Statistical Analysis of Method Comparison Studies

Version 0.2 (compiled February 16, 2009)

University of Adelaide
Thursday 19 February 2009

www.biostat.ku.dk/~bxc/MethComp/Courses/Adel.2009

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

Contents

Workshop program **5**

1 Introduction to computing **7**
1.1 Software . . . 7
1.1.1 Installation . . . 7
1.1.2 The `MethComp` package . . . 7
1.1.2.1 R and `BRugs` / `R2WinBUGS` . . . 8
1.1.2.2 Using `WinBUGS` from `MCmcmc` . . . 8

2 Introduction to the `MethComp` package **11**
2.1 Data structures . . . 11
2.2 Function overview . . . 11
2.2.1 Graphical functions . . . 12
2.2.2 Data manipulating functions . . . 12
2.2.3 Analysis functions . . . 12
2.2.4 Reporting functions . . . 13

3 Practical exercises **15**
3.1 Milk: Single measurements by two methods . . . 15
3.2 Fat measurements: Exchangeable replicates . . . 17
3.3 Oximetry: Linked replicates with non-constant bias . . . 19
3.4 Oximetry: Transformation . . . 22

4 Solutions to exercises **25**
4.1 Milk: Single measurements by two methods . . . 25
4.2 Fat measurements: Exchangeable replicates . . . 32
4.3 Oximetry: Linked replicates and non-constant bias . . . 38
4.4 Oximetry: Transformation . . . 49

Bibliography **57**

5 `MethComp` manual (version 0.5.2) **59**
abconv . . . 59
AltReg . . . 60
BA.est . . . 61
BlandAltman . . . 63
bothlines . . . 65
cardiac . . . 66
check.MCmcmc . . . 67

corr.measures . . . 68
DA.reg . . . 69
Deming . . . 70
Enzyme . . . 72
fat . . . 72
glucose . . . 73
hba.MC . . . 74
hba1c . . . 74
MCmcmc . . . 75
Meth.sim . . . 78
Meth . . . 79
milk . . . 81
ox.MC . . . 82
ox . . . 82
PEFR . . . 83
perm.repl . . . 84
plot.MCmcmc . . . 85
plot.VarComp . . . 86
plvol . . . 87
sbp . . . 88
scint . . . 89
TDI . . . 89
to.wide . . . 90
VitCap . . . 91

Index **92**

Workshop program

Thursday 19th February 2009

09:00 – 09:15	**Registration and introduction**
09:15 – 10:30	**Lecture**: Measurement: Repeatability, reproducibility. Comparing two methods without replicates. Comparing methods with replicate measurements.
10:30 – 11:00	**Morning Tea**
11:00 – 12:30	**Practical**: Plotting data, Bland-Altman-plot, limits of agreement. Wide and long representation of data. Analysis using means and using single replicates. Comparing the two approaches. Exercises: Milk & Fat.
12:30 – 13:15	**Lunch**
13:15 – 14:15	**Lecture**: A general model for analysis of method comparison studies with replicate measuremenst and non-constant bias. Practical approaches to analysis and reporting.
14:15 – 15:00	**Practical**: Use of the methods introduced in the seminar using the `BA.est`, `AltReg` and `MCmcmc` function to analyze method comparison studies with different exchangeability structure. Exercises: Oximetry & transformation.
15:00 –	**Afternoon Tea**

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

1.1 Software

The most convenient software for desk-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 contains extra functions. The more advanced models covered in this course are only implemented in R in a special package `MethComp`.

In order to be able to write scripts (programs) in R and keep them for future use (and modification for other purposes) a good editor with an interface to R is convenient. `TinN-R` is the answer. (`TinN` = Tinn `is` not Notepad). R also has a built-in text editor which is a bit more primitive; it is accessed via File → Open script or File → New script.

1.1.1 Installation

R can be obtained from www.r-project.org. Click on `CRAN`, choose a mirror (that is, from where you want to download it), click on the link to Windows and after that choose `base`. Download `R-2.8.1-win32.exe` to your computer, and run this installation file.

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

```
install.packages( c("R2WinBUGS","coda","BRugs","Epi") )
```

This will install the four mentioned packages provided you are connected to the net. Alternatively you can clik 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 really not essential for this course.

1.1.2 The `MethComp` package

Finally you will have to install the (still) non-official package for R, `MethComp`[1], which contains all the functions for analysis of method comparison studies. It is available from

[1] It will soon be an official package for R but it has only been under development during the last year or so.

http://staff.pubhealth.ku.dk/~bxc/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 file `MethComp_0.5.1.zip` and unpack it in the folder `c:\Program Files\R\R-2.8.1\library` (or wherever you have installed R).

The function `MCmcmc` from this package uses Markov chain simulatioin (MCMC) for estimation; you can choose to use either `BRugs` or `WinBUGS` for the MCMC-sampling using the argument `program=`. This can be set to either `BRugs` or `WinBUGS` — see the help page for the documentation. The default for `MCmcmc` is to use the `BRugs` package if installed. In most cases this will be the simplest option.

If you are not deeply interested in the funtionaing of the different versions of `BUGS`that are used by `MCmcmc` you can safely skip the next two sections.

1.1.2.1 R and `BRugs` / `R2WinBUGS`

`BUGS` (Bayesian inference Using Gibbs Sampling) is a programming language for specification of models that allow description in hierarchical terms, specifically as directed acyclic graphs (DAGs). It was first released in the 1990s for a Unix platform, but is now available in many guises for various platforms. `BUGS` is the generic name for any of these.

Three versions of `BUGS` are accessible from within R: `WinBUGS`, `openBUGS` and `JAGS`; we shall only be concerned with the two first ones here. The R package that allows the user to access `BUGS` from within R is `R2WinBUGS`.

`BUGS` has a special programming language so `BUGS` code statements need to be specified in a separate file.

`WinBUGS` is a stand-alone program, whereas `openBUGS` comes packaged for R in the R-package `BRugs`. The package `R2WinBUGS` has interfaces to both `WinBUGS` and `BRugs`, and although they use the same syntax etc. the output from the two are slightly different.

`BUGS` is used from the `MCmcmc` function, but all the writing of programs and postprocessing of results is taken care of by the function, so the only thing you really need is to specify whether `MCmcmc` is to use `BRugs` or `WinBUGS` for the MCMC-simulation and in the latter case the location where `WinBUGS` is installed.

1.1.2.2 Using `WinBUGS` from `MCmcmc`

`WinBUGS` can be obtained from the `WinBUGS` homepage http://www.mrc-bsu.cam.ac.uk/bugs. `WinBUGS` will only work if you have a certificate which is free. To obtain one, register at the `WinBUGS` homepage and you will get an e-mail with the certificate and which tells you how to install the certificate.

If you specify `program=WinBUGS` there will be a call to `WinBUGS`, and therefore the place on your computer where `WinBUGS` is installed must be supplied. That can either be done in the call to the function:

```
MCmcmc( ..., bugs.directory="c:/Program Files/WinBUGS14" )
```

(or wherever you installed `WinBUGS`).

The default for `MCmcmc` is to look for the R-option `bugs.directory`. Therefore, if you start your R-session by saying:

```
options(bugs.directory="c:/Program Files/WinBUGS14")
```

you don't have to bother about this any more in you session.

Even more sophisticated, you can add the line defining the option to the file `.Rprofile` which you find in the folder `c:\Program Files\R\R-2.8.1\etc`. Then R will automatically set this option every time you fire it up.

Chapter 2

Introduction to 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 lives off a particular structure of data.

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`.

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 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`.

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 printed manual at the end of this document or 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 )
```

2.2.1 Graphical functions

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

`BlandAltman` draws a Bland-Altman plot and computes limits of agreement, assuming that data are supplied as two vectors.

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

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

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.

`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 no. replicates for a `Meth` object.

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

2.2.3 Analysis functions

`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.

`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 [1].

`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`.

2.2.4 Reporting functions

Some of these functions all take a `MCmcmc` object as input, others will postprocess the output of `DA.reg`, `BA.est` or `AltReg`[1].

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

`plot.MCmcmc` Plots the conversion lines between methods 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.

[1] It is the intention to collect the results of these function in a single class, `MethComp`, with a common set of reporting functions that automatically recognize where the result came from.

Chapter 3

Practical exercises

3.1 Milk: Single measurements by two methods

The purpose of this exercise is to assess to what degree two methods can be used interchangeably, or rather to quantify how much they differ, so that an informed clinical decision can be made as to which one is preferable. Moreover we will illustrate various ways of relating the two methods to each other.

The `milk` data from the `MethComp` package contains measurements of fat content of human milk (g/100 ml) determined by the measurement of glycerol released by enzymatic hydrolysis of triglycerides (Trig) and measurements by the standard Gerber method (Gerber).

Load the dataset and take a look at its structure:

```
> data(milk)
> str(milk)
```

You can get a bit more substantial insingt by typing `?milk`.

The data is arranged in the long form, i.e. with one measurement per line and two variables, item and method. If you want to have the two methods beside each other, you can use the `to.wide` function:

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

The purpose of this exercise is to assess to what degree the two methods can be used interchangeably, or rather to quantify how much they differ, so that an informed clinical decision can be made as to which one is preferable.

Also it will introduce some ways that you can display data with the facilities in the `MethComp` package.

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)
   ```

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

```
> BA.plot(milk)
> BA.plot(milk,ymax=0.5)
```

You will want to have a look at the help page for `BA.plot` and also for `BlandAltman` which is the function that really does the plotting. Note that options from `BA.plot` are passed on to the function `BlandAltman`.

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 which.

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 may want to consult the `DA.reg` function for this purpose).

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.

3.2 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.

1. Load the dataframe `fat` and examine the names in the dataframe:

   ```
   > 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`. You would for example do something like:

   ```
   > vis <- Meth( fat, c(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 invole `plot.Meth` when the arument is os 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 accomodate the actual replication struture 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; that is an upper 95% confidence interval 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.

9. Finally, try to see what happens if you base the limits of agreement on the means over the averages. The function `BA.plot` has a facility for this type of calculation.

3.3 Oximetry: Linked replicates with non-constant bias

The `ox` data from the `MethComp` package contains data from 61 children who had their blood oxygen content measured using two methods at the Royal Children's Hospital in Melbourne. The standard chemical method analysing gases in the blood based on co-oximetry (named "CO") is to be compared to a new method using a pulse oximeter to measure light reflectance transcutaneously (named "pulse"). Most children have three replicates on each method, which are linked, so replicate 1 for each of the two methods is done at the same time. Replicate measurements were taken in quick succession, so we assume that the linked pairs of measurements are exchangeable within person.

The purpose of this exercise is to demonstrate the facility in the `MethComp` package to estimate the variance between linked replicates (the item by replicate effect) while allowing for a random method by item effect and differing residual variances between methods. We also consider the possibility of 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 replicares 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` obejct, 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. In particular be aware that the estimates are the standard deviation of the random effects, and hence are on the same scale as the original data.

5. 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 measurements by the same method; separately for each of the methods.

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 wheter 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 regressing the absolute values of the residuals from this first 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:

   ```
   > MC0 <- MCmcmc( ox, bias="const", random=c("mi","ir"), n.iter=5000 )
   ```

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

   ```
   > print(MC0)
   ```

10. 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(MC0, 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):

    ```
    > post(MC0)
    ```

    Are the residual variances equal?

11. Expand the model to allow for non-constant bias. This is the default option for `MCmcmc`, so you may omit the `bias` argument:

    ```
    > MC1 <- MCmcmc( ox, bias="lin", random=c("mi","ir"), n.iter=5000 )
    ```

    Summarize the results of the `MethComp` fit and use the `plot.MethComp` function to display the equations relating the mean measurements on each method as above.

    ```
    > print(MC1)
    > plot(MC1, pl.obs = TRUE)
    ```

    Is $\beta_{2|1}$ different from 1.00?

12. What are the implications for comparing oximetry measurements made on the same infant?

3.4 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. Therefore, 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:

   ```
   > round( ftable( DA.reg( oxt ) ), 3 )
   ```

   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:

   ```
   > ( LoAt <- BA.est( oxt )$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. Instead try to plot the two methods against each other on the original scale, and then superpose the estimated conversion lines from the model.

   The model we have is:

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

   This leads to a prediction of one method from the other as:

   $$y_{\text{CO}|\text{pulse}} = \alpha_{\text{CO}} - \alpha_{\text{pulse}} + y_{\text{pulse}} \pm 2\sqrt{\tau^2_{\text{CO}} + \tau^2_{\text{pulse}} + \sigma^2_{\text{CO}} + \sigma^2_{\text{pulse}}}$$

   Use this set of conversion lines ($y \pm 2 \times$ s.d.) on the logit-scale, to draw the corresponding curves on the original %-saturation scale.

   (Hint: Work out a set of say 100 $x$es and ys on each line on the logit scale, and then transform them all by the inverse logit and plot them as curves.)

   How do the conversion lines (curves, really) capture the actual datapoints as compared to the limits based on the original untransformed data?

7. Now try to see if a log-transform of the data works as well.

8. 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.

   Which of the transformations would you prefer — and on what grounds?

9. 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 4

Solutions to exercises

4.1 Milk: Single measurements by two methods

First we load the dataset and take a look at its structure:

```
> 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  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ id    : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Trig  : num  0.96 1.16 0.97 1.01 1.25 1.22 1.46 1.66 1.75 1.72 ...
 $ Gerber: num  0.85 1 1 1 1.2 1.2 1.38 1.65 1.68 1.7 ...
 - attr(*, "reshapeWide")=List of 5
  ..$ v.names: chr "y"
  ..$ timevar: chr "meth"
  ..$ idvar  : chr "id"
  ..$ times  : Factor w/ 2 levels "Gerber","Trig": 2 1
  ..$ varying: chr [1, 1:2] "Trig" "Gerber"

> head(mw)

  item id Trig Gerber
1    1  1 0.96   0.85
```

```
2     2  2 1.16   1.00
3     3  3 0.97   1.00
4     4  4 1.01   1.00
5     5  5 1.25   1.20
6     6  6 1.22   1.20
```

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(3,3,3,3)) # slightly nicer look to the graph
> with( mw, plot( Trig ~ Gerber, pch=16,
+                 xlim=range(milk$y), ylim=range(milk$y) ) ) # Note: identical axes
> abline(0,1)
```

   The last statement just adds the identity line.

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)
> str(milk)
```

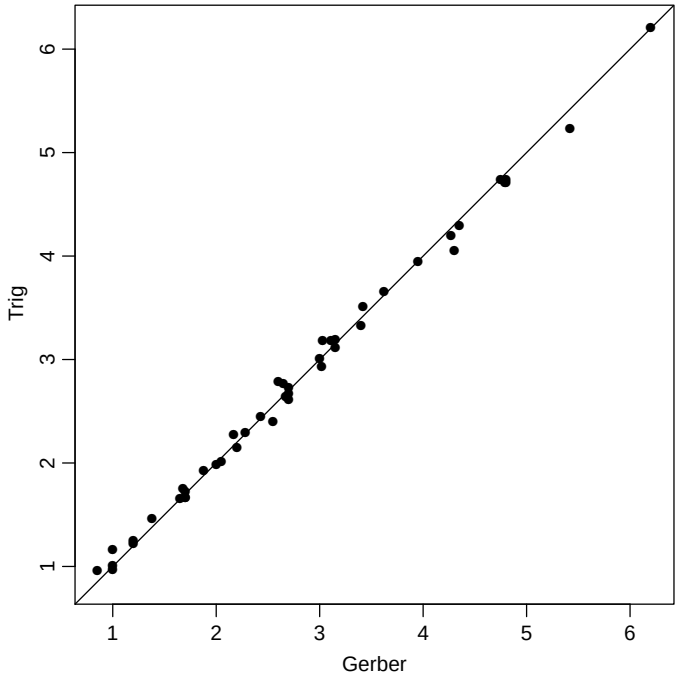

Figure 4.1: *Scatter plot of the* `milk` *data.*

```
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  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:

```
> BA.plot(milk)

Limits of agreement:
Trig - Gerber    2.5% limit   97.5% limit      SD(diff)
-0.0002222222 -0.1748120735  0.1743676290  0.0872949256
```

or, in a slightly nicer form:

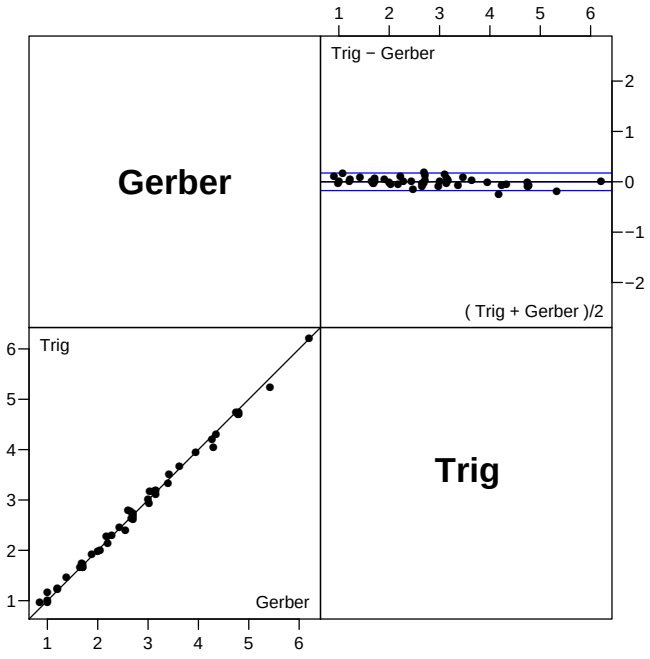

Figure 4.2: *Overview plot of the* `milk` *data, using* `plot.Meth()`*, i.e. the generic method for* `Meth` *objects.*

```
> par(mgp=c(3,1,0)/1.6, mar=c(3,3,3,3))
> BA.plot(milk,ymax=0.5)

Limits of agreement:
Trig - Gerber    2.5% limit   97.5% limit      SD(diff)
-0.0002222222 -0.1748120735  0.1743676290  0.0872949256
```

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

5. This means that the difference between 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
```

   Strangely enough, the slope is significantly different from 1, although the resulting relationship is not impressive. 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:

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

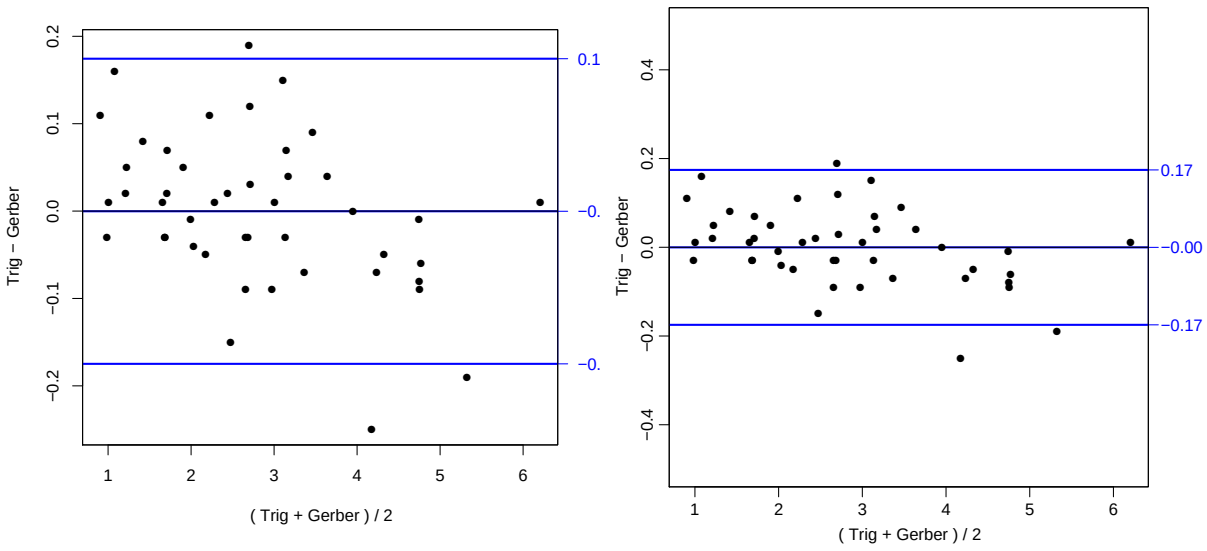

Figure 4.3: *Bland-Altman plots of the* `milk` *data, left panel with the same extent of the data on both axes, the right one with explicitly defined y-axis and explicitly defined margins — note how the right hand margin on the left plot is too narrow to accommodate the LoA.*

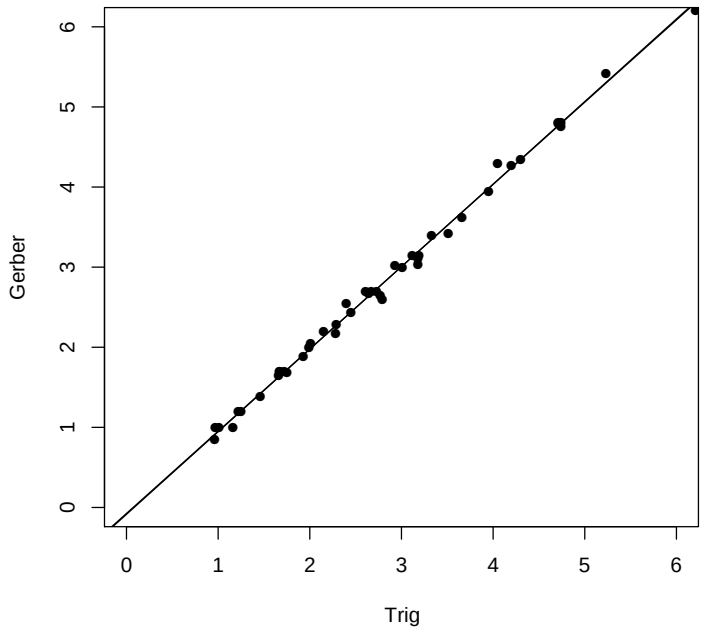

Figure 4.4: *Scatter plot of data with the two different regression lines. They are practically indistinguishable.*

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 returns a 3-dimensional array. Therefore it is desirable to show the output using `ftable` (flat `table`):

```
> round( ftable( DA.reg(milk) ), 3 )

                alpha   beta sd.pred beta=1 s.d.=K
From:  To:
Gerber Gerber   0.000  1.000      NA     NA     NA
       Trig     0.078  0.972   0.079  0.005  0.383
Trig   Gerber  -0.080  1.029   0.081  0.005  0.383
       Trig     0.000  1.000      NA     NA     NA
```

The `alpha` and `beta` columns are intercept and slopes relating the two methods based on the regression of th eidfferences 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 last two columns of the output here are p-values for the hyptheses of slope equal to 1 and constant standard deviation across the range of mesuremensts.

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 0.02980128  -2.502167 1.622649e-02
Trig         1.02667683 0.00970774 105.758594 1.323266e-53
```

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

```
> with( mw, plot( Trig, Gerber, pch=16, xlim=c(0,6), ylim=c(0,6) ) )
> with( mw, bothlines( Trig, Gerber ) )
```

   The regression lines are virtually indistinguishable.

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 ) )

   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 eqaul.

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 `BlandAltman`), with the argument `reg.line=` — a number giving the number of decimals to be used for the display of the resulting conversion equations.

```
> BA.plot( milk, reg.line=3, limy=c(-0.5,0.5) )

Limits of agreement:
Trig - Gerber    2.5% limit   97.5% limit      SD(diff)
-0.0002222222 -0.1748120735  0.1743676290  0.0872949256

Trig-Gerber = 0.079 - 0.028 (Trig+Gerber)/2 (95% p.i.: +/-0.161)
 res.sd = 0.080    se(beta) = 0.009 , P = 0.0046

 Gerber = -0.080 + 1.029 Trig (95% p.i.: +/-0.163)
 Trig = 0.078 + 0.972 Gerber (95% p.i.: +/-0.158)
```

   The regression lines are virtually indistinguishable.

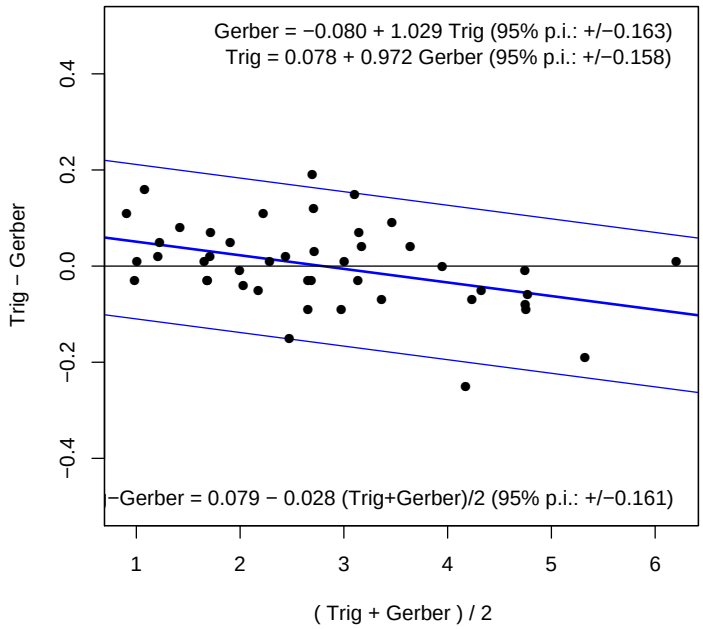

Figure 4.5: *Bland-Altman plot of the* `milk` *data with the regression of the differences on the averages and the resulting conversion equations between methods.*

4.2 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.

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

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

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

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

Classes 'Meth' and 'data.frame':        258 obs. of  4 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 ...

> 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 replicate number for pairing the measurements; so we just keep the ordering among the replicates when using `to.wide`:

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

Note:
 Replicate measurements are taken as separate items!

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

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.Meth( vis )

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

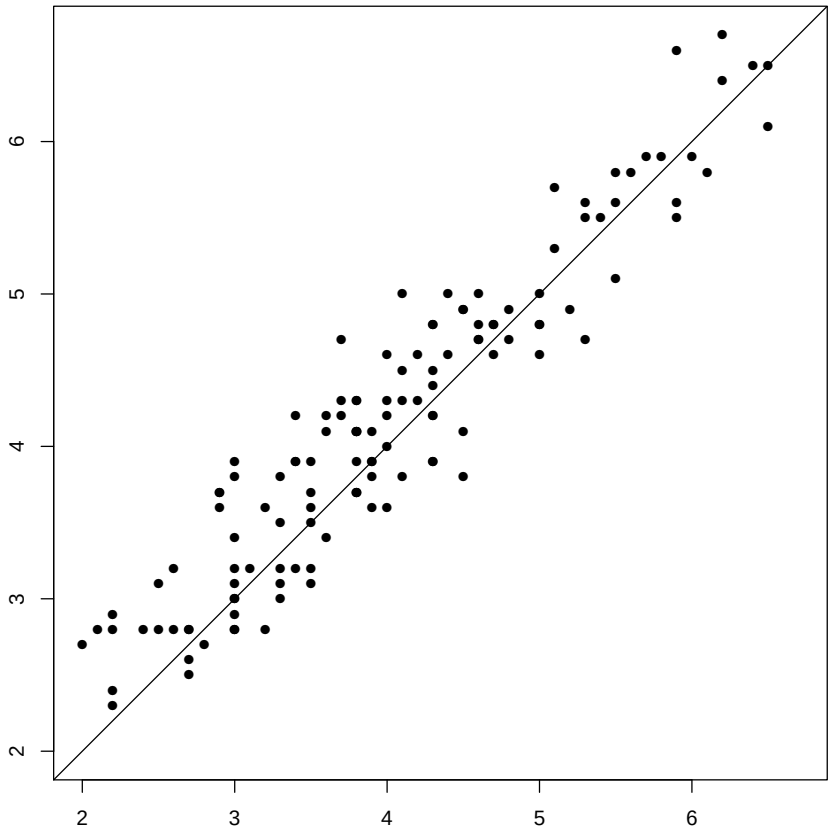

Figure 4.6: *Two observers measuring visceral fat.*

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

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

These two plots are shown in figure 4.7 where it is pretty clar 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)

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

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

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). For convenience we `flat-table` and round the result:

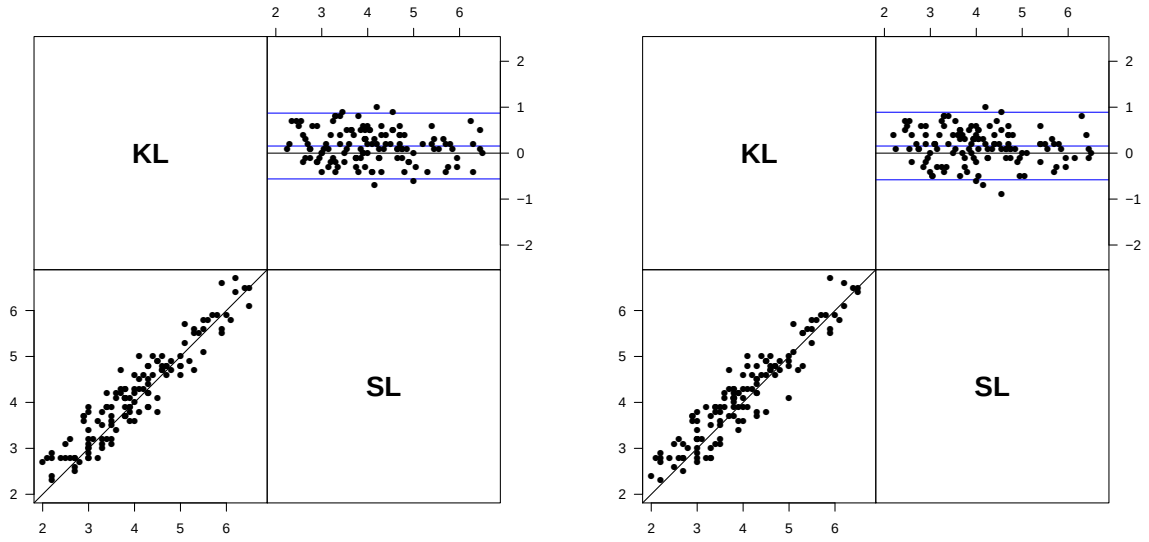

Figure 4.7: *Plot of two methods of measuring visceral fat, using different pairings of the replicates; the left panel is using the pairing in the original coding, the right panel is with a random permutation of replicates.*

```
> round( ftable( DA.reg( vis ) ), 3 )

             alpha   beta sd.pred beta=1 s.d.=K
From: To:
KL    KL     0.000  1.000      NA      NA      NA
      SL     0.326  0.957   0.349   0.158   0.275
SL    KL    -0.340  1.044   0.365   0.158   0.275
      SL     0.000  1.000      NA      NA      NA
```

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

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 ) )

$Bias
       KL        SL
0.0000000 0.1550388

$VarComp
   IxR       MxI       res
KL   0 0.1806773 0.1926961
SL   0 0.1806773 0.1732051

$LoA
               Mean      Lower    Upper        SD
SL - KL   0.1550388 -0.5727534 0.882831 0.3638961

$RepCoef
          SD     Coef.
```

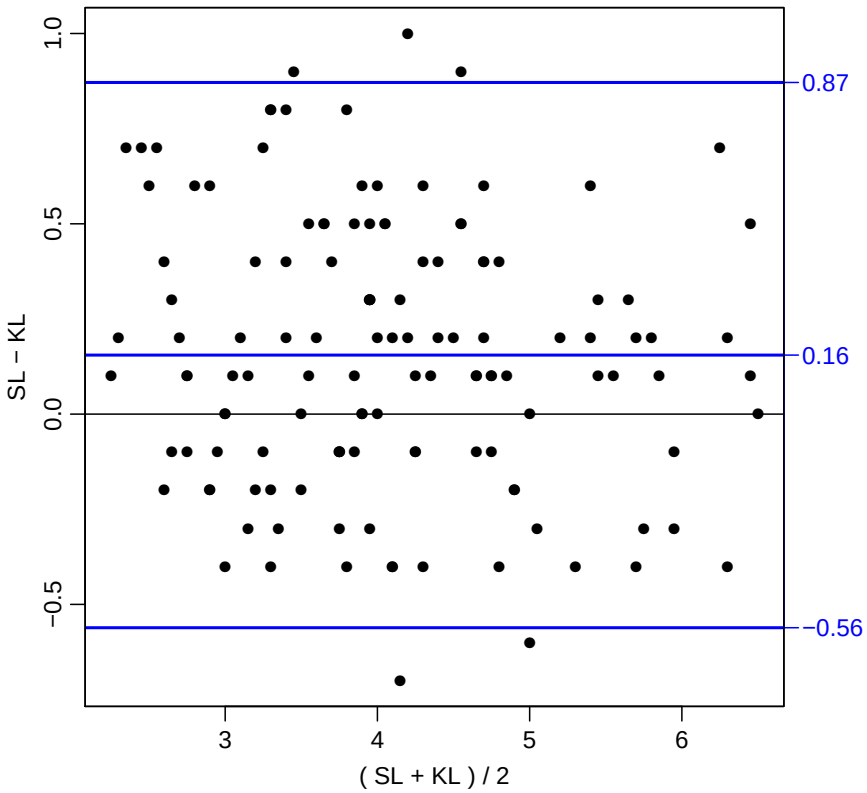

Figure 4.8: *Bland-Altman plot of two observers measuring visceral fat.*

```
KL 0.2725134 0.5450269
SL 0.2449490 0.4898979
```

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

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

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

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

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

   As predicted by the theory, the limits based on the *ad-hoc* paired replicates are roughly equal to those derived from the proper variance component model — see figure 4.9.

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 over plot:

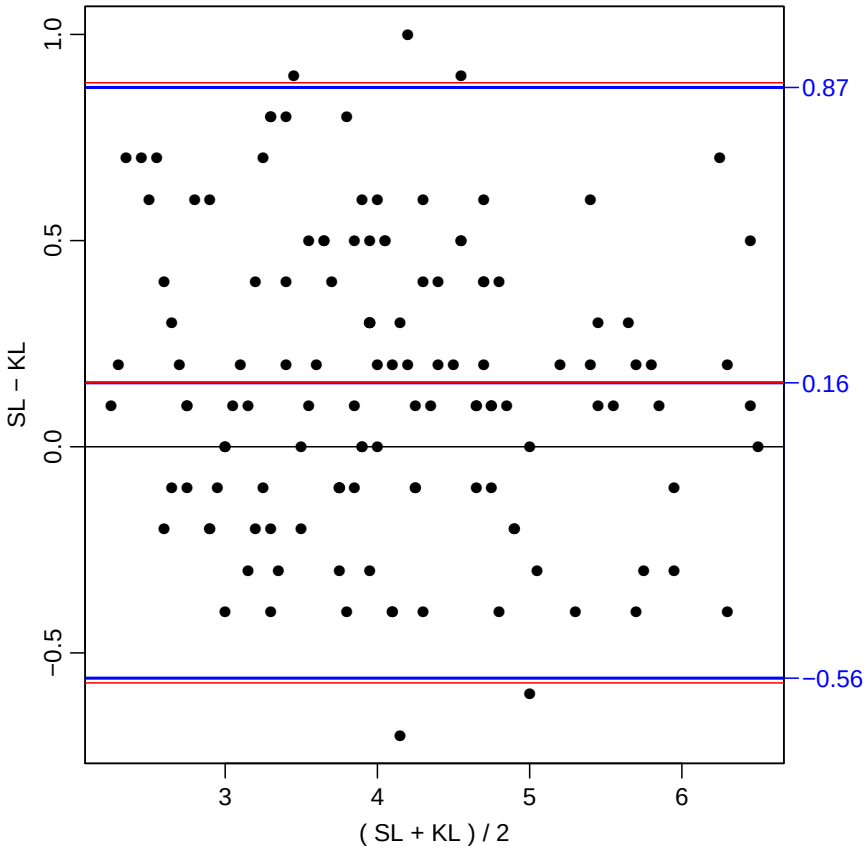

Figure 4.9: *Bland-Altman-plot of two methods of measuring visceral fat, using different pairings of the replicates. The blue lines are the LoA based on taking the paired replicates as items, the red lines are based on the estimates from the proper variance component model.*

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

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

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

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

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

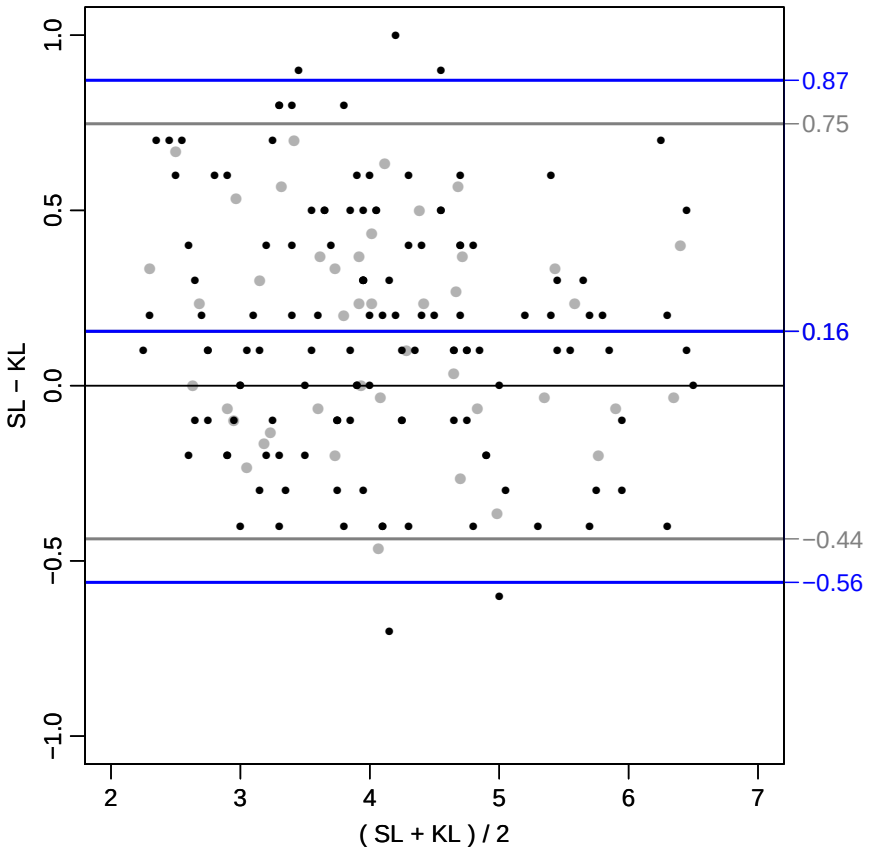

Figure 4.10: *Bland-Altman-plot of two methods of measuring visceral fat, based on the arbitrary pairing of the replicates (black) and on the mean over replicates (grey).*

4.3 Oximetry: Linked replicates and non-constant bias

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

```
> 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 )
> 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.

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 )

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

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 4.12:

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

Limits of agreement:
 pulse - CO  2.5% limit 97.5% limit    SD(diff)
  -2.477401  -14.828597    9.873795    6.175598
```

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) )

$Bias
       CO     pulse
 0.000000 -2.470446

$VarComp
           IxR      MxI      res
CO    3.415692 2.928042 2.224868
pulse 3.415692 2.928042 3.994451

$LoA
                   Mean     Lower    Upper       SD
pulse - CO   -2.470446 -14.80779 9.866901 6.168674
```

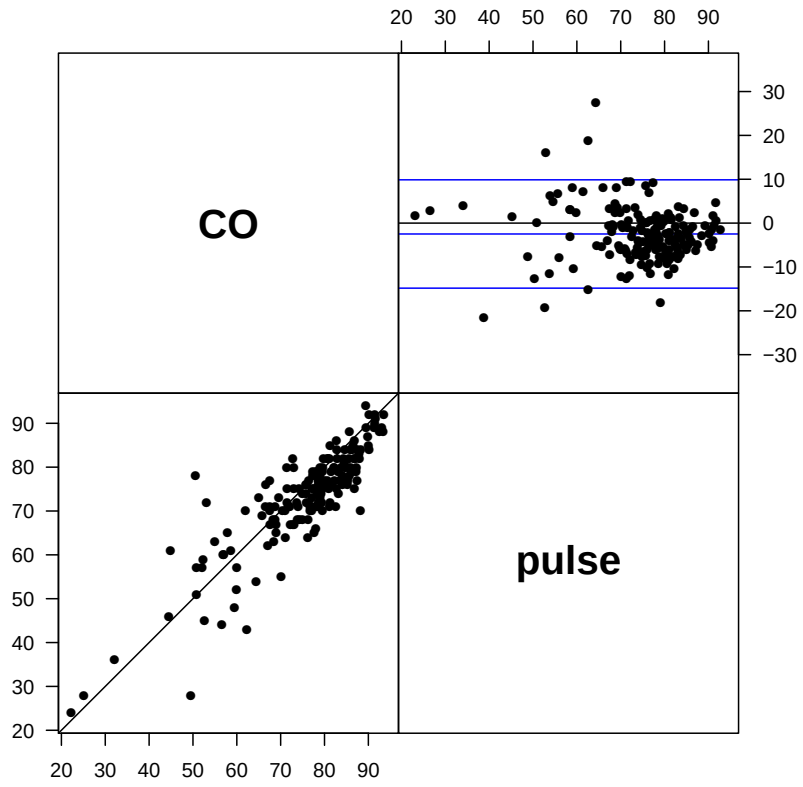

Figure 4.11: *A scatterplot (lower left) and Bland-Altman plot (upper right) of the oximetry data, using the linked replicates as items.*

```
$RepCoef
             SD    Coef.
CO    5.764892 11.52978
pulse 7.432710 14.86542
```

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 differnece between two 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.

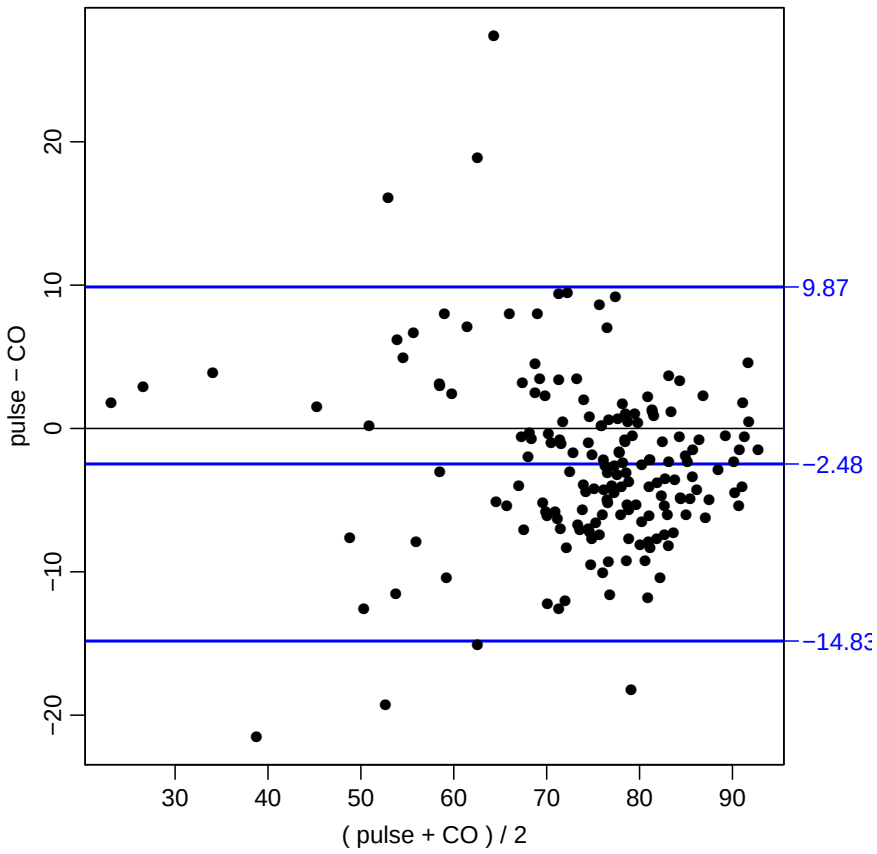

Figure 4.12: *A Bland-Altman plot of the oximetry data, using the linked replicates as items.*

Hence, the upper confidence limit for the absolute differnce 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 )

AltReg uses 354 obs. out of 354 in the supplied data.


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


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


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


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


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

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


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


iteration 8 criterion: 0.007169339
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse IxR sd. MxI sd.
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.sd.
CO      2.087
pulse   4.064


iteration 9 criterion: 0.005073329
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse IxR sd. MxI sd.
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.sd.
CO      2.077
pulse   4.069


iteration 10 criterion: 0.003706483
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse IxR sd. MxI sd.
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.sd.
CO      2.069
pulse   4.073


iteration 11 criterion: 0.002686239
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse IxR sd. MxI sd.
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.sd.
CO      2.064
pulse   4.075


iteration 12 criterion: 0.001930229
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse IxR sd. MxI sd.
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.sd.
```

```
CO      2.060
pulse   4.077


iteration 13 criterion: 0.001381185
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse IxR sd. MxI sd.
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.sd.
CO      2.057
pulse   4.078


iteration 14 criterion: 0.000986339
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse IxR sd. MxI sd.
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.sd.
CO      2.055
pulse   4.079
```

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

```
> round( ARox, 4 )

       From
To      Intercept: CO   pulse Slope: CO  pulse IxR sd. MxI sd. res.sd.
  CO           0.0000 -2.1591    1.0000 1.0629  3.5210  2.9785  2.0548
  pulse        2.0314  0.0000    0.9409 1.0000  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[,5:7] / BAox$VarComp, 4 )

       From
To      IxR sd. MxI sd. res.sd.
  CO     1.0308  1.0172  0.9235
  pulse  0.9699  0.9571  1.0212
```

Clarly, 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 apprimately formal assessment of whether the slopes are 1 and wheter the variance is constant from the regression of the differences on the avrages, using `DA.reg`:

```
> ftable( DA.reg( ox ) )

                     alpha          beta       sd.pred        beta=1        s.d.=K
From: To:
CO    CO      0.000000e+00  1.000000e+00            NA            NA            NA
      pulse   1.863840e+00  9.426203e-01  5.978569e+00  1.424526e-01  1.496146e-06
pulse CO     -1.977297e+00  1.060873e+00  6.342499e+00  1.424526e-01  1.496146e-06
      pulse   0.000000e+00  1.000000e+00            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. Running the `MCmcmc` routine and using the corresponding `print` function produces the following output:

```
> MC0 <- MCmcmc( ox, bias="const", random=c("mi","ir"), n.iter=500 )

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 500 iterations
  (of which 250 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 1000 observations.

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4:

> print(MC0)

Conversion formula:
 y_to = alpha + beta * y_from +/- 2*sd.pred:

      From:     CO                  pulse
             alpha   beta sd.pred  alpha   beta sd.pred
To:
CO           0.000  1.000   3.007  2.435  1.000   4.644
pulse       -2.435  1.000   4.644  0.000  1.000   5.800

Variance components with 95 % cred.int.:
                     50%  2.5%   97.5%
sigma.ir[CO]     144.528 2.881 594.930
sigma.ir[pulse]  144.528 2.881 594.930
sigma.mi[CO]     131.977 2.425 590.431
sigma.mi[pulse]  131.977 2.425 590.431
sigma.res[CO]      2.126 0.525   3.380
sigma.res[pulse]   4.101 3.059   4.963
sigma.tot[CO]    210.108 4.680 816.550
sigma.tot[pulse] 210.693 5.502 816.561

Mean parameters with 95 % cred.int.:
                   50%    2.5%  97.5% P(>0/1)
alpha[pulse.CO] -2.435  -8.016 33.494    0.25
alpha[CO.pulse]  2.435 -33.494  8.016    0.75
beta[pulse.CO]   1.000   1.000  1.000    0.00
beta[CO.pulse]   1.000   1.000  1.000    0.00

 Note that intercepts in conversion formulae are adjusted to get
 conversion fromulae that represent the same line both ways,
```

```
- therefore are the median of the alphas above not identical
to the intercepts given in the conversion formulae.
```

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):

```
> plot( MC0, pl.obs=TRUE )
```

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(MC0))
```

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.

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

```
> MC1 <- MCmcmc( ox, bias="lin", random=c("mi","ir"), n.iter=500 )

Comparison of 2 methods, using 354 measurements
on 61 items, with up to 3 replicate measurements,
```

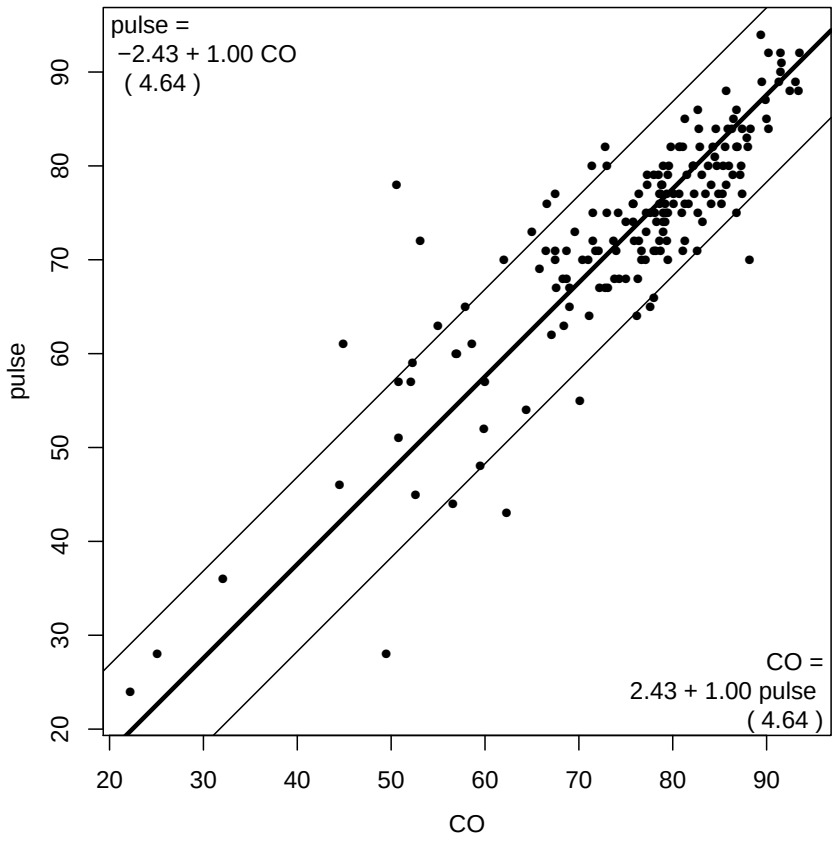

Figure 4.13: *A scatterplot of the oximetry data with the linear equations displayed. The slope of the linear relationship between methods has been constrained to 1.00.*

```
(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 500 iterations
  (of which 250 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 1000 observations.

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4:
```

11. In order to be reasonably sure about the validity of inference based on the mcmc-estiamtes we should check tha 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( MC1 ) )
    ```

12. Once we have established that the mixing of the chains is satisfactory, and hence that we

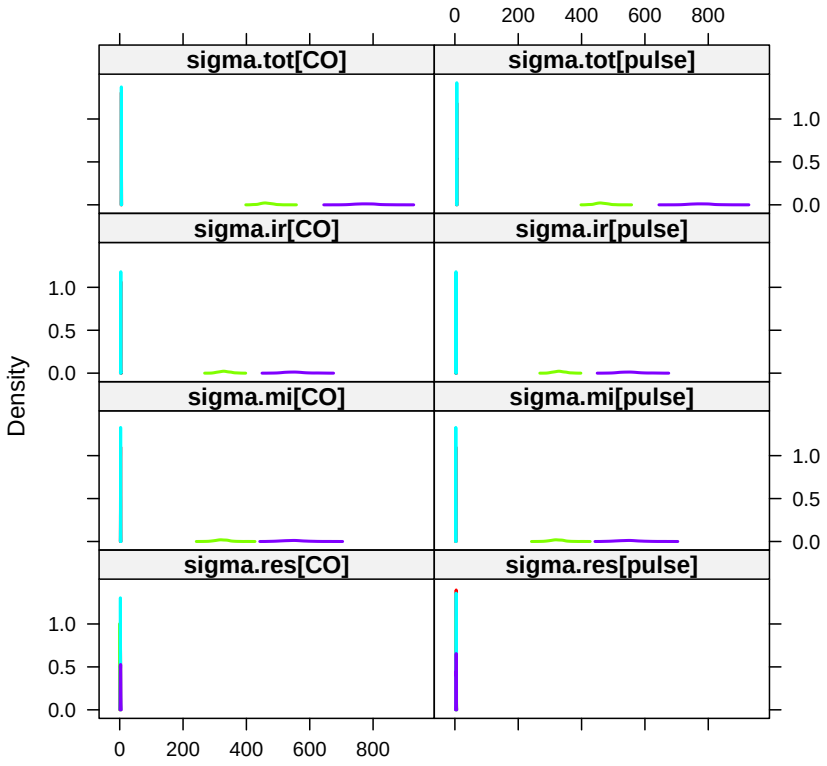

Figure 4.14: *Smoothed density plots of the variance components estimated using* `MethComp`.

are willing to accpt that the samples are samples from the statitionary distribution i.e. the correct posterior, we can can use the samples to derive estimates as posterior medians:

```
> print( MC1 )

Conversion formula:
 y_to = alpha + beta * y_from +/- 2*sd.pred:

      From:       CO                      pulse
               alpha    beta sd.pred   alpha    beta sd.pred
To:
CO             0.000   1.000   2.423 -10.333   1.169   5.276
pulse          8.839   0.855   4.563   0.000   1.000   6.104

Variance components with 95 % cred.int.:
                    50%  2.5%  97.5%
sigma.ir[CO]      3.815 3.017 13.625
sigma.ir[pulse]   3.376 2.617 11.174
sigma.mi[CO]      3.207 2.367  7.117
sigma.mi[pulse]   2.761 1.962  6.512
sigma.res[CO]     1.713 0.174  2.711
sigma.res[pulse] 4.316 3.661  5.054
sigma.tot[CO]     5.486 4.703 14.023
sigma.tot[pulse] 6.260 5.452 12.373

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

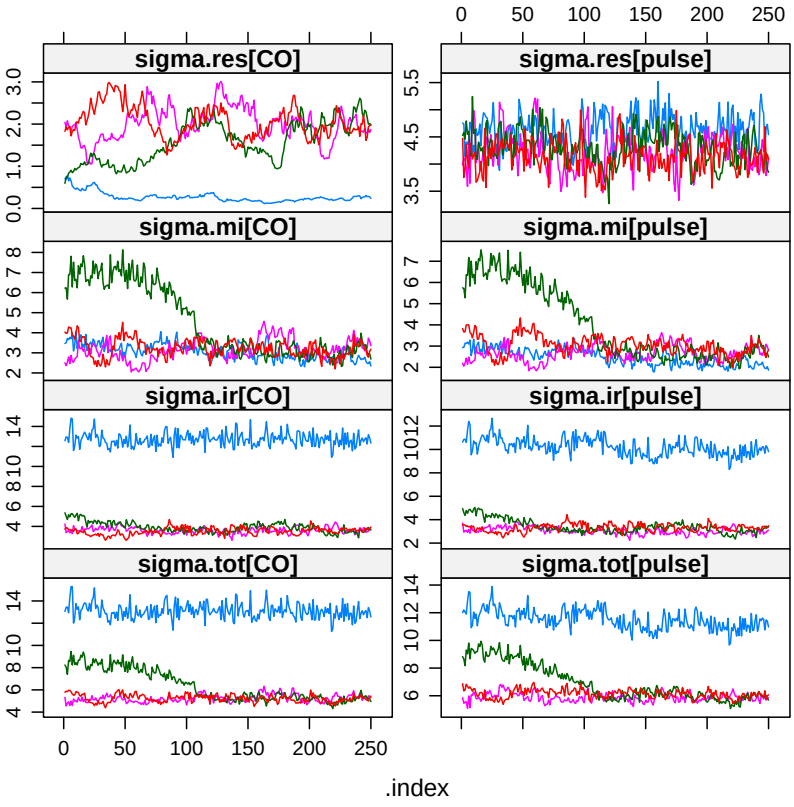

Figure 4.15: *Traces of the chains for the variance components estimated using* `MCmcmc`.

```
alpha[pulse.CO]   8.841  -0.770 16.737   0.956
alpha[CO.pulse] -10.331 -22.189  0.783   0.044
beta[pulse.CO]    0.855   0.754  0.983   0.012
beta[CO.pulse]    1.169   1.017  1.327   0.988

 Note that intercepts in conversion formulae are adjusted to get
 conversion fromulae that represent the same line both ways,
 - therefore are the median of the alphas above not identical
 to the intercepts given in the conversion formulae.

> MC1$summary

NULL
```

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:

```
> plot(MC1, pl.obs = TRUE)
```

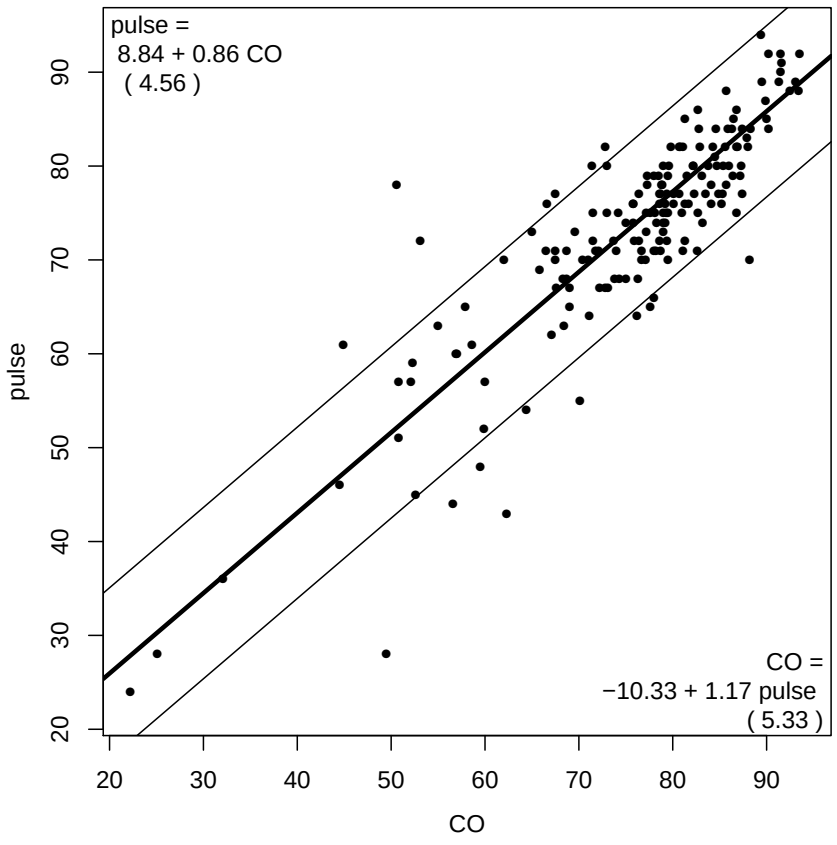

Figure 4.16: *Conversion between methods based on* `MCmcmc`*-output.*

4.4 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 )

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

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 )

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

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

```
> round( ftable( DA.reg( oxt ) ), 3 )
```

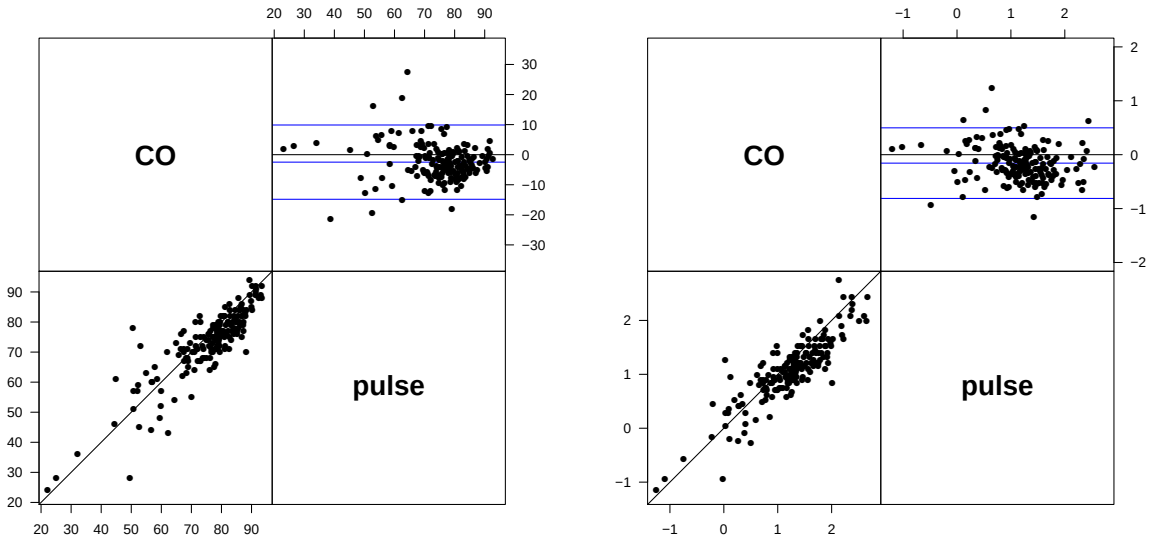

Figure 4.17: *Original (left) and* logit*-transformed oximetry data. Clearly, the logit-transform removes the tendency to diminishing variance at the upper end of the measurements, whereas the outliers in the middle of the scale have not been remedied..*

```
              alpha   beta sd.pred beta=1 s.d.=K
From: To:
CO    CO      0.000  1.000      NA     NA     NA
      pulse  -0.034  0.900   0.306  0.009  0.246
pulse CO      0.038  1.111   0.340  0.009  0.246
      pulse   0.000  1.000      NA     NA     NA
```

It appears that there is no clear evidence of variance inhomogeneity, but there is some indication 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:

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

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

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.

5. The natural thing would be to present LoA and the Bland-Altman plot on the original scale. Since we used the logit-transform; differences between logits are log-odds-ratios, so the y-axis should be shown as odds-ratios (i.e. percent saturation relative to percent non-saturation), and the x-axis either as percentage saturation or saturation-odds:

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> with( to.wide(oxt), plot( 100/(1+exp(-(CO+pulse)/2)), exp( CO-pulse ),
+                     log="y", pch=16,
+                     ylab="Odds-ratio: CO vs. pulse",
+                     xlab="Average oxygen saturation") )

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

This plot is however not very instructive, as the odds-ratio is not an immediately understandable quantity, to the extent that it is possible, reporting of results should be done on the original (clinically relevant) scale.

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.

The model we have is:

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

This leads to a prediction of one method from the other as:

$$y_{\text{CO}|\text{pulse}} = \alpha_{\text{CO}} - \alpha_{\text{pulse}} + y_{\text{pulse}} \pm 2\sqrt{\tau^2_{\text{CO}} + \tau^2_{\text{pulse}} + \sigma^2_{\text{CO}} + \sigma^2_{\text{pulse}}}$$

From the output from the `BA.est` function we see that $\alpha_{\text{CO}} - \alpha_{\text{pulse}} = 0.156$, and that the prediction s.d. is $\sqrt{2 \times 0.0246 + 0.0256 + 0.0320} = 0.327$. Hence the conversion line from logit(pulse) to logit(CO) is:

$$y_{\text{CO}|\text{pulse}} = 0.156 + y_{\text{pulse}} \pm 2 \times 0.327$$

So we generate points to plot the resulting conversion intervals on the original scale.

```
> logit <- function(p) log(p/(1-p))
> tigol <- function(x) 1/(1+exp(-x))
> pu.pt <- seq(0.001,0.999,0.001)
> logit.pu <- logit( pu.pt )
> logit.CO <- 0.156 + logit.pu
> logit.CO <- cbind( logit.CO, logit.CO+2*0.327, logit.CO-2*0.327 )
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( NA, NA,  xlab="pulse", ylab="CO", xlim=c(20,100), ylim=c(20,100),
+               xaxs="i", yaxs="i" )
> abline( h=seq(0,100,5), v=seq(0,100,5), col=gray(0.95) )
> abline( h=seq(0,100,10), v=seq(0,100,10), col=gray(0.85) )
> matlines( 100*pu.pt, 100*tigol(logit.CO),
+           type="l", lwd=c(3,1,1), lty=1, col="black" )
> # Add in the results from the analysis on the original scale
> LoA <- BA.est(ox)$LoA #$
> matlines( 100*pu.pt, cbind( 100*pu.pt-LoA[1],
+                             100*pu.pt-LoA[2],
+                             100*pu.pt-LoA[3]),
+           type="l", lwd=c(3,1,1), lty=1, col=gray(0.7) )
> # Redraw the prediction limits
> matlines( 100*pu.pt, 100*tigol(logit.CO),
+           type="l", lwd=c(3,1,1), lty=1, col="black" )
> # And finally add the points:
> with( to.wide(ox), points( pulse, CO, pch=16, cex=0.2 ) )
> box()
```

The resulting plot is shown in figure 4.20

As an alternative to the explicit calculations we might have used the output from `BA.est`:

```
> logit.CO <- outer( logit.pu, LoAt[1:3], "-" )
> matlines( pu.pt, 100*tigol(logit.CO) )
```

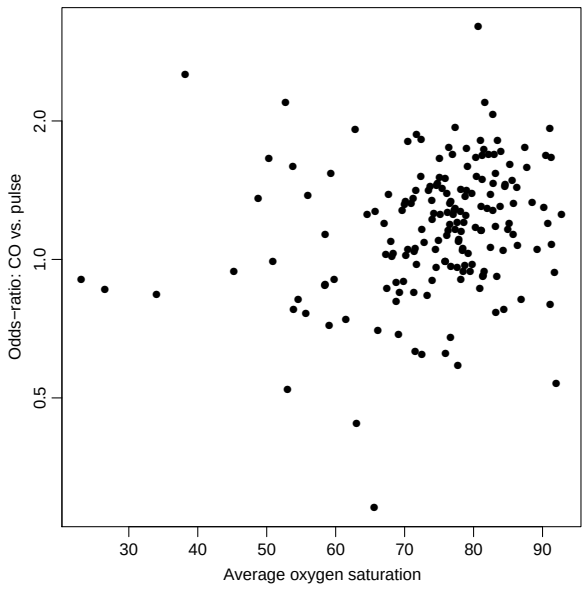

Figure 4.18: *Modified Bland-Altman plot for the logit-transformed data. A difference on the logit-scale corresponds to a log-odds-ratio, so the differences are displayed as odds-ratios (i.e. ratios of oxygen saturation odds), and the averages are the averages of the logit-transformed values, back-transformed to the the percentage scale.*

Note the use of the `outer` function which creates a matrix with `length(logit.pu)` rows and `length(LoAt[1:3])` colums, where the (i, j)th element is the difference between `logit.pu[i]` and `LoAt[1:3][j]`.

7. It would be interesting to see if a log-transform would have worked equally well — note that the log-transform requires a well-defined origin to be well-defined, and that we have. But on the other hand we do have a strict upper limit of 100% which the measurements cannot exceed, and this is not recognized by the mathematical form of the function.

```
> # First the empty coordinate system
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( NA, NA,  xlab="pulse", ylab="CO", xlim=c(20,100), ylim=c(20,100),
+                xaxs="i", yaxs="i" )
> abline( h=seq(0,100,5), v=seq(0,100,5), col=gray(0.95) )
> abline( h=seq(0,100,10), v=seq(0,100,10), col=gray(0.85) )
> # Transform data
> oxl <- transform( ox, y = log(y) )
> # Values for method 1 where prediction are computed
> y1.pr <- seq(10,99.9,0.1)
> # Get the limits of agreement (method 2 minus method 1)
```

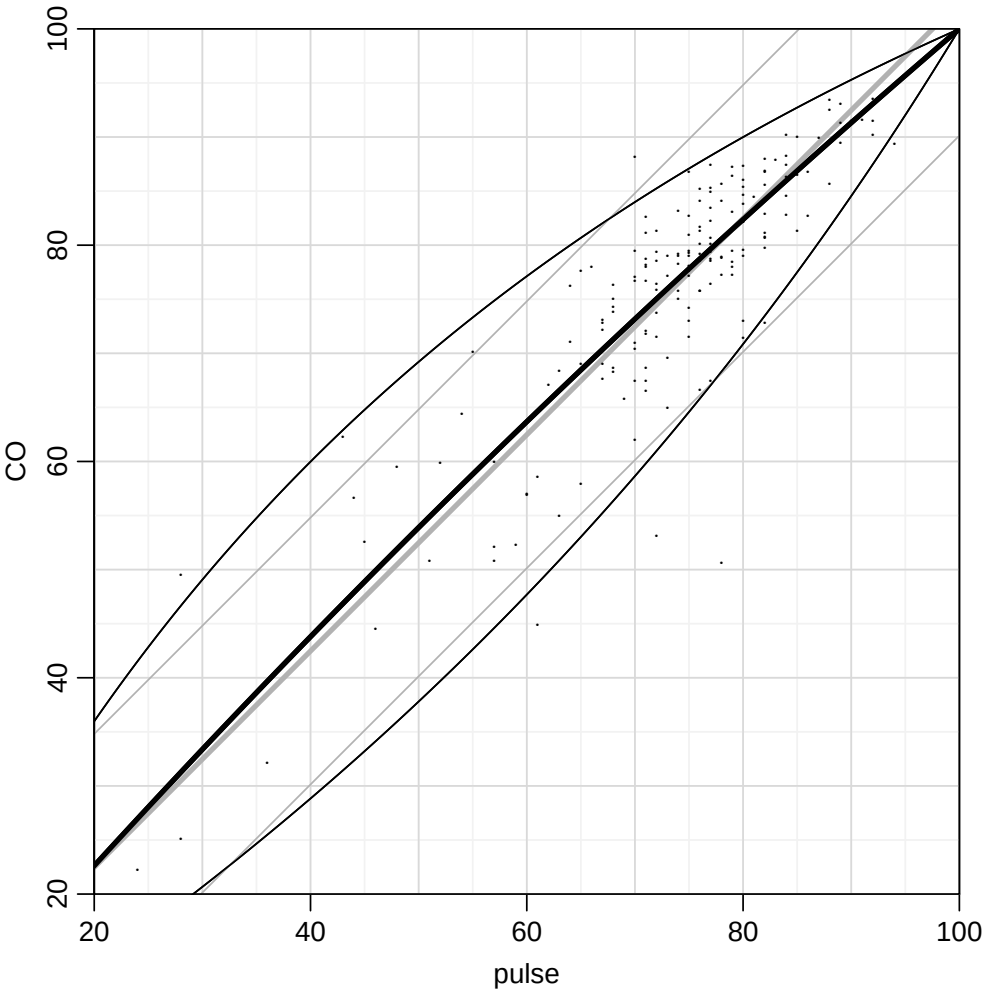

Figure 4.19: *Prediction between pulse and CO-oximetry assuming a constant difference on the* logit *scale. The limits using the original scale are shown too in light gray.*

```
> ( LoAl <- BA.est( oxl )$LoA )
> # Plot the points
> matlines( y1.pr, exp( outer( log(y1.pr), LoAl[1:3], "-" ) ),
+           lty=1, col="black", lwd=c(3,1,1) )
> # Add the points:
> with( to.wide(ox), points( pulse, CO, pch=16, cex=0.3 ) )
> box()
```

As can clearly be seen from figure 4.20, this transformation is unsatisfactory; it does not take the upper limits of the measurement into account.

8. The code used to produce the plot is easily modified to accommodate other transformations — we just replace the functions log and exp by the desired transformation and its inverse.

   Two other transformations of proportions that differ from the logit are the log−log transform and the complementary log−log transform:

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

   Applying these goes like this:

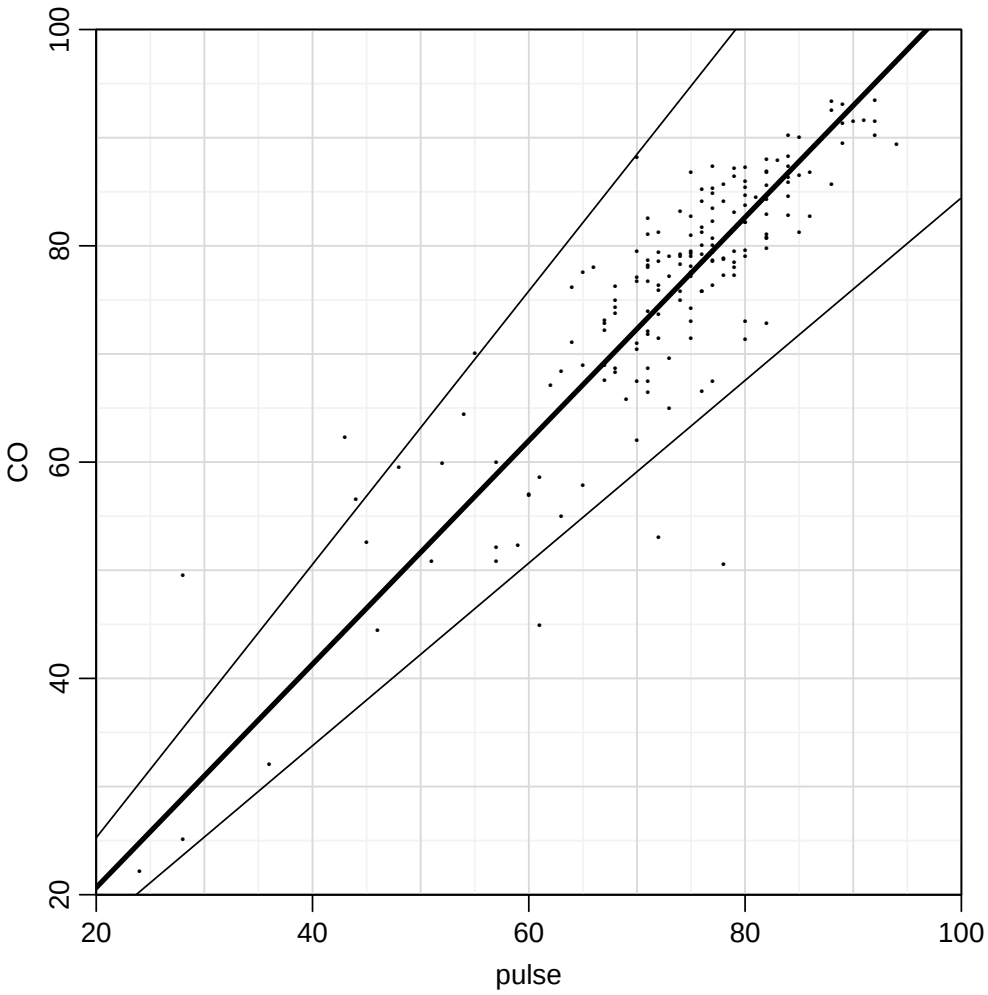

Figure 4.20: *Prediction between pulse and CO-oximetry assuming a constant difference on the* log *scale.*

```
> # First the empty coordinate system
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( NA, NA,  xlab="pulse", ylab="CO", xlim=c(0,100), ylim=c(0,100),
+               xaxs="i", yaxs="i" )
> abline( h=seq(0,100,5), v=seq(0,100,5), col=gray(0.95) )
> abline( h=seq(0,100,10), v=seq(0,100,10), col=gray(0.85) )
> # Define the functions needed to transform
>    ll <- function( p ) log(-log(p))
>  i.ll <- function( x ) exp(-exp(x))
>   cll <- function( p ) ll(1-p)
> i.cll <- function( x ) 1-i.ll(x)
> # Transform data
> oxll <- transform( ox, y =  ll(y/100) )
> oxcl <- transform( ox, y = cll(y/100) )
> # Values for method 1 where prediction are computed
> y1.pr <- seq(0.1,99.9,0.1)
> # Get the limits of agreement (method 2 minus method 1)
> ( LoAll <- BA.est( oxll )$LoA )
> ( LoAcl <- BA.est( oxcl )$LoA )
> # Plot the two sets of prediction limits between methods
> matlines( y1.pr, i.ll( cbind( ll(y1.pr/100) - LoAll[1],
+                               ll(y1.pr/100) - LoAll[2],
+                               ll(y1.pr/100) - LoAll[3] ) )*100,
+           lty=1, col="blue", lwd=c(3,1,1) )
> matlines( y1.pr, i.cll( cbind( cll(y1.pr/100) - LoAcl[1],
+                                cll(y1.pr/100) - LoAcl[2],
+                                cll(y1.pr/100) - LoAcl[3] ) )*100,
+           lty=1, col="red", lwd=c(3,1,1) )
> matlines( y1.pr, tigol( cbind( logit(y1.pr/100) - LoAt[1],
+                                logit(y1.pr/100) - LoAt[2],
+                                logit(y1.pr/100) - LoAt[3] ) )*100,
+           lty=1, col=gray(0.6), lwd=c(3,1,1) )
> text( cnr(95, 5), labels= "log-log", col="blue", adj=c(1,0), font=2 )
> text( cnr( 5,95), labels="clog-log", col="red" , adj=c(0,1), font=2 )
> # Add the points:
> with( to.wide(ox), points( pulse, CO, pch=16, cex=0.3 ) )
> box()
```

Clearly, there is no way to decide which one of these transformations is the better for the given dataset — the logit and the log−log transform are very similar close to 100% and the logit and the clog−log are similar close to 0. Incidentally all three sets of limits captures exactly the same number of points as the naïve limits using the original scale.

Intuitively one would choose the logit-transform, but that is merely based on the fact that this is what we are used to.

9. 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 the `MethComp` function. The only thing needed is to transform the measurment variable to the logit scale

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

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

   ```
   > ox.logit <- MCmcmc( oxt, bias="lin", random=c("mi","ir"), n.iter=500 )

   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 500 iterations
  (of which 250 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 1000 observations.

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4:

> print( ox.logit )

Conversion formula:
```

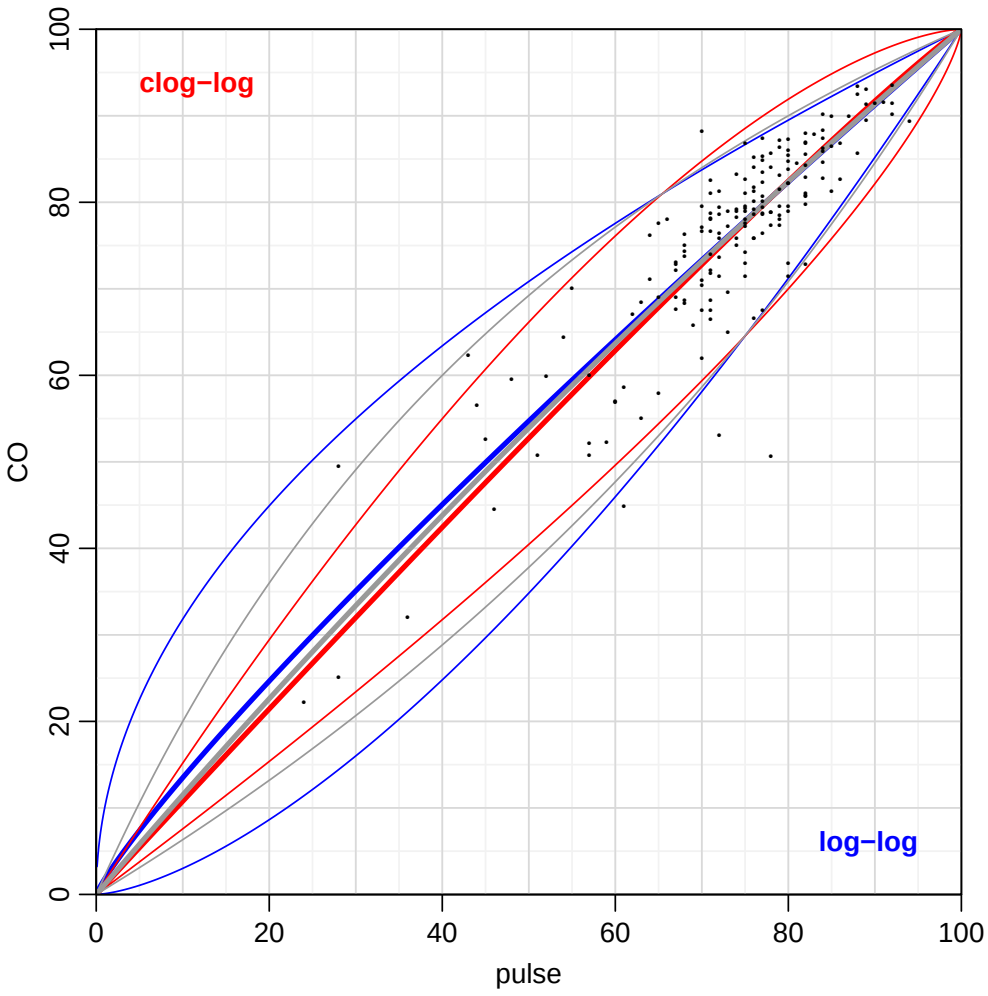

Figure 4.21: *Prediction between pulse and CO-oximetry using* log−log *and* clog−log *transformations. The results from using the* logit *transform is also given (light gray).*

```
 y_to = alpha + beta * y_from +/- 2*sd.pred:

      From:     CO                  pulse
             alpha   beta sd.pred  alpha   beta sd.pred
To:
CO           0.000  1.000   0.197  0.030  1.117   0.262
pulse       -0.027  0.895   0.234  0.000  1.000   0.279

Variance components with 95 % cred.int.:
                   50%  2.5% 97.5%
sigma.ir[CO]     0.249 0.208 0.295
sigma.ir[pulse]  0.223 0.187 0.264
sigma.mi[CO]     0.168 0.127 0.219
sigma.mi[pulse]  0.150 0.113 0.190
sigma.res[CO]    0.139 0.078 0.195
sigma.res[pulse] 0.197 0.151 0.238
sigma.tot[CO]    0.334 0.294 0.376
sigma.tot[pulse] 0.334 0.300 0.370

Mean parameters with 95 % cred.int.:
                   50%   2.5% 97.5% P(>0/1)
alpha[pulse.CO] -0.027 -0.161 0.138   0.368
alpha[CO.pulse]  0.030 -0.179 0.164   0.632
beta[pulse.CO]   0.895  0.781 1.002   0.029
beta[CO.pulse]   1.117  0.998 1.280   0.971

 Note that intercepts in conversion formulae are adjusted to get
 conversion fromulae that represent the same line both ways,
 - therefore are the median of the alphas above not identical
 to the intercepts given in the conversion formulae.
```

Bibliography

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

Chapter 5

MethComp manual (version 0.5.2)

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

Description

If a method comparison model is defined as $y_{mi} = \alpha_m + \beta_m \mu_i$, $m = 1, 2$ the coefficients of the linear conversion form method 1 to 2 are computed as well as the point where the linear conversion function intersects the identity line. 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, this is assumed to be a numerical vector of length 4 pointing to the columns of `a1` with the intercepts and slopes.
`a2`	Numerical vector of intercepts for second method.
`b2`	Numerical vector of slopes for second method.
`col.names`	Names for the resulting three vectors.

Value

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

Author(s)

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

References

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

See Also

`BA.plot`, `MCmcmc`

Examples

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

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

Description

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

Usage

```
AltReg( data,
      linked = FALSE,
         IxR = linked,
         MxI = TRUE,
      varMxI = FALSE,
         eps = 0.001,
     maxiter = 50,
     int.loc = 0,
       trace = FALSE,
      sd.lim = 0.01 )
```

Arguments

`data`	Data frame with the data in the usual `Meth` format, 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.
`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.
`int.loc`	Scalar. The location where the intercept is evaluated when returning the linear conversion paramaters between methods.
`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.

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

A matrix with one row per method compared. There are columns for intercept and slope for each of the methods, as well as columns for each of the three variance components.

Suppose methods are labelled `m1`, `m2` and `m3`. Prediction of a measurement `y1` by method `m1` from an observation `y2` by method `m2` is obtained as `y1= A + B y2` where `A` and `B` are from the row labelled `m1`, columns labelled `a m1` and labelled `b m1`, respectivlely.

Author(s)

Bendix Carstensen, ⟨bxc@steno.dk⟩

References

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

See Also

`BA.est` `Meth.sim`

Examples

```
dfr <- Meth.sim( Ni = 30,
                 Nm = 3,
               beta = c(0.9,0.8,1.1),
          sigma.mi  = c(4,5,8),
          sigma.ir  = 3,
          sigma.mir = c(5,4,3),
             m.thin = 1,
             i.thin = 1 )
levels(dfr$meth) <- paste( "m",1:3,sep="" )
str(dfr)
summary(dfr)
plot(dfr,var.names=TRUE)
# AltReg( dfr, linked=TRUE, trace=TRUE )
# AltReg( dfr, linked=TRUE, varMxI=TRUE, trace=TRUE )
data( sbp )
# AltReg( dfr, linked=TRUE, varMxI=TRUE, trace=TRUE )
```

`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=linked,
                           MxI=TRUE,
                       varMxI=TRUE, bias=TRUE, alpha=0.05 )
VC.est( data,
         IxR = has.repl(data), linked = IxR,
         MxI = has.repl(data), matrix = MxI,
      varMxI = TRUE, bias = TRUE, print = FALSE )
```

Arguments

data	A data frame representing method comparison data with replicate measurements, i.e. with columns meth, item, repl and y.
linked	Logical. Are the replicated linked within item across methods?
IxR	Logical. Should in 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.
MxI	Logical. Should the method by item interaction (matrix effect) be included in the model.
matrix	Logical. Alias for MxI.
varMxI	Logical. Should the method by item interaction have a variance that varies between methods. Ignored if only two methods are compared.
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.
print	Logical. Should the estimated bias and variance components be printed?

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, and returns the estimates. VC.est is also used as part of the fitting algorithm in AltReg, where each iteration step requires fit of this model.

Value

A list with four elements; BA.est returns a list with elements Bias, VarComp, LoA, RepCoef; VC.est returns (invisibly!) a list with elements Bias, VarComp, Mu, RanEff. These list componenets are:

Bias	Vector of estimates of α_m, the first element is always 0.
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 inder identical circumstances; the repeatability coefficient the numerical extent of the prediction interval for this difference.
Mu	Estimates of the item-specific parameters.
RanEff	Estimates of the randome effects form thr 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.

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 )
BA.est( ox )
BA.est( ox, linked=FALSE )
data( sbp )
BA.est( sbp )
BA.est( sbp, linked=FALSE )
# Check what you get from VC.est
str( VC.est( sbp ) )
```

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

Description

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

Usage

```
BlandAltman(x, y,
          x.name = NULL,
          y.name = NULL,
         maintit = "",
             cex = 1,
             pch = 16,
      col.points = "black",
       col.lines = "blue",
            limx = NULL,
            limy = NULL,
            ymax = NULL,
            eqax = FALSE,
            xlab = NULL,
            ylab = NULL,
           print = TRUE,
        reg.line = FALSE,
          digits = 2,
            mult = FALSE,
           alpha,
             ... )

BA.plot( y1, y2,
     meth.names = NULL,
      mean.repl = FALSE,
    comp.levels = 2:1,
            ... )
```

Arguments

x	Numerical vector of measurements by 1st method.
y	Numerical vector of measurements by 2nd method. Must of same length as x.
x.name	Label for the 1st method (x).
y.name	Label for the 2nd method (y).

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

Value

A list with 2 elements:

lim.agree	A vector of length 3 with Limits of Agreement.
p.value	P-value for the hypothesis that the mean difference is 0. Usually a lame thing to use.

Author(s)

Bendix Carstensen ⟨bxc@steno.dk⟩, http://www.biostat.ku.dk/~bxc.

References

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

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

B Carstensen. Limits of agreement: How to use the regression of differences on averages. Preprint, Department of Biostatistics, University of Copenhagen, http://cms.ku.dk/sund-sites/ifsv-sites/english/about/departments/biostatistics/reports/2008/researchreport08-06.pdf

See Also

BA.plot, MCmcmc.

Examples

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

# (almost) Right to use replicates singly
par( mfrow=c(1,1) )
oxw <- to.wide( ox )
CO <- oxw[,"CO"]
pulse <- oxw[,"pulse"]
BlandAltman( CO, pulse, mult=TRUE )
BlandAltman( CO, pulse, eqax=TRUE )

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

`bothlines` *Add regression lines to a plot*

Description

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

Usage

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

Arguments

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

Value

None.

Author(s)

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

See Also

abline.

Examples

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

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

Description

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

Usage

```
data(cardiac)
```

Format

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

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

`item` a numeric vector giving the item number.

`repl` a numeric vector with replicate number.

`y` the measuremnts of cardiac output.

Details

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

Source

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

Examples

```
data(cardiac)
cardiac <- Meth(cardiac)
summary(cardiac)
# Visually check exchangeability
plot( cardiac )
plot( perm.repl( cardiac ) )
BA.est(cardiac)
# Run MCmcmc using BRugs for an insufficient amount of iterations
card.mi.ir <- MCmcmc( cardiac, beta=FALSE, random=c("mi","ir"), n.iter=100, trace=T )
print( card.mi.ir )
```

`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, subset,
                               col = NULL,
                               pch = 16,
                               cex = 0.2, ... )
```

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 plost for the intercept parameters.
`scales`	Character vector of length two, indicating whether x- and y-axes of the plots should be constrained to be the same across panels.
`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 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 indicationg 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`, `post` calls `densityplot`, `pairs` calls `pairs`.

Details

A Lattice plot is returned, which means that it must printed 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.

See Also

MCmcmc,plot.MCmcmc

Examples

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

---

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

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

References

Shortly...

See Also

MCmcmc.

Examples

```
cbind( zz <- 1:15, middle(zz), ends(zz) )
data( sbp )
bp <- subset( sbp, repl==1 & meth!="J" )
bp <- Meth( bp )
summary( bp )
plot( bp )
bw <- to.wide( bp )
with( bw, corr.measures( R, S ) )
# See how it gets better with less and less data:
summ.corr <-
rbind(
with( subset( bw, middle( R+S, 0.6 ) ), corr.measures( R, S ) ),
with( subset( bw, middle( R+S, 0.4 ) ), corr.measures( R, S ) ),
with(         bw                      , corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.3 ) ), corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.4 ) ), corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.6 ) ), corr.measures( R, S ) ),
with( subset( bw,   ends( R+S, 0.8 ) ), corr.measures( R, S ) ) )
rownames( summ.corr ) <- c("middle 40%",
                           "middle 60%",
                           "total",
                           "outer 70%",
                           "outer 60%",
                           "outer 40%",
                           "outer 20%")
summ.corr
```

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

Description

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

Usage

```
DA.reg(data)
```

Arguments

data	A Meth object. May also be a data frame with columns meth, item and y.

Details

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

Value

A three-dimensional array, with dimensions From, To (both with levels equal to the methods in data) and an unnamed dimension with levels "alpha", "beta", "sd.pred", "beta=1" and "s.d.=K". Converting from method j to method k using

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

with prediction standard deviation σ, just requires the entries [l,k,c("alpha","beta","sd.pred"]. The two last entries ae p-values for the hypothese the $\beta = 1$ and that the standard error is constant over the range.

Author(s)

Bendix Carstensen, Steno Daibetes Center, bxc$steno.dk

References

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

Examples

```
data( milk )
DA.reg( milk )
round( ftable( DA.reg( milk ) ), 3 )
data( sbp )
round( ftable( DA.reg( sbp ) ), 3 )
```

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

Description

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

Usage

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

Arguments

x	numerical variable.
y	numerical variable.
vr	The assumed known ratio of the (residual) variance of the ys relative to that of the xs. 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, http://www.biostat.ku.dk/~bxc.

References

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

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

See Also

MCmcmc

Examples

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

```
# Comparing classical regression and "Deming extreme"
summary(lm(y~x))
Deming(x,y,vr=1000000)
```

`Enzyme` *Enzyme activity data*

Description

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

Usage

```
data(Enzyme)
```

Format

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

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

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

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

Source

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

Examples

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

`fat` *Measurements of subcutaneous and visceral fat*

Description

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

Usage

```
data(fat)
```

Format

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

`Id` Person id.

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

`Rep` Replicate — exchangeable within person and observer.

`Sub` Subcutaneous fat measured in mm.

`Vic` Visceral fat measured in mm.

Examples

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

`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)
```

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

Description

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

Usage

```
  MCmcmc( data,
          bias = "linear",
           IxR = has.repl(data), linked = IxR,
           MxI = TRUE,           matrix = MxI,
        varMxI = TRUE,
      n.chains = 4,
        n.iter = 2000,
      n.burnin = n.iter/2,
        n.thin = ceiling((n.iter-n.burnin)/1000),
bugs.directory = getOption("bugs.directory"),
         debug = FALSE,
bugs.code.file = "model.txt",
       clearWD = TRUE,
        bugsWD = "bugsWD",
     code.only = FALSE,
      ini.mult = 2,
      list.ini = TRUE,
           org = FALSE,
       program = "BRugs",
           ... )
## S3 method for class 'MCmcmc':
summary( object, alpha=0.05, ...)
## S3 method for class 'MCmcmc':
print( x, across, digits=3, alpha=0.05, ... )
## S3 method for class 'MCmcmc':
subset( x, subset=NULL, allow.repl=FALSE, chains=NULL, ... )
## S3 method for class 'MCmcmc':
mcmc( x, ... )
```

Arguments

data	Data frame with variables meth, item, repl and y, possibly a Meth object. y represents a measurement on an item (typically patient or sample) by method meth, in replicate repl.
bias	Character. Indicating how the bias between metods should be modelled. Possible values are "none", "constant", "linear" and "proportional". Only the first three letters are significant. Case insensitive.
IxR	Logical. Are the replicates linked across methods, i.e. should a random item by repl be included in the model.
linked	Logical, alias for IxR.
MxI	Logical, should a meth by item effect be included in the model?
matrix	Logical, alias for MxI.
varMxI	Logical, should the method by item effect have method-specific variances. Ignored if only two methods are compared.
n.chains	How many chains should be run by WinBUGS — passed on to bugs.
n.iter	How many total iterations — passed on to bugs.
n.burnin	How many of these should be burn-in — passed on to bugs.
n.thin	How many should be sampled — passed on to bugs.
bugs.directory	Where is WinBUGS (>=1.4) installed — passed on to bugs. The default is to use a parameter from options(). If you use this routinely, this is most conveniently set in your .Rprofile file.
debug	Should WinBUGS remain open after running — passed on to bugs.
clearWD	Should the working directory be cleared for junk files after the running of WinBUGS — passed on to bugs.
bugsWD	Name of the folder where the bugs files are put. The code file is also put in this folder.
bugs.code.file	Where should the bugs code go?
code.only	Should MCmcmc just create a bugs code file and a set of inits? See the list.ini argument.
ini.mult	Numeric. What factor should be used to randomly perturb the initial values for the variance componets, see below in details.
list.ini	List of lists of starting values for the chains, or logical inidcating whether starting values should be generated. If TRUE (the default), the function VC.est will be used to generate initial values for the chains. list.ini is a list of length n.chains. Each element of which is a list with the following vectors as elements: mu - length I alpha - length M beta - length M sigma.mi - length M - if M is 2 then length 1 sigma.ir - length 1 sigma.mi - length M sigma.res -length M If code.only==TRUE, list.ini indicates whether a list of initial values is returned (invisibly) or not. If code.only==FALSE, list.ini==FALSE is ignored.
org	Logical. Should the posterior of the original model parameters be returned too? If TRUE, the MCmcmc object will have an attribute, original, with the posterior of the parameters in the model actually simulated.
program	Which program should be used for the MCMC simulation. Possible values are "brugs","openbugs","ob" (openBUGS), "winbugs","wb" (WinBUGS).
...	Additional arguments passed on to bugs.
object	A MCmcmc object
alpha	1 minus the the confidence level
x	A MCmcmc object

across	Should the summary of conversion formulae be printed with α, β and prediction sd. across or down?
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 sd's, that is). Furthermore, the posterior of the point where the conversion lines intersects the identity as well as the prediction sd's between any pairs of methods are included.

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

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

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

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

and hence no guarantee that the parameters derived as posterior medians produce conversion lines that are the same in both directions. Therefore, summary.MCmcmc computes the estimate for $\alpha_{2\cdot 1}$ alpha.2.1 as

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

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

Value

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

Furthermore, the object have the following attibutes:

random	Character vector indicatinf which random effects ("ir","mi") were included in the model.
methods	Character vector with the method names.
data	The dataframe used in the analysis. This is used in plot.MCmcmc when plotting points.
mcmc.par	A list giving the number of chains etc. used to generate the object.
original	If org=TRUE, an mcmc.list object with the posterior of the original model parameters, i.e. the variance components and the unidentifiable mean parameters.

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

Author(s)

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

References

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

See Also

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

Examples

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

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

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

Description

A dataframe is simulated that represents data from a method comparison study based on parameters specified by the user. It is returned as a Meth object.

Usage

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

Arguments

Ni	The number of items (patient, animal, sample, unit etc.)
Nm	The number of methods of measurement.
Nr	The (maximal) number of replicate measurements for each (item,method) pair.
nr	The minimal number of replicate measurements for each (item,method) pair. If nr<Nr, the number of replicates for each (meth,item) pair is uniformly distributed on the points nr:Nr, otherwise nr is ignored. Different number of replicates is only meaningful if replicates are not linked, hence nr is also ignored when sigma.ir>0.
alpha	A vector of method-specific intercepts for the linear equation relating the "true" underlying item mean measurement to the mean measurement on each method.

`beta`	A vector of method-specific slopes for the linear equation relating the "true" underlying item mean measurement to the mean measurement on each method.
`mu.range`	The range across items of the "true" mean measurement. Item means are uniformly spaced across the range. If a vector length `Ni` is given, the values of that vector will be used as "true" means.
`sigma.mi`	A vector of method-specific standard deviations for a method by item random effect. Some or all components can be zero.
`sigma.ir`	Method-specific standard deviations for the item by replicate random effect.
`sigma.mir`	A vector of method-specific residual standard deviations for a method by item by replicate random effect (residual variation). All components must be greater than zero.
`m.thin`	Fraction of the observations from each method to keep.
`i.thin`	Fraction of the observations from each item to keep. If both `m.thin` and `i.thin` are given the thinning is by their componentwise product.

Details

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

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

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

Value

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

Author(s)

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

See Also

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

Examples

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

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

Description

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

Usage

```
Meth( meth,
             item,
             repl,
                y,
              ...,
             print = FALSE)
## S3 method for class 'Meth':
summary( object, ... )
## S3 method for class 'Meth':
plot( x, y = NULL,
                   col.LA = "blue",
                cex.name = 2,
               var.range,
              diff.range,
               var.names = FALSE,
                     ... )
## S3 method for class 'Meth':
subset(x, ... )
## S3 method for class 'Meth':
sample(x, size, ... )
## S3 method for class 'Meth':
transform(`_data`, ... )
```

Arguments

meth	Vector of methods, numeric, character of factor. May also be a dataframe. If this has columns, meth, item, (repl) and y, these are used.
item	Vector of items. If meth is a dataframe, item is taken as the columns of the meth dataframe to use as vectors of meth, item, (repl) and y.
repl	Vector of replicate numbers.
y	Vector of measurements. For the plot method the argument is either a vector indices or names of methods to plot.
print	Logical: Should a summary result be printed?
object	A Meth object.
x	A Meth object.
col.LA	What color should be used for the limits of agreement.
cex.name	Character expansion factor for plotting method names
var.range	The range of the 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.
size	The number or fraction (if size<1) of items to sample.
_data	A Meth object.
...	Ignored by the Meth and the summary functions. In the plot function, parameters passed on the panel function plotting methods against each other, as well as 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: tab - tabulation, plot - plotting and a couple of analysis methods (not fixed yet).

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 `sample` returns a subset of the `Meth` object with complete interformtion a sample of the items.

Author(s)

Bendix Carstensen, ⟨bxc@steno.dk⟩

Examples

```
data(fat)
# Different ways of selecting columns and generating replicate numbers
Sub1 <- Meth(fat,c(2,1,3,4),print=TRUE)
Sub2 <- Meth(fat,c(2,1,NA,4),print=TRUE)
Sub3 <- Meth(fat,c(2,1,4),print=TRUE)
summary( Sub3 )
plot( Sub3 )
# 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( dev, item, d.ana-d.samp, y, print=TRUE ) )
hb2  <- with( subset(hba1c,type=="Cap"), Meth( dev, item, d.ana-d.samp, y, print=TRUE ) )
summary( hb1 )
summary( hb2 )
```

`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)
plot(milk)
plot( y[meth=="Trig"]~y[meth=="Gerber"],data=milk,
      xlab="Fat (g/100 ml; Gerber)",
      ylab="Fat (g/100 ml; Trig.)")
abline(0,1)
```

---

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

---

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)
with( ox, table(table(item)) )
par( mfrow=c(1,2), mar=c(4,4,1,4) )
BA.plot( ox, ymax=20 )
BA.plot( ox, ymax=20, mean.repl=TRUE )
```

`PEFR` *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. (PEFR=Peak Expiratory Flow Rate).

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. This is numbers between 165 and 656 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 data frame with columns meth, item and y, possibly a Meth object.

Details

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

Value

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

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

Author(s)

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

See Also

perm.repl

Examples

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

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

Description

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

Usage

```
plot.MCmcmc( x,
             axlim = range( attr(x,"data")$y, na.rm=TRUE ),
             which,
          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),
           pl.obs = 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
`which`	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?
`pl.obs`	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 )
data( cardiac )
MCcard <- MCmcmc( cardiac, beta=FALSE, random=c("mi","ir"), n.iter=500 )
print( MCcard )
plot( MCcard )
plot( MCcard, pl.obs=TRUE )
```

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

Description

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

Usage

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

Arguments

`x`	A `MCmcmc` object.
`which`	For which of the compared methods should the plot be made?
`lwd.line`	Line width for drawing the density.
`col.line`	Color for drawing the densities.
`lty.line`	Line type for drawing the densities.
`grid`	Logical. Should a vertical grid be set up? If numeric it is set up at the values specified. If `same.ax`, the range of the grid is taken to be the extent of the x-axis for all plots.
`col.grid`	The color of the grid.

rug	Should a small rug at the bottom show posterior quantiles?
probs	Numeric vector with numbers in the range from 0 to 100, indicating the posterior percentiles to be shown in the rug.
tot.var	Should the posterior of the total variance also be shown?
same.ax	Should the same axes be used for all methods?
meth.names	Should the names of the methods be put on the plots?
VC.names	Should the names of the variance components be put on the first plot ("first"), the last ("last"), all ("all") or none ("none"). Only the first letter is needed.
...	Parameters passed on the density furnction that does the smoothing of the posterior samples.

Details

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

Value

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

Author(s)

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

See Also

plot.MCmcmc, MCmcmc, check.MCmcmc

Examples

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

---

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

Description

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

Usage

```
data(plvol)
```

Format

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

meth a factor with levels Hurley 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)
```

`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.

Examples

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

scint *Relative renal function by Scintigraphy*

Description

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

Usage

```
data(scint)
```

Format

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

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

`item` Patient identification.

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

`age` Age of the patient.

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

Source

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

Examples

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

TDI *Compute Lin's Total deviation index*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

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

`TDI` The approximate value of the TDI

`Limits of Agreement`
Limits of agreement

Note

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

Author(s)

Bendix Carstensen, bxc@steno.dk

References

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

See Also

`BA.plot`,`corr.measures`

Examples

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

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

Description

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

Usage

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

Arguments

`data` A dataframe

`warn` Logical. Should a warning be printed when replicates are taken as items?

`vars` The variables representing measurements by different methods. Either a character vector of names, or a numerical vector with the number of the variables in the dataframe.

Details

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

Value

A dataframe.

Author(s)

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

See Also

perm.repl

Examples

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

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

Description

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

Usage

```
data(VitCap)
```

Format

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

`meth` a factor with levels `StNew StSkil ExpNew 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)
summary( Vcap <- Meth( VitCap ) )
plot( Vcap )
```

Index

abconv, 55
abline, *61*
AltReg, 56, *58*

BA.est, *56*, 57, *57*
BA.plot, *55*, *59*, *61*, *74*, *86*
BA.plot *(BlandAltman)*, 59
BlandAltman, 59, *60*
bothlines, 61
bugs, *71*, *72*

cardiac, 62
check.MCmcmc, 63, *74*, *83*
corr.measures, 64, *86*

DA.reg, 65
Deming, 66
density, *83*
densityplot, *63*

ends *(corr.measures)*, 64
Enzyme, 68

fat, 68

glucose, 69

has.repl *(perm.repl)*, 80
hba.MC, 70
hba1c, 70, *70*

Lattice, *63*, *64*

make.repl *(perm.repl)*, 80
mcmc.list, *73*
mcmc.MCmcmc *(MCmcmc)*, 71
MCmcmc, *55*, *61*, *64*, *65*, *67*, *70*, 71, *75*, *78*, *81–83*
Meth, *66*, *72*, *74*, 75, *75*, *80*
Meth.sim, *57*, 74
middle *(corr.measures)*, 64
milk, 77

ox, 78, *78*
ox.MC, 78

pairs, *63*
pairs.MCmcmc *(check.MCmcmc)*, 63
PEFR, 79
perm.repl, *59*, 80, *80*, *87*
plot.MCmcmc, *64*, *73*, *74*, 81, *83*
plot.Meth, *75*
plot.Meth *(Meth)*, 75
plot.VarComp, 82
plvol, 83
post.MCmcmc *(check.MCmcmc)*, 63
print.MCmcmc, *74*, *81*
print.MCmcmc *(MCmcmc)*, 71

reshape, *86*

sample.Meth *(Meth)*, 75
sbp, 84
scint, 85
subset.MCmcmc *(MCmcmc)*, 71
subset.Meth *(Meth)*, 75
summary.MCmcmc *(MCmcmc)*, 71
summary.Meth, *75*
summary.Meth *(Meth)*, 75

TDI, 85
to.long *(to.wide)*, 86
to.wide, 86
trace.MCmcmc *(check.MCmcmc)*, 63
transform.Meth *(Meth)*, 75

VC.est, *72*
VC.est *(BA.est)*, 57
VitCap, 87

xyplot, *63*