Package: pslr (via r-universe)

November 18, 2026

Type Package

Title Public Suffix List Engine

Version 1.2.2.9000

Description A focused implementation of the Public Suffix List (PSL). Bundles a reproducible, pinned PSL snapshot and implements the official prevailing-rule algorithm to answer public-suffix (eTLD) and registrable-domain (eTLD+1) queries. Distinguishes ICANN and PRIVATE rule sections, accepts Unicode and ASCII hostnames via 'punycoder' canonicalization, and supports an explicit, validated offline refresh path. The matcher is compiled with 'cpp11' and requires no external system library. Used as the PSL engine by the 'rurl' package.

License MIT + file LICENSE

Language en-US

URL https://bart-turczynski.gitlab.io/pslr/, https://gitlab.com/bart-turczynski/pslr, https://bart-turczynski.r-universe.dev/pslr, https://CRAN.R-project.org/package=pslr

BugReports https://gitlab.com/bart-turczynski/pslr/-/work_items

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports digest, punycoder (>= 1.3.0), tools, utils

LinkingTo cpp11

Suggests curl, knitr, oysteR, rmarkdown, rosv, testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

X-schema.org-keywords public suffix list, PSL, eTLD, eTLD+1, registrable domain, ICANN, domain parsing, domain extraction, domain names, TLD, cookie domains, IDNA, public suffix

Repository https://bart-turczynski.r-universe.dev

Date/Publication 2026-11-18 15:00:27 UTC

RemoteUrl https://gitlab.com/bart-turczynski/pslr

RemoteRef HEAD

RemoteSha 693f1535b006e603f84e3177326a305f60ef82c5

Contents

is_public_suffix 2
psl_cache_prune 3
psl_diff 5
psl_refresh 7
psl_rules 9
psl_use 9
psl_version 10
public_suffix 11
public_suffix_rule 13
registrable_domain 14
suffix_extract 16

Index **18**

`is_public_suffix` *Is a host itself a public suffix?*

Description

`TRUE` exactly when the valid canonical host equals its own public suffix under the selected policy. Returns `NA` whenever `public_suffix()` would return `NA` (missing or invalid input, or an unresolved host under `unknown = "na"`). Under the default `unknown = "default"`, an unlisted single label such as `"madeuptld"` is `TRUE` via the implicit `*` rule; ask `unknown = "na"` to test explicit membership instead.

Usage

```
is_public_suffix(
  domain,
  section = "all",
  unknown = "default",
  invalid = "na",
  engine = psl_default_engine()
)
```

Arguments

domain	Character vector of DNS hostnames (not URLs). Each element may be a mixed-case ASCII, Unicode, or A-label hostname, a single label, or a hostname with exactly one terminal root dot. See **Input contract**.
section	Which rule sections are eligible: "all" (default; ICANN and PRIVATE), "icann", or "private". Section filtering happens before prevailing-rule selection, so "private" does not silently add ICANN rules; a host matching no rule in the section falls through to the implicit default rule unless unknown = "na".
unknown	"default" (default) applies the spec's implicit * rule, so an unlisted single label is its own public suffix; "na" returns NA when no explicit rule in the selected section matches.
invalid	"na" (default) returns NA for each invalid element without a warning; "error" aborts on the first invalid element, reporting its 1-based index.
engine	The psl_engine to query against; defaults to the session-global engine selected by psl_use(), so most callers never set it. Pass an engine from psl_engine() to resolve hosts against a specific snapshot in isolation.

Value

A logical vector with length(domain), preserving the names of domain.

Input contract

NA is treated as missing (returns NA), not invalid. Invalid elements include empty or whitespace-only strings, leading or consecutive dots, URL syntax, IPv6 addresses, canonical dotted-decimal IPv4 literals, and labels that fail hostname/IDNA validation. Wrong argument types and non-scalar or unknown option values always abort regardless of invalid.

See Also

public_suffix()

Examples

```
is_public_suffix("com")
is_public_suffix("example.com")
is_public_suffix("madeuptld")
is_public_suffix("madeuptld", unknown = "na")
```

---

psl_cache_prune *Prune unreferenced local Public Suffix List snapshots*

Description

Reclaims cached snapshots that nothing points at any more. The call is explicit, strictly offline, and destructive by design: it deletes snapshot files. It never changes which list is active, and every reference that could still be followed survives it.

Usage

```
psl_cache_prune(keep = 1L)
```

Arguments

keep	Number of unreferenced snapshots to retain *in addition to* every referenced one, as a single non-negative whole number. The default `1` keeps one snapshot of history beyond what is still referenced; `0` keeps only referenced snapshots.

Details

A snapshot is protected, and therefore never removed, when it is any of:

- the selected cache snapshot, the one `psl_use()` resolves with `source = "cache"`;
- named by any source's stored state, across every source this cache knows, not only the official endpoint;
- active in the calling R session; or
- among the `keep` most recently first-retrieved snapshots that none of the above already protects.

Retention is ordered by when a snapshot's bytes were *first retrieved*, as recorded in its descriptor when it was published, rather than by file modification time. A copy, a restore from backup, or any tool that rewrites timestamps changes mtime but not the fact of when the bytes were received, so first retrieval is the only stable order for history.

Pruning runs under the cache's publication lock and removes only complete `.dat`/`.rds` snapshot pairs, so it can never leave a reference pointing at bytes that are gone. Snapshot state pslr cannot fully read is treated as a reference it cannot see: if any source or selection stream is damaged, nothing is collected at all, and the damage is left visible rather than quietly erased. `psl_cache_prune()` is an explicit mutator, so it also performs the one-time migration of a pre-v2 cache before pruning; it never deletes legacy files, and it never reads or writes the reminder preference, which is configuration and lives outside the cache.

One honest limitation: pruning cannot discover engines held by *other* R processes, which already hold their parsed rules in memory. It therefore guarantees persistent referential integrity – no stored reference is ever left dangling – and not continued on-disk availability for another process's detached engine descriptor.

This is distinct from `psl_cache_clear()`, which only flushes this session's in-memory match-result cache and deletes nothing from disk.

Value

Invisibly, a data.frame with one row per removed snapshot and the columns `checksum` (character, the `"sha256:<hex>"` identity), `bytes_path` and `descriptor_path` (character paths removed, `NA` when that half was already absent), and `bytes_reclaimed` (numeric). The frame has zero rows when nothing was pruned, including when there is no cache at all. A prune that removes only part of what it selected signals an error of class `pslr_prune_partial_error` whose `result` field is this same frame, describing what was removed before the failure.

See Also

psl_snapshots(), which lists what a prune would consider; psl_refresh(), which publishes the snapshots this reclaims; psl_use().

Examples

```
# Deleting cached snapshots is never a side effect of running an example:
if (interactive()) {
  psl_cache_prune() # keep referenced snapshots plus one more
  psl_cache_prune(keep = 0) # keep only referenced snapshots
}
```

---

psl_diff *Compare two Public Suffix List snapshots*

Description

Reports which rules were added, removed, or changed between two Public Suffix List snapshots that are already available locally. The comparison is canonical rather than textual: both sides are parsed and canonicalized first, so comments, blank lines, whitespace, letter case, source ordering, and raw Unicode spelling are never reported as changes.

Usage

```
psl_diff(old, new, ...)
```

Arguments

old, new	The two snapshots to compare. Each accepts, independently, one of: "bundled" (the snapshot installed with the package), "cache" (the snapshot psl_use() would resolve, i.e. the current cache selection), a path to a local PSL source file, a psl_engine from psl_engine(), or a rule table with the psl_rules() schema. "bundled" and "cache" are reserved names, so a file with either of those names must be given as a path carrying a directory component.
...	These dots are for future extension and must be empty.

Details

The call is strictly offline. No input form triggers a request, and neither side changes the session-global list that psl_use() controls, so diffing the cache leaves the next query answering from exactly the list it did before.

Value

A base data.frame with one row per changed logical identity, ordered by that identity, and columns, in order: `change` (`"added"`, `"removed"`, or `"changed"`), `rule` (the logical identity), `old_rule` and `new_rule` (canonical rule text including any `*.` or `!` marker), `old_kind` and `new_kind` (`"normal"`, `"wildcard"`, or `"exception"`), and `old_section` and `new_section` (`"icann"` or `"private"`). Every field belonging to a side the identity is absent from is `NA`. Two snapshots that agree return the same columns and types with zero rows.

Provenance is attached as the attributes `old_version` and `new_version`, each a one-row `psl_version()` frame describing that side's snapshot, or `NULL` when the input was a rule table and therefore carries no provenance.

Which snapshots are available

The upstream endpoint publishes only the current list, and the list itself asks callers not to depend on version-control URLs, so `psl_diff()` resolves no dates and downloads no history. Installing the package performs no request and provides only the bundled snapshot; local collection begins at the first explicit `psl_refresh()`, after which each distinct validated download is retained as an immutable file until `psl_cache_prune()` collects it. `psl_snapshots()` lists what is resolvable, with a `path` for each set of stored bytes; any of those paths, or any historical revision the caller materializes as a file by other means, can be passed here directly.

Change semantics

A rule's *logical identity* is its canonical labels with the leading `*.` or `!` marker removed, so `*.example.com` and `example.com` are the same identity in two different states. `change` is:

`added` The identity exists only in `new`.

`removed` The identity exists only in `old`.

`changed` The identity exists in both, but its canonical rule kind or its ICANN/PRIVATE section differs.

Identities that exist in both snapshots in the same state are not reported. Section membership is a compared attribute rather than part of the identity, so a rule moving between the ICANN and PRIVATE sections is one `changed` row rather than an unrelated `removed`/`added` pair. A list may legally carry the same labels once in each section; such rules collapse into the single row for their identity, with the values of that side joined in ICANN-then-PRIVATE order.

See Also

`psl_snapshots()`, `psl_refresh()`, `psl_rules()`, `psl_version()`

Examples

```
# Two small lists on disk, written offline.
write_list <- function(private) {
  path <- tempfile(fileext = ".dat")
  writeLines(
    c(
      "// ===BEGIN ICANN DOMAINS===",
```

```
      "com",
      "// ===END ICANN DOMAINS===",
      "// ===BEGIN PRIVATE DOMAINS===",
      private,
      "// ===END PRIVATE DOMAINS==="
    ),
    path
  )
  path
}
old <- write_list(c("a.example.com", "*.b.example.com"))
new <- write_list(c("b.example.com", "c.example.com"))

# One row each: a.example.com removed, b.example.com changed kind,
# c.example.com added.
psl_diff(old, new)

# Agreeing snapshots return the same columns with zero rows.
psl_diff("bundled", "bundled")

# Provenance travels with the result when the input form carries it.
attr(psl_diff("bundled", old), "old_version")$checksum
```

`psl_refresh` *Refresh the cached Public Suffix List from upstream*

Description

Revalidates the Public Suffix List against its source and publishes any changed bytes into the user cache. This is the only function in the package that accesses the network, and only when you call it explicitly.

Usage

```
psl_refresh(
  url = "https://publicsuffix.org/list/public_suffix_list.dat",
  ...,
  activate = FALSE,
  force = FALSE
)
```

Arguments

`url`	Absolute `https` URL of the list source. Defaults to the official list. URLs with another scheme, embedded credentials, a query string, or a fragment are rejected, and a redirect that leaves `https` or the original origin is refused.
`...`	These dots are for future extensions and must be empty. They also make `activate` and `force` named-only. The two flags are easy to confuse – both logical, both

about doing more than a bare check – so `psl_refresh(url, TRUE)` is unreadable whichever order it means. Naming them makes every call self-documenting, and the old positional form is now a clear error instead of a silent change of meaning.

`activate` When `TRUE`, the snapshot the source now points at becomes the active list for the session, exactly as `psl_use()` would activate it – after *every* successful outcome, including a skip and a `304`. When `FALSE` (default), the cache is updated but the active list is unchanged.

`force` When `FALSE` (default), a check whose courtesy window has not elapsed makes no request at all, respecting upstream download guidance. `TRUE` bypasses that local window only; the request it then makes still carries a validator when one is available.

Details

A refresh has exactly four successful outcomes:

`skipped_recently` No request: the last successful check is still inside its courtesy window (at least 24 hours).

`not_modified` One conditional request answered `304`; the local bytes were verified and remain current.

`downloaded_unchanged` One `200` whose validated bytes hash to the snapshot already held, so no new snapshot is created.

`updated` One `200` whose validated bytes are a new snapshot.

Downloaded bytes are fully validated – size ceiling, UTF-8, official section markers, rule grammar, and canonicalization of every rule – before anything references them, and exact same-section duplicates warn once and are deduplicated. Publication is append-only: snapshot bytes and their descriptor are written before any reference to them, so an interrupted refresh never exposes a dangling reference. Refreshes of one source are serialized across processes; a second concurrent refresh of the same source makes no request and signals a busy error. A failed refresh records only a coarse attempt and leaves the cache, the selected snapshot, and the active matcher untouched.

Value

Invisibly, a `psl_refresh_result` with stable fields `outcome`, `request_url`, `effective_url`, `http_status`, `checked_at`, `previous_checksum`, `checksum`, `activated`, `validator`, `bytes_downloaded`, and `snapshot`. Operational failures signal a classed error rooted at `pslr_refresh_error` instead.

See Also

`psl_use()`, `psl_version()`

Examples

```
if (interactive()) {
  psl_refresh()
  psl_refresh(force = TRUE, activate = TRUE)
}
```

psl_rules *Rules of the active Public Suffix List*

Description

Returns the explicit rules of the active list as a base data.frame, one row per rule. The implicit default * rule is not included.

Usage

```
psl_rules(section = "all")
```

Arguments

section	Which rule sections to return: "all" (default), "icann", or "private".

Value

A base data.frame with columns, in order: rule (original source rule text), canonical_rule (the canonicalized rule, including the *. or ! marker), kind ("normal", "wildcard", or "exception"), section ("icann" or "private"), and labels (integer rule depth, counting a wildcard label). Rows are ordered first by section (ICANN before PRIVATE) and then by source-file order.

See Also

psl_version(), public_suffix_rule(), psl_diff()

Examples

```
head(psl_rules("icann"))
nrow(psl_rules("private"))
```

psl_use *Choose the active Public Suffix List for this session*

Description

Switches the list backing every query in the current R session. The change is session-only and is validated before any session state changes; a failure leaves the previously active list usable. A successful switch invalidates the match-result cache.

Usage

```
psl_use(source = "bundled", path = NULL)
```

Arguments

source	Where to load the list from: "bundled" (the pinned package snapshot), "cache" (the latest successfully validated snapshot from psl_refresh()), or "path" (a custom file).
path	For source = "path", a single readable PSL-format UTF-8 file containing one complete ICANN section and one complete PRIVATE section, using official markers. Must be NULL for any other source.

Details

A custom path is held to the same runtime duplicate policy as psl_refresh(): exact same-section duplicates warn once and are deduplicated, while conflicting rule kinds for the same labels are fatal. Cache and custom-path sources are read in source form and indexed under the runtime normalizer; they never reuse the bundled generated index.

Value

Invisibly, the psl_version() row for the newly active list.

See Also

psl_refresh(), psl_version(), psl_rules()

Examples

```
psl_use("bundled")
if (interactive()) {
  psl_use("cache")
  psl_use("path", path = "my_list.dat")
}
```

---

psl_version *Identity of the active Public Suffix List*

Description

Returns a one-row data.frame describing the list currently active in this R session: its source-snapshot provenance and the normalization identifiers actually used to index the active matcher. Reproducing a query result requires both the active-list identity and these normalization identifiers (PRD s10), so a reproducibility-sensitive workflow should record this row.

Usage

```
psl_version()
```

Details

The columns, in order, are:

`source` `"bundled"`, `"cache"`, or `"path"`.

`url` Source URL of the active snapshot: the upstream download URL for the bundled list; `NA` for a `"cache"` or `"path"` source.

`path` File path of a `"cache"` or `"path"` source; `NA` otherwise.

`retrieved_at` Network retrieval timestamp, or `NA`.

`list_date` Upstream list date, or `NA` when unknown.

`commit` Upstream commit SHA, or `NA` when unknown.

`size` Source byte size (integer).

`checksum` Source checksum, including its algorithm prefix (e.g. `"sha256:..."`).

`normalizer` The dependency providing canonicalization, currently `"punycoder"`.

`normalizer_version` Its installed package version.

`normalization_profile` Its stable case-mapping / IDNA / validation profile identifier.

`unicode_version` The Unicode data version used by that profile.

Unavailable metadata is a typed `NA`, never omitted. The normalization identifiers describe the implementation used by the current session, whether the active list came from the bundled snapshot, the user cache, or a custom path; an in-memory compatibility rebuild (PRD s8.3) updates them without altering the shipped source identity or checksum.

Value

A one-row base data.frame with the columns described in Details.

See Also

`psl_use()`, `psl_refresh()`, `psl_rules()`

Examples

```
psl_version()
```

`public_suffix` *Public suffix of a host*

Description

Returns the public suffix (effective top-level domain, eTLD) of each host under the selected Public Suffix List policy, following the official prevailing-rule algorithm.

Usage

```
public_suffix(
  domain,
  section = "all",
  output = "ascii",
  unknown = "default",
  invalid = "na",
  engine = psl_default_engine()
)
```

Arguments

`domain`	Character vector of DNS hostnames (not URLs). Each element may be a mixed-case ASCII, Unicode, or A-label hostname, a single label, or a hostname with exactly one terminal root dot. See **Input contract**.
`section`	Which rule sections are eligible: `"all"` (default; ICANN and PRIVATE), `"icann"`, or `"private"`. Section filtering happens before prevailing-rule selection, so `"private"` does not silently add ICANN rules; a host matching no rule in the section falls through to the implicit default rule unless `unknown = "na"`.
`output`	`"ascii"` (default) returns lowercase A-labels; `"unicode"` decodes them after matching. A terminal root dot is preserved either way.
`unknown`	`"default"` (default) applies the spec's implicit `*` rule, so an unlisted single label is its own public suffix; `"na"` returns `NA` when no explicit rule in the selected section matches.
`invalid`	`"na"` (default) returns `NA` for each invalid element without a warning; `"error"` aborts on the first invalid element, reporting its 1-based index.
`engine`	The `psl_engine` to query against; defaults to the session-global engine selected by `psl_use()`, so most callers never set it. Pass an engine from `psl_engine()` to resolve hosts against a specific snapshot in isolation.

Value

A character vector with `length(domain)`, preserving the names of `domain`. Other attributes are dropped.

Input contract

`NA` is treated as missing (returns `NA`), not invalid. Invalid elements include empty or whitespace-only strings, leading or consecutive dots, URL syntax, IPv6 addresses, canonical dotted-decimal IPv4 literals, and labels that fail hostname/IDNA validation. Wrong argument types and non-scalar or unknown option values always abort regardless of `invalid`.

See Also

`registrable_domain()`, `is_public_suffix()`, `suffix_extract()`, `public_suffix_rule()`

Examples

```
public_suffix("www.example.com")
public_suffix("example.co.uk")
public_suffix("example.com.")
public_suffix("madeuptld", unknown = "na")
```

`public_suffix_rule` *Inspect the prevailing PSL rule for each host*

Description

Inspect the prevailing PSL rule for each host

Usage

```
public_suffix_rule(
  domain,
  section = "all",
  unknown = "default",
  invalid = "na",
  engine = psl_default_engine()
)
```

Arguments

`domain`	Character vector of DNS hostnames (not URLs). Each element may be a mixed-case ASCII, Unicode, or A-label hostname, a single label, or a hostname with exactly one terminal root dot. See **Input contract**.
`section`	Which rule sections are eligible: `"all"` (default; ICANN and PRIVATE), `"icann"`, or `"private"`. Section filtering happens before prevailing-rule selection, so `"private"` does not silently add ICANN rules; a host matching no rule in the section falls through to the implicit default rule unless `unknown = "na"`.
`unknown`	`"default"` (default) applies the spec's implicit `*` rule, so an unlisted single label is its own public suffix; `"na"` returns `NA` when no explicit rule in the selected section matches.
`invalid`	`"na"` (default) returns `NA` for each invalid element without a warning; `"error"` aborts on the first invalid element, reporting its 1-based index.
`engine`	The `psl_engine` to query against; defaults to the session-global engine selected by `psl_use()`, so most callers never set it. Pass an engine from `psl_engine()` to resolve hosts against a specific snapshot in isolation.

Value

A base data.frame with one row per input and columns, in order: `input` (original), `host_ascii` (canonical A-label host), `rule` (the canonical rule including `*.` or `!`, `"*"` for the implicit default), `kind` (`"normal"`, `"wildcard"`, `"exception"`, or `"default"`), `rule_section` (`"icann"`, `"private"`, or `NA` for the default/no result), and `public_suffix_ascii` (the derived A-label public suffix). Invalid rows are `NA` in every derived column. A valid host left unresolved by `unknown = "na"` keeps `host_ascii` while the rule and suffix columns are `NA`. An exception `rule` retains its `!` for auditability. Zero-length input returns a zero-row frame; all-invalid input keeps one row per input.

Input contract

`NA` is treated as missing (returns `NA`), not invalid. Invalid elements include empty or whitespace-only strings, leading or consecutive dots, URL syntax, IPv6 addresses, canonical dotted-decimal IPv4 literals, and labels that fail hostname/IDNA validation. Wrong argument types and non-scalar or unknown option values always abort regardless of `invalid`.

See Also

`public_suffix()`, `suffix_extract()`

Examples

```
public_suffix_rule("www.example.co.uk")
public_suffix_rule("madeuptld")
```

`registrable_domain` *Registrable domain of a host*

Description

Returns the registrable domain (eTLD+1) of each host: its public suffix plus one host label to the left. It is `NA` when no such label exists (the host is itself a public suffix) or when the public suffix is `NA`.

Usage

```
registrable_domain(
  domain,
  section = "all",
  output = "ascii",
  unknown = "default",
  invalid = "na",
  engine = psl_default_engine()
)
```

Arguments

domain	Character vector of DNS hostnames (not URLs). Each element may be a mixed-case ASCII, Unicode, or A-label hostname, a single label, or a hostname with exactly one terminal root dot. See **Input contract**.
section	Which rule sections are eligible: "all" (default; ICANN and PRIVATE), "icann", or "private". Section filtering happens before prevailing-rule selection, so "private" does not silently add ICANN rules; a host matching no rule in the section falls through to the implicit default rule unless unknown = "na".
output	"ascii" (default) returns lowercase A-labels; "unicode" decodes them after matching. A terminal root dot is preserved either way.
unknown	"default" (default) applies the spec's implicit * rule, so an unlisted single label is its own public suffix; "na" returns NA when no explicit rule in the selected section matches.
invalid	"na" (default) returns NA for each invalid element without a warning; "error" aborts on the first invalid element, reporting its 1-based index.
engine	The psl_engine to query against; defaults to the session-global engine selected by psl_use(), so most callers never set it. Pass an engine from psl_engine() to resolve hosts against a specific snapshot in isolation.

Value

A character vector with length(domain), preserving the names of domain. Other attributes are dropped.

Input contract

NA is treated as missing (returns NA), not invalid. Invalid elements include empty or whitespace-only strings, leading or consecutive dots, URL syntax, IPv6 addresses, canonical dotted-decimal IPv4 literals, and labels that fail hostname/IDNA validation. Wrong argument types and non-scalar or unknown option values always abort regardless of invalid.

See Also

public_suffix(), is_public_suffix(), suffix_extract()

Examples

```
registrable_domain("www.example.co.uk")
registrable_domain("com")
registrable_domain("foo.madeuptld", unknown = "na")
```

`suffix_extract` *Split hosts into subdomain, registrant label, and public suffix*

Description

Split hosts into subdomain, registrant label, and public suffix

Usage

```
suffix_extract(
  domain,
  section = "all",
  output = "ascii",
  unknown = "default",
  invalid = "na",
  engine = psl_default_engine()
)
```

Arguments

`domain`	Character vector of DNS hostnames (not URLs). Each element may be a mixed-case ASCII, Unicode, or A-label hostname, a single label, or a hostname with exactly one terminal root dot. See **Input contract**.
`section`	Which rule sections are eligible: `"all"` (default; ICANN and PRIVATE), `"icann"`, or `"private"`. Section filtering happens before prevailing-rule selection, so `"private"` does not silently add ICANN rules; a host matching no rule in the section falls through to the implicit default rule unless `unknown = "na"`.
`output`	`"ascii"` (default) returns lowercase A-labels; `"unicode"` decodes them after matching. A terminal root dot is preserved either way.
`unknown`	`"default"` (default) applies the spec's implicit `*` rule, so an unlisted single label is its own public suffix; `"na"` returns `NA` when no explicit rule in the selected section matches.
`invalid`	`"na"` (default) returns `NA` for each invalid element without a warning; `"error"` aborts on the first invalid element, reporting its 1-based index.
`engine`	The `psl_engine` to query against; defaults to the session-global engine selected by `psl_use()`, so most callers never set it. Pass an engine from `psl_engine()` to resolve hosts against a specific snapshot in isolation.

Value

A base data.frame with one row per input and columns, in order: `input` (original, unchanged), `host` (canonical host in `output` form), `subdomain` (labels left of the registrable domain; `""` when none), `domain` (the single registrant label left of the suffix), `suffix` (the public suffix), and `registrable_domain` (eTLD+1). `domain`, `subdomain`, and `registrable_domain` are `NA` when the host is itself a public suffix. If public-suffix resolution is `NA`, every derived column except `input` and a successfully normalized `host` is `NA`. Zero-length input returns a zero-row frame; all-invalid input keeps one row per input. Root dots are preserved on `host`, `suffix`, and `registrable_domain` only.

Input contract

NA is treated as missing (returns NA), not invalid. Invalid elements include empty or whitespace-only strings, leading or consecutive dots, URL syntax, IPv6 addresses, canonical dotted-decimal IPv4 literals, and labels that fail hostname/IDNA validation. Wrong argument types and non-scalar or unknown option values always abort regardless of `invalid`.

See Also

`public_suffix()`, `public_suffix_rule()`

Examples

```
suffix_extract("www.example.co.uk")
suffix_extract(c("example.com", "com", NA))
```

Index

data.frame, *4*, *6*, *9–11*, *14*, *16*

is_public_suffix, 2
is_public_suffix(), *12*, *15*

psl_cache_prune, 3
psl_cache_prune(), *6*
psl_diff, 5
psl_diff(), *9*
psl_engine(), *5*
psl_refresh, 7
psl_refresh(), *5, 6*, *10, 11*
psl_rules, 9
psl_rules(), *5, 6*, *10, 11*
psl_snapshots(), *5, 6*
psl_use, 9
psl_use(), *4*, *5*, *8*, *11*
psl_version, 10
psl_version(), *6*, *8–10*
public_suffix, 11
public_suffix(), *2, 3*, *14, 15*, *17*
public_suffix_rule, 13
public_suffix_rule(), *9*, *12*, *17*

registrable_domain, 14
registrable_domain(), *12*

suffix_extract, 16
suffix_extract(), *12*, *14, 15*