

# Package: ankiR (via r-universe)

May 19, 2026

**Title** Read and Analyze 'Anki' Flashcard Databases

**Version** 0.6.6

**Description** Comprehensive toolkit for reading and analyzing 'Anki' flashcard collection databases. Provides functions to access notes, cards, decks, note types, and review logs with a tidy interface. Features extensive analytics including retention rates, learning curves, forgetting curve fitting, and review patterns. Supports 'FSRS' (Free Spaced Repetition Scheduler) analysis with stability, difficulty, retrievability metrics, parameter comparison, and workload predictions. Includes visualization functions, comparative analysis, time-based analytics, card quality assessment, sibling card analysis, interference detection, predictive features, session simulation, and an interactive Shiny dashboard. Academic/exam preparation tools for medical students and board exam preparation. Export capabilities include CSV, Org-mode, Markdown, SuperMemo, Mochi, Obsidian SR, and JSON formats with progress reports.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Imports** rlang, jsonlite, DBI, RSQLite, tibble, scales

**Suggests** testthat (>= 3.0.0), knitr, rmarkdown, ggplot2, withr, dplyr, shiny

**Config/testthat/edition** 3

**URL** <https://github.com/chrislongros/ankiR>

**BugReports** <https://github.com/chrislongros/ankiR/issues>

**VignetteBuilder** knitr

**Depends** R (>= 4.1.0)

**Language** en-US

**Repository** <https://chrislongros.r-universe.dev>

**Date/Publication** 2026-04-19 12:37:10 UTC

**RemoteUrl** <https://github.com/chrislongros/ankiR>

**RemoteRef** v0.6.6

**RemoteSha** 8d28242f43468228376459deba340033b062fa44

## Contents

analyze_addon_import . . . . .	5
anki_ab_comparison . . . . .	6
anki_backlog_calculator . . . . .	7
anki_base_path . . . . .	7
anki_benchmark . . . . .	8
anki_best_review_times . . . . .	8
anki_buried . . . . .	9
anki_burnout_detection . . . . .	10
anki_card_complexity . . . . .	11
anki_card_content . . . . .	11
anki_card_recommendations . . . . .	12
anki_cards . . . . .	13
anki_cards_fsrs . . . . .	13
anki_cards_full . . . . .	14
anki_cohort_analysis . . . . .	15
anki_collection . . . . .	16
anki_compare_by_age . . . . .	17
anki_compare_deck_difficulty . . . . .	17
anki_compare_decks . . . . .	18
anki_compare_forecasts . . . . .	19
anki_compare_groups . . . . .	19
anki_compare_periods . . . . .	20
anki_consistency . . . . .	21
anki_coverage_analysis . . . . .	22
anki_dashboard . . . . .	23
anki_db_path . . . . .	23
anki_decks . . . . .	24
anki_due . . . . .	24
anki_empty_cards . . . . .	25
anki_exam_readiness . . . . .	26
anki_export_importable . . . . .	27
anki_export_revlog . . . . .	27
anki_field_contents . . . . .	28
anki_find_similar . . . . .	29
anki_fit_forgetting_curve . . . . .	30
anki_forecast . . . . .	31
anki_forecast_enhanced . . . . .	31
anki_forecast_monte_carlo . . . . .	32

anki_gamification . . . . .	33
anki_health_check . . . . .	34
anki_heatmap_data . . . . .	35
anki_interference_analysis . . . . .	35
anki_learning_curve . . . . .	36
anki_learning_efficiency . . . . .	37
anki_learning_velocity . . . . .	38
anki_leeches . . . . .	38
anki_long_cards . . . . .	39
anki_mature . . . . .	40
anki_media_list . . . . .	40
anki_media_missing . . . . .	41
anki_media_path . . . . .	42
anki_media_stats . . . . .	42
anki_media_unused . . . . .	43
anki_models . . . . .	43
anki_monthly_summary . . . . .	44
anki_new . . . . .	45
anki_notes . . . . .	45
anki_plot_difficulty . . . . .	46
anki_plot_forecast . . . . .	47
anki_plot_forgetting_curve . . . . .	47
anki_plot_heatmap . . . . .	48
anki_plot_hours . . . . .	49
anki_plot_intervals . . . . .	49
anki_plot_monte_carlo . . . . .	50
anki_plot_retention . . . . .	51
anki_plot_stability . . . . .	51
anki_plot_weekdays . . . . .	52
anki_profiles . . . . .	53
anki_progress_report . . . . .	53
anki_quality_report . . . . .	54
anki_quick_summary . . . . .	55
anki_report . . . . .	55
anki_response_time . . . . .	56
anki_response_time_outliers . . . . .	57
anki_retention_by_type . . . . .	57
anki_retention_rate . . . . .	58
anki_retention_stability . . . . .	59
anki_review_quality . . . . .	59
anki_revlog . . . . .	60
anki_roi_analysis . . . . .	61
anki_schema_version . . . . .	62
anki_search . . . . .	62
anki_search_enhanced . . . . .	63
anki_session_analysis . . . . .	65
anki_session_stats . . . . .	65
anki_sibling_analysis . . . . .	66

anki_similar_cards . . . . .	67
anki_simulate_session . . . . .	67
anki_stats_daily . . . . .	68
anki_stats_deck . . . . .	69
anki_streak . . . . .	69
anki_streak_analytics . . . . .	70
anki_study_plan . . . . .	71
anki_study_priorities . . . . .	72
anki_summary . . . . .	72
anki_suspended . . . . .	73
anki_tag_analysis . . . . .	74
anki_tags . . . . .	74
anki_time_by_hour . . . . .	75
anki_time_by_weekday . . . . .	75
anki_timestamp_to_date . . . . .	76
anki_timestamp_to_datetime . . . . .	77
anki_to_csv . . . . .	77
anki_to_html . . . . .	78
anki_to_json . . . . .	79
anki_to_markdown . . . . .	80
anki_to_mochi . . . . .	81
anki_to_obsidian_sr . . . . .	81
anki_to_org . . . . .	82
anki_to_supernote . . . . .	83
anki_today . . . . .	84
anki_ts_anomalies . . . . .	84
anki_ts_autocorrelation . . . . .	85
anki_ts_decompose . . . . .	86
anki_ts_forecast . . . . .	86
anki_ts_intervals . . . . .	87
anki_ts_learning . . . . .	88
anki_ts_maturation . . . . .	88
anki_ts_plot . . . . .	89
anki_ts_retention . . . . .	90
anki_ts_stability . . . . .	90
anki_ts_workload . . . . .	91
anki_weak_areas . . . . .	92
anki_workload_projection . . . . .	92
date_to_anki_timestamp . . . . .	93
fsrs_compare_parameters . . . . .	94
fsrs_current_retrievability . . . . .	95
fsrs_decay_distribution . . . . .	95
fsrs_difficulty_distribution . . . . .	96
fsrs_export_reviews . . . . .	97
fsrs_forgetting_index . . . . .	97
fsrs_from_csv . . . . .	98
fsrs_get_parameters . . . . .	99
fsrs_memory_states . . . . .	99

<i>analyze_addon_import</i>	5
fsrs_prepare_for_optimizer . . . . .	100
fsrs_stability_distribution . . . . .	101
import_addon_export . . . . .	101
plot.anki_decomposition . . . . .	102
print.anki_gamification . . . . .	102
print.anki_mc_forecast . . . . .	103
<b>Index</b>	<b>104</b>

---

*analyze\_addon\_import*    *Analyze Imported Addon Data*

---

### **Description**

Run common analyses on data imported from ankiR Stats addon.

### **Usage**

```
analyze_addon_import(data)
```

### **Arguments**

`data`                    Data imported via `import_addon_export()`

### **Value**

A list with analysis results

### **Examples**

```
## Not run:
data <- import_addon_export("ankir_export.json")
analysis <- analyze_addon_import(data)
analysis$retention_trend

## End(Not run)
```

---

anki\_ab\_comparison     *A/B Comparison*

---

## Description

Compare retention, efficiency, and performance across different note types, deck settings, card formats, or custom groups.

## Usage

```
anki_ab_comparison(  
  path = NULL,  
  profile = NULL,  
  by = "note_type",  
  min_reviews = 100  
)
```

## Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Comparison dimension: "note_type", "deck", "tag", "created_period" (default "note_type")
min_reviews	Minimum reviews per group to include (default 100)

## Value

A tibble with comparison statistics

## Examples

```
## Not run:  
# Compare note types  
comp <- anki_ab_comparison(by = "note_type")  
  
# Compare decks  
comp <- anki_ab_comparison(by = "deck")  
  
## End(Not run)
```

---

`anki_backlog_calculator`*Backlog Calculator*

---

**Description**

Calculate how long it would take to clear your review backlog at different review rates, and project backlog growth if you stop studying.

**Usage**

```
anki_backlog_calculator(path = NULL, profile = NULL, new_cards_per_day = 0)
```

**Arguments**

<code>path</code>	Path to collection.anki2 (auto-detected if NULL)
<code>profile</code>	Profile name (first profile if NULL)
<code>new_cards_per_day</code>	Expected new cards per day (0 = maintenance mode)

**Value**

A list with backlog analysis and projections

**Examples**

```
## Not run:  
backlog <- anki_backlog_calculator()  
backlog$current  
backlog$scenarios  
  
## End(Not run)
```

---

`anki_base_path`*Get Anki base path*

---

**Description**

Returns the default Anki2 directory for the current platform.

**Usage**

```
anki_base_path()
```

**Value**

Character string path to Anki2 directory

**Examples**

```
## Not run:
anki_base_path()

## End(Not run)
```

---

anki_benchmark	<i>Benchmark against FSRS averages</i>
----------------	--

---

**Description**

Compare your statistics against typical FSRS users (based on published research).

**Usage**

```
anki_benchmark(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with your stats vs benchmarks

**Examples**

```
## Not run:
anki_benchmark()

## End(Not run)
```

---

anki_best_review_times	<i>Find optimal review times</i>
------------------------	----------------------------------

---

**Description**

Analyzes retention and performance by hour of day and day of week to identify when you learn best.

**Usage**

```
anki_best_review_times(path = NULL, profile = NULL, min_reviews = 50)
```

**Arguments**

path Path to collection.anki2 (auto-detected if NULL)  
 profile Profile name (first profile if NULL)  
 min\_reviews Minimum reviews per time slot for analysis (default 50)

**Value**

A list with optimal time analysis

**Examples**

```
## Not run:
times <- anki_best_review_times()
times$best_hours
times$best_days

## End(Not run)
```

---

anki_buried	<i>Get buried cards</i>
-------------	-------------------------

---

**Description**

Get buried cards

**Usage**

```
anki_buried(path = NULL, profile = NULL, include_notes = FALSE)
```

**Arguments**

path Path to collection.anki2 (auto-detected if NULL)  
 profile Profile name (first profile if NULL)  
 include\_notes If TRUE, join with note data

**Value**

A tibble of buried cards

**Examples**

```
## Not run:
buried <- anki_buried()

## End(Not run)
```

---

`anki_burnout_detection`*Detect Burnout Warning Signs*

---

## Description

Analyzes study patterns to detect potential burnout indicators: declining retention, increasing response time, shorter sessions, more "Again" presses, decreased consistency.

## Usage

```
anki_burnout_detection(  
  path = NULL,  
  profile = NULL,  
  days = 30,  
  baseline_days = 90  
)
```

## Arguments

<code>path</code>	Path to collection.anki2 (auto-detected if NULL)
<code>profile</code>	Profile name (first profile if NULL)
<code>days</code>	Number of days to analyze (default 30)
<code>baseline_days</code>	Days to use as baseline for comparison (default 90)

## Value

A list with burnout indicators and recommendations

## Examples

```
## Not run:  
burnout <- anki_burnout_detection()  
burnout$risk_level  
burnout$indicators  
  
## End(Not run)
```

---

anki\_card\_complexity *Analyze card complexity*

---

**Description**

Measures card complexity based on field content, media usage, etc.

**Usage**

```
anki_card_complexity(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with card complexity metrics

**Examples**

```
## Not run:  
complexity <- anki_card_complexity()  
  
## End(Not run)
```

---

anki\_card\_content *Card Content Analysis*

---

**Description**

Analyze card content: word count, cloze density, complexity score, and correlations with retention.

**Usage**

```
anki_card_content(path = NULL, profile = NULL, sample_size = 1000)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
sample_size	Max cards to analyze (default 1000, NULL for all)

**Value**

A list with content analysis

**Examples**

```
## Not run:  
content <- anki_card_content()  
content$summary  
content$complexity_retention  
  
## End(Not run)
```

---

anki\_card\_recommendations

*Generate card recommendations*

---

**Description**

Generates actionable recommendations for improving your collection: cards to unsuspend, leeches to rewrite, near-duplicates to merge, etc.

**Usage**

```
anki_card_recommendations(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with categorized recommendations

**Examples**

```
## Not run:  
recs <- anki_card_recommendations()  
recs$leeches_to_rewrite  
recs$cards_to_unsuspend  
  
## End(Not run)
```

---

anki_cards	<i>Read cards from Anki collection</i>
------------	--

---

**Description**

Read cards from Anki collection

**Usage**

```
anki_cards(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of cards with columns: cid, nid, did, type, queue, due, ivl, reps, lapses

**Examples**

```
## Not run:  
anki_cards()  
  
## End(Not run)
```

---

anki_cards_fsrs	<i>Read cards with FSRS-6 parameters</i>
-----------------	--

---

**Description**

Extracts cards along with their FSRS (Free Spaced Repetition Scheduler) memory state parameters. FSRS-6 stores stability, difficulty, and a per-card decay parameter in the card's data field.

**Usage**

```
anki_cards_fsrs(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of cards with FSRS parameters:

- cid - Card ID
- nid - Note ID
- did - Deck ID
- type - Card type (0=new, 1=learning, 2=review, 3=relearning)
- queue - Queue (-1=suspended, 0=new, 1=learning, 2=review, 3=day learning, 4=preview)
- due - Due date/position
- ivl - Current interval in days
- reps - Number of reviews
- lapses - Number of lapses
- stability - FSRS stability (S) in days
- difficulty - FSRS difficulty (D), range 1-10
- desired\_retention - Target retention rate
- decay - FSRS-6 decay parameter (w20), typically 0.1-0.8

**Examples**

```
## Not run:
cards_fsrs <- anki_cards_fsrs()
# Calculate current retrievability
cards_fsrs$retrievability <- fsrs_retrievability(
  stability = cards_fsrs$stability,
  days_elapsed = as.numeric(Sys.Date() - as.Date("1970-01-01")) -
    cards_fsrs$due / 86400,
  decay = cards_fsrs$decay
)

## End(Not run)
```

---

anki\_cards\_full

*Get cards with full joined data*

---

**Description**

Returns cards joined with notes, decks, and models for complete analysis.

**Usage**

```
anki_cards_full(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with card data joined with note content, deck names, and model info

**Examples**

```
## Not run:
cards_full <- anki_cards_full()

## End(Not run)
```

---

anki\_cohort\_analysis *Cohort Analysis (Vintage Analysis)*

---

**Description**

Compare card performance by when they were added. Cards added in the same period form a "cohort" and their learning outcomes are compared.

**Usage**

```
anki_cohort_analysis(
  path = NULL,
  profile = NULL,
  cohort_size = "month",
  min_cards = 20
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
cohort_size	Size of each cohort: "week", "month", "quarter" (default "month")
min_cards	Minimum cards per cohort to include (default 20)

**Value**

A tibble with cohort statistics

### Examples

```
## Not run:
cohorts <- anki_cohort_analysis()
cohorts

## End(Not run)
```

---

anki_collection	<i>Open an Anki collection</i>
-----------------	--------------------------------

---

### Description

Opens an Anki collection database and returns an object with methods to access notes, cards, decks, note types, and review logs.

### Usage

```
anki_collection(path = NULL, profile = NULL)
```

### Arguments

path	Path to collection.anki2 file. If NULL, auto-detected from default Anki location.
profile	Profile name. If NULL, uses first available profile.

### Value

An anki\_collection object with methods:

- notes() - Get all notes
- cards() - Get all cards
- revlog() - Get review log
- decks() - Get deck information
- models() - Get note types (models)
- tables() - List all tables
- close() - Close database connection

### Examples

```
## Not run:
col <- anki_collection()
col$notes()
col$decks()
col$close()

## End(Not run)
```

---

anki\_compare\_by\_age     *Compare retention by card age*

---

**Description**

Analyze how retention varies by how long you've been studying cards.

**Usage**

```
anki_compare_by_age(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with retention by card age

**Examples**

```
## Not run:  
anki_compare_by_age()  
  
## End(Not run)
```

---

anki\_compare\_deck\_difficulty  
                          *Compare performance by deck difficulty*

---

**Description**

Compare performance by deck difficulty

**Usage**

```
anki_compare_deck_difficulty(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble ranking decks by difficulty metrics

**Examples**

```
## Not run:  
anki_compare_deck_difficulty()  
  
## End(Not run)
```

---

anki_compare_decks	<i>Compare statistics between decks</i>
--------------------	---

---

**Description**

Side-by-side comparison of deck statistics.

**Usage**

```
anki_compare_decks(path = NULL, profile = NULL, decks = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
decks	Optional vector of deck names to compare (NULL for all)

**Value**

A tibble with comparative statistics

**Examples**

```
## Not run:  
anki_compare_decks()  
anki_compare_decks(decks = c("Medical", "Anatomy"))  
  
## End(Not run)
```

---

anki\_compare\_forecasts      *Compare Forecast Methods*

---

**Description**

Run multiple forecasting methods and compare their predictions.

**Usage**

```
anki_compare_forecasts(  
  path = NULL,  
  profile = NULL,  
  days_ahead = 30,  
  methods = c("holt", "arima", "seasonal", "monte_carlo")  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days_ahead	Number of days to forecast (default 30)
methods	Methods to compare (default: all available)

**Value**

A list with comparison results

**Examples**

```
## Not run:  
comp <- anki_compare_forecasts(days_ahead = 14)  
comp$summary  
  
## End(Not run)
```

---

anki\_compare\_groups      *Compare Two Specific Groups*

---

**Description**

Detailed statistical comparison between two groups (decks, note types, etc.)

**Usage**

```
anki_compare_groups(path = NULL, profile = NULL, group_a, group_b, by = "deck")
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
group_a	First group name/pattern
group_b	Second group name/pattern
by	Comparison dimension: "note_type", "deck", "tag"

**Value**

A list with detailed comparison statistics

**Examples**

```
## Not run:
comp <- anki_compare_groups("Basic", "Cloze", by = "note_type")
comp$winner
comp$differences

## End(Not run)
```

---

anki\_compare\_periods    *Compare two time periods*

---

**Description**

Compare study statistics between two time periods.

**Usage**

```
anki_compare_periods(
  path = NULL,
  profile = NULL,
  period1 = NULL,
  period2 = NULL,
  period_names = c("Period 1", "Period 2")
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
period1	First period as c(start_date, end_date)
period2	Second period as c(start_date, end_date)
period_names	Names for the periods (default: "Period 1", "Period 2")

**Value**

A tibble comparing the two periods

**Examples**

```
## Not run:  
# Compare this month vs last month  
anki_compare_periods(  
  period1 = c("2024-01-01", "2024-01-31"),  
  period2 = c("2024-02-01", "2024-02-29"),  
  period_names = c("January", "February")  
)  
  
## End(Not run)
```

---

anki_consistency	<i>Analyze study consistency</i>
------------------	----------------------------------

---

**Description**

Measures how consistent your study habits are.

**Usage**

```
anki_consistency(path = NULL, profile = NULL, days = 90)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to analyze

**Value**

A list with consistency metrics

**Examples**

```
## Not run:  
anki_consistency()  
  
## End(Not run)
```

---

`anki_coverage_analysis`*Analyze topic coverage*

---

### Description

Shows percentage complete, mature, and retained by topic (tag or subdeck).

### Usage

```
anki_coverage_analysis(  
  path = NULL,  
  profile = NULL,  
  by = "tag",  
  pattern = NULL,  
  min_cards = 10  
)
```

### Arguments

<code>path</code>	Path to collection.anki2 (auto-detected if NULL)
<code>profile</code>	Profile name (first profile if NULL)
<code>by</code>	Analysis type: "tag" or "deck" (default "tag")
<code>pattern</code>	Optional pattern to filter tags/decks
<code>min_cards</code>	Minimum cards to include in analysis (default 10)

### Value

A tibble with coverage analysis per topic

### Examples

```
## Not run:  
coverage <- anki_coverage_analysis()  
coverage <- anki_coverage_analysis(pattern = "Anatomy")  
  
## End(Not run)
```

---

anki_dashboard	<i>Launch interactive Anki dashboard</i>
----------------	--

---

**Description**

Opens a Shiny dashboard with comprehensive collection analytics.

**Usage**

```
anki_dashboard(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

Opens a Shiny app in the browser

**Examples**

```
## Not run:  
anki_dashboard()  
  
## End(Not run)
```

---

anki_db_path	<i>Get path to Anki database</i>
--------------	----------------------------------

---

**Description**

Returns the full path to an Anki collection database file.

**Usage**

```
anki_db_path(profile = NULL, base_path = NULL)
```

**Arguments**

profile	Profile name. If NULL, uses the first available profile.
base_path	Path to Anki2 directory (auto-detected if NULL)

**Value**

Character string path to collection.anki2 or collection.anki21

**Examples**

```
## Not run:
anki_db_path()
anki_db_path("User 1")

## End(Not run)
```

---

anki_decks	<i>Read decks from Anki collection</i>
------------	--

---

**Description**

Read decks from Anki collection

**Usage**

```
anki_decks(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of decks with columns: did, name

**Examples**

```
## Not run:
anki_decks()

## End(Not run)
```

---

anki_due	<i>Get cards due for review</i>
----------	---------------------------------

---

**Description**

Get cards due for review

**Usage**

```
anki_due(path = NULL, profile = NULL, days_ahead = 0)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days_ahead	Number of days to look ahead (default 0 = today only)

**Value**

A tibble of due cards

**Examples**

```
## Not run:
due_today <- anki_due()
due_week <- anki_due(days_ahead = 7)

## End(Not run)
```

---

anki\_empty\_cards      *Find cards with empty fields*

---

**Description**

Identifies cards that have empty required fields.

**Usage**

```
anki_empty_cards(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of cards with empty fields

**Examples**

```
## Not run:
empty <- anki_empty_cards()

## End(Not run)
```

---

anki\_exam\_readiness    *Track exam readiness*

---

### Description

Projects whether you'll complete cards before an exam, at what retention level. Useful for medical board exam preparation.

### Usage

```
anki_exam_readiness(  
  path = NULL,  
  profile = NULL,  
  target_date,  
  deck_pattern = NULL,  
  new_cards_per_day = 20,  
  target_retention = 0.9  
)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
target_date	Exam date (Date or character "YYYY-MM-DD")
deck_pattern	Pattern to match deck names (e.g., "Step1", "USMLE")
new_cards_per_day	Expected new cards per day (default 20)
target_retention	Target retention rate (default 0.90)

### Value

A list with exam readiness analysis

### Examples

```
## Not run:  
readiness <- anki_exam_readiness(target_date = "2024-06-15", deck_pattern = "Step1")  
  
## End(Not run)
```

---

anki\_export\_importable      *Export to Anki-importable format*

---

**Description**

Export to Anki-importable format

**Usage**

```
anki_export_importable(data, file, tags = NULL)
```

**Arguments**

data	Data frame with front and back columns
file	Output file path
tags	Optional tags

**Value**

Invisibly returns number of cards

**Examples**

```
## Not run:  
cards <- data.frame(front = c("Q1", "Q2"), back = c("A1", "A2"))  
anki_export_importable(cards, "new_cards.txt")  
  
## End(Not run)
```

---

anki\_export\_revlog      *Export review history*

---

**Description**

Exports the full review log to CSV.

**Usage**

```
anki_export_revlog(  
  file,  
  path = NULL,  
  profile = NULL,  
  include_card_info = FALSE  
)
```

**Arguments**

file	Output file path
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
include_card_info	If TRUE, join with card data

**Value**

Invisibly returns the exported data

**Examples**

```
## Not run:
anki_export_revlog("my_reviews.csv")

## End(Not run)
```

---

anki\_field\_contents *Parse note fields into columns*

---

**Description**

Splits the field content (separated by \x1f) into separate columns.

**Usage**

```
anki_field_contents(path = NULL, profile = NULL, model_id = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
model_id	Optional model ID to filter notes. If NULL, uses first model.

**Value**

A tibble with nid and separate columns for each field

**Examples**

```
## Not run:
fields <- anki_field_contents()

## End(Not run)
```

---

anki\_find\_similar      *Search cards by content similarity*

---

## Description

Find cards with similar content to a given card or text.

## Usage

```
anki_find_similar(  
  query,  
  path = NULL,  
  profile = NULL,  
  n = 10,  
  method = "tfidf",  
  within_deck = FALSE  
)
```

## Arguments

query	Either a card ID or text to search for similar cards
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
n	Number of similar cards to return (default 10)
method	Similarity method: "tfidf", "jaccard", or "ngram"
within_deck	Only search within same deck as query card

## Value

A tibble with similar cards and similarity scores

## Examples

```
## Not run:  
# Find cards similar to card ID 1234567890  
anki_find_similar(1234567890)  
  
# Find cards similar to specific text  
anki_find_similar("mitochondria powerhouse")  
  
## End(Not run)
```

---

`anki_fit_forgetting_curve`*Fit forgetting curve from review data*

---

## Description

Fits a forgetting curve to actual review data and compares it to the theoretical FSRS curve. Can analyze individual cards or aggregate data.

## Usage

```
anki_fit_forgetting_curve(  
  path = NULL,  
  profile = NULL,  
  cid = NULL,  
  min_reviews = 5,  
  max_cards = 1000  
)
```

## Arguments

<code>path</code>	Path to collection.anki2 (auto-detected if NULL)
<code>profile</code>	Profile name (first profile if NULL)
<code>cid</code>	Optional card ID for individual card analysis
<code>min_reviews</code>	Minimum reviews required for analysis (default 5)
<code>max_cards</code>	Maximum cards to analyze for aggregate (default 1000)

## Value

A list with fitted curve parameters and comparison data

## Examples

```
## Not run:  
curve <- anki_fit_forgetting_curve()  
plot(curve$data$days_elapsed, curve$data$observed_retention, type = "p")  
lines(curve$data$days_elapsed, curve$data$fitted_retention, col = "blue")  
  
## End(Not run)
```

---

anki_forecast	<i>Get forecast of upcoming reviews</i>
---------------	---

---

**Description**

Predicts how many reviews will be due each day.

**Usage**

```
anki_forecast(path = NULL, profile = NULL, days = 30)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to forecast (default 30)

**Value**

A tibble with daily forecast

**Examples**

```
## Not run:  
forecast <- anki_forecast()  
  
## End(Not run)
```

---

anki_forecast_enhanced	<i>Enhanced Time Series Forecasting</i>
------------------------	---

---

**Description**

Improved forecasting with ARIMA, seasonal patterns, and workload ceilings.

**Usage**

```
anki_forecast_enhanced(  
  path = NULL,  
  profile = NULL,  
  metric = "reviews",  
  days_ahead = 30,  
  method = "auto",  
  confidence = 0.95,  
  workload_ceiling = NULL  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
metric	Metric to forecast: "reviews", "time", "retention", "cards_learned"
days_ahead	Number of days to forecast (default 30)
method	Forecasting method: "auto", "arima", "ets", "holt", "seasonal"
confidence	Confidence level for prediction intervals (default 0.95)
workload_ceiling	Maximum daily workload (NULL for none)

**Value**

A tibble with forecast results

**Examples**

```
## Not run:
fc <- anki_forecast_enhanced("reviews", days_ahead = 30)
plot(fc$date, fc$forecast, type = "l")

## End(Not run)
```

---

anki\_forecast\_monte\_carlo

*Monte Carlo Forecasting*

---

**Description**

Forecast future reviews using Monte Carlo simulation with bootstrapping. Unlike statistical methods (ARIMA, Holt-Winters), this approach:

- Makes no distributional assumptions
- Preserves day-of-week patterns naturally
- Handles irregular study habits (missed days, catch-up sessions)
- Provides empirical confidence intervals

**Usage**

```
anki_forecast_monte_carlo(
  path = NULL,
  profile = NULL,
  days_ahead = 30,
  n_sim = 1000,
  method = "weekday",
  block_size = 7,
  include_trend = TRUE,
  seed = NULL
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days_ahead	Number of days to forecast (default 30)
n_sim	Number of simulations (default 1000)
method	Bootstrap method: "weekday" (preserves day-of-week), "block" (preserves sequences), "simple" (iid sampling)
block_size	Block size for block bootstrap (default 7)
include_trend	Whether to include trend component (default TRUE)
seed	Random seed for reproducibility (NULL for random)

**Value**

A list with forecast distribution, summary statistics, and simulation data

**Examples**

```
## Not run:
mc <- anki_forecast_monte_carlo(days_ahead = 30, n_sim = 1000)

# Summary
mc$summary

# Probability of >100 reviews on day 7
mc$prob_above(day = 7, threshold = 100)

# Full simulation matrix
dim(mc$simulations) # n_sim x days_ahead

## End(Not run)
```

---

anki\_gamification      *Gamification Stats*

---

**Description**

Calculate XP, level, achievements, and progress toward goals based on your Anki review history. Makes studying more engaging!

**Usage**

```
anki_gamification(
  path = NULL,
  profile = NULL,
  xp_per_review = 10,
  xp_per_correct = 5,
  xp_per_streak = 25
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
xp_per_review	XP awarded per review (default 10)
xp_per_correct	Bonus XP for correct answer (default 5)
xp_per_streak	Bonus XP per day of streak (default 25)

**Value**

A list with XP, level, achievements, and stats

**Examples**

```
## Not run:
stats <- anki_gamification()
stats$level
stats$achievements

## End(Not run)
```

---

anki\_health\_check      *Collection health check*

---

**Description**

Performs a comprehensive health check on your Anki collection, identifying common issues and problems.

**Usage**

```
anki_health_check(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with health check results

**Examples**

```
## Not run:
health <- anki_health_check()

## End(Not run)
```

---

anki_heatmap_data	<i>Get review data formatted for calendar heatmaps</i>
-------------------	--

---

**Description**

Get review data formatted for calendar heatmaps

**Usage**

```
anki_heatmap_data(path = NULL, profile = NULL, year = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
year	Optional year to filter (NULL for all)

**Value**

A tibble with date and count columns suitable for heatmap visualization

**Examples**

```
## Not run:  
heatmap_data <- anki_heatmap_data()  
# Use with ggplot2:  
# ggplot(heatmap_data, aes(week, weekday, fill = reviews)) + geom_tile()  
  
## End(Not run)
```

---

anki_interference_analysis	<i>Detect card interference</i>
----------------------------	---------------------------------

---

**Description**

Find cards that are often confused with each other, based on similar failure patterns or content similarity combined with poor retention.

**Usage**

```
anki_interference_analysis(  
  path = NULL,  
  profile = NULL,  
  min_lapses = 3,  
  time_window_days = 7,  
  max_pairs = 50  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
min_lapses	Minimum lapses for a card to be considered (default 3)
time_window_days	Window to look for related failures (default 7)
max_pairs	Maximum pairs to return (default 50)

**Value**

A tibble of potentially interfering card pairs

**Examples**

```
## Not run:
interference <- anki_interference_analysis()

## End(Not run)
```

---

anki\_learning\_curve *Track card learning progression over time*

---

**Description**

Shows how cards have progressed through intervals over time.

**Usage**

```
anki_learning_curve(
  path = NULL,
  profile = NULL,
  card_ids = NULL,
  n_cards = 100
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
card_ids	Optional vector of card IDs to track (NULL for sample)
n_cards	Number of cards to sample if card_ids is NULL

**Value**

A tibble with review history per card

**Examples**

```
## Not run:  
curve <- anki_learning_curve()  
  
## End(Not run)
```

---

```
anki_learning_efficiency  
  Analyze learning efficiency
```

---

**Description**

Calculate how much "real learning" is happening vs time spent on failed reviews. Measures the ratio of successful retention to total study time.

**Usage**

```
anki_learning_efficiency(  
  path = NULL,  
  profile = NULL,  
  days = 30,  
  by_deck = FALSE  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to analyze (default 30, NULL for all)
by_deck	If TRUE, calculate efficiency per deck

**Value**

A tibble with efficiency metrics

**Examples**

```
## Not run:  
eff <- anki_learning_efficiency()  
eff <- anki_learning_efficiency(days = 90, by_deck = TRUE)  
  
## End(Not run)
```

---

 anki\_learning\_velocity

*Learning Velocity Analysis*


---

### Description

Track learning rate over time: cards learned per day, time to graduation, acceleration or deceleration of learning.

### Usage

```
anki_learning_velocity(path = NULL, profile = NULL, period = "month")
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
period	Analysis period: "week", "month", "quarter", "all" (default "month")

### Value

A list with velocity metrics and trends

### Examples

```
## Not run:
velocity <- anki_learning_velocity()
velocity$current
velocity$trend

## End(Not run)
```

---

 anki\_leeches

*Find leech cards (high lapse count)*


---

### Description

Leeches are cards that you keep forgetting. By default, Anki marks cards as leeches after 8 lapses.

### Usage

```
anki_leeches(path = NULL, profile = NULL, threshold = 8, include_notes = FALSE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
threshold	Minimum number of lapses to consider a leech (default 8)
include_notes	If TRUE, join with note data

**Value**

A tibble of leech cards ordered by lapses

**Examples**

```
## Not run:  
leeches <- anki_leeches()  
leeches <- anki_leeches(threshold = 5)  
  
## End(Not run)
```

---

anki_long_cards	<i>Find cards with very long content</i>
-----------------	--

---

**Description**

Identifies cards that might be too complex.

**Usage**

```
anki_long_cards(path = NULL, profile = NULL, threshold = 2000)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
threshold	Character count threshold (default 2000)

**Value**

A tibble of long cards

**Examples**

```
## Not run:  
long_cards <- anki_long_cards()  
  
## End(Not run)
```

---

anki_mature	<i>Get mature cards (interval &gt;= 21 days)</i>
-------------	--

---

**Description**

Get mature cards (interval >= 21 days)

**Usage**

```
anki_mature(path = NULL, profile = NULL, min_interval = 21)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
min_interval	Minimum interval to consider mature (default 21)

**Value**

A tibble of mature cards

**Examples**

```
## Not run:
mature <- anki_mature()

## End(Not run)
```

---

anki_media_list	<i>List media files in collection</i>
-----------------	---------------------------------------

---

**Description**

Returns all media files stored in the collection's media folder.

**Usage**

```
anki_media_list(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with filename, size, and extension

**Examples**

```
## Not run:  
media <- anki_media_list()  
  
## End(Not run)
```

---

anki\_media\_missing      *Find missing media references*

---

**Description**

Identifies media files referenced in notes but not present in the media folder.

**Usage**

```
anki_media_missing(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with missing filenames and the notes referencing them

**Examples**

```
## Not run:  
missing <- anki_media_missing()  
  
## End(Not run)
```

---

anki_media_path	<i>Get media folder path</i>
-----------------	------------------------------

---

**Description**

Get media folder path

**Usage**

```
anki_media_path(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

Character string path to collection.media folder

**Examples**

```
## Not run:  
anki_media_path()  
  
## End(Not run)
```

---

anki_media_stats	<i>Get media statistics</i>
------------------	-----------------------------

---

**Description**

Get media statistics

**Usage**

```
anki_media_stats(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with media statistics

**Examples**

```
## Not run:  
stats <- anki_media_stats()  
  
## End(Not run)
```

---

anki_media_unused	<i>Find unused media files</i>
-------------------	--------------------------------

---

**Description**

Identifies media files that are not referenced in any note.

**Usage**

```
anki_media_unused(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of unused media files

**Examples**

```
## Not run:  
unused <- anki_media_unused()  
  
## End(Not run)
```

---

anki_models	<i>Read note types (models) from Anki collection</i>
-------------	--

---

**Description**

Read note types (models) from Anki collection

**Usage**

```
anki_models(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of note types with columns: mid, name, flds (list of field names), tmpls (list of template names)

**Examples**

```
## Not run:  
anki_models()  
  
## End(Not run)
```

---

anki\_monthly\_summary *Get monthly summary statistics*

---

**Description**

Get monthly summary statistics

**Usage**

```
anki_monthly_summary(path = NULL, profile = NULL, months = 12)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
months	Number of months to include (default 12)

**Value**

A tibble with monthly statistics

**Examples**

```
## Not run:  
anki_monthly_summary()  
  
## End(Not run)
```

---

anki_new	<i>Get new cards (never reviewed)</i>
----------	---------------------------------------

---

**Description**

Get new cards (never reviewed)

**Usage**

```
anki_new(path = NULL, profile = NULL, deck = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
deck	Optional deck name to filter

**Value**

A tibble of new cards

**Examples**

```
## Not run:  
new_cards <- anki_new()  
  
## End(Not run)
```

---

anki_notes	<i>Read notes from Anki collection</i>
------------	--

---

**Description**

Read notes from Anki collection

**Usage**

```
anki_notes(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of notes with columns: nid, mid, tags, flds, sfld

**Examples**

```
## Not run:  
anki_notes()  
  
## End(Not run)
```

---

anki\_plot\_difficulty *Plot difficulty distribution*

---

**Description**

Histogram of card difficulties (FSRS).

**Usage**

```
anki_plot_difficulty(path = NULL, profile = NULL, by_deck = FALSE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by_deck	If TRUE, facet by deck

**Value**

A ggplot2 object

**Examples**

```
## Not run:  
anki_plot_difficulty()  
  
## End(Not run)
```

---

anki\_plot\_forecast      *Plot review forecast*

---

**Description**

Shows predicted upcoming review workload.

**Usage**

```
anki_plot_forecast(path = NULL, profile = NULL, days = 30)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to forecast (default 30)

**Value**

A ggplot2 object

**Examples**

```
## Not run:  
anki_plot_forecast()  
  
## End(Not run)
```

---

anki\_plot\_forgetting\_curve  
*Plot forgetting curve comparison*

---

**Description**

Creates a visualization comparing your fitted forgetting curve to FSRS defaults.

**Usage**

```
anki_plot_forgetting_curve(curve_data)
```

**Arguments**

curve_data	Output from anki_fit_forgetting_curve()
------------	---

**Value**

A ggplot object

**Examples**

```
## Not run:
curve <- anki_fit_forgetting_curve()
anki_plot_forgetting_curve(curve)

## End(Not run)
```

---

anki\_plot\_heatmap      *Plot review heatmap calendar*

---

**Description**

Creates a calendar heatmap showing review activity.

**Usage**

```
anki_plot_heatmap(
  path = NULL,
  profile = NULL,
  year = NULL,
  colors = c("#ebedf0", "#9be9a8", "#40c463", "#30a14e", "#216e39")
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
year	Year to display (default: current year)
colors	Vector of colors for the gradient (low to high)

**Value**

A ggplot2 object

**Examples**

```
## Not run:
anki_plot_heatmap()
anki_plot_heatmap(year = 2024)

## End(Not run)
```

---

anki\_plot\_hours      *Plot reviews by hour of day*

---

**Description**

Plot reviews by hour of day

**Usage**

```
anki_plot_hours(path = NULL, profile = NULL, days = 90)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to include

**Value**

A ggplot2 object

**Examples**

```
## Not run:
anki_plot_hours()

## End(Not run)
```

---

anki\_plot\_intervals      *Plot interval distribution*

---

**Description**

Histogram of card intervals.

**Usage**

```
anki_plot_intervals(path = NULL, profile = NULL, log_scale = TRUE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
log_scale	If TRUE, use log scale for x-axis

**Value**

A ggplot2 object

**Examples**

```
## Not run:  
anki_plot_intervals()  
  
## End(Not run)
```

---

anki\_plot\_monte\_carlo *Plot Monte Carlo Forecast*

---

**Description**

Visualize Monte Carlo forecast with confidence bands.

**Usage**

```
anki_plot_monte_carlo(  
  mc_forecast,  
  show_bands = c("95", "80"),  
  show_simulations = 20,  
  cumulative = FALSE  
)
```

**Arguments**

mc_forecast	Output from anki_forecast_monte_carlo()
show_bands	Confidence bands to show: "95", "80", "50", or combinations
show_simulations	Number of individual simulations to overlay (0 for none)
cumulative	Plot cumulative reviews instead of daily

**Value**

A ggplot2 object

**Examples**

```
## Not run:  
mc <- anki_forecast_monte_carlo(days_ahead = 30)  
anki_plot_monte_carlo(mc)  
anki_plot_monte_carlo(mc, cumulative = TRUE)  
  
## End(Not run)
```

---

anki\_plot\_retention    *Plot retention over time*

---

**Description**

Shows how retention rate has changed over time.

**Usage**

```
anki_plot_retention(path = NULL, profile = NULL, days = 90, window = 7)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to include (default 90)
window	Rolling window size in days for smoothing (default 7)

**Value**

A ggplot2 object

**Examples**

```
## Not run:  
anki_plot_retention()  
anki_plot_retention(days = 365, window = 14)  
  
## End(Not run)
```

---

anki\_plot\_stability    *Plot stability distribution*

---

**Description**

Histogram of FSRS stability values.

**Usage**

```
anki_plot_stability(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A ggplot2 object

**Examples**

```
## Not run:  
anki_plot_stability()  
  
## End(Not run)
```

---

anki\_plot\_weekdays      *Plot reviews by day of week*

---

**Description**

Plot reviews by day of week

**Usage**

```
anki_plot_weekdays(path = NULL, profile = NULL, days = 90)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to include

**Value**

A ggplot2 object

**Examples**

```
## Not run:  
anki_plot_weekdays()  
  
## End(Not run)
```

---

anki_profiles	<i>List Anki profiles</i>
---------------	---------------------------

---

**Description**

Returns the names of all Anki profiles found in the Anki2 directory.

**Usage**

```
anki_profiles(base_path = NULL)
```

**Arguments**

base\_path      Path to Anki2 directory (auto-detected if NULL)

**Value**

Character vector of profile names

**Examples**

```
## Not run:  
anki_profiles()  
  
## End(Not run)
```

---

anki_progress_report	<i>Generate progress report</i>
----------------------	---------------------------------

---

**Description**

Generates a shareable HTML or Markdown progress report with charts.

**Usage**

```
anki_progress_report(  
  path = NULL,  
  profile = NULL,  
  output_path = "anki_progress.html",  
  format = "html",  
  period = "month",  
  title = "Anki Progress Report"  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
output_path	Path for output file
format	Output format: "html" or "markdown"
period	Period to analyze: "week", "month", "year"
title	Report title

**Value**

Invisibly returns the report content

**Examples**

```
## Not run:  
anki_progress_report(output_path = "my_progress.html", period = "month")  
  
## End(Not run)
```

---

anki\_quality\_report *Analyze card quality metrics*

---

**Description**

Comprehensive quality analysis combining multiple metrics.

**Usage**

```
anki_quality_report(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with quality metrics

**Examples**

```
## Not run:  
quality <- anki_quality_report()  
  
## End(Not run)
```

---

anki\_quick\_summary      *Quick Collection Summary*

---

**Description**

Get a one-liner overview of your Anki collection.

**Usage**

```
anki_quick_summary(path = NULL, profile = NULL, print = TRUE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
print	Whether to print the summary (default TRUE)

**Value**

Invisibly returns a list with summary stats

**Examples**

```
## Not run:  
anki_quick_summary()  
  
## End(Not run)
```

---

anki\_report      *Generate collection summary report*

---

**Description**

Creates a summary of your Anki collection statistics.

**Usage**

```
anki_report(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with collection statistics

**Examples**

```
## Not run:  
report <- anki_report()  
print(report)  
  
## End(Not run)
```

---

anki\_response\_time     *Analyze response time by card properties*

---

**Description**

Shows how response time varies by difficulty, interval, etc.

**Usage**

```
anki_response_time(path = NULL, profile = NULL, days = 90)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to analyze

**Value**

A tibble with response time analysis

**Examples**

```
## Not run:  
anki_response_time()  
  
## End(Not run)
```

---

anki\_response\_time\_outliers  
*Analyze response time outliers*

---

**Description**

Find reviews with suspicious response times (too fast or too slow).

**Usage**

```
anki_response_time_outliers(  
  path = NULL,  
  profile = NULL,  
  min_time_ms = 500,  
  max_time_ms = 120000  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
min_time_ms	Minimum expected time in ms (default 500)
max_time_ms	Maximum expected time in ms (default 120000 = 2 min)

**Value**

A tibble with outlier reviews

**Examples**

```
## Not run:  
outliers <- anki_response_time_outliers()  
  
## End(Not run)
```

---

anki\_retention\_by\_type  
*Calculate retention by content type*

---

**Description**

Break down retention by card characteristics: cloze vs basic, with/without media, short vs long content.

**Usage**

```
anki_retention_by_type(path = NULL, profile = NULL, days = 30)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to analyze (default 30)

**Value**

A list with retention breakdowns by different content types

**Examples**

```
## Not run:
ret_type <- anki_retention_by_type()
ret_type$by_card_type
ret_type$by_media

## End(Not run)
```

---

anki\_retention\_rate    *Calculate actual retention rate from review history*

---

**Description**

Calculates the proportion of reviews that were successful (not "Again").

**Usage**

```
anki_retention_rate(
  path = NULL,
  profile = NULL,
  days = 30,
  by_deck = FALSE,
  queue_only = TRUE
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to look back (default 30, NULL for all)
by_deck	If TRUE, calculate retention per deck
queue_only	If TRUE (default), only count reviews from the review queue (review_type == 1), matching Anki's own retention definition. Set FALSE to include learning/relearning reviews.

**Value**

A tibble with retention statistics

**Examples**

```
## Not run:  
retention <- anki_retention_rate()  
retention <- anki_retention_rate(days = 90, by_deck = TRUE)  
  
## End(Not run)
```

---

anki\_retention\_stability      *Retention Stability Analysis*

---

**Description**

Analyze how stable your retention is over time.

**Usage**

```
anki_retention_stability(path = NULL, profile = NULL, window = 7)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
window	Rolling window size in days (default 7)

**Value**

A tibble with retention stability metrics

---

anki\_review\_quality      *Review Quality Score*

---

**Description**

Detects low-quality reviews: pattern clicking, rushed reviews, suspicious timing patterns, and other indicators of disengaged studying.

**Usage**

```
anki_review_quality(
  path = NULL,
  profile = NULL,
  days = 30,
  min_time_ms = 800,
  max_time_ms = 60000
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to analyze (default 30)
min_time_ms	Minimum expected review time in ms (default 800)
max_time_ms	Maximum expected review time in ms (default 60000)

**Value**

A list with quality metrics and flagged reviews

**Examples**

```
## Not run:
quality <- anki_review_quality()
quality$score
quality$issues

## End(Not run)
```

---

anki\_revlog

*Read review log from Anki collection*

---

**Description**

Read review log from Anki collection

**Usage**

```
anki_revlog(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of review log entries with columns: rid, cid, ease, ivl, time, review\_date

**Examples**

```
## Not run:  
anki_revlog()  
  
## End(Not run)
```

---

anki_roi_analysis	<i>Calculate spaced repetition ROI</i>
-------------------	--

---

**Description**

Estimate the return on investment of your spaced repetition practice. Calculates knowledge half-life extension per minute of study.

**Usage**

```
anki_roi_analysis(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with ROI analysis

**Examples**

```
## Not run:  
roi <- anki_roi_analysis()  
  
## End(Not run)
```

---

anki\_schema\_version     *Anki Schema Version Detection*

---

### Description

Detect the Anki database schema version to help debug compatibility issues.

### Usage

```
anki_schema_version(path = NULL, profile = NULL)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

### Value

A list with schema information

### Examples

```
## Not run:
schema <- anki_schema_version()

## End(Not run)
```

---

anki\_search     *Search cards like Anki's browser*

---

### Description

Searches cards using a simplified version of Anki's search syntax. Supports: deck:name, tag:name, is:new, is:due, is:suspended, is:buried, is:learn, is:review, prop:ivl>N, prop:lapses>N, prop:reps>N

### Usage

```
anki_search(query, path = NULL, profile = NULL)
```

### Arguments

query	Search query string
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble of matching cards

**Examples**

```
## Not run:
# Find suspended cards
anki_search("is:suspended")

# Find cards in a specific deck
anki_search("deck:Default")

# Find cards with many lapses
anki_search("prop:lapses>5")

# Combine conditions
anki_search("deck:Medical is:review prop:ivl>30")

## End(Not run)
```

---

anki\_search\_enhanced *Enhanced Search for Anki Cards*

---

**Description**

Advanced search functionality supporting additional search operators beyond the basic `anki_search()`.

**Usage**

```
anki_search_enhanced(
  pattern,
  path = NULL,
  profile = NULL,
  case_sensitive = FALSE
)
```

**Arguments**

<code>pattern</code>	Search pattern with support for advanced operators
<code>path</code>	Path to collection.anki2 (auto-detected if NULL)
<code>profile</code>	Profile name (first profile if NULL)
<code>case_sensitive</code>	Whether search is case-sensitive (default FALSE)

**Details**

Supported search operators:

- added:N - Cards added in last N days
- rated:N - Cards rated in last N days
- rated:N:ease - Cards rated with specific ease (1-4) in last N days
- note:type - Cards with specific note type name
- card:N - Card template number (0-indexed)
- ivl:N - Cards with interval  $\geq$  N days
- ivl:<N - Cards with interval  $<$  N days
- prop:ease>N - Cards with ease factor  $>$  N (e.g., prop:ease>2.5)
- prop:lapses>N - Cards with lapses  $>$  N
- prop:reps>N - Cards with reps  $>$  N
- re:pattern - Regex search in card content
- OR - OR operator between terms
- Standard operators: deck:, tag:, is:, flag:

**Value**

A tibble with matching cards

**Examples**

```
## Not run:
# Cards added in last 7 days
anki_search_enhanced("added:7")

# Cards rated "Again" in last 3 days
anki_search_enhanced("rated:3:1")

# Leeches with high lapses
anki_search_enhanced("is:leech prop:lapses>5")

# Regex search
anki_search_enhanced("re:^The\\s+")

# OR search
anki_search_enhanced("deck:German OR deck:Spanish")

## End(Not run)
```

---

anki\_session\_analysis *Analyze study sessions*

---

**Description**

Identifies and analyzes discrete study sessions based on gaps between reviews.

**Usage**

```
anki_session_analysis(  
  path = NULL,  
  profile = NULL,  
  gap_minutes = 30,  
  min_session_reviews = 5  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
gap_minutes	Gap in minutes to consider a new session (default 30)
min_session_reviews	Minimum reviews to count as a session (default 5)

**Value**

A list with session analysis

**Examples**

```
## Not run:  
sessions <- anki_session_analysis()  
  
## End(Not run)
```

---

anki\_session\_stats *Analyze study sessions*

---

**Description**

Identifies study sessions and calculates session statistics. A session is defined as reviews with gaps less than session\_gap minutes.

**Usage**

```
anki_session_stats(path = NULL, profile = NULL, session_gap = 30, days = 90)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
session_gap	Gap in minutes that defines session boundaries (default 30)
days	Number of days to analyze

**Value**

A tibble with session statistics

**Examples**

```
## Not run:
sessions <- anki_session_stats()

## End(Not run)
```

---

anki\_sibling\_analysis *Analyze sibling card effects*

---

**Description**

Analyzes how sibling cards (cards from the same note) affect each other's retention. For example, does seeing the forward card help with the reverse?

**Usage**

```
anki_sibling_analysis(path = NULL, profile = NULL, max_gap_days = 7)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
max_gap_days	Maximum days between sibling reviews to consider related (default 7)

**Value**

A list with sibling analysis

**Examples**

```
## Not run:
siblings <- anki_sibling_analysis()

## End(Not run)
```

---

anki\_similar\_cards     *Find similar/duplicate cards*

---

### Description

Uses text similarity to find potential duplicate cards.

### Usage

```
anki_similar_cards(  
  path = NULL,  
  profile = NULL,  
  threshold = 0.8,  
  max_comparisons = 50000  
)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
threshold	Similarity threshold (0-1, default 0.8)
max_comparisons	Maximum number of comparisons (default 50000)

### Value

A tibble of similar card pairs

### Examples

```
## Not run:  
dupes <- anki_similar_cards(threshold = 0.9)  
  
## End(Not run)
```

---

anki\_simulate\_session     *Simulate a study session*

---

### Description

Given a time budget, predict how many cards you'll review, expected retention, and workload distribution.

**Usage**

```
anki_simulate_session(
  path = NULL,
  profile = NULL,
  minutes = 30,
  include_new = TRUE
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
minutes	Available study time in minutes (default 30)
include_new	Whether to include new cards (default TRUE)

**Value**

A tibble with session simulation

**Examples**

```
## Not run:
sim <- anki_simulate_session(minutes = 30)

## End(Not run)
```

---

anki_stats_daily	<i>Calculate daily review statistics</i>
------------------	--

---

**Description**

Calculate daily review statistics

**Usage**

```
anki_stats_daily(path = NULL, profile = NULL, from = NULL, to = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
from	Optional start date (Date or character "YYYY-MM-DD")
to	Optional end date (Date or character "YYYY-MM-DD")

**Value**

A tibble with daily statistics

**Examples**

```
## Not run:
daily <- anki_stats_daily()
daily <- anki_stats_daily(from = "2024-01-01")

## End(Not run)
```

---

anki_stats_deck	<i>Calculate per-deck statistics</i>
-----------------	--------------------------------------

---

**Description**

Calculate per-deck statistics

**Usage**

```
anki_stats_deck(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with deck statistics

**Examples**

```
## Not run:
stats <- anki_stats_deck()

## End(Not run)
```

---

anki_streak	<i>Calculate current review streak</i>
-------------	--

---

**Description**

Calculate current review streak

**Usage**

```
anki_streak(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with current\_streak, longest\_streak, and streak\_history

**Examples**

```
## Not run:  
streak <- anki_streak()  
streak$current_streak  
streak$longest_streak  
  
## End(Not run)
```

---

anki\_streak\_analytics *Advanced Streak Analytics*

---

**Description**

Detailed streak analysis including best streaks, recovery time after breaks, weekend vs weekday patterns, and streak predictions.

**Usage**

```
anki_streak_analytics(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with detailed streak statistics

**Examples**

```
## Not run:  
streaks <- anki_streak_analytics()  
streaks$current  
streaks$history  
  
## End(Not run)
```

---

anki_study_plan	<i>Create study plan</i>
-----------------	--------------------------

---

### Description

Generates a daily study plan based on exam date and current progress.

### Usage

```
anki_study_plan(  
  path = NULL,  
  profile = NULL,  
  target_date,  
  daily_minutes = 60,  
  deck_pattern = NULL  
)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
target_date	Exam/target date
daily_minutes	Available study time per day in minutes (default 60)
deck_pattern	Optional pattern to filter decks

### Value

A list with study plan

### Examples

```
## Not run:  
plan <- anki_study_plan(target_date = "2024-06-15", daily_minutes = 90)  
  
## End(Not run)
```

---

anki\_study\_priorities *Generate study priority list*

---

### Description

Identifies which topics/decks should be prioritized based on progress and retention.

### Usage

```
anki_study_priorities(path = NULL, profile = NULL, by = "tag", n = 10)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Analysis type: "tag" or "deck" (default "tag")
n	Number of priority topics to return (default 10)

### Value

A tibble with prioritized topics

### Examples

```
## Not run:
priorities <- anki_study_priorities(n = 10)

## End(Not run)
```

---

anki\_summary *Quick collection summary*

---

### Description

Returns a one-liner overview of your Anki collection.

### Usage

```
anki_summary(path = NULL, profile = NULL)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with key metrics

**Examples**

```
## Not run:  
anki_summary()  
  
## End(Not run)
```

---

anki_suspended	<i>Get suspended cards</i>
----------------	----------------------------

---

**Description**

Get suspended cards

**Usage**

```
anki_suspended(path = NULL, profile = NULL, include_notes = FALSE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
include_notes	If TRUE, join with note data

**Value**

A tibble of suspended cards

**Examples**

```
## Not run:  
suspended <- anki_suspended()  
  
## End(Not run)
```

---

anki_tag_analysis	<i>Analyze tag usage</i>
-------------------	--------------------------

---

**Description**

Provides detailed tag statistics and identifies orphan/unused tags.

**Usage**

```
anki_tag_analysis(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with tag analysis

**Examples**

```
## Not run:  
tags <- anki_tag_analysis()  
  
## End(Not run)
```

---

anki_tags	<i>Extract unique tags with counts</i>
-----------	--

---

**Description**

Parses tags from all notes and returns unique tags with their frequency.

**Usage**

```
anki_tags(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with columns: tag, count

**Examples**

```
## Not run:  
tags <- anki_tags()  
  
## End(Not run)
```

---

anki\_time\_by\_hour      *Analyze reviews by hour of day*

---

**Description**

Shows when you study most during the day.

**Usage**

```
anki_time_by_hour(path = NULL, profile = NULL, days = 90)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to analyze (NULL for all)

**Value**

A tibble with hourly statistics

**Examples**

```
## Not run:  
anki_time_by_hour()  
  
## End(Not run)
```

---

anki\_time\_by\_weekday      *Analyze reviews by day of week*

---

**Description**

Shows which days you study most.

**Usage**

```
anki_time_by_weekday(path = NULL, profile = NULL, days = 90)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to analyze (NULL for all)

**Value**

A tibble with daily statistics

**Examples**

```
## Not run:  
anki_time_by_weekday()  
  
## End(Not run)
```

---

anki\_timestamp\_to\_date

*Convert Anki timestamp to date*

---

**Description**

Converts Anki's millisecond-epoch timestamps to Date objects.

**Usage**

```
anki_timestamp_to_date(x)
```

**Arguments**

x                    Numeric timestamp in milliseconds since Unix epoch (1970-01-01)

**Value**

Date object

**Examples**

```
# Convert a typical Anki timestamp  
anki_timestamp_to_date(1368291917470)
```

---

`anki_timestamp_to_datetime`*Convert Anki timestamp to datetime*

---

**Description**

Converts Anki's millisecond-epoch timestamps to POSIXct datetime objects.

**Usage**

```
anki_timestamp_to_datetime(x)
```

**Arguments**

x                      Numeric timestamp in milliseconds since Unix epoch (1970-01-01)

**Value**

POSIXct datetime object in local timezone

**Examples**

```
# Convert a typical Anki timestamp
anki_timestamp_to_datetime(1368291917470)
```

---

`anki_to_csv`*Export deck to CSV*

---

**Description**

Exports cards from a deck to a CSV file with note content.

**Usage**

```
anki_to_csv(  
  deck,  
  file = NULL,  
  path = NULL,  
  profile = NULL,  
  include_html = FALSE  
)
```

**Arguments**

deck	Deck name or ID (partial match supported)
file	Output file path (default: deck name + .csv)
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
include_html	If FALSE, strip HTML tags from fields

**Value**

Invisibly returns the exported data

**Examples**

```
## Not run:
anki_to_csv("Default")
anki_to_csv("Medical", file = "medical_cards.csv")

## End(Not run)
```

---

anki_to_html	<i>Export collection report to HTML</i>
--------------	---

---

**Description**

Export collection report to HTML

**Usage**

```
anki_to_html(file, path = NULL, profile = NULL)
```

**Arguments**

file	Output HTML file path
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

Invisibly returns the file path

**Examples**

```
## Not run:
anki_to_html("report.html")

## End(Not run)
```

---

anki_to_json	<i>Export collection as JSON</i>
--------------	----------------------------------

---

## Description

Full collection export as structured JSON for web dashboards or custom apps.

## Usage

```
anki_to_json(  
    path = NULL,  
    profile = NULL,  
    output_path = "anki_collection.json",  
    include_revlog = TRUE,  
    include_content = TRUE  
)
```

## Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
output_path	Path for output JSON file
include_revlog	Whether to include review log (can be large)
include_content	Whether to include card content

## Value

Invisibly returns the exported data structure

## Examples

```
## Not run:  
anki_to_json(output_path = "collection.json")  
  
## End(Not run)
```

---

anki_to_markdown	<i>Export deck to Markdown format</i>
------------------	---------------------------------------

---

### Description

Exports cards to Markdown flashcard format.

### Usage

```
anki_to_markdown(  
  deck,  
  file = NULL,  
  path = NULL,  
  profile = NULL,  
  format = c("obsidian", "logseq", "basic")  
)
```

### Arguments

deck	Deck name or ID
file	Output file path
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
format	Format: "obsidian", "logseq", or "basic"

### Value

Invisibly returns the number of cards exported

### Examples

```
## Not run:  
anki_to_markdown("Medical")  
  
## End(Not run)
```

---

anki\_to\_mochi      *Export to Mochi format*

---

### Description

Exports cards to Mochi's JSON import format.

### Usage

```
anki_to_mochi(
  path = NULL,
  profile = NULL,
  output_path = "mochi_cards.json",
  deck_pattern = NULL
)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
output_path	Path for output JSON file
deck_pattern	Optional pattern to filter decks

### Value

Invisibly returns the number of cards exported

### Examples

```
## Not run:
anki_to_mochi(output_path = "mochi_import.json")

## End(Not run)
```

---

anki\_to\_obsidian\_sr      *Export to Obsidian Spaced Repetition plugin format*

---

### Description

Exports cards in a format compatible with the Obsidian Spaced Repetition plugin.

**Usage**

```
anki_to_obsidian_sr(
  path = NULL,
  profile = NULL,
  output_dir = ".",
  model_id = NULL,
  deck_pattern = NULL,
  include_scheduling = FALSE
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
output_dir	Directory for output files (default: current directory)
model_id	Optional model ID to export (NULL for all)
deck_pattern	Optional pattern to filter decks
include_scheduling	Include scheduling info as YAML frontmatter

**Value**

Invisibly returns the number of cards exported

**Examples**

```
## Not run:
anki_to_obsidian_sr(output_dir = "~/obsidian/vault/flashcards")

## End(Not run)
```

---

anki_to_org	<i>Export deck to Org-mode format</i>
-------------	---------------------------------------

---

**Description**

Exports cards to Emacs Org-mode flashcard format (org-drill compatible).

**Usage**

```
anki_to_org(
  deck,
  file = NULL,
  path = NULL,
  profile = NULL,
  include_scheduling = FALSE
)
```

**Arguments**

deck	Deck name or ID
file	Output file path (default: deck name + .org)
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
include_scheduling	If TRUE, include scheduling data as properties

**Value**

Invisibly returns the number of cards exported

**Examples**

```
## Not run:
anki_to_org("Medical")
anki_to_org("Vocabulary", file = "vocab.org")

## End(Not run)
```

---

anki\_to\_supermemo      *Export deck to SuperMemo Q&A format*

---

**Description**

Export deck to SuperMemo Q&A format

**Usage**

```
anki_to_supermemo(deck, file = NULL, path = NULL, profile = NULL)
```

**Arguments**

deck	Deck name or ID
file	Output file path
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

Invisibly returns the number of cards exported

**Examples**

```
## Not run:
anki_to_supermemo("Medical")

## End(Not run)
```

---

anki_today	<i>Today's activity summary</i>
------------	---------------------------------

---

**Description**

Returns what happened today in your Anki studies.  
Detailed breakdown of today's Anki activity.

**Usage**

```
anki_today(path = NULL, profile = NULL)
```

```
anki_today(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with today's stats  
A list with today's activity breakdown

**Examples**

```
## Not run:
anki_today()

## End(Not run)
## Not run:
anki_today()

## End(Not run)
```

---

anki_ts_anomalies	<i>Detect anomalies in review patterns</i>
-------------------	--

---

**Description**

Identifies unusual days (very high or low activity).

**Usage**

```
anki_ts_anomalies(path = NULL, profile = NULL, threshold = 2)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
threshold	Number of standard deviations for anomaly (default 2)

**Value**

A tibble with anomalous days

**Examples**

```
## Not run:  
anomalies <- anki_ts_anomalies()  
  
## End(Not run)
```

---

anki\_ts\_autocorrelation

*Calculate autocorrelation of review patterns*

---

**Description**

Identifies cyclical patterns in studying.

**Usage**

```
anki_ts_autocorrelation(path = NULL, profile = NULL, max_lag = 30)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
max_lag	Maximum lag to compute (default 30)

**Value**

A tibble with autocorrelation values

**Examples**

```
## Not run:  
acf_data <- anki_ts_autocorrelation()  
# Peaks at lag 7 indicate weekly patterns  
  
## End(Not run)
```

---

anki\_ts\_decompose      *Decompose time series into trend, seasonal, and residual*

---

### Description

Uses classical decomposition on review data.

### Usage

```
anki_ts_decompose(  
  path = NULL,  
  profile = NULL,  
  metric = "reviews",  
  frequency = 7  
)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
metric	Metric to decompose: "reviews", "retention", or "time"
frequency	Seasonal frequency (default 7 for weekly pattern)

### Value

A decomposition object (or list with components)

### Examples

```
## Not run:  
dec <- anki_ts_decompose()  
plot(dec)  
  
## End(Not run)
```

---

anki\_ts\_forecast      *Forecast future reviews using simple methods*

---

### Description

Uses exponential smoothing or linear trend for forecasting.

### Usage

```
anki_ts_forecast(path = NULL, profile = NULL, days = 30, method = "linear")
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to forecast
method	Forecasting method: "ets" (exponential smoothing) or "linear"

**Value**

A tibble with forecasted values

**Examples**

```
## Not run:
forecast <- anki_ts_forecast(days = 30)

## End(Not run)
```

---

anki\_ts\_intervals      *Analyze interval progression over time*

---

**Description**

Tracks how card intervals change over time periods.

**Usage**

```
anki_ts_intervals(path = NULL, profile = NULL, by = "week")
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Aggregation period: "day", "week", or "month"

**Value**

A tibble with interval statistics over time

**Examples**

```
## Not run:
ts <- anki_ts_intervals()
plot(ts$date, ts$median_ivl, type = "l")

## End(Not run)
```

anki\_ts\_learning      *Analyze learning rate (new cards learned)*

---

**Description**

Tracks how many new cards are introduced and learned over time.

**Usage**

```
anki_ts_learning(path = NULL, profile = NULL, by = "week")
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Aggregation period: "day", "week", or "month"

**Value**

A tibble with learning statistics over time

**Examples**

```
## Not run:  
ts <- anki_ts_learning()  
  
## End(Not run)
```

---

anki\_ts\_maturation      *Analyze card maturation over time*

---

**Description**

Tracks cumulative mature cards over time.

**Usage**

```
anki_ts_maturation(  
  path = NULL,  
  profile = NULL,  
  by = "week",  
  mature_threshold = 21  
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Aggregation period: "day", "week", or "month"
mature_threshold	Days to consider a card mature (default 21)

**Value**

A tibble with maturation statistics

**Examples**

```
## Not run:
ts <- anki_ts_maturation()

## End(Not run)
```

---

anki_ts_plot	<i>Plot time series with trend line</i>
--------------	---

---

**Description**

Plot time series with trend line

**Usage**

```
anki_ts_plot(ts_data, y_col, title = NULL)
```

**Arguments**

ts_data	A tibble from any anki_ts_* function
y_col	Column name to plot on y-axis
title	Plot title

**Value**

A ggplot object

**Examples**

```
## Not run:
ts <- anki_ts_intervals()
anki_ts_plot(ts, "median_ivl", "Median Interval Over Time")

## End(Not run)
```

---

anki\_ts\_retention      *Analyze retention over time*

---

### Description

Tracks retention rate changes over time periods.

### Usage

```
anki_ts_retention(path = NULL, profile = NULL, by = "week")
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Aggregation period: "day", "week", or "month"

### Value

A tibble with retention statistics over time

### Examples

```
## Not run:
ts <- anki_ts_retention()

## End(Not run)
```

---

anki\_ts\_stability      *Analyze FSRS stability over time*

---

### Description

Tracks how memory stability evolves.

### Usage

```
anki_ts_stability(path = NULL, profile = NULL, by = "week")
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Aggregation period: "day", "week", or "month"

**Value**

A tibble with stability statistics over time

**Examples**

```
## Not run:  
ts <- anki_ts_stability()  
  
## End(Not run)
```

---

anki_ts_workload	<i>Analyze workload trends</i>
------------------	--------------------------------

---

**Description**

Tracks review workload over time.

**Usage**

```
anki_ts_workload(path = NULL, profile = NULL, by = "week")
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by	Aggregation period: "day", "week", or "month"

**Value**

A tibble with workload statistics over time

**Examples**

```
## Not run:  
ts <- anki_ts_workload()  
  
## End(Not run)
```

---

anki\_weak\_areas      *Find weak areas by tag or deck*

---

### Description

Identifies tags or subdecks with the lowest retention or highest lapse rates.

### Usage

```
anki_weak_areas(path = NULL, profile = NULL, n = 10, by = "tag", days = 90)
```

### Arguments

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
n	Number of weak areas to return (default 10)
by	Analysis type: "tag" or "deck" (default "tag")
days	Number of days to analyze (default 90)

### Value

A tibble of weak areas

### Examples

```
## Not run:
weak <- anki_weak_areas(n = 10, by = "tag")

## End(Not run)
```

---

anki\_workload\_projection      *Workload Projection (Rough Estimate)*

---

### Description

Project future review workload with scenario analysis. This provides rough ballpark estimates for planning purposes, not accurate FSRS simulation.

### Usage

```
anki_workload_projection(
  path = NULL,
  profile = NULL,
  days = 30,
  scenarios = NULL
)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
days	Number of days to project (default 30)
scenarios	List of scenario parameters

**Details**

This function uses simplified heuristics to estimate workload:

- New cards reviewed ~4x during learning phase
- Mature card reviews based on recent average
- Retention affects review frequency

For accurate FSRS simulation, use Anki's built-in simulator (Tools > FSRS > Simulate) or the FSRS Helper add-on, which implement the full FSRS algorithm.

This function is useful for:

- Quick "what if" scenario comparisons
- Rough planning estimates
- Programmatic analysis in R

**Value**

A tibble with workload projections for each scenario

**Examples**

```
## Not run:
proj <- anki_workload_projection(days = 30)

## End(Not run)
```

---

date\_to\_anki\_timestamp

*Convert date to Anki timestamp*

---

**Description**

Converts a Date or POSIXct to Anki's millisecond-epoch format.

**Usage**

```
date_to_anki_timestamp(x)
```

**Arguments**

x                    Date or POSIXct object

**Value**

Numeric timestamp in milliseconds since Unix epoch

**Examples**

```
date_to_anki_timestamp(Sys.Date())
```

---

fsrs\_compare\_parameters  
*Compare FSRS parameters*

---

**Description**

Compares your optimized FSRS parameters against defaults and community averages.

**Usage**

```
fsrs_compare_parameters(path = NULL, profile = NULL)
```

**Arguments**

path                Path to collection.anki2 (auto-detected if NULL)  
profile             Profile name (first profile if NULL)

**Value**

A list with parameter comparison

**Examples**

```
## Not run:  
comp <- fsrs_compare_parameters()  
  
## End(Not run)
```

---

fsrs\_current\_retrievability  
*Calculate current retrievability for all cards*

---

**Description**

Computes the current probability of recall for all FSRS-enabled cards.

**Usage**

```
fsrs_current_retrievability(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A tibble with cards and their current retrievability

**Examples**

```
## Not run:  
r <- fsrs_current_retrievability()  
# Cards with low retrievability need review soon  
r[r$retrievability < 0.8, ]  
  
## End(Not run)
```

---

fsrs\_decay\_distribution  
*Analyze FSRS decay parameter distribution*

---

**Description**

In FSRS-6, each card can have its own decay parameter. This analyzes the distribution across your collection.

**Usage**

```
fsrs_decay_distribution(path = NULL, profile = NULL, by_deck = FALSE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by_deck	If TRUE, analyze by deck

**Value**

A tibble with decay distribution statistics

**Examples**

```
## Not run:  
decay <- fsrs_decay_distribution()  
  
## End(Not run)
```

---

fsrs\_difficulty\_distribution  
*Analyze FSRS difficulty distribution*

---

**Description**

Analyze FSRS difficulty distribution

**Usage**

```
fsrs_difficulty_distribution(path = NULL, profile = NULL, by_deck = FALSE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by_deck	If TRUE, calculate distribution per deck

**Value**

A tibble with difficulty statistics

**Examples**

```
## Not run:  
diff_dist <- fsrs_difficulty_distribution()  
  
## End(Not run)
```

---

fsrs\_export\_reviews    *Export reviews for external FSRS analysis*

---

**Description**

Export reviews for external FSRS analysis

**Usage**

```
fsrs_export_reviews(file, path = NULL, profile = NULL, format = "csv")
```

**Arguments**

file	Output file path (.csv or .json)
path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
format	Output format: "csv" or "json"

**Value**

Invisibly returns the exported data

**Examples**

```
## Not run:  
fsrs_export_reviews("reviews.csv")  
  
## End(Not run)
```

---

fsrs\_forgetting\_index    *Calculate forgetting index*

---

**Description**

Estimates the percentage of cards currently below target retention.

**Usage**

```
fsrs_forgetting_index(path = NULL, profile = NULL, target_retention = 0.9)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
target_retention	Target retention (default 0.9)

**Value**

A list with forgetting analysis

**Examples**

```
## Not run:  
fsrs_forgetting_index()  
  
## End(Not run)
```

---

fsrs_from_csv	<i>Import review data from CSV for analysis</i>
---------------	---

---

**Description**

Import review data from CSV for analysis

**Usage**

```
fsrs_from_csv(  
  file,  
  date_col = "date",  
  rating_col = "rating",  
  card_col = "card_id",  
  date_format = "%Y-%m-%d"  
)
```

**Arguments**

file	Path to CSV file
date_col	Name of date column
rating_col	Name of rating column
card_col	Name of card ID column
date_format	Date format string

**Value**

A tibble with standardized review data

**Examples**

```
## Not run:  
reviews <- fsrs_from_csv("reviews.csv", date_col = "date", rating_col = "grade")  
  
## End(Not run)
```

---

fsrs\_get\_parameters     *Get FSRS parameters from Anki deck config*

---

**Description**

Extracts the FSRS parameters stored in Anki's deck configuration.

**Usage**

```
fsrs_get_parameters(path = NULL, profile = NULL)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)

**Value**

A list with FSRS parameters or NULL if not found

**Examples**

```
## Not run:  
params <- fsrs_get_parameters()  
  
## End(Not run)
```

---

fsrs\_memory\_states     *Calculate memory state for cards*

---

**Description**

Calculates the current FSRS memory state (stability, difficulty, retrievability) for all cards in the collection.

**Usage**

```
fsrs_memory_states(path = NULL, profile = NULL, include_projection = FALSE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
include_projection	Include projected future states

**Value**

A tibble with memory states

**Examples**

```
## Not run:  
states <- fsrs_memory_states()  
  
## End(Not run)
```

---

fsrs\_prepare\_for\_optimizer

*Prepare review data for r-fsrs optimizer*

---

**Description**

Converts Anki review history to the format expected by the r-fsrs package for parameter optimization.

**Usage**

```
fsrs_prepare_for_optimizer(path = NULL, profile = NULL, min_reviews = 3)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
min_reviews	Minimum reviews per card to include (default 3)

**Value**

A list of review items suitable for r-fsrs

**Examples**

```
## Not run:  
# Prepare data for r-fsrs  
items <- fsrs_prepare_for_optimizer()  
  
# Use with r-fsrs (if installed)  
# library(fsrs)  
# params <- compute_parameters(items)  
  
## End(Not run)
```

---

fsrs\_stability\_distribution  
*Analyze FSRS stability distribution*

---

**Description**

Analyze FSRS stability distribution

**Usage**

```
fsrs_stability_distribution(path = NULL, profile = NULL, by_deck = FALSE)
```

**Arguments**

path	Path to collection.anki2 (auto-detected if NULL)
profile	Profile name (first profile if NULL)
by_deck	If TRUE, calculate distribution per deck

**Value**

A tibble with stability statistics

**Examples**

```
## Not run:  
stab_dist <- fsrs_stability_distribution()  
  
## End(Not run)
```

---

import\_addon\_export *Import ankiR Stats Addon Export*

---

**Description**

Import data exported from the ankiR Stats Anki addon. This allows analyzing Anki data in R without direct database access.

**Usage**

```
import_addon_export(path)
```

**Arguments**

path	Path to the JSON export file
------	------------------------------

**Value**

A list containing cards, revlog, daily\_stats, and summary

**Examples**

```
## Not run:
data <- import_addon_export("ankir_export.json")
data$summary
data$cards
data$daily_stats

## End(Not run)
```

---

```
plot.anki_decomposition
      Plot time series decomposition
```

---

**Description**

Plot time series decomposition

**Usage**

```
## S3 method for class 'anki_decomposition'
plot(x, ...)
```

**Arguments**

<code>x</code>	An <code>anki_decomposition</code> object
<code>...</code>	Additional arguments (unused)

**Value**

A ggplot object (if ggplot2 available) or base R plots

---

```
print.anki_gamification
      Print Gamification Stats
```

---

**Description**

Print Gamification Stats

**Usage**

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

**Arguments**

<code>x</code>	Gamification stats object
<code>...</code>	Additional arguments

---

```
print.anki_mc_forecast
```

*Print Monte Carlo Forecast*

---

**Description**

Print Monte Carlo Forecast

**Usage**

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

**Arguments**

<code>x</code>	An <code>anki_mc_forecast</code> object
<code>...</code>	Additional arguments (ignored)

# Index

analyze\_addon\_import, 5  
anki\_ab\_comparison, 6  
anki\_backlog\_calculator, 7  
anki\_base\_path, 7  
anki\_benchmark, 8  
anki\_best\_review\_times, 8  
anki\_buried, 9  
anki\_burnout\_detection, 10  
anki\_card\_complexity, 11  
anki\_card\_content, 11  
anki\_card\_recommendations, 12  
anki\_cards, 13  
anki\_cards\_fsrs, 13  
anki\_cards\_full, 14  
anki\_cohort\_analysis, 15  
anki\_collection, 16  
anki\_compare\_by\_age, 17  
anki\_compare\_deck\_difficulty, 17  
anki\_compare\_decks, 18  
anki\_compare\_forecasts, 19  
anki\_compare\_groups, 19  
anki\_compare\_periods, 20  
anki\_consistency, 21  
anki\_coverage\_analysis, 22  
anki\_dashboard, 23  
anki\_db\_path, 23  
anki\_decks, 24  
anki\_due, 24  
anki\_empty\_cards, 25  
anki\_exam\_readiness, 26  
anki\_export\_importable, 27  
anki\_export\_revlog, 27  
anki\_field\_contents, 28  
anki\_find\_similar, 29  
anki\_fit\_forgetting\_curve, 30  
anki\_forecast, 31  
anki\_forecast\_enhanced, 31  
anki\_forecast\_monte\_carlo, 32  
anki\_gamification, 33  
anki\_health\_check, 34  
anki\_heatmap\_data, 35  
anki\_interference\_analysis, 35  
anki\_learning\_curve, 36  
anki\_learning\_efficiency, 37  
anki\_learning\_velocity, 38  
anki\_leeches, 38  
anki\_long\_cards, 39  
anki\_mature, 40  
anki\_media\_list, 40  
anki\_media\_missing, 41  
anki\_media\_path, 42  
anki\_media\_stats, 42  
anki\_media\_unused, 43  
anki\_models, 43  
anki\_monthly\_summary, 44  
anki\_new, 45  
anki\_notes, 45  
anki\_plot\_difficulty, 46  
anki\_plot\_forecast, 47  
anki\_plot\_forgetting\_curve, 47  
anki\_plot\_heatmap, 48  
anki\_plot\_hours, 49  
anki\_plot\_intervals, 49  
anki\_plot\_monte\_carlo, 50  
anki\_plot\_retention, 51  
anki\_plot\_stability, 51  
anki\_plot\_weekdays, 52  
anki\_profiles, 53  
anki\_progress\_report, 53  
anki\_quality\_report, 54  
anki\_quick\_summary, 55  
anki\_report, 55  
anki\_response\_time, 56  
anki\_response\_time\_outliers, 57  
anki\_retention\_by\_type, 57  
anki\_retention\_rate, 58  
anki\_retention\_stability, 59  
anki\_review\_quality, 59

anki\_revlog, 60  
anki\_roi\_analysis, 61  
anki\_schema\_version, 62  
anki\_search, 62  
anki\_search\_enhanced, 63  
anki\_session\_analysis, 65  
anki\_session\_stats, 65  
anki\_sibling\_analysis, 66  
anki\_similar\_cards, 67  
anki\_simulate\_session, 67  
anki\_stats\_daily, 68  
anki\_stats\_deck, 69  
anki\_streak, 69  
anki\_streak\_analytics, 70  
anki\_study\_plan, 71  
anki\_study\_priorities, 72  
anki\_summary, 72  
anki\_suspended, 73  
anki\_tag\_analysis, 74  
anki\_tags, 74  
anki\_time\_by\_hour, 75  
anki\_time\_by\_weekday, 75  
anki\_timestamp\_to\_date, 76  
anki\_timestamp\_to\_datetime, 77  
anki\_to\_csv, 77  
anki\_to\_html, 78  
anki\_to\_json, 79  
anki\_to\_markdown, 80  
anki\_to\_mochi, 81  
anki\_to\_obsidian\_sr, 81  
anki\_to\_org, 82  
anki\_to\_supermemo, 83  
anki\_today, 84  
anki\_ts\_anomalies, 84  
anki\_ts\_autocorrelation, 85  
anki\_ts\_decompose, 86  
anki\_ts\_forecast, 86  
anki\_ts\_intervals, 87  
anki\_ts\_learning, 88  
anki\_ts\_maturation, 88  
anki\_ts\_plot, 89  
anki\_ts\_retention, 90  
anki\_ts\_stability, 90  
anki\_ts\_workload, 91  
anki\_weak\_areas, 92  
anki\_workload\_projection, 92  
  
date\_to\_anki\_timestamp, 93  
  
fsrs\_compare\_parameters, 94  
fsrs\_current\_retrievability, 95  
fsrs\_decay\_distribution, 95  
fsrs\_difficulty\_distribution, 96  
fsrs\_export\_reviews, 97  
fsrs\_forgetting\_index, 97  
fsrs\_from\_csv, 98  
fsrs\_get\_parameters, 99  
fsrs\_memory\_states, 99  
fsrs\_prepare\_for\_optimizer, 100  
fsrs\_stability\_distribution, 101  
  
import\_addon\_export, 101  
  
plot.anki\_decomposition, 102  
print.anki\_gamification, 102  
print.anki\_mc\_forecast, 103