

# Psychometric Modelling of Longitudinal Genetically-Informative Twin Data: Supplementary material

I. Schwabe    Z. Gu    J. Tijmstra    P. Hatemi    S. Pohl

This document contains supplementary material for the article "Psychometric Modelling of Longitudinal Genetically-Informative Twin Data".

Below, you can find all JAGS scripts that were used either in the simulation studies or in the data application. All scripts as well as the R script that was used for the data cleaning can also be found online at the personal github page of the first author at <https://github.com/ingaschwabe/Psychometric-Modelling-of-Longitudinal-Twin-Data>.

## Longitudinal AE decomposition

### IRT approach (Latent trait model)

Following JAGS script fits a longitudinal AE decomposition with an incorporated IRT Rasch model at the phenotypic level (Latent trait AE twin model). The script was used for the first simulation study and the data application.

```
1 #=====
2 # Following JAGS script fits a longitudinal latent trait-state AE twin model
3 # T = total number of time points (equal to 2 here)
4 # n_dz = total number of DZ twin families; n_dz = total number of MZ families
5 # n_items = total number of questionnaire items
6 #
7 # Note that this script works for 2 time points but can easily be adjusted
8 # to work for more time points.
9 # Any questions: Contact Inga Schwabe (I.Schwabe@uvt.nl)
10 #=====
11
12 model{
13     # Twin model part for MZ twins
14     for (fam in 1:n_mz){
15         a1_mz[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
16
17         # Modelling phenotypes:
18         pheno_mz_twin1[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T],
19                                         tau_e[1:T,1:T])
20         pheno_mz_twin2[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T],
21                                         tau_e[1:T,1:T])
22
23         # IRT part: Rasch model for both time points
24         for (k in 1:n_items){
25             logit(p_mz_twin1_t1[fam,k]) <- pheno_mz_twin1[fam,1] - beta[k]
26             data_mz_twin1_t1[fam,k] ~ dbern(p_mz_twin1_t1[fam,k])
27
28             logit(p_mz_twin2_t1[fam,k]) <- pheno_mz_twin2[fam,1] - beta[k]
29             data_mz_twin2_t1[fam,k] ~ dbern(p_mz_twin2_t1[fam,k])
30         }
31
32         for (k in 1:n_items){
33             logit(p_mz_twin1_t2[fam,k]) <- pheno_mz_twin1[fam,2] - beta[k]
34             data_mz_twin1_t2[fam,k] ~ dbern(p_mz_twin1_t2[fam,k])
35
36             logit(p_mz_twin2_t2[fam,k]) <- pheno_mz_twin2[fam,2] - beta[k]
37             data_mz_twin2_t2[fam,k] ~ dbern(p_mz_twin2_t2[fam,k])
38         }
39     }
```

```

40
41      # Twin model part for DZ twins
42  for (fam in 1:n_dz){
43
44      a1_dz [fam ,1:T] ~ dmnorm(mu_a [1:T], tau_a [1:T,1:T])
45      a2_dz [fam ,1:T] <- a1_dz [fam ,1:T]/sqrt(2)
46
47      a3_dz_twin1 [fam ,1:T] ~ dmnorm(mu_a [1:T], tau_a [1:T,1:T])
48      a3_dz_twin2 [fam ,1:T] ~ dmnorm(mu_a [1:T], tau_a [1:T,1:T])
49
50      a4_dz_twin1 [fam ,1:T] <- a3_dz_twin1 [fam ,1:T]/sqrt(2)
51      a4_dz_twin2 [fam ,1:T] <- a3_dz_twin2 [fam ,1:T]/sqrt(2)
52
53      # Modelling phenotypes:
54      pheno_dz_twin1 [fam ,1:T] ~ dmnorm(mu [1:T] + a2_dz [fam ,1:T] +
55                                         a4_dz_twin1 [fam ,1:T], tau_e [1:T,1:T])
56      pheno_dz_twin2 [fam ,1:T] ~ dmnorm(mu [1:T] + a2_dz [fam ,1:T] +
57                                         a4_dz_twin2 [fam ,1:T], tau_e [1:T,1:T])
58
59      # IRT part (Rasch model for both time points):
60  for (k in 1:n_items){
61      logit(p_dz_twin1_t1 [fam ,k]) <- pheno_dz_twin1 [fam ,1] - beta[k]
62      data_dz_twin1_t1 [fam ,k] ~ dbern(p_dz_twin1_t1 [fam ,k])
63
64      logit(p_dz_twin2_t1 [fam ,k]) <- pheno_dz_twin2 [fam ,1] - beta[k]
65      data_dz_twin2_t1 [fam ,k] ~ dbern(p_dz_twin2_t1 [fam ,k])
66  }
67
68  for (k in 1:n_items){
69      logit(p_dz_twin1_t2 [fam ,k]) <- pheno_dz_twin1 [fam ,2] - beta[k]
70      data_dz_twin1_t2 [fam ,k] ~ dbern(p_dz_twin1_t2 [fam ,k])
71
72      logit(p_dz_twin2_t2 [fam ,k]) <- pheno_dz_twin2 [fam ,2] - beta[k]
73      data_dz_twin2_t2 [fam ,k] ~ dbern(p_dz_twin2_t2 [fam ,k])
74  }
75  }
76
77  # Prior distributions
78  # Normal prior for beta item parameters:
79  for (k in 1:n_items){
80      beta[k] ~ dnorm(0, .1)
81  }
82
83  # The expected value of the first time point is set to 0 to identify the
84  # measurement scale.
85  # A normal prior is set on the remaining time point averages

```

```

86     mu[1] <- 0
87     mu[2] ~ dnorm(0, .1)
88
89     # Inverse Wishart prior distribution for covariance matrices
90     tau_a[1:T,1:T] ~ dwish(omega_a[,], T+1)
91     tau_e[1:T,1:T] ~ dwish(omega_e[,], T+1)
92
93     # Calculate inverse
94     Sigma_a[1:T,1:T] <- inverse(tau_a[,])
95     Sigma_e[1:T,1:T] <- inverse(tau_e[,])
96 }
```

## Sum-score approach

Following JAGS script fits a longitudinal AE twin model on sum-scores. The script was used for the first simulation study and the data application.

```

1  #=====
2  # Following JAGS script fits a longitudinal AE twin model using sum-scores
3  # T = total number of time points
4  # n_dz = total number of DZ twin families; n_dz = total number of MZ families
5  #
6  # Any questions: Contact Inga Schwabe (I.Schwabe@uvt.nl)
7  #=====
8
9  model{
10
11    # Twin model part for MZ twins
12    for (fam in 1:n_mz){
13      a1_mz[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
14
15    # Modelling phenotypes:
16    data_mz_twin1[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T], tau_e[1:T,1:T])
17    data_mz_twin2[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T], tau_e[1:T,1:T])
18  }
19
20  # Twin model part for DZ twins
21  for (fam in 1:n_dz){
22
23    a1_dz[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
24    a2_dz[fam,1:T] <- a1_dz[fam,1:T]/sqrt(2)
25
26    a3_dz_twin1[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
27    a3_dz_twin2[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
28
29    a4_dz_twin1[fam,1:T] <- a3_dz_twin1[fam,1:T]/sqrt(2)
30    a4_dz_twin2[fam,1:T] <- a3_dz_twin2[fam,1:T]/sqrt(2)
31
32    # Modelling phenotypes:
33    data_dz_twin1[fam,1:T] ~ dmnorm(mu[1:T] + a2_dz[fam,1:T] +
34                                a4_dz_twin1[fam,1:T], tau_e[1:T,1:T])
35    data_dz_twin2[fam,1:T] ~ dmnorm(mu[1:T] + a2_dz[fam,1:T] +
36                                a4_dz_twin2[fam,1:T], tau_e[1:T,1:T])
37  }
38
39  # Prior distributions
40
41  # Normal prior for expected values of time points
42  for (timepoint in 1:T){

```

```

43     mu[timepoint] ~ dnorm(0, .1)
44 }
45
46 # Inverse Wishart prior distribution for covariance matrices
47 tau_a[1:T,1:T] ~ dwish(omega_a[,], T+1)
48 tau_e[1:T,1:T] ~ dwish(omega_e[,], T+1)
49
50 # calculate inverse
51 Sigma_a[1:T,1:T] <- inverse(tau_a[,])
52 Sigma_e[1:T,1:T] <- inverse(tau_e[,])
53 }
```

## Longitudinal ACE model

### IRT approach (Latent trait model)

Following JAGS script fits a longitudinal ACE twin model with an incorporated IRT Rasch model at the phenotypic level (Latent trait ACE twin model).

```
1  #####  
2 # Following JAGS script fits a longitudinal ACE latent trait-state AE twin model  
3 # T = total number of time points (equal to 2 here)  
4 # n_dz = total number of DZ twin families; n_dz = total number of MZ families  
5 # n_items = total number of questionnaire items  
6 #  
7 # Note that this script works for 2 time points but can easily be adjusted  
8 # to work for more time points.  
9 # Any questions: Contact Inga Schwabe (I.Schwabe@uvt.nl)  
10 #####  
11  
12 model{  
13     # Twin model part for MZ twins  
14     for (fam in 1:n_mz){  
15         a1_mz[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])  
16         c_mz[fam, 1:T] ~ dmnorm(mu_c[1:T], tau_c[1:T,1:T])  
17  
18         # Modelling phenotypes:  
19         pheno_mz_twin1[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T] +  
20                                     c_mz[fam,1:T], tau_e[1:T,1:T])  
21         pheno_mz_twin2[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T] +  
22                                     c_mz[fam,1:T], tau_e[1:T,1:T])  
23  
24         # IRT part: Rasch model for both time points  
25         for (k in 1:n_items){  
26             logit(p_mz_twin1_t1[fam,k]) <- pheno_mz_twin1[fam,1] - beta[k]  
27             data_mz_twin1_t1[fam,k] ~ dbern(p_mz_twin1_t1[fam,k])  
28  
29             logit(p_mz_twin2_t1[fam,k]) <- pheno_mz_twin2[fam,1] - beta[k]  
30             data_mz_twin2_t1[fam,k] ~ dbern(p_mz_twin2_t1[fam,k])  
31         }  
32  
33         for (k in 1:n_items){  
34             logit(p_mz_twin1_t2[fam,k]) <- pheno_mz_twin1[fam,2] - beta[k]  
35             data_mz_twin1_t2[fam,k] ~ dbern(p_mz_twin1_t2[fam,k])  
36  
37             logit(p_mz_twin2_t2[fam,k]) <- pheno_mz_twin2[fam,2] - beta[k]  
38             data_mz_twin2_t2[fam,k] ~ dbern(p_mz_twin2_t2[fam,k])  
39         }  
40     }
```

```

41 }
42
43 # Twin model part for DZ twins
44 for (fam in 1:n_dz){
45   a1_dz[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
46   a2_dz[fam,1:T] <- a1_dz[fam,1:T]/sqrt(2)
47
48   a3_dz_twin1[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
49   a3_dz_twin2[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
50
51   a4_dz_twin1[fam,1:T] <- a3_dz_twin1[fam,1:T]/sqrt(2)
52   a4_dz_twin2[fam,1:T] <- a3_dz_twin2[fam,1:T]/sqrt(2)
53
54   c_dz[fam,1:T] ~ dmnorm(mu_c[1:T], tau_c[1:T,1:T])
55
56 # Modelling phenotypes:
57 pheno_dz_twin1[fam,1:T] ~ dmnorm(mu[1:T] + a2_dz[fam,1:T] +
58                                     a4_dz_twin1[fam,1:T] + c_dz[fam,1:T],
59                                     tau_e[1:T,1:T])
60 pheno_dz_twin2[fam,1:T] ~ dmnorm(mu[1:T] + a2_dz[fam,1:T] +
61                                     a4_dz_twin2[fam,1:T] + c_dz[fam,1:T],
62                                     tau_e[1:T,1:T])
63
64 # IRT part (Rasch model for both time points):
65 for (k in 1:n_items){
66   logit(p_dz_twin1_t1[fam,k]) <- pheno_dz_twin1[fam,1] - beta[k]
67   data_dz_twin1_t1[fam,k] ~ dbern(p_dz_twin1_t1[fam,k])
68
69   logit(p_dz_twin2_t1[fam,k]) <- pheno_dz_twin2[fam,1] - beta[k]
70   data_dz_twin2_t1[fam,k] ~ dbern(p_dz_twin2_t1[fam,k])
71 }
72
73 for (k in 1:n_items){
74   logit(p_dz_twin1_t2[fam,k]) <- pheno_dz_twin1[fam,2] - beta[k]
75   data_dz_twin1_t2[fam,k] ~ dbern(p_dz_twin1_t2[fam,k])
76
77   logit(p_dz_twin2_t2[fam,k]) <- pheno_dz_twin2[fam,2] - beta[k]
78   data_dz_twin2_t2[fam,k] ~ dbern(p_dz_twin2_t2[fam,k])
79 }
80 }
81
82 # Prior distributions
83 # Normal prior for beta item parameters:
84 for (k in 1:n_items){
85   beta[k] ~ dnorm(0, .1)
86 }
```

```

87
88  # The expected value of the first time point is set to 0 to identify the
89  # measurement scale. A normal prior is set on the remaining time point average
90  mu[1] <- 0
91  mu[2] ~ dnorm(0, .1)
92
93  # Inverse Wishart prior distribution for covariance matrices
94  tau_a[1:T,1:T] ~ dwish(omega_a[,], T+1)
95  tau_e[1:T,1:T] ~ dwish(omega_e[,], T+1)
96  tau_c[1:T,1:T] ~ dwish(omega_c[,], T+1)
97
98  # Calculate inverse:
99  Sigma_a[1:T,1:T] <- inverse(tau_a[,])
100 Sigma_e[1:T,1:T] <- inverse(tau_e[,])
101 Sigma_c[1:T,1:T] <- inverse(tau_c[,])
102 }
```

### IRT approach (Latent trait model): Alternative prior

Following JAGS script fits a longitudinal ACE twin model with an incorporated IRT Rasch model at the phenotypic level (Latent trait ACE twin model). This script was used in the additional simulation study.

```

1  #=====
2  # Following JAGS script fits a longitudinal ACE latent-state AE twin model
3  # T = total number of time points (equal to 2 here)
4  # n_dz = total number of DZ twin families; n_dz = total number of MZ families
5  # n_items = total number of questionnaire items
6  #
7  # Note that this script works for 2 time points but can easily be adjusted
8  # to work for more time points.
9  # Any questions: Contact Inga Schwabe (I.Schwabe@uvt.nl)
10 #=====
11
12 model{
13     # Twin model part for MZ twins
14     for (fam in 1:n_mz){
15
16         a1_mz[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
17         c_mz[fam, 1:T] ~ dmnorm(mu_c[1:T], tau_c[1:T,1:T])
18
19         # Modelling phenotypes:
20         pheno_mz_twin1[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T] +
21                                         c_mz[fam,1:T], tau_e[1:T,1:T])
22         pheno_mz_twin2[fam,1:T] ~ dmnorm(mu[1:T] + a1_mz[fam,1:T] +
23                                         c_mz[fam,1:T], tau_e[1:T,1:T])
24
25         # IRT part: Rasch model for both time points
26         for (k in 1:n_items){
27             logit(p_mz_twin1_t1[fam,k]) <- pheno_mz_twin1[fam,1] - beta[k]
28             data_mz_twin1_t1[fam,k] ~ dbern(p_mz_twin1_t1[fam,k])
29
30             logit(p_mz_twin2_t1[fam,k]) <- pheno_mz_twin2[fam,1] - beta[k]
31             data_mz_twin2_t1[fam,k] ~ dbern(p_mz_twin2_t1[fam,k])
32         }
33
34         for (k in 1:n_items){
35             logit(p_mz_twin1_t2[fam,k]) <- pheno_mz_twin1[fam,2] - beta[k]
36             data_mz_twin1_t2[fam,k] ~ dbern(p_mz_twin1_t2[fam,k])
37
38             logit(p_mz_twin2_t2[fam,k]) <- pheno_mz_twin2[fam,2] - beta[k]
39             data_mz_twin2_t2[fam,k] ~ dbern(p_mz_twin2_t2[fam,k])
40         }
41     }

```

```

42
43 # Twin model part for DZ twins
44 for (fam in 1:n_dz){
45   a1_dz[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
46   a2_dz[fam,1:T] <- a1_dz[fam,1:T]/sqrt(2)
47
48   a3_dz_twin1[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
49   a3_dz_twin2[fam,1:T] ~ dmnorm(mu_a[1:T], tau_a[1:T,1:T])
50
51   a4_dz_twin1[fam,1:T] <- a3_dz_twin1[fam,1:T]/sqrt(2)
52   a4_dz_twin2[fam,1:T] <- a3_dz_twin2[fam,1:T]/sqrt(2)
53
54   c_dz[fam,1:T] ~ dmnorm(mu_c[1:T], tau_c[1:T,1:T])
55
56 # Modelling phenotypes:
57 pheno_dz_twin1[fam,1:T] ~ dmnorm(mu[1:T] + a2_dz[fam,1:T] +
58                                     a4_dz_twin1[fam,1:T] + c_dz[fam,1:T],
59                                     tau_e[1:T,1:T])
60 pheno_dz_twin2[fam,1:T] ~ dmnorm(mu[1:T] + a2_dz[fam,1:T] +
61                                     a4_dz_twin2[fam,1:T] + c_dz[fam,1:T],
62                                     tau_e[1:T,1:T])
63
64 # IRT part (Rasch model for both time points):
65 for (k in 1:n_items){
66   logit(p_dz_twin1_t1[fam,k]) <- pheno_dz_twin1[fam,1] - beta[k]
67   data_dz_twin1_t1[fam,k] ~ dbern(p_dz_twin1_t1[fam,k])
68
69   logit(p_dz_twin2_t1[fam,k]) <- pheno_dz_twin2[fam,1] - beta[k]
70   data_dz_twin2_t1[fam,k] ~ dbern(p_dz_twin2_t1[fam,k])
71 }
72
73 for (k in 1:n_items){
74   logit(p_dz_twin1_t2[fam,k]) <- pheno_dz_twin1[fam,2] - beta[k]
75   data_dz_twin1_t2[fam,k] ~ dbern(p_dz_twin1_t2[fam,k])
76
77   logit(p_dz_twin2_t2[fam,k]) <- pheno_dz_twin2[fam,2] - beta[k]
78   data_dz_twin2_t2[fam,k] ~ dbern(p_dz_twin2_t2[fam,k])
79 }
80 }
81
82 # Prior distributions
83 # Normal prior for beta item parameters:
84 for (k in 1:n_items){
85   beta[k] ~ dnorm(0, .1)
86 }
87

```

```

88  # The expected value of the first time point is set to 0 to identify the
89  # measurement scale.
90  # A normal prior is set on the remaining time point averages
91  mu[1] <- 0
92  mu[2] ~ dnorm(0, .1)
93
94  # Inverse Wishart prior distribution for covariance matrices
95  tau_a[1:T,1:T] ~ dwish(omega_a, nu_a)
96  tau_c[1:T,1:T] ~ dwish(omega_c, nu_c)
97  tau_e[1:T,1:T] ~ dwish(omega_e, nu_e)
98
99  # Calculate inverse:
100 Sigma_a[1:T,1:T] <- inverse(tau_a[,])
101 Sigma_e[1:T,1:T] <- inverse(tau_e[,])
102 Sigma_c[1:T,1:T] <- inverse(tau_c[,])
103
104 # Hyperprios for the inverse Wishart priors
105
106 # Hyperpriors covariance matrix
107 for (d in 1:2){
108     omega_a[d,d] <- lambda_a[d]
109     omega_c[d,d] <- lambda_c[d]
110     omega_e[d,d] <- lambda_e[d]
111
112     lambda_a[d] <- exp(Loglambda_a[d])
113     lambda_c[d] <- exp(Loglambda_c[d])
114     lambda_e[d] <- exp(Loglambda_e[d])
115
116     Loglambda_a[d] ~ dunif(-100,100)
117     Loglambda_c[d] ~ dunif(-100,100)
118     Loglambda_e[d] ~ dunif(-100,100)
119 }
120
121 # Hyperpriors degree of freedom
122 # 2 = total number of timepoints
123 nu_a ~ dunif(2,100)
124 nu_c ~ dunif(2,100)
125 nu_e ~ dunif(2,100)
126
127 # Set off diag elements to 0
128 omega_a[1,2] <- 0
129 omega_a[2,1] <- 0
130
131 omega_c[1,2] <- 0
132 omega_c[2,1] <- 0
133

```

```
134      omega_e[1,2] <- 0
135      omega_e[2,1] <- 0
136  }
```