Introduction to the MethComp package

(compiled Friday 20th May, 2011, 01:31)

Bendix Carstensen Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bxc@steno.dk
www.biostat.ku.dk/~bxc

Contents

Chapter 1

Overview of MethComp

The purpose of the MethComp package is to provide computational tools to manipulate, display and analyze data from method comparison studies. A method comparison study is a study where two methods of quantitative measurement are compared by measuring the same set of items with both methods.

There may be more than two methods, and there may be replicate measurements on each item by each method.

1.1 Data structures

In general we are concerned with measurements by different methods, on different items (persons, samples), possibly replicated.

Often such data are represented by a row of measurements for each item, with possible replicates listed either below or beside each other. This implicitly assumes that some replicate measurements belong together, which is not necessarily the case in all situations.

All functions in MethComp assume data to be represented in the "long" form, with one measurement on each row, and columns to indicate method, item and replicate. Specifically, we assume the following columns are available in a data frame:

- `meth` The measurement method. Numeric or factor.
- `item` Identification of item (person, sample). Numeric or factor.
- `repl` Replicate number. Numeric or factor.
- `y` The measurement by method `meth` on item `item`, replicate number `repl`.

There is a class, "`Meth`" for this kind of data frame. It is a data frame with the facors `meth`, `item` and `repl` representing the classification, and the numerical variable `y` representing the measurements.

A dataframe with method comparison data in the *long* format is converted to a `Meth` object by using the `Meth` function on it:

```
> data( ox )
> str( ox )
```

```
'data.frame':        354 obs. of  4 variables:
 $ meth: Factor w/ 2 levels "CO","pulse": 1 1 1 1 1 1 1 1 1 1 ...
 $ item: num  1 1 1 2 2 2 3 3 3 4 ...
 $ repl: num  1 2 3 1 2 3 1 2 3 1 ...
 $ y   : num  78 76.4 77.2 68.7 67.6 68.3 82.9 80.1 80.7 62.3 ...

> ox <- Meth( ox )

The following variables from the dataframe
"ox" are used as the Meth variables:
meth: meth
item: item
repl: repl
   y: y
        #Replicates
Method    1   2   3 #Items #Obs: 354 Values:  min  med  max
  CO      1   4  56     61       177         22.2 78.6 93.5
  pulse   1   4  56     61       177         24.0 75.0 94.0

> summary( ox )

        #Replicates
Method    1   2   3 #Items #Obs: 354 Values:  min  med  max
  CO      1   4  56     61       177         22.2 78.6 93.5
  pulse   1   4  56     61       177         24.0 75.0 94.0
```

If variables `meth`, `item`, `repl` and `y` are not availabe in the data frame we may create them on the fly or give the variable positions as arguments to the `Meth` function:

```
> data( fat )
> str( fat )

'data.frame':        258 obs. of  5 variables:
 $ Id : num  1 1 1 3 3 3 5 5 5 11 ...
 $ Obs: Factor w/ 2 levels "KL","SL": 1 1 1 1 1 1 1 1 1 1 ...
 $ Rep: num  1 2 3 1 2 3 1 2 3 1 ...
 $ Sub: num  1.6 1.7 1.7 2.8 2.9 2.8 2.7 2.8 2.9 3.9 ...
 $ Vic: num  4.5 4.4 4.7 6.4 6.2 6.5 3.6 3.9 4 4.3 ...

> sc <- Meth( fat, 2, 1, 3, 4 )

The following variables from the dataframe
"fat" are used as the Meth variables:
meth: Obs
item: Id
repl: Rep
   y: Sub
       #Replicates
Method          3 #Items #Obs: 258 Values:  min med max
    KL         43     43       129         0.39 1.7 4.2
    SL         43     43       129         0.51 1.7 4.1

> str( sc )

Classes 'Meth' and 'data.frame':        258 obs. of  5 variables:
 $ meth: Factor w/ 2 levels "KL","SL": 1 1 1 1 1 1 1 1 1 1 ...
 $ item: Factor w/ 43 levels "1","2","3","4",..: 1 1 1 3 3 3 5 5 5 11 ...
 $ repl: Factor w/ 3 levels "1","2","3": 1 2 3 1 2 3 1 2 3 1 ...
 $ y   : num  1.6 1.7 1.7 2.8 2.9 2.8 2.7 2.8 2.9 3.9 ...
 $ Vic : num  4.5 4.4 4.7 6.4 6.2 6.5 3.6 3.9 4 4.3 ...
```

```
> summary( sc )

       #Replicates
Method          3 #Items #Obs: 258 Values:  min med max
    KL         43     43       129         0.39 1.7 4.2
    SL         43     43       129         0.51 1.7 4.1
```

We may even give some of them as names of the columns in the dataframe:

```
> vi <- Meth( fat, 2,1,"Rep","Vic" )

The following variables from the dataframe
"fat" are used as the Meth variables:
meth: Obs
item: Id
repl: Rep
   y: Vic
       #Replicates
Method          3 #Items #Obs: 258 Values:  min med max
    KL         43     43       129          2.0 3.9 6.5
    SL         43     43       129          2.3 4.1 6.7
```

However, more complicated operations on the dataframe is best done on the fly using the `with` function (from the `base` package):

```
> data( hba1c )
> str( hba1c )

'data.frame':        835 obs. of  6 variables:
 $ dev   : Factor w/ 3 levels "BR.V2","BR.VC",..: 2 2 2 2 2 2 2 2 1 1 ...
 $ type  : Factor w/ 2 levels "Cap","Ven": 2 2 2 2 1 1 1 1 2 2 ...
 $ item  : num  12 12 12 12 12 12 12 12 12 12 ...
 $ d.samp: num  1 1 1 1 1 1 1 1 1 1 ...
 $ d.ana : num  2 3 4 5 2 3 4 5 2 3 ...
 $ y     : num  8.7 8.7 8.7 8.7 9.2 9 8.8 8.7 9.4 9.3 ...

> hb1  <- with( hba1c,
+               Meth( meth = interaction(dev,type),
+                     item = item,
+                     repl = d.ana-d.samp,
+                        y = y, print=TRUE ) )

            #Replicates
Method          3     4 #Items #Obs: 835 Values:  min med  max
  BR.V2.Cap     0    38     38       152          5.3 8.0 12.6
  BR.VC.Cap    19    19     38       133          5.3 8.2 12.1
  Tosoh.Cap     0    38     38       152          5.0 7.8 11.8
  BR.V2.Ven    19    19     38       133          5.5 8.1 12.0
  BR.VC.Ven    19    19     38       133          5.3 8.0 11.6
  Tosoh.Ven    20    18     38       132          5.3 8.0 12.1

> str( hb1 )

Classes 'Meth' and 'data.frame':        835 obs. of  4 variables:
 $ meth: Factor w/ 6 levels "BR.V2.Cap","BR.VC.Cap",..: 5 5 5 5 2 2 2 2 4 4 ...
 $ item: Factor w/ 38 levels "1","2","3","4",..: 12 12 12 12 12 12 12 12 12 12 ...
 $ repl: Factor w/ 5 levels "0","1","2","3",..: 2 3 4 5 2 3 4 5 2 3 ...
 $ y   : num  8.7 8.7 8.7 8.7 9.2 9 8.8 8.7 9.4 9.3 ...
```

Objects of class `Meth` (which inherits from `data.frame`) has methods such as `summary`, `plot`, `subset` and `transform`. The functions mostly do not require the data to be in `Meth` format — if a dataframe with the right columns is supplied, it is normally converted internally to `Meth` format.

1.1.1 Wide format data

Sometimes data frames comes in the wide format, that is with measurements by different methods in different columns. In this case a `Meth` object is formed by giving the variables containing measurements by different methods as a vector argument to `y`, either as numbers of columns or names of columns:

```
> data( rainman )
> str( rainman )

'data.frame':        30 obs. of  6 variables:
 $ SAND: int  120 48 88 32 24 100 52 80 72 96 ...
 $ ME  : int  175 50 150 45 25 125 70 145 85 110 ...
 $ TM  : int  120 50 75 22 22 80 50 75 90 110 ...
 $ AJ  : int  105 45 75 28 25 91 48 68 55 84 ...
 $ BM  : int  100 50 60 30 20 80 45 55 60 65 ...
 $ LO  : int  100 70 80 30 20 70 50 60 60 65 ...

> RM <- Meth( rainman, item=1, y=2:6 )

The following variables from the dataframe
"rainman" are used as the Meth variables:
item: SAND
   y: ME TM AJ BM LO
       #Replicates
Method          3 #Items #Obs: 150 Values:  min med max
    AJ         10     10        30           18  57 120
    BM         10     10        30           15  62 120
    LO         10     10        30           20  55 100
    ME         10     10        30           24  90 200
    TM         10     10        30           20  75 120

> head( RM )

  meth item repl   y
1   ME  120    1 175
2   ME   48    1  50
3   ME   88    1 150
4   ME   32    1  45
5   ME   24    1  25
6   ME  100    1 125
```

1.2 Function overview

The following is a brief overview of the functions in the `MethComp` package. The full documentation is in the help pages for the functions, and an illustration of the way they work can be obtained by referring to the examples in the help pages. The help page for `plot.meth` is brought up by:

```
> ?plot.Meth
```

The example code from the manual page can be run directly by:

```
> example( plot.Meth )
```

1.2.1 Graphical functions

The graphical functions generally have a lot of arguments that can be used to fine-tune the looks of the plots. Refer to the help page for each to see them all.

`BA.plot` Makes a Bland-Altman plot of two methods from a data frame with method comparison data, and computes limits of agreement. The plotting is really done by a call to the function `BlandAltman`. The default is to plot the two first methods against each other.

`BlandAltman` draws a Bland-Altman plot and computes limits of agreement.

`bothlines` Adds regression lines of y on x and vice versa to a scatter plot. Optionally, the Deming regression line can be added too.

`plot.Meth` Plots all methods against each other in a square matrix, both as a scatter plot (below diagonal) and as a Bland-Altman plot (above diagonal).

`plot.MethComp` plots the estimated conversion between methods with a ± 2 sd interval, corresponding to approx. 95% prediction interval. Recognizes transformations applied to data.

1.2.2 Data manipulating functions

`make.repl` Generates (or replaces) a `repl` column in a `Meth` object.

`perm.repl` Randomly permutes replicates within (`method`,`item`) and assigns new replicate numbers.

`to.wide` Transforms a data frame in the long form to the wide form where separate columns for each method are generated, with one row per (`item`,`repl`).

`to.long` Reverses the result of `to.wide`. The function can also generate a long form dataset from a dataset with different methods beside each other.

`summary.Meth` Tabulates items by method and no. replicates for a `Meth` object.

`Meth.sim` Simulates a dataset from a method comparison experiment for given parameters for bias, exchangeability and variance component sizes.

1.2.3 Analysis functions

`DA.reg` Regresses the differeneces between methods on the averages and derives approximate linear conversion equations, based on [**?**].

`Deming` Performs Deming regression, i.e. regression with errors in both variables.

`BA.est` Estimates in the variance components models underlying the concept of limits of agreement, and returns the bias and the variance components. The model used assumes constant bias between methods.

`AltReg` Estimates via alternating regressions in the general model. Returns estimates of mean conversion parameters and variance components.

`MCmcmc` Estimates via `BUGS` in the general model with non-constant bias (and in the future) possibly non-constant standard deviations of the variance components. Produces a `MCmcmc` object, which is an `mcmc.list` object with some extra attributes. `mcmc.list` objects are handeled by the `coda` package, so this is required when calling `MCmcmc`.

1.2.4 Reporting functions

Some of these functions take an `MCmcmc` object as input, others will postprocess the output of `DA.reg`, `BA.est` or `AltReg`.

The functions `DA.reg`, `BA.est`, `AltReg` return objetcs that have class `MethComp`, whereas the result of `MCmcmc` can be converted to an object of this type by the `MethComp` function. The reason for this is that the results of the `MCmcmc` function is output from an MCMC-simulation which we may want to monitor by special functions. The `MethComp` function only extracts the central summaries from the `MCmcmc` object assuming the chains have reached convergence.

`print.MethComp` Prints a table of conversion equation between methods analyzed, with prediction standard deviations.

`print.MCmcmc` Prints a table of conversion equation between methods analyzed, with prediction standard deviations, but also gives summaries of the posteriors for the parameters that constitute the conversion algorithms.

`plot.MethComp`, `plot.MCmcmc` Plots the conversion lines between methods with prediction limits.

`post.MCmcmc` Plots smoothed posterior densities for the estimates. Primarily of interest for the variance components, but it has aruments to produce the posterior of the intercepts and the slopes of the conversion lines between methods too.

`check.MCmcmc` Makes diagnistic plots of the traces of the chains included in the `MCmcmc` object.

Chapter 2

Worked examples

2.1 Fat measurements: Exchangeable replicates

The `fat` data from the `MethComp` package contains measurements of subcutaneous and visceral fat on 43 persons, by two observers, KL and SL. Each measurement is replicated 3 times.

First we examine the names in the dataframe, and then use `Meth` to convert it to a form that comply with that required by the functions in the `MethComp` package for analyzing visceral fat — we convert it to a `Meth` object:

```
> data(fat)
> str(fat)

'data.frame':        258 obs. of  5 variables:
 $ Id : num  1 1 1 3 3 3 5 5 5 11 ...
 $ Obs: Factor w/ 2 levels "KL","SL": 1 1 1 1 1 1 1 1 1 1 ...
 $ Rep: num  1 2 3 1 2 3 1 2 3 1 ...
 $ Sub: num  1.6 1.7 1.7 2.8 2.9 2.8 2.7 2.8 2.9 3.9 ...
 $ Vic: num  4.5 4.4 4.7 6.4 6.2 6.5 3.6 3.9 4 4.3 ...

> vis <- Meth( fat, 2,1,3,5 )

The following variables from the dataframe
"fat" are used as the Meth variables:
meth: Obs
item: Id
repl: Rep
   y: Vic
      #Replicates
Method          3 #Items #Obs: 258 Values:  min med max
    KL         43     43       129          2.0 3.9 6.5
    SL         43     43       129          2.3 4.1 6.7

> str(vis)

Classes 'Meth' and 'data.frame':        258 obs. of  5 variables:
 $ meth: Factor w/ 2 levels "KL","SL": 1 1 1 1 1 1 1 1 1 1 ...
 $ item: Factor w/ 43 levels "1","2","3","4",..: 1 1 1 3 3 3 5 5 5 11 ...
 $ repl: Factor w/ 3 levels "1","2","3": 1 2 3 1 2 3 1 2 3 1 ...
 $ y   : num  4.5 4.4 4.7 6.4 6.2 6.5 3.6 3.9 4 4.3 ...
 $ Sub : num  1.6 1.7 1.7 2.8 2.9 2.8 2.7 2.8 2.9 3.9 ...

> summary(vis)
```

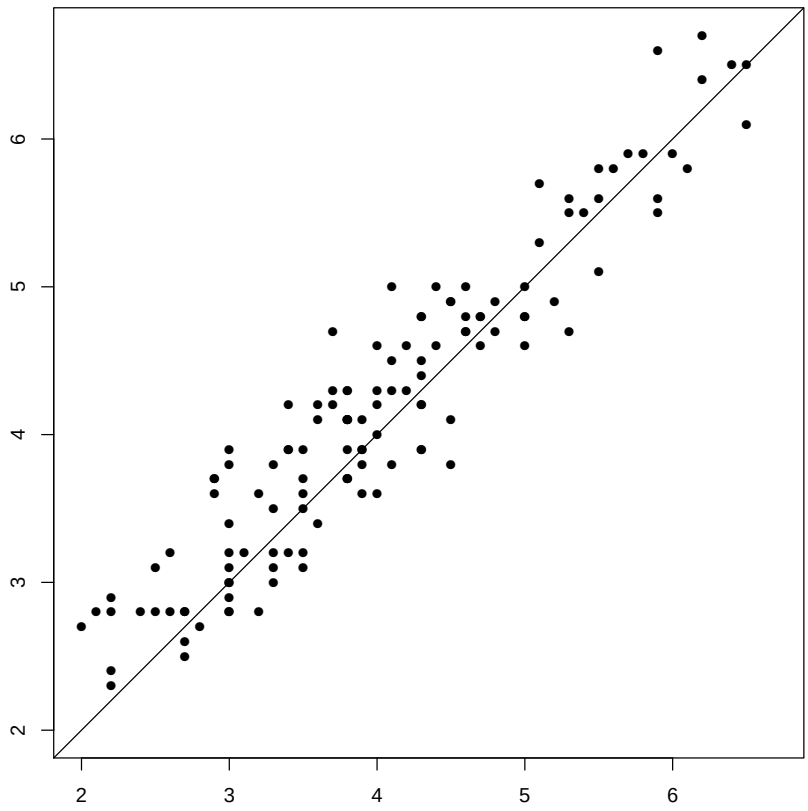

Figure 2.1: *Two observers measuring visceral fat.*

```
       #Replicates
Method          3 #Items #Obs: 258 Values:  min med max
    KL         43     43       129          2.0 3.9 6.5
    SL         43     43       129          2.3 4.1 6.7
```

The two methods plotted against each other requires that we use the replicate number for pairing the measurements; so we just keep the ordering among the replicates when using `to.wide`:

```
> pw <- to.wide( vis )

Note:
 Replicate measurements are taken as separate items!

> par( mar=c(3,3,1,1) )
> with(pw, plot( SL ~ KL, pch=16, xlim=range(vis$y), ylim=range(vis$y) ) )
> abline( 0,1 )
```

Since replicates are exchangeable *witin* (method, item) we should get the same sort of overview of the data after a random permutation of the replicates. Plotting the data using the original replicate numbers for pairing and then a random permutation is shown in figure 2.2:

```
> plot( vis )
```

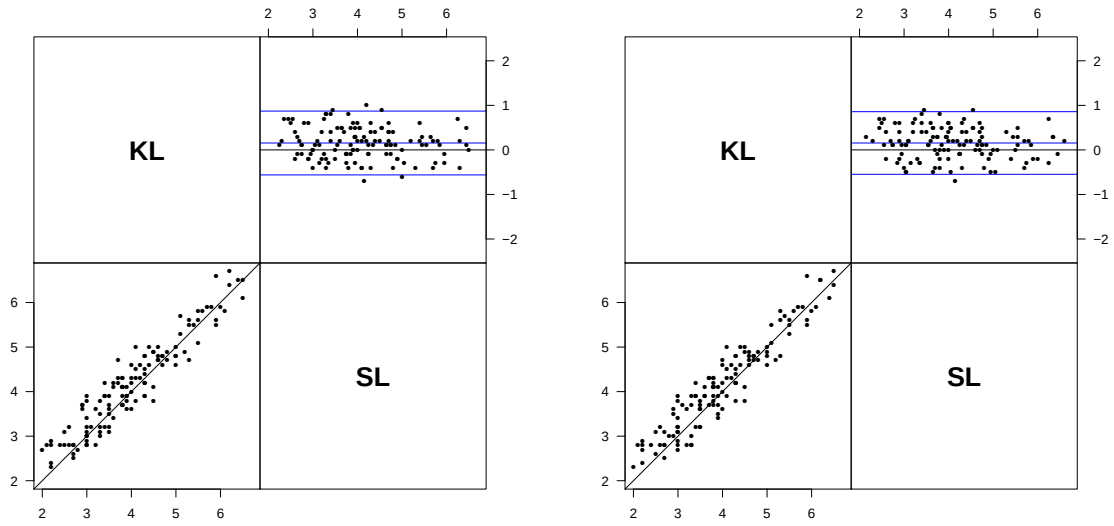

Figure 2.2: *Plot of two methods of measuring visceral fat, using different pairings of the replicates; the left panel is using the pairing in the original coding, the right panel is with a random permutation of replicates.*

```
Note:
 Replicate measurements are taken as separate items!
```

```
> plot( perm.repl( vis ) )
```

```
Note:
 Replicate measurements are taken as separate items!
```

These two plots are shown in figure 2.2 where it is pretty clar that the random permutation of replicates has little effect.

`BA.plot` produces a Bland-Altman plot and computes the limits of agreement using the pairing of replicates across methods based on the numbering of replicates. However we do not want the replicates to be connected, so we must specify this explicitly:

```
> par( mar=c(3,3,3,3), mgp=c(3,1,0)/1.6 )
> BA.plot( vis, conn.repl=FALSE )
```

```
Limits of agreement:
    SL - KL  2.5% limit 97.5% limit    SD(diff)
  0.1550388  -0.5612718   0.8713493   0.3581553
```

We see that using this approximation we get limits of agreement for KL−SL of $(-0.86, 0.55)$.

Moreover, there seems to be no indication that the difference between observers or the variance varies with the level of measurement. This can be a bit more formally tested using the `DA.reg` function (again using the existing pairing of replicates):

```
> DA.reg( vis )
```

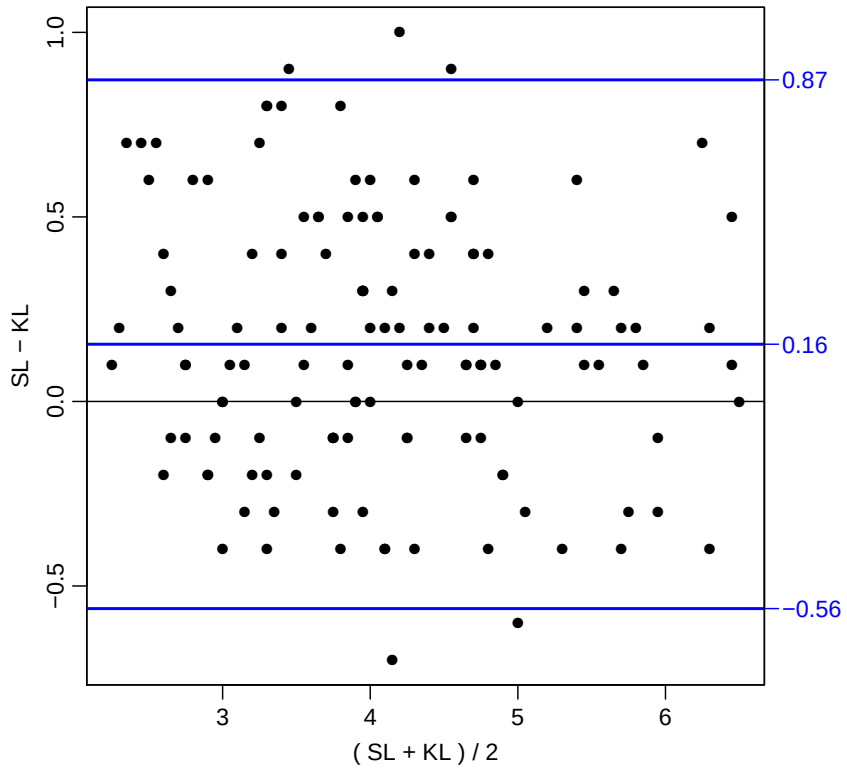

Figure 2.3: *Bland-Altman plot of two observers measuring visceral fat.*

```
 Conversion between methods:
            alpha   beta sd.pred  beta=1  sd.|A=4  slope(sd)  sd.=K
To: From:
KL  KL      0.000  1.000      NA      NA       NA         NA     NA
    SL     -0.340  1.044   0.365   0.158    0.366     -0.024  0.275
SL  KL      0.326  0.957   0.349   0.158    0.366     -0.024  0.275
    SL      0.000  1.000      NA      NA       NA         NA     NA
```

From the last two columns (p-values for tests of constant difference and constant sd.) it is clear that there are no obvious violations of the assumptions about constant difference or about constant variation across the range of measurements.

Setting up a proper variance component model we get only slightly different limits of agreement (note that we must specify the replicates to be exchangeable):

```
> ( vis.est <- BA.est( vis, linked=FALSE ) )

 Conversion between methods:
            alpha   beta     sd   LoA: lower  upper
To: From:
KL  KL      0.000  1.000  0.273       -0.545  0.545
    SL     -0.155  1.000  0.364       -0.883  0.573
SL  KL      0.155  1.000  0.364       -0.573  0.883
    SL      0.000  1.000  0.245       -0.490  0.490

 Variance components (sd):
   IxR   MxI   res
```

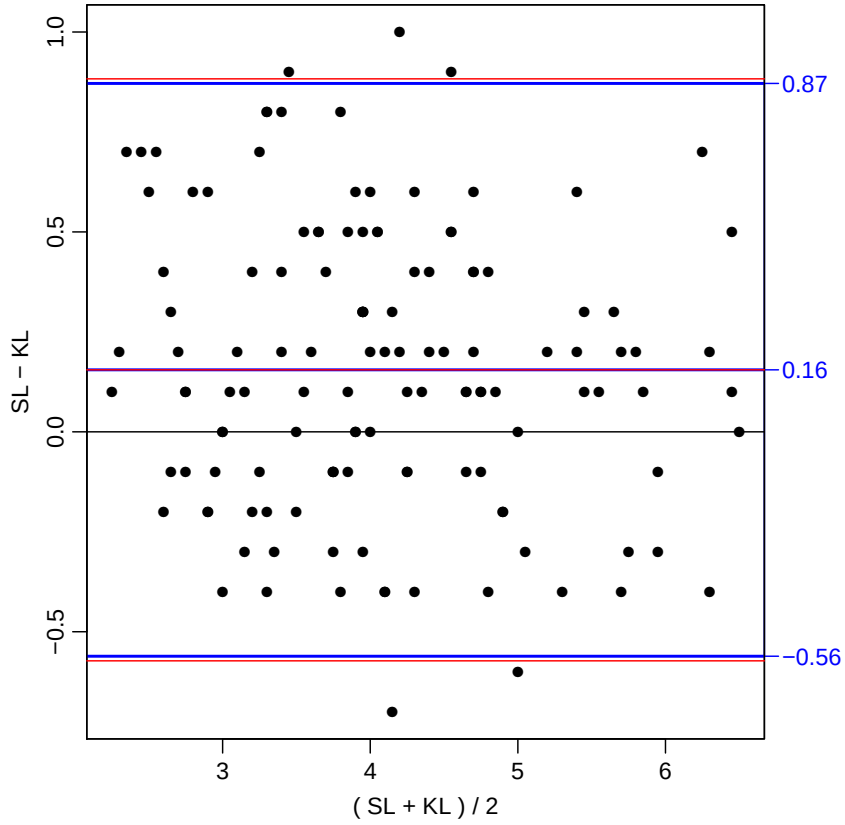

Figure 2.4: *Bland-Altman-plot of two methods of measuring visceral fat, using different pairings of the replicates. The blue lines are the LoA based on taking the paired replicates as items, the red lines are based on the estimates from the proper variance component model.*

```
KL    0 0.181 0.193
SL    0 0.181 0.173
```

Moreover we get the coefficient of reproducibility for each of the methods; that is an upper 95% confidence interval for the absolute difference between two measurements by the same method on the same

We can visualize the difference between the *ad-hoc*-computed LoA and the model based ones by plotting them in the same graph:

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( vis, conn.repl=FALSE )

Limits of agreement:
    SL - KL  2.5% limit 97.5% limit    SD(diff)
  0.1550388  -0.5612718   0.8713493   0.3581553

> abline( h=vis.est$LoA[1:3], col="red" )
```

As predicted by the theory, the limits based on the *ad-hoc* paired replicates are roughly equal to those derived from the proper variance component model — see figure 2.4.

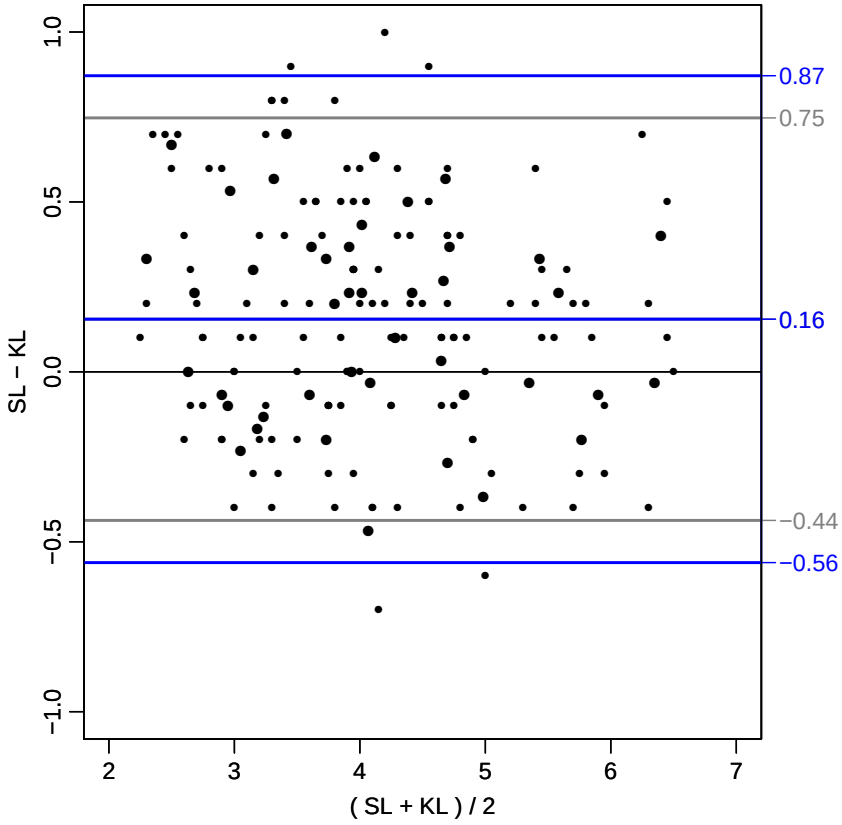

Figure 2.5: *Bland-Altman-plot of two methods of measuring visceral fat, based on the arbitrary pairing of the replicates (black) and on the mean over replicates (grey).*

In order to illustrate the effect of basing the limits of agreement on the mean over the replicates we use the argument `mean.repl`, and the trick of using `par(new=T)` to over plot:

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot(vis,mean.repl=T,limy=c(-1,1),limx=c(2,7),col=gray(0.7),col.lines=gray(0.5), conn.repl=FALS

Limits of agreement:
    SL - KL  2.5% limit 97.5% limit    SD(diff)
  0.1550388  -0.4371295   0.7472070   0.2960841

> par(new=T)
> BA.plot(vis,mean.repl=F,limy=c(-1,1),limx=c(2,7),cex=0.7,conn.repl=FALSE)

Limits of agreement:
    SL - KL  2.5% limit 97.5% limit    SD(diff)
  0.1550388  -0.5612718   0.8713493   0.3581553
```

The two superposed Bland-Altman plots are shown in figure **??**.

2.2 Cardiac output: Linked replicates?

The dataset is adapted from table 4 in: JM Bland and DG Altman: Measuring agreement in method comparison studies. Statistical Methods in Medical Research,

8:136-160, 1999. Originally supplied to Bland & Altman by Dr LS Bowling, see: Bowling LS, Sageman WS, O'Connor SM, Cole R, Amundson DE. Lack of agreement between measurement of ejection fraction by impedance cardiography versus radionuclide ventriculography. Critical Care Medicine 1993; 21: 1523-27.

It consists of measurements of cardiac output on 12 persons. For each person the cardiac output is measured repeatedly (three to six times) by impedance cardiography (IC) and radionuclide ventriculography (RV).

The dataset is supplied with the MethComp package, and comes with the correct variable names, so it can immediately be transformed into a Meth object:

```
> data( cardiac )
> cardiac <- Meth( cardiac )

The following variables from the dataframe
"cardiac" are used as the Meth variables:
meth: meth
item: item
repl: repl
   y: y
      #Replicates
Method  3  4  5  6 #Items #Obs: 120 Values:  min   med  max
    IC  1  3  3  5     12        60         2.32 4.610 7.40
    RV  1  3  3  5     12        60         2.85 5.105 7.89
```

It is not clear from the description of the dataset whether replicates are linked across methods or not, but a quick check can be made graphically by making a Bland-Altman plot on the data as supplied and on the dat where replicates are randomly permuted, and then compare them as in figure 2.2.

```
> par( mfrow=c(1,2), mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot(           cardiac , limy=c(-3,3) )

Limits of agreement:
    RV - IC  2.5% limit 97.5% limit    SD(diff)
  0.6021667  -1.3199476   2.5242809   0.9610571

> BA.plot( perm.repl(cardiac), limy=c(-3,3) )

Limits of agreement:
    RV - IC  2.5% limit 97.5% limit    SD(diff)
  0.6021667  -1.3471230   2.5514563   0.9746448
```

A slightly more formal handle can be obtained by fitting models assuming constant difference between methods. The models are fitted, one with an item(=person) by replicate effect, and one without:

```
> BA.est( cardiac, linked=TRUE )

 Conversion between methods:
           alpha   beta     sd   LoA: lower  upper
To: From:
IC  IC     0.000  1.000  0.449       -0.898  0.898
    RV    -0.705  1.000  1.022       -2.748  1.339
RV  IC     0.705  1.000  1.022       -1.339  2.748
    RV     0.000  1.000  0.374       -0.749  0.749

 Variance components (sd):
     IxR   MxI   res
IC 0.193 0.661 0.317
RV 0.193 0.661 0.265
```

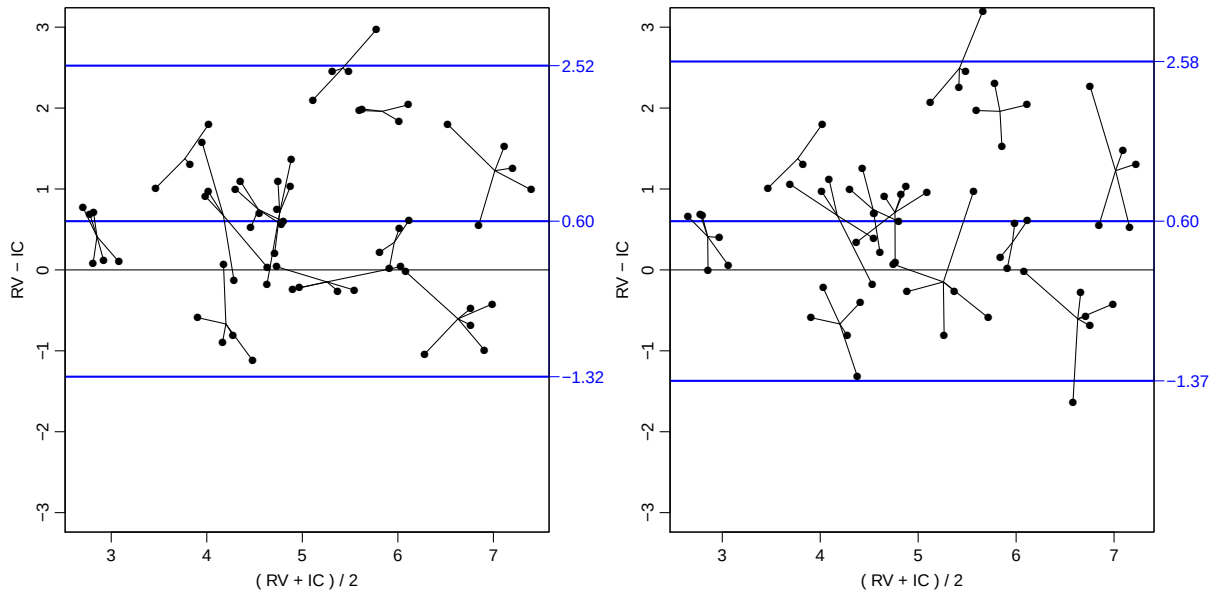

Figure 2.6: *Bland-Altman plots of the cardiac data. The left panel is the orginal data using replicate numbers to pair mesurements, the right is using a random permutation of replicates for the pairing. Even if replicates are claimed to be linked, the replicates the LoA in the right panel are not substantially wider.*

```
> BA.est( cardiac, linked=FALSE )

 Conversion between methods:
             alpha   beta     sd   LoA: lower  upper
To: From:
IC  IC       0.000  1.000  0.525        -1.050  1.050
    RV      -0.702  1.000  1.049        -2.801  1.396
RV  IC       0.702  1.000  1.049        -1.396  2.801
    RV       0.000  1.000  0.463        -0.926  0.926

 Variance components (sd):
   IxR   MxI   res
IC   0 0.654 0.371
RV   0 0.654 0.328
```

We see that there is a some variation between replicates, which we would not expect to see if replicates were exchangeable. In the model where we (erroneously) assume replicates to be exchangeable, we see that it is the residual variances that gets inflated. We can check the assumptions about constant bias and constant variance across the range of measurements by fitting a straight line to the differences as function of the averages (using the given linking of replicates). Note that the argument reg.line=3 gives printed output and graph annotation of the relationship between methods with three digits after the decimal point:

```
> BA.card <- BA.plot( cardiac, limy=c(-2,4), reg.line=3 )

Limits of agreement:
    RV - IC  2.5% limit 97.5% limit    SD(diff)
  0.6021667  -1.3199476   2.5242809   0.9610571
```

```
RV-IC = 0.422 + 0.036 (RV+IC)/2 (95% p.i.: +/-1.937)
 res.sd = 0.968    se(beta) = 0.103 , P = 0.7300

 IC = -0.415 + 0.965 RV (95% p.i.: +/-1.903)
 RV = 0.430 + 1.036 IC (95% p.i.: +/-1.972)
```

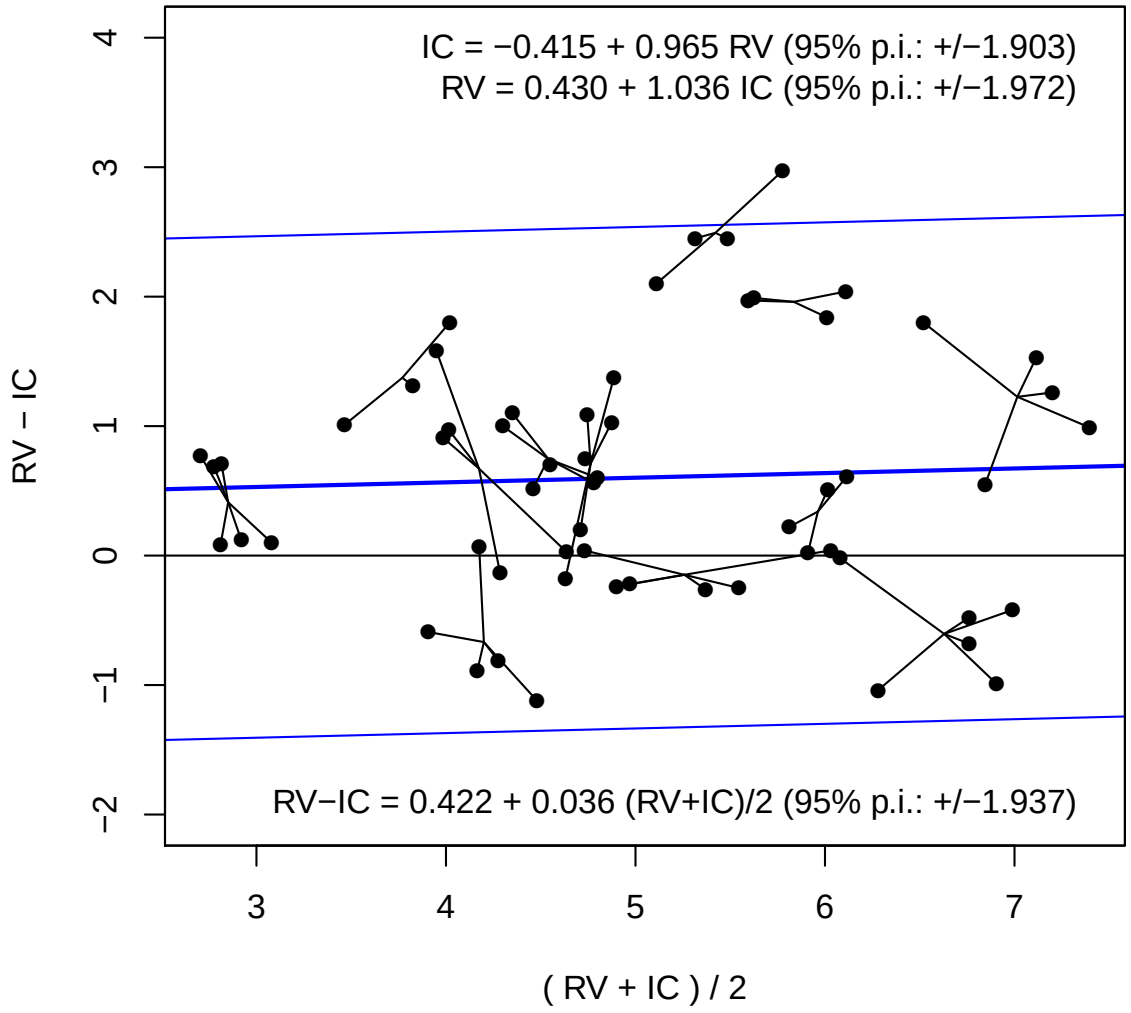

Figure 2.7: *Bland-Altman plot of the cardiac data with a fitted regression line.*

There is a some indication that the variance is not constant, but seen from the figure it does not seem alarming, it presumbaly hinges on the 6 points to the far left of the plot. An informal test of this can be obtained by using the function `DA.reg`, which regresses the Differences between methods on the Averages, and additionally regresses the

absolute values of the residuals from this analysis on the averages, so as to give an indication as to whether the residual standard deviation depends linearly on the mean:

```
> DA.reg( cardiac )
```

```
 Conversion between methods:
            alpha   beta sd.pred  beta=1  sd.|A=4.8  slope(sd)  sd.=K
To: From:
IC  IC      0.000  1.000      NA      NA         NA         NA     NA
    RV     -0.415  0.965   0.951   0.730      0.943      0.168  0.021
RV  IC      0.430  1.036   0.986   0.730      0.943      0.168  0.021
    RV      0.000  1.000      NA      NA         NA         NA     NA
```

If we fit a variance component model using `BA.est` as before, we can explore what effect it has on the repeatability (the prediction of a method from itself) if we include the variation between replicates or not:

```
> BA.est( cardiac, linked=TRUE, IxR.pr=FALSE )
```

```
 Conversion between methods:
            alpha   beta     sd   LoA: lower  upper
To: From:
IC  IC      0.000  1.000  0.449       -0.898  0.898
    RV     -0.705  1.000  1.022       -2.748  1.339
RV  IC      0.705  1.000  1.022       -1.339  2.748
    RV      0.000  1.000  0.374       -0.749  0.749

 Variance components (sd):
     IxR   MxI   res
IC 0.193 0.661 0.317
RV 0.193 0.661 0.265
```

```
> BA.est( cardiac, linked=TRUE, IxR.pr=TRUE )
```

```
 Conversion between methods:
            alpha   beta     sd   LoA: lower  upper
To: From:
IC  IC      0.000  1.000  0.525       -1.050  1.050
    RV     -0.705  1.000  1.022       -2.748  1.339
RV  IC      0.705  1.000  1.022       -1.339  2.748
    RV      0.000  1.000  0.463       -0.926  0.926

 Variance components (sd):
     IxR   MxI   res
IC 0.193 0.661 0.317
RV 0.193 0.661 0.265
```

The former is for the situation where we consider the variation between replicate measurements as a part of the repeatability conditions (even if the replicates are linked), the latter where we consider the variation between replicates to be irrelevant to the assessment of repeatability. However there is not much indication of linked estimates, since the other two variance components are virtually unchanged between the two analyses, and hence the predictions between methods based on the two approaches will be the same.

2.3 Systolic blood pressure: Linked replicates by two methods

We first load the systolic blood pressure data from the `MethComp` package.

```
> data( sbp )
> sbp <- Meth( sbp )

The following variables from the dataframe
"sbp" are used as the Meth variables:
meth: meth
item: item
repl: repl
   y: y
      #Replicates
Method          3 #Items #Obs: 765 Values:  min med max
     J         85     85       255           74 120 228
     R         85     85       255           76 120 226
     S         85     85       255           77 135 228

> str(sbp)

Classes 'Meth' and 'data.frame':        765 obs. of  4 variables:
 $ meth: Factor w/ 3 levels "J","R","S": 1 1 1 1 1 1 1 1 1 1 ...
 $ item: Factor w/ 85 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ repl: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ y   : num  100 108 76 108 124 122 116 114 100 108 ...

> plot( sbp )

Note:
 Replicate measurements are taken as separate items!
```

The resulting plot is shown in figure 2.8, clearly shows that the two manual measurements are in much closer agreement than any of them are with the automatic.

`plot.Meth` pairs replicates according to their numbering and treat them as separate items, so the plots fail to take the dependence of observations nto account.

We want to restrict our attention to the comparison of the two manual methods, but using the replicate measurements.

In this context it is important that we recognize whether the replicates are linked across the two methods or not. In this case they are, *i.e.* replicates are not exchangeable within methods and items.

```
> par( mar=c(3,3,3,3), mgp=c(3,1,0)/1.6 )
> sbp <- subset( sbp, meth %in% c("J","R") )
> str( sbp )

Classes 'Meth' and 'data.frame':        510 obs. of  4 variables:
 $ meth: Factor w/ 2 levels "J","R": 1 1 1 1 1 1 1 1 1 1 ...
 $ item: Factor w/ 85 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ repl: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ y   : num  100 108 76 108 124 122 116 114 100 108 ...

> BA.plot( sbp )

Limits of agreement:
      R - J  2.5% limit 97.5% limit    SD(diff)
-0.08627451 -4.60761840  4.43506938  2.26067194
```

A slightly more informative plot can be obtained by explicitly regulating the y-dimension of the plot by the argument `ymax=`:

```
> BA.plot( sbp, ymax=15 )

Limits of agreement:
      R - J  2.5% limit 97.5% limit    SD(diff)
-0.08627451 -4.60761840  4.43506938  2.26067194
```

The resulting plots are shown in figure 2.9.

In order to properly partition the variance and produce limits of agreement or a translation between the two observers, we should fit the relevant variance component model, assuming linked replicates:

$$y_{mir} = \alpha_m + \mu_i + a_{ir} + c_{mi} + e_{mir}, \quad a_{ir} \sim \mathcal{N}(0, \omega^2), \quad c_{mi} \sim \mathcal{N}(0, \tau_m^2), \quad e_{mir} \sim \mathcal{N}(0, \sigma_m^2)$$

Since we only have two methods, we cannot identify separate variance components τ_1 and τ_2, so we are forced to assume that $\tau_1 = \tau_2$, hence the use of `pdIdent` and not `pdDiag` in the specification of the matrix effects (*i.e.* the method by item interactions). The model above is fitted to the dataset by:

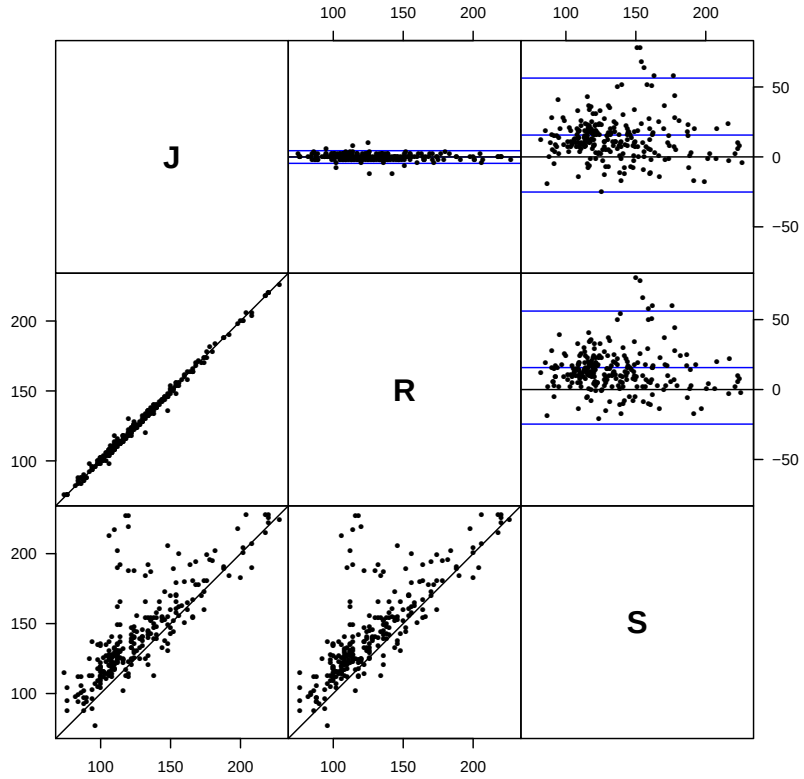

Figure 2.8: *Graphical overview of the* `sbp` *data. The methods J and R are two human observers, whereas method S is an automatic device.*

```
> m1 <- lme( y ~ meth + item,
+           random=list( item = pdIdent( ~ meth-1 ),
+                        repl = ~ 1 ),
+           weights = varIdent( form = ~1 | meth ),
+           data = sbp )
> m1

Linear mixed-effects model fit by REML
  Data: sbp
  Log-restricted-likelihood: -1163.807
  Fixed: y ~ meth + item
 (Intercept)        methR        item2        item3        item4        item5
103.47872449  -0.08627451   5.82189382 -22.17810618   1.89313629  13.45293925
       item6        item7        item8        item9       item10       item11
 25.82189382   5.82189382   7.96437876   2.92875753  -2.54706075   0.78627258
      item12       item13       item14       item15       item16       item17
 10.85751506   8.19084839   1.89313629   1.29771210  15.29771210  -2.10686371
      item18       item19       item20       item21       item22       item23
 14.63104543  33.29771210  43.29771210  53.36895457  40.17810618  66.03562124
      item24       item25       item26       item27       item28       item29
 60.48856049  39.22646963  27.22646963  37.59542419  45.22646963 115.89313629
      item30       item31       item32       item33       item34       item35
 95.66666667 -15.14248494  14.85751506  18.63104543  22.03562124  15.89313629
      item36       item37       item38       item39       item40       item41
-12.70228790   4.19084839 105.29771210  25.00000000  30.55980296  -9.07124247
      item42       item43       item44       item45       item46       item47
 -8.10686371  17.52418172  58.55980296  -2.17810618  24.26209086  11.22646963
      item48       item49       item50       item51       item52       item53
 31.08398468  49.22646963 -11.80915161  52.63104543  -1.44019704   1.15522715
      item54       item55       item56       item57       item58       item59
 -4.47581828 -24.17810618   1.59542419   5.45293925  75.45293925  52.92875753
      item60       item61       item62       item63       item64       item65
 35.96437876  93.52418172 -11.73790914  24.26209086  36.92875753  33.59542419
      item66       item67       item68       item69       item70       item71
```

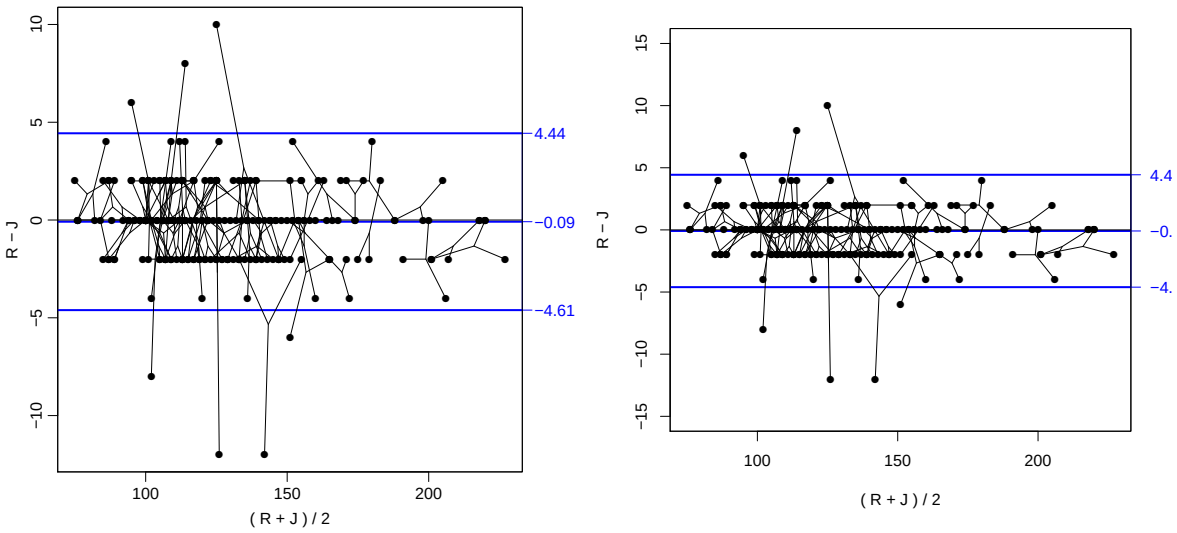

Figure 2.9: *Bland-Altman plot of the* sbp *data. Replicates are linked between methods, so the single replicates in the data has been used as single measurements when doing the Bland-Altman plot. Measurements from the same person are joined by thin lines. The only difference between the two plots is the scaling of the y-axis.*

```
 53.82189382  29.59542419   9.52418172  13.22646963  17.52418172 112.63104543
      item72       item73       item74       item75       item76       item77
 30.55980296  53.89313629 -19.44019704  70.48856049  75.59542419  13.22646963
      item78       item79       item80       item81       item82       item83
 15.29771210   4.55980296   6.26209086  36.78627258   4.78627258   6.92875753
      item84       item85
 -2.10686371  12.48856049

Random effects:
 Formula: ~meth - 1 | item
 Structure: Multiple of an Identity
            methJ     methR
StdDev: 0.2483701 0.2483701

 Formula: ~1 | repl %in% item
        (Intercept) Residual
StdDev:    5.932962  1.48587

Variance function:
 Structure: Different standard deviations per stratum
 Formula: ~1 | meth
 Parameter estimates:
       J        R
1.000000 1.122211
Number of Observations: 510
Number of Groups:
          item repl %in% item
            85            255
```

Now, the output from `lme` is pretty difficult to read, but the residual standard deviations are $\sigma_J = 1.485870$ and $\sigma_R = 1.485870 \times 1.122211 = 1.6674599$, whereas $\tau = 0.2483701$ (largely negligible) and $\omega = 5.932962$, by far the largest variance component. Also from the output we get the difference between methods R and J to be -0.08627451.

An easier way to get the relevant estimates is to use the wrapper `BA.est`, where the only necessary specification is the dataset (assuming that columns `meth`, `item`, `repl` and `y` are present) and whether replicates are linked across methods:

```
> BA.est( sbp, linked=TRUE )

 Conversion between methods:
            alpha   beta     sd   LoA: lower  upper
To: From:
J   J       0.000  1.000  2.101       -4.203  4.203
    R       0.086  1.000  2.261       -4.435  4.608
R   J      -0.086  1.000  2.261       -4.608  4.435
    R       0.000  1.000  2.358       -4.716  4.716

 Variance components (sd):
    IxR   MxI   res
J 5.933 0.248 1.486
R 5.933 0.248 1.667
```

Which is identical to the quantities we fished out of the `lme` output. Actually `BA.est` fits exactly the model we fitted, and then extracts the quantities that we are interested in.

The limits of agreement between the two manual observers is then for R$-$J $-0.0863 \pm 1.96 \times \sqrt{2 \times 0.248^2 + 1.486^2 + 1.667^2} = (-4.51, 4.34)$, i.e. on average they agree, but in order to be sure to enclose 95% of all differences we need an interval approximately as 0 ± 4.5 mmHg.

One way of seeing the lack of exchangeability is to make the overview plot using a random permuation of the replicates. If replicates were truely exchangeable within methods the plot would look similar when permuting the replicates — and it does not!

For completeness we reload the data to get observations by all three methods included, and then make overview plots after random permutation of replicates within (method,item):

```
> data(sbp)
> sbp <- Meth( sbp )

The following variables from the dataframe
"sbp" are used as the Meth variables:
meth: meth
item: item
repl: repl
   y: y
       #Replicates
Method          3 #Items #Obs: 765 Values:  min med max
     J         85     85       255           74 120 228
     R         85     85       255           76 120 226
     S         85     85       255           77 135 228

> str(sbp)

Classes 'Meth' and 'data.frame':        765 obs. of  4 variables:
 $ meth: Factor w/ 3 levels "J","R","S": 1 1 1 1 1 1 1 1 1 1 ...
 $ item: Factor w/ 85 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ repl: Factor w/ 3 levels "1","2","3": 1 1 1 1 1 1 1 1 1 1 ...
 $ y   : num  100 108 76 108 124 122 116 114 100 108 ...

> plot( perm.repl(sbp) )

Note:
 Replicate measurements are taken as separate items!
```

The two resulting plots are shown in figure 2.10.

The analysis should be based on a model where a random item by replicate effect is included to accomodate the linking of replicates:

```
> BA.est( sbp, linked=TRUE )

 Conversion between methods:
             alpha    beta      sd   LoA: lower   upper
To: From:
J   J        0.000   1.000   2.305       -4.610   4.610
    R        0.086   1.000   2.272       -4.459   4.631
    S      -15.620   1.000  20.326      -56.272  25.032
R   J       -0.086   1.000   2.272       -4.631   4.459
    R        0.000   1.000   2.187       -4.375   4.375
    S      -15.706   1.000  20.317      -56.339  24.927
S   J       15.620   1.000  20.326      -25.032  56.272
    R       15.706   1.000  20.317      -24.927  56.339
    S        0.000   1.000  12.930      -25.860  25.860

 Variance components (sd):
    IxR    MxI   res
J 5.887  0.338 1.630
R 5.887  0.001 1.547
S 5.887 18.077 9.143
```

The substantial item by replicate interaction (`IR`) clearly indicates that replicates are linked between methods.

The resulting estimates from this model gives limits of agreement for R−J based on the method by item and the residual variances:

$$-0.0863 \pm 1.96 \times \sqrt{0.3385^2 + 0.0011^2 + 1.6301^2 + 1.5467^2} = -0.0863 \pm 4.4540 = (-4.54, 4.37)$$

which is in agreement with the limits computed based on the simplistic way of taking replicates as items — a procedure wich is actually close to correct if replicates are linked.

Alternatively this could be formulated as a 95% prediction interval for R given a measurement by J, y_{J}, which would be

$$y_{\mathrm{R}}|y_{\mathrm{J}} = y_{\mathrm{J}} - 0.0863 \pm 4.4540 = y_{\mathrm{J}} + (-4.54; 4.37)$$

The above analysis is based on the correct analysis of the entire dataset, including the information from the machine measurement S. If we fit the model on the restricted dataset, we of course get a common method by item interaction term because we then only have two methods:

```
> BA.est( subset( sbp, meth!="S" ), linked=TRUE )

 Conversion between methods:
            alpha   beta     sd   LoA: lower  upper
To: From:
J   J       0.000  1.000  2.101       -4.203  4.203
    R       0.086  1.000  2.261       -4.435  4.608
R   J      -0.086  1.000  2.261       -4.608  4.435
    R       0.000  1.000  2.358       -4.716  4.716

 Variance components (sd):
```

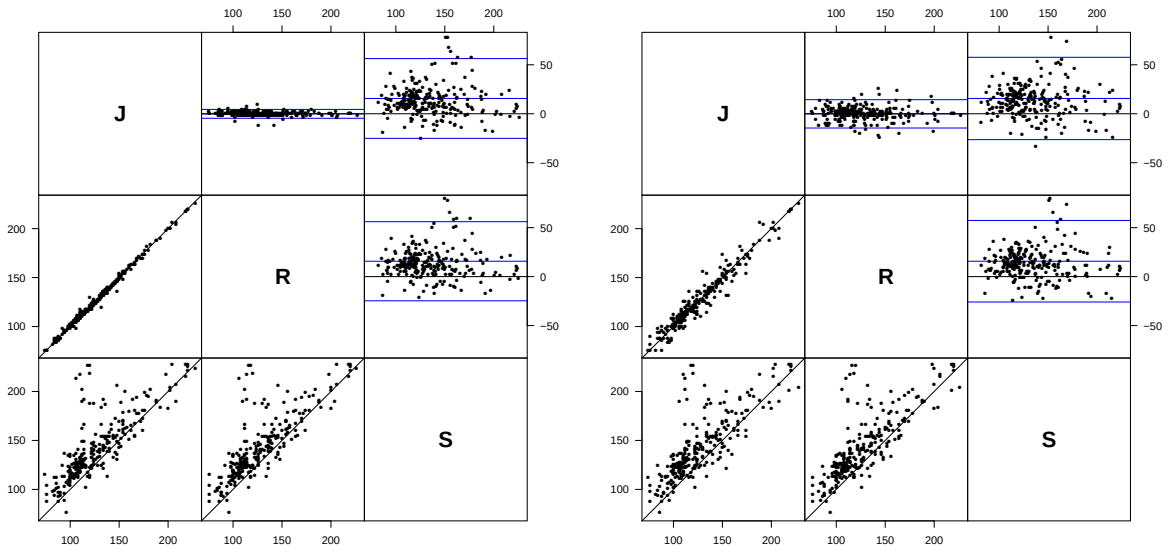

Figure 2.10: *Graphical overview of the* `sbp` *data; the left panel with the original replicate numbers used for matching; the other with replicates permuted randomly within methods.*

```
    IxR   MxI   res
J 5.933 0.248 1.486
R 5.933 0.248 1.667
```

Based on these estimates we get the limits of agreement for R−J to be:

$$-0.0863 \pm 1.96 \times \sqrt{2 \times 0.2484^2 + 1.4859^2 + 1.6674^2} = 0.0863 \pm 4.4313 = (-4.52, 4.35)$$

i.e. effectively the same as before, based on all three methods. Again these limits are those computed by `BA.est`.

Bibliography

Chapter 3

MethComp manual

Version 1.7

Date 2011-05-06

Title Functions for analysis of method comparison studies.

Author Bendix Carstensen, Lyle Gurrin.

Maintainer Bendix Carstensen <bxc@steno.dk>

Depends R (>= 2.0.0), nlme

Suggests R2WinBUGS, coda, BRugs, lattice

Description
Methods (standard and advanced) for comparison of measurement methods.

License GPL (>= 2)

URL http://www.pubhealth.ku.dk/~bxc/MethComp/

---

abconv	*Derive linear conversion coefficients from a set of indeterminate coefficients*

Description

If a method comparison model is defined as $y_{mi} = \alpha_m + \beta_m \mu_i, m = 1, 2$ y_mi = alpha_m + beta_m*mu_i, m=1,2 the coefficients of the linear conversion from method 1 to 2 are computed as: $\alpha_{2|1} = -\alpha_2 - \alpha_1\beta_2/\beta_1$ alpha_(2|1) = -alpha_2-alpha_1*beta_2/beta_1 $\beta_{2|1} = \beta_2/\beta_1$ Morover the the point where the linear conversion function intersects the identity line is computed too.. The function is designed to work on numerical vectors of posterior samples from BUGS output.

Usage

```
abconv( a1, b1 = 1:4, a2 = NULL, b2 = NULL,
        col.names = c("alpha.2.1", "beta.2.1", "id.2.1") )
```

Arguments

`a1`	Numerical vector of intercepts for first method. Alternatively a dataframe where the vectors are selected from.
`b1`	Numerical vector of slopes for first method. If `a1` is a dataframe, `b1` is assumed to be a numerical vector of length 4 pointing to the columns of `a1` with the intercepts and slopes.
`a2`	Numerical vector of intercepts for second method.
`b2`	Numerical vector of slopes for second method.
`col.names`	Names for the resulting three vectors.

Value

A dataframe with three columns: intercept and slope for the conversion from method 1 to method 2, and the value where the conversion is the identity.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc

References

B Carstensen: Comparing and predicting between several methods of measurement, Biostatistics, 5, pp 399-413, 2004

See Also

`BA.plot`, `MCmcmc`

Examples

```
abconv( 0.3, 0.9, 0.8, 0.8 )
```

`AltReg` *Estimate in a method comparison model with replicates*

Description

Estimates in the general model for method comparison studies with replicate measurements by each method, allowing for a linear relationship between methods, using the method of alternating regressions.

Usage

```
AltReg( data,
      linked = FALSE,
         IxR = linked,
         MxI = TRUE,
      varMxI = FALSE,
         eps = 0.001,
     maxiter = 50,
       trace = FALSE,
      sd.lim = 0.01,
   Transform = NULL,
   trans.tol = 1e-6 )
```

Arguments

`data`	Data frame with the data in long format, (or a `Meth` object) i.e. it must have columns `meth`, `item`, `repl` and `y`
`linked`	Logical. Are the replicates linked across methods? If true, a random `item` by `repl` is included in the model, otherwise not.
`IxR`	Logical, alias for linked.
`MxI`	Logical, should the method by item effect (matrix effect) be in the model?
`varMxI`	Logical, should the method by item effect have method-specific variances. Ignored if only two methods are compared. See details.
`eps`	Convergence criterion, the test is the max of the relative change since last iteration in both mean and variance parameters.
`maxiter`	Maximal number of iterations.
`trace`	Should a trace of the iterations be printed? If `TRUE` iteration number, convergence criterion and current estimates of means and sds are printed.

`sd.lim` Estimated standard deviations below `sd.lim` are disregarded in the evaluation of convergence. See details.

`Transform` A character string, or a list of two functions, each other's inverse. The measurements are transformed by this before analysis. Possibilities are: "exp", "log", "logit", "pctlogit" (transforms percentages by the logit), "sqrt", "sq" (square), "cll" (complementary log-minus-log), "ll" (log-minus-log). For further details see `choose.trans`.

`trans.tol` The tolerance used to check whether the supplied transformation and its inverse combine to the identity. Only used if `Transform` is a list of two functions.

Details

When fitting a model with both IxR and MxI interactions it may become very unstable to have different variances of the MxI random effects for each method, and hence the default option is to have a constant MxI variance across methods. On the other hand it may be grossly inadequate to assume these variances to be identical.

If only two methods are compared, it is not possible to separate different variances of the MxI effect, and hence the `varMxI` is ignored in this case.

The model fitted is formulated as:

$$y_{mir} = \alpha_m + \beta_m(\mu_i + a_{ir} + c_{mi}) + e_{mir}$$

and the relevant parameters to report are the estimates sds of a_{ir} and c_{mi} multiplied with the corresonidng β_m. Therefore, different values of the variances for MxI and IxR are reported also when `varMxI==FALSE`. Note that `varMxI==FALSE` is the default and that this is the opposite of the default in `BA.est`.

Value

An object of class `c("MethComp","AltReg")`, which is a list with three elements:

`Conv` A 3-way array with the 2 first dimensions named "To:" and "From:", with methods as levels. The third dimension is classifed by the linear parameters "alpha", "beta", and "sd".

`VarComp` A matrix with methods as rows and variance components as columns. Entries are the estimated standard deviations.

`data` The original data used in the analysis, with untransformed measurements (`ys`). This is needed for plotting purposes.

Moreover, if a transformation was applied before analysis, an attribute "Transform" is present; a list with two elements `trans` and `inv`, both of which are functions, the first the transform, the last the inverse.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <bxc@steno.dk>,
http://www.biostat.ku.dk/~bxc.

References

B Carstensen: Comparing and predicting between several methods of measurement. Biostatistics (2004), 5, 3, pp. 399–413.

See Also

BA.est, DA.reg, Meth.sim, MethComp

Examples

```
data( ox )
ox <- Meth( ox )
ox.AR <- AltReg( ox, linked=TRUE, trace=TRUE, Transform="pctlogit" )
str( ox.AR )
ox.AR
# plot the resulting conversion between methods
plot(ox.AR,pl.type="conv",axlim=c(20,100),points=TRUE,xaxs="i",yaxs="i",pch=16)
# - or the rotated plot
plot(ox.AR,pl.type="BA",axlim=c(20,100),points=TRUE,xaxs="i",yaxs="i",pch=16)
```

BA.est *Bias and variance components for a Bland-Altman plot.*

Description

A variance component model is fitted to method comparison data with replicate measurements in each method by item stratum. The purpose is to simplify the construction of a correct Bland-Altman-plot when replicate measurements are available, and to give the REML-estimates of the relevant variance components.

Usage

```
BA.est( data, linked=TRUE, IxR=has.repl(data),
                           MxI=has.repl(data),
                      varMxI=TRUE,
                      IxR.pr=FALSE,
                        bias=TRUE, alpha=0.05,
                Transform = NULL,
                trans.tol = 1e-6 )
```

```
  ## S3 method for class 'BA.est'
bias( obj, ref=1, ... )
  VC.est( data,
           IxR = has.repl(data), linked = IxR,
           MxI = has.repl(data), matrix = MxI,
         varMxI = TRUE, bias = TRUE, print = FALSE )
```

Arguments

`data`	A `Meth` object representing method comparison data with replicate measurements, i.e. with columns `meth`, `item`, `repl` and `y`.
`linked`	Logical. Are replicates linked within item across methods?
`IxR`	Logical. Should an item by repl interaction be included in the model. This is needed when the replicates are linked within item across methods, so it is just another name for the `linked` argument. If `linked=` is gven, this is ignored.
`MxI`	Logical. Should the method by item interaction (matrix effect) be included in the model.
`matrix`	Logical. Alias for `MxI`.
`varMxI`	Logical. Should the method by item interaction have a variance that varies between methods. Ignored if only two methods are compared.
`IxR.pr`	Logical. Should the item by repl interaction variation be included in the prediction standard deviation?
`bias`	Logical. Should a systematic bias between methods be estimated? If `FALSE` no bias between methods are assumed, i.e. $\alpha_m = 0, m = 1, \ldots M$.
`alpha`	Numerical. Significance level. By default the value 2 is used when computing prediction intervals, otherwise the $1 - \alpha/2$ t-quantile is used. The number of d.f. is taken as the number of units minus the number of items minus the number of methods minus 1 ($I - M - 1$).
`Transform`	Transformation applied to data (`y`) before analysis. See `check.trans` for possible values.
`trans.tol`	Numerical. The tolerance used to check whether the supplied transformation and its inverse combine to the identity.
`obj`	A `BA.est` object from which to extract the biases between methods.
`ref`	Numeric or character. The reference method for the biases: the method with bias 0.
`print`	Logical. Should the estimated bias and variance components be printed?
`...`	Further argumenst passed on. Curently ignored.

Details

The model fitted is:

$$y = \alpha_m + \mu_i + c_{mi} + a_{ir} + e_{mir}, \quad \text{var}(c_{mi}) = \tau_m^2, \quad \text{var}(a_{ir}) = \omega^2, \quad \text{var}(e_{mir}) = \sigma_m^2,$$

We can only fit separate variances for the τs if more than two methods are compared (i.e. `nM` > 2), hence varMxI is ignored when `nM`==2.

The function `VC.est` is the workhorse; `BA.est` just calls it. `VC.est` figures out which model to fit by `lme`, extracts results and returns estimates. `VC.est` is also used as part of the fitting algorithm in `AltReg`, where each iteration step requires fit of this model.

Value

`BA.est` returns an object of class `c("MethComp","BA.est")`, a list with four elements `Conv`, `VarComp`, `LoA`, `RepCoef`; `VC.est` returns (invisibly!) a list with elements `Bias`, `VarComp`, `Mu`, `RanEff`. These list components are:

`Conv` 3-dimensional array with dimensions "To", "From" and unnamed. The first two dimensions have the methods compared as levels, that last one `c("alpha","beta","sd","LoA: lower","upper")`. It represents the mean conversions between methods and the prediciton standard deviation.

Where "To" and "From" take the same value the value of the "sd" component is $\sqrt{2}$ times the residual variation for the method. If `IxR.pr=TRUE` the variation between replicates are included too, i.e. $\sqrt{2(\sigma_m^2 + \omega^2)}$ sqrt[2(sigma_mˆ2+omegaˆ2)].

`VarComp` A matrix of variance components (on the SD scale) with methods as rows and variance components "IxR", "MxI" and "res" as columns.

`LoA` Four-column matrix with mean difference, lower and upper limit of agreement and prediction SD. Each row in the matrix represents a pair of methods.

`RepCoef` Two-column matrix of repeatability SDs and repeatability coefficients. The SDs are the standard deviation of the difference between two measurements by the same method on the item under identical circumstances; the repeatability coefficient the numerical extent of the prediction interval for this difference, i.e. $2\sqrt{2}$

times the sd.

`Mu` Estimates of the item-specific parameters.

`RanEff` Estimates of the random effects from the model (BLUPS). This is a (possibly empty) list with possible elements named `MxI` and `IxR` according to whether these random effects are in the model.

The returned object has an attribute, `Transform` with the transformation applied to data before analysis, and its inverse — see `choose.trans`.

Author(s)

Bendix Carstensen

References

Carstensen, Simpson & Gurrin: Statistical models for assessing agreement in method comparison studies with replicate measurements, The International Journal of Biostatistics: Vol. 4 : Iss. 1, Article 16.
http://www.bepress.com/ijb/vol4/iss1/16.

See Also

`BA.plot`, `perm.repl`

Examples

```
data( ox )
ox <- Meth( ox )
summary( ox )
BA.est( ox )
BA.est( ox, linked=FALSE )
BA.est( ox, linked=TRUE, Transform="pctlogit" )
data( sbp )
BA.est( sbp )
BA.est( sbp, linked=FALSE )
# Check what you get from VC.est
str( VC.est( sbp ) )
```

`BlandAltman` *Bland-Altman plot of differences versus averages.*

Description

For two vectors of equal length representing measurements of the same quantity by two different methods, the differences are plotted versus the average. The limits of agreement (prediction limits for the differences) are plotted, optionally a regression of differences of means is given too.

Usage

```
BlandAltman(x, y,
          x.name = NULL,
          y.name = NULL,
         maintit = "",
             cex = 1,
             pch = 16,
       col.points = "black",
        col.lines = "blue",
             limx = NULL,
             limy = NULL,
             ymax = NULL,
             eqax = FALSE,
             xlab = NULL,
             ylab = NULL,
            print = TRUE,
         reg.line = FALSE,
           digits = 2,
             mult = FALSE,
            alpha,
              ... )
BA.plot( y1, y2,
    meth.names = NULL,
     mean.repl = FALSE,
     conn.repl = !mean.repl,
      lwd.conn = 1,
      col.conn = "black",
   comp.levels = 2:1,
             ... )
```

Arguments

`x`	Numerical vector of measurements by 1st method.
`y`	Numerical vector of measurements by 2nd method. Must of same length as `x`.
`x.name`	Label for the 1st method (`x`).
`y.name`	Label for the 2nd method (`y`).
`maintit`	Main title for the plot
`cex`	Character expansion for the points.
`pch`	Plot symbol for points.
`col.points`	Color for the points.
`col.lines`	Color for the lines indicating limits of agreement.

`limx`	x-axis limits.
`limy`	y-axis limits.
`ymax`	Scalar. The y-axis will extend from -ymax to +ymax.
`eqax`	Logical. Should the range on x- and y- axes be the same?
`xlab`	x-axis label.
`ylab`	y-axis label.
`print`	Logical: Should the limits of agreement and the c.i.s of these be printed?
`reg.line`	If `TRUE`, the regression line of `x-y` on `(x+y)/2` is drawn. If numerical the regression equation is printed with the given number of digits after the decimal points.
`digits`	How many decimal places should be used when printing limits of agreement? Used both for the printing of results and for annotation of the plot.
`mult`	Logical. Should data be log-transformed and reporting be on a multiplicative scale?
`alpha`	1 minus confidence level used when computing confidence intervals and limits of agreement, i.e. the t(1-alpha/2) quantile is used. If not supplied the standard value of 2 is used for computing LoA.
`y1`	Measurements by method 1. Alternatively a `Meth` object or a dataframe with columns `meth`, `item`, `y`, and possibly `repl`.
`y2`	Corresponding measurements by method 2. Ignored if `y1` is a dataframe.
`meth.names`	Names for the two methods. Used for annotation of the plot. If not supplied and `y1` is a dataframe names are derived from the factor level names of `meth`.
`mean.repl`	Logical. If there are replicate measurements by each method should the means by `item` and `meth` be formed before further ado. WARNING: This will give too narrow limits of agreement.
`conn.repl`	Logical. Should replicates from the same item be connected?
`lwd.conn`	Line width of connecting lines
`col.conn`	Color of connecting lines
`comp.levels`	Levels of the `meth` factor to compare. May be used to switch the order of the methods compared by specifying `comp.meth=2:1`.
`...`	Further arguments passed on from `BA.plot` to `BlandAltman` and possibly further to the `plot` function. The arguments passed to `BlandAltman` are used for fine-tuning the appearance of the plot.

Value

An object of class `BA.check`; list with 3 elements:

`LoA`	A vector of length 3 with Limits of Agreement.
`p.value`	P-values for three hypothese: 1) Constant variance - this is the test of 0 slope in the regression of absolute residuals on averages. 2) Constant difference - this is the test of 0 slop in the regression of differences on averages. 3) Difference equal to 0 - this is usually a lame thing to use.
`reg.res`	A 3×4 matrix with (in the first row) the results from regressing the averages on the means, and in the two other rows the derived relationships between methods. In each line the intercept (`alpha`), slope (`beta`), the prediction standard deviation (`pr.sd`) and half the width of the prdiction interval (`pr.int`).

Author(s)

Bendix Carstensen `<bxc@steno.dk>`, http://www.biostat.ku.dk/~bxc.

References

JM Bland and DG Altman: Statistical methods for assessing agreement between two methods of clinical measurement, Lancet, i, 1986, pp. 307-310.

JM Bland and DG Altman. Measuring agreement in method comparison studies. Statistical Methods in Medical Research, 8:136-160, 1999.

B Carstensen: Comparing methods of measurement: Extending the LoA by regression. Stat Med. 2010 Feb 10;29(3):401-10.

See Also

`BA.plot`, `MCmcmc`.

Examples

```
data( ox )
par( mfrow=c(1,2) )
# Wrong to use mean over replicates
mtab <- with( ox, tapply( y, list(item, meth), mean ) )
CO <- mtab[,"CO"]
pulse <- mtab[,"pulse"]
BlandAltman( CO, pulse )

# (almost) Right to use replicates singly
par( mfrow=c(1,1) )
oxw <- to.wide( ox )
```

```
CO <- oxw[,"CO"]
pulse <- oxw[,"pulse"]
BlandAltman( CO, pulse, mult=TRUE )
BlandAltman( CO, pulse, eqax=TRUE )

data( plvol )
BA.plot( plvol )
BA.plot( plvol, reg.line=TRUE )
BA.plot( plvol, reg.line=2 )
```

`bothlines` *Add regression lines to a plot*

Description

Add the regression lines of y on x AND x on y to the plot. Optionally add the line obtained by allowing errors in both variables (Deming regression).

Usage

```
bothlines(x, y, Dem = FALSE, sdr = 1, col = "black", ...)
```

Arguments

`x`	Numeric vector
`y`	Numeric vector
`Dem`	Logical. Should the Deming regression line be added too?
`sdr`	Numeric. The assumed ratio of standard deviations used in the Deming regression.
`col`	Colour of the lines. Can be a vector of up to 3 elements, one for each line.
`...`	Additional arguments passed on to `abline`, which does the actual plotting.

Value

None.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc

See Also

abline.

Examples

```
data( ox )
oxw <- to.wide(ox)
attach( oxw )
plot( CO, pulse )
abline(0,1)
bothlines( CO, pulse, Dem=TRUE, col=rainbow(3), lwd=2 )
plot( CO, pulse,pch=16 )
abline(0,1, col=gray(0.7), lwd=2)
bothlines( CO, pulse, Dem=TRUE, col=c(rep("transparent",2),"black"), lwd=2 )
```

---

cardiac *Measurement of cardiac output by two different methods.*

Description

For each subject cardiac output is measured repeatedly (three to six times) by impedance cardiography (IC) and radionuclide ventriculography (RV).

Usage

```
data(cardiac)
```

Format

A data frame with 120 observations on the following 4 variables.

`meth` a factor with levels `IC` `RV`

`item` a numeric vector giving the item number.

`repl` a numeric vector with replicate number.

`y` the measuremnts of cardiac output.

Details

It is not entirely clear from the source whether the replicates are exchangeable within (method,item) or whether they represent pairs of measurements. From the description it looks as if replicates are linked between methods, but in the paper they are treated as if they were not.

Source

The dataset is adapted from table 4 in: JM Bland and DG Altman: Measuring agreement in method comparison studies. Statistical Methods in Medical Research, 8:136-160, 1999. Originally supplied to Bland \& Altman by Dr LS Bowling, see: Bowling LS, Sageman WS, O'Connor SM, Cole R, Amundson DE. Lack of agreement between measurement of ejection fraction by impedance cardiography versus radionuclide ventriculography. Critical Care Medicine 1993; 21: 1523-27.

Examples

```
data(cardiac)
cardiac <- Meth(cardiac)
summary(cardiac)
# Visually check exchangeability
plot( cardiac )
plot( perm.repl( cardiac ) )
BA.est(cardiac)
# Run MCmcmc using BRugs for an insufficient amount of iterations
## Not run: card.mi.ir <- MCmcmc( cardiac,
                                  beta=FALSE, random=c("mi","ir"),
                                  n.iter=100, trace=T )
print( card.mi.ir )
## End(Not run)
```

`CardOutput` *Measurements of Cardiac output.*

Description

Two different ways of measuring cardiac output and oxygen saturation in 15 critically ill persons.

Usage

```
data(CardOutput)
```

Format

A data frame with 15 observations on the following 8 variables.

`Age` Patient age

`Diag` Diagnosis, a factor with levels `sepsis`, `cardiogenic`, `hypothermia`

`VO2` Oxygen consumption

`Svo2` Mixed venous O2 saturation

`Scvo2` Central venous oxygen saturation

`TCO` Thermodilution-derived cardiac output

`FCO` Fick-derived cardiac output.

`Sex` Sex, a factor with levels `F`, `M`

Source

Avi A. Weinbroum, Philippe Biderman, Dror Soffer, Joseph M. Klausner & Oded Szold:

Reliability of cardiac output calculation by the fick principle and central venous oxygen saturation in emergency conditions.

Journal of Clinical Monitoring and Computing (2008) 22: 361-366

Examples

```
data(CardOutput)
```

`check.MCmcmc` *Functions to graphically assess the convergence of the MCMC-simulation in a MCmcmc object*

Description

These functions display traces, posterior densities and autocorrelation functions for the relevant subset of the parameters in a MCmcmc object.

Usage

```
## S3 method for class 'MCmcmc'
trace( obj, what = "sd",
                            scales = c("same", "free"),
                            layout = "col",
                            aspect = "fill", ...)
```

```
## S3 method for class 'MCmcmc'
post( obj, what ="sd",
                             check = TRUE,
                           scales = "same",
                           layout = "row",
```

```
                              lwd = 2,
                              col,
                      plot.points = FALSE,
                           aspect = "fill", ... )
## S3 method for class 'MCmcmc'
pairs( x, what = "sd",
                            subset,
                               col = NULL,
                               pch = 16,
                               cex = 0.2,
                            scales = "free", ... )
```

Arguments

`obj`	A `MCmcmc` object.
`x`	A `MCmcmc` object.
`what`	Character indicating what parameters to plot. Possible values are `"sd"` or `"var"` which gives plots for the variance components (on the sd. scale), `"beta"` or `"slope"`, which gives plots for slope parameters and `"alpha"` or `"int"`, which gives plots for the intercept parameters.
`scales`	Character vector of length two, with possible values "same" or "free", indicating whether x- and y-axes of the plots should be constrained to be the same across panels. For `pairs` only the first element is used to decide whether all panles should have the same axes.
`layout`	Character. If `"col"` parameters are displayed columnwise by method, if `"row"` they are displayed row-wise.
`aspect`	How should the panels be scaled. Default (`"fill"`) is to make a panels take up as much place as possible.
`check`	Logical. Should the density plots be separate for each chain (in order to check convergence) or should the chains be merged.
`lwd`	Width of the lines used for plotting of the posterior densities.
`col`	Color of the lines points used for plotting of the posterior densities.
`plot.points`	Logical. Should a rug with actual data points be plotted beneath the density.
`pch`	Plot symbol for the points.
`subset`	Character or numerical indicating the columns of the posterior that should be plotted by `pairs`.
`cex`	Plot character size for points in `pairs`.
`...`	Further aruments passed on to the `Lattice` function called: `trace` calls `xyplot` from the `coda` package, `post` calls `densityplot` from the `coda` package, `pairs` calls `pairs` from the `graphics` package.

Details

A `Lattice` plot is returned, which means that it must `print`ed when these functions are called in a batch program or inside another function or for-loop.

`trace` plots traces of the sampled chains, `post` plots posterior densities of the parameters and `pairs` plots a scatter-plot matrix of bivariate marginal posterior distributions.

Value

A `Lattice` plot.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <bxc@steno.dk>,
http://www.biostat.ku.dk/~bxc.

See Also

`MCmcmc`, `plot.MCmcmc`, `ox.MC`, `sbp.MC`

Examples

```
# Load a provided MCmcmc object
data( ox.MC )
trace.MCmcmc( ox.MC, what="beta" )
pairs.MCmcmc( ox.MC, what="sd" )
```

`choose.trans` *Functions to handle transformations of measuremnt results.*

Description

Choose a function and inverse based on a text string; check whether two functions actually are each others inverse.

Usage

```
choose.trans( tr )
check.trans( trans, y, trans.tol = 1e-05 )
```

Arguments

`tr`	A character string, or a list of two functions, they should be each other's inverse. Names of the list are ignored.
`trans`	A list of two functions, each other's inverse.
`y`	Vector of numerical values where the functions should be each other's inverse.
`trans.tol`	Numerical constant indication how precise the evaulation should be.

Value

`choose.trans` returns a named list with two elements "trans" and "inv", both functions which are each other's inverse. This is intended to be stored as an attribute `"Transform"` with the resulting object and used in plotting and reporting. All results will be on the transformed scale. If the `tr` argument to `choose.trans` is a character constant, the appropriate named list of two functions will be generated. Possibilities are: "exp", "log", "logit", "pctlogit" (transforms percentages by the logit), "sqrt", "sq" (square), "cll" (complementary log-minus-log), "ll" (log-minus-log). If there is no match `NULL` is returned, which will correspond to no transformation.

`check.trans` returns nothing.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc.

Examples

```
choose.trans( "logit" )
```

`corr.measures` *Correlation measures for method comparison studies. Please don't use them!*

Description

Computes correlation, mean squared difference, concordance correlation coefficient and the association coefficient. `middle` and `ends` are useful utilities for illustrating the shortcomings of the association measures, see the example.

Usage

```
corr.measures(x, y)
middle(w, rm = 1/3)
  ends(w, rm = 1/3)
```

Arguments

x	vector of measurements by one method.
y	vector of measurements by another method.
w	numerical vector.
rm	fraction of data to remove.

Details

These measures are all flawed since they are based on the correlation in various guises. They fail to address the relevant problem of AGREEMENT. It is recommended NOT to use them. The example gives an example, illustrating what happens when increasingly large chunks of data in the middle are removed.

Value

`corr.measures` return a vector with 4 elements. `middle` and `ends` return a logical vector pointing to the middle or the ends of the `w` after removing a fraction of `rm` from data.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc

References

Shortly...

See Also

`MCmcmc`.

Examples

```
cbind( zz <- 1:15, middle(zz), ends(zz) )
data( sbp )
bp <- subset( sbp, repl==1 & meth!="J" )
bp <- Meth( bp )
summary( bp )
plot( bp )
bw <- to.wide( bp )
with( bw, corr.measures( R, S ) )
# See how it gets better with less and less data:
summ.corr <-
rbind(
with( subset( bw, middle( R+S, 0.6 ) ), corr.measures( R, S ) ),
with( subset( bw, middle( R+S, 0.4 ) ), corr.measures( R, S ) ),
```

```
with(         bw                       , corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.3 ) ), corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.4 ) ), corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.6 ) ), corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.8 ) ), corr.measures( R, S ) ) )
rownames( summ.corr ) <- c("middle 40%",
                           "middle 60%",
                           "total",
                           "outer 70%",
                           "outer 60%",
                           "outer 40%",
                           "outer 20%")
summ.corr
```

`DA.reg` *Make a regresion of differeneces on averages*

Description

For each pair of methods in `data`, a regression of the differences on the averages between methods is made and a linear relationship between methods with prediction standard deviations is derived.

Usage

```
DA.reg(data,
  Transform = NULL,
  trans.tol = 1e-6)
```

Arguments

`data`	A `Meth` object. May also be a data frame with columns `meth`, `item` and `y`.
`Transform`	A character string, or a list of two functions, each other's inverse. The measurements are transformed by this before analysis. Possibilities are: "exp", "log", "logit", "pctlogit" (transforms percentages by the logit), "sqrt", "sq" (square), "cll" (complementary log-minus-log), "ll" (log-minus-log). For further details see `choose.trans`.
`trans.tol`	The tolerance used to check whether the supplied transformation and its inverse combine to the identity. Only used if `Transform` is a list of two functions.

Details

If the input object contains replicate measurements these are taken as separate items in the order they appear in the dataset.

Value

A `MethComp` object, i.e. a list with three components, `Conv`, `VarComp`, and `data`. `Conv` is a three-dimensional array, with dimensions `To`, `From` (both with levels equal to the methods in `data`) and an unnamed dimension with levels `"alpha"`, `"beta"`, `"sd.pred"`, `"beta=1"` and `"s.d.=K"`. Converstng from method l to method k using

$$y_{k|l} = \alpha + \beta y_l$$

with prediction standard deviation σ, just requires the entries `[k,l,c("alpha","beta","sd.pred"]`. The two last entries are p-values for the hypotheses: 1) $\beta = 1$ and 2) standard errors are constant over the range. The latter is derived by regressiin the absoulte values of the residuals on the averages.

The `VarComp` element of the list is `NULL`, and only present for compatibility with the print method for `MethComp` objects.

The `data` element is the input datframe. The mesurements iny are left un-transformed.

Author(s)

Bendix Carstensen, Steno Diabetes Center, `bxc$steno.dk`

References

B Carstensen: Limits of agreement: How to use the regression of differences on averages. Technical Report 08.6, Department of Biostatistics, University of Copenhagen, http://www.pubhealth.ku.dk/bs/publikationer/Research_report_08-6.pdf, 2008.

Examples

```
data( milk )
DA.reg( milk )
data( sbp )
print( DA.reg( sbp ), digits=3 )
```

`Deming` *Regression with errors in both variables (Deming regression)*

Description

The function makes a regression of y on x, assuming that both x and y are measured with error. This problem only has an analytical solution if the ratio of the variances is known, hence this is required as an input parameter.

Usage

```
Deming(x, y, vr = sdr^2, sdr = sqrt(vr),
       boot = FALSE, keep.boot = FALSE, alpha = 0.05)
```

Arguments

`x`	numerical variable.
`y`	numerical variable.
`vr`	The assumed known ratio of the (residual) variance of the `y`s relative to that of the `x`s. Defaults to 1.
`sdr`	do. for standard deviations. Defaults to 1. `vr` takes precedence if both are given.
`boot`	Should bootstrap estimates of standard errors of parameters be done? If `boot==TRUE`, 1000 bootstrap samples are done, if `boot` is numeric, `boot` samples are made.
`keep.boot`	Should the 4-column matrix of bootstrap samples be returned? If `TRUE`, the summary is printed, but the matrix is returned invisibly. Ignored if `boot=FALSE`
`alpha`	What significance level should be used when displaying confidence intervals?

Details

The formal model underlying the procedure is based on a so called functional relationship:

$$x_i = \xi_i + e_{1i}, \qquad y_i = \alpha + \beta\xi_i + e_{2i}$$

with $\text{var}(e_{1i}) = \sigma$, $\text{var}(e_{2i}) = \lambda\sigma$, where λ is the known variance ratio.

The estimates of the residual variance is based on a weighting of the sum of squared deviations in both directions, divided by $n - 2$. The ML estimate would use $2n$ instead, but in the model we actually estimate $n + 2$ parameters — α, β and the n ξs.

This is not in Peter Sprent's book (see references).

Value

If `boot==FALSE` a named vector with components `Intercept`, `Slope`, `sigma.x`, `sigma.y`, where `x` and `y` are substituted by the variable names.

If `boot==TRUE` a matrix with rows `Intercept`, `Slope`, `sigma.x`, `sigma.y`, and colums giving the estimates, the bootstrap standard error and the bootstrap estimate and c.i. as the 0.5, $\alpha/2$ and $1-\alpha/2$ quantiles of the sample.

If `keep.boot==TRUE` this summary is printed, but a matrix with columns `Intercept`, `Slope`, `sigma.x`, `sigma.y` and `boot` rows is returned.

Author(s)

Bendix Carstensen, Steno Diabetes Center, `<bxc@steno.dk>`,
http://www.biostat.ku.dk/~bxc.

References

Peter Sprent: Models in Regression, Methuen & Co., London 1969, ch.3.4.

WE Deming: Statistical adjustment of data, New York: Wiley, 1943. [This is a reference taken from a reference list — I never saw the book myself].

See Also

`MCmcmc`

Examples

```
# Some data
x <- runif(100,0,5) + rnorm(100)
y <- 2 + 3 * x + rnorm(100,sd=2)
# Deming regression with equal variances, variance ratio 2.
Deming(x,y)
Deming(x,y,vr=2)
Deming(x,y,boot=TRUE)
bb <- Deming(x,y,boot=TRUE,keep.boot=TRUE)
str(bb)
# Plot data with the two classical regression lines
plot(x,y)
abline(lm(y~x))
ir <- coef(lm(x~y))
abline(-ir[1]/ir[2],1/ir[2])
abline(Deming(x,y,sdr=2)[1:2],col="red")
abline(Deming(x,y,sdr=10)[1:2],col="blue")
# Comparing classical regression and "Deming extreme"
summary(lm(y~x))
Deming(x,y,vr=1000000)
```

`Enzyme` *Enzyme activity data*

Description

Three measurement of enzyme activity on 24 patients. The measurements is of the enzymes sucrase and alkaline phosphatase. The interest is to compare the 'homogenate' and 'pellet' methods.

Usage

```
data(Enzyme)
```

Format

A data frame with 72 observations on the following 3 variables.

`meth` a factor with levels `SucHom SucPel Alkphos`, representing three different measurements, i.e. homogenate and pellet values of sucrase, as well as homogenate values of alkaline.

`item` a numeric vector, the person ID for the 24 patients

`y` a numeric vector, the measurements on the enzyme activity.

Source

R. L. Carter; Restricted Maximum Likelihood Estimation of Bias and Reliability in the Comparison of Several Measuring Methods; Biometrics, Dec., 1981, Vol. 37, No. 4, pp. 733-741.

Examples

```
data(Enzyme)
Enzyme <- Meth( Enzyme )
summary( Enzyme )
plot(Enzyme)
```

fat *Measurements of subcutaneous and visceral fat*

Description

43 persons had Subcutaneous and Visceral fat thickness measured at Steno Diabetes Center in 2006 by two observers; all measurements were done three times. The interest is to compare the measurements by the two observers. Persons are items, observers are methods, the three replicates are exchangeable within (person,observer)=(item,method)

Usage

```
data(fat)
```

Format

A data frame with 258 observations on the following 6 variables.

`Id` Person id.

`Obs` Observers, a factor with levels `KL` and `SL`.

`Rep` Replicate — exchangeable within person and observer.

`Sub` Subcutaneous fat measured in cm.

`Vic` Visceral fat measured in cm.

Examples

```
data(fat)
str(fat)
vic <- Meth( fat, meth=2, item=1, repl="Rep", y="Vic" )
str(vic)
BA.est( vic, linked=FALSE )
```

glucose *Glucose measurements by different methods*

Description

74 persons in 5 centres in Finland had blood glucose measured by 11 different methods, based on 4 different types of blood. Each person had blood sampled at 0, 30, 60 and 120 min after a 75 g glucose load.

Usage

```
data(glucose)
```

Format

A data frame with 1302 observations on the following 6 variables.

`meth` Method of measurement. A factor with 11 levels: `n.plas1` `n.plas2` `h.cap` `h.blood` `h.plas` `h.serum` `m.plas` `m.serum` `o.cap` `s.serum` `k.plas`.

`type` Type of blood sample. A factor with 4 levels: `blood` `plasma` `serum` `capil`

`item` Person id.

`time` Time of blood sampling. Minutes since glucose load.

`cent` Center of sampling. Except for the two first methods, `n.plas1` and `n.plas2`, samples were analyzed at the centres too

`y` Glucose measurement in mmol/l.

Source

The study was conducted at the National Public Health Institute in Helsinki by Jaana Lindstrom.

References

B Carstensen, J Lindstrom, J Sundvall, K Borch-Johnsen1, J Tuomilehto & the DPS Study Group: Measurement of Blood Glucose: Comparison between different Types of Specimens. Annals of Clinical Biochemistry, to appear.

Examples

```
data( glucose )
str( glucose )
# Use only plasma and serum as methods and make a Bland-Altman plot
gluc <- subset( glucose, type %in% c("plasma","serum") )
gluc$meth <- gluc$type
gluc$repl <- gluc$time
BA.plot( gluc )
```

`hba.MC` *A MCmcmc object from the hba1c data*

Description

This object is included for illustrative purposes. It is a result of a 5-hour run using MCmcmc, with `n.iter=100000`.

Usage

```
data(hba.MC)
```

Format

The format is a `MCmcmc` object.

Details

The data are the venous measurements from the `hba1c` dataset, using the day of analysis as replicate. Measurements are taken to be linked within replicate (=day of analysis).

Examples

```
data(hba.MC)
attr(hba.MC,"mcmc.par")
# print.MCmcmc(hba.MC)
# One of the chains is really fishy (it's the first one)
# trace.MCmcmc(hba.MC)
# trace.MCmcmc(hba.MC,"beta")
# Try to have a look, excluding the first chain
# hba.MCsub <- subset.MCmcmc(hba.MC,chains=-1)
# trace.MCmcmc(hba.MCsub)
# trace.MCmcmc(hba.MCsub,"beta")
# A MCmcmc object also has class mcmc.list, so we can use the
# coda functions for covergence diagnostics:
# acfplot( subset.MCmcmc(hba.MC, subset="sigma"))
```

`hba1c` *Measurements of HbA1c from Steno Diabetes Center*

Description

Three analysers (machines) for determination of HbA1c (glycosylated haemoglobin) were tested on samples from 38 individuals. Each had drawn a venous and capillary blood sample. These were analysed on five different days.

Usage

```
data(hba1c)
```

Format

A data frame with 835 observations on the following 6 variables.

`dev` Type of machine used. A factor with levels `BR.V2`, `BR.VC` and `Tosoh`.

`type` Type of blood analysed (capillary or venous). A factor with levels `Cap` `Ven`

`item` Person-id. A numeric vector

`d.samp` Day of sampling.

`d.ana` Day of laboratory analysis.

`y` The measured value of HbA1c.

Details

In the terminology of method comparison studies, methods is the cross-classification of `dev` and `type`, and replicate is `d.ana`. It may be of interest to look at the effect of time between `d.ana` and `d.samp`, i.e. the time between sampling and analysis.

Source

Bendix Carstensen, Steno Diabetes Center.

References

These data were analysed as example in: Carstensen: Comparing and predicting between several methods of measurement, Biostatistics 5, pp. 399–413, 2004.

Examples

```
data(hba1c)
str(hba1c)
hb1  <- with( hba1c,
              Meth( meth = interaction(dev,type),
                    item = item,
                    repl = d.ana-d.samp,
                       y = y, print=TRUE ) )
```

`MCmcmc` *Fit a model for method comparison studies using WinBUGS*

Description

A model linking each of a number of methods of measurement linearly to the "true" value is set up in BUGS and run via the function `bugs` from the `R2WinBUGS` package.

Usage

```
  MCmcmc( data,
          bias = "linear",
           IxR = has.repl(data), linked = IxR,
           MxI = TRUE,           matrix = MxI,
        varMxI = nlevels(factor(data$meth)) > 2,
      n.chains = 4,
        n.iter = 2000,
      n.burnin = n.iter/2,
        n.thin = ceiling((n.iter-n.burnin)/1000),
bugs.directory = getOption("bugs.directory"),
         debug = FALSE,
bugs.code.file = "model.txt",
       clearWD = TRUE,
     code.only = FALSE,
      ini.mult = 2,
      list.ini = TRUE,
           org = FALSE,
       program = "BRugs",
     Transform = NULL,
     trans.tol = 1e-6,
           ... )
## S3 method for class 'MCmcmc'
summary( object, alpha=0.05, ...)
## S3 method for class 'MCmcmc'
```

```
print( x, digits=3, alpha=0.05, ... )
## S3 method for class 'MCmcmc'
subset( x, subset=NULL, allow.repl=FALSE, chains=NULL, ... )
## S3 method for class 'MCmcmc'
mcmc( x, ... )
```

Arguments

`data`	Data frame with variables `meth`, `item`, `repl` and `y`, possibly a `Meth` object. `y` represents a measurement on an `item` (typically patient or sample) by method `meth`, in replicate `repl`.
`bias`	Character. Indicating how the bias between metods should be modelled. Possible values are `"none"`, `"constant"`, `"linear"` and `"proportional"`. Only the first three letters are significant. Case insensitive.
`IxR`	Logical. Are the replicates linked across methods, i.e. should a random `item` by `repl` be included in the model.
`linked`	Logical, alias for `IxR`.
`MxI`	Logical, should a `meth` by `item` effect be included in the model?
`matrix`	Logical, alias for `MxI`.
`varMxI`	Logical, should the method by item effect have method-specific variances. Ignored if only two methods are compared.
`n.chains`	How many chains should be run by WinBUGS — passed on to `bugs`.
`n.iter`	How many total iterations — passed on to `bugs`.
`n.burnin`	How many of these should be burn-in — passed on to `bugs`.
`n.thin`	How many should be sampled — passed on to `bugs`.
`bugs.directory`	Where is WinBUGS (>=1.4) installed — passed on to `bugs`. The default is to use a parameter from options(). If you use this routinely, this is most conveniently set in your `.Rprofile` file.
`debug`	Should WinBUGS remain open after running — passed on to `bugs`.
`clearWD`	Should the working directory be cleared for junk files after the running of WinBUGS — passed on to `bugs`.
`bugs.code.file`	Where should the bugs code go?
`code.only`	Should `MCmcmc` just create a bugs code file and a set of inits? See the `list.ini` argument.
`ini.mult`	Numeric. What factor should be used to randomly perturb the initial values for the variance componets, see below in details.

`list.ini` List of lists of starting values for the chains, or logical inidcating whether starting values should be generated. If `TRUE` (the default), the function `VC.est` will be used to generate initial values for the chains. `list.ini` is a list of length `n.chains`. Each element of which is a list with the following vectors as elements:

`mu` - length I

`alpha` - length M

`beta` - length M

`sigma.mi` - length M - if M is 2 then length 1

`sigma.ir` - length 1

`sigma.mi` - length M

`sigma.res` - length M

If `code.only==TRUE`, `list.ini` indicates whether a list of initial values is returned (invisibly) or not. If `code.only==FALSE`, `list.ini==FALSE` is ignored.

`org` Logical. Should the posterior of the original model parameters be returned too? If `TRUE`, the `MCmcmc` object will have an attribute, `original`, with the posterior of the parameters in the model actually simulated.

`program` Which program should be used for the MCMC simulation. Possible values are "brugs","openbugs","ob" (openBUGS), "winbugs","wb" (WinBUGS).

`Transform` Transformation of data (`y`) before analysis. See `choose.trans`.

`trans.tol` The tolerance used to check whether the supplied transformation and its inverse combine to the identity.

`...` Additional arguments passed on to `bugs`.

`object` A `MCmcmc` object

`alpha` 1 minus the the confidence level

`x` A `MCmcmc` object

`digits` Number of digits after the decimal point when printing.

`subset` Numerical, character or list giving the variables to keep. If numerical, the variables in the `MCmcmc` object with these numbers are selected. If character, each element of the character vector is "grep"ed against the variable names, and the matches are selected to the subset. If a list each element is used in turn, numerical and character elements can be mixed.

`allow.repl` Should duplicate columns be allowed in the result?

`chains` Numerical vector giving the number of the chains to keep.

Details

This function uses features currently only available under Windows, so the function returns NULL unless the operating system is Windows.

The model set up for an observation y_{mir} is:

$$y_{mir} = \alpha_m + \beta_m(\mu_i + b_{ir} + c_{mi}) + e_{mir}$$

where b_{ir} is a random `item` by `repl` interaction (included if `"ir" %in% random`) and c_{mi} is a random `meth` by `item` interaction (included if `"mi" %in% random`). The μ_i's are parameters in the model but are not monitored — only the αs, βs and the variances of b_{ir}, c_{mi} and e_{mir} are monitored and returned. The estimated parameters are only determined up to a linear transformation of the μs, but the linear functions linking methods are invariant. The identifiable conversion parameters are:

$$\alpha_{m\cdot k} = \alpha_m - \alpha_k\beta_m/\beta_k, \quad \beta_{m\cdot k} = \beta_m/\beta_k$$

The posteriors of these are derived and included in the `posterior`, which also will contain the posterior of the variance components (the sd's, that is). Furthermore, the posterior of the point where the conversion lines intersects the identity as well as the prediction sd's between any pairs of methods are included.

The function `summary.MCmcmc` method gives estimates of the conversion parameters that are consistent. Clearly,

$$\text{median}(\beta_{1\cdot 2}) = 1/\text{median}(\beta_{2\cdot 1})$$

because the inverse is a monotone transformation, but there is no guarantee that

$$\text{median}(\alpha_{1\cdot 2}) = \text{median}(-\alpha_{2\cdot 1}/\beta_{2\cdot 1})$$

and hence no guarantee that the parameters derived as posterior medians produce conversion lines that are the same in both directions. Therefore, `summary.MCmcmc` computes the estimate for $\alpha_{2\cdot 1}$ as

$$(\text{median}(\alpha_{1\cdot 2}) - \text{median}(\alpha_{2\cdot 1})/\text{median}(\beta_{2\cdot 1}))/2$$

and the estimate of $\alpha_{1\cdot 2}$ correspondingly. The resulting parameter estimates defines the same lines.

Value

If `code.only==FALSE`, an object of class `MCmcmc` which is a `mcmc.list` object of the relevant parametes, i.e. the posteriors of the conversion parameters and the variance components transformed to the scales of each of the methods.

Furthermore, the object have the following attibutes:

`random` Character vector indicatinf which random effects ("ir","mi") were included in the model.

`methods`	Character vector with the method names.
`data`	The dataframe used in the analysis. This is used in `plot.MCmcmc` when plotting points.
`mcmc.par`	A list giving the number of chains etc. used to generate the object.
`original`	If `org=TRUE`, an `mcmc.list` object with the posterior of the original model parameters, i.e. the variance components and the unidentifiable mean parameters.
`Transform`	The transformation used to the measurements before the analysis.

If `code.only==TRUE`, a list containing the initial values is generated.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc, Lyle Gurrin, University of Melbourne,
http://www.epi.unimelb.edu.au/about/staff/gurrin-lyle.

References

B Carstensen: Comparing and predicting between several methods of measurement, Biostatistics, 5, pp 399-413, 2004

See Also

`BA.plot`, `plot.MCmcmc`, `print.MCmcmc`, `check.MCmcmc`

Examples

```
data( ox )
str( ox )
MCmcmc( ox, MI=TRUE, IR=TRUE, code.only=TRUE, bugs.code.file="" )

### What is written here is not necessarily correct on your machine.
# ox.MC <- MCmcmc( ox, MI=TRUE, IR=TRUE, n.iter=100, program="winbugs" )
# ox.MC <- MCmcmc( ox, MI=TRUE, IR=TRUE, n.iter=100 )
# data( ox.MC )
#  str( ox.MC )
#print( ox.MC )
```

`Meth` *Create a Meth object representing a method comparison study*

Description

Creates a dataframe with columns `meth`, `item`, (`repl`) and `y`.

Usage

```
Meth( data=NULL,
      meth="meth", item="item", repl=NULL, y="y",
      print=!is.null(data), keep.vars=!is.null(data) )
## S3 method for class 'Meth'
summary( object, ... )
## S3 method for class 'Meth'
plot(x, y = NULL,
          col.LoA = "blue", col.pt = "black", cex.name = 2,
        var.range,
       diff.range,
        var.names = FALSE,
              pch = 16,
              cex = 0.7,
        Transform,
              ... )
## S3 method for class 'Meth'
subset(x, ... )
## S3 method for class 'Meth'
sample( x,
                     how = "random",
                       N = if( how=="items" ) nlevels( x$item ) else nrow(x),
                     ... )
## S3 method for class 'Meth'
transform(`_data`, ... )
```

Arguments

`data`	A dataframe.
`meth`	Vector of methods, numeric, character or factor. Can also be a number or character referring to a column in `data`.
`item`	Vector of items, numeric, character or factor. Can also be a number or character referring to a column in `data`.
`repl`	Vector of replicate numbers, numeric, character or factor. Can also be a number or character referring to a column in `data`.

`y`	Vector of measurements. Can also be a character or numerical vector pointing to columns in `data` which contains the measurements by different methods or a dataframe with columns representing measurements by different methods. In this case the argument `meth` is ignored, and the names of the columns are taken as method names. For the `plot` method the argument is either a vector of indices or names of methods to plot.
`print`	Logical: Should a summary result be printed?
`keep.vars`	Logical. Should the remaining variables from the dataframe `data` be transferred to the `Meth` object.
`object`	A `Meth` object.
`x`	A `Meth` object.
`col.LoA`	What color should be used for the limits of agreement.
`col.pt`	What color should be used for the points.
`cex.name`	Character expansion factor for plotting method names
`var.range`	The range of both axes in the scatter plot and the x-axis in the Bland-Altman plot be?
`diff.range`	The range of yaxis in the Bland-Altman plot. Defaults to a range as the x-axis, but centered around 0.
`var.names`	If logical: should the individual panels be labelled with the variable names?. If character, then the values of the character will be used to label the methods.
`pch`	Plot character for points.
`cex`	Plot charcter expansion for points.
`Transform`	Transformation used to the measurements prior to plotting. Function or character, see `choose.trans` for possible values.
`how`	Character. What sampling strategy should be used, one of `"random"`, `"linked"` or `"item"`. Only the first letter is significant. See details for explanation.
`N`	How many observations should be sampled?
`_data`	A `Meth` object.
`...`	Ignored by the `Meth` and the `summary` and `sample` functions. In the `plot` function, parameters passed on to both the panel function plotting methods against each other, as well as to those plotting differences against means.

Details

In order to perform analyses of method comparisons it is convenient to have a dataframe with classifying factors , `meth`, `item`, and possibly `repl` and the response

variable `y`. This function creates such a dataframe, and gives it a class, `Meth`, for which there is a number of methods: `summary` - tabulation, `plot` - plotting and a couple of analysis methods.

If there are replicates in the values of `item` it is assumed that those observations represent replicate measurements and different replicate numbers are given to those.

`sample.Meth` samples a `Meth` object with replacement. If `how=="random"`, a random sample of the rows are sampled, the existing values of `meth`, `item` and `y` are kept but new replicate numbers are generated. If `how=="linked"`, a random sample of the linked observations (i.e. observations with identical `item` and `repl` values) are sampled with replacement and replicate numbers are kept. If `how=="item"`, items are sampled with replacement, and their observations are included the sampled numner of times.

Value

The `Meth` function returns a `Meth` object which is a dataframe with columns `meth`, `item`, (`repl`) and `y`. `summary.Meth` returns a table classified by method and no. of replicate measurements, extended with columns of the total number of items, total number of observations and the range of the measurements. The `subset.Meth` returns a subset of the `Meth` rows.

Author(s)

Bendix Carstensen, `<bxc@steno.dk>`

Examples

```
data(fat)
# Different ways of selecting columns and generating replicate numbers
Sub1 <- Meth(fat,meth=2,item=1,repl=3,y=4,print=TRUE)
Sub2 <- Meth(fat,2,1,3,4,print=TRUE)
Sub3 <- Meth(fat,meth="Obs",item="Id",repl="Rep",y="Sub",print=TRUE)
summary( Sub3 )
plot( Sub3 )

# Use observation in different columns as methods
data( CardOutput )
head( CardOutput )
sv <- Meth( CardOutput, y=c("Svo2","Scvo2") )
# Note that replicates are generated if a non-unique item-id is used
sv <- Meth( CardOutput, y=c("Svo2","Scvo2"), item="Age" )
str( sv )
# A summary is not created if the the first argument (data=) is not used:
sv <- Meth( y=CardOutput[,c("Svo2","Scvo2")], item=CardOutput$VO2 )
summary(sv)
```

```
# Sample items
ssv <- sample.Meth( sv, how="item", N=8 )

# More than two methods
data( sbp )
plot( Meth( sbp ) )
# Creating non-unique replicate numbers per (meth,item) creates a warning:
data( hba1c )
hb1  <- with( hba1c,
              Meth( meth=dev, item=item, repl=d.ana-d.samp, y=y, print=TRUE ) )
hb2  <- with( subset(hba1c,type=="Cap"),
              Meth( meth=dev, item=item, repl=d.ana-d.samp, y=y, print=TRUE ) )
```

`Meth.sim` *Simulate a dataframe containing replicate measurements on the same items using different methods.*

Description

Simulates a dataframe representing data from a method comparison study. It is returned as a `Meth` object.

Usage

```
Meth.sim( Ni = 100,
          Nm = 2,
          Nr = 3,
          nr = Nr,
       alpha = rep(0,Nm),
        beta = rep(1,Nm),
    mu.range = c(0, 100),
    sigma.mi = rep(5,Nm),
    sigma.ir = 2.5,
   sigma.mir = rep(5,Nm),
     m.thin = 1,
     i.thin = 1 )
```

Arguments

`Ni`	The number of items (patient, animal, sample, unit etc.)
`Nm`	The number of methods of measurement.

`Nr`	The (maximal) number of replicate measurements for each (item,method) pair.
`nr`	The minimal number of replicate measurements for each (item,method) pair. If `nr<Nr`, the number of replicates for each (meth,item) pair is uniformly distributed on the points `nr:Nr`, otherwise `nr` is ignored. Different number of replicates is only meaningful if replicates are not linked, hence `nr` is also ignored when `sigma.ir>0`.
`alpha`	A vector of method-specific intercepts for the linear equation relating the "true" underlying item mean measurement to the mean measurement on each method.
`beta`	A vector of method-specific slopes for the linear equation relating the "true" underlying item mean measurement to the mean measurement on each method.
`mu.range`	The range across items of the "true" mean measurement. Item means are uniformly spaced across the range. If a vector length `Ni` is given, the values of that vector will be used as "true" means.
`sigma.mi`	A vector of method-specific standard deviations for a method by item random effect. Some or all components can be zero.
`sigma.ir`	Method-specific standard deviations for the item by replicate random effect.
`sigma.mir`	A vector of method-specific residual standard deviations for a method by item by replicate random effect (residual variation). All components must be greater than zero.
`m.thin`	Fraction of the observations from each method to keep.
`i.thin`	Fraction of the observations from each item to keep. If both `m.thin` and `i.thin` are given the thinning is by their componentwise product.

Details

Data are simulated according to the following model for an observation y_{mir}:

$$y_{mir} = \alpha_m + \beta_m(\mu_i + b_{ir} + c_{mi}) + e_{mir}$$

where b_{ir} is a random `item` by `repl` interaction (with standard deviation for method m the corresponding component of the vector $\sigma_i r$), c_{mi} is a random `meth` by `item` interaction (with standard deviation for method m the corresponding component of the vector $\sigma_m i$) and e_{mir} is a residual error term (with standard deviation for method m the corresponding component of the vector $\sigma_m ir$). The μ_i's are uniformly spaced in a range specified by `mu.range`.

Value

A `Meth` object, i.e. dataframe with columns `meth`, `item`, `repl` and `y`, representing results from a method comparison study.

Author(s)

Lyle Gurrin, University of Melbourne,
http://www.epi.unimelb.edu.au/about/staff/gurrin-lyle

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc

See Also

`summary.Meth`, `plot.Meth`, `MCmcmc`

Examples

```
Meth.sim( Ni=4, Nr=3 )
xx <- Meth.sim( Nm=3, Nr=5, nr=2, alpha=1:3, beta=c(0.7,0.9,1.2), m.thin=0.7 )
summary( xx )
plot( xx )
```

`MethComp` *Summarize conversion equations and prediction intervals between methods.*

Description

Takes the results from `BA.est`, `AltReg` or `MCmcmc` and returns a `MethComp` object, suitable for displaying the relationship between methods in print pr graphic form.

Usage

```
MethComp(obj)
## S3 method for class 'MethComp'
print(x, digits=3, ... )
## S3 method for class 'MethComp'
plot(x,
                    wh.cmp = 1:2,
                   pl.type = "convert",
                     axlim = range(x$data$y,na.rm=TRUE),
                    diflim = axlim-mean(axlim),
                    points = FALSE,
                      grid = TRUE,
                    N.grid = 10,
                  col.grid = grey(0.9),
                 col.lines = "black",
                col.points = "black",
                       eqn = tolower(substr(pl.type,1,1))=="c" &
```

```
                          is.null(attr(x,"Transform")),
                col.eqn = col.lines,
               font.eqn = 2,
                 digits = 1,

                    ... )
## S3 method for class 'MethComp'
lines(x,
                  wh.cmp = getOption("MethComp.wh.cmp"),
                 pl.type = getOption("MethComp.pl.type"),

              col.lines = "black",
                    lwd = c(3,1),
                  alpha = NULL,
                    ... )
## S3 method for class 'MethComp'
points(x,
                  wh.cmp = getOption("MethComp.wh.cmp"),
                 pl.type = getOption("MethComp.pl.type"),

            col.points = "black",
                    ... )
```

Arguments

`obj`	A `MethComp` or `MCmcmc` object.
`x`	A `MethComp` object.
`digits`	How many digits should be used when displaying conversion equations and variance components?
`wh.cmp`	Numeric of length 2. Which two methods should be plotted.
`pl.type`	Character. If "conv" it will be a plot of two methods against each other, otherwise it will be a plot of the 2nd minus the 1st versus the average; a Bland-Altman type plot.
`axlim`	The extent of the axes of the measurements.
`diflim`	The extent of the axis of the differences.
`points`	Logical. Should the points be included in the plot.
`grid`	Logical. Should there be a grid?
`N.grid`	Numeric. How many gridlines? If a vector of length>1, it will be taken as the position of the gridlines.
`col.grid`	Color of the gridlines.
`col.lines`	Color of the conversion lines.

lwd	Numerical vector of length 2. Width of the conversion line and the prediction limits respectively.
alpha	1 minus the confidence level for the prediction interval. If not given, the prediction interval is constructed as plus/minus twice the SD.
col.points	Color of the points.
eqn	Logical. Should the conversion equation be printed on the plot.
col.eqn	Color of the conversion formula
font.eqn	font for the conversion formula
...	Further arguments.

Details

Using MethComp on the results from BA.est or AltReg is not necessary, as these two functions already return objetcs of class MethComp.

plot.MethComp plots the conversion function with prediction limits; always using the original scale of measurements. It also sets the options "MethComp.wh.cmp" indicating which two methods are plotted and "MethComp.pl.type" indicating whether a plot of methods against each other or a Bland-Altman type plot of differences versus averages. By default the conversion lines are plotted.

lines.MethComp and points.MethComp adds conversion lines with prediction limits and points to a plot.

Value

MethComp returns a MethComp object, which is a list with three elements, Conv, a three-way array giving the linear conversion equations between methods, VarComp, a two-way array classified by methods and variance components and data, a copy of the original Meth object supplied — see the description under BA.est.

A MethComp object has an attribute Transform, which is either NULL, or a named list with elements trans and inv, both of which are functions. The first is the transformation applied to measurements before analysis; the results are all given on the transformed scale. The second is the inverse transformation; this is only used when plotting the resulting relationship between methods.

The methods print, plot, lines and points return nothing.

Author(s)

Bendix Carstensen, Steno Diabetes Center, <bxc@steno.dk>.

See Also

BA.est AltReg MCmcmc

Examples

```
data( ox )
BA.ox <- BA.est( ox, linked=TRUE )
print( BA.ox )
AR.ox <- AltReg( ox, linked=TRUE  )
print( AR.ox )
plot( AR.ox )
```

`milk` *Measurement of fat content of human milk by two different methods.*

Description

Fat content of human milk determined by measurement of glycerol released by enzymic hydrolysis of triglycerides (Trig) and measurement by the Standard Gerber method (Gerber). Units are (g/100 ml).

Usage

```
data(milk)
```

Format

A data frame with 90 observations on the following 3 variables.

`meth` a factor with levels `Gerber Trig`

`item` sample id

`y` a numeric vector

Source

The dataset is adapted from table 3 in: JM Bland and DG Altman: Measuring agreement in method comparison studies. Statistical Methods in Medical Research, 8:136-160, 1999. See: Lucas A, Hudson GJ, Simpson P, Cole TJ, Baker BA. An automated enzymic micromethod for the measurement of fat in human milk. Journal of Dairy Research 1987; 54: 487-92.

Examples

```
data(milk)
str(milk)
milk <- Meth(milk)
plot(milk)
abline(0,1)
```

ox *Measurement of oxygen saturation in blood*

Description

61 children had their blood oxygen content measured at the Children's Hospital in Melbourne, either with a chemical method analysing gases in the blood (`CO`) or by a pulse oximeter measuring transcutaneously (`pulse`). Replicates are linked between methods; i.e. replicate 1 for each of the two methods are done at the same time. However, replicate measurements were taken in quick succession so the pairs of measurements are exchangeable within person.

Usage

```
data(ox)
```

Format

A data frame with 354 observations on the following 4 variables.

`meth` Measurement methods, factor with levels `CO`, `pulse`

`item` Id for the child

`repl` Replicate of measurements. There were 3 measurements for most children, 4 had only 2 replicates with each method, one only 1

`y` Oxygen saturation in percent.

Examples

```
data(ox)
str(ox)
ox <- Meth(ox)
with( ox, table(table(item)) )
# The effect of basing LoA on means over replicates:
par( mfrow=c(1,2), mar=c(4,4,1,4) )
BA.plot( ox, ymax=20 )
BA.plot( ox, ymax=20, mean.repl=TRUE )
```

`ox.MC` *A MCmcmc object from the oximetry data.*

Description

This object is included for illustrative purposes. It is a result of using `MCmcmc`, with `n.iter=20000`.

Usage

```
data(ox.MC)
```

Format

The format is a `MCmcmc` object.

Details

The data are the `ox` dataset, where measurements are linked within replicate (=day of analysis).

Examples

```
data(ox.MC)
attr(ox.MC,"mcmc.par")
## Not run:
print.MCmcmc(ox.MC)
trace.MCmcmc(ox.MC)
trace.MCmcmc(ox.MC,"beta")
 post.MCmcmc(ox.MC)
 post.MCmcmc(ox.MC,"beta")
## End(Not run)
# A MCmcmc object also has class mcmc.list, so we can use the
# coda functions for covergence diagnostics:
## Not run:  acfplot( subset.MCmcmc(ox.MC, subset="sigma"))
```

`PBreg` *Passing-Bablok regression*

Description

Implementation of the Passing-Bablok's procedure for assessing of the equality of measurements by two different analytical methods.

Usage

```
  PBreg(x, y=NULL, conf.level=0.05, wh.meth=1:2)
  ## S3 method for class 'PBreg'
print(x,...)
```

Arguments

`x`	a numeric vector of measurements by method A, alternatively a data frame of exactly two columns, first column with measurements by method A, second column with measurements by method B. If `x` is a `Meth` object, the methods from that are used in the regression.
`y`	a numeric vector of measurements by method B - must be of the same length as `x`. If not provided, `x` must be a data frame of exactly 2 columns.
`conf.level`	confidence level for calculation of confidence boundaries.
`wh.meth`	Which of the methods from the `Meth` object are used in the regression.
`...`	other parameters, currently ignored.

Details

This is an implementation of the original Passing-Bablok procedure of fitting unbiased linear regression line to data in the method comparison studies. It calcualtes the unbiased slope and intercept, along with their confidence intervals. However, the tests for linearity is not yet fully implemented.

It doesn't matter which results are assigned to "Method A" and "Method B", however the "Method A" results will be plotted on the x-axis by the `plot` method.

Value

`PBreg` returns an object of class `"PBreg"`, for which the `print` and `plot` methods are defined.

An object of class `"PBreg"` is a list composed of the following elements:

`coefficients`	a matrix of 3 columns and 2 rows, containing the estimates of the intercept and slope, along with their confidence boundaries.
`residuals`	defined as in the `"lm"` class, as the response minus the fitted value.
`fitted.values`	the fitted values.
`model`	the model data frame used.
`n`	a vector of two values: the number of observations read, and the number of observations used.

`S`	A vector of all slope estimates.
`adj`	A vector of fit parameters, where *Ss* is the number of estimated slopes (`length(S)`), K is the offset for negative slopes, *M1* and *M2* are the locations of confidence boundaries in `S`, and l and L are the numbers of points above and below the fitted line, used in cusum calculation.
`cusum`	A vector of cumulative sums of residuals sorted by the D-rank.
`Di`	A vector of D-ranks.

Note

Please note that this method can become very computationally intensive for larger numbers of observations. One can expect a reasonable computation times for datasets with fewer than 100 observations.

Author(s)

Michal J. Figurski `<mfigrs@gmail.com>`

References

Passing, H. and Bablok, W. (1983), A New Biometrical Procedure for Testing the Equality of Measurements from Two Different Analytical Methods. *Journal of Clinical Chemistry and Clinical Biochemistry*, Vol 21, 709–720

See Also

`plot.PBreg`, `Deming`.

Examples

```
## Model data frame generation
a <- data.frame(x=seq(1, 30)+rnorm(mean=0, sd=1, n=30),
                y=seq(1, 30)*rnorm(mean=1, sd=0.4, n=30))

## Call to PBreg
x <- PBreg(a)
print(x)
par(mfrow=c(2,2))
plot(x, s=1:4)

# A real data example
data(milk)
milk <- Meth(milk)
summary(milk)
PBmilk <- PBreg( milk )
plot( PBmilk )
```

PEFR — *Peak Expiratory Flow Rate (PEFR) measurements with Wright peak flow and mini Wright peak flow meter.*

Description

Measurement of PEFR with Wright peak flow and mini Wright peak flow meter on 17 individuals.

Usage

```
data(PEFR)
```

Format

A data frame with 68 observations on the following 3 variables.

`meth` a factor with levels `Wright` and `Mini`, representing measurements by a Wright peak flow meter and a mini Wright meter respectively, in random order.

`item` Numeric vector, the person ID.

`y` Numeric vector, the measurements, i.e. PEFR for the two measurements with a Wright peak flow meter and a mini Wright meter respectively. The measurement unit is l/min.

repl Numeric vector, replicate number. Replicates are exchangeable within item.

Source

J. M. Bland and D. G. Altman (1986) Statistical Methods for Assessing Agreement Between Two Methods of Clinical Measurement, Lancet. 1986 Feb 8;1(8476):307-10.

Examples

```
data(PEFR)
PEFR <- Meth(PEFR)
summary(PEFR)
plot(PEFR)
plot(perm.repl(PEFR))
```

`perm.repl` *Manipulate the replicate numbering within (item,method)*

Description

Replicate numbers are generated within (item,method) in a dataframe representing a method comparison study. The function assumes that observations are in the correct order within each (item,method), i.e. if replicate observations are non-exchangeable within method, linked observations are assumed to be in the same order within each (item,method).

Usage

```
make.repl( data )
 has.repl( data )
perm.repl( data )
```

Arguments

`data`	A `Meth` object or a data frame with columns `meth`, `item` and `y`.

Details

`make.repl` just adds replicate numbers in the order of the data.frame rows. `perm.repl` is designed to explore the effect of permuting the replicates within (item,method). If replicates are truly exchangeable within methods, the inference should be independent of this permutation.

Value

`make.repl` returns a dataframe with a column, `repl` added or replaced, whereas `has.repl` returns a logical indicating wheter a combination of (`meth`,`item`) wioth more that one valid y- value.

`perm.repl` returns a dataframe of class `Meth` where the rows (i.e. replicates) are randomly permuted within (`meth`,`item`), and subsequently ordered by (`meth`,`item`,`repl`).

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc

See Also

`perm.repl`

Examples

```
data(ox)
xx <- subset( ox, item<4 )[,-3]
cbind( xx, make.repl(xx) )
cbind( make.repl(xx), perm.repl(xx) )
data( ox )
xx <- subset( ox, item<4 )
cbind( xx, perm.repl(xx) )
# Replicates are linked in the oximetry dataset, so randomly permuting
# them clearly inflates the limits of agreement:
par( mfrow=c(1,2), mar=c(4,4,1,4) )
BA.plot(           ox , ymax=30, digits=1 )
BA.plot( perm.repl(ox), ymax=30, digits=1 )
```

`plot.MCmcmc` *Plot estimated conversion lines and formulae.*

Description

Plots the pairwise conversion formulae between methods from a `MCmcmc` object.

Usage

```
plot.MCmcmc( x,
             axlim = range( attr(x,"data")$y, na.rm=TRUE ),
           wh.cmp,
         lwd.line = c(3,1), col.line = rep("black",2), lty.line=rep(1,2),
              eqn = TRUE, digits = 2,
            grid = FALSE, col.grid=gray(0.8),
          points = FALSE,
         col.pts = "black", pch.pts = 16, cex.pts = 0.8,
             ... )
```

Arguments

`x`	A `MCmcmc` object
`axlim`	The limits for the axes in the panels
`wh.cmp`	Numeric vector or vector of method names. Which of the methods should be included in the plot?
`lwd.line`	Numerical vector of length 2. The width of the conversion line and the prediction limits. If the second values is 0, no prediction limits are drawn.

`col.line`	Numerical vector of length 2. The color of the conversion line and the prediction limits.
`lty.line`	Numerical vector of length 2. The line types of the conversion line and the prediction limits.
`eqn`	Should the conversion equations be printed on the plot?. Defaults to `TRUE`.
`digits`	How many digits after the decimal point shoudl be used when printing the conversion equations.
`grid`	Should a grid be drawn? If a numerical vector is given, the grid is drawn at those values.
`col.grid`	What color should the grid have?
`points`	Logical or character. Should the points be plotted. If `TRUE` or `"repl"` paired values of single replicates are plotted. If `"perm"`, replicates are randomly permuted within (item, method) befor plotting. If `"mean"`, means across replicates within item, method are formed and plotted.
`col.pts`	What color should the observation have.
`pch.pts`	What plotting symbol should be used.
`cex.pts`	What scaling should be used for the plot symbols.
`...`	Parameters to pass on. Currently not used.

Value

Nothing. The lower part of a (M-1) by (M-1) matrix of plots is drawn, showing the pairwise conversion lines. In the corners of each is given the two conversion equations together with the prediction standard error.

See Also

`MCmcmc`, `print.MCmcmc`

Examples

```
## Not run: data( hba1c )
## Not run: str( hba1c )
## Not run: hba1c <- transform( subset( hba1c, type=="Ven" ),
                    meth = dev,
                    repl = d.ana )
## End(Not run)
## Not run: hb.res <- MCmcmc( hba1c, n.iter=50 )
## Not run: data( hba.MC )
## Not run: str( hba.MC )
## Not run: par( ask=TRUE )
## Not run: plot( hba.MC )
## Not run: plot( hba.MC, pl.obs=TRUE )
```

`plot.PBreg` *Passing-Bablok regression - plot method*

Description

A plot method for the `"PBreg"` class object, that is a result of Passing-Bablok regression.

Usage

```
## S3 method for class 'PBreg'
plot(x,
                pch=21, bg="#2200aa33",
                xlim=c(0, max(x$model)), ylim=c(0, max(x$model)),
                xlab=x$meths[1], ylab=x$meths[2],
                subtype=1, ...)
```

Arguments

`x`	an object of class `"PBreg"`
`pch`	Which plotting character should be used for the points.
`bg`	Background colour.
`xlim`	Limits for the x-axis.
`ylim`	Limits for the y-axis.
`xlab`	Label on the x-axis.
`ylab`	Label on the y-axis.
`subtype`	a numeric value or vector, that selects the desired plot subtype. Subtype **1** is an x-y plot of raw data with regression line and confidence boundaries for the fit as a shaded area. This is the default. Subtype **2** is a ranked residuals plot. Subtype **3** is the "Cusum" plot useful for assessing linearity of the fit. Plot subtypes 1 through 3 are standard plots from the 1983 paper by Passing and Bablok - see the reference. Plot subtype **4** is a histogram (with overlaid density line) of the individual slopes. The range of this plot is limited to 7 x IQR for better visibility.
`...`	other parameters as in `"plot"`, some of which are pre-defined for improved appearance. This affects only the subtype 1 plot.

Author(s)

Michal J. Figurski `<mfigrs@gmail.com>`

References

Passing, H. and Bablok, W. (1983), A New Biometrical Procedure for Testing the Equality of Measurements from Two Different Analytical Methods. *Journal of Clinical Chemistry and Clinical Biochemistry*, **Vol 21**, 709–720

See Also

`PBreg`, `Deming`.

Examples

```
## Model data frame generation
a <- data.frame(x=seq(1, 30)+rnorm(mean=0, sd=1, n=30),
                y=seq(1, 30)*rnorm(mean=1, sd=0.4, n=30))

## Call to PBreg
x <- PBreg(a)
print(x)
par(mfrow=c(2,2))
plot(x, s=1:4)
```

`plot.VarComp` *Plot the a posteriori densities for variance components*

Description

When a method comparison model i fitted and stored in a `MCmcmc` object, then the posterior distributions of the variance components are plotted, in separate displays for method.

Usage

```
plot.VarComp( x,
          which,
       lwd.line = rep(2, 4),
       col.line = c("red", "green", "blue", "black"),
       lty.line = rep(1, 4),
           grid = TRUE,
       col.grid = gray(0.8),
            rug = TRUE,
          probs = c(5, 50, 95),
        tot.var = FALSE,
        same.ax = TRUE,
     meth.names = TRUE,
```

```
        VC.names = "first",
             ... )
```

Arguments

`x`	A `MCmcmc` object.
`which`	For which of the compared methods should the plot be made?
`lwd.line`	Line width for drawing the density.
`col.line`	Color for drawing the densities.
`lty.line`	Line type for drawing the densities.
`grid`	Logical. Should a vertical grid be set up? If numeric it is set up at the values specified. If `same.ax`, the range of the grid is taken to be the extent of the x-axis for all plots.
`col.grid`	The color of the grid.
`rug`	Should a small rug at the bottom show posterior quantiles?
`probs`	Numeric vector with numbers in the range from 0 to 100, indicating the posterior percentiles to be shown in the rug.
`tot.var`	Should the posterior of the total variance also be shown?
`same.ax`	Should the same axes be used for all methods?
`meth.names`	Should the names of the methods be put on the plots?
`VC.names`	Should the names of the variance components be put on the first plot (`"first"`), the last (`"last"`), all (`"all"`) or none (`"none"`). Only the first letter is needed.
`...`	Parameters passed on the `density` furnction that does the smoothing of the posterior samples.

Details

The function generates a series of plots, one for each method compared in the `MCmcmc` object supplied (or those chosen by `which=`). Therefore the user must take care to set `mfrow` or `mfcol` to capture all the plots.

Value

A list with one element for each method. Each element of this is a list of densities, i.e. of objects of class `density`, one for each variance component.

Author(s)

Bendix Carstensen, www.biostat.ku.dk/~bxc

See Also

`plot.MCmcmc`, `MCmcmc`, `check.MCmcmc`

Examples

```
data( ox.MC )
par( mfrow=c(2,1) )
plot.VarComp( ox.MC, grid=c(0,15) )
```

---

plvol — *Measurements of plasma volume measured by two different methods.*

Description

For each subject (`item`) the plasma volume is expressed as a percentage of the expected value for normal individuals. Two alternative sets of normal values are used, named Nadler and Hurley respectively.

Usage

```
data(plvol)
```

Format

A data frame with 198 observations on the following 3 variables.

`meth` a factor with levels `Hurley` and `Nadler`

`item` a numeric vector

`y` a numeric vector

Source

The datset is adapted from table 2 in: JM Bland and DG Altman: Measuring agreement in method comparison studies. Statistical Methods in Medical Research, 8:136-160, 1999. Originally supplied to Bland \& Altman by C Dore, see: Cotes PM, Dore CJ, Liu Yin JA, Lewis SM, Messinezy M, Pearson TC, Reid C. Determination of serum immunoreactive erythropoietin in the investigation of erythrocytosis. New England Journal of Medicine 1986; 315: 283-87.

Examples

```
data(plvol)
str(plvol)
plot( y[meth=="Nadler"]~y[meth=="Hurley"],data=plvol,
      xlab="Plasma volume (Hurley) (pct)",
      ylab="Plasma volume (Nadler) (pct)" )
abline(0,1)
par( mar=c(4,4,1,4) )
BA.plot(plvol)
```

---

rainman *Perception of points in a swarm*

Description

Five raters were asked to guess the number of points in a swarm for 10 different figures (which - unknown to the raters - were each repeated three times).

Usage

```
data(rainman)
```

Format

A data frame with 30 observations on the following 6 variables.

`SAND` The true number of points in the swarm. Each picture is replicated thrice

`ME` Ratings from judge 1

`TM` Ratings from judge 2

`AJ` Ratings from judge 3

`BM` Ratings from judge 4

`LO` Ratings from judge 5

Details

The raters had approximately 10 seconds to judge each picture, and they thought it were 30 different pictures. Before starting the experiment they were shown 6 (unrelated) pictures and were told the number of points in each of those pictures. The SAND column contains the picture id (which is also the true number of points in the swarm).

Source

Collected by Claus Ekstrom.

Examples

```
library(MethComp)
data( rainman )
str( rainman )
RM <- Meth( rainman, item=1, y=2:6 )
head( RM )
BA.est( RM, linked=FALSE )
library(lme4)
mf <- lmer( y ~ meth + item + (1|MI),
                data = transform( RM, MI=interaction(meth,item) ) )
summary( mf )
mr <- lmer( y ~ (1|meth) + (1|item) + (1|MI),
                data = transform( RM, MI=interaction(meth,item) ) )
summary( mr )

#
# Point swarms were generated by the following program
#
## Not run:
set.seed(2) # Original
npoints <- sample(4:30)*4
nplots <- 10
pdf(file="swarms.pdf", onefile=TRUE)

s1 <- sample(npoints[1:nplots])
print(s1)
for (i in 1:nplots) {
  n <- s1[i]
  set.seed(n)
  x <- runif(n)
  y <- runif(n)
  plot(x,y, xlim=c(-.15, 1.15), ylim=c(-.15, 1.15), pch=20, axes=F,
       xlab="", ylab="")
}
s1 <- sample(npoints[1:nplots])
print(s1)
for (i in 1:nplots) {
  n <- s1[i]
  set.seed(n)
  x <- runif(n)
  y <- runif(n)
  plot(y,x, xlim=c(-.15, 1.15), ylim=c(-.15, 1.15), pch=20, axes=F,
       xlab="", ylab="")
}
s1 <- sample(npoints[1:nplots])
print(s1)
for (i in 1:nplots) {
```

```
  n <- s1[i]
  set.seed(n)
  x <- runif(n)
  y <- runif(n)
  plot(-x,y, xlim=c(-1.15, .15), ylim=c(-.15, 1.15), pch=20, axes=F,
       xlab="", ylab="")
}
dev.off()

## End(Not run)
```

`sbp` *Systolic blood pressure measured by three different methods.*

Description

For each subject (`item`) there are three replicate measurements by three methods (two observers, J and R and the automatic machine, S). The replicates are linked within (method,item).

Usage

```
data(sbp)
```

Format

A data frame with 765 observations on the following 4 variables:

`meth` Methods, a factor with levels `J`(observer 1), `R`(observer 2) and `S`(machine)

`item` Person id, numeric.

`repl` Replicate number, a numeric vector

`y` Systolic blood pressure masurement, a numeric vector

Source

The dataset is adapted from table 1 in: JM Bland and DG Altman: Measuring agreement in method comparison studies. Statistical Methods in Medical Research, 8:136-160, 1999. Originally supplied to Bland \& Altman by E. O'Brien, see: Altman DG, Bland JM. The analysis of blood pressure data. In O'Brien E, O'Malley K eds. Blood pressure measurement. Amsterdam: Elsevier, 1991: 287-314.

See Also

`sbp.MC`

Examples

```
data(sbp)
par( mfrow=c(2,2), mar=c(4,4,1,4) )
BA.plot( sbp, comp=1:2 )
BA.plot( sbp, comp=2:3 )
BA.plot( sbp, comp=c(1,3) )
BA.est( sbp, linked=TRUE )
```

---

sbp.MC *A MCmcmc object from the sbp data*

Description

This object is included for illustrative purposes. It is a result of using `MCmcmc`, with `n.iter=100000` on the dataset `sbp` from this package.

Usage

```
data(sbp.MC)
```

Format

The format is a `MCmcmc` object.

Details

The basic data are measurements of systolic blood pressure from the `sbp` dataset. Measurements are taked to be linked within replicate. The code used to generate the object was:

```
library(MethComp)
data( sbp )
spb <- Meth( sbp )
sbp.MC <- MCmcmc( sbp, linked=TRUE, n.iter=100000 ) )
```

Examples

```
data(sbp.MC)
# How was the data generated
attr(sbp.MC,"mcmc.par")

# Traceplots
trace.MCmcmc(sbp.MC)
trace.MCmcmc(sbp.MC,"beta")
```

```
# A MCmcmc object also has class mcmc.list, so we can use the
# standard coda functions for covergence diagnostics:
acfplot( subset.MCmcmc(sbp.MC,subset="sigma") )

# Have a look at the correlation between the 9 variance parameters
pairs.MCmcmc( sbp.MC )

# Have a look at whether the MxI variance componnts are the same between methods:
pairs.MCmcmc( sbp.MC, subset=c("ir"), eq=TRUE,
              panel=function(x,y,...)
                    {
                    abline(0,1)
                    abline(v=median(x),h=median(y),col="gray")
                    points(x,y,...)
                    }
            )
```

`scint` *Relative renal function by Scintigraphy*

Description

Measurements of the relative kidney function (=renal function) for 111 patients. The percentage of the total renal function present in the left kidney is determined by one reference method, `DMSA` (static) and by one of two dynamic methods, `DTPA` or `EC`.

Usage

```
data(scint)
```

Format

A data frame with 222 observations on the following 5 variables:

`meth` Measurement method, a factor with levels `DMSA`, `DTPA`, `EC`.

`item` Patient identification.

`y` Percentage of total kidney function in the left kidney.

`age` Age of the patient.

`sex` Sex of the patient, a factor with levels `F`, `M`.

Source

F. C. Domingues, G. Y. Fujikawa, H. Decker, G. Alonso, J. C. Pereira, P. S. Duarte: Comparison of Relative Renal Function Measured with Either 99mTc-DTPA or 99mTc-EC Dynamic Scintigraphies with that Measured with 99mTc-DMSA Static Scintigraphy. International Braz J Urol Vol. 32 (4): 405-409, 2006

Examples

```
data(scint)
str(scint)
# Make a Bland-Altman plot for each of the possible comparisons:
par(mfrow=c(1,2),mgp=c(3,1,0)/1.6,mar=c(3,3,1,3))
BA.plot(scint,comp.levels=c(1,2),ymax=15,digits=1,cex=2)
BA.plot(scint,comp.levels=c(1,3),ymax=15,digits=1,cex=2)
```

TDI *Compute Lin's Total deviation index*

Description

This index calculates a value such that a certain fraction of difference between methods will be numerically smaller than this.

Usage

```
TDI( y1, y2, p = 0.05, boot = 1000, alpha = 0.05 )
```

Arguments

`y1`	Measurements by one method.
`y2`	Measurements by the other method
`p`	The fraction of items with differences numerically exceeding the TDI
`boot`	If numerical, this is the number of bootstraps. If `FALSE` no confidence interval for the TDI is produced.
`alpha`	1 - confidende degree.

Details

If `boot==FALSE` a single number, the TDI is returned. If `boot` is a number, the median and the 1-`alpha`/2 central interval based on `boot` resamples are returned too, in a named vector of length 4.

Value

A list with 3 components. The names of the list are preceeded by the criterion percentage, i.e. the percentage of the population that the TDI is devised to catch.

`TDI` The numerically computed value for the TDI. If `boot` is numeric, a vector of median and a bootstrap c.i. is appended.

`TDI` The approximate value of the TDI

`Limits of Agreement`
Limits of agreement

Note

The TDI is a measure which esentially is a number K such that the interval [-K,K] contains the limits of agreement.

Author(s)

Bendix Carstensen, bxc@steno.dk

References

LI Lin: Total deviation index for measuring individual agreement with applications in laboratory performance and bioequivalence, Statistics in Medicine, 19, 255-270 (2000)

See Also

`BA.plot`,`corr.measures`

Examples

```
data(plvol)
pw <- to.wide(plvol)
with(pw,TDI(Hurley,Nadler))
```

`to.wide` *Functions to convert between long and wide representations of data.*

Description

These functions are merely wrappers for `reshape`. Given the complicated syntax of `reshape` and the particularly simple structure of this problem, the functions facilitate the conversion enormously.

Usage

```
to.wide( data, warn )
to.long( data, vars )
```

Arguments

`data`	A dataframe
`warn`	Logical. Should a warning be printed when replicates are taken as items?
`vars`	The variables representing measurements by different methods. Either a character vector of names, or a numerical vector with the number of the variables in the dataframe.

Details

If `data` represents method comparisons with exchangeable replicates within method, the transformation to wide format does not necessarily make sense.

Value

A dataframe.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://www.biostat.ku.dk/~bxc

See Also

`perm.repl`

Examples

```
data( milk )
str( milk )
mw <- to.wide( milk )
str( mw )
( mw <- subset( mw, item < 3 ) )
to.long( mw, 3:4 )
```

`VitCap` *Merits of two instruments designed to measure certain aspects of human lung function (Vital Capacity)*

Description

Measurement on certain aspects of human lung capacity for 72 patients on 4 instrument-operative combination, i.e. two different instruments and two different users, a skilled one and a new one.

Usage

```
data(VitCap)
```

Format

A data frame with 288 observations on the following 5 variables.

`meth` a factor with levels `StNew`, `StSkil`, `ExpNew` and `ExpSkil`, representing the instrument by user combinations. See below.

`item` a numeric vector, the person ID, i.e. the 72 patients

`y` a numeric vector, the measurements, i.e. vital capacity.

`user` a factor with levels `New` `Skil`, for the new user and the skilled user

`instrument` a factor with levels `Exp` and `St`, for the experimental instrument and the standard one.

Source

V. D. Barnett, Simultaneous Pairwise Linear Structural Relationships, Biometrics, Mar. 1969, Vol. 25, No. 1, pp. 129-142.

Examples

```
data(VitCap)
Vcap <- Meth( VitCap )
str( Vcap )
plot( Vcap )
```

Index