

# Package ‘LinkOrgs’

December 23, 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 (>= 3.3.3)

**License** CC BY-NC-ND 4.0

**Encoding** UTF-8

**LazyData** true

**Imports**

data.table, plyr, Rfast, 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

## Contents

AssessMatchPerformance . . . . .	2
BuildBackend . . . . .	3
dropboxURL2downloadURL . . . . .	4
GetCalibratedDistThres . . . . .	5
LinkOrgs . . . . .	6
pDistMatch_discrete . . . . .	9
pDistMatch_euclidean . . . . .	11
pFuzzyMatch_discrete . . . . .	12
pFuzzyMatch_euclidean . . . . .	14
print2 . . . . .	16
url2dt . . . . .	16
<b>Index</b>	<b>18</b>

---

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

<code>x, y</code>	Data frames that were merged to produce <code>z</code> . Used to determine the universe of possible match pairs.
<code>z</code>	Data frame containing the predicted matches to be evaluated. Must contain columns specified by <code>by.x</code> and <code>by.y</code> .
<code>z_true</code>	Data frame containing the ground-truth (reference) matches. Must contain columns specified by <code>by.x</code> and <code>by.y</code> .
<code>by</code>	Character string specifying the column name for merging when both data frames use the same column name.
<code>by.x</code>	Character string specifying the column name in <code>x</code> used for merging. Defaults to <code>by</code> if not specified.
<code>by.y</code>	Character string specifying the column name in <code>y</code> used for merging. Defaults to <code>by</code> if not specified.
<code>openBrowser</code>	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_orenames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orenames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orenames_x" = x_orenames)
y <- data.frame("orenames_y" = y_orenames)
z <- data.frame("orenames_x" = x_orenames[1:2], "orenames_y" = y_orenames[1:2])
z_true <- data.frame("orenames_x" = x_orenames, "orenames_y" = y_orenames)

# Obtain match performance data
PerformanceMatrix <- AssessMatchPerformance(x = x,
                                             y = y,
                                             z = z,
                                             z_true = z_true,
                                             by.x = "orenames_x",
                                             by.y = "orenames_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

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

### 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	<i>LinkOrgs</i>
----------	-----------------

---

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

<code>x, y</code>	Data frames to be merged.
<code>by, by.x, by.y</code>	Character vector(s) that specify the column names used for merging data frames <code>x</code> and <code>y</code> . The merging variables should be organizational names. See <code>?base::merge</code> for more details regarding syntax.
<code>embedx, embedy</code>	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.
<code>embedDistMetric</code>	Optional custom distance metric function for embedding-based matching.
<code>algorithm</code>	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.
<code>conda_env</code>	Character string; specifies a conda environment where JAX and related packages have been installed (see <code>?LinkOrgs::BuildBackend</code> ). Used only when <code>algorithm='ml'</code> or <code>DistanceMeasure='ml'</code> . Default is "LinkOrgs_env".
<code>conda_env_required</code>	Logical; specifies whether conda environment is required. Default is TRUE.
<code>ReturnDiagnostics</code>	Logical; if TRUE, various match-level diagnostics are returned in the merged data frame. Default is FALSE.
<code>ReturnProgress</code>	Logical; if TRUE, progress messages are printed during execution. Default is TRUE.
<code>ToLower</code>	Logical; if TRUE, converts names to lowercase before matching. Default is TRUE.
<code>NormalizeSpaces</code>	Logical; if TRUE, removes extra whitespace from names. Default is TRUE.
<code>RemovePunctuation</code>	Logical; if TRUE, removes punctuation from names. Default is TRUE.
<code>MaxDist</code>	Numeric; maximum allowed distance between two matched strings. If <code>AveMatchNumberPerAlias</code> is specified, it takes priority over this parameter.
<code>MaxDist_network</code>	Numeric; maximum allowed distance for network-based matching when using <code>algorithm = "bipartite"</code> or <code>"markov"</code> .

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 <code>?stringdist::stringdist</code> 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 <code>algorithm='ml'</code> (or <code>DistanceMeasure='ml'</code> ), 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 <code>conda_env</code> 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 <code>algorithm = "deezymatch"</code> ).

## 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_orenames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orenames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orenames_x" = x_orenames)
y <- data.frame("orenames_y" = y_orenames)

# Perform merge with fuzzy matching
linkedOrgs <- LinkOrgs(x = x,
                       y = y,
                       by.x = "orenames_x",
                       by.y = "orenames_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.

<code>by.x</code>	Character string; column name in <code>x</code> containing organization names.
<code>by.y</code>	Character string; column name in <code>y</code> containing organization names.
<code>embedDistMetric</code>	Not used in discrete matching (included for API consistency).
<code>return_stringdist</code>	Logical; if TRUE, returns string distances. Default is TRUE.
<code>onlyUTF</code>	Logical; if TRUE, processes only UTF-8 strings. Default is TRUE.
<code>qgram</code>	Integer; the q-gram size for string distance calculation. Default is 2.
<code>DistanceMeasure</code>	Character; algorithm for computing pairwise string distances. Options include "jaccard", "osa", "jw". See <code>?stringdist::stringdist</code> for all options. Default is "jaccard".
<code>MaxDist</code>	Numeric; maximum allowed distance between matched strings. Pairs with distances greater than this threshold are excluded. Default is 0.20.
<code>ReturnProgress</code>	Logical; if TRUE, progress information is available (currently disabled). Default is TRUE.
<code>nCores</code>	Integer; number of CPU cores for parallel processing. Default is NULL (uses single core).
<code>ReturnMaxDistThreshold</code>	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_orenames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orenames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orenames_x" = x_orenames)
y <- data.frame("orenames_y" = y_orenames)

# Compute distances
distances <- pDistMatch_discrete(x = x,
                                  y = y,
                                  by.x = "orenames_x",
                                  by.y = "orenames_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 <code>?stringdist::stringdist</code> for all options. Default is "jaccard".
AveMatchNumberPerAlias	Numeric; target average number of matches per alias. If specified, automatically calibrates MaxDist using <code>GetCalibratedDistThres()</code> .
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_orenames <- c("apple", "oracle", "enron inc.", "mcdonalds corporation")
y_orenames <- c("apple corp", "oracle inc", "enron", "mcdonalds co")
x <- data.frame("orenames_x" = x_orenames)
y <- data.frame("orenames_y" = y_orenames)

# Perform fuzzy matching
matched <- pFuzzyMatch_discrete(x = x,
                                y = y,
                                by.x = "orenames_x",
                                by.y = "orenames_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 <a href="#">LinkOrgs()</a> with <code>algorithm = "ml"</code> ).



---

print2	<i>print2</i>
--------	---------------

---

### 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	<i>Download CSV from URL to data.table</i>
--------	--

---

### 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

AssessMatchPerformance, [2](#)

BuildBackend, [3](#)

data.table::fread(), [16](#), [17](#)

dropboxURL2downloadURL, [4](#)

dropboxURL2downloadURL(), [16](#), [17](#)

GetCalibratedDistThres, [5](#)

GetCalibratedDistThres(), [13–15](#)

LinkOrgs, [6](#)

LinkOrgs(), [4](#), [14](#), [15](#)

pDistMatch\_discrete, [9](#)

pDistMatch\_discrete(), [12](#), [14](#)

pDistMatch\_euclidean, [11](#)

pDistMatch\_euclidean(), [14](#), [15](#)

pFuzzyMatch\_discrete, [12](#)

pFuzzyMatch\_discrete(), [10](#), [15](#)

pFuzzyMatch\_euclidean, [14](#)

pFuzzyMatch\_euclidean(), [12](#), [14](#)

print2, [16](#)

stringdist::stringdist(), [10](#)

url2dt, [16](#)

url2dt(), [5](#)