Package 'MBmca'

August 29, 2016

Type Package

Title Nucleic Acid Melting Curve Analysis on Microbead Surfaces with R

Version 0.0.3-5

LazyData true

Depends R (>= 3.0.0), robustbase (>= 0.9), chipPCR (>= 0.0.7)

Date 2015-01-14

Description The MBmca package provides data sets and lightweight utilities for
nucleic acid melting curve analysis and presentation on microbead surfaces
but also for reactions in solution (e.g., qPCR).

License GPL (>= 2)

URL https://github.com/michbur/MBmca/

Author Stefan Roediger [aut, cre],
Michal Burdukiewicz [ctb]

Maintainer Stefan Roediger <stefan_roediger@gmx.de>

NeedsCompilation no

Repository CRAN

Date/Publication 2015-01-13 13:58:41

R topics documented:

MBmca-package	*Microbead Surface Nucleic Acid Melting Curve Analysis*

Description

Nucleic acid Melting Curve Analysis is a powerful method to investigate the interaction of double stranded nucleic acids. The MBmca package provides data sets and lightweight utilities for nucleic acid melting curve analysis and presentation on microbead surfaces. However, the function of the package can also be used for the analysis of reactions in solution (e.g., qPCR). Methods include melting curve data pre-processing (smooth, normalize, rotate, background subtraction), data inspection (comparison of multiplex melting curves) with location parameters (mean, median), deviation parameters (standard of the melting peaks including the second derivative. The second derivative melting peaks is implemented as parameter to further characterize the melting behavior. Plot functions to illustrate data quality, smoothed curves and derivatives are available too.

Details

Package:	MBmca
Type:	Package
Version:	0.0.3-4
Date:	2014-08-17
License: GPL (>= 2)	

Author(s)

Stefan Roediger <stefan_roediger@gmx.de>

References

A Highly Versatile Microscope Imaging Technology Platform for the Multiplex Real-Time Detection of Biomolecules and Autoimmune Antibodies. S. Roediger, P. Schierack, A. Boehm, J. Nitschke, I. Berger, U. Froemmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W. Lehmann and C. Schroeder. *Advances in Biochemical Bioengineering/Biotechnology*. 133:35–74, 2013. http://www.ncbi.nlm.nih.gov/pubmed/22437246

Surface Melting Curve Analysis with R. S. Roediger, A. Boehm and I. Schimke. *The R Journal*. 5(2):37–52, 2013. http://journal.r-project.org

Nucleic acid detection based on the use of microbeads: a review. S. Roediger, C. Liebsch, C. Schmidt, W. Lehmann, U. Resch-Genger, U. Schedler, P. Schierack. *Microchim Acta* 2014:1–18. DOI: 10.1007/s00604-014-1243-4

`diffQ`	*Calculation of the melting temperature (Tm) from the first derivative*

Description

`diffQ` is used to calculate the melting temperature (Tm) but also for elementary graphical operations (e.g., show the Tm or the derivative). It does not require smoothed data for the MCA. The parameter `rsm` can be used to double the temperature resolution by calculation of the mean temperature and mean fluorescence. Note: mcaSmoother has the n parameter with a similar functionality. First the approximate Tm is determined as the `min()` and/or `max()` from the first derivative. The first numeric derivative (Forward Difference) is estimated from the values of the function values obtained during an experiment since the exact function of the melting curve is unknown. The method used in `diffQ` is suitable for independent variables that are equally and unequally spaced. Alternatives for the numerical differentiation include Backward Differences, Central Differences or Three-Point (Forward or Backward) Difference based on Lagrange Estimation (currently not implemented in `diffQ`). The approximate peak value is the starting-point for a function based calculation. The function takes a defined number n (maximum 8) of the left and the right neighbor values and fits a quadratic polynomial. The quadratic regression of the X (temperature) against the Y (fluorescence) range gives the coefficients. The optimal quadratic polynomial is chosen based on the highest adjusted R-squared value. `diffQ` returns an objects of the class `list`. To accessing components of lists is done as described elsewhere either by name or by number. `diffQ` has a simple plot function. However, for sophisticated analysis and plots its recommended to use `diffQ` as presented in the examples as part of algorithms.

Usage

```
diffQ(xy, fct = min, fws = 8, col = 2, plot = FALSE, verbose = FALSE,
  warn = TRUE, peak = FALSE, negderiv = TRUE, deriv = FALSE,
  derivlimits = FALSE, derivlimitsline = FALSE, vertiline = FALSE,
  rsm = FALSE, inder = FALSE)
```

Arguments

`xy`	is a `data.frame` containing in the first column the temperature and in the second column the fluorescence values. Preferably the output from `mcaSmoother` is used.
`fct`	accepts `min` or `max` as option and is used to define whether to find a local minimum ("negative peak") or local maximum ("positive peak").
`fws`	defines the number (n) of left and right neighbors to use for the calculation of the quadratic polynomial.
`col`	is a graphical parameter used to define the length of the line used in the plot.
`plot`	shows a plot of a single melting curve. To draw multiple curves in a single plot set `plot = FALSE` and create and empty plot instead (see examples).
`verbose`	shows additional information (e.g., approximate derivative, ranges used for calculation, approximate Tm) of the calculation.

warn	diffQ tries to keep the user as informed as possible about the quality of the analysis. However, in some scenarios are the warning and message about analysis not needed or disturbing. warn can be used to stop the flodding of the output.
peak	shows the peak in the plot (see examples).
negderiv	uses the positive first derivative instead of the negative.
deriv	shows the first derivative with the color assigned to col (see examples).
derivlimits	shows the neighbors (fws) used to calculate the Tm as points in the plot (see examples).
derivlimitsline	
	shows the neighbors (fws) used to calculate the Tm as line in the plot (see examples).
vertiline	draws a vertical line at the Tms (see examples).
rsm	performs a doubling of the temperature resolution by calculation of the mean temperature and mean fluorescence between successive temperature steps. Note: mcaSmoother has the "n" parameter with a similar but advanced functionality.
inder	Interpolates first derivatives using the five-point stencil. See chipPCR package for details.

Value

diffQ()	returns a comprehensive list (if parameter verbose is TRUE) with results from the first derivative. The list includes a data.frame of the derivative ("xy"). The temperature range ("limits.xQ") and fluorescence range ("limits.diffQ") to calculate the peak value. "fluo.x" is the approximate fluorescence at the approximate melting temperature. The calculated melting temperature ("Tm") with the corresponding fluorescence intensity ("fluoTm"). The number of neighbors ("fws"), the adjusted R-squared ("adj.r.squared") and the normalized-root-mean-squared-error ("NRMSE") to fit. The quality of the calculated melting temperature ("Tm") can be checked with devsum which reports the relative deviation (in percent) between the approximate melting temperature and the calculated melting temperature, if NRMSE is less than 0.08 and the adjusted R-squared is less than 0.85. A relative deviation larger than 10 percent will result in a warning. Reducing fws might improve the result.
Tm	returns the calculated melting temperature ("Tm").
fluoTm	returns the calculated fluorescence at the calculated melting temperature.
Tm.approx	returns the approximate melting temperature.
fluo.x	returns the approximate fluorescence at the calculated melting temperature.
xy	returns the approximate derivative value used for the calculation of the melting peak.
limits.xQ	returns a data range of temperature values used to calculate the melting temperature.
limits.diffQ	returns a data range of fluorescence values used to calculate the melting temperature.

adj.r.squared	returns the adjusted R-squared from the quadratic model fitting function (see also fit).
NRMSE	returns the normalized root-mean-squared-error (NRMSE) from the quadratic model fitting function (see also fit).
fws	returns the number of points used for the calculation of the melting temperature.
devsum	returns measures to show the difference between the approximate and calculated melting temperature.
temperature	returns measures to investigate the temperature resolution of the melting curve. Raw fluorescence measurements at irregular temperature resolutions (intervals) can introduce artifacts and thus lead to wrong melting point estimations.
temperature$T.delta	
	returns the difference between two successive temperature steps.
temperature$mean.T.delta	
	returns the mean difference between two temperature steps.
temperature$sd.T.delta	
	returns the standard deviation of the temperature.
temperature$RSD.T.delta	
	returns the relative standard deviation (RSD) of the temperature in percent.
fit	returns the summary of the results of the quadratic model fitting function.

Author(s)

Stefan Roediger

References

A Highly Versatile Microscope Imaging Technology Platform for the Multiplex Real-Time Detection of Biomolecules and Autoimmune Antibodies. S. Roediger, P. Schierack, A. Boehm, J. Nitschke, I. Berger, U. Froemmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W. Lehmann and C. Schroeder. *Advances in Biochemical Bioengineering/Biotechnology*. 133:33–74, 2013. http://www.ncbi.nlm.nih.gov/pubmed/22437246

Nucleic acid detection based on the use of microbeads: a review. S. Roediger, C. Liebsch, C. Schmidt, W. Lehmann, U. Resch-Genger, U. Schedler, P. Schierack. *Microchim Acta* 2014:1–18. DOI: 10.1007/s00604-014-1243-4

Roediger S, Boehm A, Schimke I. Surface Melting Curve Analysis with R. *The R Journal* 2013;5:37–53.

See Also

diffQ2, mcaSmoother

Examples

```
# First Example
# Plot the first derivative of different samples for single melting curve
# data. Note that the argument "plot" is TRUE.
```

```
data(MultiMelt)
par(mfrow = c(1,2))
sapply(2L:14, function(i) {
        tmp <- mcaSmoother(MultiMelt[, 1], MultiMelt[, i])
        diffQ(tmp, plot = TRUE)
  }
)
par(mfrow = c(1,1))
# Second example
# Plot the first derivative of different samples from MultiMelt
# in a single plot.
data(MultiMelt)

# First create an empty plot
plot(NA, NA, xlab = "Temperature", ylab ="-d(refMFI)/d(T)",
        main = "Multiple melting peaks in a single plot", xlim = c(65,85),
        ylim = c(-0.4,0.01), pch = 19, cex = 1.8)
# Prepossess the selected melting curve data (2,6,12) with mcaSmoother
# and apply them to diffQ. Note that the argument "plot" is FALSE
# while other arguments like derivlimitsline or peak are TRUE.
sapply(c(2,6,12), function(i) {
tmp <- mcaSmoother(MultiMelt[, 1], MultiMelt[, i],
    bg = c(41,61), bgadj = TRUE)
diffQ(tmp, plot = FALSE, derivlimitsline = TRUE, deriv = TRUE,
      peak = TRUE, derivlimits = TRUE, col = i, vertiline = TRUE)
  }
)
legend(65, -0.1, colnames(MultiMelt[, c(2,6,12)]), pch = c(15,15,15),
col = c(2,6,12))

# Third example
# First create an empty plot
plot(NA, NA, xlim = c(50,85), ylim = c(-0.4,2.5),
     xlab = "Temperature",
     ylab ="-refMFI(T) | refMFI'(T) | refMFI''(T)",
     main = "1st and 2nd Derivatives",
     pch = 19, cex = 1.8)

# Prepossess the selected melting curve data with mcaSmoother
# and apply them to diffQ and diffQ2. Note that
# the argument "plot" is FALSE while other
# arguments like derivlimitsline or peak are TRUE.

tmp <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 2],
    bg = c(41,61), bgadj = TRUE)
lines(tmp, col= 1, lwd = 2)

# Note the different use of the argument derivlimits in diffQ and diffQ2
diffQ(tmp, fct = min, derivlimitsline = TRUE, deriv = TRUE,
    peak = TRUE, derivlimits = FALSE, col = 2, vertiline = TRUE)
diffQ2(tmp, fct = min, derivlimitsline = TRUE, deriv = TRUE,
    peak = TRUE, derivlimits = TRUE, col = 3, vertiline = TRUE)
```

```
# Add a legend to the plot
legend(65, 1.5, c("Melting curve",
  "1st Derivative",
  "2nd Derivative"),
  pch = c(19,19,19), col = c(1,2,3))

# Fourth example
# Different curves may potentially have different quality in practice.
# For example, using data from MultiMelt should return a
# valid result and plot.
data(MultiMelt)

diffQ(cbind(MultiMelt[, 1], MultiMelt[, 2]), plot = TRUE)$Tm
# limits_xQ
#   77.88139

# Imagine an experiment that went terribly wrong. You would
# still get an estimate for the Tm. The output from diffQ,
# with an error attached, lets the user know that this Tm
# is potentially meaningless. diffQ() will give several
# warning messages.

set.seed(1)
y = rnorm(55,1.5,.8)
diffQ(cbind(MultiMelt[, 1],y), plot = TRUE)$Tm

# The distribution of the curve data indicates noise.
# The data should be visually inspected with a plot
# (see examples of diffQ). The Tm calculation (fit,
# adj. R squared ~ 0.157, NRMSE ~ 0.279) is not optimal
# presumably due to noisy data. Check raw melting
# curve (see examples of diffQ).
# Calculated Tm
#       56.16755


# Sixth example
# Curves may potentially have a low temperature resolution. The rsm
# parameter can be used to double the temperature resolution.
# Use data from MultiMelt column 15 (MLC2v2).
data(MultiMelt)
tmp <- cbind(MultiMelt[, 1], MultiMelt[, 15])

# Use diffQ without and with the rsm parameter and plot
# the results in a single row
par(mfrow = c(1,2))

diffQ(tmp, plot = TRUE)$Tm
  text(60, -0.15, "without rsm parameter")

diffQ(tmp, plot = TRUE, rsm = TRUE)$Tm
  text(60, -0.15, "with rsm parameter")
par(mfrow = c(1,1))
```

---

diffQ2 *Calculation of the melting temperatures (Tm, Tm1D2 and Tm2D2) from the first and the second derivative*

Description

diffQ2() calls instances of diffQ() to calculate the Tm1D2 and Tm2D2. The options are similar to diffQ(). Both diffQ() and diffQ2() return objects of the class list. To accessing components of lists is done as described elsewhere either be name or by number. diffQ2 has no standalone plot function. For sophisticated analysis and plots its recommended to use diffQ2 as presented in the examples as part of algorithms.

Usage

```
diffQ2(xy, fct = max, fws = 8, col = 2, plot = FALSE, verbose = FALSE,
  peak = FALSE, deriv = FALSE, negderiv = TRUE, derivlimits = FALSE,
  derivlimitsline = FALSE, vertiline = FALSE, rsm = FALSE,
  inder = FALSE, warn = TRUE)
```

Arguments

xy	is a data.frame containing in the first column the temperature and in the second column the fluorescence values. Preferably the output from mcaSmoother is used.
fct	accepts min or max as option and is used to define whether to find a local minimum ("negative peak") or local maximum ("positive peak").
fws	defines the number (n) of left and right neighbors to use for the calculation of the quadratic polynomial.
col	is a graphical parameter used to define the length of the line used in the plot.
plot	shows a plot of a single melting curve with (Tm) as vertical line and the second derivatives (Tm1D2 and Tm2D2). To draw multiple curves in a single plot set plot = FALSE and create and empty plot instead (see examples).
verbose	shows additional information (e.g., first and second approximate derivatives, ranges used for calculation, approximate Tm, Tm1D2, Tm2D2) of the calculation.
peak	shows the peak in the plot.
deriv	shows the first derivative with the color assigned to col (see examples).
negderiv	calculates the negative derivative (default). If FALSE the positive first negative is calculated.
derivlimits	shows the number (n) used to calculate the Tm as points in the plot (see examples).
derivlimitsline	
	shows the number (n) used to calculate the Tm as line in the plot (see examples).
vertiline	draws a vertical line at the Tms (see examples).

rsm	performs a doubling of the temperature resolution by calculation of the mean temperature and mean fluorescence between successive temperature steps. Note: mcaSmoother has the "n" parameter with a similar but advanced functionality.
inder	Interpolates derivatives using the five-point stencil. See chipPCR package for details.
warn	diffQ tries to keep the user as informed as possible about the quality of the analysis. However, in some scenarios are the warning and message about analysis not needed or disturbing. warn can be used to stop the flodding of the output.

Value

$TmD1	TmD1 returns a comprehensive list (if parameter verbose is TRUE) with results from the first derivative. The list includes a data.frame of the derivative ("xy"). The temperature range ("limits.xQ") and fluorescence range ("limits.diffQ") to calculate the peak value. "fluo.x" is the approximate fluorescence at the approximate melting temperature. The calculated melting temperature ("Tm") with the corresponding fluorescence intensity ("fluoTm"). The number of points ("fws") and the adjusted R-squared ("adj.r.squared") to fit.
$TmD1$Tm	returns the calculated melting temperature ("Tm") from the first derivative.
$TmD1$fluoTm	returns the calculated fluorescence at the calculated melting temperature ("Tm").
$TmD1$Tm.approx	returns the approximate melting temperature ("Tm") from the first derivative.
$TmD1$fluo.x	returns the approximate fluorescence at the calculated melting temperature ("Tm").
$TmD1$xy	is a data.frame containing in the first column the temperature and in the second column the fluorescence values. Preferably the output from mcaSmoother is used.
$TmD1$limits.xQ	returns a data range of temperature values used to calculate the melting temperature.
$TmD1$limits.diffQ	returns a data range of fluorescence values used to calculate the melting temperature.
$TmD1$adj.r.squared	returns the adjusted R-squared from the quadratic model fitting function (see also fit) of the first derivative.
$TmD1$NRMSE	returns the normalized root-mean-squared-error (NRMSE) from the quadratic model fitting function (see also fit) of the first derivative.
$TmD1$fws	returns the number of points used for the calculation of the melting temperature of the first derivative.
$TmD1$devsum	returns measures to show the difference between the approximate and calculated melting temperature of the first derivative.
$TmD1$fit	returns the summary of the results of the quadratic model fitting function of the first derivative.
$Tm1D2	returns the "left" melting temperature ("Tm1D2 ") values from the second derivative.

$Tm1D2$Tm   returns the "left" calculated melting temperature ("Tm1D2") from the second
                   derivative.

$Tm1D2$fluoTm   returns the "left" calculated fluorescence at the calculated melting temperature
                   ("Tm1D2") from the second derivative.

$Tm1D2$Tm.approx
                   returns the "left" approximate melting temperature ("Tm1D2") from the second
                   derivative.

$Tm1D2$fluo.x   returns the "left" approximate fluorescence at the calculated melting temperature
                   ("Tm1D2") from the second derivative.

$Tm1D2$xy   is a data.frame containing in the first column the temperature and in the second
                   column the fluorescence values of the "left" melting temperature ("Tm1D2")
                   from the second derivative. Preferably the output from mcaSmoother is used.

$Tm1D2$limits.xQ
                   returns a data range of temperature values used to calculate the melting temper-
                   ature of the "left" melting temperature ("Tm1D2") from the second derivative.

$Tm1D2$limits.diffQ
                   returns a data range of fluorescence values used to calculate the melting temper-
                   ature of the "left" melting temperature ("Tm1D2") from the second derivative.

$Tm1D2$adj.r.squared
                   returns the adjusted R-squared from the quadratic model fitting function (see
                   also fit) of the "left" melting temperature ("Tm1D2") from the second deriva-
                   tive.

$Tm1D2$NRMSE   returns normalized root-mean-squared-error (NRMSE) from the quadratic model
                   fitting function (see also fit) of the "left" melting temperature ("Tm1D2") from
                   the second derivative.

$Tm1D2$fws   returns the number of points used for the calculation of the melting temperature
                   of the "left" melting temperature ("Tm1D2") from the second derivative.

$Tm1D2$devsum   returns measures to show the difference between the approximate and alculated
                   melting temperature of the "left" melting temperature ("Tm1D2") from the sec-
                   ond derivative.

$Tm1D2$fit   returns the summary of the results of the quadratic model fitting function of the
                   "left" melting temperature ("Tm1D2") from the second derivative.

$Tm2D2   returns the "right" melting temperature ("Tm2D2 ") values from the second
                   derivative.

$Tm2D2$Tm   returns the "right" calculated melting temperature ("Tm2D2") from the second
                   derivative.

$Tm2D2$fluoTm   returns the "right" calculated fluorescence at the calculated melting temperature
                   ("Tm2D2") from the second derivative.

$Tm2D2$Tm.approx
                   returns the "right" approximate melting temperature ("Tm1D2") from the second
                   derivative.

$Tm2D2$fluo.x   returns the "left" approximate fluorescence at the calculated melting temperature
                   ("Tm2D2") from the second derivative.

$Tm2D2$xy   is a data.frame containing in the first column the temperature and in the second column the fluorescence values of the "right" melting temperature ("Tm2D2") from the second derivative. Preferably the output from mcaSmoother is used.

$Tm2D2$limits.xQ

             returns a data range of temperature values used to calculate the melting temperature of the "right" melting temperature ("Tm2D2") from the second derivative.

$Tm2D2$limits.diffQ

             returns a data range of fluorescence values used to calculate the melting temperature of the "right" melting temperature ("Tm"D2") from the second derivative.

$Tm2D2$adj.r.squared

             returns the adjusted R-squared from the quadratic model fitting function (see also fit) of the "right" melting temperature ("Tm2D2") from the second derivative.

$Tm2D2$NRMSE   returns normalized root-mean-squared-error (NRMSE) from the quadratic model fitting function (see also fit) of the "right" melting temperature ("Tm2D2") from the second derivative.

$Tm2D2$fws   returns the number of points used for the calculation of the melting temperature of the "right" melting temperature ("Tm2D2") from the second derivative.

$Tm2D2$devsum   returns measures to show the difference between the approximate and calculated melting temperature of the "right" melting temperature ("Tm2D2") from the second derivative.

$Tm2D2$fit   returns the summary of the results of the quadratic model fitting function of the "right" melting temperature ("Tm2D2") from the second derivative.

$xTm1.2.D2   returns only the "left" and right calculated melting temperature ("Tm1D2, Tm2D2") from the second derivative.

$yTm1.2.D2   returns only the "left" and right calculated fluorescence ("Tm1D2, Tm2D2") from the second derivative.

$temperature   returns measures to investigate the temperature resolution of the melting curve. Raw fluorescence measurements at irregular temperature resolutions (intervals) can introduce artifacts and thus lead to wrong melting point estimations.

$temperature$T.delta

             returns the difference between two successive temperature steps.

$temperature$mean.T.delta

             returns the mean difference between two temperature steps.

$temperature$sd.T.delta

             returns the standard deviation of the temperature.

$temperature$RSD.T.delta

             returns the relative standard deviation (RSD) of the temperature in percent.

Author(s)

Stefan Roediger

References

A Highly Versatile Microscope Imaging Technology Platform for the Multiplex Real-Time Detection of Biomolecules and Autoimmune Antibodies. S. Roediger, P. Schierack, A. Boehm, J. Nitschke, I. Berger, U. Froemmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W. Lehmann and C. Schroeder. *Advances in Biochemical Bioengineering/Biotechnology*. 133:33–74, 2013. http://www.ncbi.nlm.nih.gov/pubmed/22437246

Nucleic acid detection based on the use of microbeads: a review. S. Roediger, C. Liebsch, C. Schmidt, W. Lehmann, U. Resch-Genger, U. Schedler, P. Schierack. *Microchim Acta* 2014:1–18. DOI: 10.1007/s00604-014-1243-4

Roediger S, Boehm A, Schimke I. Surface Melting Curve Analysis with R. *The R Journal* 2013;5:37–53.

See Also

diffQ, mcaSmoother

Examples

```
# First Example
# Plot the first and the second derivative melting curves of MLC-2v
# for a single melting curve. Should give a warning message but the graph
# will show you that the calculation is ok
data(MultiMelt)
tmp <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 14])
diffQ2(tmp, fct = min, verbose = FALSE, plot = TRUE)


# Second Example
# Calculate the maximum fluorescence of a melting curve, Tm,
# Tm1D2 and Tm2D2 of HPRT1 for 12 microbead populations and assign the
# values to the matrix HPRT1
data(MultiMelt)
HPRT1 <- matrix(NA,12,4,
dimnames = list(colnames(MultiMelt[, 2L:13]),
    c("Fluo", "Tm", "Tm1D2", "Tm2D2")))
for (i in 2L:13) {
      tmp <- mcaSmoother(MultiMelt[, 1],
                    MultiMelt[, i])
      tmpTM <- diffQ2(tmp, fct = min, verbose = TRUE)
      HPRT1[i-1, 1] <- max(tmp$y)
      HPRT1[i-1, 2] <- tmpTM$TmD1$Tm
      HPRT1[i-1, 3] <- tmpTM$Tm1D2$Tm
      HPRT1[i-1, 4] <- tmpTM$Tm2D2$Tm
}
HPRT1


# Third Example
# Use diffQ2 to determine the second derivative.

data(MultiMelt)
HPRT1 <- matrix(NA,12,4,
```

```
dimnames = list(colnames(MultiMelt[, 2L:13]),
    c("Fluo", "Tm", "Tm1D2", "Tm2D2")))
for (i in 2L:13) {
        tmp <- mcaSmoother(MultiMelt[, 1],
                    MultiMelt[, i])
        tmpTM <- diffQ2(tmp, fct = min, verbose = TRUE)
        HPRT1[i-1, 1] <- max(tmp[["y.sp"]])
        HPRT1[i-1, 2] <- tmpTM[["TmD1"]][["Tm"]]
        HPRT1[i-1, 3] <- tmpTM[["Tm1D2"]][["Tm"]]
        HPRT1[i-1, 4] <- tmpTM[["Tm2D2"]][["Tm"]]
}
plot(HPRT1[, 1], HPRT1[, 2],
        xlab = "refMFI", ylab = "Temperature",
        main = "HPRT1", xlim = c(2.1,2.55),
        ylim = c(72,82), pch = 19,
        col = 1:12, cex = 1.8)
points(HPRT1[, 1], HPRT1[, 3], pch = 15)
points(HPRT1[, 1], HPRT1[, 4], pch = 15)
abline(lm(HPRT1[, 2] ~ HPRT1[, 1]))
abline(lm(HPRT1[, 3] ~ HPRT1[, 1]))
abline(lm(HPRT1[, 4] ~ HPRT1[, 1]))

# Fourth Example
# Use diffQ2 with inder parameter to determine the second derivative.
data(MultiMelt)

tmp <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 14])
diffQ2(tmp, fct = min, verbose = FALSE, plot = TRUE, inder = FALSE)
diffQ2(tmp, fct = min, verbose = FALSE, plot = TRUE, inder = TRUE)
par(mfrow = c(1,1))
```

DMP   *Bimodal melting curve experiment on the surface of microbeads.*

Description

A melting curve experiment with six microbead populations and short oligonucleotide probes (direct hybridization). Detection probes for human VIM (vimentin), MLC-2v (myosin regulatory light chain 2, ventricular/cardiac muscle isoform), SERCA2 (Atp2a2 - ATPase, Calcium-transporting ATPase sarcoplasmic reticulum type, slow twitch skeletal muscle isoform), HRPT1 (hyperparathyroidism 1) and the artificial sequences Poly(dA)20 (20 nt of dA) and aCS (artificial Control Sequence).

Format

A data frame with the melting curves of six different capture and detection probe pairs on six microbeads populations for VIM, MLC-2v, SERCA2, Poly(dA)20, aCS and HPRT1. First column contains the temperature (in degree Celsius, 1 degree Celsius per step) followed by melting curves of VIM, MLC-2v, SERCA2, Poly(dA)20, aCS and HPRT1 with bimodal melting pattern. The dyes and quencher used were Atto 647N and BHQ2.

T a numeric vector, Temperature in degree Celsius.

VIM.no.Q a numeric vector, VIM without quencher and without Poly(dT)20 region.

MLC2v.w.Q.w.dT20 a numeric vector, MLC-2v with quencher-labeled detection probe and fluorescent Poly(dA)20 detection probe.

SERCA2.no.Q.w.dT20 a numeric vector, SERCA2 without quencher-labeled detection probe and Poly(dA)20 detection probe.

PolydA20.w.Q a numeric vector, Poly(dT)20 with fluorescent Poly(dA)20 detection probe (quencher labeled).

aCS.w.Q.w.dT20 a numeric vector, artificial Control Sequence without quencher-labeled detection probe and fluorescent Poly(dA)20 detection probe.

HPRT1.no.Q.w.dT20 a numeric vector, HPRT1 without quencher-labeled detection probe and fluorescent Poly(dA)20 detection probe.

Source

Data were measured with the VideoScan platform:

A Highly Versatile Microscope Imaging Technology Platform for the Multiplex Real-Time Detection of Biomolecules and Autoimmune Antibodies. S. Roediger, P. Schierack, A. Boehm, J. Nitschke, I. Berger, U. Froemmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W. Lehmann and C. Schroeder. *Advances in Biochemical Bioengineering/Biotechnology*. 133:35–74, 2013. http://www.ncbi.nlm.nih.gov/pubmed/22437246

Surface Melting Curve Analysis with R. S. Roediger, A. Boehm and I. Schimke. *The R Journal*. 5(2):37–52, 2013. http://journal.r-project.org/

See Also

MFIerror, mcaSmoother, diffQ, diffQ, MultiMelt, DualHyb

Examples

```
data(DMP)
```

DualHyb   *Surface melting curve data from direct hybridization experiment of short oligonucleotide probes with bimodal melting curve pattern.*

Description

A melting curve experiment with four microbead populations and short oligonucleotide probes (direct hybridization) and longer probes (dual-hybridization probes) capture probe. Detection probes for human VIM (vimentin), MLC-2v (myosin regulatory light chain 2, ventricular/cardiac muscle isoform) and SERCA2 (Atp2a2 - ATPase, Calcium-transporting ATPase sarcoplasmic reticulum type, slow twitch skeletal muscle isoform). One sequence of VIM contained a mutation at position 41.

Format

A data frame with the melting curves of three different capture and detection probe pairs for HRPT1 and MLC-2v. First column contains the temperature (in degree Celsius, 0.5 degree Celsius per step) followed by melting curves of HRPT1 on 12 microbead populations and melting curves of MLC-2v on 12 microbead populations.

T a numeric vector, Temperature in degree Celsius.

MLC2v a numeric vector, MLC-2v with quencher-labeled detection probe

SERCA2 a numeric vector, SERCA2 without quencher-labeled detection probe

VIM.w.Mutation a numeric vector, mutated VIM with quencher-labeled detection probe

VIM.wo.Mutation a numeric vector, native VIM with quencher-labeled detection probe

Details

The melting curve was conducted with short oligonucleotide probes (direct hybridization) and longer probes (dual-hybridization probes) on the surface of microbeads (sequences and materials according to Roediger et al. (2012)) using the VideoScan platform by Roediger et al. (2012). The dyes and quencher used were Atto 647N and BHQ2.

Source

Data were measured with the VideoScan platform:

A Highly Versatile Microscope Imaging Technology Platform for the Multiplex Real-Time Detection of Biomolecules and Autoimmune Antibodies. S. Roediger, P. Schierack, A. Boehm, J. Nitschke, I. Berger, U. Froemmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W. Lehmann and C. Schroeder. *Advances in Biochemical Bioengineering/Biotechnology*. 133:35–74, 2013. http://www.ncbi.nlm.nih.gov/pubmed/22437246

Surface Melting Curve Analysis with R. S. Roediger, A. Boehm and I. Schimke. *The R Journal*. 5(2):37–52, 2013. http://journal.r-project.org/

Nucleic acid detection based on the use of microbeads: a review. S. Roediger, C. Liebsch, C. Schmidt, W. Lehmann, U. Resch-Genger, U. Schedler, P. Schierack. *Microchim Acta* 2014:1–18. DOI: 10.1007/s00604-014-1243-4

See Also

MFIerror, mcaSmoother, diffQ, diffQ2, DMP, MultiMelt

Examples

```
data(DualHyb)
```

---

mcaPeaks   *Function to estimate the approximate local minima and maxima of*
                           melting curve data.

Description

The `mcaPeaks()` is used to estimate the approximate local minima and maxima of melting curve
data. This can be useful to define a temperature range for melting curve analysis, quality control of
the melting curve or to define a threshold of peak heights. Melting curves may consist of multiple
significant and insignificant melting peaks. `mcaPeaks()` uses estimated the temperatures and fluo-
rescence values of the local minima and maxima. The original data remain unchanged and only the
approximate local minima and maxima are returned. `mcaPeaks()` takes modified code proposed
earlier by Brian Ripley (https://stat.ethz.ch/pipermail/r-help/2002-May/021934.html).

Usage

```
mcaPeaks(x, y, span = 3)
```

Arguments

x	x is the column of a data frame for the temperature.
y	y is the column of a data frame for the fluorescence values.
span	span is used to adjust the window span.

Value

p.min	returns a `data.frame` with the temperatures ("T (minima)") and the fluorescence ("F (minima)") for the local minima of the processed data.
p.max	returns a `data.frame` with the temperatures ("T (minima)") and the fluorescence ("F (minima)") for the local maxima of the processed data.

Author(s)

Stefan Roediger

See Also

[mcaSmoother](), [smooth.spline]()

Examples

```
# First Example
data(DMP)
# Smooth and Min-Max normalize melting curve data with mcaSmoother()
tmp <- mcaSmoother(DMP[, 1], DMP[,6], minmax = TRUE, n = 2)

# Extract the first derivative melting curve data
```

```
tmp.d <- diffQ(tmp, verbose = TRUE)$xy

# Determine the approximate local minima and maxima of a curve
peak.val <-mcaPeaks(tmp.d[, 1], tmp.d[, 2])
peak.val

# Plot the first derivative melting curve
# Add a horizontal line and points for the approximate local minima
# and maxima to the plot
plot(tmp.d, type = "l",
     main = "Show the approximate local minima and maxima of a curve")
  abline(h = 0)
  points(peak.val$p.min, col = 1, pch = 19)
  points(peak.val$p.max, col = 2, pch = 19)
  legend(25, 0.08, c("Minima", "Maxima"), col = c(1,2), pch = c(19,19))

# Second example
# Signifcant peaks can be distinguished by peak hight
plot(tmp.d, type = "l",
     main = "Show the approximate local minima and maxima of a curve")
  abline(h = 0)
  points(peak.val$p.min, col = 1, pch = 19)
  points(peak.val$p.max, col = 2, pch = 19)
  legend(25, 0.08, c("Minima", "Maxima"), col = c(1,2), pch = c(19,19))

# Test which local maxima peak is above the median + 3 * Median Absolute
# Add a threshold (th) line to the plot
th.max <- median(peak.val$p.max[, 2]) + 3 * mad(peak.val$p.max[, 2])
abline(h = th.max, col = 3)

# add the approximate temperatures of the local minima and
# maxima to the plot
T.val <- c(peak.val$p.min[, 1], peak.val$p.max[, 1])
text(T.val, rep(-0.05, length(T.val)), round(T.val,0))

# Use a temperature range from the plot to calculate the Tm of
# the maximum Trange is used between 37 and 74 degree Celsius

tmp <- mcaSmoother(DMP[, 1], DMP[, 6], minmax = TRUE, Trange = c(37,74),
    n = 2)
# Tm 48.23, fluoTm 0.137
diffQ(tmp, fct = max, plot = TRUE)
```

---

mcaSmoother   *Function to pre-process melting curve data.*

Description

The function mcaSmoother() is used for data pre-processing. Measurements from experimental systems may occasionally include missing values (NA). mcaSmoother() uses approx() to fill up

NAs under the assumption that all measurements were equidistant. The original data remain un-
changed and only the NAs are substituted. Following it calls smooth.spline() to smooth the
curve. Different strengths can be set using the option df.fact (f default~0.95). Internally it takes
the degree of freedom value from the spline and multiplies it with a factor between 0.6 and 1.1.
Values lower than 1 result in stronger smoothed curves. The outcome of the differentiation depends
on the temperature resolution of the melting curve. It is recommended to use a temperature res-
olution of at least 0.5 degree Celsius. In addition equal distances (e.g., 50 -> 50.5 -> 60 degree
Celsius) rather than unequal distances (e.g., 50 -> 50.4 -> 60.1 degree Celsius) for the temperature
steps are recommended. The parameter n can be used to increase the temperature resolution of the
melting curve data. mcaSmoother uses the spline function for this purpose. A temperature range for
a simple linear background correction. The linear trend is estimated by a robust linear regression
using lmrob(). In case criteria for a robust linear regression are violated lm() is automatically
used. The parameter n can be combined with the parameter Trange to make transform all melting
curves of question to have the #same range and similar resolution. Optionally a Min-Max normal-
ization between 0 and 1 can be used by setting the option minmax to TRUE. This is useful in many
situations. For example, if the fluorescence values between samples vary considerably (e.g., due
to high background, different reporter dyes,...), particularly in solution or for better comparison of
results.

Usage

```
mcaSmoother(x, y, bgadj = FALSE, bg = NULL, Trange = NULL,
  minmax = FALSE, df.fact = 0.95, n = NULL)
```

Arguments

x	is the column of a data frame for the temperature.
y	is the column of a data frame for the fluorescence values.
bgadj	is used to adjust the background signal. This causes mcaSmoother to use the data subset defined by bg for the linear regression and background correction.
bg	is used to define the range for the background reduction (e.g., bg = c(50,55), between 50 and 55 degree Celsius)).
Trange	is used to define the temperature range (e.g., Trange = c(50,95), between 50 and 95 degree Celsius) for melting curve analysis.
minmax	is used to scale the fluorescence a Min-Max normalization between 0 and 1 can be used by setting the option minmax to TRUE.
df.fact	is a factor to smooth the curve. Different strengths can be set using the option df.fact (f default ~ 0.95). Internally it takes the degree of freedom value from the spline and multiplies it with a factor between 0.6 and 1.1. Values lower than 1 result in stronger smoothed curves.
n	is number of interpolations to take place. This parameter uses the spline function and increases the temperature resolution of the melting curve data.

Value

xy	returns a data.frame with the temperature ("x") in the first and the pre-processed fluorescence values ("y.sp") in the second column.

Author(s)

Stefan Roediger

References

A Highly Versatile Microscope Imaging Technology Platform for the Multiplex Real-Time De-
tection of Biomolecules and Autoimmune Antibodies. S. Roediger, P. Schierack, A. Boehm, J.
Nitschke, I. Berger, U. Froemmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W.
Lehmann and C. Schroeder. *Advances in Biochemical Bioengineering/Biotechnology*. 133:33–74,
2013. <http://www.ncbi.nlm.nih.gov/pubmed/22437246>

Nucleic acid detection based on the use of microbeads: a review. S. Roediger, C. Liebsch, C.
Schmidt, W. Lehmann, U. Resch-Genger, U. Schedler, P. Schierack. *Microchim Acta* 2014:1–18.
DOI: 10.1007/s00604-014-1243-4

Roediger S, Boehm A, Schimke I. Surface Melting Curve Analysis with R. *The R Journal* 2013;5:37–
53.

See Also

`MFIerror`, `lmrob`, `smooth.spline`, `spline`, `lm`, `approx`

Examples

```
# First Example
# Use mcaSmoother with different n to increase the temperature
# resolution of the melting curve artificially. Compare the
# influence of the n on the Tm and fluoTm values
data(MultiMelt)

Tm <- vector()
fluo <- vector()
for (i in seq(1,3.5,0.5)) {
  res.smooth <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 14], n = i)
  res <- diffQ(res.smooth)
  Tm <- c(Tm, res$Tm)
  fluo <- c(fluo, res$fluoTm)
}
plot(fluo, Tm, ylim = c(76,76.2))
abline(h = mean(Tm))
text(fluo, seq(76.1,76.05,-0.02),
     paste("n:", seq(3.5,1,-0.5), sep = " "), col = 2)
abline(h = c(mean(Tm) + sd(Tm), mean(Tm) - sd(Tm)), col = 2)

legend(-0.22, 76.2, c("mean Tm", "mean Tm +/- SD Tm"),
col = c(1,2), lwd = 2)

# Second Example
# Use mcaSmoother with different strengths of smoothing
# (f, 0.6 = strongest, 1 = weakest).
data(DMP)
plot(DMP[, 1], DMP[,6],
```

```
      xlim = c(20,95), xlab = "Temperature",
      ylab = "refMFI", pch = 19, col = 8)
f <- c(0.6, 0.8, 1.0)
for (i in c(1:3)) {
  lines(mcaSmoother(DMP[, 1],
        DMP[,6], df.fact = f[i]),
        col = i, lwd = 2)
}
legend(20, 1.5, paste("f", f, sep = ": "),
       cex = 1.2, col =  1:3, bty = "n",
       lty = 1, lwd = 4)

# Third Example
# Plot the smoothed and trimmed melting curve
data(MultiMelt)
tmp <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 14])
tmp.trimmed <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 14],
    Trange = c(49,85))
plot(tmp, pch = 19, xlab = "Temperature", ylab = "refMFI",
  main = "MLC-2v, mcaSmoother using Trange")
  points(tmp.trimmed, col = 2, type = "b", pch = 19)
  legend(50, 1, c("smoothed values",
  "trimmed smoothed values"),
   pch = c(19,19), col = c(1,2))

# Fourth Example
# Use mcaSmoother with different n to increase the temperature
# resolution of the melting curve. Caution, this operation may
# affect your data negatively if the resolution is set to high.
# Higher resolutions will just give the impression of better
# data quality. res.st uses the default resolution (no
# alteration)
# res.high uses the double resolution.
data(MultiMelt)
res.st <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 14])
res.high <- mcaSmoother(MultiMelt[, 1], MultiMelt[, 14], n = 2)

par(fig = c(0,1,0.5,1))
plot(res.st, xlab = "Temperature", ylab = "F",
main = "Effect of n parameter on the temperature
resolution")
points(res.high, col = 2, pch = 2)
  legend(50, 1, c(paste("default resolution.", nrow(res.st),
  "Temperature steps", sep = " "),
  paste("double resolution.", nrow(res.high),
  "Temperature steps", sep = " ")),
  pch = c(1,2), col = c(1,2))
par(fig = c(0,0.5,0,0.5), new  =  TRUE)
diffQ(res.st, plot = TRUE)
  text(65, 0.025, paste("default resolution.", nrow(res.st),
"Temperature steps", sep = " "))
par(fig = c(0.5,1,0,0.5), new = TRUE)
diffQ(res.high, plot = TRUE)
```

```
  text(65, 0.025, paste("double resolution.", nrow(res.high),
"Temperature steps", sep = " "))

# Fifth example
# Different experiments may have different temperature
# resolutions and temperature ranges. The example uses a
# simulated melting curve with a temperature resolution of
# 0.5 and 1 degree Celsius and a temperature range of
# 35 to 95 degree Celsius.
#
# Coefficients of a 3 parameter sigmoid model.  Note:
# The off-set, temperature range and temperature resolution
# differ between both simulations. However, the melting
# temperatures should be very
# similar finally.
b <- -0.5; e <- 77

# Simulate first melting curve with a temperature
# between 35 - 95 degree Celsius and 1 degree Celsius
# per step temperature resolution.

t1 <- seq(35, 95, 1)
f1 <- 0.3 + 4 / (1 + exp(b * (t1 - e)))

# Simulate second melting curve with a temperature
# between 41.5 - 92.1 degree Celsius and 0.5 degree Celsius
# per step temperature resolution.
t2 <- seq(41.5, 92.1, 0.5)
f2 <- 0.2 + 2 / (1 + exp(b * (t2 - e)))

# Plot both simulated melting curves
plot(t1, f1, pch = 15, ylab = "MFI",
     main = "Simulated Melting Curves",
     xlab = "Temperature", col = 1)
points(t2, f2, pch = 19, col = 2)
legend(50, 1,
       c("35 - 95 degree Celsius, 1 degree Celsius per step",
       "41.5 - 92.1 degree Celsius, 0.5 degree Celsius per step",
       sep = " "), pch = c(15,19), col = c(1,2))

# Use mcaSmoother with n = 2 to increase the temperature
# resolution of the first simulated melting curve. The minmax
# parameter is used to make the peak heights compareable. The
# temperature range was limited between 45 to 90 degree Celsius for
# both simulations

t1f1 <- mcaSmoother(t1, f1, Trange= c(45, 90), minmax = TRUE, n = 2)
t2f2 <- mcaSmoother(t2, f2, Trange= c(45, 90), minmax = TRUE, n = 1)

# Perform a MCA on both altered simulations. As expected, the melting
# temperature are almost identical.
par(mfrow = c(2,1))
# Tm 77.00263, fluoTm -0.1245848
```

```
diffQ(t1f1, plot = TRUE)
text(60, -0.08,
     "Raw data: 35 - 95 degree Celsius,\n 1 degree Celsius per step")


# Tm 77.00069, fluoTm -0.1245394
diffQ(t2f2, plot = TRUE)
text(60, -0.08, "Raw data: 41.5 - 92.1 degree Celsius,
      \n 0.5 degree Celsius per step")
par(mfrow = c(1,1))
```

MFIerror	*Multiple comparison of the temperature dependent variance of the refMFI*

Description

MFIerror is used for a fast multiple comparison of the temperature dependent variance of the refMFI. MFIerror returns an object of the class data.frame with columns "Temperature", "Location" (Mean, Median), "Deviation" (Standard Deviation, Median Absolute Deviation) and "Coefficient of Variation".

Usage

```
MFIerror(x, y, CV = FALSE, RSD = FALSE, rob = FALSE, errplot = TRUE,
  type = "p", pch = 19, length = 0.05, col = "black")
```

Arguments

x	is the column of a data frame for the temperature.
y	are multiple columns of fluorescence values from a data.frame (e.g., [, c(1:n)]).
CV	If CV is true the coefficient of variation (RSD, CV) is plotted. If set to FLASE the deviation as Standard Deviation or Median Absolute Deviation is plotted.
RSD	Setting the option RSD=TRUE shows the relative standard deviation (RSD) in percent.
rob	Using the option rob as TRUE the median and the median absolute deviation (MAD) is plotted instead of the mean and standard deviation.
errplot	sets MFIerror() to plot the results (default). In the default setting (CV=FALSE) the mean with the standard deviations is plotted.
type	is a graphical parameter setting the plot use lines, points or both (see plot).
pch	is a graphical parameter used to define the symbol used in the plot.
length	length is a graphical parameter used to define the length of the error bar used in the plot.
col	col is a graphical parameter used to define the color of the error bar used in the plot.

Value

res   returns a `data.frame` containing the "Temperature", "Location" (mean, me-
               dian), "Deviation" (standard deviation, median absolute deviation), "Coefficient
               of Variance" (CV, RSD) sequential in the columns.

Author(s)

Stefan Roediger

See Also

[mcaSmoother](#)

Examples

```
# First Example
# Temperature dependent variance of the refMFI using standard measures
# (Mean, Standard Deviation (SD)).
# Use Standard Deviation (SD) in the plot

data(MultiMelt)
MFIerror(MultiMelt[, 1], MultiMelt[, c(2L:13)])

# Second Example
# Temperature dependent relative variance of the refMFI using robust
# measures (Median, Median Absolute Deviation (MAD)). The parameter
# errplot is set to FALSE in order to prevent the plot of the
# coefficient of variation versus the temperature.

MFIerror(MultiMelt[, 1], MultiMelt[, c(2L:13)], errplot = FALSE,
  RSD = TRUE, rob = TRUE)

# Third Example
# Temperature dependent relative variance of the refMFI using
# robust measures (Median, Median Absolute Deviation (MAD)).
MFIerror(MultiMelt[, 1], MultiMelt[, c(2L:13)], RSD = TRUE,
  rob = TRUE)
```

MultiMelt   *Surface melting curve data from direct hybridization experiment of
                         short oligonucleotides.*

Description

A melting curve experiment with twelve microbead populations and the short oligonucleotide cap-
ture probe and detection probe for human HRPT1 (hyperparathyroidism 1) and MLC-2v (myosin
regulatory light chain 2, ventricular/cardiac muscle isoform).

Format

A data frame with the melting curves of two different capture and detection probe pairs for HRPT1 and MLC-2v. First column contains the temperature (in degree Celsius, 1 degree Celsius per step) followed by melting curves of HRPT1 on twelve microbead populations and melting curves of MLC-2v on twelve microbead populations.

T a numeric vector for the temperature in degree Celsius

HPRT1.1 a numeric vector, as HPRT1.1 of detection/capture probe HPRT1/HPRT1-cap on microbead population 1

HPRT1.2 a numeric vector, as HPRT1.2 on microbead population 2

HPRT1.3 a numeric vector, as HPRT1.3 on microbead population 3

HPRT1.4 a numeric vector, as HPRT1.4 on microbead population 4

HPRT1.5 a numeric vector, as HPRT1.5 on microbead population 5

HPRT1.6 a numeric vector, as HPRT1.6 on microbead population 6

HPRT1.7 a numeric vector, as HPRT1.7 on microbead population 7

HPRT1.8 a numeric vector, as HPRT1.8 on microbead population 8

HPRT1.9 a numeric vector, as HPRT1.9 on microbead population 9

HPRT1.10 a numeric vector, as HPRT1.10 on microbead population 10

HPRT1.11 a numeric vector, as HPRT1.11 on microbead population 11

HPRT1.12 a numeric vector, as HPRT1.12 on microbead population 12

MLC2v1 a numeric vector, as MLC2v1 of detection/capture probe MLC-2v/MLC-2v-cap on microbead population 1

MLC2v2 a numeric vector, as MLC2v2 on microbead population 2

MLC2v3 a numeric vector, as MLC2v3 on microbead population 3

MLC2v4 a numeric vector, as MLC2v4 on microbead population 4

MLC2v5 a numeric vector, as MLC2v5 on microbead population 5

MLC2v6 a numeric vector, as MLC2v6 on microbead population 6

MLC2v7 a numeric vector, as MLC2v7 on microbead population 7

MLC2v8 a numeric vector, as MLC2v8 on microbead population 8

MLC2v9 a numeric vector, as MLC2v9 on microbead population 9

MLC2v10 a numeric vector, as MLC2v10 on microbead population 10

MLC2v11 a numeric vector, as MLC2v11 on microbead population 11

MLC2v12 a numeric vector, as MLC2v12 on microbead population 12

Details

The melting curve was conducted with short oligonucleotide probes on the surface of microbeads using the VideoScan platform according to Roediger et al. (2012). The dyes and quencher used were Atto 647N and BHQ2.

Source

Data were measured with the VideoScan platform:

A Highly Versatile Microscope Imaging Technology Platform for the Multiplex Real-Time Detection of Biomolecules and Autoimmune Antibodies. S. Roediger, P. Schierack, A. Boehm, J. Nitschke, I. Berger, U. Froemmel, C. Schmidt, M. Ruhland, I. Schimke, D. Roggenbuck, W. Lehmann and C. Schroeder. *Advances in Biochemical Bioengineering/Biotechnology*. 133:35–74, 2013. <http://www.ncbi.nlm.nih.gov/pubmed/22437246>

Surface Melting Curve Analysis with R. S. Roediger, A. Boehm and I. Schimke. *The R Journal*. 5(2):37–52, 2013, 2013. <http://journal.r-project.org/>

See Also

MFIerror, mcaSmoother, diffQ, diffQ2, DMP, DualHyb

Examples

```
data(MultiMelt)
```

Index