Package ‘chromer’

May 8, 2026

Title Interface to Chromosome Counts Database API

Version 0.10

Date 2025-07-28

Description A programmatic interface to the Chromosome Counts Database (<https://ccdb.tau.ac.il/>), Rice et al. (2014) <doi:10.1111/nph.13191>. This package is part of the 'ROpenSci' suite (<https://ropensci.org>).

Depends R (>= 2.15)

Imports stats, dplyr, tibble, httr

Suggests roxygen2 (>= 5.0.1), testthat

URL https://docs.ropensci.org/chromer/, https://github.com/ropensci/chromer

BugReports https://github.com/ropensci/chromer/issues

License MIT + file LICENSE

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Matthew Pennell [aut] (ORCID: <https://orcid.org/0000-0002-2886-3970>), Paula Andrea Martinez [aut] (ORCID: <https://orcid.org/0000-0002-8990-1985>), Karl W Broman [aut, cre] (ORCID: <https://orcid.org/0000-0002-4914-6671>)

Maintainer Karl W Broman <broman@wisc.edu>

Repository CRAN

Date/Publication 2025-07-28 11:50:02 UTC

Contents

chrom_counts 2
summarize_counts 3

Index **5**

`chrom_counts` *Returns chromosome counts from Chromosome Counts Database API*

Description

This function calls the Chromsome Counts Database (CCDB) API and returns all counts for specified higher taxa.

Usage

```
chrom_counts(
  taxa,
  rank = c("species", "genus", "family", "majorGroup"),
  full = FALSE,
  foptions = list()
)
```

Arguments

`taxa`	Taxonomic name(s) to query. Can be either a single name, a vector of multiple names or a list. If supplying multiple names, these must all be of the same `rank`.
`rank`	Rank to query.
`full`	Logical. Whether to return full records. Defaults to `FALSE` which returns only partial records. Partial records includes the resolved name as well as the gametophytic (n) and sporophytic (2n) counts.
`foptions`	additional options to be passed to `httr::GET`

Details

When using the API to query for species, both matched names and resolved names are searched. This means that all records for potential synonyms will be returned as well. Currently species binomials must be specified by either 'genus species' (i.e., space between genus and species) or 'genus_species'.

To search for subspecies (subsp.) or varieties (var.), you can use search terms like:

`"Solanum acaule var. albicans"`.

Searching for `"Solanum acaule"` will return all subspecies and varieties.

Currently the only acceptable search terms when specifying `"majorGroup"` are `"Angiosperms"`, `"Gymnosperms"`, `"Pteridophytes"`, or `"Bryophytes"`.

Value

A `data.frame` containing all records matched by query

Examples

```
## Not run:

## Get all counts for genus Castilleja
chrom_counts("Castilleja", "genus")

## Get all counts for both Castilleja and Lachemilla
chrom_counts(c("Castilleja", "Lachemilla"), "genus")

## Get all counts for Castilleja miniata
chrom_counts("Castilleja miniata", "species")

## Get all counts for only Castilleja miniata subsp. elata
chrom_counts("Castilleja miniata subsp. elata", "species")

## Note that searching for "Castilleja miniata" will return
## all subspecies and varieties

## Get all counts for the Orobanchaceae
chrom_counts("Orobanchaceae", "family")


## End(Not run)
```

---

summarize_counts *Summarize chromosome counts from API call*

Description

This function processes and cleans the data returned from the API call for use in downstream analysis.

Usage

```
summarize_counts(counts)
```

Arguments

counts	A 'chrom.counts' object inherited from chrom_counts.

Details

The results from the API call are a bit messy and difficult to use for downstream analyses. This function cleans up the data in three ways. First, it combines aggregates and summarizes all records from each species. Second, many of the counts are combined with text characters (e.g., "#-#", "c.#", and "#, #, #". This function uses regular expressions to pull out all and any numeric values from these strings. Third, some of the records are gametophytic (n) counts and others are from sporophytes (2n); the function simply divides the sporophytic counts in half so that all measurements are on a common scale.

IMPORTANT: Use this function with caution. Parsing the counts programmatically may be useful but it may generate erroneous results in some cases if input is in an odd format. For example, if the count is "#+-#", the function will return both the first and second # as valid counts . Given the creativity(?) of researchers in entering data, it is hard to predict all possible ways that the counts may be represented. Therefore, some manual checking will probably be necessary.

Value

A `data.frame` containing the resolved binomial, the count type (gametophytic or sporophytic), the counts, the inferred gametophytic count (for sporophytic records) and the number of records supporting each count.

Examples

```
## Not run:

## Get all counts for genus Castilleja
res <- chrom_counts("Castilleja", "genus")

## summarize the results
summarize_counts(res)


## End(Not run)
```

Index

chrom_counts, 2, *3*

summarize_counts, 3