Package: sensortowerR (via r-universe)

May 13, 2026

Title Interface to 'Sensor Tower' Mobile App Intelligence API

Version 1.0.0

Description Interface to the 'Sensor Tower' API <https://app.sensortower.com/api/docs/app_analysis> for mobile app analytics and market intelligence. Provides a small, consistent set of functions to retrieve app metadata, publisher information, download and revenue estimates, active user metrics, category rankings, and market trends. Four core verbs ('st_metrics', 'st_rankings', 'st_app'/'st_apps', 'st_filter') cover the common workflows with standardized parameters and tidyverse-friendly output. Supports both iOS and Android app ecosystems with unified data structures for cross-platform analysis.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports dplyr, glue, httr, httr2, jsonlite, lubridate, openssl, purrr, rlang, stats, stringr, tibble, tidyr, utils

Suggests gt, gtExtras, knitr, pkgbuild, rcmdcheck, rmarkdown, rhub, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Author Phillip Black [aut, cre]

Maintainer Phillip Black <pblack@gameeconomistconsulting.com>

URL https://github.com/econosopher/sensortowerR

BugReports https://github.com/econosopher/sensortowerR/issues

Depends R (>= 4.1.0)

Config/pak/sysreqs libicu-dev libssl-dev

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

Date/Publication 2026-05-13 11:16:48 UTC

RemoteUrl https://github.com/econosopher/sensortowerr

RemoteRef HEAD

RemoteSha abe3624bbdac5e9cf33efed8c95b3d3cfe290388

Contents

`calculate_yoy_growth` *Calculate year-over-year growth rates*

Description

Helper function to calculate YoY growth rates from the output of st_yoy_metrics

Usage

```
calculate_yoy_growth(yoy_data, baseline_year = NULL)
```

Arguments

`yoy_data`	Output from st_yoy_metrics
`baseline_year`	The year to use as baseline (default: earliest year)

Value

A tibble with growth rates relative to baseline year

`filter_helpers` *Custom Filter Helper Functions*

Description

Functions to validate, test, and manage Sensor Tower custom field filters

format_arpu *Format ARPU (Average Revenue Per User)*

Description

Format ARPU (Average Revenue Per User)

Usage

```
format_arpu(val, digits = 2)
```

Arguments

val	Numeric ARPU value
digits	Number of decimal places (default: 2)

Value

Formatted ARPU string

Examples

```
format_arpu(5.234)    # "$5.23"
format_arpu(0.99)     # "$0.99"
```

format_currency *Format currency values with appropriate suffixes*

Description

Format currency values with appropriate suffixes

Usage

```
format_currency(val, digits = 2)
```

Arguments

val	Numeric value to format as currency
digits	Number of digits after decimal for millions/billions (default: 2)

Value

Formatted currency string

Examples

```
format_currency(1234567)      # "$1.23M"
format_currency(1234567890)   # "$1.23B"
format_currency(123)          # "$123"
```

`format_downloads` *Format download counts with appropriate suffixes*

Description

Format download counts with appropriate suffixes

Usage

```
format_downloads(val)
```

Arguments

`val`	Numeric value to format as downloads

Value

Formatted download count string

Examples

```
format_downloads(1234567)      # "1.2M"
format_downloads(1234567890)   # "1.2B"
```

`format_large_number` *Format large numbers with K/M/B suffixes*

Description

Format large numbers with K/M/B suffixes

Usage

```
format_large_number(val, digits = 1, prefix = "")
```

Arguments

`val`	Numeric value to format
`digits`	Number of digits after decimal for millions/billions (default: 1)
`prefix`	Optional prefix (e.g., "$" for currency)

Value

Formatted string with appropriate suffix

Examples

```
format_large_number(1234567)      # "1.2M"
format_large_number(1234567890)   # "1.2B"
format_large_number(1234567, prefix = "$") # "$1.2M"
```

---

format_market_share *Format market share as percentage*

Description

Format market share as percentage

Usage

```
format_market_share(val, digits = 1)
```

Arguments

val	Numeric value as decimal (0-1 scale)
digits	Number of decimal places (default: 1)

Value

Formatted market share percentage string

Examples

```
format_market_share(0.234)    # "23.4%"
format_market_share(0.05)     # "5.0%"
```

---

format_percent *Format percentages*

Description

Format percentages

Usage

```
format_percent(val, digits = 1)
```

Arguments

val	Numeric value to format as percentage (0-100 scale)
digits	Number of decimal places (default: 1)

Value

Formatted percentage string

Examples

```
format_percent(23.456)    # "23.5%"
format_percent(0.234, digits = 2)    # "0.23%"
```

---

format_retention *Format retention rates*

Description

Format retention rates

Usage

```
format_retention(val, digits = 1)
```

Arguments

val	Numeric value as decimal (0-1 scale)
digits	Number of decimal places (default: 1)

Value

Formatted retention percentage string

Examples

```
format_retention(0.234)    # "23.4%"
format_retention(0.85)     # "85.0%"
```

format_users *Format user counts (DAU/MAU/WAU)*

Description

Format user counts (DAU/MAU/WAU)

Usage

```
format_users(val)
```

Arguments

val	Numeric value to format as user count

Value

Formatted user count string

Examples

```
format_users(1234567)      # "1.2M"
format_users(1234)         # "1.2K"
```

formatting_helpers *Formatting Helper Functions*

Description

Functions for formatting numeric values in a human-readable way, particularly useful for revenue, download counts, and percentages.

lookup_category_names *Helper function to look up category names*

Description

Helper function to look up category names

Usage

```
lookup_category_names(category_ids, platform = "ios")
```

Arguments

category_ids	Character vector of category IDs
platform	Character string. "ios" or "android"

Value

Character vector of category names

---

st_active_users *Fetch Active User Metrics with a Tidy Long Output*

Description

Lightweight wrapper around [st_batch_metrics()] focused only on active-user metrics (‘dau‘, ‘wau‘, ‘mau‘). Returns a tidy long-format tibble that is designed for straightforward tidyverse chaining.

Usage

```
st_active_users(
  os,
  app_list,
  metrics = c("dau", "wau", "mau"),
  date_range = list(start_date = Sys.Date() - 90, end_date = Sys.Date() - 1),
  countries,
  granularity = "monthly",
  parallel = FALSE,
  verbose = TRUE,
  auth_token = NULL,
  max_cores = 2
)
```

Arguments

os	Character. Required. Operating system: ‘"ios"‘, ‘"android"‘, or ‘"unified"‘.
app_list	List or data frame containing app information. Same formats accepted by [st_batch_metrics()].
metrics	Character vector. Active-user metrics to return. Must be one or more of ‘"dau"‘, ‘"wau"‘, ‘"mau"‘. Default is all three.
date_range	List with ‘start_date‘ and ‘end_date‘.
countries	Character vector of country codes.
granularity	Character granularity (‘"daily"‘, ‘"weekly"‘, ‘"monthly"‘, ‘"quarterly"‘). Default ‘"monthly"‘.
parallel	Logical. Whether to use parallel processing.
verbose	Logical. Whether to print progress messages.
auth_token	Character string. Sensor Tower API token. Defaults to ‘SENSORTOWER_AUTH_TOKEN‘ environment variable.
max_cores	Integer. Maximum cores for parallel processing.

Value

A tibble with columns including ‘original_id‘, ‘app_name‘, ‘app_id‘, ‘app_id_type‘, ‘date‘, ‘country‘, ‘metric‘, and ‘value‘.

`st_analyze_filter` *Analyze Custom Filter Performance*

Description

Provides a summary analysis of apps matching a custom filter, including top performers, growth metrics, and category breakdown.

Usage

```
st_analyze_filter(
  filter_id,
  measure = "DAU",
  regions = "US",
  top_n = 10,
  auth_token = NULL
)
```

Arguments

`filter_id`	Character. The custom fields filter ID to analyze
`measure`	Character. Metric to analyze: "DAU", "revenue", or "units"
`regions`	Character vector. Region codes (default "US")
`top_n`	Integer. Number of top apps to show (default 10)
`auth_token`	Optional. Character string. Your Sensor Tower API token.

Value

A list containing summary statistics and top apps

st_api_diagnostics *Diagnose API Issues*

Description

This function helps diagnose common API issues by testing various ID formats and endpoints to determine the best approach for fetching data.

Usage

```
st_api_diagnostics(
  app_id,
  verbose = TRUE,
  auth_token = Sys.getenv("SENSORTOWER_AUTH_TOKEN")
)
```

Arguments

app_id	Character string. The app ID to diagnose (can be unified, iOS, or Android)
verbose	Logical. Show detailed diagnostic output. Default is TRUE.
auth_token	Character string. Your Sensor Tower API authentication token.

Value

A list with diagnostic results including: - 'id_type': Detected ID type - 'platform_ids': Resolved platform-specific IDs - 'endpoint_results': Results from testing various endpoints - 'recommendations': Suggested approach for this app

Examples

```
## Not run:
# Diagnose Star Trek Fleet Command
diagnosis <- st_api_diagnostics("5ba4585f539ce75b97db6bcb")

# Check iOS app
diagnosis <- st_api_diagnostics("1427744264")

## End(Not run)
```

st_app *Fetch App Details*

Description

Thin v1.0.0 facade for app metadata lookups. By default it dispatches to the legacy detailed app metadata endpoint. When ‘fields‘ is supplied, it dispatches to ‘st_app_enriched()‘ and optionally subsets to the requested fields.

Usage

```
st_app(app_id, os = "unified", fields = NULL, auth_token = NULL)
```

Arguments

app_id	Character scalar or vector of app identifiers.
os	Operating system context. One of ‘"ios"‘, ‘"android"‘, or ‘"unified"‘.
fields	Optional character vector of enriched fields to keep. When supplied, the request is routed through ‘st_app_enriched()‘.
auth_token	Optional Sensor Tower API token.

Value

A tibble returned by the dispatched legacy implementation.

Examples

```
## Not run:
st_app("553834731", os = "ios")

st_app(
  c("553834731", "com.supercell.clashofclans"),
  fields = c("revenue_90d_ww", "downloads_30d_ww")
)

## End(Not run)
```

st_app_enriched *Fetch Enriched Metrics for Specific Apps*

Description

Retrieves comprehensive metrics including retention, MAU, DAU, demographics, and other aggregate tags for specific apps by their unified app IDs.

Usage

```
st_app_enriched(
  unified_app_ids,
  os = "unified",
  regions = "WW",
  auth_token = NULL
)
```

Arguments

unified_app_ids	Character vector. One or more unified app IDs (24-character hex strings). Required. Use ‘st_app_info()‘ to find these.
os	Character string. Operating system context for the request. Must be "unified" (default), "ios", or "android".
regions	Character vector. Region codes for data filtering. Defaults to "WW" (worldwide).
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable SENSORTOWER_AUTH_TOKEN.

Details

This function is designed for the common workflow of: 1. Search for apps by name using ‘st_app_info()‘ 2. Get their unified IDs 3. Fetch enriched metrics for those specific apps using this function

Value

A [tibble][tibble::tibble] with enriched metrics including: - **Identification**: ‘unified_app_id‘, ‘unified_app_name‘ - **Active Users**: ‘dau_30d_us‘, ‘dau_30d_ww‘, ‘wau_4w_us‘, ‘wau_4w_ww‘, ‘mau_month_us‘, ‘mau_month_ww‘ - **Retention**: ‘retention_1d_us/ww‘, ‘retention_7d_us/ww‘, ‘retention_14d_us/ww‘, ‘retention_30d_us/ww‘, ‘retention_60d_us/ww‘ - **Demographics**: ‘genders_us‘, ‘genders_ww‘, ‘age_us‘, ‘age_ww‘, ‘male_share_us‘, ‘female_share_us‘ - **Revenue/Downloads**: ‘revenue_30d_ww‘, ‘revenue_90d_ww‘, ‘revenue_alltime_us/ww‘, ‘downloads_30d_ww‘, ‘downloads_alltime_us/ww‘ - **Monetization**: ‘rpd_alltime_us/ww‘, ‘arpu_90d_us/ww‘ - **Launch**: ‘release_date_us/ww‘, ‘earliest_release_date‘

Recommended Workflow

“‘ # Step 1: Search for apps by name apps <- st_app_info("Royal Match")

Step 2: Get unified IDs app_ids <- apps$unified_app_id

Step 3: Fetch enriched metrics metrics <- st_app_enriched(app_ids) “‘

Data Availability Notes

- **IMPORTANT: Geographic Limitations** - All enriched metrics are only available for **US market (‘_us‘ suffix)** and **Worldwide aggregates (‘_ww‘ suffix)**. Per-country data (e.g., GB, DE, FR, JP) is NOT available through this endpoint. For per-country data, use [st_sales_report()] for revenue/downloads or [st_batch_metrics()] for MAU/DAU time-series. - Retention data (D1, D7, D14, D30, D60) is aggregated for the "last quarter" - not time-series data. D90 retention is NOT available through the API. - Demographics (age/gender) are primarily available for US market only. - Not all metrics are available for all apps - smaller apps may have NA values. - This returns **snapshot data**, not time-series. For historical trends, use [st_batch_metrics()] or [st_sales_report()].

See Also

[st_app_info()] for searching apps by name, [st_app_lookup()] for resolving app IDs, [st_sales_report()] for time-series revenue/download data, [st_batch_metrics()] for time-series DAU/WAU/MAU data

Examples

```
## Not run:
# Get enriched data for specific apps
royal_match <- st_app_info("Royal Match")
enriched <- st_app_enriched(royal_match$unified_app_id)

# Get data for multiple apps at once
game_ids <- c("5f16a8019f7b275235017614", "660af7c66237390ce7c829fc")
multi_enriched <- st_app_enriched(game_ids)

# View retention metrics
multi_enriched %>%
  select(unified_app_name, starts_with("retention"))

## End(Not run)
```

---

st_app_tag *Fetch Apps by Custom Fields and Tags*

Description

Retrieves apps filtered by custom fields and tags from Sensor Tower. This function uses the /v1/app_tag/apps endpoint.

Usage

```
st_app_tag(
  app_id_type,
  custom_fields_filter_id,
  name = NULL,
  value = NULL,
  global = TRUE,
  last_known_id = NULL,
  auth_token = NULL,
  base_url = "https://api.sensortower.com"
)
```

Arguments

app_id_type	Character string. Operating System. Must be one of "itunes" (iOS) or "unified". Required.
custom_fields_filter_id	Character string. ID of a Sensor Tower custom field filter. Required. Use the filter ID from relevant endpoint.
name	Optional. Character string. Name of Custom or Global Field. Defaults to "Stock Ticker".
value	Optional. Character string. Tag value for custom or global field provided. Leave blank to fetch all possible apps.
global	Optional. Logical. Filter by global or organization custom fields. Defaults to TRUE (false means organization custom fields).
last_known_id	Optional. Character string. Supply last_known_id from previous request to get next page. Leave blank to get first page.
auth_token	Optional. Character string. Your Sensor Tower API token.
base_url	Optional. Character string. The base URL for the API.

Value

A [tibble][tibble::tibble] with app data including IDs and metadata.

---

st_apps *Search or Filter Apps*

Description

Thin v1.0.0 facade for app discovery by query string or server-side filter.

Usage

```
st_apps(
  query = NULL,
  filter = NULL,
  os = "ios",
  country = "US",
  limit = 100,
  auth_token = NULL
)
```

Arguments

query	Optional search string for app-name search.
filter	Optional ‘st_filter‘ object or 24-character filter ID string.
os	Operating system context for query searches.
country	Two-letter country code used by the filtered-app workflow.
limit	Positive integer row limit.
auth_token	Optional Sensor Tower API token.

Value

A tibble returned by the dispatched legacy implementation.

Examples

```
## Not run:
st_apps(query = "Royal Match", os = "unified", limit = 25)

filt <- st_filter(genre = "Puzzle")
st_apps(filter = filt, country = "US", limit = 50)

## End(Not run)
```

---

st_batch_app_lookup *Batch Look up app information*

Description

This function takes a vector of unified app IDs and returns a single data frame containing the corresponding unified_app_id, app_name, platform-specific IDs (ios_app_id, android_app_id), and icon_url. It handles chunking requests to the st_app_details endpoint (limit 100).

Usage

```
st_batch_app_lookup(
  app_ids,
  auth_token = Sys.getenv("SENSORTOWER_AUTH_TOKEN"),
  verbose = FALSE
)
```

Arguments

app_ids	Character vector of unified app IDs.
auth_token	Character string. Your Sensor Tower API authentication token.
verbose	Logical. Whether to show progress messages. Default is FALSE.

Value

A tibble with columns:

- unified_app_id: The Sensor Tower unified app ID
- app_name: The app's display name
- ios_app_id: iOS app ID if found
- android_app_id: Android app ID if found
- icon_url: URL of the app icon

Returns empty tibble if no apps can be found or details cannot be fetched.

Examples

```
## Not run:
# Look up multiple games
apps <- st_batch_app_lookup(c("5ba4585f539ce75b97db6bcb", "5d10d6bfecb4db5f93902f23"))

## End(Not run)
```

---

st_build_filter_url *Build Sensor Tower Filter URL*

Description

Constructs a URL for the Sensor Tower web interface where you can create custom filters. Visit this URL, configure your filters, and then copy the custom_fields_filter_id from the resulting URL.

Usage

```
st_build_filter_url(
  os = "unified",
  category = NULL,
  countries = NULL,
  base_url = "https://app.sensortower.com/top-charts"
)
```

Arguments

os	Character string. Operating system filter. One of "ios", "android", or "unified". Defaults to "unified".
category	Optional. Category ID to pre-select.
countries	Optional. Character vector of country codes to pre-select.
base_url	Character string. Base URL for Sensor Tower. Defaults to "https://app.sensortower.com/top-charts".

Value

Character string. The constructed URL.

Examples

```
## Not run:
# Build URL for iOS games in US
url <- st_build_filter_url(os = "ios", category = 6014, countries = "US")

# Open in browser
browseURL(url)

## End(Not run)
```

---

st_build_web_url *Create Sensor Tower Web URL from Parameters*

Description

Builds a Sensor Tower web interface URL from API parameters. This is the reverse of st_parse_web_url().

Usage

```
st_build_web_url(
  os = "unified",
  measure = "revenue",
  category = NULL,
  regions = "US",
  start_date = NULL,
```

```
  end_date = NULL,
  custom_fields_filter_id = NULL,
  custom_tags_mode = NULL,
  ...
)
```

Arguments

os	Operating system
measure	Measure type
category	Category ID
regions	Region codes (converted to country parameters)
start_date	Start date
end_date	End date
custom_fields_filter_id	Custom filter ID
custom_tags_mode	Custom tags mode
...	Additional parameters

Value

Character string URL

---

st_cache_info *Show App ID Cache Statistics*

Description

Display information about the current app ID cache

Usage

```
st_cache_info()
```

Value

No return value, called for side effects (displaying cache statistics).

st_categories *List Available Sensor Tower Categories*

Description

Returns a tibble of app categories recognized by the Sensor Tower API, mapping category IDs to category names for different platforms (iOS/Android). Useful for finding valid inputs for the ‘category‘ parameter in other functions.

Usage

```
st_categories(platform = NULL)
```

Arguments

platform	Optional character string. Filter results for a specific platform ("ios" or "android"). If NULL (default), returns categories for both platforms.

Value

A tibble with columns ‘platform‘ (character, "ios" or "android"), ‘category_id‘ (character, e.g., "6014"), and ‘category_name‘ (character, e.g., "Games").

Examples

```
# Get all categories
all_cats <- st_categories()
head(all_cats)

# Get only iOS categories
ios_cats <- st_categories(platform = "ios")
head(ios_cats)

# Find game categories on iOS
ios_games <- subset(st_categories("ios"), grepl("Game", category_name))
head(ios_games)
```

st_clear_app_cache *Clear App Name Cache*

Description

Clears the internal cache of app name lookups. Useful for testing or when you want to refresh app name data.

Usage

```
st_clear_app_cache()
```

Value

No return value, called for side effects (clearing the cache).

---

st_clear_id_cache *Clear App ID Cache*

Description

Clears the in-memory and on-disk cache of app ID mappings

Usage

```
st_clear_id_cache(disk = TRUE)
```

Arguments

disk	Logical. Also remove the on-disk cache file (default TRUE).

Value

No return value, called for side effects (clearing the cache).

---

st_custom_fields *Custom Fields Filter Functions*

Description

Functions to work with Sensor Tower custom fields filters.

---

st_custom_fields_utils *Custom Fields Utility Functions*

Description

Utility functions for common custom fields filtering scenarios in Sensor Tower. These functions provide pre-built filters for frequently used queries.

st_custom_fields_values *Get Custom Fields Values*

Description

Retrieves a list of all accessible custom fields and their possible values. This is useful for discovering what custom fields are available to filter by.

Usage

```
st_custom_fields_values(
  term = NULL,
  auth_token = NULL,
  base_url = "https://api.sensortower.com"
)
```

Arguments

term	Optional. Character string. Search term to filter field names.
auth_token	Optional. Character string. Your Sensor Tower API token.
base_url	Optional. Character string. The base URL for the API.

Value

A tibble containing custom fields and their possible values

Examples

```
## Not run:
# Get all custom fields
fields <- st_custom_fields_values()

# Search for specific fields
date_fields <- st_custom_fields_values(term = "date")

## End(Not run)
```

st_custom_fields_workflow *Custom Fields Filter Workflow Helper Functions*

Description

Helper functions to streamline working with custom fields filters in Sensor Tower. These functions combine the custom fields endpoints with data retrieval functions to provide a complete workflow.

st_demographics *Fetch Demographics Data for Apps*

Description

Retrieves user demographics (age and gender breakdown) for specific apps from the Sensor Tower Usage Intelligence API. This function queries the demographics endpoint directly using platform-specific app IDs.

Usage

```
st_demographics(
  unified_app_id = NULL,
  ios_app_id = NULL,
  android_app_id = NULL,
  os = NULL,
  country = "US",
  date_granularity = "all_time",
  start_date = NULL,
  end_date = NULL,
  auth_token = NULL,
  verbose = TRUE
)
```

Arguments

unified_app_id	Character string. Sensor Tower unified app ID (24-character hex). Will be resolved to platform-specific IDs automatically.
ios_app_id	Character string. iOS app ID (numeric, e.g., "1234567890").
android_app_id	Character string. Android package name (e.g., "com.example.app").
os	Character string. Operating system: "ios" or "android". Required if using platform-specific IDs. When using unified_app_id, defaults to trying both platforms.
country	Character string. Country code (e.g., "US", "GB"). Default is "US". Only single country supported per request.
date_granularity	Character string. Either "all_time" (default) or "quarterly". All-time data goes back to Q4 2015. Quarterly data begins Q1 2021.
start_date	Date or character string. Start date for quarterly data in "YYYY-MM-DD" format. Ignored for all_time granularity.
end_date	Date or character string. End date for quarterly data in "YYYY-MM-DD" format. Ignored for all_time granularity.
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable SENSORTOWER_AUTH_TOKEN.
verbose	Logical. If TRUE, prints progress messages.

Value

A [tibble][tibble::tibble] with demographics metrics including: - **app_id**: The platform-specific app ID - **os**: Platform (ios or android) - **country**: Country code - **female_percent**: Percentage of female users (0-100) - **male_percent**: Percentage of male users (0-100) - **average_age**: Average user age - **age_13_17**, **age_18_24**, **age_25_34**, **age_35_44**, **age_45_54**, **age_55_64**, **age_65_plus**: Age group percentages - **confidence**: Data confidence level

Data Availability

- Quarterly data begins Q1 2021 - All-time data goes back to Q4 2015 - Demographics are primarily available for US market - Data availability depends on app's user base size

Recommended Workflow

‘‘‘ # Step 1: Search for app by name app <- st_app_info("Royal Match")

Step 2: Get demographics using unified ID demographics <- st_demographics(unified_app_id = app$unified_app_id[1]) ‘‘‘

See Also

[st_app_info()] for searching apps by name, [st_app_lookup()] for resolving app IDs, [st_retention()] for retention metrics

Examples

```
## Not run:
# Get demographics for an app using unified ID
demo <- st_demographics(unified_app_id = "5f16a8019f7b275235017614")

# Get demographics for iOS app directly
demo <- st_demographics(
  ios_app_id = "553834731",
  os = "ios",
  country = "US"
)

## End(Not run)
```

---

st_discover_fields *Discover Available Custom Fields*

Description

Searches and displays available custom fields that can be used for filtering. This is helpful for discovering what fields are available before creating filters.

Usage

```
st_discover_fields(search_term = NULL, show_values = FALSE, auth_token = NULL)
```

Arguments

search_term	Optional. Character string to search for in field names
show_values	Logical. Whether to show possible values for each field
auth_token	Optional. Character string. Your Sensor Tower API token.

Value

A tibble with custom fields information

Examples

```
## Not run:
# Find all game-related fields
game_fields <- st_discover_fields("game")

# Find all date fields
date_fields <- st_discover_fields("date")

# Show all fields with their values
all_fields <- st_discover_fields(show_values = TRUE)

## End(Not run)
```

---

st_extract_filter_id *Extract Filter ID from Sensor Tower URL*

Description

Extracts the custom_fields_filter_id parameter from a Sensor Tower URL. This is helpful when copying URLs from the web interface.

Usage

```
st_extract_filter_id(url)
```

Arguments

url	Character string. A Sensor Tower URL containing custom_fields_filter_id

Value

Character string. The extracted filter ID, or NULL if not found.

Examples

```
## Not run:
url <- "https://app.sensortower.com/top-charts?custom_fields_filter_id=687df26ac5a19ebcfe817d7f"
filter_id <- st_extract_filter_id(url)

## End(Not run)
```

`st_extract_url_params` *Extract All Parameters from Sensor Tower URL*

Description

Extracts and displays all parameters from a Sensor Tower web URL in a readable format. Useful for understanding complex URLs.

Usage

```
st_extract_url_params(url)
```

Arguments

`url`	Character string. A Sensor Tower web interface URL

Value

Data frame with parameter names and values

Examples

```
## Not run:
url <- "https://app.sensortower.com/market-analysis/top-apps?os=unified&measure=DAU"
params_df <- st_extract_url_params(url)
View(params_df)

## End(Not run)
```

`st_facets_metrics` *Low-Level Access to Sensor Tower Facets Metrics*

Description

Performs a GET request against Sensor Tower's ‘"/v1/facets/metrics"‘ route. This helper stays intentionally low-level so package code can work with new facets-based endpoints while higher-level wrappers are added incrementally.

Usage

```
st_facets_metrics(
  query = character(),
  params = list(),
  auth_token = NULL,
  host = c("api", "app"),
  verbose = FALSE
)
```

Arguments

query	Character vector of raw query fragments appended verbatim after encoded ‘params‘. This is useful when Sensor Tower documents nested or repeated parameters that are awkward to express as a regular named R list. Example: ‘c("filters[app_ids][]=553834731")‘.
params	Named list of regular query parameters. Vector values are serialized as comma-separated strings to match Sensor Tower’s ‘style=form, explode=false‘ usage in the OpenAPI spec.
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable ‘SENSORTOWER_AUTH_TOKEN‘.
host	Character string. Which host to target: ‘"api"‘ (default) or ‘"app"‘. The ‘app‘ host uses ‘https://app.sensortower.com/api‘.
verbose	Logical. If ‘TRUE‘, prints the request URL with the auth token redacted.

Details

As of March 17, 2026, the facets route is live and the retention contract is validated against production. The machine-readable Sensor Tower docs remain gated behind a signed-in web session, so this helper still provides a stable escape hatch for raw or partially documented facets requests.

Value

A parsed JSON response. Rectangular top-level responses are returned as a tibble; nested responses are returned as a named list.

Examples

```
## Not run:
# Retention request using regular query parameters
response <- st_facets_metrics(
  params = list(
    facets = "retention",
    bundle = "retention_daily",
    breakdown = c("date", "app_id"),
    start_date = "2025-01-01",
    end_date = "2025-01-31",
    app_ids = "553834731"
  )
)
```

```
## End(Not run)
```

---

st_filter *Create a Sensor Tower Filter Object*

Description

Thin v1.0.0 facade for building or wrapping Sensor Tower custom-field filter IDs. When criteria are supplied, the function translates them into the legacy custom-fields request structure and creates a server-side filter ID using the existing implementation.

Usage

```
st_filter(
  date_from = NULL,
  date_to = NULL,
  genre = NULL,
  monetization = NULL,
  publisher = NULL,
  sdk = NULL,
  custom_fields = NULL,
  filter_id = NULL,
  combine = c("and", "or"),
  auth_token = NULL
)

## S3 method for class 'st_filter'
print(x, ...)

## S3 method for class 'st_filter'
format(x, ...)

## S3 method for class 'st_filter'
as.character(x, ...)

## S3 method for class 'st_filter'
c(..., recursive = FALSE, combine = c("and", "or"))
```

Arguments

date_from, date_to	Optional date bounds for a release-date criterion.
genre	Optional character vector of game genres.
monetization	Optional monetization criteria. Accept either a character vector such as ‘c("free", "iap")‘ or a named list like ‘list(free = TRUE, iap = TRUE)‘.
publisher	Optional character vector of publisher names.

sdk	Optional character vector of SDK names.
custom_fields	Optional raw custom-fields specification. Can be a list-of-lists already matching the legacy request format or a named list of field names and values.
filter_id	Optional existing 24-character Sensor Tower filter ID to wrap.
combine	Logical operator metadata for multi-criterion filters: ‘"and"‘ or ‘"or"‘.
auth_token	Optional Sensor Tower API token.
x	An ‘st_filter‘ object (S3 method argument).
...	Additional arguments. For ‘c.st_filter()‘, further ‘st_filter‘ objects to combine; ignored by ‘print.st_filter()‘, ‘format.st_filter()‘, and ‘as.character.st_filter()‘.
recursive	Unused; present for S3 method-signature compatibility with [base::c()].

Value

An object of class ‘st_filter‘, implemented as a list with elements: - ‘criteria‘: normalized criteria supplied to ‘st_filter()‘ - ‘combine‘: ‘"and"‘ or ‘"or"‘ - ‘filter_id‘: the wrapped or created server-side filter ID

Examples

```
## Not run:
st_filter(genre = c("Puzzle", "Strategy"))

st_filter(
  date_from = "2024-01-01",
  date_to = "2024-12-31",
  monetization = c("free", "iap")
)

existing <- st_filter(filter_id = "687df26ac5a19ebcfe817d7f")
as.character(existing)

## End(Not run)
```

---

st_game_summary *Fetch Game Market Summary Data*

Description

Retrieves aggregate download and revenue estimates by game categories, countries, and date ranges from Sensor Tower’s ‘games_breakdown‘ endpoint. Use this for market/category denominator series; do not approximate market totals by batching top charts, rankings, or large app rosters.

Usage

```
st_game_summary(
  categories = 7001,
  countries,
  os,
  date_granularity,
  start_date,
  end_date,
  auth_token = NULL,
  base_url = "https://api.sensortower.com",
  enrich_response = TRUE
)
```

Arguments

categories	Character string or numeric vector. Game category IDs to analyze. Defaults to 7001 (a popular game category). Use ‘st_categories()‘ to find valid category IDs.
countries	Character vector or comma-separated string. Country codes (e.g., ‘"US"‘, ‘c("US", "GB")‘, ‘"WW"‘ for worldwide) to analyze. Required.
os	Character string. Operating System. Must be one of "ios", "android", or "unified". Required. Note: The underlying API only supports "ios" and "android"; when ‘os = "unified"‘ this function automatically fetches both platforms and combines them into a single table with total columns.
date_granularity	Character string. Time granularity for aggregation. Must be one of "daily", "weekly", "monthly", or "quarterly". Required.
start_date	Character string or Date object. Start date for the query in "YYYY-MM-DD" format. Required.
end_date	Character string or Date object. End date for the query in "YYYY-MM-DD" format, inclusive. Required.
auth_token	Optional. Character string. Your Sensor Tower API token.
base_url	Optional. Character string. The base URL for the API.
enrich_response	Optional. Logical. If ‘TRUE‘ (default), enriches the response with readable column names and processes the data.

Value

A [tibble][tibble::tibble] with aggregate game market summary data including: - **Category information**: Game category details - **Geographic data**: Country-level breakdowns - **Downloads**: iOS (iPhone + iPad combined) and Android download estimates - **Revenue**: iOS (iPhone + iPad combined) and Android revenue estimates, expressed in dollars when ‘enrich_response = TRUE‘ - **Totals (unified only)**: ‘Total Downloads‘, ‘Total Revenue‘ - **Time series**: Data broken down by specified granularity

Automatic Data Combination: For iOS and unified platforms, iPhone and iPad data are automatically combined into single "iOS Downloads" and "iOS Revenue" columns for simplified analysis.

API Endpoint Used

- **Game Summary**: `GET /v1/{os}/games_breakdown` (API only supports `os = "ios"` or `os = "android"`; unified is synthesized)

Revenue Units

The raw `games_breakdown` endpoint returns revenue in cents. With `enrich_response = TRUE`, this function converts revenue fields to dollars before combining devices or platforms. With `enrich_response = FALSE`, raw endpoint fields are returned unchanged.

Field Mappings and Processing

The API returns abbreviated field names which are automatically mapped to descriptive names and processed: - **iOS**: `iu` + `au` = iOS Downloads (combined), `ir` + `ar` = iOS Revenue (combined) - **Android**: `u` = Android Downloads, `r` = Android Revenue - **Common**: `ca` = Category, `cc` = Country Code, `d` = Date

iPhone and iPad data are automatically combined for simplified analysis. For `os = "unified"`, iOS and Android aggregate rows are summed by date and country across the requested category basket.

See Also

[st_categories()], [st_rankings()], [st_metrics()]

Examples

```
## Not run:
# Specific categories and countries
rpg_summary <- st_game_summary(
  categories = c(7001, 7002),
  countries = c("US", "GB", "DE"),
  os = "ios",
  date_granularity = "weekly",
  start_date = "2024-01-01",
  end_date = "2024-03-31"
)

# Monthly summary for iOS games in the US
ios_monthly <- st_game_summary(
  os = "ios",
  countries = "US",
  date_granularity = "monthly",
  start_date = "2024-01-01",
  end_date = "2024-06-30"
)

## End(Not run)
```

st_get_app_names *Get App Names from Publisher Apps Result*

Description

Helper function to create a name lookup table from the result of ‘st_publisher_apps()‘. This handles canonical ID mapping automatically, so you can join sales data (which uses canonical IDs) back to app names.

Usage

```
st_get_app_names(apps_df, include_canonical = TRUE)
```

Arguments

apps_df	A tibble returned by ‘st_publisher_apps()‘.
include_canonical	Logical. If TRUE, includes mappings for canonical IDs that were resolved during aggregation. Defaults to TRUE.

Value

A tibble with columns ‘unified_app_id‘ and ‘app_name‘ suitable for joining with sales data or other API results.

Examples

```
## Not run:
# Get apps with canonical ID resolution
apps <- st_publisher_apps("647eb849d9d91f31a54f1792", aggregate_related = TRUE)

# Get name lookup table
name_lookup <- st_get_app_names(apps)

# Use with sales data
sales <- st_unified_sales_report(apps$unified_app_id, ...)
sales_with_names <- sales %>%
  left_join(name_lookup, by = "unified_app_id")

## End(Not run)
```

st_get_filter_collection *Get Pre-Built Filter Collections*

Description

Returns commonly used filter IDs for quick access to pre-defined app segments.

Usage

```
st_get_filter_collection(
  collection = c("top_genres", "monetization_models", "platform_exclusive",
    "market_segments"),
  auth_token = NULL
)
```

Arguments

collection	Character. Name of the collection: - "top_genres": Major game genres - "monetization_models": Different monetization approaches - "platform_exclusive": Platform-specific apps - "market_segments": Market segment filters
auth_token	Optional. Character string. Your Sensor Tower API token.

Value

A named list of filter IDs

Examples

```
## Not run:
# Get filter IDs for top game genres
genre_filters <- st_get_filter_collection("top_genres")

# Use a filter from the collection
puzzle_apps <- st_get_filtered_apps(
  filter_id = genre_filters$puzzle,
  measure = "DAU",
  regions = "US"
)

## End(Not run)
```

st_get_filtered_apps *Get Top Apps with Custom Filter*

Description

Retrieves top apps using a custom fields filter. This combines filter creation with data retrieval in a single workflow.

Usage

```
st_get_filtered_apps(
  field_name = NULL,
  field_values = NULL,
  filter_id = NULL,
  measure = "DAU",
  regions = "US",
  date = NULL,
  end_date = NULL,
  limit = 100,
  enrich_response = TRUE,
  auth_token = NULL,
  ...
)
```

Arguments

field_name	Character. Name of the custom field to filter by (or NULL to use filter_id)
field_values	Character vector. Values to filter for (or NULL to use filter_id)
filter_id	Character. Existing filter ID to use (alternative to field_name/values)
measure	Character. Metric to measure: "DAU", "WAU", "MAU", "revenue", or "units"
regions	Character vector. Region codes (e.g., "US", "WW")
date	Character or Date. Start date for the query
end_date	Optional. Character or Date. End date for the query
limit	Integer. Maximum number of apps to return (default 100)
enrich_response	Logical. Whether to enrich with additional metrics
auth_token	Optional. Character string. Your Sensor Tower API token.
...	Additional parameters passed to st_top_charts

Value

A tibble with top apps data

Examples

```
## Not run:
# Get top Word games by DAU
word_games <- st_get_filtered_apps(
  field_name = "Game Sub-genre",
  field_values = "Word",
  measure = "DAU",
  regions = "US",
  limit = 20
)

# Use existing filter ID
apps <- st_get_filtered_apps(
  filter_id = "603697f4241bc16eb8570d37",
  measure = "revenue",
  regions = "US"
)

## End(Not run)
```

`st_get_unified_mapping` *Get Unified ID Mapping for Apps*

Description

Retrieves the mapping between platform-specific app IDs and unified app IDs. This function handles cases where platform IDs from st_top_charts may not be directly searchable, using app names as a fallback resolution method.

Usage

```
st_get_unified_mapping(
  app_ids,
  app_names = NULL,
  os = "unified",
  auth_token = Sys.getenv("SENSORTOWER_AUTH_TOKEN")
)
```

Arguments

`app_ids`	Character vector of app IDs (can be iOS, Android, or unified hex IDs)
`app_names`	Character vector of app names (optional, helps with resolution)
`os`	Character string. Operating system: "ios", "android", or "unified"
`auth_token`	Character string. Sensor Tower API authentication token. Defaults to environment variable SENSORTOWER_AUTH_TOKEN.

Details

This function uses an ID-first approach (no name-based resolution): 1. For hex IDs (24-char), uses st_app_lookup to get platform IDs 2. For platform IDs, first tries to look them up via st_app_lookup 3. If direct lookup fails, searches the unified index using the platform ID as the term and matches exact IDs within nested ios_apps/android_apps 4. Returns the best available mapping for each app using IDs only

Note: Platform IDs from st_top_charts may be regional or legacy IDs that aren't directly searchable. In these cases, name-based search provides the most reliable resolution to unified IDs.

Value

A data frame with columns: - 'input_id': The original ID provided - 'unified_app_id': The unified app ID (hex format) - 'unified_app_name': The unified app name - 'ios_app_id': iOS app ID (if available) - 'android_app_id': Android app ID (if available) - 'publisher_id': Publisher ID - 'publisher_name': Publisher name

Examples

```
## Not run:
# Get mapping with app names for better resolution
mapping <- st_get_unified_mapping(
  app_ids = c("943599237", "com.bandainamcogames.dbzdokkan"),
  app_names = c("Dragon Ball Z Dokkan Battle", "Dragon Ball Z Dokkan Battle"),
  os = "unified"
)

## End(Not run)
```

---

st_gt_dashboard *Create FiveThirtyEight-styled GT Dashboard from Top Charts Data*

Description

Creates a professional, FiveThirtyEight-themed GT table dashboard from Sensor Tower top charts data with customizable styling and metric options.

Usage

```
st_gt_dashboard(
  data,
  title = "Top Mobile Games",
  subtitle = NULL,
  ranking_metric = "revenue_180d_ww",
  show_demographics = TRUE,
  show_engagement = TRUE,
  show_retention = TRUE,
```

```
  retention_region = "us",
  show_rpd = TRUE,
  bar_charts = TRUE,
  bar_chart_columns = NULL,
  heatmap_retention = TRUE,
  compact_mode = TRUE,
  width = 1800,
  height = 700,
  save_path = NULL,
  icon_cache_dir = "inst/images/app_icons",
  raw = FALSE,
 color_scheme = list(revenue = "#FF6600", downloads = "#008FD5", engagement = "#9C27B0",
  rpd = "#4CAF50", retention_low = "#FFCDD2", retention_mid = "#C8E6C9", retention_high
    = "#4CAF50")
)
```

Arguments

data	Data frame from st_top_charts() or similar Sensor Tower function
title	Character string for the table title (default: "Top Mobile Games")
subtitle	Character string for subtitle. If NULL, auto-generates based on data
ranking_metric	Character string specifying which metric to use for ranking. Options: "revenue_180d_ww", "revenue_30d_ww", "downloads_180d_ww", "downloads_30d_ww", etc. (default: "revenue_180d_ww")
show_demographics	Logical, whether to show demographic columns (age, gender) (default: TRUE)
show_engagement	Logical, whether to show engagement metrics (DAU, WAU, MAU) (default: TRUE)
show_retention	Logical, whether to show retention metrics (default: TRUE)
retention_region	Character string for retention region ("us", "ww", etc.) (default: "us")
show_rpd	Logical, whether to show Revenue Per Download (default: TRUE)
bar_charts	Logical, whether to show bar chart visualizations (default: TRUE)
bar_chart_columns	Character vector of column patterns to add bar charts to. If NULL, applies to all numeric columns except RPD and retention.
heatmap_retention	Logical, whether to apply heatmap to retention columns (default: TRUE)
compact_mode	Logical, whether to use compact row heights (default: TRUE)
width	Numeric, table width in pixels (default: 1800)
height	Numeric, table height in pixels (default: 700)
save_path	Character string, path to save the table image. If NULL, returns GT object
icon_cache_dir	Character string, directory to cache app icons (default: "inst/images/app_icons")

raw	Logical, whether to return a minimally styled table without custom formatting, bar charts, or heatmaps (default: FALSE)
color_scheme	List with color codes for different metric types: - revenue: Revenue metrics color (default: "#FF6600") - downloads: Downloads metrics color (default: "#008FD5") - engagement: Engagement metrics color (default: "#9C27B0") - rpd: RPD metrics color (default: "#4CAF50") - retention_low: Low retention color (default: "#FFCDD2") - retention_mid: Mid retention color (default: "#C8E6C9") - retention_high: High retention color (default: "#4CAF50")

Value

GT object (if save_path is NULL) or saves image and returns path

Examples

```
## Not run:
# Basic usage - one line after st_top_charts()
top_rpgs <- st_top_charts(category = 7014, measure = "revenue")
st_gt_dashboard(top_rpgs)

# Raw mode for minimal styling
st_gt_dashboard(top_rpgs, raw = TRUE)

# Customize the dashboard
st_gt_dashboard(
  top_rpgs,
  title = "Top Role-Playing Games Q4 2024",
  ranking_metric = "revenue_30d_ww",
  show_retention = FALSE,
  save_path = "dashboard.png"
)

# Change color scheme
st_gt_dashboard(
  top_rpgs,
  color_scheme = list(
    revenue = "#E74C3C",
    downloads = "#3498DB",
    engagement = "#9B59B6"
  )
)

## End(Not run)
```

---

st_is_valid_filter_id *Validate Custom Field Filter ID Format*

Description

Checks if a filter ID matches the expected 24-character hexadecimal format used by Sensor Tower.

Usage

```
st_is_valid_filter_id(filter_id)
```

Arguments

filter_id	Character string. The filter ID to validate

Value

Logical. TRUE if valid format, FALSE otherwise

Examples

```
## Not run:
# Valid filter ID
st_is_valid_filter_id("687df26ac5a19ebcfe817d7f")  # TRUE

# Invalid filter IDs
st_is_valid_filter_id("invalid")  # FALSE
st_is_valid_filter_id("687df26ac5a19ebcfe817d7")  # FALSE (too short)

## End(Not run)
```

---

st_metrics *Fetch Sensor Tower Metrics*

Description

Thin v1.0.0 facade for sales/download metrics. The function validates a standardized argument set, dispatches to the legacy implementation that best matches the request, and then normalizes the result into a stable schema.

Usage

```
st_metrics(
  app_id,
  metrics = c("revenue", "downloads"),
  os = "unified",
  countries = "WW",
  date_from = Sys.Date() - 90,
  date_to = Sys.Date(),
  granularity = "daily",
  revenue_unit = c("dollars", "cents"),
  shape = c("long", "wide"),
  cache = TRUE,
  auth_token = NULL
)
```

Arguments

app_id	Character scalar or vector of app identifiers. Each entry can be a Sensor Tower unified app ID, an iOS numeric app ID, or an Android package name.
metrics	Character vector of metrics to return. Supported values are ‘"revenue"‘ and ‘"downloads"‘.
os	Character scalar. One of ‘"ios"‘, ‘"android"‘, or ‘"unified"‘.
countries	Character vector of 2-letter country codes. Use ‘"WW"‘ for worldwide aggregates.
date_from, date_to	Date bounds for the query. Accept ‘Date‘ objects or ISO date strings.
granularity	Character scalar. One of ‘"daily"‘, ‘"weekly"‘, ‘"monthly"‘, or ‘"quarterly"‘.
revenue_unit	Character scalar. ‘"dollars"‘ (default) returns revenue in base currency units. ‘"cents"‘ returns revenue in cents for compatibility with the legacy API surface.
shape	Character scalar. ‘"long"‘ returns one row per metric observation. ‘"wide"‘ returns one row per app/date/country with separate metric columns.
cache	Logical. If ‘TRUE‘, use a process-local cache keyed on the normalized arguments.
auth_token	Optional Sensor Tower API token. If ‘NULL‘, falls back to ‘SENSORTOWER_AUTH_TOKEN‘.

Value

If ‘shape = "long"‘, a tibble with columns: - ‘app_id‘: identifier supplied to ‘st_metrics()‘ - ‘os‘: normalized operating system - ‘country‘: 2-letter country code - ‘date‘: observation date - ‘metric‘: one of ‘"revenue"‘ or ‘"downloads"‘ - ‘value‘: metric value; revenue is in dollars by default and cents when ‘revenue_unit = "cents"‘

If ‘shape = "wide"‘, a tibble with columns: - ‘app_id‘ - ‘os‘ - ‘country‘ - ‘date‘ - one numeric column per requested metric

Examples

```
## Not run:
st_metrics(
  app_id = "553834731",
  os = "ios",
  countries = "US",
  date_from = Sys.Date() - 30,
  date_to = Sys.Date() - 1
)

st_metrics(
  app_id = c("553834731", "com.supercell.clashofclans"),
  os = "unified",
  shape = "wide",
  countries = c("US", "GB"),
  revenue_unit = "cents"
)
```

```
## End(Not run)
```

---

st_parse_web_url *Parse Sensor Tower Web URL to API Parameters*

Description

Converts a Sensor Tower web interface URL into API-compatible parameters that can be used with sensortowerR functions. This is helpful when you want to replicate a web query in R.

Usage

```
st_parse_web_url(url, verbose = TRUE)
```

Arguments

url	Character string. A Sensor Tower web interface URL
verbose	Logical. Whether to print parameter mapping details. Defaults to TRUE.

Value

List of API-compatible parameters suitable for use with st_top_charts() and other sensortowerR functions

Examples

```
## Not run:
# Parse a web URL
url <- "https://app.sensortower.com/market-analysis/top-apps?os=unified&measure=DAU"
params <- st_parse_web_url(url)

# Use the parameters in an API call
data <- do.call(st_top_charts, params)

# Or modify parameters before using
params$limit <- 50
data <- do.call(st_top_charts, params)

## End(Not run)
```

st_publisher_apps *Get All Apps from a Publisher*

Description

Retrieves a list of apps associated with a specified unified publisher ID from the Sensor Tower API. Targets the '/v1/unified/publishers/apps' endpoint.

Usage

```
st_publisher_apps(
  unified_id = NULL,
  publisher_id = NULL,
  aggregate_related = FALSE,
  auth_token = Sys.getenv("SENSORTOWER_AUTH_TOKEN"),
  verbose = TRUE
)
```

Arguments

unified_id	Character. Unified ID to resolve apps for. May be either: - Unified Publisher ID (24-char hex) - Unified App ID (24-char hex) belonging to a publisher The API returns the unified publisher and all associated apps in both cases.
publisher_id	Deprecated alias for 'unified_id'.
aggregate_related	Logical. If TRUE, ensures each app's unified_app_id is the canonical ID that aggregates ALL regional SKUs. This solves the problem where games like "Watcher of Realms" are published under multiple regional publishers (Moonton, Vizta Games, Skystone Games, etc.) and may return different unified_app_ids. When TRUE, the function looks up each app by name to find the true unified_app_id that combines all regional versions. Defaults to FALSE for backwards compatibility.
auth_token	Character. Your Sensor Tower API authentication token. Defaults to the value stored in the 'SENSORTOWER_AUTH_TOKEN' environment variable.
verbose	Logical. If TRUE, prints progress messages during aggregation. Defaults to TRUE.

Value

A [tibble][tibble::tibble] containing details of the apps associated with the publisher. The exact columns depend on the API response but often include app IDs, names, platform, etc. Returns an empty tibble if the publisher ID is invalid, has no apps, or an error occurs.

Solving Regional Publisher Issues

Many publishers have regional subsidiaries or partners that publish the same game under different app IDs in different regions. For example, Moonton's "Watcher of Realms" is published by Moonton in some regions, Vizta Games in others, and Skystone Games in others.

When 'aggregate_related = TRUE', this function ensures you get the unified_app_id that represents the FULL game across all regional publishers, which is required for accurate revenue/download aggregation via 'st_unified_sales_report()'.

API Endpoint Used

- 'GET /v1/unified/publishers/apps'

Examples

```
## Not run:
# Ensure SENSORTOWER_AUTH_TOKEN is set in your environment
# Sys.setenv(SENSORTOWER_AUTH_TOKEN = "your_secure_auth_token_here")

# Basic usage - get publisher's apps
apps_list <- st_publisher_apps(unified_id = "647eb849d9d91f31a54f1792")

# With regional SKU aggregation - ensures canonical unified_app_ids
apps_list <- st_publisher_apps(
  unified_id = "647eb849d9d91f31a54f1792",
  aggregate_related = TRUE
)

# Then use with st_unified_sales_report() for accurate data
sales <- st_unified_sales_report(
  unified_app_id = apps_list$unified_app_id,
  countries = "WW",
  start_date = "2024-01-01",
  end_date = "2024-12-31",
  date_granularity = "monthly"
)

## End(Not run)
```

---

st_publisher_portfolio

Publisher Portfolio Analysis

Description

Fetches comprehensive portfolio data for a publisher including revenue, downloads, MAU, and rankings. Returns a tidy data frame ready for visualization or GT table creation.

Usage

```
st_publisher_portfolio(
  publisher = NULL,
  publisher_id = NULL,
  start_date = "2023-01-01",
  end_date = NULL,
  countries = "WW",
  metrics = c("revenue", "downloads", "mau"),
  include_rankings = TRUE,
  include_portfolio_total = TRUE,
  granularity = "yearly",
  min_revenue = 1e+05,
  auth_token = Sys.getenv("SENSORTOWER_AUTH_TOKEN"),
  verbose = TRUE,
  use_cache = FALSE,
  cache_dir = NULL
)
```

Arguments

publisher	Character. Publisher name to search for (e.g., "Lilith Games", "Supercell", "King"). The function will search for the publisher and use the first match.
publisher_id	Character. Optional. If provided, skips the publisher search and uses this unified_publisher_id directly.
start_date	Date or character. Start date for metrics (default: "2023-01-01").
end_date	Date or character. End date for metrics (default: last day of previous month).
countries	Character. Countries for metrics (default: "WW" for worldwide).
metrics	Character vector. Which metrics to fetch. Options: "revenue", "downloads", "mau". Default: all three.
include_rankings	Logical. Whether to fetch subgenre rankings from top charts. Default: TRUE.
include_portfolio_total	Logical. Whether to add a portfolio total row. Default: TRUE.
granularity	Character. How to aggregate the data: "yearly" (default), "quarterly", or "monthly".
min_revenue	Numeric. Minimum revenue threshold to include an app. Default: 100000 (apps with at least $100K in any year).
auth_token	Character. Sensor Tower API token. Defaults to SENSORTOWER_AUTH_TOKEN environment variable
verbose	Logical. Print progress messages. Default: TRUE.
use_cache	Logical. Use cached data if available. Default: FALSE. When TRUE, requires cache_dir to be specified.
cache_dir	Character. Directory for cached data. Default: NULL (no caching). Must be explicitly set to enable caching. Use tempdir() for temporary caching.

Value

A tibble with portfolio data including: - app_name: Game name - subgenre: Game sub-genre - subgenre_rank: Rank within sub-genre - revenue_{year}: Revenue by year - downloads_{year}: Downloads by year - mau_{year}: Average MAU by year (if requested) - revenue_yoy, downloads_yoy, mau_yoy: Year-over-year growth percentages

Examples

```
## Not run:
# Simple usage - just provide publisher name
lilith_portfolio <- st_publisher_portfolio("Lilith Games")

# Piped workflow
library(dplyr)

"Supercell" %>%
  st_publisher_portfolio(
    start_date = "2023-01-01",
    metrics = c("revenue", "downloads")
  ) %>%
  filter(revenue_2024 > 1000000) %>%
  arrange(desc(revenue_2024))

# Custom date range and countries
portfolio <- st_publisher_portfolio(
  publisher = "King",
  start_date = "2022-01-01",
  end_date = "2024-12-31",
  countries = c("US", "GB", "DE"),
  metrics = c("revenue", "downloads", "mau"),
  include_rankings = TRUE
)

## End(Not run)
```

---

st_rankings *Fetch Sensor Tower Rankings*

Description

Thin v1.0.0 facade that standardizes the app, publisher, and category ranking entrypoints while dispatching to the existing implementations.

Usage

```
st_rankings(
  entity = c("app", "publisher", "category"),
  os = "ios",
```

```
  category = NULL,
  country = "US",
  chart_type = "topfreeapplications",
  date = Sys.Date() - 1,
  limit = 100,
  filter = NULL,
  auth_token = NULL
)
```

Arguments

entity	Ranking entity to fetch: '"app"', '"publisher"', or '"category"'.
os	Operating system. One of '"ios"', '"android"', or '"unified"'.
category	Optional category identifier forwarded to the legacy ranking implementation.
country	Two-letter country code.
chart_type	Chart type for category rankings. Ignored for app and publisher rankings.
date	Ranking date. Accepts a 'Date' object or ISO date string.
limit	Positive integer row limit.
filter	Optional 'st_filter' object or 24-character filter ID string.
auth_token	Optional Sensor Tower API token.

Value

A tibble with standardized columns 'rank', 'id', 'name', 'os', 'category', 'country', and 'date', plus any entity-specific columns returned by the dispatched implementation.

Examples

```
## Not run:
st_rankings(entity = "app", os = "ios", category = 6014, country = "US")

st_rankings(
  entity = "category",
  os = "android",
  category = "game",
  chart_type = "topgrossing",
  country = "GB"
)

## End(Not run)
```

st_ratings_facets *Fetch Rating Metrics from Sensor Tower's Facets API*

Description

Retrieves rating metrics from Sensor Tower's new facets-based ratings endpoint. This wrapper targets ‘"/v1/facets/metrics?facets=ratings"‘ and returns a tidy tibble with the live response columns preserved.

Usage

```
st_ratings_facets(
  app_ids,
  bundle = c("ratings_incremental", "ratings_cumulative"),
  breakdown = c("app_id", "date"),
  start_date,
  end_date,
  date_granularity = NULL,
  regions = NULL,
  android_localized_estimates = TRUE,
  auth_token = NULL,
  verbose = FALSE
)
```

Arguments

app_ids	Character vector of iOS app IDs or Android package names. Sensor Tower documents a maximum of 1,000 IDs per request.
bundle	Character string. Rating bundle to request: ‘"ratings_incremental"‘ or ‘"ratings_cumulative"‘.
breakdown	Breakdown fields or a comma-separated breakdown string. Supported combinations are ‘"app_id"‘, ‘"app_id,date"‘, ‘"region"‘, ‘"region,date"‘, and ‘"app_version"‘.
start_date	Start date in ‘YYYY-MM-DD‘ format or as ‘Date‘.
end_date	End date in ‘YYYY-MM-DD‘ format or as ‘Date‘.
date_granularity	Optional date granularity. Required when ‘breakdown‘ includes ‘date‘.
regions	Optional character vector of region codes.
android_localized_estimates	Logical. Whether to apply Android country weighting. Defaults to ‘TRUE‘ to match the current documented default.
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable ‘SENSORTOWER_AUTH_TOKEN‘.
verbose	Logical. If ‘TRUE‘, prints the request URL with the auth token redacted.

Details

Validated against the live Sensor Tower API on March 17, 2026.

Value

A [tibble][tibble::tibble] containing rating metrics such as ‘rating_average_incremental‘, ‘rating_count_incremental‘, or their cumulative counterparts.

See Also

[st_facets_metrics()] for raw facets access

Examples

```
## Not run:
ratings <- st_ratings_facets(
  app_ids = "553834731",
  bundle = "ratings_incremental",
  breakdown = c("app_id", "date"),
  start_date = "2024-01-01",
  end_date = "2024-01-07",
  date_granularity = "day"
)

## End(Not run)
```

---

st_retention *Fetch Retention Data for Apps*

Description

Retrieves retention metrics (D1-D90) for specific apps from the Sensor Tower Usage Intelligence API. This function queries the retention endpoint directly using platform-specific app IDs.

Usage

```
st_retention(
  unified_app_id = NULL,
  ios_app_id = NULL,
  android_app_id = NULL,
  os = NULL,
  country = "US",
  date_granularity = "all_time",
  start_date = NULL,
  end_date = NULL,
  auth_token = NULL,
  verbose = TRUE
)
```

Arguments

unified_app_id	Character string. Sensor Tower unified app ID (24-character hex). Will be resolved to platform-specific IDs automatically.
ios_app_id	Character string. iOS app ID (numeric, e.g., "1234567890").
android_app_id	Character string. Android package name (e.g., "com.example.app").
os	Character string. Operating system: "ios" or "android". Required if using platform-specific IDs. When using unified_app_id, defaults to "ios" but will try both platforms.
country	Character string. Country code (e.g., "US", "GB"). Default is "US". Only single country supported per request.
date_granularity	Character string. Either "all_time" (default) or "quarterly". All-time data goes back to Q4 2015. Quarterly data begins Q1 2021.
start_date	Date or character string. Start date for quarterly data in "YYYY-MM-DD" format. Ignored for all_time granularity.
end_date	Date or character string. End date for quarterly data in "YYYY-MM-DD" format. Ignored for all_time granularity.
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable SENSORTOWER_AUTH_TOKEN.
verbose	Logical. If TRUE, prints progress messages.

Value

A [tibble][tibble::tibble] with retention metrics including: - **app_id**: The platform-specific app ID - **os**: Platform (ios or android) - **country**: Country code - **retention_d1** through **retention_d90**: Retention percentages (0-1 scale) - **confidence**: Data confidence level (red=low, yellow=medium, green=high) - **baseline_downloads**: Total downloads in baseline period - **baseline_start_date**, **baseline_end_date**: Dates for baseline period

Data Availability

- Quarterly data begins Q1 2021 - All-time data goes back to Q4 2015 - Data is only available for apps with sufficient user base - Confidence levels: red (<=3), yellow (4-6), green (>=7)

Recommended Workflow

“‘ # Step 1: Search for app by name app <- st_app_info("Royal Match")

Step 2: Get retention data using unified ID retention <- st_retention(unified_app_id = app$unified_app_id[1]) “‘

See Also

[st_app_info()] for searching apps by name, [st_app_lookup()] for resolving app IDs, [st_demographics()] for user demographics data

Examples

```
## Not run:
# Get retention for an app using unified ID
retention <- st_retention(unified_app_id = "5f16a8019f7b275235017614")

# Get retention for iOS app directly
retention <- st_retention(
  ios_app_id = "553834731",
  os = "ios",
  country = "US"
)

# Get quarterly retention data
retention <- st_retention(
  unified_app_id = "5f16a8019f7b275235017614",
  date_granularity = "quarterly",
  start_date = "2024-01-01",
  end_date = "2024-09-30"
)

## End(Not run)
```

---

st_retention_facets *Fetch Retention Metrics from Sensor Tower's Facets API*

Description

Retrieves retention curves from Sensor Tower's new facets-based retention endpoint. This wrapper targets ‘"/v1/facets/metrics?facets=retention"‘ and returns a tidy tibble with the live response columns preserved.

Usage

```
st_retention_facets(
  app_ids = NULL,
  unified_app_ids = NULL,
  bundle = c("retention_daily", "retention_weekly", "retention_monthly"),
  breakdown = c("date", "app_id"),
  start_date,
  end_date,
  regions = NULL,
  auth_token = NULL,
  verbose = FALSE
)
```

Arguments

app_ids	Character vector of platform-specific app IDs. May contain iOS numeric IDs or Android bundle IDs. Supply exactly one of ‘app_ids‘ or ‘unified_app_ids‘.
unified_app_ids	Character vector of Sensor Tower unified app IDs. Supply exactly one of ‘app_ids‘ or ‘unified_app_ids‘.
bundle	Character string. Retention bundle to request: ‘"retention_daily"‘, ‘"retention_weekly"‘, or ‘"retention_monthly"‘.
breakdown	Character vector of breakdown fields. Supported combinations are the ones documented by Sensor Tower: ‘"date"‘, ‘"app_id"‘, ‘"unified_app_id"‘, ‘"date,app_id"‘, ‘"date,unified_app_id"‘, ‘"unified_app_id,app_id"‘, and ‘"date,unified_app_id,app_id"‘.
start_date	Start date in ‘YYYY-MM-DD‘ format or as ‘Date‘.
end_date	End date in ‘YYYY-MM-DD‘ format or as ‘Date‘.
regions	Optional character vector of region codes. When omitted, Sensor Tower returns worldwide estimates.
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable ‘SENSORTOWER_AUTH_TOKEN‘.
verbose	Logical. If ‘TRUE‘, prints the request URL with the auth token redacted.

Details

Validated against the live Sensor Tower API on March 17, 2026. The production response currently returns daily retention columns including ‘est_retention_d14‘ and ‘est_retention_d365‘.

Value

A [tibble][tibble::tibble] with one row per requested breakdown combination. The response preserves Sensor Tower’s live metric column names such as ‘est_retention_d1‘, ‘est_retention_d14‘, ‘est_retention_w52‘, or ‘est_retention_m12‘.

See Also

[st_retention()] for the legacy retention endpoint, [st_facets_metrics()] for raw facets access

Examples

```
## Not run:
retention <- st_retention_facets(
  app_ids = "553834731",
  bundle = "retention_daily",
  breakdown = c("date", "app_id"),
  start_date = "2025-01-01",
  end_date = "2025-01-31"
)

## End(Not run)
```

st_reviews_by_rating_facets *Fetch Review Metrics by Rating from Sensor Tower's Facets API*

Description

Retrieves review metrics broken down by star rating from Sensor Tower's new facets-based review endpoint. This wrapper targets ‘"/v1/facets/metrics?facets=reviews_by_rating"‘.

Usage

```
st_reviews_by_rating_facets(
  app_id,
  breakdown = c("date", "review_rating"),
  start_date,
  end_date,
  date_granularity = NULL,
  regions = NULL,
  languages = NULL,
  review_keywords = NULL,
  review_sentiments = NULL,
  review_tags = NULL,
  search_terms = NULL,
  rating_filters = NULL,
  auth_token = NULL,
  verbose = FALSE
)
```

Arguments

app_id	Single iOS app ID or Android package name.
breakdown	Breakdown fields or a comma-separated breakdown string. Supported combinations are ‘"review_rating"‘, ‘"date,review_rating"‘, ‘"region,review_rating"‘, ‘"language,review_rating"‘, and ‘"app_version,review_rating"‘.
start_date	Start date in ‘YYYY-MM-DD‘ format or as ‘Date‘.
end_date	End date in ‘YYYY-MM-DD‘ format or as ‘Date‘.
date_granularity	Optional date granularity. Required when ‘breakdown‘ includes ‘date‘.
regions	Optional character vector of iOS region codes.
languages	Optional character vector of Android language codes.
review_keywords	Optional character vector of review-keyword filters.
review_sentiments	Optional character vector of sentiment filters.
review_tags	Optional character vector of review-tag filters.

search_terms	Optional character vector of content search terms.
rating_filters	Optional character vector or integer vector of star filters (‘1‘ through ‘5‘).
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable ‘SENSORTOWER_AUTH_TOKEN‘.
verbose	Logical. If ‘TRUE‘, prints the request URL with the auth token redacted.

Details

Validated against the live Sensor Tower API on March 17, 2026.

Value

A [tibble][tibble::tibble] containing fields such as ‘review_rating‘, ‘review_rating_count‘, ‘review_rating_percentage‘, and ‘review_rating_average‘.

See Also

[st_facets_metrics()] for raw facets access

Examples

```
## Not run:
reviews <- st_reviews_by_rating_facets(
  app_id = "553834731",
  breakdown = c("date", "review_rating"),
  start_date = "2024-01-01",
  end_date = "2024-01-07",
  date_granularity = "day",
  regions = "US"
)

## End(Not run)
```

---

st_session_metrics *Fetch Session Metrics Time Series Data*

Description

Retrieves session metrics time series data (session count, session duration, time spent) for apps from the Sensor Tower Usage Intelligence API.

Usage

```
st_session_metrics(
  unified_app_id = NULL,
  ios_app_id = NULL,
  android_app_id = NULL,
  start_date,
  end_date,
  metrics = c("session_count", "session_duration", "time_spent"),
  regions = "US",
  time_period = "week",
  date_granularity = "monthly",
  os = NULL,
  breakdown = "unified_app_id",
  auth_token = NULL,
  verbose = TRUE
)
```

Arguments

unified_app_id	Character string or vector. Sensor Tower unified app ID(s) (24-character hex). Maximum 100 apps per request.
ios_app_id	Character string or vector. iOS app ID(s) for non-unified queries.
android_app_id	Character string or vector. Android package name(s) for non-unified queries.
start_date	Date or character string. Start date in "YYYY-MM-DD" format. Data is available from 2021-01-01 onward.
end_date	Date or character string. End date in "YYYY-MM-DD" format.
metrics	Character vector. Metrics to retrieve. Options include: - "time_spent" (average seconds per user per day) - "total_time_spent" (total seconds across all users) - "session_duration" (average session length in seconds) - "session_count" (average sessions per user per day) - "total_session_count" (total sessions across all users) Default is c("session_count", "session_duration", "time_spent").
regions	Character vector. Region/country codes (e.g., "US", "GB"). Default is "US". Use NULL for all regions.
time_period	Character string. Session metrics time period. Options: "day", "week". Default is "week". Returns averaged session metrics for each period within a month.
date_granularity	Character string. Aggregate data by granularity. Options: "daily", "weekly", "monthly". Default is "monthly". Note: "daily" granularity may not be supported by the API for all apps; use "weekly" or "monthly" if you receive empty results with "daily".
os	Character string. Filter by platform for unified apps. Options: "ios", "android", or NULL for both. Default is NULL.
breakdown	Character string. Fields for data aggregation. Options: "unified_app_id", "app_id", "region". Default is "unified_app_id".

auth_token	Character string. Your Sensor Tower API token. Defaults to environment variable SENSORTOWER_AUTH_TOKEN.
verbose	Logical. If TRUE, prints progress messages.

Value

A [tibble][tibble::tibble] with session metrics including: - **unified_app_id** or **app_id**: The app identifier - **date**: Date of the data point - **time_spent**: Average seconds spent per user per day - **total_time_spent**: Total seconds across all users - **session_duration**: Average session length in seconds - **session_count**: Average sessions per user per day - **total_session_count**: Total session count across all users

Data Availability

- Data is available from 2021-01-01 onward - Session metrics require Usage Intelligence subscription - Maximum 100 apps per request

See Also

[st_retention()] for retention metrics, [st_demographics()] for user demographics, [st_batch_metrics()] for MAU/DAU/WAU metrics

Examples

```
## Not run:
# Get session metrics for a unified app
sessions <- st_session_metrics(
  unified_app_id = "5fbc3849d0b8414136857afc",
  start_date = "2024-01-01",
  end_date = "2024-12-01"
)

# Get specific metrics with weekly granularity
sessions <- st_session_metrics(
  unified_app_id = "5fbc3849d0b8414136857afc",
  start_date = "2024-01-01",
  end_date = "2024-03-01",
  metrics = c("session_count", "session_duration"),
  date_granularity = "weekly"
)

# Get session data for Android app directly
sessions <- st_session_metrics(
  android_app_id = "com.example.app",
  start_date = "2024-01-01",
  end_date = "2024-06-01"
)

## End(Not run)
```

st_test_filter *Test a Custom Filter ID*

Description

Tests whether a custom filter ID works with the Sensor Tower API by making a minimal test request. This helps verify that the filter exists and is accessible with your authentication.

Usage

```
st_test_filter(filter_id, os = "ios", verbose = TRUE, auth_token = NULL)
```

Arguments

filter_id	Character string. The filter ID to test
os	Character string. Operating system to test with. One of "ios", "android", or "unified". Defaults to "ios".
verbose	Logical. Whether to print detailed test results. Defaults to TRUE.
auth_token	Optional. Character string. Your Sensor Tower API token. Defaults to environment variable SENSORTOWER_AUTH_TOKEN.

Value

List with test results including success status and any error messages

Examples

```
## Not run:
# Test a filter ID
result <- st_test_filter("687df26ac5a19ebcfe817d7f")

# Test silently
result <- st_test_filter("687df26ac5a19ebcfe817d7f", verbose = FALSE)

# Test with different OS
result <- st_test_filter("687df26ac5a19ebcfe817d7f", os = "unified")

## End(Not run)
```

st_yoy_metrics *Year-over-Year Metrics Comparison*

Description

Fetches metrics for the same date range across multiple years for year-over-year comparisons. Allows flexible date ranges and supports all available metrics including revenue, downloads, and active users.

Usage

```
st_yoy_metrics(
  os,
  unified_app_id = NULL,
  ios_app_id = NULL,
  android_app_id = NULL,
  publisher_id = NULL,
  years = NULL,
  period_start,
  period_end,
  metrics = c("revenue", "downloads"),
  countries,
  cache_dir = NULL,
  auth_token = Sys.getenv("SENSORTOWER_AUTH_TOKEN"),
  verbose = TRUE,
  granularity,
  use_single_fetch = TRUE
)
```

Arguments

os	Character. Required. Operating system: "ios", "android", or "unified". This determines which platform's data is returned.
unified_app_id	Character vector. Sensor Tower unified app ID(s). Must be 24-character hex format (e.g., "5ba4585f539ce75b97db6bcb").
ios_app_id	Character vector. iOS app ID(s) (e.g., "1234567890").
android_app_id	Character vector. Android package name(s) (e.g., "com.example.app").
publisher_id	Character vector. Publisher ID(s) (alternative to app IDs).
years	Integer vector. Years to compare (e.g., c(2022, 2023, 2024)). If NULL, uses current year and previous year.
period_start	Character string or Date. Start of the comparison period. Can be "MM-DD" format (e.g., "01-01" for Jan 1) or a full date. If a full date is provided, only the month and day are used.
period_end	Character string or Date. End of the comparison period. Can be "MM-DD" format (e.g., "03-31" for Mar 31) or a full date. If a full date is provided, only the month and day are used.

metrics	Character vector. Metrics to fetch. Supports "revenue", "downloads", "dau", "wau", and "mau". Default is both revenue and downloads.
countries	Character vector. Country codes (e.g., "US", "GB", "JP"). Required.
cache_dir	Character. Directory for caching API responses (optional).
auth_token	Character string. Sensor Tower API token.
verbose	Logical. Print progress messages.
granularity	Character. Date granularity for the data (e.g., "daily", "monthly").
use_single_fetch	Logical. If TRUE, uses a single API call to fetch all data. Defaults to TRUE for efficiency.

Details

This function is designed for year-over-year comparisons:

- **Flexible date ranges**: Compare any period (e.g., Q1, specific months, custom ranges) - **Multiple years**: Compare across 2+ years in a single call - **Smart date handling**: Automatically handles leap years and invalid dates - **YoY calculations**: Includes both percentage and absolute change - **Caching**: Reuses cached data to minimize API calls

The function will apply the same calendar period (month/day range) to each specified year, making it easy to compare seasonal trends, campaign periods, or any custom date range across years.

Value

A tibble in tidy/long format with columns: - ‘app_id‘: The app ID used for fetching data - ‘app_id_type‘: Type of app ID ("ios", "android", or "unified") - ‘entity_id‘: App or publisher ID - ‘entity_name‘: App or publisher name - ‘entity_type‘: "app" or "publisher" - ‘year‘: Year of the data - ‘date_start‘: Start date of the period (YYYY-MM-DD) - ‘date_end‘: End date of the period (YYYY-MM-DD) - ‘country‘: Country code - ‘metric‘: The metric name (e.g., "revenue", "downloads", "dau") - ‘value‘: Metric value (units depend on metric type) - ‘yoy_change‘: Year-over-year change (percentage) - ‘yoy_change_absolute‘: Year-over-year change (absolute value)

Examples

```
## Not run:
# Compare Q1 performance across years
q1_comparison <- st_yoy_metrics(
  os = "ios",
  ios_app_id = "553834731",  # Candy Crush iOS
  years = c(2022, 2023, 2024),
  period_start = "01-01",
  period_end = "03-31",
  countries = "US",
  metrics = c("revenue", "downloads")
)

# Compare holiday season (Nov-Dec) across years
holiday_comparison <- st_yoy_metrics(
  os = "unified",
```

```
  unified_app_id = "5ba4585f539ce75b97db6bcb",
  years = c(2021, 2022, 2023),
  period_start = "11-01",
  period_end = "12-31",
  countries = c("US", "GB", "JP"),
  metrics = c("revenue", "downloads", "dau")
)

# Compare specific campaign period using full dates
campaign_comparison <- st_yoy_metrics(
  os = "android",
  android_app_id = "com.king.candycrushsaga",
  years = NULL,  # Uses current and previous year
  period_start = as.Date("2024-02-14"),  # Valentine's campaign
  period_end = as.Date("2024-02-28"),
  countries = c("US", "GB", "JP"),
  metrics = c("revenue", "downloads", "dau", "wau")
)

## End(Not run)
```

Index