Statistical Analysis of Method Comparison Studies

Haukeland University Hospital, Bergen, Norway
19–20 March 2014
http://BendixCarstensen.com/MethComp/Courses/Bergen.2014
Version 2.1

Compiled Thursday 20th March, 2014, 08:01
from: C:/Bendix/undervis/MethComp/Bergen.2014/pracs/pracs.tex

Bendix Carstensen Steno Diabetes Center, Gentofte, Denmark
& Department of Biostatistics, University of Copenhagen
bxc@steno.dk
http://BendixCarstensen.com

Contents

Program

Wednesday 19 March 2014

10:30 – 11:15	**Lecture 1**: Welcome and introduction. Simple comparisons of measurement methods. Introduction to computing.
11:15 – 11:30	**Morning Tea**
11:30 – 12:00	**Practical 1**: Limits of agreement, Bland-Altman-plots: • Milk • Plasma volume.
12:00 – 12:30	**Lecture 2**: Linear bias between methods. Variable SD.
12:30 – 13:15	**Lunch**
13:15 – 14:00	**Lecture 3**: Replicate measurements — exchangeable or linked? Allocation of sources of variation.
14:00 – 15:00	**Practical 2**: Data with replicate measurements by each method: • Fat. • Systolic blood pressure.
15:00 – 15:30	Coffee break
15:30 – 15:50	**Lecture 4**: Repeatability, reproducibility Coefficient of variation
15:50 – 16:20	**Lecture 5**: A general model for method comparisons. Linear relationship between methods.
16:30 – 18:00	**Practical 3**: • Oximetry data.

Thursday 20 March 2014

09:00 – 09:30	Recap of Wednesday.
09:30 – 10:15	**Lecture 6:** Converting between methods. Variance components. Transformations.
10:15 – 10:30	Coffee break
10:30 – 11:45	**Practical 4**: • Oximetry data — transformation. • Analyzing your own data. . . ?
11:45 – 12:00	Summary and closure.

Chapter 1

Introduction to computing

This course is both theoretical and practical, i.e. the aim is to convey a basic understanding of the problems in method comparison studies, but also to convey practical skills in handling the statistical analysis.

The practicals assume that you bring your own laptop with R on it.

In the following is a brief overview of the software and other files you must download.

1.1 Software: R

The most convenient software for desktop calculator type of calculations and simulation as well as simple statistical computing is the free software package R for statistics and graphics. R can be extended with *packages* that contain extra functions. The more advanced models covered in this course are only implemented in R in the `MethComp` package. Hence we shall assume that you have the latest version of R on your computer.

1.1.1 Installation

R can be obtained from cran.r-project.org. Download the appropriate version to your computer, and run this installation file.

Then fire up R, and at the command prompt type:

```
> install.packages( c("coda","rjags","Epi") )
```

This will install the mentioned packages provided you are connected to the net. Alternatively you can click in Packages→Install package(s), and choose the packages from the menu it brings up.

`Epi` is a package designed for epidemiological use. It contains some functions for display of estimates that may be useful, but is otherwise not essential for this course.

1.1.2 Installing the `MethComp` package

Finally you will have to install the `MethComp` which contains functions for analysis of method comparison studies. It is available from http://BendixCarstensen.com/MethComp/Archive/?C=M;O=D — this link should bring up the latest version of the package at the top of the display. Download the zip-file and then from the menu select Packages→ Install package(s) from local zip files

1.1.3 MCMC sampling in the `MethComp` package

The function `MCmcmc` in the `MethComp` package uses Markov chain simulation (MCMC) for estimation; it is recommended that you use the platform-independent `JAGS` program, available from http://mcmc-jags.sourceforge.net/. Once you have installed `JAGS` it and the `rjags` package the `MCmcmc` function should work.

Alternatively, you may choose to use `BRugs` or `WinBUGS` for the MCMC-sampling using the argument `program=` in `MCmcmc`, if you have installed either `openBUGS` or `WinBUGS`. These versions are however slightly slower for the models implemented in `MCmcmc`, and `WinBUGS` requires that you specify the installation folder for it.

1.2 The `MethComp` package

The purpose of the `MethComp` package is to provide computational tools to manipulate, display and analyze data from method comparison studies. The package requires a particular structure of data.

1.2.1 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 the replicate measurements listed in the same line 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`.

1.2.1.1 Meth

There is a class, "`Meth`" for this kind of data frame. A data frame is converted to a `Meth` object by using the `Meth` function on it. Objects of class `Meth` (which inherits from the class `data.frame`) has specific methods such as `summary`, `plot`, `subset` and `transform` (the latter two only to keep the class attribute). The analysis functions mostly do not require the data to be in `Meth` format — if a data frame with the right columns is supplied, it is converted internally. There are several ways of creating a data frame of class `Meth` from an existing data frame — see the documentation for the function `Meth`.

1.2.1.2 MethComp

Results from an analysis with estimated conversions between methods and (if applicable) variance components. Produced by different functions.

1.2.1.3 MCmcmc

Results from an MCMC analysis of a model by `MCmcmc`. Can be converted to a `MethComp` object, for more handy printing and plotting of essential results.

1.2.2 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 manual or by consulting the help pages on the fly by typing e.g.:

```
> ?plot.Meth
```

which will bring up the manual page for the function `plot.Meth`. The example code from the manual page can be run directly by:

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

Neither here nor in the text for the practicals are extensive descriptions of how to use the functions in the package. It is therefor essential that you (repeatedly!) consult the help pages for the functions as described above.

1.2.2.1 Graphical functions

`plot.Meth` Plots all methods against all others, both as a scatter plot and as a Bland-Altman plot.

`BA.plot` Makes a Bland-Altman plot of two methods from a data frame with method comparison data, and computes limits of agreement. Alternatively, it can plot the two methods against each other in a scatter plot.

`plot.MethComp` Plots estimated conversion lines with prediction limits between methods, based on a `MethComp` object, which is an estimated method comparison model.

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

1.2.2.2 Data manipulating functions

`make.repl` Generates (or replaces) a `repl` column in a data frame with columns `meth`, `item` and `y`.

`perm.repl` Randomly permutes replicates within (method,item) and assigns new replicate numbers. Useful when plotting data from experiments with exchangeable replicates.

`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,replicate).

`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 number of 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.

1.2.2.3 Analysis functions

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

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

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

`AltReg` Estimates via alternating regressions in the general model. Returns estimates of mean conversion parameters and variance components. The fitting algorithm is not terribly efficient, so it is advisable to use the argument `trace=T` to make sure that something actually is happening.

`MCmcmc` Estimates via `BUGS` in the general model with non-constant bias. Produces a `MCmcmc` object, which is an `mcmc.list` object with some extra attributes. `mcmc.list` objects are handled by the `coda` package, so this is required when calling `MCmcmc`.

1.2.2.4 Reporting functions

The functions `DA.reg`, `BA.est` or `AltReg` return objects of class `MethComp`. This is the core type of output in the `MethComp` package, it contains all relevant information on the relationship between the methods.

The function `MCmcmc` returns an object of class `MCmcmc`, which can be converted to a `MethComp` object by the MethComp function by e.g.:

```
> MCox <- MCmcmc( ox, random=c("mi","ir"), n.iter=5000 )
> mcox <- MethComp( MCox )
```

The reason for this is that an `MCmcmc` objects contains a lot of additional information about the MCMC fitting, which is necessary in order to check the proper convergence of the MCMC sampling.

`print.MethComp` Prints a table of conversion equations between methods analyzed, with prediction standard deviations. Also gives summaries of the posteriors for the parameters that constitute the conversion algorithms.

`plot.MethComp` Plots the conversion lines between methods with prediction limits. There are also `points` and `lines` functions that will add the observations and the conversion line with prediction limits.

`post.MCmcmc` Plots smoothed posterior densities for the estimates. This is primarily of interest for the variance component estimates, but it has arguments to produce the posterior distribution of the parameters of the mean conversion between methods.

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

Chapter 2

Practicals

2.1 Milk: Single measurements by two methods

First, load the `MethComp` package, and check what version of R and the package you actually have, and *then* load the dataset and take a look at its structure:

```
> library( MethComp )
> sessionInfo()
> data(milk)
> str(milk)
> head(milk)
```

The data is arranged in the long form, i.e. with one measurement per line and two variables, item and method. Using the `to.wide` function puts the data in a more familiar format for inspection. Try:

```
> mw <- to.wide( milk )
> str( mw )
> head( mw )
```

1. Plot the two sets of measurements against each other, e.g. by using the two variables from the dataset in the wide form.

2. To get an overview of the relationship you can exploit the fact that the dataset has variables `item`, `meth` and `y` and convert it to a `Meth` object. Then you can use the facilities for a `Meth` object. Try:

   ```
   > milk <- Meth( milk )
   > summary( milk )
   > plot( milk )
   ```

   You will want to consult the help page for the `plot.Meth` function to see the possible arguments to enhance the plot.

3. You can also be more explicit about the Bland-Altman comparison between the two methods:

   ```
   > BA.plot( milk )
   > BA.plot( milk, diflim=c(-1,1)/2 )
   ```

You will want to have a look at the help page for `BA.plot`, to see the possible arguments to `BA.plot` used to control the look of the plot.

4. What are the limits of agreement between the two methods?

5. Formulate in plain words what this means. Remember to explicitly state which method is subtracted from the other.e

6. Inspect the plot and try to assess whether the assumptions underlying the reporting of limits of agreement are fulfilled. (*Hint:* Try to regress the differences on the averages, and translate the resulting regression equation to a linear relationship between the two methods. You might want to consult the `DA.reg` function).

7. Fit the two regression lines (i.e. regress `Gerber` on `Trig` and vice versa) and show them in a plot of the two methods:

```
> summary( lm( Trig ~ Gerber, data=mw ) )$coef
> summary( lm( Gerber ~ Trig, data=mw ) )$coef
```

   How do they relate to the equation derived from the regression of the difference on the average?

8. Finally, try to make a regression allowing for errors in both variables, the so-called Deming regression:

```
> with( mw, Deming( Trig, Gerber ) )
```

   Compare this with the relationship derived from the regression of the difference on the average.

9. Use the results to provide an improved prediction equation for Gerber based on a measured value by Trig.

2.2 Plasma volume: Single measurements by two methods

The `plvol` data from the `MethComp` package contains measurements of plasma volume is expressed as a percentage of the expected value for normal individuals. First load the `MethComp` package and the dataset with the measurements:

```
> library( MethComp )
> data( plvol )
```

1. Plot the the measurements from the two methods against each other. You may want to transform the datyaset to the "wide" form using `to.wide()`

2. Make a Bland-Altman plot and compute the limits of agreement. Try:

```
> BA.plot(plvol)
```

Are these limits a reasonable summary of the data?

3. Now try to make a log-transform of the data and re-do the analysis. *Hint:* You may use the `mult=TRUE` option to `BA.plot` to achieve this:

```
> BA.plot( plvol, mult=TRUE )
```

4. Does the log-transform give a better description of data? Hint: Explore the variance homogeneity by using the function `DA.reg`.

5. Formulate a conclusion for the data in plain words, based on the log-transformed analysis.

2.3 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. By the vary mature of the experiemnt, the replicates are exchangeable.

1. Load the data frame `fat` and examine the names in the data frame:

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

   Then use `Meth` to convert it to a form that comply with that required by the functions in the `MethComp` package for analyzing the measurements of visceral fat between the two observers. You will need to look closely at the arguments of `Meth` to accomplish this. You could for example do something like:

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

2. Plot the two methods against each other, using the replicate number for pairing the measurements; you would use the function `to.wide` to get the data in a form so that you can plot them. Alternatively you can try out the function `plot.Meth` directly on the `Meth` object — you just need to use `plot` on the object, R will automatically invoke `plot.Meth` when the argument is of class `Meth`.

3. Since replicates are exchangeable *within* (method, item) we should get the same sort of overview of the data after a random permutation of the replicates. Try plotting the data using the original replicate numbers for pairing and then a random permutation created by the `perm.repl` function:

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

4. Now use `BA.plot` to produce a Bland-Altman plot and compute the limits of agreement using the pairing of replicates across methods based on the numbering of replicates. What are the limits of agreement computed this way?

5. The assumptions behind the limits of agreement is that the difference between methods is constant and that the variation is constant across the range of observations. This can be formally tested by regressing the differences on the averages and after that regressing the absolute values of the residuals on the means. Try to use the `DA.reg` function (again using the existing pairing of replicates) to do this. Explore how this changes by permutation of the replicates.

6. Now set up a proper variance component model to accommodate the actual replication structure of the data. Remember to indicate the exchangeability structure of the data when calling `BA.est`, by using the argument `linked=FALSE`.

7. From `BA.est` you will get the coefficient of reproducibility for each of the methods; this is an upper 95% confidence limit for the absolute difference between two measurements by the same method on the same item. Does this differ between methods?

8. Compare the limits of agreement obtained from the naïve approach using replicates as items with the correct one using the proper model. Read the help page for `BA.plot` carefully, particularly forz the argument `model=`.

9. Finally, try to see what happens if you base the limits of agreement on the means over the averages. You can use the function `mean.Meth` to transform a `Meth` object with replicates to one with only the mean over the replicates, so using for example:

```
> BA.plot( mean(vis) )
```

2.4 Systolic blood pressure: Linked replicates by two methods

The dataset with systolic blood pressure measurements is taken from table 1 in [**?**], where a more detailed description can be found.

1. Load the systolic blood pressure data from the `MethComp` package, and take a look at the data using `?sbp`, `str()`:

```
> data( sbp )
> str( sbp )
```

   Since the colums have the right names you can easily turn the data-frame into a `Meth` object and inspect it using `plot()`:

```
> sbp <- Meth( sbp )
> str( sbp )
> plot( sbp )
```

   What is the impression of the relationship between the methods, and their relative uncertainty?

2. We want to restrict our attention to the comparison of the two manual methods (J and R), but still using the replicate measurements. Are the replicates exchangeable within method and item? Make a Bland-Altman plot of the data for the two manual methods, and derive the limits of agreement, e.g.:

```
> sbp <- subset( sbp, meth %in% c("J","R") )
> BA.plot( sbp )
```

   Try to use the argument `diflim=` (the meaning of this is found on the help page for the function `BA.plot`). How does the use of the replicates for this Bland-Altman plot and limits of agreement correspond to the exchangeability structure of data?

3. (Optional, somewhat esoteric) Now fit a proper variance component model, assuming linked replicates:

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

   The releveant code to make `lme` to fit the right model is here:

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

   Note that the model cannot be fitted by `lmer`, since we must have different residual variances for the two methods. What is the average bias between methods, and what is the estimated size of the variance components?

4. An easier way to get the relevant estimates is to use the wrapper `BA.est` — remeber to consult the help page first.

5. What is the limits of agreement between observers R and J? Give a precise interpretation of this.

6. How can you — solely based on data — find out whether replicates are linked or exchangeable?

7. Compute limits of agreement based on the variance components from the model for the entire datset. How does these agree with those just based on methods R and J?

8. How do these limits of agreement compare to those obtined by just treating replicates as items (as done by the function `DA.reg`) ?

9. Formulate the result as a 95% prediction interval for a measurement by method R given a a measurement by method J, y_{J}, say.

2.5 Oximetry: Linked replicates and non-constant bias

1. Start by loading the dataset and take a look at its structure:

```
> library(MethComp)
```

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

The dataframe is already in the correct form for use with the `MethComp` package, with variables named `item`, `meth`, `repl` and `y`, but it would more convenient to convert it to a `Meth` object:

```
> ox <- Meth(ox)
> summary( ox )
```

How may replicates are there on each child?

2. Now plot the two sets of measurements against each other using the `plot.Meth` function (remember that when we have turned the dataframe into a `Meth` object, then `plot` will automatically invoke the `plot.Meth` function:

```
> plot( ox )
```

3. Use the `BA.plot` function to generate a Bland-Altman plot of the data. What is the estimated average difference between measurements from the two methods? What are the limits of agreement between the two methods?

```
> BA.plot(ox)
```

Are these limits large compared to the average oximetry measure and the range of the data?

4. The Bland-Altman procedure for generating the limits of agreement is based on a model with constant bias. Moreover, it does not divide the variation between different sources. With replicate measurements we can allocate the variation to the different sources using a variance component model:

   - method by item ("matrix" effect).
   - item by replicate (variation between linked sets).
   - residual variation for each method.

The model can be fit by using the function `BA.est()`:

```
> BA.est(ox)
```

Make sure that you understand what each of the variance components mean.

5. Note that the `MxI` variance components are the same for `CO` and `pulse` since separate parameters cannot be estimated when there are only two methods. Compare the magnitude of the `IxR` variance component for the item by replicate effect to both the `MxI` variance component for the method by item effect and the residuals variances. Is this what you would expect given that the replicates are linked?

6. Give a confidence interval for the absolute difference between two repeat measuremnts by the same method.

7. Now expand the model allowing for non-constant bias, i.e. by a linear relationship between the methods. Use the `AltReg` function to estimate in this model. How do the variance components change?

8. You can get an approximate assessment of whether the slopes are different from 1 by regressing the differences between the linked replicates on the averages, and testing whether the slope is 0. Likewise, we can approximately assess whether the variance is constant across the range of the measuremnts by regression the absolute values of the residuals from this regression on the averages. Both of these are implemented in the function `DA.reg`. What is the conclusion of this analysis?

9. One of the drawbacks of using the `BA.est` or `AltReg` functions is that we do not get standard errors or confidence intervals for the estimated variance parameters. The `MCmcmc` function produces summaries of the posterior distribution of estimated parameters in a Bayesian setup. You must use the argument `bias="const"` in the call to `MCmcmc` to fit a model with constant bias:

   ```
   > ox.mi.ir <- MCmcmc( ox, random=c("mi","ir"), n.iter=5000, bias="const")
   ```

   Summarize the results by using the `print` function on the resulting `MCmcm` object `ox.mi.ir`:

   ```
   > print(ox.mi.ir)
   ```

10. You can get a summary of the results from the function by converting it to a `MethComp` object — this is an object that is designed to hold results from method comparisons - that is intercepts and slopes as well as variance components. Use the `plot` function for `MCmcmc` objects to produce a scatterplot displaying the linear equations relating one method to the other (recall that the slope has been constrained to be 1):

    ```
    > plot( ox.mi.ir, pl.obs=TRUE )
    ```

    Use the `post.MCmcmc` function to display smoothed posterior densities for the variance components separately for each method (although only the residual variances differ between methods):

    ```
    > par(mfrow = c(2,1))
    > post(ox.mi.ir)
    ```

    Are the residual variances equal?

2.6 Oximetry: Transformation

In the first exercise on the oximetry data, we just used the original ys, measured in percent, as the response variable. We also saw taht on this scale there was in indication of heteroschedasticity while there was little indication that the bias was non-constant. However, since the measurements are in percent, it would be natural to apply a transformation to the data before doing the analysis. This exercise is a continuation / replication of the previous using a transformation of the measurements.

1. First, get the data and take a look at the data without transformation:

```
> data( ox )
> ox <- Meth( ox )
> plot( ox )
```

2. Now, transform the measurements by the logit-transform of the percentages (remember that these are numbers between 0 and 100):

```
> oxt <- transform( ox, y=log(y/(100-y)) )
> plot( oxt )
```

3. Make a quick check of the assumptions underlying the LoA; constant bias and variance by using the `DA.reg` function:

```
> DA.reg( oxt )
```

   What is the conclusion?

4. Now compute the limits of agreement on the logit-scale, based on the model assuming constant bias, using the correct model for linked replicates:

```
> BAoxt <- BA.est( oxt )
> BAoxt$LoA
```

   How would you interpret these limits of agreement in terms of the original data?

5. Try to transform the LoA to the odds-ratio scale (that is the fraction of saturation to non-saturation — admittedly somewhat odd (!)), and use this to make a Bland-Altman plot with an interpretable scale. How do you find the interpretability of the plot?

6. The natural thing would instead be to present LoA and the Bland-Altman plot on the original scale. That is to take the analysis on the logit scale and show the conversion between methods on the original scale by back-transformation. Since this a more general problem, there is an automatic transformation mechanism in most functions from the `MethComp` package. This is activated by the `Trans=` argument. You can see the details of this on the help page for the function `choose.trans` — just type:

```
> ?choose.trans
```

So try the transformed analysis (and check the constancy of the variance:

```
> DA.reg( ox, Trans="pctlogit" )
> BAox <- BA.est( ox, Trans="pctlogit" )
```

Use the `plot` method for `MethComp` objects (which is what is returned by `BA.est`, to show both a Bland-Altman plot and a plot to convert measuremnst between the two methods.

7. Two other frequently used transformations of proportions are the log−log transform and the complementary log−log transform:

$$\text{loglog}(p) = \log(-\log(p)) \qquad \text{cloglog}(p) = \log(-\log(1-p))$$

Try to use these transformations, and show the conversions between methods. (Hint look at the help page for `choose.trans`. Which of the transformations would you prefer — and on what grounds?

8. So far we have only considered models with constant bias, and it would be prudent to check whether the bias between methods on the logit scale is actually constant. Such an analysis is parallel to the one we did on the original scale, using either the `AltReg` or the `MCmcmc` functions. Do the analysis using one of these approaches and see how it differs from the prediction limits based on the constant-bias for logits.

Chapter 3

Solutions to exercises

3.1 Milk: Single measurements by two methods

First we load the `MethComp` package so we have access to the dataset and then take a look at its structure:

```
> library(MethComp)
> data( milk )
> str( milk )
  'data.frame':        90 obs. of  3 variables:
   $ meth: Factor w/ 2 levels "Gerber","Trig": 2 2 2 2 2 2 2 2 2 2 ...
   $ item: int  1 2 3 4 5 6 7 8 9 10 ...
   $ y   : num  0.96 1.16 0.97 1.01 1.25 1.22 1.46 1.66 1.75 1.72 ...

> head( milk )

    meth item    y
  1 Trig    1 0.96
  2 Trig    2 1.16
  3 Trig    3 0.97
  4 Trig    4 1.01
  5 Trig    5 1.25
  6 Trig    6 1.22
```

The data is arranged in the long form, i.e. with one measurement per line and two variables, item and method. Using the `to.wide` function puts the data in a more familiar format:

```
> mw <- to.wide( milk )
> str( mw )
  'data.frame':        45 obs. of  4 variables:
   $ item  : Factor w/ 45 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
   $ repl  : Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
   $ Gerber: num  0.85 1 1 1 1.2 1.2 1.38 1.65 1.68 1.7 ...
   $ Trig  : num  0.96 1.16 0.97 1.01 1.25 1.22 1.46 1.66 1.75 1.72 ...

> head( mw )

    item repl Gerber Trig
  1    1    1   0.85 0.96
  2    2    1   1.00 1.16
  3    3    1   1.00 0.97
  4    4    1   1.00 1.01
  5    5    1   1.20 1.25
  6    6    1   1.20 1.22
```

1. We plot the two sets of measurements against each other, using the two variables from the dataset in the wide form:

```
> par( mgp=c(3,1,0)/1.6, mar=c(3,3,1,1) )
> with( mw, plot( Trig ~ Gerber, pch=16,
+                 xlim=range(milk$y),
+                 ylim=range(milk$y) ) ) # Note: identical axes
> abline(0,1)
```

   The last statement above just adds the identity line to the plot.

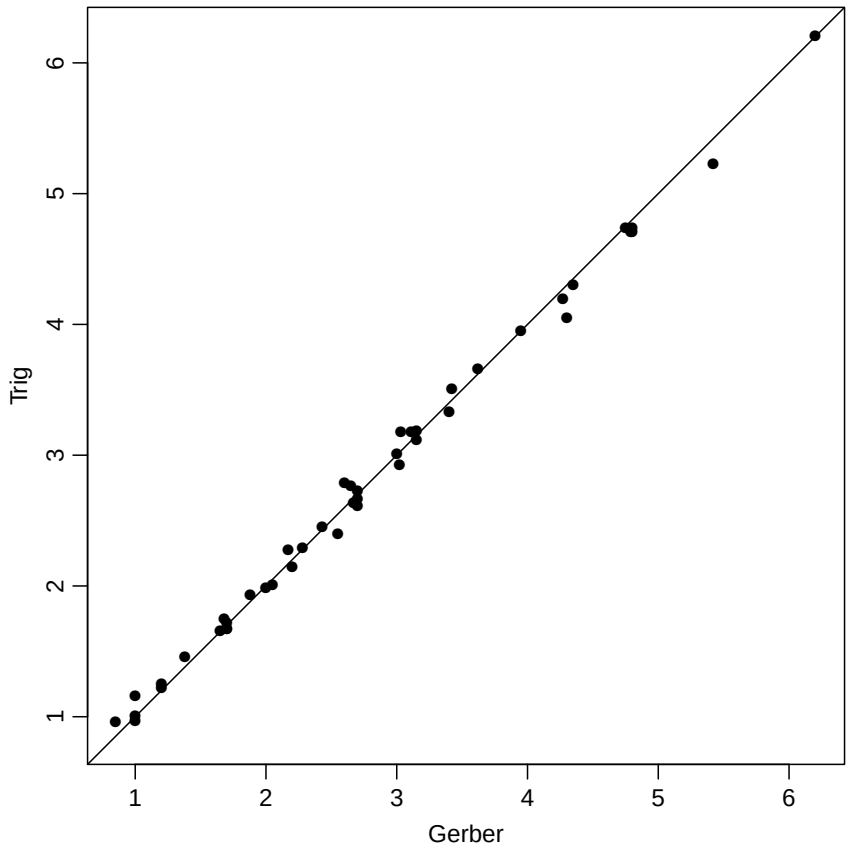

Figure 3.1: *Scatter plot of the* `milk` *data.*

2. Exploiting that the `milk` dataset has variables `item`, `meth` and `y`, we can without further ado convert it to a `Meth` object and then use the facilities for that:

```
> summary( milk )

      meth         item           y
 Gerber:45   Min.   : 1   Min.   :0.850
 Trig  :45   1st Qu.:12   1st Qu.:1.728
             Median :23   Median :2.670
             Mean   :23   Mean   :2.804
             3rd Qu.:34   3rd Qu.:3.487
             Max.   :45   Max.   :6.210

> milk <- Meth( milk )
```

```
  The following variables from the dataframe
  "milk" are used as the Meth variables:
  meth: meth
  item: item
     y: y
           #Replicates
  Method            1 #Items #Obs: 90 Values:  min  med  max
    Gerber         45     45       45         0.85 2.67 6.20
    Trig           45     45       45         0.96 2.67 6.21

> str( milk )

  Classes 'Meth' and 'data.frame':        90 obs. of  4 variables:
   $ meth: Factor w/ 2 levels "Gerber","Trig": 2 2 2 2 2 2 2 2 2 2 ...
   $ item: Factor w/ 45 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
   $ repl: Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
   $ y   : num  0.96 1.16 0.97 1.01 1.25 1.22 1.46 1.66 1.75 1.72 ...

> summary(milk)

           #Replicates
  Method            1 #Items #Obs: 90 Values:  min  med  max
    Gerber         45     45       45         0.85 2.67 6.20
    Trig           45     45       45         0.96 2.67 6.21

> par( mgp=c(3,1,0)/1.6 )
> plot( milk, var.names=TRUE )
```

Note the use of the `var.names=` argument to annotate the individual panels with the variable names to avoid confusion of what is on the axes.

3. We can get a proper Bland-Altman plot with a explicit calculation of the limits of agreement:

```
> zz <- BA.plot( milk )
```

or in a slightly nicer form:

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

Note that we put the result from `BA.plot` in an object, this gives basically the output from `DA.reg`:

```
> str( zz )

  List of 3
   $ Conv   : num [1:2, 1:2, 1:12] 0 -0.000222 0.000222 0 1 ...
    ..- attr(*, "dimnames")=List of 3
    .. ..$ To:  : chr [1:2] "Gerber" "Trig"
    .. ..$ From:: chr [1:2] "Gerber" "Trig"
    .. ..$      : chr [1:12] "alpha" "beta" "sd.pr" "beta=1" ...
   $ VarComp: NULL
   $ data   :Classes 'Meth' and 'data.frame':        90 obs. of  4 variables:
    ..$ meth: Factor w/ 2 levels "Gerber","Trig": 2 2 2 2 2 2 2 2 2 2 ...
    ..$ item: Factor w/ 45 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8 9 10 ...
    ..$ repl: Factor w/ 1 level "1": 1 1 1 1 1 1 1 1 1 1 ...
    ..$ y   : num [1:90] 0.96 1.16 0.97 1.01 1.25 1.22 1.46 1.66 1.75 1.72 ...
   - attr(*, "class")= chr [1:2] "MethComp" "DA.reg"
   - attr(*, "RandomRaters")= logi FALSE
   - attr(*, "pl.type")= Named chr [1:3] "BA" "const" "const"
    ..- attr(*, "names")= chr [1:3] "pl.type" "dif.type" "sd.type"
```

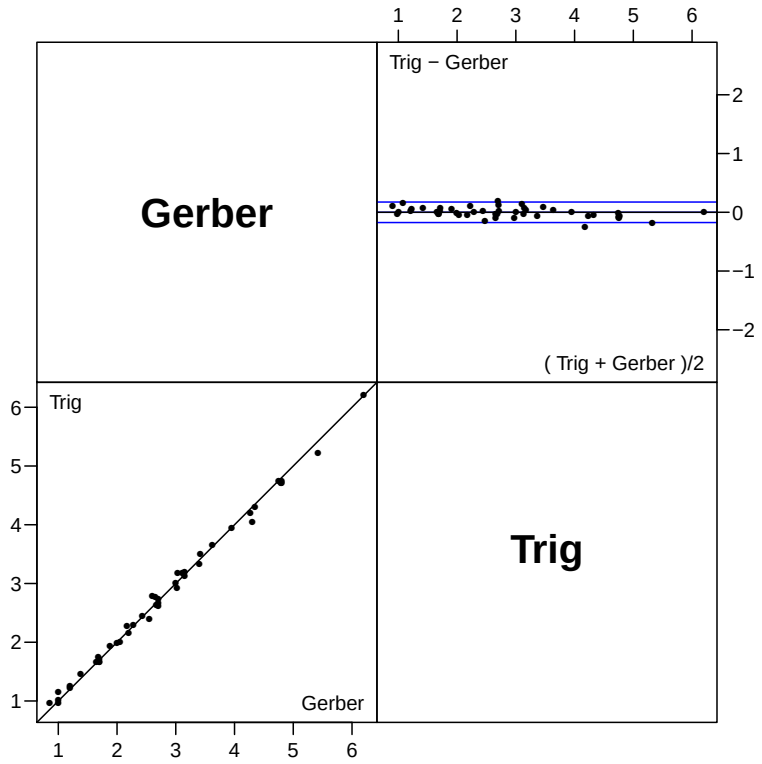

Figure 3.2: *Overview plot of the* `milk` *data, using* `plot.Meth()`*, i.e. the generic plotting method for* `Meth` *objects.*

```
> round( ftable( zz$Conv ), 3 )

                alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K LoA-
To:     From:
Gerber  Gerber  0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA     N
        Trig    0.000  1.000  0.087  1.000   0.000   0.000   0.087  0.058  0.008  0.317 -0.17
Trig    Gerber  0.000  1.000  0.087  1.000   0.000   0.000  -0.087  0.058  0.008  0.317 -0.17
        Trig    0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA     N
```

4. From the figure and the printout, we see that the limits of agreement are $(-0.17, 0.17)$g/100 ml.

5. . . . which means that the difference between a pair of future measurements by Gerber and Trig with 95% probability will be between -0.17 and 0.17 g/100ml.

6. The Bland-Altman plot looks very nice with an average that is very flat. However, regressing the differences on the averages gives:

```
> summary( lm( I(Gerber-Trig) ~ I((Gerber+Trig)/2), data=mw ) )$coef

                       Estimate Std. Error   t value    Pr(>|t|)
(Intercept)          -0.07904017 0.02906123 -2.719781 0.009386433
I((Gerber + Trig)/2)  0.02827097 0.00944454  2.993367 0.004559424
```

although the resulting relationship is not impressively different from a horizontal slope, the slope *is* significantly different from 1. In general we have:

$$y - x = \alpha + \beta\left(\frac{x+y}{2}\right) \qquad \Leftrightarrow \qquad y = \frac{\alpha}{1-\beta/2} + \left(\frac{1+\beta/2}{1-\beta/2}\right)x$$

so the regression coefficients of the difference on the mean ($\alpha = -0.079, \beta = 0.028$) implies the relationships:

$$\begin{aligned} \text{Gerber} &= -0.079/(1-0.014) + (1+0.014)/(1-0.014)\text{Trig} = -0.080 + 1.029\text{Trig} \\ \text{Trig} &= 0.078 + 0.972\text{Gerber} \end{aligned}$$

Note that this type of regression is tantamount to minimizing the squared deviations orthogonal to the identity line, and *not* orthogonal to the regression line.

This relationship can be obtained directly by the function `DA.reg`, which regresses the differences on the averages and returns the relationships for the original variables, as well as approximate tests for the hypotheses of constant difference and constant standard deviation:

```
> DA.reg(milk)

   Conversion between methods:
                  alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To:    From:
Gerber Gerber   0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

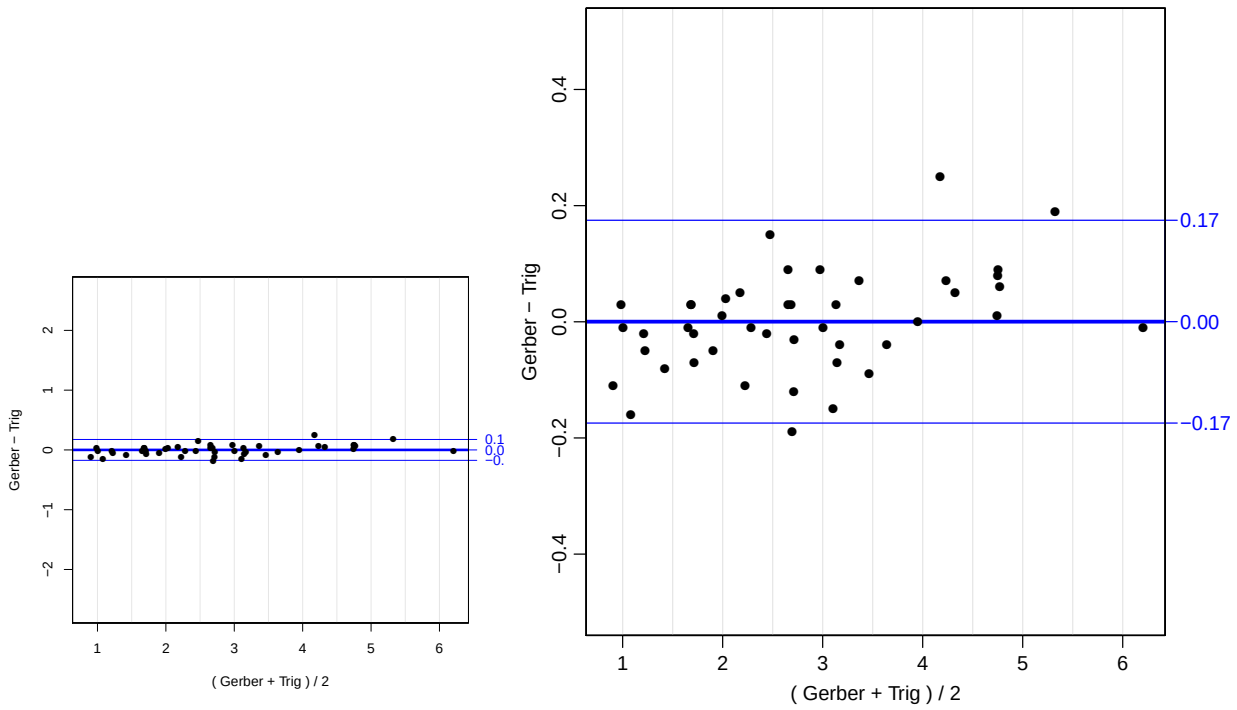

Figure 3.3: *Bland-Altman plots of the* `milk` *data, left panel with the same extent of the data on both axes (the default), the right one with explicitly defined y-axis and explicitly defined margins — note how the right hand margin on the left plot is too narrow to accommodate the annotation of the LoA in full.*

```
        Trig    -0.080  1.029  0.081  0.005  -0.079   0.028   0.080  0.059  0.006  0.383
Trig    Gerber   0.078  0.972  0.079  0.005   0.079  -0.028  -0.080  0.059  0.006  0.383
        Trig     0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

The `alpha` and `beta` columns are intercept and slopes relating the two methods based on the regression of the differences on the averages. The `sd.pred` is the prediction standard deviation derived from the this regression, $(\sigma/(1+\beta/2)$ and $\sigma/(1-\beta/2)$, respectively, where σ^2 is the residual variance from the regression of differences on means.

The range of the measurements is broadly speaking from 1 to 5 g/100ml, i.e. the contribution of the slope is about 0.15, largely in the same ballpark as the limits of agreement. Hence, if future measurements will be in this range too, the slope can hardly be ignored. Unless of course deviations less than some 0.4 g/100ml are considered irrelevant.

The columns labelled with `(t-f)` (Refers to "To" and "From") are intercept and slope of the regression of differences on averages, and the columns labelels with `(sd)` refer to the the intercept and slope of the SD as a function o fthe average, derived from regressing the absolute residuals on the average.

The column labeled `beta=1` is the p-value of the test for the null that the slope of the differences against the averages is 0, which is the same as the hypothesis that the lnear relationship between methods has slope 1. The column `sd=K` is the p-value for the null that the SD is constant as a fubction of the average between methods.

7. The two regression lines also show slopes significantly different from 1, with roughly the same slope as those derived from the regression of the differences on the averages, although this will not be the case in general.

```
> summary( lm( Trig ~ Gerber, data=mw ) )$coef

               Estimate  Std. Error    t value     Pr(>|t|)
(Intercept) 0.08308899 0.028301786   2.935821 5.323062e-03
Gerber      0.97028609 0.009174537 105.758594 1.323266e-53

> summary( lm( Gerber ~ Trig, data=mw ) )$coef

               Estimate  Std. Error    t value     Pr(>|t|)
(Intercept) -0.07456776 0.029801280  -2.502167 1.622649e-02
Trig         1.02667683 0.009707739 105.758594 1.323266e-53
```

We can plot the two lines using the function `bothlines`:

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> with( mw, plot( Trig, Gerber, pch=16, xlim=c(0,7), ylim=c(0,7) ) )
> with( mw, bothlines( Trig, Gerber ) )
```

8. A regression allowing for errors in both variables, is the so-called Deming regression which gives a result which is very close to that from the ordinary regression of the differences on the averages:

```
> with( mw, Deming( Trig, Gerber ) )
```

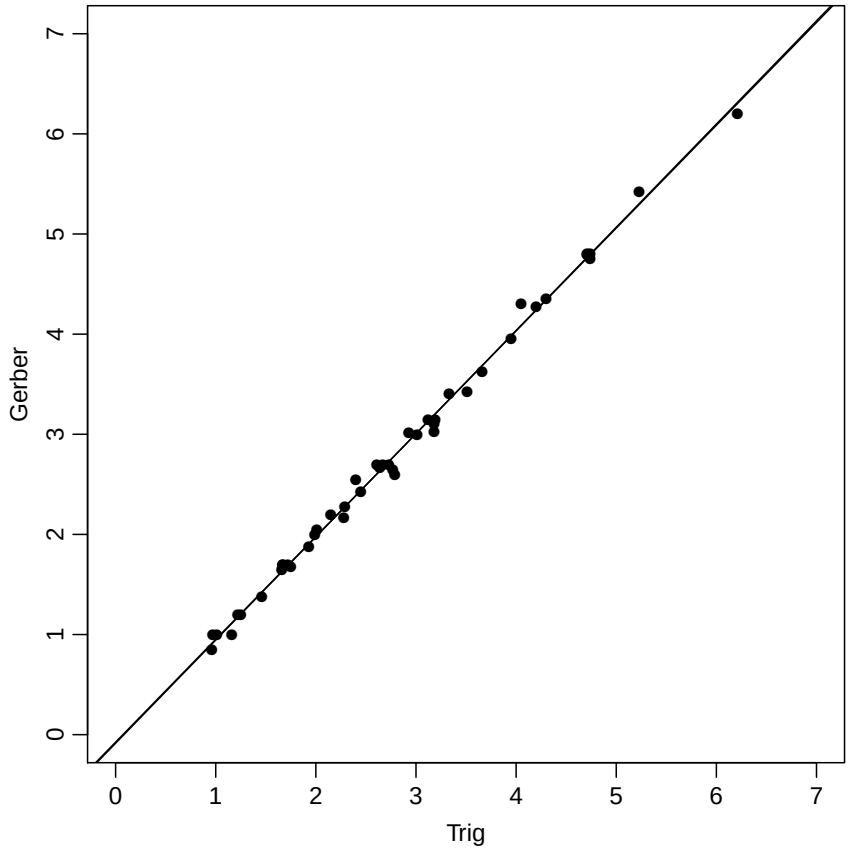

Figure 3.4: *Scatter plot of data with the two different regression lines. The two lines are practically indistinguishable.*

```
  Intercept       Slope   sigma.Trig sigma.Gerber
-0.08025171  1.02870424  0.05679647   0.05679647
```

Deming regression assumes that the ratio of the residual sd.s is known; the default for the `Deming` function is to assume that they are equal. Essentially this assumption has to be pulled out of thin air in the absence of replicate measurements.

9. The advantage of regression of the differences on averages is that it provides an estimate of the residual standard deviation, which can be used for construction of prediction limits. This calculation can be done using `BA.plot` (which uses `DA.reg` and `plot.MethComp`), the latter with the argument `reg.line=` — a number giving the number of decimals to be used for the display of the resulting conversion equations.

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> BA.plot( milk, dif.type="lin", eqn=TRUE, digits=3, diflim=c(-1,1)/2 )

  Relationships between methods:
   Gerber-Trig = -0.079+0.028(Gerber+Trig)/2 (0.080)
   Gerber = -0.080+1.029Trig (0.081)
   Trig = 0.078+0.972Gerber (0.079)
```

If we instead want a plot that shows a conversion of one method to the other we use the argument `pl.type="conv"`:

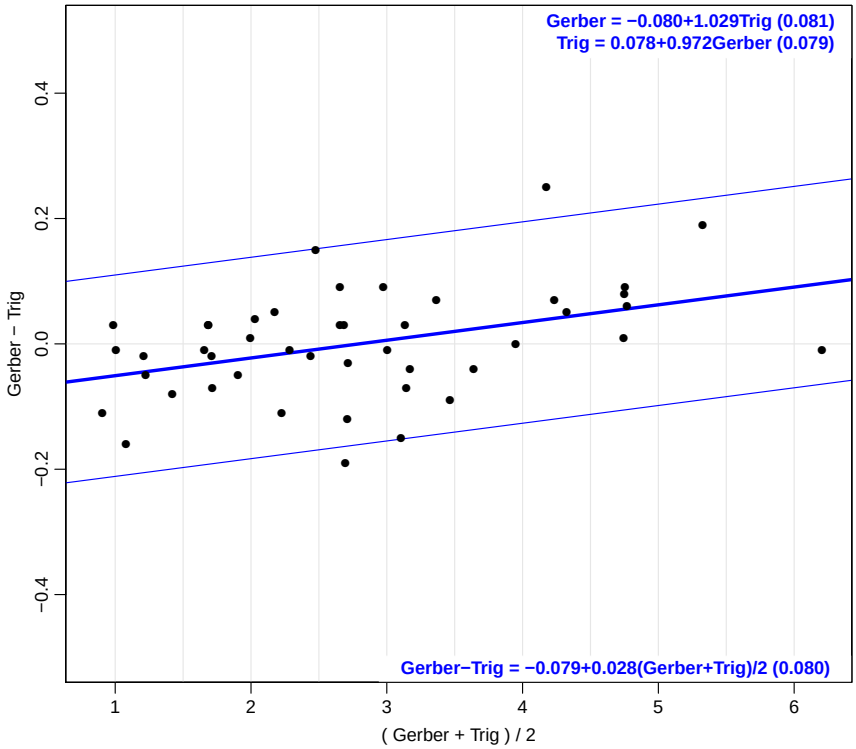

Figure 3.5: *Bland-Altman plot of the* `milk` *data with the regression of the differences on the averages and the resulting conversion equations between methods.*

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> BA.plot( milk, pl.type="conv", dif.type="lin", eqn=TRUE, digits=3 )

  Relationships between methods:
   Gerber-Trig = -0.079+0.028(Gerber+Trig)/2 (0.080)
   Gerber = -0.080+1.029Trig (0.081)
   Trig = 0.078+0.972Gerber (0.079)

> abline(0,1,col="red")
```

3.2 Plasma volume: Single measurements by two methods

We first load the `MethComp` package and the dataset `plvol` with the measurements of plasma volume is expressed as a percentage of the expected value for normal individuals.

```
> library( MethComp )
> data( plvol )
```

1. Measurements by the two methods are plotted against each other after transforming the dataset to the wide form:

```
> pw <- to.wide(plvol)
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> with(pw, plot( Hurley ~ Nadler, pch=16, xlim=range(plvol$y), ylim=range(plvol$y) ) )
> abline( 0,1 )
```

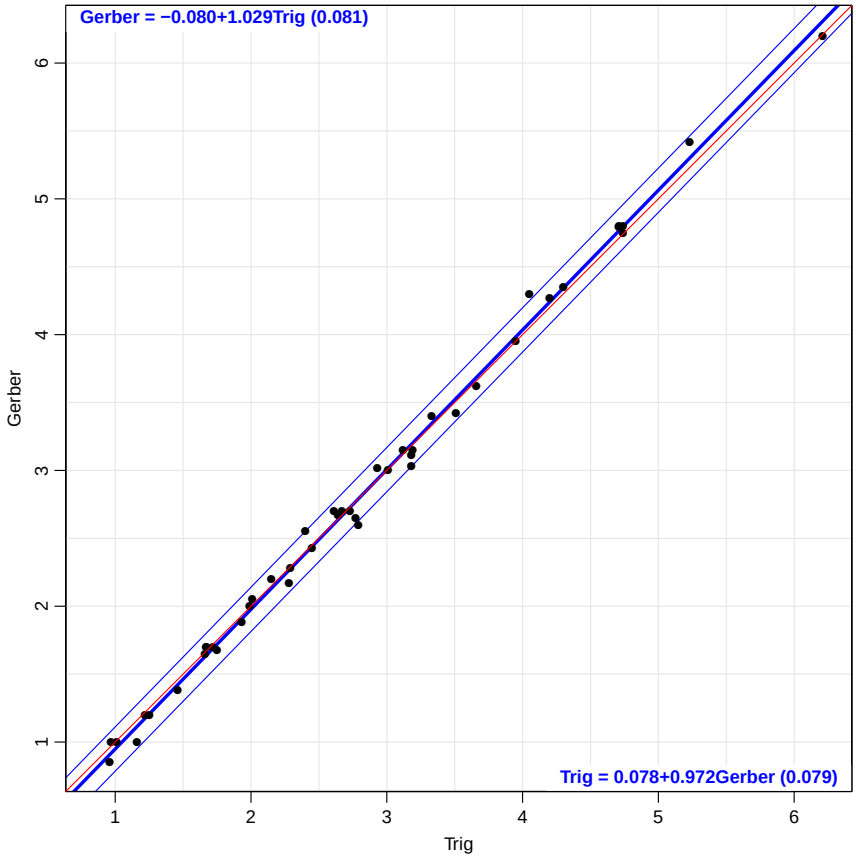

Figure 3.6: *Plot of the* `milk` *data with the regression of the differences on the averages shown as a conversion plot and the resulting conversion equations between methods.*

2. `BA.plot` produces a Bland-Altman plot and computes the limits of agreement (we use the `diflim` argument to get a sensible range on the y-axis — otherwise the extent is as the x-axis):

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( plvol, diflim=c(-20,5), h.grid=seq(-20,10,5) )
> abline( h=0 )
```

   Clearly, there is both a decreasing difference with increasing value of the plasma volume as well as an increase in variance with measurement level. Hence, these limits do not provide a reasonable summary of the data — they are much wider than necessary for a given level of the plasma volume.

3. If we log-transform the data and re-do the analysis we may get something more sensible. We can use the `mult=TRUE` option to `BA.plot` to achieve this in one go:

```
> BA.plot( plvol, mult=TRUE, diflim=c(0.8,1.05),
+          digits=2, h.grid=seq(80,110,5)/100 )
> abline( h=1, lwd=2 )
```

4. It is immediately apparent from the plot that the log-transform gives a much better description of data.

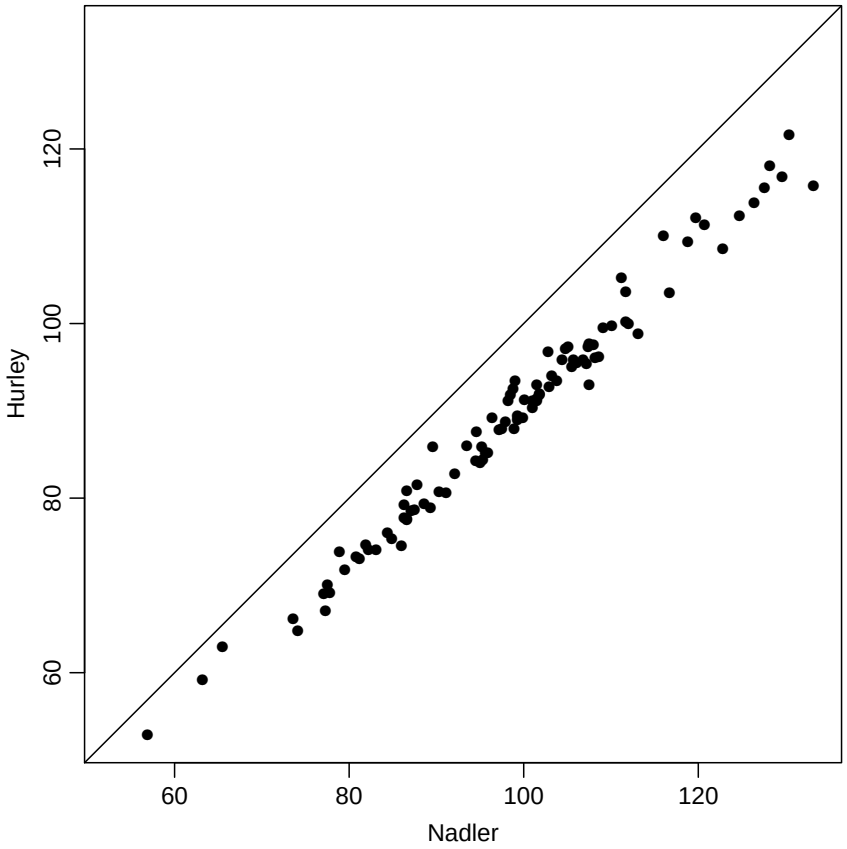

Figure 3.7: *Plot of two methods of measuring plasma volume.*

We can get a more formal assessment of the effect of the log-transform by using the `DA.reg` function:

```
> DA.reg( plvol )

   Conversion between methods:
                 alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To:    From:
Hurley Hurley    0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
       Nadler   -0.870  0.915  1.951  0.000  -0.908  -0.089   2.037  0.006  0.021  0.067
Nadler Hurley    0.951  1.093  2.132  0.000   0.908   0.089  -2.037  0.006  0.021  0.067
       Nadler    0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA

> DA.reg( plvol, Transform="log" )

  Note: Response transformed by:  .Primitive("log")


   Conversion between methods:
                 alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To:    From:
Hurley Hurley    0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
       Nadler   -0.113  1.003  0.022  0.818  -0.113   0.003   0.022  0.064 -0.009  0.389
Nadler Hurley    0.113  0.997  0.022  0.818   0.113  -0.003  -0.022  0.064 -0.009  0.389
       Nadler    0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

It is pretty clear that on th log-scale the difference between methos is constant (`beta=1`), as is the residual variance (`sd=K`).

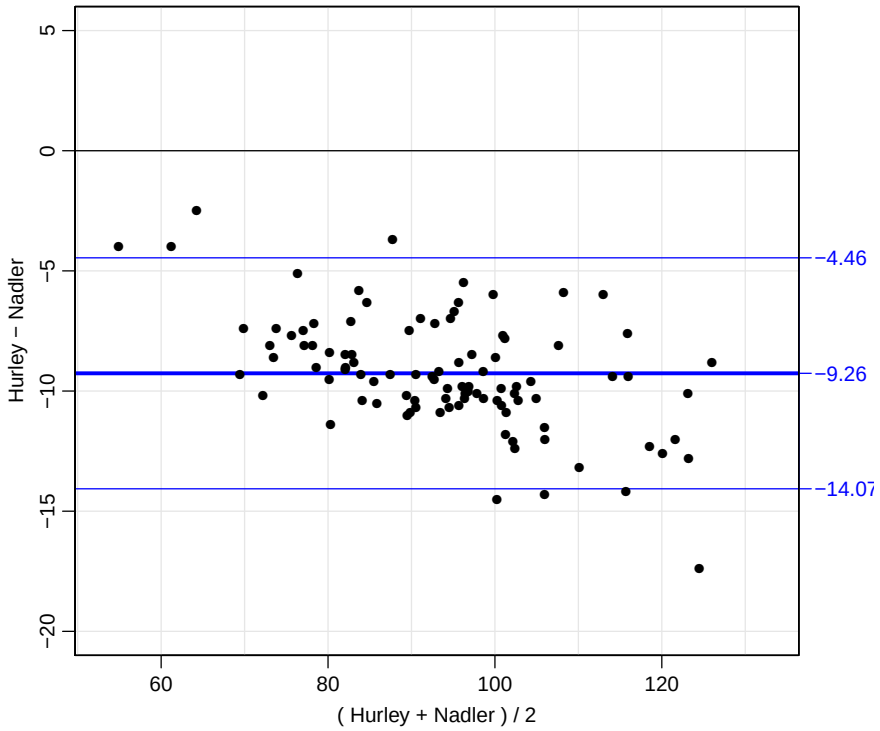

Figure 3.8: *Bland-Altman plot of two methods of measuring plasma volume.*

5. The estimated *ratio* between the Nadler and Hurley methods is 0.90 and the ratio of future measurements by the two methods is with 95% probability between 0.87 and 0.95. Another way of saying this is that for a given measurement by the Hurley method, the Nadler method will yield a measurement which is between 87 and 95% of this (with 95% probability).

3.3 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.

```
R version 3.0.2 (2013-09-25)
Platform: i386-w64-mingw32/i386 (32-bit)

locale:
[1] LC_COLLATE=Danish_Denmark.1252  LC_CTYPE=Danish_Denmark.1252
[3] LC_MONETARY=Danish_Denmark.1252 LC_NUMERIC=C
[5] LC_TIME=Danish_Denmark.1252

attached base packages:
[1] utils     datasets  graphics  grDevices stats     methods   base

other attached packages:
[1] MethComp_1.27  nlme_3.1-109   foreign_0.8-55

loaded via a namespace (and not attached):
[1] grid_3.0.2      lattice_0.20-23 tools_3.0.2
```

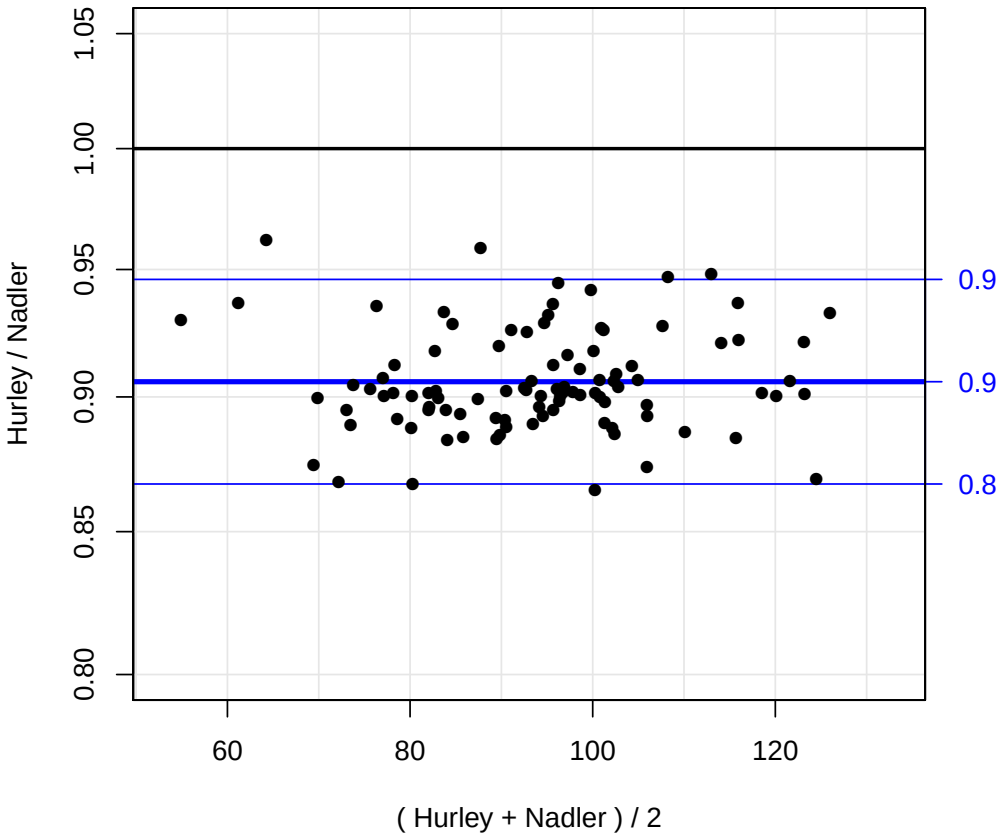

Figure 3.9: *Bland-Altman plot of two methods of measuring plasma volume, using log-transformed data, i.e. a relative scale.*

1. First we load the `fat` data and examine the names in the data frame, and then use `Meth` to convert it to a form that comply with that required by the functions in the `MethComp` package for analyzing the agreement between the two methods (observers) of 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 ...

> head(fat)

    Id Obs Rep Sub Vic
  1  1  KL   1 1.6 4.5
  2  1  KL   2 1.7 4.4
  3  1  KL   3 1.7 4.7
  4  3  KL   1 2.8 6.4
  5  3  KL   2 2.9 6.2
  6  3  KL   3 2.8 6.5
```

```
> 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 ...
   $ 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)

         #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
```

2. The two methods plotted against each other requires that we use the replication number for pairing the measurements; so we just keep the ordering among the replicates when using `to.wide`. But note that since the replicates are exchangeable the pairing is (or rather, should be) arbitrary.

```
> pw <- to.wide( vis )
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> with(pw, plot( SL ~ KL, pch=16, xlim=range(vis$y), ylim=range(vis$y) ) )
> abline( 0,1 )
```

3. Since replicates are exchangeable *within* (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 **??**:

```
> plot( vis )


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

   These two plots are shown in figure 3.11 where it is pretty clear that the random permutation of replicates has little effect.

4. `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.

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

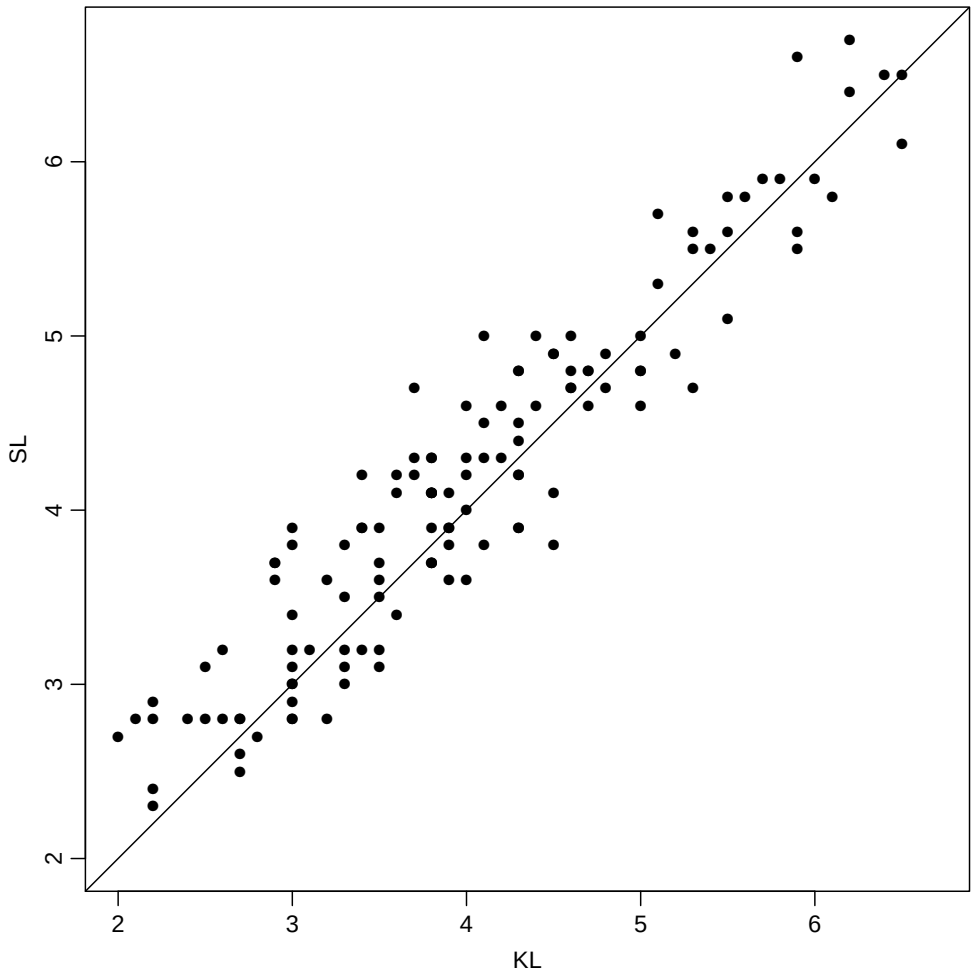

Figure 3.10: *Two observers measuring visceral fat.*

We see that using this approximation we get limits of agreement for KL−SL of $(-0.88, 0.57)$.

5. 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 )

   Conversion between methods:
             alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To: From:
KL  KL       0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
    SL      -0.340  1.044  0.365  0.158  -0.333   0.043   0.357  0.462 -0.024  0.275
SL  KL       0.326  0.957  0.349  0.158   0.333  -0.043  -0.357  0.462 -0.024  0.275
    SL       0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA

> DA.reg( perm.repl(vis) )

   Conversion between methods:
             alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To: From:
KL  KL       0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
    SL      -0.340  1.044  0.359  0.152  -0.333   0.043   0.351  0.419 -0.016  0.462
```

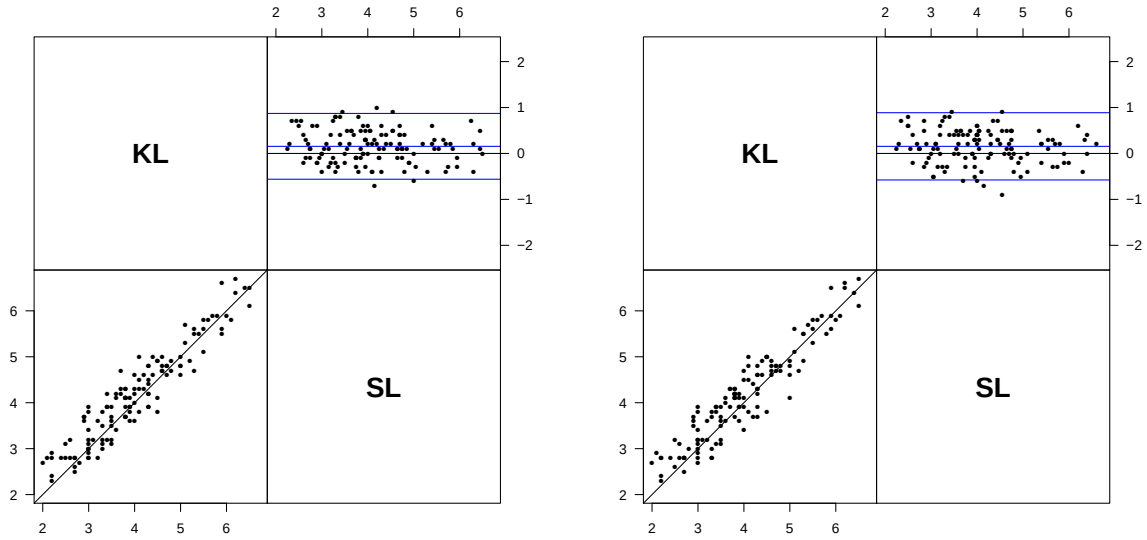

Figure 3.11: *Plot of two methods of measuring visceral fat, using different pairings of the replicates; the left panel is using the pairing in the original coding, the right panel is with a random permutation of replicates.*

```
SL  KL      0.326  0.957  0.344  0.152   0.333  -0.043  -0.351  0.419 -0.016  0.462
    SL      0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

From the two columns with p-values for tests of constant difference ("`beta=1`") and constant SD ("`sd=K`"), it is clear that there are no obvious violations of the assumptions about constant difference or about constant variation across the range of measurements. Moreover, this does not change appreaciably if replicates are permuted randomly within methods:

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

   Conversion between methods:
             alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To: From:
KL  KL       0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
    SL      -0.341  1.045  0.375  0.169  -0.333   0.044   0.367  0.403 -0.006  0.784
SL  KL       0.326  0.957  0.359  0.169   0.333  -0.044  -0.367  0.403 -0.006  0.784
    SL       0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

6. 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.pr LoA-lo LoA-up
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
```

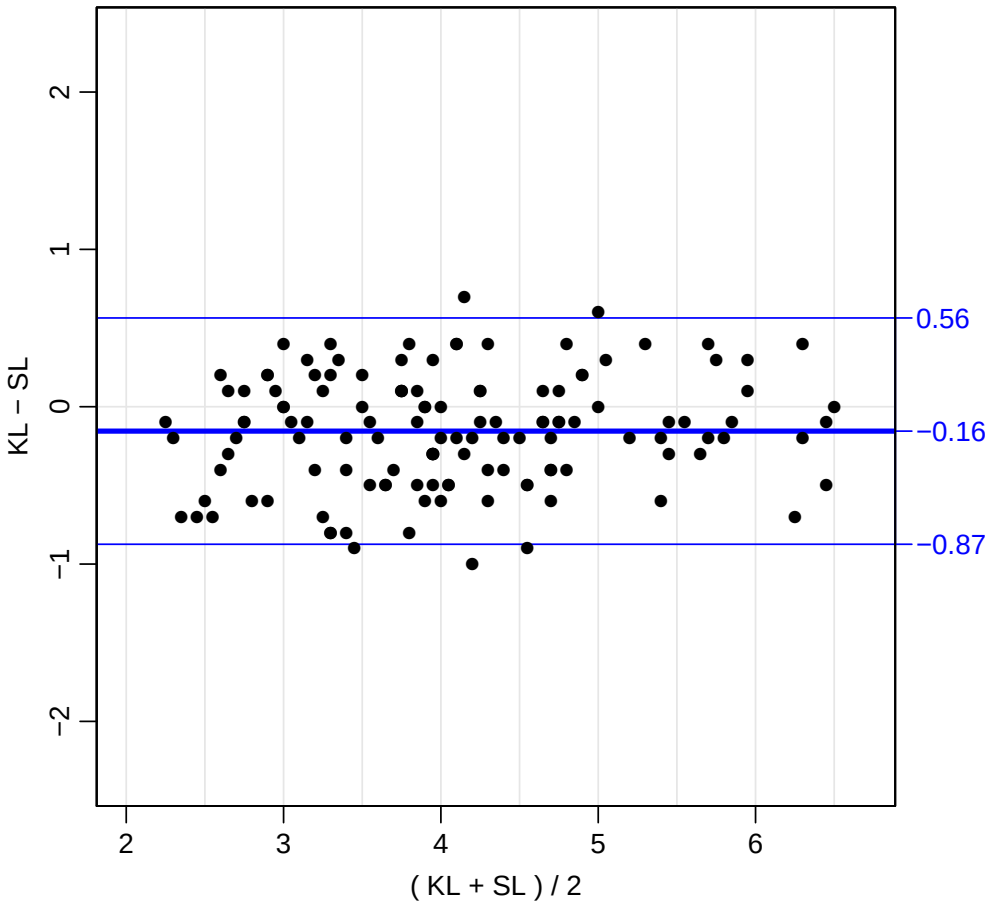

Figure 3.12: *Bland-Altman plot of two observers measuring visceral fat.*

```
 Variance components (sd):
   IxR   MxI   res
KL   0 0.181 0.193
SL   0 0.181 0.173
```

```
> vis.est$LoA
```

```
                Mean      Lower     Upper        SD
SL - KL   0.1550388 -0.5727534 0.8828309 0.3638961
```

7. Moreover we can get the coefficient of reproducibility for each of the methods; that is an upper 95% confidence limit for the absolute difference between two measurements by the same method on the same item by taking 2.8 ($2\sqrt{2}$) times the prediction standard deviation, when predicting from one method to itself. For KL it is $2.8 \times 0.273 = 0.76$ cm and for SL it is $2.8 \times 0.245 = 0.69$ cm, that is, effectively the same for both observers (methods).

8. We can visualize the difference between the *ad-hoc*-computed LoA and the model based ones by plotting them in the same graph. Note that we must use the argument `model=NULL`, since `BA.plot` otherwise by default will fit the relevant variance component model and base the LoA on the results from this:

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( vis, model=NULL )
> abline( h=-vis.est$LoA[1:3], col="red" )
```

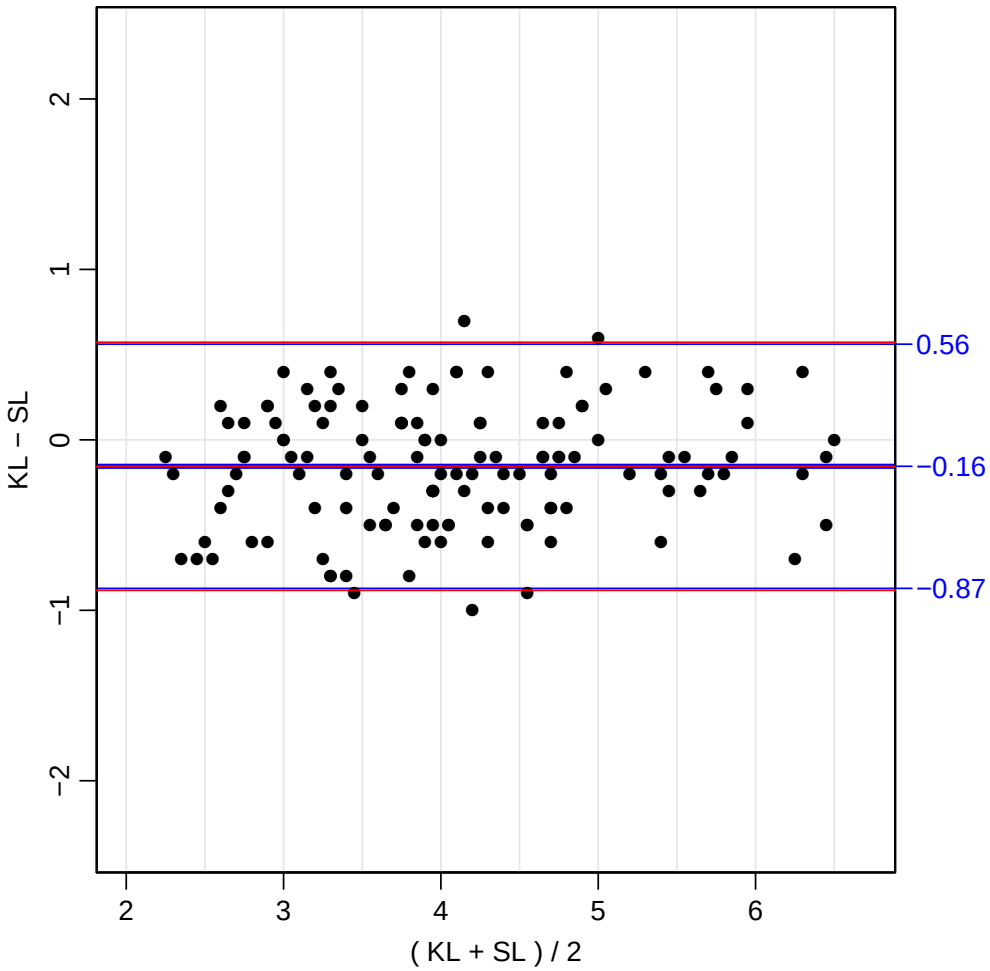

Figure 3.13: *Bland-Altman-plot of two methods of measuring visceral fat, using different pairings of the replicates. The blue lines are the LoA based on taking the paired replicates as items, the red lines are based on the estimates from the proper variance component model.*

As predicted by the theory, the limits based on the *ad-hoc* paired replicates are indistinguishable from those derived from the proper variance component model — see figure 3.13.

9. 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 overplot:

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( vis, diflim=c(-1.5,1.5), axlim=c(2,7),
+          col.p=gray(0.7),col.l=gray(0.5), repl.conn=TRUE )
> par(new=T)
> BA.plot( mean(vis), diflim=c(-1.5,1.5), axlim=c(2,7), cex=0.6, col.p="blue" )
```

The two superposed Bland-Altman plots are shown in figure 3.14.

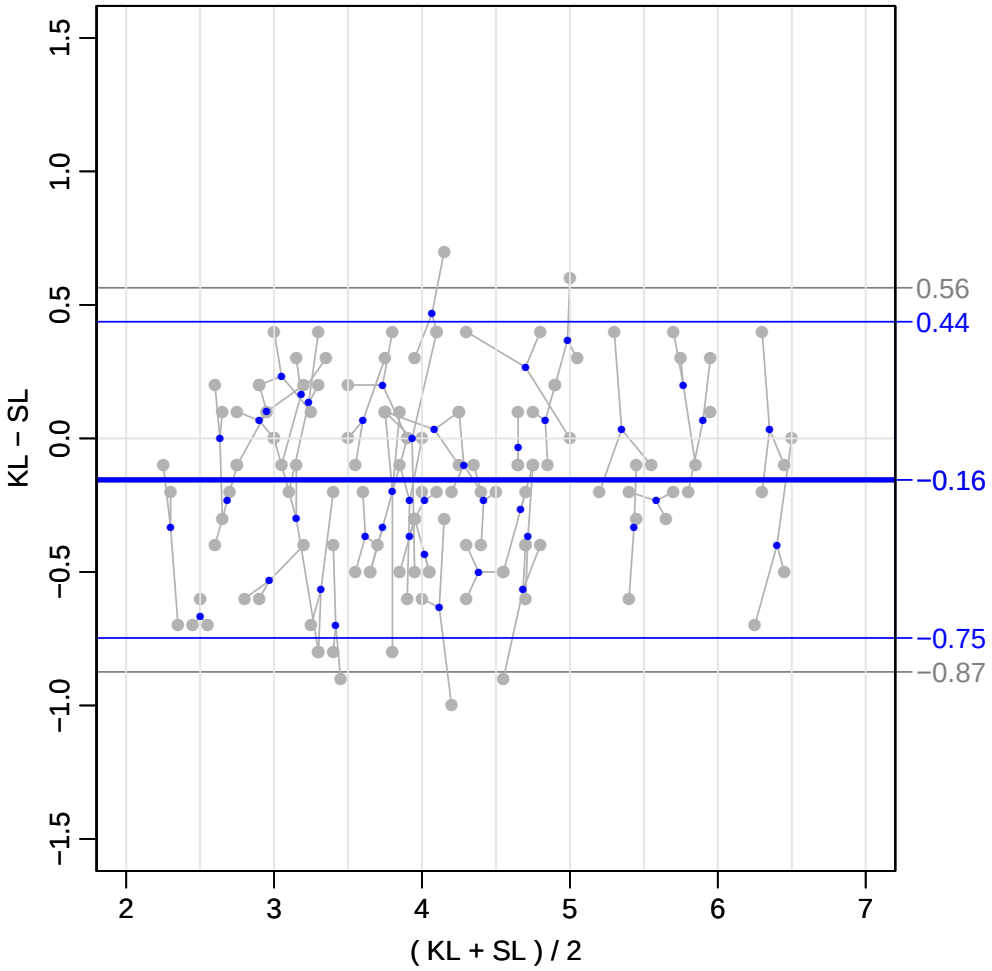

Figure 3.14: *Bland-Altman-plot of two methods of measuring visceral fat, based on the arbitrary pairing of the replicates (black) and on the mean over replicates (grey).*

3.4 Systolic blood pressure: Linked replicates by two methods

The dataset with systolic blood pressure measurements is taken from table 1 in [**?**], where a more detailed description can be found.

1. There are really three methods in the dataset:

```
> library( MethComp )
> data( sbp )
> str( sbp )

  '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: num  1 2 3 4 5 6 7 8 9 10 ...
   $ repl: num  1 1 1 1 1 1 1 1 1 1 ...
   $ y   : num  100 108 76 108 124 122 116 114 100 108 ...

> 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
```

```
> plot( sbp )
```

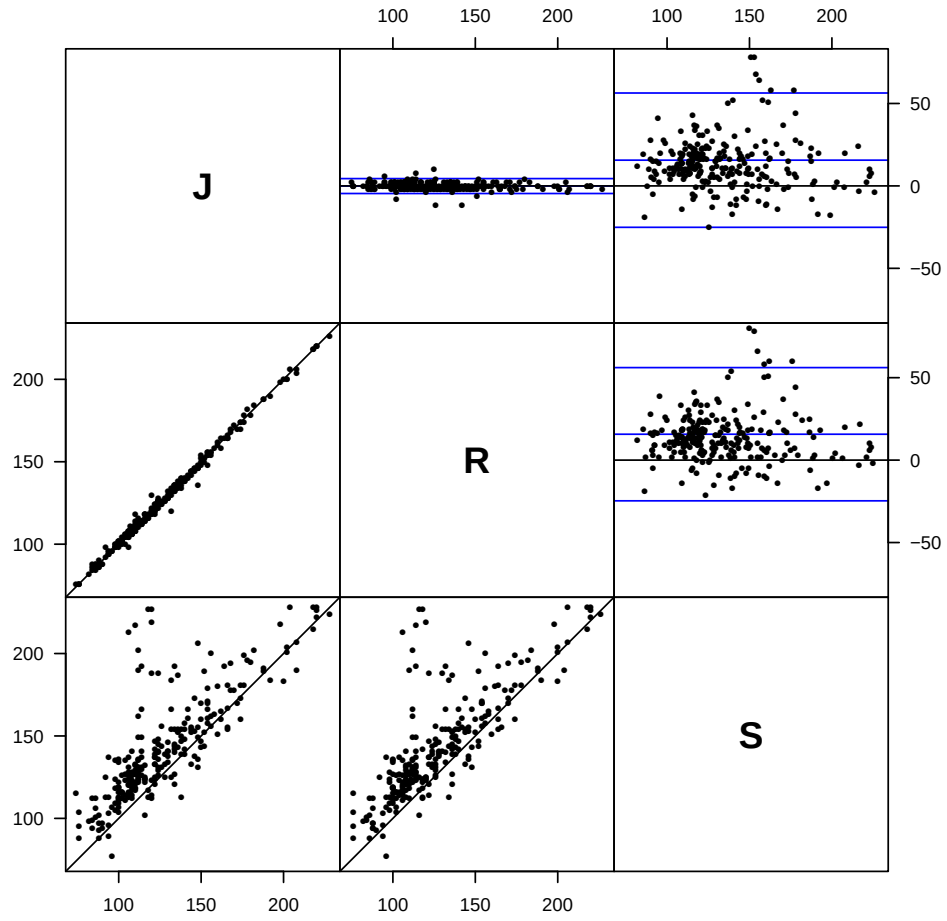

Figure 3.15: *Graphical overview of the* **sbp** *data. The methods J and R are two human observers, whereas method S is an automatic device.*

The resulting plot is shown in figure 3.15, it clearly shows that the two manual measurements are in much closer agreement than any of them are with the automatic.

2. We now restrict 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 (see the help page for the dataset), *i.e.* replicates are *not* exchangeable within methods and items.

```
> par(mar=c(3,3,1,3),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 ...
   $ 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 ...
   $ y   : num  100 108 76 108 124 122 116 114 100 108 ...

> BA.plot( sbp, model="linked" )
```

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

```
> par(mar=c(3,3,1,3),mgp=c(3,1,0)/1.6)
> BA.plot( sbp, model="linked", diflim=c(-15,15), h.grid=-8:8*2 )
```

The resulting plots are shown in figure 3.16. It is noted that the differences are all even integers, thus the original mesurements are so too (or odd, actually).

3. 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

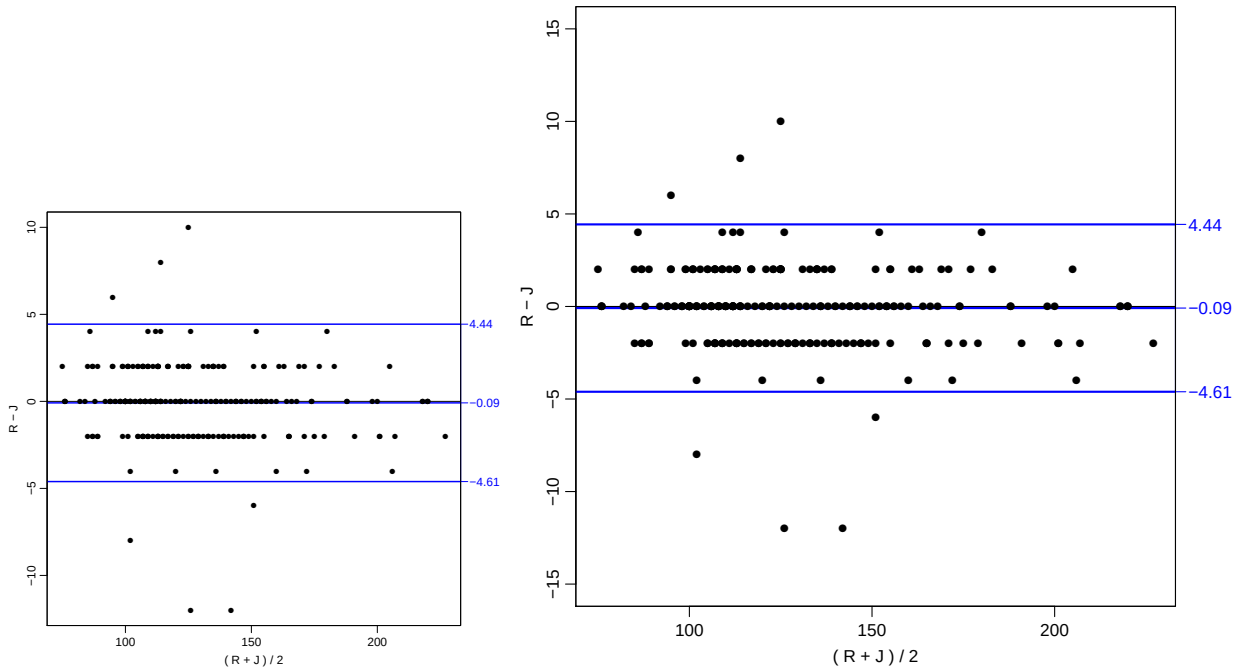

Figure 3.16: *Bland-Altman plot of the* sbp *data. Replicates are linked between methods, so the single replicates in the data has been used as single measurements when doing the Bland-Altman plot. The only difference between the two plots is the scaling of the y-axis.*

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 (note that we must assure that item is a factor in order for `lme` to fit the right model):

```
> 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
 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.

4. 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.pr LoA-lo LoA-up
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 above with `lme`, and then extracts the quantities that we are interested in.

5. The limits of agreement between the two manual observers is then for R−J $-0.0863 \pm 2 \times \sqrt{2 \times 0.248^2 + 1.486^2 + 1.667^2} = (-4.61, 4.44)$, 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.

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

```
> plot( sbp )
```

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

The two resulting plots are shown in figure 3.17. Interestingly, though, it does not seem that the spread relative to the machine method, S, changes much by permutation of replicates. Another indication of non-exchangeability of the replicates is the substantial item by replicate interaction (`IR`) that largely disappears upon permutation of replicates:

```
> p.sbp <- perm.repl( sbp )
> BA.est( p.sbp, MxI=TRUE, linked=TRUE )

   Conversion between methods:
                alpha    beta   sd.pr  LoA-lo  LoA-up
  To: From:
  J   J         0.000   1.000   7.383 -14.765  14.765
      R         0.086   1.000   7.409 -14.732  14.905
  R   J        -0.086   1.000   7.409 -14.905  14.732
      R         0.000   1.000   7.436 -14.872  14.872

   Variance components (sd):
      IxR MxI   res
  J 2.031   0 5.220
  R 2.031   0 5.258
```

It drops from 5.9 to 1.9, and the residual variance increases correspondingly.

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

```
> data( sbp )
> sbp <- Meth( sbp, print=FALSE )
> ( mod.cmp <- BA.est( sbp, linked=TRUE ) )

   Conversion between methods:
                alpha    beta   sd.pr  LoA-lo  LoA-up
  To: From:
```

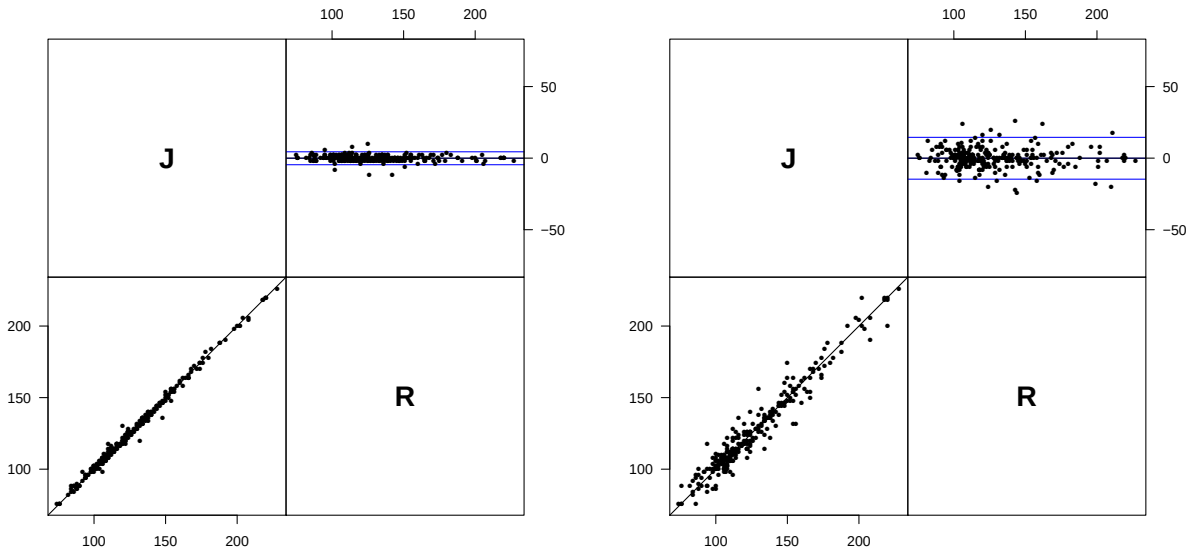

Figure 3.17: *Graphical overview of the* **`sbp`** *data; the left panel with the original replicate numbers used for matching; the other with replicates permuted randomly within methods.*

```
  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
```

```
> round( ftable(mod.cmp$Conv), 4 )
```

```
                alpha     beta    sd.pr   beta=1  in(t-f)  sl(t-f)  sd(t-f)   in(sd)   sl(sd)
  To: From:
  J   J        0.0000   1.0000   2.3052   1.0000   0.0000   0.0000   2.3052   2.3052   0.0000
      R        0.0863   1.0000   2.2724   1.0000   0.0863   0.0000   2.2724   2.2724   0.0000
      S      -15.6196   1.0000  20.3260   1.0000 -15.6196   0.0000  20.3260  20.3260   0.0000
  R   J       -0.0863   1.0000   2.2724   1.0000  -0.0863   0.0000   2.2724   2.2724   0.0000
      R        0.0000   1.0000   2.1873   1.0000   0.0000   0.0000   2.1873   2.1873   0.0000
      S      -15.7059   1.0000  20.3166   1.0000 -15.7059   0.0000  20.3166  20.3166   0.0000
  S   J       15.6196   1.0000  20.3260   1.0000  15.6196   0.0000  20.3260  20.3260   0.0000
      R       15.7059   1.0000  20.3166   1.0000  15.7059   0.0000  20.3166  20.3166   0.0000
      S        0.0000   1.0000  12.9299   1.0000   0.0000   0.0000  12.9299  12.9299   0.0000
```

```
> round( mod.cmp$VarComp, 4 )
```

```
       IxR     MxI    res
  J 5.8872  0.3385 1.6301
  R 5.8872  0.0011 1.5467
  S 5.8872 18.0771 9.1428
```

The resulting estimates from this model gives limits of agreement for R−J based on the method by item and the residual variances, as computed in the resulting `MethComp` object from `BA.est`:

$$-0.0863 \pm 1.96 \times \sqrt{0.3385^2 + 0.0011^2 + 1.6301^2 + 1.5467^2} = -0.0863 \pm 4.4540 = (-4.53, 4.46)$$

Pretty close to those we got from just using the two methods alone.

8. These limits of agreement are very close to 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:

```
> da.cmp <- DA.reg( sbp )
> round( ftable(da.cmp$Conv), 4 )
```

```
                alpha     beta    sd.pr   beta=1  in(t-f)  sl(t-f)  sd(t-f)   in(sd)   sl(sd)
  To: From:
  J   J        0.0000   1.0000       NA       NA   0.0000   0.0000       NA       NA       NA
      R       -1.0756   1.0091   2.2579   0.0476  -1.0707   0.0091   2.2476   1.5054   0.0030
      S       -8.4482   0.9499  19.8388   0.2259  -8.6654  -0.0514  20.3489 -11.5473   0.2049
  R   J        1.0658   0.9910   2.2374   0.0476   1.0707  -0.0091  -2.2476   1.5054   0.0030
      R        0.0000   1.0000       NA       NA   0.0000   0.0000       NA       NA       NA
      S       -7.1775   0.9404  19.5594   0.1459  -7.3981  -0.0615  20.1605 -11.0874   0.2006
  S   J        8.8941   1.0528  20.8860   0.2259   8.6654   0.0514 -20.3489 -11.5473   0.2049
      R        7.6327   1.0634  20.7996   0.1459   7.3981   0.0615 -20.1605 -11.0874   0.2006
      S        0.0000   1.0000       NA       NA   0.0000   0.0000       NA       NA       NA
```

9. Alternatively the limits of agreement could be formulated as a 95% prediction interval for R given a measurement by J, y_J, which would be:

$$y_R|y_J = y_J - 0.0863 \pm 4.4540 = y_J + (-4.54; 4.37)$$

3.5 Oximetry: Linked replicates and non-constant bias

1. Having loaded the data we first transform the data frame `ox` into a `Meth` object:

```
> library(MethComp)
> 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 ...
```

```
> head(ox)
```

```
  meth item repl    y
1   CO    1    1 78.0
2   CO    1    2 76.4
3   CO    1    3 77.2
4   CO    2    1 68.7
5   CO    2    2 67.6
6   CO    2    3 68.3
```

```
> 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
```

The `summary` method for `Meth` objects reveals that most children have three replicates by each method: a few have 2 and one child only has one measurement by each method.

2. Having converted the data frame to a `Meth` object we can plot the two sets of measurements against each other using the `plot.Meth` function, which produces the plot in figure **??**. Note that since we have replicate measurements, these must be paired up in some way in order to plot the measurements from the two methods against each other. In this case, the default behaviour is OK, since the replicates *are* actually linked.

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

3. We use the `BA.plot` function to generate a more detailed version of the Bland-Altman plot than the one resulting from the `plot.Meth` function, which is displayed in 3.19:

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

From the printed output of the `BA.plot` function we find that the estimated average difference between measurements by pulse and CO is -2.5%. The limits of agreement between the two methods are $(-14.8, 9.9)$ respectively. The average difference of about 2.5 is fairly small compared to the median oximetry measurement of 75 but the limits of agreement are quite wide (25% across).

4. We run the `BA.est` function to fit a linear mixed effect model that estimates the relevant variance components:

```
> ( BAox <- BA.est(ox) )

   Conversion between methods:
                  alpha    beta   sd.pr  LoA-lo  LoA-up
  To:   From:
```

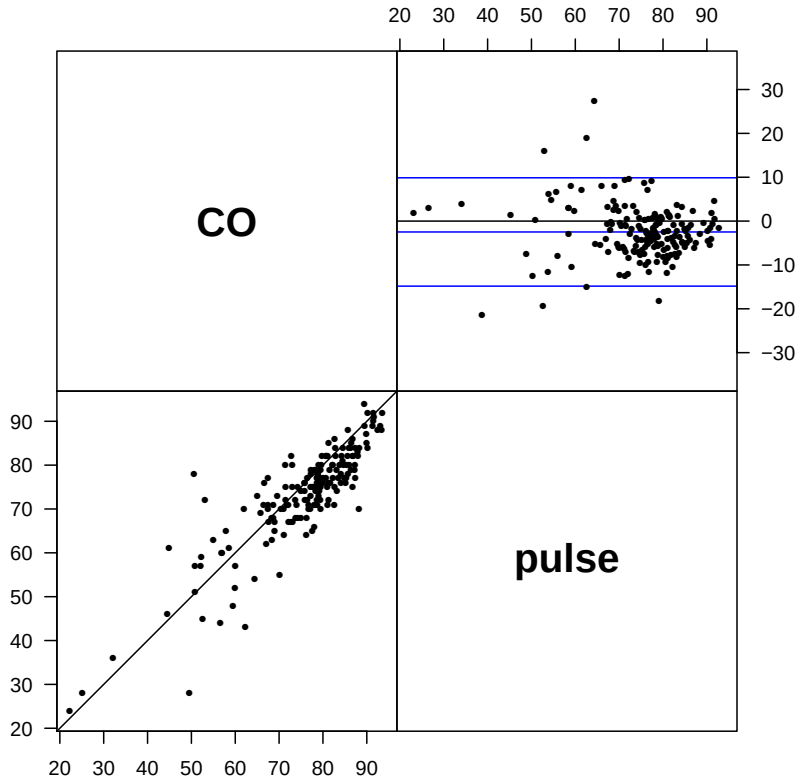

Figure 3.18: *A scatterplot (lower left) and Bland-Altman plot (upper right) of the oximetry data, using the linked replicates as items.*

```
CO    CO       0.000   1.000   3.146  -6.293   6.293
      pulse    2.470   1.000   6.169  -9.867  14.808
pulse CO      -2.470   1.000   6.169 -14.808   9.867
      pulse    0.000   1.000   5.649 -11.298  11.298

 Variance components (sd):
        IxR   MxI   res
CO    3.416 2.928 2.225
pulse 3.416 2.928 3.994
```

5. The residual variances for `CO` and `pulse` are clearly different; the estimated residual variance for co-oximetry (`res` in the output) is 2.22, almost half as large as the corresponding value for pulse oximetry of 3.99. The estimated value of the `IxR` variance component is 3.42, which is larger than the estimate of 2.93 for the `MxI` variance component (note that `MxI.CO` and `MxI.pulse` are the same since we have only two methods of measurement). These variance components lie in between the estimated residual variance for the two methods.

   There is no basis for expecting the `IxR` variance component to have any particular size relative to the other variance components. It represents the variation between replicates which may or may not be relevant for the assessment of repeatability, depending on the circumstances.

6. The `RepCoef` component of the `BA.est` result contains the coefficients of repeatability; the `SD` column is the standard deviation of the difference between two

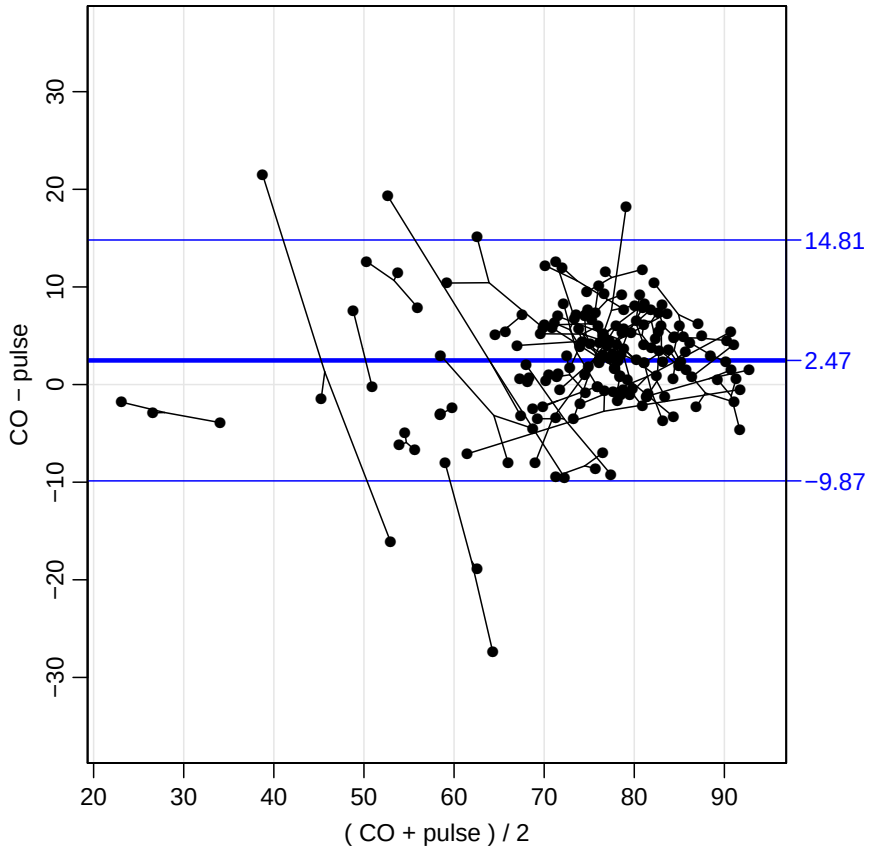

Figure 3.19: *A Bland-Altman plot of the oximetry data, using the linked replicates as items.*

repeat measures by the same method, incorporating the item by replicate variance component, i.e. $\sqrt{2\omega^2 + 2\sigma^2}$. The `Coef.` column is this multiplied by 2 (or if `alpha=` is given as argument the appropriate normal quantile) giving the upper confidence limit for the absolute difference between two measurements.

Hence, the upper confidence limit for the absolute difference between is 11.5% for CO and 14.9% for pulse oximetry.

7. If we want to allow for a non-constant difference between the methods, we would invoke the general model:

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

As outlined, this can be fitted by alternating regressions which conveniently are implemented in the function `AltReg`. In order to follow the convergence we use the parameter `trace=T`, which causes the function to print an account of current parameter estimates after every iteration.

```
> ARox <- AltReg( ox, linked=TRUE, trace=T )

  iteration 1 criterion: 1
         alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   res
  CO     0.911 0.988 1.861        74.419 74.417     1.000 0.974 3.371 3.502 2.292
  pulse -1.039 1.014 1.860        74.422 74.419     1.027 1.000 3.460 3.595 3.958

  iteration 2 criterion: 0.07508045
         alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   res
  CO    -0.714 1.011 1.255        74.419 74.956      1.00  0.99 3.399 3.311 2.251
  pulse -2.006 1.022 3.020        73.878 74.419      1.01  1.00 3.433 3.344 3.981

  iteration 3 criterion: 0.0594666
         alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   res
  CO    -2.363 1.035 1.215        74.419 75.433     1.000 1.005 3.425 3.173 2.211
  pulse -2.971 1.030 3.082        73.412 74.419     0.995 1.000 3.407 3.156 4.002

  iteration 4 criterion: 0.04281372
         alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   res
  CO    -4.019 1.058 1.177        74.419 75.831     1.000 1.019 3.447 3.084 2.175
  pulse -3.963 1.039 3.139        73.034 74.419     0.982 1.000 3.384 3.027 4.021

  iteration 5 criterion: 0.02856943
         alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   res
  CO    -5.668 1.081 1.143        74.419 76.145     1.000  1.03 3.466 3.031 2.145
  pulse -5.009 1.049 3.186        72.744 74.419     0.971  1.00 3.365 2.943 4.036

  iteration 6 criterion: 0.01820552
         alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   res
  CO    -7.307 1.103 1.113        74.419 76.382     1.000 1.039 3.482 3.003 2.121
  pulse -6.124 1.062 3.223        72.530 74.419     0.962 1.000 3.351 2.890 4.048

  iteration 7 criterion: 0.01140264
         alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   res
  CO    -8.936 1.126  1.09        74.419 76.556     1.000 1.046 3.493 2.989 2.102
  pulse -7.314 1.076  3.25        72.377 74.419     0.956 1.000 3.340 2.858 4.057

  iteration 8 criterion: 0.007169339
          alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI
  CO    -10.562 1.148 1.071        74.419 76.680     1.000 1.051 3.502 2.982
  pulse  -8.576 1.092 3.269        72.269 74.419     0.951 1.000 3.331 2.837
          res
  CO    2.087
  pulse 4.064
```

```
iteration 9 criterion: 0.005074459
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI
CO    -12.190 1.169 1.057        74.419 76.768     1.000 1.055 3.508 2.980
pulse  -9.904 1.109 3.282        72.193 74.419     0.948 1.000 3.325 2.824
        res
CO    2.077
pulse 4.069

iteration 10 criterion: 0.003705422
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI
CO    -13.826 1.191 1.047        74.419 76.830     1.000 1.058 3.513 2.978
pulse -11.290 1.126 3.292        72.140 74.419     0.945 1.000 3.321 2.816
        res
CO    2.069
pulse 4.073

iteration 11 criterion: 0.002686236
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI
CO    -15.476 1.213 1.039        74.419 76.873     1.000  1.06 3.516 2.978
pulse -12.727 1.145 3.298        72.104 74.419     0.944  1.00 3.318 2.810
        res
CO    2.064
pulse 4.075

iteration 12 criterion: 0.001930191
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI
CO    -17.144 1.236 1.034        74.419 76.903     1.000 1.061 3.518 2.978
pulse -14.211 1.165 3.303        72.079 74.419     0.942 1.000 3.315 2.807
        res
CO    2.060
pulse 4.077

iteration 13 criterion: 0.001381194
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI
CO    -18.834 1.258 1.030        74.419 76.924     1.000 1.062 3.520 2.978
pulse -15.736 1.185 3.306        72.061 74.419     0.941 1.000 3.314 2.804
        res
CO    2.057
pulse 4.078

iteration 14 criterion: 0.0009863462
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI
CO    -20.548 1.281 1.027        74.419 76.938     1.000 1.063 3.521 2.978
pulse -17.301 1.205 3.308        72.049 74.419     0.941 1.000 3.313 2.802
        res
CO    2.055
pulse 4.079

AltReg converged after  14 iterations
Last convergence criterion was  0.0009863462

> ARox

 Conversion between methods:
              alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:   From:
CO    CO      0.000  1.000  2.906   0.000   0.000   2.906
      pulse  -2.159  1.063  6.385  -2.093   0.061   6.190
pulse CO      2.031  0.941  6.007   2.093  -0.061   6.190
      pulse   0.000  1.000  5.769   0.000   0.000   5.769

 Variance components (sd):
         s.d.
Method    IxR   MxI   res
  CO    3.521 2.978 2.055
  pulse 3.313 2.802 4.079
```

We can now compare the variance components between the model with constant bias and the model with linear bias:

```
> round( ARox$VarComp, 4 )

          s.d.
Method      IxR    MxI    res
  CO    3.5210 2.9785 2.0548
  pulse 3.3127 2.8023 4.0792

> round( BAox$VarComp, 4 )

          IxR   MxI    res
CO    3.4157 2.928 2.2249
pulse 3.4157 2.928 3.9945

> round( ARox$VarComp / BAox$VarComp, 4 )

          s.d.
Method      IxR    MxI    res
  CO    1.0308 1.0172 0.9235
  pulse 0.9699 0.9571 1.0212
```

Clearly, there is not much difference between the two models in terms of the variance components, and the slope between the methods do not seem to differ much from 1.

8. We can get an approximately formal assessment of whether the slopes are 1 and whether the variance is constant from the regression of the differences on the averages, using `DA.reg`:

```
> DA.reg( ox )

 Conversion between methods:
              alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To:   From:
CO    CO      0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
      pulse  -1.977  1.061  6.342  0.142  -1.919   0.059   6.155 17.602 -0.162  0.000
pulse CO      1.864  0.943  5.979  0.142   1.919  -0.059  -6.155 17.602 -0.162  0.000
      pulse   0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

It seems that there is little justification for the addition of the non-constant bias, and neither for the maintaining of the constant variance assumption. However we shall leave these concerns aside to be treated in another practical.

9. In order to get some more information on the variance components than just estimates we use the `MCmcmc`-function to estimate in the model, so that we get estimates of the uncertainty of the variance components from simulations.

   Briefly, the `MCmcmc` function estimates in the model by drawing random samples from the distribution of the parameter estimates. This allows us to construct confidence intervals for the parameters, but also easily for any function of the parameters we can think of; notably ratios of variance estimates. Formally we set up a full Bayesian model with priors, but the priors specified are quite vague, so their practical influence is small.

   To run the function we must specify the dataset, the random effects to include in the model, the number of iterations, and whether we want a model with constant or linear bias between methods:

```
> system.time(
+ ox.mi.ir <- MCmcmc( ox, random=c("mi","ir"), n.iter=5000,
+                      bias="const") )

  Comparison of 2 methods, using 354 measurements
  on 61 items, with up to 3 replicate measurements,
  (replicate values are in the set: 1 2 3 )
  ( 2 * 61 * 3 = 366 ):

  No. items with measurements on each method:
          #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

  Simulation run of a model with
  - fixed bias (slope==1)
  - method by item and item by replicate interaction:
  - using 4 chains run for 5000 iterations
    (of which 2500 are burn-in),
  - monitoring every 3 values of the chain:
  - giving a posterior sample of 3333 observations.

  Initialization and burn-in:
  Compiling model graph
     Resolving undeclared variables
     Allocating nodes
     Graph Size: 2511

  Initializing model

  Sampling:
     user  system elapsed
    20.73    0.09   20.97
```

We can summarize the results by using the `print` function on the resulting `MCmcmc` object `ox.mi.ir`:

```
> print(ox.mi.ir)

   Conversion between methods:
                alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
  To:   From:
  CO    CO      0.000  1.000  3.127   0.000   0.000   3.127
        pulse   2.479  1.000  4.637   2.479   0.000   4.637
  pulse CO     -2.479  1.000  4.637  -2.479   0.000   4.637
        pulse   0.000  1.000  5.756   0.000   0.000   5.756

   Variance components (sd):
         s.d.
  Method    IxR  MxI   res
    CO    3.492 2.98 2.211
    pulse 3.492 2.98 4.070

  Variance components with 95 % cred.int.:
      method    CO              pulse
      qnt      50%  2.5% 97.5%   50%  2.5% 97.5%
  SD
  IxR        3.492 2.831 4.220 3.492 2.831 4.220
  MxI        2.980 2.245 3.835 2.980 2.245 3.835
  res        2.211 1.235 3.034 4.070 3.433 4.834
  tot        5.117 4.541 5.807 6.163 5.526 6.943

  Mean parameters with 95 % cred.int.:
                     50%   2.5%  97.5% P(>0/1)
```

```
alpha[pulse.CO] -2.479 -3.808 -1.183       0
alpha[CO.pulse]  2.479  1.183  3.808       1
beta[pulse.CO]   1.000  1.000  1.000       0
beta[CO.pulse]   1.000  1.000  1.000       0

 Note that intercepts in conversion formulae are adjusted to get
 conversion formulae that represent the same line both ways,
 and hence the median interceps in the posterior do not agree
 exactly with those given in the conversion formulae.
```

We see the resulting conversion equations, but also get estimates and confidence intervals for the variance component parameters.

10. We can get a summary of the results by converting it to a `MethComp` object, which will print a summary like the one obtained from `BA.est`, `DA.reg` and `AltReg`.

```
> MC.ox <- MethComp( ox.mi.ir )
> MC.ox

 Conversion between methods:
              alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:   From:
CO    CO      0.000  1.000  3.127   0.000   0.000   3.127
      pulse   2.479  1.000  4.637   2.479   0.000   4.637
pulse CO     -2.479  1.000  4.637  -2.479   0.000   4.637
      pulse   0.000  1.000  5.756   0.000   0.000   5.756

 Variance components (sd):
       s.d.
Method    IxR  MxI   res
  CO    3.492 2.98 2.211
  pulse 3.492 2.98 4.070
```

11. The `plot` function produces a scatterplot displaying the linear equations relating one method to the other (recall that the slope has been constrained to be 1):

```
> args(plot.MethComp)

function (x, wh.comp = 1:2, pl.type = "conv", dif.type = "lin",
    sd.type = "const", axlim = range(x$data$y, na.rm = TRUE),
    diflim = axlim - mean(axlim), points = FALSE, repl.conn = FALSE,
    col.conn = "gray", lwd.conn = 1, grid = TRUE, h.grid = TRUE,
    col.grid = grey(0.9), lwd = c(3, 1, 1), col.lines = "black",
    col.points = "black", pch.points = 16, cex.points = 1, eqn = is.null(attr(x,
        "Transform")), col.eqn = col.lines, font.eqn = 2, digits = 2,
    mult = FALSE, alpha = NULL, ...)
NULL

> plot( MC.ox, points=TRUE )

Relationships between methods:
 CO-pulse = 2.48 (4.64)
 CO = 2.48+pulse (4.64)
 pulse = -2.48+CO (4.64)
```

The `post.MCmcmc` function produces smoothed posterior densities for the variance components separately for each method (note that only the residual variance is different between methods since the `MI` and `IR` variance components are constrained to be the same):

```
> print( post.MCmcmc( ox.mi.ir ) )
```

The graph strongly supports the contention that the two residual variances are not equal since the support for the posterior density of each hardly overlap at all.

12. We now estimate both intercept and slope parameters using `MCmcmc` and summarize the results using the `print` routine:

```
> system.time(
+ ox.lin <- MCmcmc( ox, bias="lin", random=c("mi","ir"), n.iter=5000 ) )

  Comparison of 2 methods, using 354 measurements
  on 61 items, with up to 3 replicate measurements,
  (replicate values are in the set: 1 2 3 )
  ( 2 * 61 * 3 = 366 ):

  No. items with measurements on each method:
          #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

  Simulation run of a model with
  - method by item and item by replicate interaction:
  - using 4 chains run for 5000 iterations
    (of which 2500 are burn-in),
```

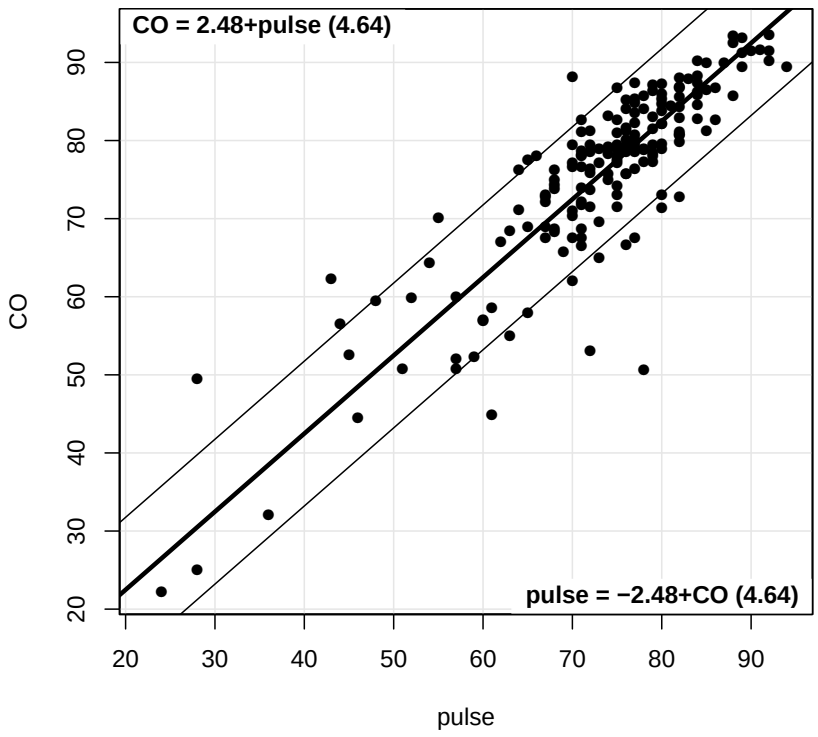

Figure 3.20: *A scatterplot of the oximetry data with the linear equations displayed. The slope of the linear relationship between methods has been constrained to 1.00.*

```
- monitoring every 3 values of the chain:
- giving a posterior sample of 3333 observations.

Initialization and burn-in:
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 2868

Initializing model

Sampling:
   user  system elapsed
  31.95    0.00   31.98
```

13. In order to be reasonably sure about the validity of inference based on the MCMC-estimates we should check that we have sufficient mixing of the chains. One possibility is to take a look using the traces of the sampled values through the functions `check.sd` and `check.beta`, that produces plots of the traces from the (default 4) chains used in the sampling:

```
> print( trace.MCmcmc( ox.lin ) )
```

14. Once we have established that the mixing of the chains is satisfactory, and hence that we are willing to accept that the samples are samples from the stationary distribution

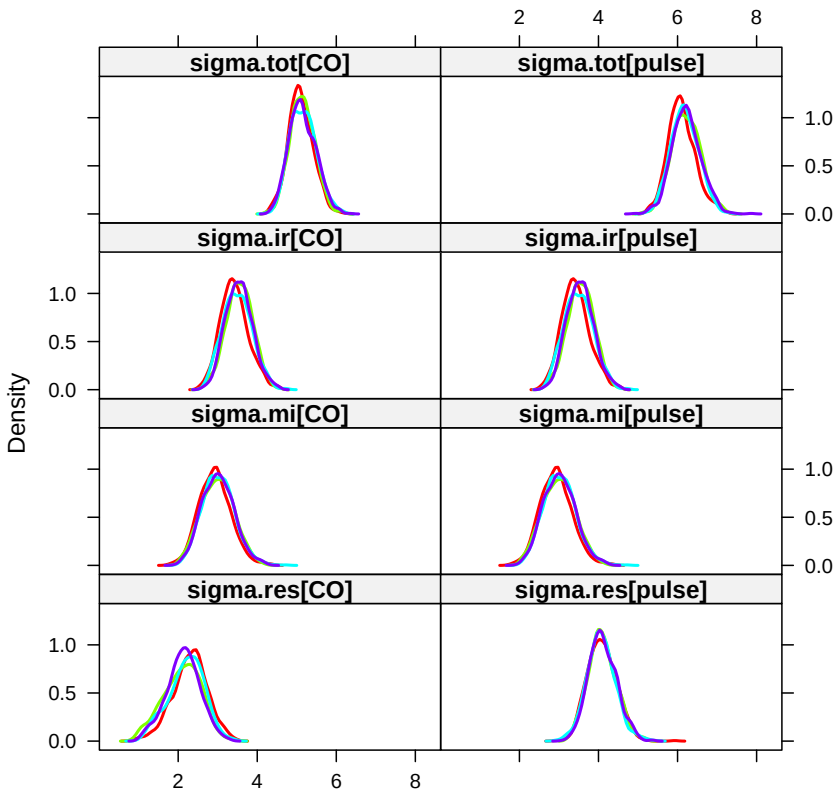

Figure 3.21: *Smoothed density plots of the variance components estimated using* `MethComp`.

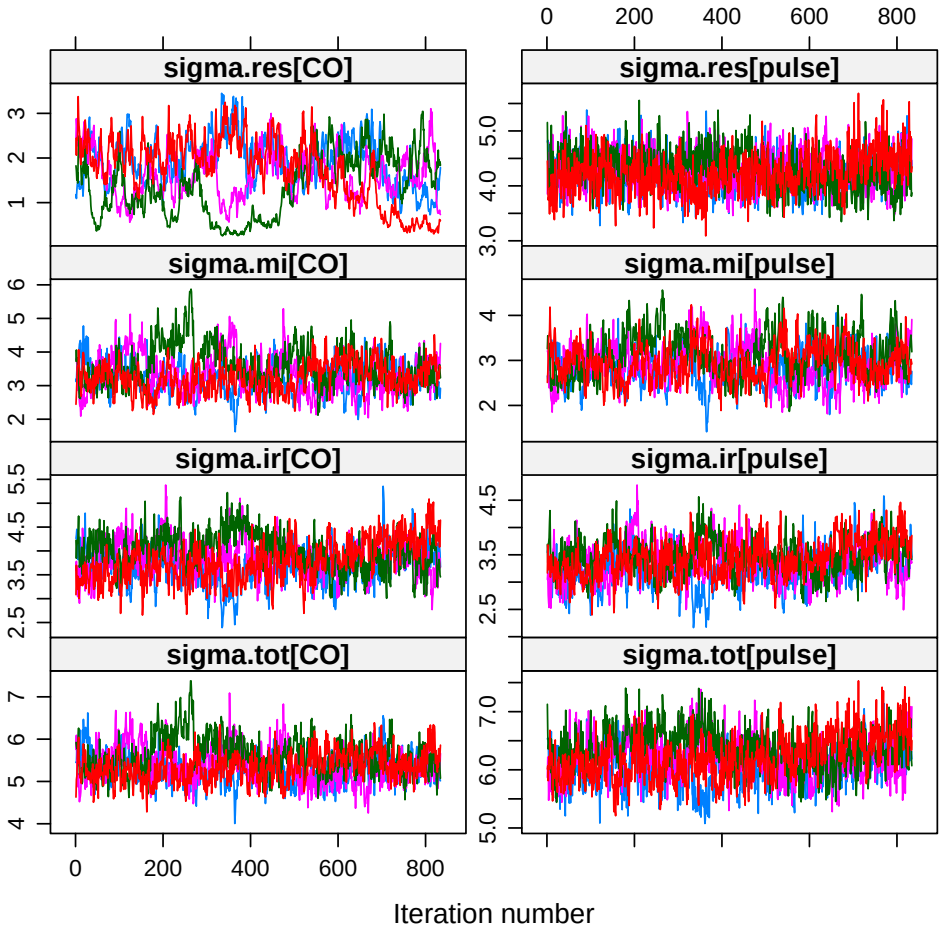

Figure 3.22: *Traces of the chains for the variance components estimated using* `MCmcmc`.

i.e. the correct posterior, we can can use the samples to derive estimates as posterior medians:

```
> print( ox.lin )

 Conversion between methods:
              alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:   From:
CO    CO      0.000  1.000  2.425   0.000   0.000   2.425
      pulse  -6.845  1.128  5.129  -6.434   0.120   4.821
pulse CO      6.070  0.887  4.534   6.434  -0.120   4.806
      pulse   0.000  1.000  5.995   0.000   0.000   5.995

 Variance components (sd):
       s.d.
Method    IxR   MxI   res
  CO    3.821 3.293 1.714
  pulse 3.372 2.919 4.239

Variance components with 95 % cred.int.:
    method    CO              pulse
    qnt      50%  2.5% 97.5%   50%  2.5% 97.5%
SD
IxR        3.821 3.047 4.613 3.372 2.689 4.071
MxI        3.293 2.411 4.514 2.919 2.160 3.902
```

```
res        1.714 0.398 2.847 4.239 3.606 4.979
tot        5.376 4.693 6.326 6.196 5.493 6.966

Mean parameters with 95 % cred.int.:
                    50%    2.5%  97.5% P(>0/1)
alpha[pulse.CO]   6.077  -2.800 14.409   0.933
alpha[CO.pulse] -6.837 -18.612  2.809   0.067
beta[pulse.CO]    0.887   0.775  1.001   0.026
beta[CO.pulse]    1.128   0.999  1.291   0.974

 Note that intercepts in conversion formulae are adjusted to get
 conversion formulae that represent the same line both ways,
 and hence the median interceps in the posterior do not agree
 exactly with those given in the conversion formulae.
```

```
> ox.lin$summary

  NULL
```

```
> MethComp( ox.lin )

 Conversion between methods:
              alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:   From:
CO    CO      0.000  1.000  2.425   0.000   0.000   2.425
      pulse  -6.845  1.128  5.129  -6.434   0.120   4.821
pulse CO      6.070  0.887  4.534   6.434  -0.120   4.806
      pulse   0.000  1.000  5.995   0.000   0.000   5.995

 Variance components (sd):
       s.d.
Method    IxR   MxI   res
  CO    3.821 3.293 1.714
  pulse 3.372 2.919 4.239
```

The summary output provides reasonable evidence that the slope of the linear relationship is different from 1.00, in fact close to 0.90 for the prediction of pulse oximetry from co-oximetry. This implies that the average difference in measurements between the two methods will increase with the magnitude of the underlying measurement. The `plot` method for `MCmcmc` can be used to display the observed data, fitted line with prediction limits and equations:

```
> post.MCmcmc( ox.lin )
```

3.6 Oximetry: Transformation

In the first exercise on the oximetry data, we just used the original ys, measured in percent, as the response variable. We also saw taht on this scale there was in indication of heteroschedasticity while there was little indication that the bias was non-constant.

However, since the measurements are in percent, it would be natural to apply a transformation to the data before doing the analysis. This exercise is a continuation / replication of the previous using a transformation of the measurements.

1. First, we load the data and take a look at the data without transformation:

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

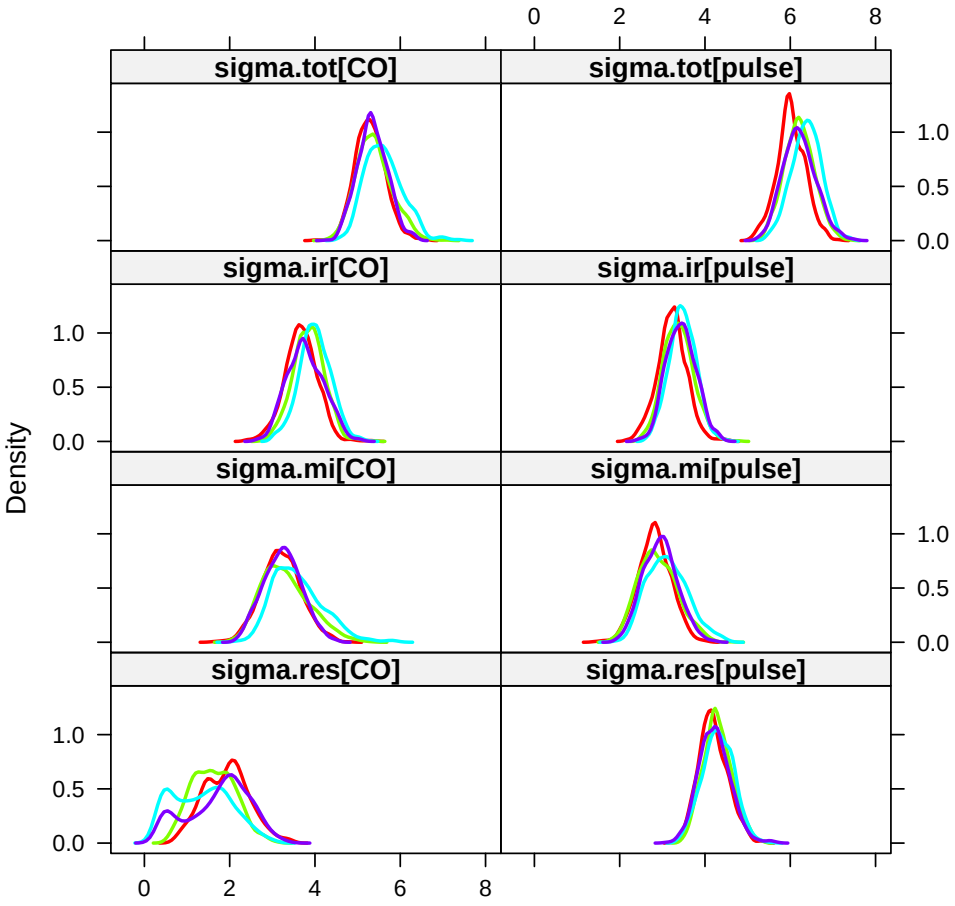

Figure 3.23: *Conversion between methods based on* `MCmcmc`*-output.*

```
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
```

```
> plot( ox )
```

2. Now, we transform the measurements by the logit-transform of the percentages (remember that these are numbers between 0 and 100):

```
> oxt <- transform( ox, y=log(y/(100-y)) )
> plot( oxt )
```

3. A check of the assumptions underlying the LoA; constant bias and variance can be made by using the `DA.reg` function:

```
> DA.reg( oxt )

   Conversion between methods:
                alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
To:   From:
CO    CO        0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
      pulse     0.038  1.111  0.340  0.009   0.036   0.105   0.322  0.348 -0.038  0.246
pulse CO       -0.034  0.900  0.306  0.009  -0.036  -0.105  -0.322  0.348 -0.038  0.246
      pulse     0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

It appears that there is no clear evidence of variance inhomogeneity, but there is some evidence of a non-constant difference between the methods on the logit-scale.

4. Now we compute the limits of agreement, based on the model assuming constant bias, using the correct model for linked replicates. We will also need the analysis on the orginal scale too:

```
> BAoxt <- BA.est( oxt )
> BAoxt$LoA

                    Mean      Lower     Upper        SD
pulse - CO   -0.1563956 -0.8106768 0.4978856 0.3271406
```

5. We note that the LoA are for the logit-transformed data, so if we transform these values by the exponential we get odds-ratios, since the LoA are *differences* of log-odds.

```
> exp( BAoxt$LoA )[-4]

  [1] 0.8552208 0.4445571 1.6452388

> BAox  <- BA.est( ox  )
> BAox$LoA[-4]
```

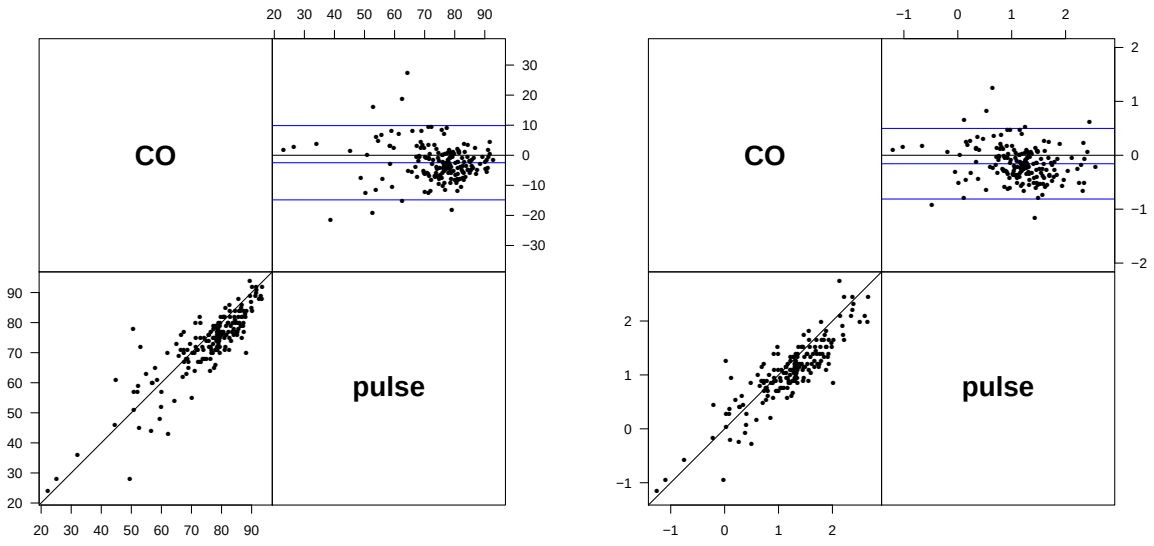

Figure 3.24: *Original (left) and* logit*-transformed oximetry data. Clearly, the logit-transform removes the tendency to diminishing variance at the upper end of the measurements, whereas the outliers in the middle of the scale have not been remedied..*

```
[1]  -2.470446 -14.807793   9.866901
```

This is the odds ratio of pulse versus CO; where odds is defined as saturation divided by one minus saturation — hardly a clinically relvant term.

6. Therefore, it would be more instructive to plot the two methods against each other on the original scale, and then superpose the estimated conversion lines from the model on the logit-transformed scale. This can be quite simply achieved by the `Transform=` argument to the `BA.est` function (we just check the constant variance and horizontal slope by `DA.reg`):

```
> DA.reg( ox, Trans="pctlogit" )

  Note: Response transformed by:  function (p) log(p/(100 - p))


   Conversion between methods:
                alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
  To:   From:
  CO    CO      0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
        pulse   0.038  1.111  0.340  0.009   0.036   0.105   0.322  0.348 -0.038  0.246
  pulse CO     -0.034  0.900  0.306  0.009  -0.036  -0.105  -0.322  0.348 -0.038  0.246
        pulse   0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA

> BAoxl <- BA.est( ox, Trans="pctlogit" )
> BAoxl

  Note: Response transformed by:  function (p) log(p/(100 - p))


   Conversion between methods:
                alpha   beta  sd.pr LoA-lo LoA-up
  To:   From:
  CO    CO      0.000  1.000  0.226 -0.452  0.452
        pulse   0.156  1.000  0.327 -0.498  0.811
  pulse CO     -0.156  1.000  0.327 -0.811  0.498
        pulse   0.000  1.000  0.253 -0.506  0.506

   Variance components (sd):
          IxR   MxI   res
  CO    0.221 0.157 0.160
  pulse 0.221 0.157 0.179
```

You can see the available transformations by referring to the help page of `choose.trans`. The function used here is the `pctlogit` defined as $p \mapsto log(p/(100-p))$, i.e. a logit transform of percentages.

Once you have done the analysis on the transformed scale, we can plot the result in two different ways; either as a conversion plot or as a Bland-Altman plot:

```
> plot( BAoxl, pl.type="conv", points=TRUE,
+              axlim=c(20,100), xaxs="i", yaxs="i" )
> plot( BAoxl, pl.type="BA"  , points=TRUE,
+              axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
```

We can overlay the results from the un-transformed analysis, using the `new=TRUE` argument which prevents R from erasing an existing plot before overlaying the new:

```
> plot( BAoxl, pl.type="conv", points=TRUE,
+       axlim=c(20,100), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAox , pl.type="conv", col.lines="gray",
+              axlim=c(20,100), xaxs="i", yaxs="i" )

  Relationships between methods:
   CO-pulse = 2.47 (6.17)
   CO = 2.47+pulse (6.17)
   pulse = -2.47+CO (6.17)


> plot( BAoxl, pl.type="BA", points=TRUE,
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAox, pl.type="BA", col.lines="gray",
+              axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )

  Relationships between methods:
   CO-pulse = 2.47 (6.17)
   CO = 2.47+pulse (6.17)
   pulse = -2.47+CO (6.17)
```

The resulting plot is shown in figure 3.25

7. We can quickly do the analyses with the other two transformations; in this case we have to supply the transformations (and their inverse) as R-functions:

```
> BAoxll  <- BA.est( ox, Trans=list( function(p) log(-log(p/100)),
+                                    function(x) 100*exp(-exp(x)) ) )
> BAoxcll <- BA.est( ox, Trans=list( function(p) log(-log(1-p/100)),
+                                    function(x) 100*(1-exp(-exp(x))) ) )
```

Once this is done then, we can easily plot the two resulting curves in the same plot as the other two we did previously:

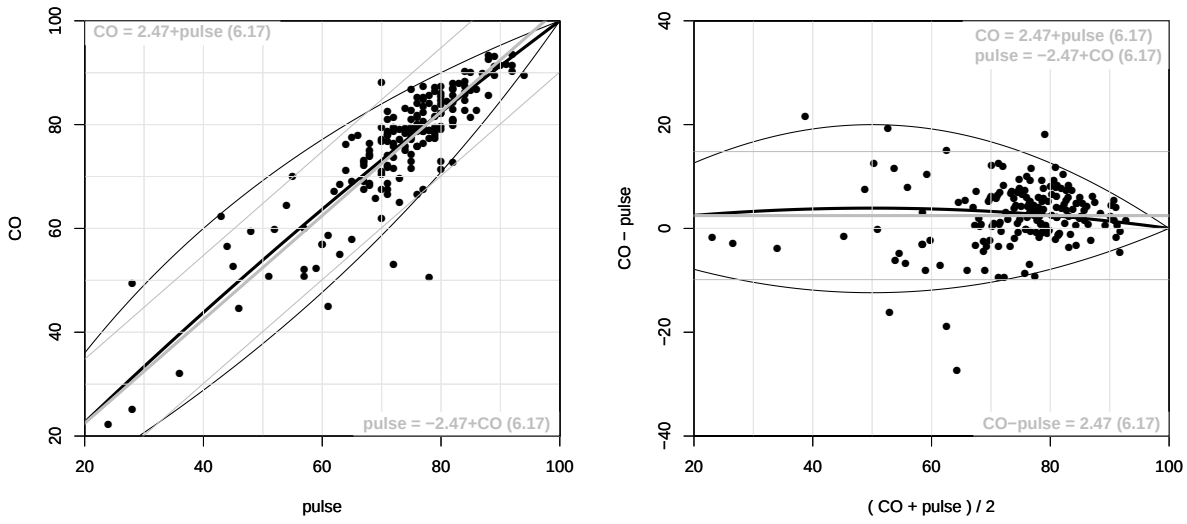

Figure 3.25: *Prediction between pulse and CO-oximetry assuming a constant difference on the* logit *scale. The limits using the original scale are shown too in light gray.*

```
> plot( BAoxl, pl.type="conv",
+       axlim=c(20,100), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAoxll, pl.type="conv",, col.lines="blue",
+       axlim=c(20,100), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAoxcll, pl.type="conv", col.lines="red",
+       axlim=c(20,100), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAox, pl.type="conv", points=TRUE, col.lines="gray",
+              axlim=c(20,100), xaxs="i", yaxs="i" )

  Relationships between methods:
   CO-pulse = 2.47 (6.17)
   CO = 2.47+pulse (6.17)
   pulse = -2.47+CO (6.17)
```



```
> plot( BAoxl, pl.type="BA",
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAoxll, pl.type="BA",, col.lines="blue",
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAoxcll, pl.type="BA", col.lines="red",
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( BAox, pl.type="BA", col.lines="gray", points=TRUE,
+              axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )

  Relationships between methods:
   CO-pulse = 2.47 (6.17)
   CO = 2.47+pulse (6.17)
   pulse = -2.47+CO (6.17)
```

8. Recall the results for the transformed data when we regressed the differences on the averages:

```
> DA.reg( ox, Trans="pctlogit" )

  Note: Response transformed by:  function (p) log(p/(100 - p))


   Conversion between methods:
                alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K
  To:   From:
  CO    CO      0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
        pulse   0.038  1.111  0.340  0.009   0.036   0.105   0.322  0.348 -0.038  0.246
  pulse CO     -0.034  0.900  0.306  0.009  -0.036  -0.105  -0.322  0.348 -0.038  0.246
        pulse   0.000  1.000     NA     NA   0.000   0.000      NA     NA     NA     NA
```

The rough estimate of the slope is 1.1, and this is actually significantly different for 1.

We estimate both intercept and slope parameters using `MCmcmc` and summarise the results using the `print` routine.

```
> system.time(
+ MCoxl <- MCmcmc( ox, bias="lin", random=c("mi","ir"), n.iter=5000,
+                   Trans="pctlogit" ) )
```

```
Comparison of 2 methods, using 354 measurements
on 61 items, with up to 3 replicate measurements,
(replicate values are in the set: 1 2 3 )
( 2 * 61 * 3 = 366 ):

No. items with measurements on each method:
        #Replicates
Method    1   2   3 #Items #Obs: 354 Values:  min      med      max
  CO      1   4  56     61       177    -1.254049 1.300981 2.666159
  pulse   1   4  56     61       177    -1.152680 1.098612 2.751535

Simulation run of a model with
- method by item and item by replicate interaction:
- using 4 chains run for 5000 iterations
  (of which 2500 are burn-in),
- monitoring every 3 values of the chain:
- giving a posterior sample of 3333 observations.

Initialization and burn-in:
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 2869

Initializing model

Sampling:
   user  system elapsed
  42.04    0.09   42.21

> MethComp( MCox1 )

Note: Response transformed by:  function (p) log(p/(100 - p))


 Conversion between methods:
```

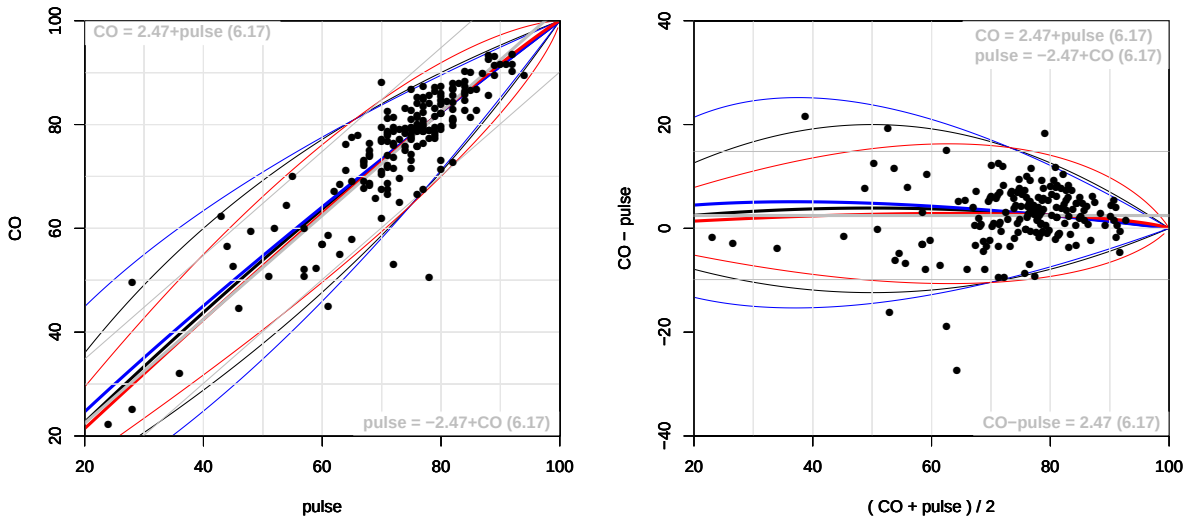

Figure 3.26: *Prediction between pulse and CO-oximetry assuming a constant difference on the* logit *scale. The red lines are limits based on the complementary log−log transform, and the blue lines the log−log transform. The limits using the original scale are shown too in light gray.*

```
                alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
  To:   From:
  CO    CO      0.000  1.000  0.183   0.000   0.000   0.183
        pulse   0.008  1.140  0.264   0.008   0.131   0.247
  pulse CO     -0.007  0.877  0.232  -0.008  -0.131   0.247
        pulse   0.000  1.000  0.284   0.000   0.000   0.284

   Variance components (sd):
         s.d.
  Method    IxR   MxI   res
    CO    0.258 0.179 0.129
    pulse 0.225 0.157 0.201

> system.time(
+ ARoxl <- AltReg( ox, linked=TRUE, Trans="pctlogit", trace=FALSE ) )

  AltReg converged after  15 iterations
  Last convergence criterion was  0.0008526646
     user  system elapsed
    11.68    0.00   11.70

> MethComp( ARoxl )

  Note: Response transformed by:  function (p) log(p/(100 - p))


   Conversion between methods:
                alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
  To:   From:
  CO    CO      0.000  1.000  0.202   0.000   0.000   0.202
        pulse   0.042  1.105  0.341   0.040   0.100   0.324
  pulse CO     -0.038  0.905  0.309  -0.040  -0.100   0.324
        pulse   0.000  1.000  0.271   0.000   0.000   0.271

   Variance components (sd):
           s.d.
  Method    IxR   MxI   res
    CO    0.232 0.160 0.143
    pulse 0.210 0.145 0.191
```

We see the estimates are not the same by the two methods, but the estimates from the `AltReg` method are well within the posterior credible intervals from the `MCmcmc` function.

Finally we can put the results from the two different estimation approaches on top of each other and compare with the prediciton limits derived by assuming constant bias on the logit-scale:

```
> plot( BAoxl, pl.type="comp", points=TRUE, pch=16,
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( ARoxl, pl.type="comp",, col.lines="blue",
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
> par( new=TRUE )
> plot( MethComp(MCoxl), pl.type="comp", col.lines="red",
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )
```

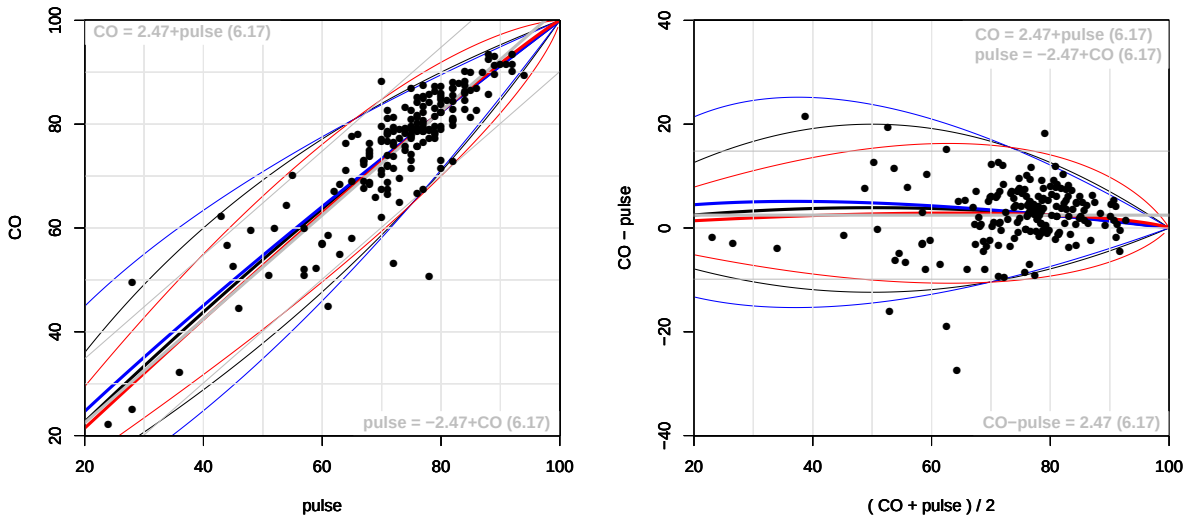

Figure 3.27: *Prediction between pulse and CO-oximetry, The black line assumes constant bias on the logit scale, the red (`MCmcmc`) and the blue (`AltReg`) allows a linear relationship on that scale.*

Chapter 4

MethComp manual

Version 1.27

Date 2014-02-20

Title Analysis of Method Comparison studies.

Author Bendix Carstensen, Lyle Gurrin, Claus Ekstrom, Michal Figurski

Maintainer Bendix Carstensen ⟨bxc@steno.dk⟩

Depends R (>= 3.0.0), nlme

Suggests R2WinBUGS, BRugs, rjags, coda, lattice, lme4

Description Methods (standard and advanced) for analysis of agreement between measurement methods.

License GPL (>= 2)

URL http://BendixCarstensen.com/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://BendixCarstensen.com

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://BendixCarstensen.com.

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 )
## Not run:
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)
## End(Not run)
```

`Ancona` *Data from a rating experiment of recorgnizing point counts.*

Description

At the course ”Statsitical Analysis of Method Comparison Studies” ai the SISMEC conference in Ancona, on 28 September 2011, the participants on the course were used as raters of ten pictures of points. Pitures were shown 3 times each to the participants, and they assessed the number of points in each.

Usage

```
data(Ancona)
```

Format

A data frame with 510 observations on the following 4 variables.

`rater` a factor with 17 levels

`item` a numeric vector indicating the pictures shown. The value is the actual number of points.

`repl` a numeric vector, replicate number

`score` a numeric vector, the number of points in `item`

Source

The course ”Statsitical Analysis of Method Comparison Studies” ai the SISMEC conference in Ancona, on 28 September 2011.

Examples

```
library( MethComp )
data( Ancona )
Anc <- Meth( Ancona, 1, 2, 3, 4 )
```

`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),
                          corMxI=FALSE,
                          varMxI=TRUE,
                          IxR.pr=FALSE,
                            bias=TRUE, alpha=0.05,
                 Transform = NULL,
                 trans.tol = 1e-6,
                 random.raters = FALSE,
                 lmecontrol = lmeControl(msMaxIter=300),
                 weightfunction = c("mean", "median")
 )
  ## 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,
         corMxI = FALSE,
         varMxI = TRUE,
           bias = TRUE,
          print = FALSE,
 random.raters = FALSE,
    lmecontrol = lmeControl(msMaxIter=300)
 )
```

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 given, this is ignored.
`MxI`	Logical. Should the method by item interaction (matrix effect) be included in the model.
`matrix`	Logical. Alias for `MxI`.
`corMxI`	Logical. Should the method by item interaction allow coorelated effects within item. Ignored if only two methods are compared.
`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.
random.raters	Logical. Should methods/raters be considered as random. Defaults to FALSE which corresponds to a fixed effect of methods/raters.
lmecontrol	A list of control parameters passed on to lme.
weightfunction	Function to weigh variance components for random raters. Defaults to mean but can also be median.
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 arguments passed on. Currently 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. The function VC.est is actually just a wrapper for the functions VC.est.fixed that handles the case with fixed methods (usually 2 or three) i.e. the classical method comparison problem, and VC.est.random that handles the situation where "methods" are merely a random sample of raters from some population of raters; and therefore are regarded as random.

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, the last one c("alpha","beta","sd.pred","LoA: lower","upper"). It represents the mean conversions between methods and the prediction 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" )
## Not run:
data( sbp )
BA.est( sbp )
BA.est( sbp, linked=FALSE )
# Check what you get from VC.est
str( VC.est( sbp ) )
## End(Not run)
```

`BA.plot` *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. Works with `Meth` and `MethComp` objects too.

Usage

```
BA.plot( y1, y2, meth.names = NULL,
                      wh.comp = 1:2,
                      pl.type = "BA",
                     dif.type = "const",
                      sd.type = "const",
                        model = if( inherits(y1,"Meth") & has.repl(y1) ) "linked"
                                else NULL,
                         eqax = FALSE,
                         axlim = if( is.data.frame(y1) ) range(y1$y) else range(c(y1,y2)),
                        diflim = NULL,
                          grid = TRUE,
                        h.grid = TRUE,
                       col.grid = grey(0.9),
                         points = TRUE,
                    col.points = "black",
                    cex.points = 1,
                    pch.points = 16,
                           lwd = c(3,1,1),
                     col.lines = "blue",
                     repl.conn = FALSE,
                      col.conn = col.points,
                      lwd.conn = 1,
                          xlab = NULL,
                          ylab = NULL,
                           eqn = FALSE,
                       col.eqn = col.lines,
                      font.eqn = 2,
                        digits = 2,
                     Transform = if( mult ) "log" else NULL,
                          mult = FALSE,
                         alpha = NULL,
                           ... )
```

Arguments

`y1`	Numerical vector of measurements by 1st method. Can also be a `Meth` or a `MethComp` object, see details.
`y2`	Numerical vector of measurements by 2nd method. Must of same length as `x`. Ignored if a `Meth` or a `MethComp` objects is given for `y1`.
`meth.names`	Label for the method names.
`wh.comp`	Which methods should be compared. Either numerical or character.
`pl.type`	What type of plot should be made, `"BA"` for differences versus averages, `"conv"` for method 1 versus method 2.
`dif.type`	How should difference depend on the averages. `"const"` or `"lin"`.
`sd.type`	How should the standard deviation depend on the averages. `"const"` or `"lin"`.
`model`	Should a variance component model be used to compute the limits of agreement? If `NULL` a simple analysis is made; other possibilities are `"exch"` or `"linked"` for exchangeable or linked replicates. If a `Meth` object with replicate measurements is supplied, the default is to assume linked replicates.
`eqax`	Should the axes be identical? If a Bland-Altman plot is drawn, the axis for the differences will have the same extent as the axis for the averages, but centered on 0 (see `diflim`).

`axlim`	The limits of the axes.
`diflim`	The limits of the difference axis.
`grid`	Should there be a grid? If numeric, how many gridlines? If a vector of length>1, it will be taken as the position of the vertical gridlines.
`h.grid`	Position of the horizontal gridlines. Ignored if `pl.type="conv"`.
`col.grid`	Color of the gridlines.
`points`	Logical. Should the observed points be drawn?
`col.points`	What color should they have?
`cex.points`	How large should they be?
`pch.points`	What plot character for the points
`lwd`	Numerical vector of 3, giving the width of the conversion line (mean difference) and the limits of agreement.
`col.lines`	What color should the lines have.
`repl.conn`	Should replicate measurements be connected (within items)?
`col.conn`	Color of connecting lines.
`lwd.conn`	Width of connecting lines.
`xlab`	x-axis label.
`ylab`	y-axis label.
`eqn`	Logical. Should the equations linking the methods be shown on the plot? If a Bland-Altman plot is made, both the equations linking the methods and the equation for the differences versus the averages are shown.
`col.eqn`	Color for equations
`font.eqn`	Font for equations
`digits`	How many digits after the decimal point should be used when showing the equations.
`Transform`	Transformation applied to data prior to analysis. Plots are made on the original scale after back-transformation.
`mult`	Logical. If TRUE, ratios of measurement instead of differences will be plotted in the Bland-Altman plot on a logarithmic axis, and limits of agreement will be given on this scale? This gives the same analysis as using `Transform="log"`, but a different plot. Using another transformation than the log is accommodated, but no LoA is shown on the axis.
`alpha`	1 minus the confidence level. If `NULL` a multiplier of 2 is used for constructing prediction limits, otherwise a t-quantile with d.f. equal th number of items minus 1.
`...`	Further parameters passed on to `plot.MethComp`

Details

A plot of the relationship between the methods is produced; either a Bland-Altman plot of the differences versus averages, or a 45 degree rotation as a conversion between the methods. If `model=NULL` a simple regression of averages on differences is made by calling `DA.reg`, and the specified conversion plotted.

Value

An object of class `MethComp` and either `DA.reg` (if `model=NULL`) or `BA.est` (if `model` is character).

Author(s)

Bendix Carstensen ⟨bxc@steno.dk⟩, http://BendixCarstensen.com.

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.est, DA.reg, MCmcmc.

Examples

```
data( ox )
ox <- Meth( ox )
# The simplest possible Bland-Altman plot
BA.plot( ox )

## With bells and whistles, comparing the naive and model
par( mfrow=c(2,2) )
BA.plot( ox, model=NULL, repl.conn=TRUE, col.lines="blue",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
         las=1, eqn=TRUE, dif.type="lin", pl.type="BA", sd.type="lin",
         grid=1:9*10, digits=3,font.eqn=1)
par(new=TRUE)
BA.plot( ox, model="linked", repl.conn=TRUE, col.lines="red",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
         las=1, eqn=FALSE, dif.type="lin", pl.type="BA", sd.type="lin",
        grid=1:0*10, digits=3)
BA.plot( ox, model=NULL, repl.conn=TRUE, col.lines="blue",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
         las=1, eqn=TRUE, dif.type="lin", pl.type="conv", sd.type="lin",
        grid=1:9*10, digits=3,font.eqn=1)
par(new=TRUE)
BA.plot( ox, model="linked", repl.conn=TRUE, col.lines="red",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
         las=1, eqn=FALSE, dif.type="lin", pl.type="conv", sd.type="lin",
         grid=1:9*10, digits=3)
# The same again, but now logit-transformed
BA.plot( ox, model=NULL, repl.conn=TRUE, col.lines="blue",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
         las=1, eqn=TRUE, dif.type="lin", pl.type="BA", sd.type="lin",
         grid=1:9*10, digits=3,font.eqn=1,Transform="pctlogit")
par(new=TRUE)
BA.plot( ox, model="linked", repl.conn=TRUE, col.lines="red",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
         las=1, eqn=FALSE, dif.type="lin", pl.type="BA", sd.type="lin",
         grid=1:0*10, digits=3,Transform="pctlogit")
BA.plot( ox, model=NULL, repl.conn=TRUE, col.lines="blue",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
         las=1, eqn=TRUE, dif.type="lin", pl.type="conv", sd.type="lin",
         grid=1:9*10, digits=3,font.eqn=1,Transform="pctlogit")
par(new=TRUE)
BA.plot( ox, model="linked", repl.conn=TRUE, col.lines="red",
         axlim=c(0,100), diflim=c(-50,50), xaxs="i", yaxs="i",
```

```
las=1, eqn=FALSE, dif.type="lin", pl.type="conv", sd.type="lin",
grid=1:9*10, digits=3,Transform="pctlogit")
```

`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://BendixCarstensen.com

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://BendixCarstensen.com.

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://BendixCarstensen.com

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 regression of differences 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,
```

```
        print = TRUE,
random.raters = FALSE,
     DA.slope = TRUE )
 DA2y( a=0, b=0, s=NA )
 y2DA( A=0, B=1, S=NA )
```

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.
`print`	Should the results be printed?
`random.raters`	If methods really are a random selection of raters, neither intercept nor slope different from 0 are sensible, so if this is `TRUE`, intercept and slope in the regression of difference on averages are fixed to 0. Meaning that we are essentially looking at the raw differences as residuals.
`DA.slope`	If this is TRUE, a slope of the differences in the verages is estimated, otherwise the relationship is assumed constant.
`a`	Intercept in the linear relation of the differences `y1-y2` to the averages `(y1+y2)/2`. If a vector of length>1, this is used instead of `a`, `b` and `s`, and `b` and `s` are ignored.
`b`	Slope in the linear relstion of the differences to the averages.
`s`	SD from the regression of the differences in the averages. Can be `NA`.
`A`	Intercept in the linear relation of y1 on y2.
`B`	Slope in the linear relation of y1 on y2.
`S`	SD for the linear relation of y1 on y2. Can be `NA`.

Details

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

The functions `DA2y` and `y2DA` are convenience functions that convert the estimates of intercept, slope and sd from the regression of $D = y_1 - y_2$ on $A = (y_1 + y_2)/2$, back and forth to the resulting intercept, slope and sd in the relationship between y_1 and y_2, cf. Carstensen (2010), equation 6.

`DA2y` takes intercept(`a`), slope(`b`) and sd(`s`) from the relationship `(y1-y2)=a+b((y1+y2)/2)+e` with sd(`e`)=`s`, and returns a two by 3 matrix with columns `"int","slope","sd"` and rows `"y1|2","y2|1"`.

`y2DA` takes intercept(`A`), slope(`B`) and sd(`S`) from the relationship `y1=A+B y2 + E` with sd(`E`)=`E`, and returns a vector of length 3 with names `"int(t-f)","slope(t-f)","sd(t-f)"`, where `t` refers to "to" (`y1` and `f` to "from" `y2`.

Value

`DA.reg` returns 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"`, referring to the linear relationship of `To` to `From`, `"int(t-f)"`, `"slope(t-f)"`, `"sd(t-f)"`, referring to the regression of the

differences on the averages, and `"int(sd)"`, `"slope(sd)"`, and `"s.d.=K"`, referring to the regression of the absoulte residuals on the averages, and `LoA-lo`, `LoA-hi`, the limits of agreement.

Converting from method l to method k using

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

with prediction standard deviation σ, just requires the entries `[k,1,c("alpha","beta","sd.pred")]`, if we assume the s.d. is constant.

The next entry is the p-values for the hypothesis $\beta = 1$, intercept and slope of the SD of the differences as a linear function of the average and finally p-value of the hypothesis that standard errors are constant over the range. The latter three are derived by regressing the absolute values of the residuals on the averages, and can be used to produce LoA where the s.d. increases (or decreases) by the mean, using the function `DA2y`.

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 dataframe. The measurements in `y` are left un-transformed, even if data are transformed (i.e. if the `Transform` attribute of the object is non-null).

`DA2y` returns a 2 by 3 matrix with rownames `c("y1|2","y2|1")` and columnnames `c("int","slope","sd")`, calculated under the assumption that the differences were formed as `D <- y1 - y2`.

`y2DA` returns a 3-component vector with names `c("DA-int","DA-slope","DA-sd")`, referring to differences `D=y1-y2` as a linear function of `A=(y1+y2)/2`.

Author(s)

Bendix Carstensen, Steno Diabetes Center, `bxc$steno.dk`, http://BendixCarstensen.com/MethComp

References

B. Carstensen: Comparing methods of measurement: Extending the LoA by regression. Stat Med, 29:401-410, 2010.

Examples

```
data( milk )
DA.reg( milk )
data( sbp )
print( DA.reg(sbp), digits=3 )
# Slope, intercept : y1 = 0.7 + 1.2*y2 (0.4)
A <- c(0.7,1.2,0.4)
( y2DA( A ) )
( DA2y( y2DA( A ) ) )
```

`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://BendixCarstensen.com.

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

```
# 'True' values
M <- runif(100,0,5)
# Measurements:
x <-         M + rnorm(100)
y <- 2 + 3 * M + rnorm(100,sd=2)
# Deming regression with equal variances,  resp. 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 functions `jags.model` and `coda.samples` form the `rjags` packgae or `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.code.file = "model.txt",
       clearWD = TRUE,
     code.only = FALSE,
      ini.mult = 2,
      list.ini = TRUE,
           org = FALSE,
     Transform = NULL,
     trans.tol = 1e-6,
       program = "JAGS",
bugs.directory = getOption("bugs.directory"),
         debug = FALSE,
           ... )
## 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 methods 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`.

`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 components, see below in details.

`list.ini` List of lists of starting values for the chains, or logical indicating 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.

`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.

`program` Which program should be used for the MCMC simulation. Possible values are "`BRugs`", "`openbugs`", "`ob`" (openBUGS/BRugs), "`winbugs`", "`wb`" (WinBUGS), "`jags`" (JAGS). Case insensitive. Defaults to "`JAGS`" since: 1) JAGS is available on all platforms and 2) JAGS seems to be faster than BRugs on (some) windows machines.

`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`.
`...`	Additional arguments passed on to `bugs`.
`object`	An `MCmcmc` object
`alpha`	1 minus the the confidence level
`x`	An `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

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 SDs, that is). Furthermore, the posterior of the point where the conversion lines intersects the identity as well as the prediction SDs 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 parameters, 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 attributes:

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

methods	Character vector with the method names.
data	The data frame 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, and the generated BUGS code is printed to the console.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://BendixCarstensen.com, 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 )
ox <- Meth( ox )
# Writes the BUGS program to your console
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 ) # runs JAGS
#  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, which = 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'
mean(x, na.rm=TRUE, simplify=TRUE, ... )
## S3 method for class 'Meth'
sort(x, ... )
## 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.
`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.
`which`	A vector of indices or names of methods to plot. If `NULL` all methods in the object are plotted.
`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.
`na.rm`	Logical. Should `NA`s be removed before calculations?
`simplify`	Should a `Meth` object with one row per (meth,item) be returned?
`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 `mean.Meth` returns a `Meth` object where means have been computed over replicates, and put in a variable `mean.y`. If `simplify=TRUE`, a smaller `Meth` object will be returned with only one row per (meth,item), and the means in the variable `y`. This is useful if the definition of a particular measurement method is the mean of a specified number of replicate measuements. The functions `mean.Meth`, `median.Meth`, `max.Meth`, and `min.Meth` behaves similarly, whereas `sort.Meth` just sorts the replicates within each (meth,item), and puts the results in a variable `sort.y` added `Meth` object.

The `subset.Meth` returns a subset of the `Meth` rows. If a subset of the methods is selected, the new `meth` variable will have levels equal to the actually present levels of `meth` in the new `Meth` object. This is not the case if subsetting is done using "`[`".

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://BendixCarstensen.com

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, DA.reg, 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.comp = 1:2,
      pl.type = "conv",
     dif.type = "lin",
      sd.type = "const",
        axlim = range(x$data$y,na.rm=TRUE),
       diflim = axlim-mean(axlim),
       points = FALSE,
    repl.conn = FALSE,
     col.conn = "gray",
     lwd.conn = 1,
         grid = TRUE,
       h.grid = TRUE,
     col.grid = grey(0.9),
          lwd = c(3,1,1),
    col.lines = "black",
   col.points = "black",
   pch.points = 16,
   cex.points = 1,
          eqn = is.null(attr(x,"Transform")),
      col.eqn = col.lines,
     font.eqn = 2,
       digits = 2,
         mult = FALSE,
        alpha = NULL,
          ... )
## S3 method for class 'MethComp'
lines(x,
                   wh.comp = getOption("MethComp.wh.comp"),
```

```
                    pl.type = getOption("MethComp.pl.type"),
                   dif.type = getOption("MethComp.dif.type"),
                    sd.type = getOption("MethComp.sd.type"),
                  col.lines = "black",
                        lwd = c(3,1,1),
                     digits = 3,
                       mult = FALSE,
                      alpha = NULL,
                        ... )
## S3 method for class 'MethComp'
points(x,
                     wh.comp = getOption("MethComp.wh.comp"),
                     pl.type = getOption("MethComp.pl.type"),
                  col.points = "black",
                  pch.points = 16,
                  cex.points = 1,
                   repl.conn = FALSE,
                    col.conn = "gray",
                    lwd.conn = 1,
                        mult = FALSE,
                         ... )
```

Arguments

`obj`	A `MethComp` or `MCmcmc` object.
`x`	A `MethComp` object.
`wh.comp`	Numeric or character 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 1st minus the 2nd versus the average; a Bland-Altman type plot.
`dif.type`	Character. If "lin" (the default) a linear relationship between methods is allowed. Otherwise a constant difference is assumed and LoA can be indicated on the plot.
`sd.type`	Should the estimated dependence of the SD (from `DA.reg` be used when plotting prediction limits?
`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.
`repl.conn`	Logical. Should replcate measurements be connected; this assumes linked replicates.
`col.conn`	Color of the lines connecting replicates.
`lwd.conn`	Width of the connection lines.
`grid`	Should there be a grid? If numeric, how many gridlines? If a vector of length>1, it will be taken as the position of the vertical gridlines.
`h.grid`	Position of the horizontal gridlines. Ignored if `pl.type="conv"`.
`col.grid`	Color of the gridlines.
`col.lines`	Color of the conversion lines.
`lwd`	Numerical vector of length 3. Width of the conversion line and the prediction limits.
`pch.points`	Plot character for points.
`cex.points`	Character expansion for points.
`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
`digits`	The number of digits after the decimal point in the conversion formulae.
`mult`	Logical. Should ratios be plotted on a log-scale instead of differences on a linear scale? See description of the argument for `BA.plot`.
`alpha`	1 minus the confidence level for the prediction interval. If not given, the prediction interval is constructed as plus/minus twice the SD.
`...`	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 )
## Not run:
AR.ox <- AltReg( ox, linked=TRUE  )
print( AR.ox )
plot( AR.ox )
## End(Not run)
```

`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)) )
summary( ox )
# The effect of basing LoA on means over replicates:
par( mfrow=c(1,2), mar=c(4,4,1,4) )
BA.plot(      ox , diflim=c(-20,20), axlim=c(20,100), repl.conn=TRUE )
BA.plot( mean(ox), diflim=c(-20,20), axlim=c(20,100) )
```

`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 `Meth` object, alternatively a numeric vector of measurements by method A, or a data frame of exactly two columns, first column with measurements by method A, second column with measurements by method B.
`y`	a numeric vector of measurements by method B - must be of the same length as `x`. If not provided, `x` must be the `Meth` object or a data frame of exactly 2 columns.
`conf.level`	confidence level for calculation of confidence boundaries - 0.05 is the default.
`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`, `predict` 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.
`I`	A vector of all intercept estimates.
`adj`	A vector of fit parameters, where *Ss* is the number of estimated slopes (`length(S)`), *K* is the offset for slopes <(-1), *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`, `predict.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)
par(mfrow=c(2,2))
plot(PBmilk, s=1:4)
```

`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

```
## S3 method for class 'MCmcmc'
plot( 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, colors =
                list(CI = "#ccaaff50", fit = "blue", ref = "#99999955",
                bars = "gray", dens = "#8866aaa0", ref2 = c("#1222bb99",
                "#bb221299")), ...)
```

Arguments

`x`	an object of class `"PBreg"`
`pch`	Which plotting character should be used for the points.
`bg`	Background colour for the plotting character.
`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 5 x IQR for better visibility.
`colors`	A list of 6 elements allowing customization of colors of various plot elements. For plot subtype 1: "CI" is the color of the shaded confidence interval area; and "fit" is the color of fit line. For plot subtypes 2 & 3: "ref" is the color of the horizontal reference line. For plot subtype 4: "bars" is the bar background color, "dens" is the color of the density line, and "ref2" is a vector of two colors for lines indicating the median and confidence limits.
`...`	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)

## Or the same using "Meth" object
a <- Meth(a, y=1:2)
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

```
## S3 method for class 'VarComp'
plot( 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)
```

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

Description

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

Usage

```
## S3 method for class 'PBreg'
predict(object, newdata = object$model$x, interval="none",
  level=0.95,...)
```

Arguments

`object`	an object of class `"PBreg"`
`newdata`	an optional vector of new values of `x` to make predictions for. If omitted, the fitted values will be used.
`interval`	type of interval calculation - either `confidence` or `none`. The former is the default.
`level`	tolerance/confidence level.
`...`	Not used.

Value

If `interval` is `"confidence"` this function returns a data frame with three columns: ”fit”, ”lwr” and ”upr” - similarly to `predict.lm`.

If `interval` is `"none"` a vector of predicted values is returned.

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)
predict(x, interval="none")

## Or the same using "Meth" object
a <- Meth(a, y=1:2)
x <- PBreg(a)
print(x)
predict(x)
```

`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, wh.comp=1:2 )
BA.plot( sbp, wh.comp=2:3 )
BA.plot( sbp, wh.comp=c(1,3) )
## Not run:  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 taken 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, program="JAGS" ) )
```

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 convergence 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 components are the same between methods:
## Not run:
pairs.MCmcmc( sbp.MC, subset=c("mi"), eq=TRUE,
```

```
                panel=function(x,y,...)
                      {
                      abline(0,1)
                      abline(v=median(x),h=median(y),col="gray")
                      points(x,y,...)
                      }
               )
## End(Not run)
```

`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=TRUE )
to.long( data, vars )
```

Arguments

`data`	A `Meth` object.
`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. Also recognizes a

Value

A dataframe.

Author(s)

Bendix Carstensen, Steno Diabetes Center, http://BendixCarstensen.com

See Also

`perm.repl`

Examples

```
data( milk )
str( milk )
mw <- to.wide( milk )
str( mw )
( mw <- subset( mw, as.integer(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 )
```