

# Package ‘wcvpmatch’

March 23, 2026

**Title** Taxonomic Name Reconciliation Against the 'WCVP' Backbone

**Version** 0.0.1

**Description** Standardizes and reconciles scientific plant names against a World Checklist of Vascular Plants ('WCVP')-style taxonomic backbone. The package parses names into taxonomic components and applies staged exact and fuzzy matching for binomial and trinomial inputs, including infraspecific rank-aware checks. It also returns accepted-name context and row-level matching flags to support reproducible, auditable preprocessing for downstream biodiversity, spatial, and trait analyses. A user-supplied backbone can be passed through 'target\_df'; when the optional companion package 'wcvpdata' is installed, its default checklist can also be used.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Imports** assertthat, cli, dplyr, fozziejoin, lifecycle, magrittr, memoise, purrr, stringdist, stringr, tibble, tidyr

**Depends** R (>= 4.1.0)

**LazyData** true

**LazyDataCompression** xz

**Suggests** rlang (>= 1.0.0), testthat (>= 3.0.0), wcvpdata

**Additional\_repositories** <https://paulesantos.r-universe.dev>

**Config/testthat/edition** 3

**URL** <https://github.com/PaulESantos/wcvpmatch>

**BugReports** <https://github.com/PaulESantos/wcvpmatch/issues>

**NeedsCompilation** no

**Author** Paul Efren Santos Andrade [aut, cre, cph]

**Maintainer** Paul Efren Santos Andrade <paulefrens@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-03-23 09:10:03 UTC

## Contents

build_genus_index . . . . .	2
classify_snames . . . . .	3
fia . . . . .	4
prefilter_target_by_genus . . . . .	5
wcvp_direct_match . . . . .	6
wcvp_direct_match_species_within_genus . . . . .	7
wcvp_fuzzy_match_genus . . . . .	7
wcvp_fuzzy_match_species_within_genus . . . . .	8
wcvp_genus_match . . . . .	9
wcvp_matching . . . . .	10
wcvp_setup_info . . . . .	12
wcvp_suffix_match_species_within_genus . . . . .	12

<b>Index</b>	<b>14</b>
--------------	-----------

---

build_genus_index	<i>Build a Genus Index for Fast Prefiltering</i>
-------------------	--

---

### Description

#### [Experimental]

Creates a compact genus-level index from the target backbone. The index stores one row per genus and a list-column with candidate `plant_name_id` values associated with each genus.

If `plant_name_id` is not present in `target_df`, a surrogate integer ID is created to keep the index usable with custom backbones.

### Usage

```
build_genus_index(target_df = NULL)
```

### Arguments

<code>target_df</code>	Optional custom target table. If <code>NULL</code> , the optional <code>wcvpdata</code> checklist is used when available; otherwise pass a backbone explicitly.
------------------------	---

### Value

A tibble with columns:

**genus** Genus name (character).

**plant\_name\_id** List-column of unique IDs per genus.

**n\_records** Number of IDs per genus.

**genus\_nchar** Number of characters in the genus name.

**Examples**

```
library(wcvpmatch)
build_genus_index()
```

---

classify_spnames	<i>Classify Scientific Plant Names into Taxonomic Components</i>
------------------	--

---

**Description****[Experimental]**

Parse and classify scientific plant names into taxonomic components: genus, specific epithet, infraspecific rank, infraspecific epithet, and author.

Output is aligned to a backbone convention:

- Orig.Genus in Title Case (first letter uppercase, rest lowercase).
- Orig.Species and Orig.Infraspecies epithets in lowercase.
- Infra.Rank in lowercase (subsp., var., subvar., f., subf.).
- Author is recovered from the input and preserved in its original casing/punctuation (no forced uppercasing).
- Orig.Name is reconstructed as: genus + species + (rank + infra) + author.

Robustness rules:

- cf. / aff. are removed from parsing but preserved as flags (has\_cf, has\_aff).
- Hybrid markers (x/\u00D7) as standalone tokens are removed with had\_hybrid = TRUE.
- sp. / spp. triggers genus-only classification (Rank = 1, Orig.Species = NA) and sets is\_sp/is\_spp.
- If an infraspecific rank is present but the infraspecific epithet is missing, sets rank\_missing\_infra = TRUE and keeps Infra.Rank while Orig.Infraspecies = NA.
- If rank appears "late" (after author-like tokens), parsing is best-effort and rank\_late = TRUE.
- If there is no explicit rank and a third token exists, the function can infer an unranked infraspecific epithet when the third token looks epithet-like (all lowercase), and does not look like the start of an author. In that case implied\_infra = TRUE, Orig.Infraspecies is filled, Infra.Rank = NA, and Rank = 3.

**Usage**

```
classify_spnames(splist)
```

**Arguments**

splist                      Character vector. Scientific plant names.

**Value**

A tibble with one row per input name and standardized columns/flags:

**sorter** Numeric index of original order.

**Input.Name** Original input string as provided by user.

**Orig.Name** Reconstructed standardized name aligned to backbone + original-cased author.

**Orig.Genus** Genus in Title Case.

**Orig.Species** Specific epithet in lowercase, or NA for genus-only (sp. / spp.).

**Author** Recovered author string (original casing/punctuation) or "".

**Orig.Infraspecies** Infraspecific epithet in lowercase (ranked or implied), or NA.

**Infra.Rank** Infraspecific rank in lowercase (subsp., var., subvar., f., subf.), or NA.

**Rank** Numeric level: 1 genus-only, 2 genus+species, 3 includes infraspecific epithet.

**has\_cf,has\_aff,is\_sp,is\_spp,had\_hybrid,rank\_late,rank\_missing\_infra,had\_na\_author,implied\_infra**  
Logical flags.

**Examples**

```
library(wcvpmatch)
classify_spnames(c("Opuntia sp.", "Rosa canina subsp. coriifolia (Fr.) Leffler"))
classify_spnames(c("Cydonia japonica tricolor")) # implied unranked infra epithet
```

---

 fia

---

*Cleaned Master Tree Species List from FIA*


---

**Description**

A cleaned dataset containing tree species recorded by the Forest Inventory and Analysis (FIA) program of the U.S. Forest Service. This dataset is used in the examples and README of the wcvpmatch package. The data was downloaded in November 2022 from the official webpage of the Forest Inventory and Analysis National Program, available at the following [link](#), and was originally used during the development of the treemendous package. For wcvpmatch, the variable names have been standardized to Orig.Genus and Orig.Species.

**Usage**

```
fia
```

**Format**

A data frame with 2169 rows and 2 variables:

**Orig.Genus** Genus name of the species binomial

**Orig.Species** Specific epithet of the species binomial

---

 prefilter\_target\_by\_genus

*Prefilter Target Backbone by Input Genera (Exact + Fuzzy)*


---

## Description

### [Experimental]

Reduces the target backbone to genera relevant for the current input names. This is designed as a pre-step before `wcvp_matching()` to reduce search space.

Strategy:

- Exact genus candidates are always included.
- Optional fuzzy genus candidates are included when `include_fuzzy = TRUE`.
- Returned object preserves the standard target schema used by the package.

## Usage

```
prefilter_target_by_genus(
  df,
  target_df = NULL,
  genus_index = NULL,
  include_fuzzy = TRUE,
  max_dist = 1,
  method = "osa"
)
```

## Arguments

<code>df</code>	Input tibble/data.frame with either Genus/Species or Orig.Genus/Orig.Species.
<code>target_df</code>	Optional custom target table. If NULL, the optional <code>wcvpdata</code> checklist is used when available; otherwise pass a backbone explicitly.
<code>genus_index</code>	Optional pre-built index from <code>build_genus_index()</code> . If NULL, it is built on the fly.
<code>include_fuzzy</code>	Logical. If TRUE, include fuzzy-matched genera.
<code>max_dist</code>	Maximum fuzzy distance for genus matching (used when <code>include_fuzzy = TRUE</code> ).
<code>method</code>	String distance method passed to <code>fuzzyjoin</code> .

## Value

A prefiltered `target_df` tibble compatible with `wcvp_matching(target_df = ...)`. Attributes:

- candidate\_genera** Character vector of selected genera.
- exact\_genera** Character vector of exact matched genera.
- fuzzy\_genera** Character vector of fuzzy matched genera.

## Examples

```
library(wcvpmatch)
df <- data.frame(Genus = "Opuntia", Species = "yanganucensis")
prefilter_target_by_genus(df)
```

---

wcvp_direct_match	<i>Direct Match Species &amp; Genus Binomial or Trinomial names</i>
-------------------	---

---

## Description

### [Experimental]

Tries to directly match Genus + Species | Genus + Species + Rank + Infraspecies to WCVP data.

## Usage

```
wcvp_direct_match(df, target_df = NULL)
```

## Arguments

df	tibble containing the species binomial split into the columns Orig.Genus and Orig.Species.
target_df	Optional custom target table. If NULL, the optional wcvpdata checklist is used when available; otherwise pass a backbone explicitly.

## Value

Returns a tibble with the additional logical column `direct_match`, indicating whether the binomial was successfully matched (TRUE) or not (FALSE). Returns original columns plus `Matched.Genus`, `Matched.Species`, `Matched.Infra.Rank`, and `Matched.Infraspecies`.

## Examples

```
library(wcvpmatch)
# Simple binomial match
df_parsed <- classify_snames("Opuntia yanganucensis")
wcvp_direct_match(df_parsed)
```

---

`wcvp_direct_match_species_within_genus`*Direct Match Species within Genus*

---

**Description****[Experimental]**

Tries to directly match the specific epithet within an already matched genus in 'WCVP'.

**Usage**

```
wcvp_direct_match_species_within_genus(df, target_df = NULL)
```

**Arguments**

<code>df</code>	tibble containing the species binomial split into the columns <code>Orig.Genus</code> and <code>Orig.Species</code> .
<code>target_df</code>	Optional custom target table. If <code>NULL</code> , the optional <code>wcvpdata</code> checklist is used when available; otherwise pass a backbone explicitly.

**Value**

Returns a tibble with the additional logical column `direct_match_species_within_genus`, indicating whether the specific epithet was successfully matched within the matched genus (`TRUE`) or not (`FALSE`).

---

`wcvp_fuzzy_match_genus`*Fuzzy Match Genus Name*

---

**Description****[Experimental]**

Tries to fuzzy match the genus name to the 'WCVP' table (using the optional `wcvpdata` checklist by default when available).

**Usage**

```
wcvp_fuzzy_match_genus(df, target_df = NULL, max_dist = 1, method = "osa")
```

**Arguments**

df	tibble containing the species binomial split into the columns Orig.Genus and Orig.Species.
target_df	Optional custom target table. If NULL, the optional wcvpdata checklist is used when available; otherwise pass a backbone explicitly.
max_dist	Maximum edit distance used for fuzzy genus matching.
method	String distance method passed to fozziejoin (for example "osa").

**Value**

Returns a tibble with the additional logical column `fuzzy_match_genus`, indicating whether the genus was successfully matched (TRUE) or not (FALSE). Further, the additional column `fuzzy_genus_dist` returns the distance for every match.

**Examples**

```
library(wcvpmatch)
df <- data.frame(Orig.Genus = "Opuntiaa", Orig.Species = "yanganucensis")
wcvp_fuzzy_match_genus(df)
```

---

wcvp\_fuzzy\_match\_species\_within\_genus  
*Fuzzy Match Species within Genus*

---

**Description****[Experimental]**

Tries to fuzzy match the species epithet within a matched genus against 'WCVP' (using the optional wcvpdata checklist by default when available).

**Usage**

```
wcvp_fuzzy_match_species_within_genus(  
  df,  
  target_df = NULL,  
  max_dist = 1,  
  method = "osa"  
)
```

**Arguments**

df	tibble containing the species binomial split into the columns Orig.Genus and Orig.Species.
target_df	Optional custom target table. If NULL, the optional wcvpdata checklist is used when available; otherwise pass a backbone explicitly.
max_dist	Maximum edit distance used for fuzzy species matching within genus.
method	String distance method passed to fozziejoin (for example "osa").

**Value**

Returns a tibble with the additional logical column `fuzzy_match_species_within_genus`, indicating whether the specific epithet was successfully fuzzy matched within the matched genus (TRUE) or not (FALSE).

---

wcvp_genus_match	<i>Match Genus name</i>
------------------	-------------------------

---

**Description****[Experimental]**

Tries to match the genus name to the 'WCVP' table (using the optional wcvpdata checklist by default when available).

**Usage**

```
wcvp_genus_match(df, target_df = NULL)
```

**Arguments**

df	tibble containing the species binomial split into the columns Orig.Genus and Orig.Species.
target_df	Optional custom target table. If NULL, the optional wcvpdata checklist is used when available; otherwise pass a backbone explicitly.

**Value**

Returns a tibble with the additional logical column `genus_match`, indicating whether the genus was successfully matched (TRUE) or not (FALSE).

wcvp\_matching

*Match Scientific Names Against WCVF***Description****[Experimental]**

Runs a matching pipeline with exact and partial matching for binomial and trinomial names, including infraspecific rank validation.

**Usage**

```
wcvp_matching(
  df,
  target_df = NULL,
  prefilter_genus = TRUE,
  allow_duplicates = FALSE,
  max_dist = 1,
  method = "osa",
  add_name_distance = FALSE,
  name_distance_method = "osa",
  profile = FALSE,
  output_name_style = c("snake_case", "legacy")
)
```

**Arguments**

df	Input tibble/data.frame with either Genus/Species or Orig.Genus/Orig.Species. For trinomials, include Infra.Rank and Infrasppecies (or Orig.Infra.Rank/Orig.Infrasppecies).
target_df	Optional custom target table. If NULL, data are read from the optional wcvpdata checklist when available; otherwise pass target_df explicitly.
prefilter_genus	Logical. If TRUE, prefilter target_df to candidate genera (exact + fuzzy) before running the matching pipeline.
allow_duplicates	Logical. If TRUE, duplicated taxon keys are deduplicated internally for matching and then expanded back to original rows. Output includes input_index for traceability to the original input.
max_dist	Maximum distance used in all fuzzy matching stages (genus, species, infrasppecies).
method	A string indicating the fuzzy matching method (passed to fozziejoin). Supported methods: <ul style="list-style-type: none"> <li>• "levenshtein": Levenshtein edit distance (default).</li> <li>• "osa": Optimal string alignment.</li> <li>• "damerau_levenshtein" or "dl": Damerau-Levenshtein distance.</li> <li>• "hamming": Hamming distance (equal-length strings only).</li> </ul>

- "lcs": Longest common subsequence.
- "qgram": Q-gram similarity (requires q).
- "cosine": Cosine similarity (requires q).
- "jaccard": Jaccard similarity (requires q).
- "jaro": Jaro similarity.
- "jaro\_winkler" or "jw": Jaro-Winkler similarity.
- "soundex": Soundex codes based on the National Archives standard.

**add\_name\_distance**

Logical. If TRUE, add `matched_dist` as pairwise distance between input name (Input.Name fallback Orig.Name) and `matched_taxon_name`.

**name\_distance\_method**

Method passed to `stringdist::stringdist` when `add_name_distance = TRUE` (for example "osa").

**profile**

Logical. If TRUE, attach a timing table in the "timings" attribute of the returned tibble, with elapsed seconds per pipeline stage.

**output\_name\_style**

Naming style for output columns:

- "snake\_case" returns standardized lower snake\_case names.
- "legacy" keeps the historical mixed naming convention.

**Value**

Tibble with matched names, process flags, and taxonomic context columns: `matched_plant_name_id`, `matched_taxon_name`, `taxon_status`, `accepted_plant_name_id`, `accepted_taxon_name`, `is_accepted_name`.

**Examples**

```
library(wcvpmatch)
# Match a single name
wcvp_matching(data.frame(Genus = "Opuntia", Species = "yanganucensis"))

# Match multiple names with snake_case output
names <- c("Aniba heterotepala", "Anthurium quipuscoae")
df <- classify_spnames(names)
wcvp_matching(df, output_name_style = "snake_case")

# Attach per-stage timings for profiling
out <- wcvp_matching(df, output_name_style = "snake_case", profile = TRUE)
attr(out, "timings")
```

---

wcvp_setup_info	<i>Check Default Backbone Setup</i>
-----------------	-------------------------------------

---

**Description**

Reports whether the optional companion package wcvpdata is available for use as the default WCVP backbone and, if not, explains how to install it from r-universe.

**Usage**

```
wcvp_setup_info(inform = TRUE)
```

**Arguments**

inform            Logical. If TRUE (default), print a short setup message.

**Value**

Invisibly returns a named list with setup status fields: default\_backbone\_available, wcvpdata\_installed, wcvpdata\_has\_backbone, wcvpdata\_version, repository, and install\_command.

**Examples**

```
library(wcvpmatch)
wcvp_setup_info()
```

---

wcvp_suffix_match_species_within_genus	<i>Suffix Match Species within Genus</i>
--	--

---

**Description****[Experimental]**

Tries to match the specific epithet by exchanging common suffixes within an already matched genus in 'WCVP'. The following suffixes are captured: c("a", "i", "is", "um", "us", "ae").

**Usage**

```
wcvp_suffix_match_species_within_genus(df, target_df = NULL)
```

**Arguments**

df                tibble containing the species binomial split into the columns Orig.Genus and Orig.Species.

target\_df        Optional custom target table. If NULL, the optional wcvpdata checklist is used when available; otherwise pass a backbone explicitly.

**Value**

Returns a tibble with the additional logical column `suffix_match_species_within_genus`, indicating whether the specific epithet was successfully matched within the matched genus (TRUE) or not (FALSE).

**Examples**

```
library(wcvpmatch)
df <- data.frame(Orig.Genus = "Opuntia", Orig.Species = "yanganucensa", Matched.Genus = "Opuntia")
wcvp_suffix_match_species_within_genus(df)
```

# Index

## \* datasets

- [fia](#), [4](#)
- [build\\_genus\\_index](#), [2](#)
- [classify\\_snames](#), [3](#)
- [fia](#), [4](#)
- [prefilter\\_target\\_by\\_genus](#), [5](#)
- [wcvp\\_direct\\_match](#), [6](#)
- [wcvp\\_direct\\_match\\_species\\_within\\_genus](#),  
[7](#)
- [wcvp\\_fuzzy\\_match\\_genus](#), [7](#)
- [wcvp\\_fuzzy\\_match\\_species\\_within\\_genus](#),  
[8](#)
- [wcvp\\_genus\\_match](#), [9](#)
- [wcvp\\_matching](#), [10](#)
- [wcvp\\_setup\\_info](#), [12](#)
- [wcvp\\_suffix\\_match\\_species\\_within\\_genus](#),  
[12](#)