Package 'VFP'

January 27, 2025

Version 1.4.3

Date 2025-01-27

Title Variance Function Program

Author Andre Schuetzenmeister [cre, aut],
Florian Dufey [aut],
Andrea Geistanger [ctb]

Maintainer Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Depends R (>= 3.1.0), gnm

Imports VCA, MASS, utils, stats, graphics, grDevices, methods

Suggests RUnit, knitr, rmarkdown, prettydoc

Description

Variance function estimation for models proposed by W. Sadler in his variance function program ('VFP', www.aacb.asn.au/AACB/Resources/Variance-Function-Program). Here, the idea is to fit multiple variance functions to a data set and consequently assess which function reflects the relationship 'Var ~ Mean' best. For 'in-vitro diagnostic' ('IVD') assays modeling this relationship is of great importance when individual test-results are used for defining follow-up treatment of patients.

VignetteBuilder knitr, rmarkdown

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2025-01-27 08:50:05 UTC

Contents

VFP-package *(V)ariance (F)unction (P)rogram*

Description

The intended use of this package is to implement variance functions proposed in Sadler's VFP stand-alone software (see reference below), from which the name was borrowed as well. Main function of this package is `fit.vfp` for fitting non-linear variance-function models. Usually, these models are fitted to analysis-results of precision performance data e.g. frequently generated for in-vitro diagnostics (IVD). R-package VFP is designed to work best on objects of class 'VCA' as generated by R-package `VCA` but it is not restricted to these. There are several functions operating on S3-objects of class 'VFP', e.g. `plot.VFP`, `print.VFP`, `summary.VFP`, and `predict.VFP`. Function `predictMean` is of special interest when a functional relationship is used to derive limit of quantitation (LoQ) or functional sensitivity, as the concentration at which the IVD-imprecision expressed as coefficient of variation (CV) undercuts a specific threshold.

Details

Package:	VFP
Type:	Package
Version:	1.4.3
Date:	2025-01-27
License:	GPL (>=3)
LazyLoad:	yes

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>, Florian Dufey <florian.dufey@roche.com>, Andrea Geistanger <andrea.geistanger@roche.com>

References

CLSI EP05-A3: Evaluation of Precision of Quantitative Measurement Procedures; Approved Guideline - Third Edition. (2014)

Sadler WA, Smith MH. Use and Abuse of Imprecision Profiles: Some Pitfalls Illustarted by Computing and Plotting Confidence Intervals. Clin Chem 1990; 36/7:1346-1350

Sadler WA, Smith MH. A reliable method of estimating the variance function in immunoassays. Comput Stat Data Anal 1986; 3:227-239

Sadler WA, Smith MH. Estimation of imprecision in immunoassays quality assessment programmes. Ann Clin Biochem 1987; 24:98-102

www.aacb.asn.au/AACB/Resources/Variance-Function-Program

`.onLoad` *Keep backward compatibility by assigning new to old function names when loading the package. Old name did not comply to new CRAN package check rules.*

Description

Keep backward compatibility by assigning new to old function names when loading the package. Old name did not comply to new CRAN package check rules.

Usage

```
.onLoad(libname, pkgname)
```

Arguments

`libname`	(character) string giving the library directory where the package defining the namespace was found
`pkgname`	(character) string giving the name of the package

`add_grid` *Add a Grid to an Existing Plot.*

Description

It is possible to use automatically determined grid lines (`x=NULL, y=NULL`) or specifying the number of cells `x=3, y=4` as done by `grid`. Additionally, x- and y-locations of grid-lines can be specified, e.g. `x=1:10, y=seq(0,10,2)`.

Usage

```
add_grid(x = NULL, y = NULL, col = "lightgray", lwd = 1L, lty = 3L)
```

Arguments

`x`	(integer, numeric) single integer specifies number of cells, numeric vector specifies vertical grid-lines
`y`	(integer, numeric) single integer specifies number of cells, numeric vector specifies horizontal grid-lines
`col`	(character) color of grid-lines
`lwd`	(integer) line width of grid-lines
`lty`	(integer) line type of grid-lines

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

as_rgb *Convert Color-Name or RGB-Code to Possibly Semi-Transparent RGB-code.*

Description

Function takes the name of a color and converts it into the rgb space. Parameter "alpha" allows to specify the transparency within [0,1], 0 meaning completey transparent and 1 meaning completey opaque. If an RGB-code is provided and alpha != 1, the RGB-code of the transparency adapted color will be returned.

Usage

```
as_rgb(col = "black", alpha = 1)
```

Arguments

col	(character) name of the color to be converted/transformed into RGB-space (code). Only those colors can be used which are part of the set returned by function colors(). Defaults to "black".
alpha	(numeric) value specifying the transparency to be used, 0 = completely transparent, 1 = opaque.

Value

RGB-code

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
# convert character string representing a color to RGB-code
# using alpha-channel of .25 (75\% transparent)
as_rgb("red", alpha=.25)

# same thing now using the RGB-code of red (alpha=1, i.e. as_rgb("red"))
as_rgb("#FF0000FF", alpha=.25)
```

B2mIntra_98 *Example Data B2mIntra_98.VFP (Beta-2-microglobulin RIA) from the Variance Function Program 12.0 from Sadler*

Description

Results in this file are all serum/CSF clinical specimen duplicates during 1998 from a low throughput Beta-2-microglobulin RIA.

Usage

```
data(B2mIntra_98)
```

Format

data.frame with 554 observations and 2 variables.

References

www.aacb.asn.au/AACB/Resources/Variance-Function-Program

bt_coef *Back-Transformation of Estimated Coefficients.*

Description

This function performs back-transformation from re-parameterized forms in the 'VFP'-package into the original form.

Usage

```
bt_coef(object, K = NULL, signJ = NULL, model = NULL, ...)
```

Arguments

object	(object) of class 'gnm' representing a fitted model in re-parameterized form
K	(numeric) constant value 'K'
signJ	(integer) either 1 or -1
model	(integer) specifying which model shall be back-transformed
...	additional parameters

Details

In the 'VFP' package models are re-parameterized to have better control over the constraint solution-space, i.e. only models may be fitted generating non-negative fitted values. This function is intended to be for internal use only.

Value

(numeric) vector of coefficients in original parameterized form

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com> Florian Dufey <florian.dufey@roche.com>

`coef.VFP` *Extract Model-Coefficients from VFP-Objects.*

Description

Extract Model-Coefficients from VFP-Objects.

Usage

```
## S3 method for class 'VFP'
coef(object, model.no = NULL, ...)
```

Arguments

`object`	(object) of class "VFP"
`model.no`	(integer) specifying one of models 1:10, must be one of the fitted models
`...`	additional parameters passed forward

Value

(numeric) model coefficients

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com> Florian Dufey <florian.dufey@roche.com>

Examples

```
library(VCA)
data(VCAdata1)
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
mat <- get_mat(lst) # automatically selects "total"
res <- fit_vfp(model.no=1:9, Data=mat)
coef(res)
```

condition_handler *Condition-Handling Without Losing Information.*

Description

Function is intented to wrap expressions provided and catching all potentially useful information generated by the wrapped expression, i.e. errors, warnings, and messages.

Usage

```
condition_handler(expr, file = NULL)
```

Arguments

expr	(expression) for which exception handling should be provided
file	(character) string specifying a file to which all captured output shall be written

Value

(list) with element "result", "status" (0 = no warnings, no errors), 1 = warnings were caught, 2 = errors were caught no result generated, "warnings", "errors", "messages"

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
condition_handler(warning("This is a warning!"))
f <- function(expr){warning("This a warning!"); eval(expr)}
condition_handler(f(1/2))
condition_handler(stop("This is an error!"))
condition_handler(1/"a")
```

derive_cx *Determine C5 and C95 or any Concentration Cx.*

Description

This function makes use of a precision profile. The concentration is sought at which 100 * 'Cx'% of the measurements lie above 'cutoff' theoretically as each X-value corresponds to a normal distribution with mean=X and SD as read off the precision profile. In case of e.g. "C5" exactly 5% will be above cutoff, whereas for "C95" 95% will be larger than cutoff. This follows the CLSI EP12 guideline whenever an internal continuous result (ICR) is available and measurement imprecision can be assumed to be normally distributed. The CLSI EP12 recommends to base derivation of C5 and C95 on the results of intermediate precision analyses using multiple samples. This includes between-day and between-run as additional variance components besides repeatability.

Usage

```
derive_cx(
  vfp,
  model.no = NULL,
  start = NULL,
  cutoff = NULL,
  Cx = 0.05,
  tol = 1e-06,
  plot = FALSE
)
```

Arguments

vfp	(VFP) object representing a precision profile to be used
model.no	(integer) specifying which model to use, if NULL the "best" model will be used automatically
start	(numeric) start concentration, e.g. for C5 smaller than r, for C95 larger than R
cutoff	(numeric) the cutoff of to be used
Cx	(numeric) the probability, e.g. for C5 use 0.05 and for C95 use 0.95
tol	(numeric) tolerance value determining when the iterative bisections search can terminate, i.e. when the difference becomes smalle enough
plot	(logical) TRUE = plot the result

Value

(numeric) Mean and SD of concentration Cx

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
## Not run:
# perform variance component analysis
library(VCA)
data(VCAdata1)
# perform VCA-anaylsis
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
# transform list of VCA-objects into required matrix
mat <- get_mat(lst) # automatically selects "total"
mat
# fit all models batch-wise
res <- fit_vfp(model.no=1:9, Data=mat)
# now search for the C5 concentration
derive_cx(res, start=15, cutoff=20, Cx=0.05, plot=TRUE)
derive_cx(res, start=25, cutoff=20, Cx=0.95, plot=TRUE)
derive_cx(res, start=25, cutoff=20, Cx=0.25, plot=TRUE)
```

```
derive_cx(res, start=25, cutoff=20, Cx=0.75, plot=TRUE)

#
p <- c(seq(.01, .12, .01), seq(.15, .85, .05), seq(.88, .99, .01))
system.time(x <- derive_cx(res, Cx=p, cutoff=20))

## End(Not run)
```

`fit_ep17` *Fit CLSI EP17 Model Using log-transformed X and Y.*

Description

This function fits the model proposed in CLSI EP17 by log-transforming CV (Y) as well as mean-values (X) und performing a linear regression of these. More specifically CV = A * Conc^B, where Conc = mean concentration of a sample and CV is on the percent-scale, is fitted by ordinary least squares (OLS) estimation of log(CV) = A + B * log(Conc). Fitted values are subsequently back-transformed using formula cv = exp(a) * C^b, where cv, a and b represent estimates of CV, A and B. Therefore, this model does not fall within the same class as models 1 to 9, although the predictor function is identical to that of model 9. This also has the consequence that regression statistics, like AIC or deviance, are not directly comparable to those of models 1 to 9.

Usage

```
fit_ep17(x, y, DF, typeY = c("vc", "sd", "cv"), k = 2, ...)
```

Arguments

`x`	(numeric) mean concentrations of samples
`y`	(numeric) variability at 'x' on VC-, SD-, or CV-scale
`DF`	(numeric) vector of degrees of freedom linked to variabilities 'y' used in derivation of deviance and AIC
`typeY`	(character) specifying the scale of 'y'-values
`k`	(numeric) numeric specifying the 'weight' of the equivalent degrees of freedom (edf) part in the AIC formula.
`...`	additional arguments

Details

The AIC is computed following the implementation of `extractAIC.lm` in the 'stats' package with the adaption of using 'n = sum(df)' instead of 'n' being the number of residuals. The 'df' come from a precision analysis, thus, there are far more observations used to fit this model than indicated by the number of residuals.

Value

(list) with items "x" and "y" as provided, and "x.out" and "y.out" representing X- and Y-coordiantes of fitted values for plotting

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
# data from appendix D of CLSI EP17-A2 (pg. 54)
EP17.dat <- data.frame(
 Lot=c(rep("Lot1", 9), rep("Lot2", 9)),
Mean=c( 0.04, 0.053, 0.08, 0.111, 0.137, 0.164, 0.19, 0.214, 0.245,
0.041, 0.047, 0.077, 0.106, 0.136, 0.159, 0.182, 0.205, 0.234),
CV=c(40.2, 29.6, 19.5, 15.1, 10.0, 7.4, 6.0, 7.5, 5.4,
 44.1, 28.8, 15.1, 17.8, 11.4, 9.2, 8.4, 7.8, 6.2),
 SD=c(0.016, 0.016, 0.016, 0.017, 0.014, 0.012, 0.011, 0.016, 0.013,
 0.018, 0.014, 0.012, 0.019, 0.016, 0.015, 0.015, 0.016, 0.014),
 DF=rep(1, 18)
)

EP17.dat$VC <- EP17.dat$SD^2

lot1 <- subset(EP17.dat, Lot=="Lot1")
lot2 <- subset(EP17.dat, Lot=="Lot2")

# function fit_ep17 is not exported, use package namesspace in call
fit.lot1 <- VFP:::fit_ep17(x=lot1$Mean, y=lot1$CV, typeY="cv", DF=lot1$DF)
```

`fit_vfp` *Estimate (Im)Precision-Profiles Modeling the Relationship 'Var ~ Mean'.*

Description

This function fits one out of ten or any subset of ten non-linear functions at once. This is done on precision data consisting of information about the variance, concentration at which this variance was observed and the respective degrees of freedom. Provided data must contain at least three columns with this information. There are following variance-functions covered:

constant variance $sigma^2$

constant CV $sigma^2 = beta_1 * u^2$

mixed constant, proportional variance $sigma^2 = beta_1 + beta_2 * u^2$

constrained power model, constant exponent $sigma^2 = (beta_1 + beta_2 * u)^K$

alternative constrained power model $sigma^2 = beta_1 + beta_2 * u^K$

alternative unconstrained power model for VF's with a minimum $sigma^2 = beta_1 + beta_2 * u + beta_3 * u^J$

alternative unconstrained power model $sigma^2 = beta_1 + beta_2 * u^J$

unconstrained power model (default model of Sadler) $sigma^2 = (beta_1 + beta_2 * u)^J$

CLSI EP17 similar model $sigma^2 = beta_1 * u^J$

Exact CLSI EP17 model (fitted by linear regression on logarithmic scale) $cv = beta_1 * u^J$

Fitting all ten models is somehow redundant if constant 'C' is chosen to be equal to 2, since models 3 and 5 are equivalent and these are constrained versions of model 7 where the exponent is also estimated. The latter also applies to model 4 which is a constrained version of model 8. Note that model 10 fits the same predictor function as model 9 using simple linear regression on a logarithmic scale. Therefore, regression statistics like AIC, deviance etc. is not directly comparable to that of models 1 to 9. Despite these possible redundancies, as computation time is not critical here for typical precision-profiles (of in-vitro diagnostics precision experiments) we chose to offer batch-processing as well. During computation, all models are internally reparameterized so as to guarantee that the variance function is positive in the range 'u' from 0 to 'u_max'. In models 7 and 8, 'J' is restricted to 0.1<J<10 to avoid the appearance of sharp hooks. Occasionally, these restrictions may lead to a failure of convergence. This is then a sign that the model parameters are on the boundary and that the model fits the data very badly. This should not be taken as reason for concern. It occurs frequently for model 6 when the variance function has no minimum, which is normally the case.

Usage

```
fit_vfp(
  Data,
  model.no = 1:9,
  K = 2,
  startvals = NULL,
  quiet = TRUE,
  col.mean = "Mean",
  col.var = "VC",
  col.df = "DF",
  col.sd = NULL,
  col.cv = NULL,
  minVC = NA,
  ...
)
```

Arguments

Data	(data.frame, matrix) containing mean-values, estimated variances and degrees of freedom for each sample
model.no	(integer) in 1:10, may be any combination of these integer-values specifying models 1 to 10 which are consequently fitted to the data; defaults to 1:9
K	(numeric) value defining the constant used in models 4 and 5; defaults to 2
startvals	(numeric) vector of start values for the optimization algorithm, only used if a single model is specified by the user, otherwise, start values will be sought automatically
quiet	(logical) TRUE = all output messages (warnings, errors) and regular console output is suppressed and can be found in elements "stderr" and "stdout" of the returned object

col.mean	(character) string specifying a column/variable in 'Data' containing mean-values; defaults to "Mean"
col.var	(character) string specifying a column/variable in 'Data' containing the variances; Defaults to "VC"
col.df	(character) string specifying a column/variable in 'Data' containing the degrees of freedom; defaults to "DF"
col.sd	(character) string (optional) specifying a column/ variable in 'Data' containing the SD-values, used for model 10 to derive more precise CV-values compared to derivation from 'col.var' in case 'col.cv' is not specified directly. Note that column "col.var" must nevertheless be set and existing
col.cv	(character) string (optional) specifying a column/ variable in 'Data' containing the CV-values, if missing, first 'col.sd' is checked, if missing 'col.var' used to derive per-sample CV-values. Note that column "col.var" must nevertheless be set and existing
minVC	(numeric) value assigned to variances being equal to zero, defaults to NA, which results in removing these observations; could also be set to the smallest possible positive double (.Machine$double.eps)
...	additional parameters passed forward, e.g. 'vc' of function get_mat for selecting a specific variance component in case of 'Data' being a list of 'VCA'-objects (see examples)

Details

Functions predict.VFP and predict_mean are useful to make inference based on a fitted model. #' It is possible to derive concentrations at which a predefined variability is reached, which is sometimes referred to as #' "functional sensitivity" and/or "limit of quantitation" (LoQ). Funtion predict_mean returns the fitted value #' at which a user-defined variance ("vc"), SD or CV is reached with its corresponding 100(1-alpha)% CI derived from the CI of the fitted model. The plotting method for objects of class 'VFP' can automatically add this information to a plot using arguments 'Prediction' and 'Pred.CI' (see plot.VFP for details. Function predict.VFP makes #' predictions for specified mean-values based on fitted models.

Note, that in cases where a valid solution was found in the re- parameterized space but the final fit with 'gnm' in the original parameter-space did not converge no variance-covariance matrix can be estimated. Therefore, no confidence-intervals will be available downstream.

Value

(object) of class 'VFP' with elements:

Models	objects of class 'gnm' representing the fitted model
RSS	residual sum of squares for each fitted model
AIC	the Akaike information criterion for each fitted model
Deviance	the deviance for each fitted model
Formulas	formula(s) as expression for each fitted model
Mu.Func	function(s) to transform mean value to re-parameterized scale

Mu	list of transformed mean values
Data	the original data set provided to fit model(s)
K	constant as specified by the user
startvals	start values as specified by the user
errors	messages of all errors caught
output	if 'quiet = TRUE' all output that was redirected to a file
warning	messages of all warnings caught
notes	all other notes caught during execution

Author(s)

Florian Dufey <florian.dufey@roche.com>, Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

See Also

plot.VFP, predict.VFP, predict_mean

Examples

```
# load VCA-package and data
library(VCA)
data(VCAdata1)
# perform VCA-anaylsis
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by = "sample")
# transform list of VCA-objects into required matrix
mat <- get_mat(lst)    # automatically selects "total"
mat

# fit all 9 models batch-wise
res <- fit_vfp(model.no = 1:9, Data = mat)

# if 'mat' is not required for later usage, following works
# equally well
res2 <- fit_vfp(lst, 1:9)

# plot best-fitting model
plot(res)
plot(res, type = "cv")
plot(res, type = "cv", ci.type = "lines", ci.col = "red",
     Grid = list(col = "wheat"),
Points = list(pch = 2, lwd = 2, col = "black"))

# now derive concentation at which a specific reproducibility-
# imprecision of 10\% is reached and add this to the plot
pred <- plot(res, type = "cv", ci.type = "band",
        ci.col = as_rgb("red", .25), Grid = list(col = "orange"),
        Points = list(pch = 2, lwd = 2, col = "black"),
        Prediction = list(y = 10, col = "red"), Pred.CI = TRUE)

# (invisibly) returned object contains all relevant information
```

```
pred

# same for repeatability
mat.err <- get_mat(lst, "error")
res.err <- fit_vfp(1:9, Data = mat.err)

# without extracting 'mat.err'
res.err2 <- fit_vfp(lst, 1:9, vc = "error")

plot(res.err)

########################################################################
# another example using CA19_9 data from CLSI EP05-A3

data(CA19_9)

# fit reproducibility model to data
fits.CA19_9 <- anovaVCA(result~site/day, CA19_9, by = "sample")

# fit within-laboratory-model treating site as fixed effect
fits.ip.CA19_9 <- anovaMM(result~site/(day), CA19_9, by = "sample")

# the variability "between-site" is not part of "total"
fits.ip.CA19_9[[1]]
fits.CA19_9[[1]]

# extract repeatability
rep.CA19_9 <- get_mat(fits.CA19_9, "error")

# extract reproducibility
repro.CA19_9 <- get_mat(fits.CA19_9, "total")

# extract intermediate-precision (within-lab)
ip.CA19_9 <- get_mat(fits.ip.CA19_9, "total")

# fit model (a+bX)^C (model 8) to all three matrices

mod8.repro   <- fit_vfp(repro.CA19_9, 8)
mod8.ip    <- fit_vfp(ip.CA19_9, 8)
mod8.rep    <- fit_vfp(rep.CA19_9, 8)

# plot reproducibility precision profile first
# leave enough space in right margin for a legend
plot(mod8.repro, mar = c(5.1, 7, 4.1, 15),
    type = "cv", ci.type = "none", Model = FALSE,
    Line = list(col = "blue", lwd = 3),
    Points = list(pch = 15, col = "blue", cex = 1.5),
    xlim = c(10, 450), ylim = c(0, 10),
    Xlabel = list(text = "CA19-9, kU/L (LogScale) - 3 Patient Pools,
 3 QC Materials",
        cex = 1.5), Title = NULL,
    Ylabel = list(text = "% CV", cex = 1.5),
    Grid = NULL, Crit = NULL, log = "x")
```

```
# add intermediate precision profile
plot  (mod8.ip, type = "cv", add = TRUE, ci.type = "none",
    Points = list(pch = 16, col = "deepskyblue", cex = 1.5),
    Line = list(col = "deepskyblue", lwd = 3), log = "x")

# add repeatability precision profile
plot( mod8.rep, type = "cv", add = TRUE, ci.type = "none",
    Points = list(pch = 17, col = "darkorchid3", cex = 1.5),
    Line = list(col = "darkorchid3", lwd = 3), log = "x")

# add legend to right margin
legend_rm(x = "center", pch = 15:17, col = c("blue", "deepskyblue",
"darkorchid3"), cex = 1.5, legend = c("Reproducibility",
"Within-Lab Precision", "Repeatability"), box.lty = 0)
#'
```

`get_mat` *Transform list of VCA-object into VFP-matrix required for fitting.*

Description

Transform list of VCA-object into VFP-matrix required for fitting.

Usage

```
get_mat(obj, vc = 1)
```

Arguments

`obj`	(list) of VCA-objects
`vc`	(integer, character) either an integer specifying a variance component or the name of a variance component; can also be a vector of integers specifying a continuous sequence of variance components always including 'error' (repeatability)

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
library(VCA)
data(VCAdata1)
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
get_mat(lst)  # automatically selects 'total'
# pooled version of intermediate precision (error+run+day)
get_mat(lst, 4:6)
# only repeatability ('error')
```

```
get_mat(lst, "error")
# use remlVCA instead
lst2 <- remlVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
get_mat(lst2)
```

`Glucose` *Example Data Glucose.VFP from the Variance Function Program 12.0 from Sadler*

Description

Intra-day precision of arterial glucose measurement on the Precision PCx instrument (Abbott Laboratories, Abbott Park, Illinois, USA). The data were kindly provided by Peter Watkinson, The John Radcliffe Hospital, Oxford, UK. They provide an excellent example of a situation where a simple variance function should be imposed in preference to accepting the fit of one of the flexible functions.

Usage

```
data(Glucose)
```

Format

data.frame with 206 observations and 2 variables.

References

www.aacb.asn.au/AACB/Resources/Variance-Function-Program

`legend_rm` *Add Legend to Right Margin.*

Description

This function accepts all parameters applicable in and forwards them to function `legend`. There will be only made some modifications to the X-coordinate ensuring that the legend is plotted in the right margin of the graphic device. Make sure that you have reserved sufficient space in the right margin, e.g. 'plot.VFP(....., mar=c(4,5,4,10))'.

Usage

```
legend_rm(
  x = c("center", "bottomright", "bottom", "bottomleft", "left", "topleft", "top",
    "topright", "right"),
  y = NULL,
  offset = 0.05,
  ...
)
```

Arguments

x	(character, numeric) either one of the character strings "center","bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" or a numeric values specifying the X-coordinate in user coordinates
y	(numeric) value specifying the Y-coordiante in user coordinates, only used in case 'x' is numeric
offset	(numeric) value in [0, 0.5] specifying the offset as fraction in regard to width of the right margin
...	all parameters applicable in function legend

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
library(VCA)
data(VCAdata1)
# perform VCA-anaylsis
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
# transform list of VCA-objects into required matrix
mat <- get_mat(lst) # automatically selects "total"
mat

# fit all 9 models batch-wise
res <- fit_vfp(model.no=1:9, Data=mat)

plot(res, mar=c(5.1, 4.1, 4.1,15), Crit=NULL)

legend_rm(cex=1.25, text.font=10,
 legend=c(
    paste0("AIC:    ", signif(as.numeric(res$AIC["Model_6"]), 4)),
   paste0("Dev:    ", signif(as.numeric(res$Deviance["Model_6"]), 4)),
    paste0("RSS:    ", signif(as.numeric(res$RSS["Model_6"]),4))))
```

MultiLotReproResults *Result of a Real-World Precsion Experiment on 10 Samples*

Description

(data.frame) representing the results of an imprecision experiment. There are 10 observations and four variables which can be used directly as input for function 'fit.vfp'. Different samples are indicated by rownames.

Usage

```
data(MultiLotReproResults)
```

Format

data.frame with 10 observations and 4 variables.

`plot.VFP` *Plot VFP-Ojbects.*

Description

Function takes an object of class 'VFP' and plots a fitted variance-function either on the original variance-scale ('type="vc"') or on the CV-scale ("cv"). The corresponding 100x(1-alpha)% confidencen interval around the variance-function can be plotted either as lines ('ci.type="lines') or as per default as CI-band.

Usage

```
## S3 method for class 'VFP'
plot(
  x,
  model.no = NULL,
  type = c("vc", "sd", "cv"),
  add = FALSE,
  alpha = 0.05,
  ci.col = "gray90",
  ci.type = c("band", "lines", "none"),
  dispersion = NULL,
  browse = FALSE,
  BG = "white",
  Title = list(),
  Xlabel = list(),
  Ylabel = list(),
  Line = list(),
  Points = list(),
  Grid = list(),
  Crit = list(),
  ylim = NULL,
  xlim = NULL,
  Prediction = NULL,
  Pred.CI = NULL,
  Model = TRUE,
  CI.method = c("chisq", "t", "normal"),
  use.log = FALSE,
  Npred = 200,
  ...
)
```

Arguments

x	(VFP) object as returned by function 'fit_vfp'
model.no	(integer) specifying which model to plot, must be one of the fitted models
type	(character) either "vc" to generate a plot on the original variance-scale or "cv" to plot it on the coefficient of variation scale, i.e. CV = 100*sqrt(VC)/Mean or "log" to plot after a variance stabilizing transformation. The latter will not work if 'Prediction' is specified!.
add	(logical) TRUE = the current (im)precision profile is added to an existing plot, FALSE = a new plot will be generated
alpha	(numeric) value specifying the 100x(1-alpha)% confidencen interval to be plotted around the function
ci.col	(character) string specifying a color used for the CI-region
ci.type	(character) either "band" to plto the CI as polygon, or "lines" to plot the CI-bounds as two separate line using color 'ci.col'
dispersion	(numeric) NULL = the dispersion parameter will be estimated, numeric value = the dispersion parameter will be used as specified
browse	(logical) TRUE = if multiple models were fitted, all will be displayed one after the other in increasing order of their respective AIC, mouse-klick on the plot triggers switch to the next model
BG	(character) string specifying a background color
Title	(list) passed to function mtext controlling content and style of the main title
Xlabel	(list) passed to function mtext controlling the labeling of the X-axis
Ylabel	(list) passed to function mtext controlling the labeling of the Y-axis
Line	(list) passed to function lines controlling the visual appearance of the line representing the fitted variance-function
Points	(list) passed to function points controlling how data used to fit a variance function shall be plotted
Grid	(list) passed to function add_grid controlling the appearance of a grid, set to NULL to omit
Crit	(list) passed to function legend per default used to present the optimality criterion for choosing the best fitting model, per default this is AIC and additionally residual sum of squares (RSS) is shown for the plotted model in the upper-right corner
ylim	(numeric) vector of length two specifying plot-limits in Y-direction, if NULL these will be automatically determined
xlim	(numeric) vector of length two specifying plot-limits in X-direction, if NULL these will be automatically determined
Prediction	(list) with elements 'y' specifying values on VC-, SD- or CV-scale depending on 'type' for which predictions on the X-axis are requested or 'x' specifying mean-values on the X-axis for which predictions on the Y-axis are requested; furthermore, all graphical parameters accepted by function lines indicating predictions; NULL to omit (default); additionally to arguments accepted by function

	'lines', one can specify parameters 'x.line' and 'y.line' used to indicated values at the respective axis in margin-lines see mtext for details; 'cex' and 'font' for specifying how these values are plotted. The same color as lines is used per default but can be changed setting 'text.col'. Can also be (numeric) vector (not a list) which results in using default graphical settings. See also parameter Pred.CI.
Pred.CI	(list) with all parameters accepted by function rect; if not NULL, the 100x(1-alpha)% CI of predicted values will be added as a semi-transparent rectangle per default (see examples).
Model	(logical) TRUE = plots the fitted model as subtitle below the main title, FALSE = omits this
CI.method	(character) one of "t", "normal", "chisq" specifying which CI-method to use for deriving confidence intervals
use.log	(logical) TRUE = X- and Y-axis will be log-transformed
Npred	(integer) specifying the number of data points used to plot the fitted model, the larger the smoother (maybe slower if too large)
...	additional parameters passed forward

Value

(matrix) of predictions at user-specified X- or Y-coordinates is invisibly return in case Prediction is not NULL

Author(s)

Andre Schuetzemeister <andre.schuetzeneister@roche.com>

See Also

fit_vfp, predict.VFP, predict_mean

Examples

```
library(VCA)
data(VCAdata1)
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
mat <- get_mat(lst) # automatically selects "total"
mat

res <- fit_vfp(model.no=1:9, Data=mat)
plot(res)
plot(res, type="cv")
plot(res, type="cv", ci.type="lines", ci.col="red",
Grid=list(col="wheat"), Points=list(pch=2, lwd=2, col="black"))

# same for repeatability
mat.err <- get_mat(lst, "error")
res.err <- fit_vfp(1:9, Data=mat.err)
plot(res.err)
```

```
# add predictions to plot, e.g. functional sensitivity
plot(res.err, type="cv", xlim=c(0, 4), Prediction=10)

# variability at X-values are of interest
plot(res.err, type="cv", xlim=c(0, 4), Prediction=list(x=0.5))

# one can specify X- and Y-values in the "Prediction" list-argument
plot(res.err, type="cv", xlim=c(0, 4),
Prediction=list(x=c(0.25, 0.5), y=15))

########################################################################
# another example using CA19_9 data from CLSI EP05-A3

data(CA19_9)

# fit reproducibility model to data
fits.CA19_9 <- anovaVCA(result~site/day, CA19_9, by="sample")

# fit within-laboratory-model treating site as fixed effect
fits.ip.CA19_9 <- anovaMM(result~site/(day), CA19_9, by="sample")

# the variability "between-site" is not part of "total"
fits.ip.CA19_9[[1]]
fits.CA19_9[[1]]

# extract repeatability
rep.CA19_9 <- get_mat(fits.CA19_9, "error")

# extract reproducibility
repro.CA19_9 <- get_mat(fits.CA19_9, "total")

# extract intermediate-precision (within-lab)
ip.CA19_9 <- get_mat(fits.ip.CA19_9, "total")

# fit model (a+bX)^C (model 8) to all three matrices

mod8.repro  <- fit_vfp(repro.CA19_9, 8)
mod8.ip <- fit_vfp(ip.CA19_9, 8)
mod8.rep <- fit_vfp(rep.CA19_9, 8)

# plot reproducibility precision profile first
# leave enough space in right margin for a legend
plot(mod8.repro, mar=c(5.1, 7, 4.1, 15),
type="cv", ci.type="none", Model=FALSE,
Line=list(col="blue", lwd=3),
Points=list(pch=15, col="blue", cex=1.5),
xlim=c(10, 450), ylim=c(0,10),
Xlabel=list(text="CA19-9, kU/L (LogScale) - 3 Patient Pools, 3 QC Materials",
cex=1.5), Title=NULL,
Ylabel=list(text="% CV", cex=1.5),
Grid=NULL, Crit=NULL, log="x")
```

```
# add intermediate precision profile
plot (mod8.ip, type="cv", add=TRUE, ci.type="none",
Points=list(pch=16, col="deepskyblue", cex=1.5),
Line=list(col="deepskyblue", lwd=3), log="x")

# add repeatability precision profile
plot(mod8.rep, type="cv", add=TRUE, ci.type="none",
Points=list(pch=17, col="darkorchid3", cex=1.5),
Line=list(col="darkorchid3", lwd=3), log="x")

# add legend to right margin
legend_rm( x="center", pch=15:17, col=c("blue", "deepskyblue", "darkorchid3"),
cex=1.5, legend=c("Reproducibility", "Within-Lab Precision", "Repeatability"),
box.lty=0)

# repeatability precision profile with some beautifications
plot(mod8.rep, BG="darkgray",
Points=list(pch=17, cex=1.5, col="blue"), Line=list(col="blue"),
Grid=list(x=seq(0, 400, 50), y=seq(0, 100, 10), col="white"),
Xlabel=list(cex=1.5, text="CA19-9 [U/mL]", col="blue"),
Ylabel=list(cex=1.5, text="Repeatability on Variance-Scale", col="blue"),
Crit=list(text.col="white", text.font=2, cex=1.25))
```

`powfun2simple` *Internal Function Model 2.*

Description

Internal Function Model 2.

Usage

```
powfun2simple(x)
```

Arguments

`x`	(numeric) parameter

`powfun3` *Internal Function Model 3.*

Description

Internal Function Model 3.

Usage

```
powfun3(x)
```

Arguments

x	(numeric) parameter

---

powfun3simple *Internal Function Model 3.*

Description

Internal Function Model 3.

Usage

```
powfun3simple(x)
```

Arguments

x	(numeric) parameter

---

powfun4 *Internal Function Model 4.*

Description

Internal Function Model 4.

Usage

```
powfun4(x, potenz)
```

Arguments

x	(numeric) parameter 1
potenz	(numeric) parameter 2

powfun4simple *Internal Function Model 4.*

Description

Internal Function Model 4.

Usage

```
powfun4simple(x, potenz)
```

Arguments

x	(numeric) parameter 1
potenz	(numeric) parameter 2

powfun5 *Internal Function Model 5.*

Description

Internal Function Model 5.

Usage

```
powfun5(x)
```

Arguments

x	(numeric) parameter 1

powfun5simple *Internal Function Model 5.*

Description

Internal Function Model 5.

Usage

```
powfun5simple(x, K)
```

Arguments

x	(numeric) parameter 1
K	(numeric) parameter 2

powfun6 *Internal Function Model 6.*

Description

Internal Function Model 6.

Usage

```
powfun6(x)
```

Arguments

x	(numeric) parameter 1

powfun6simple *Internal Function Model 6.*

Description

Internal Function Model 6.

Usage

```
powfun6simple(x)
```

Arguments

x	(numeric) parameter 1

powfun7 *Internal Function Model 7.*

Description

Internal Function Model 7.

Usage

```
powfun7(x)
```

Arguments

x	(numeric) parameter 1

powfun7simple *Internal Function Model 7.*

Description

Internal Function Model 7.

Usage

```
powfun7simple(x)
```

Arguments

x	(numeric) parameter 1

powfun8 *Internal Function Model 8.*

Description

Internal Function Model 8.

Usage

```
powfun8(x, C, signJ)
```

Arguments

x	(numeric) parameter 1
C	(numeric) parameter 2
signJ	(numeric) parameter 3

powfun8simple *Internal Function Model 8.*

Description

Internal Function Model 8.

Usage

```
powfun8simple(x)
```

Arguments

x	(numeric) parameter 1

powfun9simple *Internal Function Model 9.*

Description

Internal Function Model 9.

Usage

```
powfun9simple(x)
```

Arguments

x	(numeric) parameter 1

precision_plot *Precision Performance Plot of Qualitative Tests.*

Description

This function visualizes what is described in the CLSI EP12 guideline for qualitative test with internal continuous response (ICR). The hit rate, i.e. the number of measurements deemed to have a certain condition. The C5 and C95 concentrations will be derived per default by this function but it can be set to any set of hit rates. The histograms representing normal distribution of imprecisions at specific concentrations will be scaled to nicely fit into the plot, i.e. the area under the plot will not be equal to 1.

Usage

```
precision_plot(
  vfp,
  model.no = NULL,
  cutoff,
  prob = c(0.05, 0.5, 0.95),
  col = c("blue", "black", "red"),
  Cutoff = list(),
  Title = list(),
  Xlabel = list(),
  Ylabel = list(),
  HRLine = list(),
  Legend = FALSE,
  nclass = -1,
  BG = "gray90",
  digits = 3,
  alpha = 0.15,
```

```
  alpha2 = 0,
  xlim = NULL,
  col.grid = "white",
  Nrand = 1e+06
)
```

Arguments

vfp	(VFP) object modeling imprecision over the measuring range
model.no	(integer) specifying the VFP-model to used
cutoff	(numeric) specifying one or two cutoff(s), the latter will implicitly define an equivical zone with implications on how 'prob' will be interpreted (see 'prob' for details)
prob	(numeric) values 0 < x < 1 specifying coverage probability of an respecitive normal distribution at cutoff, in case of two cutoffs all elements of 'prob' < 0.5 will be evaluated in regard to cutoff 1, and all 'prob' > 0.5 in regard to cutoff 2
col	(character) strings specifying colors of the different distributions, which will be plotted semi-transparent using 'alpha1' for specifying the level of transparency (1=opaque, 0=fully transparent)
Cutoff	(list) specifying all parameters of the abline function. Vertical lines representing one or two cutoffs can be specified, the color will be re-used for a label in the upper margin. Set to NULL to omit.
Title	(list) specifying all parameters applicable in function mtext for specifying a main title of the plot
Xlabel	(list) specifying all parameters applicable in function mtext for specifying the X-axis label of the plot
Ylabel	(list) specifying all parameters applicable in function mtext for specifying the Y-axis label of the plot
HRLine	(list) specifying all parameters applicable in lines of the line representing the hit rate developing from 0% to 100%
Legend	(logical) TRUE = a legend is added to the plot
nclass	(integer) number of classes in the histograms representing normal distributions of imprecision at Cx-concentrations, number<10 will lead to automatically determining appropriate numbers per histogram (default)
BG	(character) string specifying a background color
digits	(integer) number of significant digits used to indicated concentrations Cx
alpha	(numeric) value 0<=x<=1 specifying the level of transparency of histograms
alpha2	(numeric) similar to 'alpha' referring to the coverage probability, i.e. setting it to a value < 0 will highlight coverage probabilities in histograms
xlim	(numeric) plotting limits in X-direction
col.grid	(character) string specifying a color name to be used for the grid providing orientation in X- and Y-direction
Nrand	(integer) specifying the number of data points simulated to represent a normal distribution

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
## Not run:
# perform variance component analysis
library(VCA)
data(VCAdata1)
# perform VCA-anaylsis
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
# transform list of VCA-objects into required matrix
mat <- get_mat(lst) # automatically selects "total"
mat
# fit all models batch-wise, the best fitting will be used automatically
res <- fit_vfp(model.no=1:9, Data=mat)
# plot hit and visualize imprecision usign default settings
precision_plot(res, cutoff=20)
# without normal distribution at cutoff do
precision_plot(res, cutoff=20, prob=c(.05, .95), col=c("blue", "red"))
# highlight the proportion > cutoff (hit rate) more
precision_plot(res, cutoff=20, prob=c(.05, .95), col=c("blue", "red"), alpha2=.5)
# plot with legend
precision_plot(res, cutoff=20, prob=c(.05, .95), col=c("blue", "red"), alpha2=.5, Legend=TRUE)
# use different probabilities and colors
precision_plot(res, cutoff=20, prob=c(.05, .95), col="black", alpha2=.3)

# now using two cutoffs, i.e. with equivocal zone
precision_plot( res, cutoff=c(17, 19), prob=c(.05, .95), col=c("mediumblue", "red3"),
alpha2=.5, HRLine=list(col=c("mediumblue", "red3")))

## End(Not run)
```

---

predict.VFP *Predict Method for Objects of Class 'VFP'.*

Description

Predictions are made for the variance (type="vc"), standard deviation ("sd") or coefficient of variation ("cv") and their corresponding confidence intervals. The latter are calculated primarily on the variance scale and then transformed to the other scales, if required.

Usage

```
## S3 method for class 'VFP'
predict(
  object,
  model.no = NULL,
```

```
  newdata = NULL,
  alpha = 0.05,
  dispersion = NULL,
  type = c("vc", "sd", "cv"),
  CI.method = c("chisq", "t", "normal"),
  use.log = FALSE,
  ...
)
```

Arguments

object	(object) of class "VFP"
model.no	(integer) specifying a fitted model stored in 'object'
newdata	(numeric) optionally, a vector specifying mean-values for which predictions on the user-defined scale ('type') are requested. If omitted, fitted values will be returned.
alpha	(numeric) value specifying the 100 x (1-alpha)% confidence interval of predicted values
dispersion	(numeric) NULL = the dispersion will be set =1 (should usually not be changed; For the Saddler model, the dispersion is 1.), numeric value = the dispersion parameter will be used as specified
type	(character) specifying on which scale the predicted values shall be returned, possible are "vc" = variance, "sd"=standard deviation, "cv"=coefficient of variation
CI.method	(character) one of "t", "normal", "chisq" specifying which CI-method to use
use.log	(logical) TRUE = X- and Y-axis will be log-transformed
...	additional parameters passed forward to function predict.gnm

Value

(data.frame) with numeric variables:

Mean	value at which predictions were requested
Fitted	prediction at 'Mean'
SE	standard error of prediction
Scale	residual scale
LCL	lower confidence limit of the 100x(1-'alpha')% CI
UCL	upper confidence limit of the 100x(1-'alpha')% CI

Examples

```
library(VCA)
data(VCAdata1)
lst <- anovaVCA(y~(device+lot)/day/run, VCAdata1, by="sample")
mat <- get_mat(lst) # automatically selects "total"
res <- fit_vfp(model.no=1:9, Data=mat)
```

```
predict(res)
predict(res, dispersion=0.95)
```

`predict_mean` *Finding X-Value for Given Y-Value Using a Bisection-Approach.*

Description

For given variability-values (Y-axis) on one of three scales (see 'type'), those values on the X-axis are determined which give fitted values equal to the specification.

Usage

```
predict_mean(
  obj,
  type = c("vc", "sd", "cv"),
  model.no = NULL,
  alpha = 0.05,
  newdata = NULL,
  tol = 1e-04,
  ci = TRUE,
  ...
)
```

Arguments

`obj`	(object) of class 'VFP'
`type`	(character) "vc" = variance, "sd" = standard deviation = sqrt(variance), "cv" = coefficient of variation
`model.no`	(integer) specifying which model to use in case 'obj' represents multiple fitted models
`alpha`	(numeric) value specifying the 100x(1-alpha)% confidence interval for the predicted value(s)
`newdata`	(numeric) values representing variability-values on a specific scale ('type')
`tol`	(numeric) tolerance value relative to 'newdata' specifying the stopping criterion for the bisection algorithm, also used to evaluate equality of lower and upper bounds in a bisection step for checking whether a boundary can be determined or not
`ci`	(logical) indicates whether confidence intervals for predicted concentrations are required (TRUE) or not (FALSE), if 'newdata' contains many values the overall computation time can be minimized to 1/3 leaving out runs of the bisection-algorithm for LCL and UCL
`...`	additional parameter passed forward or used internally

Details

This is achieved using a bisection algorithm which converges according to the specified tolerance 'tol'. In case of 'type="cv"', i.e. if specified Y-values are coefficients of variation, these are interpreted as percentages (15 = 15%).

Value

(data.frame) with variables "Mean" (X-value), "VC", "SD" or "CV" depending on 'type', "Diff" the difference to the specified Y-value, "LCL" and "UCL" as limits of the 100x(1-alpha)% CI.

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

See Also

fit_vfp, predict.VFP, plot.VFP

Examples

```
# perform variance component analyses first
library(VCA)
data(CA19_9)
fits.CA19_9 <- anovaVCA(result~site/day, CA19_9, by="sample")

# extract repeatability
mat.CA19_9 <- get_mat(fits.CA19_9, "error")
res.CA19_9 <- fit_vfp(mat.CA19_9, 1:9)
summary(res.CA19_9)
print(res.CA19_9)

# predict CA19_9-concentration with 5\% CV
predict_mean(res.CA19_9, newdata=5)

# this is used in function plot.VFP as well
plot(res.CA19_9, Prediction=list(y=5), type="cv")
plot(res.CA19_9, Prediction=list(y=5), type="cv",
xlim=c(0, 80), ylim=c(0, 10))
```

---

predict_model_ep17 *Predict Method for Objects of Class 'modelEP17'.*

Description

This is a helper function not intented to be used directly.

Usage

```
predict_model_ep17(
  object,
  newdata = NULL,
  alpha = 0.05,
  ci.type = c("vc", "sd", "cv"),
  CI.method = c("chisq", "t", "normal"),
  use.log = FALSE,
  ...
)
```

Arguments

object	(object) of class 'modelEP17'
newdata	(numeric) vector of data points at which prediction shall be made
alpha	(numeric) value defining the 100(1-alpha)% confidence interval for predictions
ci.type	(character) string specifying on which scale prediction shall be made
CI.method	(character) string specifying which type of CI shall be used
use.log	(logical) TRUE X- and Y-values will be returned on log-scale
...	additional arguments

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

---

print.VFP *Print Objects of Class 'VFP'*

Description

Print Objects of Class 'VFP'

Usage

```
## S3 method for class 'VFP'
print(x, model.no = NULL, digits = 4, ...)
```

Arguments

x	(object) of class 'VFP'
model.no	(integer) specifying a fitted model in 'x', if NULL the best fitting model will be printed, i.e. the one with min(AIC)
digits	(integer) number of significant digits
...	additional parameters passed forward

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
library(VCA)
data(CA19_9)
fits.CA19_9 <- anovaVCA(result~site/day, CA19_9, by="sample")
# extract repeatability
mat.CA19_9 <- get_mat(fits.CA19_9, "error")
res.CA19_9 <- fit_vfp(mat.CA19_9, 1:9)
res.CA19_9
```

RealData1 *Result of a Real-World Precsion Experiment on 10 Samples*

Description

(data.frame) representing the results of an imprecision experiment. There are 10 observations and six variables which can be used directly as input for function 'fit.vfp'. Different samples are indicated by variable "No".

Usage

```
data(RealData1)
```

Format

data.frame with 10 observations and 6 variables.

ReproData *Serum Work Area Example Data of a Reproducibility Experiment*

Description

Multi-site precision experiment using 6 human serum-samples and 2 precision control samples, which were all measured at each site on 12 days, with 2 runs per day and 2 replicates per run, i.e. there are 8 (sample) x 3 (sites/labs) x 12 (days) x 2 (runs) x 2 (replicates) = 1152 observations overall.

Usage

```
data(ReproData)
```

Format

data.frame with 1152 observations and 5 variables.

signif2 *Adapted Version of Function 'signif'*

Description

This function adapts base-function `signif` by always returning integer values in case the number of requested significant digits is less than the the number of digits in front of the decimal separator.

Usage

```
signif2(x, digits = 4, force = TRUE, ...)
```

Arguments

`x`	(numeric) value to be rounded to the desired number of significant digits
`digits`	(integer) number of significant digits
`force`	(logical) TRUE = force the return value to have at least 4 significant digits, i.e. to integers with less digits zeros will be appended after the decimal separator, otherwise the return value will be casted from character to numeric
`...`	additional parameters

Value

number with 'digits' significant digits, if 'force=TRUE' "character" objects will be returned otherwise objects of mode "numeric"

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

summary.VFP *Summary Objects of Class 'VFP'*

Description

Summary Objects of Class 'VFP'

Usage

```
## S3 method for class 'VFP'
summary(
  object,
  model.no = NULL,
  digits = 4,
  type = c("simple", "complex"),
  ...
)
```

Arguments

object	(object) of class 'VFP'
model.no	(integer) specifying a fitted model in 'x', if NULL the best fitting model will be printed, i.e. the one with min(RSS)
digits	(integer) number of significant digits
type	(character) "simple" = short summary, "complex" = calls the summary-method for objects of class "gnm"
...	additional parameters passed forward

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com>

Examples

```
library(VCA)
data(CA19_9)
fits.CA19_9 <- anovaVCA(result~site/day, CA19_9, by="sample")
# extract repeatability
mat.CA19_9 <- get_mat(fits.CA19_9, "error")
res.CA19_9 <- fit_vfp(mat.CA19_9, 1:9)
summary(res.CA19_9)
print(res.CA19_9)
```

T4S9_99 *Example Data T4S9_99.VFP (T4 RIA) from the Variance Function Program 12.0 from Sadler*

Description

Clinical specimen T4 RIA duplicates produced during 1999 by a staff member.

Usage

```
data(T4S9_99)
```

Format

data.frame with 8553 observations and 2 variables.

References

www.aacb.asn.au/AACB/Resources/Variance-Function-Program

t_coef *Transformation of Coefficients.*

Description

This function performs transformation from the original parameterization into the 'VFP'-package internal re-parameterized form.

Usage

```
t_coef(
  coeffs0,
  K = NULL,
  Maxi = NULL,
  model = NULL,
  signJ = NULL,
  eps = sqrt(.Machine$double.eps),
  ...
)
```

Arguments

coeffs0	(numeric) vector of function coefficients to be transformed into the re-parameterized form
K	(numeric) constant value 'K'
Maxi	(numeric) max. value
model	(integer) specifying which model shall be back-transformed
signJ	(integer) either 1 or -1
eps	(numeric) constant used instead of zero in case of log-transformation
...	additional parameters

Details

In the 'VFP' package models are re-parameterized to have better control over the constrained solution-space, i.e. only models may be fitted generating non-negative fitted values. This function is intended to be for internal use only.

Value

(numeric) vector of coefficients in re-parameterized form

Author(s)

Andre Schuetzenmeister <andre.schuetzenmeister@roche.com> Florian Dufey <florian.dufey@roche.com>

Index