                                                                         |
|-----------|-------------------------------------------------------------------------|
| <b>x</b>  | Vector or data frame with Elefeld PK-PD model parameters                |
| <b>pd</b> | Logical. Should PD parameters be returned in addition to PK parameters. |

**Value**

List of parameters used by Elefeld PK-PD model.

---

|      |                                 |
|------|---------------------------------|
| emax | <i>E<sub>max</sub> function</i> |
|------|---------------------------------|

---

### Description

E<sub>max</sub> function. c<sub>50</sub> is the concentration eliciting a 50 identifying the slope of the E<sub>max</sub> curve at c<sub>50</sub>, E<sub>0</sub> is the response value with no drug present, E<sub>m</sub> is the maximum effect size.

### Usage

```
emax(ce, pars)
```

### Arguments

|      |                                                                                                       |
|------|-------------------------------------------------------------------------------------------------------|
| ce   | Vector of effect-site concentrations.                                                                 |
| pars | Named vector of parameter values with names (c <sub>50</sub> ,gamma,e <sub>0</sub> ,e <sub>m</sub> ). |

### Value

Numeric vector of same length as ce.

### Examples

```
pars_emax <- c(c50 = 1.5, gamma = 1.47, e0 = 100, emx = 100)
ce_seq <- seq(0,4,0.1)
plot(ce_seq, emax(ce_seq, pars_emax), type = "l",
     xlab = "Effect-site concentrtrion (ug/mL)", ylab = "BIS")
```

---

|              |                                                           |
|--------------|-----------------------------------------------------------|
| emax_eleveld | <i>E<sub>max</sub> function for Eleveld (2018) model.</i> |
|--------------|-----------------------------------------------------------|

---

### Description

The parameter gamma takes one of two values depending on whether ce ≤ c<sub>50</sub>.

### Usage

```
emax_eleveld(ce, pars)
```

### Arguments

|      |                                                                                                                   |
|------|-------------------------------------------------------------------------------------------------------------------|
| ce   | Vector of effect-site concentrations.                                                                             |
| pars | Vector of parameter values in order (c <sub>50</sub> ,gamma,gamma <sub>2</sub> ,e <sub>0</sub> ,e <sub>m</sub> ). |

### Value

Numeric vector of same length as ce.

**Examples**

```
pars_emax_eleveld <- c(c50 = 1.5, e0 = 100, gamma = 1.47, gamma2 = 1.89)
ce_seq <- seq(0,4,0.1)
plot(ce_seq, emax_eleveld(ce_seq, pars_emax_eleveld), type = "l",
     xlab = "Effect-site concentrtrion (ug/mL)", ylab = "BIS")
```

---

|          |                              |
|----------|------------------------------|
| emax_inv | <i>Inverse Emax function</i> |
|----------|------------------------------|

---

**Description**

Inverse Emax function to return effect-site concentrations required to reach target effect.

**Usage**

```
emax_inv(pdresp, pars)
```

**Arguments**

|        |                                                                 |
|--------|-----------------------------------------------------------------|
| pdresp | PD response values                                              |
| pars   | Named vector of parameter values with names (c50,gamma,E0,Emx). |

**Value**

Numeric vector of same length as pdresp.

**Examples**

```
pars_emax <- c(c50 = 1.5, gamma = 4, e0 = 100, emx = 100)
ce_seq <- seq(0,4,0.1)
all.equal(emax_inv(emax(ce_seq, pars_emax), pars_emax), ce_seq)
```

---

|                  |                              |
|------------------|------------------------------|
| emax_inv_eleveld | <i>Inverse Emax function</i> |
|------------------|------------------------------|

---

**Description**

Inverse of Emax function used by Eleveld population PK model.

**Usage**

```
emax_inv_eleveld(pdresp, pars)
```

**Arguments**

|        |                                                                 |
|--------|-----------------------------------------------------------------|
| pdresp | PD response values                                              |
| pars   | Named vector of parameter values with names (c50,gamma,E0,Emx). |

### Value

Numeric vector of same length as pdresp.

### Examples

```
pars_emax_eleveld <- c(c50 = 1.5, e0 = 100, gamma = 1.47, gamma2 = 1.89)
ce_seq <- seq(0,4,0.1)
all.equal(emax_inv_eleveld(emax_eleveld(ce_seq, pars_emax_eleveld), pars_emax_eleveld), ce_seq)
```

---

|               |                                                                        |
|---------------|------------------------------------------------------------------------|
| emax_inv_remi | <i>Inverse Emax function implemented by Eleveld remifentanil model</i> |
|---------------|------------------------------------------------------------------------|

---

### Description

Inverse Emax function to return effect-site concentrations required to reach target effect.

### Usage

```
emax_inv_remi(pdresp, pars)
```

### Arguments

|        |                                                                 |
|--------|-----------------------------------------------------------------|
| pdresp | PD response values                                              |
| pars   | Named vector of parameter values with names (c50,gamma,E0,Emx). |

### Value

Numeric vector of same length as pdresp.

### Examples

```
pars_emax <- c(e0 = 19, emx = 5.6, c50 = 12.7, gamma = 2.87)
emax_inv_remi(emax_remi(10, pars_emax), pars_emax)
```

---

|           |                                                                |
|-----------|----------------------------------------------------------------|
| emax_remi | <i>Emax function implemented by Eleveld remifentanil model</i> |
|-----------|----------------------------------------------------------------|

---

### Description

Emax function. c50 is the concentration eliciting a 50 identifying the slope of the Emax curve at c50, E0 is the response value with no drug present, Emx is the maximum effect size.

### Usage

```
emax_remi(ce, pars)
```

**Arguments**

`ce`                      Vector of effect-site concentrations.

`pars`                    Named vector of parameter values with names (c50,gamma,e0,emx).

**Value**

Numeric vector of same length as `ce`.

**Examples**

```
pars_emax <- c(e0 = 19, emx = 5.6, c50 = 12.7, gamma = 2.87)
ce_seq <- seq(0,60,0.1)
plot(ce_seq, emax_remi(ce_seq, pars_emax), type = "l",
     xlab = "Effect-site concentrtrion (ug/mL)", ylab = "Spectral Edge Frequency (HZ)")
```

---

|             |                                                                                                                                                      |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| format_pars | <i>Format parameters for use in Rcpp functions Order parameters for 1-4 compartment models to be used in Rcpp functions in predict_pkmod method.</i> |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------|

---

**Description**

Format parameters for use in Rcpp functions

Order parameters for 1-4 compartment models to be used in Rcpp functions in predict\_pkmod method.

**Usage**

```
format_pars(pars, ncmt = 3)
```

**Arguments**

`pars`                    Vector of named parameters. Names can be capitalized or lowercase and can include variations of "V1" as "V" or clearance terms rather than elimination rate constants.

`ncmt`                    Number of compartments in the model. This should be a value between 1 and 4. If `ncmt = 4`, it assumes that the fourth compartment is an effect-site without a corresponding volume parameter.

**Value**

Numeric vector of transformed parameter values.

**Examples**

```
format_pars(c(V1 = 8.9, CL = 1.4, q2 = 0.9, v2 = 18), ncmt = 2)
format_pars(c(V1 = 8.9, CL = 1.4, q2 = 0.9, v2 = 18, cl2 = 3), ncmt = 2)
```

---

|            |                                           |
|------------|-------------------------------------------|
| infer_pkfn | <i>Identify pkfn from parameter names</i> |
|------------|-------------------------------------------|

---

### Description

Identify structural PK model function (i.e., 'pkfn') from parameter names. Models available are 1-, 2-, and 3-compartment mammillary models, or 3-compartment with an effect site, corresponding to functions 'pkmod1cpt', 'pkmod2cpt', 'pkmod3cpt', and 'pkmod3cptm', respectively.

### Usage

```
infer_pkfn(parnms)
```

### Arguments

parnms            Vector of parameter names.

### Value

Returns one of the following functions: 'pkmod1cpt', 'pkmod2cpt', 'pkmod3cpt', or 'pkmod3cptm' based on the parameter names entered.

### Examples

```
# 1-compartment
infer_pkfn(c("CL", "V"))
infer_pkfn(c("Cl", "v1"))
# 2-compartment
infer_pkfn(c("CL", "v", "v2", "q"))
# 3-compartment
infer_pkfn(c("CL", "v", "v2", "q", "Q2", "V3"))
# 3-compartment with effect-site
infer_pkfn(c("CL", "v", "v2", "q", "Q2", "V3", "ke0"))
```

---

|            |                          |
|------------|--------------------------|
| inf_manual | <i>infusion schedule</i> |
|------------|--------------------------|

---

### Description

Returns a data frame describing a set of infusions to be administered. Output can be used as argument "inf" in predict\_pkmod.

### Usage

```
inf_manual(inf_tms, inf_rate, duration = NULL)
```

**Arguments**

|          |                                                                                                                                                                                                                              |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| inf_tms  | Vector of time to begin infusions. If duration is NULL, times expected to include both infusion start and infusion end times.                                                                                                |
| inf_rate | Vector of infusion rates. Must have length equal to either length(inf_tms) or 1. If length(inf_rate)=1, the same infusion rate will be administered at each time. Either inf_rate or target must be specified, but not both. |
| duration | Optional duration of infusions.                                                                                                                                                                                              |

**Value**

Matrix of infusion rates, start and end times.

**Examples**

```
# specify start and end times, as well as infusion rates (including 0).
inf_manual(inf_tms = c(0,0.5,4,4.5,10), inf_rate = c(100,0,80,0,0))
# specify start times, single infusion rate and single duration
inf_manual(inf_tms = c(0,4), inf_rate = 100, duration = 0.5)
# multiple infusion rates, single duration
inf_manual(inf_tms = c(0,4), inf_rate = c(100,80), duration = 0.5)
# multiple sequential infusion rates
inf_manual(inf_tms = seq(0,1,1/6), inf_rate = 100, duration = 1/6)
# single infusion rate, multiple durations
inf_manual(inf_tms = c(0,4), inf_rate = 100, duration = c(0.5,1))
# multiple infusion rates, multiple durations
inf_manual(inf_tms = c(0,4), inf_rate = c(100,80), duration = c(0.5,1))
```

---

inf\_tci

---

*Target-controlled infusion*


---

**Description**

Apply a TCI algorithm to a set of targets and a 'pkmod' or 'poppkmod' object to calculate infusion rates.

**Usage**

```
inf_tci(
  pkmod,
  target_vals,
  target_tms,
  type = c("plasma", "effect"),
  dtm = NULL,
  custom_alg = NULL,
  inittm = 0,
  ignore_pd = FALSE,
  ...
)
```



**Arguments**

|             |                                                                                                                                                                                                                                                                                                                                       |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkmod       | 'pkmod' object created by 'pkmod()' or a 'popppkmod' object created by 'popppkmod()'.                                                                                                                                                                                                                                                 |
| target_vals | A vector of numeric values indicating PK or PD targets for TCI algorithm.                                                                                                                                                                                                                                                             |
| target_tms  | A vector of numeric values indicating times at which the TCI algorithm should begin targeting each value.                                                                                                                                                                                                                             |
| type        | Type of TCI algorithm to be used. Options are plasma- or effect-site targeting.                                                                                                                                                                                                                                                       |
| dtm         | TCI update frequency. Defaults to 1/6, corresponding to 10-second intervals if model parameters are in terms of minutes.                                                                                                                                                                                                              |
| custom_alg  | Custom TCI algorithm to be used instead of default plasma- or effect-site targeting algorithms. The algorithm should be a function that takes minimum arguments 'Ct', 'pkmod', and 'dtm' and returns a single infusion rate. See 'tci_plasma' or 'tci_effect' for examples and vignette on custom models/algorithms for more details. |
| inittm      | Initial time to start TCI algorithm. Cannot be greater than the minimum value of 'target_tms'.                                                                                                                                                                                                                                        |
| ignore_pd   | Logical. Should the PD component of the pkmod object (if present) be ignored. By default, predict.tciinf will assume that 'value' refers to PD targets if a PD model is specified.                                                                                                                                                    |
| ...         | Arguments passed to TCI algorithm                                                                                                                                                                                                                                                                                                     |

**Examples**

```
# 3-compartment model with effect-site
my_mod <- pkmod(pars_pk = c(v1 = 8.995, v2 = 17.297, v3 = 120.963, c1 = 1.382,
q2 = 0.919, q3 = 0.609, ke0 = 1.289))
# plasma targeting
inf_tci(my_mod, target_vals = c(2,3,4,4), target_tms = c(0,2,3,10), "plasma")
# effect-site targeting
inf_tci(my_mod, target_vals = c(2,3,4,4), target_tms = c(0,2,3,10), "effect")
# popppkmod object
data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),
HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
elvd_mod <- popppkmod(data, drug = "ppf", model = "eleveld")
inf_tci(elvd_mod, target_vals = c(2,3,4,4), target_tms = c(0,2,3,10), "effect")
```

init\_pkmod

*Create an object with class "pkmod"***Description**

Create an object with class "pkmod"

**Usage**

```
init_pkmod(
  pars_pk = NULL,
  init = NULL,
  pkfn = NULL,
  pars_pd = NULL,
  pdfn = NULL,
  pdinv = NULL,
  pcmt = NULL,
  ecmt = NULL,
  sigma_add = NULL,
  sigma_mult = NULL,
  log_response = NULL,
  Omega = NULL
)
```

**Arguments**

|                           |                                                                                                                                                                                                                          |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pars_pk</code>      | Vector or matrix of named PK parameters. If not specified, the pkmod function will be inferred from the parameter names. Print <code>'list_parnms()'</code> for acceptable parameter names.                              |
| <code>init</code>         | Vector of initial concentrations. Will default to values of zero in all compartments if not specified.                                                                                                                   |
| <code>pkfn</code>         | PK model function. Functions provided in 'tci' include <code>'pkmod1cpt'</code> , <code>'pkmod2cpt'</code> , <code>'pkmod3cpt'</code> , and <code>'pkmod3cptm'</code> . User-defined functions should be specified here. |
| <code>pars_pd</code>      | PD model parameters if a PD model is specified                                                                                                                                                                           |
| <code>pdfn</code>         | PD model function                                                                                                                                                                                                        |
| <code>pdinv</code>        | Inverse PD model function for use in TCI algorithms                                                                                                                                                                      |
| <code>pcmt</code>         | Index of plasma compartment. Defaults to first compartment if not specified.                                                                                                                                             |
| <code>ecmt</code>         | Index of effect-site compartment if a PD model is specified. Will default to last compartment if unspecified.                                                                                                            |
| <code>sigma_add</code>    | Standard deviation of additive residual error                                                                                                                                                                            |
| <code>sigma_mult</code>   | Standard deviation of multiplicative residual error                                                                                                                                                                      |
| <code>log_response</code> | Logical value indicating if the response should be logged prior to adding error.                                                                                                                                         |
| <code>Omega</code>        | Optional matrix of random effect parameters. Column names should correspond to names of <code>pars_pk</code> , <code>pars_pd</code> , and <code>sigma_add</code> or <code>sigma_mult</code> .                            |

**Value**

A list with class `pkmod` for which `print`, `plot`, `predict`, and `simulate` methods exist.

**Examples**

```
# create a pkmod object for a one compartment model with plasma targeting
init_pkmod()
```

---

|               |                                        |
|---------------|----------------------------------------|
| init_poppkmod | <i>Initialize a 'poppkmod' object.</i> |
|---------------|----------------------------------------|

---

## Description

Generate a 'poppkmod' object from an existing population PK model for propofol or remifentanil using patient covariates. Available models for propofol are the Marsh, Schnider, and Eleveld models. Available models for remifentanil are the Minto, Kim, and Eleveld models. Input is a data frame with rows corresponding to individuals and columns recording patient covariates.

## Usage

```
init_poppkmod(data = NULL, drug = NULL, model = NULL, sample = NULL, PD = NULL)
```

## Arguments

|        |                                                                                                                                                                 |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| data   | Data frame of patient covariates. ID values, if used, should be in a column labeled "id" or "ID"                                                                |
| drug   | "ppf" for propofol or "remi" for remifentanil. Defaults to "ppf".                                                                                               |
| model  | Model name. Options are "marsh", "schnider", or "eleveld" if drug = "ppf", or "minto", "kim", or "eleveld" if drug = "remi".                                    |
| sample | Logical. Should parameter values be sampled from interindividual distribution (TRUE) or evaluated at point estimates for covariates (FALSE)? Defaults to FALSE. |
| PD     | Should the PD component be evaluated for PK-PD models. Defaults to TRUE.                                                                                        |

## Value

'poppkmod' object

## Examples

```
data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),
  HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
init_poppkmod(data, drug = "ppf", model = "eleveld")
init_poppkmod(data, drug = "remi", model = "kim")
```

---

list\_parnms

*Identify pkfn from parameter names*


---

**Description**

Identify structural PK model function (i.e., ‘pkfn’) from parameter names. Models available are 1-, 2-, and 3-compartment mammillary models, or 3-compartment with an effect site, corresponding to functions ‘pkmod1cpt’, ‘pkmod2cpt’, ‘pkmod3cpt’, and ‘pkmod3cptm’, respectively.

**Usage**

```
list_parnms()
```

**Value**

Returns one of the following functions: ‘pkmod1cpt’, ‘pkmod2cpt’, ‘pkmod3cpt’, or ‘pkmod3cptm’ based on the parameter names entered.

**Examples**

```
list_parnms()
```

---

list\_pkmods

*Print population PK models available in ‘tci’*


---

**Description**

Print population PK models available in ‘tci’ for propofol (Marsh, Schnider, Eleveld) and remifentanyl (Minto, Kim, Eleveld).

**Usage**

```
list_pkmods()
```

**Value**

Prints function names, model types, and required covariates for each model.

**Examples**

```
list_pkmods()
```

---

|                |                                                                    |
|----------------|--------------------------------------------------------------------|
| log_likelihood | <i>Evaluate the log likelihood of a vector of parameter values</i> |
|----------------|--------------------------------------------------------------------|

---

**Description**

Evaluate the log likelihood of parameters given observed data. Can be applied to PK or PK-PD models.

**Usage**

```
log_likelihood(lpars, pkmod, inf, tms, obs)
```

**Arguments**

|       |                                                                                                                                                                                                         |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpars | Named vector of logged parameter values to be evaluated. This should include any PK or PD parameters, as well as residual error standard deviations (sigma_add or sigma_mult) that are to be evaluated. |
| pkmod | 'pkmod' object. Mean values are a subset of log(pars_pk), log(pars_pd), log(sigma_add), log(sigma_mult). PK-PD parameter values not specified in 'lpars' will be inferred from 'pkmod'.                 |
| inf   | Infusion schedule                                                                                                                                                                                       |
| tms   | Times associated with observations                                                                                                                                                                      |
| obs   | Observed values (concentrations or PD response values)                                                                                                                                                  |

**Value**

Numeric value of length 1

**Examples**

```
my_mod <- pkmod(pars_pk = c(cl = 10, q2 = 2, q3 = 20, v = 15, v2 = 30, v3 = 50,
  ke0 = 1.2), sigma_mult = 0.2)
inf <- inf_manual(inf_tms = 0, inf_rate = 80, duration = 2)
tms <- c(1,2,4,8,12)
obs <- simulate(my_mod, inf = inf, tms = tms)
# evaluate log-likelihood at a new set of parameters
lpars = log(c(cl=11,q2=3,q3=25,v=15,v2=30,v3=50,ke0=1.15,sigma_mult=0.3))
log_likelihood(lpars, my_mod, inf, tms, obs)

# estimate for a subset of parameters (exclude q2, v2, v3)
lpars_sub = log(c(cl=11,q3=25,v=15,ke0=1.15,sigma_mult=0.3))
log_likelihood(lpars_sub, my_mod, inf, tms, obs)

# add a pd response and replace multiplicative error with additive error
my_mod_pd <- update(my_mod, pars_pd = c(c50 = 2.8, gamma = 1.47, e0 = 93,
  emx = 93), pdfn = emax, pdinv = emax_inv, ecmtpt = 4, sigma_mult = 0, sigma_add = 4)
# simulate observations
obs_pd <- simulate(my_mod_pd, inf = inf, tms = seq(0,12,0.5))
```

```
# evaluate likelihood at new parameters
lpars_pd <- log(c(c1=11,q3=25,v=15,ke0=1.15,sigma_add=4,c50=5,gamma=1))
log_likelihood(lpars_pd, my_mod_pd, inf, tms = seq(0,12,0.5), obs_pd)
```

---

|                   |                                                                        |
|-------------------|------------------------------------------------------------------------|
| log_posterior_neg | <i>Evaluate the negative log posterior value of a parameter vector</i> |
|-------------------|------------------------------------------------------------------------|

---

## Description

Evaluate the negative log posterior value of a parameter vector given a set of observations and prior distribution for log-normally distributed PK or PK-PD parameters.

## Usage

```
log_posterior_neg(lpars, pkmod, inf, tms, obs)
```

## Arguments

|       |                                                                                                                                                                                         |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpars | Logged parameter values. Can be a subset of the full set of PK or PK-PD parameter values.                                                                                               |
| pkmod | 'pkmod' object. Mean values are a subset of log(pars_pk), log(pars_pd), log(sigma_add), log(sigma_mult). PK-PD parameter values not specified in 'lpars' will be inferred from 'pkmod'. |
| inf   | Infusion schedule                                                                                                                                                                       |
| tms   | Times associated with observations                                                                                                                                                      |
| obs   | Observed values (concentrations or PD response values)                                                                                                                                  |

## Value

Numeric value of length 1

## Examples

```
# evaluate negative log posterior at a new set of parameters
lpars = log(c(c1=11,q2=3,q3=25,v=20,v2=40,v3=80,ke0=1.15,sigma_add=0.3))
prior_vcov <- matrix(diag(c(0.265,0.346,0.209,0.610,0.565,0.597,0.702,0.463)),
  8,8, dimnames = list(NULL,names(lpars)))
my_mod <- pkmod(pars_pk = c(c1 = 10, q2 = 2, q3 =20, v = 15, v2 = 30, v3 = 50,
  ke0 = 1.2), sigma_add = 0.2, log_response = TRUE, Omega = prior_vcov)
inf <- inf_manual(inf_tms = 0, inf_rate = 80, duration = 2)
tms <- c(1,2,4,8,12)
obs <- simulate(my_mod, inf = inf, tms = tms)
log_posterior_neg(lpars, my_mod, inf, tms, obs)

# evaluate log-prior for subset of parameters (remove volume parameters)
lpars_sub = log(c(c1=11,q2=3,q3=25,ke0=1.15,sigma_add=0.15))
my_mod <- update(my_mod, Omega = matrix(diag(c(0.265,0.346,0.209,0.702,0.463)),
  5,5, dimnames = list(NULL,names(lpars_sub))))
```

```
log_posterior_neg(lpars_sub, my_mod, inf, tms, obs)

# add a pd response and replace multiplicative error with additive error
# evaluate likelihood at new parameters
lpars_pd_eval = log(c(c1=11,q3=25,v=15,ke0=1.15,sigma_add=4,c50=5,gamma=1))
prior_vcov_pd <- matrix(diag(c(0.265,0.209,0.610,0.702,0.230,0.242,0.1)),7,7,
dimnames = list(NULL,names(lpars_pd_eval)))
my_mod_pd <- update(my_mod, pars_pd = c(c50 = 2.8, gamma = 1.47, e0 = 93, emx = 93),
pdfn = emax, pdinv = emax_inv, ecmt = 4, sigma_mult = 0, sigma_add = 4,
Omega = prior_vcov_pd)
# simulate observations
obs_pd <- simulate(my_mod_pd, inf = inf, tms = seq(0,12,0.5))
log_posterior_neg(lpars_pd_eval, my_mod_pd, inf, tms, obs_pd)
```

log\_prior

*Calculate logged-prior probability for a set of parameters***Description**

Calculate logged-prior probability for a set of parameters, assuming that parameter values are log-normally distributed. Mean values are set as the logged parameter values in the ‘pkmod’ object. Variances are given by the diagonal elements of ‘prior\_vcov’.

**Usage**

```
log_prior(lpars, pkmod)
```

**Arguments**

|       |                                                                                                                                                                                         |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| lpars | Logged parameter values. Can be a subset of the full set of PK or PK-PD parameter values.                                                                                               |
| pkmod | ‘pkmod’ object. Mean values are a subset of log(pars_pk), log(pars_pd), log(sigma_add), log(sigma_mult). PK-PD parameter values not specified in ‘lpars’ will be inferred from ‘pkmod’. |

**Value**

Numeric value of length 1

**Examples**

```
# evaluate log-prior for pk parameters + residual
lpars = log(c(c1=11,q2=3,q3=25,v=15,v2=30,v3=50,ke0=1.15,sigma_add=0.15))
prior_vcov <- matrix(diag(c(0.265,0.346,0.209,0.610,0.565,0.597,0.702,0.463)), 8,8,
dimnames = list(NULL,names(lpars)))
my_mod <- pkmod(pars_pk = c(c1 = 10, q2 = 2, q3 =20, v = 15, v2 = 30, v3 = 50, ke0 = 1.2),
sigma_add = 0.2, log_response = TRUE, Omega = prior_vcov)
log_prior(lpars, my_mod)
```

```
# evaluate log-prior for subset of parameters (remove volume parameters)
lpars_sub = log(c(cl=11,q2=3,q3=25,ke0=1.15,sigma_add=0.15))
prior_vcov_sub <- matrix(diag(c(0.265,0.346,0.209,0.702,0.463)), 5,5,
  dimnames = list(NULL,names(lpars_sub)))
my_mod <- update(my_mod, Omega = prior_vcov_sub)
log_prior(lpars_sub, my_mod)
```

olc

*Simulate open-loop control*

## Description

Simulate open-loop control with target-controlled infusion for a 'pkmod' object. Infusion rates are calculated using 'pkmod\_prior' to reach 'target\_vals' at 'target\_tms'. Data values are simulated using 'pkmod\_true' at 'obs\_tms'. 'pkmod\_prior' and 'pkmod\_true' do not need to have the same structure.

## Usage

```
olc(
  pkmod_prior,
  pkmod_true,
  target_vals,
  target_tms,
  obs_tms,
  type = c("effect", "plasma"),
  custom_alg = NULL,
  resp_bounds = NULL,
  seed = NULL
)
```

## Arguments

|             |                                                                                                                                                                                      |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkmod_prior | 'pkmod' object describing a PK/PK-PD model that is used to calculate TCI infusion rates and is updated as data are simulated and incorporated. Must have an associated Omega matrix. |
| pkmod_true  | 'pkmod' object describing the patient's "true" response. This model will be used to simulate observations.                                                                           |
| target_vals | A vector of numeric values indicating PK or PD targets for TCI algorithm.                                                                                                            |
| target_tms  | A vector of numeric values indicating times at which the TCI algorithm should begin targeting each value.                                                                            |
| obs_tms     | Times at which data values should be simulated from 'pkmod_true'.                                                                                                                    |
| type        | Type of TCI algorithm to be used. Options are "plasma" and "effect". Defaults to "effect". Will be overwritten if 'custom_alg' is non-null.                                          |
| custom_alg  | Custom TCI algorithm to overwrite default plasma- or effect-site targeting.                                                                                                          |
| resp_bounds | Optional vector of two values indicating minimum and maximum values possible for the response.                                                                                       |
| seed        | An integer used to initialize the random number generator.                                                                                                                           |



## Examples

```
pkmod_prior <- pkmod(pars_pk = c(c1 = 10, q2 = 2, q3 = 20, v = 15, v2 = 30, v3 = 50, ke0 = 1.2))
pkmod_true <- pkmod(pars_pk = c(c1 = 16, q2 = 4, q3 = 10, v = 20, v2 = 20, v3 = 80, ke0 = 0.8),
  sigma_add = 0.1, log_response = TRUE)
target_vals <- c(2,3,4,3,3)
target_tms <- c(0,5,10,36,60)
obs_tms <- c(1,2,4,8,12,16,24,36,48)
sim <- olc(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms)
len <- 500
tms <- seq(0,60,length.out = len)
df <- data.frame(time = rep(tms,2),
  value = c(predict(pkmod_true, sim$inf,tms)[,1],
    predict(pkmod_prior, sim$inf,tms)[,1]),
  type = c(rep("true",len),rep("prior",len)))
library(ggplot2)
ggplot(df, aes(x = time, y = value, color = type)) +
  geom_step(data = data.frame(time = target_tms, value = target_vals),
    aes(x = time, y = value), inherit.aes = FALSE) +
  geom_line() +
  geom_point(data = sim$obs, aes(x = time, y = obs), inherit.aes = FALSE)
```

---

pkmod

*Create a pkmod object*

---

## Description

User function to create pkmod objects with validation checks. Multiple rows are permitted for arguments ‘pars\_pk’, ‘pars\_pd’, ‘init’

## Usage

```
pkmod(
  pars_pk = NULL,
  init = NULL,
  pkfn = NULL,
  pars_pd = NULL,
  pdfn = NULL,
  pdinv = NULL,
  pcmt = NULL,
  ecmt = NULL,
  sigma_add = 0,
  sigma_mult = 0,
  log_response = FALSE,
  Omega = NULL
)
```

**Arguments**

|              |                                                                                                                                                                                |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pars_pk      | Vector or matrix of named PK parameters. If not specified, the pkmod function will be inferred from the parameter names. Print 'list_parmns()' for acceptable parameter names. |
| init         | Vector of initial concentrations. Will default to values of zero in all compartments if not specified.                                                                         |
| pkfn         | PK model function. Functions provided in 'tci' include 'pkmod1cpt', 'pkmod2cpt', 'pkmod3cpt', and 'pkmod3cptm'. User-defined functions should be specified here.               |
| pars_pd      | PD model parameters if a PD model is specified                                                                                                                                 |
| pdfn         | PD model function                                                                                                                                                              |
| pdinv        | Inverse PD model function for use in TCI algorithms                                                                                                                            |
| pcmpt        | Index of plasma compartment. Defaults to first compartment if not specified.                                                                                                   |
| ecmpt        | Index of effect-site compartment if a PD model is specified. Will default to last compartment if unspecified.                                                                  |
| sigma_add    | Standard deviation of additive residual error                                                                                                                                  |
| sigma_mult   | Standard deviation of multiplicative residual error                                                                                                                            |
| log_response | Logical value indicating if the response should be logged prior to adding error.                                                                                               |
| Omega        | Optional matrix of random effect parameters. Column names should correspond to names of pars_pk, pars_pd, and sigma_add or sigma_mult.                                         |

**Value**

Returns a list with class "pkmod" if validation checks are passed. Returns an error if not.

**Examples**

```
# 1-compartment model
pkmod(pars_pk = c(CL = 10, V1 = 10))
# 2-compartment model
pkmod(pars_pk = c(CL = 10, V1 = 10, Q = 3, v2 = 20))
# 2-compartment model with random effects matrix
Omega <- matrix(diag(c(0.3,0.2,0,0.4)), 4,4, dimnames = list(NULL,c("CL","V1","Q","v2")))
pkmod(pars_pk = c(CL = 10, V1 = 10, Q = 3, v2 = 20), Omega = Omega)
```

---

pkmod1cpt

*One compartment IV infusion with first-order elimination.*

---

**Description**

One compartment IV infusion with first-order elimination.

**Usage**

```
pkmod1cpt(tm, kR, pars, init = 0)
```

**Arguments**

|      |                                                                    |
|------|--------------------------------------------------------------------|
| tm   | Vector of times to evaluate the PK function at                     |
| kR   | Infusion rate (e.g. ml/min).                                       |
| pars | Named vector of parameters with names ('k10','v1') or ('cl','v1'). |
| init | Initial concentration. Defaults to 0.                              |

**Value**

Numeric vector of concentrations for a constant infusion rate

**Examples**

```
pkmod1cpt(1,1,c(k10 = 0.5, v1 = 1))
pkmod1cpt(1,1,c(KE = 0.5, v1 = 1))
pkmod1cpt(1,1,c(CL = 0.5, v1 = 1))
```

---

|           |                                                                  |
|-----------|------------------------------------------------------------------|
| pkmod2cpt | <i>Two compartment IV infusion with first-order elimination.</i> |
|-----------|------------------------------------------------------------------|

---

**Description**

Two compartment IV infusion with first-order elimination. Elimination from peripheral compartment is assumed to be zero unless 'k20' is specified.

**Usage**

```
pkmod2cpt(tm, kR, pars, init = c(0, 0))
```

**Arguments**

|      |                                                                                              |
|------|----------------------------------------------------------------------------------------------|
| tm   | Vector of times to evaluate the PK function at                                               |
| kR   | Infusion rate (e.g. ml/min).                                                                 |
| pars | Named vector of parameters with names ('K10','K12','K21','V1','V2') or ('CL','Q','V1','V2'). |
| init | Initial concentration. Defaults to 0 in both compartments.                                   |

**Value**

Numeric matrix of concentrations for a constant infusion rate

**Examples**

```
pkmod2cpt(1,1,c(CL = 15, V1 = 10, Q2 = 10, V2 = 20))
pkmod2cpt(1,1,c(CL = 15, v1 = 10, Q2 = 10, V2 = 20, cl2 = 4))
```

---

pkmod3cpt

*Three compartment IV infusion with first-order elimination.*


---

### Description

Three compartment IV infusion with first-order elimination. Elimination is assumed to occur only from central compartment if 'k20', 'k30' are not specified.

### Usage

```
pkmod3cpt(tm, kR, pars, init = c(0, 0, 0))
```

### Arguments

|      |                                                                                              |
|------|----------------------------------------------------------------------------------------------|
| tm   | Vector of times to evaluate the PK function at                                               |
| kR   | Infusion rate (e.g. ml/min).                                                                 |
| pars | Named vector of parameters with names ('K10','K12','K21','V1','V2') or ('CL','Q','V1','V2'). |
| init | Initial concentration. Defaults to 0 in all compartments.                                    |

### Value

Numeric matrix of concentrations for a constant infusion rate

### Examples

```
pkmod3cpt(1,1,c(CL = 15, Q2 = 10, Q3 = 5, V1 = 10, V2 = 20, V3 = 50))
```

---

pkmod3cptm

*Solution to three-compartment IV model with effect-site*


---

### Description

3 compartment IV infusion with first-order absorption between compartments and with an additional effect-site compartment. The analytical solutions implemented in this function are provided in "ADVAN-style analytical solutions for common pharmacokinetic models" by Abuhelwa et al. 2015.

### Usage

```
pkmod3cptm(tm, kR, pars, init = c(0, 0, 0, 0))
```

**Arguments**

|      |                                                                          |
|------|--------------------------------------------------------------------------|
| tm   | Vector of times to evaluate the PK function at                           |
| kR   | Infusion rate (e.g. ml/min).                                             |
| pars | Named vector of parameters with names (k10,k12,k21,k13,k31,v1,v2,v3,ke0) |
| init | Initial concentration                                                    |

**Details**

This function takes in arguments for each of the absorption and elimination rate constants of a three-compartment model as well as initial concentrations, c0. ke0 gives the rate of elimination from the effect-site compartment into the central compartment (i.e. k41). The rate of absorption into the effect-site compartment is set at 1/10,000 the value of ke0. The function returns a set of functions that calculate the concentration in each of the four compartments as a function of time.

**Value**

Numeric matrix of concentrations for a constant infusion rate

**Examples**

```
pars_3cpt <- c(k10=1.5,k12=0.15,k21=0.09,k13=0.8,k31=0.8,v1=10,v2=15,v3=100,ke0=1)
pkmod3cptm(1,1,pars_3cpt)
```

---

pkmod\_eleveld\_ppf      *Eleveld population PK model for propofol*

---

**Description**

Function takes patient covariate values required for the Eleveld PK or PK-PD model for propofol and returns a 'pkmod' object with the appropriate model parameters.

**Usage**

```
pkmod_eleveld_ppf(
  AGE,
  TBW,
  HGT,
  MALE,
  OPIATE = TRUE,
  ARTERIAL = TRUE,
  PMA = NULL,
  PD = TRUE,
  ...
)
```

**Arguments**

|          |                                                                         |
|----------|-------------------------------------------------------------------------|
| AGE      | Age (years)                                                             |
| TBW      | Weight (kg)                                                             |
| HGT      | Height (cm)                                                             |
| MALE     | Sex, logical                                                            |
| OPIATE   | Logical indicating presence of opiates. Defaults to TRUE.               |
| ARTERIAL | PK based on arterial sampling rather than venous. Defaults to TRUE.     |
| PMA      | Post-menstrual age. Calculated as AGE + 40 weeks if not provided.       |
| PD       | Logical. Should PD parameters be returned in addition to PK parameters. |
| ...      | Arguments passed to 'pkmod'                                             |

**Value**

'pkmod' object with Eleveld propofol population PK or PK-PD parameters

**Examples**

```
pkmod_eleveld_ppf(AGE = 40, TBW = 56, HGT=150, MALE = TRUE)
```

---

|                    |                                                     |
|--------------------|-----------------------------------------------------|
| pkmod_eleveld_remi | <i>Eleveld population PK model for remifentanyl</i> |
|--------------------|-----------------------------------------------------|

---

**Description**

Function takes patient covariate values required for the Eleveld PK or PK-PD model for propofol and returns a 'pkmod' object with the appropriate model parameters.

**Usage**

```
pkmod_eleveld_remi(AGE, MALE, TBW, HGT = NULL, BMI = NULL, PD = TRUE, ...)
```

**Arguments**

|      |                                                                         |
|------|-------------------------------------------------------------------------|
| AGE  | Age (years)                                                             |
| MALE | Sex, logical                                                            |
| TBW  | Total body weight (kg).                                                 |
| HGT  | Height (cm). Used to calculate BMI if not provided.                     |
| BMI  | Body mass index                                                         |
| PD   | Logical. Should PD parameters be returned in addition to PK parameters. |
| ...  | Arguments passed to 'pkmod'                                             |

**Value**

'pkmod' object with Eleveld remifentanyl population PK or PK-PD parameters

**Examples**

```
pkmod_eleveld_remi(AGE = 40, TBW = 56, HGT=150, MALE = TRUE)
```

---

|           |                                                 |
|-----------|-------------------------------------------------|
| pkmod_kim | <i>Kim population PK model for remifentanil</i> |
|-----------|-------------------------------------------------|

---

**Description**

Evaluate Kim population PK model at patient covariate values. Published in Kim et al. (2017). "Disposition of Remifentanil in Obesity: A New Pharmacokinetic Model Incorporating the Influence of Body Mass" Anesthesiology Vol. 126, 1019–1032. doi: <https://doi.org/10.1097/ALN.0000000000001635>

**Usage**

```
pkmod_kim(AGE, TBW, HGT = NULL, BMI = NULL, MALE = NULL, FFM = NULL, ...)
```

**Arguments**

|      |                                                                |
|------|----------------------------------------------------------------|
| AGE  | Age (years)                                                    |
| TBW  | Total body weight (kg).                                        |
| HGT  | Height (cm). Used to calculate BMI if BMI is not provided.     |
| BMI  | Body mass index. Used to calculate LBM if LBM is not provided. |
| MALE | Logical. Used to calculate LBM if LBM is not provided.         |
| FFM  | Fat-free mass. Can be used instead of BMI and MALE.            |
| ...  | Arguments passed to 'pkmod'                                    |

**Value**

'pkmod' object with Schnider population PK parameters

**Examples**

```
pkmod_kim(AGE = 40, TBW = 75, BMI = 30, MALE = TRUE)
pkmod_kim(AGE = 40, TBW = 75, FFM = 52.83)
```

---

|             |                                                                                                           |
|-------------|-----------------------------------------------------------------------------------------------------------|
| pkmod_marsh | <div>Population</div> <div>PK and PK-PD functions</div> <div>Marsh population PK model for propofol</div> |
|-------------|-----------------------------------------------------------------------------------------------------------|

---

**Description**

Evaluates the Marsh propofol model at patient covariates (total body mass) and returns a ‘pkmod’ object. KE0 parameter set to 1.2 in accordance with recommendations from Absalom et al., 2009 "Pharmacokinetic models for propofol- Defining and illuminating the devil in the detail."

**Usage**

```
pkmod_marsh(TBW, ...)
```

**Arguments**

|     |                             |
|-----|-----------------------------|
| TBW | Weight (kg)                 |
| ... | Arguments passed to ‘pkmod’ |

**Value**

‘pkmod’ object with Marsh population PK parameters

**Examples**

```
pkmod_marsh(TBW = 50)
```

---

|             |                                            |
|-------------|--------------------------------------------|
| pkmod_minto | Minto population PK model for remifentanyl |
|-------------|--------------------------------------------|

---

**Description**

Evaluate Minto population PK model at patient covariate values. Published in Minto et al. (1997). "Influence of Age and Gender on the Pharmacokinetics and Pharmacodynamics of Remifentanyl: I. Model Development." Anesthesiology 86:10-23 doi: <https://doi.org/10.1097/00000542-199701000-00004>. Residual error standard deviations are taken from Eleveld et al. (2017). "An Allometric Model of Remifentanyl Pharmacokinetics and Pharmacodynamics." Anesthesiology Vol. 126, 1005–1018 doi: <https://doi.org/10.1097/ALN.0000000000001634>.



**Usage**

```
pkmod_minto(
  AGE,
  HGT = NULL,
  TBW = NULL,
  MALE = NULL,
  LBM = NULL,
  PD = TRUE,
  ...
)
```

**Arguments**

|      |                                                                         |
|------|-------------------------------------------------------------------------|
| AGE  | Age (years)                                                             |
| HGT  | Height (cm). Used to calculate LBM if LBM is not provided.              |
| TBW  | Weight (kg). Used to calculate LBM if LBM is not provided.              |
| MALE | Logical. Used to calculate LBM if LBM is not provided.                  |
| LBM  | Lean body mass (kg). Can be provided instead of TBW, HGT, and MALE      |
| PD   | Logical. Should PD parameters be returned in addition to PK parameters. |
| ...  | Arguments passed to 'pkmod'                                             |

**Value**

'pkmod' object with Schnider population PK parameters

**Examples**

```
pkmod_minto(AGE = 40, HGT=170, LBM = 43.9)
pkmod_minto(AGE = 40, HGT=170, TBW=50, MALE=TRUE)
```

---

|                |                                                  |
|----------------|--------------------------------------------------|
| pkmod_schnider | <i>Schnider population PK model for propofol</i> |
|----------------|--------------------------------------------------|

---

**Description**

Evaluate Schnider population PK model at patient covariate values. Published in Schnider et al. (1998). "The influence of method of administration and covariates on the pharmacokinetics of propofol in adult volunteers." *Anesthesiology* 88 (5):1170-82.

**Usage**

```
pkmod_schnider(AGE, HGT, LBM = NULL, TBW = NULL, MALE = NULL, ...)
```

**Arguments**

|      |                                                               |
|------|---------------------------------------------------------------|
| AGE  | Age (years)                                                   |
| HGT  | Height (cm)                                                   |
| LBM  | Lean body mass (kg). Can be provided instead of TBW, and MALE |
| TBW  | Weight (kg). Used to calculate LBM if LBM is not provided.    |
| MALE | Logical. Used to calculate LBM if LBM is not provided.        |
| ...  | Arguments passed to 'pkmod'                                   |

**Value**

'pkmod' object with Schnider population PK parameters

**Examples**

```
pkmod_schnider(AGE = 40,HGT=170,LBM = 43.9)
pkmod_schnider(AGE = 40,HGT=170,TBW=50,MALE=TRUE)
```

---

|              |                                      |
|--------------|--------------------------------------|
| plot.sim_tci | <i>Plot method for sim_tci class</i> |
|--------------|--------------------------------------|

---

**Description**

Plot object with class "sim\_tci" created by 'simulate\_tci()'.

**Usage**

```
## S3 method for class 'sim_tci'
plot(
  x,
  ...,
  yvar = NULL,
  id = NULL,
  type = c("true", "prior", "posterior"),
  show_inf = FALSE,
  show_data = FALSE,
  show_updates = FALSE,
  wrap_id = FALSE
)
```

**Arguments**

|      |                                                                                                                                          |
|------|------------------------------------------------------------------------------------------------------------------------------------------|
| x    | Object with class "sim_tci" created by 'simulate_tci()'                                                                                  |
| ...  | Other arguments. Not currently used.                                                                                                     |
| yvar | Response variable. Options are concentrations ("c1","c2",...) or "pdresp" for a PD response. Only one variable may be plotted at a time. |

|              |                                                                                                                       |
|--------------|-----------------------------------------------------------------------------------------------------------------------|
| id           | Subset of IDs to plot. Will default to all if unspecified. Can be displayed in separate plots via 'wrap_id' argument. |
| type         | Type of response to plot. Options are "prior", "true", or "posterior" if closed-loop control was used.                |
| show_inf     | Logical. Display infusion rates alongside response values.                                                            |
| show_data    | Logical. Display simulated data values in addition to responses.                                                      |
| show_updates | Logical, for closed-loop only. Show update times along x-axis.                                                        |
| wrap_id      | Logical. Separate plots by ID value.                                                                                  |

## Value

Plots simulation results

## Examples

```
data <- data.frame(ID = 1:2, AGE = c(30,40), TBW = c(70,80),
  HGT = c(160,170), MALE = c(FALSE,TRUE))
pkmod_prior <- poppkmod(data, drug = "ppf", model = "eleveld")
pkmod_true <- poppkmod(data, drug = "ppf", model = "eleveld", sample = TRUE)
obs_tms <- seq(1/6,10,1/6)
target_vals = c(75,60,50,50)
target_tms = c(0,3,6,10)

# open-loop simulation (without update_tms)
sim_ol <- simulate_tci(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms,
  seed = 200)
plot(sim_ol, id = 1, type = "true")
plot(sim_ol, yvar = "c4", type = "true")
plot(sim_ol, yvar = "c4", type = "true", wrap_id = TRUE, show_inf = TRUE)

# closed-loop simulation (with update_tms)
## Not run:
sim_cl <- simulate_tci(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms,
  update_tms = c(2,4,6), seed = 200)
plot(sim_cl, type = "posterior", id = 1, show_inf = TRUE)
plot(sim_cl, type = "posterior", wrap_id = TRUE, show_data = TRUE)
plot(sim_cl, yvar = "c4", wrap_id = TRUE)

## End(Not run)
```

## Description

Create a ‘poppkmod’ object using an existing population PK model for propofol or remifentanyl using patient covariates. Available models for propofol are the Marsh, Schnider, and Eleveld models. Available models for remifentanyl are the Minto, Kim, and Eleveld models. Input is a data frame with rows corresponding to individuals and columns recording patient covariates. An ID column is optional, but will be generated as 1:nrow(data) if not supplied. Covariates required by each model are

### Propofol

- Marsh: TBW
- Schnider: (AGE, HGT, TBW, MALE) or (AGE, HGT, LBW)
- Eleveld: AGE, TBW, HGT, MALE

### Remifentanyl

- Minto: (AGE, HGT, TBW, MALE) or (AGE, HGT, LBW)
- Kim: (AGE, TBW, BMI, HGT) or (AGE, TBW, FFM)
- Eleveld: (AGE, MALE, TBW, HGT) or (AGE, MALE, TBW, BMI)

### Abbreviations

- TBW = Total body weight (kg)
- LBW = Lean body weight (kg)
- FFM = Fat-free mass (kg)
- AGE = Age (years)
- HGT = Height (cm)
- MALE = Male (1/0, TRUE/FALSE)
- BMI = Body mass index ( $\text{kg}/\text{m}^2$ )

## Usage

```
poppkmod(
  data,
  drug = c("ppf", "remi"),
  model = c("marsh", "schnider", "eleveld", "minto", "kim"),
  sample = FALSE,
  PD = TRUE,
  ...
)
```

## Arguments

|      |                                                                                                  |
|------|--------------------------------------------------------------------------------------------------|
| data | Data frame of patient covariates. ID values, if used, should be in a column labeled "id" or "ID" |
| drug | "ppf" for propofol or "remi" for remifentanyl. Defaults to "ppf".                                |

|        |                                                                                                                                                                 |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| model  | Model name. Options are "marsh", "schnider", or "eleveld" if drug = "ppf", or "minto", "kim", or "eleveld" if drug = "remi".                                    |
| sample | Logical. Should parameter values be sampled from interindividual distribution (TRUE) or evaluated at point estimates for covariates (FALSE)? Defaults to FALSE. |
| PD     | Logical. If applicable, should the PD component be evaluated for PK-PD models. Defaults to TRUE.                                                                |
| ...    | Arguments passed on to each pkmod object                                                                                                                        |

**Value**

‘poppkmod’ object

**Examples**

```
data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),
  HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
poppkmod(data, drug = "ppf", model = "eleveld")
poppkmod(data, drug = "remi", model = "kim")
```

---

predict.pkmod

*Predict method for pkmod objects*

---

**Description**

Predict concentrations from a pkmod object - can be a user defined function

**Usage**

```
## S3 method for class 'pkmod'
predict(object, inf, tms, return_times = FALSE, ...)
```

**Arguments**

|              |                                                                                                                                                 |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| object       | An object with class pkmod.                                                                                                                     |
| inf          | A matrix with columns "begin", "end", "inf_rate" indicating when infusions should be administered. Can be created by ‘inf_manual’ or ‘inf_tci’. |
| tms          | Times at which to calculate predicted concentrations.                                                                                           |
| return_times | Logical. Should prediction times be returned along with responses? Defaults to FALSE.                                                           |
| ...          | List or vector of values to be passed on to update.pkmod.                                                                                       |

**Value**

Matrix of predicted concentrations associated with a pkmod object and and infusion schedule.

**Examples**

```
# dosing schedule
dose <- inf_manual(inf_tms = c(0,0.5,4,4.5,10), inf_rate = c(100,0,80,0,0))
# pkmod object
my_mod <- pkmod(pars_pk = c(CL = 15, V1 = 10, Q2 = 10, V2 = 20))
# predict at specific times
predict(my_mod, inf = dose, tms = c(1.5,2.5,3))
# predict with an Emax PD function
my_mod_pd <- pkmod(pars_pk = c(v1 = 8.995, v2 = 17.297, v3 = 120.963, cl = 1.382,
q2 = 0.919, q3 = 0.609, ke0 = 1.289),
pars_pd = c(c50 = 2.8, gamma = 1.47, gamma2 = 1.89, e0 = 93, emx = 93),
pdfn = emax, pdinv = emax_inv, ecmt = 4)
predict(my_mod_pd, inf = dose, tms = c(1.5,2.5,3))
# predict with a subset of new PK-PD parameters
predict(my_mod_pd, inf = dose, tms = c(1.5,2.5,3), pars_pk = c(ke0 = 0.8),
pars_pd = c(c50 = 2, e0 = 100))
```

---

|                  |                                         |
|------------------|-----------------------------------------|
| predict.poppkmod | <i>Predict method for pkmod objects</i> |
|------------------|-----------------------------------------|

---

**Description**

Predict concentrations from a pkmod object - can be a user defined function

**Usage**

```
## S3 method for class 'poppkmod'
predict(object, inf, tms, ...)
```

**Arguments**

|        |                                                                                                                                               |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| object | An object with class poppkmod.                                                                                                                |
| inf    | A matrix with columns "begin","end","inf_rate" indicating when infusions should be administered. Can be created by 'inf_manual' or 'inf_tci'. |
| tms    | Times at which to calculate predicted concentrations.                                                                                         |
| ...    | List or vector of values to be passed on to predict.pkmod.                                                                                    |

**Value**

Matrix of predicted concentrations associated with a poppkmod object and and infusion schedule.

**Examples**

```
# dosing schedule
dose <- inf_manual(inf_tms = c(0,0.5,4,4.5,10), inf_rate = c(100,0,80,0,0))
# poppkmod object
data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),
HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
```

```
elvd_mod <- poppkmod(data, drug = "ppf", model = "eleveld")
predict(elvd_mod, inf = dose, tms = c(1.5,2.5,3))
```

---

|             |                    |
|-------------|--------------------|
| print.pkmod | <i>Print pkmod</i> |
|-------------|--------------------|

---

## Description

Print method for pkmod objects

## Usage

```
## S3 method for class 'pkmod'
print(x, ..., digits = 3)
```

## Arguments

|        |                                                |
|--------|------------------------------------------------|
| x      | Object with class "pkmod" created by 'pkmod()' |
| ...    | Arguments passed to 'update.pkmod()'           |
| digits | Number of significant digits to print          |

## Value

Prints description of pkmod

## Examples

```
# 1-parameter model
pkmod(pars_pk = c(CL = 10, V1 = 10))
```

---

|                |                       |
|----------------|-----------------------|
| print.poppkmod | <i>Print poppkmod</i> |
|----------------|-----------------------|

---

## Description

Print method for poppkmod objects

## Usage

```
## S3 method for class 'poppkmod'
print(x, ..., digits = 3)
```

## Arguments

|        |                                                   |
|--------|---------------------------------------------------|
| x      | Object with class "pkmod" created by 'poppkmod()' |
| ...    | Additional arguments. Not used                    |
| digits | Number of significant digits to print             |

**Value**

Prints description of pkmod

**Examples**

```
data <- data.frame(ID = 1:5,
  AGE = seq(20,60,by=10),
  TBW = seq(60,80,by=5),
  HGT = seq(150,190,by=10),
  MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
poppkmod(data, drug = "ppf", model = "eleveld")
poppkmod(data, drug = "remi", model = "kim")
```

---

|               |                                       |
|---------------|---------------------------------------|
| print.sim_tci | <i>Print method for sim_tci class</i> |
|---------------|---------------------------------------|

---

**Description**

Print object with class "sim\_tci" created by 'simulate\_tci()'.

**Usage**

```
## S3 method for class 'sim_tci'
print(x, ...)
```

**Arguments**

|     |                                                         |
|-----|---------------------------------------------------------|
| x   | Object with class "sim_tci" created by 'simulate_tci()' |
| ... | Other arguments. Not currently used.                    |

**Value**

Prints a description of the simulation.

**Examples**

```
data <- data.frame(ID = 1:3, AGE = c(20,30,40), TBW = c(60,70,80),
  HGT = c(150,160,170), MALE = c(TRUE,FALSE,TRUE))
pkmod_prior <- poppkmod(data, drug = "ppf", model = "eleveld")
pkmod_true <- poppkmod(data, drug = "ppf", model = "eleveld", sample = TRUE)
obs_tms <- seq(1/6,10,1/6)
target_vals = c(75,60,50,50)
target_tms = c(0,3,6,10)

# open-loop simulation (without update_tms)
sim_ol <- simulate_tci(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms,
  seed = 200)

# closed-loop simulation (with update_tms)
```



```
## Not run:
sim_cl <- simulate_tci(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms,
  update_tms = c(2,4,6,8), seed = 200)

## End(Not run)
```

---

|            |                                                                                                      |
|------------|------------------------------------------------------------------------------------------------------|
| sample_iiv | <i>Sample PK or PK-PD parameters from the distribution of inter- or intra-individual variability</i> |
|------------|------------------------------------------------------------------------------------------------------|

---

## Description

Sample PK or PK-PD parameters from the distribution of random effects for a ‘pkmod’ or ‘poppkmod’ object, as described by the Omega matrix (see ?pkmod).

## Usage

```
sample_iiv(mod, log_normal = TRUE, ...)
```

## Arguments

|            |                                                                                                                                      |
|------------|--------------------------------------------------------------------------------------------------------------------------------------|
| mod        | ‘pkmod’ or ‘poppkmod’ object with associated Omega matrix describing random effect variances                                         |
| log_normal | Logical. Assumes random effects are log-normally distributed and multiplicative if TRUE, additive and normally distributed if FALSE. |
| ...        | Arguments passed to update.pkmod.                                                                                                    |

## Value

Returns model passed to "mod" argument with randomly sampled PK or PK-PD parameters.

## Examples

```
# sample from single PK model
sample_iiv(pkmod_schnider(AGE = 40, HGT=170, TBW=50, MALE=TRUE))
# sample from `poppkmod`
data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),
  HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
sample_iiv(poppkmod(data = data, drug = "ppf", model = "eleveld"))
```

---

|              |                                                |
|--------------|------------------------------------------------|
| sample_pkmod | <i>Sample parameters from a 'pkmod' object</i> |
|--------------|------------------------------------------------|

---

**Description**

Sample parameters from a 'pkmod' object

**Usage**

```
sample_pkmod(pkmod, log_normal = TRUE, ...)
```

**Arguments**

|            |                                                                                                                                      |
|------------|--------------------------------------------------------------------------------------------------------------------------------------|
| pkmod      | 'pkmod' object with associated Omega matrix describing random effect variances                                                       |
| log_normal | Logical. Assumes random effects are log-normally distributed and multiplicative if TRUE, additive and normally distributed if FALSE. |
| ...        | Arguments passed to update.pkmod                                                                                                     |

**Value**

'pkmod' object with updated parameters.

**Examples**

```
sample_pkmod(pkmod_schnider(AGE = 40, HGT=170, TBW=50, MALE=TRUE))
sample_pkmod(pkmod_eleveld_ppf(AGE = 40, TBW = 56, HGT=150, MALE = TRUE, PD = FALSE))
```

---

|                |                                          |
|----------------|------------------------------------------|
| simulate.pkmod | <i>Simulate method for pkmod objects</i> |
|----------------|------------------------------------------|

---

**Description**

Simulate observations from a pkmod object

**Usage**

```
## S3 method for class 'pkmod'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  ...,
  inf,
  tms,
  obs_cmt = 1,
  resp_bounds = NULL
)
```

**Arguments**

|             |                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| object      | An object with class "pkmod" generated by 'pkmod'                                                                                       |
| nsim        | Number of observations to simulate at each time point. Defaults to 1.                                                                   |
| seed        | An integer used to initialize the random number generator.                                                                              |
| ...         | Arguments passed to 'update.pkmod'.                                                                                                     |
| inf         | A matrix of infusion rates with columns 'begin', 'end', and 'inf_rate'. This can be created manually, by 'inf_manual', or by 'inf_tci'. |
| tms         | Times at which to simulate observations.                                                                                                |
| obs_cmpt    | Integer value indicating compartment in which observations are taken. Overridden if a PD model is included.                             |
| resp_bounds | Optional vector of two values indicating minimum and maximum values possible for the response.                                          |

**Examples**

```
# simulate data from a 2 compartment model with multiplicative error
dose <- inf_manual(inf_tms = c(0,0.5,4,4.5,10), inf_rate = c(100,0,80,0,0))
my_mod <- pkmod(pars_pk = c(CL = 10, V1 = 10, Q2 = 4, V2 = 30))
inf <- inf_tci(my_mod, target_vals = c(2,3,4,4), target_tms = c(0,2,3,10), "plasma")
simulate(my_mod, nsim = 3, inf = inf, tms = c(1,2,4,6,10), sigma_mult = 0.2)
# simulate with PD model
my_mod_pd <- pkmod(pars_pk = c(v1 = 8.995, v2 = 17.297, v3 = 120.963, cl = 1.382,
q2 = 0.919, q3 = 0.609, ke0 = 1.289),
pars_pd = c(c50 = 2.8, gamma = 1.47, gamma2 = 1.89, e0 = 93, emx = 93),
pdfn = emax, pdinv = emax_inv, ecmt = 4)
simulate(my_mod_pd, inf = dose, tms = c(1.5,2.5,3))
```

simulate.poppkmod

*Summary method for 'poppkmod' objects***Description**

Summarize parameter distribution Simulate method for poppkmod objects

**Usage**

```
## S3 method for class 'poppkmod'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  ...,
  inf,
  tms,
  obs_cmpt = 1,
  resp_bounds = NULL
)
```

**Arguments**

|             |                                                                                                                                         |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------|
| object      | An object with class "popppkmod" generated by 'popppkmod'                                                                               |
| nsim        | Number of observations to simulate at each time point. Defaults to 1.                                                                   |
| seed        | An integer used to initialize the random number generator.                                                                              |
| ...         | Arguments passed to 'update.pkmod'.                                                                                                     |
| inf         | A matrix of infusion rates with columns 'begin', 'end', and 'inf_rate'. This can be created manually, by 'inf_manual', or by 'inf_tci'. |
| tms         | Times at which to simulate observations.                                                                                                |
| obs_cmpt    | Integer value indicating compartment in which observations are taken. Overridden if a PD model is included.                             |
| resp_bounds | Optional vector of two values indicating minimum and maximum values possible for the response.                                          |

**Details**

Simulate observations from a popppkmod object

**Examples**

```
# simulate data from a 2 compartment model with multiplicative error
dose <- inf_manual(inf_tms = c(0,0.5,4,4.5,10), inf_rate = c(100,0,80,0,0))
# popppkmod object
data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),
  HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
elvd_mod <- popppkmod(data, drug = "ppf", model = "eleveld")
inf <- inf_tci(elvd_mod, target_vals = c(2,3,4,4), target_tms = c(0,2,3,10), "plasma")
simulate(elvd_mod, nsim = 3, inf = inf, tms = c(1,2,4,6,10))
```

---

simulate\_clc

*Simulate closed-loop control using Bayesian updates*

---

**Description**

Simulate closed-loop control using Bayesian updates for 'pkmod' or 'popppkmod' objects. Infusion rates are calculated using 'pkmod\_prior' to reach 'target\_vals' at 'target\_tms'. Data values are simulated using 'pkmod\_true' at 'obs\_tms'. 'pkmod\_prior' and 'pkmod\_true' do not need to have the same structure. Model parameters are updated at each update time using all available simulated observations. Processing delays can be added through the 'delay' argument, such that observations aren't made available to the update mechanism until 'update\_tms >= obs\_tms + delay'.

**Usage**

```
simulate_clc(
  pkmod_prior,
  pkmod_true,
  target_vals,
  target_tms,
  obs_tms,
  update_tms,
  type = c("effect", "plasma"),
  custom_alg = NULL,
  resp_bounds = NULL,
  delay = 0,
  seed = NULL,
  verbose = TRUE
)
```

**Arguments**

|             |                                                                                                                                                                                                                                                                                                                                                      |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkmod_prior | 'pkmod' or 'poppkmod' object describing a PK/PK-PD model that is used to calculate TCI infusion rates and is updated as data are simulated and incorporated. Must have an associated Omega matrix.                                                                                                                                                   |
| pkmod_true  | 'pkmod' or 'poppkmod' object describing the patient's "true" response. This model will be used to simulate observations.                                                                                                                                                                                                                             |
| target_vals | A vector of numeric values indicating PK or PD targets for TCI algorithm.                                                                                                                                                                                                                                                                            |
| target_tms  | A vector of numeric values indicating times at which the TCI algorithm should begin targeting each value.                                                                                                                                                                                                                                            |
| obs_tms     | Times at which data values should be simulated from 'pkmod_true'.                                                                                                                                                                                                                                                                                    |
| update_tms  | Times at which 'pkmod_prior' should be updated using all available simulated observations.                                                                                                                                                                                                                                                           |
| type        | Type of TCI algorithm to be used. Options are "plasma" and "effect". Defaults to "effect". Will be overwritten if 'custom_alg' is non-null.                                                                                                                                                                                                          |
| custom_alg  | Custom TCI algorithm to overwrite default plasma- or effect-site targeting.                                                                                                                                                                                                                                                                          |
| resp_bounds | Optional vector of two values indicating minimum and maximum values possible for the response.                                                                                                                                                                                                                                                       |
| delay       | Optional numeric value indicating a temporal delay between when observations are simulated and when they should be made available for updating 'pkmod_prior'. For example, a delay should be set to account for a processing time delay in Bispectral Index measurements or the time required to measure drug concentrations from collected samples. |
| seed        | An integer used to initialize the random number generator.                                                                                                                                                                                                                                                                                           |
| verbose     | Logical. Print progress as simulation is run.                                                                                                                                                                                                                                                                                                        |

**Examples**

```
data <- data.frame(ID = 1:3, AGE = c(20,30,40), TBW = c(60,70,80),
```

```

HGT = c(150,160,170), MALE = c(TRUE,FALSE,TRUE))
pkmod_prior <- poppkmod(data, drug = "ppf", model = "eleveld")
pkmod_true <- poppkmod(data, drug = "ppf", model = "eleveld", sample = TRUE)
obs_tms <- seq(1/6,10,1/6)
update_tms <- c(2,4,6,8)
target_vals = c(75,60,50,50)
target_tms = c(0,3,6,10)
## Not run:
sim <- simulate_clc(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms,
update_tms, seed = 200)
len <- 500
tms <- seq(0,10,length.out = len)
resp <- data.frame(rbind(predict(pkmod_true, sim$inf, tms),
predict(pkmod_prior, sim$inf, tms),
predict(sim$pkmod_post, sim$inf, tms)))
resp$type = c(rep("true",len*3),rep("prior",len*3),rep("posterior",len*3))
library(ggplot2)
ggplot(resp) + geom_line(aes(x = time, y = pdresp, color = factor(id))) +
facet_wrap(~type) + labs(x = "Hours", y = "Bispectral Index") +
  geom_step(data = data.frame(time = target_tms, value = target_vals),
  aes(x = time, y = value), inherit.aes = FALSE)

## End(Not run)

```

---

simulate\_olc

Simulate open-loop control using TCI

---

## Description

Simulate open-loop control using TCI for ‘pkmod’ or ‘poppkmod’ objects. Infusion rates are calculated using ‘pkmod\_prior’ to reach ‘target\_vals’ at ‘target\_tms’. Data values are simulated using ‘pkmod\_true’ at ‘obs\_tms’. ‘pkmod\_prior’ and ‘pkmod\_true’ do not need to have the same structure, but are required to have the same number of IDs (i.e., N) if ‘poppkmod’ objects are used.

## Usage

```

simulate_olc(
  pkmod_prior,
  pkmod_true,
  target_vals,
  target_tms,
  obs_tms,
  type = c("effect", "plasma"),
  custom_alg = NULL,
  resp_bounds = NULL,
  seed = NULL
)

```

**Arguments**

|             |                                                                                                                                                                                      |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkmod_prior | 'pkmod' object describing a PK/PK-PD model that is used to calculate TCI infusion rates and is updated as data are simulated and incorporated. Must have an associated Omega matrix. |
| pkmod_true  | 'pkmod' object describing the patient's "true" response. This model will be used to simulate observations.                                                                           |
| target_vals | A vector of numeric values indicating PK or PD targets for TCI algorithm.                                                                                                            |
| target_tms  | A vector of numeric values indicating times at which the TCI algorithm should begin targeting each value.                                                                            |
| obs_tms     | Times at which data values should be simulated from 'pkmod_true'.                                                                                                                    |
| type        | Type of TCI algorithm to be used. Options are "plasma" and "effect". Defaults to "effect". Will be overwritten if 'custom_alg' is non-null.                                          |
| custom_alg  | Custom TCI algorithm to overwrite default plasma- or effect-site targeting.                                                                                                          |
| resp_bounds | Optional vector of two values indicating minimum and maximum values possible for the response.                                                                                       |
| seed        | An integer used to initialize the random number generator.                                                                                                                           |

**Examples**

```

data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),
  HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))
pkmod_prior <- poppkmod(data, drug = "ppf", model = "eleveld")
pkmod_true <- poppkmod(data, drug = "ppf", model = "eleveld", sample = TRUE)
obs_tms <- seq(1/6,10,1/6)
target_vals = c(75,60,50,50)
target_tms = c(0,3,6,10)
sim <- simulate_olc(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms)
len <- 500
tms <- seq(0,10,length.out = len)
resp <- data.frame(rbind(predict(pkmod_true, sim$inf, tms),
  predict(pkmod_prior, sim$inf, tms)))
resp$type = c(rep("true",len*5),rep("prior",len*5))
library(ggplot2)
ggplot(resp) + geom_line(aes(x = time, y = pdresp, color = factor(id))) + facet_wrap(~type) +
  labs(x = "Hours", y = "Bispectral Index") + theme_bw() +
  geom_step(data = data.frame(time = target_tms, value = target_vals),
    aes(x = time, y = value), inherit.aes = FALSE)

```

## Description

Simulate responses from a 'pkmod' or 'popppkmod' object using TCI control. Infusion rates are calculated to reach targets using 'pkmod\_prior'. Data values are simulated using 'pkmod\_true'. 'pkmod\_prior' and 'pkmod\_true' do not need to have the same parameters or structure and should be different when simulating responses under model misspecification. If update times (argument 'update\_tms') are provided then the function 'simulate\_clc' is called for "closed-loop" control and model parameters will be updated via Bayes' rule using available data. Only parameters with values in the Omega matrix will be updated. A data processing delay can be added through the argument 'delay'. See '?bayes\_update?' for more details. If update times are not specified, then the function 'simulate\_olc' will be called to implement "open-loop" control and model parameters will not be updated. Simulation results have class 'tci\_sim' and can be plotted using 'plot.tci\_sim()'.

## Usage

```
simulate_tci(
  pkmod_prior,
  pkmod_true,
  target_vals,
  target_tms,
  obs_tms,
  update_tms = NULL,
  type = c("effect", "plasma"),
  custom_alg = NULL,
  resp_bounds = NULL,
  delay = 0,
  seed = NULL,
  verbose = TRUE
)
```

## Arguments

|             |                                                                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pkmod_prior | 'pkmod' or 'popppkmod' object describing a PK/PK-PD model that is used to calculate TCI infusion rates and is updated as data are simulated and incorporated. Must have an associated Omega matrix. |
| pkmod_true  | 'pkmod' or 'popppkmod' object describing the patient's "true" response. This model will be used to simulate observations.                                                                           |
| target_vals | A vector of numeric values indicating PK or PD targets for TCI algorithm.                                                                                                                           |
| target_tms  | A vector of numeric values indicating times at which the TCI algorithm should begin targeting each value.                                                                                           |
| obs_tms     | Times at which data values should be simulated from 'pkmod_true'.                                                                                                                                   |
| update_tms  | Times at which 'pkmod_prior' should be updated using all available simulated observations.                                                                                                          |
| type        | Type of TCI algorithm to be used. Options are "plasma" and "effect". Defaults to "effect". Will be overwritten if 'custom_alg' is non-null.                                                         |
| custom_alg  | Custom TCI algorithm to overwrite default plasma- or effect-site targeting.                                                                                                                         |



|             |                                                                                                                                                                                                                                                                                                                                                      |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| resp_bounds | Optional vector of two values indicating minimum and maximum values possible for the response.                                                                                                                                                                                                                                                       |
| delay       | Optional numeric value indicating a temporal delay between when observations are simulated and when they should be made available for updating 'pkmod_prior'. For example, a delay should be set to account for a processing time delay in Bispectral Index measurements or the time required to measure drug concentrations from collected samples. |
| seed        | An integer used to initialize the random number generator.                                                                                                                                                                                                                                                                                           |
| verbose     | Logical. Print progress as simulation is run.                                                                                                                                                                                                                                                                                                        |

### Examples

```
data <- data.frame(ID = 1:3, AGE = c(20,30,40), TBW = c(60,70,80),
  HGT = c(150,160,170), MALE = c(TRUE,FALSE,TRUE))
pkmod_prior <- poppkmod(data, drug = "ppf", model = "eleveld")
pkmod_true <- poppkmod(data, drug = "ppf", model = "eleveld", sample = TRUE)
obs_tms <- seq(1/6,10,1/6)
target_vals = c(75,60,50,50)
target_tms = c(0,3,6,10)

# open-loop simulation (without updates)
sim_ol <- simulate_tci(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms,
  seed = 200)
plot(sim_ol)

# closed-loop simulation (with updates)
## Not run:
sim_cl <- simulate_tci(pkmod_prior, pkmod_true, target_vals, target_tms, obs_tms,
  update_tms = c(2,4,6,8), seed = 200)
plot(sim_cl, wrap_id = TRUE, show_inf = TRUE, show_data = TRUE)

## End(Not run)
```

---

tci\_documentation      tci\_documentation

---

### Description

Functions to implement target-controlled infusion algorithms

### Details

tci package documentation

This package contains functions to implement target-controlled infusion (TCI) algorithms for compartmental PK models under intravenous administration. TCI algorithms for plasma or effect-site targeting are included and can be extended to pharmacodynamic responses. Custom PK-PD models and custom TCI algorithms can be specified. Functions are provided to simulate responses from PK/PK-PD models under open- or closed-loop control.

---

|            |                                                                                     |
|------------|-------------------------------------------------------------------------------------|
| tci_effect | <i>Effect-site TCI algorithm with plasma targeting within small range of target</i> |
|------------|-------------------------------------------------------------------------------------|

---

## Description

Modified effect-site TCI algorithm that switches to plasma-targeting when the plasma concentration is within 20% of the target and the effect-site concentration is within 0.5% of the target. The modification decreases computation time and prevents oscillatory behavior in the effect-site concentrations.

## Usage

```
tci_effect(Ct, pkmod, dtm = 1/6, cptol = 0.2, cetol = 0.05, ...)
```

## Arguments

|       |                                                                                                                          |
|-------|--------------------------------------------------------------------------------------------------------------------------|
| Ct    | Numeric vector of target effect-site concentrations.                                                                     |
| pkmod | PK model                                                                                                                 |
| dtm   | TCI update frequency. Defaults to 1/6, corresponding to 10-second intervals if model parameters are in terms of minutes. |
| cptol | Percentage of plasma concentration required to be within to switch to plasma targeting.                                  |
| cetol | Percentage of effect-site concentration required to be within to switch to plasma targeting.                             |
| ...   | Arguments passed on to 'tci_plasma' and 'tci_effect_only' functions, including to update.pkmod.                          |

## Value

Numeric value

## Examples

```
my_mod <- pkmod(pars_pk = c(v1 = 8.995, v2 = 17.297, v3 = 120.963, c1 = 1.382,
q2 = 0.919, q3 = 0.609, ke0 = 1.289))
tci_effect(Ct = 2, pkmod = my_mod)
# update parameters
tci_effect(Ct = 2, pkmod = my_mod, pars_pk = c(v1 = 12, c1 = 2))
```

---

|                 |                                                |
|-----------------|------------------------------------------------|
| tci_effect_only | <i>TCI algorithm for effect-site targeting</i> |
|-----------------|------------------------------------------------|

---

### Description

Function for calculating a TCI infusion schedule corresponding to a set of target concentrations. This function makes use of formulas described by Shafer and Gregg (1992) in "Algorithms to rapidly achieve and maintain stable drug concentrations at the site of drug effect with a computer-controlled infusion pump"

### Usage

```
tci_effect_only(
  Ct,
  pkmod,
  dtm = 1/6,
  tmax_search = 10,
  maxrt = 1200,
  grid_len = 1200,
  ...
)
```

### Arguments

|             |                                                                                                                                                                                              |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ct          | Numeric vector of target effect-site concentrations.                                                                                                                                         |
| pkmod       | PK model                                                                                                                                                                                     |
| dtm         | Frequency of TCI updates. Default is 1/6 minutes = 10 seconds.                                                                                                                               |
| tmax_search | Outer bound on times searched to find a maximum concentration following an infusion of duration dtm. Defaults to 20 minutes. May need to be increased if a drug has a slow elimination rate. |
| maxrt       | Maximum infusion rate of TCI pump. Defaults to 1200.                                                                                                                                         |
| grid_len    | Number of time points used to identify time of maximum concentration. Can be increased for more precision.                                                                                   |
| ...         | List or vector of named values to be passed on to update.pkmod.                                                                                                                              |

### Value

Numeric value

### Examples

```
# three compartment model with effect site
my_mod <- pkmod(pars_pk = c(v1 = 8.995, v2 = 17.297, v3 = 120.963,
  cl = 1.382, q2 = 0.919, q3 = 0.609, ke0 = 1.289))
tci_effect_only(Ct = 2, pkmod = my_mod)
# update parameters
tci_effect_only(Ct = 2, pkmod = my_mod, pars_pk = c(v1 = 12, cl = 2))
```

---

|             |                                           |
|-------------|-------------------------------------------|
| tc_i_plasma | <i>TCI algorithm for plasma targeting</i> |
|-------------|-------------------------------------------|

---

**Description**

TCI algorithm based on the algorithm described by Jacobs (1990).

**Usage**

```
tc_i_plasma(Ct, pkmod, dtm = 1/6, maxrt = 1200, ...)
```

**Arguments**

|       |                                                                                                                                                                     |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Ct    | Target plasma concentration                                                                                                                                         |
| pkmod | PK model                                                                                                                                                            |
| dtm   | Duration of the infusion                                                                                                                                            |
| maxrt | Maximum infusion rate. Defaults to 200 ml/min in reference to the maximum infusion rate of 1200 ml/h permitted by existing TCI pumps (e.g. Anestfusor TCI program). |
| ...   | Arguments passed on to update.pkmod.                                                                                                                                |

**Value**

Numeric value

**Examples**

```
# plasma targeting
my_mod <- pkmod(pars_pk = c(CL = 10, V1 = 10))
tc_i_plasma(Ct = 2, my_mod)
# update CL parameter
tc_i_plasma(Ct = 2, my_mod, pars_pk = c(CL = 15))
```

---

|              |                                |
|--------------|--------------------------------|
| update.pkmod | <i>Update method for pkmod</i> |
|--------------|--------------------------------|

---

**Description**

Update parameters or initial values of a pkmod object

**Usage**

```
## S3 method for class 'pkmod'
update(object, ...)
```

**Arguments**

|        |                                  |
|--------|----------------------------------|
| object | Object with class pkmod          |
| ...    | Updated values passed to object. |

**Value**

Returns the pkmod object with elements replaced

**Examples**

```
# initial pkmod object
(my_mod <- pkmod(pars_pk = c(CL = 10, V1 = 10)))
# update a subset of parameters and initial values
update(my_mod, pars_pk = c(CL = 20, V1 = 2), init = 3, sigma_add = 1, log_response = TRUE)
```

---

|                |                                |
|----------------|--------------------------------|
| validate_pkmod | <i>pkmod validation checks</i> |
|----------------|--------------------------------|

---

**Description**

Function to provide validation checks for a pkmod object

**Usage**

```
validate_pkmod(x)
```

**Arguments**

|   |                         |
|---|-------------------------|
| x | Object of class "pkmod" |
|---|-------------------------|

**Value**

Returns a list with class "pkmod" if validation checks are passed. Returns an error if not.

**Examples**

```
validate_pkmod(init_pkmod(pars_pk = c(CL = 10, V1 = 10, Q2 = 4, V2=20)))
```

---

|                   |                                                         |
|-------------------|---------------------------------------------------------|
| validate_poppkmod | <i>Perform validation checks on a 'poppkmod' object</i> |
|-------------------|---------------------------------------------------------|

---

**Description**

Perform validation checks on a 'poppkmod' object created by 'init\_poppkmod'.

**Usage**

```
validate_poppkmod(x)
```

**Arguments**

|   |                                                       |
|---|-------------------------------------------------------|
| x | Object with class "poppkmod" created by init_poppkmod |
|---|-------------------------------------------------------|

**Value**

'poppkmod' object or error

**Examples**

```
data <- data.frame(ID = 1:5, AGE = seq(20,60,by=10), TBW = seq(60,80,by=5),  
HGT = seq(150,190,by=10), MALE = c(TRUE,TRUE,FALSE,FALSE,FALSE))  
validate_poppkmod(init_poppkmod(data, drug = "ppf", model = "eleveld"))
```

# Index

## \* datasets

    eleveld\_pd, [8](#)  
    eleveld\_pk, [9](#)

apply\_tci, [3](#)  
assign\_pars, [4](#)

bayes\_update, [5](#)

clc, [6](#)

eleveld\_pd, [8](#)  
eleveld\_pk, [9](#)  
elvdparams, [10](#)  
emax, [11](#)  
emax\_eleveld, [11](#)  
emax\_inv, [12](#)  
emax\_inv\_eleveld, [12](#)  
emax\_inv\_remi, [13](#)  
emax\_remi, [13](#)

format\_pars, [14](#)

inf\_manual, [15](#)  
inf\_tci, [16](#)  
infer\_pkfn, [15](#)  
init\_pkmod, [17](#)  
init\_poppkmod, [19](#)

list\_parnms, [20](#)  
list\_pkmods, [20](#)  
log\_likelihood, [21](#)  
log\_posterior\_neg, [22](#)  
log\_prior, [23](#)

olc, [24](#)

pkmod, [25](#)  
pkmod1cpt, [26](#)  
pkmod2cpt, [27](#)  
pkmod3cpt, [28](#)

pkmod3cptm, [28](#)  
pkmod\_eleveld\_ppf, [29](#)  
pkmod\_eleveld\_remi, [30](#)  
pkmod\_kim, [31](#)  
pkmod\_marsh, [32](#)  
pkmod\_minto, [32](#)  
pkmod\_schnider, [33](#)  
plot.sim\_tci, [34](#)  
poppkmod, [35](#)  
predict.pkmod, [37](#)  
predict.poppkmod, [38](#)  
print.pkmod, [39](#)  
print.poppkmod, [39](#)  
print.sim\_tci, [40](#)

sample\_iiv, [41](#)  
sample\_pkmod, [42](#)  
simulate.pkmod, [42](#)  
simulate.poppkmod, [43](#)  
simulate\_clc, [44](#)  
simulate\_olc, [46](#)  
simulate\_tci, [47](#)

tci\_documentation, [49](#)  
tci\_effect, [50](#)  
tci\_effect\_only, [51](#)  
tci\_plasma, [52](#)

update.pkmod, [52](#)

validate\_pkmod, [53](#)  
validate\_poppkmod, [54](#)