Introduction to the MethComp package

(compiled Friday 20th May, 2011, 02:09)

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

Contents

Chapter 1

Overview of MethComp

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

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

1.1 Data structures

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

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

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

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

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

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

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

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

> ox <- Meth( ox )

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

> summary( ox )

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

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

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

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

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

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

> str( sc )

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

```
> summary( sc )

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

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

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

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

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

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

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

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

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

> str( hb1 )

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

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

1.1.1 Wide format data

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

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

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

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

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

> head( RM )

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

1.2 Function overview

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

```
> ?plot.Meth
```

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

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

1.2.1 Graphical functions

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

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

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

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

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

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

1.2.2 Data manipulating functions

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

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

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

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

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

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

1.2.3 Analysis functions

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

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

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

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

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

1.2.4 Reporting functions

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

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

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

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

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

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

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

Chapter 2

Worked examples

2.1 Fat measurements: Exchangeable replicates

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

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

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

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

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

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

> str(vis)

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

> summary(vis)
```

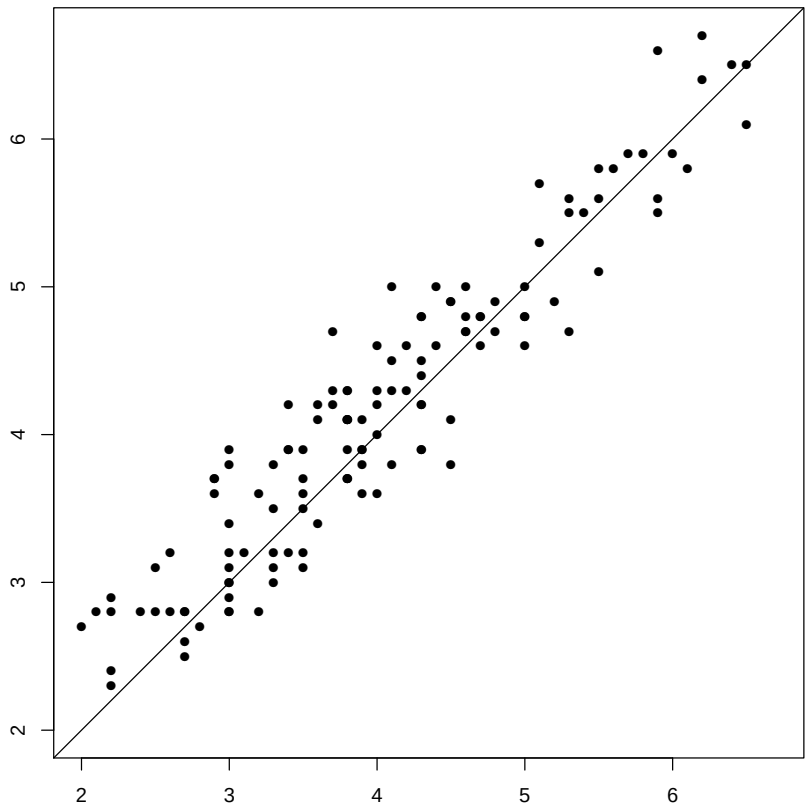

Figure 2.1: *Two observers measuring visceral fat.*

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

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

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

Note:
 Replicate measurements are taken as separate items!

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

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

```
> plot( vis )
```

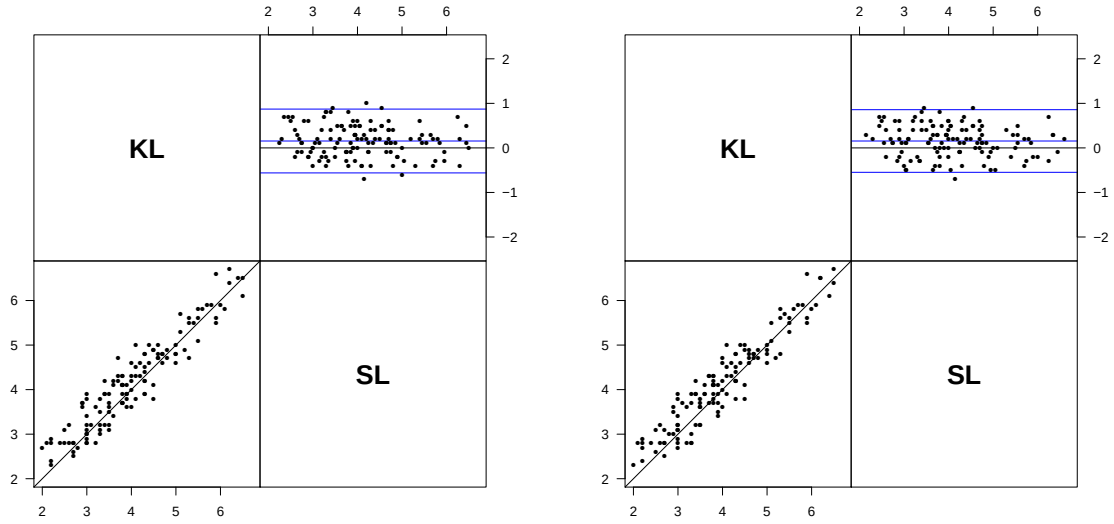

Figure 2.2: *Plot of two methods of measuring visceral fat, using different pairings of the replicates; the left panel is using the pairing in the original coding, the right panel is with a random permutation of replicates.*

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

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

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

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

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

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

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

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

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

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

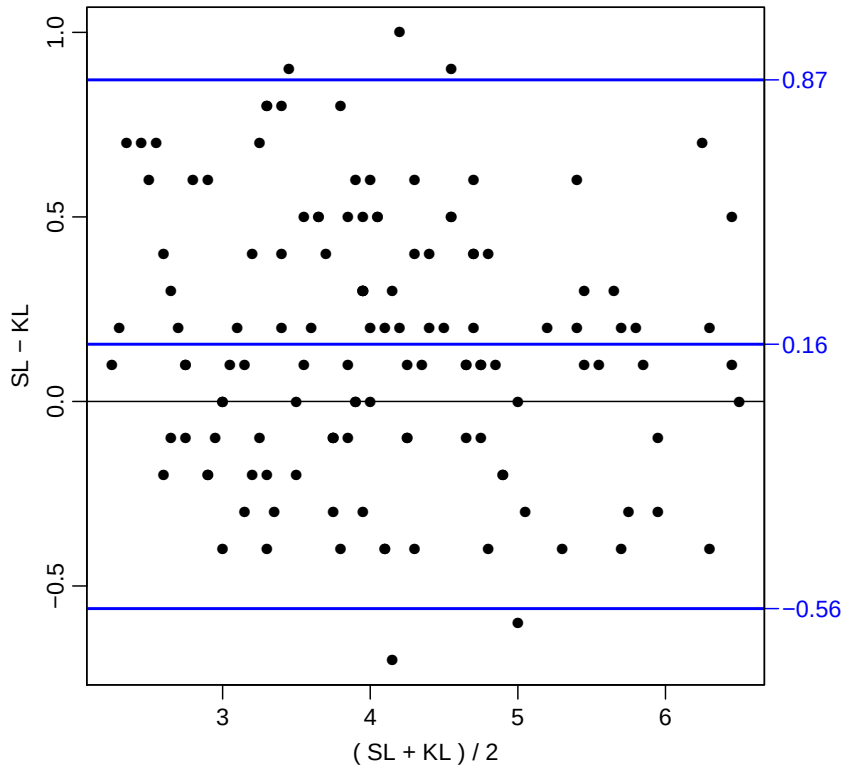

Figure 2.3: *Bland-Altman plot of two observers measuring visceral fat.*

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

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

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

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

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

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

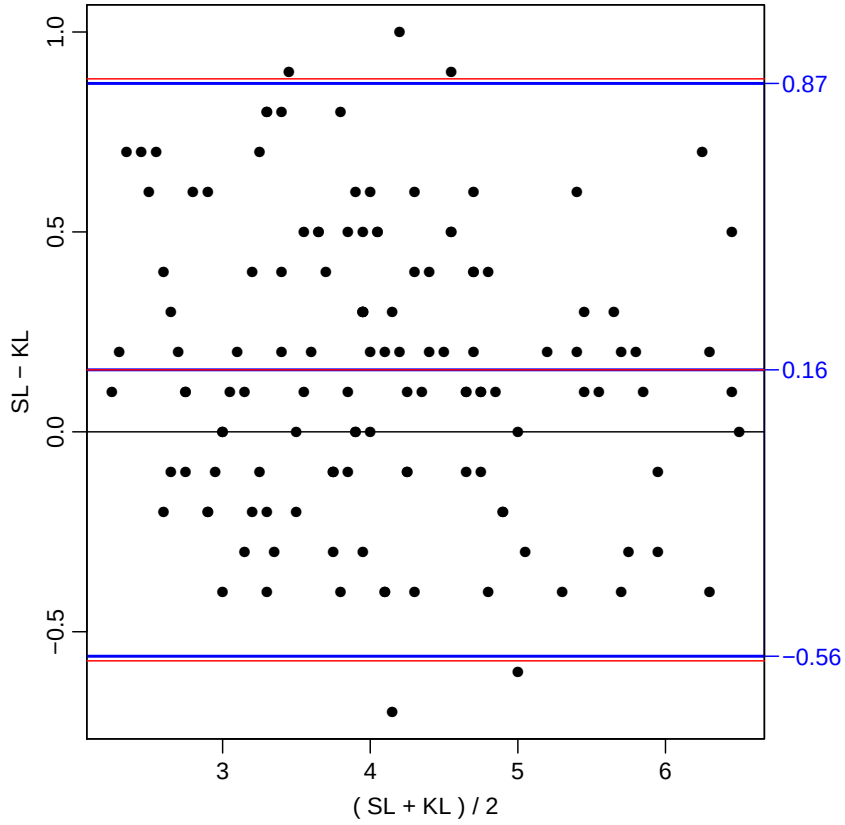

Figure 2.4: *Bland-Altman-plot of two methods of measuring visceral fat, using different pairings of the replicates. The blue lines are the LoA based on taking the paired replicates as items, the red lines are based on the estimates from the proper variance component model.*

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

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

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

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

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

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

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

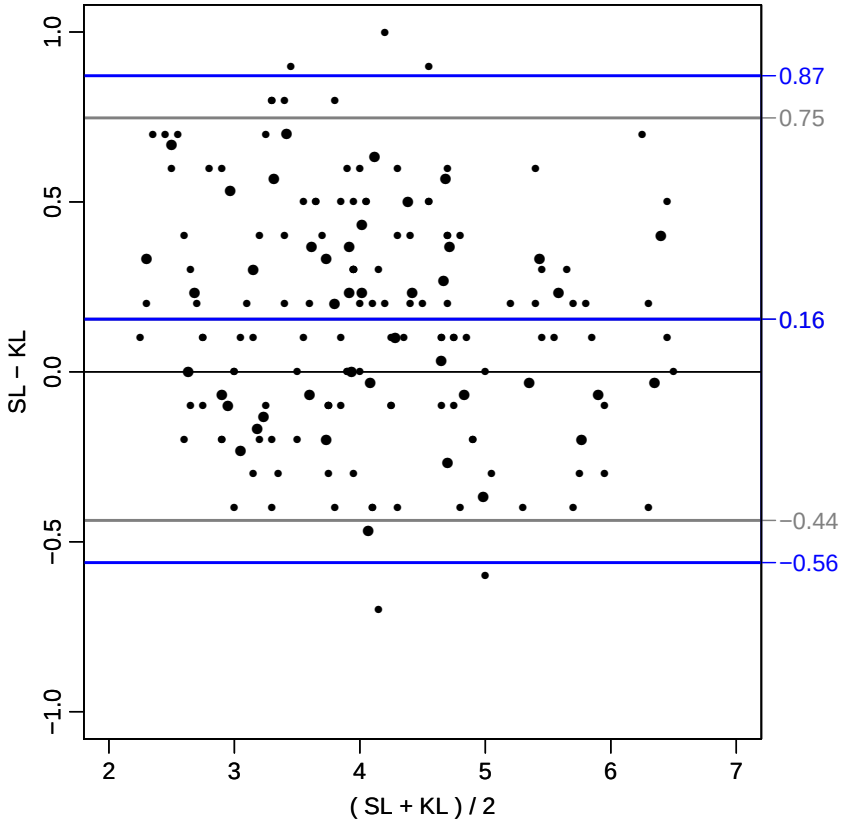

Figure 2.5: *Bland-Altman-plot of two methods of measuring visceral fat, based on the arbitrary pairing of the replicates (black) and on the mean over replicates (grey).*

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

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

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

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

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

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

2.2 Cardiac output: Linked replicates?

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

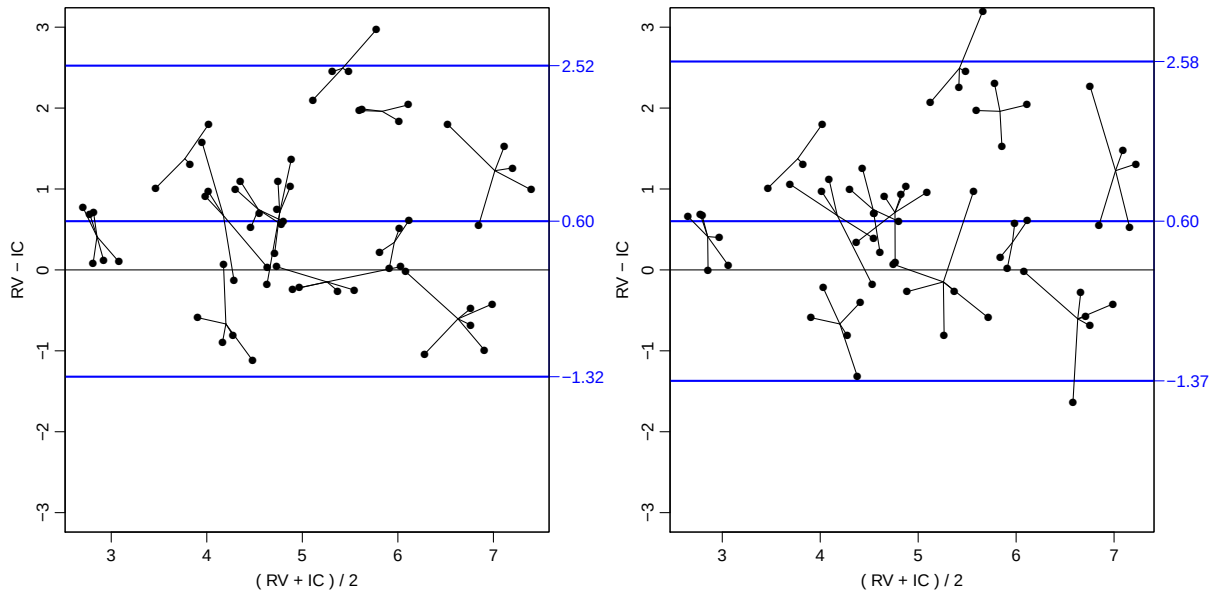

Figure 2.6: *Bland-Altman plots of the cardiac data. The left panel is the orginal data using replicate numbers to pair mesurements, the right is using a random permutation of replicates for the pairing. Even if replicates are claimed to be linked, the replicates the LoA in the right panel are not substantially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

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

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

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

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

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

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

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

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

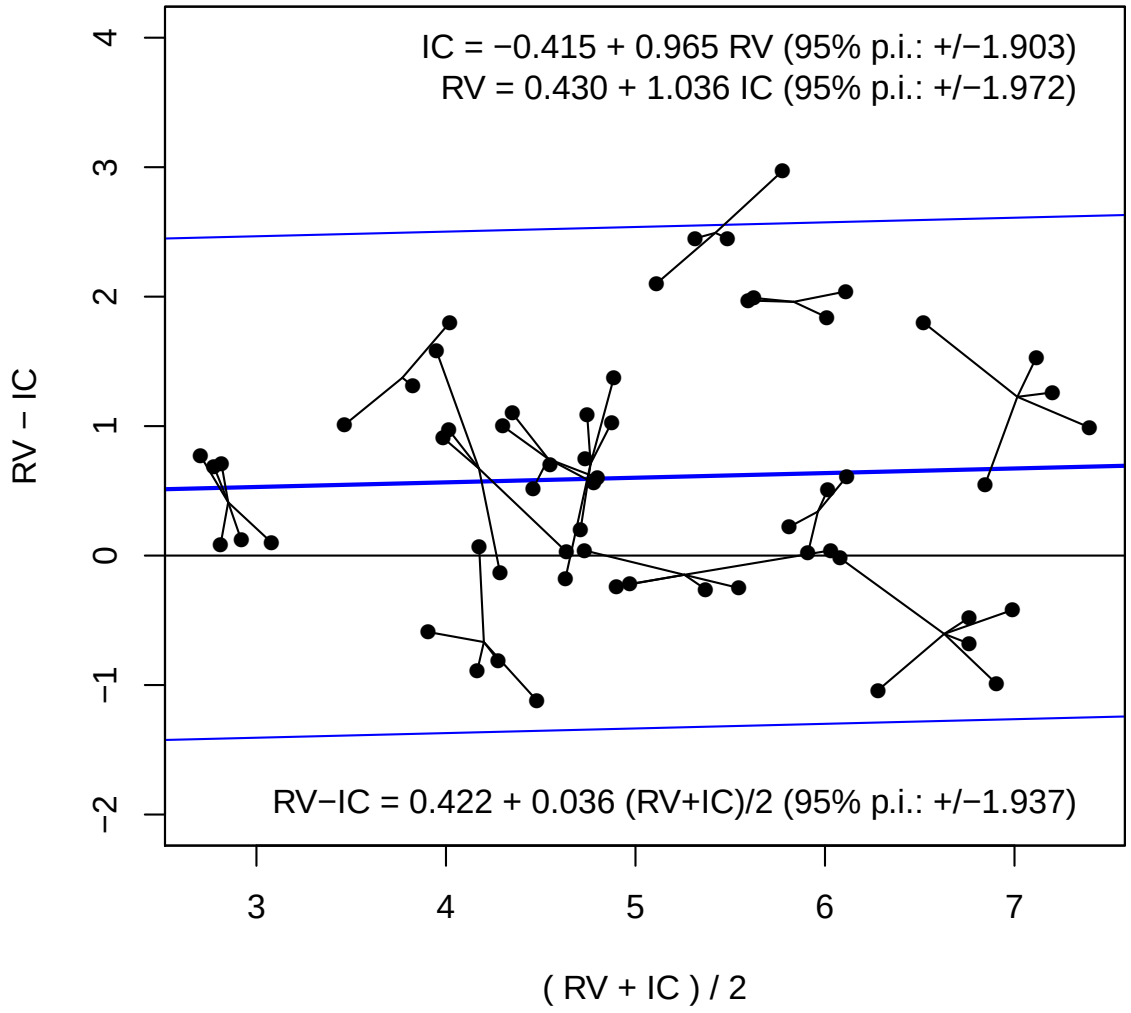

Figure 2.7: *Bland-Altman plot of the cardiac data with a fitted regression line.*

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

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

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

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

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

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

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

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

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

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

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

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

2.3 Systolic blood pressure: Linked replicates by two methods

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

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

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

> str(sbp)

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

> plot( sbp )

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

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

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

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

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

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

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

> BA.plot( sbp )

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

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

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

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

The resulting plots are shown in figure 2.9.

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

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

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

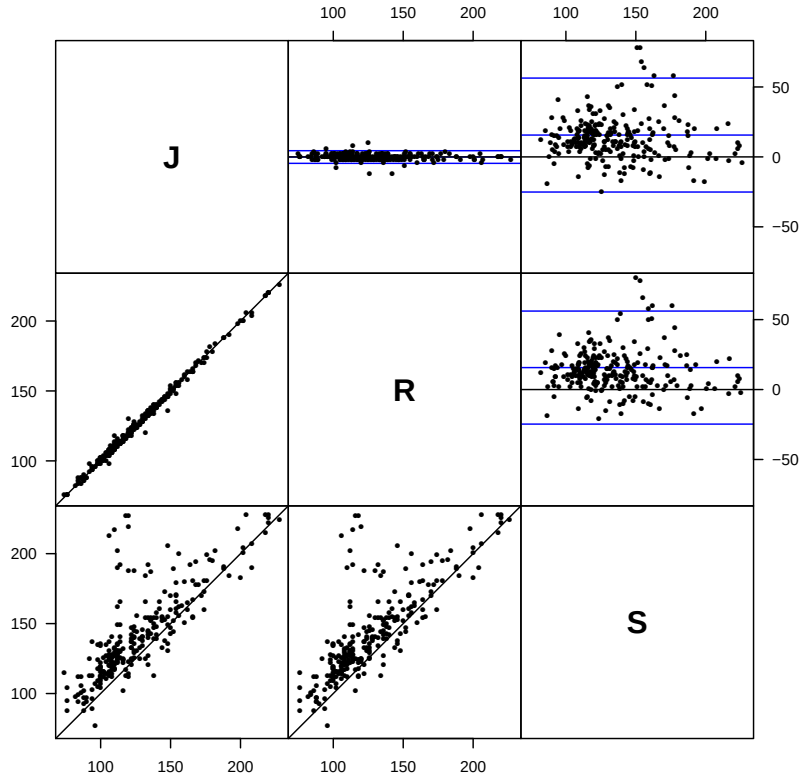

Figure 2.8: *Graphical overview of the* `sbp` *data. The methods J and R are two human observers, whereas method S is an automatic device.*

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

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

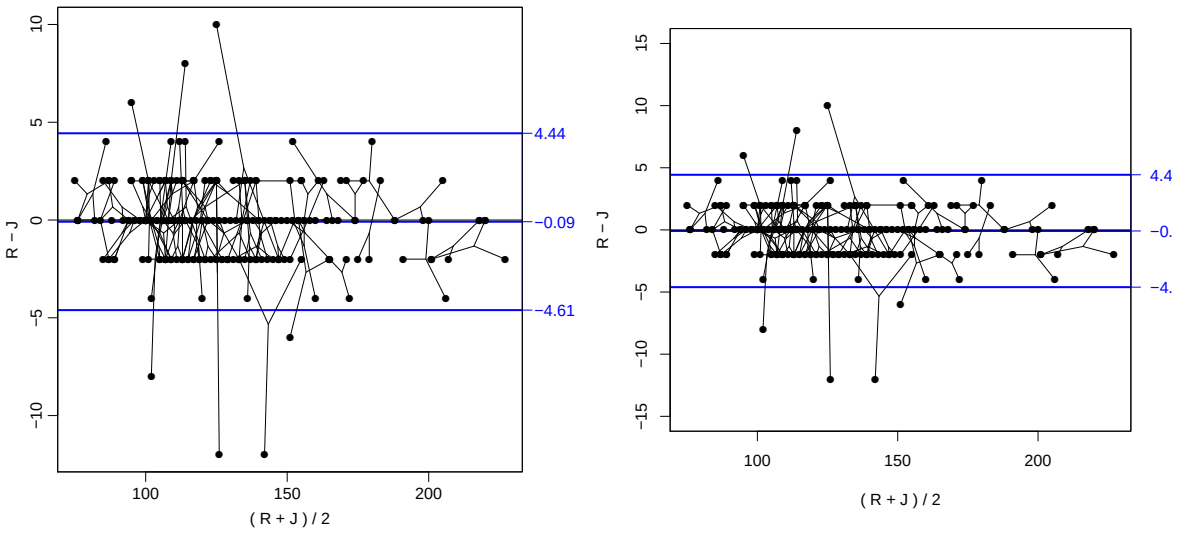

Figure 2.9: *Bland-Altman plot of the* sbp *data. Replicates are linked between methods, so the single replicates in the data has been used as single measurements when doing the Bland-Altman plot. Measurements from the same person are joined by thin lines. The only difference between the two plots is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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

> str(sbp)

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

> plot( perm.repl(sbp) )

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

The two resulting plots are shown in figure 2.10.

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

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

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

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

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

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

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

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

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

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

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

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

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

 Variance components (sd):
```

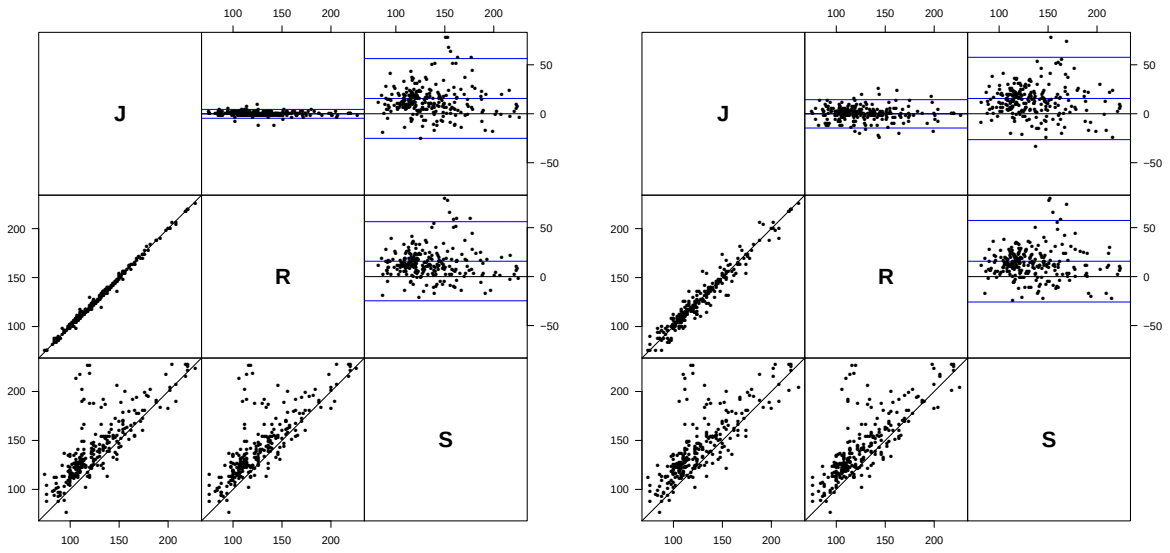

Figure 2.10: *Graphical overview of the* `sbp` *data; the left panel with the original replicate numbers used for matching; the other with replicates permuted randomly within methods.*

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

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

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

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

Bibliography