

# Package: antedep (via r-universe)

May 25, 2026

**Title** Antedependence Models for Longitudinal Data

**Version** 0.1.0

**Author** Chenyang Li [aut, cre], Dale Zimmerman [aut, ctb]

**Maintainer** Chenyang Li <chenyang-li@uiowa.edu>

**Description** Fitting, simulation, and inference for antedependence models for longitudinal data, as described in Zimmerman and Nunez-Anton (2009, ISBN:9781420011074). Supports integer-valued antedependence (INAD) models for count data with thinning operators (binomial, Poisson, negative binomial) and flexible innovation distributions (Poisson, Bell, negative binomial), categorical antedependence models for discrete-state longitudinal outcomes, and Gaussian antedependence (AD) models for continuous data. Implements maximum likelihood estimation via time-separable optimization and block coordinate descent, with confidence intervals based on Louis' identity and profile likelihood.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**URL** <https://tanchyking.github.io/antedep/>,  
<https://github.com/TanchyKing/antedep>

**BugReports** <https://github.com/TanchyKing/antedep/issues>

**Imports** nloptr (>= 1.2.0), stats, ggplot2, MASS

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Depends** R (>= 3.5)

**LazyData** true

**VignetteBuilder** knitr

**Config/pak/sysreqs** cmake

**Repository** <https://tanchyking.r-universe.dev>

**Date/Publication** 2026-04-22 16:17:37 UTC

**RemoteUrl** <https://github.com/tanchyking/antedep>

**RemoteRef** HEAD

**RemoteSha** 355a34576c8e5bf6dce875ea92c774f068fa33f5

## Contents

aic_cat . . . . .	3
aic_gau . . . . .	4
aic_inad . . . . .	5
Bell . . . . .	6
bic_cat . . . . .	7
bic_gau . . . . .	8
bic_inad . . . . .	9
bic_order_cat . . . . .	10
bic_order_gau . . . . .	11
bic_order_inad . . . . .	12
bolus_inad . . . . .	13
cattle_growth . . . . .	13
ci_cat . . . . .	14
ci_gau . . . . .	16
ci_inad . . . . .	17
cochlear_implant . . . . .	18
em_cat . . . . .	19
em_gau . . . . .	20
em_inad . . . . .	21
fit_cat . . . . .	23
fit_gau . . . . .	25
fit_inad . . . . .	27
labor_force_cat . . . . .	29
logL_cat . . . . .	29
logL_gau . . . . .	31
logL_inad . . . . .	32
logL_inad_i . . . . .	34
partial_corr . . . . .	35
plot_prism . . . . .	37
plot_profile . . . . .	38
print.cat_ci . . . . .	40
print.cat_fit . . . . .	40
print.cat_lrt . . . . .	41
print.gau_bic_order . . . . .	41
print.gau_ci . . . . .	42
print.gau_contrast_test . . . . .	42
print.gau_fit . . . . .	43

print.gau_homogeneity_test . . . . .	43
print.gau_mean_test . . . . .	44
print.gau_order_test . . . . .	44
print.homogeneity_tests_inad . . . . .	45
print.inad_ci . . . . .	45
print.inad_fit . . . . .	46
print.test_homogeneity_inad . . . . .	46
race_100km . . . . .	47
run_homogeneity_tests_inad . . . . .	47
run_order_tests_cat . . . . .	48
run_stationarity_tests_cat . . . . .	49
run_stationarity_tests_gau . . . . .	50
run_stationarity_tests_inad . . . . .	52
simulate_cat . . . . .	53
simulate_gau . . . . .	55
simulate_inad . . . . .	56
summary.cat_ci . . . . .	58
summary.gau_ci . . . . .	58
summary.inad_ci . . . . .	59
test_contrast_gau . . . . .	59
test_homogeneity_cat . . . . .	61
test_homogeneity_gau . . . . .	63
test_homogeneity_inad . . . . .	64
test_one_sample_gau . . . . .	66
test_order_cat . . . . .	68
test_order_gau . . . . .	70
test_order_inad . . . . .	72
test_stationarity_cat . . . . .	74
test_stationarity_gau . . . . .	76
test_stationarity_inad . . . . .	78
test_timeinvariance_cat . . . . .	80
test_two_sample_gau . . . . .	81

**Index****84**

aic\_cat

*Akaike information criterion for fitted categorical AD models***Description**

Computes AIC using the fitted log likelihood and a parameter count for categorical antedependence parameters.

**Usage**

aic\_cat(fit)

**Arguments**

`fit` A fitted model object of class "cat\_fit" returned by `fit_cat`.

**Details**

The AIC is computed as:

$$AIC = -2 \times \ell + 2k$$

where  $\ell$  is the log-likelihood and  $k$  is the number of free parameters.

**Value**

A numeric scalar AIC value.

**Examples**

```
set.seed(1)
y <- simulate_cat(40, 5, order = 1, n_categories = 2)
fit <- fit_cat(y, order = 1)
aic_cat(fit)
```

---

aic\_gau

*Akaike information criterion for fitted Gaussian AD models*

---

**Description**

Computes AIC using the fitted log likelihood and a parameter count that respects identifiability constraints for the Gaussian antedependence parameters.

**Usage**

```
aic_gau(fit)
```

**Arguments**

`fit` A fitted model object returned by `fit_gau`.

**Details**

The AIC is computed as:

$$AIC = -2 \times \ell + 2k$$

where  $\ell$  is the log-likelihood and  $k$  is the number of free parameters.

This function applies to Gaussian AD fits from `fit_gau`.

**Value**

A numeric scalar AIC value.

**Examples**

```
set.seed(1)
y <- simulate_gau(n_subjects = 30, n_time = 5, order = 1, phi = 0.3)
fit <- fit_gau(y, order = 1)
aic_gau(fit)
```

---

 aic\_inad

*Akaike information criterion for fitted INAD models*


---

**Description**

Computes AIC using the fitted log likelihood and a parameter count that respects structural zeros and identifiability constraints.

**Usage**

```
aic_inad(fit)
```

**Arguments**

`fit` A fitted model object returned by `fit_inad`.

**Details**

The AIC is computed as:

$$AIC = -2 \times \ell + 2k$$

where  $\ell$  is the log-likelihood and  $k$  is the number of free parameters.

**Value**

A numeric scalar AIC value.

**Examples**

```
set.seed(1)
y <- simulate_inad(
  n_subjects = 40,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2
)
fit <- fit_inad(y, order = 1, thinning = "binom", innovation = "pois", max_iter = 20)
aic_inad(fit)
```

---

 Bell

*The Bell distribution*


---

### Description

Density, distribution function, quantile function and random generation for the Bell distribution with parameter theta.

### Usage

```
dbell(x, theta, log = FALSE)
```

```
pbell(x, theta)
```

```
rbell(n, theta, max_z = 100L)
```

```
qbell(p, theta, max_z = 100L)
```

### Arguments

x	vector of nonnegative integers (for dbell and pbell).
theta	scalar nonnegative Bell parameter.
log	logical; if TRUE, probabilities p are given as log(p).
n	number of observations to generate (for rbell).
max_z	maximum support value used for approximation in rbell and qbell.
p	numeric vector of probabilities between 0 and 1 inclusive (for qbell).

### Details

Let  $B_x$  denote the xth Bell number. The Bell distribution has probability mass function

$$P(X = x) = \theta^x \exp(-\exp(\theta) + 1) \frac{B_x}{x!},$$

for nonnegative integers  $x$  and  $\theta \geq 0$ .

For  $\theta > 0$ , the Bell mean is  $E[X] = \theta e^\theta$ . At  $\theta = 0$ , the distribution is degenerate at 0.

The functions follow the standard naming used in base R: dbell for the density, pbell for the distribution function, qbell for the quantile function and rbell for random generation.

### Value

For dbell, a numeric vector of probabilities. For pbell, a numeric vector of cumulative probabilities. For qbell, an integer vector of quantiles. For rbell, an integer vector of random values.

**Examples**

```
dbell(0:5, theta = 1)
pbell(0:5, theta = 1)
qbell(c(0.25, 0.5, 0.9), theta = 1)
set.seed(1)
rbell(10, theta = 1)
```

---

 bic\_cat

*Bayesian information criterion for fitted categorical AD models*


---

**Description**

Computes BIC using the fitted log likelihood and a parameter count for categorical antedependence parameters.

**Usage**

```
bic_cat(fit, n_subjects = NULL)
```

**Arguments**

`fit` A fitted model object of class "cat\_fit" returned by `fit_cat`.  
`n_subjects` Number of subjects. If NULL, extracted from fit.

**Details**

The BIC is computed as:

$$BIC = -2 \times \ell + k \times \log(N)$$

where  $\ell$  is the log-likelihood,  $k$  is the number of free parameters, and  $N$  is the number of subjects.

**Value**

A numeric scalar BIC value.

**Examples**

```
set.seed(1)
y <- simulate_cat(40, 5, order = 1, n_categories = 2)

# Fit models of different orders
fit0 <- fit_cat(y, order = 0)
fit1 <- fit_cat(y, order = 1)
fit2 <- fit_cat(y, order = 2)

# Compare BIC
c(BIC_0 = bic_cat(fit0), BIC_1 = bic_cat(fit1), BIC_2 = bic_cat(fit2))
```

---

bic_gau	<i>Bayesian information criterion for fitted Gaussian AD models</i>
---------	---

---

### Description

Computes BIC using the fitted log likelihood and a parameter count that respects identifiability constraints for the Gaussian antedependence parameters.

### Usage

```
bic_gau(fit, n_subjects = NULL)
```

### Arguments

`fit` A fitted model object returned by `fit_gau`.

`n_subjects` Number of subjects, typically `nrow(y)`. If `NULL`, inferred from `fit$settings$n_subjects`.

### Details

The BIC is computed as:

$$BIC = -2 \times \ell + k \times \log(N)$$

where  $\ell$  is the log-likelihood,  $k$  is the number of free parameters, and  $N$  is the number of subjects.

This function applies to Gaussian AD fits from `fit_gau`. For categorical and INAD models, use `bic_cat` and `bic_inad`.

### Value

A numeric scalar BIC value.

### Examples

```
set.seed(1)
y <- simulate_gau(n_subjects = 30, n_time = 5, order = 1, phi = 0.3)
fit <- fit_gau(y, order = 1)
bic_gau(fit, n_subjects = nrow(y))
```

---

`bic_inad`*Bayesian information criterion for fitted INAD models*

---

### Description

Computes BIC using the fitted log likelihood and a parameter count that respects structural zeros and identifiability constraints.

### Usage

```
bic_inad(fit, n_subjects = NULL)
```

### Arguments

<code>fit</code>	A fitted model object returned by <code>fit_inad</code> .
<code>n_subjects</code>	Number of subjects, typically <code>nrow(y)</code> . If <code>NULL</code> , inferred from <code>fit\$settings\$n_subjects</code> or legacy <code>length(fit\$settings\$blocks)</code> when available (with a warning).

### Details

The BIC is computed as:

$$BIC = -2 \times \ell + k \times \log(N)$$

where  $\ell$  is the log-likelihood,  $k$  is the number of free parameters, and  $N$  is the number of subjects.

### Value

A numeric scalar BIC value.

### Examples

```
set.seed(1)
y <- simulate_inad(
  n_subjects = 40,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2
)
fit <- fit_inad(y, order = 1, thinning = "binom", innovation = "pois", max_iter = 20)
bic_inad(fit, n_subjects = nrow(y))
```

---

 bic\_order\_cat

*BIC-based order selection for categorical AD models*


---

**Description**

Fits AD models of increasing orders and selects the best by BIC.

**Usage**

```
bic_order_cat(
  y,
  max_order = 2,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  criterion = "bic"
)
```

**Arguments**

y	Integer matrix of categorical data (n_subjects x n_time).
max_order	Maximum order to consider. Default is 2.
blocks	Optional block membership vector.
homogeneous	Whether to use homogeneous parameters across blocks.
n_categories	Number of categories (inferred if NULL).
criterion	Which criterion to use: "bic" (default) or "aic".

**Value**

A list containing:

table	Data frame with order, log_l, n_params, aic, bic
bic	Named numeric vector of BIC values by order
best_order	Order with lowest criterion value
criterion	Criterion used for order selection ("bic" or "aic")
fits	List of fitted models

**Examples**

```
y <- simulate_cat(100, 5, order = 1, n_categories = 2)
result <- bic_order_cat(y, max_order = 2)
print(result$table)
print(result$best_order)
```

---

bic\_order\_gau                    *BIC-based order selection for Gaussian AD models*

---

### Description

Fits AD models of increasing orders and selects the best by BIC.

### Usage

```
bic_order_gau(y, max_order = 2L, ...)
```

### Arguments

**y**                    Numeric matrix with `n_subjects` rows and `n_time` columns.  
**max\_order**           Maximum order to consider.  
**...**                Additional arguments passed to [fit\\_gau](#).

### Value

A list with class `gau_bic_order` containing:

**fits** List of fitted models  
**bic** BIC values for each order  
**best\_order** Order with lowest BIC  
**table** Summary table

### See Also

[bic\\_order\\_cat](#), [bic\\_order\\_inad](#), [fit\\_gau](#)

### Examples

```
set.seed(1)
y <- simulate_gau(n_subjects = 80, n_time = 6, order = 1, phi = 0.4)
ord <- bic_order_gau(y, max_order = 2)
ord$best_order
ord$table
```

---

bic_order_inad	<i>BIC-based order selection for INAD models</i>
----------------	--

---

### Description

Fits INAD models across candidate orders and reports BIC-based selection.

### Usage

```
bic_order_inad(
  y,
  max_order = 2,
  thinning = "binom",
  innovation = "pois",
  blocks = NULL,
  ...
)
```

### Arguments

y	Integer matrix.
max_order	Maximum order (1 or 2).
thinning	Thinning operator.
innovation	Innovation distribution.
blocks	Optional block assignments.
...	Additional arguments.

### Value

A list with class "bic\_order\_inad" containing:

**fits** List of fitted INAD models by candidate order

**bic** Named numeric vector of BIC values by order

**best\_order** Order with minimum BIC

**table** Data frame with order, logLik, n\_params, and BIC

**settings** Input and derived settings used for selection

### Examples

```
set.seed(1)
y <- simulate_inad(
  n_subjects = 30,
  n_time = 5,
  order = 1,
  thinning = "binom",
```

```

    innovation = "pois",
    alpha = 0.3,
    theta = 2
  )
ord <- bic_order_inad(y, max_order = 2, thinning = "binom", innovation = "pois", max_iter = 10)
ord$best_order

```

---

 bolus\_inad

*Morphine bolus analgesia counts*


---

### Description

Morphine bolus self administration counts for two treatment groups recorded at 12 four hour time points. The data are stored in matrix form to facilitate use with antedependence models.

### Usage

```
bolus_inad
```

### Format

A list with four components:

**y** integer matrix of dimension N by n\_time containing all subjects and time points

**y\_2mg** integer matrix with rows corresponding to the 2 mg treatment group

**y\_1mg** integer matrix with rows corresponding to the 1 mg treatment group

**blocks** integer vector of length N giving the block or treatment group indicator, 1 for 2 mg and 2 for 1 mg

### Source

Dataset bolus from the **cold** package, converted to matrix form and grouped by treatment.

---

 cattle\_growth

*Cattle growth data (Treatments A and B)*


---

### Description

Longitudinal cattle growth measurements for two treatment groups from Zimmerman and Nunez-Anton antedependence book companion data. This dataset is continuous-response data suitable for Gaussian AD modeling.

### Usage

```
cattle_growth
```

**Format**

A list with five components:

**y** numeric matrix of dimension N by n\_time containing all subjects

**y\_A** numeric matrix for Treatment A subjects

**y\_B** numeric matrix for Treatment B subjects

**blocks** integer vector of length N (1 = Treatment A, 2 = Treatment B)

**time** integer vector of measurement occasions

**Source**

[https://homepage.divms.uiowa.edu/~dzimmer/Data-for-AD/cattle\\_growth\\_data\\_Treatment%20A.txt](https://homepage.divms.uiowa.edu/~dzimmer/Data-for-AD/cattle_growth_data_Treatment%20A.txt) and [https://homepage.divms.uiowa.edu/~dzimmer/Data-for-AD/cattle\\_growth\\_data\\_Treatment%20B.txt](https://homepage.divms.uiowa.edu/~dzimmer/Data-for-AD/cattle_growth_data_Treatment%20B.txt)

---

 ci\_cat

---

*Confidence intervals for fitted categorical AD models*


---

**Description**

Computes Wald-based confidence intervals for the transition probability parameters of a fitted categorical antedependence model.

**Usage**

```
ci_cat(fit, y = NULL, level = 0.95, parameters = "all")
```

**Arguments**

<b>fit</b>	A fitted model object of class "cat_fit" from fit_cat().
<b>y</b>	Optional data matrix. If NULL, fit\$cell_counts is used (observed counts for closed-form fits; expected counts for EM fits).
<b>level</b>	Confidence level (default 0.95).
<b>parameters</b>	Which parameters to compute CIs for: "all" (default), "marginal", or "transition".

**Details**

Confidence intervals are computed using the Wald method based on the asymptotic normality of maximum likelihood estimators.

For a probability estimate  $\hat{\pi}$  based on count N, the standard error is:

$$SE(\hat{\pi}) = \sqrt{\frac{\hat{\pi}(1 - \hat{\pi})}{N}}$$

For conditional probabilities  $\hat{\pi}_{j|i}$  based on conditioning count  $N_i$ , the standard error is:

$$SE(\hat{\pi}_{j|i}) = \sqrt{\frac{\hat{\pi}_{j|i}(1 - \hat{\pi}_{j|i})}{N_i}}$$

The confidence interval is then:

$$\hat{\pi} \pm z_{\alpha/2} \times SE(\hat{\pi})$$

Note: CIs are truncated to the interval from 0 to 1 when they exceed these bounds.

Missing-data fits with `na_action = "marginalize"` are not currently supported because observed cell counts are not stored for that path.

### Value

A list of class "cat\_ci" containing:

marginal	Data frame of CIs for marginal parameters (if requested)
transition	List of data frames of CIs for transition parameters (if requested)
level	Confidence level used
settings	Model settings from fit

### References

Agresti, A. (2013). *Categorical Data Analysis* (3rd ed.). Wiley.

### See Also

[fit\\_cat](#)

### Examples

```
# Fit a model
set.seed(123)
y <- simulate_cat(200, 5, order = 1, n_categories = 2)
fit <- fit_cat(y, order = 1)

# Compute confidence intervals
ci <- ci_cat(fit)
print(ci)

# Just marginal CIs
ci_marg <- ci_cat(fit, parameters = "marginal")
```

---

ci\_gau

*Confidence intervals for fitted Gaussian AD models*


---

**Description**

Computes approximate Wald confidence intervals for selected parameters from a fitted Gaussian AD model.

**Usage**

```
ci_gau(fit, level = 0.95, parameters = "all")
```

**Arguments**

fit	A fitted model object returned by <code>fit_gau</code> .
level	Confidence level between 0 and 1.
parameters	Which parameters to include: "all" (default), "mu", "phi", or "sigma".

**Details**

This helper currently supports complete-data Gaussian AD fits.

Standard errors are based on large-sample approximations:

- $SE(\hat{\mu}_t) \approx \hat{\sigma}_t / \sqrt{n}$
- $SE(\hat{\sigma}_t) \approx \hat{\sigma}_t / \sqrt{2n}$
- $SE(\hat{\phi}) \approx \sqrt{(1 - \hat{\phi}^2)/n}$  for free  $\phi$  entries

**Value**

An object of class `gau_ci`, a list with elements `settings`, `level`, `mu`, `phi`, and `sigma`. Each non-NULL element is a data frame with columns `param`, `est`, `se`, `lower`, `upper`, and `level`.

**See Also**

[fit\\_gau](#), [ci\\_cat](#), [ci\\_inad](#)

**Examples**

```
y <- simulate_gau(n_subjects = 80, n_time = 6, order = 1, phi = 0.4)
fit <- fit_gau(y, order = 1)
ci <- ci_gau(fit)
ci$mu
ci$phi
ci$sigma
```

ci\_inad

*Confidence intervals for fitted INAD models***Description**

Computes confidence intervals for selected parameters from a fitted INAD model. For the fixed effect case, Wald intervals for time varying alpha and theta are computed via Louis identity for supported thinning-innovation combinations. For block effects tau, profile likelihood intervals are computed by fixing one component of tau and re maximizing the log likelihood over nuisance parameters. For negative binomial innovations, Wald intervals for the innovation size parameter are computed using a one dimensional observed information approximation per time point, holding other parameters fixed at their fitted values.

**Usage**

```
ci_inad(
  y,
  fit,
  blocks = NULL,
  level = 0.95,
  idx_time = NULL,
  ridge = 0,
  profile_maxeval = 2500,
  profile_xtol_rel = 1e-06
)
```

**Arguments**

y	Integer matrix with n_subjects rows and n_time columns.
fit	A fitted model object returned by <code>fit_inad</code> .
blocks	Optional integer vector of length n_subjects. Required for block effect intervals. If provided, should match <code>fit\$settings\$blocks</code> .
level	Confidence level between 0 and 1.
idx_time	Optional integer vector of time indices for which to compute intervals. Default is all time points.
ridge	Nonnegative ridge value added to the observed information matrix used for Louis based Wald intervals.
profile_maxeval	Maximum number of function evaluations used in the profile likelihood refits.
profile_xtol_rel	Relative tolerance used in the profile likelihood refits.

**Value**

An object of class `inad_ci`, a list with elements `settings`, `level`, `alpha`, `theta`, `nb_inno_size`, and `tau`. Each non NULL interval element is a data frame with columns `param`, `est`, `lower`, `upper`, and possibly `se` and `width`.

### Examples

```
data("bolus_inad", package = "antedep")
y <- bolus_inad$y
blocks <- bolus_inad$blocks
fit <- fit_inad(y, order = 1, thinning = "binom", innovation = "bell", blocks = blocks)
ci <- ci_inad(y, fit, blocks = blocks)
ci$alpha
ci$theta
ci$tau
```

---

cochlear\_implant

*Cochlear implant speech recognition data*

---

### Description

Longitudinal speech recognition outcomes for two groups (A/B), including incomplete records, from Zimmerman and Nunez-Anton antedependence book companion data. This dataset is continuous-response data suitable for Gaussian AD modeling.

### Usage

```
cochlear_implant
```

### Format

A list with six components:

**y** numeric matrix of dimension N by n\_time containing all subjects

**y\_A** numeric matrix for Group A subjects

**y\_B** numeric matrix for Group B subjects

**blocks** integer vector of length N (1 = Group A, 2 = Group B)

**group** character vector of group labels ("A"/"B")

**time** integer vector of measurement occasions

### Source

[https://homepage.divms.uiowa.edu/~dzimmer/Data-for-AD/speech\\_recognition\\_data.txt](https://homepage.divms.uiowa.edu/~dzimmer/Data-for-AD/speech_recognition_data.txt)

---

em\_cat

*EM algorithm for categorical AD model estimation*


---

### Description

Fits categorical antedependence models with missing outcomes using the Expectation-Maximization (EM) algorithm for orders 0 and 1.

### Usage

```
em_cat(
  y,
  order = 1,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  max_iter = 100,
  tol = 1e-06,
  epsilon = 1e-08,
  safeguard = TRUE,
  verbose = FALSE
)
```

### Arguments

y	Integer matrix with n_subjects rows and n_time columns. Values are category codes in 1, . . . , n_categories; NA is allowed.
order	Antedependence order. Supported values are 0 and 1. Order 2 is not yet implemented in em_cat().
blocks	Optional block/group vector of length n_subjects. Any coding is accepted (e.g., non-sequential integers or factor levels).
homogeneous	Logical. If TRUE, a single parameter set is fitted across blocks. If FALSE, separate parameters are fitted by block.
n_categories	Number of categories. If NULL, inferred from observed data.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance on absolute log-likelihood change.
epsilon	Small positive constant used for smoothing and numerical stability.
safeguard	Logical; if TRUE, apply step-halving when an M-step update decreases observed-data log-likelihood.
verbose	Logical; if TRUE, print EM progress.

## Details

For complete data (no missing values), this function defers to `fit_cat` with closed-form MLEs.

For missing data and orders 0/1, each EM iteration computes expected sufficient statistics with a forward-backward E-step, then updates probabilities by normalized expected counts in the M-step. If `safeguard = TRUE`, a step-halving line search is applied to the M-step update whenever the observed-data likelihood decreases.

A final E-step is run before returning so that `log_1/AIC/BIC` and expected cell counts correspond exactly to the returned parameter values.

## Value

A `cat_fit` object with fields matching `fit_cat`. In EM mode, `cell_counts` stores expected counts from the final E-step, with `settings$cell_counts_type = "expected"`.

## See Also

`fit_cat`, `logL_cat`

## Examples

```
set.seed(1)
y <- simulate_cat(n_subjects = 40, n_time = 5, order = 1, n_categories = 3)
y[sample(length(y), 10)] <- NA
fit <- em_cat(y, order = 1, n_categories = 3, max_iter = 20, tol = 1e-5)
fit$settings$na_action
```

---

em\_gau

*EM algorithm for Gaussian AD model estimation*

---

## Description

Convenience wrapper around `fit_gau` with `na_action = "em"` to provide a parallel entry point to `em_inad`.

## Usage

```
em_gau(  
  y,  
  order = 1,  
  blocks = NULL,  
  estimate_mu = TRUE,  
  max_iter = 100,  
  tol = 1e-06,  
  verbose = FALSE,  
  ...  
)
```

**Arguments**

y	Numeric matrix (n_subjects x n_time), may contain NA.
order	Integer 0, 1, or 2.
blocks	Optional vector of block membership (length n_subjects).
estimate_mu	Logical, whether to estimate mu (default TRUE).
max_iter	Maximum EM iterations.
tol	EM convergence tolerance.
verbose	Logical, print EM progress.
...	Additional arguments passed to <a href="#">fit_gau</a> .

**Details**

This is an alias-style helper for users who prefer explicit em\_\* entry points across model families.

**Value**

An `gau_fit` object as returned by [fit\\_gau](#).

**See Also**

[fit\\_gau](#), [em\\_inad](#), [em\\_cat](#), [fit\\_cat](#)

**Examples**

```
set.seed(1)
y <- simulate_gau(n_subjects = 35, n_time = 5, order = 1, phi = 0.3)
y[sample(length(y), 8)] <- NA
fit <- em_gau(y, order = 1, max_iter = 20, tol = 1e-5)
fit$settings$na_action
```

---

em\_inad

*EM algorithm for INAD model estimation*


---

**Description**

Fits INAD models using the Expectation-Maximization algorithm. This is an alternative to direct likelihood optimization.

**Usage**

```
em_inad(
  y,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  blocks = NULL,
  max_iter = 200,
  tol = 1e-07,
  alpha_init = NULL,
  theta_init = NULL,
  tau_init = NULL,
  nb_inno_size = NULL,
  safeguard = TRUE,
  verbose = FALSE
)
```

**Arguments**

y	Integer matrix with n_subjects rows and n_time columns.
order	Model order (1 or 2). Order 0 does not require EM.
thinning	Thinning operator: "binom", "pois", or "nbinom".
innovation	Innovation distribution: "pois", "bell", or "nbinom".
blocks	Optional integer vector of length n_subjects for block effects.
max_iter	Maximum number of EM iterations.
tol	Convergence tolerance for log-likelihood change.
alpha_init	Optional initial values for alpha parameters.
theta_init	Optional initial values for theta parameters.
tau_init	Optional initial values for tau parameters.
nb_inno_size	Size parameter for negative binomial innovation (if used).
safeguard	Logical; if TRUE, use step-halving when likelihood decreases.
verbose	Logical; if TRUE, print iteration progress.

**Details**

For Gaussian and CAT EM entry points, see [em\\_gau](#) and [em\\_cat](#). For CAT specifically, `fit_cat()` supports `na_action = "em"` for orders 0/1 and `na_action = "marginalize"` for order 2 missing-data fits.

**Value**

A list with class "inad\_fit" containing estimated parameters.

**See Also**

[em\\_gau](#), [em\\_cat](#), [fit\\_inad](#), [fit\\_cat](#)

**Examples**

```

set.seed(1)
y <- simulate_inad(
  n_subjects = 50,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.25,
  theta = 2
)
fit <- em_inad(y, order = 1, thinning = "binom", innovation = "pois", max_iter = 20, tol = 1e-6)
fit$log_l

```

fit\_cat

*Fit categorical antedependence model by maximum likelihood***Description**

Computes maximum likelihood estimates for the parameters of an AD(p) model for categorical longitudinal data. The model is parameterized by transition probabilities, and MLEs are obtained in closed form.

**Usage**

```

fit_cat(
  y,
  order = 1,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  na_action = c("fail", "complete", "marginalize", "em"),
  em_max_iter = 100,
  em_tol = 1e-06,
  em_epsilon = 1e-08,
  em_safeguard = TRUE,
  em_verbose = FALSE
)

```

**Arguments**

y	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns. Each entry should be a category code from 1 to <code>c</code> , where <code>c</code> is the number of categories.
order	Antedependence order <code>p</code> . Must be 0, 1, or 2. Default is 1.
blocks	Optional integer vector of length <code>n_subjects</code> specifying group membership. If <code>NULL</code> , all subjects are treated as one group.

homogeneous	Logical. If TRUE (default), parameters are shared across all groups (blocks are ignored for estimation). If FALSE, separate transition probabilities are estimated for each group.
n_categories	Number of categories. If NULL (default), inferred from the maximum value in y.
na_action	Handling of missing values in y. One of "fail" (default, error if any missing), "complete" (drop subjects with any missing values), or "marginalize" (maximize observed-data likelihood by integrating over missing outcomes), or "em" (use <code>em_cat</code> for orders 0 and 1).
em_max_iter	Maximum EM iterations used when na_action = "em".
em_tol	EM convergence tolerance used when na_action = "em".
em_epsilon	Numerical smoothing constant used when na_action = "em".
em_safeguard	Logical; if TRUE, use step-halving safeguard in <code>em_cat</code> when na_action = "em".
em_verbose	Logical; print EM progress when na_action = "em".

### Details

For AD(p), the model decomposes as:

$$P(Y_1, \dots, Y_n) = P(Y_1, \dots, Y_p) \times \prod_{k=p+1}^n P(Y_k | Y_{k-p}, \dots, Y_{k-1})$$

MLEs are computed as empirical proportions:

- Marginal/joint probabilities: count / N
- Transition probabilities: conditional count / marginal count

Empty cells receive probability 0 (if denominator is also 0).

When na\_action = "em", `fit_cat()` dispatches to `em_cat`. In that case, `em_safeguard` controls step-halving protection against likelihood-decreasing updates, and returned `log_l/AIC/BIC/cell_counts` are synchronized via a final E-step under the returned parameters. For order = 2, na\_action = "em" is not available and errors explicitly; use na\_action = "marginalize".

### Value

A list of class "cat\_fit" containing:

marginal	List of marginal/joint probabilities for initial time points
transition	List of transition probability arrays for k = p+1 to n
log_l	Log-likelihood at MLE
aic	Akaike Information Criterion
bic	Bayesian Information Criterion
n_params	Number of free parameters
cell_counts	List of cell counts: observed counts for closed-form fits (na_action = "fail"/"complete"), expected counts from the final E-step for EM fits (na_action = "em"), and NULL for na_action = "marginalize"
convergence	Optimizer convergence code (0 for closed-form solutions)
settings	List of model settings

## References

Xie, Y. and Zimmerman, D. L. (2013). Antedependence models for nonstationary categorical longitudinal data with ignorable missingness: likelihood-based inference. *Statistics in Medicine*, 32, 3274-3289.

## Examples

```
# Simulate binary AD(1) data
set.seed(123)
y <- simulate_cat(n_subjects = 100, n_time = 5, order = 1, n_categories = 2)

# Fit model
fit <- fit_cat(y, order = 1)
print(fit)

# Compare orders
fit0 <- fit_cat(y, order = 0)
fit1 <- fit_cat(y, order = 1)
fit2 <- fit_cat(y, order = 2)
c(AIC_0 = fit0$aic, AIC_1 = fit1$aic, AIC_2 = fit2$aic)

# EM fit with missing data
y_miss <- y
y_miss[sample(length(y_miss), size = round(0.15 * length(y_miss)))] <- NA
fit_em <- fit_cat(
  y_miss,
  order = 1,
  na_action = "em",
  em_max_iter = 80,
  em_tol = 1e-6
)
fit_em$settings$n_iter
fit_em$settings$cell_counts_type
```

---

fit\_gau

*Fit Gaussian antedependence model by maximum likelihood*


---

## Description

Fits an AD(0), AD(1), or AD(2) model for Gaussian longitudinal data by maximum likelihood. Missing values can be handled by complete-case deletion or by EM (see [em\\_gau](#) for an explicit EM wrapper).

## Usage

```
fit_gau(
  y,
```

```

order = 1,
blocks = NULL,
na_action = c("fail", "complete", "em"),
estimate_mu = TRUE,
em_max_iter = 100,
em_tol = 1e-06,
em_verbose = FALSE,
...
)

```

### Arguments

<code>y</code>	Numeric matrix ( <code>n_subjects</code> x <code>n_time</code> ). May contain NA.
<code>order</code>	Integer 0, 1, or 2.
<code>blocks</code>	Optional vector of block membership (length <code>n_subjects</code> ).
<code>na_action</code>	One of "fail", "complete", or "em".
<code>estimate_mu</code>	Logical, whether to estimate mu (default TRUE).
<code>em_max_iter</code>	Maximum EM iterations (only used when <code>na_action</code> = "em").
<code>em_tol</code>	EM convergence tolerance (only used when <code>na_action</code> = "em").
<code>em_verbose</code>	Logical, print EM progress (only used when <code>na_action</code> = "em").
<code>...</code>	Passed through to the EM fitter.

### Details

For missing data with `na_action` = "em", AD orders 0 and 1 are the primary production path. AD order 2 is available, but the current EM implementation uses simplified second-order updates and should be treated as provisional for high-stakes inference.

For observed-data likelihood evaluation under MAR without fitting, use `logL_gau` with `na_action` = "marginalize". In contrast, `fit_gau` handles missingness via complete-case fitting (`na_action` = "complete") or EM (`na_action` = "em").

### Value

A list with components including `mu`, `phi`, `sigma`, `tau`, `log_l`, `n_obs`, `n_missing`.

### See Also

[em\\_gau](#), [fit\\_cat](#), [fit\\_inad](#)

### Examples

```

set.seed(1)
y <- simulate_gau(n_subjects = 30, n_time = 5, order = 1, phi = 0.3)
fit <- fit_gau(y, order = 1)
fit$log_l

y_miss <- y

```

```

y_miss[1, 2] <- NA
fit_em <- fit_gau(y_miss, order = 1, na_action = "em", em_max_iter = 20)
fit_em$settings$na_action

```

---

fit\_inad

*Fit INAD antedependence model by maximum likelihood*


---

## Description

Fits INAD models by maximum likelihood.

## Usage

```

fit_inad(
  y,
  order = 1,
  thinning = c("binom", "pois", "nbinom"),
  innovation = c("pois", "bell", "nbinom"),
  blocks = NULL,
  max_iter = 50,
  tol = 1e-06,
  verbose = FALSE,
  init_alpha = NULL,
  init_theta = NULL,
  init_tau = 0.4,
  init_nb_inno_size = 1,
  nb_inno_size_ub = 50,
  na_action = c("fail", "complete", "marginalize")
)

```

## Arguments

y	Integer matrix n_sub by n_time.
order	Integer in {0, 1, 2}.
thinning	One of "binom", "pois", "nbinom".
innovation	One of "pois", "bell", "nbinom".
blocks	Optional integer vector length n_sub. Default NULL.
max_iter	Max iterations for FE coordinate descent.
tol	Tolerance for FE log likelihood stopping.
verbose	Logical.
init_alpha	Optional initial alpha. For order 1 numeric length 1 or n_time. For order 2 matrix n_time by 2 or list(alpha1, alpha2).
init_theta	Optional initial theta numeric length 1 or n_time.

init_tau	Optional initial tau. Scalar expands to $c(0, x, \dots, x)$ . Vector forces first to 0.
init_nb_inno_size	Optional initial size for innovation nbinom, length 1 or n_time.
nb_inno_size_ub	Upper bound for innovation negative binomial size parameter when innovation = "nbinom". Default is 50.
na_action	How to handle missing values: <ul style="list-style-type: none"> <li>• "fail": stop if any NA is present.</li> <li>• "complete": fit using complete-case subjects only.</li> <li>• "marginalize": maximize observed-data likelihood under MAR.</li> </ul>

### Details

No fixed effect: time separable optimization using  $\log L_{\text{inad}_i}$  with theta eliminated by moment equations for order 1 and 2.

Fixed effect: block coordinate descent using nloptr BOBYQA, updating tau, alpha, theta, and nb\_inno\_size if needed.

### Value

A list of class "inad\_fit" containing:

**alpha** Estimated antedependence parameter(s)  
**theta** Estimated innovation parameter(s)  
**tau** Estimated block effects (if applicable)  
**nb\_inno\_size** Estimated innovation NB size parameter(s), when innovation = "nbinom"  
**log\_l** Maximized log-likelihood  
**aic** Akaike information criterion  
**bic** Bayesian information criterion  
**n\_params** Number of free parameters  
**convergence** Convergence code  
**settings** Model and fitting settings

### Examples

```
set.seed(1)
y <- simulate_inad(
  n_subjects = 60,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2
)
fit <- fit_inad(y, order = 1, thinning = "binom", innovation = "pois", max_iter = 20)
fit$log_l
```

---

labor_force_cat	<i>Labor force longitudinal categorical data (Table 1)</i>
-----------------	--

---

**Description**

Five-year employment-status sequences reconstructed from Table 1 in the labor-force example used in Xie and Zimmerman score/Wald antedependence testing work. Category coding is 1 = employed, 2 = unemployed.

**Usage**

```
labor_force_cat
```

**Format**

A list with five components:

**y** integer matrix of dimension N by 5 containing expanded subject-level sequences

**counts** data frame with columns Y1, Y2, Y3, Y4, Y5, Count

**n\_categories** number of categories (2)

**time** integer vector of calendar years (1967:1971)

**status\_labels** character vector c("employed", "unemployed")

**Source**

Table 1 (labor-force example) from: Xie, Y. and Zimmerman, D. L. (2013). Score and Wald tests for antedependence in categorical longitudinal data.

---

logL_cat	<i>Log-likelihood for categorical AD models (with missing data support)</i>
----------	---

---

**Description**

Evaluates the log-likelihood of an AD(p) model for categorical longitudinal data at given parameter values.

**Usage**

```
logL_cat(
  y,
  order,
  marginal,
  transition = NULL,
  blocks = NULL,
  homogeneous = TRUE,
```

```

n_categories = NULL,
na_action = c("fail", "complete", "marginalize")
)

```

### Arguments

<code>y</code>	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns. Each entry should be a category code from 1 to <code>c</code> .
<code>order</code>	Ante-dependence order <code>p</code> . Must be 0, 1, or 2.
<code>marginal</code>	List of marginal/joint probabilities for initial time points. Structure depends on order (see Details).
<code>transition</code>	List of transition probability arrays for time points $k = p+1$ to $n$ . Each element should be an array of dimension $c^p \times c$ where the last dimension corresponds to the current time point.
<code>blocks</code>	Optional integer vector of length <code>n_subjects</code> specifying group membership. Required if <code>homogeneous = FALSE</code> .
<code>homogeneous</code>	Logical. If TRUE (default), same parameters used for all subjects. If FALSE, <code>marginal</code> and <code>transition</code> should be lists indexed by block.
<code>n_categories</code>	Number of categories. If NULL, inferred from data.
<code>na_action</code>	Handling of missing values in <code>y</code> . One of "fail" (default, error if any missing), "complete" (drop subjects with any missing values), or "marginalize" (integrate over missing categorical outcomes under the AD model).

### Details

The log-likelihood for AD( $p$ ) decomposes into contributions from initial time points and transition time points.

For order 0 (independence), the log-likelihood is the sum of log marginal probabilities at each time point.

Parameter structure for `marginal`:

- Order 0: List with elements  $t_1, t_2, \dots, t_n$ , each a vector of length `c`
- Order 1: List with element  $t_1$  (vector of length `c`)
- Order 2: List with  $t_1$  (vector),  $t_{2\_given\_1to1}$  (`c` x `c` matrix)

Parameter structure for `transition`:

- Order 0: Not used (NULL or empty list)
- Order 1: List with elements  $t_2, t_3, \dots, t_n$ , each `c` x `c` matrix
- Order 2: List with elements  $t_3, t_4, \dots, t_n$ , each `c` x `c` x `c` array

### Value

Scalar log-likelihood value.

## References

Xie, Y. and Zimmerman, D. L. (2013). Antedependence models for nonstationary categorical longitudinal data with ignorable missingness: likelihood-based inference. *Statistics in Medicine*, 32, 3274-3289.

## Examples

```
set.seed(1)
y <- simulate_cat(n_subjects = 40, n_time = 5, order = 1, n_categories = 3)
fit <- fit_cat(y, order = 1, n_categories = 3)
logL_cat(
  y = y,
  order = 1,
  marginal = fit$marginal,
  transition = fit$transition,
  n_categories = 3
)
```

---

logL\_gau

---

*Log-likelihood for Gaussian AD models (with missing data support)*


---

## Description

Computes the log-likelihood for Gaussian antedependence models of order 0, 1, or 2. Supports missing data under MAR assumption via `na_action` parameter.

## Usage

```
logL_gau(
  y,
  order,
  mu = NULL,
  phi = NULL,
  sigma = NULL,
  blocks = NULL,
  tau = 0,
  na_action = c("fail", "complete", "marginalize")
)
```

## Arguments

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns. May contain NA.
<code>order</code>	Antedependence order, one of 0, 1, or 2.
<code>mu</code>	Mean vector (length <code>n_time</code> ).
<code>phi</code>	Dependence coefficient(s). For order 1: vector of length <code>n_time-1</code> . For order 2: matrix with 2 columns or vector of length $2*(n\_time-2)$ .

sigma	Innovation standard deviations (length n_time).
blocks	Integer vector of block membership (length n_subjects), or NULL.
tau	Block effects, first element constrained to zero
na_action	How to handle missing values: <ul style="list-style-type: none"> <li>• fail: Error if any NA is present (default)</li> <li>• complete: Use only complete cases</li> <li>• marginalize: Compute observed-data likelihood</li> </ul>

### Details

For complete data (no NA), all three na\_action options give the same result.

For missing data:

- marginalize: Uses MVN marginalization to compute  $P(Y_{\text{obs}})$ . This is the correct observed-data likelihood for MAR missing data.
- complete: Removes subjects with any missing values. May lose information.
- fail: Stops with error. Useful to ensure no missing data present.

### Value

Scalar log-likelihood value.

### Examples

```
set.seed(1)
y <- simulate_gau(n_subjects = 30, n_time = 5, order = 1, phi = 0.3)
fit <- fit_gau(y, order = 1)
logL_gau(y, order = 1, mu = fit$mu, phi = fit$phi, sigma = fit$sigma)
```

---

logL\_inad

*Log-likelihood for INAD models (with missing data support)*

---

### Description

If blocks is NULL, this computes the log likelihood as the sum of per time contributions from logL\_inad\_i for computational convenience.

### Usage

```
logL_inad(
  y,
  order = 1,
  thinning = c("binom", "pois", "nbinom"),
  innovation = c("pois", "bell", "nbinom"),
  alpha,
```

```

    theta,
    nb_inno_size = NULL,
    blocks = NULL,
    tau = 0,
    na_action = c("fail", "complete", "marginalize")
  )

```

### Arguments

y	Integer matrix n_sub by n_time.
order	Integer in {0, 1, 2}.
thinning	One of "binom", "pois", "nbinom".
innovation	One of "pois", "bell", "nbinom".
alpha	Thinning parameters. For order 1, numeric length 1 or n_time. For order 2, either a matrix n_time by 2 or a list(alpha1, alpha2).
theta	Innovation parameter(s). Numeric length 1 or n_time. For Poisson and negative binomial innovations, this is the innovation mean. For Bell innovations, this is the Bell rate parameter (mean $\theta e^\theta$ ).
nb_inno_size	Size parameter for innovation "nbinom". Numeric length 1 or n_time.
blocks	Optional integer vector of length n_sub. If NULL, no fixed effect.
tau	Optional numeric vector. Only used if blocks is not NULL.
na_action	How to handle missing values: <ul style="list-style-type: none"> <li>• "fail": error if any NA is present.</li> <li>• "complete": use only complete-case subjects.</li> <li>• "marginalize": observed-data likelihood under MAR via truncated-state recursion.</li> </ul>

### Value

A scalar log likelihood.

### Examples

```

set.seed(1)
y <- simulate_inad(
  n_subjects = 60,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2
)
fit <- fit_inad(y, order = 1, thinning = "binom", innovation = "pois", max_iter = 20)
logL_inad(
  y,
  order = 1,

```

```

    thinning = "binom",
    innovation = "pois",
    alpha = fit$alpha,
    theta = fit$theta
  )

```

---

logL\_inad\_i

*INAD log likelihood contribution at time i (no fixed effect)*


---

### Description

Returns the time  $i$  contribution, summed over subjects, for the no fixed effect model.

### Usage

```

logL_inad_i(
  y,
  i,
  order = 1,
  thinning = c("binom", "pois", "nbinom"),
  innovation = c("pois", "bell", "nbinom"),
  alpha,
  theta,
  nb_inno_size = NULL
)

```

### Arguments

<code>y</code>	Integer matrix <code>n_sub</code> by <code>n_time</code> .
<code>i</code>	Time index in <code>1..ncol(y)</code> .
<code>order</code>	Integer in <code>{0, 1, 2}</code> .
<code>thinning</code>	One of "binom", "pois", "nbinom".
<code>innovation</code>	One of "pois", "bell", "nbinom".
<code>alpha</code>	Thinning parameters. For order 1, numeric length 1 or <code>n_time</code> . For order 2, either a matrix <code>n_time</code> by 2 or a list(alpha1, alpha2).
<code>theta</code>	Innovation parameter at time $i$ , or a vector length 1 or <code>n_time</code> . For Poisson and negative binomial innovations, this is the innovation mean. For Bell innovations, this is the Bell rate parameter (mean $\theta e^\theta$ ).
<code>nb_inno_size</code>	Size parameter for innovation "nbinom". Numeric length 1 or <code>n_time</code> .

### Value

A scalar log likelihood contribution for time  $i$ .

**Examples**

```

set.seed(1)
y <- simulate_inad(
  n_subjects = 50,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2
)
fit <- fit_inad(y, order = 1, thinning = "binom", innovation = "pois", max_iter = 20)
logL_inad_i(
  y = y,
  i = 3,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = fit$alpha,
  theta = fit$theta
)

```

---

partial\_corr

---

*Compute intervenor-adjusted partial correlation matrix*


---

**Description**

Computes the partial correlation between  $Y[i]$  and  $Y[j]$  adjusting for the "intervenor" variables  $Y[i+1], \dots, Y[j-1]$ . Under an antedependence model of order  $p$ , partial correlations for  $|i-j| > p$  should be approximately zero.

**Usage**

```
partial_corr(y, test = FALSE, n_digits = 3)
```

**Arguments**

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>test</code>	Logical; if TRUE, returns significance flags based on approximate threshold $2/\sqrt{n_{\text{eff}}}$ where $n_{\text{eff}} = n_{\text{subjects}} - (\text{lag} - 1)$ . Default FALSE.
<code>n_digits</code>	Integer; number of decimal places for rounding. Default 3.

**Details**

The intervenor-adjusted partial correlation between  $Y[i]$  and  $Y[j]$  ( $i < j$ ) is computed as the correlation between the residuals from regressing  $Y[i]$  and  $Y[j]$  on the intervenor set  $Y[i+1], \dots, Y[j-1]$ .

For adjacent time points ( $|i-j| = 1$ ), the partial correlation equals the ordinary correlation since there are no intervenors.

The diagonal of both returned matrices contains variances (not correlations). This keeps scale information available alongside correlation structure.

The significance test uses an approximate threshold of  $2/\sqrt{n_{\text{eff}}}$ , which corresponds roughly to a 95% confidence bound under normality. This is a rough screening tool, not a formal hypothesis test.

### Value

A list with components:

correlation	Matrix with correlations (upper triangle) and variances (diagonal)
partial_correlation	Matrix with partial correlations (lower triangle) and variances (diagonal)
significant	(If test=TRUE) Matrix flagging significant partial correlations (1 = significant)
n_subjects	Number of subjects
n_time	Number of time points

### References

Zimmerman, D. L. and Nunez-Anton, V. (2009). Antedependence Models for Longitudinal Data. CRC Press.

### See Also

[plot\\_prism](#) for visual diagnostics

### Examples

```
data("bolus_inad")
pc <- partial_corr(bolus_inad$y, test = TRUE)

# View partial correlations (lower triangle)
pc$partial_correlation

# Extract variances from the diagonal
variances <- diag(pc$partial_correlation)

# Check which are "significant" (rough screen for AD order)
pc$significant
```

---

plot_prism	<i>PRISM plot (Partial Residual Intervenor Scatterplot Matrix)</i>
------------	--

---

### Description

Creates a matrix of scatterplots for diagnosing antedependence structure. The upper triangle shows ordinary scatterplots of  $Y[i]$  vs  $Y[j]$ . The lower triangle shows PRISM plots: residuals from regressing  $Y[i]$  and  $Y[j]$  on the intervenor variables  $Y[i+1], \dots, Y[j-1]$ .

### Usage

```
plot_prism(
  y,
  time_labels = NULL,
  pch = 20,
  cex = 0.6,
  col_upper = "steelblue",
  col_lower = "firebrick",
  main = "PRISM Diagnostic Plot"
)
```

### Arguments

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>time_labels</code>	Optional character vector of time point labels. Default uses column names or "T1", "T2", etc.
<code>pch</code>	Point character for scatterplots. Default 20 (filled circle).
<code>cex</code>	Point size. Default 0.6.
<code>col_upper</code>	Color for upper triangle plots. Default "steelblue".
<code>col_lower</code>	Color for lower triangle (PRISM) plots. Default "firebrick".
<code>main</code>	Overall title. Default "PRISM Diagnostic Plot".

### Details

Under an antedependence model of order  $p$ , the partial correlation between  $Y[i]$  and  $Y[j]$  given the intervenors should be zero when  $|i-j| > p$ . This means PRISM plots in the lower triangle should show no association for lags greater than  $p$ .

Interpretation:

- Upper triangle: Shows marginal associations between time points
- Lower triangle (PRISM): Shows conditional associations after removing effects of intervenor variables
- If AD(1) holds: Only the first sub-diagonal of lower triangle should show association
- If AD(2) holds: First two sub-diagonals should show association

**Value**

Invisibly returns NULL. Called for side effect (plotting).

**References**

Zimmerman, D. L. and Nunez-Anton, V. (2009). Antedependence Models for Longitudinal Data. CRC Press. Chapter 2.

**See Also**

[partial\\_corr](#) for numerical partial correlations

**Examples**

```
data("bolus_inad")
plot_prism(bolus_inad$y)

# With custom labels
plot_prism(bolus_inad$y, time_labels = paste0("Hour ", seq(0, 44, by = 4)))
```

---

plot\_profile

*Profile plot (spaghetti plot) for longitudinal data*

---

**Description**

Creates a profile plot showing individual subject trajectories with overlaid mean trajectory and standard deviation bands.

**Usage**

```
plot_profile(
  y,
  time_points = NULL,
  blocks = NULL,
  block_labels = NULL,
  title = "Profile Plot",
  xlab = "Time",
  ylab = "Measurement",
  ylim = NULL,
  show_sd = TRUE,
  individual_alpha = 0.3,
  individual_color = "grey50",
  mean_color = "blue",
  sd_color = "red",
  mean_lwd = 2
)
```

**Arguments**

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns, or a data frame with measurements.
<code>time_points</code>	Optional numeric vector of time points for x-axis. Default uses <code>1:n_time</code> or attempts to extract from column names.
<code>blocks</code>	Optional integer vector of block memberships for stratified plotting. If provided, creates separate panels for each block.
<code>block_labels</code>	Optional character vector of labels for blocks.
<code>title</code>	Plot title. Default "Profile Plot".
<code>xlab</code>	X-axis label. Default "Time".
<code>ylab</code>	Y-axis label. Default "Measurement".
<code>ylim</code>	Optional y-axis limits as <code>c(min, max)</code> .
<code>show_sd</code>	Logical; if TRUE (default), show +/- 1 SD error bars.
<code>individual_alpha</code>	Alpha (transparency) for individual trajectories. Default 0.3.
<code>individual_color</code>	Color for individual trajectories. Default "grey50".
<code>mean_color</code>	Color for mean trajectory. Default "blue".
<code>sd_color</code>	Color for SD error bars. Default "red".
<code>mean_lwd</code>	Line width for mean trajectory. Default 2.

**Details**

This function provides a quick visual summary of longitudinal data showing:

- Individual subject trajectories (light grey lines)
- Mean trajectory across subjects (bold colored line)
- +/- 1 standard deviation bands (error bars)

When `blocks` is provided, the plot is faceted by block membership, allowing comparison of trajectories across treatment groups or other strata.

**Value**

A `ggplot2` object (invisibly). Called primarily for side effect (plotting).

**Examples**

```
data("bolus_inad")

# Basic profile plot
plot_profile(bolus_inad$y)

# With block stratification
plot_profile(bolus_inad$y, blocks = bolus_inad$blocks,
             block_labels = c("2mg", "1mg"))
```

```
# Customized
plot_profile(bolus_inad$y,
             time_points = seq(0, 44, by = 4),
             title = "Bolus Counts Over Time",
             xlab = "Hours", ylab = "Count")
```

---

print.cat\_ci                    *Print method for cat\_ci objects*

---

### Description

Print method for cat\_ci objects

### Usage

```
## S3 method for class 'cat_ci'
print(x, ...)
```

### Arguments

x                    A cat\_ci object  
 ...                  Additional arguments (ignored)

### Value

Invisibly returns x.

---

print.cat\_fit                    *Print method for cat\_fit objects*

---

### Description

Print method for cat\_fit objects

### Usage

```
## S3 method for class 'cat_fit'
print(x, ...)
```

### Arguments

x                    A cat\_fit object  
 ...                  Additional arguments (ignored)

**Value**

Invisibly returns x.

---

*print.cat\_lrt*      *Print method for cat\_lrt objects*

---

**Description**

Print method for cat\_lrt objects

**Usage**

```
## S3 method for class 'cat_lrt'  
print(x, ...)
```

**Arguments**

x                    A cat\_lrt object  
...                  Additional arguments (ignored)

**Value**

Invisibly returns x.

---

*print.gau\_bic\_order*      *Print method for BIC order selection*

---

**Description**

Print method for BIC order selection

**Usage**

```
## S3 method for class 'gau_bic_order'  
print(x, ...)
```

**Arguments**

x                    Object of class gau\_bic\_order.  
...                  Unused.

**Value**

Invisibly returns x.

---

print.gau_ci	<i>Print method for AD confidence intervals</i>
--------------	---

---

**Description**

Print method for AD confidence intervals

**Usage**

```
## S3 method for class 'gau_ci'  
print(x, ...)
```

**Arguments**

x	An object of class gau_ci.
...	Unused.

**Value**

The input object, invisibly.

---

print.gau_contrast_test	<i>Print method for AD contrast test</i>
-------------------------	--

---

**Description**

Print method for AD contrast test

**Usage**

```
## S3 method for class 'gau_contrast_test'  
print(x, ...)
```

**Arguments**

x	Object of class gau_contrast_test.
...	Unused.

**Value**

Invisibly returns x.

---

print.gau_fit	<i>Print method for gau_fit objects</i>
---------------	---

---

**Description**

Print method for gau\_fit objects.

**Usage**

```
## S3 method for class 'gau_fit'  
print(x, ...)
```

**Arguments**

x	A gau_fit object.
...	Additional arguments (ignored).

**Value**

The input object, invisibly.

---

print.gau_homogeneity_test	<i>Print method for AD homogeneity test</i>
----------------------------	---

---

**Description**

Print method for AD homogeneity test

**Usage**

```
## S3 method for class 'gau_homogeneity_test'  
print(x, ...)
```

**Arguments**

x	Object of class gau_homogeneity_test.
...	Unused.

**Value**

Invisibly returns x.

---

`print.gau_mean_test` *Print method for AD mean test*

---

**Description**

Print method for AD mean test

**Usage**

```
## S3 method for class 'gau_mean_test'  
print(x, ...)
```

**Arguments**

<code>x</code>	Object of class <code>gau_mean_test</code> .
<code>...</code>	Unused.

**Value**

Invisibly returns `x`.

---

`print.gau_order_test` *Print method for AD order test*

---

**Description**

Print method for AD order test

**Usage**

```
## S3 method for class 'gau_order_test'  
print(x, ...)
```

**Arguments**

<code>x</code>	Object of class <code>gau_order_test</code> .
<code>...</code>	Unused.

**Value**

Invisibly returns `x`.

---

```
print.homogeneity_tests_inad  
    Print method for homogeneity_tests_inad
```

---

### **Description**

Print method for homogeneity\_tests\_inad

### **Usage**

```
## S3 method for class 'homogeneity_tests_inad'  
print(x, digits = 4, ...)
```

### **Arguments**

x	Object of class homogeneity_tests_inad
digits	Number of digits for printing
...	Unused

### **Value**

Invisibly returns x.

---

```
print.inad_ci    Print method for INAD confidence intervals
```

---

### **Description**

Print method for INAD confidence intervals

### **Usage**

```
## S3 method for class 'inad_ci'  
print(x, ...)
```

### **Arguments**

x	An object of class inad_ci.
...	Unused.

### **Value**

The input object, invisibly.

---

```
print.inad_fit          Print method for INAD model fits
```

---

**Description**

Print method for INAD model fits

**Usage**

```
## S3 method for class 'inad_fit'
print(x, digits = 4, ...)
```

**Arguments**

x	An object of class inad_fit.
digits	Number of digits to print.
...	Unused.

**Value**

The input object, invisibly.

---

```
print.test_homogeneity_inad
          Print method for test_homogeneity_inad
```

---

**Description**

Print method for test\_homogeneity\_inad

**Usage**

```
## S3 method for class 'test_homogeneity_inad'
print(x, digits = 4, ...)
```

**Arguments**

x	Object of class test_homogeneity_inad
digits	Number of digits for printing
...	Unused

**Value**

Invisibly returns x.

---

race_100km	<i>100km race split-time data</i>
------------	-----------------------------------

---

**Description**

Split times (in minutes) from a 100km race example with 10 consecutive sections. This is continuous-response longitudinal data suitable for Gaussian AD modeling.

**Usage**

```
race_100km
```

**Format**

A list with five components:

**y** numeric matrix of dimension N by 10 containing section split times

**age** numeric vector of subject ages (may include missing values)

**subject** integer subject identifiers

**time** integer vector of section indices (1:10)

**section\_labels** character vector of split-time column names

**Source**

Combined table extracted from textbook race example data, stored in data-raw/external/100km\_race\_combined\_extract

---

run_homogeneity_tests_inad
----------------------------

*Run all homogeneity tests for INAD*

---

**Description**

Convenience function to run all three homogeneity tests at once and return a summary.

**Usage**

```
run_homogeneity_tests_inad(  
  y,  
  blocks,  
  order = 1,  
  thinning = "binom",  
  innovation = "pois",  
  ...  
)
```

**Arguments**

<code>y</code>	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>blocks</code>	Integer vector of length <code>n_subjects</code> specifying group membership.
<code>order</code>	Antedependence order (0, 1, or 2).
<code>thinning</code>	Thinning operator: "binom", "pois", or "nbinom".
<code>innovation</code>	Innovation distribution: "pois", "bell", or "nbinom".
<code>...</code>	Additional arguments passed to <code>fit_inad</code> .

**Value**

A list with class "homogeneity\_tests\_inad" containing results for all three tests and a summary table.

**Examples**

```
data("bolus_inad")
tests <- run_homogeneity_tests_inad(bolus_inad$y, bolus_inad$blocks,
                                   order = 1, thinning = "nbinom",
                                   innovation = "bell")

print(tests)
```

---

`run_order_tests_cat`    *Run all pairwise order tests*

---

**Description**

Performs sequential likelihood ratio tests for AD orders 0 vs 1, 1 vs 2, etc.

**Usage**

```
run_order_tests_cat(
  y,
  max_order = 2,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  test = c("lrt", "score", "mlrt", "wald")
)
```

**Arguments**

y	Integer matrix of categorical data (n_subjects x n_time).
max_order	Maximum order to test. Default is 2.
blocks	Optional block membership vector.
homogeneous	Whether to use homogeneous parameters across blocks.
n_categories	Number of categories (inferred if NULL).
test	Type of test statistic for each pairwise comparison. One of "lrt" (default), "score", "mlrt", or "wald". Passed to <code>test_order_cat</code> .

**Details**

This function performs forward selection: starting from order 0, it tests whether increasing the order provides significant improvement. The selected order is the highest order where the test was significant (at  $\alpha = 0.05$ ).

**Value**

A list containing:

tests	List of <code>test_order_cat</code> results for each comparison
table	Summary data frame with all comparisons
fits	List of all fitted models
selected_order	Recommended order based on sequential testing at $\alpha=0.05$

**Examples**

```
y <- simulate_cat(200, 6, order = 1, n_categories = 2)
result <- run_order_tests_cat(y, max_order = 2)
print(result$table)
cat("Selected order:", result$selected_order, "\n")
```

---

run\_stationarity\_tests\_cat

*Run all stationarity-related tests for categorical AD*

---

**Description**

Performs tests for time-invariance and stationarity constraints. For order = 1, the stationarity test corresponds to strict stationarity; for order > 1, it tests marginal-constancy plus time-invariant transitions. Currently supports complete data only.

**Usage**

```
run_stationarity_tests_cat(
  y,
  order = 1,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  test = c("lrt", "score", "mlrt")
)
```

**Arguments**

<code>y</code>	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns. Each entry should be a category code from 1 to <code>c</code> .
<code>order</code>	Antedependence order <code>p</code> . Default is 1.
<code>blocks</code>	Optional integer vector of length <code>n_subjects</code> specifying group membership.
<code>homogeneous</code>	Logical. If TRUE (default), parameters are shared across all groups.
<code>n_categories</code>	Number of categories. If NULL, inferred from data.
<code>test</code>	Type of test statistic. One of "lrt" (default), "score", or "mlrt".

**Value**

A list containing:

**time\_invariance** Result of `test_timeinvariance_cat`

**stationarity** Result of `test_stationarity_cat`

**table** Summary data frame

**Examples**

```
y <- simulate_cat(200, 6, order = 1, n_categories = 2)
result <- run_stationarity_tests_cat(y, order = 1)
print(result$table)
```

---

run\_stationarity\_tests\_gau

*Run all stationarity-related tests for Gaussian AD*

---

**Description**

Runs a standard set of stationarity constraints for Gaussian AD models.

**Usage**

```
run_stationarity_tests_gau(  
  y,  
  order = 1L,  
  blocks = NULL,  
  verbose = FALSE,  
  max_iter = 2000L,  
  rel_tol = 1e-08,  
  ...  
)
```

**Arguments**

y	Numeric matrix with n_subjects rows and n_time columns.
order	Antedependence order (0, 1, or 2).
blocks	Optional vector of block memberships (length n_subjects).
verbose	Logical; if TRUE, prints progress.
max_iter	Maximum number of optimization iterations for constrained fits.
rel_tol	Relative tolerance for constrained optimization.
...	Additional arguments passed to <a href="#">fit_gau</a> for the unconstrained fit.

**Value**

A list with class "stationarity\_tests\_gau" containing:

**fit\_unconstrained** Unconstrained Gaussian AD fit  
**tests** Named list of [test\\_stationarity\\_gau](#) results  
**summary** Summary table of all constraints

**See Also**

[test\\_stationarity\\_gau](#)

**Examples**

```
set.seed(1)  
y <- simulate_gau(n_subjects = 80, n_time = 6, order = 1, phi = 0.4, sigma = 1)  
out <- run_stationarity_tests_gau(y, order = 1, verbose = FALSE)  
out$summary
```

---

`run_stationarity_tests_inad`*Run all stationarity-related tests for INAD*

---

**Description**

Run all stationarity-related tests for INAD

**Usage**

```
run_stationarity_tests_inad(  
  y,  
  order = 1,  
  thinning = "binom",  
  innovation = "pois",  
  blocks = NULL,  
  verbose = FALSE,  
  ...  
)
```

**Arguments**

<code>y</code>	Integer matrix.
<code>order</code>	Model order (1 or 2).
<code>thinning</code>	Thinning operator.
<code>innovation</code>	Innovation distribution.
<code>blocks</code>	Optional block assignments.
<code>verbose</code>	Logical.
<code>...</code>	Additional arguments.

**Value**

A list with class "stationarity\_tests\_inad".

**Examples**

```
set.seed(1)  
y <- simulate_inad(  
  n_subjects = 25,  
  n_time = 5,  
  order = 1,  
  thinning = "binom",  
  innovation = "pois",  
  alpha = 0.3,  
  theta = 2  
)
```

```

out <- run_stationarity_tests_inad(
  y,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  verbose = FALSE,
  max_iter = 15
)
out$summary

```

---

simulate\_cat

*Simulate categorical antedependence series*


---

### Description

Generate simulated longitudinal categorical data from an AD(p) model with specified transition probabilities.

### Usage

```

simulate_cat(
  n_subjects,
  n_time,
  order = 1,
  n_categories = 2,
  marginal = NULL,
  transition = NULL,
  blocks = NULL,
  homogeneous = TRUE,
  seed = NULL
)

```

### Arguments

n_subjects	Number of subjects to simulate.
n_time	Number of time points.
order	Antedependence order p. Must be 0, 1, or 2. Default is 1.
n_categories	Number of categories c. Default is 2 (binary).
marginal	List of marginal/joint probabilities for initial time points. If NULL, uniform probabilities are used. See Details for structure.
transition	List of transition probability arrays for time points k = p+1 to n. If NULL, uniform transitions are used. See Details.
blocks	Optional integer vector of length n_subjects specifying group membership. Used with homogeneous = FALSE.
homogeneous	Logical. If TRUE (default), same parameters for all subjects. If FALSE, marginal and transition should be lists indexed by block.
seed	Optional random seed for reproducibility.

## Details

Data are simulated sequentially:

1. For  $k = 1$ : Draw  $Y(1)$  from marginal distribution
2. For  $k = 2$  to  $p$ : Draw  $Y(k)$  conditional on  $Y(1), \dots, Y(k-1)$
3. For  $k = p+1$  to  $n$ : Draw  $Y(k)$  conditional on  $Y(k-p), \dots, Y(k-1)$

Parameter structure for marginal:

- Order 0: List with elements  $t_1, t_2, \dots, t_n$ , each a vector of length  $c$  summing to 1
- Order 1: List with element  $t_1$  (vector of length  $c$ )
- Order 2: List with  $t_1$  (vector),  $t_{2\_given\_1to1}$  ( $c \times c$  matrix where rows represent conditioning values and columns represent outcomes)

Parameter structure for transition:

- Order 0: Not used (NULL)
- Order 1: List with elements  $t_2, t_3, \dots, t_n$ , each  $c \times c$  matrix where rows are previous values and columns are current values (rows sum to 1)
- Order 2: List with elements  $t_3, t_4, \dots, t_n$ , each  $c \times c \times c$  array where first two indices are conditioning values and third is outcome

## Value

Integer matrix with  $n\_subjects$  rows and  $n\_time$  columns, where each entry is a category code from 1 to  $c$ .

## References

Xie, Y. and Zimmerman, D. L. (2013). Antedependence models for nonstationary categorical longitudinal data with ignorable missingness: likelihood-based inference. *Statistics in Medicine*, 32, 3274-3289.

## Examples

```
y <- simulate_cat(n_subjects = 30, n_time = 5, order = 1, n_categories = 3, seed = 1)
dim(y)
```

---

simulate_gau	<i>Simulate Gaussian antedependence series</i>
--------------	--

---

**Description**

Generate longitudinal continuous data from a Gaussian antedependence (AD) model of order 0, 1, or 2 using a conditional regression on predecessors.

**Usage**

```
simulate_gau(
  n_subjects,
  n_time,
  order = 1L,
  mu = NULL,
  phi = NULL,
  sigma = NULL,
  blocks = NULL,
  tau = 0,
  seed = NULL
)
```

**Arguments**

<code>n_subjects</code>	number of subjects
<code>n_time</code>	number of time points
<code>order</code>	antedependence order, 0, 1 or 2
<code>mu</code>	mean parameter; NULL, scalar, or length <code>n_time</code>
<code>phi</code>	dependence parameter; ignored when <code>order = 0</code> . For <code>order = 1</code> , NULL, scalar, or length <code>n_time</code> . For <code>order = 2</code> , NULL or a 2 by <code>n_time</code> matrix.
<code>sigma</code>	innovation standard deviation; NULL, scalar, or length <code>n_time</code>
<code>blocks</code>	integer vector of length <code>n_subjects</code> indicating block membership for each subject; if NULL, no block effect is applied
<code>tau</code>	group effect vector indexed by block; <code>tau[1]</code> is forced to 0. If scalar <code>x</code> , it is expanded to <code>c(0, x, ..., x)</code> with length equal to the number of blocks
<code>seed</code>	optional random seed for reproducibility

**Details**

For `order = 0`, each time point is generated independently as  $Y[, t] = \mu[t] + \tau[\text{block}] + \text{eps}$ , with  $\text{eps} \sim N(0, \sigma[t]^2)$ .

For `order = 1`, for  $t \geq 2$ :  $Y[, t] = m_t + \phi[t] * (Y[, t - 1] - m_{\{t-1\}}) + \text{eps}_t$ , where  $m_t = \mu[t] + \tau[\text{block}]$  and  $\text{eps}_t \sim N(0, \sigma[t]^2)$ .

For `order = 2`, for  $t \geq 3$ :  $Y[, t] = m_t + \phi[1, t] * (Y[, t - 1] - m_{\{t-1\}}) + \phi[2, t] * (Y[, t - 2] - m_{\{t-2\}})$

If `blocks` is provided, each subject `s` belongs to a block and receives a mean shift `tau[blocks[s]]`. `tau[1]` is forced to 0.

**Value**

numeric matrix with dimension n\_subjects by n\_time

**Examples**

```
y <- simulate_gau(
  n_subjects = 20,
  n_time = 6,
  order = 1,
  phi = 0.4,
  seed = 42
)
dim(y)
```

---

simulate\_inad

*Simulate INAD antedependence series*

---

**Description**

Generate longitudinal count data from an INAD model using a thinning operator and an innovation distribution.

**Usage**

```
simulate_inad(
  n_subjects,
  n_time,
  order = 1L,
  thinning = c("binom", "pois", "nbinom"),
  innovation = c("pois", "bell", "nbinom"),
  alpha = NULL,
  theta = NULL,
  nb_inno_size = NULL,
  blocks = NULL,
  tau = 0,
  seed = NULL
)
```

**Arguments**

n_subjects	number of subjects
n_time	number of time points
order	antedependence order, 0, 1 or 2
thinning	thinning operator, one of "binom", "pois", "nbinom"
innovation	innovation distribution, one of "pois", "bell", "nbinom"

alpha	thinning parameter or vector or matrix; if NULL, defaults are used depending on the order
theta	innovation parameter or vector; if NULL, defaults are used depending on the innovation type. For Poisson and negative binomial innovations, theta is the innovation mean parameter. For Bell innovations, theta is the Bell rate parameter, with innovation mean $\theta * \exp(\theta)$ .
nb_inno_size	size (dispersion) parameter for negative binomial innovations when innovation = "nbinom"; must be positive. If NULL, defaults to 1. Larger values correspond to less overdispersion (approaching Poisson as size $\rightarrow$ Inf).
blocks	integer vector of length n_subjects indicating block membership for each subject; if NULL, no block effect is applied
tau	group effect vector indexed by block; tau[1] is forced to 0. If scalar x, it is expanded to c(0, x, ..., x) with length equal to the number of blocks
seed	optional random seed for reproducibility

### Details

Time 1 observations are generated from the innovation distribution alone. For times 2 to n\_time, counts are generated as thinning of previous counts plus independent innovations. When order = 0, all time points are generated from the innovation distribution and the thinning operator and alpha are ignored.

If blocks is provided, innovations include a block effect. For Poisson and negative binomial innovations, the innovation mean is  $\theta[t] + \tau[\text{blocks}[i]]$ . For Bell innovations, the innovation mean is  $\theta[t] * \exp(\theta[t]) + \tau[\text{blocks}[i]]$ .

### Value

integer matrix of counts with dimension n\_subjects by n\_time

### Examples

```
y <- simulate_inad(
  n_subjects = 20,
  n_time = 6,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2,
  seed = 42
)
dim(y)
```

---

summary.cat_ci	<i>Summary method for cat_ci objects</i>
----------------	--

---

**Description**

Summary method for cat\_ci objects

**Usage**

```
## S3 method for class 'cat_ci'
summary(object, ...)
```

**Arguments**

object	A cat_ci object
...	Additional arguments (ignored)

**Value**

A data frame summarizing all CIs

---

summary.gau_ci	<i>Summary method for gau_ci objects</i>
----------------	--

---

**Description**

Summary method for gau\_ci objects

**Usage**

```
## S3 method for class 'gau_ci'
summary(object, ...)
```

**Arguments**

object	A gau_ci object.
...	Unused.

**Value**

A data frame stacking available confidence intervals with columns param, component, est, se, lower, upper, and level. Returns NULL if no intervals are available.

---

summary.inad_ci	<i>Summary method for inad_ci objects</i>
-----------------	---

---

**Description**

Summary method for inad\_ci objects

**Usage**

```
## S3 method for class 'inad_ci'
summary(object, ...)
```

**Arguments**

object	An inad_ci object.
...	Unused.

**Value**

A data frame stacking available confidence intervals with columns param, component, est, se, lower, upper, and level. Returns NULL if no intervals are available.

---

test_contrast_gau	<i>Test linear hypotheses on the mean under antedependence</i>
-------------------	--

---

**Description**

Tests the null hypothesis  $C * \mu = c$  for a specified contrast matrix  $C$  and vector  $c$ , under an AD( $p$ ) covariance structure. This implements Theorem 7.2 of Zimmerman & Núñez-Antón (2009).

**Usage**

```
test_contrast_gau(y, C, c = NULL, p = 1L)
```

**Arguments**

y	Numeric matrix with n_subjects rows and n_time columns.
C	Contrast matrix with c rows and n_time columns, where c is the number of contrasts being tested. Rows must be linearly independent.
c	Right-hand side vector of length equal to nrow(C). Default is a vector of zeros.
p	Antedependence order of the covariance structure. This is the same order parameter named order in <a href="#">fit_gau</a> .

## Details

The Wald test statistic (Theorem 7.2) is:

$$(C\bar{Y} - c)^T (C\hat{\Sigma}C^T)^{-1} (C\bar{Y} - c)$$

where  $\hat{\Sigma}$  is the REML estimator of the covariance matrix under the AD(p) model.

Common examples include:

- Testing if mean is constant: C is the first-difference matrix
- Testing for linear trend: C tests deviations from linearity

## Value

A list with class `gau_contrast_test` containing:

**method** Inference method used ("wald").

**C** Contrast matrix

**c** Right-hand side vector

**mu\_hat** Estimated mean vector

**contrast\_est** Estimated value of  $C * \mu$

**statistic** Wald test statistic

**df** Degrees of freedom (number of contrasts)

**p\_value** P-value from chi-square distribution

## References

Zimmerman, D.L. and Núñez-Antón, V. (2009). Antedependence Models for Longitudinal Data. Chapman & Hall/CRC. Chapter 7.

## Examples

```
y <- simulate_gau(n_subjects = 50, n_time = 5, order = 1)

# Test if mean is constant (all differences = 0)
# C is 4x5 matrix of first differences
C <- matrix(0, nrow = 4, ncol = 5)
for (i in 1:4) {
  C[i, i] <- 1
  C[i, i+1] <- -1
}
test <- test_contrast_gau(y, C = C, p = 1)
print(test)
```

---

test\_homogeneity\_cat *Likelihood ratio test for homogeneity across groups (categorical AD data)*

---

### Description

Tests whether multiple groups share the same transition probability parameters in a categorical antedependence model.

### Usage

```
test_homogeneity_cat(
  y = NULL,
  blocks = NULL,
  order = 1,
  n_categories = NULL,
  fit_null = NULL,
  fit_alt = NULL,
  test = c("lrt", "score", "mlrt")
)
```

### Arguments

y	Integer matrix with n_subjects rows and n_time columns. Each entry should be a category code from 1 to c. Can be NULL if both fit_null and fit_alt are provided.
blocks	Integer vector of length n_subjects specifying group membership. Required unless pre-fitted models are provided.
order	Antedependence order p. Default is 1.
n_categories	Number of categories. If NULL, inferred from data.
fit_null	Optional pre-fitted homogeneous model (class "cat_fit" with homogeneous = TRUE). If provided, y is not required for fitting under H0.
fit_alt	Optional pre-fitted heterogeneous model (class "cat_fit" with homogeneous = FALSE). If provided, y is not required for fitting under H1.
test	Type of test statistic. One of "lrt" (default), "score", or "mlrt".

### Details

The null hypothesis is that all G groups share the same transition probability parameters:

$$H_0 : \pi^{(1)} = \pi^{(2)} = \dots = \pi^{(G)}$$

The alternative hypothesis allows each group to have its own parameters.

The degrees of freedom are:

$$df = (G - 1) \times k$$

where G is the number of groups and k is the number of free parameters per population.

**Value**

A list of class "cat\_lrt" containing:

**method** Inference method used: one of "lrt", "score", "mlrt", or "wald".

**lrt\_stat** Likelihood ratio test statistic

**df** Degrees of freedom

**p\_value** P-value from chi-square distribution

**fit\_null** Fitted homogeneous model (H0)

**fit\_alt** Fitted heterogeneous model (H1)

**n\_groups** Number of groups

**table** Summary data frame

**References**

Xie, Y. and Zimmerman, D. L. (2013). Antedependence models for nonstationary categorical longitudinal data with ignorable missingness: likelihood-based inference. *Statistics in Medicine*, 32, 3274-3289.

**See Also**

[fit\\_cat](#), [test\\_order\\_cat](#)

**Examples**

```
# Simulate data with different transition probabilities for two groups
set.seed(123)
marg1 <- list(t1 = c(0.7, 0.3))
marg2 <- list(t1 = c(0.4, 0.6))
trans1 <- list(t2 = matrix(c(0.9, 0.1, 0.2, 0.8), 2, byrow = TRUE),
  t3 = matrix(c(0.9, 0.1, 0.2, 0.8), 2, byrow = TRUE))
trans2 <- list(t2 = matrix(c(0.5, 0.5, 0.5, 0.5), 2, byrow = TRUE),
  t3 = matrix(c(0.5, 0.5, 0.5, 0.5), 2, byrow = TRUE))

y1 <- simulate_cat(100, 3, order = 1, n_categories = 2,
  marginal = marg1, transition = trans1)
y2 <- simulate_cat(100, 3, order = 1, n_categories = 2,
  marginal = marg2, transition = trans2)
y <- rbind(y1, y2)
blocks <- c(rep(1, 100), rep(2, 100))

# Test homogeneity
test <- test_homogeneity_cat(y, blocks, order = 1)
print(test)
```

---

test_homogeneity_gau	<i>Likelihood ratio test for homogeneity across groups (Gaussian AD data)</i>
----------------------	---

---

### Description

Tests the null hypothesis that  $G$  groups have the same  $AD(p)$  covariance structure against the alternative that they have different  $AD(p)$  structures. This implements Theorem 6.6 of Zimmerman & Núñez-Antón (2009).

### Usage

```
test_homogeneity_gau(y, blocks, p = 1L, use_modified = TRUE)
```

### Arguments

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>blocks</code>	Integer vector of length <code>n_subjects</code> indicating group membership.
<code>p</code>	Antedependence order. This is the same order parameter named <code>order</code> in <a href="#">fit_gau</a> .
<code>use_modified</code>	Logical. If <code>TRUE</code> (default), use modified test statistic for better small-sample approximation.

### Details

The test compares:

- $H_0$ : All  $G$  groups have the same  $AD(p)$  covariance matrix  $\Sigma(\theta)$
- $H_1$ : Groups have different  $AD(p)$  covariance matrices  $\Sigma(\theta_g)$

The likelihood ratio test statistic (Theorem 6.6) involves comparing pooled and within-group RSS values. The degrees of freedom are  $(G-1)(2n - p)(p + 1)/2$ .

This test is useful for determining whether a common covariance structure can be assumed across treatment groups before performing mean comparisons.

### Value

A list with class `gau_homogeneity_test` containing:

**method** Inference method used ("lrt").

**statistic** Test statistic value

**statistic\_modified** Modified test statistic (if `use_modified = TRUE`)

**df** Degrees of freedom

**p\_value** P-value from chi-square distribution

**p\_value\_modified** P-value from modified test

**G** Number of groups

**group\_sizes** Sample sizes for each group

**order** Antedependence order

**References**

- Zimmerman, D.L. and Núñez-Antón, V. (2009). Antedependence Models for Longitudinal Data. Chapman & Hall/CRC. Section 6.4.
- Kenward, M.G. (1987). A method for comparing profiles of repeated measurements. Applied Statistics, 36, 296-308.

**See Also**

[test\\_order\\_gau](#), [test\\_two\\_sample\\_gau](#)

**Examples**

```
# Simulate data from two groups with same covariance
n1 <- 30
n2 <- 35
y1 <- simulate_gau(n1, n_time = 6, order = 1, phi = 0.5, sigma = 1)
y2 <- simulate_gau(n2, n_time = 6, order = 1, phi = 0.5, sigma = 1)
y <- rbind(y1, y2)
blocks <- c(rep(1, n1), rep(2, n2))

# Test homogeneity
test <- test_homogeneity_gau(y, blocks, p = 1)
print(test)
```

---

test\_homogeneity\_inad *Likelihood ratio test for homogeneity across groups (INAD data)*

---

**Description**

Tests hypotheses about parameter equality across treatment or grouping factors in integer-valued antedependence models. Implements the homogeneity testing framework from Section 3.7 of Li & Zimmerman (2026).

**Usage**

```
test_homogeneity_inad(
  y,
  blocks,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  test = c("all", "mean", "dependence"),
  fit_pooled = NULL,
  fit_inadfe = NULL,
  fit_hetero = NULL,
  ...
)
```

**Arguments**

<code>y</code>	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>blocks</code>	Integer vector of length <code>n_subjects</code> specifying group membership.
<code>order</code>	Antependence order (0, 1, or 2).
<code>thinning</code>	Thinning operator: "binom", "pois", or "nbinom".
<code>innovation</code>	Innovation distribution: "pois", "bell", or "nbinom".
<code>test</code>	Type of homogeneity test: <ul style="list-style-type: none"> <li>• "all": Tests M1 (pooled) vs M3 (fully heterogeneous)</li> <li>• "mean": Tests M1 (pooled) vs M2 (shared dependence, different means)</li> <li>• "dependence": Tests M2 (INADFE) vs M3 (fully heterogeneous)</li> </ul>
<code>fit_pooled</code>	Optional pre-computed pooled fit (M1).
<code>fit_inadfe</code>	Optional pre-computed INADFE fit (M2).
<code>fit_hetero</code>	Optional pre-computed heterogeneous fit (M3).
<code>...</code>	Additional arguments passed to <code>fit_inad</code> .

**Details**

The function supports three nested model comparisons as described in the paper:

**M1 (Pooled):** All parameters are common across groups. This corresponds to fitting `fit_inad(y, blocks = NULL)`.

**M2 (INADFE):** The thinning parameters  $\alpha$  are shared across groups, but innovation means differ via block effects  $\tau$ . This is the standard INADFE model fitted via `fit_inad(y, blocks = blocks)`.

**M3 (Fully Heterogeneous):** Both  $\alpha$  and  $\theta$  parameters can differ across groups. This is fitted by running separate `fit_inad` calls for each group.

The three test types correspond to:

- "all": H0: M1 vs H1: M3 (complete homogeneity vs complete heterogeneity)
- "mean": H0: M1 vs H1: M2 (test for group differences in means only)
- "dependence": H0: M2 vs H1: M3 (test for group differences in dependence)

Degrees of freedom are computed as the difference in free parameters between the null and alternative models.

**Value**

A list with class "test\_homogeneity\_inad" containing:

**method** Inference method used ("lrt").

**statistic** Test statistic value

**lrt\_stat** Likelihood ratio test statistic

**df** Degrees of freedom

**p\_value** P-value from chi-square distribution

**test** Type of test performed

**fit\_null** Fitted model under H0  
**fit\_alt** Fitted model under H1  
**bic\_null** BIC under H0  
**bic\_alt** BIC under H1  
**bic\_selected** Which model BIC prefers  
**table** Summary data frame

## References

Li, C. and Zimmerman, D.L. (2026). Integer-valued antedependence models for longitudinal count data. *Biostatistics*. Section 3.7.

## See Also

[fit\\_inad](#), [test\\_order\\_inad](#), [test\\_stationarity\\_inad](#)

## Examples

```
data("bolus_inad")
y <- bolus_inad$y
blocks <- bolus_inad$blocks

# Test for any group differences (M1 vs M3)
test_all <- test_homogeneity_inad(y, blocks, order = 1,
                                thinning = "nbinom", innovation = "bell",
                                test = "all")

print(test_all)

# Test only for mean differences (M1 vs M2)
test_mean <- test_homogeneity_inad(y, blocks, order = 1,
                                  thinning = "nbinom", innovation = "bell",
                                  test = "mean")

print(test_mean)

# Test for dependence differences given different means (M2 vs M3)
test_dep <- test_homogeneity_inad(y, blocks, order = 1,
                                  thinning = "nbinom", innovation = "bell",
                                  test = "dependence")

print(test_dep)
```

**Description**

Tests the null hypothesis that the mean vector equals a specified value  $\mu = \mu_0$  against the alternative  $\mu \neq \mu_0$ , under an AD(p) covariance structure. This implements Theorem 7.1 of Zimmerman & Núñez-Antón (2009).

**Usage**

```
test_one_sample_gau(y, mu0, p = 1L, order = NULL, use_modified = TRUE)
```

**Arguments**

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>mu0</code>	Hypothesized mean vector under the null (length <code>n_time</code> ).
<code>p</code>	Antedependence order of the covariance structure. This is the same order parameter named <code>order</code> in <code>fit_gau</code> .
<code>order</code>	Optional alias for <code>p</code> . Supply only one of <code>p</code> or <code>order</code> .
<code>use_modified</code>	Logical. If TRUE (default), use the modified test statistic (formula 7.7) for better small-sample approximation.

**Details**

The test exploits the AD structure to gain power over tests that don't assume any covariance structure. The likelihood ratio test statistic (Theorem 7.1) is:

$$N \sum_{i=1}^n [\log RSS_i(\mu_0) - \log RSS_i(\hat{\mu})]$$

where  $RSS_i(\mu)$  is the residual sum of squares from the regression of  $Y_i - \mu_i$  on its  $p$  predecessors  $Y_{(i-1)} - \mu_{(i-1)}$ , ...,  $Y_{(i-p)} - \mu_{(i-p)}$ .

The test has  $n$  degrees of freedom (one for each component of  $\mu$ ).

**Value**

A list with class `gau_mean_test` containing:

- method** Inference method used ("lrt").
- test\_type** "one-sample"
- mu0** Hypothesized mean under null
- mu\_hat** MLE of mean (sample mean)
- statistic** Test statistic value
- statistic\_modified** Modified test statistic (if `use_modified = TRUE`)
- df** Degrees of freedom (`n_time`)
- p\_value** P-value from chi-square distribution
- p\_value\_modified** P-value from modified test
- order** Antedependence order used

**References**

Zimmerman, D.L. and Núñez-Antón, V. (2009). Antedependence Models for Longitudinal Data. Chapman & Hall/CRC. Chapter 7.

**See Also**

[test\\_two\\_sample\\_gau](#), [test\\_order\\_gau](#)

**Examples**

```
# Simulate data with known mean
mu_true <- c(10, 11, 12, 13, 14, 15)
y <- simulate_gau(n_subjects = 50, n_time = 6, order = 1, mu = mu_true)

# Test if mean is zero
test <- test_one_sample_gau(y, mu0 = rep(0, 6), p = 1)
print(test)

# Test if mean equals true value (should not reject)
test2 <- test_one_sample_gau(y, mu0 = mu_true, p = 1)
print(test2)
```

---

test\_order\_cat

*Likelihood ratio test for antedependence order (categorical AD data)*


---

**Description**

Tests whether a higher-order AD model provides significantly better fit than a lower-order model for categorical longitudinal data.

**Usage**

```
test_order_cat(
  y = NULL,
  order_null = 0,
  order_alt = 1,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  fit_null = NULL,
  fit_alt = NULL,
  test = c("lrt", "score", "mlrt", "wald")
)
```

**Arguments**

<code>y</code>	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns. Each entry should be a category code from 1 to <code>c</code> . Can be NULL if both <code>fit_null</code> and <code>fit_alt</code> are provided.
<code>order_null</code>	Order under the null hypothesis (default 0).
<code>order_alt</code>	Order under the alternative hypothesis (default 1). Must be greater than <code>order_null</code> .
<code>blocks</code>	Optional integer vector of length <code>n_subjects</code> specifying group membership.
<code>homogeneous</code>	Logical. If TRUE (default), parameters are shared across all groups.
<code>n_categories</code>	Number of categories. If NULL, inferred from data.
<code>fit_null</code>	Optional pre-fitted model under null hypothesis (class "cat_fit"). If provided, <code>y</code> is not required for fitting under H0.
<code>fit_alt</code>	Optional pre-fitted model under alternative hypothesis. If provided, <code>y</code> is not required for fitting under H1.
<code>test</code>	Type of test statistic. One of "lrt" (default), "score", "mlrt", or "wald".

**Details**

The likelihood ratio test statistic is:

$$\lambda = -2[\ell_0 - \ell_1]$$

where  $\ell_0$  and  $\ell_1$  are the maximized log-likelihoods under the null and alternative hypotheses.

Under H0,  $\lambda$  follows a chi-square distribution with degrees of freedom equal to the difference in the number of free parameters.

For testing AD(p) vs AD(p+1), the degrees of freedom are:

$$df = (c - 1)^2 \times c^p \times (n - p - 1)$$

where  $c$  is the number of categories and  $n$  is the number of time points.

If `y` contains missing values and models are fit internally, this function defaults to `na_action = "marginalize"` for fitting. Score- and Wald-based variants currently require complete data.

**Value**

A list of class "cat\_lrt" containing:

**method** Inference method used: one of "lrt", "score", "mlrt", or "wald".

**lrt\_stat** Likelihood ratio test statistic

**df** Degrees of freedom

**p\_value** P-value from chi-square distribution

**fit\_null** Fitted model under H0

**fit\_alt** Fitted model under H1

**order\_null** Order under null

**order\_alt** Order under alternative

**table** Summary data frame

## References

Xie, Y. and Zimmerman, D. L. (2013). Antedependence models for nonstationary categorical longitudinal data with ignorable missingness: likelihood-based inference. *Statistics in Medicine*, 32, 3274-3289.

## See Also

[fit\\_cat](#), [bic\\_order\\_cat](#)

## Examples

```
# Simulate AD(1) data
set.seed(123)
y <- simulate_cat(200, 6, order = 1, n_categories = 2)

# Test AD(0) vs AD(1)
test_01 <- test_order_cat(y, order_null = 0, order_alt = 1)
print(test_01$table)

# Test AD(1) vs AD(2)
test_12 <- test_order_cat(y, order_null = 1, order_alt = 2)
print(test_12$table)

# Using pre-fitted models
fit0 <- fit_cat(y, order = 0)
fit1 <- fit_cat(y, order = 1)
test_prefitted <- test_order_cat(fit_null = fit0, fit_alt = fit1)
```

---

test\_order\_gau

*Likelihood ratio test for antedependence order (Gaussian AD data)*

---

## Description

Tests the null hypothesis that the data follow an AD(p) model against the alternative that they follow an AD(p+q) model, using the likelihood ratio test described in Theorem 6.4 and 6.5 of Zimmerman & Núñez-Antón (2009).

## Usage

```
test_order_gau(
  y,
  p = 0L,
  q = 1L,
  mu = NULL,
  use_modified = TRUE,
  order_null = NULL,
  order_alt = NULL
)
```

**Arguments**

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>p</code>	Order under the null hypothesis (default 0). This is the same antedependence order parameter named <code>order</code> in <code>fit_gau</code> .
<code>q</code>	Order increment under the alternative (default 1, so alternative is AD(p+q)).
<code>mu</code>	Optional mean vector. If NULL, the saturated mean (sample means) is used.
<code>use_modified</code>	Logical. If TRUE (default), use the modified test statistic (formula 6.9) which has better small-sample properties.
<code>order_null</code>	Optional alias for <code>p</code> (null order).
<code>order_alt</code>	Optional absolute order under the alternative. When supplied, <code>q</code> is computed as <code>order_alt - p</code> .

**Details**

The test is based on the intervenor-adjusted sample partial correlations. Under the null hypothesis AD(p), the partial correlations  $r_{(i,i-k|(i-k+1:i-1))}$  should be zero for  $k > p$ .

The likelihood ratio test statistic (Theorem 6.4) is:

$$-N \sum_{j=1}^q \sum_{i=p+j+1}^n \log(1 - r_{i,i-p-j \cdot (i-p-j+1:i-1)}^2)$$

which is asymptotically chi-square with  $(2n - 2p - q - 1)(q/2)$  degrees of freedom.

The modified test (formula 6.9) adjusts for small-sample bias using Kenward's (1987) correction.

**Value**

A list with class `gau_order_test` containing:

<b>method</b>	Inference method used ("lrt").
<b>p</b>	Order under null hypothesis
<b>q</b>	Order increment
<b>statistic</b>	Test statistic value
<b>statistic_modified</b>	Modified test statistic (if <code>use_modified = TRUE</code> )
<b>df</b>	Degrees of freedom
<b>p_value</b>	P-value from chi-square distribution
<b>p_value_modified</b>	P-value from modified test (if <code>use_modified = TRUE</code> )
<b>n_subjects</b>	Number of subjects
<b>n_time</b>	Number of time points

**References**

- Zimmerman, D.L. and Núñez-Antón, V. (2009). Antedependence Models for Longitudinal Data. Chapman & Hall/CRC. Chapter 6.
- Kenward, M.G. (1987). A method for comparing profiles of repeated measurements. Applied Statistics, 36, 296-308.

**See Also**

[test\\_one\\_sample\\_gau](#), [test\\_homogeneity\\_gau](#)

**Examples**

```
# Simulate AD(1) data
y <- simulate_gau(n_subjects = 50, n_time = 6, order = 1, phi = 0.5)

# Test AD(0) vs AD(1)
test01 <- test_order_gau(y, p = 0, q = 1)
print(test01)

# Test AD(1) vs AD(2)
test12 <- test_order_gau(y, p = 1, q = 1)
print(test12)
```

---

test_order_inad	<i>Likelihood ratio test for antedependence order (INAD data)</i>
-----------------	---

---

**Description**

Performs a likelihood ratio test comparing INAD models of different orders.

**Usage**

```
test_order_inad(
  y,
  order_null = 0,
  order_alt = 1,
  thinning = "binom",
  innovation = "pois",
  blocks = NULL,
  use_chibar = TRUE,
  weights = NULL,
  fit_null = NULL,
  fit_alt = NULL,
  ...
)
```

**Arguments**

y	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
order_null	Order under null hypothesis (0 or 1).
order_alt	Order under alternative hypothesis (1 or 2). Must be <code>order_null + 1</code> .
thinning	Thinning operator: "binom", "pois", or "nbinom".

innovation	Innovation distribution: "pois", "bell", or "nbinom".
blocks	Optional integer vector for block effects.
use_chibar	Logical; if TRUE, use chi-bar-square for boundary test.
weights	Optional weights for chi-bar-square mixture.
fit_null	Optional pre-computed null fit.
fit_alt	Optional pre-computed alternative fit.
...	Additional arguments passed to fit_inad.

### Details

The test compares nested INAD models of orders `order_null` and `order_alt = order_null + 1` using:

$$\lambda = 2(\ell_{alt} - \ell_{null})$$

where  $\ell_{null}$  and  $\ell_{alt}$  are maximized log-likelihoods under the null and alternative models.

The default p-value uses the chi-square approximation with degrees of freedom matching the number of additional dependence parameters introduced under the higher-order model. When `use_chibar = TRUE`, a chi-bar-square mixture p-value is also reported for boundary-aware inference.

Missing-data inputs are supported through the same `na_action` options available in `fit_inad`. If `y` has missing values and `na_action` is not supplied via `...`, this function defaults to `na_action = "marginalize"`.

### Value

A list with class `"test_order_inad"` containing:

<b>method</b>	Inference method used ("lrt").
<b>fit_null</b>	Fitted model under H0
<b>fit_alt</b>	Fitted model under H1
<b>statistic</b>	Test statistic value
<b>lrt_stat</b>	Likelihood ratio test statistic
<b>df</b>	Degrees of freedom
<b>p_value</b>	Chi-square p-value
<b>p_value_chibar</b>	Chi-bar-square p-value (if <code>use_chibar = TRUE</code> )
<b>bic_null</b>	BIC under H0
<b>bic_alt</b>	BIC under H1
<b>bic_selected</b>	Which model BIC prefers
<b>table</b>	Two-row model comparison table
<b>settings</b>	Input and derived settings for the test

### References

Li, C. and Zimmerman, D.L. (2026). Integer-valued antedependence models for longitudinal count data. *Biostatistics*.

**See Also**

[fit\\_inad](#), [bic\\_order\\_inad](#), [test\\_stationarity\\_inad](#)

**Examples**

```
set.seed(1)
y <- simulate_inad(
  n_subjects = 40,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2
)
out <- test_order_inad(
  y,
  order_null = 0,
  order_alt = 1,
  thinning = "binom",
  innovation = "pois",
  max_iter = 20
)
out$statistic
```

---

test\_stationarity\_cat *Likelihood ratio test for stationarity (categorical AD data)*

---

**Description**

Tests whether a categorical antedependence process satisfies stationarity constraints in the AD parameterization.

**Usage**

```
test_stationarity_cat(
  y,
  order = 1,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  test = c("lrt", "score", "mlrt")
)
```

**Arguments**

**y** Integer matrix with `n_subjects` rows and `n_time` columns. Each entry should be a category code from 1 to `c`.

order	Antedependence order p. Default is 1.
blocks	Optional integer vector of length n_subjects specifying group membership.
homogeneous	Logical. If TRUE (default), parameters are shared across all groups.
n_categories	Number of categories. If NULL, inferred from data.
test	Type of test statistic. One of "lrt" (default), "score", or "mlrt".

### Details

The tested constraints are:

1. The marginal distribution  $P(Y_k)$  is constant for all k
2. The transition probabilities  $P(Y_k | Y(k-p), \dots, Y(k-1))$  are constant for all  $k > p$

For AD order 1, these two constraints correspond to strict stationarity. For AD order greater than 1, this function should be interpreted as testing marginal-constancy plus time-invariant transitions; these constraints are not, in general, sufficient for full strict stationarity.

This is stronger than time-invariance alone, which only requires condition 2.

This function currently supports complete data only.

The null hypothesis is tested against the general (non-stationary) AD(p) model. The degrees of freedom are computed from the fitted parameter counts:

$$df = n_{params}(H_1) - n_{params}(H_0)$$

where  $H_1$  is the unconstrained non-stationary model and  $H_0$  is the stationary model.

### Value

A list of class "cat\_lrt" containing:

**method** Inference method used: one of "lrt", "score", or "mlrt".

**lrt\_stat** Likelihood ratio test statistic

**df** Degrees of freedom

**p\_value** P-value from chi-square distribution

**fit\_null** Fitted stationary model ( $H_0$ )

**fit\_alt** Fitted non-stationary model ( $H_1$ )

**table** Summary data frame

### References

Xie, Y. and Zimmerman, D. L. (2013). Antedependence models for nonstationary categorical longitudinal data with ignorable missingness: likelihood-based inference. *Statistics in Medicine*, 32, 3274-3289.

### See Also

[test\\_timeinvariance\\_cat](#), [test\\_order\\_cat](#)

**Examples**

```
# Simulate stationary AD(1) data
set.seed(123)
y <- simulate_cat(200, 6, order = 1, n_categories = 2)

# Test stationarity
test <- test_stationarity_cat(y, order = 1)
print(test)
```

---

test\_stationarity\_gau *Likelihood ratio test for stationarity (Gaussian AD data)*

---

**Description**

Tests whether time-varying Gaussian AD covariance parameters can be constrained to be constant over time.

**Usage**

```
test_stationarity_gau(
  y,
  order = 1L,
  blocks = NULL,
  constrain = "both",
  fit_unconstrained = NULL,
  verbose = FALSE,
  max_iter = 2000L,
  rel_tol = 1e-08,
  ...
)
```

**Arguments**

y	Numeric matrix with n_subjects rows and n_time columns.
order	Antedependence order (0, 1, or 2).
blocks	Optional vector of block memberships (length n_subjects).
constrain	Constraint to test: for order 0: "sigma" (or "all"); for order 1: "phi", "sigma", or "both"/"all"; for order 2: "phi1", "phi2", "phi", "sigma", or "all"/"both".
fit_unconstrained	Optional pre-computed unconstrained fit from <a href="#">fit_gau</a> .
verbose	Logical; if TRUE, prints fitting progress.
max_iter	Maximum number of optimization iterations for constrained fit.
rel_tol	Relative tolerance for constrained optimization.
...	Additional arguments passed to <a href="#">fit_gau</a> when fit_unconstrained is not provided.

## Details

The mean structure is kept unrestricted in both models (time-specific means plus optional block shifts), and the test constrains covariance parameters: innovation standard deviations and/or antedependence coefficients.

The likelihood-ratio statistic is:

$$\lambda = 2(\ell_{alt} - \ell_{null})$$

where  $\ell_{null}$  and  $\ell_{alt}$  are maximized log-likelihoods under the constrained and unconstrained models, respectively.

Degrees of freedom are computed from the number of constraints implied by `constrain`.

## Value

A list with class "test\_stationarity\_gau" containing:

**method** Inference method used ("lrt").  
**fit\_unconstrained** Unconstrained Gaussian AD fit  
**fit\_constrained** Constrained Gaussian AD fit  
**constraint** Human-readable null constraint description  
**lrt\_stat** Likelihood-ratio statistic  
**df** Degrees of freedom  
**p\_value** Chi-square p-value  
**bic\_unconstrained** BIC of unconstrained model  
**bic\_constrained** BIC of constrained model  
**bic\_selected** Model selected by BIC  
**table** Two-row model summary table

## References

Zimmerman, D.L. and Nunez-Anton, V. (2009). Antedependence Models for Longitudinal Data. Chapman & Hall/CRC. Chapter 6.

## See Also

[run\\_stationarity\\_tests\\_gau](#), [test\\_order\\_gau](#), [fit\\_gau](#)

## Examples

```
set.seed(1)
y <- simulate_gau(n_subjects = 80, n_time = 6, order = 1, phi = 0.4, sigma = 1)

# Test jointly constant phi and sigma (order 1)
out <- test_stationarity_gau(y, order = 1, constrain = "both")
out$p_value
```

---

 test\_stationarity\_inad

*Likelihood ratio test for stationarity (INAD data)*


---

### Description

Tests whether time-varying INAD parameters can be constrained to be constant over time.

### Usage

```
test_stationarity_inad(
  y,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  blocks = NULL,
  constrain = "both",
  fit_unconstrained = NULL,
  verbose = FALSE,
  ...
)
```

### Arguments

y	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
order	Model order (1 or 2).
thinning	Thinning operator: "binom", "pois", or "nbinom".
innovation	Innovation distribution: "pois", "bell", or "nbinom".
blocks	Optional integer vector for block effects.
constrain	Which parameters to constrain: "alpha", "theta", "both" for order 1; "alpha1", "alpha2", "alpha", "theta", "all" for order 2.
fit_unconstrained	Optional pre-computed unconstrained fit.
verbose	Logical; if TRUE, print progress.
...	Additional arguments.

### Details

For order 1, the test can constrain alpha, theta, or both. For order 2, it can constrain alpha1, alpha2, alpha (both), theta, or all supported time-varying parameters.

Degrees of freedom are computed from the number of equality constraints imposed under the null model relative to the unconstrained model.

Missing-data inputs are supported through the same `na_action` options available in `fit_inad`. If `y` has missing values and `na_action` is not supplied via `...`, this function defaults to `na_action = "marginalize"`.

**Value**

A list with class "test\_stationarity\_inad" containing:

**method** Inference method used ("lrt").  
**fit\_unconstrained** Unconstrained INAD fit  
**fit\_constrained** Constrained INAD fit  
**constraint** Human-readable null constraint description  
**statistic** Test statistic value  
**lrt\_stat** Likelihood ratio test statistic  
**df** Degrees of freedom  
**p\_value** Chi-square p-value  
**bic\_unconstrained** BIC of unconstrained model  
**bic\_constrained** BIC of constrained model  
**bic\_selected** Model selected by BIC  
**table** Two-row model summary table

**References**

Li, C. and Zimmerman, D.L. (2026). Integer-valued antedependence models for longitudinal count data. *Biostatistics*.

**See Also**

[run\\_stationarity\\_tests\\_inad](#), [test\\_order\\_inad](#), [fit\\_inad](#)

**Examples**

```
set.seed(1)
y <- simulate_inad(
  n_subjects = 30,
  n_time = 5,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  alpha = 0.3,
  theta = 2
)
out <- test_stationarity_inad(
  y,
  order = 1,
  thinning = "binom",
  innovation = "pois",
  constrain = "both",
  max_iter = 20
)
out$p_value
```

---

```
test_timeinvariance_cat
```

*Likelihood ratio test for time-invariance (categorical data)*

---

## Description

Tests whether transition probabilities are constant over time in a categorical antedependence model.

## Usage

```
test_timeinvariance_cat(
  y,
  order = 1,
  blocks = NULL,
  homogeneous = TRUE,
  n_categories = NULL,
  test = c("lrt", "score", "mlrt")
)
```

## Arguments

<code>y</code>	Integer matrix with <code>n_subjects</code> rows and <code>n_time</code> columns. Each entry should be a category code from 1 to <code>c</code> .
<code>order</code>	Antedependence order <code>p</code> . Default is 1.
<code>blocks</code>	Optional integer vector of length <code>n_subjects</code> specifying group membership.
<code>homogeneous</code>	Logical. If TRUE (default), parameters are shared across all groups.
<code>n_categories</code>	Number of categories. If NULL, inferred from data.
<code>test</code>	Type of test statistic. One of "lrt" (default), "score", or "mlrt".

## Details

The null hypothesis is that all transition probabilities (for  $k > p$ ) are equal across time:

$$H_0 : \pi_{y_k | y_{k-p}, \dots, y_{k-1}} \text{ is constant for } k = p + 1, \dots, n$$

This reduces (n-p) separate transition matrices/arrays to a single one.

The degrees of freedom are:

$$df = (c - 1) \times c^p \times (n - p - 1)$$

This function currently supports complete data only. If `y` contains missing values, use model-fitting functions (for example `fit_cat`) directly with missing-data handling instead of this test wrapper.

**Value**

A list of class "cat\_lrt" containing:

**method** Inference method used: one of "lrt", "score", "mlrt", or "wald".

**lrt\_stat** Likelihood ratio test statistic

**df** Degrees of freedom

**p\_value** P-value from chi-square distribution

**fit\_null** Fitted time-invariant model (H0)

**fit\_alt** Fitted time-varying model (H1)

**table** Summary data frame

**References**

Xie, Y. and Zimmerman, D. L. (2013). Antedependence models for nonstationary categorical longitudinal data with ignorable missingness: likelihood-based inference. *Statistics in Medicine*, 32, 3274-3289.

**See Also**

[fit\\_cat](#), [test\\_order\\_cat](#)

**Examples**

```
# Simulate data with time-invariant transitions
set.seed(123)
y <- simulate_cat(200, 6, order = 1, n_categories = 2)

# Test time-invariance
test <- test_timeinvariance_cat(y, order = 1)
print(test)
```

---

test\_two\_sample\_gau     *Two-sample test for equality of mean profiles under antedependence*

---

**Description**

Tests the null hypothesis that two groups have equal mean profiles  $\mu_1 = \mu_2$  against the alternative  $\mu_1 \neq \mu_2$ , assuming a common AD(p) covariance structure. This implements Theorem 7.3 of Zimmerman & Núñez-Antón (2009).

**Usage**

```
test_two_sample_gau(y, blocks, p = 1L, order = NULL, use_modified = TRUE)
```

**Arguments**

<code>y</code>	Numeric matrix with <code>n_subjects</code> rows and <code>n_time</code> columns.
<code>blocks</code>	Integer vector of length <code>n_subjects</code> indicating group membership (must contain exactly two unique values, typically 1 and 2).
<code>p</code>	Antedependence order of the common covariance structure. This is the same order parameter named <code>order</code> in <code>fit_gau</code> .
<code>order</code>	Optional alias for <code>p</code> . Supply only one of <code>p</code> or <code>order</code> .
<code>use_modified</code>	Logical. If TRUE (default), use modified test statistic.

**Details**

This test is also known as a "profile comparison" test. The likelihood ratio test statistic (Theorem 7.3) compares the pooled RSS (under H0: common mean) to the sum of within-group RSS (under H1: separate means):

$$N \sum_{i=1}^n [\log RSS_i(\mu) - \log RSS_i(\mu_1, \mu_2)]$$

where `RSS_i(mu)` uses a common mean and `RSS_i(mu_1, mu_2)` uses group-specific means.

**Value**

A list with class `gau_mean_test` containing:

<b>method</b>	Inference method used ("lrt").
<b>test_type</b>	"two-sample"
<b>mu1_hat</b>	Estimated mean for group 1
<b>mu2_hat</b>	Estimated mean for group 2
<b>mu_pooled</b>	Pooled mean estimate under H0
<b>statistic</b>	Test statistic value
<b>statistic_modified</b>	Modified test statistic
<b>df</b>	Degrees of freedom ( <code>n_time</code> )
<b>p_value</b>	P-value from chi-square distribution
<b>p_value_modified</b>	P-value from modified test
<b>order</b>	Antedependence order used

**References**

Zimmerman, D.L. and Núñez-Antón, V. (2009). Antedependence Models for Longitudinal Data. Chapman & Hall/CRC. Chapter 7.

**Examples**

```
# Simulate data from two groups with different means
n1 <- 30
n2 <- 35
y1 <- simulate_gau(n1, n_time = 6, order = 1, mu = rep(10, 6))
y2 <- simulate_gau(n2, n_time = 6, order = 1, mu = rep(12, 6))
y <- rbind(y1, y2)
blocks <- c(rep(1, n1), rep(2, n2))

# Test equality of profiles
test <- test_two_sample_gau(y, blocks, p = 1)
print(test)
```

# Index

## \* datasets

- bolus\_inad, 13
  - cattle\_growth, 13
  - cochlear\_implant, 18
  - labor\_force\_cat, 29
  - race\_100km, 47
- aic\_cat, 3  
aic\_gau, 4  
aic\_inad, 5
- Bell, 6  
bic\_cat, 7, 8  
bic\_gau, 8  
bic\_inad, 8, 9  
bic\_order\_cat, 10, 11, 70  
bic\_order\_gau, 11  
bic\_order\_inad, 11, 12, 74  
bolus\_inad, 13
- cattle\_growth, 13  
ci\_cat, 14, 16  
ci\_gau, 16  
ci\_inad, 16, 17  
cochlear\_implant, 18
- dbell (Bell), 6
- em\_cat, 19, 21, 22, 24  
em\_gau, 20, 22, 25, 26  
em\_inad, 20, 21, 21
- fit\_cat, 4, 7, 15, 20–22, 23, 26, 62, 70, 81  
fit\_gau, 4, 8, 11, 16, 20, 21, 25, 51, 59, 63, 67, 71, 76, 77, 82  
fit\_inad, 5, 9, 17, 22, 26, 27, 66, 73, 74, 78, 79
- labor\_force\_cat, 29  
logL\_cat, 20, 29  
logL\_gau, 26, 31
- logL\_inad, 32  
logL\_inad\_i, 34
- partial\_corr, 35, 38  
pbell (Bell), 6  
plot\_prism, 36, 37  
plot\_profile, 38  
print.cat\_ci, 40  
print.cat\_fit, 40  
print.cat\_lrt, 41  
print.gau\_bic\_order, 41  
print.gau\_ci, 42  
print.gau\_contrast\_test, 42  
print.gau\_fit, 43  
print.gau\_homogeneity\_test, 43  
print.gau\_mean\_test, 44  
print.gau\_order\_test, 44  
print.homogeneity\_tests\_inad, 45  
print.inad\_ci, 45  
print.inad\_fit, 46  
print.test\_homogeneity\_inad, 46
- qbell (Bell), 6
- race\_100km, 47  
rbell (Bell), 6  
run\_homogeneity\_tests\_inad, 47  
run\_order\_tests\_cat, 48  
run\_stationarity\_tests\_cat, 49  
run\_stationarity\_tests\_gau, 50, 77  
run\_stationarity\_tests\_inad, 52, 79
- simulate\_cat, 53  
simulate\_gau, 55  
simulate\_inad, 56  
summary.cat\_ci, 58  
summary.gau\_ci, 58  
summary.inad\_ci, 59
- test\_contrast\_gau, 59  
test\_homogeneity\_cat, 61

test\_homogeneity\_gau, [63](#), [72](#)  
test\_homogeneity\_inad, [64](#)  
test\_one\_sample\_gau, [66](#), [72](#)  
test\_order\_cat, [49](#), [62](#), [68](#), [75](#), [81](#)  
test\_order\_gau, [64](#), [68](#), [70](#), [77](#)  
test\_order\_inad, [66](#), [72](#), [79](#)  
test\_stationarity\_cat, [74](#)  
test\_stationarity\_gau, [51](#), [76](#)  
test\_stationarity\_inad, [66](#), [74](#), [78](#)  
test\_timeinvariance\_cat, [75](#), [80](#)  
test\_two\_sample\_gau, [64](#), [68](#), [81](#)