# Package: dina (via r-universe)

October 28, 2026

**Type** Package

**Title** Bayesian Estimation of DINA Model

**Version** 2.0.3

**Description** Estimate the Deterministic Input, Noisy ``And" Gate (DINA) cognitive diagnostic model parameters using the Gibbs sampler described by Culpepper (2015) <doi:10.3102/1076998615595403>.

**URL** https://github.com/tmsalab/dina, https://tmsalab.github.io/dina/

**BugReports** https://github.com/tmsalab/dina/issues

**License** GPL (>= 2)

**Depends** R (>= 4.3.0), simcdm (>= 0.1.0)

**Imports** Rcpp (>= 1.1.0)

**LinkingTo** Rcpp, RcppArmadillo (>= 15.0.2-2), simcdm, rgen

**Suggests** CDM, covr, testthat

**RoxygenNote** 7.3.3

**Encoding** UTF-8

**NeedsCompilation** yes

**Author** Steven Andrew Culpepper [aut, cph] (ORCID: <https://orcid.org/0000-0003-4226-6176>), James Joseph Balamuta [aut, cre] (ORCID: <https://orcid.org/0000-0003-2826-8458>)

**Maintainer** James Joseph Balamuta <balamut2@illinois.edu>

**Repository** https://cran.r-universe.dev

**Date/Publication** 2026-10-28 16:50:54 UTC

**RemoteUrl** https://github.com/cran/dina

**RemoteRef** HEAD

**RemoteSha** 352735e7a46456dc7ed4ff2cb4b50b45b3e0352a

# Contents

dina-package . . . . . 2
dina . . . . . 2
DINA_Gibbs . . . . . 6

**Index** **7**

---

`dina-package` *dina: Bayesian Estimation of DINA Model*

---

## Description

Estimate the Deterministic Input, Noisy "And" Gate (DINA) cognitive diagnostic model parameters using the Gibbs sampler described by Culpepper (2015) doi:10.3102/1076998615595403.

## Author(s)

**Maintainer**: James Joseph Balamuta <balamut2@illinois.edu> (ORCID)

Authors:

- Steven Andrew Culpepper <sculpepp@illinois.edu> (ORCID) [copyright holder]

## See Also

Useful links:

- https://github.com/tmsalab/dina
- https://tmsalab.github.io/dina/
- Report bugs at https://github.com/tmsalab/dina/issues

---

`dina` *Generate Posterior Distribution with Gibbs sampler*

---

## Description

Function for sampling parameters from full conditional distributions. The function returns a list of arrays or matrices with parameter posterior samples. Note that the output includes the posterior samples in objects.

## Usage

```
dina(Y, Q, chain_length = 10000)
```

## Arguments

| | |
|---|---|
| Y | A $N \times J$ matrix of observed responses. |
| Q | A $N \times K$ matrix indicating which skills are required for which items. |
| chain_length | Number of MCMC iterations. |

## Value

A list with samples from the posterior distribution with each entry named:

- CLASSES = individual attribute profiles,
- PIs = latent class proportions,
- SigS = item slipping parameters, and
- GamS = item guessing parameters.

## Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

## See Also

simcdm::sim_dina_items() and simcdm::attribute_classes()

## Examples

```
## Not run:

####################################
# Tatsuoka Fraction Subtraction Data
####################################

# This example requires data from the CDM package.
if(requireNamespace("CDM")) {

data(fraction.subtraction.data, package = "CDM")
data(fraction.subtraction.qmatrix, package = "CDM")
Y_1984 = as.matrix(fraction.subtraction.data)
Q_1984 = as.matrix(fraction.subtraction.qmatrix)
K_1984 = ncol(fraction.subtraction.qmatrix)
J_1984 = ncol(Y_1984)

# Creating matrix of possible attribute profiles
As_1984 = rep(0, K_1984)

for(j in 1:K_1984) {
    temp = combn(1:K_1984, m = j)
    tempmat = matrix(0, ncol(temp), K_1984)
    for(j in 1:ncol(temp)) tempmat[j, temp[, j]] = 1
    As_1984 = rbind(As_1984, tempmat)
}
```

```
As_1984 = as.matrix(As_1984)

# Generate samples from posterior distribution
# May take 8 minutes
chainLength = 5000
burnin = 1000
chain_samples = burnin:chainLength

outchain_1984 = dina(Y = Y_1984, Q = Q_1984,
                     chain_length = chainLength)

# Summarize posterior samples for g and 1-s
mgs_1984 = apply(outchain_1984$GamS[, chain_samples], 1, mean)
sgs_1984 = apply(outchain_1984$GamS[, chain_samples], 1, sd)
mss_1984 = 1 - apply(outchain_1984$SigS[, chain_samples], 1, mean)
sss_1984 = apply(outchain_1984$SigS[, chain_samples], 1, sd)
output_1984 = cbind(mgs_1984, sgs_1984, mss_1984, sss_1984)
colnames(output_1984) = c('g Est','g SE','1-s Est','1-s SE')
rownames(output_1984) = colnames(Y_1984)
print(output_1984, digits = 3)

# Summarize marginal skill distribution using posterior samples for latent
# class proportions
marg_PIs = t(As_1984) %*% outchain_1984$PIs
PI_Est = apply(marg_PIs[, chain_samples], 1, mean)
PI_Sd = apply(marg_PIs[, chain_samples], 1, sd)
PIoutput = cbind(PI_Est, PI_Sd)
colnames(PIoutput) = c('EST', 'SE')
rownames(PIoutput) = paste('Skill', 1:K_1984)
print(PIoutput, digits = 3)

}

#########################################################
# de la Torre (2009) Simulation Replication w/ N = 200
#########################################################
N = 200
K = 5
J = 30
delta0 = rep(1, 2^K)

# Creating Q matrix
Q = matrix(rep(diag(K), 2), 2*K, K, byrow = TRUE)

for(mm in 2:K) {
    temp = combn(1:K, m = mm)
    tempmat = matrix(0, ncol(temp), K)
    for(j in 1:ncol(temp)) tempmat[j, temp[, j]] = 1
    Q = rbind(Q, tempmat)
}

Q = Q[1:J,]
```

```
# Setting item parameters and generating attribute profiles
ss = gs = rep(.2, J)
PIs = rep(1/(2^K), 2^K)
CLs = c(1:(2^K)) %*% rmultinom(n = N, size = 1, prob = PIs)

# Defining matrix of possible attribute profiles
As = rep(0,K)

for(j in 1:K) {
    temp = combn(1:K, m = j)
    tempmat = matrix(0, ncol(temp), K)
    for(j in 1:ncol(temp)) tempmat[j, temp[, j]] = 1
    As = rbind(As, tempmat)
}

As = as.matrix(As)

# Sample true attribute profiles
Alphas = As[CLs,]

# Simulate data under DINA model
Y_sim = simcdm::sim_dina_items(Alphas, Q, ss, gs)

## Execute MCMC DINA routine ----

# NOTE: This example uses a small chain length to reduce
# computation time to illustrate the pedagogical concept.
# In a real-life scenario, increase the chain length to
# at least 5,000.

chainLength = 200
burnin = 100

outchain = dina(Y_sim, Q, chain_length = chainLength)

## Summarize posterior samples for g and 1-s ----

chain_samples = burnin:chainLength
mGs = apply(outchain$GamS[, chain_samples], 1, mean)
sGs = apply(outchain$GamS[, chain_samples], 1, sd)
m1mSS = 1 - apply(outchain$SigS[, chain_samples], 1, mean)
s1mSS = apply(outchain$SigS[, chain_samples], 1, sd)
output = cbind(mGs, sGs, m1mSS, s1mSS)
colnames(output) = c('g Est', 'g SE', '1-s Est', '1-s SE')
rownames(output) = paste('Item', 1:J)
print(output, digits = 3)

## Summarize marginal skill distribution ----

# Via posterior samples for latent class proportions
PIoutput = cbind(apply(outchain$PIs, 1, mean), apply(outchain$PIs, 1, sd))
colnames(PIoutput) = c('EST', 'SE')
rownames(PIoutput) = apply(As, 1, paste0, collapse='')
```

```
print(PIoutput, digits = 3)

## End(Not run)
```

---

`DINA_Gibbs` *Deprecated functions*

---

## Description

Functions found within this help documentation have been deprecated.

## Usage

```
DINA_Gibbs(...)
```

## Arguments

| | |
|---|---|
| `...` | Old parameters |

## Details

Deprecated functions

- `DINA_Gibbs` in favor of `dina`

# Index

dina, 2
dina-package, 2
DINA_Gibbs, 6

simcdm::attribute_classes(), *3*
simcdm::sim_dina_items(), *3*