

# Package ‘charlatan’

October 12, 2022

**Type** Package

**Title** Make Fake Data

**Description** Make fake data, supporting addresses, person names, dates, times, colors, coordinates, currencies, digital object identifiers ('DOIs'), jobs, phone numbers, 'DNA' sequences, doubles and integers from distributions and within a range.

**Version** 0.5.0

**License** MIT + file LICENSE

**URL** <https://docs.ropensci.org/charlatan/> (website)  
<https://github.com/ropensci/charlatan> (devel)

**BugReports** <https://github.com/ropensci/charlatan/issues>

**LazyData** true

**Encoding** UTF-8

**Language** en-US

**VignetteBuilder** knitr

**Imports** R6 (>= 2.2.0), tibble (>= 1.2), whisker

**Suggests** testthat, knitr, rmarkdown, ipaddress, stringi

**RoxygenNote** 7.2.1

**Collate** 'address-provider-en\_GB.R' 'address-provider-en\_NZ.R'  
'address-provider-en\_US.R' 'address-provider-nl\_NL.R'  
'base-provider.R' 'datetime-provider.R' 'address-provider.R'  
'available\_locales.R' 'charlatan-package.R'  
'charlatan\_settings.R' 'color-provider-en\_US.R'  
'color-provider-uk\_UA.R' 'color-provider.R' 'color.R'  
'company-provider-bg\_BG.R' 'company-provider-cs\_CZ.R'  
'company-provider-de\_DE.R' 'company-provider-en\_US.R'  
'company-provider-es\_MX.R' 'company-provider-fa\_IR.R'  
'company-provider-fr\_FR.R' 'company-provider-hr\_HR.R'  
'company-provider-it\_IT.R' 'company-provider.R' 'company.R'  
'coordinate-provider.R' 'coordinates.R'  
'credit\_card-provider.R' 'credit\_card.R' 'currency-provider.R'

'currency.R' 'date\_time.R' 'doi-provider.R' 'doi.R'  
 'element-provider.R' 'elements.R' 'file-provider.R'  
 'fraudster.R' 'generate.R' 'globals.R'  
 'internet-provider-bg\_BG.R' 'internet-provider-cs\_CZ.R'  
 'internet-provider-de\_DE.R' 'internet-provider-en\_AU.R'  
 'internet-provider-en\_NZ.R' 'internet-provider-fa\_IR.R'  
 'internet-provider-fr\_FR.R' 'internet-provider-hr\_HR.R'  
 'internet-provider.R' 'job.R' 'jobs-provider-da\_DK.R'  
 'jobs-provider-en\_US.R' 'jobs-provider-fa\_IR.R'  
 'jobs-provider-fi\_FI.R' 'jobs-provider-fr\_CH.R'  
 'jobs-provider-fr\_FR.R' 'jobs-provider-hr\_HR.R'  
 'jobs-provider-nl\_NL.R' 'jobs-provider-pl\_PL.R'  
 'jobs-provider-ru\_RU.R' 'jobs-provider-uk\_UA.R'  
 'jobs-provider-zh\_TW.R' 'jobs-provider.R'  
 'lorem-provider-ar\_AA.R' 'lorem-provider-el\_GR.R'  
 'lorem-provider-en\_US.R' 'lorem-provider-he\_IL.R'  
 'lorem-provider-ja\_JP.R' 'lorem-provider-la.R'  
 'lorem-provider-ru\_RU.R' 'lorem-provider-zh\_CN.R'  
 'lorem-provider-zh\_TW.R' 'lorem-provider.R' 'misc-provider.R'  
 'missing-data-provider.R' 'missing.R' 'name.R'  
 'numerics-provider.R' 'numerics.R' 'onload.R'  
 'person-provider-bg\_BG.R' 'person-provider-cs\_CZ.R'  
 'person-provider-da\_DK.R' 'person-provider-de\_AT.R'  
 'person-provider-de\_DE.R' 'person-provider-en\_GB.R'  
 'person-provider-en\_NZ.R' 'person-provider-en\_US.R'  
 'person-provider-es\_ES.R' 'person-provider-es\_MX.R'  
 'person-provider-fa\_IR.R' 'person-provider-fi\_FI.R'  
 'person-provider-fr\_CH.R' 'person-provider-fr\_FR.R'  
 'person-provider-hr\_HR.R' 'person-provider-it\_IT.R'  
 'person-provider-ja\_JP.R' 'person-provider-ko\_KR.R'  
 'person-provider-lt\_LT.R' 'person-provider-lv\_LV.R'  
 'person-provider-ne\_NP.R' 'person-provider-nl\_NL.R'  
 'person-provider-no\_NO.R' 'person-provider-pl\_PL.R'  
 'person-provider.R' 'phone-number-provider-all.R'  
 'phone\_number.R' 'phonenumbers-provider.R'  
 'sequence-provider.R' 'sequences.R' 'taxonomy-provider.R'  
 'taxonomy.R' 'useragent-provider.R' 'zzz.R'

**NeedsCompilation** no

**Author** Scott Chamberlain [aut, cre] (<<https://orcid.org/0000-0003-1444-9135>>),  
 Kyle Voytovich [aut],  
 Martin Pedersen [ctb],  
 Brooke Anderson [rev] (Brooke Anderson reviewed the package for  
 rOpenSci, see <https://github.com/ropensci/onboarding/issues/94>),  
 Tristan Mahr [rev] (Tristan Mahr reviewed the package for rOpenSci, see  
<https://github.com/ropensci/onboarding/issues/94>),  
 rOpenSci [fnd] (<https://ropensci.org>)

**Maintainer** Scott Chamberlain <myrmecocystus+r@gmail.com>

Repository CRAN

Date/Publication 2022-08-18 12:00:09 UTC

**R topics documented:**

charlatan-package . . . . .	3
available_locales_df . . . . .	4
BaseProvider . . . . .	5
charlatan_locales . . . . .	8
charlatan_settings . . . . .	8
ch_color . . . . .	9
ch_company . . . . .	10
ch_credit . . . . .	11
ch_currency . . . . .	12
ch_doi . . . . .	12
ch_generate . . . . .	13
ch_gene_sequence . . . . .	14
ch_job . . . . .	14
ch_missing . . . . .	15
ch_name . . . . .	16
ch_phone_number . . . . .	16
coordinates . . . . .	17
date_time . . . . .	18
elements . . . . .	19
fraudster . . . . .	19
numerics . . . . .	20
taxonomy . . . . .	21
<b>Index</b>	<b>23</b>

---

charlatan-package	<i>charlatan</i>
-------------------	------------------

---

**Description**

Make fake data, supporting addresses, person names, dates, times, colors, coordinates, currencies, digital object identifiers (DOIs), jobs, phone numbers, DNA sequences, doubles and integers from distributions and within a range.

**Package API**

- `ch_generate()`: generate a data.frame with fake data
- `fraudster()`: single interface to all fake data methods
- High level interfaces: There are high level functions prefixed with `ch_` that wrap low level interfaces, and are meant to be easier to use and provide easy way to make many instances of a thing.

- Low level interfaces: All of these are R6 objects that a user can initialize and then call methods on the them.

### Author(s)

Scott Chamberlain <myrmecocystus+r@gmail.com>

Kyle Voytovich

Martin Pedersen

### Examples

```
# generate individual types of data
ch_name()
ch_phone_number()
ch_job()

# generate a data.frame
ch_generate()

# one interface to all data types - generate the class first
# reports the locale to be used, can change optionally
(x <- fraudster())
x$job()
x$name()
x$color_name()
x$hex_color()

# low level interfaces to "data providers"
# these are exported by hidden from package man page
# as most users will likely not interact with these
x <- ColorProvider$new()
x$color_name()
x$hex_color()
```

---

available\_locales\_df *Available locales*

---

### Description

A data.frame of locales available in **charlatan**

### Format

A data frame with 45 rows and 4 variables:

**Language** language two letter code

**Country** country two letter code

**Variant** a variant code, if applicable

**Name** official locale two letter code

**See Also**

data.frame used in [charlatan\\_locales\(\)](#)

---

BaseProvider

*BaseProvider*

---

**Description**

BaseProvider

BaseProvider

**Methods****Public methods:**

- [BaseProvider\\$random\\_element\(\)](#)
- [BaseProvider\\$random\\_element\\_prob\(\)](#)
- [BaseProvider\\$random\\_int\(\)](#)
- [BaseProvider\\$random\\_digit\(\)](#)
- [BaseProvider\\$random\\_digit\\_not\\_zero\(\)](#)
- [BaseProvider\\$random\\_digit\\_or\\_empty\(\)](#)
- [BaseProvider\\$random\\_digit\\_not\\_zero\\_or\\_empty\(\)](#)
- [BaseProvider\\$random\\_letter\(\)](#)
- [BaseProvider\\$numerify\(\)](#)
- [BaseProvider\\$lexify\(\)](#)
- [BaseProvider\\$bothify\(\)](#)
- [BaseProvider\\$check\\_locale\(\)](#)
- [BaseProvider\\$randomize\\_nb\\_elements\(\)](#)
- [BaseProvider\\$clone\(\)](#)

**Method** [random\\_element\(\)](#): pick a random element from vector/list

*Usage:*

`BaseProvider$random_element(x)`

*Arguments:*

x vector or list

*Returns:* a single element from x

**Method** [random\\_element\\_prob\(\)](#): pick a random element with probability from vector/list

*Usage:*

`BaseProvider$random_element_prob(x)`

*Arguments:*

x vector or list

**Method** `random_int()`: any number of random integers from a min, max

*Usage:*

```
BaseProvider$random_int(min = 0, max = 9999, size = 1)
```

*Arguments:*

`min` the minimum value. default: 0

`max` the maximum value. default: 9999

`size` number of values to return. default: 1

*Returns:* random integer

**Method** `random_digit()`: random integer between 0 and 9

*Usage:*

```
BaseProvider$random_digit()
```

**Method** `random_digit_not_zero()`: random integer between 1 and 9

*Usage:*

```
BaseProvider$random_digit_not_zero()
```

**Method** `random_digit_or_empty()`: random integer between 0 and 9 or empty character string

*Usage:*

```
BaseProvider$random_digit_or_empty()
```

**Method** `random_digit_not_zero_or_empty()`: random integer between 1 and 9 or empty character string

*Usage:*

```
BaseProvider$random_digit_not_zero_or_empty()
```

**Method** `random_letter()`: random letter

*Usage:*

```
BaseProvider$random_letter()
```

**Method** `numerify()`: replace a template with numbers

*Usage:*

```
BaseProvider$numerify(text = "###")
```

*Arguments:*

`text` (character) a string

**Method** `lexify()`: replace a template with letters

*Usage:*

```
BaseProvider$lexify(text = "????")
```

*Arguments:*

`text` (character) a string

**Method** `bothify()`: both numerify and lexify together

*Usage:*

```
BaseProvider$bothify(text = "## ??")
```

*Arguments:*

text (character) a string

**Method** `check_locale()`: check a locale to see if it exists, if not, stop with error message

*Usage:*

```
BaseProvider$check_locale(x)
```

*Arguments:*

x a locale name, e.g. 'bg\_BG'

*Returns:* returns nothing if locale is supported; stops w/ message if not

**Method** `randomize_nb_elements()`: Returns a random value near number

*Usage:*

```
BaseProvider$randomize_nb_elements(
  number = 10,
  le = FALSE,
  ge = FALSE,
  min = NULL,
  max = NULL
)
```

*Arguments:*

number value to which the result must be near

le result must be lower or equal to number

ge result must be greater or equal to number

min the minimum value. default: NULL

max the maximum value. default: NULL

*Returns:* a random int near number

**Method** `clone()`: The objects of this class are cloneable with this method.

*Usage:*

```
BaseProvider$clone(deep = FALSE)
```

*Arguments:*

deep Whether to make a deep clone.

## Examples

```
(x <- BaseProvider$new())

x$numerify("##%asdf221?")
x$lexify("##%asdf221?")
x$bothify("##%asdf221?")

z <- PhoneNumberProvider$new()
x$numerify(z$render())
```

```
x$random_element(letters)
x$random_int()
x$random_digit()
x$random_digit_not_zero()
x$random_digit_or_empty()
x$random_digit_not_zero_or_empty()
x$random_letter()
x$check_locale("es_ES")
## fails
# x$check_locale("es_EQ")

x$randomize_nb_elements()
```

---

charlatan\_locales      *Available locales*

---

### Description

Available locales

### Usage

```
charlatan_locales()
```

### Value

a data.frame of the available locales in this package. See [available\\_locales\\_df](#) for structure.

Not all functions support all locales. Check the docs for each one to see what locales they support.

You can find out more about each locale by running your locale though `stringi::stri_locale_info()`

### Examples

```
charlatan_locales()
```

---

charlatan\_settings      *charlatan settings*

---

### Description

charlatan settings

### Usage

```
charlatan_settings(messy = NULL)
```



**Arguments**

messy (logical) make some messy data. Default: NULL

**More deets**

- messy - When FALSE, nothing is different from normal. When TRUE, we select incorrect/wrong values with probability X. Messy mode is only available for **en-US** for now, and only for some data types. The default setting is NULL, meaning it is ignored.

**Examples**

```

charlatan_settings()
charlatan_settings(messy = TRUE)
charlatan_settings(messy = FALSE)

# with PersonProvider - overrides local messy param in all cases
x <- PersonProvider$new()
x$messy
charlatan_settings(messy = TRUE)
x <- PersonProvider$new()
x$messy

```

---

ch\_color *Create fake colors*

---

**Description**

Create fake colors

**Usage**

```

ch_color_name(n = 1, locale = NULL)

ch_safe_color_name(n = 1, locale = NULL)

ch_hex_color(n = 1)

ch_safe_hex_color(n = 1)

ch_rgb_color(n = 1)

ch_rgb_css_color(n = 1)

```

**Arguments**

n (integer) number of things to get, any non-negative integer

locale (character) the locale to use. See ColorProvider\$new()\$allowed\_locales() for locales supported. Affects the ch\_color\_name and ch\_safe\_color\_name functions

**See Also**[ColorProvider](#)**Examples**

```
ch_color_name()
ch_color_name(10)
ch_color_name(500)

ch_safe_color_name()
ch_safe_color_name(10)

ch_hex_color()
ch_hex_color(10)
ch_hex_color(1000)

ch_safe_hex_color()
ch_safe_hex_color(10)

ch_rgb_color()
ch_rgb_color(10)

ch_rgb_css_color()
ch_rgb_css_color(10)

ch_color_name(locale = "uk_UA")
ch_color_name(n = 10, locale = "uk_UA")

ch_safe_color_name(locale = "uk_UA")
ch_safe_color_name(n = 10, locale = "uk_UA")
```

---

`ch_company`*Create fake company names and other company bits*

---

**Description**

Create fake company names and other company bits

**Usage**

```
ch_company(n = 1, locale = NULL)
```

**Arguments**

<code>n</code>	(integer) number of things to get, any non-negative integer
<code>locale</code>	(character) the locale to use. See <code>CompanyProvider\$new()\$allowed_locales()</code> for locales supported.

**See Also**

[CompanyProvider](#)

**Examples**

```
ch_company()
ch_company(10)
ch_company(500)

ch_company(locale = "fr_FR", n = 10)
ch_company(locale = "cs_CZ", n = 10)
ch_company(locale = "es_MX", n = 10)
ch_company(locale = "hr_HR", n = 10)
```

---

ch_credit	<i>Create fake credit card data</i>
-----------	-------------------------------------

---

**Description**

Create fake credit card data

**Usage**

```
ch_credit_card_provider(n = 1)

ch_credit_card_number(n = 1)

ch_credit_card_security_code(n = 1)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

**See Also**

[CreditCardProvider](#)

**Examples**

```
ch_credit_card_provider()
ch_credit_card_provider(n = 4)

ch_credit_card_number()
ch_credit_card_number(n = 10)
ch_credit_card_number(n = 500)

ch_credit_card_security_code()
ch_credit_card_security_code(n = 10)
ch_credit_card_security_code(n = 500)
```

---

ch_currency	<i>Create fake currencies</i>
-------------	-------------------------------

---

**Description**

Create fake currencies

**Usage**

```
ch_currency(n = 1)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

**See Also**

[CurrencyProvider](#)

**Examples**

```
ch_currency()  
ch_currency(10)  
ch_currency(500)
```

---

ch_doi	<i>Create fake DOIs (Digital Object Identifiers)</i>
--------	------------------------------------------------------

---

**Description**

Create fake DOIs (Digital Object Identifiers)

**Usage**

```
ch_doi(n = 1)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

**See Also**

[DOIProvider](#)

**Examples**

```
ch_doi()  
ch_doi(10)  
ch_doi(100)
```

---

ch_generate	<i>Generate a fake dataset</i>
-------------	--------------------------------

---

### Description

Generate a fake dataset

### Usage

```
ch_generate(..., n = 10, locale = NULL)
```

### Arguments

...	columns to include. must be in the allowed set. See <b>Allowed column names</b> below. Three default columns are included (name, job, phone_number) if nothing is specified - but are overridden by any input.
n	(integer) number of things to get, any non-negative integer
locale	(character) the locale to use. options: only supported for data types that have locale support, See each data provider for details.

### Allowed column names

- name (default included)
- job (default included)
- phone\_number (default included)
- currency
- color\_name
- rgb\_color
- rgb\_css\_color

### Examples

```
ch_generate()
ch_generate(n = 1)
ch_generate(n = 100)

ch_generate('job')
ch_generate('job', 'name')
ch_generate('job', 'color_name')

# locale
ch_generate(locale = "en_US")
ch_generate(locale = "fr_FR")
ch_generate(locale = "fr_CH")
```

---

ch\_gene\_sequence      *Create fake gene sequences*

---

**Description**

Create fake gene sequences

**Usage**

```
ch_gene_sequence(n = 1, length = 30)
```

**Arguments**

n                    (integer) number of things to get, any non-negative integer  
length              (integer) length of sequence to create

**See Also**

[SequenceProvider](#)

**Examples**

```
ch_gene_sequence()  
ch_gene_sequence(10)  
ch_gene_sequence(100)  
  
ch_gene_sequence(length = 500)  
ch_gene_sequence(10, length = 500)
```

---

ch\_job                    *Create fake jobs*

---

**Description**

Create fake jobs

**Usage**

```
ch_job(n = 1, locale = NULL)
```

**Arguments**

n                    (integer) number of things to get, any non-negative integer  
locale              (character) the locale to use. Run `JobProvider$new()$allowed_locales()`  
for locales supported (default: en\_US)

**See Also**[JobProvider](#)**Examples**

```
ch_job()
ch_job(10)
ch_job(500)

ch_job(locale = "da_DK", n = 10)
ch_job(locale = "fi_FI", n = 10)
ch_job(locale = "fr_FR", n = 10)
ch_job(locale = "fr_CH", n = 10)
ch_job(locale = "hr_HR", n = 10)
ch_job(locale = "fa_IR", n = 10)
ch_job(locale = "pl_PL", n = 10)
ch_job(locale = "ru_RU", n = 10)
ch_job(locale = "uk_UA", n = 10)
ch_job(locale = "zh_TW", n = 10)
```

---

ch\_missing

*Create missing data*

---

**Description**

Create missing data

**Usage**

```
ch_missing(x, n = 1)
```

**Arguments**

x	Input vector, can be any class - only 1 vector
n	(integer) number of things to get, any non-negative integer

**See Also**[MissingDataProvider](#)**Examples**

```
ch_missing(letters)
ch_missing(letters, 10)
ch_missing(letters, 20)
```

---

ch\_name *Create fake person names*

---

**Description**

Create fake person names

**Usage**

```
ch_name(n = 1, locale = NULL, messy = FALSE)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

locale (character) the locale to use. See `PersonProvider$new()$allowed_locales()` for locales supported (default: en\_US)

messy (logical) make some messy data. Default: FALSE

**See Also**

[PersonProvider](#)

**Examples**

```
ch_name()  
ch_name(10)  
ch_name(500)  
  
ch_name(locale = "fr_FR", n = 10)  
ch_name(locale = "fr_CH", n = 10)  
ch_name(locale = "fa_IR", n = 10)  
ch_name(locale = "fi_FI", n = 10)
```

---

ch\_phone\_number *Create fake phone numbers*

---

**Description**

Create fake phone numbers

**Usage**

```
ch_phone_number(n = 1, locale = NULL)
```



**Arguments**

n (integer) number of things to get, any non-negative integer

locale (character) the locale to use. See `PhoneNumberProvider$new()$allowed_locales()` for locales supported (default: en\_US)

**See Also**

[PhoneNumberProvider](#)

**Examples**

```
ch_phone_number()
ch_phone_number(10)
ch_phone_number(500)

# locales
ch_phone_number(locale = "fr_FR")
ch_phone_number(locale = "uk_UA")
ch_phone_number(locale = "en_CA")
ch_phone_number(locale = "lv_LV")
```

---

coordinates

*Create fake coordinates*

---

**Description**

Create fake coordinates

**Usage**

```
ch_lon(n = 1)
ch_lat(n = 1)
ch_position(n = 1, bbox = NULL)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

bbox a bounding box of the form [w, s, e, n]

**See Also**

[CoordinateProvider](#)

**Examples**

```
ch_lon()
ch_lon(10)

ch_lat()
ch_lat(10)

ch_position()
ch_position(10)
ch_position(bbox = c(-120, 30, -110, 60))
```

---

date\_time

*Create dates and times*

---

**Description**

Create dates and times

**Usage**

```
ch_timezone(n = 1)
ch_unix_time(n = 1)
ch_date_time(n = 1)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

**See Also**

[DateTimeProvider](#)

**Examples**

```
ch_timezone()
ch_timezone(10)

ch_unix_time()
ch_unix_time(20)

ch_date_time()
ch_date_time(20)
```

---

elements	<i>Get elements</i>
----------	---------------------

---

**Description**

Get elements

**Usage**

```
ch_element_symbol(n = 1)
```

```
ch_element_element(n = 1)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

**See Also**

[ElementProvider](#)

**Examples**

```
ch_element_symbol()  
ch_element_symbol(10)  
ch_element_symbol(50)
```

```
ch_element_element()  
ch_element_element(10)  
ch_element_element(50)
```

---

fraudster	<i>Fraudster - catch all client to make all types of fake data</i>
-----------	--------------------------------------------------------------------

---

**Description**

Fraudster - catch all client to make all types of fake data

**Usage**

```
fraudster(locale = NULL)
```

**Arguments**

locale (character) the locale to use. options: en\_US (default), fr\_FR, fr\_CH, hr\_FR, fa\_IR, pl\_PL, ru\_RU, uk\_UA, zh\_TW.

**Examples**

```
# English - the default locale
(x <- fraudster())
x$job()
x$name()
x$color_name()
x$safe_color_name()
x$hex_color()
x$safe_hex_color()
x$rgb_color()
x$rgb_css_color()

# different locales
## French
(y <- fraudster(locale = "fr_FR"))
y$job()

## Croatian
(z <- fraudster(locale = "hr_HR"))
z$job()

## Ukranian
(w <- fraudster(locale = "uk_UA"))
w$job()
w$color_name()

# geospatial
x$lat()
x$lon()
x$position()

# DOIs (Digital Object Identifier)
x$doi()
```

---

numerics

*Create numbers*

---

**Description**

Create numbers

**Usage**

```
ch_double(n = 1, mean = 0, sd = 1)

ch_integer(n = 1, min = 1, max = 1000)

ch_unif(n = 1, min = 0, max = 9999)
```

```
ch_norm(n = 1, mean = 0, sd = 1)
ch_lnorm(n = 1, mean = 0, sd = 1)
ch_beta(n = 1, shape1, shape2, ncp = 0)
```

### Arguments

n	(integer) number of things to get, any non-negative integer
mean	mean value
sd	standard deviation
min	minimum value
max	maximum value
shape1, shape2	non-negative parameters of the Beta distribution
ncp	non-centrality parameter

### Examples

```
ch_double()
ch_double(10)
ch_double(100)

ch_integer()
ch_integer(10)
ch_integer(100)

ch_unif()
ch_norm()
ch_lnorm()
ch_beta(shape1 = 1, shape2 = 1)
```

---

taxonomy	<i>Create fake taxonomic names</i>
----------	------------------------------------

---

### Description

Create fake taxonomic names

### Usage

```
ch_taxonomic_genus(n = 1)
ch_taxonomic_epithet(n = 1)
ch_taxonomic_species(n = 1)
```

**Arguments**

n (integer) number of things to get, any non-negative integer

**Names**

Names were taken from Theplantlist. 500 genera names and 500 epithets were chosen at random from the set of 10,000 names in the dataset in the `taxize` package. Theplantlist is, as it says on the tin, composed of plant names - so these fake names are derived from plant names if that matters to you. These may generate names that match those of real taxa, but may not as well.

**Taxonomic authority**

Randomly, the taxonomic authority is in parentheses - which represents that the given authority was not the original authority.

**See Also**

[TaxonomyProvider](#)

**Examples**

```
ch_taxonomic_genus()  
ch_taxonomic_genus(10)  
ch_taxonomic_genus(500)  
  
ch_taxonomic_epithet()  
ch_taxonomic_epithet(10)  
ch_taxonomic_epithet(500)  
  
ch_taxonomic_species()  
ch_taxonomic_species(10)  
ch_taxonomic_species(500)
```

# Index

- \* **data**
  - available\_locales\_df, 4
- \* **package**
  - charlatan-package, 3
- available\_locales\_df, 4, 8
- BaseProvider, 5
- ch\_beta (numerics), 20
- ch\_color, 9
- ch\_color\_name (ch\_color), 9
- ch\_company, 10
- ch\_credit, 11
- ch\_credit\_card\_number (ch\_credit), 11
- ch\_credit\_card\_provider (ch\_credit), 11
- ch\_credit\_card\_security\_code (ch\_credit), 11
- ch\_currency, 12
- ch\_date\_time (date\_time), 18
- ch\_doi, 12
- ch\_double (numerics), 20
- ch\_element\_element (elements), 19
- ch\_element\_symbol (elements), 19
- ch\_gene\_sequence, 14
- ch\_generate, 13
- ch\_generate(), 3
- ch\_hex\_color (ch\_color), 9
- ch\_integer (numerics), 20
- ch\_job, 14
- ch\_lat (coordinates), 17
- ch\_lnorm (numerics), 20
- ch\_lon (coordinates), 17
- ch\_missing, 15
- ch\_name, 16
- ch\_norm (numerics), 20
- ch\_phone\_number, 16
- ch\_position (coordinates), 17
- ch\_rgb\_color (ch\_color), 9
- ch\_rgb\_css\_color (ch\_color), 9
- ch\_safe\_color\_name (ch\_color), 9
- ch\_safe\_hex\_color (ch\_color), 9
- ch\_taxonomic\_epithet (taxonomy), 21
- ch\_taxonomic\_genus (taxonomy), 21
- ch\_taxonomic\_species (taxonomy), 21
- ch\_timezone (date\_time), 18
- ch\_unif (numerics), 20
- ch\_unix\_time (date\_time), 18
- charlatan (charlatan-package), 3
- charlatan-package, 3
- charlatan\_locales, 8
- charlatan\_locales(), 5
- charlatan\_settings, 8
- ColorProvider, 10
- CompanyProvider, 11
- CoordinateProvider, 17
- coordinates, 17
- CreditCardProvider, 11
- CurrencyProvider, 12
- date\_time, 18
- DateTimeProvider, 18
- DOIProvider, 12
- ElementProvider, 19
- elements, 19
- fraudster, 19
- fraudster(), 3
- JobProvider, 15
- MissingDataProvider, 15
- numerics, 20
- PersonProvider, 16
- PhoneNumberProvider, 17
- SequenceProvider, 14
- taxonomy, 21
- TaxonomyProvider, 22