Package: dialr (via r-universe)

October 24, 2026

Title Parse, Format, and Validate International Phone Numbers

Version 0.4.2

Description Parse, format, and validate international phone numbers using Google's 'libphonenumber' java library, <https://github.com/google/libphonenumber>.

License GPL (>= 3)

URL https://socialresearchcentre.github.io/dialr/, https://github.com/socialresearchcentre/dialr, https://github.com/socialresearchcentre/dialrjars, https://github.com/google/libphonenumber

BugReports https://github.com/socialresearchcentre/dialr/issues

Depends R (>= 3.6.0)

Imports dialrjars (>= 8.10.11.1), rJava

Suggests covr, dplyr, knitr, pillar, rmarkdown, testthat (>= 2.1.0)

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.2.3

SystemRequirements java (>= 1.6)

Author Danny Smith [aut, cre, cph], The Social Research Centre [cph]

Maintainer Danny Smith <danny@gorcha.org>

Config/pak/sysreqs default-jdk

Repository https://cran.r-universe.dev

Date/Publication 2023-10-16 22:31:08 UTC

RemoteUrl https://github.com/cran/dialr

RemoteRef HEAD

RemoteSha b9fea62646be4012f02ffd0481b7af178b085ba7

Contents

check_cc . . . 2
dialr-example . . . 3
dialr-match . . . 4
dialr-phone . . . 5
dialr-region . . . 8
dialr-type . . . 9
dialr-valid . . . 10
get_carrier . . . 12
get_cc . . . 13
get_geocode . . . 13
get_timezone . . . 15

Index **16**

`check_cc` *Check ISO country code*

Description

Check whether an ISO country code is valid.

Usage

```
check_cc(country)
```

Arguments

`country`	A character vector of ISO country codes.

Value

A logical vector flagging which elements are valid codes.

Examples

```
check_cc(c("AU", "US", "CN", "WRONG", NA))
```

`dialr-example` *Get an example phone number*

Description

Produces example phone numbers for the given `region`, `type` and `valid` combinations. Input vectors are recycled as necessary if a vector of length 1 is provided.

Usage

```
get_example(region, type = NULL, valid = TRUE)
```

Arguments

`region`	A character vector of ISO country codes.
`type`	A character vector of phone number types for each region. If `NULL` (default), returns an example "FIXED_LINE" number. Returns an empty phone number if `type` is not valid for the provided `region`.
`valid`	A logical vector. For each `FALSE` entry, `get_example` returns an example invalid number, and `type` is ignored.

Value

A phone vector.

libphonenumber reference

`get_example()`: `PhoneNumberUtil.getExampleNumberForType()`; `PhoneNumberUtil.getExampleNumber()` if `type` is `NULL` or `NA`; `PhoneNumberUtil.getInvalidExampleNumber()` if `valid` is `FALSE`.

See Also

`get_supported_regions()` for valid region codes, `get_types_for_region()` to get valid phone types for a region.

Other phone functions: `dialr-match`, `dialr-phone`, `dialr-region`, `dialr-type`, `dialr-valid`, `dialr`

Examples

```
# Get a basic example number
get_example("AU")

# Get an example mobile number
get_example("AU", type = "MOBILE")

# Example phone number for an invalid type
get_example("AU", type = "VOICEMAIL")
```

```
# Get an example invalid number
get_example("AU", valid = FALSE)

# Get a combination of the previous examples
get_example(c("AU", "AU",     "AU",        "AU" ),
            c(NA,   "MOBILE", "VOICEMAIL", NA   ),
            c(TRUE, TRUE,     TRUE,        FALSE))
```

`dialr-match` *Phone number equality checks*

Description

Check if two vectors contain matching phone numbers. See Details section for a full list of match types. `is_match()` with default arguments is used to implement == and != for phone vectors.

`is_match()` accepts phone or atomic vectors. Atomic vectors are converted to character for comparison. Note that although they can contain formatting character vectors are not parsed with a default region, so they will only ever be an `"EXACT_MATCH"` if a country calling code is specified with + at the start. See Examples.

Usage

```
is_match(e1, e2, detailed = FALSE, strict = TRUE, not_number_na = TRUE)
```

Arguments

`e1`	A phone or character vector.
`e2`	A phone or character vector.
`detailed`	If `FALSE`, `is_match()` returns a logical vector. If `TRUE`, it returns a character vector with the match type. See Details section for possible return values.
`strict`	If `TRUE`, only `"EXACT_MATCH"` is treated as a match. If `FALSE`, `"EXACT_MATCH"`, `"NSN_MATCH"` and `"SHORT_NSN_MATCH"` are all considered a match. Ignored if `detailed = TRUE`.
`not_number_na`	If `TRUE`, `"NOT_A_NUMBER"` is converted to `NA`.

Details

Possible return values for `is_match(x, detailed = TRUE)`:

- `"EXACT_MATCH"`: The country_code, NSN, presence of a leading zero for Italian numbers and any extension present are the same.
- `"NSN_MATCH"`: Either or both values has no region specified, and the NSNs and extensions are the same.
- `"SHORT_NSN_MATCH"`: Either or both values has no region specified, or the region specified is the same, and one NSN could be a shorter version of the other number. This includes the case where one has an extension specified, and the other does not.

- "NOT_A_NUMBER": One of the input phone numbers failed to parse.
- "NO_MATCH": All others.

For example, the numbers +1 345 657 1234 and 657 1234 are a "SHORT_NSN_MATCH". The numbers +1 345 657 1234 and 345 657 are a "NO_MATCH".

Value

A logical or character vector.

libphonenumber reference

is_match(): PhoneNumberUtil.isNumberMatch()

See Also

Other phone functions: dialr-example, dialr-phone, dialr-region, dialr-type, dialr-valid, dialr

Examples

```
is_match(phone("0412 345 678", "AU"), phone("+61412345678", "AU"))

phone("0412 345 678", "AU") == phone("+61412345678", "AU")
phone("0412 345 678", "AU") != phone("+61412345678", "AU")

# character vectors are only fully specified with a country calling code
is_match("0412345678", "0412345678", detailed = TRUE)
is_match("+61412345678", "+61412345678", detailed = TRUE)

is_match(phone("0412345678", "AU"), "0412345678", detailed = TRUE)
is_match(phone("+61412345678", "AU"), "+61412345678", detailed = TRUE)
```

---

dialr-phone *Phone number parsing and formatting*

Description

A phone vector stores phone numbers parsed with libphonenumber for formatting and further processing.

Usage

```
phone(
  x = character(),
  region = character(),
  show_progress = getOption("dialr.show_progress")
)
```

```
phone_reparse(x)

is.phone(x)

## S3 method for class 'phone'
print(x, n = 10, ...)

## S3 method for class 'phone'
format(
  x,
  format = c("E164", "NATIONAL", "INTERNATIONAL", "RFC3966"),
  home = NULL,
  clean = TRUE,
  strict = FALSE,
  ...
)

## S3 method for class 'phone'
as.character(x, raw = TRUE, ...)
```

Arguments

x	A character vector of phone numbers.
region	A character vector of ISO country codes with the default region for each phone number in x. A region vector of length 1 will be recycled to the length of x. If NA or "", numbers written in international format (with a leading +) will be parsed without a default region.
show_progress	Should a progress bar be displayed? Defaults to the option dialr.show_progress.
n	Number of elements to print.
...	Additional arguments for specific methods.
format	Phone number format to use from one of four standards: • "E164": general format for international telephone numbers from ITU-T Recommendation E.164 • "NATIONAL": national notation from ITU-T Recommendation E.123 • "INTERNATIONAL": international notation from ITU-T Recommendation E.123 • "RFC3966": "tel" URI syntax from the IETF tel URI for Telephone Numbers See notes from the libphonenumber javadocs for more details. format defaults to "E164". The default can be set in option dialr.format.
home	ISO country code for home region. If provided, numbers will be formatted for dialing from the home region.
clean	Should non-numeric characters be removed? If TRUE, keeps numbers and leading "+".

strict	Should invalid phone numbers be removed? If TRUE, invalid phone numbers are replaced with NA.
raw	If TRUE, the raw phone number is returned. Otherwise elements are cleaned with format().

Details

libphonenumber defines the PhoneNumberUtil class, with a set of functions for extracting information from and performing processing on a parsed Phonenumber object. A text phone number must be parsed before any other operations (e.g. checking phone number validity, formatting) can be performed. When parsing a phone number a "default region" is required to determine the processing context for non-international numbers.

A phone vector stores the raw phone number, the default region and a java Phonenumber object for each element. The java object is cached so should persist between R sessions. In case of issues, use phone_reparse() to recreate the phone vector from the original phone number and region.

Phone number parsing functions display a progress bar in interactive sessions by default. This can be disabled globally by setting option dialr.show_progress to FALSE, or locally using the show_progress function argument.

libphonenumber reference

phone(): Phone numbers are parsed using PhoneNumberUtil.parseAndKeepRawInput(). A phone vector stores the returned Phonenumber.PhoneNumber object alongside the original raw text and default region for later reference.

format(): PhoneNumberUtil.format() by default, or PhoneNumberUtil.formatOutOfCountryCallingNumber() if home is provided.

See Also

Other phone functions: dialr-example, dialr-match, dialr-region, dialr-type, dialr-valid, dialr

Examples

```
# Create a phone vector
x <- phone(c(0, 0123, "0412 345 678", "61412987654", "03 9123 4567", "+12015550123"), "AU")

is.phone(x)
print(x)
as.character(x)
format(x)
format(x, home = "AU")

# Parse international number with no default region
phone("+61412345678", NA)

# Will fail to parse if number is not in international format
phone("0412345678", NA)
```

```
# A combination can be used
phone(c("+61412345678", "0412345678"), c(NA, "AU"))
```

`dialr-region` *Phone number region*

Description

In libphonenumber a phone number region is represented by a 2 digit ISO country code. `get_region(x)` returns the 2-digit ISO country code for each element of a phone vector.

Use `get_supported_regions()` to see a full list of supported regions.

Region can also be retrieved from an international calling code. `get_region_for_calling_code(x)` returns the main region for each provided calling code. Since multiple regions can share a single calling code, `get_regions_for_calling_code(x)` returns a list of character vectors of regions for each.

Usage

```
get_region(x)

get_supported_regions()

get_region_for_calling_code(x)

get_regions_for_calling_code(x)
```

Arguments

x	A phone vector, or a vector of calling codes.

Value

A character vector of country codes.

`get_regions_for_calling_code()` returns a list of character vectors for each provided calling code.

libphonenumber reference

`get_region()`: `PhoneNumberUtil.getRegionCodeForNumber()`

`get_supported_regions()`: `PhoneNumberUtil.getSupportedRegions()`

`get_region_for_calling_code()`: `PhoneNumberUtil.getRegionCodeForCountryCode()`

`get_regions_for_calling_code()`: `PhoneNumberUtil.getRegionCodesForCountryCode()`

See Also

Other phone functions: `dialr-example`, `dialr-match`, `dialr-phone`, `dialr-type`, `dialr-valid`, `dialr`

Examples

```
# Get regions for a phone vector
x <- phone(c(0, 0123, "0412 345 678", "61412987654", "03 9123 4567", "+12015550123"), "AU")
get_region(x)

# All supported region codes
get_supported_regions()

# Primary region for a calling code
get_region_for_calling_code(c(1, 61, 84))

# All regions for a calling code
get_regions_for_calling_code(c(1, 61, 84))
```

`dialr-type` *Phone number type*

Description

In addition to validity, libphonenumber can identify phone number type - it is able to distinguish Fixed-line, Mobile, Toll-free, Premium Rate, Shared Cost, VoIP, Personal Numbers, UAN, Pager, and Voicemail (whenever feasible).

`get_type(x)` returns the phone number type for each element of a phone vector.

Valid phone number types differ by region. `get_types_for_region(x)` returns a list of character vectors of valid types for each provided ISO country code. Use `get_supported_types()` to see a full list of supported types.

Usage

```
get_type(x, strict = FALSE)

get_supported_types()

get_types_for_region(x)
```

Arguments

`x`	A phone vector, or a character vector of ISO country codes.
`strict`	If `TRUE`, invalid phone numbers return `NA`.

Value

A character vector of phone types.

`get_types_for_region()` returns a list of character vectors for each provided country code.

libphonenumber reference

`get_type()`: `PhoneNumberUtil.getNumberType()`

`get_supported_types()`: `PhoneNumberUtil.PhoneNumberType`

`get_types_for_region()`: `PhoneNumberUtil.getSupportedTypesForRegion()`

See Also

Other phone functions: `dialr-example`, `dialr-match`, `dialr-phone`, `dialr-region`, `dialr-valid`, `dialr`

Examples

```
# Get phone types for a phone vector
x <- phone(c(0, 0123, "0412 345 678", "61412987654", "03 9123 4567", "+12015550123"), "AU")
get_type(x)

# All supported phone types
get_supported_types()

# Get supported types for specified regions
get_types_for_region("AU")
get_types_for_region(c("GB", "US"))
get_types_for_region(get_supported_regions())[1:5]
```

`dialr-valid` *Phone number validity checks*

Description

For each element of `x`:

- `is_parsed(x)`: Was this successfully parsed?
- `is_valid(x)`: Is this a valid phone number?
- `is_possible(x)`: Is this a possible phone number? Return type depends on `detailed`.

Usage

```
is_parsed(x)

is_valid(x)

is_possible(x, detailed = FALSE, type = NULL)
```

Arguments

x	A phone vector.
detailed	If FALSE, is_possible returns a logical vector. If TRUE, it returns a character vector with "IS_POSSIBLE" or the reason for failure. See Details section for possible return values.
type	If provided, checks if x is possible for the given phone number type.

Details

Possible return values for `is_possible(x, detailed = TRUE)`:

- "INVALID_COUNTRY_CODE": The number has an invalid country calling code.
- "INVALID_LENGTH": The number is longer than the shortest valid numbers for this region, shorter than the longest valid numbers for this region, and does not itself have a number length that matches valid numbers for this region.
- "IS_POSSIBLE": The number length matches that of valid numbers for this region.
- "IS_POSSIBLE_LOCAL_ONLY": The number length matches that of local numbers for this region only (i.e. numbers that may be able to be dialled within an area, but do not have all the information to be dialled from anywhere inside or outside the country).
- "TOO_LONG": The number is longer than all valid numbers for this region.
- "TOO_SHORT": The number is shorter than all valid numbers for this region.

libphonenumber reference

`is_valid()`: `PhoneNumberUtil.isValidNumber()`

`is_possible()`: `PhoneNumberUtil.isPossibleNumber()`

`is_possible(detailed = TRUE)`: `PhoneNumberUtil.isPossibleNumberWithReason()`

`is_possible(type = type)`: `PhoneNumberUtil.isPossibleNumberForType()`

`is_possible(detailed = TRUE, type = type)`: `PhoneNumberUtil.sPossibleNumberForTypeWthReason()`

See Also

Other phone functions: dialr-example, dialr-match, dialr-phone, dialr-region, dialr-type, dialr

Examples

```
x <- phone(c(0, 0123, "0412 345 678", "61412987654", "03 9123 4567", "+12015550123"), "AU")

is_parsed(x)
is_valid(x)

is_possible(x)
is_possible(x, detailed = TRUE)

is_possible(x, type = "MOBILE")
is_possible(x, detailed = TRUE, type = "MOBILE")
```

get_carrier *Phone number carrier information*

Description

Returns a carrier name for each phone number, in the language provided in `locale`.

Usage

```
get_carrier(
  x,
  strict = FALSE,
  safe = FALSE,
  locale = getOption("dialr.locale")
)
```

Arguments

`x`	A phone vector.
`strict`	Should invalid phone numbers be removed? If `TRUE`, invalid phone numbers are replaced with `NA`.
`safe`	If `TRUE`, gets the name of the carrier for a given phone number only when it is 'safe' to display to users. A carrier name is considered safe if the number is valid and for a region that doesn't support mobile number portability. All other phone numbers return `""`.
`locale`	The Java locale used to retrieve localised results. The default is set in option `dialr.locale`.

Details

The carrier name is the one the number was originally allocated to, however if the country supports mobile number portability the number might not belong to the returned carrier anymore. If no mapping is found `""` is returned.

Value

A carrier name for each phone number for the given locale, or `""` if the number is invalid.

libphonenumber reference

`get_geocode()`: `PhoneNumberToCarrierMapper.getNameForValidNumber()` by default, or `PhoneNumberToCarrierMapp`
if `safe = TRUE`.

Examples

```
x <- phone(c(0, 0123, "0412 345 678", "61412987654", "03 9123 4567", "+12015550123"), "AU")
get_carrier(x)
get_carrier(x, strict = TRUE)
get_carrier(x, safe = TRUE)
```

---

get_cc *Get ISO country code*

Description

Get ISO country code from a country name.

Usage

```
get_cc(country)
```

Arguments

country	A character vector of country names.

Value

A vector of ISO country codes (NA where not found).

Examples

```
get_cc("Australia")
get_cc(c("Australia", "China", "United states"))
```

---

get_geocode *Phone number geographical information*

Description

Returns a text description for each phone number, in the language provided in locale.

Usage

```
get_geocode(x, home = NULL, strict = FALSE, locale = getOption("dialr.locale"))
```

Arguments

x	A phone vector.
home	ISO country code for home region. See Details.
strict	Should invalid phone numbers be removed? If TRUE, invalid phone numbers are replaced with NA.
locale	The Java locale used to retrieve localised results. The default is set in option dialr.locale.

Details

The description might consist of the name of the country where the phone number is from, or the name of the geographical area the phone number is from if more detailed information is available.

If a phone number is from the region specified in home, only a lower-level description will be returned, if one exists. Otherwise, the phone number's region will be returned, with optionally some more detailed information.

For example, for a user from the region "US" (United States), we would show "Mountain View, CA" for a particular number, omitting the United States from the description. For a user from the United Kingdom (region "GB"), for the same number we may show "Mountain View, CA, United States" or even just "United States".

Value

A text description for each phone number for the given locale, or "" if the number is invalid or could belong to multiple countries.

libphonenumber reference

get_geocode(): PhoneNumberOfflineGeocoder.getDescriptionForValidNumber().

Examples

```
x <- phone(c(0, 0123, "0412 345 678", "61412987654", "03 9123 4567", "+12015550123"), "AU")
get_geocode(x)
get_geocode(x, strict = TRUE)

# Specify a home country
get_geocode(x, home = "AU")
get_geocode(x, home = "US")

# Specify a language
get_geocode(x, home = "DE", locale = "de")
```

`get_timezone` *Phone number time zone*

Description

Retrieve a list of CLDR time zones to which a phone number belongs.

Usage

```
get_timezone(
  x,
  strict = FALSE,
  show_progress = getOption("dialr.show_progress")
)
```

Arguments

`x`	A phone vector.
`strict`	Should invalid phone numbers be removed? If `TRUE`, invalid phone numbers are replaced with `NA`.
`show_progress`	Should a progress bar be displayed? Defaults to the option `dialr.show_progress`.

Details

This function assumes the phone number is geo-localizable. Fixed-line and mobile numbers are considered possible candidates for geo-localization.

Value

A character vector of time zones to which each phone number belongs, separated by `;`, or the default unknown time zone `"Etc/Unknown"` if no other time zone was found.

libphonenumber reference

`get_timezone()`: `PhoneNumberToTimeZonesMapper.getTimeZonesForGeographicalNumber()`.

Examples

```
x <- phone(c(0, 0123, "0412 345 678", "61412987654", "03 9123 4567", "+12015550123"), "AU")
get_timezone(x)
get_timezone(x, strict = TRUE)

# Return a list
strsplit(get_timezone(x), ";")
```

Index

∗ **phone functions**
    dialr-example, 3
    dialr-match, 4
    dialr-phone, 5
    dialr-region, 8
    dialr-type, 9
    dialr-valid, 10

as.character.phone *(*dialr-phone*)*, 5

check_cc, 2

dialr, *3*, *5*, *7, 8*, *10, 11*
dialr-example, 3
dialr-match, 4
dialr-phone, 5
dialr-region, 8
dialr-type, 9
dialr-valid, 10

format.phone *(*dialr-phone*)*, 5

get_carrier, 12
get_cc, 13
get_example *(*dialr-example*)*, 3
get_geocode, 13
get_region *(*dialr-region*)*, 8
get_region_for_calling_code
        *(*dialr-region*)*, 8
get_regions_for_calling_code
        *(*dialr-region*)*, 8
get_supported_regions *(*dialr-region*)*, 8
get_supported_regions(), *3*
get_supported_types *(*dialr-type*)*, 9
get_timezone, 15
get_type *(*dialr-type*)*, 9
get_types_for_region *(*dialr-type*)*, 9
get_types_for_region(), *3*

is.phone *(*dialr-phone*)*, 5
is_match *(*dialr-match*)*, 4
is_parsed *(*dialr-valid*)*, 10
is_possible *(*dialr-valid*)*, 10
is_valid *(*dialr-valid*)*, 10
ISO country code, *2*, *6*, *9*, *13, 14*
ISO country codes, *2, 3*, *6*, *9*, *13*

phone, *3, 4*, *8, 9*, *11, 12*, *14, 15*
phone *(*dialr-phone*)*, 5
phone number type, *11*
phone number types, *3*
phone_reparse *(*dialr-phone*)*, 5
print.phone *(*dialr-phone*)*, 5

region, *3*

type, *3*