Package 'LinkOrgs'

December 24, 2025

Title LinkOrgs: Algorithms for Organizational Record Linkage

Version 0.01

Author Connor Jerzak [aut, cre] (<https://orcid.org/0000-0003-1914-8905>), Brian Libgober [aut] (<https://orcid.org/0000-0001-9638-0228>)

Maintainer Connor Jerzak <connor.jerzak@gmail.com>

Description An R package for organizational records using the algorithms of Jerzak & Libgober (2023+). The linkage is done based on organizational names and using half a billion open collaborated records on those names from LinkedIn users. It also contains functions implementing string matching performance metrics, as well as a fast, parallized version of fuzzy string matching.

Depends R (>= 3.5.0)

License CC BY-NC-ND 4.0

Encoding UTF-8

Imports data.table,plyr,stringdist,parallel,stringr,dplyr,fastmatch,reticulate,foreach,doParallel,digest

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

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Contents

AssessMatchPerformance

AssessMatchPerformance

Description

Computes match performance metrics (true/false positive and true/false negative rates) by comparing predicted matches against a ground-truth dataset.

Usage

```
AssessMatchPerformance(
  x,
  y,
  z,
  z_true,
  by,
  by.x = by,
  by.y = by,
  openBrowser = F
)
```

Arguments

x, y	Data frames that were merged to produce z. Used to determine the universe of possible match pairs.
z	Data frame containing the predicted matches to be evaluated. Must contain columns specified by by.x and by.y.
z_true	Data frame containing the ground-truth (reference) matches. Must contain columns specified by by.x and by.y.
by	Character string specifying the column name for merging when both data frames use the same column name.
by.x	Character string specifying the column name in x used for merging. Defaults to by if not specified.
by.y	Character string specifying the column name in y used for merging. Defaults to by if not specified.
openBrowser	Logical; if TRUE, opens browser for debugging. Default is FALSE.

Value

A named numeric vector with the following elements:

TruePositives Number of matches in z that are also in z_true.

FalsePositives Number of matches in z that are not in z_true.

FalseNegatives Number of matches in z_true that are not in z.

TrueNegatives Number of non-matches correctly identified (total pairs minus TP, FP, and FN).

MatchedDatasetSize Number of rows in the predicted matches z.

Examples

```
# Create synthetic data
x_orgnames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orgnames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orgnames_x" = x_orgnames)
y <- data.frame("orgnames_y" = y_orgnames)
z <- data.frame("orgnames_x" = x_orgnames[1:2], "orgnames_y" = y_orgnames[1:2])
z_true <- data.frame("orgnames_x" = x_orgnames, "orgnames_y" = y_orgnames)

# Obtain match performance data
PerformanceMatrix <- AssessMatchPerformance(x = x,
                                            y = y,
                                            z = z,
                                            z_true = z_true,
                                            by.x = "orgnames_x",
                                            by.y = "orgnames_y")
print(PerformanceMatrix)
```

`BuildBackend` *Build Backend for LinkOrgs Machine Learning Models*

Description

Creates and configures a conda environment with all necessary Python packages (JAX, TensorFlow, Optax, Equinox, JMP) for running the machine learning components of LinkOrgs.

Usage

```
BuildBackend(conda_env = "LinkOrgs_env", conda = "auto", tryMetal = T)
```

Arguments

`conda_env`	Character string; name of the conda environment to create. Default is `"LinkOrgs_env"`.
`conda`	Character string; path to a conda executable, or `"auto"` to let reticulate automatically find an appropriate conda binary. Default is `"auto"`.
`tryMetal`	Logical; if `TRUE` and running on Apple Silicon (arm64 macOS), attempts to install `jax-metal` for GPU acceleration. Default is `TRUE`.

Details

This function requires an Internet connection to download packages. The conda environment will include:

- TensorFlow 2.15
- TensorFlow Probability 0.23
- JAX 0.4.26 and JAXlib 0.4.26
- Optax 0.2.2
- Equinox 0.11.4
- JMP 0.0.4

- NumPy 1.26.4
- jax-metal 0.1.0 (Apple Silicon only, if `tryMetal = TRUE`)

You can find available conda Python paths via: `system("which python")`

Value

Invisibly returns `NULL`. Called for its side effect of creating and configuring the conda environment. Prints "Done building LinkOrgs backend!" upon successful completion.

See Also

`LinkOrgs()` for using the ML backend after setup.

Examples

```
## Not run:
# Build with default settings
BuildBackend()

# Build with a specific conda path
BuildBackend(conda = "/opt/miniconda3/bin/conda")

# Build without attempting Metal support on macOS
BuildBackend(tryMetal = FALSE)

## End(Not run)
```

---

dropboxURL2downloadURL

Convert Dropbox Share URL to Direct Download URL

Description

Converts a Dropbox share link to a direct download URL by replacing the domain with `dl.dropboxusercontent.com`.

Usage

```
dropboxURL2downloadURL(url)
```

Arguments

`url`	Character string; a Dropbox share URL (e.g., `"https://www.dropbox.com/s/..."` or `"https://dropbox.com/s/..."`).

Details

Dropbox share links require modification to enable direct file downloads. This function replaces:

- `https://www.dropbox.com` with `https://dl.dropboxusercontent.com`
- `www.dropbox.com` with `dl.dropboxusercontent.com`
- `dropbox.com` with `dl.dropboxusercontent.com`

Value

Character string; the converted direct download URL. If the input is not a Dropbox URL, it is returned unchanged.

See Also

`url2dt()` which uses this function internally.

Examples

```
# Convert a Dropbox share link
direct_url <- dropboxURL2downloadURL(
  "https://www.dropbox.com/s/abc123/myfile.csv?dl=0"
)
# Returns: "https://dl.dropboxusercontent.com/s/abc123/myfile.csv?dl=0"
```

`GetCalibratedDistThres` *GetCalibratedDistThres*

Description

Calibrates a distance threshold based on a target average number of matches per alias. Samples pairwise distances from a subset of observations to determine the threshold that would yield approximately the desired number of matches.

Usage

```
GetCalibratedDistThres(
  x = NULL,
  y = NULL,
  by.x = NULL,
  by.y = NULL,
  AveMatchNumberPerAlias = 5L,
  qgram = 2L,
  DistanceMeasure = "jaccard",
  nCores = NULL,
  mode = "euclidean"
)
```

Arguments

`x`	Input data. For `mode = "euclidean"`: an embedding matrix where rows are observations and columns are embedding dimensions. For `mode = "discrete"`: a data frame containing the column specified by `by.x`.
`y`	Input data. For `mode = "euclidean"`: an embedding matrix where rows are observations and columns are embedding dimensions. For `mode = "discrete"`: a data frame containing the column specified by `by.y`.
`by.x`	Column name in `x` to use for matching. Only used when `mode = "discrete"`.
`by.y`	Column name in `y` to use for matching. Only used when `mode = "discrete"`.

`AveMatchNumberPerAlias`
: Target average number of matches per observation. Used to calibrate the distance threshold. Default is 5.

`qgram`
: The q-gram size for string distance calculation. Only used when `mode = "discrete"`. Default is 2.

`DistanceMeasure`
: The string distance measure to use. Only used when `mode = "discrete"`. Options include "jaccard", "osa", "jw". See `?stringdist::stringdist` for all options. Default is "jaccard".

`nCores`
: Number of CPU cores for parallel computation. Only used when `mode = "discrete"`. Default is NULL (auto-detect).

`mode`
: Character string specifying the distance computation mode. Must be either "euclidean" (for embedding-based matching) or "discrete" (for string-based matching). Default is "euclidean".

Value

A numeric value representing the calibrated distance threshold.

`LinkOrgs` *LinkOrgs*

Description

Implements the organizational record linkage algorithms of Libgober and Jerzak (2023+) using half-a-billion open-collaborated records.

Usage

```
LinkOrgs(
  x = NULL,
  y = NULL,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  embedx = NULL,
  embedy = NULL,
  embedDistMetric = NULL,
  algorithm = "ml",
  conda_env = "LinkOrgs_env",
  conda_env_required = T,
  ReturnDiagnostics = F,
  ReturnProgress = T,
  ToLower = T,
  NormalizeSpaces = T,
  RemovePunctuation = T,
  MaxDist = NULL,
  MaxDist_network = NULL,
  AveMatchNumberPerAlias = 10,
  AveMatchNumberPerAlias_network = 2,
```

```
  DistanceMeasure = "jaccard",
  qgram = 2,
  RelThresNetwork = 1.5,
  ml_version = "v1",
  openBrowser = F,
  ExportEmbeddingsOnly = FALSE,
  ReturnDecomposition = FALSE,
  python_executable,
  nCores = NULL,
  deezyLoc = NULL
)
```

Arguments

x, y	Data frames to be merged.
by, by.x, by.y	Character vector(s) that specify the column names used for merging data frames x and y. The merging variables should be organizational names. See ?base::merge for more details regarding syntax.
embedx, embedy	Optional pre-computed embedding matrices. If provided, these will be used instead of computing embeddings from names. Rows correspond to observations and columns to embedding dimensions.
embedDistMetric	Optional custom distance metric function for embedding-based matching.
algorithm	Character; specifies which algorithm should be used. Options are "fuzzy", "ml", "bipartite", "markov", and "transfer". Default is "ml", which uses a machine-learning approach with Transformer networks and up to 11 million parameters to predict match probabilities using half a billion open-collaborated records as training data.
conda_env	Character string; specifies a conda environment where JAX and related packages have been installed (see ?LinkOrgs::BuildBackend). Used only when algorithm='ml' or DistanceMeasure='ml'. Default is "LinkOrgs_env".
conda_env_required	Logical; specifies whether conda environment is required. Default is TRUE.
ReturnDiagnostics	Logical; if TRUE, various match-level diagnostics are returned in the merged data frame. Default is FALSE.
ReturnProgress	Logical; if TRUE, progress messages are printed during execution. Default is TRUE.
ToLower	Logical; if TRUE, converts names to lowercase before matching. Default is TRUE.
NormalizeSpaces	Logical; if TRUE, removes extra whitespace from names. Default is TRUE.
RemovePunctuation	Logical; if TRUE, removes punctuation from names. Default is TRUE.
MaxDist	Numeric; maximum allowed distance between two matched strings. If AveMatchNumberPerAlias is specified, it takes priority over this parameter.
MaxDist_network	Numeric; maximum allowed distance for network-based matching when using algorithm = "bipartite" or "markov".

`AveMatchNumberPerAlias`
: Numeric; target average number of matches per alias. Used to automatically calibrate `MaxDist`. Takes priority over `MaxDist` if both are specified. Default is `10`.

`AveMatchNumberPerAlias_network`
: Numeric; target average number of matches per alias for network-based candidate selection. Default is `2`.

`DistanceMeasure`
: Character; algorithm for computing pairwise string distances. Options include `"osa"`, `"jaccard"`, `"jw"`, or `"ml"` for embedding-based distance. See `?stringdist::stringdist` for all string distance options. Default is `"jaccard"`.

`qgram`
: Integer; the q-gram size used in string distance measures. Default is `2`.

`RelThresNetwork`
: Numeric; relative threshold multiplier for network distances. Default is `1.5`.

`ml_version`
: Character; specifies which version of the ML algorithm to use. Options are `"v0"` (9M parameters) through `"v4"`. Default is `"v1"` (11M parameters).

`openBrowser`
: Logical; if `TRUE`, opens browser for debugging. Default is `FALSE`.

`ExportEmbeddingsOnly`
: Logical; if `TRUE` with `algorithm='ml'` (or `DistanceMeasure='ml'`), returns only ML embeddings for x and/or y without matching, for offline linkage. Default is `FALSE`.

`ReturnDecomposition`
: Logical; if `TRUE`, returns a list containing the merged data frame along with intermediate results. Default is `FALSE`.

`python_executable`
: Path to Python executable. Usually not needed if `conda_env` is specified.

`nCores`
: Integer; number of CPU cores to use for parallel processing. Default is `NULL` (auto-detect based on data size).

`deezyLoc`
: Path to DeezyMatch installation (for `algorithm = "deezymatch"`).

Details

`LinkOrgs` automatically processes the name text for each dataset (specified by `by` or `by.x` and `by.y`). Text preprocessing includes:

- **Case normalization**: Set `ToLower = FALSE` to preserve case sensitivity.
- **Space normalization**: Set `NormalizeSpaces = FALSE` to preserve whitespace.
- **Punctuation removal**: Set `RemovePunctuation = FALSE` to preserve punctuation.

To use combined machine learning and network methods, set `algorithm` to `"bipartite"` or `"markov"`, and `DistanceMeasure` to `"ml"`.

Value

If `ExportEmbeddingsOnly = TRUE`, returns a list with `embedx` and/or `embedy` data frames containing the input names and their embeddings. If `ReturnDecomposition = TRUE`, returns a list with `z` (merged data), `z_RawNames` (raw name matches), and `z_Network` (network matches). Otherwise, returns the merged data frame `z`.

Examples

```
# Create synthetic data
x_orgnames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orgnames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orgnames_x" = x_orgnames)
y <- data.frame("orgnames_y" = y_orgnames)

# Perform merge with fuzzy matching
linkedOrgs <- LinkOrgs(x = x,
                       y = y,
                       by.x = "orgnames_x",
                       by.y = "orgnames_y",
                       algorithm = "fuzzy",
                       MaxDist = 0.6)

print(linkedOrgs)
```

`pDistMatch_discrete` *Compute Discrete String Distances (Internal)*

Description

Computes pairwise string distances between organization names in two data frames using discrete distance measures (e.g., Jaccard, OSA, Jaro-Winkler). Uses trigram indexing for efficient candidate filtering and parallel processing for speed.

Usage

```
pDistMatch_discrete(
  x,
  y,
  by = NULL,
  by.x = NULL,
  by.y = NULL,
  embedDistMetric = NULL,
  return_stringdist = T,
  onlyUFT = T,
  qgram = 2,
  DistanceMeasure = "jaccard",
  MaxDist = 0.2,
  ReturnProgress = T,
  nCores = NULL,
  ReturnMaxDistThreshold = F
)
```

Arguments

x, y	Data frames containing organization names to be matched.
by	Character string; column name for matching when both data frames use the same column name. Overridden by `by.x` and `by.y` if specified.

`by.x` Character string; column name in `x` containing organization names.

`by.y` Character string; column name in `y` containing organization names.

`embedDistMetric` Not used in discrete matching (included for API consistency).

`return_stringdist` Logical; if `TRUE`, returns string distances. Default is `TRUE`.

`onlyUFT` Logical; if `TRUE`, processes only UTF-8 strings. Default is `TRUE`.

`qgram` Integer; the q-gram size for string distance calculation. Default is `2`.

`DistanceMeasure` Character; algorithm for computing pairwise string distances. Options include `"jaccard"`, `"osa"`, `"jw"`. See `?stringdist::stringdist` for all options. Default is `"jaccard"`.

`MaxDist` Numeric; maximum allowed distance between matched strings. Pairs with distances greater than this threshold are excluded. Default is `0.20`.

`ReturnProgress` Logical; if `TRUE`, progress information is available (currently disabled). Default is `TRUE`.

`nCores` Integer; number of CPU cores for parallel processing. Default is `NULL` (uses single core).

`ReturnMaxDistThreshold` Logical; if `TRUE`, returns the distance threshold used. Default is `FALSE`.

Details

This function uses a two-stage approach for efficient matching:

1. **Trigram indexing**: Builds an index of character trigrams for each name, then filters candidate pairs to those sharing at least 5% of trigrams.
2. **Distance computation**: Computes exact string distances only for filtered candidates, returning pairs with distances at or below `MaxDist`.

The function automatically swaps `x` and `y` if `y` has fewer rows than `x` for more efficient parallelization.

Value

A data frame with three columns:

ix Integer; row index in `x` of the matched record.

iy Integer; row index in `y` of the matched record.

stringdist Numeric; the string distance between the matched pair.

Returns an empty data frame if no matches are found below `MaxDist`.

See Also

`pFuzzyMatch_discrete()` for the higher-level wrapper that returns merged data, `stringdist::stringdist()` for available distance measures.

Examples

```
# Create synthetic data
x_orgnames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orgnames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orgnames_x" = x_orgnames)
y <- data.frame("orgnames_y" = y_orgnames)

# Compute distances
distances <- pDistMatch_discrete(x = x,
                                 y = y,
                                 by.x = "orgnames_x",
                                 by.y = "orgnames_y",
                                 MaxDist = 0.5)
```

---

pDistMatch_euclidean *Compute Euclidean Distances Between Embeddings (Internal)*

Description

Computes pairwise Euclidean distances between embedding vectors from two sets of observations. This function is used internally for ML-based matching where organization names have been converted to numeric embeddings.

Usage

```
pDistMatch_euclidean(
  embedx,
  embedy,
  MaxDist = NULL,
  embedDistMetric = NULL,
  ReturnProgress = T
)
```

Arguments

embedx	Numeric matrix; embeddings for the first set of observations. Rows correspond to observations and columns to embedding dimensions.
embedy	Numeric matrix; embeddings for the second set of observations. Rows correspond to observations and columns to embedding dimensions.
MaxDist	Numeric; maximum allowed Euclidean distance. Pairs with distances greater than this threshold are excluded. If NULL, all pairs are returned.
embedDistMetric	Optional function; custom distance metric. If NULL, Euclidean distance is computed. The function should take two arguments (expanded x vector and transposed y matrix) and return a distance vector.
ReturnProgress	Logical; if TRUE, progress information is available (currently disabled). Default is TRUE.

Details

This function computes Euclidean distances between all pairs of embedding vectors. For efficiency:

- Automatically swaps `embedx` and `embedy` if `embedy` has fewer rows for better vectorization.
- Uses JAX for GPU acceleration when available (detected automatically).
- Rounds embeddings to reduce precision overhead when embedding precision differs.

The function is typically called by `pFuzzyMatch_euclidean()` rather than directly by users.

Value

A data frame with three columns:

ix Integer; row index in `embedx` of the matched record.

iy Integer; row index in `embedy` of the matched record.

stringdist Numeric; the Euclidean distance between the matched pair's embeddings (named `stringdist` for consistency with discrete matching).

Returns an empty data frame if no matches are found below `MaxDist`.

See Also

`pFuzzyMatch_euclidean()` for the higher-level wrapper that returns merged data, `pDistMatch_discrete()` for string-distance-based matching.

Examples

```
## Not run:
# Create synthetic embeddings
embedx <- matrix(rnorm(4 * 256), nrow = 4)
embedy <- matrix(rnorm(4 * 256), nrow = 4)

# Compute distances
distances <- pDistMatch_euclidean(embedx = embedx,
                                  embedy = embedy,
                                  MaxDist = 5.0)

## End(Not run)
```

`pFuzzyMatch_discrete` *Fuzzy Match with Discrete String Distances*

Description

Performs fuzzy matching between two data frames using string distance measures (e.g., Jaccard, OSA, Jaro-Winkler). This is a wrapper around `pDistMatch_discrete()` that returns the merged data frame with matched records.

Usage

```
pFuzzyMatch_discrete(
  x = NULL,
  by.x = NULL,
  embedx = NULL,
  y = NULL,
  by.y = NULL,
  embedy = NULL,
  embedDistMetric = NULL,
  MaxDist = NULL,
  qgram = 2,
  DistanceMeasure = "jaccard",
  AveMatchNumberPerAlias = NULL,
  nCores = NULL,
  ...
)
```

Arguments

x, y	Data frames to be merged.
by.x	Character string; column name in x containing organization names.
embedx, embedy	Optional embedding matrices (not used in discrete matching, included for API consistency).
by.y	Character string; column name in y containing organization names.
embedDistMetric	Optional custom distance metric (not used in discrete matching).
MaxDist	Numeric; maximum allowed distance between matched strings. Pairs with distances greater than this threshold are excluded. If AveMatchNumberPerAlias is specified, it takes priority.
qgram	Integer; the q-gram size for string distance calculation. Default is 2.
DistanceMeasure	Character; algorithm for computing pairwise string distances. Options include "jaccard", "osa", "jw". See ?stringdist::stringdist for all options. Default is "jaccard".
AveMatchNumberPerAlias	Numeric; target average number of matches per alias. If specified, automatically calibrates MaxDist using GetCalibratedDistThres().
nCores	Integer; number of CPU cores for parallel processing. Default is NULL (uses single core).
...	Additional arguments (currently unused).

Details

This function uses trigram indexing to efficiently filter candidate matches before computing exact string distances. This approach significantly speeds up matching for large datasets.

Value

A data frame containing matched records from x and y, with columns from both data frames (suffixed with .x and .y respectively) and a stringdist column indicating the distance between each matched pair.

See Also

pDistMatch_discrete() for the underlying distance computation, GetCalibratedDistThres() for automatic threshold calibration, pFuzzyMatch_euclidean() for embedding-based matching.

Examples

```
# Create synthetic data
x_orgnames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orgnames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orgnames_x" = x_orgnames)
y <- data.frame("orgnames_y" = y_orgnames)

# Perform fuzzy matching
matched <- pFuzzyMatch_discrete(x = x,
                                y = y,
                                by.x = "orgnames_x",
                                by.y = "orgnames_y",
                                MaxDist = 0.5)
```

---

pFuzzyMatch_euclidean *Fuzzy Match with Euclidean Distance on Embeddings*

Description

Performs fuzzy matching between two data frames using Euclidean distance on pre-computed embeddings. This is a wrapper around pDistMatch_euclidean() that returns the merged data frame with matched records.

Usage

```
pFuzzyMatch_euclidean(
  x = NULL,
  by.x = NULL,
  embedx = NULL,
  y = NULL,
  by.y = NULL,
  embedy = NULL,
  embedDistMetric = NULL,
  MaxDist = NULL,
  AveMatchNumberPerAlias = NULL,
  ...
)
```

Arguments

x, y	Data frames to be merged.
by.x	Character string; column name in x containing organization names.
embedx, embedy	Numeric matrices containing embeddings for records in x and y respectively. Rows correspond to observations and columns to embedding dimensions. These are typically produced by the ML backend (see LinkOrgs() with algorithm = "ml").

by.y — Character string; column name in y containing organization names.

embedDistMetric — Optional custom distance metric function for computing distances between embeddings. If NULL, Euclidean distance is used.

MaxDist — Numeric; maximum allowed Euclidean distance between matched embeddings. Pairs with distances greater than this threshold are excluded. If AveMatchNumberPerAlias is specified, it takes priority.

AveMatchNumberPerAlias — Numeric; target average number of matches per alias. If specified, automatically calibrates MaxDist using GetCalibratedDistThres().

... — Additional arguments (currently unused).

Details

This function is typically used internally by LinkOrgs() when algorithm = "ml" or DistanceMeasure = "ml". It computes Euclidean distances between embedding vectors rather than string distances.

Value

A data frame containing matched records from x and y, with columns from both data frames (suffixed with .x and .y respectively) and a stringdist column indicating the Euclidean distance between each matched pair's embeddings.

See Also

pDistMatch_euclidean() for the underlying distance computation, GetCalibratedDistThres() for automatic threshold calibration, pFuzzyMatch_discrete() for string-distance-based matching.

Examples

```
## Not run:
# Create synthetic data with embeddings
x <- data.frame("orgnames_x" = c("apple", "oracle"))
y <- data.frame("orgnames_y" = c("apple corp", "oracle inc"))

# Assume embedx and embedy are pre-computed embedding matrices
# (typically produced by LinkOrgs ML backend)
embedx <- matrix(rnorm(2 * 256), nrow = 2)
embedy <- matrix(rnorm(2 * 256), nrow = 2)

matched <- pFuzzyMatch_euclidean(x = x,
                                 y = y,
                                 by.x = "orgnames_x",
                                 by.y = "orgnames_y",
                                 embedx = embedx,
                                 embedy = embedy,
                                 MaxDist = 5.0)

## End(Not run)
```

print2 *print2*

Description

Prints a message with a timestamp prefix.

Usage

```
print2(text, quiet = F)
```

Arguments

text	Character string to print.
quiet	Logical; if TRUE, suppress output. Default is FALSE.

Value

Invisibly returns NULL. Called for its side effect of printing.

Examples

```
print2("Hello world!")
```

url2dt *Download CSV from URL to data.table*

Description

Downloads a zipped CSV file from a URL and loads it into memory as a data.table. Automatically handles Dropbox URLs by converting them to direct download links.

Usage

```
url2dt(url)
```

Arguments

url	Character string; the URL pointing to a `.csv.zip` or `.csv.gz` file. Dropbox share links are automatically converted to direct download URLs.

Details

This function:

1. Converts Dropbox share links to direct download URLs using `dropboxURL2downloadURL()`
2. Downloads the file to a temporary directory
3. Unzips (if `.zip`) or decompresses (if `.gz`) the file
4. Reads the CSV file using `data.table::fread()`
5. Cleans up the temporary file

Value

A data.table containing the downloaded data.

See Also

`dropboxURL2downloadURL()` for URL conversion, `data.table::fread()` for the underlying CSV reader.

Examples

```
## Not run:
# Download from Dropbox
my_dt <- url2dt("https://www.dropbox.com/s/example/data.csv.zip?dl=0")

## End(Not run)
```

Index