Statistical Analysis of Method Comparison Studies

University of Copenhagen
8–10 February 2011
Version 1

www.biostat.ku.dk/~bxc/MethComp/Courses/Cph.2011

Compiled Sunday 20th February, 2011, 22:25
from: `C:/Bendix/undervis/MethComp/Cph.2011/pracs/pracs.tex`

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

Contents

Program

0.1 Program

The program will be structured with one hour lecture and 2 hours practicals (approx.) every morning and afternoon and 1 hours lunch break; so the three days are structured as:

09:00 – 10:00	Lecture 1
10:00 – 10:30	Morning Tea
10:30 – 12:00	Practical 1
12:00 – 13:00	Lunch
13:00 – 14:00	Lecture 2
14:00 – 14:30	Coffee break
14:30 – 16:00	Practical 2

Tuesesday 8 February 2011	
09:15 – 10:30	**Lecture 1**: Welcome and introduction. Simple comparisons of measurement methods. Correlation. Introduction to computing.
10:30 – 11:00	**Morning Tea**
11:00 – 12:00	**Practical 1**: Limits of agreement, Bland-Altman-plots: • Milk • Plasma volume.
12:00 – 13:00	**Lunch**
13:00 – 14:00	**Lecture 2**: Setting up (your own) data. `Meth` objects. Non-constant difference between methods Designs with replicate measurements — allocation of sources of variation.
14:00 – 14:30	Coffee break
14:00 – 16:00	**Practical 2**: Data with replicate measurements by each method: • Fat. • Systolic blood pressure. • HbA_{1c}

Wednesday 9 February 2011	
09:00 – 09:15	Recap of Tuesday.
09:15 – 10:15	**Lecture 3**:
	A general model for method comparisons.
	Repeatability and reproducibility
10:15 – 10:45	**Morning Tea**
10:45 – 12:00	**Practical 3**:
	• Oximetry data.
12:00 – 13:00	**Lunch**
13:00 – 14:00	**Lecture 4:**
	Linear relationship between methods.
	Converting between methods.
	Variance components.
	TRansformations.
14:00 – 14:30	**Afternoon Tea**
14:30 – 16:00	**Practical 4**:
	• Oximetry data — transformation.
	• Analyzing your own data.

Thursday 10 February 2011	
09:00 – 09:15	Recap of Wednesday.
09:15 – 10:00	**Lecture 5**:
	Implementation in BUGS: Using the `MCmcmc` function.
10:00 – 10:30	**Morning Tea**
10:30 – 12:00	**Practical 5**:
	• SBP-data: Three methods with replicate measurements.
12:00 – 13:00	**Lunch**
13:00 – 14:00	**Lecture 6**:
	More elaborate designs, variance component models.
	Summary and overview.
14:00 – 14:30	**Afternoon Tea**
14:30 – 16:00	**Practical 6**:
	• Recap of practicals; tidying you code.
	• Analysing your own data.

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 contain extra functions. The more advanced models covered in this course are only implemented in R in the `MethComp` package.

In order to be able to write scripts (programs) in R and keep them for future use (and modification for other purposes) a good text editor with an interface to R is convenient. `TinN-R` is one possible answer. You can get it from `http://sourceforge.net/projects/tinn-r/files/Tinn-R%20setup/2.3.7.1/Tinn-R_2.3.7.1_setup.exe/download`[1].

R also has a built-in text editor which is a bit more primitive; it is accessed via $\boxed{\text{File}}\rightarrow\boxed{\text{Open script}}$ or $\boxed{\text{File}}\rightarrow\boxed{\text{New script}}$.

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-2.12.1-win.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 click in $\boxed{\text{Packages}}\rightarrow\boxed{\text{Install package(s)}}$, and choose the packages from the menu it brings up.

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

[1] `TinN` = `Tinn is not Notepad`)

1.1.2 The MethComp package

Finally you will have to install the (still) non-official package for R, MethComp[2], which contains all the functions for analysis of method comparison studies. It is available from 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_1.3.zip and then from the menu select $\boxed{\text{Packages}}\rightarrow$ $\boxed{\text{Install package(s) from local zip files}}$

The function MCmcmc from this package uses Markov chain simulation (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 functioning of the different versions of BUGSthat 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 first two 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 is slightly different.

BUGS is used from the MCmcmc function, but all the writing of programs and post-processing 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 license key which is free. To obtain one, register at the WinBUGS homepage and you will get an e-mail with the key 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:

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

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

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

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 requires 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 number of replicates for a `Meth` object.

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

2.2.3 Analysis functions

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

`DA.reg` Regresses the differences between methods on the averages and derives approximate linear conversion equations, based on [1].

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

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

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

2.2.4 Reporting functions

The functions `DA.reg`, `BA.est` or `AltReg` return objects of class `MethComp`, whereas `MCmcmc` return an object of class `MCmcmc`, which can be converted by the MethComp function. Thus you should do something like:

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

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

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

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

Chapter 3

Practicals

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, and introduce some ways that you can display data with the facilities in the `MethComp` package.

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).

First, 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)
```

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. (*Hint:* Take a look at the `reg.line` argument to the `BA.est` function).

3.2 Plasma volume: Single measurements by two methods

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

1. Plot the the measurements from the two methods against each other.

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

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

   Are these limits a reasonable summary of the data?

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

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

   Note that the explanation of the parameter `mult` is not on the help page for `BA.plot` but in that for `BlandAltman`.

   Does the log-transform give a better description of data?

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

3.3 Fat measurements: Exchangeable replicates

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

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, 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 — look at the help-page for this.

3.4 Systolic blood pressure: Linked replicates by two methods

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

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

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

   Since the colums have the right names you can easily turn the data-frame into a `Meth` object:

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

   What is the immediate impression of the relationship of the methods to each other?

   How are the replicate measurements handled by `plot.meth`?

2. We want to restrict our attention to the comparison of the two manual methods (J and R), but still using the replicate measurements.

   Are the replicates exchangeable within method and item?

   Make a Bland-Altman plot of the data for the two manual methods, and derive the limits of agreement, e.g.:

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

   Try to use the argument `ymax=` (the meaning of this is found on the help page for the function `BlandAltman`).

   How does the use of the replicates for this Bland-Altman plot and limits of agreement correspond to the exchangeability structure of data?

3. Fit the proper model for the data, reflecting the non-exchangeability of replicates:

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

   The code in `lme` to do this is:

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

   Find the bias between methods, as well as the variance components.

4. A more direct way of getting at the variance components is to use the wrapper `BA.est()`, try:

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

   Try to locate the values in teh output fron `BA.est` in the output from `lme`.

5. Use these estimates to construct limits of agreement for the difference J−R, and compare these with the limits obtained by using the paired replicates as items.

6. One way of demonstrating the lack of exchangeability of replicates is to make the overview plot using a random permutation of the replicates. If replicates were exchangeable within methods the plot would look similar when permuting the replicates. Try to use the function `perm.repl()` to make a random permutation of replicates — for the sake of completeness reload the dataset so you have all three methods available:

   ```
   > data(sbp)
   > sbp <- Meth(sbp)
   > plot(sbp)
   > plot(perm.repl(sbp))
   ```

   In order to compare results, you may want to open a new window between the two plotting commands using the command `windows()` or `x11()` (equivalent).

7. Compute limits of agreement based on the variance components from the model for the entire dataset.

8. Formulate this as a 95% prediction interval for a measurement by method R given a measurement by method J, y_{J}.

9. Fit the model on the dataset with only measurements by the two physicians and compute the limits of agreement based on estimates from this. Compare with the previously computed limits of agreement.

3.5 Measurement of HbA$_{1c}$ I: Machine and specimen as method

The `hba1c` data from the `MethComp` package contains measurements of HbA$_{1c}$, i.e. the fraction of the hemoglobin in the blood that is glycosylated, and is usually reported a s a percentage. Glycosylation of the hemoglobin depends on the glucose (sugar) concentration in the blood. The red blood cells that contain the hemoglobin have an average lifetime of 3 months, so HbA$_{1c}$ is therefore a marker of long term (i.e. 3 month) blood glucose regulation. It is used for monitoring of diabetes patients — normal person have a level of HbA$_{1c}$ about 4–5% whereas diabetes usually have higher values, the normal treatment target for HbA$_{1c}$ is a value below 6.6%.

At Steno Diabetes Center, HbA$_{1c}$ is monitored routinely for all patients, and the laboratory therefore has a machine to analyze blood samples for HbA$_{1c}$. At a certain point the machine (Biorad, version Classic `BR.VC`) were to be replaced, so two candidate machines were brought in and blood samples from a number of patients were measured on all three machines. Blood was sampled both as capillary blood and venous blood. Finally blood was stored an analyze on different days.

The primary aim of the study was to investigate which of the machines were the more accurate, secondary aims to see if there were substantial differences between measurements based on capillary and venous blood and finally to provide a conversion algorithm between "old" measurements and "new" measurements to avoid breaks in the clinical series for patients.

1. Load the `hba1c` data and take a look at the structure, e.g.:

```
> data( hba1c )
> with( hba1c, table( d.samp, d.ana ) )
> with( hba1c, table( dev, type, d.ana ) )
```

2. Note that the dataset does not have the standard structure, it lacks a definition of method and replicate. Provide these by using the interaction between `dev` and `type` and the day of analysis as replicate number.

   You may want to use the function `transform` and to create the interaction, the function (surprise, surpise) `interaction`, ie. create an updated dataframe, `hb`, say:

```
> hb <- transform( hba1c, meth = interaction( dev, type ),
+                          repl = d.ana )
```

3. Make an overview plot of the data in order to get an impression of the likely variations worth considering:

```
> plot.meth( hb )
```

   What is the major first impression of the precision and relative bias of the different instruments?

4. Can we consider the replicates exchangeable within methods?

5. Specify a "standard model" for analyzing these data and fit it using `MethComp`:

```
> m0 <- MethComp( hb )
```

   Remember to to put it into an object; the result is quite large, and therefore it is more handily represented by its default print method, so just type the name of the

```
> m0
```

6. There is a zillion arguments to `MethComp`, but for a start we just use the default settings — in "real" applications one would use a larger number of iterations in order to be on the safe side. Since there are 6 methods we can plot the variance components associated with each of them in a 2 by 3 layout, try:

```
> par(mfrow=c(2,3))
> plot.VarComp(m0)
```

7. The posterior distributions of the variance components may not be very well determined, so try to re-fit the model using substantially more iterations. Also, try to enclose the call to `MethComp` in a `system.time()` in order to see how much time it takes, e.g.:

```
> system.time(
+ m1 <- MethComp( hb, n.iter=1000, n.chains=5 )
+            )
```

8. After a longer simulation try to do a more detailed plot by fiddling the graphics parameters a bit:

```
> par(mfrow=c(2,3),mar=c(3,1,2,1),mgp=c(3,1,0)/1.6)
> plot.VarComp(m1,grid=seq(0,1.5,0.1))
```

9. Try to form conclusions about the machines and speciemns based on the posterior distributions of the variance components.

3.6 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?

3.7 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:

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

   What is the conclusion?

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

```
> ( 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:

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

   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.

3.8 Systolic blood pressure: Linked replicates by three methods

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

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

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

   Since the colums have the right names you can easily turn the data-frame into a `Meth` object:

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

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

2. Assess more precisely the relationship between the methods' uncertainty, first by using `BA.est`. How would you interpret the values of the estimated variance components?

3. Make a rough assessment of whether the pairwise differences between methods are constant and wheter the variances are constant too. You can use `DA.reg` for this purpose.

   Is there any new conclusions compared to the output from `BA.est`?

4. Now fit a proper model, both with linear relationships between methods and proper allocation of the variance components, using the `MCmcmc` function. Strat with 200 iterations to see if it works on your computer, then try with 1000 and then 10,000, in order to geta feeeling for how long time it takes on your comouter:

   ```
   > MCsbp <- MCmcmc( sbp, n.iter=100 )
   ```

5. Once you have the result from the MCMC sampling, you can inspect the results by using the function `MethComp`. Compare with the results from `BA.est` and `DA.reg`.

6. Check the convergence of the chains by using `trace` on the object:

   ```
   > trace.MCmcmc( MCsbp )
   ```

   You may want to discard some of the sampling path; you should look at the documentation for `mcmc.list` objects from the `coda` package.

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)

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

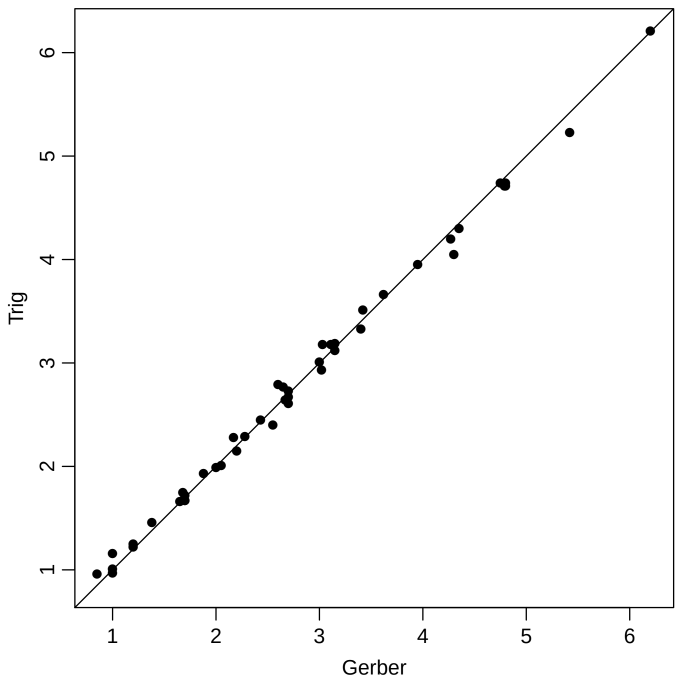


Figure 4.1: *Scatter plot of the* `milk` *data.*

```
> str(milk)

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

> summary(milk)

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

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

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

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

```
> 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:

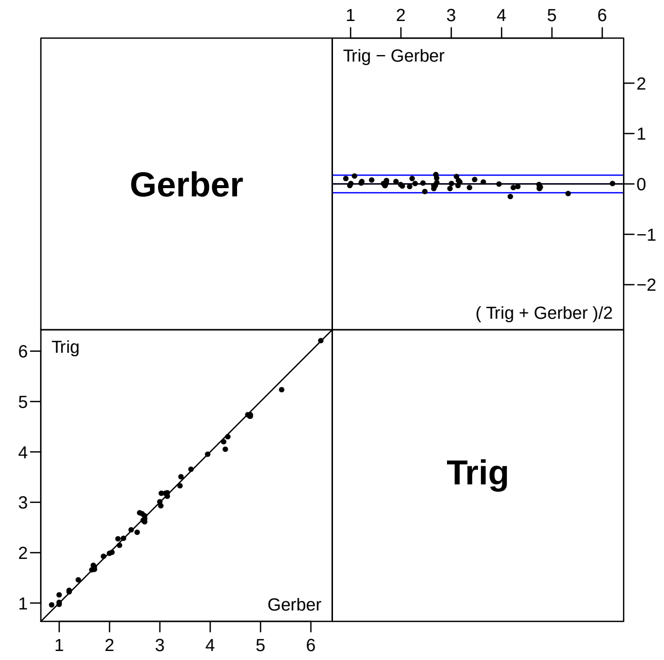


Figure 4.2: *Overview plot of the* `milk` *data, using* `plot.Meth()`*, i.e. the generic method for* `Meth` *objects.*

```
> 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:

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

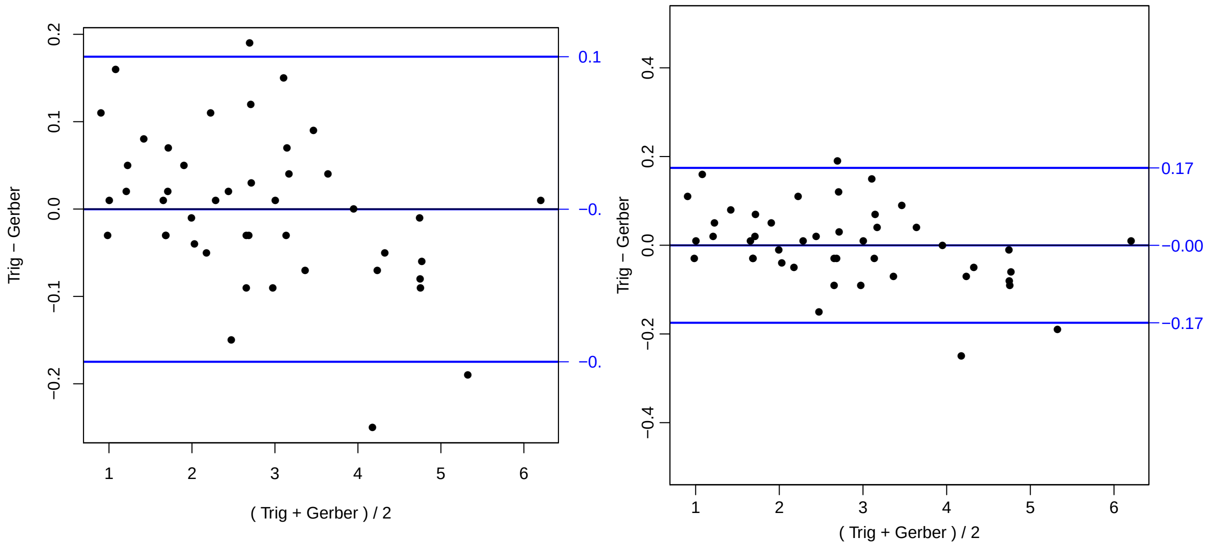


Figure 4.3: *Bland-Altman plots of the* `milk` *data, left panel with the same extent of the data on both axes, the right one with explicitly defined y-axis and explicitly defined margins — note how the right hand margin on the left plot is too narrow to accommodate the LoA.*

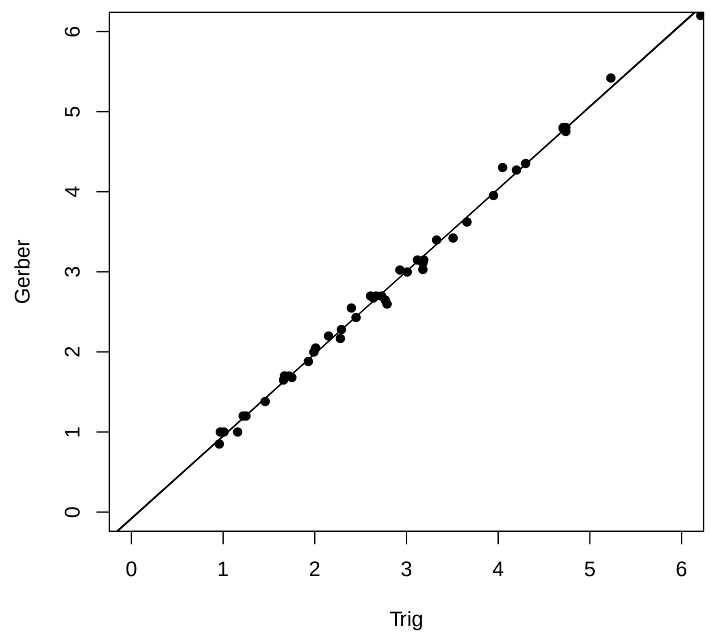


Figure 4.4: *Scatter plot of data with the two different regression lines. They are practically indistinguishable.*

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 rgeresses the differences on the averages and returns the relationships for the original variables, as well as approximate tests for the hypotheses of constant difference and constant standard deviation:

```
> DA.reg(milk)

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

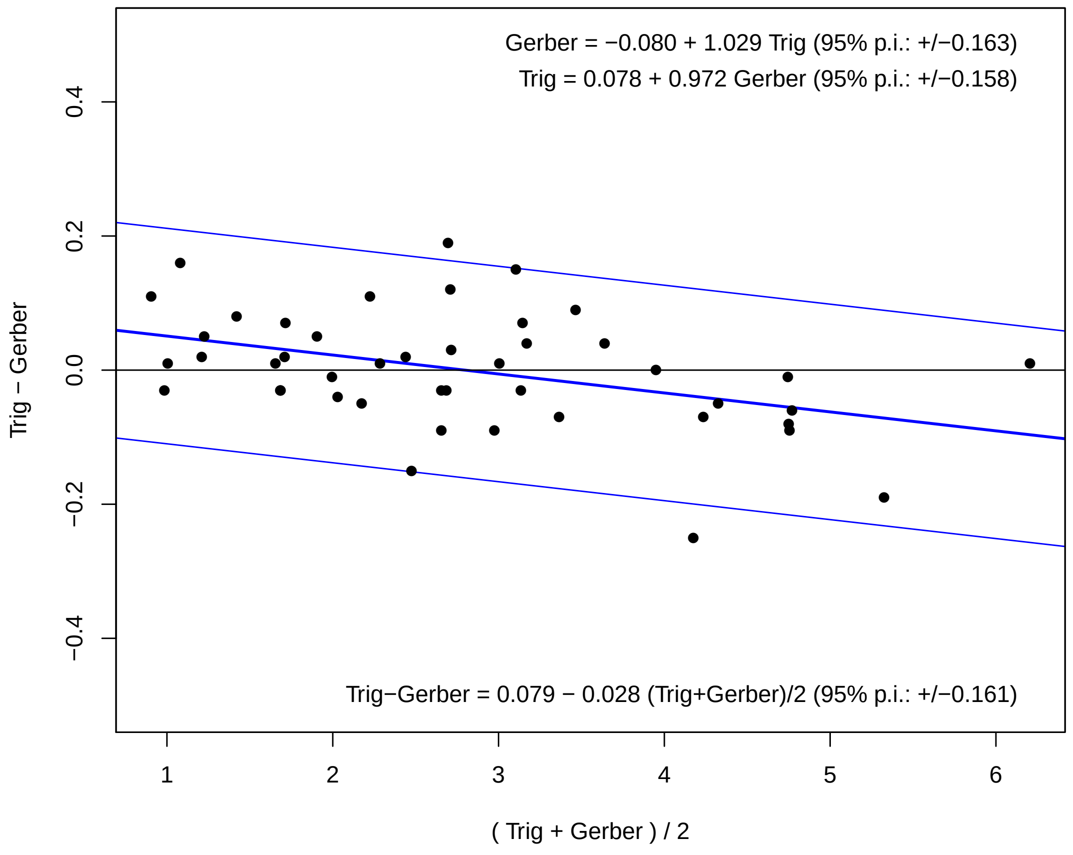


Figure 4.5: *Bland-Altman plot of the* `milk` *data with the regression of the differences on the averages and the resulting conversion equations between methods.*

4.2 Plasma volume: Single measurements by two methods

The `plvol` data from the `MethComp` package contains measurements of plasma volume expressed as a percentage of the expected value for normal individuals, measured by two different methods,

1. The two methods plotted against each other:

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

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

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

Limits of agreement:
Nadler - Hurley      2.5% limit     97.5% limit        SD(diff)
       9.262626        4.456798       14.068455        2.402914
```

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

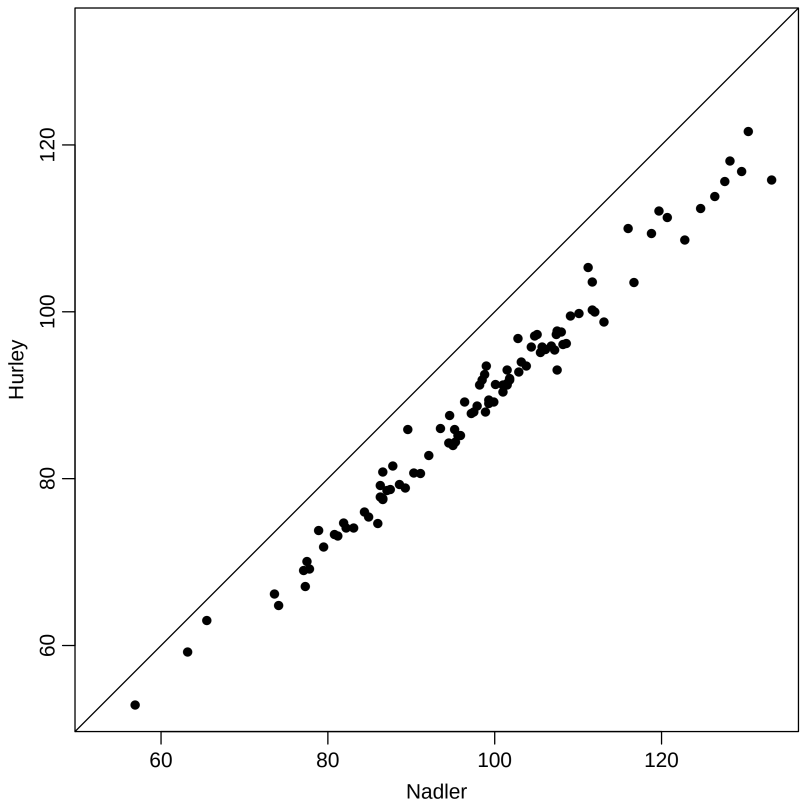


Figure 4.6: *Plot of two methods of measuring plasma volume.*

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

```
> BA.plot( plvol, mult=TRUE, ymax=log(1.25), digits=3 )

Limits of agreement:
Nadler / Hurley      2.5% limit     97.5% limit   SD(log-ratio)
     1.10395572      1.05706708      1.15292420      0.02170083
```

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

4. The estimated *ratio* between the Nadler and Hurley methods is 0.90 and the ratio of future measurements by the two methods is with 95% probability between 0.87 and 0.95. Alternatively, we may sat that for a given measurement by the Hurley method, the Nadler method will with 95% probability yield a measurement which is between 87 and 95% of this.

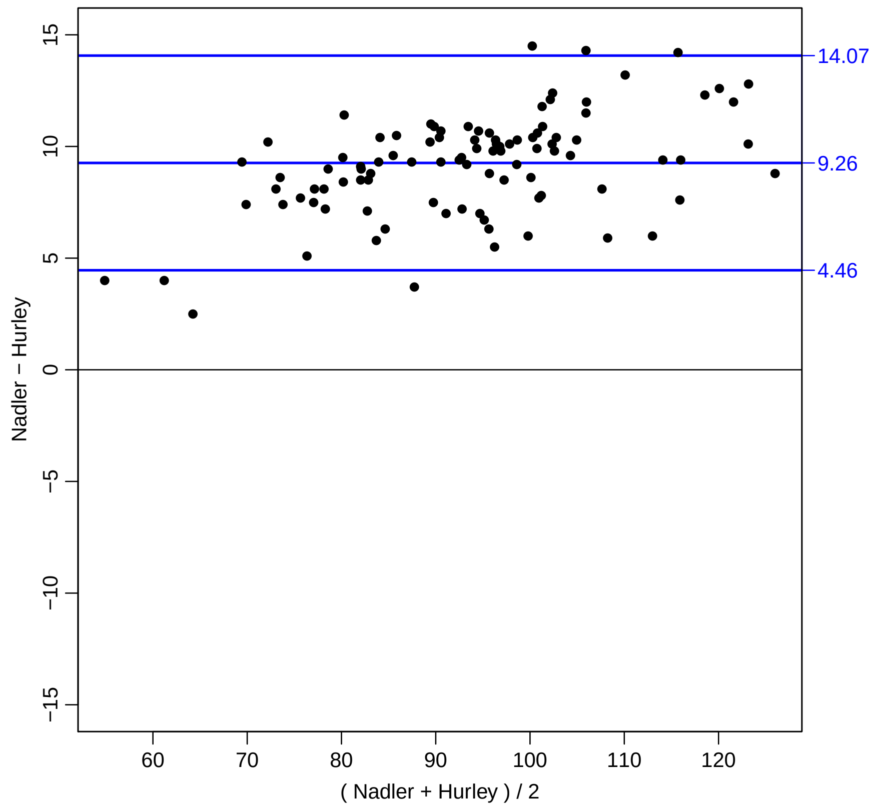


Figure 4.7: *Bland-Altman plot of two methods of measuring plasma volume.*

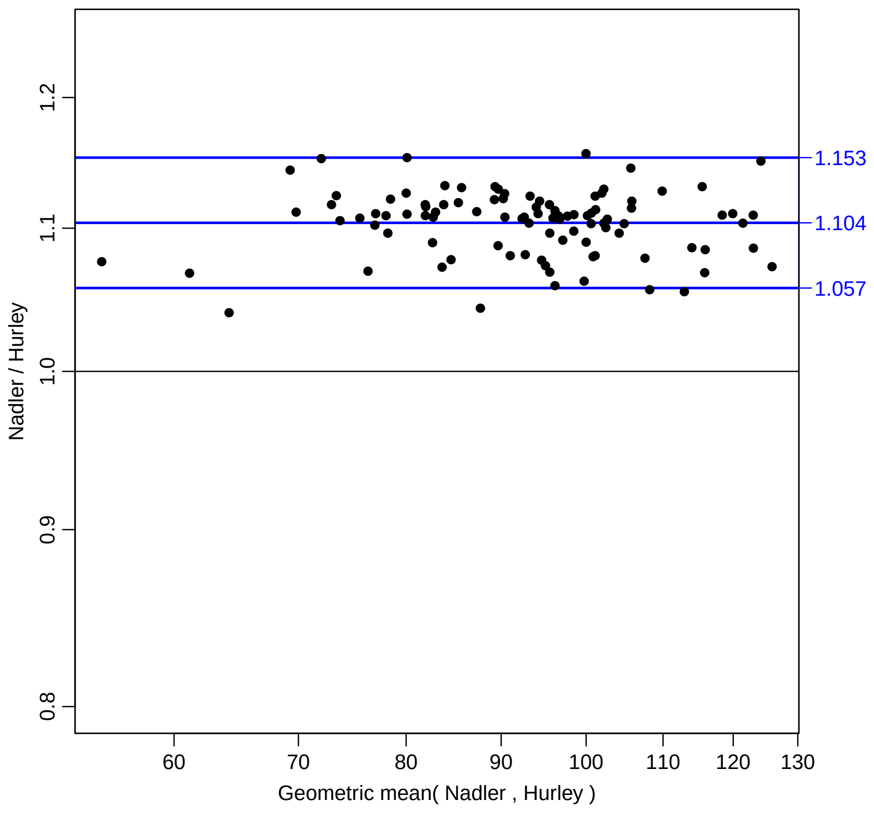


Figure 4.8: *Bland-Altman plot of two methods of measuring plasma volume, using log-transformed data, i.e. a relative scale.*

4.3 Fat measurements: Exchangeable replicates

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

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 1 ...
 $ Rep: num  1 2 3 1 2 3 1 2 3 1 ...
 $ Sub: num  1.6 1.7 1.7 2.8 2.9 2.8 2.7 2.8 2.9 3.9 ...
 $ Vic: num  4.5 4.4 4.7 6.4 6.2 6.5 3.6 3.9 4 4.3 ...

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

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

> str(vis)

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

> summary(vis)

       #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 )
```

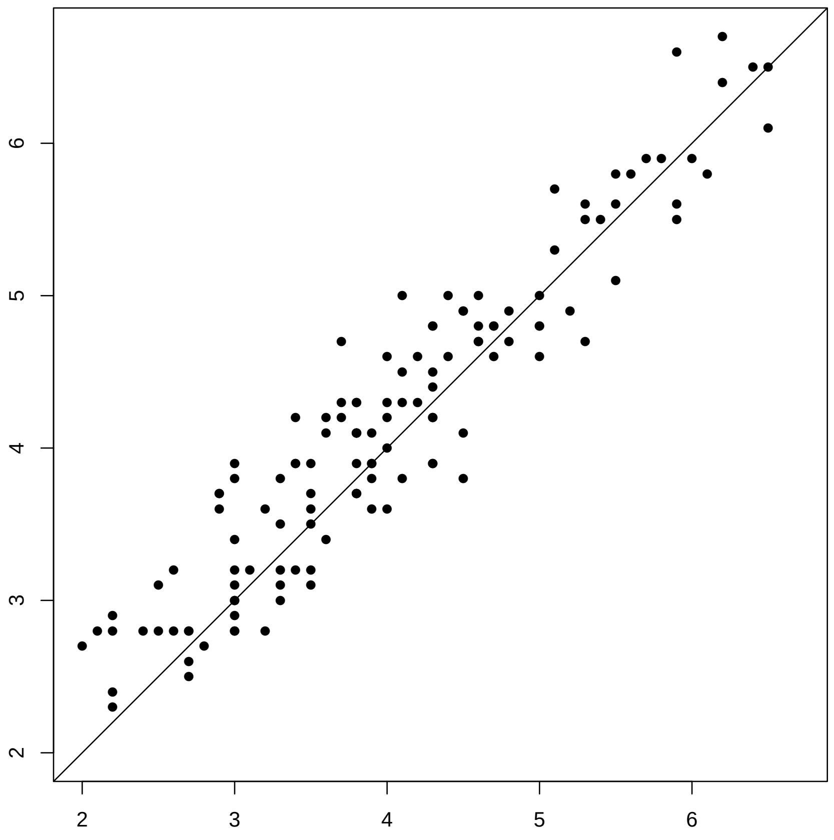


Figure 4.9: *Two observers measuring visceral fat.*

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

```
> plot( vis )

Note:
 Replicate measurements are taken as separate items!

> plot( perm.repl( vis ) )

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

   These two plots are shown in figure 4.10 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
```

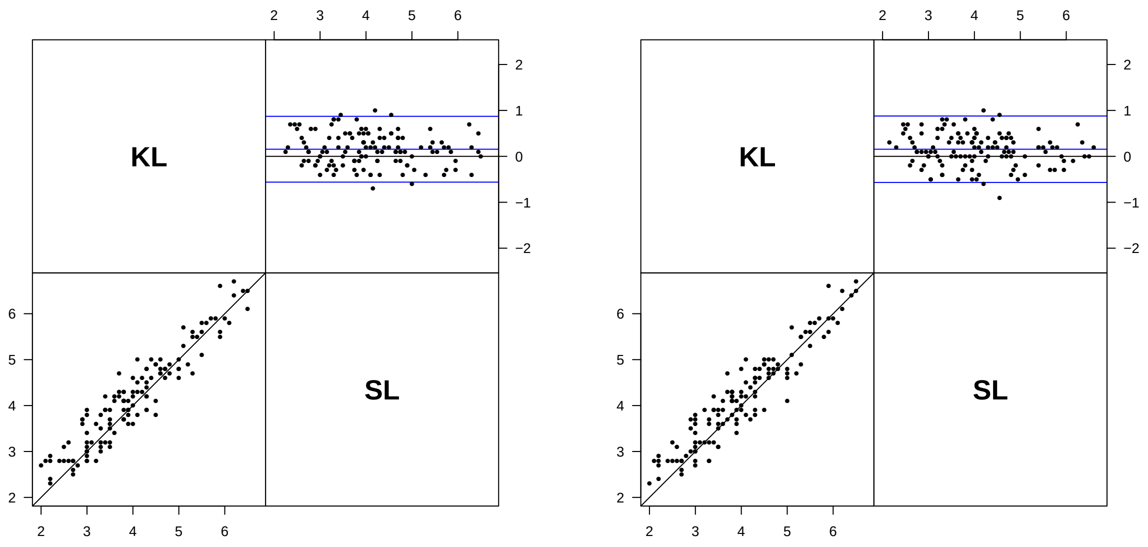


Figure 4.10: *Plot of two methods of measuring visceral fat, using different pairings of the replicates; the left panel is using the pairing in the original coding, the right panel is with a random permutation of replicates.*

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):

```
> DA.reg( vis )

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

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

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

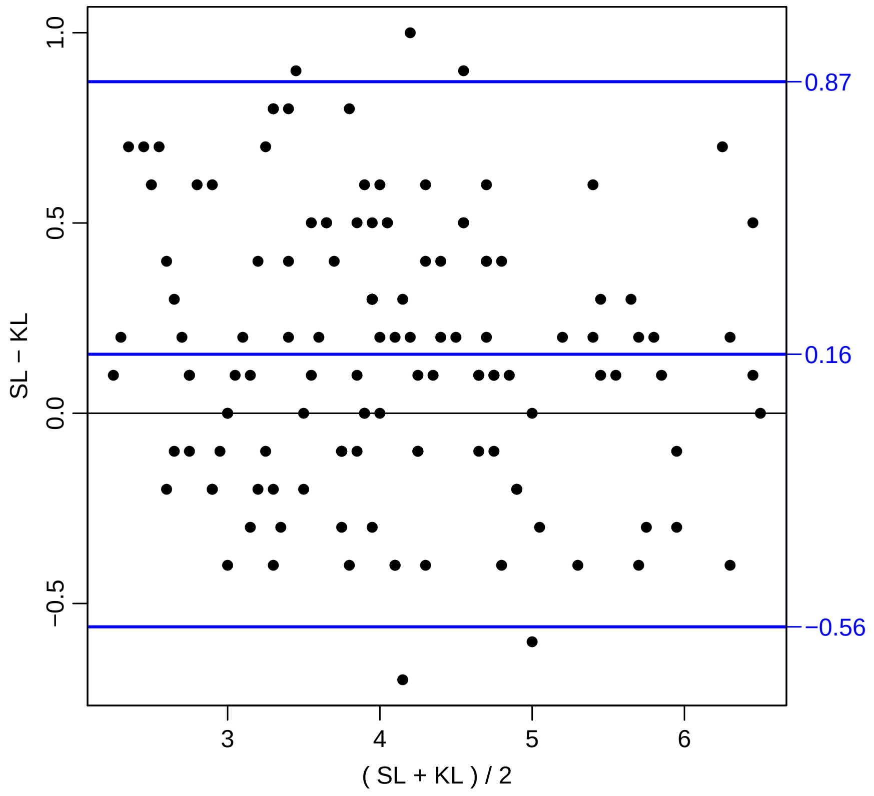


Figure 4.11: *Bland-Altman plot of two observers measuring visceral fat.*

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

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 4.12.

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:

```
> 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))
```

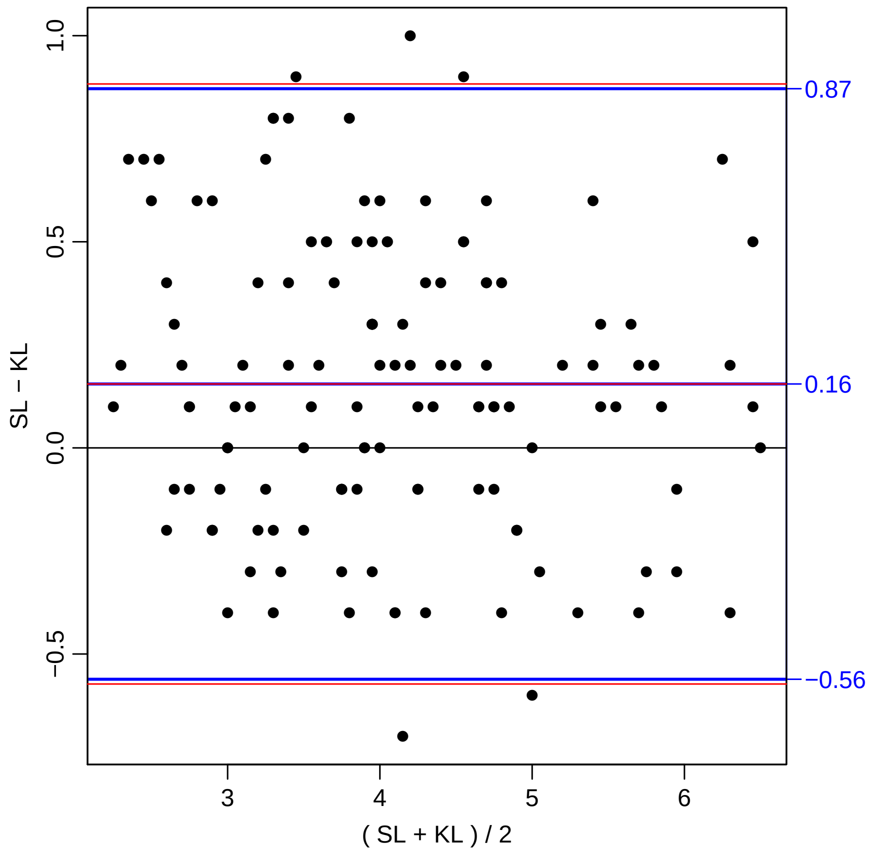


Figure 4.12: *Bland-Altman-plot of two methods of measuring visceral fat, using different pairings of the replicates. The blue lines are the LoA based on taking the paired replicates as items, the red lines are based on the estimates from the proper variance component model.*

```
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 ??.

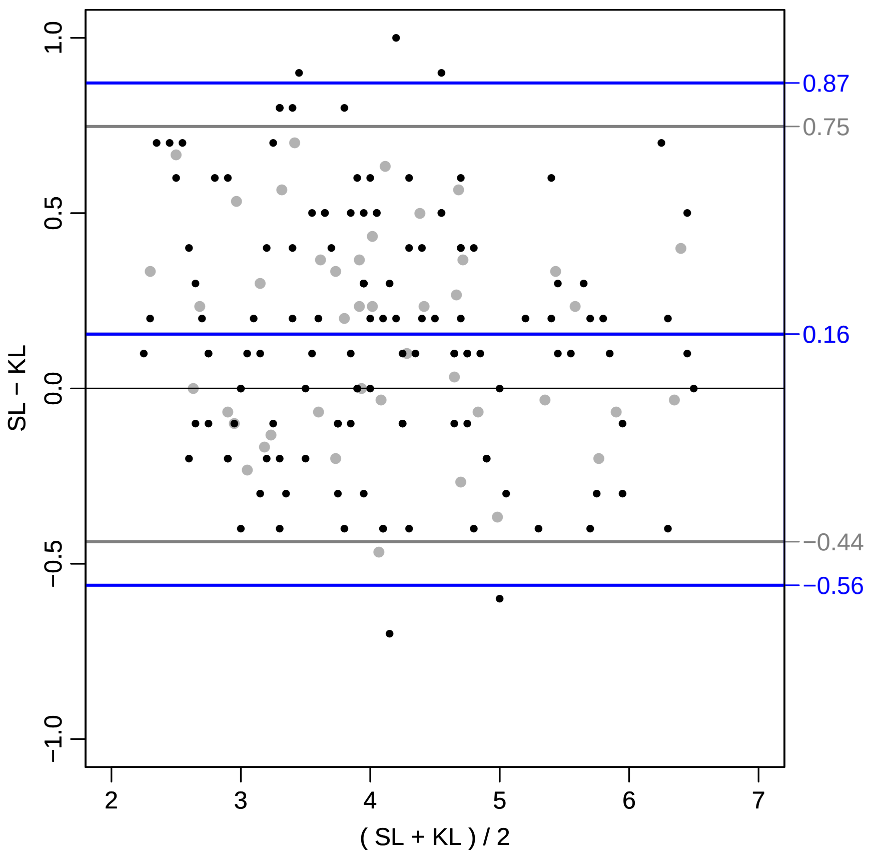


Figure 4.13: *Bland-Altman-plot of two methods of measuring visceral fat, based on the arbitrary pairing of the replicates (black) and on the mean over replicates (grey).*

4.4 Systolic blood pressure: Linked replicates by two methods

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

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

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

   > sbp <- Meth( sbp )

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

   > plot(sbp)

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

   The resulting plot is shown in figure 4.14, clearly shows that the two manual measurements are in much closer agreement than any of them are with the automatic.

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

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

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

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

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

   > BA.plot( sbp )

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

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

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

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

The resulting plots are shown in figure 4.15.

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

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

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

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

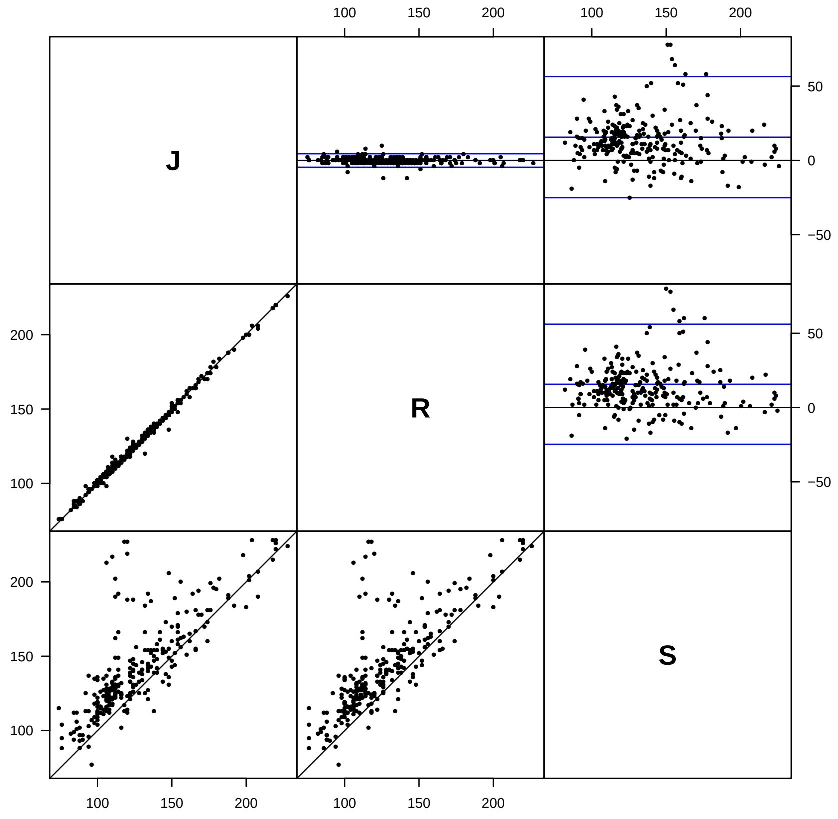


Figure 4.14: *Graphical overview of the* `sbp` *data. The methods J and R are two human observers, whereas method S is an automatic device.*

```
Linear mixed-effects model fit by REML
  Data: sbp
  Log-restricted-likelihood: -1163.807
  Fixed: y ~ meth + item
 (Intercept)        methR        item2        item3        item4        item5
103.47872449  -0.08627451   5.82189382 -22.17810618   1.89313629  13.45293925
       item6        item7        item8        item9       item10       item11
 25.82189382   5.82189382   7.96437876   2.92875753  -2.54706075   0.78627258
      item12       item13       item14       item15       item16       item17
 10.85751506   8.19084839   1.89313629   1.29771210  15.29771210  -2.10686371
      item18       item19       item20       item21       item22       item23
 14.63104543  33.29771210  43.29771210  53.36895457  40.17810618  66.03562124
      item24       item25       item26       item27       item28       item29
 60.48856049  39.22646963  27.22646963  37.59542419  45.22646963 115.89313629
      item30       item31       item32       item33       item34       item35
 95.66666667 -15.14248494  14.85751506  18.63104543  22.03562124  15.89313629
      item36       item37       item38       item39       item40       item41
-12.70228790   4.19084839 105.29771210  25.00000000  30.55980296  -9.07124247
      item42       item43       item44       item45       item46       item47
 -8.10686371  17.52418172  58.55980296  -2.17810618  24.26209086  11.22646963
      item48       item49       item50       item51       item52       item53
 31.08398468  49.22646963 -11.80915161  52.63104543  -1.44019704   1.15522715
      item54       item55       item56       item57       item58       item59
 -4.47581828 -24.17810618   1.59542419   5.45293925  75.45293925  52.92875753
      item60       item61       item62       item63       item64       item65
 35.96437876  93.52418172 -11.73790914  24.26209086  36.92875753  33.59542419
      item66       item67       item68       item69       item70       item71
 53.82189382  29.59542419   9.52418172  13.22646963  17.52418172 112.63104543
      item72       item73       item74       item75       item76       item77
 30.55980296  53.89313629 -19.44019704  70.48856049  75.59542419  13.22646963
      item78       item79       item80       item81       item82       item83
 15.29771210   4.55980296   6.26209086  36.78627258   4.78627258   6.92875753
      item84       item85
 -2.10686371  12.48856049

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

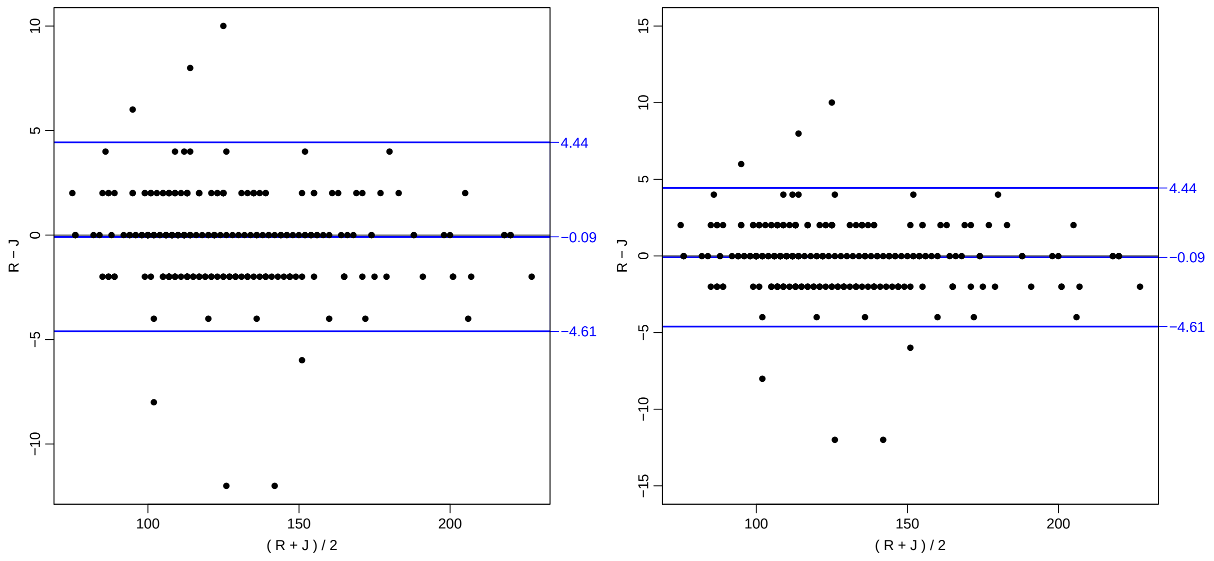


Figure 4.15: *Bland-Altman plot of the* sbp *data. Replicates are linked between methods, so the single replicates in the data has been used as single measurements when doing the Bland-Altman plot. The only difference between the two plots is the scaling of the y-axis.*

```
StdDev: 0.2483701 0.2483701

 Formula: ~1 | repl %in% item
        (Intercept) Residual
StdDev:    5.932962 1.485870

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

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

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

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

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

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

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

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

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

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

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

The following variables from the dataframe
"sbp" are used as the Meth variables:
meth: meth
item: item
```

```
repl: repl
   y: y
       #Replicates
Method          3 #Items #Obs: 765 Values:  min med max
     J         85     85       255           74 120 228
     R         85     85       255           76 120 226
     S         85     85       255           77 135 228
```

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

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

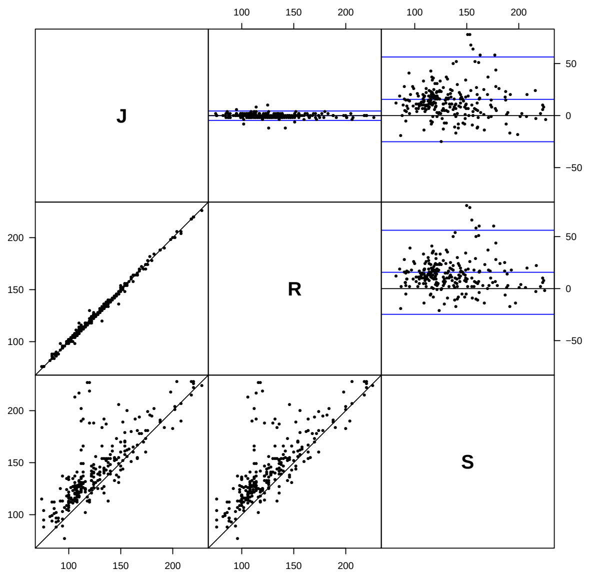
The two resulting plots are shown in figure 4.16.

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

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

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

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



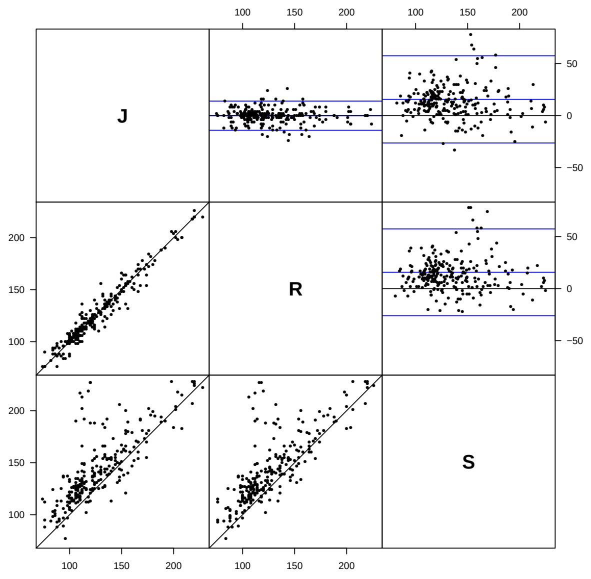


Figure 4.16: *Graphical overview of the* `sbp` *data; the left panel with the original replicate numbers used for matching; the other with replicates permuted randomly within methods.*

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

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

 Conversion between methods:
            alpha    beta      sd   LoA: lower   upper
To: From:
J   J        0.000   1.000   7.505      -15.009  15.009
    R        0.086   1.000   7.508      -14.931  15.103
    S      -15.620   1.000  20.814      -57.247  26.008
R   J       -0.086   1.000   7.508      -15.103  14.931
    R        0.000   1.000   7.512      -15.025  15.025
    S      -15.706   1.000  20.815      -57.336  25.924
S   J       15.620   1.000  20.814      -26.008  57.247
    R       15.706   1.000  20.815      -25.924  57.336
    S        0.000   1.000  12.770      -25.541  25.541

 Variance components (sd):
   IxR    MxI   res
J 1.723  0.000 5.306
R 1.723  0.000 5.312
S 1.723 17.986 9.030
```

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

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

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

8. Alternatively this could be formulated as a 95% prediction interval for R given a measurement by J, y_J, which would be

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

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

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

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

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

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

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

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

4.5 Measurement of HbA_{1c} I: Machine and specimen as method

1. First we load the hba1c data and take a look at the structure of data:

```
> data( hba1c )
> with( hba1c, table( d.samp, d.ana ) )

      d.ana
d.samp   1   2   3   4   5
     1  38 114 113 114  76
     2   0  38 114 114 114

> with( hba1c, ftable( dev, type, d.ana ) )

           d.ana  1  2  3  4  5
dev   type
BR.V2 Cap        19 38 38 38 19
      Ven         0 19 38 38 38
BR.VC Cap         0 19 38 38 38
      Ven         0 19 38 38 38
Tosoh Cap        19 38 38 38 19
      Ven         0 19 37 38 38
```

2. The dataset does not have the standard structure, it lacks a definition of method and replicate. We can provide these by using the interaction between dev and type and the day of analysis as replicate number:

```
> hb <- transform( hba1c, meth = interaction( dev, type ),
+                         repl = d.ana )
> str( hb )

'data.frame':       835 obs. of  8 variables:
 $ dev   : Factor w/ 3 levels "BR.V2","BR.VC",..: 2 2 2 2 2 2 2 2 1 1 ...
 $ type  : Factor w/ 2 levels "Cap","Ven": 2 2 2 2 1 1 1 1 2 2 ...
 $ item  : num  12 12 12 12 12 12 12 12 12 12 ...
 $ d.samp: num  1 1 1 1 1 1 1 1 1 1 ...
 $ d.ana : num  2 3 4 5 2 3 4 5 2 3 ...
 $ y     : num  8.7 8.7 8.7 8.7 9.2 9 8.8 8.7 9.4 9.3 ...
 $ meth  : Factor w/ 6 levels "BR.V2.Cap","BR.VC.Cap",..: 5 5 5 5 2 2 2 2 4 4 ...
 $ repl  : num  2 3 4 5 2 3 4 5 2 3 ...

> hb <- Meth( hb, 7, 3, 5, 6 )

The following variables from the dataframe
"hb" are used as the Meth variables:
meth: meth
item: item
repl: d.ana
   y: y
          #Replicates
Method         3    4 #Items #Obs: 835 Values:  min med  max
  BR.V2.Cap    0   38     38       152          5.3 8.0 12.6
  BR.VC.Cap   19   19     38       133          5.3 8.2 12.1
  Tosoh.Cap    0   38     38       152          5.0 7.8 11.8
  BR.V2.Ven   19   19     38       133          5.5 8.1 12.0
  BR.VC.Ven   19   19     38       133          5.3 8.0 11.6
  Tosoh.Ven   20   18     38       132          5.3 8.0 12.1
```

Note that the replication structure i slightly different between machines and specimens (venous/capillary). This is because of technical limitations; only some machines and specimens allow analysis on the same day as the sampling.

3. In figure 4.17 is an overview plot of the data. This plot is made under the assumption that replicates are linked by replicate number, in this case day of analysis. This is presumably a sensible assumption, but we will see later.

```
> plot( hb )

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

There is a tendency that comparisons with the machine `BR-VC` have a higher variance than other comparisons.

4. It is difficult to say if we can consider the replicates exchangeable within methods. But since samples are analyzed on different days we would suspect that there were some linking, so an individual by replicate interaction ma be in its place.

5. The "standard model" for analyzing data of this kind is:

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

This is actually the most elaborate model fitted by `MethComp` but also the default:

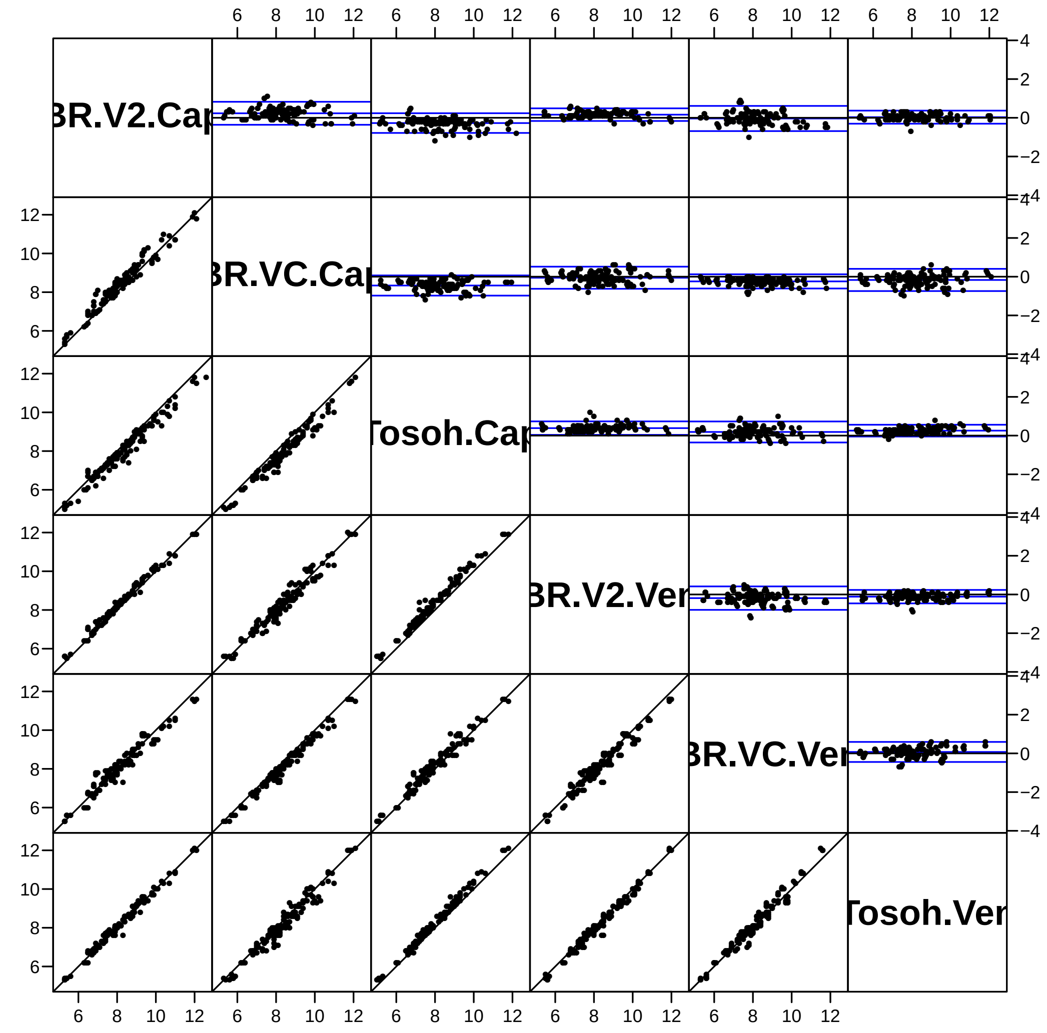


Figure 4.17: *Overview of the HbA$_{1c}$-data. Replicate measurements on the same day of analysis are lined as items.*

```
> m0 <- MCmcmc( hb, linked=TRUE, n.iter=100 )

Comparison of 6 methods, using 835 measurements
on 38 items, with up to 4 replicate measurements,
(replicate values are in the set: 1 2 3 4 5 )
( 6 * 38 * 4 = 912 ):

No. items with measurements on each method:
            #Replicates
Method         3     4 #Items #Obs: 835 Values:  min med  max
  BR.V2.Cap    0    38     38       152          5.3 8.0 12.6
  BR.VC.Cap   19    19     38       133          5.3 8.2 12.1
  Tosoh.Cap    0    38     38       152          5.0 7.8 11.8
  BR.V2.Ven   19    19     38       133          5.5 8.1 12.0
  BR.VC.Ven   19    19     38       133          5.3 8.0 11.6
  Tosoh.Ven   20    18     38       132          5.3 8.0 12.1

Simulation run of a model with
- method by item and item by replicate interaction:
- using 4 chains run for 100 iterations
  (of which 50 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 200 observations.

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4: Sampling has been
```

The resulting `MethComp` object `m0` is quite big, so it is more handily represented by its default print method:

```
> m0

 Conversion between methods:
                       alpha     beta       sd
To:       From:
BR.V2.Cap BR.V2.Cap    0.000    1.000    1.115
          BR.VC.Cap    2.360    0.735    1.434
          Tosoh.Cap    3.779    0.618    0.792
          BR.V2.Ven  -56.098    8.659   10.230
          BR.VC.Ven   11.106    0.542    1.349
          Tosoh.Ven    4.455    0.520    1.375
BR.VC.Cap BR.V2.Cap   -3.209    1.471    3.854
          BR.VC.Cap    0.000    1.000    2.736
          Tosoh.Cap    0.915    0.677    2.948
          BR.V2.Ven  -66.986   10.488   21.663
          BR.VC.Ven   -6.047    1.940    3.379
          Tosoh.Ven    1.000    0.866    4.143
Tosoh.Cap BR.V2.Cap   -6.115    1.769    1.764
          BR.VC.Cap   -1.350    2.429    5.813
          Tosoh.Cap    0.000    1.000    0.938
          BR.V2.Ven -133.118   16.928   27.680
          BR.VC.Ven    2.418    0.752    1.408
          Tosoh.Ven  -21.414    3.585    4.582
BR.V2.Ven BR.V2.Cap    6.479    0.228    1.535
          BR.VC.Cap    6.387    0.133    1.559
          Tosoh.Cap    7.864    0.059    1.407
          BR.V2.Ven    0.000    1.000    1.983
          BR.VC.Ven    7.503    0.050    1.481
          Tosoh.Ven    9.646    0.141    1.402
BR.VC.Ven BR.V2.Cap  -20.494    5.656    6.656
          BR.VC.Cap    3.117    0.650    1.998
          Tosoh.Cap   -3.217    1.407    2.215
          BR.V2.Ven -150.192   21.172   39.484
          BR.VC.Ven    0.000    1.000    2.032
          Tosoh.Ven -216.421    8.181   10.162
Tosoh.Ven BR.V2.Cap   -8.563    2.196    2.864
          BR.VC.Cap   -1.155    1.183    3.182
          Tosoh.Cap    5.973    0.620    1.397
          BR.V2.Ven  -68.531   12.047   13.370
          BR.VC.Ven   26.453    1.601    3.569
          Tosoh.Ven    0.000    1.000    1.973
 Variance components (sd):
         s.d.
Method       IxR   MxI   res
  BR.V2.Cap 0.289 0.754 0.221
  BR.VC.Cap 0.019 1.397 0.104
  Tosoh.Cap 0.129 1.929 0.145
  BR.V2.Ven 0.047 1.436 0.052
  BR.VC.Ven 0.360 0.001 0.663
  Tosoh.Ven 0.094 1.393 0.072

Variance components with 95 % cred.int.:
    method BR.V2.Cap             BR.VC.Cap             Tosoh.Cap             BR.V2.Ven             BR.VC.
    qnt          50%  2.5% 97.5%       50%  2.5% 97.5%       50%  2.5% 97.5%       50%  2.5% 97.5%
SD
IxR            0.289 0.000 2.085     0.019 0.000 0.134     0.129 0.000 2.210     0.047 0.007 2.109     0.
MxI            0.754 0.134 1.543     1.397 1.015 2.852     1.929 0.213 6.032     1.436 0.201 2.404     0.
res            0.221 0.091 0.294     0.104 0.090 0.122     0.145 0.112 0.172     0.052 0.039 0.106     0.
tot            1.341 0.743 2.094     1.402 1.021 2.857     2.243 1.172 6.037     1.797 1.075 2.405     1.

Mean parameters with 95 % cred.int.:
                                50%     2.5%   97.5% P(>0/1)
alpha[BR.VC.Cap.BR.V2.Cap]   -3.679 -322.128   0.948    0.25
alpha[Tosoh.Cap.BR.V2.Cap]   -6.777 -204.053   0.269    0.25
alpha[BR.V2.Ven.BR.V2.Cap]    5.719    1.562   7.476    1.00
alpha[BR.VC.Ven.BR.V2.Cap]  -39.858 -773.122   7.192    0.25
alpha[Tosoh.Ven.BR.V2.Cap]   -9.696 -316.784   8.168    0.25
alpha[BR.V2.Cap.BR.VC.Cap]    2.034   -3.195   8.188    0.75
alpha[Tosoh.Cap.BR.VC.Cap]   -2.303 -101.967   7.229    0.50
alpha[BR.V2.Ven.BR.VC.Cap]    6.496    5.002   9.064    1.00
alpha[BR.VC.Ven.BR.VC.Cap]    2.307 -373.373   6.881    0.75
alpha[Tosoh.Ven.BR.VC.Cap]   -1.514   -7.137   8.182    0.50
alpha[BR.V2.Cap.Tosoh.Cap]    3.391   -0.287   7.863    0.75
alpha[BR.VC.Cap.Tosoh.Cap]    0.435  -86.251   7.698    0.50
alpha[BR.V2.Ven.Tosoh.Cap]    7.673    5.683   8.544    1.00
alpha[BR.VC.Ven.Tosoh.Cap]   -3.604  -27.073   7.380    0.25
alpha[Tosoh.Ven.Tosoh.Cap]    3.228  -85.296   8.153    0.50
alpha[BR.V2.Cap.BR.V2.Ven]  -50.395 -118.651  -1.892    0.00
alpha[BR.VC.Cap.BR.V2.Ven]  -63.054 -817.005 -12.567    0.00
alpha[Tosoh.Cap.BR.V2.Ven] -136.327 -273.509 -56.292    0.00
alpha[BR.VC.Ven.BR.V2.Ven] -162.390 -977.833  -4.364    0.00
alpha[Tosoh.Ven.BR.V2.Ven]  -88.836 -794.913   2.982    0.25
alpha[BR.V2.Cap.BR.VC.Ven]    3.881  -70.374   8.148    0.75
alpha[BR.VC.Cap.BR.VC.Ven]   -7.148  -20.013   8.126    0.25
alpha[Tosoh.Cap.BR.VC.Ven]    2.139  -88.337   6.121    0.75
alpha[BR.V2.Ven.BR.VC.Ven]    6.915    2.824   8.895    1.00
alpha[Tosoh.Ven.BR.VC.Ven]   -2.972  -96.793   8.175    0.50
alpha[BR.V2.Cap.Tosoh.Ven]    3.902 -137.338   8.183    0.75
alpha[BR.VC.Cap.Tosoh.Ven]    0.696 -145.298   3.294    0.50
alpha[Tosoh.Cap.Tosoh.Ven]  -21.364 -129.563   7.217    0.50
alpha[BR.V2.Ven.Tosoh.Ven]    6.939   -3.279   9.062    0.75
alpha[BR.VC.Ven.Tosoh.Ven]  -63.265 -229.082   7.416    0.50
beta[BR.VC.Cap.BR.V2.Cap]     1.471    0.297  39.942    0.75
beta[Tosoh.Cap.BR.V2.Cap]     1.769    0.936  25.942    0.75
beta[BR.V2.Ven.BR.V2.Cap]     0.228    0.062   0.834    0.00
beta[BR.VC.Ven.BR.V2.Cap]     5.656    0.102  96.064    0.72
beta[Tosoh.Ven.BR.V2.Cap]     2.196    0.059  39.158    0.75
beta[BR.V2.Cap.BR.VC.Cap]     0.735    0.025   3.370    0.25
beta[Tosoh.Cap.BR.VC.Cap]     2.429    0.082  13.228    0.50
beta[BR.V2.Ven.BR.VC.Cap]     0.133    0.011   0.395    0.00
beta[BR.VC.Ven.BR.VC.Cap]     0.650    0.307  46.216    0.25
beta[Tosoh.Ven.BR.VC.Cap]     1.183    0.056   4.496    0.51
beta[BR.V2.Cap.Tosoh.Cap]     0.618    0.039   1.069    0.25
beta[BR.VC.Cap.Tosoh.Cap]     0.677    0.076  12.135    0.50
beta[BR.V2.Ven.Tosoh.Cap]     0.059    0.029   0.142    0.00
beta[BR.VC.Ven.Tosoh.Cap]     1.407    0.084   4.447    0.75
beta[Tosoh.Ven.Tosoh.Cap]     0.620    0.063  12.022    0.50
beta[BR.V2.Cap.BR.V2.Ven]     8.659    1.199  16.038    1.00
beta[BR.VC.Cap.BR.V2.Ven]    10.488    2.531  90.591    1.00
beta[Tosoh.Cap.BR.V2.Ven]    16.928    7.024  34.262    1.00
beta[BR.VC.Ven.BR.V2.Ven]    21.172    1.546 120.507    1.00
```

```
beta[Tosoh.Ven.BR.V2.Ven]    12.047    0.909  88.243    0.75
beta[BR.V2.Cap.BR.VC.Ven]     0.542    0.010   9.785    0.28
beta[BR.VC.Cap.BR.VC.Ven]     1.940    0.022   3.254    0.75
beta[Tosoh.Cap.BR.VC.Ven]     0.752    0.225  11.972    0.25
beta[BR.V2.Ven.BR.VC.Ven]     0.050    0.008   0.647    0.00
beta[Tosoh.Ven.BR.VC.Ven]     1.601    0.034  13.063    0.50
beta[BR.V2.Cap.Tosoh.Ven]     0.520    0.026  16.879    0.25
beta[BR.VC.Cap.Tosoh.Ven]     0.866    0.222  17.829    0.49
beta[Tosoh.Cap.Tosoh.Ven]     3.585    0.083  15.958    0.50
beta[BR.V2.Ven.Tosoh.Ven]     0.141    0.011   1.100    0.25
beta[BR.VC.Ven.Tosoh.Ven]     8.181    0.077  29.166    0.50

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

6. We can also get a graphical overview of the relationships between the methods by using the function `plot.MethComp`. Since the resulting object is of class `MethComp`, it suffices to say:

```
> plot( m0, grid=TRUE )
```

7. There is a zillion arguments to `MethComp` (did you remember to type "`?MethComp`"?), but for a start we just use the default settings — in "real" applications one would use a larger number of iterations in order to be on the safe side. Since there are 6 methods we can plot the variance components associated with each of them in a 2 by 3 layout, try:

```
> par(mfrow=c(2,3))
> plot.VarComp(m0)
```

8. Clearly, the posterior distributions of the variance components in figure 4.19 are not very well determined, so we re-fit the model using substantially more iterations. Try to enclose the call to `MethComp` in a `system.time()` in order to see how much time it takes.

```
> system.time(
+ m1 <- MCmcmc( hb, n.iter=100, n.chains=5 )
+            )

Comparison of 6 methods, using 835 measurements
on 38 items, with up to 4 replicate measurements,
(replicate values are in the set: 1 2 3 4 5 )
( 6 * 38 * 4 = 912 ):

No. items with measurements on each method:
            #Replicates
Method         3    4 #Items #Obs: 835 Values:  min med  max
  BR.V2.Cap    0   38     38       152          5.3 8.0 12.6
  BR.VC.Cap   19   19     38       133          5.3 8.2 12.1
  Tosoh.Cap    0   38     38       152          5.0 7.8 11.8
  BR.V2.Ven   19   19     38       133          5.5 8.1 12.0
  BR.VC.Ven   19   19     38       133          5.3 8.0 11.6
  Tosoh.Ven   20   18     38       132          5.3 8.0 12.1

Simulation run of a model with
- method by item and item by replicate interaction:
- using 5 chains run for 100 iterations
  (of which 50 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 250 observations.

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4: Initializing chai
   user  system elapsed
  23.80    0.23   24.48
```

9. Having done this more elaborate simulation we can get a more detailed plot by fiddling the graphics parameters a bit:

```
> par(mfrow=c(2,3),mar=c(3,1,2,1),mgp=c(3,1,0)/1.6)
> plot.VarComp(m1,grid=seq(0,1.8,0.1))
```

10. Based on the posterior distributions shown in figure 4.20, the following conclusions may be drawn:

    - The method×item effect is largest for `BR.VC` (the existing machine) and smallest for `Tosoh`. This indicates that `Tosoh` has be best stability of measurements across patients.
    - The residual variance is pretty much the same across machines, but substantially smaller for venous than for capillary samples.
    - The item×replicate variance component may be large, but is very badly estimated, i.e. there is presumably not much information about it in the dataset.

      One explanation may be that there is a systematic effect of replicate — recall that replicates are not exchangeable because they refer to different days of analysis. Hence a

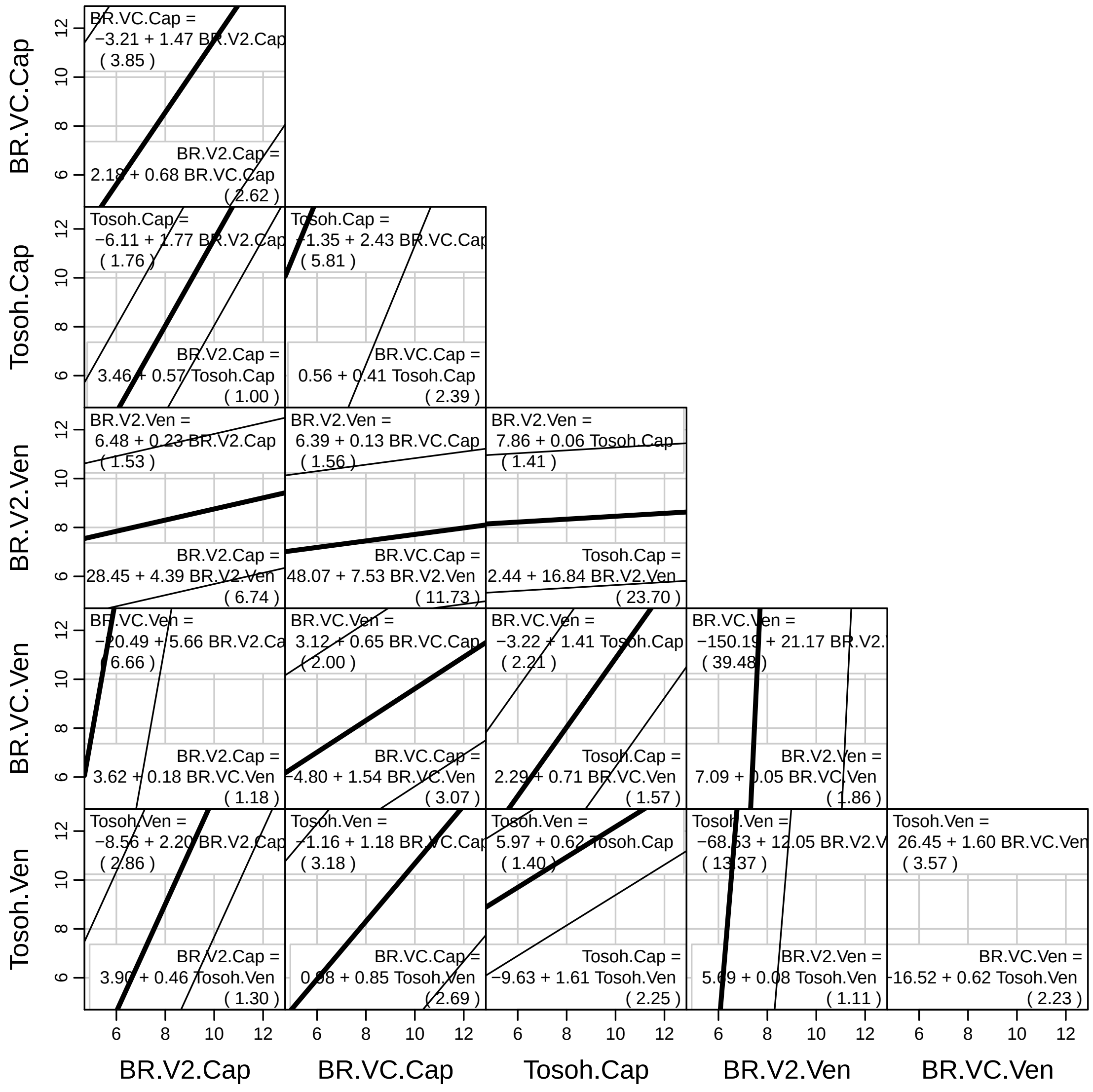


Figure 4.18: *Estimated translation formulae between methods, based on the posterior distribution of the identifiable translation parameters.*

possibility would be to explore whether there was a systematic effect of analysis day alone or analysis day by machine. This systematic feature is however not accommodated by the `MethComp` function. This can be implemented by using the `code.only` argument of `MethComp`, which produces the `BUGS` code in a separate file, which can then be edited to accommodate the systematic effects mentioned.

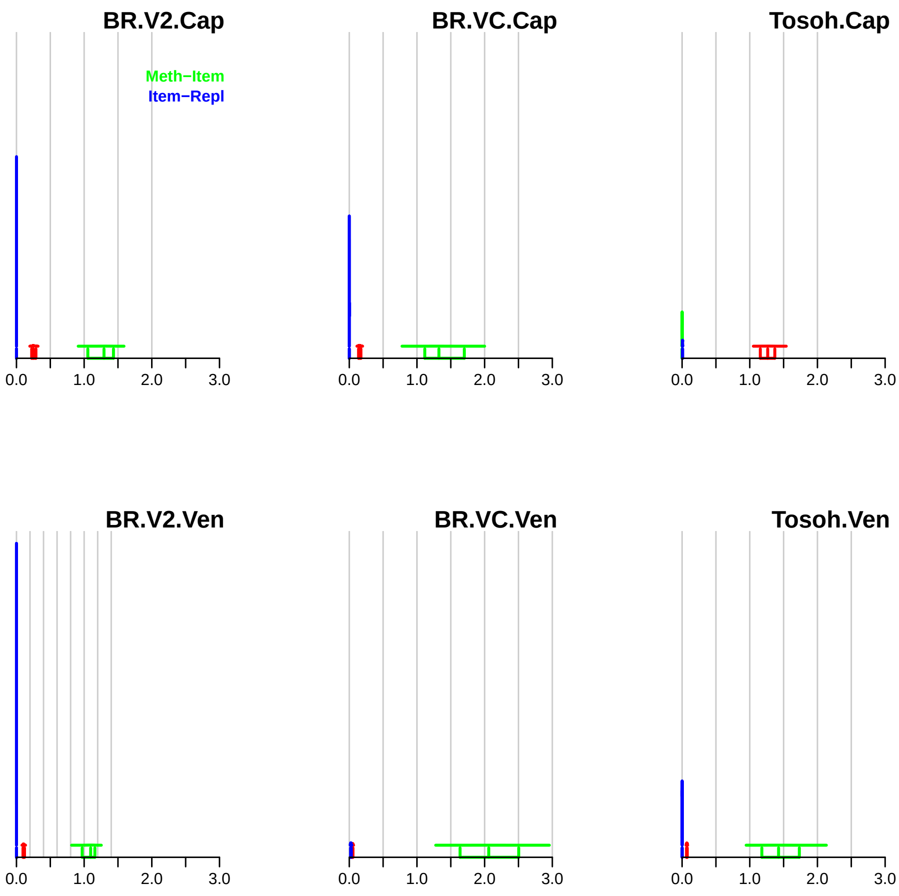


Figure 4.19: *Posterior distributions of the variance components for the 6 methods, based on 1000 burn-in and 1000 samples from 3 chains.*

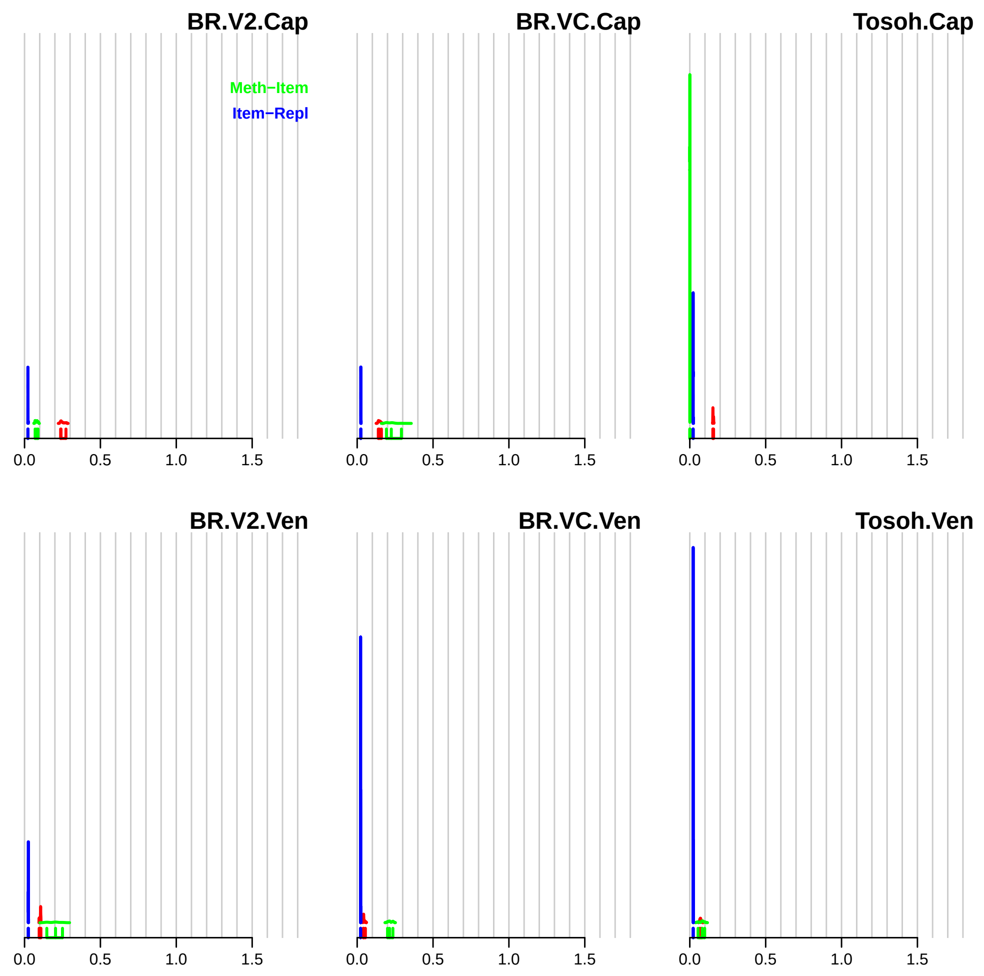


Figure 4.20: *Posterior distributions of the variance components for the 6 methods, based on 5000 burn-in and 5000 samples from 5 chains.*

4.6 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 )

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

   > summary( ox )

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

   The `summary` method for `Meth` objects reveals that most children have three replicates by each method.

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 4.22:

   ```
   > 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) )

 Conversion between methods:
              alpha    beta      sd   LoA: lower   upper
To:   From:
CO    CO      0.000   1.000   3.146       -6.293   6.293
      pulse   2.470   1.000   6.169       -9.867  14.808
pulse CO     -2.470   1.000   6.169      -14.808   9.867
      pulse   0.000   1.000   5.649      -11.298  11.298

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

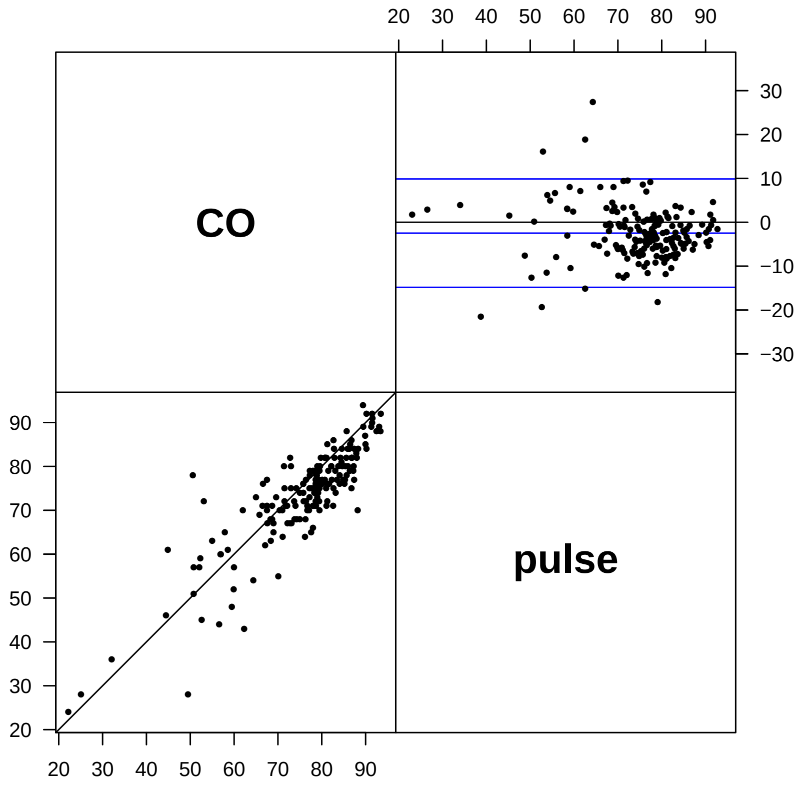


Figure 4.21: *A scatterplot (lower left) and Bland-Altman plot (upper right) of the oximetry data, using the linked replicates as items.*

```
CO    3.416 2.928 2.225
pulse 3.416 2.928 3.994
```

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

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

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

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

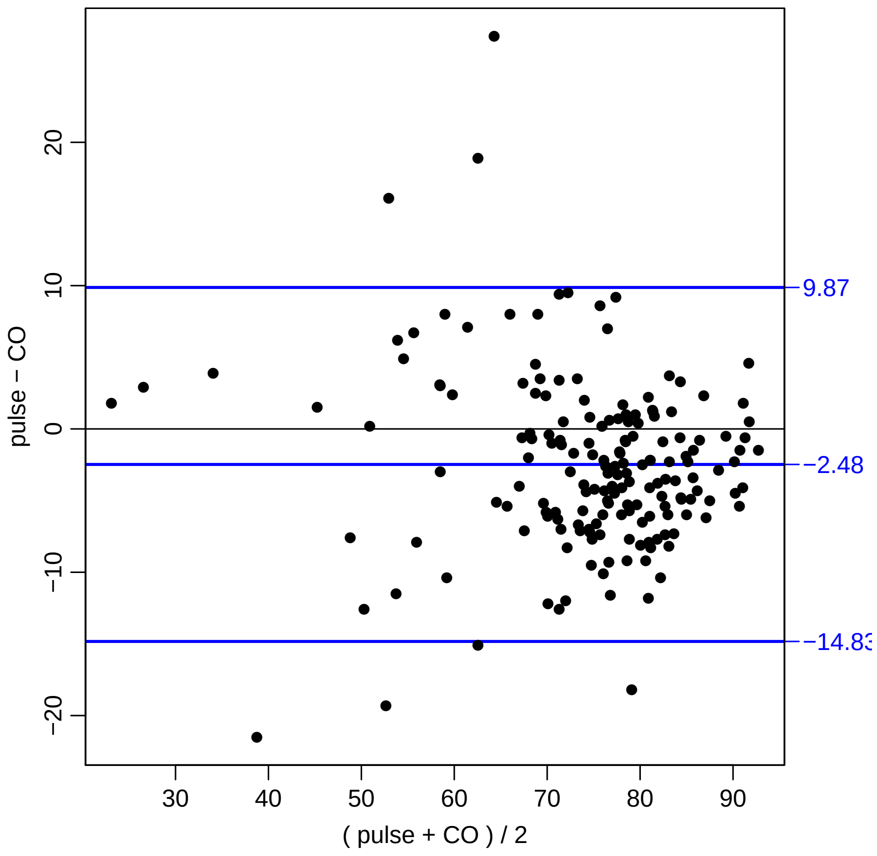


Figure 4.22: *A Bland-Altman plot of the oximetry data, using the linked replicates as items.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

AltReg converged after  14 iterations
Last convergence criterion was  0.0009863462
```

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

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

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

> round( BAox$VarComp, 4 )

         IxR   MxI    res
CO    3.4157 2.928 2.2249
pulse 3.4157 2.928 3.9945

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

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

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`:

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

```
 Conversion between methods:
             alpha   beta sd.pred beta=1 s.d.=K
To:   From:
CO    CO      0.000  1.000      NA     NA     NA
      pulse  -1.977  1.061   6.342  0.142  0.000
pulse CO      1.864  0.943   5.979  0.142  0.000
      pulse   0.000  1.000      NA     NA     NA
```

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

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

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

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

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

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

No. items with measurements on each method:
        #Replicates
Method    1   2   3 #Items #Obs: 354 Values:  min  med  max
  CO      1   4  56     61       177         22.2 78.6 93.5
  pulse   1   4  56     61       177         24.0 75.0 94.0

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

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4: Sampling has been
```

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

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

 Conversion between methods:
             alpha   beta     sd
To:   From:
CO    CO      0.000  1.000  3.128
      pulse   0.912  1.000  4.631
pulse CO     -0.912  1.000  4.631
      pulse   0.000  1.000  5.705

 Variance components (sd):
       s.d.
```

```
Method      IxR     MxI   res
  CO    119.332 115.383 2.212
  pulse 119.332 115.383 4.034

Variance components with 95 % cred.int.:
    method      CO                   pulse
    qnt        50%    2.5%   97.5%     50%    2.5%   97.5%
SD
IxR        119.332   2.955 557.698 119.332   2.955 557.698
MxI        115.383   2.335 559.671 115.383   2.335 559.671
res          2.212   0.540   4.699   4.034   0.370   4.958
tot        171.723   4.577 776.595 172.290   5.602 776.587

Mean parameters with 95 % cred.int.:
                   50%    2.5%  97.5% P(>0/1)
alpha[pulse.CO] -0.912  -3.533 44.966   0.493
alpha[CO.pulse]  0.912 -44.966  3.533   0.507
beta[pulse.CO]   1.000   1.000  1.000   0.000
beta[CO.pulse]   1.000   1.000  1.000   0.000

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

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

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

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

 Conversion between methods:
             alpha   beta     sd
To:   From:
CO    CO      0.000  1.000  3.128
      pulse   0.912  1.000  4.631
pulse CO     -0.912  1.000  4.631
      pulse   0.000  1.000  5.705

 Variance components (sd):
       s.d.
Method      IxR     MxI   res
  CO    119.332 115.383 2.212
  pulse 119.332 115.383 4.034
```

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

```
> plot( ox.mi.ir, 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( ox.mi.ir ) )
```

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

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

```
> ox.lin <- 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,
(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: Sampling has been
```

13. In order to be reasonably sure about the validity of inference based on the mcmc-estimates 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( ox.lin ) )
```

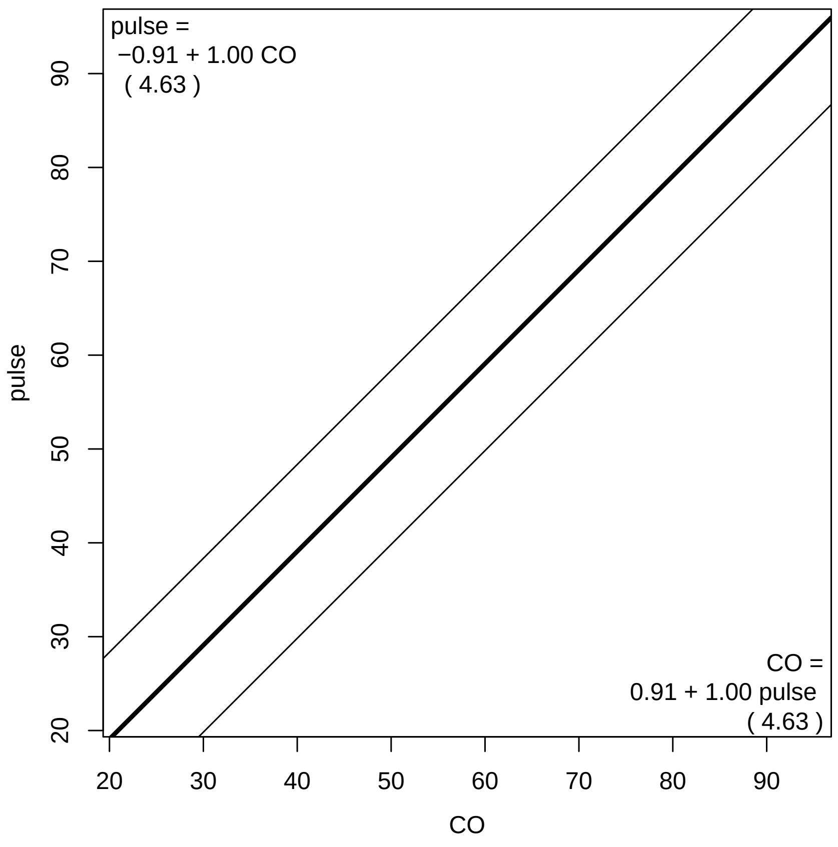


Figure 4.23: *A scatterplot of the oximetry data with the linear equations displayed. The slope of the linear relationship between methods has been constrained to 1.00.*

14. Once we have established that the mixing of the chains is satisfactory, and hence that we 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( ox.lin )

 Conversion between methods:
              alpha    beta      sd
To:   From:
CO    CO      0.000   1.000   2.462
      pulse -10.151   1.170   5.345
pulse CO      8.672   0.854   4.516
      pulse   0.000   1.000   5.977

 Variance components (sd):
       s.d.
Method    IxR   MxI   res
  CO    3.718 3.201 1.741
  pulse 3.141 2.800 4.226

Variance components with 95 % cred.int.:
    method    CO                pulse
    qnt      50%  2.5% 97.5%   50%  2.5% 97.5%
SD
IxR        3.718 3.059 5.472 3.141 2.659 4.142
MxI        3.201 2.152 7.725 2.800 1.900 5.894
res        1.741 0.895 2.835 4.226 3.594 4.975
tot        5.311 4.505 9.511 6.025 5.382 8.322

Mean parameters with 95 % cred.int.:
```

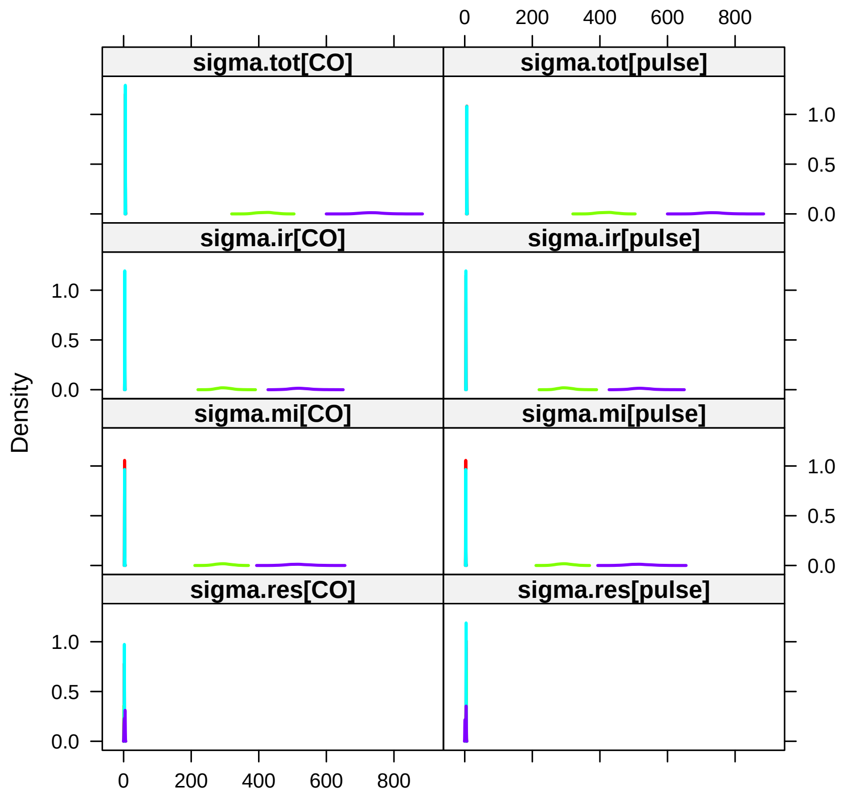


Figure 4.24: *Smoothed density plots of the variance components estimated using* `MethComp`.

```
                    50%    2.5%  97.5% P(>0/1)
alpha[pulse.CO]    8.659   2.261 18.920       1
alpha[CO.pulse] -10.167 -25.430 -2.423       0
beta[pulse.CO]     0.854   0.733  0.935       0
beta[CO.pulse]     1.170   1.069  1.365       1

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

> ox.lin$summary

NULL

> MethComp( ox.lin )

 Conversion between methods:
              alpha    beta      sd
To:   From:
CO    CO      0.000   1.000   2.462
      pulse -10.151   1.170   5.345
pulse CO      8.672   0.854   4.516
      pulse   0.000   1.000   5.977

 Variance components (sd):
      s.d.
Method    IxR   MxI   res
  CO    3.718 3.201 1.741
  pulse 3.141 2.800 4.226
```

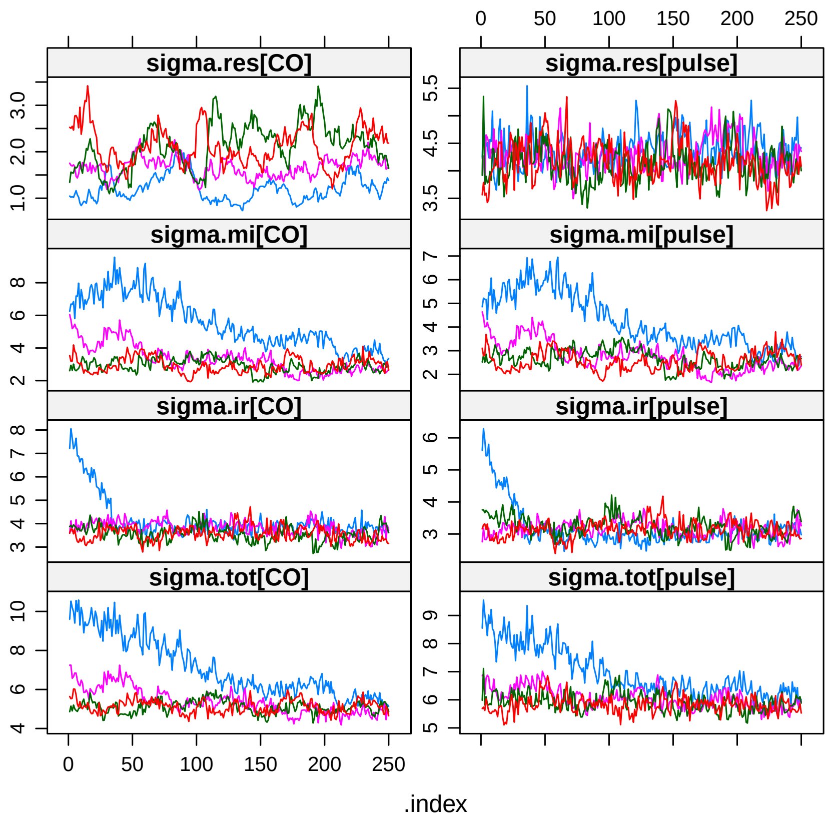


Figure 4.25: *Traces of the chains for the variance components estimated using* `MCmcmc`*.*

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 differce 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( ox.lin, pl.obs = TRUE )
```

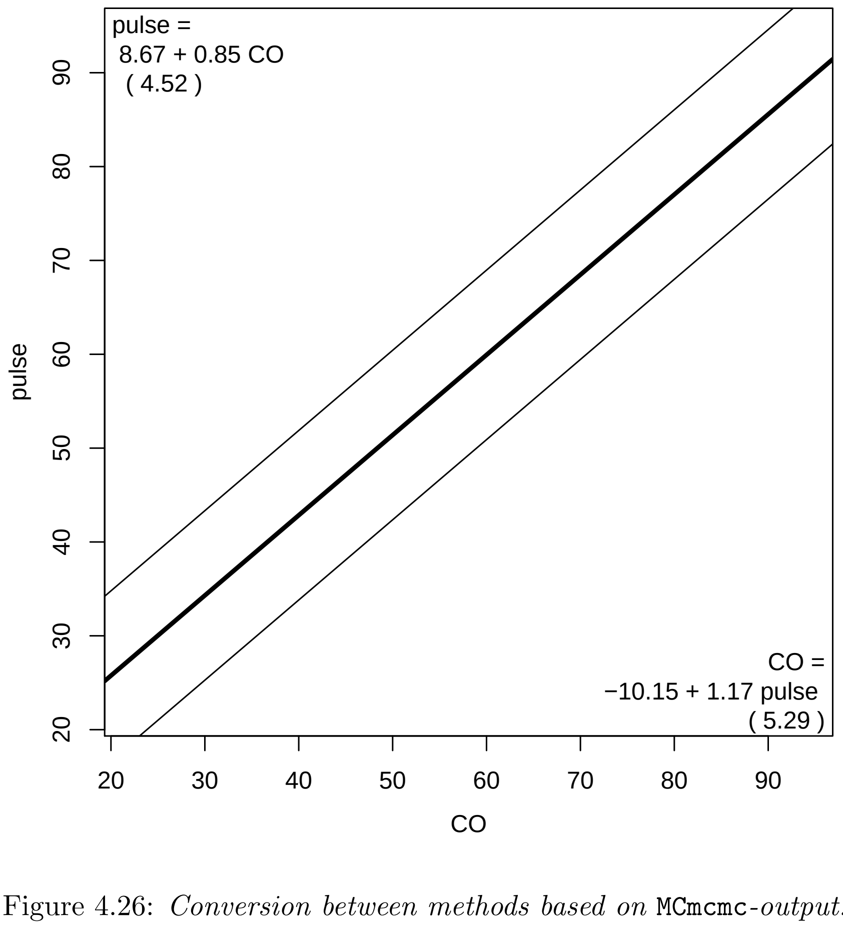


Figure 4.26: *Conversion between methods based on* `MCmcmc`*-output.*

4.7 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 )

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

> plot( ox )

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:

```
> DA.reg( oxt )

 Conversion between methods:
             alpha   beta sd.pred beta=1 sd.|A=1.21 slope(sd)  sd.=K
To:   From:
CO    CO      0.000  1.000      NA     NA         NA        NA     NA
      pulse   0.038  1.111   0.340  0.009      0.303    -0.038  0.246
pulse CO     -0.034  0.900   0.306  0.009      0.303    -0.038  0.246
      pulse   0.000  1.000      NA     NA         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:

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

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

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

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

[1] 0.8552208 0.4445571 1.6452388
```

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

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

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

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


 Conversion between methods:
                 alpha   beta sd.pred beta=1 sd.|A=77 slope(sd)  sd.=K
To:   From:
CO    CO         0.000  1.000      NA     NA       NA        NA     NA
      pulse      0.038  1.111   0.340  0.009    0.303    -0.038  0.246
pulse CO        -0.034  0.900   0.306  0.009    0.303    -0.038  0.246
      pulse      0.000  1.000      NA     NA       NA        NA     NA

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

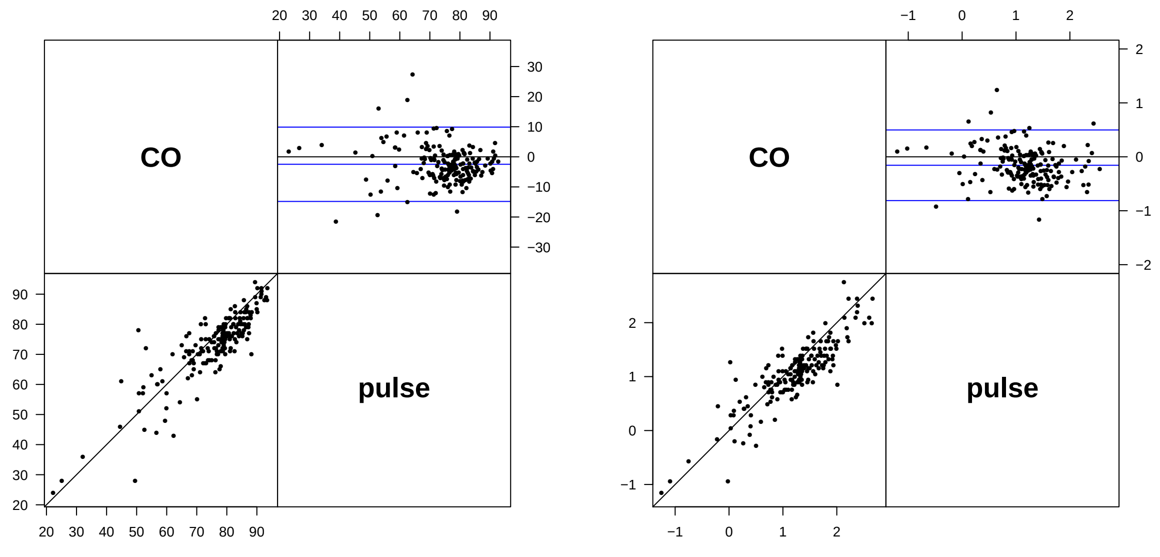


Figure 4.27: *Original (left) and* logit*-transformed oximetry data. Clearly, the logit-transform removes the tendency to diminishing variance at the upper end of the measurements, whereas the outliers in the middle of the scale have not been remedied..*

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

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

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

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

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

Note:
 Replicate measurements are taken as separate items!

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

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

Note:
 Replicate measurements are taken as separate items!

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

The resulting plot is shown in figure 4.28

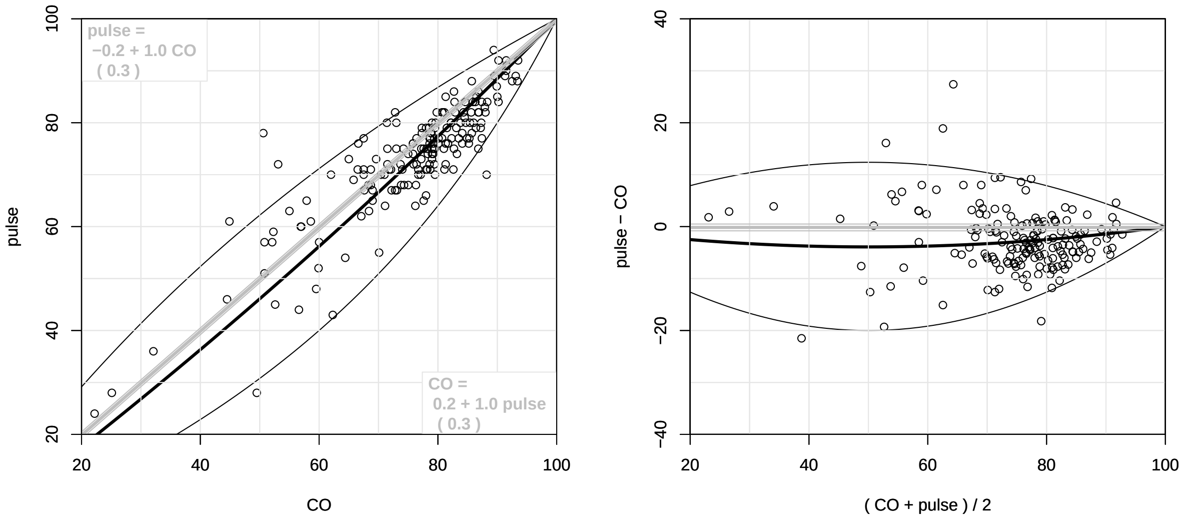


Figure 4.28: *Prediction between pulse and CO-oximetry assuming a constant difference on the* logit *scale. The limits using the original scale are shown too in light gray.*

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

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

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

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

Note:
 Replicate measurements are taken as separate items!

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

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

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

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

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


 Conversion between methods:
             alpha   beta sd.pred beta=1 sd.|A=77 slope(sd)  sd.=K
To:   From:
CO    CO     0.000  1.000      NA     NA       NA        NA     NA
      pulse  0.038  1.111   0.340  0.009    0.303    -0.038  0.246
pulse CO    -0.034  0.900   0.306  0.009    0.303    -0.038  0.246
      pulse  0.000  1.000      NA     NA       NA        NA     NA
```

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

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

```
> system.time(
+ MCoxl <- MCmcmc( ox, bias="lin", random=c("mi","ir"), n.iter=50000,
+                  Trans="pctlogit" ) )
```

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

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

Simulation run of a model with
- method by item and item by replicate interaction:
- using 4 chains run for 50000 iterations
  (of which 25000 are burn-in),
- monitoring every 25 values of the chain:
- giving a posterior sample of 4000 observations.

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4: Sampling has been
   user  system elapsed
 572.47    0.19  573.72


> MethComp( MCoxl )


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


 Conversion between methods:
             alpha   beta     sd
To:   From:
CO    CO      0.000  1.000  0.171
      pulse  -0.010  1.156  0.267
pulse CO      0.008  0.865  0.231
      pulse   0.000  1.000  0.290

 Variance components (sd):
       s.d.
Method    IxR   MxI   res
```

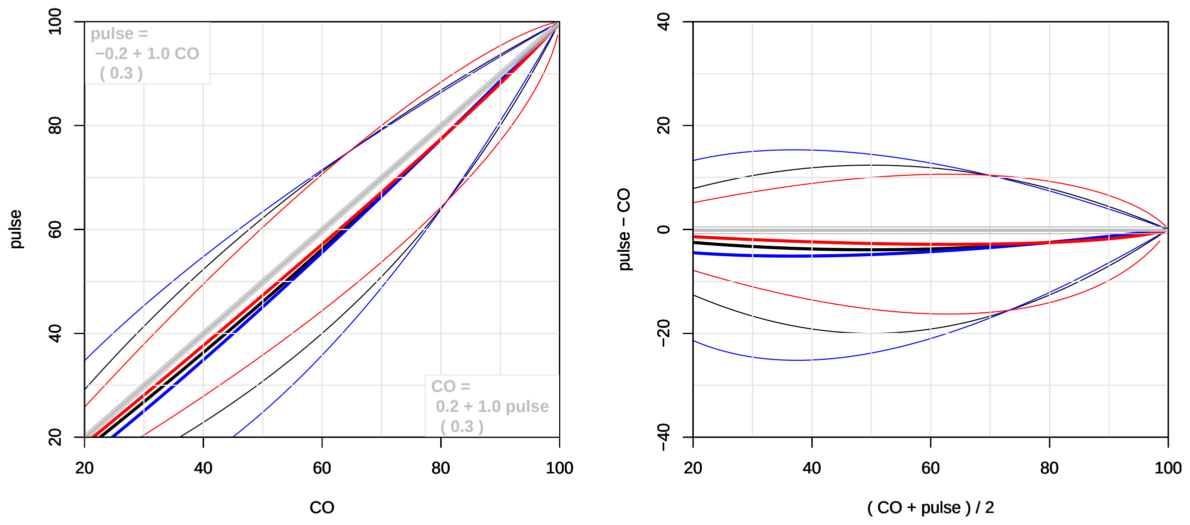


Figure 4.29: *Prediction between pulse and CO-oximetry assuming a constant difference on the* logit *scale. The red lines are limits based on the complementary log–log transform, and the blue lines the log–log transform. The limits using the original scale are shown too in light gray.*

```
  CO    0.258 0.180 0.121
  pulse 0.224 0.156 0.205

> ARoxl <- AltReg( ox, linked=TRUE, Trans="pctlogit", trace=TRUE )

iteration 1 criterion: 1
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO     0.003 0.998 0.098          1.151 1.151     1.000 0.994 0.220 0.197 0.161
pulse -0.003 1.003 0.098          1.151 1.151     1.006 1.000 0.222 0.198 0.178

iteration 2 criterion: 0.08547255
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.024 1.032 0.100          1.151 1.181     1.000 1.013 0.222 0.185 0.158
pulse -0.039 1.019 0.121          1.121 1.151     0.987 1.000 0.220 0.182 0.181

iteration 3 criterion: 0.0732349
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.054 1.068 0.097          1.151 1.209      1.00 1.031 0.224 0.175 0.155
pulse -0.075 1.036 0.125          1.094 1.151      0.97 1.000 0.218 0.170 0.183

iteration 4 criterion: 0.05672292
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.087 1.104 0.094          1.151 1.234     1.000 1.047 0.226 0.168 0.153
pulse -0.111 1.055 0.129          1.071 1.151     0.955 1.000 0.216 0.161 0.185

iteration 5 criterion: 0.03987535
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.121 1.140 0.092          1.151 1.255     1.000 1.061 0.228 0.164 0.150
pulse -0.146 1.075 0.133          1.052 1.151     0.942 1.000 0.215 0.155 0.187

iteration 6 criterion: 0.02601184
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.157 1.176 0.089          1.151 1.272     1.000 1.073 0.229 0.162 0.149
pulse -0.181 1.096 0.136          1.038 1.151     0.932 1.000 0.213 0.151 0.188

iteration 7 criterion: 0.01624239
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.194 1.211 0.087          1.151 1.284     1.000 1.082 0.230 0.161 0.147
pulse -0.216 1.120 0.139          1.027 1.151     0.925 1.000 0.213 0.148 0.189

iteration 8 criterion: 0.009992423
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.233 1.247 0.086          1.151 1.293     1.000 1.089 0.231 0.160 0.146
pulse -0.251 1.145 0.140          1.020 1.151     0.919 1.000 0.212 0.147 0.190

iteration 9 criterion: 0.006183976
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.272 1.282 0.084          1.151 1.300     1.000 1.094 0.231 0.160 0.145
pulse -0.286 1.172 0.142          1.014 1.151     0.914 1.000 0.211 0.146 0.190

iteration 10 criterion: 0.004311325
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.312 1.318 0.084          1.151 1.304     1.000 1.097 0.232 0.160 0.144
pulse -0.322 1.201 0.142          1.011 1.151     0.911 1.000 0.211 0.145 0.191

iteration 11 criterion: 0.003151143
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.353 1.354 0.083          1.151 1.308     1.000   1.1 0.232 0.160 0.144
pulse -0.359 1.231 0.143          1.008 1.151     0.909   1.0 0.211 0.145 0.191

iteration 12 criterion: 0.002286339
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.395 1.391 0.082          1.151 1.310     1.000 1.102 0.232 0.160 0.144
pulse -0.397 1.262 0.144          1.006 1.151     0.907 1.000 0.211 0.145 0.191

iteration 13 criterion: 0.001650499
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.439 1.428 0.082          1.151 1.312     1.000 1.103 0.232 0.160 0.143
pulse -0.436 1.294 0.144          1.005 1.151     0.906 1.000 0.210 0.145 0.191

iteration 14 criterion: 0.001187758
```

```
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.483 1.466 0.082         1.151 1.313     1.000 1.104 0.232 0.160 0.143
pulse -0.475 1.328 0.144         1.004 1.151     0.905 1.000 0.210 0.145 0.191

iteration 15 criterion: 0.0008526646
       alpha  beta sigma Intercept: CO pulse Slope: CO pulse   IxR   MxI   res
CO    -0.528 1.506 0.082         1.151 1.314     1.000 1.105 0.232 0.160 0.143
pulse -0.516 1.362 0.144         1.003 1.151     0.905 1.000 0.210 0.145 0.191

AltReg converged after  15 iterations
Last convergence criterion was  0.0008526646
```

```
> MethComp( ARoxl )

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


 Conversion between methods:
             alpha   beta     sd
To:   From:
CO    CO      0.000  1.000  0.202
      pulse   0.042  1.105  0.341
pulse CO     -0.038  0.905  0.309
      pulse   0.000  1.000  0.271

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

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

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

```
> plot( BAoxl, pl.type="comp", points=TRUE, pch=16,
+       axlim=c(20,100), diflim=c(-40,40), xaxs="i", yaxs="i" )

Note:
 Replicate measurements are taken as separate items!

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

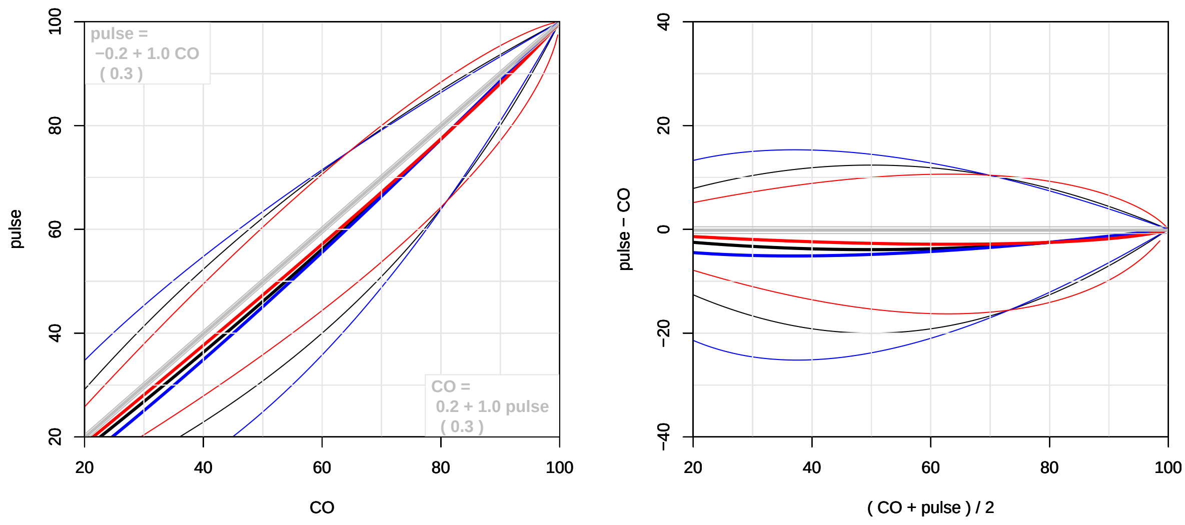


Figure 4.30: *Prediction between pulse and CO-oximetry, The black line assumes constant bias on the logit scale, the red (`MCmcmc`) and the blue (`AltReg`) allows a linear relationship on that scale.*

4.8 Systolic blood pressure: Linked replicates by three methods

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

1. First we load the package and then the systolic blood pressure data from the `MethComp` package, and then take a look at the data using `str()`:

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

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

Since the columns have the right names you can easily turn the data-frame into a `Meth` object:

```
> sbp <- Meth( sbp )

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

> str( sbp )

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

> plot(sbp,var.names=T)

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

Clearly, the automated method seems to be considerably less accurate then the two manual measurements; at least the two manual measurements are in closer agreement than with the automated.

2. We can formally assess the differences between methods, and allocate the sources of variation between the methods using the `BA.est` function; remember to specify thet the replicates are linked across the methods:

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

 Conversion between methods:
            alpha    beta      sd   LoA: lower   upper
To: From:
J   J       0.000   1.000   2.305       -4.610   4.610
    R       0.086   1.000   2.272       -4.459   4.631
```

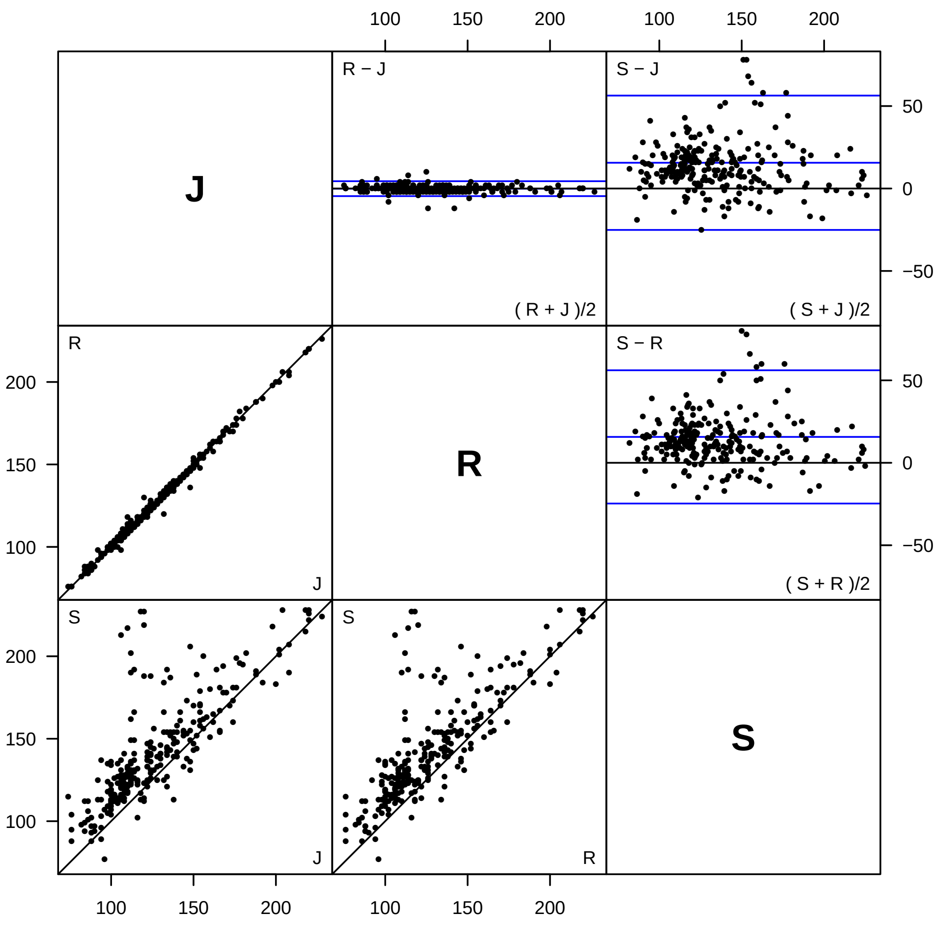


Figure 4.31: *scatter-plots and Bland-Altman plots for three methods of measuring systolic blood pressure. J and R are clinicians, S is an automated method.*

```
    S      -15.620   1.000  20.326      -56.272  25.032
R   J       -0.086   1.000   2.272       -4.631   4.459
    R        0.000   1.000   2.187       -4.375   4.375
    S      -15.706   1.000  20.317      -56.339  24.927
S   J       15.620   1.000  20.326      -25.032  56.272
    R       15.706   1.000  20.317      -24.927  56.339
    S        0.000   1.000  12.930      -25.860  25.860

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

We see in the variance components that method "S" is the one with the largest measurement error, but also the one where the M×I variation is the largest. Whereas the residual variation is estimated from the differences between the replicates with the same method, and therefore represent the repeatability, the matrix-effect M×I assesses how the mean of the measurements on a particular patient varies between methods. So this variance component is influenced by how many closely agreeing methods are in the comparison. Hence it is not surprising that this is virtually 0 for the two closely agreeing methods.

We can also see that variation between measurement occasions is quite large, the s.d. is around 6 mmHg, almost in the same order of magnitude as the s.d. of method S.

3. Two assumptions behind the model fitted by `BA.est` are that the differences between methods are constant, and that the variance of the differences is the same across the range of measurements. These assumptions can be assessed approximately by regressing differences between methods on averages. The function `DA.reg` does this for all pairs of methods, and reports the pairwise results. This is of course only a partial assessment of the assumptions. The replicates are linked, so we should *not* apply a random permutation of replicates:

```
> DA.reg( sbp )

 Conversion between methods:
           alpha   beta sd.pred beta=1 s.d.=K
To: From:
J   J      0.000  1.000      NA     NA     NA
    R     -1.076  1.009   2.258  0.048  0.486
    S     -8.448  0.950  19.839  0.226  0.000
R   J      1.066  0.991   2.237  0.048  0.486
    R      0.000  1.000      NA     NA     NA
    S     -7.178  0.940  19.559  0.146  0.000
S   J      8.894  1.053  20.886  0.226  0.000
    R      7.633  1.063  20.800  0.146  0.000
    S      0.000  1.000      NA     NA     NA
```

We see that the hypothesis of constant difference between methods is rejected for the pairing of R and J but not very convincingly. So there is little evidence for the non-constancy of differences. On the other hand for all comparisons with method S we see that there is evidence that the variance is non-constant; this is also apparent from the plot in figure 4.33.

4. We can explore whether a log-transformation would alleviate some of these problems; this would correspond to assuming that the relative blood.pressure were of interest, and using the coefficient of variation as the relevant measure of variation. This is easily accomplished using the `Trans=` argument to `DA.reg`:

```
> DA.reg( sbp, Trans="log" )

Note: Response transformed by:


 Conversion between methods:
           alpha   beta sd.pred beta=1 s.d.=K
To: From:
J   J      0.000  1.000      NA     NA     NA
    R     -0.046  1.010   0.019  0.060  0.068
    S     -0.387  1.054   0.142  0.195  0.028
R   J      0.046  0.990   0.018  0.060  0.068
    R      0.000  1.000      NA     NA     NA
    S     -0.332  1.043   0.140  0.298  0.031
S   J      0.367  0.948   0.135  0.195  0.028
    R      0.319  0.959   0.134  0.298  0.031
    S      0.000  1.000      NA     NA     NA
```

This actually makes things worse; the p-values for constant variances are all almost significant. So we stick to the original scale, even if the constant variance assumptions is violated, so the conclusions w.r.t. method S should be cautious.

5. In order to get a handle on the uncertainty of the variance components, we fit a model with `MCmcmc` using 25,000 iterations:

```
> system.time(
+ MCsbp <- MCmcmc( sbp, bias="lin", linked=TRUE, n.iter=25000 ) )
```

```
Comparison of 3 methods, using 765 measurements
on 85 items, with up to 3 replicate measurements,
(replicate values are in the set: 1 2 3 )
( 3 * 85 * 3 = 765 ):

No. items with measurements on each method:
       #Replicates
Method          3 #Items #Obs: 765 Values:  min med max
     J         85     85       255           74 120 228
     R         85     85       255           76 120 226
     S         85     85       255           77 135 228

Simulation run of a model with
- method by item and item by replicate interaction:
- using 4 chains run for 25000 iterations
  (of which 12500 are burn-in),
- monitoring every 13 values of the chain:
- giving a posterior sample of 3846 observations.

Initializing chain 1: Initializing chain 2: Initializing chain 3: Initializing chain 4: Sampling has been
   user  system elapsed
 675.27    0.98  683.19

> MCsbp

 Conversion between methods:
            alpha    beta      sd
To: From:
J   J        0.000   1.000   2.182
    R       -1.092   1.009   2.292
    S      -51.474   1.254  25.086
R   J        1.082   0.991   2.272
    R        0.000   1.000   2.377
    S      -49.908   1.242  24.845
S   J       41.059   0.798  19.995
    R       40.190   0.805  20.002
    S        0.000   1.000  28.222

 Variance components (sd):
     s.d.
Method   IxR    MxI   res
     J 5.996  0.238 1.503
     R 5.943  0.348 1.622
     S 4.786 17.851 8.937

Variance components with 95 % cred.int.:
    method      J                       R                       S
    qnt       50%   2.5%  97.5%    50%   2.5%  97.5%    50%   2.5%  97.5%
SD
IxR         5.996  5.386  6.734  5.943  5.330  6.670  4.786  4.008  5.698
MxI         0.238  0.004  0.782  0.348  0.025  0.866 17.851 15.209 21.185
res         1.503  0.831  2.055  1.622  0.921  2.077  8.937  8.042  9.992
tot         6.199  5.592  6.931  6.176  5.572  6.891 20.537 18.284 23.559

Mean parameters with 95 % cred.int.:
                50%    2.5%   97.5% P(>0/1)
alpha[R.J]   1.082  -0.240   2.316   0.952
alpha[S.J]  41.065  26.468  54.102   1.000
alpha[J.R]  -1.092  -2.360   0.239   0.048
alpha[S.R]  40.151  25.325  53.359   1.000
alpha[J.S] -51.466 -77.474 -29.229   0.000
alpha[R.S] -49.956 -75.860 -27.746   0.000
beta[R.J]    0.991   0.981   1.001   0.035
beta[S.J]    0.798   0.698   0.908   0.000
beta[J.R]    1.009   0.999   1.019   0.965
beta[S.R]    0.805   0.703   0.917   0.000
beta[J.S]    1.254   1.101   1.432   1.000
beta[R.S]    1.242   1.091   1.422   1.000

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

6. The results from the `MCmcmc` function is an `MCmcmc` object which basically contains the posterior samples for each of the parameters. The results can be summarized in the same form as summarize from `BA.est`, `DA.reg` and `AltReg`, by converting them to a `MethComp` object, that gives the conversion equations between methods, the prediction s.d.s and the variance components (but not the uncertainty of the variance components):

```
> MethComp( MCsbp )

 Conversion between methods:
            alpha    beta      sd
To: From:
J   J       0.000   1.000   2.182
    R      -1.092   1.009   2.292
    S     -51.474   1.254  25.086
R   J       1.082   0.991   2.272
    R       0.000   1.000   2.377
    S     -49.908   1.242  24.845
S   J      41.059   0.798  19.995
    R      40.190   0.805  20.002
    S       0.000   1.000  28.222

 Variance components (sd):
      s.d.
Method   IxR    MxI   res
     J 5.996  0.238 1.503
     R 5.943  0.348 1.622
     S 4.786 17.851 8.937
```

We see that the residual ("pure" measurement error) as well as the between patients-variation within method (`MxI`) are substantially larger for method S. Also we see that the variation between replicates (which is common for the methods) is quite large, almost 6 mmHg in s.d.

7. We can check the convergence of the chains by using `trace.MCmcmc` to show how the sample paths for each of the parameters; the default is to show the traces for the variance components:

```
> trace.MCmcmc( MCsbp )
```

We can explicitly ask for the trace for the intercept and slope parameters:

```
> trace.MCmcmc( MCsbp, "int" )

> trace.MCmcmc( MCsbp, "slope" )
```

These have bee put side-by side in figure **??**

8. For comparison, we can also estimate in the same model using the `AltReg` function. Note that the default for the `AltReg` function is to assume that the method*times*item random effects have the same s.d. for all methods, which is not the model we are interested in, so we must use the argument `varMxI=TRUE`.

```
> ARsbp <- AltReg( sbp, linked=TRUE, varMxI=TRUE, maxiter=50 )

AltReg converged after  35 iterations
Last convergence criterion was  0.0009529158

> ARsbp
```

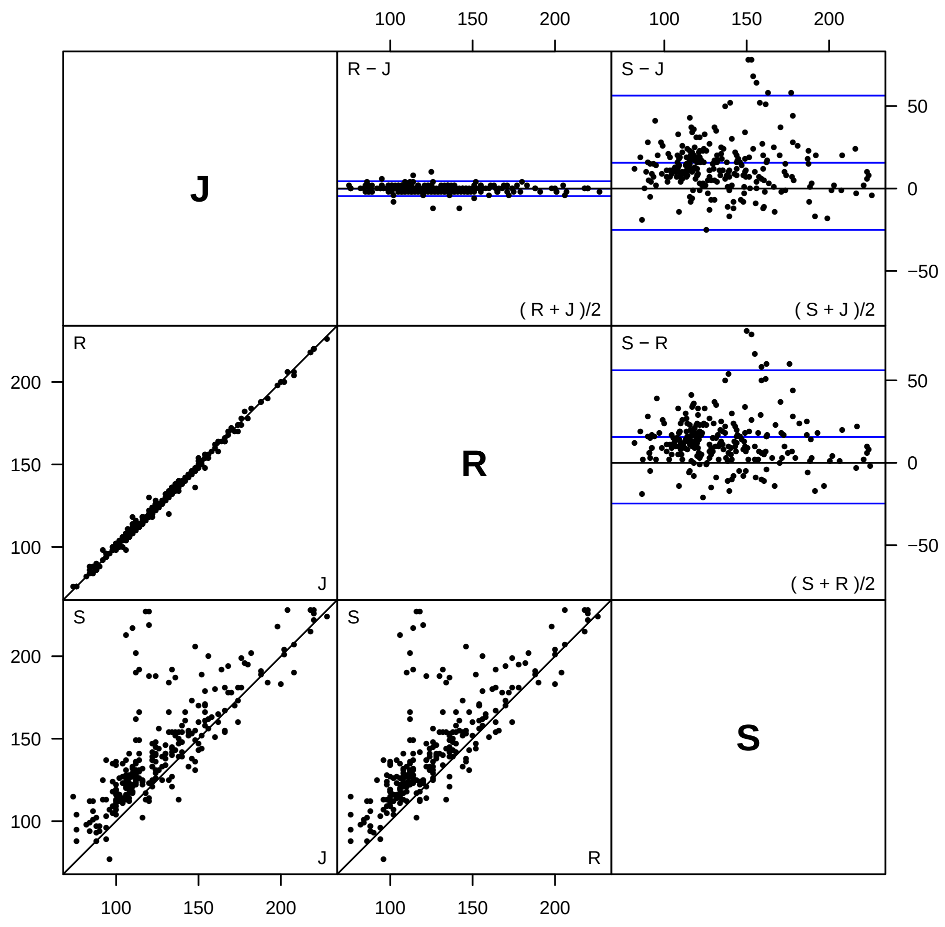


Figure 4.32: *scatter-plots and Bland-Altman plots for three methods of measuring systolic blood pressure. J and R are clinicians, S is an automated method.*

```
 Conversion between methods:
             alpha    beta      sd
To: From:
J   J        0.000   1.000   2.204
    R       -1.091   1.009   2.261
    S      -12.253   0.986  20.525
R   J        1.081   0.991   2.241
    R        0.000   1.000   2.296
    S      -11.060   0.977  20.343
S   J       12.425   1.014  20.812
    R       11.318   1.023  20.819
    S        0.000   1.000  12.984

 Variance components (sd):
        s.d.
Method   IxR    MxI   res
     J 5.908  0.003 1.559
     R 5.854  0.001 1.623
     S 5.990 18.611 9.181
```

We used the `AltReg` function with the `trace=FALSE` (the default) argument to save space here, but you would presumably want to use `trace=TRUE` it if you run this on your screen).

We see that we get approximately the same estimates for the variance components, but the estimates for the intercepts and slopes for the conversions are not the same.

9. Finally, when you feel confident about the results from the MCMC simulation you can plot the estimated conversions between the methods. To that end you first convert the `MCmcmc` object to a `MethComp` object and then use `plot` (which means you use the function `plot.MethComp`. At the same time we can plot the corresponding results from the `AltReg` analyses (which by default have class `MethComp`.

```
> par( mfrow=c(2,3), mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( MethComp(MCsbp), 1:2   ,points=T, axlim=c(50,250), pch=16, digits=2 )
> plot( MethComp(MCsbp), 3:2   ,points=T, axlim=c(50,250), pch=16, digits=2 )
> plot( MethComp(MCsbp), c(1,3),points=T, axlim=c(50,250), pch=16, digits=2 )
> plot(          ARsbp , 1:2   ,points=T, axlim=c(50,250), pch=16, digits=2 )
> plot(          ARsbp , 3:2   ,points=T, axlim=c(50,250), pch=16, digits=2 )
> plot(          ARsbp , c(1,3),points=T, axlim=c(50,250), pch=16, digits=2 )
```

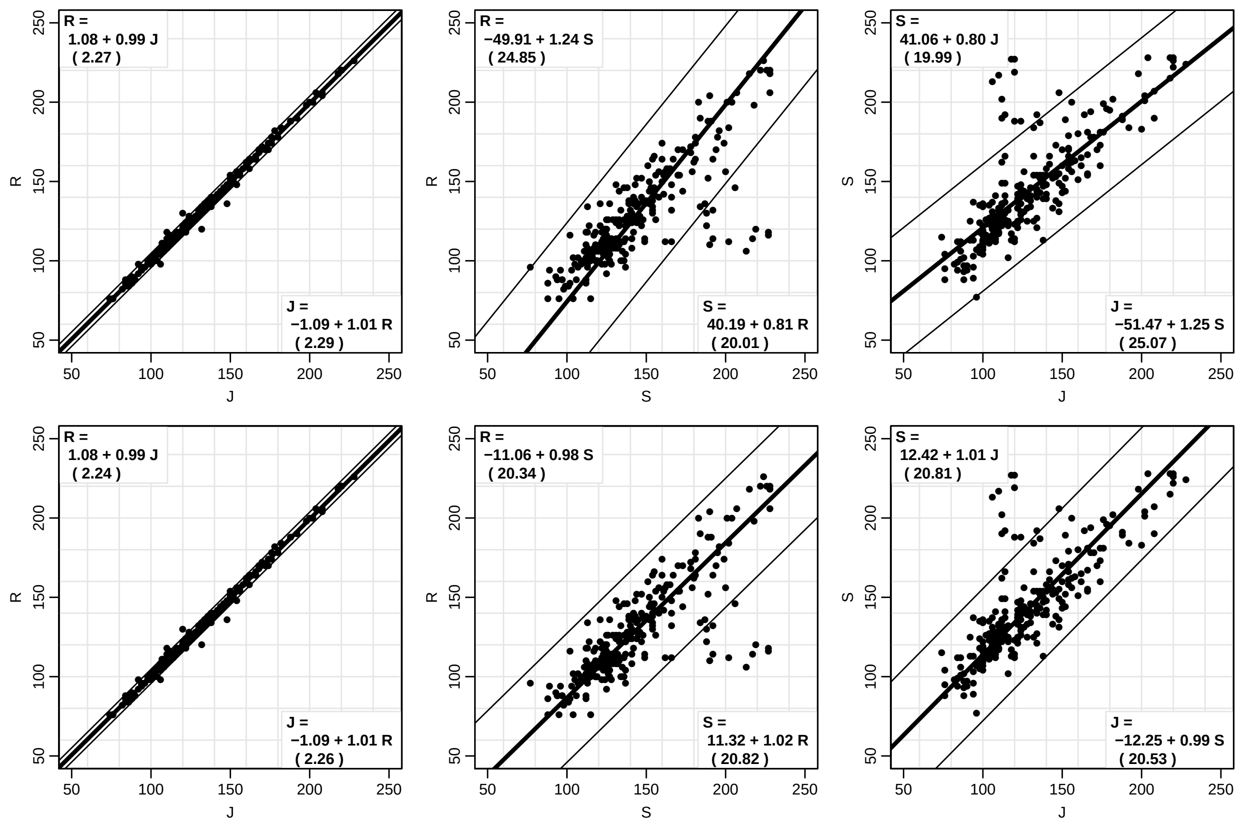


Figure 4.33: *Conversion plots between pairs of methods, based on the MCMC modeling using* `MCmcmc` *(upper panels), and alternating regressions using* `AltReg` *(lower panels).*

From figure 4.33 we see that the predictions are different, but given the quite wide limits when method S is involved, there is no substantial difference between the two approaches.

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 1.3

Date 2011-02-03

Title Functions for analysis of method comparison studies.

Author Bendix Carstensen, Lyle Gurrin.

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

Depends R (>= 2.0.0), nlme

Suggests R2WinBUGS, coda, BRugs, lattice

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

License GPL (>= 2)

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

R topics documented:

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

Description

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

Usage

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

Arguments

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

Value

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

Author(s)

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

References

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

See Also

BA.plot, MCmcmc

Examples

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

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

Description

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

Usage

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

Arguments

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

`trace`	Should a trace of the iterations be printed? If `TRUE` iteration number, convergence criterion and current estimates of means and sds are printed.
`sd.lim`	Estimated standard deviations below `sd.lim` are disregarded in the evaluation of convergence. See details.
`Transform`	A character string, or a list of two functions, each other's inverse. The measurements are transformed by this before analysis. Possibilities are: "exp", "log", "logit", "pctlogit" (transforms percentages by the logit), "sqrt", "sq" (square), "cll" (complementary log-minus-log), "ll" (log-minus-log). For further details see `choose.trans`.
`trans.tol`	The tolerance used to check whether the supplied transformation and its inverse combine to the identity. Only used if `Transform` is a list of two functions.

Details

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

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

The model fitted is formulated as:

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

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

Value

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

`Conv`	A 3-way array with the 2 first dimensions named "To:" and "From:", with methods as levels. The third dimension is classifed by the linear parameters "alpha", "beta", and "sd".
`VarComp`	A matrix with methods as rows and variance components as columns. Entries are the estimated standard deviations.
`data`	The original data used in the analysis, with untransformed measurements (ys). This is needed for plotting purposes.

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

Author(s)

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

References

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

See Also

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

Examples

```
data( ox )
ox <- Meth( ox )
ox.AR <- AltReg( ox, linked=TRUE, trace=TRUE, Transform="pctlogit" )
str( ox.AR )
```

```
ox.AR
# plot the resulting conversion between methods
plot(ox.AR,pl.type="conv",axlim=c(20,100),points=TRUE,xaxs="i",yaxs="i",pch=16)
# - or the rotated plot
plot(ox.AR,pl.type="BA",axlim=c(20,100),points=TRUE,xaxs="i",yaxs="i",pch=16)
```

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

Description

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

Usage

```
  BA.est( data, linked=TRUE, IxR=has.repl(data),
                             MxI=has.repl(data),
                          varMxI=TRUE,
                          IxR.pr=FALSE,
                            bias=TRUE, alpha=0.05,
                Transform = NULL,
                trans.tol = 1e-6 )
  ## S3 method for class 'BA.est'
bias( obj, ref=1, ... )
  VC.est( data,
           IxR = has.repl(data), linked = IxR,
           MxI = has.repl(data), matrix = MxI,
        varMxI = TRUE, bias = TRUE, print = FALSE )
```

Arguments

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

`ref`	Numeric or character. The reference method for the biases: the method with bias 0.
`print`	Logical. Should the estimated bias and variance components be printed?
`...`	Further argumenst passed on. Curently ignored.

Details

The model fitted is:

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

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

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

Value

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

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

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

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

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

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

times the sd.

`Mu` Estimates of the item-specific parameters.

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

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

Author(s)

Bendix Carstensen

References

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

See Also

`BA.plot`, `perm.repl`

Examples

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

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

Description

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

Usage

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

Arguments

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

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

Value

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

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

Author(s)

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

References

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

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

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

See Also

BA.plot, MCmcmc.

Examples

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

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

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

`bothlines` *Add regression lines to a plot*

Description

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

Usage

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

Arguments

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

Value

None.

Author(s)

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

See Also

abline.

Examples

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

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

Description

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

Usage

```
data(cardiac)
```

Format

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

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

`item` a numeric vector giving the item number.

`repl` a numeric vector with replicate number.

`y` the measuremnts of cardiac output.

Details

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

Source

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

Examples

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

`CardOutput` *Measurements of Cardiac output.*

Description

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

Usage

```
data(CardOutput)
```

Format

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

`Age` Patient age

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

`VO2` Oxygen consumption

`Svo2` Mixed venous O2 saturation

`Scvo2` Central venous oxygen saturation

`TCO` Thermodilution-derived cardiac output

`FCO` Fick-derived cardiac output.

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

Source

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

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

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

Examples

```
data(CardOutput)
```

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

Description

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

Usage

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




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

Arguments

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

`pch`	Plot symbol for the points.
`subset`	Character or numerical indicating the columns of the posterior that should be plotted by `pairs`.
`cex`	Plot character size for points in `pairs`.
`...`	Further aruments passed on to the `Lattice` function called: `trace` calls `xyplot` from the `coda` package, `post` calls `densityplot` from the `coda` package, `pairs` calls `pairs` from the `graphics` package.

Details

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

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

Value

A `Lattice` plot.

Author(s)

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

See Also

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

Examples

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

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

Description

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

Usage

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

Arguments

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

Value

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

`check.trans` returns nothing.

Author(s)

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

Examples

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

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

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

References

Shortly...

See Also

MCmcmc.

Examples

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

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

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

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

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

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

Author(s)

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

References

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

Examples

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

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

Description

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

Usage

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

Arguments

`x`	numerical variable.
`y`	numerical variable.
`vr`	The assumed known ratio of the (residual) variance of the `y`s relative to that of the `x`s. Defaults to 1.
`sdr`	do. for standard deviations. Defaults to 1. `vr` takes precedence if both are given.

`boot`	Should bootstrap estimates of standard errors of parameters be done? If `boot==TRUE`, 1000 bootstrap samples are done, if `boot` is numeric, `boot` samples are made.
`keep.boot`	Should the 4-column matrix of bootstrap samples be returned? If `TRUE`, the summary is printed, but the matrix is returned invisibly. Ignored if `boot=FALSE`
`alpha`	What significance level should be used when displaying confidence intervals?

Details

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

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

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

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

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

Value

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

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

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

Author(s)

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

References

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

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

See Also

`MCmcmc`

Examples

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

`Enzyme` *Enzyme activity data*

Description

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

Usage

```
data(Enzyme)
```

Format

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

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

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

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

Source

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

Examples

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

`fat` *Measurements of subcutaneous and visceral fat*

Description

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

Usage

```
data(fat)
```

Format

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

`Id` Person id.

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

`Rep` Replicate — exchangeable within person and observer.

`Sub` Subcutaneous fat measured in cm.

`Vic` Visceral fat measured in cm.

Examples

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

`glucose` *Glucose measurements by different methods*

Description

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

Usage

```
data(glucose)
```

Format

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

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

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

`item` Person id.

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

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

`y` Glucose measurement in mmol/l.

Source

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

References

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

Examples

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

---

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

Description

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

Usage

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

Format

The format is a `MCmcmc` object.

Details

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

Examples

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

---

hba1c *Measurements of HbA1c from Steno Diabetes Center*

Description

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

Usage

```
data(hba1c)
```

Format

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

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

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

`item` Person-id. A numeric vector

`d.samp` Day of sampling.

`d.ana` Day of laboratory analysis.

`y` The measured value of HbA1c.

Details

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

Source

Bendix Carstensen, Steno Diabetes Center.

References

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

Examples

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

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

Description

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

Usage

```
MCmcmc( data,
        bias = "linear",
         IxR = has.repl(data), linked = IxR,
         MxI = TRUE,           matrix = MxI,
       varMxI = nlevels(factor(data$meth)) > 2,
     n.chains = 4,
       n.iter = 2000,
     n.burnin = n.iter/2,
        n.thin = ceiling((n.iter-n.burnin)/1000),
bugs.directory = getOption("bugs.directory"),
         debug = FALSE,
bugs.code.file = "model.txt",
       clearWD = TRUE,
     code.only = FALSE,
      ini.mult = 2,
      list.ini = TRUE,
           org = FALSE,
       program = "BRugs",
     Transform = NULL,
     trans.tol = 1e-6,
           ... )
## S3 method for class 'MCmcmc'
summary( object, alpha=0.05, ...)
## S3 method for class 'MCmcmc'
print( x, digits=3, alpha=0.05, ... )
## S3 method for class 'MCmcmc'
subset( x, subset=NULL, allow.repl=FALSE, chains=NULL, ... )
## S3 method for class 'MCmcmc'
mcmc( x, ... )
```

Arguments

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

	`mu` - length I
	`alpha` - length M
	`beta` - length M
	`sigma.mi` - length M - if M is 2 then length 1
	`sigma.ir` - length 1
	`sigma.mi` - length M
	`sigma.res` - length M
	If `code.only==TRUE`, `list.ini` indicates whether a list of initial values is returned (invisibly) or not. If `code.only==FALSE`, `list.ini==FALSE` is ignored.
`org`	Logical. Should the posterior of the original model parameters be returned too? If `TRUE`, the `MCmcmc` object will have an attribute, `original`, with the posterior of the parameters in the model actually simulated.
`program`	Which program should be used for the MCMC simulation. Possible values are "brugs","openbugs","ob" (openBUGS), "winbugs","wb" (WinBUGS).
`Transform`	Transformation of data (`y`) before analysis. See `choose.trans`.
`trans.tol`	The tolerance used to check whether the supplied transformation and its inverse combine to the identity.
`...`	Additional arguments passed on to `bugs`.
`object`	A `MCmcmc` object
`alpha`	1 minus the the confidence level
`x`	A `MCmcmc` object
`digits`	Number of digits after the decimal point when printing.
`subset`	Numerical, character or list giving the variables to keep. If numerical, the variables in the `MCmcmc` object with these numbers are selected. If character, each element of the character vector is "grep"ed against the variable names, and the matches are selected to the subset. If a list each element is used in turn, numerical and character elements can be mixed.
`allow.repl`	Should duplicate columns be allowed in the result?
`chains`	Numerical vector giving the number of the chains to keep.

Details

This function uses features currently only available under Windows, so the function returns NULL unless the operating system is Windows.
The model set up for an observation y_{mir} is:

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

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

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

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

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

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

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

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

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

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

Value

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

Furthermore, the object have the following attibutes:

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

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

Author(s)

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

References

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

See Also

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

Examples

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

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

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

Description

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

Usage

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

Arguments

`data`	A dataframe.
`meth`	Vector of methods, numeric, character or factor. Can also be a number or character referring to a column in `data`.
`item`	Vector of items, numeric, character or factor. Can also be a number or character referring to a column in `data`.
`repl`	Vector of replicate numbers, numeric, character or factor. Can also be a number or character referring to a column in `data`.
`y`	Vector of measurements. Can also be a character or numerical vector pointing to columns in `data` which contains the measurements by different methods or a dataframe with columns representing measurements by different methods. In this case the argument `meth` is ignored, and the names of the columns are taken as method names. For the `plot` method the argument is either a vector of indices or names of methods to plot.
`print`	Logical: Should a summary result be printed?
`keep.vars`	Logical. Should the remaining variables from the dataframe `data` be transferred to the `Meth` object.
`object`	A `Meth` object.
`x`	A `Meth` object.
`col.LoA`	What color should be used for the limits of agreement.
`col.pt`	What color should be used for the points.
`cex.name`	Character expansion factor for plotting method names
`var.range`	The range of both axes in the scatter plot and the x-axis in the Bland-Altman plot be?
`diff.range`	The range of yaxis in the Bland-Altman plot. Defaults to a range as the x-axis, but centered around 0.
`var.names`	If logical: should the individual panels be labelled with the variable names?. If character, then the values of the character will be used to label the methods.
`pch`	Plot character for points.
`cex`	Plot charcter expansion for points.

`Transform`	Transformation used to the measurements prior to plotting. Function or character, see `choose.trans` for possible values.
`how`	Character. What sampling strategy should be used, one of `"random"`, `"linked"` or `"item"`. Only the first letter is significant. See details for explanation.
`N`	How many observations should be sampled?
`_data`	A `Meth` object.
`...`	Ignored by the `Meth` and the `summary` and `sample` functions. In the `plot` function, parameters passed on to both the panel function plotting methods against each other, as well as to those plotting differences against means.

Details

In order to perform analyses of method comparisons it is convenient to have a dataframe with classifying factors , `meth`, `item`, and possibly `repl` and the response variable `y`. This function creates such a dataframe, and gives it a class, `Meth`, for which there is a number of methods: `summary` - tabulation, `plot` - plotting and a couple of analysis methods.

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

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

Value

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

Author(s)

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

Examples

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

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

# Sample items
```

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

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

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

Description

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

Usage

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

Arguments

`Ni`	The number of items (patient, animal, sample, unit etc.)
`Nm`	The number of methods of measurement.
`Nr`	The (maximal) number of replicate measurements for each (item,method) pair.
`nr`	The minimal number of replicate measurements for each (item,method) pair. If `nr<Nr`, the number of replicates for each (meth,item) pair is uniformly distributed on the points `nr:Nr`, otherwise `nr` is ignored. Different number of replicates is only meaningful if replicates are not linked, hence `nr` is also ignored when `sigma.ir>0`.
`alpha`	A vector of method-specific intercepts for the linear equation relating the "true" underlying item mean measurement to the mean measurement on each method.
`beta`	A vector of method-specific slopes for the linear equation relating the "true" underlying item mean measurement to the mean measurement on each method.
`mu.range`	The range across items of the "true" mean measurement. Item means are uniformly spaced across the range. If a vector length `Ni` is given, the values of that vector will be used as "true" means.
`sigma.mi`	A vector of method-specific standard deviations for a method by item random effect. Some or all components can be zero.

`sigma.ir`	Method-specific standard deviations for the item by replicate random effect.
`sigma.mir`	A vector of method-specific residual standard deviations for a method by item by replicate random effect (residual variation). All components must be greater than zero.
`m.thin`	Fraction of the observations from each method to keep.
`i.thin`	Fraction of the observations from each item to keep. If both `m.thin` and `i.thin` are given the thinning is by their componentwise product.

Details

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

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

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

Value

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

Author(s)

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

See Also

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

Examples

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

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

Description

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

Usage

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

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

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

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

Arguments

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

Details

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

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

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

Value

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

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

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

Author(s)

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

See Also

BA.est AltReg MCmcmc

Examples

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

---

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

Description

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

Usage

```
data(milk)
```

Format

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

meth a factor with levels Gerber Trig

item sample id

y a numeric vector

Source

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

Examples

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

`ox` *Measurement of oxygen saturation in blood*

Description

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

Usage

```
data(ox)
```

Format

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

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

`item` Id for the child

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

`y` Oxygen saturation in percent.

Examples

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

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

Description

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

Usage

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

Format

The format is a MCmcmc object.

Details

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

Examples

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

PBreg	*Passing-Bablok regression*

Description

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

Usage

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

Arguments

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

Details

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

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

Value

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

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

`coefficients`	a matrix of 3 columns and 2 rows, containing the estimates of the intercept and slope, along with their confidence boundaries.
`residuals`	defined as in the `"lm"` class, as the response minus the fitted value.
`fitted.values`	the fitted values.
`model`	the model data frame used.
`n`	a vector of two values: the number of observations read, and the number of observations used.
`S`	A vector of all slope estimates.
`adj`	A vector of fit parameters, where *Ss* is the number of estimated slopes (`length(S)`), *K* is the offset for negative slopes, *M1* and *M2* are the locations of confidence boundaries in `S`, and *l* and *L* are the numbers of points above and below the fitted line, used in cusum calculation.
`cusum`	A vector of cumulative sums of residuals sorted by the D-rank.
`Di`	A vector of D-ranks.

Note

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

Author(s)

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

References

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

See Also

`plot.PBreg`, `Deming`.

Examples

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

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

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

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

Description

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

Usage

```
data(PEFR)
```

Format

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

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

`item` Numeric vector, the person ID.

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

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

Source

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

Examples

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

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

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

Author(s)

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

See Also

`perm.repl`

Examples

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

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

Description

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

Usage

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

Arguments

`x`	A `MCmcmc` object
`axlim`	The limits for the axes in the panels
`wh.cmp`	Numeric vector or vector of method names. Which of the methods should be included in the plot?
`lwd.line`	Numerical vector of length 2. The width of the conversion line and the prediction limits. If the second values is 0, no prediction limits are drawn.
`col.line`	Numerical vector of length 2. The color of the conversion line and the prediction limits.
`lty.line`	Numerical vector of length 2. The line types of the conversion line and the prediction limits.
`eqn`	Should the conversion equations be printed on the plot?. Defaults to `TRUE`.
`digits`	How many digits after the decimal point shoudl be used when printing the conversion equations.
`grid`	Should a grid be drawn? If a numerical vector is given, the grid is drawn at those values.
`col.grid`	What color should the grid have?
`points`	Logical or character. Should the points be plotted. If `TRUE` or `"repl"` paired values of single replicates are plotted. If `"perm"`, replicates are randomly permuted within (item, method) befor plotting. If `"mean"`, means across replicates within item, method are formed and plotted.
`col.pts`	What color should the observation have.
`pch.pts`	What plotting symbol should be used.
`cex.pts`	What scaling should be used for the plot symbols.
`...`	Parameters to pass on. Currently not used.

Value

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

See Also

`MCmcmc`, `print.MCmcmc`

Examples

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

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

Description

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

Usage

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

Arguments

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

Author(s)

Michal J. Figurski <mfigrs@gmail.com>

References

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

See Also

PBreg, Deming.

Examples

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

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

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

Description

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

Usage

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

Arguments

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

Details

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

Value

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

Author(s)

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

See Also

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

Examples

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

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

Description

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

Usage

```
data(plvol)
```

Format

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

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

`item` a numeric vector

`y` a numeric vector

Source

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

Examples

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

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

Description

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

Usage

```
data(sbp)
```

Format

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

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

`item` Person id, numeric.

`repl` Replicate number, a numeric vector

`y` Systolic blood pressure masurement, a numeric vector

Source

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

See Also

`sbp.MC`

Examples

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

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

Description

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

Usage

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

Format

The format is a MCmcmc object.

Details

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

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

Examples

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

# Conversion between methods and variance components
print.MCmcmc(sbp.MC)

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

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

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

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

scint *Relative renal function by Scintigraphy*

Description

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

Usage

```
data(scint)
```

Format

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

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

`item` Patient identification.

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

`age` Age of the patient.

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

Source

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

Examples

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

TDI *Compute Lin's Total deviation index*

Description

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

Usage

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

Arguments

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

Details

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

Value

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

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

Note

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

Author(s)

Bendix Carstensen, bxc@steno.dk

References

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

See Also

`BA.plot`,`corr.measures`

Examples

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

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

Description

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

Usage

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

Arguments

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

Details

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

Value

A dataframe.

Author(s)

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

See Also

`perm.repl`

Examples

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

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

Description

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

Usage

```
data(VitCap)
```

Format

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

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

`item` a numeric vector, the person ID, i.e. the 72 patients `y` a numeric vector, the measurements, i.e. vital capacity.

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

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

Source

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

Examples

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