

# Package ‘CalciOMatic’

January 2, 2012

**Type** Package

**Title** Automatic Calcium Imaging Analysis

**Version** 1.1-3

**Date** 2009-10-06

**Author** Sebastien Joucla, Christophe Pouzat

**Maintainer** Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**Description** Simulate and analyse calcium imaging data obtained with ratiometric dyes

**License** GPL (>= 2)

**Depends** cobs

**URL** <http://sites.google.com/site/sebastienjoucla>

**Repository** CRAN

**Date/Publication** 2009-10-16 14:04:41

## R topics documented:

CalciOMatic-package . . . . .	2
anova4Fits . . . . .	3
caBiExp . . . . .	5
caFromDf . . . . .	7
caFromRatio . . . . .	9
caMonoBiExpFromIG . . . . .	11
caMonoExp . . . . .	13
directFit . . . . .	14
fluo . . . . .	18
getThetaRatio . . . . .	20
igDirect . . . . .	22
igRatio . . . . .	26
inVitro . . . . .	28

mkFluo4DirectFit . . . . .	30
mkFunction4DirectFit . . . . .	32
mkFunction4RatioFit . . . . .	34
plot.direct_fit . . . . .	36
plot.fluo_rawdata . . . . .	38
plot.ratio_fit . . . . .	40
plot.ratio_fit_list . . . . .	42
plotCalciOMatic . . . . .	45
ratioExpPhysio . . . . .	48
ratioExpSimul . . . . .	51
ratioFitFromCa . . . . .	54
ratioFitFromDf . . . . .	56
transientConvexPart . . . . .	59

<b>Index</b>	<b>62</b>
--------------	-----------

---

CalciOMatic-package      *Automatic Calcium Imaging Analysis*

---

## Description

Simulate and analyse calcium imaging data obtained with ratiometric dyes. The package provides tools to fit parametric models of calcium dynamics on experimental data. Two methods are available: the classical 'ratiometric' method and a new 'direct' method, which does not imply any data ratioing and fits directly the fluorescence transients recorded at two excitation wavelengths. The latter method allows for the construction of meaningful confidence intervals on the calcium dynamics parameters.

## Details

Package: CalciOMatic  
 Type: Package  
 Version: 1.1-3  
 Date: 2009-10-06  
 License: GPL (>= 2)  
 Depends: cobs

## Author(s)

Sebastien Joucla, Christophe Pouzat

Maintainer: Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

## References

Joucla S, Pippow A, Kloppenburg P and Pouzat C (2009) Quantitative estimation of calcium dynamics from ratiometric measurements: A direct, non-ratioing, approach. J Neurophysiol, in revision

## Examples

```
## Load the data set from cockroach olfactory interneurons
data(inVitro)

## Define the calibrated parameters of the calcium indicator (Fura-2)
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d   <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Characteristics of the CCD camera, obtained from 'calibration' experiments
G     <- 0.146
s_ro  <- 16.4

## Create the data.frame containing the physiological data:
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=G, s_ro=s_ro,
                             alphamethod=TRUE)

## Fit the physiological data with the direct method:
physioDirectFit <- directFit(physioData,
                             transients=2,
                             Sqrt=TRUE,
                             type="mono",
                             AfterPeak=14)

## Plot the raw and fitted data, as well as plots of goodness of fit
plot(physioDirectFit, numTransient=2, items=1:6)
```

---

 anova4Fits

*Perform an ANalysis Of VAriance between two fit objects*


---

## Description

The function `Anova_4_Fits` performs an ANOVA between two objects inheriting from the "nls" class, in order to determine which one best fits the raw data.

## Usage

```
anova4Fits(Fit_1, Fit_2)
```

**Arguments**

Fit\_1            the first "nls" object to compare  
Fit\_2            the second "nls" object to compare

**Details**

The residual sum of squares of both models are compared, the least of both tells which model is the most appropriate to fit the raw data.

**Value**

An integer (1 or 2) indicating which model best fits the raw data.

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[directFit](#)

**Examples**

```
## Parameters of the biexponential calcium transient
t0n <- 1
Time <- seq(0,30,0.1)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5
mu <- 0
dtau <- 10

## Calibration parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Experiment-specific parameters
nb_B <- 5
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 1000
P_B <- 1000
phi <- 1.25
S_B_340 <- 100/P/T_340
S_B_380 <- 100/P/T_380

## Create a biexponential calcium decay
Ca_Bi <- caBiExp(t = Time, t0n = t0n,
```

```

Ca0 = Ca0, dCa = dCa, tau = tau,
fact=1/(1+exp(-mu)), dtau = dtau)

## Simulate the corresponding ratiometric experiment
df_Bi <- ratioExpSimul(nb_B   = nb_B,
                      Ca     = Ca_Bi,
                      R_min  = R_min,
                      R_max  = R_max,
                      K_eff  = K_eff,
                      K_d    = K_d,
                      B_T    = B_T,
                      phi    = phi,
                      S_B_340 = S_B_340,
                      S_B_380 = S_B_380,
                      T_340  = T_340,
                      T_380  = T_380,
                      P      = P,
                      P_B    = P_B,
                      ntransients = 1,
                      G      = 1,
                      s_ro   = 0)

## Perform a monoexponential and a biexponential direct fit
direct_fit_mono <- directFit(df = df_Bi,
                           transients = 1,
                           SQRT = TRUE,
                           ratio = NULL,
                           type = "mono")

direct_fit_bi   <- directFit(df = df_Bi,
                           transients = 1,
                           SQRT = TRUE,
                           ratio = NULL,
                           type = "bi")

## Test which model ('mono' or 'bi') bests predicts the 'experimental' data
idx <- anova4Fits(Fit_1 = direct_fit_mono, Fit_2 = direct_fit_bi)
print(idx)

```

---

caBiExp

*BiExponential Time Course of Intracellular Calcium Concentration*


---

## Description

The function `caBiExp` returns a vector of intracellular calcium concentration ( $[Ca^{2+}]$ ) vs time values  $t$ . A  $dCa$  jump occurs at  $t_0n$ , followed by a biexponential return to baseline value  $Ca_0$ . The fast time constant ( $\tau$ ) has a weight  $fact$  (between 0 and 1), the slow time constant ( $\tau+d\tau$ ) has a weight  $1-fact$ .

**Usage**

```
caBiExp(t = 1, t0n = 1, Ca0 = 0.05, dCa = 0.1, tau = 3, fact = 1, dtau = 2)
```

**Arguments**

t	a vector of time values at which $[Ca^{2+}]$ is computed (in s)
t0n	the time of the $[Ca^{2+}]$ jump (in s)
Ca0	the baseline $[Ca^{2+}]$ (in $\mu M$ )
dCa	the $[Ca^{2+}]$ jump occurring at t0n (in $\mu M$ )
tau	the fast time constant of the $[Ca^{2+}]$ biexponential return to baseline (in s)
fact	the relative weight of the fast time constant of the biexponential decay (a real number between 0 and 1). The relative weight of the slow time constant is given by 1-fact
dtau	added to tau, defines the slow time constant of the $[Ca^{2+}]$ biexponential return to baseline (in s)

**Value**

A vector containing the  $[Ca^{2+}]$  values. The vector has the two following attributes:

Time	a copy of argument t
t0n	a copy of argument t0n

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[caMonoExp](#), [caMonoBiExpFromIG](#)

**Examples**

```
## Simulate a biexponential calcium transient
Ca <- caBiExp(t = seq(0,20,0.1),
             t0n = 2,
             Ca0 = 0.25,
             dCa = 1,
             tau = 2,
             fact = 0.7,
             dtau = 8)

## Plot the calcium transient vs. time
plot(attr(Ca,"Time"), Ca, type="l")

## Add a vertical dashed line at t0n
abline(v = attr(Ca,"t0n"), lty = 2)
```

---

caFromDf *Get Calcium Concentration From a Fluorescence Data Frame, Using the Ratiometric Transformation*

---

### Description

The function `caFromDf` applies the ratiometric transformation to the vectors of fluorescence (including background fluorescence) contained in a data frame and returns the corresponding intracellular calcium concentration. The structure of the data frame is defined in the [ratioExpSimul](#) function.

### Usage

```
caFromDf(df, numTransient = 1, Plot = FALSE)
```

### Arguments

`df` a data frame of class "fluo\_rawdata" containing all relevant information (fluorescence transients, background fluorescence, calibration parameters and exposure times). The structure of the input data frame is defined in the [ratioExpSimul](#)

`numTransient` an integer: The index of the transient to analyse in the input data frame `df`

`Plot` a logical value: Set to TRUE to plot the calcium transient deduced from the ratiometric transformation

### Details

The way  $[Ca^{2+}]$  is estimated by the ratiometric transformation is described in the help of the [caFromRatio](#) function.

### Value

A vector of intracellular calcium concentration ratiometric transformation.

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[ratioExpSimul](#), [ratioExpPhysio](#), [caFromRatio](#)

### Examples

```
## (0) 'Experimental' parameters

## Parameters of the monoexponential calcium transient
tOn <- 1
Time <- seq(0,10,0.1)
Ca0 <- 0.10
```

```

dCa <- 0.25
tau <- 1.5

## Calibration parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Experiment-specific parameters
nb_B <- 5
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 1000
P_B <- 1000
phi <- 1.25
S_B_340 <- 100/P/T_340
S_B_380 <- 100/P/T_380

## (1) Create a monoexponential calcium decay
Ca_Mono <- caMonoExp(t = Time, t0n = t0n,
                    Ca0 = Ca0, dCa = dCa, tau = tau)

## (2) Simulate the corresponding ratiometric experiment
df_Mono <- ratioExpSimul(nb_B = nb_B,
                        Ca = Ca_Mono,
                        R_min = R_min,
                        R_max = R_max,
                        K_eff = K_eff,
                        K_d = K_d,
                        B_T = B_T,
                        phi = phi,
                        S_B_340 = S_B_340,
                        S_B_380 = S_B_380,
                        T_340 = T_340,
                        T_380 = T_380,
                        P = P,
                        P_B = P_B,
                        ntransients = 1,
                        G = 1,
                        s_ro = 0)

## (3) Get the noisy calcium transient from the data frame
Ca_noisy <- caFromDf(df = df_Mono,
                   numTransient = 1,
                   Plot = FALSE)

## (4) Plot the simulated noisy calcium transient
## over the ideal calcium transient
## plot(attr(Ca_noisy,"Time"), Ca_noisy, type = "l", col = "blue")
## lines(Time, Ca_Mono, col="red", lwd = 2)

```

```
## abline(v = attr(Ca_noisy,"t0n"), lty = 2)
```

---

caFromRatio	<i>Get Calcium Concentration From Fluorescence Signals, Using the Ratiometric Transformation</i>
-------------	--

---

### Description

The function `caFromRatio` applies the ratiometric transformation to vectors of fluorescence (including background fluorescence) and returns the corresponding intracellular calcium concentration.

### Usage

```
caFromRatio(adu_B_340, adu_340,
            adu_B_380, adu_380,
            T_340 = 0.015, T_380 = 0.006,
            P, P_B,
            R_min = 0.136, R_max = 2.701, K_eff = 3.637,
            Plot = FALSE)
```

### Arguments

<code>adu_B_340</code>	a vector of background fluorescence values (photon counts) recorded at 340 nm
<code>adu_340</code>	a vector of fluorescence values recorded at 340 nm
<code>adu_B_380</code>	a vector of background fluorescence values recorded at 380 nm
<code>adu_380</code>	a vector of fluorescence values recorded at 380 nm
<code>T_340</code>	the exposure time at 340 nm (in s)
<code>T_380</code>	the exposure time at 380 nm (in s)
<code>P</code>	the number of pixels of the Region Of Interest (ROI)
<code>P_B</code>	the number of pixels of the Background Region
<code>R_min</code>	the minimum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
<code>R_max</code>	the maximum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
<code>K_eff</code>	the effective dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
<code>Plot</code>	a logical value: Set to TRUE to plot the calcium transient estimated by the ratiometric transformation

## Details

The calcium imaging technique makes use of the ability of a fluorescent dye (e.g. Fura) to bind with calcium ions present inside the cell. Briefly, the photons emitted by the calcium-free and calcium-bound forms of the dye are recorded by a CCD camera, following the illumination of the tissue by a light source at relevant wavelengths (corresponding to maxima of excitation of the free and bound forms of the dye). In the case of a ratiometric dye, an algebraic relationship links the intracellular calcium concentration to the photon counts at both wavelengths (340 and 380 nm, in the case of Fura-2). It is thus possible to retrieve the intracellular calcium concentration from the ratio of the photon counts recorded at these two wavelengths (after subtraction of the background fluorescence): This is the ratiometric transformation. The ratio  $R$  is defined as:

$$R = \frac{P^{-1} \cdot adu_{340} - P_B^{-1} \cdot adu_{B,340}}{P^{-1} \cdot adu_{380} - P_B^{-1} \cdot adu_{B,380}} \cdot \frac{T_{380}}{T_{340}} = \frac{R_{min} \cdot K_{eff} + R_{max} \cdot [Ca^{2+}]}{K_{eff} + [Ca^{2+}]}$$

Then, the intracellular calcium concentration is given by:

$$[Ca^{2+}] = K_{eff} \cdot \frac{R - R_{min}}{R_{max} - R}$$

## Value

A vector of intracellular calcium concentration calculated with the ratiometric transformation described above. This vector comes with the estimated covariance matrix as attribute (see Joucla et al. (2009) for more details).

## Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

## References

Joucla S, Pippow A, Kloppenburg P and Pouzat C (2009) Quantitative estimation of calcium dynamics from ratiometric measurements: a direct, non-ratioing, method, Journal of Neurophysiology, in revision

## Examples

```
## (0) "Experimental" parameters

## Parameters of the monoexponential calcium transient
tOn <- 1
Time <- seq(0,10,0.1)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5

## Calibration parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)
```

```

## Experiment-specific parameters
nb_B    <- 5
B_T     <- 100.0
T_340   <- 0.015
T_380   <- 0.006
P       <- 1000
P_B     <- 1000
phi     <- 1.25
S_B_340 <- 100/P/T_340
S_B_380 <- 100/P/T_380

## (1) Create a monoexponential calcium decay
Ca_Mono <- caMonoExp(t = Time, tOn = tOn,
                    Ca0 = Ca0, dCa = dCa, tau = tau)

## (2) Create the background and transient fluorescence signals
adu_B_340 <- rep(fluc(Ca=rep(0,nb_B),
                    R_min=R_min$value, R_max=R_max$value,
                    K_eff=K_eff$value, K_d=K_d$value,
                    B_T=0, phi=phi, S_B=S_B_340, T_stim=T_340, P=P, P_B=P_B))

adu_340 <- rep(fluc(Ca=Ca_Mono,
                    R_min=R_min$value, R_max=R_max$value,
                    K_eff=K_eff$value, K_d=K_d$value,
                    B_T=B_T, phi=phi, S_B=S_B_340, T_stim=T_340, P=P, P_B=P_B))

adu_B_380 <- rep(fluc(Ca=rep(0,nb_B),
                    R_min=1, R_max=1, K_eff=K_eff$value, K_d=K_d$value,
                    B_T=0, phi=phi, S_B=S_B_380, T_stim=T_380, P=P, P_B=P_B))

adu_380 <- rep(fluc(Ca=Ca_Mono,
                    R_min=1, R_max=1, K_eff=K_eff$value, K_d=K_d$value,
                    B_T=B_T, phi=phi, S_B=S_B_380, T_stim=T_380, P=P, P_B=P_B))

## (3) Get the noisy calcium transient from the ratiometric transformation
Ca <- caFromRatio(adu_B_340, adu_340,
                 adu_B_380, adu_380,
                 T_340 = 0.015, T_380 = 0.006,
                 P, P_B,
                 R_min = R_min, R_max = R_max, K_eff = K_eff,
                 Plot = TRUE)

## (4) Superimpose the original calcium transient
lines(Ca_Mono, lty=2, col="red")

```

**Description**

The `caMonoBiExpFromIG` function returns a vector of intracellular calcium concentration ( $[Ca^{2+}]$ ). A  $dCa$  jump occurs at  $tOn$ , followed by either a monoexponential or a biexponential return to the baseline value  $Ca0$ , depending on the fields of the input list `ig`. The `caMonoBiExpFromIG` function is a low-level function of the `ratioFitFromCa` and `directFit` functions.

**Usage**

```
caMonoBiExpFromIG(t = 1, tOn = 1, ig = NULL)
```

**Arguments**

<code>t</code>	a vector of time values at which $[Ca^{2+}]$ is computed (in s)
<code>tOn</code>	the time of the $[Ca^{2+}]$ jump (in s)
<code>ig</code>	an object of class "initial_guess", giving the parameters of the decay. This object is a list with the following numerical fields: "log_Ca0", "log_dCa", "log_tau" in the case of a monoexponential decay. In the case of a biexponential decay, the two following fields are also included: "mu" and "log_dttau"

**Value**

A vector containing the  $[Ca^{2+}]$  values. The vector has the two following attributes:

<code>Time</code>	a copy of argument <code>t</code>
<code>tOn</code>	a copy of argument <code>tOn</code>

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[caMonoExp](#), [caBiExp](#)

**Examples**

```
## Parameters of the calcium transient
Ca0 <- 0.10
dCa <- 0.25
tau <- 1

## List of initial guesses
ig_mono <- list(log_Ca0 = log(Ca0),
               log_dCa = log(dCa),
               log_tau = log(tau))
class(ig_mono) <- "initial_guess"

## Build the calcium transient
Ca <- caMonoBiExpFromIG(t = seq(0,10,0.1),
                       tOn = 2,
```

```

        ig = ig_mono)

## Plot the calcium transient vs. time
plot(attr(Ca,"Time"), Ca, type="l")

## Add a vertical dashed line at t0n
abline(v = attr(Ca,"t0n"), lty = 2)

```

---

caMonoExp

*MonoExponential Time Course of Intracellular Calcium Concentration*


---

### Description

The function caMonoExp returns a vector of intracellular calcium concentration ( $[Ca^{2+}]$ ) vs time values  $t$ . A dCa jump occurs at  $t0n$ , followed by a monoexponential return to baseline value  $Ca0$ , with time constant  $\tau$ .

### Usage

```
caMonoExp(t = 1, t0n = 1, Ca0 = 0.05, dCa = 0.1, tau = 3)
```

### Arguments

t	a vector of time values at which $[Ca^{2+}]$ is computed (in s)
t0n	the time of the $[Ca^{2+}]$ jump (in s)
Ca0	the baseline $[Ca^{2+}]$ (in $\mu M$ )
dCa	the $[Ca^{2+}]$ jump occurring at $t0n$ (in $\mu M$ )
tau	the time constant of the $[Ca^{2+}]$ monoexponential return to baseline (in s)

### Value

A vector containing the  $[Ca^{2+}]$  values. The vector has the two following attributes:

Time	a copy of argument t
t0n	a copy of argument t0n

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[caBiExp](#), [caMonoBiExpFromIG](#)

## Examples

```
## Simulate a monoexponential calcium transient
Ca <- caMonoExp(t    = seq(0,10,0.1),
               tOn  = 2,
               Ca0  = 0.25,
               dCa  = 1,
               tau  = 2)

## Plot the calcium transient vs. time
plot(attr(Ca,"Time"),Ca,type="l")

## Add a vertical dashed line at tOn
abline(v = attr(Ca,"tOn"), lty = 2)
```

---

directFit	<i>Perform a Direct Fit of Fluorescence Signals Obtained with a Ratiometric Dye</i>
-----------	---

---

## Description

The function `directFit` performs a direct fit on fluorescence signals obtained with a ratiometric dye. The calcium dynamics are fitted with either a mono- or a biexponential decay, depending on the value of `type`.

## Usage

```
directFit(df, transients = c(1, 2, 3), SQR = TRUE, ratio = NULL,
          type = "mono", Plot = FALSE, Fit = TRUE,
          AfterPeak = FALSE, Trace = FALSE, WarnOnly = TRUE)
```

## Arguments

<code>df</code>	a data frame of class "fluo_rawdata" containing all relevant information (fluorescence transients, background fluorescence, calibration parameters and exposure times). The structure of the input data frame must be the same as the one defined in <a href="#">ratioExpSimul</a>
<code>transients</code>	a vector of integers giving the numbers of the transients to fit simultaneously
<code>SQR</code>	a logical value. Set to TRUE (default) to fit the square root of the fluorescence signals
<code>ratio</code>	an object of class "initial_guess" or "calcium_fit_ratio". If <code>ratio</code> is a list of class "initial_guess", it gives initial guesses (IGs) for the calcium dynamics parameters ( <code>log_Ca0</code> , <code>log_dCa</code> , <code>log_tau</code> for a monoexponential decay, eventually <code>mu</code> and <code>log_dtau</code> for a biexponential decay). If <code>ratio</code> is a "ratio_fit" object, the fitted values of its parameters are used as initial guesses. If <code>ratio</code> is none of the above objects, a ratiometric fit is performed on <code>df</code> to find initial guesses for the calcium dynamics parameters

type	a character string (either "mono" or "bi"), specifying the type of calcium exponential decay to consider
Plot	a logical value. Set to TRUE to plot the original signals, the initial guess and the fit results
Fit	a logical value. Set to TRUE to perform the fit, or to FALSE to compute an initial guess only
AfterPeak	a logical or numerical value. Set to FALSE to perform the fit on the whole fluorescence transients, to TRUE to consider only the part before the fluorescence jump and the convex part after the fluorescence peak (for both signals), or to an integer to skip a given number of samples after the fluorescence jump
Trace	a logical value. Set to TRUE to print results for successive steps of the optimization algorithm
WarnOnly	a logical value. Set to TRUE to go on even if the fit produced an error

### Details

The fit is performed using the `nls` function, which determines the nonlinear (weighted) least-squares estimates of the parameters of a nonlinear model. The algorithm is set to the default Gauss-Newton.

The initial guesses for the experiment-specific parameters are calculated with the `igDirect` function. If the "USE\_se" field of the calibration parameters and alpha are set to TRUE, the initial guesses for these parameters are given by their experimental mean value.

The quality of the direct fit is based on the probabilistic properties of the fluorescence signals, which are described as realizations of Poisson processes. For values of parameter above about 10, the Poissonian distribution can be approximated by a Gaussian distribution with variance equal to the mean. Applying the square root transformation (by setting the `SQRT` logical argument to TRUE) to such a process leads to a stabilization of the variance, which becomes equal to  $\frac{1}{4}$ . Thus, one is brought back to a standard nonlinear regression setting. Moreover, in this situation, it is possible to account for the limited precision with which the calibration parameters are known. For this purpose, these parameters are also fitted, and a weight of  $\frac{1}{\sigma_{exp}^2}$  is applied to each of them ( $\frac{1}{\sigma_{exp}}$  referring to the experimental standard error to the mean (sem)). The whole signal to fit is thus the following:

$$(B_{340}, F_{340}, B_{380}, F_{380}, R_{min}, R_{max}, K_{eff}, K_d),$$

with the following weights:

$$\left( 4, 4, 4, 4, \frac{1}{\sigma_{Rmin}^2}, \frac{1}{\sigma_{Rmax}^2}, \frac{1}{\sigma_{Keff}^2}, \frac{1}{\sigma_{Kd}^2} \right)$$

### Value

An object that inherits from both "nls" and "direct\_fit" classes. The object has the following attributes:

"Name"	a character string telling which type of fit has been performed
"Time"	the whole time vector, which includes NAs/NaNs for the background fluorescence signals and the supplementary calibration parameters, when relevant

"RawData" the raw signal, which is created by the concatenation of the background fluorescence at 340 nm, the fluorescence transient at 340 nm, the background fluorescence at 380 nm, the fluorescence transient at 380 nm, and, when relevant, the mean values of the selected calibration parameters. This signal is the one passed to the nls formula

"RawDataFrame" a copy of the input data frame

"FitFunction" the function passed to the nls formula

"Subset" the indices of the Time vector used for the fit

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[transientConvexPart](#), [mkFluo4DirectFit](#), [igDirect](#), [ratioExpSimul](#), [ratioExpPhysio](#)

### Examples

```
## Direct Fit On Simulated Data
## (parameters set to the value in Table 2 from
## Joucla et al. (2009) (Journal of Neurophysiology))
## =====

## Parameters of the monoexponential calcium transient
tOn <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Experiment-specific parameters
nb_B <- 1
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 200
P_B <- 200
phi <- 2
S_B_340 <- 30
S_B_380 <- 80

## Create a monoexponential calcium decay
Ca <- caMonoExp(t = Time, tOn = tOn,
                Ca0 = Ca0, dCa = dCa, tau = tau)
```

```

## Simulate the corresponding ratiometric experiment
df <- ratioExpSimul(nb_B = nb_B,
                   Ca    = Ca,
                   R_min = R_min,
                   R_max = R_max,
                   K_eff = K_eff,
                   K_d   = K_d,
                   B_T   = B_T,
                   phi   = phi,
                   S_B_340 = S_B_340,
                   S_B_380 = S_B_380,
                   T_340  = T_340,
                   T_380  = T_380,
                   P      = P,
                   P_B    = P_B,
                   ntransients = 1,
                   G      = 1,
                   s_ro   = 0)

## Perform a monoexponential and a biexponential ratiometric fit
direct_fit <- directFit(df = df,
                      transients = 1,
                      SQRT = TRUE,
                      ratio = NULL,
                      type = "mono")

## Plot the raw and fitted data as well as goodness of fit tests
plot(direct_fit, numTransient=1, items=1:6)

## Direct Fit On Physiological Data
## (reproduces Fig. 6 of Joucla et al. (2009))
## =====

## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151,   USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729,   USE_se=TRUE)
K_d   <- list(value=0.583, mean=0.583, se=0.123,   USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,

```

```

                                alphamethod=TRUE)

## Fit the physiological data with the direct method
## (skip 14 samples of the signal after t0n)
physiolDirectFit <- directFit(physioData,
                              transients=2,
                              Sqrt=TRUE,
                              type="mono",
                              AfterPeak=14)

## Plot the raw and fitted data as well as goodness of fit tests
plot(physiolDirectFit, numTransient=2, items=1:6)

```

---

fluo *Convert Intracellular Calcium Concentration into Fluorescence Values*

---

## Description

The function `fluo` converts an intracellular calcium concentration to a photon count, depending on the values of the calibration parameters ( $R_{min}$ ,  $R_{max}$ ,  $K_{eff}$  and  $K_d$ ) and the experiment-specific parameters ( $B_T$ ,  $\phi$ ,  $S_B$ ,  $T_{stim}$ ,  $P$  and  $P_B$ )

## Usage

```
fluo(Ca = 1, R_min = 0.136, R_max = 2.701, K_eff = 3.637, K_d = 0.583,
     B_T = 100, phi = 1.25, S_B = 10, T_stim = 0.015, P = 400, P_B = 400)
```

## Arguments

Ca	the intracellular calcium concentration (in $\mu\text{M}$ )
R_min	the minimum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
R_max	the maximum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
K_eff	the effective dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
K_d	the dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
B_T	the total dye concentration in the cell (in $\mu\text{M}$ )
phi	a dimensionless scaling experiment-specific parameter
S_B	the background (+ dark current) fluorescence intensity (in count/pixel/sec)
T_stim	the exposure time (in s)
P	the number of pixels of the Region Of Interest (ROI)
P_B	the number of pixels of the Background Region

## Details

The calcium imaging technique makes use of the ability of a fluorescent dye (e.g. Fura) to bind with calcium ions presents inside a neural cell. Briefly, the photons emitted by a neural tissue are recorded by a CCD camera, following the illumination of the tissue at a relevant wavelength (corresponding to the excitation properties of the free and/or bound dye). The amount of photons emitted depends on the intracellular calcium concentration (with which the dye is bound), and, in the case of a ratiometric dye, an algebraic relationship links both variables. The latter is given by:

$$F_{340} = \left( \frac{B_T \cdot \phi}{K_d + [Ca^{2+}]} \cdot (R_{min} \cdot K_{eff} + R_{max} \cdot [Ca^{2+}]) + S_{B,340} \right) \cdot T_{stim,340} \cdot P$$

$$F_{380} = \left( \frac{B_T \cdot \phi}{K_d + [Ca^{2+}]} \cdot (K_{eff} + [Ca^{2+}]) + S_{B,380} \right) \cdot T_{stim,380} \cdot P$$

The function fluo determines photon counts according to one of these two equations, depending on the values of R\_min, R\_max and B\_T

## Value

An object of class "fluo\_transient", which is a vector containing the fluorescence values calculated as described above. The object has several attributes, which are:

Ca	a copy of argument Ca
R_min	a copy of argument R_min
R_max	a copy of argument R_max
K_eff	a copy of argument K_eff
K_d	a copy of argument K_d
B_T	a copy of argument B_T
T_stim	a copy of argument T_stim
P	a copy of argument P
S_B	a copy of argument S_B
phi	a copy of argument phi

## Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

## References

Joucla S, Pippow A, Kloppenburg P and Pouzat C (2009) Quantitative estimation of calcium dynamics from ratiometric measurements: a direct, non-ratioing, method, *Journal of Neurophysiology*, in revision

## See Also

[caBiExp](#), [caMonoExp](#)

**Examples**

```

## Parameters of the monoexponential calcium transient
tOn <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5

## Calibrated parameters
R_min <- 0.136
R_max <- 2.701
K_eff <- 3.637
K_d <- 0.583

## Experiment-specific parameters
nb_B <- 1
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 200
P_B <- 200
phi <- 2
S_B_340 <- 30
S_B_380 <- 80

## Create a monoexponential calcium decay
Ca <- caMonoExp(t = Time, tOn = tOn,
                Ca0 = Ca0, dCa = dCa, tau = tau)

## Define Background and Signal fluorescences at 340 and 380 nm
B_340 <- fluo(Ca=rep(0,nb_B), R_min=R_min, R_max=R_max, K_eff=K_eff, K_d=K_d,
              B_T=0, phi=phi, S_B=S_B_340, T_stim=T_340, P=P, P_B=P_B)

F_340 <- fluo(Ca=Ca, R_min=R_min, R_max=R_max, K_eff=K_eff, K_d=K_d,
              B_T=B_T, phi=phi, S_B=S_B_340, T_stim=T_340, P=P, P_B=P_B)

B_380 <- fluo(Ca=rep(0,nb_B), R_min=1, R_max=1, K_eff=K_eff, K_d=K_d,
              B_T=0, phi=phi, S_B=S_B_380, T_stim=T_380, P=P, P_B=P_B)

F_380 <- fluo(Ca=Ca, R_min=1, R_max=1, K_eff=K_eff, K_d=K_d,
              B_T=B_T, phi=phi, S_B=S_B_380, T_stim=T_380, P=P, P_B=P_B)

## Plot the fluorescence transients at 340 and 380 nm
layout(matrix(1:2,nrow=2))
plot(Time, F_340, type="l", bty="n")
plot(Time, F_380, type="l", bty="n")

```

---

getThetaRatio

---

*Compute confidence intervals for parameters of calcium dynamics  
model fitted with the ratiometric approach*


---

**Description**

The function `getThetaRatio` estimates confidence intervals for parameters a calcium dynamic model fitted to data estimated with the ratiometric transformation. The way these CIs are computed depends on the `ciMode` parameter.

**Usage**

```
getThetaRatio(calcium_ratio_fit,
              ciMode = c("normalApprox", "likelihoodRatio"), ...)
```

**Arguments**

<code>calcium_ratio_fit</code>	an object of class "ratio_fit"
<code>ciMode</code>	should the normal approximation ("normal") or the likelihood ratio ("ratio") be used to obtain the CI?
<code>...</code>	not used

**Details**

The `ciMode` argument specifies which approach to use to estimate the CIs. If set to "normal", the quadratic approximation of the log-likelihood applies, and the 95% CIs are given as  $t(0.975, dof) \times se(p)$ , where  $t$  is the Student quantile function,  $dof$  is the number of degrees of freedom,  $se(p)$  is the standard error associated to the estimation of parameter  $p$  (given by the inverse of the square root of the diagonal of the hessian matrix returned by "optim"). If `ciMode` is set to "likelihoodRatio", we make use of the likelihood ratio statistics (Davison, 2003).

**Value**

A matrix with 2 rows and N columns, corresponding to the number of parameters of the calcium dynamics model. Each column gives the lower and upper bound of the 95% confidence interval for each parameter.

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**References**

Davison AC (2003), Statistical Models, Cambridge University Press

**Examples**

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
```

```

K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d   <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=FALSE)

## Retrieve the calcium concentration from the data frame
Ca_noisy <- caFromDf(df      = physioData,
                    numTransient = 2,
                    Plot      = FALSE)

## Perform a ratiometric fit
physioRatioFit <- ratioFitFromCa(Ca = Ca_noisy,
                                 t   = attr(Ca_noisy,"Time"),
                                 tOn = attr(Ca_noisy, "tOn"),
                                 type = "mono",
                                 AfterPeak = 14)

## Compute the confidence interval
## using the likelihood ratio statistics
CI <- getThetaRatio(physioRatioFit,
                    ciMode = "likelihoodRatio")

print(CI)

```

---

igDirect

*Provide an Initial Guess For the Experiment-Specific Parameters of a Direct Fit*


---

## Description

The function `igDirect` provides an initial guess for the experiment-specific parameters of fluorescence transients obtained with a ratiometric dye (the background fluorescence `log_S_B_340` and `log_S_B_380`, as well as the scaling coefficient `log_phi`)

## Usage

```

igDirect(adu_B_340, adu_340, adu_B_380, adu_380,
         ig_ratio, t, tOn = 1, subset = 1:length(t),
         R_min = 0.136, R_max = 2.701, K_eff = 3.637, K_d = 0.583,
         B_T = 100, T_340 = 0.015, T_380 = 0.006, P = 400, P_B = 400)

```

**Arguments**

adu_B_340	the background fluorescence at 340 nm
adu_340	the fluorescence transient at 340 nm
adu_B_380	the background fluorescence at 380 nm
adu_380	the fluorescence transient at 380 nm
ig_ratio	the initial guess list for the parameters of the $[Ca^{2+}]$ transient, returned by the igRatio function
t	a vector of time values at which the fluorescence values were obtained (in s)
tOn	the time of the fluorescence jump (in s)
subset	a vector of time indices to consider (generally the whole fluorescence signals)
R_min	the minimum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
R_max	the maximum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
K_eff	the effective dissociation constant of the dye in the cell (in $\mu M$ ). This parameter is obtained from calibration experiments
K_d	the dissociation constant of the dye in the cell (in $\mu M$ ). This parameter is obtained from calibration experiments
B_T	the total concentration of the dye in the cell (in $\mu M$ )
T_340	the exposure time at 340 nm (in s)
T_380	the exposure time at 380 nm (in s)
P	the number of pixels of the Region Of Interest (ROI)
P_B	the number of pixels of the Background Region

**Details**

The initial guesses for  $\log\_S\_B\_340$  and  $\log\_S\_B\_380$  are obtained by averaging the  $adu\_B\_340$  and  $adu\_B\_380$  signals, divided by  $T\_340 \cdot P\_B$  and  $T\_380 \cdot P\_B$  respectively, and by taking their logarithm.

The initial guess for  $\log\_phi$  is obtained by applying a linear (zero-intercept) regression between the following signals, and by taking the logarithm of the slope:

$$c \left( \frac{adu_{340}}{T_{340} \cdot P} - S_{B,340}, \frac{adu_{380}}{T_{380} \cdot P} - S_{B,380} \right)$$

$$\frac{[B_T]}{K_f + [Ca^{2+}]_{ratio}} \cdot c (R_{min} \cdot K_{eff} + R_{max} \cdot [Ca^{2+}]_{ratio}, K_{eff} + [Ca^{2+}]_{ratio})$$

In these formulas,  $[Ca^{2+}]_{ratio}$  refers to the calcium concentration transient estimated with the initial guess of the parameters listed in the  $ig\_ratio$  argument

**Value**

A named list of class "initial\_guess", containing initial guesses (IG) for the logarithms of the three experiment-specific parameters: The background fluorescences at 340 and 380 nm (`log_S_B_340` and `log_S_B_380` respectively) and the amplitude coefficient `log_phi`

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**References**

see the [fluo](#) documentation for details about the data generation model

**See Also**

[igRatio](#), [fluo](#)

**Examples**

```
## Parameters of the monoexponential calcium transient
t0n <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5

## Create a monoexponential calcium decay
Ca <- caMonoExp(t = Time, t0n = t0n,
                Ca0 = Ca0, dCa = dCa, tau = tau)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Experiment-specific parameters
nb_B <- 1
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 200
P_B <- 200
phi <- 2
S_B_340 <- 30
S_B_380 <- 80

## Define Background and Transient fluorescence
## signals at 340 and 380 nm
adu_B_340 <- fluo(Ca=rep(0,nb_B),
                 R_min=R_min$value, R_max=R_max$value,
                 K_eff=K_eff$value, K_d=K_d$value,
```

```

      B_T=0, phi=phi, S_B=S_B_340,
      T_stim=T_340, P=P, P_B=P_B)

adu_340 <- fluo(Ca=Ca,
               R_min=R_min$value, R_max=R_max$value,
               K_eff=K_eff$value, K_d=K_d$value,
               B_T=B_T, phi=phi, S_B=S_B_340,
               T_stim=T_340, P=P, P_B=P_B)

adu_B_380 <- fluo(Ca=rep(0,nb_B),
                 R_min=1, R_max=1,
                 K_eff=K_eff$value, K_d=K_d$value,
                 B_T=0, phi=phi, S_B=S_B_380,
                 T_stim=T_380, P=P, P_B=P_B)

adu_380 <- fluo(Ca=Ca,
               R_min=1, R_max=1,
               K_eff=K_eff$value, K_d=K_d$value,
               B_T=B_T, phi=phi, S_B=S_B_380,
               T_stim=T_380, P=P, P_B=P_B)

## Add Poissonian noise to these signals
adu_B_340 <- rpois(length(adu_B_340), adu_B_340)
adu_340 <- rpois(length(adu_340), adu_340)
adu_B_380 <- rpois(length(adu_B_380), adu_B_380)
adu_380 <- rpois(length(adu_380), adu_380)

## Extract the noisy calcium transient
## (from the ratiometric transformation)
Ca_noisy <- caFromRatio(adu_B_340, adu_340,
                       adu_B_380, adu_380,
                       T_340, T_380,
                       P, P_B,
                       R_min, R_max, K_eff,
                       Plot = FALSE)

## Perform a ratiometric fit to determine
## the calcium dynamics parameters
ratio_fit <- ratioFitFromCa(Ca_noisy, t=Time, tOn, type="mono")

## List the fitted parameters and create
## the corresponding calcium transient
ig_mono <- as.list(ratio_fit$par)
names(ig_mono) <- c("log_Ca0", "log_dCa", "log_tau")
class(ig_mono) <- "initial_guess"

## Perform an Initial Guess for the Experiment-Specific Parameters
ig_direct <- igDirect(adu_B_340 = adu_B_340,
                    adu_340 = adu_340,
                    adu_B_380 = adu_B_380,
                    adu_380 = adu_380,
                    ig_ratio = ig_mono,
                    t = Time, tOn = tOn, subset = 1:length(Time),

```

```

R_min = R_min$value, R_max = R_max$value,
K_eff = K_eff$value, K_d = K_d$value,
B_T = 100, T_340 = T_340, T_380 = T_380, P = P, P_B = P_B)

## Compare the initial guess with the known values of the parameters
print(exp(as.vector(unlist(ig_direct))))
print(c(Ca0, dCa, tau, phi, S_B_340, S_B_380))

```

---

igRatio

*Provide an Initial Guess For a Calcium Concentration Ratiometric Fit*


---

### Description

The function `igRatio` provides an initial guess for the parameters of an intracellular calcium concentration transient obtained after a ratiometric transformation. The transients considered here are either mono- or biexponential.

### Usage

```
igRatio(Ca, t, t0n = 1, type = "mono")
```

### Arguments

Ca	a vector of $[Ca^{2+}]$ values (in $\mu M$ )
t	a vector of time values at which $[Ca^{2+}]$ is computed (in s)
t0n	the time of the $[Ca^{2+}]$ jump (in s)
type	a character string (either "mono" or "bi"), indicating which type of exponential decay to consider

### Details

This function provides initial guesses for three or five parameters, depending on the type of exponential return to baseline. If `type` is set to "mono", three parameters are guessed:

- `log_Ca0` is obtained by averaging the signal prior to `t0n` (and taking its logarithm)
- `log_dCa` is obtained by subtracting the IG for `Ca0` from the maximum value of the signal (and taking its logarithm)
- `log_tau` is obtained from a linear regression on a rescaled (by `dCa`) and time offset (by `Ca0`) version of the original signal (and taking its logarithm)

If `type` is set to "bi", two more parameters (`mu` and `log_dtau`) are guessed. For that purpose, the slow time constant of the signal is first guessed: successive linear fits of the end part of the signal log-normalized signal (of increasing lengths, from `T` to `Tend`, with decreasing values of `T`) are performed, until the fitted time constant does not change anymore. Then,  $\tau_s$  is set to `T` and its relative weight (from which `mu` arises) is deduced from the fit intercept. Considering the slope of the signal at the peak then leads to the fast time constant of the biexponential decay (`tau`), thus, to `dtau`.

**Value**

A named list of class "initial\_guess", containing initial guesses (IG) for the three/five scalar following components of the mono- or bi- exponential calcium decay:

log_Ca0	IG for the logarithm of the baseline $[Ca^{2+}]$
log_dCa	IG for the logarithm of the $[Ca^{2+}]$ jump
log_tau	IG for the logarithm of the time constant of the monoexponential decay (if type is set to "mono") or the fast time constant of the biexponential decay (if type is set to "bi")
mu	IG for the real number (between -Inf and +Inf) defining the relative weight of the fast and slow time constants of the biexponential decay (if type is set to "bi"). The weight of the fast time constant is given by $\frac{\exp(\mu)}{1+\exp(\mu)}$
log_dtau	IG for the logarithm of the $d\tau$ defining the slow time constant of the biexponential decay (if type is set to "bi"). This slow time constant is given by $\tau_s = \tau + d\tau$

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[igDirect](#), [caMonoBiExpFromIG](#)

**Examples**

```
## Parameters of the monoexponential calcium transient
tOn <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5

## Calibration parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=FALSE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=FALSE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=FALSE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=FALSE)

## Experiment-specific parameters
nb_B <- 1
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 200
P_B <- 200
phi <- 2
S_B_340 <- 30
S_B_380 <- 80
```

```

## Create a monoexponential calcium decay
Ca <- caMonoExp(t=Time,
               t0n=t0n,
               Ca0=Ca0,
               dCa=dCa,
               tau=tau)

## Simulate the corresponding ratiometric experiment
df <- ratioExpSimul(nb_B = nb_B,
                   Ca    = Ca,
                   R_min = R_min,
                   R_max = R_max,
                   K_eff = K_eff,
                   K_d   = K_d,
                   B_T   = B_T,
                   phi   = phi,
                   S_B_340 = S_B_340,
                   S_B_380 = S_B_380,
                   T_340  = T_340,
                   T_380  = T_380,
                   P      = P,
                   P_B    = P_B,
                   ntransients = 1,
                   G      = 1,
                   s_ro   = 0)

## Get the noisy calcium transient from the data frame
Ca_noisy <- caFromDf(df, numTransient=1, Plot=FALSE)

## Find an Initial Guess for the calcium transient parameters
ig_mono <- igRatio(Ca=Ca_noisy, t=Time, t0n=t0n, type="mono")

## Plot the simulated noisy calcium transient over the original
## calcium transient
plot(Time, Ca_noisy, type="l")
lines(Time, Ca, col="blue")

## Add the calcium transient corresponding to the initial guess
lines(Time, caMonoBiExpFromIG(t=Time, t0n=t0n, ig=ig_mono), lwd=2, col="red")

## Add the corresponding legend
legend("topright",c("Ideal","Noisy","Initial Guess"),
      col=c("blue","black","red"),lwd=c(1,1,2))

```

## Description

Data from 20 *in vitro* experiments performed on cockroaches (*Periplaneta americana*) antennal lobe neurons. Each data set contains a list of experimental results made of the fura 2 loading curve

(measured at 360 nm) and 3 (or 2) calcium transients (measured at 340 and 380 nm).

## Usage

`data(inVitro)`

## Format

The data are stored as integer. `inVitro` contains 20 sublists. Each sublist corresponds to one experiment and contains:

<code>time360</code>	A vector of times (in s) at which fluorescence measurements at the isosbestic wavelength (360 nm) were taken
<code>adu360</code>	A vector of fluorescence measurements at 360 nm
<code>adu360Background</code>	Background fluorescence measurements at 360 nm
<code>P360</code>	Number of on-chip binned pixels used for the measurements at 360 nm
<code>P360Background</code>	Number of on-chip binned pixels used for the measurements of background fluorescence at 360 nm. If different from <code>P360</code> then the pixels used for background measurement were different than the one used for loading curve measurements
<code>onChipBinning360</code>	Number of physical pixels binned on-chip for the 360 nm measurements
<code>exposureTime360</code>	Exposure time used at 360 nm (in s)
<code>adu340Background</code>	Background fluorescence measurements at 340 nm
<code>adu380Background</code>	Background fluorescence measurements at 380 nm
<code>furaPipette</code>	The total fura concentration in the pipette (in $\mu\text{M}$ )
<code>exposureTime340</code>	Exposure time used at 340 nm (in s)
<code>exposureTime380</code>	Exposure time used at 380 nm (in s)
<code>P</code>	Number of on-chip binned pixels
<code>PBackground</code>	Number of on-chip binned pixels used for the measurements of background fluorescence at 340 and 380 nm. If different from <code>P</code> then the pixels used for background measurement were different than the one used for transient measurements
<code>onChipBinning</code>	Number of physical pixels binned on-chip for the 340 and 380 nm measurements
<code>stim1</code> , <code>stim2</code> and <code>stim3</code>	List with calcium transient data (see below)

Each element `stim1`, `stim2` and when a third transient was measured, `stim3`, are lists with the following components:

<code>time</code>	A vector of times (in s) at which fluorescence measurements were taken
<code>adu340</code>	A vector of fluorescence measurements at 340 nm
<code>adu380</code>	A vector of fluorescence measurements at 380 nm

## Details

Break-in time is 0.

`inVitro` contains data recorded in vitro (from cultured cells) from antennal lobe neurons for a total of 20 experiments. Experiments 5 and 16 in the `inVitro` data set have only 2 transients instead of

3.

**Source**

Andreas Pippow <andreas.pippow@uni-koeln.de> did the experiments

**Examples**

```
data(inVitro)
```

---

<code>mkFluo4DirectFit</code>	<i>Define Predicted Fluorescence Signals</i>
-------------------------------	--

---

**Description**

The function `mkFluo4DirectFit` defines a single signal made of four fluorescence signals (the background and fluorescence transients at both wavelengths), for use with within the `mkFunction4DirectFit` function

**Usage**

```
mkFluo4DirectFit(Ca, phi, S_B_340, S_B_380, nb_B,
                 R_min, R_max, K_eff, K_d, B_T,
                 T_340, T_380, P, P_B, SQRT = TRUE)
```

**Arguments**

<code>Ca</code>	the time course of the intracellular calcium concentration
<code>phi</code>	the scaling experiment-specific parameter
<code>S_B_340</code>	the background fluorescence at 340 nm
<code>S_B_380</code>	the background fluorescence at 380 nm
<code>nb_B</code>	the number of background measurements
<code>R_min</code>	the minimum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
<code>R_max</code>	the maximum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
<code>K_eff</code>	the effective dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
<code>K_d</code>	the dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
<code>B_T</code>	the total concentration of the dye inside the cell (in $\mu\text{M}$ )
<code>T_340</code>	the exposure time at 340 nm
<code>T_380</code>	the exposure time at 380 nm
<code>P</code>	the number of pixels of the ROI
<code>P_B</code>	the number of pixels of the background region
<code>SQRT</code>	a logical value. Set to TRUE to apply the square root transformation to the fluorescence signals

**Value**

a vector containing, in this order: the background fluorescence (in count) at 340 nm, the fluorescence transient at 340 nm, the background fluorescence at 380 nm and the fluorescence transient at 380 nm. If the `SQRT` argument is set to `TRUE`, the square root of the whole signal is returned

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[mkFunction4DirectFit](#), [directFit](#)

**Examples**

```
## Parameters of the biexponential calcium transient
t0n <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5
mu <- 0
dtau <- 10

## Calibration parameters
R_min <- 0.136
R_max <- 2.701
K_eff <- 3.637
K_d <- 0.583

## Experiment-specific parameters
nb_B <- 1
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 200
P_B <- 200
phi <- 2
S_B_340 <- 30
S_B_380 <- 80

## Create a biexponential calcium decay
Ca_Bi <- caBiExp(t=Time,
                 t0n=t0n,
                 Ca0=Ca0,
                 dCa=dCa,
                 tau=tau,
                 fact=1/(1+exp(-mu)),
                 dtau=dtau)

## Define the whole original fluorescence vector
Fluo_bi <- mkFluo4DirectFit(Ca = Ca_Bi, phi,
```

```
S_B_340, S_B_380, nb_B,
R_min, R_max, K_eff, K_d,
B_T, T_340, T_380, P, P_B,
SQRT = TRUE)
```

---

mkFunction4DirectFit *Function for Direct Fit*

---

### Description

The function `mkFunction4DirectFit` returns a function predicting the exponential time course of fluorescence transients at two wavelengths (including background fluorescence). The function, the arguments of which depends on the value of `type`, can be passed to the `nls` formula, for a direct fit

### Usage

```
mkFunction4DirectFit(type = "mono", nb_B = 5, transients = 1,
                    alphamethod = TRUE, SQRT = TRUE)
```

### Arguments

<code>type</code>	a character string (either "mono" or "bi"), specifying which type of exponential decay should be considered
<code>nb_B</code>	the number of background measurements performed at each wavelength
<code>transients</code>	a vector of integers, specifying the numbers of the transients to fit (determines the names of the calcium dynamics parameters ( <code>log_Ca0_1</code> , <code>log_Ca0_2</code> , ...))
<code>alphamethod</code>	a logical value. Set to TRUE (respectively FALSE) to include alpha (respectively B_T) in the output function formals
<code>SQRT</code>	a logical value. Set to TRUE (the default value) to return the square root of the fluorescence signals

### Value

A function, the arguments of which depend on the value of `type`, `alphamethod` and `transients`. The list below describes all possible arguments:

<code>t</code>	the times at which the fluorescence values are expected (in s)
<code>tOn</code>	the time of the fluorescence jump (in s)
<code>adu_B_340</code>	the background fluorescence measurement(s) at 340 nm
<code>adu_340</code>	the fluorescence transient(s) at 340 nm
<code>adu_B_380</code>	the background fluorescence measurement(s) at 380 nm
<code>adu_380</code>	the fluorescence transient(s) at 380 nm
<code>T_340</code>	the exposure time at 340 nm
<code>T_380</code>	the exposure time at 380 nm

P	the number of pixels of the ROI
P_B	the number of pixels of the background region
log_Ca0_1	the logarithm of the $[Ca^{2+}]$ baseline of the first transient
log_dCa_1	the logarithm of the $[Ca^{2+}]$ jump of the first transient
log_tau_1	the logarithm of the $[Ca^{2+}]$ time constant of the first transient
mu_1	the real number (between -Inf and +Inf) defining the relative weight of the fast and slow time constants of the $[Ca^{2+}]$ biexponential decay (if type is set to "bi"). The weight of the fast time constant is given by $\frac{\exp(\mu)}{1+\exp(\mu)}$
log_dtau_1	the logarithm of the $d\tau$ defining the slow time constant of the $[Ca^{2+}]$ biexponential decay (if type is set to TRUE"bi"). This slow time constant is given by $\tau_s = \tau + d\tau$
log_phi	the logarithm of the experiment-specific amplitude coefficient
log_S_B_340	the logarithm of the background fluorescence at 340 nm
log_S_B_380	the logarithm of the background fluorescence at 380 nm
log_R_min	the logarithm of the minimum ratiometric measurement observable
log_R_max	the logarithm of the maximum ratiometric measurement observable
log_K_eff	the logarithm of the effective Fura dissociation constant in the cell (in $\mu M$ )
log_K_d	the logarithm of the Fura dissociation constant (in $\mu M$ )
alpha	the isocoefficient. Present only if alphamethod is set to TRUE. In this case, B_T is replaced with $adu_{340}/(T_{340} \cdot P) - adu_{B,340}/(T_{340} \cdot P_B) + alpha \cdot (adu_{380}/(T_{380} \cdot P) - adu_{B,380}/(T_{380} \cdot P_B))$ in the fluorescence model
B_T	the total concentration of the dye in the cell. Present only if alphamethod is set to FALSE

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[mkFluo4DirectFit](#), [directFit](#)

**Examples**

```
## Parameters of the biexponential calcium transient
tOn <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5
mu <- 0
dtau <- 10

## Calibrated parameters
R_min <- 0.136
```

```

R_max <- 2.701
K_eff <- 3.637
K_d   <- 0.583

## Experiment-specific parameters
B_T   <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P     <- 200
P_B   <- 200
phi   <- 2
S_B_340 <- 30
S_B_380 <- 80

## Define a function for fluorescence transients based on
## a monoexponential calcium concentration time course
Fluo_bi_fct <- mkFunction4DirectFit(type = "bi",
                                   nb_B = 5,
                                   transients = 1,
                                   alphasmethod = FALSE,
                                   Sqrt = TRUE)

## Create the fluorescence transients
Time <- matrix(Time, nrow=1, dimnames=list("1",NULL))
names(t0n) <- "1"
Fluo_bi <- Fluo_bi_fct(t = Time, t0n = t0n,
                      T_340 = T_340, T_380 = T_380,
                      P = P, P_B = P_B,
                      log_Ca0_1 = log(Ca0),
                      log_dCa_1 = log(dCa),
                      log_tau_1 = log(tau),
                      mu_1 = mu,
                      log_dtau_1 = log(dtau),
                      log_phi = log(phi),
                      log_S_B_340 = log(S_B_340),
                      log_S_B_380 = log(S_B_380),
                      log_R_min = log(R_min), log_R_max = log(R_max),
                      log_K_eff = log(K_eff), log_K_d = log(K_d),
                      B_T = B_T)

```

---

mkFunction4RatioFit      *Function for Ratiometric Fit with the nls function*

---

### Description

The function `mkFunction4RatioFit` returns a function predicting the exponential time course of an intracellular concentration. The function, the arguments of which depend on the value of `type`, can be used within the `RSS` function of a ratiometric fit.



```
## Check that both decays are similar
plot(Time, Ca_biexp_decay1, type="l", lwd=2)
lines(Time, Ca_biexp_decay2, col="red", lty=2, lwd=2)
```

---

plot.direct\_fit      *Plot Function for Objects of Class "direct\_fit"*

---

## Description

The function `plot.direct_fit` performs different kinds of predefined plots for objects of class `direct_fit`

## Usage

```
## S3 method for class 'direct_fit'
plot(x, y = NULL, numTransient = 1, items = 1:7,
     col = "black", col2 = "darkgray",
     main = "Fluorescence transients: Direct fit",
     xlab = c("Time (s)", "Time (s)", "Time (s)", "Time (s)",
              "Lag", "Theoretical quant.", "Residuals"),
     ylab = c(expression(sqrt(adu[340])),
               expression(res[340]),
               expression(sqrt(adu[380])),
               expression(res[380]),
               "ACF", "Sample quant.", "Counts"),
     labs = c(expression(A[1]), expression(A[2]),
               expression(B[1]), expression(B[2]),
               "C", "D", "E"),
     ylas = 1, ask = FALSE, ...)
```

## Arguments

<code>x</code>	a data frame of class <code>direct_fit</code> , as returned by the <code>directFit</code> function
<code>y</code>	argument not used (NULL by default)
<code>numTransient</code>	a vector of integers specifying which of the fitted transients should be plotted
<code>items</code>	a vector of integers (between 1 and 7), telling which panels to draw
<code>col</code>	the color of the main signal to plot (either an integer or a character string)
<code>col2</code>	the color of the secondary signal to plot (either an integer or a character string)
<code>main</code>	a character string specifying the main title of the figure
<code>xlab</code>	a vector of character strings specifying the label of the x-axes
<code>ylab</code>	a vector of character strings specifying the label of the x-axes
<code>labs</code>	a vector of character strings specifying the label at the top-left of each panel (generally a letter, a letter with an integer, or an expression)
<code>ylas</code>	an integer specifying the orientation of the yticks. Possible values are 0 or 3 (vertical), or 1 or 2 (horizontal)

ask	a logical value. Set to FALSE to draw all plots on the same figure. If set to TRUE, a single device will be opened, in which the first plot will be drawn. Three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, for interactions with the user (see details below)
...	one or more of the following plot parameters: "cex", "cex.axis", "cex.lab", "cex.main", "font", "font.axis", "font.lab", "font.main", "line.xlab", "line.ylab", "line.lab", "line.main", "adj.main", "xlim", "ylim", "tcl", "mgp.x", "mgp.y"

### Details

If the user does not want to draw all plots in the same figure, (s)he can set to logical ask value to TRUE. In that case, three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, allowing user interactions. By clicking on the arrowhead oriented left (resp. right), the user will draw the previous (resp. next) plot (within items). By clicking on the black square, the user will stay on the current plot and none of the symbols will be available anymore

### Value

This plot function does not return anything else than the plotted figures

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[plotCalciOMatic](#), [plot.fluo\\_rawdata](#), [plot.ratio\\_fit](#), [plot.ratio\\_fit\\_list](#)

### Examples

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=TRUE)
```

```

## Perform a direct fit
physioDirectFit <- directFit(physioData,
                             transients=2,
                             SQR=True,
                             type="mono",
                             AfterPeak=14)

## Plot the raw and fitted data as well as goodness of fit tests
## on the same figure
plot(x=physioDirectFit, numTransient=2, items=1:6)

## Plot the raw and fitted data as well as goodness of fit tests
## on the same figure
## plot(x=physioDirectFit, numTransient=2, items=1:6, ask=TRUE)

```

---

plot.fluo\_rawdata      *Plot Function for Objects of Class "fluo\_rawdata"*

---

## Description

The function `plot.fluo_rawdata` performs different kinds of predefined plots for objects of class `fluo_rawdata`

## Usage

```

## S3 method for class 'fluo_rawdata'
plot(x, y = NULL, numTransient = 1,
      items = 1:3, col = "black",
      main = "Ratiometric experiment: raw data",
      xlab = c("", "", "Time (s)"),
      ylab = c(expression(paste(adu[340], " (photons)")),
                expression(paste(adu[380], " (photons)")),
                expression(paste("[", Ca^{2} + phantom(),
                                  "] (" , mu, "M)"))),
      labs = c(expression(A[1]), expression(A[2]), "B"),
      ylas = 1, ask = FALSE, ...)

```

## Arguments

<code>x</code>	a data frame of class <code>fluo_rawdata</code> , as returned by the <a href="#">ratioExpSimul</a> and <a href="#">ratioExpPhysio</a> functions
<code>y</code>	argument not used (NULL by default)
<code>numTransient</code>	a vector of integers specifying which of the fitted transients should be plotted
<code>items</code>	a vector of integers (between 1 and 3), telling which panels to draw
<code>col</code>	the color of the signals to plot (either an integer or a character string)
<code>main</code>	a character string specifying the main title of the figure

xlabs	a vector of character strings specifying the label of the x-axes
ylabs	a vector of character strings specifying the label of the x-axes
labs	a vector of character strings specifying the label at the top-left of each panel (generally a letter, a letter with an integer, or an expression)
ylas	an integer specifying the orientation of the yticks. Possible values are 0 or 3 (vertical), or 1 or 2 (horizontal)
ask	a logical value. Set to FALSE to draw all plots on the same figure. If set to TRUE, a single device will be opened, in which the first plot will be drawn. Three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, for interactions with the user (see details below)
...	one or more of the following plot parameters: "cex", "cex.axis", "cex.lab", "cex.main", "font", "font.axis", "font.lab", "font.main", "line.xlab", "line.ylab", "line.lab", "line.main", "adj.main", "xlim", "ylim", "tcl", "mgp.x", "mgp.y"

### Details

If the user does not want to draw all plots in the same figure, (s)he can set to logical ask value to TRUE. In that case, three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, allowing user interactions. By clicking on the arrowhead oriented left (resp. right), the user will draw the previous (resp. next) plot (within items). By clicking on the black square, the user will stay on the current plot and none of the symbols will be available anymore

### Value

This plot function does not return anything else than the plotted figures

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[plotCalciOMatic](#), [plot.direct\\_fit](#), [plot.ratio\\_fit](#), [plot.ratio\\_fit\\_list](#)

### Examples

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
```

```
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=TRUE)

## Plot fluorescence transients and the calcium transient
## obtained after ratiometric transformation
plot(x = physioData, numTransient = 2, items=1:3)

## Wait for user action
## plot(x = physioData, numTransient = 2, items=1:3, ask = TRUE,
##      xlabs = c("Time (s)", "Time (s)", "Time (s)"))
```

---

plot.ratio\_fit

*Plot Method for Objects of Class "ratio\_fit"*


---

## Description

The function `plot.ratio_fit` performs different kinds of predefined plots for objects of class `ratio_fit`

## Usage

```
## S3 method for class 'ratio_fit'
plot(x, y = NULL, items = 1:5,
     col = "black", col2 = "darkgray",
     main = "Intracellular calcium transient: Ratiometric fit",
     xlabs = c("Time (s)", "Time (s)", "Lag",
               "Theoretical quant.", "Residuals"),
     ylabs = c(expression(paste("[", Ca^{2+phantom()}],
                               "]" ("", mu, "M)")), expression(res[Ca]),
               "ACF", "Sample quant.", "Counts"),
     labs = c(expression(A[1]), expression(A[2]),
               "B", "C", "D"),
     ylas = 1, ask = FALSE, ...)
```

## Arguments

<code>x</code>	an object of class <code>ratio_fit</code> , as returned by the <code>ratioFitFromDf</code> function
<code>y</code>	argument not used (NULL by default)
<code>items</code>	a vector of integers (between 1 and 5), telling which plots to draw
<code>col</code>	the color of the main signals to plot (either an integer or a character string)
<code>col2</code>	the color of the secondary signals to plot (either an integer or a character string)

main	a character string specifying the main title of the figure
xlabs	a vector of character strings specifying the label of the x-axes
ylabs	a vector of character strings specifying the label of the y-axes
labs	a vector of character strings specifying the label at the top-left of each panel (generally a letter, a letter with an integer, or an expression)
ylas	an integer specifying the orientation of the yticks. Possible values are 0 or 3 (vertical), or 1 or 2 (horizontal)
ask	a logical value. Set to FALSE to draw all plots on the same figure. If set to TRUE, a single device will be opened, in which the first plot will be drawn. Three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, for interactions with the user (see details below)
...	one or more of the following plot parameters: "cex", "cex.axis", "cex.lab", "cex.main", "font", "font.axis", "font.lab", "font.main", "line.xlab", "line.ylab", "line.lab", "line.main", "adj.main", "xlim", "ylim", "tcl", "mfp.x", "mfp.y"

### Details

If the user does not want to draw all plots in the same figure, (s)he can set to logical ask value to TRUE. In that case, three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, allowing user interactions. By clicking on the arrowhead oriented left (resp. right), the user will draw the previous (resp. next) plot (within items). By clicking on the black square, the user will stay on the current plot and none of the symbols will be available anymore

### Value

This plot method does not return anything else that the plotted figures

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[plotCalciOMatic](#), [plot.fluo\\_rawdata](#), [plot.direct\\_fit](#), [plot.ratio\\_fit\\_list](#)

### Examples

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
```

```

## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=FALSE)

## Perform a ratiometric fit
physioRatioFit <- ratioFitFromDf(df = physioData,
                                 transients = 2,
                                 AfterPeak = 14)

## Print the class of 'physioRatioFit'
print(class(physioRatioFit))

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=FALSE)

## Perform a ratiometric fit using \code{"optim"}
physioRatioFit <- ratioFitFromDf(df = physioData,
                                 transients = 2,
                                 AfterPeak = 14)

## Plot the results on the same figure
plot(x=physioRatioFit, items=1:4)

## Plot the results on separate figures and wait for a user action
## plot(x=physioRatioFit, items=1:4, ask=TRUE)

```

---

plot.ratio\_fit\_list     *Plot Method for Objects of Class "ratio\_fit\_list"*

---

### Description

The function `plot.ratio_fit_list` performs different kinds of predefined plots for objects of class `ratio_fit_list`

**Usage**

```
## S3 method for class 'ratio_fit_list'
plot(x, y = NULL, numTransient = 1,
     items = 1:5, col = "black", col2 = "darkgray",
     main = "Intracellular calcium transient: Ratiometric fit",
     xlabs = c("Time (s)", "Time (s)", "Lag",
               "Theoretical quantiles", "Residuals"),
     ylabs = c(expression(paste("[", Ca^{2 + phantom()},
                               "]" ("", mu, "M"))), "Residuals",
               "Autocorrelation function",
               "Sample quantiles", "Counts"),
     labs = c(expression(A[1]), expression(A[2]),
               "B", "C", "D"),
     ylas = 1, ask = FALSE, ...)
```

**Arguments**

x	an object of class <code>ratio_fit_list</code> , as returned by the <code>ratioFitFromDf</code> function
y	argument not used (NULL by default)
numTransient	a vector of integers specifying which of the fitted transients would be plotted
items	a vector of integers (between 1 and 5), telling which panels to draw
col	the color of the main signals to plot (either an integer or a character string)
col2	the color of the secondary signals to plot (either an integer or a character string)
main	a character string specifying the main title of the figure
xlabs	a vector of character strings specifying the label of the x-axes
ylabs	a vector of character strings specifying the label of the x-axes
labs	a vector of character strings specifying the label at the top-left of each panel (generally a letter, a letter with an integer, or an expression)
ylas	an integer specifying the orientation of the yticks. Possible values are 0 or 3 (vertical), or 1 or 2 (horizontal)
ask	a logical value. Set to FALSE to draw all plots on the same figure. If set to TRUE, a single device will be opened, in which the first plot will be drawn. Three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, for interactions with the user (see details below)
...	one or more of the following plot parameters: "cex", "cex.axis", "cex.lab", "cex.main", "font", "font.axis", "font.lab", "font.main", "line.xlab", "line.ylab", "line.lab", "line.main", "adj.main", "xlim", "ylim", "tcl", "mgp.x", "mgp.y"

**Details**

If the user does not want to draw all plots in the same figure, (s)he can set to logical `ask` value to TRUE. In that case, three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, allowing user interactions. By clicking on the arrowhead oriented left (resp.

right), the user will draw the previous (resp. next) plot (within items). By clicking on the black square, the user will stay on the current plot and none of the symbols will be available anymore. If ask is set to FALSE,  $N = \text{length}(\text{numTransient})$  figures will be opened simultaneously within a device similar to the one in which the first figure was plotted

### Value

This plot method does not return anything else than the plotted figures

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[plotCalciOMatic](#), [plot.fluo\\_rawdata](#), [plot.direct\\_fit](#), [plot.ratio\\_fit](#)

### Examples

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=FALSE)

## Perform a ratiometric fit
physioRatioFit <- ratioFitFromDf(df = physioData,
                                transients = 2,
                                AfterPeak = 14)

## Print the class of 'physioRatioFit'
print(class(physioRatioFit))

## Create the data frame containing the physiological data
## (experiment #2, stimulations #2 and #3)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
```

```

physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=c(2,3),
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=FALSE)

## Perform a ratiometric fit
physioRatioFit <- ratioFitFromDf(df = physioData,
                                transients = c(2,3),
                                AfterPeak = 14)

## Print the class of 'physioRatioFit'
print(class(physioRatioFit))

## Plot the results on the same figure
## plot(x = physioRatioFit, numTransient = c(2,3), items=1:4)

## Plot the results on separate figures and wait for a user action
## Click on the black square to make the second figure appear
## plot(x = physioRatioFit, numTransient = c(2,3), items=1:4, ask=TRUE)

```

---

plotCalciOMatic

*Low-Level Plot Function for CalciOMatic*


---

## Description

The low-level function `plotCalciOMatic` performs different kinds of predefined plots, depending on the value of `n`. It is generally called by plot methods dedicated to several types of objects ("`fluo_rawdata`", "`ratio_fit`", "`ratio_fit_list`" or "`direct_fit`")

## Usage

```

plotCalciOMatic(x = NULL, y = NULL, n = 1, x2 = NULL, y2 = NULL,
               col = "black", col2 = "darkgray",
               main = "MyCalciumRatiometricFit",
               xlab = "", ylab = "", lab = "A", ylas = 1,
               oma = c(4, 0, 1, 0), mar = c(0, 7, 2, 0),
               ask = FALSE, ...)

```

## Arguments

<code>x</code>	the abscissa of the main signal to plot (see details below)
<code>y</code>	the main signal to plot (see details below)
<code>n</code>	an integer, between 1 and 5, telling which kind of plot is used (see details below)
<code>x2</code>	the abscissa of the secondary signal to plot (see details below)
<code>y2</code>	the secondary signal to plot (see details below)

col	the color of the main signal (either an integer or a character string)
col2	the color of the secondary signal (either an integer or a character string)
main	a character string specifying the main title of the plot
xlab	a character string specifying the label of the x-axis
ylab	a character string specifying the label of the y-axis
lab	a character string specifying the label at the top-left of the plot (generally a letter, a letter with an integer, or an expression)
ylas	an integer specifying the orientation of the yticks. Possible values are 0 or 3 (vertical), or 1 or 2 (horizontal)
oma	a vector of length 4 specifying the outer margin of the figure
mar	a vector of length 4 specifying the margin of the panel to add
ask	a logical value. Set to FALSE to draw all plots on the same figure. If set to TRUE, a single device will be opened, in which the first plot will be drawn. Three symbols (left arrow, black square and right arrow) will be added at the bottom right of the plot, for interactions with the user (see details below)
...	one or more of the following plot parameters: "cex", "cex.axis", "cex.lab", "cex.main", "font", "font.axis", "font.lab", "font.main", "line.xlab", "line.ylab", "line.lab", "line.main", "adj.main", "xlim", "ylim", "tcl", "mgp.x", "mgp.y"

### Details

The third argument of the `plot.calciomatic` function is an integer `n` comprised between 1 and 5. The plot drawn directly depends on the value of `n`, as follows:

- `n=1` plot `y` vs. `x` (type lines and color `col`), and superimpose the plot of `y2` vs. `x2` (color `col2`). Used to plot raw data and fitted data vs. time values
- `n=2` plot `y` vs. `x` (type lines and color `col`), and add a dashed horizontal line at `y=0` (color `col2`). Used to plot fit residuals
- `n=3` plot a bar plot, with `y` of class "acf" (color `col`), and add a 95% confidence band between  $y = -1.96/\sqrt{\text{length}(y[["acf"]])}$  and  $y = +1.96/\sqrt{\text{length}(y[["acf"]])}$  (color `col2`). Used to plot the auto-correlation function of the residuals
- `n=4` plot `y` vs. `x` (type points and color `col`), add a diagonal dashed line (color `col`) and a 95% confidence band (color `col2`). Used to plot quantile-quantile plots of the fit residuals
- `n=5` plot an histogram of `x` (color `col`), and add vertical dashed lines at `-3/-2/-1/0/1/2/3` times the standard deviation of `x` (color `col2`). Used to plot the histogram of the fit residuals

### Value

When `ask` is set to its default FALSE value, a plot is drawn, and nothing else is returned by the function. If `ask` is set to TRUE, the action to execute is returned, as a character string

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>



```

        ntransients = 1,
        G           = 1,
        s_ro       = 0)

## Extract relevant data from the data frame
Time <- with(df,Time[!is.na(Time) & lambda==340])
adu_340 <- with(df,adu[!is.na(Time) & lambda==340])
adu_380 <- with(df,adu[!is.na(Time) & lambda==380])

## Plot the fluorescence data in the same figure
par(oma = c(4, 0, 1, 0), mar = c(0, 7, 2, 0))
layout(matrix(c(1,2),ncol=1))

plotCalciOMatic(x = Time, y = adu_340, n = 1, xlab = "",
                ylab = expression(paste(adu[340], "(photons)")),
                lab = "A", main = "Fluorescence Raw Data")

plotCalciOMatic(x = Time, y = adu_380, n = 1, xlab = "Time (s)",
                ylab = expression(paste(adu[380], "(photons)")),
                lab = "B", main = "")

```

---

ratioExpPhysio	<i>Gathers the Results of a Ratiometric Experiment in an Object of Class "fluo_rawdata"</i>
----------------	---

---

## Description

The function `ratioExpPhysio` gathers the results of a single ratiometric experiment including 1 or more fluorescence transients in a data frame of class `"fluo_rawdata"`, usable by the following functions: `ratioFitFromDf`, `directFit` and `plot.fluo_rawdata`

## Usage

```
ratioExpPhysio(dataset = "inVitro", expe = 1, stim = 1, idxOn = 10,
               R_min = 0.136, R_max = 2.701, K_eff = 3.637, K_d = 0.583,
               G = 0.146, s_ro = 16.4, alphamethod = TRUE)
```

## Arguments

<code>dataset</code>	a character string. The name of the variable containing results of ratiometric experiments. The minimal structure of this variable is detailed in <code>inVitro</code>
<code>expe</code>	the number of the experiment to consider (field "Exp.." of the dataset)
<code>stim</code>	a vector of integers specifying the number of the stimulations to consider (field "stim.")
<code>idxOn</code>	the index of the time at which the stimulation is applied
<code>R_min</code>	the minimum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments

R_max	the maximum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
K_eff	the effective dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
K_d	the dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
G	the gain of the CCD camera
s_ro	the standard deviation of the read-out process of the camera
alphamethod	a logical value. If set to TRUE, the fluorescence measurements at 360 nm (contained in the dataset) are used to estimate the isocoefficient $\alpha$

### Details

Details about the estimation of the isocoefficient  $\alpha$  with the fluorescence measurements at 340, 380 and 360 nm are given in Joucla et al. (2009, J Neurophysiol) (see Methods and Appendix C)

The variable entitled "dataset", which contains all experiment informations, should be a list with fields named "Exp01", "Exp02", etc. Each of this field should be a list with (at least) the following fields. The information contained in these fields are retrieved and put at the right place in the output data frame:

stim1	(eventually stim2, stim3, etc.)
adu340Background	a vector of background fluorescence recorded at 340 nm
adu380Background	a vector of background fluorescence recorded at 380 nm
P	the number of pixels used for the data binning of the raw image, for fluorescence transient
PBackground	the number of pixels used for the data binning of the raw image, for background fluorescence
furaPipette	the total concentration of the dye in the pipette (in $\mu\text{M}$ ) which is assumed to be the same in the cell, at steady-state
exposureTime340	the exposure time at 340 nm (in s)
exposureTime380	the exposure time at 380 nm (in s)

Each field of "stim1" should be a list with at least the following fields:

time	the times at which the fluorescence transient was acquired
adu340	the fluorescence transient obtained at 340 nm
adu380	the fluorescence transient obtained at 380 nm

**Value**

An object of class "fluo\_rawdata", which is a data frame with four columns:

adu	the photon counts (or Analog-to-Digital Units) at both wavelengths, including background fluorescence
Time	the times at which each value in adu was recorded. For the background fluorescence, Time is set to NA
lambda	the wavelength at which each value in adu was recorded (a factor)
transient	the number of the fluorescence transient in the input data (can be 1, 2 or 3 for transient signals, and 0 for background measurements)

Data appear in this order : (1) the background fluorescence at 340 nm, (2) the fluorescence transient(s) at 340 nm, (3) the background fluorescence at 380 nm, (4) the fluorescence transient(s) at 380 nm. The object has also the following attributes:

tOn	the time at which the stimulation is applied (in s)
T_stim	a vector containing the exposure times at 340 nm and 380 nm
R_min	a copy of arg R_min
R_max	a copy of arg R_max
K_eff	a copy of arg K_eff
K_d	a copy of arg K_d
P	a copy of field P of the input data set
P_B	a copy of arg PBackground of the input data set
B_T	the total Fura concentration in the cell (in $\mu\text{M}$ )
nb_B	the number of background measurements performed at each wavelength, before loading the dye into the cell
alpha	an estimation of the isocoefficient (only if alphamethod is set to TRUE)
G	a copy of arg G
s_ro	a copy of arg s_ro

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[ratioExpSimul](#)

**Examples**

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=TRUE)

## Plot the raw data
plot(physioData, numTransient=2)
```

---

ratioExpSimul

*Simulate Ratiometric Experiment*


---

**Description**

The function `ratioExpSimul` simulates the results of one ratiometric experiment, i.e, the photon counts obtained at both wavelengths (340 and 380 nm), knowing the time course of the intracellular calcium concentration. The photon counts are described as the realization of a Poissonian process.

**Usage**

```
ratioExpSimul(nb_B = 5, Ca,
              R_min = 0.136, R_max = 2.701, K_eff = 3.637, K_d = 0.583,
              B_T = 100, phi = 1.25, S_B_340 = 10, S_B_380 = 10,
              T_340 = 0.015, T_380 = 0.006, P = 400, P_B = 400,
              ntransients = 1, G = 1, s_ro = 0, noise = TRUE)
```

**Arguments**

<code>nb_B</code>	the number of background measurements to simulate before the fluorescence transients
<code>Ca</code>	the ideal calcium transient from which fluorescence signals arise

R_min	the minimum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
R_max	the maximum fluorescence ratio between the measurements at 340 and 380 nm. This parameter is obtained from calibration experiments
K_eff	the effective dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
K_d	the dissociation constant of the dye in the cell (in $\mu\text{M}$ ). This parameter is obtained from calibration experiments
B_T	the total concentration of the dye in the cell (in $\mu\text{M}$ )
phi	the scaling experiment-specific parameters
S_B_340	the background fluorescence at 340 nm
S_B_380	the background fluorescence at 380 nm
T_340	the exposure time at 340 nm
T_380	the exposure time at 380 nm
P	the number of pixels of the ROI
P_B	the number of pixels of the background region
ntransients	a vector of integers (above or equal to 1) specifying the indices of the transients to simulate
G	the gain of the CCD camera
s_ro	the standard deviation of the read-out process of the camera
noise	a logical. see details below

### Details

The way fluorescence values arise from intracellular calcium concentration values is described in the `fluor` function. Recording fluorescence with a CCD camera noises the photon counts, which can be described as the realization of a Poissonian process, the parameter of which is the fluorescence value itself. Ratiometric experiments are simulated thus simulated by drawing Poissonian samples from ideal fluorescence transients. These noisy data are then multiplied by the gain  $G$  of the CCD camera, and the standard deviation of the read-out noise ( $s_{ro}$ ) is finally added. In case noise is set to `FALSE`, fluorescence data arising from the fluorescence model are only digitized (rounded towards the nearest integer), but not drawn from a Poisson distribution.

### Value

An object of class "fluor\_rawdata", which is a data frame with four columns:

adu	the photon counts (or Analog-to-Digital Units) at both wavelengths, including background fluorescence
Time	the times at which each value in adu was recorded. For the background fluorescence, Time is set to NA
lambda	the wavelength at which each value in adu was recorded (a factor)
transient	the number of the fluorescence transient in the input data (can be 1, 2 or 3 for transient signals, and 0 for background measurements)

Data appear in this order : (1) the background fluorescence at 340 nm, (2) the fluorescence transient(s) at 340 nm, (3) the background fluorescence at 380 nm, (4) the fluorescence transient(s) at 380 nm. The object has also the following attributes:

tOn	the time at which the stimulation is applied (in s)
T_stim	a vector containing the exposure time at 340 nm and 380 nm
R_min	a copy of arg R_min
R_max	a copy of arg R_max
K_eff	a copy of arg K_eff
K_d	a copy of arg K_d
P	a copy of arg P
P_B	a copy of arg P_B
B_T	a copy of arg B_T
nb_B	a copy of arg nb_B
G	a copy of arg G
s_ro	a copy of arg s_ro

#### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

#### See Also

[ratioExpPhysio](#)

#### Examples

```
## Parameters of the monoexponential calcium transient
tOn <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=FALSE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=FALSE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=FALSE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=FALSE)

## Experiment-specific parameters
nb_B <- 1
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 200
P_B <- 200
phi <- 2
```

```

S_B_340 <- 30
S_B_380 <- 80

## Create a monoexponential calcium decay
Ca <- caMonoExp(t = Time, tOn = tOn,
               Ca0 = Ca0, dCa = dCa, tau = tau)

## Simulate the corresponding ratiometric experiment
simulData <- ratioExpSimul(nb_B = nb_B,
                          Ca = Ca,
                          R_min = R_min,
                          R_max = R_max,
                          K_eff = K_eff,
                          K_d = K_d,
                          B_T = B_T,
                          phi = phi,
                          S_B_340 = S_B_340,
                          S_B_380 = S_B_380,
                          T_340 = T_340,
                          T_380 = T_380,
                          P = P,
                          P_B = P_B,
                          ntransients = 1,
                          G = 1,
                          s_ro = 0)

## Plot the raw data
plot(simulData, numTransient=1)

```

---

ratioFitFromCa

*Perform a Fit on an Intracellular Calcium Concentration Vector*

---

## Description

The function `ratioFitFromCa` performs a fit on an intracellular calcium concentration transient. The transient is fitted with a mono- or a biexponential decay, depending on the value of `type`

## Usage

```

ratioFitFromCa(Ca, t, tOn, type = "mono", ig = NULL,
               Plot = FALSE, Fit = TRUE, AfterPeak = FALSE)

```

## Arguments

<code>Ca</code>	a vector of calcium concentration (in $\mu\text{M}$ )
<code>t</code>	a vector of latencies at which the calcium concentration was obtained (in s)
<code>tOn</code>	the time of the calcium concentration jump (in s)
<code>type</code>	a character string (either "mono" or "bi"), specifying the type of calcium exponential decay to consider

ig	an object of class "initial_guess", giving values of the calcium dynamics parameters to initiate the fitting process with using nls. This is a list with the following fields: ("log_Ca0", "log_dCa", "log_tau" for a monoexponential decay, eventually "mu" and "log_dtau" for a biexponential decay). If ig is not an object of class "initial_guess", initial guesses are estimated using the "igRatio" function
Plot	a logical value. Set to TRUE to plot the original signals, the initial guess and the fit results
Fit	a logical value. Set to TRUE to perform the fit, or to FALSE to compute an initial guess only
AfterPeak	a logical or numerical value. Set to FALSE to perform the fit on the whole fluorescence transients, to TRUE to consider only the part before the fluorescence jump and the convex part after the fluorescence peak, for both signals, or to an integer to skip a given number of samples after the fluorescence jump

### Details

This function can be used to fit any signal made with a first part corresponding to a baseline signal, then an (almost) instantaneous rise, and a mono- or bi-exponential return to baseline. Since the logarithm of all parameters (Ca0, dCa and tau) are fitted, the baseline and jump must be positive values.

This function uses `optim` to minimize the following residual sum of squares (RSS):

$$RSS_{ratio}(\theta_{Ca}) = (ca - [Ca^{2+}](t, \theta_{Ca}))^T \cdot \Omega^{-1} \cdot (ca - [Ca^{2+}](t, \theta_{Ca}))$$

where  $\theta_{Ca}$  stands for the parameters of the calcium dynamics model,  $ca$  is the vector of calcium concentration estimated by the ratiometric concentration and  $\Omega$  is the covariance matrix of the calcium signal. This covariance matrix is obtained with a first-order Taylor expansion of  $ca$  (see Joucla et al. (2009) for details).

The initial guesses for the calcium dynamics parameters, if not given in "ig", are calculated with the `igRatio` function

### Value

An object of class "ratio\_fit\_optim", which has the following attributes:

"Name"	a character string telling which type of fit has been performed
"type"	a copy of arg type
"Time"	a copy of arg t
"t0n"	a copy of arg t0n
"FitFunction"	the function passed to the <code>optim</code> formula
"RawData"	a copy of arg Ca
"RSSFunction"	the function that returns the residual sum of squares
"Subset"	the indices of the Time vector used for the fit
"VocChol"	the inverse of the Cholesky decomposition of the covariance matrix

**Author(s)**

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**See Also**

[transientConvexPart](#), [caFromDf](#), [igRatio](#), [ratioFitFromDf](#)

**Examples**

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=FALSE)

## Retrieve the calcium concentration from the data frame
Ca_noisy <- caFromDf(df = physioData,
                    numTransient = 2,
                    Plot = FALSE)

## Perform a ratiometric fit
physioRatioFit <- ratioFitFromCa(Ca = Ca_noisy,
                                 t = attr(Ca_noisy, "Time"),
                                 tOn = attr(Ca_noisy, "tOn"),
                                 type = "mono",
                                 AfterPeak = 14)
```

---

ratioFitFromDf

*Perform a Ratiometric Fit from a "fluo\_rawdata" object*

---

**Description**

The function `ratioFitFromDf` performs a fit on an intracellular calcium concentration transient obtained from a "fluo\_rawdata", after ratiometric transformation. The transient is fitted with a mono- or a biexponential decay, depending on the value of `type`

**Usage**

```
ratioFitFromDf(df, transients = 1, type = "mono", ig = NULL,
               Plot = FALSE, Fit = TRUE, AfterPeak = FALSE)
```

**Arguments**

df	a data frame of class "fluo_rawdata" containing all relevant information (fluorescence transients, background fluorescence, calibration parameters and exposure times). The structure of the input data frame must be the same as the one defined in <a href="#">ratioExpSimul</a>
transients	a vector of integers giving the numbers of the transients to fit
type	a character string (either "mono" or "bi"), specifying the type of calcium exponential decay to consider
ig	an object of class "initial_guess", giving values of the calcium dynamics parameters to initiate the fitting process with using nls. This is a list including the following fields: ("log_Ca0", "log_dCa", "log_tau" for a monoexponential decay, eventually "mu" and "log_dtau" for a biexponential decay). If ig is not an object of class "initial_guess", initial guesses are estimated using the "igRatio" function
Plot	a logical value. Set to TRUE to plot the original signals, the initial guess and the fit results
Fit	a logical value. Set to TRUE to perform the fit, or to FALSE to compute an initial guess only
AfterPeak	a logical or numerical value. Set to FALSE to perform the fit on the whole fluorescence transients, to TRUE to consider only the part before the fluorescence jump and the convex part after the fluorescence peak (for both signals), or to an integer to skip a given number of samples after the fluorescence jump

**Details**

The calcium concentration ( $[Ca^{2+}]$ ) is deduced from the ratiometric transformation (see [caFromRatio](#)). A mono- or bi-exponential decay is fitted to the  $[Ca^{2+}]$  signal, using the `optim` function. For more details, see [ratioFitFromCa](#).

**Value**

An object that inherits from both "nls" and either "ratio\_fit" or "ratio\_fit\_list" classes, depending whether transients is a single value or a vector. In the latter case, the output "ratio\_fit\_list" object is a list of "ratio\_fit" objects, which have the following attributes:

"Name"	a character string telling which type of fit has been performed
"Time"	the whole time vector (in s)
"RawData"	the $[Ca^{2+}]$ signal deduced from the ratiometric transformation. This signal, which is the one passed to the nls formula, has two attributes: "var" is the vector of variances estimated from the error propagation method, and "Time" is the vector of latencies at which fluorescence measurements were performed

"RawDataFrame" a copy of the input data frame  
 "FitFunction" the function passed to the nls formula  
 "Subset" the indices of the Time vector used for the fit

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

### See Also

[transientConvexPart](#), [caFromDf](#), [igRatio](#), [ratioFitFromCa](#)

### Examples

```
## Load the data from cockroach olfactory interneurons
data(inVitro)

## Calibrated parameters
R_min <- list(value=0.136, mean=0.136, se=0.00363, USE_se=TRUE)
R_max <- list(value=2.701, mean=2.701, se=0.151, USE_se=TRUE)
K_eff <- list(value=3.637, mean=3.637, se=0.729, USE_se=TRUE)
K_d <- list(value=0.583, mean=0.583, se=0.123, USE_se=TRUE)

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=2,
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
                             G=0.146, s_ro=16.4,
                             alphamethod=FALSE)

## Perform a ratiometric fit
physioRatioFit <- ratioFitFromDf(df = physioData,
                                transients = 2,
                                AfterPeak = 14)

## Print the class of 'physioRatioFit'
print(class(physioRatioFit))

## Create the data frame containing the physiological data
## (experiment #2, stimulation #2)
## G and s_ro are the respectively the gain of the CCD camera
## and the standard deviation of its read-out process
physioData <- ratioExpPhysio(dataset="inVitro",
                             expe=2, stim=c(2,3),
                             idxOn=10,
                             R_min=R_min, R_max=R_max,
                             K_eff=K_eff, K_d=K_d,
```

```
G=0.146, s_ro=16.4,
alphamethod=FALSE)

## Perform a ratiometric fit
physioRatioFit <- ratioFitFromDf(df = physioData,
                                transients = c(2,3),
                                AfterPeak = 14)

## Print the class of 'physioRatioFit'
print(class(physioRatioFit[[2]]))
```

---

transientConvexPart *Select the Convex or Concave Part of a Transient*

---

### Description

The function `transientConvexPart` extracts the indices of a given transient where the signal is monotonically convex or concave, after a local peak (maximum or minimum) at the beginning of the transient

### Usage

```
transientConvexPart(transient, t = 1, t0n = 1)
```

### Arguments

transient	the vector to work on
t	a vector of time values at which transient has been obtained (in s)
t0n	the time of the transient jump (in s)

### Details

The function `transientConvexPart` is designed to work on transients of the following form: First, prior to `t0n`, a baseline; Then, at `t0n`, a sharp (positive or negative) jump, which leads to a global maximum or minimum; Finally, a monotonic return to baseline. Real  $[Ca^{2+}]$  or fluorescence transients, on which this function is applied, are generally of this form. The function smoothes the input transient, finds the time (after the peak) at which the second derivative changes sign, and returns its index

### Value

An integer, which is the index of the transient (after the peak) at which the second derivative changes sign, and returns its index

### Author(s)

Sebastien Joucla <sebastien.joucla@parisdescartes.fr>

**Examples**

```

## Parameters of the monoexponential calcium transient
t0n <- 1
Time <- seq(0,12,length.out=160)
Ca0 <- 0.10
dCa <- 0.25
tau <- 1.5

## Calibration parameters
R_min <- 0.136
R_max <- 2.701
K_eff <- 3.637
K_d <- 0.583

## Experiment-specific parameters
nb_B <- 5
B_T <- 100.0
T_340 <- 0.015
T_380 <- 0.006
P <- 200
P_B <- 200
phi <- 20
S_B_340 <- 300
S_B_380 <- 800

## Create a monoexponential calcium decay
Ca_Mono <- caMonoExp(t=Time,
                    t0n=t0n,
                    Ca0=Ca0,
                    dCa=dCa,
                    tau=tau)

## Simulate the corresponding ratiometric experiment
df_Mono <- ratioExpSimul(nb_B = nb_B, Ca = Ca_Mono,
                       R_min = R_min, R_max = R_max,
                       K_eff = K_eff, K_d = K_d,
                       B_T = B_T, phi = phi, P = P, P_B = P_B,
                       ntransients = 1,
                       S_B_340 = S_B_340, S_B_380 = S_B_380,
                       T_340 = T_340, T_380 = T_380, G = 1, s_ro = 0)

## Get the fluorescence transients at 340 and 380 nm, respectively
t <- with(df_Mono,Time[!is.na(Time) & lambda==340])
adu_340 <- with(df_Mono,adu[!is.na(Time) & lambda==340])
adu_380 <- with(df_Mono,adu[!is.na(Time) & lambda==380])

## Calculate the indices of convex/concave starts at both wavelengths
idx_340 <- transientConvexPart(t = t, t0n = t0n, transient = adu_340)
idx_380 <- transientConvexPart(t = t, t0n = t0n, transient = adu_380)

## Plot both transients, with a specific color for the
## portions of interest

```

```
layout(matrix(c(1,2),ncol=1))

plot(t[c(1:idx_340)], adu_340[c(1:idx_340)], type="l",
      xlim = c(Time[1],Time[length(Time)]))
lines(t[c(idx_340:length(adu_340))],
      adu_340[c(idx_340:length(adu_340))], col="blue")

plot(t[c(1:idx_380)], adu_380[c(1:idx_380)], type="l",
      xlim = c(Time[1], Time[length(Time)]))
lines(t[c(idx_380:length(adu_380))],
      adu_380[c(idx_380:length(adu_380))], col="red")
```

# Index

## \*Topic **datasets**

inVitro, [28](#)

## \*Topic **package**

CalciOMatic-package, [2](#)

anova4Fits, [3](#)

caBiExp, [5](#), [12](#), [13](#), [19](#), [35](#)

caFromDf, [7](#), [56](#), [58](#)

caFromRatio, [7](#), [9](#), [57](#)

CalciOMatic (CalciOMatic-package), [2](#)

CalciOMatic-package, [2](#)

caMonoBiExpFromIG, [6](#), [11](#), [13](#), [27](#)

caMonoExp, [6](#), [12](#), [13](#), [19](#), [35](#)

directFit, [4](#), [14](#), [31](#), [33](#), [48](#)

fluo, [18](#), [24](#), [52](#)

getThetaRatio, [20](#)

igDirect, [15](#), [16](#), [22](#), [27](#)

igRatio, [24](#), [26](#), [56](#), [58](#)

inVitro, [28](#), [48](#)

mkFluo4DirectFit, [16](#), [30](#), [33](#)

mkFunction4DirectFit, [31](#), [32](#)

mkFunction4RatioFit, [34](#)

plot.direct\_fit, [36](#), [39](#), [41](#), [44](#), [47](#)

plot.fluo\_rawdata, [37](#), [38](#), [41](#), [44](#), [47](#), [48](#)

plot.ratio\_fit, [37](#), [39](#), [40](#), [44](#), [47](#)

plot.ratio\_fit\_list, [37](#), [39](#), [41](#), [42](#), [47](#)

plotCalciOMatic, [37](#), [39](#), [41](#), [44](#), [45](#)

ratioExpPhysio, [7](#), [16](#), [38](#), [48](#), [53](#)

ratioExpSimul, [7](#), [14](#), [16](#), [38](#), [50](#), [51](#), [57](#)

ratioFitFromCa, [35](#), [54](#), [57](#), [58](#)

ratioFitFromDf, [35](#), [48](#), [56](#), [56](#)

transientConvexPart, [16](#), [56](#), [58](#), [59](#)