Statistical Analysis of Method Comparison Studies

Bendix Carstensen

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

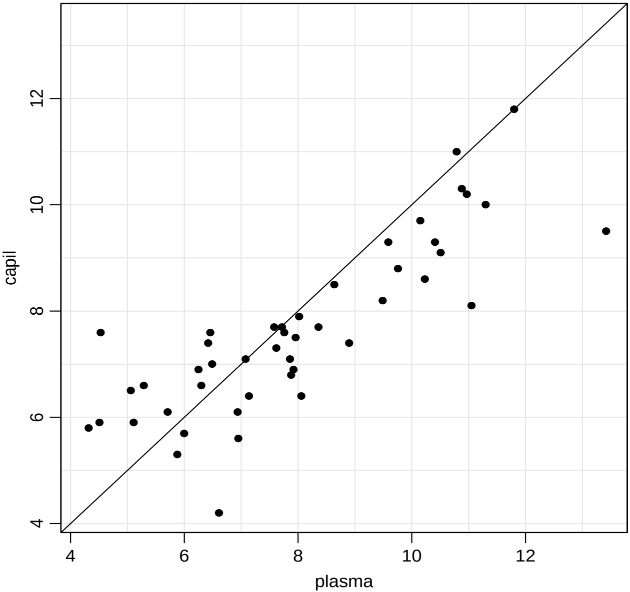
Haukeland University Hospital, Bergen, Norway
19–20 March 2014

`http://BendixCarstensen.com/MethComp/Courses/Bergen.2014`

What this is about

- Two (laboratory) methods for measuring the same clinical quantity.
- Persons are measured with both methods.
- Scaled measurements (continuous).
- Errors in both variables.

Glucose measurements



Course outlook

- **Model** based approach
- Explicit **parametric** models:
    - Assumptions are made clear
    - — relaxing assumptions is clear
- **Comparison** of methods:
  — can one replace the other?
- **Conversion** between methods:
  — if measurement is y_1 with method 1, what would it be with method 2?
- Examples from `MethComp` package for **R**.
- Code and output included on the slides
- — and on the course web-site.

Order of topics 19-20 March

- Wednesday 19th
  - One measurement by each method
  - Computing
  - Linear bias between methods
  - Variable SD
  - Practical `milk`, `plvol`
  - Replicate measurements, exchangeable / linked
  - Practical `fat`, `sbp2`
  - Repeatability, reproducibility
  - Coefficient of variation
- Thursday 20th
  - Replicate measurements and linear bias
  - Practical `ox` 1–8
  - Converting between methods
  - MCMC methods for estimation of variance components
  - Practical `ox` 9–

Comparing two methods with one measurement on each

Bendix Carstensen

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

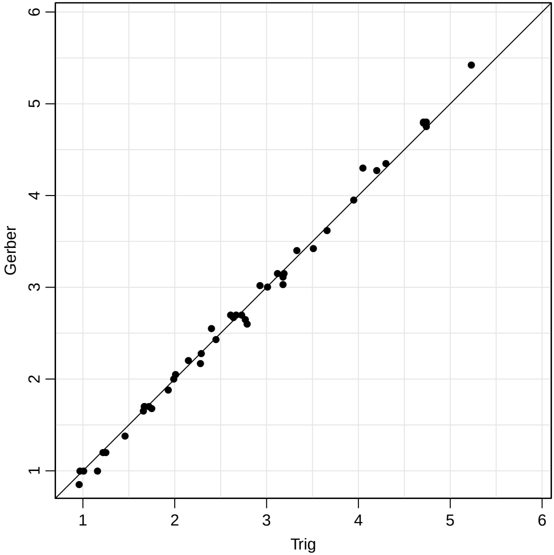
`(Comp-simple)`

Comparing measurement methods

General questions:

- Are results systematically different?
- Can one method safely be replaced by another?
- What is the size of measurement errors?
- Different centres use different methods of measurement: How can we convert from one method to another?
- How precise is the conversion?

Fat content in human milk:



The relationship looks like:

$$y_1 = a + by_2$$

```
> library(MethComp)
> # sessionInfo()
> data( milk )
> 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
```

```
> par( mar=c(3,3,1,1),mgp=c(3,1,0)/1.6 )
> BA.plot( milk, pl.type="comp", col.line="transparent",
+          lwd=c(3,0,0), axlim=c(1,6)-0.1 )
> abline(0,1)
```

Two methods — one measurement by each

- How large is the difference between a measurement with method 1 and one with method 2 on a (randomly chosen) person?

$$D_i = y_{2i} - y_{1i}, \qquad \bar{D}, \qquad \mathrm{s.d.}(D)$$

- 95% prediction interval for the difference between a measurement by method 1 and one by method 2. [1, 2]
- **Limits of agreement**:

$$\bar{D} \pm 2 \times \mathrm{s.d.}(D)$$

Limits of agreement: Interpretation

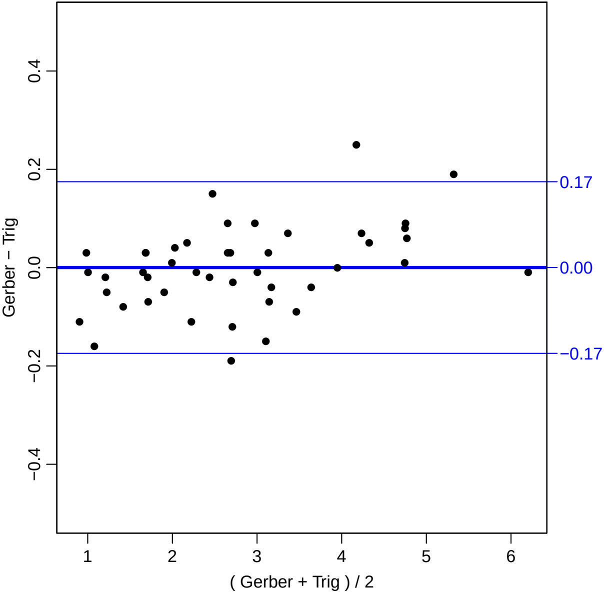
- If a new patient is measured **once** with each of the two methods, the difference between the two values will with 95% probability be within the limits of agreement.
- This is a **prediction** interval for a single (future) difference.
- Interpretation requires a **clinical** input:
  Are the limits of agreement sufficiently narrow to make the use of either of the methods clinically acceptable?
- Is it relevant to test if the mean is 0?

Limits of agreement: Test? No!

Testing whether the difference is 0 is a bad idea:

- Small study: Null accepted even if the difference is important.
- Large study: Null rejected even if the difference is clinically irrelevant.
- It is an **equivalence** problem:
  1. How small can we reasonably safely assume the differences to be?
  2. **Testing is irrelevant:**
     — not interesting if the **mean** difference is significantly different from 0.
  3. **Clinical input is required** to interpret the **prediction** interval.

Limits of agreement:



Plot differences (D_i) versus averages (A_i).

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

Model behind "Limits of agreement"

- Methods $m = 1, \ldots, M$, applied to $i = 1, \ldots, I$ individuals:

$$\begin{aligned} y_{mi} &= \alpha_m + \mu_i + e_{mi} \\ e_{mi} &\sim \mathcal{N}(0, \sigma_m^2) \quad \text{measurement error} \end{aligned}$$

- Two-way analysis of variance model, with **different** variances in columns.
- Different variances are not identifiable without replicate measurements for $M = 2$.

  The variances σ_m are based on the distance of the obs to the mean across methods, but they are always numerically identical with only 2 methods.

Limits of agreement:

- Usually interpreted as the likely difference between two future measurements, one with each method:

$$\widehat{y_2 - y_1} = \hat{D} = \alpha_2 - \alpha_1 \pm 2\,\mathrm{s.d.}(D)$$

- Convert to prediction interval for y_2 given y_1:

$$\hat{y}_{2|1} = \hat{y}_2 | y_1 = \alpha_2 - \alpha_1 + y_1 \pm 2\,\mathrm{s.d.}(D)$$

- Formally, we should replace:

$$2 \quad \rightarrow \quad \mathrm{t}_{0.975}^{(I-1)} \sqrt{1 + 1/I}$$

which equals 2 for $I = 85$ and 1.96 for $I = \infty$

Spurious correlation?

Different variances induce correlation between D_i and $A_i = (y_{1i} + y_{2i})/2$, if the variances of y_{1i} and y_{2i} are ζ_1^2 and ζ_2^2 respectively:

$$\mathrm{cov}(D_i, A_i) = \frac{1}{2}(\zeta_2^2 - \zeta_1^2) \neq 0 \quad \text{if } \zeta_1 \neq \zeta_2$$

In correlation terms:

$$\rho(D, A) = \frac{1}{2}\left(\frac{\zeta_2^2 - \zeta_1^2}{\zeta_1^2 + \zeta_2^2}\right)$$

i.e. the correlation depends on whether the difference between the variances is large relative to the sizes of the two.

. . . not really. . .

The variances we were using were the **marginal** variances of y_1 and y_2:

$$y_{mi} = \alpha_m + \mu_i + e_{mi}$$
$$\mathrm{var}(y_m) = \mathrm{var}(\mu_i) + \sigma_m^2$$

and hence the correlation expression is:

$$\rho(D, A) = \frac{1}{2}\left(\frac{\zeta_2^2 - \zeta_1^2}{\zeta_1^2 + \zeta_2^2}\right) = \frac{1}{2}\left(\frac{\sigma_2^2 - \sigma_1^2}{2\mathrm{var}(\mu_i) + \sigma_1^2 + \sigma_2^2}\right)$$

Hence only relevant if $\mathrm{var}(\mu_i)$ is small relative to σ_1^2 and σ_2^2.

Not likely in practise — the μs are normally chosen to be widely spread, so $\mathrm{var}(\mu_i) \gg \sigma_1^2, \sigma_2^2$

Introduction to computing

Bendix Carstensen

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

(`Intro-comp`)

Course structure

The 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.

- **R** for data manipulation and graphics.
- Occasionally `BUGS` (JAGS) for estimation in non-linear variance component models.

How it works

Example data sets are included in the `MethComp` package.

Functions in `MethComp` are based on a data frame with a particular structure; a `Meth` object:

`meth` — method (factor)
`item` — item, person, individual, sample (factor)
`repl` — replicate (if present) (factor)
`y` — the actual measurement (numerical)

Once converted to a `Meth` object, just use `summary`, `plot` etc.

How it looks I

```
> library( MethComp )
> 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

> ( subset( ox, as.integer(item)<3 ) )
```

How it looks II

```
    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
7  pulse    1    1 71.0
8  pulse    1    2 72.0
9  pulse    1    3 73.0
10 pulse    2    1 68.0
11 pulse    2    2 67.0
12 pulse    2    3 68.0
```

```
> subset( to.wide(ox), as.integer(item)<3 )
```

```
  item repl   CO pulse
1    1    1 78.0    71
2    1    2 76.4    72
3    1    3 77.2    73
4    2    1 68.7    68
5    2    2 67.6    67
6    2    3 68.3    68
```

Analyses in this course

- Scatter plots.
- Bland-Altman plots ($(y_2 - y_1)$ vs. $(y_1 + y_2)/2$)
- Limits of agreement.
- Models with constant bias.
- Models with linear bias.
- Conversion formulae between methods.
- Plots of converison equations.
- Reporting of variance components.
- Transformation of response.

Data objects im `MethComp`

- `Meth` Dataframe in the "long" format, with predefined variable names.
- `MethComp` Results from an analysis with estimated conversions betweenmethods and (if applicable) variance components. Produced by different functions.
- `MCmcmc` Results from a MCMC analysis of a model. Can be converted to a `MethComp` object.

Functions in the `MethComp` package

5 broad categories of functions in `MethComp`:

- Data manipulation — reshaping and changing data.
- Graphical — exploring data.
- Simulation — generating datasets or replacing variables.
- Analysis functions — fitting models to data.
- Reporting functions — displaying the results from analyses.

Data manipulation functions

- `Meth` Sets up a `Meth` object — a dataframe in the "long" format, with predefined variable names.
- `make.repl` Generates 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`/`to.long` Transforms a data frame in the long form to the wide form and vice versa.
- `Meth.sim` Simulates a dataset (a `Meth` object) from a method comparison experiment.

Graphical functions (basic)

- `plot.Meth` Plots all methods against all other, both as a scatter plot and as a Bland-Altman plot.
- `BA.plot` Makes a Bland-Altman plot of two methods from a data frame with method comparison data, and computes limits of agreement.
- `bothlines` Adds regression lines of y on x and vice versa to a scatter plot.

Analysis functions (simple)

- `DA.reg`, regresses the differences on the averages. Also regresses the absolute residuals on the averages to check whether the variance is constant. Returns a `MethComp` object.
- `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. Returns a `MethComp` object.
- `VC.est` The workhorse behind `BA.est`.
- `Deming` Performs Deming regression, i.e. regression with errors in both variables.

Analysis functions (general)

- `MCmcmc` Estimates via `BUGS` (JAGS) in the general model with non-constant bias. Produces an `MCmcmc` object. WHich can be converted to a `MethComp` object.
- `AltReg` Estimates via ad-hoc procedure (alternating regressions) in a model with linear bias between methods. Returns a matrix of estimates with the conversion parameters as well as the variance components. Returns a `MethComp` object.

Reporting functions

- `print.MethComp` Prints a table of conversion equations based on an estimated model.
- `plot.MethComp` Graphs the estimated relationship between methods based on an estimated model.
- `print.MCmcmc` Table of conversion equations between methods analyzed.
- `plot.MCmcmc` Conversion lines between methods with prediction limits.
- `post.MCmcmc` Smoothed posteriors of estimates.
- `trace.MCmcmc` Simulation traces from an `MCmcmc` object.

Does it work? I

You should get something reasonable out of this:

```
> library( MethComp )
> data( ox )
> ox <- Meth( ox )
> summary( ox )
> plot( ox )
> BA.plot( ox )
> BA.est( ox )
> ( AR.ox <- AltReg(ox,linked=TRUE,trace=TRUE) )
> MCmcmc( ox, code.only=TRUE )
> MC.ox <- MCmcmc( ox, n.iter=500 )
> print( MC.ox )
> plot( MC.ox )
> trace.MCmcmc( MC.ox )
> post.MCmcmc( MC.ox )
```

Non-constant difference

Bendix Carstensen

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

(`Non-const`)

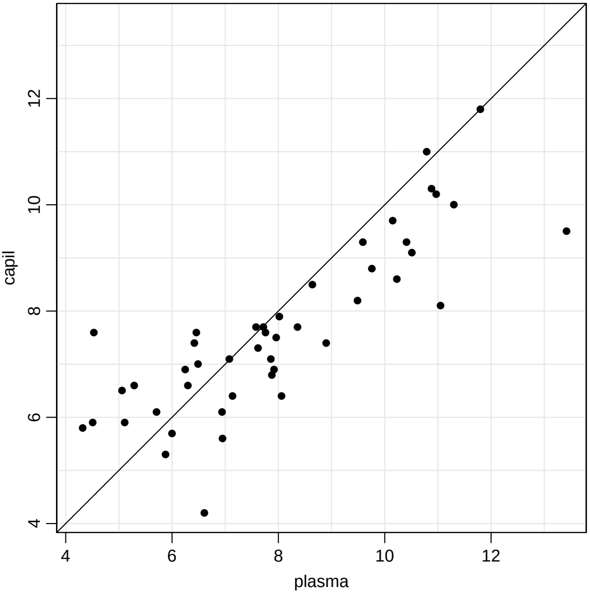
Limits of agreement — assumptions

- The difference between methods is constant
- The variances of the methods (and hence of the difference) is constant
- “Constant” means constant across the range of measurement values

Check this by:

- Regress differences on averages.
- Regress absolute residuals from this on the averages.

Glucose measurements



```
> options( width=61 )
> library(MethComp)
> data( glucose )
> gluc <-subset( glucose, type %in% levels(type)[c(2,4)] &
+                         meth %in% c("h.cap","o.cap","n.plas1"),
+                         select=c(2,3,4,6) )
> str( gluc )
```

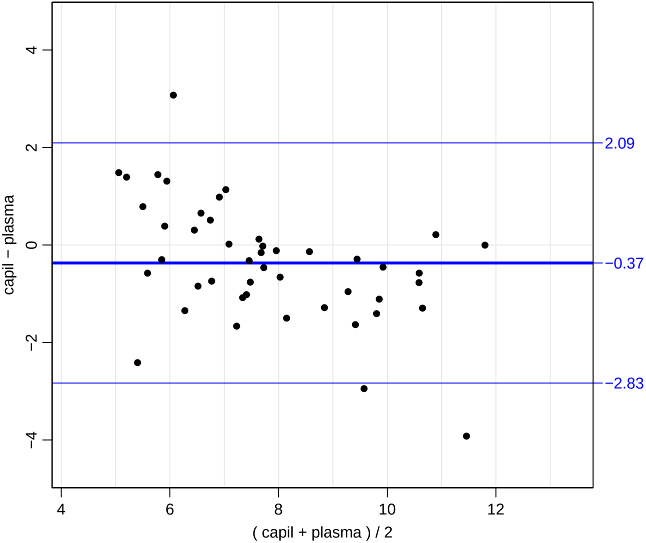
```
'data.frame': 472 obs. of  4 variables:
 $ type: Factor w/ 4 levels "blood","plasma",..: 2 4 2 4 2 4 2 4 2 4 ...
 $ item: num  1 1 1 1 1 1 1 1 2 2 ...
 $ time: num  0 0 30 30 60 60 120 120 0 0 ...
 $ y   : num  6.36 5.1 10.3 9.8 13.33 ...
```

```
> glu120 <- Meth( subset( gluc, time==120 ), meth="type", print=F )
> summary( glu120 )
```

```
         #Replicates
Method             1 #Items #Obs: 119 Values:  min  med   max
  plasma          73     73        73         4.32 7.92 13.42
  capil           46     46        46         4.20 7.45 11.80
```

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, wh.comp=2:1, pl.type="comp",
+          col.line="transparent" )
> abline( 0, 1 )
```

Glucose measurements



```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, wh.comp=2:1, pl.type="BA" )
```

Regress differences on averages

$$D_i = a + bA_i + e_i, \quad \mathrm{var}(e_i) = \sigma_D^2$$

If b is different from 0, we could use this equation to derive LoA:

$$a + bA_i \pm 2\sigma_D$$

or convert to prediction as for LoA:

$$y_{2|1} = y_1 + a + bA_i \approx y_1 + a + by_1 = a + (1+b)y_1$$

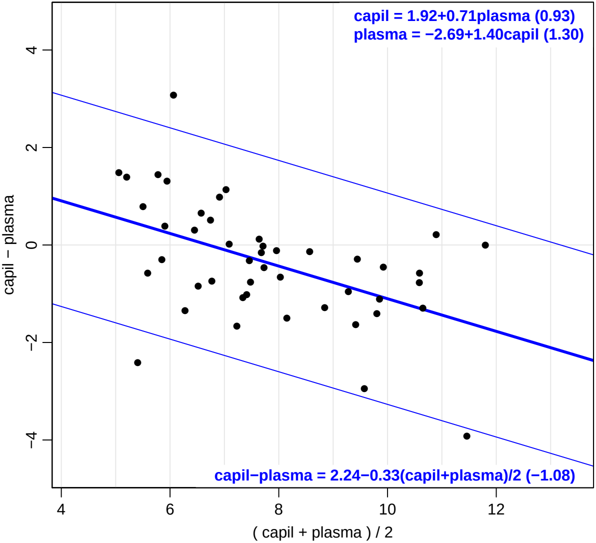
Exchanging methods would give:

$$y_{1|2} = -\,a + (1-b)y_1$$

instead of:

$$y_{1|2} = \frac{-a}{1+b} + \frac{1}{1+b}y_1$$

Variable limits of agreement



```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, dif.type="lin",wh.comp=2:1, pl.type="BA" )
```

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, dif.type="lin",wh.comp=2:1, pl.type="BA",
+          eqn=TRUE )
```

```
Relationships between methods:
 capil-plasma = 2.24-0.33(capil+plasma)/2 (-1.08)
 capil = 1.92+0.71plasma (0.93)
 plasma = -2.69+1.40capil (1.30)
```

Using the regression of D on A properly

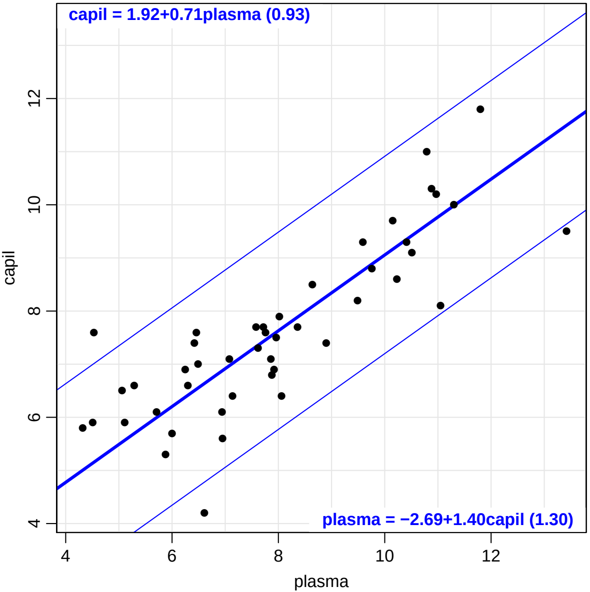
$$\begin{aligned} y_{2i} - y_{1i} &= a + b(y_{1i} + y_{2i})/2 + e_i \\ y_{2i}(1 - b/2) &= a + (1 + b/2)y_{1i} + e_i \end{aligned}$$

$$y_{2i} = \frac{a}{1 - b/2} + \frac{1 + b/2}{1 - b/2} y_{1i} + \frac{1}{1 - b/2} e_i$$

$$y_{1i} = \frac{-a}{1 + b/2} + \frac{1 - b/2}{1 + b/2} y_{2i} + \frac{1}{1 + b/2} e_i$$

Details found in [5]
This is what comes out of the functions
`DA.reg` and `BA.plot`.

Conversion equation with prediction limits



```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, dif.type="lin",wh.comp=2:1, pl.type="conv",
+          eqn=TRUE )

Relationships between methods:
 capil-plasma = 2.24-0.33(capil+plasma)/2 (-1.08)
 capil = 1.92+0.71plasma (0.93)
 plasma = -2.69+1.40capil (1.30)
```

Why does this work?

The general model for the data is:

$$\begin{aligned} y_{1i} &= \alpha_1 + \beta_1\mu_i + e_{1i}, \qquad & e_{1i} \sim \mathcal{N}(0, \sigma_1^2) \\ y_{2i} &= \alpha_2 + \beta_2\mu_i + e_{2i}, \qquad & e_{2i} \sim \mathcal{N}(0, \sigma_2^2) \end{aligned}$$

- Work out the prediction of y_{2i} given an observation of y_{1i} in terms of the αs and βs.
- Work out how differences relate to averages in terms of αs and βs.
- Use til to work out relationship between the (α, β) and (a, b)
- Then the prediction is as we just derived it.

So why is it wrong anyway?

Conceptually:

Once the β_m is introduced:

$$y_{mi} = \alpha_m + \beta_m \mu_i + e_{mi}$$

measurements by different methods are on different scales.

Hence it has formally no meaning to form the differences.

So why is it wrong anyway?

Statistically:

Under the correctly specified model, the induced model for the differences on the averages A_i, these contain the error terms, and so does the residuals.

So the covariate is not independent of the error terms.

Thus the assumptions behind regression are violated.

Then why use it?

- With only one observation per (method,item) there is not much else to do.
- If the slope linking the two methods (β_1/β_2) is not dramatically different from 1, the violations are not that big.
- Implemented in `BA.plot` and in `DA.reg`, which also checks the residuals.

For further details, see [5].

Limits of agreement — assumptions

- The difference between methods is constant
- The variances of the methods (and hence of the difference) is constant
- Residuals follow a normal distribution

Check this by:

- Regress differences on averages
- Regress absolute residuals from this on the averages
- . . . the cental limit theorem?

Regressing residuals on averages

- Residuals $\sim \mathcal{N}(0, \sigma^2)$
  $\Rightarrow$ absolute residuals half-normal.
- Mean of standard half normal is:

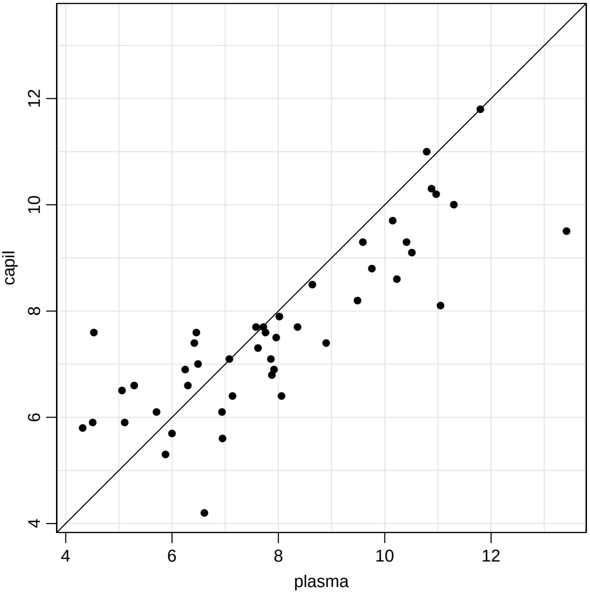
$$\int_0^\infty x(2/\sqrt{2\pi})\exp(-x^2/2)\,\mathrm{d}x = \sqrt{2/\pi}$$

- Mean of absolute residuals is $\sigma\sqrt{2/\pi}$
- Linear relationship of absolute residuals (R_i) to averages (A_i):

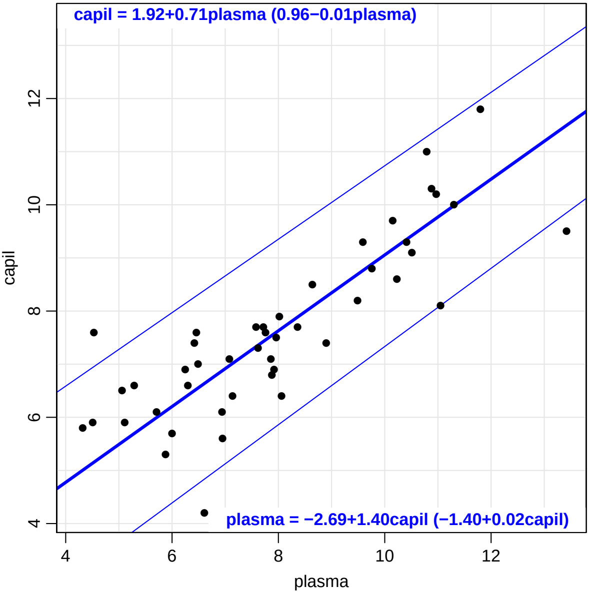
$$R_i = a + bA_i \quad \Leftrightarrow \quad \sigma(A) \approx a\sqrt{\pi/2} + b\sqrt{\pi/2}A$$

- Implemented in `DA.reg`.

Glucose measurements



Variable standard deviation



```
> ( da <- DA.reg( glu120 ) )

 Conversion between methods:
                alpha   beta  sd.pr beta=1 in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd
To:    From:
plasma plasma   0.000  1.000     NA     NA   0.000   0.000      NA     NA     N
       capil   -2.695  1.402  1.302  0.000  -2.244   0.335   1.084  1.138 -0.01
capil  plasma   1.922  0.713  0.928  0.000   2.244  -0.335  -1.084  1.138 -0.01
       capil    0.000  1.000     NA     NA   0.000   0.000      NA     NA     N

> round( ftable( da$Conv[,,-(1:4)] ), 3 )

               in(t-f) sl(t-f) sd(t-f) in(sd) sl(sd)   sd=K LoA-lo LoA-up
To:    From:
plasma plasma    0.000   0.000      NA     NA     NA     NA     NA     NA
       capil    -2.244   0.335   1.084  1.138 -0.015  0.833 -2.095  2.833
capil  plasma    2.244  -0.335  -1.084  1.138 -0.015  0.833 -2.833  2.095
       capil     0.000   0.000      NA     NA     NA     NA     NA     NA

> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, wh.comp=2:1, pl.type="BA", dif.type="const",
+          sd.type="lin", eqn=TRUE )
```

```
Relationships between methods:
 capil-plasma = -0.37 (1.70-0.07Avg.)
 capil = -0.37+plasma (1.65-0.07plasma)
 plasma = 0.37+capil (-1.75+0.07capil)
```

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, wh.comp=2:1, pl.type="BA",
+          dif.type="lin", sd.type="lin", eqn=TRUE )
```

```
Relationships between methods:
 capil-plasma = 2.24-0.33(capil+plasma)/2 (1.14-0.02Avg.)
 capil = 1.92+0.71plasma (0.96-0.01plasma)
 plasma = -2.69+1.40capil (-1.40+0.02capil)
```

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> BA.plot( glu120, wh.comp=2:1, pl.type="comp",
+          dif.type="lin", sd.type="lin", eqn=TRUE )
```

```
Relationships between methods:
 capil-plasma = 2.24-0.33(capil+plasma)/2 (1.14-0.02Avg.)
 capil = 1.92+0.71plasma (0.96-0.01plasma)
 plasma = -2.69+1.40capil (-1.40+0.02capil)
```

Variable mean and standard deviation

- 2-step procedure:
    - Regress D_i on A_i.
    - Regress R_i (absolute residuals) on A_i
- Can be done using quadratic rather than linear terms, or even splines. (Not in `MethComp` — yet, any takers?)
- Allows very flexible form of the relationships between differences and averages
- —and flexible form of the $\mathrm{s.d.}$ to the mean.
- The relationship $D \sim A$ is easily back-transformed to a relationship $y_1 \sim y_2$, with prediction intervals.
- Beware: **over-modelling!**

Comparing two methods with replicate measurements

Bendix Carstensen

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

`(Comp-repl)`

Replicate measurements on each item

Fat data; **exchangeable** replicates:

```
item repl  KL  SL
   1    1 4.5 5.0
   1    2 4.7 4.9
   1    3 4.4 4.8
   3    1 6.4 6.5
   3    2 6.2 6.4
   3    3 6.5 6.1
```

Oximetry data; linked replicates:

```
item repl   CO pulse
   1    1 78.0    71
   1    2 76.4    72
   1    3 77.2    73
   2    1 68.7    68
   2    2 67.6    67
   2    3 68.3    68
```

Replicate measurements on each item

Fat data; exchangeable replicates:

```
item repl  KL  SL
   1    1 4.5 4.9
   1    2 4.4 5.0
   1    3 4.7 4.8
   3    1 6.4 6.5
   3    2 6.2 6.4
   3    3 6.5 6.1
```

Oximetry data; **linked** replicates:

```
item repl   CO pulse
   1    1 77.2    73
   1    2 78.0    71
   1    3 76.4    72
   2    1 68.7    68
   2    2 67.6    67
   2    3 68.3    68
```

Extension of the model: exchangeable replicates

$$\begin{aligned} y_{mir} &= \alpha_m + \mu_i + c_{mi} + e_{mir} \\ &\quad \text{s.d.}(c_{mi}) = \tau_m \quad \text{— “matrix”-effect} \\ &\quad \text{s.d.}(e_{mir}) = \sigma_m \quad \text{— measurement error} \end{aligned}$$

- Replicates within (m, i) is needed to separate τ and σ.
- Even with replicates, the τs are only estimable if $M > 2$.
- Still assumes that the difference between methods is constant.
- Assumes **exchangeability** of replicates.

Extension of the model: linked replicates

$$
\begin{aligned}
y_{mir} &= \alpha_m + \mu_i + a_{ir} + c_{mi} + e_{mir} \\
&\quad \text{s.d.}(a_{ir}) = \omega \quad \text{— between replicates} \\
&\quad \text{s.d.}(c_{mi}) = \tau_m \quad \text{— “matrix”-effect} \\
&\quad \text{s.d.}(e_{mir}) = \sigma_m \quad \text{— measurement error}
\end{aligned}
$$

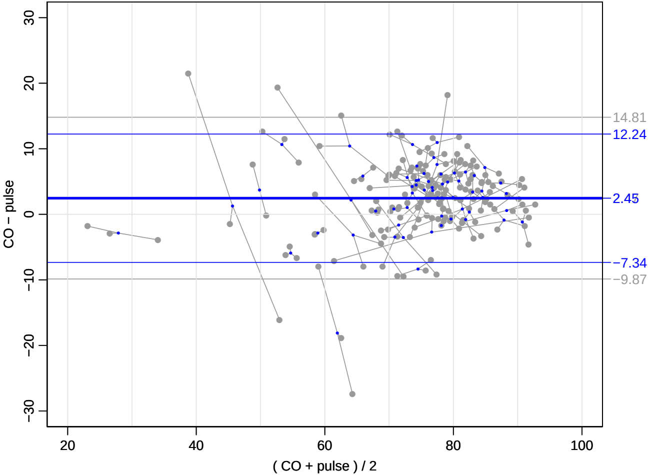
- Still assumes difference between methods constant.
- Replicates **linked** between methods: a_{ir} is common across methods; first replicate on a person is made under similar conditions for all methods, second too etc.

Replicate measurements

Three approaches to LoA with replicate measurements:

1. Means over replicates within each method by item stratum.
2. Replicates within item are taken as items.
3. Fit the model and use it for the LoA:
    - The model is a standard linear mixed model with separate variances per method.
    - The model is fitted using `BA.est(data,linked=TRUE)` — later.

Oximetry data



```
> library(MethComp)
> data( ox )
> ox <- Meth( ox, print=FALSE )
> 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
```

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( ox, pl.type="BA",
+       axlim=c(20,100), diflim=c(-30,30) )
```

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( ox, pl.type="BA", col.points=gray(0.5), repl.conn=TRUE,
+       axlim=c(20,100), diflim=c(-30,30), col.lines=gray(0.5) )
```

```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( ox, pl.type="BA", col.points=gray(0.6), repl.conn=TRUE,
+       axlim=c(20,100), diflim=c(-30,30), col.lines=gray(0.6) )
> par( new=TRUE )
> BA.plot( mean(ox), pl.type="BA", col.points="blue", cex=0.5,
+          axlim=c(20,100), diflim=c(-30,30) )
```

Replicate measurements

- The limits of agreement should still be for difference between future **single** measurements.
- Analysis based on the **means** of replicates is therefore **wrong**:
- If the model is:

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

- …then the correct limits of agreement are:

$$\alpha_1 - \alpha_2 \pm 2\sqrt{\tau_1^2 + \tau_2^2 + \sigma_1^2 + \sigma_2^2}$$

Wrong or almost right?

- $\text{var}(y_{1jr} - y_{2jr}) = \tau_1^2 + \tau_2^2 + \sigma_1^2 + \sigma_2^2$ — note that the term $a_{ir} - a_{ir}$ cancels because we are referring to the **same** replicate.
- If we are using means of replicates to form the differences we have:

$$\begin{aligned}\bar{d}_i &= \bar{y}_{1i\cdot} - \bar{y}_{2i\cdot} \\ &= \alpha_1 - \alpha_2 + \textstyle\sum_r a_{ir}/R_{1i} - \sum_r a_{ir}/R_{2i} \\ &\quad + c_{1i} - c_{2i} + \textstyle\sum_r e_{1ir}/R_{1i} - \sum_r e_{2ir}/R_{2i} \\ \Rightarrow & \\ \text{var}(\bar{d}_i) &= \tau_1^2 + \tau_2^2 + \sigma_1^2/R_{1i} + \sigma_2^2/R_{2i} \\ &< \tau_1^2 + \tau_2^2 + \sigma_1^2 + \sigma_2^2\end{aligned}$$

(Linked) replicates as items

- If replicates are taken as items, then the differences are:

$$d_{ir} = y_{1ir} - y_{2ir} = \alpha_1 - \alpha_2 + c_{1i} - c_{2i} + e_{1ir} - e_{2ir}$$

- which has variance $\tau_1^2 + \tau_2^2 + \sigma_1^2 + \sigma_2^2$, and so gives the correct limits of agreement.
- But the differences are not independent:

$$\mathrm{cov}(d_{ir}, d_{is}) = \tau_1^2 + \tau_2^2$$

- Negligible if the residual variances are very large compared to the interaction, variance likely to be only slightly downwards biased.

Exchangeable replicates as items?

- Exchangeable replicates: not clear how to produce the differences with replicates as items.
- If replicates are paired at random (se the function `perm.repl`), the variance will still be correct using the model without the $i \times r$ interaction term (a_{ir}):

$$\mathrm{var}(y_{1ir} - y_{2is}) = \tau_1^2 + \sigma_1^2 + \tau_2^2 + \sigma_2^2$$

- Differences will be positively correlated within item:

$$\mathrm{cov}(y_{1ir} - y_{2is}, y_{1it} - y_{2iu}) = \tau_1^2 + \tau_2^2$$

— slight underestimate of the true variance.

Recommendations

- Fit the correct model, and get the estimates from that, e.g. by using `BA.est`.
- If you must use over-simplified methods:
    - Use linked replicates as item.
    - If replicates are not linked; make a random linking.
    - Note: If this give a substantially different picture than using the original replicate numbering as linking key, there might be something fishy about the data.

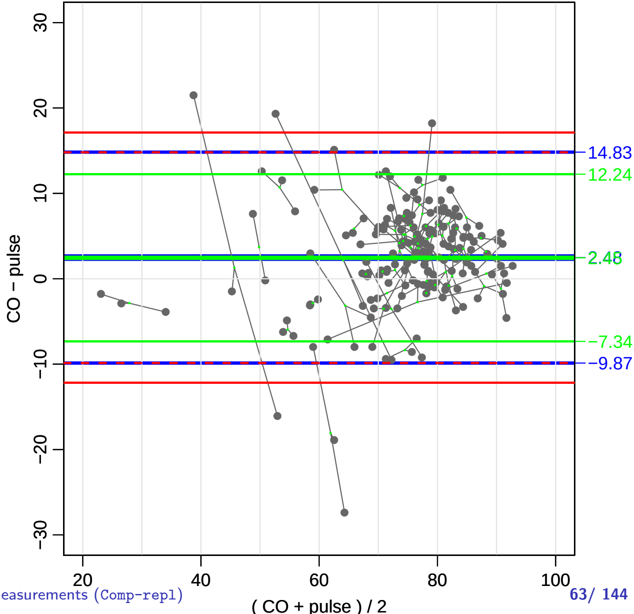
Further details, see [6].

Oximetry data (linked replicates)

Linked replicates used as items

Mean over replicates as items

Limits based on model — dashed line assuming linked, full exchangeable replicates



```
> ( ox.link <- BA.est( ox, linked=TRUE ) )

 Conversion between methods:
               alpha    beta   sd.pr  LoA-lo  LoA-up
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
CO    3.416 2.928 2.225
pulse 3.416 2.928 3.994

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

```
 Conversion between methods:
                 alpha    beta   sd.pr  LoA-lo  LoA-up
To:   From:
CO    CO         0.000   1.000   5.755 -11.509  11.509
      pulse      2.476   1.000   7.326 -12.175  17.127
pulse CO        -2.476   1.000   7.326 -17.127  12.175
      pulse      0.000   1.000   7.417 -14.835  14.835

 Variance components (sd):
      IxR   MxI   res
CO      0 2.191 4.069
pulse   0 2.191 5.245
```

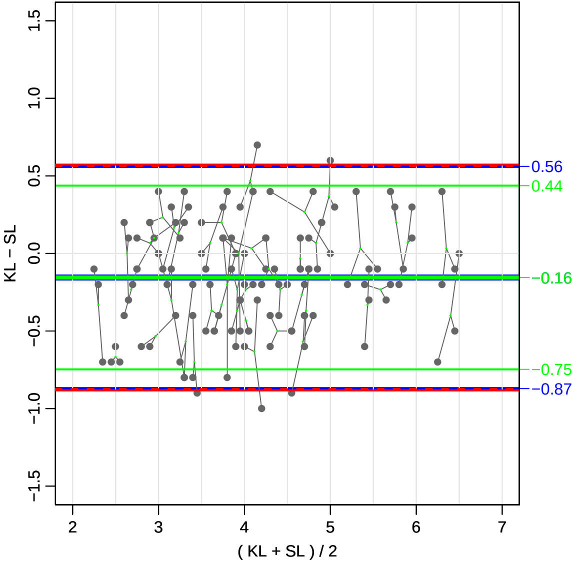
```
> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( ox, pl.type="BA", model=NULL,
+          col.points=gray(0.4), repl.conn=TRUE,
+          axlim=c(20,100), diflim=c(-30,30), col.lines="blue",
+          lwd=c(6,3,3) )
> par( new=TRUE )
> BA.plot( mean(ox), pl.type="BA", col.points="green",
+          cex.points=0.3, axlim=c(20,100), diflim=c(-30,30),
+          col.lines="green", lwd=c(4,2,2))
> abline( h=-ox.link[["LoA"]][2:3], col="red", lwd=2, lty=2 )
> abline( h=-ox.exch[["LoA"]][2:3], col="red", lwd=2, lty=1 )
```

Visceral fat data (exchangeable replicates)

Randomly paired replicates used as items

Mean over replicates as items

Limits based on model — dashed line assuming linked, full exchangeable replicates



```
> data( fat )
> 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

> ( vis.link <- BA.est( vis, linked=TRUE ) )

 Conversion between methods:
           alpha   beta  sd.pr LoA-lo LoA-up
To: From:
KL  KL     0.000  1.000  0.264 -0.528  0.528
    SL    -0.155  1.000  0.360 -0.874  0.564
SL  KL     0.155  1.000  0.360 -0.564  0.874
    SL     0.000  1.000  0.235 -0.471  0.471

 Variance components (sd):
     IxR   MxI   res
KL 0.048 0.183 0.187
SL 0.048 0.183 0.166
```

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

 Conversion between methods:
           alpha  beta sd.pr LoA-lo LoA-up
To: From:
KL  KL     0.000 1.000 0.273 -0.545  0.545
    SL    -0.155 1.000 0.364 -0.883  0.573
SL  KL     0.155 1.000 0.364 -0.573  0.883
    SL     0.000 1.000 0.245 -0.490  0.490

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

> par( mar=c(3,3,1,3), mgp=c(3,1,0)/1.6 )
> BA.plot( vis, pl.type="BA", model=NULL,
+          col.points=gray(0.4), repl.conn=TRUE,
+          axlim=c(2,7), diflim=c(-3,3)/2, col.lines="blue",
+          lwd=c(6,3,3) )
> par( new=TRUE )
> BA.plot( mean(vis), pl.type="BA", col.points="green",
+          cex.points=0.3, axlim=c(2,7), diflim=c(-3,3)/2,
+          col.lines="green", lwd=c(4,2,2))
> abline( h=-vis.link[["LoA"]][2:3], col="red", lwd=2, lty=2 )
> abline( h=-vis.exch[["LoA"]][2:3], col="red", lwd=2, lty=1 )
```

How the data is generated I

- A statistical model is a description of a machinery that may have generated data
- Illustrate how the various components make up the observed data.

```
> source("mc-ill.R")
> mc.ill
```

How the data is generated II

```
function (prefix, Nm = 2, Ni = 11, Nr = 3, alpha = c(-4, 7),
    beta = c(0.95, 1.05), sigma.ir = 5, sigma.mi = c(3, 5), sigma.mir = c(2,
        3))
{
    meth <- rep(1:Nm, Ni)
    item <- rep(1:Ni, each = Nm)
    reps <- rep(Nr, length(meth))
    dfr <- data.frame(meth = meth, item = item)[rep(1:length(meth),
        reps), ]
    dfr <- make.repl(dfr)
    dfr <- dfr[with(dfr, order(meth, item, repl)), ]
    mu <- runif(Ni, 15, 85)
    dfr$mu <- mu[dfr$item]
    dfr$alpha <- alpha[dfr$meth]
    dfr$beta <- beta[dfr$meth]
    e.ir <- rnorm(nlevels(IR <- with(dfr, interaction(item, repl))),
        mean = 0, sd = sigma.ir)
    dfr$e.ir <- e.ir[as.integer(IR)]
    e.mi <- rnorm(nlevels(MI <- with(dfr, interaction(meth, item))),
        mean = 0, sd = sigma.mi)
    dfr$e.mi <- e.mi[as.integer(MI)]
    dfr$e.mir <- rnorm(nrow(dfr), mean = 0, sd = sigma.mir[meth])
    dfr <- transform(dfr, y = alpha + beta * (mu + e.ir + e.mi) +
        e.mir, yrm = alpha + beta * (mu + e.ir + e.mi), yr = alpha +
        beta * (mu + e.ir), y0 = alpha + beta * mu)
    dfr
```

How the data is generated III

```
d1 <- subset(dfr, meth == 1)
d2 <- subset(dfr, meth == 2)
mu1 <- d1$mu
y10 <- d1$y0
y1r <- d1$yr
y1m <- d1$yrm
y1f <- d1$y
mu2 <- d2$mu
y20 <- d2$y0
y2r <- d2$yr
y2m <- d2$yrm
y2f <- d2$y
x <- 4
xx <- 1.7
clr <- rainbow(Ni)
pdf(paste("../graph/", prefix, "-ill-1.pdf", sep = ""), height = 2 *
    x + 2, width = 3 * x + 3, pointsize = 21)
layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
plot(mu1, y10, xlim = c(0, 100), ylim = c(0, 100), xlab = expression(mu),
    ylab = "y1", pch = 16, cex = xx, col = clr[d1$item])
abline(0, 1)
plot(mu2, y20, xlim = c(0, 100), ylim = c(0, 100), xlab = expression(mu),
    ylab = "y2", pch = 16, cex = xx, col = clr[d1$item])
abline(0, 1)
plot(y10, y20, xlim = c(0, 100), ylim = c(0, 100), xlab = "y1",
```

How the data is generated IV

```
    ylab = "y2", pch = 16, cex = xx, col = clr[d1$item])
abline(0, 1)
dev.off()
pdf(paste("../graph/", prefix, "-ill-2.pdf", sep = ""), height = 2 *
    x + 2, width = 3 * x + 3, pointsize = 21)
layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
plot(mu1, y10, xlim = c(0, 100), ylim = c(0, 100), col = clr[d1$item],
    xlab = expression(mu), ylab = "y1", pch = 1, lwd = 2,
    cex = xx)
segments(mu1, y10, mu1, y1r, col = grey(0.7))
points(mu1, y1r, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
plot(mu2, y20, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
    xlab = expression(mu), ylab = "y2", pch = 1, lwd = 2,
    cex = xx)
segments(mu2, y20, mu2, y2r, col = grey(0.7))
points(mu2, y2r, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
plot(y10, y20, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
    xlab = "y1", ylab = "y2", pch = 1, lwd = 2, cex = xx)
segments(y10, y20, y1r, y2r, col = clr[d1$item])
points(y1r, y2r, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
dev.off()
pdf(paste("../graph/", prefix, "-ill-3.pdf", sep = ""), height = 2 *
```

How the data is generated V

```
    x + 2, width = 3 * x + 3, pointsize = 21)
layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
plot(mu1, y10, xlim = c(0, 100), ylim = c(0, 100), col = clr[d1$item],
    xlab = expression(mu), ylab = "y1", pch = 1, lwd = 2,
    cex = xx)
segments(mu1, y10, mu1, y1r, col = clr[d1$item])
points(mu1, y1r, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(mu1, y1r, mu1, y1m, col = clr[d1$item])
points(mu1, y1m, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
plot(mu2, y20, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
    xlab = expression(mu), ylab = "y2", pch = 1, lwd = 2,
    cex = xx)
segments(mu2, y20, mu2, y2r, col = clr[d1$item])
points(mu2, y2r, col = clr[d1$item], pch = 16, cex = xx)
segments(mu2, y2r, mu2, y2m, col = clr[d1$item])
points(mu2, y2m, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
plot(y10, y20, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
    xlab = "y1", ylab = "y2", pch = 1, lwd = 2, cex = xx)
segments(y10, y20, y1r, y2r, col = clr[d1$item])
points(y1r, y2r, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(y1r, y2r, y1m, y2m, col = clr[d1$item])
points(y1m, y2m, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
```

How the data is generated VI

```
dev.off()
pdf(paste("../graph/", prefix, "-ill-4.pdf", sep = ""), height = 2 *
    x + 2, width = 3 * x + 3, pointsize = 21)
layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
plot(mu1, y10, xlim = c(0, 100), ylim = c(0, 100), col = clr[d1$item],
    xlab = expression(mu), ylab = "y1", pch = 1, lwd = 2,
    cex = xx)
segments(mu1, y10, mu1, y1r, col = clr[d1$item])
points(mu1, y1r, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(mu1, y1r, mu1, y1m, col = clr[d1$item])
points(mu1, y1m, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(mu1, y1m, mu1, y1f, col = clr[d1$item])
points(mu1, y1f, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
plot(mu2, y20, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
    xlab = expression(mu), ylab = "y2", pch = 1, lwd = 2,
    cex = xx)
segments(mu2, y20, mu2, y2r, col = clr[d1$item])
points(mu2, y2r, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(mu2, y2r, mu2, y2m, col = clr[d1$item])
points(mu2, y2m, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(mu2, y2m, mu2, y2m, col = clr[d1$item])
points(mu2, y2m, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
plot(y10, y20, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
```

How the data is generated VII

```
    xlab = "y1", ylab = "y2", pch = 1, lwd = 2, cex = xx)
segments(y10, y20, y1r, y2r, col = clr[d1$item])
points(y1r, y2r, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(y1r, y2r, y1m, y2m, col = clr[d1$item])
points(y1m, y2m, col = clr[d1$item], pch = 1, lwd = 2, cex = xx)
segments(y1m, y2m, y1f, y2f, col = clr[d1$item])
points(y1f, y2f, col = clr[d1$item], pch = 16, cex = xx)
abline(0, 1)
dev.off()
pdf(paste("../graph/", prefix, "-ill-5.pdf", sep = ""), height = 2 *
    x + 2, width = 3 * x + 3, pointsize = 21)
layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
plot(mu1, y1f, xlim = c(0, 100), ylim = c(0, 100), col = clr[d1$item],
    xlab = expression(mu), ylab = "y1", pch = 16, lwd = 2,
    cex = xx)
abline(0, 1)
plot(mu2, y2f, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
    xlab = expression(mu), ylab = "y2", pch = 16, lwd = 2,
    cex = xx)
abline(0, 1)
plot(y1f, y2f, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
    xlab = "y1", ylab = "y2", pch = 16, lwd = 2, cex = xx)
abline(0, 1)
```

How the data is generated VIII

```
        dev.off()
        pdf(paste("../graph/", prefix, "-ill-6.pdf", sep = ""), height = 2 *
            x + 2, width = 3 * x + 3, pointsize = 21)
        layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
        par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
        layout(matrix(c(1, 2, 3, 3, 3, 3), 2, 3))
        par(mai = c(3, 3, 1, 1)/4, mgp = c(3, 1, 0)/1.6)
        plot(mu1, y1f, xlim = c(0, 100), ylim = c(0, 100), col = clr[d1$item],
            xlab = expression(mu), ylab = "y1", pch = 16, lwd = 2,
            cex = xx)
        abline(0, 1)
        plot(mu2, y2f, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
            xlab = expression(mu), ylab = "y2", pch = 16, lwd = 2,
            cex = xx)
        abline(0, 1)
        plot(y1f, y2f, xlim = c(0, 100), ylim = c(0, 100), , col = clr[d1$item],
            xlab = "y1", ylab = "y2", pch = 16, lwd = 2, cex = xx)
        abline(0, 1)
        abline(alpha[2] - alpha[1] * beta[2]/beta[1], beta[2]/beta[1],
            lwd = 3, col = gray(0.6))
        dev.off()
    }
```

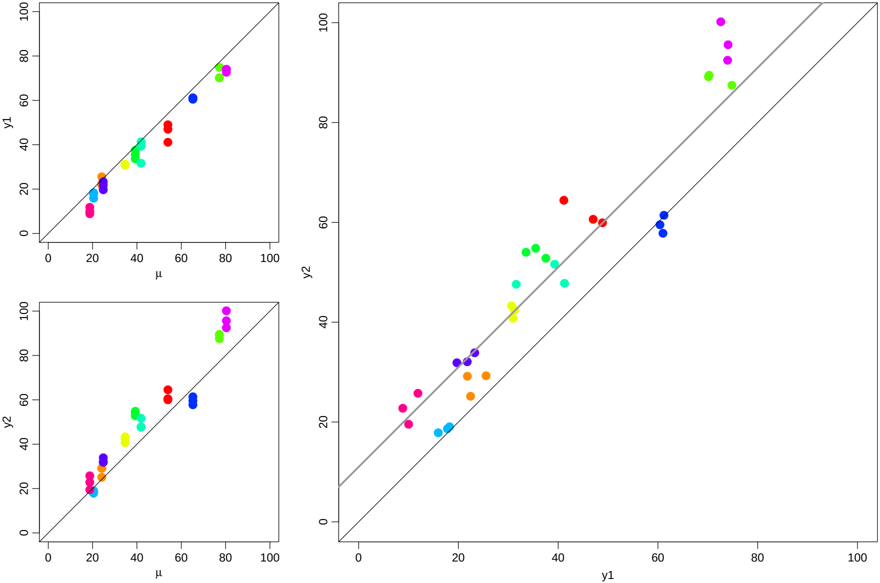
```
> library( MethComp )
> mc.ill("vcx",beta=c(1,1),sigma.ir=0)
```

How the data is generated IX

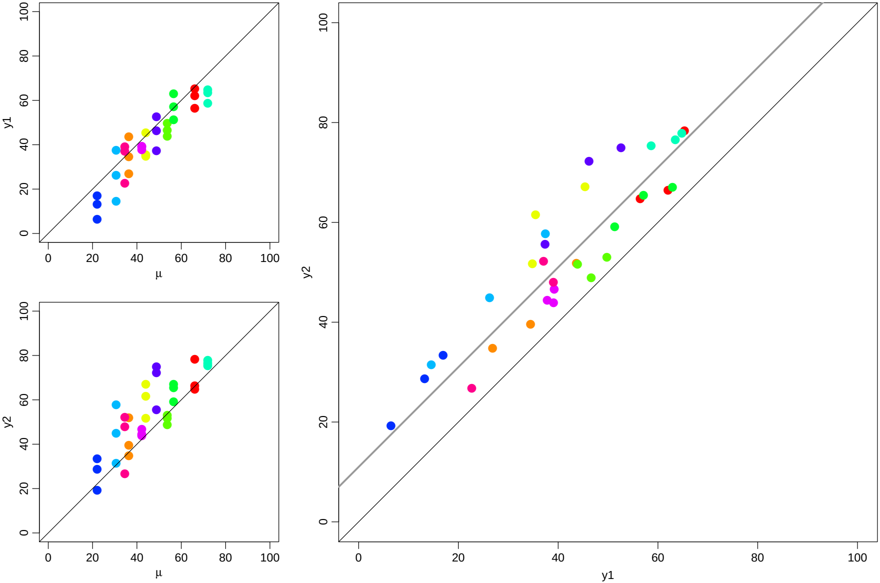
```
null device
          1
```

```
> mc.ill("vcl",beta=c(1,1),sigma.ir=5)
```

```
null device
          1
```



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



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

Repeatability and reproducibility

Bendix Carstensen

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

(`Repro`)

Accuracy of a measurement method

(ISO 5625)

- Repeatability:
  The accuracy of the method under exactly similar circumstances; i.e. the same lab, the same technician, and the same day.
  (**Repeata**bility conditions)
- Reproducibility:
  The accuracy of the method under comparable circumstances, i.e. the same machinery, the same kit, but possibly different days or laboratories or technicians.
  (**Reproduci**bility conditions)

Quantification of accuracy

- Upper limit of a 95% confidence interval for the difference between two measurements.
- Suppose the variance of the measurement is σ^2:

$$\text{var}(y_{mi1} - y_{mi2}) = 2\sigma^2$$

— standard error of difference: $\sqrt{2}\sigma$

- Confidence interval for the difference:

$$0 \pm 1.96 \times \sqrt{2}\sigma = 0 \pm 2.772\sigma \approx \pm 2.8\sigma$$

- This is called the reproducibility coefficient or simply the **reproducibility**.
  (2.8 is used as a convenient approximation).

Quantification of accuracy

- Where do we get the σ?
- Repeat measurements on the same item.
- The conditions under which the repeat (replicate) measurements are taken determines whether we are estimating repeatability or reproducibility.
- In larger experiments we must consider the **exchangeability** of the replicates — i.e. which replicates are done under (exactly) similar conditions and which are not.

Coefficient of variation

- Defined as $\mathrm{s.d.}$ relative to mean: $\mathrm{CV} = \sigma/\mu$
- Measurements with varying mean and $\mathrm{s.d.}$ may still have constant CV.
- Assumption of $\mathrm{s.d.}$ proportional to μ across the range of y, $\mathrm{s.d.}(y) = \mathrm{CV}\mu(y)$
  — implies that measurements are positive.
- LoA could be:

$$\mu \pm 2\mathrm{CV}\mu$$

- But what if $\mathrm{CV} > 0.5$ — lower bound < 0?
- Immaterial — "2" depends on the degree of confidence chosen anyway.

Coefficient of variation

- σ proportional to μ
- $\Rightarrow$ confidence intervals should be multiplicative: $\mu \overset{\times}{\div} \mathrm{erf}$ for some error-factor.
- Specifically:

$$\mathrm{s.d.}\big(\log(Y)\big) \approx \sigma \times \left.\frac{\mathrm{d}\log(y)}{\mathrm{d}y}\right|_{y=\mu} = \sigma/\mu = \mathrm{CV}$$

- …so using CV is just doing analysis on the log-scale.

Coefficient of variation

- CV small:
  CV is the same as the $\mathrm{s.d.}$ of the log-transformed data.
- CV large:
  CV is **not** same as the $\mathrm{s.d.}$ of the log-transformed data.
- . . . but it is the log-transformed analysis that is meaningful.
- Empirical question if this gives a better model.

A common misconception

> *There are other approaches that might also be used (e.g., coefficients of variation, item response theory, or the "signal to noise ratio"). [7]* [1]

- The authors seem to think that coefficient of variation is another **model**.
- It is not a different model — just the same model on a transformed **scale**,
- — focusing on the variance (of the log-transformed data)

---

[1] Guidelines for Reporting Reliability and Agreement Studies (GRRAS)

Linear bias between methods

Bendix Carstensen

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

(`Lin-bias`)

Extension with non-constant bias

$$y_{mir} = \alpha_m + \beta_m \mu_i + \text{random effects}$$

- There is now a **scaling** between the methods.
- Methods do not measure on the same scale — the relative scaling is **estimated**, between method 1 and 2 the scale is β_2/β_1.
- Consequence: Multiplication of all measurements on one method by a fixed number does not change results of analysis:
  - The αs & βs are multiplied by the same factor
  - as is the $\mathrm{s.d.}$s of the variance components for this method.

Variance components

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

- The random effects c_{mi} and e_{mir} have variances specific for each method.
- Variance of a_{ir} does not depend on m — reporting scaled to each of the methods by the corresponding β_m.
- Implies that $\omega = \mathrm{s.d.}(a_{ir})$ is irrelevant — the scale is arbitrary.
- Relevant quantities are $\beta_m\omega$ — the between replicate variation within item **as measured on the mth scale**.

Variance components

Method, Item, Replicate.

$$
\begin{aligned}
y_{mir} &= \alpha_m + \beta_m(\mu_i + a_{ir} + c_{mi}) + e_{mir} \\
\text{s.d.}(c_{mi}) &= \tau_m
\end{aligned}
$$

- Matrix-effect: Each item reacts differently to each method.
- If only two methods:
  - τ_1 and τ_2 cannot be separated.
  - Variances must be reported on the scale of each method, as $\beta_m \tau_m$.

Variance components

Method, Item, Replicate.

$$y_{mir} = \alpha_m + \beta_m(\mu_i + a_{ir} + c_{mi}) + e_{mir}$$
$$\text{s.d.}(a_{ir}) = \omega$$

- Common across methods — must be scaled relative to the methods.
- Included if replicates are linked across methods, e.g. if there is a sequence in the replicates.
- a_{ir} nuisance parameters — $(\mu_i + a_{ir})$ is the "true" value underlying measurements y_{mir}.

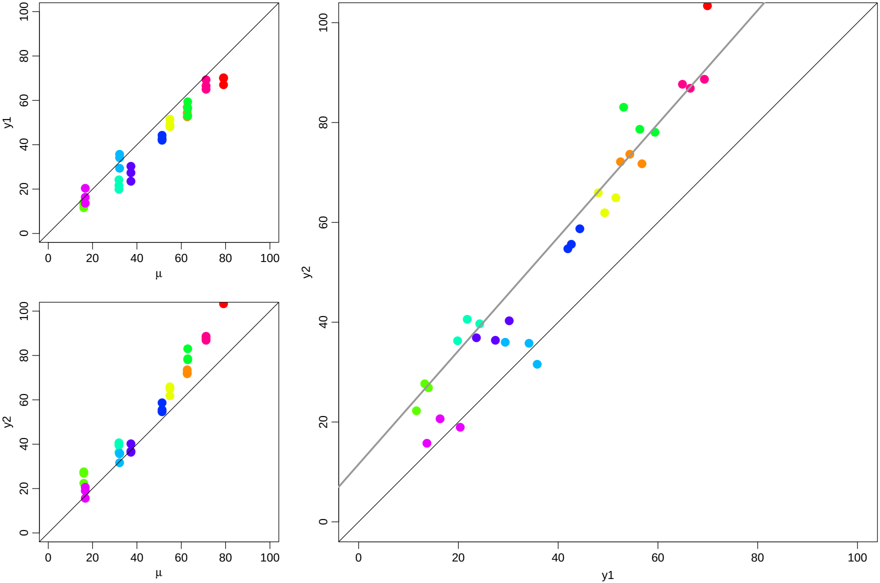
Estimation in the extended model

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

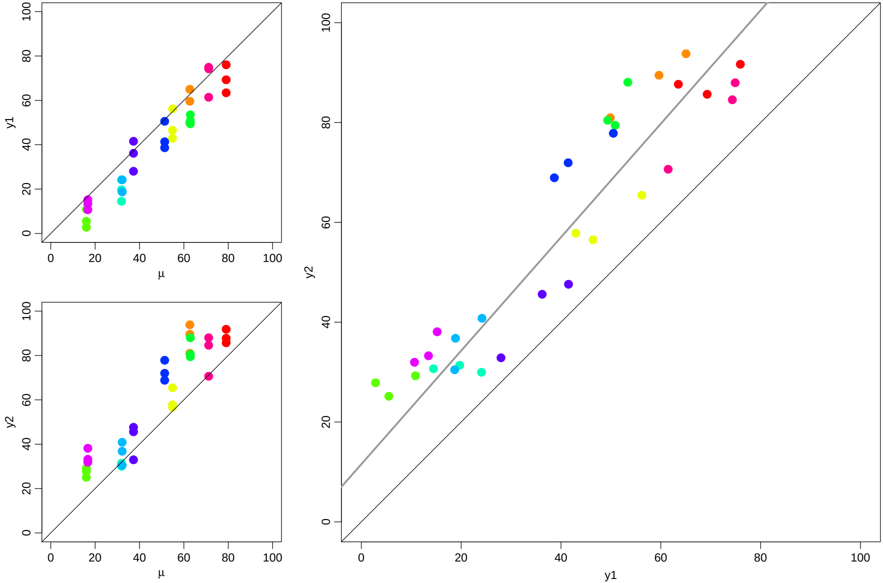
- Not a standard linear mixed model.
- Does not fit into usual software.
- Fitted in `BUGS`, using JAGS via `MCmcmc`.
- …or `AltReg` — we shall return to this later

How the data is generated I

- A statistical model is a description of a machinery that may have generated data
- Illustrate how the various components make up the observed data.



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



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

g

Converting between methods

Bendix Carstensen

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

(`Convert`)

Predicting method 2 from method 1

$$\begin{aligned}
y_{10r} &= \alpha_1 + \beta_1(\mu_0 + a_{0r} + c_{10}) + e_{10r} \\
y_{20r} &= \alpha_2 + \beta_2(\mu_0 + a_{0r} + c_{20}) + e_{20r} \\
& \Downarrow \\
y_{20r} &= \alpha_2 + \frac{\beta_2}{\beta_1}(y_{10r} - \alpha_1 - e_{10r}) \\
& \qquad + \beta_2(-c_{10} + c_{20}) + e_{20r}
\end{aligned}$$

The random effects have expectation 0, so:

$$\mathrm{E}(y_{20}|y_{10}) = \hat{y}_{20} = \alpha_2 + \frac{\beta_2}{\beta_1}(y_{10} - \alpha_1)$$

- Intercept: $\alpha_{2|1} = \alpha_2 - \alpha_1 \dfrac{\beta_2}{\beta_1}$
- Slope: $\beta_{2|1} = \dfrac{\beta_2}{\beta_1}$
- Invariant under linear transform of μ:

$$a + b\mu_i \rightarrow \tilde{\mu}_i \quad \Rightarrow \quad \alpha_m + \beta_m \mu_i \rightarrow \tilde{\alpha}_m + \tilde{\beta}_m \tilde{\mu}_i$$

$$\text{where:} \quad \tilde{\alpha}_m = \alpha_m - a\beta_m/b, \quad \tilde{\beta}_m = \beta_m/b$$

- $\Rightarrow$ the conversion is invariant too:

$$\alpha_{2|1} = \tilde{\alpha}_2 - \tilde{\alpha}_1 \frac{\tilde{\beta}_2}{\tilde{\beta}_1}$$

$$\beta_{2|1} = \frac{\tilde{\beta}_2}{\tilde{\beta}_1}$$

$$\begin{aligned} y_{20r} &= \alpha_2 + \frac{\beta_2}{\beta_1}(y_{10r} - \alpha_1 - e_{10r}) \\ &\qquad + \beta_2(-c_{10} + c_{20}) + e_{20r} \\ \mathrm{var}(\hat{y}_{20}|y_{10}) &= \left(\frac{\beta_2}{\beta_1}\right)^2(\beta_1^2\tau_1^2 + \sigma_1^2) + (\beta_2^2\tau_2^2 + \sigma_2^2) \end{aligned}$$

The prediction s.d. is:

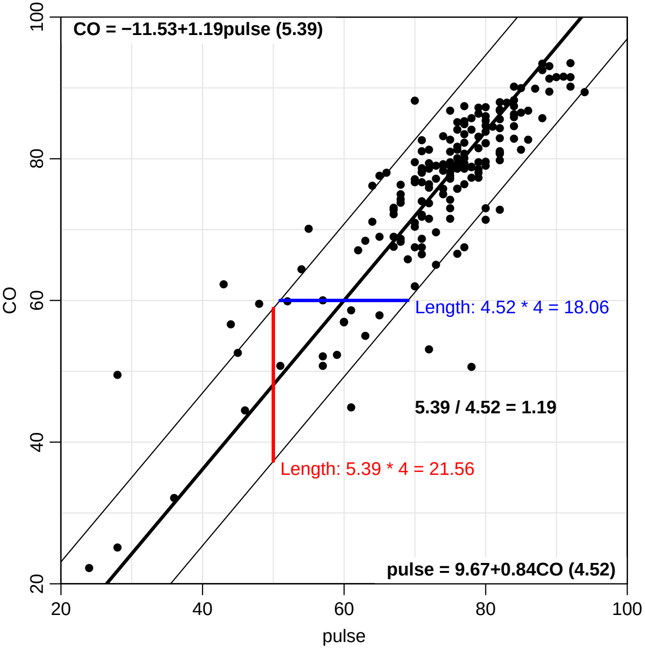
$$\sigma_{2|1} = \sqrt{\left(\frac{\beta_2}{\beta_1}\right)^2(\beta_1^2\tau_1^2 + \sigma_1^2) + (\beta_2^2\tau_2^2 + \sigma_2^2)}$$

If we do the prediction the other way round $(y_1|y_2)$ we get the same relationship i.e. a line with the inverse slope, β_1/β_2.

The width of the prediction interval in this direction is (by permutation of indices):

$$\begin{aligned}\sigma_{1|2} &= \sqrt{(\beta_1^2\tau_1^2+\sigma_1^2)+\left(\frac{\beta_1}{\beta_2}\right)^2(\beta_2^2\tau_2^2+\sigma_2^2)} \\ &= \frac{\beta_1}{\beta_2}\sqrt{\left(\frac{\beta_2}{\beta_1}\right)^2(\beta_1^2\tau_1^2+\sigma_1^2)+(\beta_2^2\tau_2^2+\sigma_2^2)} = \frac{\beta_1}{\beta_2}\sigma_{2|1}\end{aligned}$$

i.e. if we draw the prediction limits as straight lines they can be used both ways.


CO = −11.53+1.19pulse (5.39)
Length: 4.52 * 4 = 18.06
5.39 / 4.52 = 1.19
Length: 5.39 * 4 = 21.56
pulse = 9.67+0.84CO (4.52)
CO
20
40
60
80
100
pulse
20
40
60
80
100

```
> options( width=61 )
> library(MethComp)
> 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

> system.time( MCox <- MCmcmc( ox, IxR=TRUE ) )
```

```
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 2000 iterations
  (of which 1000 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 4000 observations.

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

Initializing model

Sampling:
   user  system elapsed
  13.94    0.07   14.45
```

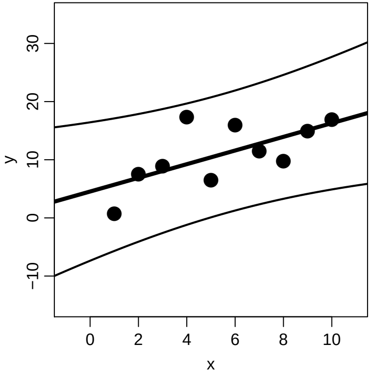
```
> ( Mox <- MethComp( MCox ) )


 Conversion between methods:
                alpha    beta   sd.pr in(t-f) sl(t-f) sd(t-f)
To:    From:
CO     CO       0.000   1.000   1.963   0.000   0.000   1.963
       pulse  -11.531   1.192   5.390 -10.519   0.175   4.917
pulse  CO       9.671   0.839   4.515  10.519  -0.175   4.911
       pulse    0.000   1.000   6.108   0.000   0.000   6.108

 Variance components (sd):
       s.d.
Method     IxR   MxI   res
  CO     3.861 3.311 1.388
  pulse  3.212 2.774 4.319


> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( Mox, points=TRUE, axlim=c(20,100), xaxs="i", yaxs="i" )

Relationships between methods:
 CO-pulse = -10.52+0.18(CO+pulse)/2 (4.92)
 CO = -11.53+1.19pulse (5.39)
 pulse = 9.67+0.84CO (4.52)


> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( Mox, points=TRUE, axlim=c(20,100), xaxs="i", yaxs="i" )
```

```
Relationships between methods:
 CO-pulse = -10.52+0.18(CO+pulse)/2 (4.92)
 CO = -11.53+1.19pulse (5.39)
 pulse = 9.67+0.84CO (4.52)
```

```
> segments( 50, Mox$Conv["CO","pulse","alpha"] +
+               Mox$Conv["CO","pulse","beta" ]*50 -
+               Mox$Conv["CO","pulse","sd.pr"]*2,
+           50, Mox$Conv["CO","pulse","alpha"] +
+               Mox$Conv["CO","pulse","beta" ]*50 +
+               Mox$Conv["CO","pulse","sd.pr"]*2,
+           col="red", lwd=3 )
> text( 51, Mox$Conv["CO","pulse","alpha"] +
+           Mox$Conv["CO","pulse","beta" ]*50 -
+           Mox$Conv["CO","pulse","sd.pr"]*2.02,
+       paste( "Length:", formatC(Mox$Conv["CO","pulse","sd.pr"],
+                                 format="f", digits=2),
+              "* 4 =", formatC(Mox$Conv["CO","pulse","sd.pr"]*4,
+                                 format="f", digits=2) ),
+       col="red", adj=c(0,1) )
> segments( Mox$Conv["pulse","CO","alpha"] +
+           Mox$Conv["pulse","CO","beta" ]*60 -
+           Mox$Conv["pulse","CO","sd.pr"]*2, 60,
+           Mox$Conv["pulse","CO","alpha"] +
+           Mox$Conv["pulse","CO","beta" ]*60 +
+           Mox$Conv["pulse","CO","sd.pr"]*2, 60,
+           col="blue", lwd=3 )
> text( Mox$Conv["pulse","CO","alpha"] +
+       Mox$Conv["pulse","CO","beta" ]*60 +
```

```
+       Mox$Conv["pulse","CO","sd.pr"]*2 + 1, 60,
+       paste( "Length:", formatC(Mox$Conv["pulse","CO","sd.pr"],
+                                 format="f", digits=2),
+              "* 4 =", formatC(Mox$Conv["pulse","CO","sd.pr"]*4,
+                                 format="f", digits=2) ),
+       col="blue", adj=c(0,1) )
> text( 70, 45, paste( formatC( Mox$Conv["CO","pulse","sd.pr"],
+                               format="f", digits=2 ), "/",
+                      formatC( Mox$Conv["pulse","CO","sd.pr"],
+                               format="f", digits=2 ), "=",
+                      formatC( Mox$Conv["CO","pulse","beta"],
+                               format="f", digits=2 ) ),
+       adj=0, font=2 )
```

What happened to the curvature?



Usually the prediction limits are curved:

$$\hat{y}|x \pm 1.96 \times \hat{\sigma}\sqrt{1 + x'x}$$

In our prediction we have ignored the last term $(x'x)$, i.e. effectively assuming that there is no estimation error on $\alpha_{2|1}$ and $\beta_{2|1}$.

```
> set.seed(17676)
> par(mar=c(3,3,1,1),mgp=c(3,1,0)/1.6)
> x <- 1:10
> y <- 3 + 1.6*x + rnorm(x,,6)
> m0 <- lm(y~x)
> plot(y~x,pch=16,ylim=c(-15,35),xlim=c(-1,11),cex=2)
> nx <- seq(-3,13,,200)
> matlines( nx, predict( m0, interval="pred", newdata=data.frame(x=nx)),
+           lwd=c(4,2,2), col="black", lty=1 )

> # The same but now with 100 points
> set.seed(17676)
> par(mar=c(3,3,1,1),mgp=c(3,1,0)/1.6)
> x <- seq(1,10,,100)
> y <- 3 + 1.6*x + rnorm(x,,6)
> m0 <- lm(y~x)
> plot(y~x,pch=16,ylim=c(-15,35),xlim=c(-1,11),cex=0.7)
> nx <- seq(-3,13,,200)
> matlines( nx, predict( m0, interval="pred", newdata=data.frame(x=nx)),
+           lwd=c(4,2,2), col="black", lty=1 )
```

Comparing to a gold standard

- The prediction $\mathrm{s.d.}$ is:

$$\sigma_{2|1} = \sqrt{\left(\frac{\beta_2}{\beta_1}\right)^2 (\beta_1^2\tau_1^2 + \sigma_1^2) + (\beta_2^2\tau_2^2 + \sigma_2^2)}$$

- If method 1 is the gold standard (no error), *i.e.* **assumed**: $\tau_1 = \sigma_1 = 0$
- Estimate relationship by regressing y_2 on y_1, deriving τ_2 and σ_2 — standard linear regresssion.
- Prediction of y_1 (what would the gold standard give?):
- Limits for $y_2|y_1$, but used the other way.

Implementation in BUGS/JAGS

Bendix Carstensen

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

(`BUGS-impl`)

Implementation in `BUGS`

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

Non-linear hierarchical model:

- The model is *symmetrical* in methods.
- Mean is overparametrized.
- Choose a prior (and hence posterior!) for the μs with finite support.
- Keeps the chains nicely in place.

This is the philosophy in the function `MCmcmc`.

Results from fitting the model

The posterior dist'n of $(\alpha_m, \beta_m, \mu_i)$ is singular.

But the relevant translation quantities **are** identifiable:

$$\begin{aligned}\alpha_{2|1} &= \alpha_2 - \alpha_1\beta_2/\beta_1\\ \beta_{2|1} &= \beta_2/\beta_1\end{aligned}$$

— so are the variance components.

Posterior medians used to devise prediction equations with limits.

Implemented model:

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

- Replicates required in data.
- **JAGS** (or `R2WinBUGS` or `BRUGS`) is required.
- Dataframe with variables `meth`, `item`, `repl` and `y` (a `Meth` object)
- The function `MCmcmc` writes a `BUGS`-program, initial values and data to files.
- Runs **JAGS** and sucks results back in to **R**, and gives a nice overview of the conversion equations.

```
> options( width=61 )
> library(MethComp)
> 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

> system.time( MCox <- MCmcmc( ox, IxR=TRUE, n.iter=10000 ) )
```

```
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 10000 iterations
  (of which 5000 are burn-in),
- monitoring every 5 values of the chain:
- giving a posterior sample of 4000 observations.

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

Initializing model

Sampling:
   user  system elapsed
  69.49    0.09   69.93
```

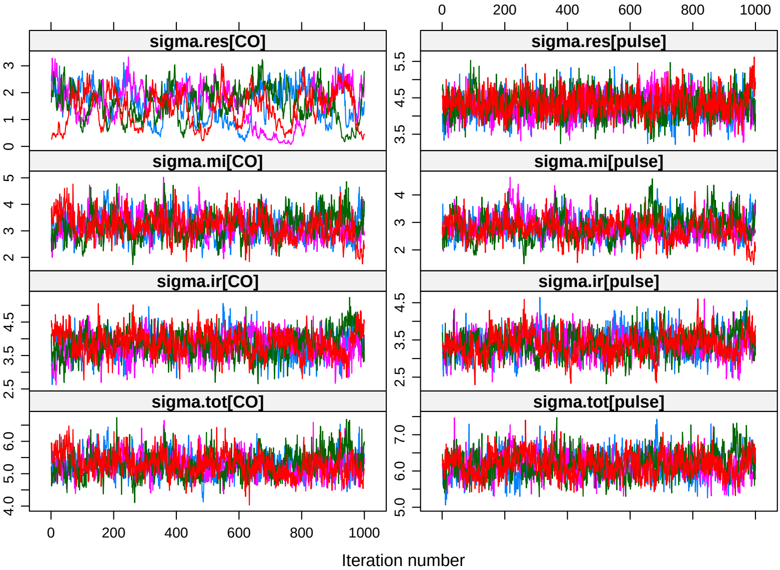
```
> MCox

 Conversion between methods:
                alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:   From:
CO    CO        0.000  1.000  2.316   0.000   0.000   2.316
      pulse    -6.933  1.129  5.124  -6.513   0.121   4.813
pulse CO        6.140  0.886  4.544   6.513  -0.121   4.819
      pulse     0.000  1.000  6.050   0.000   0.000   6.050

 Variance components (sd):
      s.d.
Method    IxR   MxI   res
  CO    3.824 3.155 1.638
  pulse 3.377 2.798 4.278

Variance components with 95 % cred.int.:
    method    CO                 pulse
    qnt      50%  2.5% 97.5%   50%  2.5% 97.5%
SD
IxR        3.824 3.074 4.546 3.377 2.741 4.054
MxI        3.155 2.323 4.150 2.798 2.024 3.760
res        1.638 0.298 2.697 4.278 3.631 5.005
tot        5.260 4.632 6.037 6.169 5.541 6.841

Mean parameters with 95 % cred.int.:
                    50%    2.5%  97.5% P(>0/1)
alpha[pulse.CO]  6.144  -2.900 13.632   0.918
alpha[CO.pulse] -6.928 -17.274  2.921   0.082
```

```
beta[pulse.CO]   0.886   0.788  1.003   0.028
beta[CO.pulse]   1.129   0.997  1.270   0.972

 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.
```

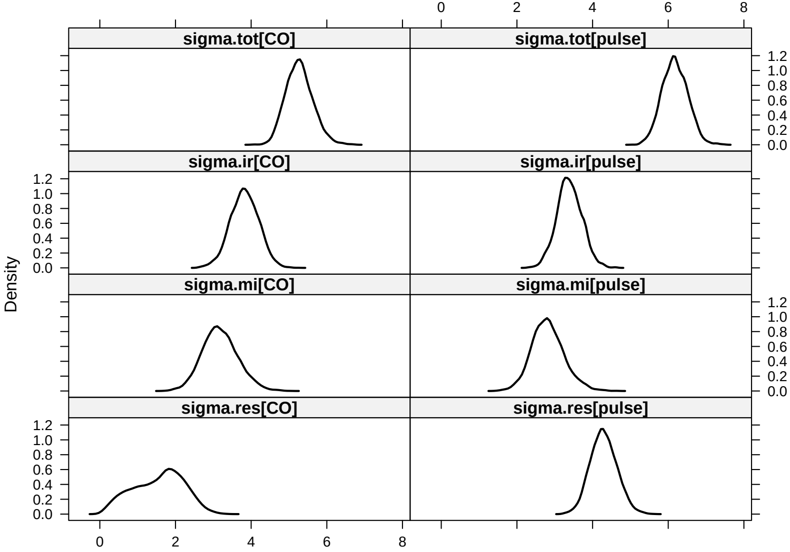
```
> MethComp( MCox )
```

```
 Conversion between methods:
             alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:   From:
CO    CO      0.000  1.000  2.316   0.000   0.000   2.316
      pulse  -6.933  1.129  5.124  -6.513   0.121   4.813
pulse CO      6.140  0.886  4.544   6.513  -0.121   4.819
      pulse   0.000  1.000  6.050   0.000   0.000   6.050

 Variance components (sd):
      s.d.
Method    IxR   MxI   res
  CO    3.824 3.155 1.638
  pulse 3.377 2.798 4.278
```

Traces of the chains



Posteriors for variance components



```
> trace.MCmcmc( MCox )

> post.MCmcmc( MCox )

> post.MCmcmc( MCox, check=FALSE )
```

```
> 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

> system.time( MCbp <- MCmcmc( sbp, IxR=TRUE, n.iter=10000 ) )
```

```
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 10000 iterations
  (of which 5000 are burn-in),
- monitoring every 5 values of the chain:
- giving a posterior sample of 4000 observations.

Initialization and burn-in:
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 5982

Initializing model

Sampling:
```

```
   user  system elapsed
 179.57    0.22  180.22

> MCbp

 Conversion between methods:
            alpha    beta   sd.pr in(t-f) sl(t-f) sd(t-f)
To: From:
J   J       0.000   1.000   2.173   0.000   0.000   2.173
    R      -1.143   1.010   2.293  -1.137   0.010   2.282
    S     -50.444   1.246  24.899 -44.929   0.219  22.176
R   J       1.132   0.990   2.271   1.137  -0.010   2.282
    R       0.000   1.000   2.374   0.000   0.000   2.374
    S     -48.832   1.234  24.689 -43.720   0.209  22.104
S   J      40.501   0.803  20.008  44.929  -0.219  22.196
    R      39.577   0.810  20.019  43.720  -0.209  22.114
    S       0.000   1.000  28.242   0.000   0.000  28.242

 Variance components (sd):
      s.d.
Method   IxR    MxI   res
     J 5.992  0.316 1.482
     R 5.935  0.184 1.658
     S 4.804 17.860 8.923

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          5.992  5.379  6.719  5.935  5.331  6.650  4.804  4.082  5.698
MxI          0.316  0.009  0.840  0.184  0.025  0.635 17.860 15.163 21.274
res          1.482  0.765  2.023  1.658  1.016  2.125  8.923  8.011 10.024
tot          6.191  5.573  6.897  6.176  5.570  6.866 20.551 18.270 23.607

Mean parameters with 95 % cred.int.:
                 50%    2.5%   97.5% P(>0/1)
alpha[R.J]    1.132  -0.209   2.332   0.952
alpha[S.J]   40.477  27.049  53.826   1.000
alpha[J.R]   -1.143  -2.375   0.209   0.048
alpha[S.R]   39.561  25.983  52.959   1.000
alpha[J.S]  -50.474 -76.857 -30.134   0.000
alpha[R.S]  -48.852 -74.837 -28.416   0.000
beta[R.J]     0.990   0.981   1.001   0.033
beta[S.J]     0.803   0.699   0.905   0.000
beta[J.R]     1.010   0.999   1.019   0.967
beta[S.R]     0.810   0.706   0.913   0.000
beta[J.S]     1.246   1.105   1.430   1.000
beta[R.S]     1.234   1.095   1.415   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.


> MethComp( MCbp )
```

```
 Conversion between methods:
            alpha    beta   sd.pr in(t-f) sl(t-f) sd(t-f)
To: From:
J   J       0.000   1.000   2.173   0.000   0.000   2.173
    R      -1.143   1.010   2.293  -1.137   0.010   2.282
    S     -50.444   1.246  24.899 -44.929   0.219  22.176
R   J       1.132   0.990   2.271   1.137  -0.010   2.282
    R       0.000   1.000   2.374   0.000   0.000   2.374
    S     -48.832   1.234  24.689 -43.720   0.209  22.104
S   J      40.501   0.803  20.008  44.929  -0.219  22.196
    R      39.577   0.810  20.019  43.720  -0.209  22.114
    S       0.000   1.000  28.242   0.000   0.000  28.242

 Variance components (sd):
     s.d.
Method   IxR    MxI   res
     J 5.992  0.316 1.482
     R 5.935  0.184 1.658
     S 4.804 17.860 8.923
```

Alternating regressions

Bendix Carstensen

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

(`Alt-reg`)

Alternating random effects regression

Carstensen [3] proposed a ridiculously complicated approach to fit the model

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

based in the observation that:

- For fixed μ the model is a linear mixed model.
- For fixed (α, β) it is a regression through 0.

This has be improved by Carstensen in [4]

Alternating random effects regression

The correctly formulated version of the slightly more general model:

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

- For fixed $\zeta_{mir} = \mu_i + a_{ir} + c_{mi}$ the model is a linear model, with residual variances different between methods.
- For fixed (α, β) scaled responses y follow a standard mixed model:

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

Estimation algorithm

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

1. Start with $\zeta_{mir} = \bar{y}_{mi\cdot}$
2. Estimate (α_m, β_m).
3. Compute the scaled responses and fit the random effects model.
4. Use the estimated μ_is, and BLUPs of a_{ir} and c_{mi} to update ζ_{mir}.
5. Check convergence in terms of identifiable parameters.

The residual variances

- The variance components are estimated in the model for the scaled response.
- The estimation of parameters (α_m, β_m) are not taken into account in the calculation of the residual variance d.f.
- Hence the residual variances must be corrected *post hoc*.
- This machinery is implemented in the function `AltReg` in the `MethComp` package.

```
> options( width=100 )
> library(MethComp)
> 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

> system.time( AR.ox <- AltReg( ox, linked=T, 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   re
CO    -10.562 1.148 1.071        74.419 76.680     1.000 1.051 3.502 2.982 2.08
pulse  -8.576 1.092 3.269        72.269 74.419     0.951 1.000 3.331 2.837 4.06

iteration 9 criterion: 0.005074459
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   re
CO    -12.190 1.169 1.057        74.419 76.768     1.000 1.055 3.508 2.980 2.07
pulse  -9.904 1.109 3.282        72.193 74.419     0.948 1.000 3.325 2.824 4.06

iteration 10 criterion: 0.003705422
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   re
CO    -13.826 1.191 1.047        74.419 76.830     1.000 1.058 3.513 2.978 2.06
pulse -11.290 1.126 3.292        72.140 74.419     0.945 1.000 3.321 2.816 4.07

iteration 11 criterion: 0.002686236
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   re
CO    -15.476 1.213 1.039        74.419 76.873     1.000  1.06 3.516 2.978 2.06
pulse -12.727 1.145 3.298        72.104 74.419     0.944  1.00 3.318 2.810 4.07

iteration 12 criterion: 0.001930191
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   re
CO    -17.144 1.236 1.034        74.419 76.903     1.000 1.061 3.518 2.978 2.06
```

```
pulse -14.211 1.165 3.303        72.079 74.419     0.942 1.000 3.315 2.807 4.07

iteration 13 criterion: 0.001381194
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   re
CO    -18.834 1.258 1.030        74.419 76.924     1.000 1.062 3.520 2.978 2.05
pulse -15.736 1.185 3.306        72.061 74.419     0.941 1.000 3.314 2.804 4.07

iteration 14 criterion: 0.0009863462
        alpha  beta sigma Intercept: CO  pulse Slope: CO pulse   IxR   MxI   re
CO    -20.548 1.281 1.027        74.419 76.938     1.000 1.063 3.521 2.978 2.05
pulse -17.301 1.205 3.308        72.049 74.419     0.941 1.000 3.313 2.802 4.07

AltReg converged after  14 iterations
Last convergence criterion was  0.0009863462
   user  system elapsed
  12.71    0.03   12.78

> AR.ox
```

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

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

Transformation of data

Bendix Carstensen

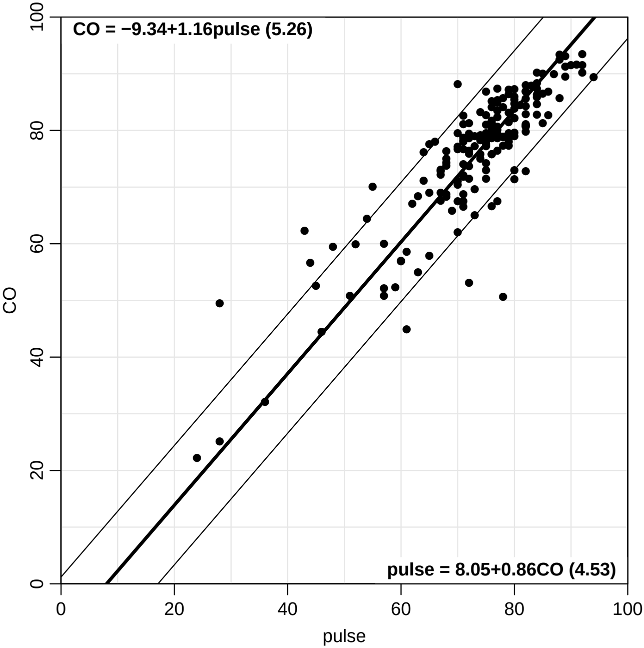
SAoMCS
19–20 March 2014
Haukeland University Hospital, Bergen, Norway
`http://BendixCarstensen.com/MethComp/Courses/Bergen.2014`

(`Transform`)

If variances are not constant

A transformation might help:

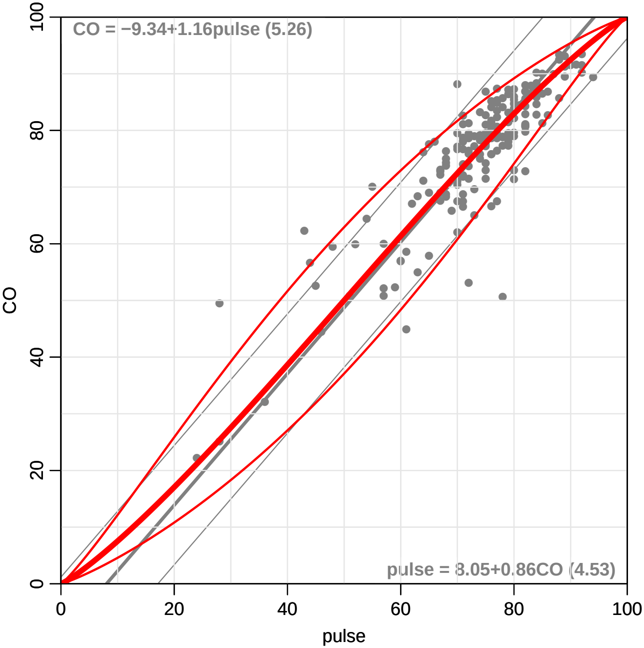
```
> library( MethComp )
> data( ox )
> ox <- Meth( ox )

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

> DA.reg(ox)

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

> round( ftable(DA.reg(ox)$Conv[,,-(1:4)]), 3 )
```


CO = −9.34+1.16pulse (5.26)
pulse = 8.05+0.86CO (4.53)
CO
pulse
0
20
40
60
80
100
0
20
40
60
80
100

```
> library(MethComp)
> 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

> system.time( MCox <- MCmcmc( ox, IxR=TRUE ) )
```

```
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 2000 iterations
  (of which 1000 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 4000 observations.

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

Initializing model

Sampling:
   user  system elapsed
  16.27    0.05   16.39
```

```
> ( Mox <- MethComp( MCox ) )

 Conversion between methods:
             alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:    From:
CO     CO     0.000  1.000  2.553   0.000   0.000   2.553
       pulse -9.341  1.161  5.263  -8.645   0.149   4.871
pulse  CO     8.045  0.861  4.526   8.645  -0.149   4.864
       pulse  0.000  1.000  5.985   0.000   0.000   5.985

 Variance components (sd):
       s.d.
Method    IxR   MxI   res
  CO    3.706 3.089 1.805
  pulse 3.173 2.647 4.232


> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( Mox, points=TRUE, axlim=c(0,100), xaxs="i", yaxs="i" )

Relationships between methods:
 CO-pulse = -8.64+0.15(CO+pulse)/2 (4.87)
 CO = -9.34+1.16pulse (5.26)
 pulse = 8.05+0.86CO (4.53)
```


CO = −9.34+1.16pulse (5.26)
pulse = 8.05+0.86CO (4.53)
CO
0
20
40
60
80
100
pulse
0
20
40
60
80
100

Using the `Transform` argument I

```
> system.time( MCox <- MCmcmc( ox, IxR=TRUE, Transform="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 2000 iterations
  (of which 1000 are burn-in),
- monitoring all values of the chain:
- giving a posterior sample of 4000 observations.

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

Using the `Transform` argument II

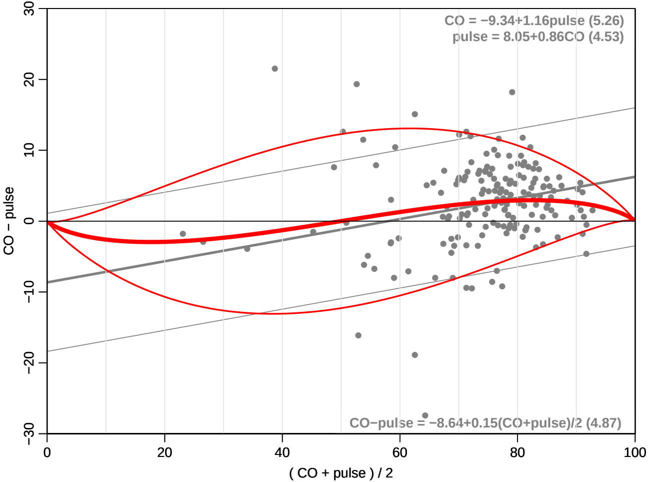
```
Initializing model

Sampling:
   user  system elapsed
  16.12    0.00   16.19

> ( Tox <- MethComp( MCox ) )

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


 Conversion between methods:
             alpha   beta  sd.pr in(t-f) sl(t-f) sd(t-f)
To:   From:
CO    CO     0.000  1.000  0.184   0.000   0.000   0.184
      pulse  0.000  1.141  0.264   0.000   0.132   0.247
pulse CO     0.000  0.876  0.232   0.000  -0.132   0.247
      pulse  0.000  1.000  0.283   0.000   0.000   0.283

 Variance components (sd):
       s.d.
Method    IxR   MxI  res
  CO    0.257 0.176 0.13
  pulse 0.224 0.154 0.20
```

Using the `Transform` argument III

```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( Mox, points=TRUE, axlim=c(0,100), xaxs="i", yaxs="i",
+       col.lines=gray(0.5), col.points=gray(0.5) )

Relationships between methods:
 CO-pulse = -8.64+0.15(CO+pulse)/2 (4.87)
 CO = -9.34+1.16pulse (5.26)
 pulse = 8.05+0.86CO (4.53)

> par( new=TRUE )
> plot( Tox, points=FALSE, axlim=c(0,100), xaxs="i", yaxs="i",
+       col.lines="red", lwd=c(5,2,2) )
```

Transformation to a Bland-Altman plot

Just convert to the differences versus the averages:



```
> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( Mox, pl.type="BA", points=TRUE, axlim=c(0,100), diflim=c(-30,30),
+       xaxs="i", yaxs="i" )

Relationships between methods:
 CO-pulse = -8.64+0.15(CO+pulse)/2 (4.87)
 CO = -9.34+1.16pulse (5.26)
 pulse = 8.05+0.86CO (4.53)

> abline( h=0 )

> par( mar=c(3,3,1,1), mgp=c(3,1,0)/1.6 )
> plot( Mox, pl.type="BA", points=TRUE, axlim=c(0,100), diflim=c(-30,30),
+       xaxs="i", yaxs="i", col.lines=gray(0.5), col.points=gray(0.5) )

Relationships between methods:
 CO-pulse = -8.64+0.15(CO+pulse)/2 (4.87)
 CO = -9.34+1.16pulse (5.26)
 pulse = 8.05+0.86CO (4.53)

> abline( h=0 )
> par( new=TRUE )
> plot( Tox, pl.type="BA", points=FALSE, axlim=c(0,100), diflim=c(-30,30),
+       xaxs="i", yaxs="i", col.lines="red", lwd=c(5,2,2) )
> abline( h=0 )
```

DG Altman and JM Bland.
Measurement in medicine: The analysis of method comparison studies.
The Statistician, 32:307–317, 1983.

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

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

B. Carstensen.
Comparing Clinical Measurement Methods: A practical guide.
Wiley, 2010.

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

B Carstensen, J Simpson, and LC Gurrin.
Statistical models for assessing agreement in method comparison studies with replicate measurements.
International Journal of Biostatistics, 4(1):Article 16, 2008.

J. Kottner, L. Audige, S. Brorson, A. Donner, B. J. Gajewski, A. Hrobjartsson, C. Roberts, M. Shoukri, and D. L. Streiner.
Guidelines for Reporting Reliability and Agreement Studies (GRRAS) were proposed.
J Clin Epidemiol, 64(1):96–106, Jan 2011.