Package ‘AMDconfigurations’

October 21, 2026

Type Package

Title Geometric Analysis of Configurations in High-Dimensional Spaces

Version 0.4.0

Description Computes Average Membership Degree (AMD) from soft-membership matrices and constructs AMD curves across candidate configuration counts. The current curve backend uses repeated fuzzy c-means fits while keeping the AMD definition, membership estimation, curve construction, and final assignment as separate operations. Fitted memberships, convergence diagnostics, preprocessing transformations, seeds, row maps, and process-level reproducibility links are retained. The package reports numerical curve features but leaves their scientific interpretation to an explicitly justified analysis. The method was introduced by Mendoza and Araújo (2022) <doi:10.1111/ecog.06289>.

License MIT + file LICENSE

URL https://github.com/mmendoza1967/AMDconfigurations, https://mmendoza1967.github.io/AMDconfigurations/

BugReports https://github.com/mmendoza1967/AMDconfigurations/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports e1071 (>= 1.7-17), grDevices, graphics, stats, utils

Suggests testthat (>= 3.2.0), knitr, rmarkdown, ranger (>= 0.18.0), digest (>= 0.6.27)

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website pkgdown

NeedsCompilation no

Author Manuel Mendoza [aut, cre] (ORCID: <https://orcid.org/0000-0002-2143-8138>)

Maintainer Manuel Mendoza <mmendoza@mncn.csic.es>

Repository CRAN

Date/Publication 2026-10-21 03:50:02 UTC

Contents

amd_reproducibility_manifest 2
amd_score 3
assign_configurations 4
compute_amd_curve 6
diagnose_amd 8
Fulldata 10
load_amd_predictor 10
plot.amd_curve 11
plot.amd_prediction 11
plot_AMD 12
predict_amd 13
print.amd_curve 14
print.amd_diagnostics 15
print.amd_prediction 15
summary.amd_curve 16
summary.amd_prediction 16
Transcdata_small 17

Index **18**

`amd_reproducibility_manifest`
Record the reproducibility environment for an AMD process

Description

Creates a compact record of the software and random-number environment used by AMD. The manifest is created once and then reused within the current R process while its random-number generator remains unchanged. A different process, including each parallel worker, or a change to `RNGkind()` receives a different manifest identifier.

Usage

```
amd_reproducibility_manifest()
```

Details

`compute_amd_curve()` records only the manifest identifier, rather than duplicating the complete session information in every curve. Retrieve the manifest immediately after each curve and save one copy of every distinct `manifest_id`. Curves sharing an identifier can share the saved copy.

Value

An object of class "amd_reproducibility_manifest" containing a schema version, a unique environment identifier, its generation and UTC creation time, the process identifier, R and package versions, RNGkind(), and sessionInfo().

Examples

```
manifest <- amd_reproducibility_manifest()
manifest$manifest_id
```

---

amd_score — *Compute Average Membership Degree from soft memberships*

Description

amd_score() defines Average Membership Degree (AMD) independently of the algorithm used to estimate the memberships. Rows represent observations, columns represent candidate configurations, entries are non-negative, and each row sums to one.

Usage

```
amd_score(membership, tolerance = sqrt(.Machine$double.eps))
```

Arguments

membership	Numeric matrix or data frame with observations in rows and configurations in columns.
tolerance	Non-negative numerical tolerance used to check membership bounds and row sums.

Details

For a membership matrix with c columns, AMD is the mean of the largest membership in each row minus 1 / c. The term 1 / c is the value attained by a completely uniform membership vector. It is not a distribution-free expectation under random assignment.

A one-column membership matrix is mathematically valid and has AMD zero. Curves used to compare alternative numbers of configurations nevertheless begin at c = 2; see compute_amd_curve().

Value

One un-named numeric AMD value in the interval [0, 1 - 1 / c], where c is the number of membership columns.

References

Mendoza, M. and Araújo, M. B. (2022). Biogeography of bird and mammal trophic structures. *Ecography*, 2022(5), e06289. doi:10.1111/ecog.06289

Examples

```
memberships <- rbind(
  c(0.90, 0.10),
  c(0.25, 0.75),
  c(0.50, 0.50)
)
amd_score(memberships)
```

`assign_configurations` *Assign observations to a user-selected number of configurations*

Description

The official workflow supplies an `amd_curve` object. The function then extracts the exact fit that produced the AMD value at the explicitly requested `c`; no new clustering is performed.

Usage

```
assign_configurations(
  data,
  c,
  its = 50,
  iter_max = 100,
  m = 2,
  scale_data = FALSE,
  verbose = TRUE,
  seeds = NULL
)
```

Arguments

`data`	Preferably an object returned by `compute_amd_curve()`. A numeric matrix or data frame with observations in rows and variables in columns is accepted only through the provisional compatibility mode.
`c`	One integer greater than or equal to 2. The number of configurations to extract or fit. It must have been evaluated when `data` is an AMD curve.
`its`	Positive integer number of random starts in independent fitting mode (default: 50). Ignored when extracting from an AMD curve.
`iter_max`	Positive integer maximum number of fuzzy c-means iterations in independent fitting mode (default: 100). Ignored when extracting from an AMD curve.
`m`	Fuzziness parameter greater than one in independent fitting mode (default: 2). Ignored when extracting from an AMD curve.
`scale_data`	Logical; standardize variables before independent fitting. Ignored when extracting from an AMD curve.
`verbose`	Logical; print progress information in independent fitting mode. Ignored when extracting from an AMD curve.

`seeds` Optional vector of positive integer seeds, one per random start in independent fitting mode. Supplied seeds make the result reproducible and do not alter the caller's random-number state. Ignored when extracting from an AMD curve. If `NULL`, seeds are sampled from the caller's active random-number generator and advance its state. Unused trailing seeds are ignored. Duplicate seeds reduce the number of unique starts and generate a warning.

Details

A provisional compatibility mode accepts a numeric matrix or data frame and independently estimates memberships with repeated fuzzy c-means starts. It is not the package's recommended scientific workflow and may be deprecated or removed in a future version.

The function never chooses `c` automatically. In particular, passing an AMD curve still requires an explicit `c`, so the user can incorporate the shape of the curve rather than being forced to accept its numerical maximum.

Future automation may assist with interpreting the curve and proposing `c`, but it must be validated against known truth and must not silently replace inspection of the complete curve.

AMD is defined independently of fuzzy c-means. Fuzzy c-means is only the membership-estimation backend used by the current implementation of `compute_amd_curve()` and by the provisional compatibility mode.

Value

A list of class `"amd_assignment"`. `schema_version` identifies the public object layout. The object contains the selected `c` (also in the compatibility alias `c_opt`), hard assignments, memberships, centers, AMD and objective values, selection metadata, and a `starts` data frame with the diagnostics for every random start. `source` distinguishes exact extraction from an AMD curve from an independent fit. Components `backend` and `selection_rule` identify how memberships were estimated and how the retained fit was selected. `complete_rows` and `omitted_rows` map fitted observations back to the input rows when incomplete rows were removed. `settings` records the fitting and preprocessing contract, while `manifest_id` links the object to `amd_reproducibility_manifest()`. In curve-extraction mode, `settings` retains the complete candidate range used to create the curve; in independent fitting mode it records the single fitted `c`. If converged starts have exactly tied minimum objectives, the first start is selected. The selected fit is always converged under the stated iteration rule.

Examples

```
set.seed(1)
X <- rbind(
  matrix(rnorm(100, mean = -2), ncol = 2),
  matrix(rnorm(100, mean = 2), ncol = 2)
)
curve <- compute_amd_curve(
  X, its = 5, nin = 2, nsp = 4, seeds = 1:15, verbose = FALSE
)

# Exact fit underlying the selected point of the inspected curve
extracted <- assign_configurations(curve, c = 2)
```

compute_amd_curve *Compute an Average Membership Degree (AMD) curve*

Description

For every candidate number of configurations, this function runs fuzzy c-means from several random initializations. Random starts are computational attempts to estimate the same solution, not data replicates. Consequently, the fit with the smallest fuzzy c-means objective among the converged starts supplies the membership matrix evaluated by `amd_score()` at each `c`.

Usage

```
compute_amd_curve(
  data,
  its,
  nin,
  nsp,
  seeds = NULL,
  verbose = TRUE,
  plot_curve = FALSE,
  open_device = TRUE,
  scale_data = FALSE,
  iter_max = 100,
  m = 2,
  preselect_top_sd = NULL
)
```

Arguments

`data`	Numeric matrix or data frame. Rows are observations and columns are variables. Rows with missing values are removed with a warning and recorded in the returned row map.
`its`	Positive integer number of random starts for every candidate `c`.
`nin`	Minimum candidate number of configurations. Must be at least 2.
`nsp`	Maximum candidate number of configurations.
`seeds`	Optional positive integer vector or matrix. Vectors retain the historical start-major order; unused trailing values are ignored. If `NULL`, seeds are sampled from the caller's active random-number generator and advance its state. A matrix must have one row per start and uniquely named columns for every evaluated `c`; this form preserves the seeds of common candidates when the upper search limit is extended.
`verbose`	Logical; print progress information when `TRUE`.

`plot_curve` Logical; plot the primary AMD curve when `TRUE`.

`open_device` Logical; when `plot_curve = TRUE`, open a new graphics device before plotting. The default is `TRUE` for compatibility with the public 0.1.0 function. It has no effect when `plot_curve = FALSE`.

`scale_data` Logical; standardize variables before fitting.

`iter_max` Positive integer maximum number of fuzzy c-means iterations.

`m` Fuzziness parameter; must be greater than one.

`preselect_top_sd` Optional positive integer. If supplied, retain only the variables with the largest standard deviations before optional scaling. Retained variables are ordered by decreasing standard deviation; their original indices and names are recorded in `settings`.

Details

AMD itself is defined independently of fuzzy c-means. Fuzzy c-means is the membership-estimation backend used by this function.

The returned `c_peak` (and its compatibility alias `c_opt`) is only the numerical maximum of the evaluated curve. In particular, a maximum at the boundary of the examined range is flagged and must not by itself be treated as evidence for that number of groups. If several evaluated values are exactly tied for the maximum, the smallest `c` is returned.

Vector seeds, including seeds generated when `seeds = NULL`, preserve the historical start-major mapping and therefore depend on the number of evaluated candidates. Use a named seed matrix when comparing nested search ranges. Extra named columns are allowed and ignored when their `c` is not in the current range. Duplicate seeds within one candidate `c` reduce the number of unique random starts and generate a warning.

Every curve contains a `manifest_id` linking it to the process-level record returned by `amd_reproducibility_manifest()`. Retrieve the manifest immediately after each curve and save one copy of every distinct identifier. Curves sharing an identifier can share the saved copy.

Value

An object of class `"amd_curve"`. `schema_version` identifies the public object layout. The primary `curve` contains AMD values from the minimum-objective converged fit at each `c`. Components `mean`, `max`, and `raw` retain the historical restart summaries for diagnostics and compatibility. These historical summaries include all starts, including nonconverged starts, and are not the primary curve. `selected_fits` contains the exact fitted memberships that produced the primary curve. `complete_rows` and `omitted_rows` map the fitted observations back to the input rows. The `settings` component records the input and retained variables and the exact centering and scaling vectors used to construct the fitted data. `manifest_id` links the curve to its process-level reproducibility manifest. Additional public components are `c_peak`, its historical alias `c_opt`, `peak_at_boundary`, the per-start `objectives`, `iterations` and `converged` matrices, selected-start diagnostics, the seed matrix, `coordinates` (the selected membership matrix at `c_peak`), `backend`, and `selection_rule`. `selected_fit` is the element of `selected_fits` corresponding to `c_peak`.

References

Mendoza, M. and Araújo, M. B. (2022). Biogeography of bird and mammal trophic structures. *Ecography*, 2022(5), e06289. doi:10.1111/ecog.06289

Examples

```
set.seed(42)
X <- rbind(
  matrix(rnorm(40, mean = -2), ncol = 2),
  matrix(rnorm(40, mean = 2), ncol = 2)
)
curve <- compute_amd_curve(
  X, its = 2, nin = 2, nsp = 3, seeds = 1:4, verbose = FALSE
)
curve$curve
```

`diagnose_amd` *Describe an AMD curve and its stored partitions*

Description

Summarise the primary curve, memberships and hard assignments without fitting new models, selecting a number of groups or changing the input. The selected fits must be those saved by `compute_amd_curve()`.

Usage

```
diagnose_amd(object, ambiguity_gap = 0.1)
```

Arguments

`object`	An `amd_curve` with a primary curve and a selected fit for every evaluated candidate. Legacy 0.1.0 objects are not sufficient.
`ambiguity_gap`	A number in `[0, 1]`. An observation is counted as ambiguous when its largest membership minus its second-largest membership is strictly less than this value. The default, 0.1, is a descriptive convention, not a validated universal decision threshold.

Details

Normalised observation entropy is `-sum(u * log(u)) / log(c)`, with `0 * log(0)` interpreted as zero. Size entropy applies the same formula to group fractions. Quantiles use `stats::quantile()` with `type = 8`. Exact top-membership ties are reported separately and the stored assignment is retained; no new tie-breaking or random-number generation takes place.

Transitions compare the same observations, not the numerical group labels. ARI and NMI are invariant to relabelling. Identical single-group partitions have ARI and NMI one even if their occupied membership columns differ. NMI is zero when only one of the partitions has a single occupied group.

This function uses memberships and assignments only. It does not calculate sample-to-sample distances, silhouette, geometric compactness, density, connectivity or resampling stability, and does not require the original data. It does not impute missing descriptive values or call a supervised reader. Its working arrays grow linearly with the number of observations per candidate, plus contingency tables between occupied groups; it does not construct an observation-by-observation distance matrix.

Value

An object of class `amd_diagnostics` with:

candidates One row per evaluated `c`, describing curve neighbours, winning memberships, normalised membership entropy, top-two gaps and hard and soft group-size balance. Boundary local maxima are `NA`, since one neighbour is missing. A local maximum here means strictly greater than both evaluated neighbours; plateaus are not classified as peaks.

groups One row per membership column at each `c`. Hard sizes use the stored winning assignment. Soft masses are sums of memberships, not of memberships raised to the fuzziness parameter. Summaries within a hard group are `NA` when that group is empty.

transitions ARI, geometric-mean normalised mutual information (NMI) and directional fragmentation between successive evaluated candidates. Fragmentation is the fraction outside each group's most frequent group in the other partition. `from_fragmentation` describes splitting of source groups; `to_fragmentation` describes mixing of source groups in destination groups. These are descriptive overlaps, not evidence of physical splitting, merging or biological stability.

curve_summary Numerical maximum, exact ties, boundary status and the evaluated range. These do not constitute a scientific choice of K.

dictionary Definitions and scales for every table column.

unavailable Diagnostics not computed by this operation and why.

complete_rows, omitted_rows The original observation row map.

settings Diagnostic conventions and provenance.

See Also

`compute_amd_curve()`, `assign_configurations()`, `amd_score()`

Examples

```
set.seed(42)
X <- rbind(matrix(rnorm(40, -2), ncol = 2),
           matrix(rnorm(40, 2), ncol = 2))
curve <- compute_amd_curve(X, its = 2, nin = 2, nsp = 3,
                           seeds = 1:4, iter_max = 1000, verbose = FALSE)
diagnostics <- diagnose_amd(curve)
diagnostics$candidates
diagnostics$transitions
# Inspect the curve and diagnostics before making an explicit choice.
assigned <- assign_configurations(curve, c = 2)
```

Fulldata — *Ecological dataset used in the example*

Description

The dataset `Fulldata` contains the ecological data used in the example script `inst/examples/example_ecology.R`. It is included in the package to provide a complete reproducible workflow for the ecological case study.

Usage

```
data("Fulldata")
```

Format

A data frame with 6,619 rows and 34 columns. Columns `lat`, `lon`, `Key`, `island`, `Clst`, and `NAME` identify and describe sampling units; `IFd`, `GIn`, `HIn`, `IFr`, `Crn`, `Pln`, `Frg`, `Nct`, and `Grn` are the nine trophic-structure variables used by the package example; and `BIO1` to `BIO19` are bioclimatic variables. See the cited article and its supporting information for the complete variable definitions and data provenance.

Source

Data prepared for Mendoza and Araújo (2022), doi:10.1111/ecog.06289.

load_amd_predictor — *Load the optional drop-37 AMD reader*

Description

Loads the external model bundle used by `predict_amd()`. The large fitted random forests are deliberately not embedded in the base package. Loading verifies a caller-supplied SHA-256, validates the model and preprocessing contracts, and never downloads or refits a model. Historical v1/v2 bundles remain reproducible with their matching historical package versions.

Usage

```
load_amd_predictor(path, sha256)
```

Arguments

`path`	Path to an `amd_predictor_bundle_v3_drop37` RDS file.
`sha256`	Expected 64-character SHA-256 from the matching bundle manifest. The optional package `digest` is required for this integrity check.

Value

An object of class "amd_predictor".

---

plot.amd_curve *Plot method for AMD curve objects*

Description

Plot method for AMD curve objects

Usage

```
## S3 method for class 'amd_curve'
plot(x, type = c("curve", "mean", "max"), open_device = FALSE, ...)
```

Arguments

x	An object of class "amd_curve".
type	Curve to plot; see plot_AMD().
open_device	Logical; open a new graphics device before plotting.
...	Additional arguments passed to plot_AMD().

Value

Invisibly returns NULL.

---

plot.amd_prediction *Plot an optional AMD reader result*

Description

The fitted AMD curve remains the primary display. The numerical AMD maximum and model-assisted proposal are shown separately. The analytical c = 1, AMD = 0 anchor is optional because including it changes the vertical scale and can make the shape among c = 2,...,14 less visible.

Usage

```
## S3 method for class 'amd_prediction'
plot(x, show_c1 = FALSE, ...)
```

Arguments

x	An amd_prediction object.
show_c1	Logical; show the unconnected analytical c = 1 anchor.
...	Additional arguments passed to graphics::plot().

Value

Invisibly returns `x`.

`plot_AMD` *Plot an AMD curve*

Description

Plots the primary AMD curve returned by `compute_amd_curve()`. Historical restart summaries remain available as diagnostic alternatives.

Usage

```
plot_AMD(
  amd_object,
  type = c("curve", "mean", "max"),
  open_device = FALSE,
  ...
)
```

Arguments

`amd_object`	An object returned by `compute_amd_curve()`.
`type`	Curve to plot: `"curve"` (default) uses the minimum-objective converged fit at every `c`; `"mean"` and `"max"` show historical summaries across all random starts, including nonconverged starts. They are retained only for diagnostics and compatibility.
`open_device`	Logical; open a new graphics device before plotting. The default uses the active device, as is conventional for S3 plot methods.
`...`	Additional graphical parameters passed to `graphics::plot()`.

Details

Objects saved by public version 0.1.0 can still be plotted from their stored `mean` or `max` restart summaries. Because those objects do not contain the minimum-objective primary curve, the method warns and does not attempt to reconstruct it.

Value

Invisibly returns `NULL`.

`predict_amd` *Propose a modal resolution from an AMD analysis*

Description

Optional assistance for interpreting a complete AMD curve. It computes the frozen `c = 2,...,14` AMD analysis (unless a compatible curve is supplied), derives curve, data, and candidate-partition descriptors, and applies five externally loaded random forests. Conditional reference values estimate correctness of the one-versus-several-modes decision and exact K under the reference simulation regime. Reliability for a new distribution remains unestablished; these values are not the probability that groups exist. The target is the numerically resolved number of density modes used as the simulation reference, not necessarily the number of generating components. AMD itself does not estimate a density.

Usage

```
predict_amd(
  data,
  predictor,
  curve = NULL,
  seed = 1L,
  allow_extrapolation = FALSE,
  verbose = TRUE,
  max_working_bytes = 512 * 1024^2
)
```

Arguments

`data` Numeric matrix or data frame, with observations in rows.

`predictor` An object returned by `load_amd_predictor()`.

`curve` Optional compatible `amd_curve`. Supplying it avoids repeating the principal AMD fits; it is checked against `data` and the frozen model contract.

`seed` Positive integer controlling AMD starts, stability refits, and any large-data sampling used by descriptors. The caller's random-number state is preserved.

`allow_extrapolation` Logical. Away from the validated numerical grid, retain conditional simulation-reference values in `reference_reliability`, labelled as unvalidated interpolation or extrapolation. This never establishes reliability for the analysed distribution and does not override feature or output-support guards.

`verbose` Logical; report progress while computing a new curve.

`max_working_bytes` Positive finite working-memory guard for the exact descriptors (default 512 MiB). The estimate is not a guarantee of total process memory and excludes the externally loaded forests. The frozen global-distance summaries use a seeded sample of 2000 rows when N > 2000; partition distances remain exact. This distinction is recorded in the resource diagnostics. No additional approximation bypasses the guard.

Details

This function does not replace inspection of the AMD curve, does not claim that its proposal is ground truth, and does not assign observations. When K > 1 is proposed, use `assign_configurations()` with the returned curve and a scientifically justified c.

The drop-37 forests omit 37 dimensional descriptors and use the corrected missing-left-pair convention. Their own correctness calibrators were fitted only on selection populations. A subsequent fixed-reader prospective bank gave 68.51 percent exact-count accuracy versus 70.91 percent mean reference confidence. A controlled distribution challenge gave 60.53 versus 66.71 percent. Thus the calibrated values cannot certify accuracy for a new distribution merely because N and D match the simulation grid.

Continuous exact-K and one/several values remain available separately in `reference_reliability`; no certainty threshold chooses or rejects K. By default these reference values are suppressed outside N = 150 or 600, D = 2, 6 or 15, assessed output support K=1:8, or assessed feature conditions. The mechanical candidate range remains 1:14. Departures from omitted training-constant predictors also suppress reference values. Neither forest unanimity nor a large reference value establishes distributional applicability. Count correctness does not validate FCM assignments or the scientific meaning of empirical configurations. See the installed `READER_EVIDENCE.txt` for evidence and limits.

Supplying a compatible curve avoids its 260 principal fits. The additional five computational restarts per c required by the reader are still run; these are not a bootstrap or other sample-level stability analysis. Curve checks establish row/dimension and objective consistency with the supplied data, not a cryptographic identity of a separately created data file.

Value

An object of class `"amd_prediction"` containing the AMD curve, proposed numerically resolved density-mode count K, separate structure and exact-modal-count conditional reference estimates. `reliability` remains `NA` because applicability to the analysed distribution has not been established. `reference_reliability` contains the own-calibrator values subject to applicability guards. `agreement` contains forest vote fractions, not probabilities of correctness. `has_structure` is retained as a compatibility alias for `proposed_multiple_modes`. The object includes candidate scores, five-seed proposals, diagnostics, and applicability.

`print.amd_curve` *Print an AMD curve object*

Description

Print an AMD curve object

Usage

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

Arguments

x	An object of class "amd_curve".
...	Unused.

Value

Invisibly returns x.

---

print.amd_diagnostics *Print descriptive AMD diagnostics*

Description

Print descriptive AMD diagnostics

Usage

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

Arguments

x	An object returned by diagnose_amd().
...	Additional arguments passed to print.data.frame().

Value

Invisibly returns x.

---

print.amd_prediction *Print an optional AMD reader result*

Description

Print an optional AMD reader result

Usage

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

Arguments

x	An object returned by predict_amd().
...	Unused.

Value

Invisibly returns `x`.

---

summary.amd_curve *Summarise an AMD curve object*

Description

Summarise an AMD curve object

Usage

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

Arguments

`object`	An object of class `"amd_curve"`.
`...`	Unused.

Value

Invisibly returns `object`.

---

summary.amd_prediction *Summarise an optional AMD reader result*

Description

Summarise an optional AMD reader result

Usage

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

Arguments

`object`	An object returned by `predict_amd()`.
`...`	Unused.

Value

A list of proposals, conditional reliability, applicability and scores.

`Transcdata_small` *Reduced transcriptomic dataset (1000 variables)*

Description

This dataset contains the 1000 variables with highest variance from the original transcriptomic dataset used in the AMDconfigurations package.

Usage

```
data("Transcdata_small")
```

Format

A numeric data frame with 931 rows (samples) and 1,000 columns (gene-expression variables), ordered by decreasing variance in the full cleaned expression matrix.

Details

The full cleaned dataset (~58,000 variables) is available at Zenodo (DOI: https://doi.org/10.5281/zenodo.18604443).

Source

Mendoza, M. and Medina, M. A. (2026), *High-dimensional transcriptomic dataset for the AMD-configurations package*, Zenodo, doi:10.5281/zenodo.18604443, CC BY 4.0.

Index

∗ **datasets**
    Fulldata, 10
    Transcdata_small, 17

amd_reproducibility_manifest, 2
amd_reproducibility_manifest(), *5*, *7*
amd_score, 3
amd_score(), *9*
assign_configurations, 4
assign_configurations(), *9*, *14*

compute_amd_curve, 6
compute_amd_curve(), *2*, *4*, *5*, *8*, *9*

diagnose_amd, 8
diagnose_amd(), *15*

Fulldata, 10

graphics::plot(), *11*, *12*

load_amd_predictor, 10
load_amd_predictor(), *13*

plot.amd_curve, 11
plot.amd_prediction, 11
plot_AMD, 12
predict_amd, 13
predict_amd(), *10*, *15*, *16*
print.amd_curve, 14
print.amd_diagnostics, 15
print.amd_prediction, 15
print.data.frame(), *15*

stats::quantile(), *8*
summary.amd_curve, 16
summary.amd_prediction, 16

Transcdata_small, 17