Online Appendix A

Single Level Reliability as a Function of ICC, Reliability Within, and Reliability Between

Let:

V_1= The total variance of scale X when it is considered at a single-level
V_B= The total variance of a scale X at the between level
V_W= The total variance of a scale X at the within level

T_1= The true score variance of scale X when it is considered at a single-level
T_B= The true score variance of a scale X at the between level
T_W= The true score variance of a scale X at the within level

R_1= The reliability of scale X when estimated at a single level
R_B= The reliability of scale X at the between level
R_W= The reliability of scale X at the within level

Where reliabilities are specified as:

$$R_1 = \frac{T_1}{V_1}$$

$$R_B = \frac{T_B}{V_B}$$

$$R_W = \frac{T_W}{V_W}$$

And therefore:

$$T_1 = R_1 V_1$$

$$T_B = R_B V_B$$

$$T_W = R_W V_W$$

Because within- and between-cluster variances are orthogonal, we can also state that:

$$V_1 = V_B + V_w$$

$$T_1 = T_B + T_w$$

and we can define:

$$ICC = \frac{V_B}{V_1}$$

Or, alternatively,

$$ICC = \frac{V_1 - V_w}{V_1} = \frac{V_1}{V_1} - \frac{V_w}{V_1} = 1 - \frac{V_w}{V_1}$$

Therefore,

$$\frac{V_w}{V_1} + ICC = 1$$

Meaning,

$$\frac{V_w}{V_1} = 1 - ICC$$

Given the above, we can show that single-level reliability is defined as the weighted sum of a scale's level-specific reliabilities, where weights are a direct function of the scale's ICC:

$$R_1 = \frac{T_1}{V_1} = \frac{T_B + T_w}{V_1} = \frac{T_B}{V_1} + \frac{T_w}{V_1} = \frac{R_B V_B}{V_1} + \frac{R_W V_W}{V_1} = R_B * \frac{V_B}{V_1} + R_W * \frac{V_W}{V_1}$$

$$= R_B * ICC + R_W * (1 - ICC)$$

Online Appendix B

Path Diagram and Full Expansion of an MSEM Equation

Equation 12 represents the full model in Figure 1 using expanded matrices for p = 6 and m = 2:

$$\begin{bmatrix} y_{1ik} \\ y_{2ik} \\ y_{3ik} \\ y_{4ik} \\ y_{5ik} \\ y_{6ik} \end{bmatrix} = \left[\begin{array}{cccccccc:cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} y_{1wik} \\ y_{2wik} \\ y_{3wik} \\ y_{4wik} \\ y_{5wik} \\ y_{6wik} \\ \eta_{1wik} \\ \eta_{2wik} \\ \hdashline y_{1bk} \\ y_{2bk} \\ y_{3bk} \\ y_{4bk} \\ y_{5bk} \\ y_{6bk} \\ \eta_{1bk} \\ \eta_{2bk} \end{array}\right] \qquad (0)$$

$$
\begin{bmatrix} y_{1wik} \\ y_{2wik} \\ y_{3wik} \\ y_{4wik} \\ y_{5wik} \\ y_{6wik} \\ \eta_{1wik} \\ \eta_{2wik} \\ \hdashline y_{1bk} \\ y_{2bk} \\ y_{3bk} \\ y_{4bk} \\ y_{5bk} \\ y_{6bk} \\ \eta_{1bk} \\ \eta_{2bk} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \hdashline \alpha_{1k} \\ \alpha_{2k} \\ \alpha_{3k} \\ \alpha_{4k} \\ \alpha_{5k} \\ \alpha_{6k} \\ \alpha_{\eta 1k} \\ \alpha_{\eta 2k} \end{bmatrix}
+
\left[\begin{array}{cccccccc:cccccccc}
0&0&0&0&0&0&\lambda_{w11}&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&\lambda_{w21}&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&\lambda_{w31}&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&\lambda_{w42}&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&\lambda_{w52}&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&\lambda_{w62}&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ \hdashline
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right]
\begin{bmatrix} y_{1wik} \\ y_{2wik} \\ y_{3wik} \\ y_{4wik} \\ y_{5wik} \\ y_{6wik} \\ \eta_{1wik} \\ \eta_{2wik} \\ \hdashline y_{1bk} \\ y_{2bk} \\ y_{3bk} \\ y_{4bk} \\ y_{5bk} \\ y_{6bk} \\ \eta_{1bk} \\ \eta_{2bk} \end{bmatrix}
+
\begin{bmatrix} \delta_{1wik} \\ \delta_{2wik} \\ \delta_{3wik} \\ \delta_{4wik} \\ \delta_{5wik} \\ \delta_{6wik} \\ \zeta_{1wik} \\ \zeta_{2wik} \\ \hdashline 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\quad (0)
$$

$$
\begin{bmatrix} \alpha_{1k} \\ \alpha_{2k} \\ \alpha_{3k} \\ \alpha_{4k} \\ \alpha_{5k} \\ \alpha_{6k} \\ \alpha_{\eta 1k} \\ \alpha_{\eta 2k} \end{bmatrix}
=
\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \\ \mu_1 \\ \mu_2 \end{bmatrix}
+
\begin{bmatrix}
0&0&0&0&0&0&\lambda_{b11}&0 \\
0&0&0&0&0&0&\lambda_{b21}&0 \\
0&0&0&0&0&0&\lambda_{b31}&0 \\
0&0&0&0&0&0&0&\lambda_{b42} \\
0&0&0&0&0&0&0&\lambda_{b52} \\
0&0&0&0&0&0&0&\lambda_{b62} \\
0&0&0&0&0&0&0&0 \\
0&0&0&0&0&0&0&0
\end{bmatrix}
\begin{bmatrix} \alpha_{1k} \\ \alpha_{2k} \\ \alpha_{3k} \\ \alpha_{4k} \\ \alpha_{5k} \\ \alpha_{6k} \\ \alpha_{\eta 1k} \\ \alpha_{\eta 2k} \end{bmatrix}
+
\begin{bmatrix} \delta_{1k} \\ \delta_{2k} \\ \delta_{3k} \\ \delta_{4k} \\ \delta_{5k} \\ \delta_{6k} \\ \zeta_{1k} \\ \zeta_{2k} \end{bmatrix}
\quad (0)
$$

The usual identifying constrains must be made within each level (e.g., fixing a loading or factor variance to 1 for each factor; constraining at least one item intercept per factor if latent means μ_1 and μ_2 are to be estimated).

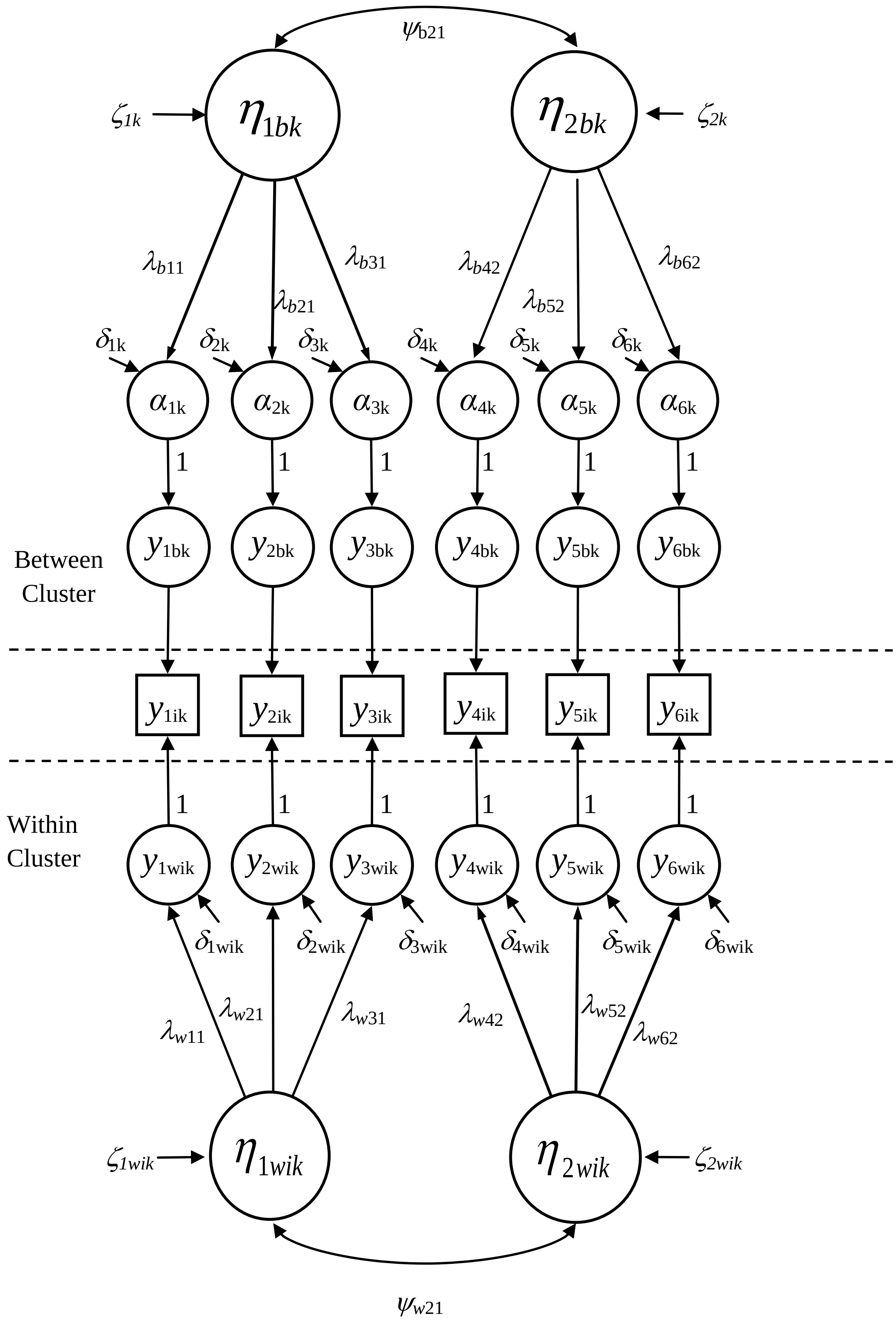

Figure B.1. *A multilevel CFA model.*

Online Appendix C

Supplementary Syntax Files

Mplus syntax for single-level α:

```
DATA:
FILE IS MATH_ACHIEVE.CSV;

VARIABLE:
NAMES ARE CLASS AS4MAMOR AS4MAENJ AS4MALIK AS4MABORr;
USEVARIABLES ARE AS4MAMOR AS4MAENJ AS4MALIK AS4MABORr;

MODEL:

AS4MAMOR WITH
AS4MAENJ  (C1)
AS4MALIK  (C2)
AS4MABORr (C3);

AS4MAENJ WITH
AS4MALIK  (C4)
AS4MABORr (C5);

AS4MALIK WITH
AS4MABORr (C6);

AS4MAMOR  (V1);
AS4MAENJ  (V2);
AS4MALIK  (V3);
AS4MABORr (V4);

OUTPUT: SAMPSTAT;

MODEL CONSTRAINT: NEW(COMP_V ALPHA);
COMP_V = V1+V2+V3+V4+2*(C1+C2+C3+C4+C5+C6);
ALPHA = (((C1+C2+C3+C4+C5+C6)/6)*16)/COMP_V;
```

Mplus syntax for two-level α:

```
DATA:
FILE IS MATH_ACHIEVE.CSV;

VARIABLE:
NAMES ARE CLASS AS4MAMOR AS4MAENJ AS4MALIK AS4MABORr;
CLUSTER = CLASS;

ANALYSIS:
TYPE=TWOLEVEL;

MODEL:
%WITHIN%
AS4MAMOR WITH
AS4MAENJ  (WC1)
AS4MALIK  (WC2)
AS4MABORr (WC3);

AS4MAENJ WITH
AS4MALIK  (WC4)
AS4MABORr (WC5);

AS4MALIK WITH
AS4MABORr (WC6);

AS4MAMOR  (WV1);
AS4MAENJ  (WV2);
AS4MALIK  (WV3);
AS4MABORr (WV4);

%BETWEEN%
AS4MAMOR WITH
AS4MAENJ  (BC1)
AS4MALIK  (BC2)
AS4MABORr (BC3);

AS4MAENJ WITH
AS4MALIK  (BC4)
AS4MABORr (BC5);

AS4MALIK WITH
AS4MABORr (BC6);

AS4MAMOR  (BV1);
AS4MAENJ  (BV2);
AS4MALIK  (BV3);
AS4MABORr (BV4);

OUTPUT: SAMPSTAT;
MODEL CONSTRAINT: NEW(COMP_V_WALPHAW COMP_V_B ALPHAB);
COMP_V_W = WV1+WV2+WV3+WV4+2*(WC1+WC2+WC3+WC4+WC5+WC6);
ALPHAW = (((WC1+WC2+WC3+WC4+WC5+WC6)/6)*16)/COMP_V_W;
COMP_V_B = BV1+BV2+BV3+BV4+2*(BC1+BC2+BC3+BC4+BC5+BC6);
ALPHAB = (((BC1+BC2+BC3+BC4+BC5+BC6)/6)*16)/COMP_V_B;
```

Mplus syntax for computing single-level ω and H:

```
DATA:
FILE IS MATH_ACHIEVE.CSV;

VARIABLE:
NAMES ARE CLASS AS4MAMOR AS4MAENJ AS4MALIK AS4MABORr;
USEVARIABLES ARE AS4MAMOR AS4MAENJ AS4MALIK AS4MABORr;

MODEL:

MATH BY AS4MAMOR* (L1)
   AS4MAENJ  (L2)
   AS4MALIK  (L3)
   AS4MABORr (L4);
MATH@1;

AS4MAMOR  (R1);
AS4MAENJ  (R2);
AS4MALIK  (R3);
AS4MABORr (R4);

OUTPUT: SAMPSTAT;

MODEL CONSTRAINT: NEW(NUM DENOM OMEGA H);
NUM = (L1+L2+L3+L4)**2;
DENOM = ((L1+L2+L3+L4)**2)+(R1+R2+R3+R4);
OMEGA = NUM/DENOM;
H = 1/(1+(1/((L1**2/R1)+(L2**2/R2)+(L3**2/R3)+(L4**2/R4))));
```

Mplus syntax for computing two-level ω and H:

```
DATA:
FILE IS MATH_ACHIEVE.CSV;

VARIABLE:
NAMES ARE CLASS AS4MAMOR AS4MAENJ AS4MALIK AS4MABORr;
CLUSTER = CLASS;

ANALYSIS:
TYPE=TWOLEVEL;

MODEL:
%WITHIN%
MATHW BY AS4MAMOR* (WL1)
   AS4MAENJ  (WL2)
   AS4MALIK  (WL3)
   AS4MABORr (WL4);
MATHW@1;

AS4MAMOR  (WR1);
AS4MAENJ  (WR2);
AS4MALIK  (WR3);
AS4MABORr (WR4);

%BETWEEN%
MATHB BY AS4MAMOR* (BL1)
   AS4MAENJ  (BL2)
   AS4MALIK  (BL3)
   AS4MABORr (BL4);
MATHB@1;

AS4MAMOR  (BR1);
AS4MAENJ  (BR2);
AS4MALIK  (BR3);
AS4MABORr (BR4);

OUTPUT: SAMPSTAT;

MODEL CONSTRAINT: NEW(NUMW DENOMW OMEGAW HW
NUMB DENOMB OMEGAB HB);
NUMW = (WL1+WL2+WL3+WL4)**2;
DENOMW = ((WL1+WL2+WL3+WL4)**2)+(WR1+WR2+WR3+WR4);
OMEGAW = NUMW/DENOMW;
HW = 1/(1+(1/((WL1**2/WR1)+(WL2**2/WR2)+(WL3**2/WR3)+(WL4**2/WR4))));

NUMB = (BL1+BL2+BL3+BL4)**2;
DENOMB = ((BL1+BL2+BL3+BL4)**2)+(BR1+BR2+BR3+BR4);
OMEGAB = NUMB/DENOMB;
HB = 1/(1+(1/((BL1**2/BR1)+(BL2**2/BR2)+(BL3**2/BR3)+(BL4**2/BR4))));

WR1 > 0; BR1 > 0;
WR2 > 0; BR2 > 0;
WR3 > 0; BR3 > 0;
WR4 > 0; BR4 > 0;
```

Monte Carlo Confidence Interval Syntax for R

```
#------------------Clear All----------------------------#

rm(list=ls())

#---------------SPECIFICATIONS------------------------#

##############################################################

# This code calculates Monte Carlo confidence
# intervals for level-specific alpha and composite reliability
# based on CFA-derived factor loadings and residual variances.
# Our calculator also requires users to input the asymptotic
# covariance matrix (ACM) of these estimates to replicate their
# assumed sampling distribution (input parameters are assumed to
# be multivariate normally distributed).

##############################################################
# Install package MASS
#install.packages("MASS")

# Load package MASS
require(MASS)

##############################################################
# User Input

conf <- .95                 # Confidence level (1 – Type I error rate)
reps <- 10000               # Number of Monte Carlo simulations
set.seed(7913025)           # Set random seed

# Factor loadings and residual variances; used to calculate all reliability estimates
wlambda <- as.matrix(c(.299,.299,.299,.299,.299,.299))       # Factor loadings within
wtheta <- as.matrix(c(.905,.905,.905,.905,.905,.905))        # Residual variances within
blambda <- as.matrix(c(.137,.137,.160,.160,.183,.183))       # Factor loadings between
btheta <- as.matrix(c(.034,.034,.027,.027,.019,.019))        # Residual variances between

# Input full ACM ordered as lambda(within), theta(within), lambda(between), theta(between)
    # The ACM can be imported from an external file (as shown), or
    # inputted directly into the syntax as a matrix
acmuse <- as.matrix(read.table("F:\\Monte Carlo CIs\\example.acov"))


##############################################################

#---------------   End User Input    ------------------------#

##############################################################

pest <- rbind(wlambda,wtheta,blambda,btheta)  # Combine parameter estimates into a single vector
nwest <- sum(nrow(wlambda),nrow(wtheta))      # Count number of within-level parameter estimates
data <- mvrnorm(reps,pest,acmuse,empirical=T) # Generate random draws form joint distribution of
                                              # parameter estimates
                                              # Note: empirical=T specifies that the resulting
                                              # data will perfectly adhere to the specified
                                              # parameters (i.e., the generated sample statistics
                                              # will be identical to the specified parameters).

# Compute within-level omegas
womega <- rowSums(data[,c(1:nrow(wlambda))])^2/ (rowSums(data[,c(1:nrow(wlambda))])^2 +
rowSums(data[,c((nrow(wlambda)+1):(nrow(wlambda)+nrow(wtheta)))]))

# Compute between-cluster omegas
bomega <-
rowSums(data[,c((nwest+1):(nwest+nrow(wlambda)))])^2/(rowSums(data[,c((nwest+1):(nwest+nrow(wlamb
da)))])^2 + rowSums(data[,c((nrow(wlambda)+nwest+1):(nwest+nrow(wlambda)+nrow(wtheta)))]))

# Compute within-level alphas
    # Compute the sum of the within-level covariances by first treating
```

```
    # the within-level true score variance estimate as the sum of the within-level
    # true-score covariance matrix. Remove the indicator true-score variances
    # from the estimated true score variance such that the result equals two times
    # each unique coavariance. Divide this result by two to obtain the sum of the
    # within-level covariances
wcovs <-  (rowSums(data[,c(1:nrow(wlambda))])^2-rowSums(data[,c(1:nrow(wlambda))]^2))/2

    # Find the average within-level covariance
avgwcov <- wcovs/((nrow(wlambda)*nrow(wlambda)-nrow(wlambda))/2)

    # Compute within-level alphas
walpha <- (nrow(wlambda)^2*avgwcov)/(rowSums(data[,c(1:nrow(wlambda))])^2 +
rowSums(data[,c((nrow(wlambda)+1):(nrow(wlambda)+nrow(wtheta)))]))

# Compute between-cluster alphas as above
bcovs <-  (rowSums(data[,c((nwest+1):(nwest+nrow(wlambda)))])^2-
rowSums(data[,c((nwest+1):(nwest+nrow(wlambda)))]^2))/2
avgbcov <- bcovs/((nrow(blambda)*nrow(blambda)-nrow(blambda))/2)
balpha <- (nrow(blambda)^2*avgbcov)/(rowSums(data[,c((nwest+1):(nwest+nrow(wlambda)))])^2 +
rowSums(data[,c((nrow(wlambda)+nwest+1):(nwest+nrow(wlambda)+nrow(wtheta)))]))

# Print results

cat('\n Confidence Limits:
         Lower   Upper
W OMEGA: ',round(quantile(womega,c((1-conf)/2,1-(1-conf)/2),na.rm=T),5),
'\nB OMEGA: ',round(quantile(bomega,c((1-conf)/2,1-(1-conf)/2),na.rm=T),5),
'\n\nW ALPHA: ',round(quantile(walpha,c((1-conf)/2,1-(1-conf)/2),na.rm=T),5),
'\nB ALPHA: ',round(quantile(balpha,c((1-conf)/2,1-(1-conf)/2),na.rm=T),5),
'\n')
```