Package 'cre.dcf'

January 12, 2026

Title Discounted Cash Flow Tools for Commercial Real Estate

Version 0.0.3

Date 2025-12-10

Description Provides 'R' utilities to build unlevered and levered discounted cash flow (DCF) tables for commercial real estate (CRE) assets. Functions generate bullet and amortising debt schedules, compute credit metrics such as debt coverage ratios (DCR), debt service coverage ratios (DSCR), interest coverage ratios, debt yield ratios, and forward loan-to-value ratios (LTV) based on net operating income (NOI). The toolkit evaluates refinancing feasibility under alternative market scenarios and supports end-to-end scenario execution from a YAML (YAML Ain't Markup Language) configuration file parsed with 'yaml'. Includes helpers for sensitivity analysis, covenant diagnostics, and reproducible vignettes.

License MIT + file LICENSE

Encoding UTF-8

Language en

Depends R (>= 4.1)

Imports checkmate, dplyr, magrittr, purrr, stats, tibble, utils, yaml

Suggests ggplot2, knitr, readr, rmarkdown, scales, testthat (>= 3.0.0), tidyr

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.3.3

LazyData true

NeedsCompilation no

Author Kevin Poisson [aut, cre]

Maintainer Kevin Poisson <kevin.poisson@parisgeo.cnrs.fr>

Repository CRAN

Date/Publication 2026-01-12 18:50:21 UTC

Contents

`add_credit_ratios` *Add credit ratios for debt service, interest cover, debt yield, and forward loan-to-value*

Description

Align a project cash-flow table with a debt schedule and compute standard credit ratios for each period:

- debt service coverage ratio (DSCR),
- interest cover ratio (ICR),
- initial and current debt yield,
- forward loan-to-value (LTV) based on next-period NOI.

Optionally, simple covenant flags are added when threshold values are supplied.

Usage

```
add_credit_ratios(
  cf_tab,
  debt_sched,
  exit_yield,
  covenants = NULL,
  dscr_basis = c("noi", "gei", "cfads"),
  cfads_ti_lc = NULL,
  ignore_balloon_in_min = FALSE,
  maturity_year = NULL
)
```

Arguments

`cf_tab`	A data.frame or tibble of project cash flows over years 0..N, typically the output of `dcf_calculate()` or `cf_make_full_table()`. It must at least contain a `year` column and either `net_operating_income` or `gei`. When available, the following columns are used: `opex`, `cf_pre_debt`, `capex_recur`, `leasing_costs`, `loan_init`.
`debt_sched`	A data.frame or tibble representing the debt schedule, typically the output of `debt_built_schedule()`. It must contain `year`, `payment`, `interest`, and `outstanding_debt`, and may also include `debt_draw` and `loan_init`.
`exit_yield`	Numeric scalar; exit yield (in decimal form, for example 0.05) used to compute forward values as `NOI_next / exit_yield`.

`covenants` Optional list with elements `dscr_min`, `ltv_max` and/or `debt_yield_min`. When supplied, the function adds simple covenant indicators to the output table.

`dscr_basis` Character string specifying the numerator used for DSCR. One of `"noi"`, `"gei"` or `"cfads"`. The default is `"noi"`.

`cfads_ti_lc` Optional object used to construct a CFADS adjustment for tenant-improvement or leasing-cost allowances. If a list, the element `annual_allowance` (numeric scalar or vector) is subtracted from NOI. If a function, it is called as `cfads_ti_lc(cf_tab)` and the returned numeric vector is subtracted from NOI.

`ignore_balloon_in_min` Logical scalar. If `TRUE` and `maturity_year` is not `NULL`, the attribute `"min_dscr_pre_maturity"` is attached to the result and stores the minimum DSCR computed only over years 1 to `maturity_year - 1`, ignoring any balloon repayment at maturity.

`maturity_year` Optional integer scalar giving the contractual maturity year of the facility. Periods with `year > maturity_year` are treated as post-maturity (no outstanding debt, no payment, no interest). This parameter is required when `ignore_balloon_in_min = TRUE`.

Value

A tibble equal to `cf_tab` with the following additional columns:

- `gei`, `noi` (created if missing),
- `payment`, `interest`, `outstanding_debt`,
- `noi_fwd`, `value_forward`,
- `dscr`, `interest_cover_ratio`,
- `debt_yield_init`, `debt_yield_current`,
- `ltv_forward`,
- covenant indicators when `covenants` is supplied.

When `ignore_balloon_in_min = TRUE` and `maturity_year` is provided, the object also carries an attribute `"min_dscr_pre_maturity"` containing the minimum DSCR before maturity.

Examples

```
cf_tab <- data.frame(
  year = 0:3,
  gei  = c(0, 120, 123, 126),
  opex = c(0, 40, 41, 42),
  loan_init = c(2000, NA, NA, NA)
)

debt_sched <- data.frame(
  year = 0:3,
  payment = c(0, 150, 150, 2150),
  interest = c(0, 100, 95, 90),
  outstanding_debt = c(2000, 2000, 1950, 1900),
  debt_draw = c(2000, 0, 0, 0)
)
```

```
out <- add_credit_ratios(
  cf_tab = cf_tab,
  debt_sched = debt_sched,
  exit_yield = 0.05,
  covenants = list(dscr_min = 1.10, ltv_max = 0.70)
)

out
```

`as_rate` *Rate conversion (decimal vs bps)*

Description

Rate conversion (decimal vs bps)

Usage

```
as_rate(dec = NULL, bps = NULL)
```

Arguments

`dec`	numeric(1). Decimal rate.
`bps`	numeric(1). Basis points.

Value

numeric(1) as decimal.

`as_yaml` *Serialize a validated configuration list to YAML*

Description

Validates a configuration list against the package grammar using `cfg_validate()` and serializes it to a YAML file on disk. This helper is intended for reproducibility and interoperability, allowing a fully specified in-memory configuration to be persisted and reused in subsequent runs or edited manually by users.

Validates `config` and writes it to `path` as `'YAML'`.

Usage

```
as_yaml(config, path)

as_yaml(config, path)
```

Arguments

config	List specification following the package grammar.
path	Output file path (for example "case.yml").

Details

The function performs validation before writing to disk. If validation fails, an error is raised and no file is written. The YAML output is a direct serialization of the validated configuration list and therefore preserves all fields, including nested structures.

Value

The input path, returned invisibly, to allow use in pipelines.

The input path, invisibly.

Examples

```
tmp <- tempfile(fileext = ".yml")
cfg <- dcf_spec_template()
cfg$entry_yield <- 0.06
as_yaml(cfg, tmp)
stopifnot(file.exists(tmp))

cfg <- dcf_spec_template()
cfg$entry_yield <- 0.06
tmp <- tempfile(fileext = ".yml")
as_yaml(cfg, tmp)
stopifnot(file.exists(tmp))
unlink(tmp)
```

---

build_lease_table *Stylised rent table (lease cash-flow)*

Description

Builds a minimal year-noi table for n_years with optionally vectorised vacancy rates.

Usage

```
build_lease_table(rent_signed, surface_m2, n_years, vac_rate_vec = 0)
```

Arguments

rent_signed	numeric. Face rent (€/m²/year) (scalar or vector).
surface_m2	numeric. Floor area (m²) (scalar or vector).
n_years	integer(1). Number of years.
vac_rate_vec	numeric. Vacancy (scalar or vector), recycled to n_years.

Value

tibble(year, noi).

Examples

```
build_lease_table(400, 2500, n_years = 5, vac_rate_vec = c(0, .05, .1))
```

---

cfg_explain *Explain effective parameters after normalization*

Description

Produces a compact tibble that reports selected effective inputs used by the engine after validation and normalization (see cfg_normalize()).

Usage

```
cfg_explain(config)
```

Arguments

config	List configuration (not a file path).

Value

A tibble with selected effective parameters and derived values.

Examples

```
cfg <- dcf_spec_template()
cfg$acq_price_ht <- 1e6
ex <- cfg_explain(cfg)
str(ex)
```

---

cfg_missing *Report missing or inconsistent fields in a config list*

Description

Runs lightweight checks aligned with cfg_validate() and returns a table of issues, if any. This is a convenience wrapper for user-facing diagnostics; it does not replace cfg_validate().

Usage

```
cfg_missing(config)
```

Arguments

config	List configuration to inspect.

Value

A tibble with columns field, problem, hint, or an empty tibble if no issues are detected.

Examples

```
tib <- cfg_missing(list())
tib
```

---

cfg_normalize *Normalize YAML into canonical Discounted Cash Flow (DCF)/debt parameters*

Description

Converts a raw YAML configuration into a set of scalars and vectors directly usable by dcf_calculate() and debt_built_schedule().

Usage

```
cfg_normalize(cfg)
```

Arguments

cfg	list parsed from YAML (raw, not yet normalized).

Value

list including in particular:

- disc_rate, exit_yield, exit_cost,
- acq_price_ht, acq_price_di,
- ltv_init, rate_annual, maturity, type,
- arrangement_fee_pct, capitalized_fees,
- noi_vec, opex_vec, capex_vec (vectors of length N).

cfg_validate *Validate YAML configuration structure*

Description

Validate YAML configuration structure

Usage

```
cfg_validate(cfg)
```

Arguments

cfg	list returned by dcf_read_config().

Value

cfg invisibly (or error if invalid).

cf_compute_levered *Equity cash flows and metrics in the presence of debt*

Description

Computes equity cash flows over $t = 0..N$ from an unlevered Discounted Cash Flow (DCF) and an annual debt schedule, then derives equity IRR and equity NPV. The convention is that free_cash_flow includes the acquisition at $t = 0$ as a negative flow and includes operating free cash flows for $t >= 1$. Sale proceeds are booked at $t = N$ via sale_proceeds.

Usage

```
cf_compute_levered(dcf_res, debt_sched, cfg)
```

Arguments

dcf_res	list. Result of dcf_calculate(). Must contain: • inputs with at least acq_price, disc_rate, exit_yield, • cashflows with at least year, free_cash_flow, sale_proceeds, net_operating_income.
debt_sched	data.frame or tibble. Debt schedule (output of debt_built_schedule()). Minimal columns: year, payment, interest, amortization, outstanding_debt. Years must be compatible with dcf_res$cashflows$year.
cfg	list. Financing parameters. Must contain ltv_init. Optional: arrangement_fee_pct (default 0) and capitalized_fees (default TRUE).

Value

A list with:

- equity_cf: numeric vector of equity cash flows,
- metrics: list with irr_equity, npv_equity, equity_0, loan_draw_0,
- full: dcf_res$cashflows enriched by add_credit_ratios().

Examples

```
dcf <- dcf_calculate(
  acq_price = 1e7, entry_yield = 0.05, exit_yield = 0.055,
  horizon_years = 10, disc_rate = 0.07
)
sch <- debt_built_schedule(
  principal = 6e6, rate_annual = 0.045, maturity = 5, type = "bullet"
)
out <- cf_compute_levered(
  dcf_res = dcf,
  debt_sched = sch,
  cfg = list(ltv_init = 0.6, arrangement_fee_pct = 0, capitalized_fees = TRUE)
)
stopifnot(is.numeric(out$metrics$irr_equity) || is.na(out$metrics$irr_equity))
stopifnot(is.numeric(out$equity_cf))
```

---

cf_make_full_table *Assemble the full cash-flow table (discounted cash flow and debt)*

Description

Builds an annual table by merging operating cash flows from a discounted cash flow model with a debt schedule; standardises gross effective income (GEI) and net operating income (NOI), computes post-debt cash flows, the equity cash flow, and discounted equity cash flows. Enforces a minimal contract on expected columns on both inputs.

Usage

```
cf_make_full_table(dcf, schedule)
```

Arguments

dcf — A list containing at least an element cashflows (data.frame or tibble) with one row per year and the following columns:

- year (integer, 0 = acquisition date),
- net_operating_income (numeric),
- capex (numeric, optional),
- free_cash_flow (numeric, pre-debt cash flow),
- sale_proceeds (numeric, sale proceeds in the exit year, 0 otherwise),

- discount_factor (numeric, strictly positive discount factor).

If gei or noi are missing, they are derived according to the convention: gei := net_operating_income and noi := gei - opex. If opex is missing, it is set to 0.

schedule — A data.frame or tibble of the debt schedule with one row per year and the required columns:

- year (integer, aligned with dcf$cashflows$year),
- debt_draw (numeric, drawdown; typically positive at year == 0),
- interest (numeric),
- amortization (numeric),
- payment (numeric, debt service = interest + amortization; must be 0 at year == 0),
- arrangement_fee (numeric, upfront or recurring fees),
- outstanding_debt (numeric, end-of-period outstanding balance).

Details

Invariants and checks:

- Stop if required columns are missing on the Discounted Cash Flow (DCF) or the debt side.
- Stop if payment[year == 0] != 0.
- Warn if debt_draw[year == 0] <= 0.

Value

A merged tibble (join on year) containing:

- all input columns from the Discounted Cash Flow (DCF) and the debt schedule,
- df (alias of discount_factor),
- cf_pre_debt (= free_cash_flow),
- cf_post_debt (= free_cash_flow - payment - arrangement_fee + debt_draw),
- equity_flow (= cf_post_debt + sale_proceeds),
- equity_disc (= equity_flow / df).

Examples

```
cf <- tibble::tibble(
  year = 0:2,
  net_operating_income = c(NA, 120, 124),
  opex = c(0, 20, 21),
  capex = c(0, 5, 5),
  free_cash_flow = c(-100, 95, 98),
  sale_proceeds = c(0, 0, 150),
  discount_factor = c(1, 1.05, 1.1025)
)
dcf <- list(cashflows = cf)
```

```
schedule <- tibble::tibble(
  year = 0:2,
  debt_draw = c(60, 0, 0),
  interest = c(0, 3, 2),
  amortization = c(0, 10, 50),
  payment = interest + amortization,
  arrangement_fee = c(0.6, 0, 0),
  outstanding_debt = c(60, 50, 0)
)

res <- cf_make_full_table(dcf, schedule)
res
```

`compare_financing_scenarios` *Compare three financing structures on a common Discounted Cash Flow (DCF) base*

Description

Build and compare three financing setups for a given unlevered DCF:

- an all-equity case,
- a bullet debt structure,
- an amortizing debt structure.

All three scenarios share the same acquisition base, interest rate, maturity and target LTV. The function returns a summary table of key investment and credit metrics, together with detailed objects for each scenario.

Usage

```
compare_financing_scenarios(
  dcf_res,
  acq_price,
  ltv,
  rate,
  maturity,
  covenants = list(dscr_min = 1.25, ltv_max = 0.65)
)
```

Arguments

`dcf_res`	List; result of `dcf_calculate()` for the unlevered project. It is assumed to contain the cash-flow table and the input exit yield in `dcf_res$inputs$exit_yield`.
`acq_price`	Numeric scalar; acquisition base consistent with the pricing convention used in `dcf_res` (for example HT, DI or value).

ltv	Numeric scalar in [0, 1); target loan-to-value ratio at origination.
rate	Numeric scalar in [0, 1]; annual interest rate used to build the debt schedules.
maturity	Integer scalar greater than or equal to 1; debt maturity in years.
covenants	Optional list of covenant thresholds, for example list(dscr_min = 1.25, ltv_max = 0.65). These values are passed to add_credit_ratios() when computing credit ratios.

Value

A list with two components:

summary	A tibble that summarizes, for the all-equity, bullet and amortizing cases, the main valuation metrics (IRR, NPV) and selected credit indicators (for example minimum DSCR and maximum forward LTV).
details	A named list with one element per scenario. Each element contains the debt schedule (schedule), the full joined project and debt cash-flow table (full), the credit-ratio table (ratios), and the leveraged metrics object (metrics).

---

compute_equity_invest *Compute equity invested at t0 (acquisition costs already included in acq_price)*

Description

Compute equity invested at t0 (acquisition costs already included in acq_price)

Usage

```
compute_equity_invest(
  acq_price,
  ltv_init,
  arrangement_fee_pct = 0,
  capitalized_fees = TRUE
)
```

Arguments

acq_price	All-in acquisition price (basis for financing).
ltv_init	Initial LTV (0–1).
arrangement_fee_pct	Arrangement fee rate (0–1).
capitalized_fees	TRUE if fees are capitalized into the loan principal.

Value

A list with: equity_0, loan_draw_0, fees_init, fees_cap.

compute_leveraged_metrics *Levered summary (equity cash flows and equity metrics)*

Description

Builds equity cash flows from a Discounted Cash Flow (DCF) table and a standardised debt schedule.

Usage

```
compute_leveraged_metrics(dcf_res, debt_sched, equity_invest)
```

Arguments

dcf_res	list. Output of dcf_calculate().
debt_sched	data.frame. Output of debt_built_schedule() (0...N).
equity_invest	numeric(1). Equity contribution at t = 0 (positive).

Value

list containing irr_equity, npv_equity, cashflows (levered table), and a reminder of the project-level metrics.

compute_noi_y1 *Quick computation of year-1 NOI*

Description

Quick computation of year-1 NOI

Usage

```
compute_noi_y1(rent_signed, lettable_area, vac_rate = 0)
```

Arguments

rent_signed	numeric(1). Face rent (€/m²/year).
lettable_area	numeric(1). Lettable area (m²).
vac_rate	numeric(1) in [0,1). Average vacancy rate.

Value

numeric(1) NOI_{y1} rounded to cents.

Examples

```
compute_noi_y1(400, 2500, vac_rate = 0.05)
```

---

compute_unleveraged_metrics *Unlevered summary (project metrics)*

Description

Derives project-level metrics from the standard DCF table.

Usage

```
compute_unleveraged_metrics(dcf_res)
```

Arguments

dcf_res	list. Output of dcf_calculate().

Value

list containing irr_project, npv_project, irr_equity, npv_equity, and cashflows.

---

cre_glossary *Glossary of CRE finance and modelling terms*

Description

Bilingual glossary (English/French) of the main commercial real estate finance and discounted cash-flow modelling terms used in the package. Definitions are intended to be short, operational and consistent with the usage in vignettes and function documentation.

Usage

```
cre_glossary
```

Format

A tibble with one row per term and the following columns:

term_id Short, unique identifier used internally (e.g. "irr", "dscr").

term_en Canonical English label.

term_fr Canonical French label.

definition_en Operational English definition (2–4 lines).

definition_fr Operational French definition (2–4 lines).

category High-level category (e.g. "discounted_cash_flow", "debt_metrics", "portfolio", "leasing").

subcategory Optional subcategory (e.g. "return", "risk", "covenant").

see_also Comma-separated list of related `term_id` values.

See Also

Vignette `vignette("glossary", package = "cre.dcf")`

`dcf_add_noi_columns` *Explicitly standardise GEI and NOI columns in a Discounted Cash Flow (DCF) cash-flow table*

Description

Guarantees the presence of numeric columns `gei` and `noi` in a cash-flow table, to make explicit the income base used for the unlevered project IRR. In this package, `gei` denotes gross effective income (after vacancy and rent-free effects) and `noi` is computed as `gei - opex`.

The input may provide `gei` directly, or a legacy column `net_operating_income` which is interpreted here as `gei` (compatibility with earlier pipelines).

Usage

```
dcf_add_noi_columns(cf_tab)
```

Arguments

`cf_tab`	data.frame|tibble Cash-flow table for periods 0..N, typically produced by `dcf_calculate()`. Required columns: `opex` and either `gei` or `net_operating_income`.

Value

A `tibble` with guaranteed numeric columns `gei` and `noi`. Existing `noi` is preserved when present, but a warning is emitted if it differs from `gei - opex` beyond a small tolerance.

Examples

```
# Minimal example with a legacy column name (net_operating_income interpreted as GEI)
cf_tab <- tibble::tibble(
  year = 0:2,
  net_operating_income = c(0, 120, 124),
  opex = c(0, 20, 21)
)
dcf_add_noi_columns(cf_tab)

# Example where GEI is provided explicitly and NOI is already present
cf_tab2 <- tibble::tibble(
  year = 0:2,
```

```
  gei  = c(0, 120, 124),
  opex = c(0, 20, 21),
  noi  = c(0, 100, 103)
)
dcf_add_noi_columns(cf_tab2)
```

`dcf_calculate` *Unlevered discounted cash flow model for a commercial real estate asset*

Description

Builds an indexed annual pro forma over years 0..N, a terminal value, and unlevered valuation metrics including net present value (NPV) and internal rate of return (IRR) for a directly held commercial real estate (CRE) asset, without debt. The income base is net operating income (NOI).

Usage

```
dcf_calculate(
  acq_price,
  entry_yield,
  exit_yield,
  horizon_years,
  disc_rate,
  exit_cost = 0,
  capex = 0,
  index_rent = 0,
  vacancy = 0,
  opex = 0,
  noi = NULL
)
```

Arguments

`acq_price`	Numeric scalar. Acquisition price (net of tax or all-in, depending on the chosen convention).
`entry_yield`	Numeric scalar in `[0, 1]`. Entry yield; in top-down mode, `NOI[1]` = `entry_yield * acq_price`.
`exit_yield`	Numeric scalar in `(0, 1]`. Exit yield.
`horizon_years`	Integer scalar greater than or equal to 1. Projection horizon `N` in years.
`disc_rate`	Numeric scalar in `(0, 1]`. Discount rate.
`exit_cost`	Numeric scalar in `[0, 1)`. Exit cost as a fraction of the sale price. Default is 0.
`capex`	Numeric scalar or numeric vector of length `N`. Capital expenditure per year. Default is 0.
`index_rent`	Numeric scalar or numeric vector of length `N`. Annual rent indexation rate. Used only in top-down mode. Default is 0.

vacancy	Numeric scalar or numeric vector of length N in [0, 1). Average annual vacancy. Used only in top-down mode. Default is 0.
opex	Numeric scalar or numeric vector of length N. Operating expenses (non-recoverable). Default is 0.
noi	Numeric scalar or numeric vector of length N, optional. Exogenous NOI path (for example computed from leases). When non-NULL, it replaces the internal NOI calculation.

Details

Time convention: year = 0..N. The acquisition is booked at year = 0 in free_cash_flow as a negative cash flow equal to the acquisition price, and the sale is booked only at year = N in sale_proceeds. The project NPV corresponds to the sum of discounted_cash_flow.

Two construction modes are available for the NOI path:

- **Top-down mode** (default): when noi is NULL, the NOI path is derived from the entry yield and acquisition price: NOI[1] = entry_yield * acq_price, then indexed with index_rent and adjusted by vacancy.
- **Bottom-up mode**: when noi is supplied (scalar or vector), it is recycled to length N and used as the NOI[1..N] path. In this case, entry_yield, index_rent, and vacancy are not used to recompute NOI.

Value

A list with:

- inputs: list of main assumptions,
- cashflows: tibble 0..N with standardised columns,
- npv: project net present value (NPV),
- irr_project: project internal rate of return (IRR), unlevered.

Examples

```
res <- dcf_calculate(
  acq_price = 1000,
  entry_yield = 0.06,
  exit_yield = 0.055,
  horizon_years = 3,
  disc_rate = 0.08,
  capex = c(5, 5, 0),
  index_rent = c(0.01, 0.01, 0.01),
  vacancy = c(0.05, 0.05, 0),
  opex = c(10, 10, 10)
)
res$npv
res$irr_project
head(res$cashflows)
```

dcf_read_config *Read a configuration YAML*

Description

Read a configuration YAML

Usage

```
dcf_read_config(
  config_file = system.file("extdata", "preset_default.yml", package = "cre.dcf")
)
```

Arguments

config_file	path; default to inst/extdata/config.yml in the package.

Value

list

dcf_spec_template *Minimal specification template for a Discounted Cash Flow (DCF) case*

Description

Returns a ready-to-edit list that matches the package's YAML grammar. Use this for interactive prototyping or to generate a YAML file.

Usage

```
dcf_spec_template()
```

Value

A named list with all required top-level keys and sane defaults.

Examples

```
cfg <- dcf_spec_template()
str(cfg, max.level = 1)
```

dcf_write_yaml_template *Write a commented YAML template for users to edit*

Description

Creates a 'YAML' file on disk from dcf_spec_template(), suitable for manual editing.

Usage

```
dcf_write_yaml_template(path)
```

Arguments

path	File path where to write the 'YAML' file (for example "my_case.yml").

Value

The input path, invisibly.

Examples

```
tmp <- tempfile(fileext = ".yml")
dcf_write_yaml_template(tmp)
stopifnot(file.exists(tmp))
unlink(tmp)
```

debt_built_schedule *Debt schedule for bullet and amortising loans*

Description

Creates an annual schedule indexed from 0..maturity with an initial draw at year = 0, interest, amortisation, total payment, and end-of-year outstanding balance. The convention is no payment at year = 0. For both loan types, the outstanding principal is 0 at maturity up to rounding.

Usage

```
debt_built_schedule(
  principal,
  rate_annual,
  maturity,
  type = c("amort", "bullet"),
  extra_amort_pct = 0,
  arrangement_fee_pct = 0
)
```

Arguments

principal	Numeric scalar. Amount borrowed at year = 0 (greater than or equal to 0).
rate_annual	Numeric scalar in [0, 1]. Annual nominal interest rate.
maturity	Integer scalar greater than or equal to 1. Duration in years; returned years are 0..maturity.
type	Character scalar. Either "amort" (constant payment) or "bullet".
extra_amort_pct	Numeric scalar in [0, 1]. Additional annual amortisation rate (used only for "bullet").
arrangement_fee_pct	Numeric scalar in [0, 1]. Arrangement fee rate applied to principal.

Value

A tibble with columns year, debt_draw, interest, amortization, payment, arrangement_fee, outstanding_debt, and loan_init.

Examples

```
sch_b <- debt_built_schedule(6e6, 0.045, maturity = 5, type = "bullet")
sch_a <- debt_built_schedule(6e6, 0.045, maturity = 5, type = "amort")
sch_b
sch_a
```

---

derive_exit_yield *Derive an exit yield from an entry yield and a spread (bps)*

Description

Derive an exit yield from an entry yield and a spread (bps)

Usage

```
derive_exit_yield(entry_yield, spread_bps)
```

Arguments

entry_yield	numeric(1) >= 0. Entry cap-rate in decimal form.
spread_bps	numeric(1). Spread in basis points (may be negative).

Value

numeric(1) Exit yield in decimal form.

Examples

```
derive_exit_yield(0.055, 50) # 0.060
```

flag_covenants *Covenant flags after computing credit ratios*

Description

Adds logical indicator columns for covenant breaches based on three ratios: debt service coverage ratio (DSCR), forward loan-to-value ratio (LTV), and current debt yield.

Usage

```
flag_covenants(cf, cov)
```

Arguments

cf	A data.frame or tibble containing at least dscr, ltv_forward, and debt_yield_current.
cov	A list of covenant thresholds. Supported elements include: • dscr_min numeric, default 1.25, • ltv_max numeric in [0, 1], default 0.65, • debt_yield_min numeric, default 0.08.

Value

The input table cf enriched with logical columns cov_dscr_breach, cov_ltv_breach, and cov_dy_breach.

Examples

```
cf <- tibble::tibble(
  year = 1:3,
  dscr = c(1.40, 1.10, NA),
  ltv_forward = c(0.60, 0.70, 0.64),
  debt_yield_current = c(0.09, 0.07, 0.08)
)
cov <- list(dscr_min = 1.25, ltv_max = 0.65, debt_yield_min = 0.08)
flag_covenants(cf, cov)
```

forward_value_from_noi *Forward value from next-period NOI*

Description

Compute a forward-value vector based on next-period NOI and an exit yield. Given a series of annual NOI values, the function constructs a vector NOI can be obtained either from a fixed forward growth rate or from a simple extrapolation of observed growth.

Usage

```
forward_value_from_noi(noi_vec, exit_yield, g_forward = NA_real_)
```

Arguments

noi_vec	Numeric vector of annual NOI values.
exit_yield	Numeric scalar; exit yield in decimal form (for example 0.05).
g_forward	Optional numeric scalar giving a constant forward growth rate. When supplied, the last element of NOI_next is constructed as the last NOI multiplied by 1 + g_forward. When g_forward is NA (the default), a capped log-growth extrapolation is used instead.

Value

A numeric vector of forward values with the same length as noi_vec.

---

get_cfg *Safe access to nested YAML values*

Description

Safe access to nested YAML values

Usage

```
get_cfg(cfg, ..., default = NULL)
```

Arguments

cfg	list configuration object.
...	nested keys.
default	value if missing.

Value

value or default.

guard_rate *Guardrail on an input rate (message if scale likely incorrect)*

Description

Guardrail on an input rate (message if scale likely incorrect)

Usage

```
guard_rate(x, name)
```

Arguments

x	numeric(1).
name	character(1). Parameter name used in messages.

Value

numeric(1) unchanged.

init_debt_fees *Initial debt fees (arrangement fee)*

Description

Initial debt fees (arrangement fee)

Usage

```
init_debt_fees(loan_draw_0, arrangement_fee_pct = 0, capitalized = TRUE)
```

Arguments

loan_draw_0	Initial loan drawdown amount (before any possible capitalization of fees).
arrangement_fee_pct	Arrangement fee rate (0–1).
capitalized	Logical: TRUE = fee is capitalized into the loan principal, FALSE = fee is paid in cash.

Value

A list: amount (numeric), capitalized (logical).

irr_partition *IRR decomposition between operations and resale*

Description

Approximates the relative contribution of:

- operational cash flows (acquisition + NOI - capex - opex),
- resale (net sale in year N), to the total IRR, using NPV shares (`share`) and mapping them to `irr_total` (`irr_contrib` = `irr_total` * `share`).

Usage

```
irr_partition(cashflows, tv_disc = NULL, irr_total, initial_investment = NULL)
```

Arguments

`cashflows`	data.frame. Must contain at least `year`, `free_cash_flow`, `discount_factor`. If `sale_proceeds` is missing, it is assumed to be zero.
`tv_disc`	numeric(1). Terminal value already discounted (net sale), if available. If `NULL`, it is derived from `sale_proceeds` and `discount_factor`.
`irr_total`	numeric(1). Total IRR (project or equity) for which the decomposition is sought (e.g. `dcf_res$irr_project` or an equity IRR).
`initial_investment`	numeric(1). Not used here (kept for API compatibility).

Value

`tibble(component, share, irr_contrib)` with two rows: "Operations" and "Resale".

irr_safe *Robust internal rate of return (adaptive bracketing)*

Description

Computes a real IRR from a vector of dated cash flows $t = 0, \ldots, T$. The algorithm first searches for a root in an initial interval `[lower, upper]`. If this interval does not *bracket* a root (that is, if the net present value function does not change sign), the upper bound is expanded multiplicatively up to `max_upper`.

If the cash-flow series exhibits no sign change (all flows are `>= 0` or all `<= 0`), or if no root can be bracketed after expansion, the function silently returns `NA_real_` (optionally with a warning if `warn = TRUE`).

Usage

```
irr_safe(
  cf,
  lower = -0.9999,
  upper = 0.1,
  max_upper = 10000,
  tol = sqrt(.Machine$double.eps),
  warn = FALSE
)
```

Arguments

cf	Numeric. Vector of cash flows $t = 0, \ldots, T$.
lower, upper	Initial search bounds for the IRR (decimal rates).
max_upper	Maximum upper bound when automatically expanding the bracketing interval.
tol	Numerical tolerance passed to uniroot.
warn	Logical. If TRUE, emits a warning when the IRR cannot be computed (no sign change or failure of bracketing).

Value

A numeric scalar (decimal rate) corresponding to the IRR, or NA_real_ if the IRR is not defined or could not be located numerically.

Examples

```
irr_safe(c(-100, 60, 60))   # IRR defined
irr_safe(c(-100, -20, -5))  # no sign change -> NA
```

---

leases_tbl_structuration

Aggregate lease events into annual vectors aligned on base_year..base_year+horizon-1

Description

Converts a list of lease events into annual vectors for rent, vacancy, free months, tenant capex (€/sqm), and a new_lease flag. The [start, end] convention is used: an event applies to years y with start <= y <= end. Overlaps within a unit resolve as: rent/vac/new_lease: last event wins; capex_sqm/free_months: accumulated at start year. Returned vectors are **non-indexed** (indexation is applied later in cfg_normalize()).

Usage

```
leases_tbl_structuration(ev, horizon, base_year)
```

Arguments

ev	list of events with fields: start, end, rent, vac, free_months, capex_sqm, new_lease.
horizon	integer(1) >= 1, number of annual steps.
base_year	integer(1), first absolute year of the horizon.

Value

list with numeric vectors of length horizon: rent, vac, free, capex_sqm, new_lease.

---

npv_rate *Net present value at constant rate*

Description

NPV of cf evaluated at times (default 0..T).

Usage

```
npv_rate(cf, rate, times = seq_along(cf) - 1L)
```

Arguments

cf	numeric. Cash flows.
rate	numeric(1). Discount rate (decimal).
times	integer. Time indices (same length as cf).

Value

numeric(1) NPV.

---

price_from_cap *Acquisition price from an entry capitalization rate*

Description

Converts a given NOI_y1 and entry_yield into a net purchase price (HT) and an all-in price including acquisition costs (via acq_cost_rate).

Usage

```
price_from_cap(noi_y1, entry_yield, acq_cost_rate = 0)
```

Arguments

noi_y1	numeric(1). Expected NOI for year 1.
entry_yield	numeric(1) in (0,1]. Entry capitalization rate.
acq_cost_rate	numeric(1) in [0,1). Acquisition cost rate.

Value

list(ht = net price, di = all-in price).

Examples

```
price_from_cap(500000, 0.05, acq_cost_rate = 0.07)
```

---

run_case *Run a full DCF case from a list or a YAML file*

Description

User-facing single entry point. Accepts either an in-memory config list or a config_file path to YAML. Both routes share the same validation and normalization pathway, ensuring identical downstream behavior.

Usage

```
run_case(
  config = NULL,
  config_file = NULL,
  debt_type = c("bullet", "amort"),
  ltv_base = c("price_di", "price_ht", "value")
)
```

Arguments

config	Optional list configuration following the YAML grammar.
config_file	Optional path to a YAML configuration file. If both config and config_file are NULL, defaults to the package example at inst/extdata/config.yml.
debt_type	Debt schedule type to use ("bullet" or "amort"). This parameter overrides any implicit type inferred in normalization.
ltv_base	Base for loan-to-value (LTV) and initial principal. One of "price_di", "price_ht", or "value".

Details

The function centralizes user ergonomics:

- Reads either a list or a YAML file.
- Validates and normalizes with `cfg_validate()` and `cfg_normalize()`.
- Computes the unlevered discounted cash flow (DCF), builds a debt schedule, computes leveraged metrics, and adds credit ratios to the full cash-flow table.
- Handles capitalized arrangement fees by adjusting the scheduled principal to avoid double-counting.

Value

A list containing pricing (acquisition price net of taxes, acquisition costs, and acquisition price including costs), all-equity metrics, leveraged metrics, a comparison table, the full cash-flow table with credit ratios, and selected configuration flags.

Examples

```
# R list route
cfg <- dcf_spec_template()
cfg$leases <- list(
  list(
    unit = "U",
    area = 1000,
    events = list(
      list(
        start = cfg$purchase_year,
       end   = cfg$purchase_year + cfg$horizon_years,  # keep NOI positive in terminal year
        rent = 200,
        free_months = 0,
        capex_sqm = 0,
        vac = 0,
        new_lease = 0
      )
    )
  )
)
out <- run_case(config = cfg, debt_type = "bullet")
names(out)
```

`run_from_config` *Canonical pipeline from a YAML file*

Description

Canonical pipeline from a YAML file

Usage

```
run_from_config(config_file, ltv_base = c("price_ht", "price_di", "value"))
```

Arguments

config_file	path to YAML.
ltv_base	"price_ht" | "price_di" | "value".

Value

list(dcf, debt, full, ratios, norm)

---

select_terminal_noi *Robust selection of terminal NOI for resale valuation*

Description

Chooses a stabilised net operating income (NOI) for terminal value calculation, using a hierarchical decision rule designed to mitigate distortions driven by vacancy, capital expenditure, or atypical end-of-horizon cash-flow patterns.

The selection logic proceeds as follows:

1. If `NOI_N` is (numerically) zero and `force_theoretical_if_noi_n_zero` is `TRUE`, use `noi_theoretical` when provided.
2. If year `N` is clean (zero vacancy, zero capex, and `NOI_N > 0`), use `NOI_N`.
3. If year `N` is distorted but year `N-1` is clean and `NOI_{N-1} > 0`, use `NOI_{N-1}`.
4. Otherwise, if `noi_theoretical` is provided, use it.
5. As a last resort, fall back to `NOI_N`. A warning is emitted only when `NOI_N <= 0`.

Usage

```
select_terminal_noi(
  noi,
  vacancy = NULL,
  capex = NULL,
  noi_theoretical = NULL,
  force_theoretical_if_noi_n_zero = TRUE
)
```

Arguments

noi	Numeric vector of length N containing annual NOI values for years 1..N.
vacancy	Optional numeric vector of length N giving annual vacancy rates. When NULL, vacancy is assumed to be zero in all years.
capex	Optional numeric vector of length N giving annual capital expenditures. When NULL, capex is assumed to be zero in all years.
noi_theoretical	Optional numeric scalar giving a stabilised theoretical NOI (for example market rent multiplied by area).
force_theoretical_if_noi_n_zero	Logical scalar. When TRUE, and NOI_N is numerically zero, noi_theoretical is used when available.

Value

Numeric scalar giving the NOI retained for capitalization.

---

simulate_shock *Apply scenario shocks to a set of Discounted Cash Flow (DCF) assumptions*

Description

Applies additive shifts (rates and yields in decimal form) or proportional scalings (NOI, CAPEX) to a list of parameters. Preserves field names.

Usage

```
simulate_shock(cfg, deltas = list())
```

Arguments

cfg	list. Base assumptions (e.g. those passed to dcf_calculate()). Fields read if present: disc_rate, exit_yield, entry_yield, capex, index_rent, vacancy.
deltas	list. Supported keywords: • d_rate (additive on disc_rate, decimal), • d_exit_yield (additive on exit_yield, decimal), • d_noi (multiplicative on entry_yield, i.e. on year-1 net operating income NOI_y1), • d_capex (multiplicative on capex), • d_index (multiplicative on index_rent), • d_vacancy (multiplicative on vacancy).

Value

list cfg_choc with the same structure as cfg.

`styles_breach_counts` *Count covenant breaches by style under the bullet-debt scenario*

Description

This helper aggregates, for a set of styles, the number of periods in which bullet-debt credit metrics breach simple covenant guardrails:

- DSCR < `min_dscr_guard`,
- forward LTV > `max_ltv_guard`.

Usage

```
styles_breach_counts(
  styles = c("core", "core_plus", "value_added", "opportunistic"),
  min_dscr_guard = 1.2,
  max_ltv_guard = 0.65
)
```

Arguments

`styles`	Character vector of style names (e.g. `"core"`, `"core_plus"`, `"value_added"`, `"opportunistic"`). The output `style` factor will follow this ordering.
`min_dscr_guard`	Numeric scalar, DSCR guardrail below which a period is counted as a DSCR breach.
`max_ltv_guard`	Numeric scalar, forward-LTV guardrail above which a period is counted as an LTV breach.

Details

It relies on `style_bullet_ratios()`, which is expected to return, for each style, a tibble of yearly ratios in the bullet-debt scenario with at least the columns: `style`, `year`, `dscr`, `ltv_forward`.

Value

A tibble with one row per style and the columns:

- `style` (factor, levels = `styles`),
- `n_dscr_breach`: number of years with `dscr` < `min_dscr_guard`,
- `n_ltv_breach`: number of years with `ltv_forward` > `max_ltv_guard`. Year 0 is excluded from the counts.

`styles_break_even_exit_yield` *Break-even exit yield for a target leveraged equity IRR, by style*

Description

For each style, this helper solves (via `uniroot()`) for the exit yield that delivers a specified target leveraged equity IRR, holding all other assumptions of the preset constant.

Usage

```
styles_break_even_exit_yield(
  styles,
  target_irr,
  interval = c(0.03, 0.1),
  config_dir = system.file("extdata", package = "cre.dcf")
)
```

Arguments

`styles`	Character vector of style identifiers.
`target_irr`	Numeric, target leveraged equity IRR to hit (in decimal).
`interval`	Numeric vector of length 2 giving the bracketing interval for the absolute exit yield (e.g. `c(0.03, 0.10)` for 3%–10%).
`config_dir`	Directory where preset YAML files are stored.

Details

It proceeds by:

- reading the YAML preset,
- defining a root-finding function that, for a candidate absolute exit yield, adjusts `exit_yield_spread_bps` accordingly,
- calling `run_case()` and returning the difference between the resulting equity IRR and `target_irr`,
- bracketing the root over a user-specified interval.

The lower the break-even exit yield, the tighter the exit_pricing assumption that must be met to reach the hurdle, and the more the style depends on favourable market_conditions at sale.

Value

A tibble with columns:

- `style` (character),
- `target_irr` (numeric),
- `be_exit_yield` (numeric, break-even exit yield in decimal, or `NA` if no root was found in `interval`).

`styles_distressed_exit` *Distressed exit diagnostic across CRE investment styles*

Description

This helper applies a simple lender-driven distressed-exit rule to a set of canonical style presets. For each style and covenant regime, it:

1. Runs the baseline case via `run_case()`.
2. Identifies the first covenant breach under the bullet-debt scenario (DSCR and forward LTV).
3. Optionally shifts very early breaches to a minimum refinancing year (refinancing window logic).
4. Re-runs the case with a shortened horizon and a fire-sale exit-yield penalty, and extracts:
   - distressed equity IRR (possibly NA),
   - distressed equity multiple and loss percentage,
   - distressed sale value.

Usage

```
styles_distressed_exit(
  styles,
  regimes,
  fire_sale_bps = 100,
  refi_min_year = 3L,
  allow_year1_distress = TRUE,
  ext_dir = system.file("extdata", package = "cre.dcf")
)
```

Arguments

`styles`	Character vector of style tags, e.g. `c("core", "core_plus", "value_added", "opportunistic")`.
`regimes`	A data frame or tibble with at least three columns: `regime` (label), `min_dscr` (numeric), `max_ltv` (numeric). Each row defines a covenant regime (strict / baseline / flexible, etc.).
`fire_sale_bps`	Numeric scalar. Widening (in basis points) applied to the exit-yield spread in the distressed run (e.g. `+100` for +100 bps).
`refi_min_year`	Integer scalar. Minimum year at which a lender-driven distressed exit can occur. If a breach is detected before this year and `allow_year1_distress = FALSE`, the distressed exit is moved to `refi_min_year`.
`allow_year1_distress`	Logical. If `TRUE`, distress can occur in year 1. If `FALSE`, breaches in years `< refi_min_year` are shifted to `refi_min_year` (refinancing window logic).
`ext_dir`	Optional directory where style presets (YAML) are stored. Defaults to the package `inst/extdata` folder.

Value

A tibble with one row per combination of style and regime, and the columns:

- `style`, `regime`, `min_dscr`, `max_ltv`,
- `breach_year`, `breach_type`,
- `irr_equity_base`, `irr_equity_distress`,
- `distress_undefined` (logical),
- `equity_multiple_base`, `equity_multiple_distress`,
- `equity_loss_pct_base`, `equity_loss_pct_distress`,
- `sale_value_distress`.

`styles_equity_cashflows` *Extract leveraged equity cash flows by style*

Description

This helper loads a set of preset styles from YAML, runs each configuration through [`run_case()`] under the leveraged (debt) scenario, and extracts the yearly equity cash flows. It is primarily used in vignettes and tests to document the time profile of equity outflows and inflows by style.

Usage

```
styles_equity_cashflows(
  styles,
  config_dir = system.file("extdata", package = "cre.dcf")
)
```

Arguments

`styles`	Character vector of style identifiers, e.g. `c("core", "core_plus", "value_added", "opportunistic")`.
`config_dir`	Directory from which preset YAML files are loaded. Defaults to the package `inst/extdata` folder.

Details

For each style, the function:

1. reads `preset_<style>.yml` from `config_dir`;
2. calls [`run_case()`] and accesses `out$leveraged$cashflows`;
3. returns the pair `(year, equity_cf)` with a style label.

The sign convention follows [`compute_leveraged_metrics()`]: the initial equity outlay at $t = 0$ is negative, subsequent net equity distributions are positive when cash is returned to equity.

Value

A tibble with columns:

- `style` (character),
- `year` (integer),
- `equity_cf` (numeric), the leveraged equity cash flow in year `year`.

`styles_exit_sensitivity` *Exit-yield sensitivity of leveraged equity IRR by style*

Description

For each style, this helper:

- loads the corresponding YAML preset,
- perturbs the `exit_yield_spread_bps` parameter by a grid of deltas,
- reruns `run_case()` for each perturbation,
- collects the leveraged equity IRR.

Usage

```
styles_exit_sensitivity(
  styles,
  delta_bps = c(-50, 0, 50),
  config_dir = system.file("extdata", package = "cre.dcf")
)
```

Arguments

`styles`	Character vector of style identifiers (e.g. `"core"`, `"core_plus"`, `"value_added"`, `"opportunistic"`).
`delta_bps`	Numeric vector of exit-yield spread shocks in basis points, applied additively to the `exit_yield_spread_bps` field of each preset.
`config_dir`	Directory where preset YAML files are stored. Defaults to the package's `inst/extdata` folder.

Details

Economically, this approximates how sensitive each style's equity IRR is to small shifts in the exit_yield, and therefore to terminal_value risk. Strategies that concentrate value creation at exit (e.g. value_added, opportunistic) should display stronger IRR reactions to a given shock.

Value

A tibble with columns:

- `style` (character),
- `shock_bps` (numeric, the applied spread shock),
- `irr_equity` (numeric, leveraged equity IRR under the shock).

`styles_growth_sensitivity`
Rental-growth (indexation) sensitivity of leveraged equity IRR by style

---

Description

This helper perturbs the global `index_rate` parameter of each style preset by a given grid of additive shocks and recomputes the leveraged equity IRR.

Usage

```
styles_growth_sensitivity(
  styles,
  delta = c(-0.01, 0, 0.01),
  config_dir = system.file("extdata", package = "cre.dcf")
)
```

Arguments

`styles`	Character vector of style identifiers.
`delta`	Numeric vector of rental-growth shocks (additive) applied to the `index_rate` parameter of the preset.
`config_dir`	Directory where preset YAML files are stored.

Details

It therefore measures how dependent each style is on rental_growth (via indexation and lease renewals) to reach its target equity_IRR. In canonical calibrations, core strategies tend to be less sensitive than value_added or opportunistic profiles, which rely more heavily on growth and lease-up.

Value

A tibble with columns:

- `style` (character),
- `shock_growth` (numeric, growth shock added to `index_rate`),
- `irr_equity` (numeric, leveraged equity IRR under the shock).

`styles_manifest` *Compute the style-by-style manifest for canonical presets*

Description

This helper runs the four canonical style presets (`"core"`, `"core_plus"`, `"value_added"`, `"opportunistic"`) through [`run_case()`] and extracts a compact set of indicators that are salient for both investors and lenders:

Usage

```
styles_manifest(
  styles = c("core", "core_plus", "value_added", "opportunistic")
)
```

Arguments

`styles`	Character vector of style names to include. Defaults to the four canonical presets: `c("core", "core_plus", "value_added", "opportunistic")`.

Details

- project IRR (all-equity),
- equity IRR (levered),
- minimum DSCR under a bullet structure,
- initial LTV at origination under a bullet structure,
- maximum forward LTV under a bullet structure,
- equity NPV.

The result is a tibble that can be reused both in vignettes and in automated tests to ensure that the canonical presets preserve the intended risk–return and leverage–coverage hierarchies.

Value

A tibble with one row per style and the columns: `style`, `class`, `irr_project`, `irr_equity`, `dscr_min_bul`, `ltv_init`, `ltv_max_fwd`, `npv_equity`.

`styles_pv_split` *Present-value split between income and resale by style*

Description

For each style preset, this helper:

- runs `run_case()` under the all-equity scenario,
- takes the cash-flow table used for the unlevered DCF,
- discounts positive cash inflows at the DCF discount rate,
- decomposes the resulting present value into:
  - income = free cash flow excluding resale proceeds,
  - resale = terminal sale proceeds.

Usage

```
styles_pv_split(
  styles,
  config_dir = system.file("extdata", package = "cre.dcf")
)
```

Arguments

`styles`	Character vector of style identifiers.
`config_dir`	Directory where preset YAML files are stored.

Details

Year 0 (initial outlay) is excluded from the income/resale split so that shares remain numerically stable and interpretable.

Value

A tibble with columns: style, pv_income, pv_resale, share_pv_income, share_pv_resale.

`styles_revalue_yield_plus_growth` *Re-evaluate styles under a yield-plus-growth discounting rule*

Description

This helper re-runs a set of preset styles under a simplified `"yield_plus_growth"` discounting convention, leaving all cash-flow assumptions unchanged. It is primarily used in vignettes and tests to check that the qualitative ordering of styles (in terms of equity IRR and NPV) is robust to the choice of discounting scheme.

Usage

```
styles_revalue_yield_plus_growth(
  styles,
  config_dir = system.file("extdata", package = "cre.dcf")
)
```

Arguments

`styles`	Character vector of style identifiers, e.g. `c("core", "core_plus", "value_added", "opportunistic")`.
`config_dir`	Directory from which preset YAML files are loaded. Defaults to the package `inst/extdata` folder.

Details

For each style, the function:

1. loads the corresponding YAML preset file;
2. overrides `disc_method <- "yield_plus_growth"`;
3. sets `disc_rate_yield_plus_growth` so that the property yield equals `entry_yield` and the growth component equals `index_rate`;
4. calls [`run_case()`] and extracts the leveraged equity IRR and NPV.

Value

A tibble with one row per style and the columns:

- `style` (character),
- `irr_equity_y`: leveraged equity IRR under the `"yield_plus_growth"` convention,
- `npv_equity_y`: leveraged equity NPV under the same convention.

sweep_sensitivities *Sensitivity grid (rate / exit yield) and monotonicity of ratios*

Description

For each (rate, exit_yield) pair, builds a bullet schedule, merges it with dcf_calculate() cash flows, computes ratios via add_credit_ratios(), and returns min_dscr (t >= 1) and max_ltv_forward (t >= 1).

Usage

```
sweep_sensitivities(
  dcf_res,
  rate_grid,
  exit_yield_grid,
  ltv = 0.6,
  maturity = 5L
)
```

Arguments

dcf_res	list. Output of dcf_calculate().
rate_grid	numeric. Grid of annual nominal rates (decimal).
exit_yield_grid	numeric. Grid of exit_yield values (decimal).
ltv	numeric(1). Initial LTV (default 0.60).
maturity	integer(1). Maturity (years) of the bullet schedule.

Value

tibble with columns rate, exit_yield, min_dscr, max_ltv_forward.

test_refi *Test the feasibility of a refinancing at year T (interest-only diagnostic)*

Description

Assesses at \(T\) the simultaneous feasibility of DSCR and forward LTV covenants assuming an interest-only payment at \(T+1\). This diagnostic isolates covenant feasibility from the precise structure of the new loan.

Usage

```
test_refi(full, year_T, covenants, new_rate, new_exit_yield)
```

Arguments

full	data.frame. Merged table (0..N) from cf_make_full_table(), containing at least: year, net_operating_income, outstanding_debt.
year_T	integer(1). Evaluation year \(T\) (0..N).
covenants	list. Thresholds: dscr_min (default 1.25), ltv_max (default 0.65).
new_rate	numeric(1). New annual nominal rate (decimal).
new_exit_yield	numeric(1). New exit yield (decimal) for forward value. NOI_{T+1} is missing (default 0 if not provided as an attribute of full or in the DCF inputs).

Value

list with status ("ok"/"fail"), reasons (character) and snapshot (tibble).

Index