

Package ‘forensicR’

October 10, 2026

Title Crime Scene Documentation and Reconstruction Tools

Version 0.0.6

Description Supports crime scene investigators in documenting and reconstructing scenes. Scene points are mapped from field measurements (baseline, triangulation, polar). Shooting incidents are reconstructed from bullet defects with explicit uncertainty, following the methods described in Haag and Haag (2020, ISBN:9780128193976) and Hueske (2015, ISBN:9781498707664). Evidence and custody are logged in a structured form, and reproducible scene reports are rendered to Word, PDF or HTML.

Date 2026-09-30

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL <https://github.com/kaquadros/forensicR>

BugReports <https://github.com/kaquadros/forensicR/issues>

Suggests gridExtra, htmltools, htmlwidgets, knitr, png, rayrender, rgl, testthat, withr

Depends R (>= 4.1.0)

Config/testthat/edition 3

Imports cli, digest, ggplot2, rlang, rmarkdown, tibble

Config/roxygen2/version 8.1.0

VignetteBuilder knitr

NeedsCompilation no

Author Karine Quadros [aut, cre],
Allan Quadros [aut]

Maintainer Karine Quadros <kquadros@dgso.org>

Repository CRAN

Date/Publication 2026-10-10 10:50:07 UTC

Contents

along_wall	3
assumed_segment	3
coords_baseline	4
coords_polar	5
coords_triangulation	6
death_scene_record	7
evidence_log	8
file_hash	9
furniture	9
furniture_catalog	11
furniture_footprint	11
furniture_from_corners	12
furniture_table	12
impact_angle	13
intersect_trajectories	13
log_custody	14
log_item	15
opening	16
origin_zone	16
path_deflections	17
plot_scene	18
plot_trajectories	19
plot_wall_elevation	20
render_scene_3d	21
render_scene_report	23
report_footer	24
room_rect	24
scene_3d_widget	25
scene_report_skeleton	26
trajectory_2pt	27
trajectory_from_angles	27
trajectory_from_defect	28
trajectory_obstructions	29
trajectory_path	30
trajectory_table	31
wall_from_room	31

Index

32

along_wall	<i>Place an object against a wall, as it is measured in the field</i>
------------	---

Description

Position is given as the distance from the wall's left end (as seen from inside the room, see [wall_from_room\(\)](#)) to the object's left edge, its length along the wall and its depth into the room.

Usage

```
along_wall(walls, side, from, length, depth, id, type, ..., room_id = "room")
```

Arguments

walls	Walls tibble from room_rect() .
side	"north", "south", "east" or "west".
from	Distance from the left end of the wall to the object's left edge.
length	Extent along the wall.
depth	Extent into the room.
id, type	Passed to furniture() .
...	Further arguments to furniture() (height, color, ...).
room_id	Group id of the room rectangle in walls.

Value

A one-row furniture tibble.

Examples

```
walls <- room_rect(0, 0, 6, 5)
along_wall(walls, "north", from = 0.5, length = 2.1, depth = 0.9, id = "Sofa", type = "sofa")
```

assumed_segment	<i>Segment with an assumed (not measured) deflection</i>
-----------------	--

Description

For the case where the bullet demonstrably passed through an object but its path after it was not documented. The segment continues the previous direction from the joint with a cone of possible deflections of standard deviation `se_deflection` (degrees, random orientation). The result is an *assumption* and is flagged as such in tables, plots and reports.

Usage

```
assumed_segment(
  previous,
  joint = NULL,
  length = 2,
  se_deflection = 10,
  id = "assumed"
)
```

Arguments

previous	The trajectory this segment continues.
joint	Exit point $c(x, y, z)$. Default: the previous anchor.
length	Length of the assumed segment.
se_deflection	Standard deviation of the deflection angle, degrees.
id	Label.

Value

A trajectory with assumed = TRUE and a virtual anchor at joint + length * u.

coords_baseline	<i>Convert baseline (rectangular coordinate) measurements to XY</i>
-----------------	---

Description

The baseline method fixes a reference line between two known points (e.g. two wall corners). Each item is located by its distance *along* the baseline from the origin and its perpendicular *offset* from the line (positive to the left of the direction of travel).

Usage

```
coords_baseline(
  along,
  offset,
  origin = c(0, 0),
  end = c(1, 0),
  id = NULL,
  z = NA_real_,
  type = "evidence"
)
```

Arguments

along	Numeric. Distance along the baseline from the origin.
offset	Numeric. Perpendicular distance from the baseline. Positive values are to the left when walking from the origin toward the end point.
origin	Numeric length-2. XY of the baseline origin. Default <code>c(0, 0)</code> .
end	Numeric length-2. XY of the baseline end point. Default <code>c(1, 0)</code> (baseline runs along the positive x-axis).
id	Optional character vector of item labels.
z	Optional numeric. Height above the floor of each item (e.g. a bullet defect in a wall). Default NA, i.e. floor level / not measured.
type	Optional character. Item type used by the 3D renderers: "evidence" (default, numbered marker), "defect" (bullet defect, dark disc on the wall) or "bloodstain" (schematic red disc).

Value

A tibble with columns `id`, `x`, `y`, `z`, `type`, `method`.

Examples

```
coords_baseline(along = c(2.4, 5.1), offset = c(1.2, -0.8), id = c("A1", "A2"))
```

coords_polar	<i>Convert polar (azimuth/distance) measurements to XY</i>
--------------	--

Description

For total-station or compass-and-tape work. Azimuths follow the survey convention: degrees clockwise from north (north = 0, east = 90).

Usage

```
coords_polar(
  distance,
  azimuth,
  station = c(0, 0),
  id = NULL,
  z = NA_real_,
  type = "evidence"
)
```

Arguments

distance	Numeric. Horizontal distance from the station.
azimuth	Numeric. Degrees clockwise from north.
station	Numeric length-2. XY of the instrument. Default <code>c(0, 0)</code> .
id	Optional character vector of item labels.
z	Optional numeric. Height above the floor of each item (e.g. a bullet defect in a wall). Default NA, i.e. floor level / not measured.
type	Optional character. Item type used by the 3D renderers: "evidence" (default, numbered marker), "defect" (bullet defect, dark disc on the wall) or "bloodstain" (schematic red disc).

Value

A tibble with columns `id`, `x`, `y`, `z`, `type`, `method`.

Examples

```
coords_polar(distance = 10, azimuth = 90) # 10 units due east
```

`coords_triangulation` *Convert triangulation measurements to XY*

Description

Locates each item from its distances to two fixed reference points. Two mirror-image solutions exist; `side` picks the one to the left ("left", default) or right of the line from `p1` to `p2`.

Usage

```
coords_triangulation(
  d1,
  d2,
  p1,
  p2,
  side = c("left", "right"),
  id = NULL,
  z = NA_real_,
  type = "evidence"
)
```

Arguments

d1, d2	Numeric. Distances from the item to reference points p1, p2.
p1, p2	Numeric length-2. XY of the two reference points.
side	"left" or "right" of the directed line p1 -> p2.
id	Optional character vector of item labels.
z	Optional numeric. Height above the floor of each item (e.g. a bullet defect in a wall). Default NA, i.e. floor level / not measured.
type	Optional character. Item type used by the 3D renderers: "evidence" (default, numbered marker), "defect" (bullet defect, dark disc on the wall) or "bloodstain" (schematic red disc).

Value

A tibble with columns id, x, y, z, type, method. Rows whose distances cannot meet (no triangle) get NA coordinates with a warning.

Examples

```
coords_triangulation(d1 = 3, d2 = 4, p1 = c(0, 0), p2 = c(5, 0))
```

death_scene_record	<i>Structured death scene observation record</i>
--------------------	--

Description

Captures what a crime scene investigator documents at a death scene. It records observations only; it makes **no** estimate of time since death, which is the medical examiner's or coroner's determination.

Usage

```
death_scene_record(
  rigor = "not assessed",
  livor = "not assessed",
  livor_position = NA_character_,
  decomposition = NA_character_,
  insects = NA_character_,
  ambient_temp_c = NA_real_,
  environment = NA_character_,
  clothing = NA_character_,
  body_position = NA_character_,
  observed_at = Sys.time(),
  observed_by = NA_character_,
  notes = NA_character_
)
```

Arguments

rigor	One of "absent", "developing", "complete", "resolving", "not assessed".
livor	One of "absent", "unfixed", "fixed", "not assessed".
livor_position	Character. Where livor is present and whether it is consistent with the body position found (free text).
decomposition	Character. Free text or agency scale.
insects	Character. Presence/type observed, if any.
ambient_temp_c	Numeric. Ambient temperature at the body, Celsius.
environment	Character. Indoor/outdoor, HVAC state, windows, sun exposure, etc.
clothing	Character.
body_position	Character.
observed_at	POSIXct. When the observations were made.
observed_by	Character.
notes	Character.

Value

A one-row tibble of class `death_scene_record`.

evidence_log	<i>Create an empty evidence log</i>
--------------	-------------------------------------

Description

An evidence log is a list of two tibbles: `items` (one row per evidence item) and `custody` (one row per custody event). It is deliberately a plain R object, not a database: it lives in the report's project folder and travels with the case file.

Usage

```
evidence_log(case_id, agency = NA_character_)
```

Arguments

case_id	Character. Agency case number.
agency	Character. Agency name.

Value

An object of class `evidence_log`.

Examples

```
log <- evidence_log("2026-001234", "Lawrence Police Department")
```

file_hash	<i>SHA-256 hash of files</i>
-----------	------------------------------

Description

SHA-256 hash of files

Usage

```
file_hash(paths)
```

Arguments

paths	Character vector of file paths.
-------	---------------------------------

Value

Named character vector of hex digests.

furniture	<i>Define a scene object (furniture, appliance, person)</i>
-----------	---

Description

Objects are rectangular or circular footprints with a height. Rectangles are anchored at their south-west corner in their own frame and rotated counter-clockwise by angle about that corner; circles are anchored at their center. Height, color and label default from `furniture_catalog()` by type.

Usage

```
furniture(
  id,
  type,
  x,
  y,
  width = NULL,
  depth = NULL,
  diameter = NULL,
  height = NULL,
  z0 = 0,
  angle = 0,
  shape = c("rect", "circle"),
  anchor = NULL,
  color = NULL,
  alpha = 0.45,
  measured = TRUE,
```

```

    steps = NA_integer_,
    notes = NA_character_
  )

```

Arguments

<code>id</code>	Character. Unique label (e.g. "Sofa", "Table 1").
<code>type</code>	Character. A type from <code>furniture_catalog()</code> , or any other string (then height is required).
<code>x, y</code>	Anchor position, scene units: south-west corner of a rectangle (before rotation) or the center of a circle. See <code>anchor</code> .
<code>width, depth</code>	Rectangle extent along its own x and y axes.
<code>diameter</code>	Circle diameter (use with <code>shape = "circle"</code>).
<code>height</code>	Height above <code>z0</code> . Default from the catalog.
<code>z0</code>	Height of the object's base (e.g. a shelf). Default 0.
<code>angle</code>	Rotation in degrees, counter-clockwise, about the anchor.
<code>shape</code>	"rect" or "circle".
<code>anchor</code>	"sw" (default for rectangles) or "center".
<code>color</code>	Fill color. Default from the catalog.
<code>alpha</code>	Opacity 0 to 1 used in every view. Default 0.45, so what is behind an object stays visible.
<code>measured</code>	Logical. Were the dimensions measured (TRUE) or estimated (FALSE)? Estimated objects are drawn with dashed outlines and flagged in the report table.
<code>steps</code>	For <code>type = "stairs"</code> : number of steps rising along the object's own x axis. The 3D view stacks them; obstruction checks use the stepped profile.
<code>notes</code>	Character.

Value

A one-row tibble of class `furniture`. Combine several with `rbind()`.

Examples

```

f <- rbind(
  furniture("Table", "table", x = 3, y = 2, width = 1.2, depth = 0.8, angle = 15),
  furniture("Stool", "chair", x = 1.5, y = 1.2, diameter = 0.4, shape = "circle")
)

```

furniture_catalog	<i>Catalog of object types with default height, color and label</i>
-------------------	---

Description

Used by `furniture()` and friends to fill in defaults. Heights are typical values and should be replaced by measurements whenever available.

Usage

```
furniture_catalog()
```

Value

A tibble with columns type, height, color, label.

furniture_footprint	<i>Footprint polygons of scene objects</i>
---------------------	--

Description

Footprint polygons of scene objects

Usage

```
furniture_footprint(f)
```

Arguments

`f` A furniture tibble.

Value

A tibble with columns id, x, y, type, color, alpha, measured, one row per vertex.

`furniture_from_corners`*Axis-aligned object from two measured opposite corners*

Description

Axis-aligned object from two measured opposite corners

Usage

```
furniture_from_corners(id, type, p1, p2, ...)
```

Arguments

<code>id, type</code>	Passed to <code>furniture()</code> .
<code>p1, p2</code>	Numeric length-2 opposite corners $c(x, y)$.
<code>...</code>	Further arguments to <code>furniture()</code> .

Value

A one-row furniture tibble.

`furniture_table`*Summary table of scene objects for reports*

Description

Summary table of scene objects for reports

Usage

```
furniture_table(f)
```

Arguments

<code>f</code>	A furniture tibble.
----------------	---------------------

Value

A tibble with readable columns.

impact_angle	<i>Impact angle from a bullet defect's ellipse</i>
--------------	--

Description

For a projectile striking a flat surface, the angle of impact relative to the surface is approximated by $\text{asin}(\text{width} / \text{length})$ of the resulting elliptical defect (90 degrees is perpendicular). The relationship is reliable for non-yielding surfaces (drywall, wood, sheet metal); it is unreliable for glass, thick fabric, or when the defect is irregular. Near 90 degrees the estimate is inherently imprecise because the ellipse is almost circular; the Monte Carlo interval reflects that.

Usage

```
impact_angle(width, length, se = NULL, n_sim = 10000, level = 0.95)
```

Arguments

width, length	Numeric. Minor and major axis of the defect, same units.
se	Optional numeric. Measurement standard error for both axes. If supplied, a Monte Carlo interval is added (n_sim draws).
n_sim	Integer. Number of Monte Carlo draws.
level	Confidence level for the interval.

Value

A tibble with angle_deg and, if se is supplied, lower, upper and level.

Examples

```
impact_angle(width = 8, length = 16)           # 30 degrees
impact_angle(width = 8, length = 16, se = 0.5)
```

intersect_trajectories	<i>Convergence of two trajectories</i>
------------------------	--

Description

Finds the closest points between two back-projected trajectories. If the shots came from one position, the lines should pass close to each other *behind* both defects. The miss distance relative to its Monte Carlo spread tells you whether a common origin is supported; the midpoint estimates that origin.

Usage

```
intersect_trajectories(t1, t2, n_sim = 5000, level = 0.95)
```

Arguments

t1, t2	trajectory objects.
n_sim	Monte Carlo draws.
level	Interval level.

Value

A list with point (midpoint of closest approach, central estimate), miss_distance (central), behind_both (logical, central estimate lies behind both defects), summary (tibble of Monte Carlo quantiles for x, y, z and miss distance, plus p_behind_both) and samples (tibble of per-draw results).

log_custody	<i>Record a custody event</i>
-------------	-------------------------------

Description

Record a custody event

Usage

```
log_custody(  
  log,  
  item_id,  
  event,  
  from = NA_character_,  
  to = NA_character_,  
  at = Sys.time(),  
  notes = NA_character_  
)
```

Arguments

log	An evidence_log.
item_id	Character. Must already exist in log\$items, except for the initial "collected" event written by log_item() .
event	Character. E.g. "collected", "transferred", "sealed", "submitted to lab".
from, to	Character. Custodians.
at	POSIXct. Defaults to now.
notes	Character (optional).

Value

The updated evidence_log.

log_item	<i>Add an evidence item to a log</i>
----------	--------------------------------------

Description

Any photo files supplied are hashed with SHA-256 at logging time so the report can show that the images referenced are the ones collected.

Usage

```
log_item(  
  log,  
  item_id,  
  description,  
  location = NA_character_,  
  collected_by = NA_character_,  
  collected_at = Sys.time(),  
  packaging = NA_character_,  
  photos = character(),  
  notes = NA_character_  
)
```

Arguments

log	An evidence_log.
item_id	Character. Unique item label (e.g. "1", "A-3").
description	Character.
location	Character. Where found; ideally a scene point id.
collected_by	Character.
collected_at	POSIXct. Defaults to now.
packaging	Character. E.g. "paper bag, sealed, initialed".
photos	Character vector of image paths (optional).
notes	Character (optional).

Value

The updated evidence_log.

opening	<i>A door or window segment for scene plots</i>
---------	---

Description

A door or window segment for scene plots

Usage

```
opening(x1, y1, x2, y2, type = c("door", "window"), id = NULL)
```

Arguments

x1, y1, x2, y2	End points of the opening along a wall.
type	"door" or "window".
id	Group label; defaults to a unique one.

Value

A tibble with columns x, y, group, type.

origin_zone	<i>Possible origin of fire at assumed muzzle heights</i>
-------------	--

Description

Projects a trajectory back until it reaches each assumed muzzle height and reports the horizontal distance from the defect and the XY position, with a Monte Carlo interval from the trajectory's angular uncertainty. This is the standard way to bound where a shooter could have been: the trajectory alone is a line; a plausible muzzle height turns it into a position.

Usage

```
origin_zone(
  trajectory,
  heights = c(standing = 1.5, kneeling = 1, prone = 0.3),
  max_distance = 50,
  furniture = NULL,
  n_sim = 5000,
  level = 0.95
)
```

Arguments

trajectory	A trajectory, or a trajectory_path (its first segment, the one on the shooter's side, is used).
heights	Named numeric vector of muzzle heights, scene units.
max_distance	Largest horizontal distance considered, scene units (e.g. the room or lot dimension). Near-horizontal draws otherwise project to absurd distances; anything beyond this limit counts as not reachable.
furniture	Optional furniture tibble. Draws whose origin falls inside an object are counted as not reachable (nobody fires from inside a wardrobe); the fraction excluded this way is returned as p_in_furniture.
n_sim	Monte Carlo draws.
level	Interval level.

Details

The default heights are illustrative shoulder-level values for standing, kneeling and prone shooters and must be replaced by case-specific values.

Value

A tibble with one row per height: position, height, horizontal_distance (median), hd_lower, hd_upper, x, y, and p_reachable, the fraction of draws in which the back-projected line reaches that height within max_distance (it cannot when, e.g., the shot traveled upward and the height is above the defect). Quantiles are computed over the reachable draws only, so read them together with p_reachable.

Examples

```
t <- trajectory_from_angles(c(4.8, 0, 1.35), 350, -8)
origin_zone(t)
```

path_deflections	<i>Deflection angles and joint consistency of a path</i>
------------------	--

Description

Deflection angles and joint consistency of a path

Usage

```
path_deflections(path, n_sim = 5000, level = 0.95)
```

Arguments

path	A trajectory_path.
n_sim	Monte Carlo draws.
level	Interval level.

Value

A tibble with one row per joint: joint, x, y, z, deflection_deg with lower/upper, miss_in (distance from the joint to the incoming segment's line, median and interval) and miss_out (same for the outgoing segment, back-projected). Large misses relative to the measurement uncertainty mean the documented segments do not meet at the joint. assumed flags joints whose outgoing segment is an assumption.

plot_scene	<i>Plot scene points as a simple sketch</i>
------------	---

Description

Plot scene points as a simple sketch

Usage

```
plot_scene(
  points,
  units = "m",
  walls = NULL,
  furniture = NULL,
  furniture_alpha = NULL
)
```

Arguments

points	A tibble from one of the coords_*() functions, or several of them row-bound together.
units	Character. Axis unit label, e.g. "m" or "ft".
walls	Optional tibble of wall/door/window polylines with columns x, y, group and type; see room_rect() and opening() .
furniture	Optional furniture tibble; see furniture() .
furniture_alpha	Optional opacity overriding every object's own.

Value

A ggplot object.

plot_trajectories	<i>Plot trajectories in plan or elevation view with uncertainty</i>
-------------------	---

Description

Plan view: bird's-eye XY with each trajectory drawn back from its defect by back units; faint lines are Monte Carlo draws and show the cone of uncertainty. Elevation view: height against horizontal distance back from the defect, one panel per trajectory, with assumed muzzle heights dashed.

Usage

```
plot_trajectories(
  trajectories,
  view = c("plan", "elevation"),
  back = 8,
  n_draw = 150,
  points = NULL,
  walls = NULL,
  furniture = NULL,
  furniture_alpha = NULL,
  convergence = FALSE,
  heights = c(standing = 1.5, kneeling = 1, prone = 0.3),
  units = "m"
)
```

Arguments

trajectories	A trajectory, a trajectory_path, or a list of them. Path segments are drawn between their joints; assumed segments are dashed and run forward from the joint.
view	"plan" or "elevation".
back	Distance to project back, scene units.
n_draw	Number of Monte Carlo lines to draw.
points	Optional scene points tibble (from coords_*) to overlay in plan view.
walls	Optional wall/door/window polylines for plan view; see room_rect() and opening() .
furniture	Optional furniture tibble drawn in plan view.
furniture_alpha	Optional opacity overriding every object's own.
convergence	Plan view only. TRUE computes intersect_trajectories() for every pair and draws the Monte Carlo cloud of closest-approach points (only draws lying behind both defects) with the central estimate marked. Alternatively pass one result of intersect_trajectories() or a list of them. FALSE draws nothing.
heights	Muzzle heights to mark in elevation view.
units	Axis unit label.

Value

A ggplot object.

plot_wall_elevation	<i>Elevation view of one wall</i>
---------------------	-----------------------------------

Description

The standard companion to the plan view: the wall drawn face-on with the position and height of everything on or near it (bullet defects, stains, openings). Items are selected by their perpendicular distance to the wall line; heights come from the z column of the points. Trajectory anchors are added as defects, with a short arrow showing the projected direction of flight *into* the wall.

Usage

```
plot_wall_elevation(
  wall,
  points = NULL,
  trajectories = NULL,
  walls = NULL,
  tol = 0.15,
  height = 2.5,
  door_height = 2,
  window_sill = 0.9,
  window_head = 2.1,
  units = "m",
  title = NULL,
  furniture = NULL,
  tol_furniture = 0.35,
  furniture_alpha = NULL
)
```

Arguments

wall	Numeric length-4 c(x1, y1, x2, y2); the first point appears on the left. See wall_from_room() .
points	Optional scene points tibble with z.
trajectories	Optional trajectory or list of them.
walls	Optional walls tibble; door and window openings lying on this wall are drawn.
tol	Maximum perpendicular distance from the wall line for an item to be shown, scene units.
height	Wall (ceiling) height.
door_height, window_sill, window_head	Heights used to draw openings.
units	Axis unit label.

title	Plot title. Default built from the wall end points.
furniture	Optional furniture tibble. Objects whose footprint comes within tol_furniture of the wall are drawn as semi-transparent boxes with their height.
tol_furniture	Distance from the wall within which objects are shown.
furniture_alpha	Optional opacity overriding every object's own.

Value

A ggplot object.

render_scene_3d	<i>Render a 3D reconstruction of the scene (rayrender)</i>
-----------------	--

Description

Builds a schematic, to-scale 3D model of the room from the same objects used by the 2D plots and renders it with 'rayrender': walls with door and window panels, a floor grid with numbered x/y axes, numbered evidence markers, bullet defects at their measured height, schematic blood-stain discs, trajectories as rods and their angular uncertainty as translucent cones. Deliberately schematic: courtroom demonstratives must be accurate and non-prejudicial, so nothing is rendered photo-realistically.

Usage

```
render_scene_3d(
  walls,
  points = NULL,
  trajectories = NULL,
  furniture = NULL,
  furniture_alpha = NULL,
  file,
  view = c("door", "corner", "overhead", "dollhouse"),
  lookfrom = NULL,
  lookat = NULL,
  fov = NULL,
  wall_height = 2.5,
  back = 3,
  cone = TRUE,
  grid = 1,
  grid_on = c("floor", "walls"),
  grid_strength = 0.6,
  grid_labels = TRUE,
  omit_wall = NULL,
  width = 1000,
  height = 750,
```

```

    samples = 128,
    ...
)

```

Arguments

walls	Walls tibble; see room_rect() and opening() .
points	Optional scene points tibble (z and type used).
trajectories	Optional trajectory or list of them.
furniture	Optional furniture tibble; see furniture() . Objects are drawn as semi-transparent boxes (or cylinders) with a label on top.
furniture_alpha	Optional opacity overriding every object's own.
file	Output PNG path. No default: name the file explicitly, for example inside the case folder.
view	Camera preset: "door" (inside, at the door, eye level), "corner", "overhead", or "dollhouse" (from outside, south wall removed). Ignored if lookfrom is given.
lookfrom, lookat	Optional explicit camera position and target in scene coordinates c(x, y, z).
fov	Field of view in degrees. Default depends on view.
wall_height	Ceiling height.
back	Length of trajectory rods behind the defect.
cone	Logical. Draw the uncertainty cone for each trajectory.
grid	Grid spacing, scene units. 0 disables all grid lines.
grid_on	Where to draw grid lines: any of "floor" and "walls" (wall grids carry the z axis: verticals at each x/y tick and horizontals at each height tick). Use "none" for no lines but keep labels.
grid_strength	Number in 0 to 1 controlling how strongly grid lines stand out (line darkness and thickness). Wall lines are always drawn fainter than floor lines.
grid_labels	Logical. Draw the numbers: x/y along the floor edges and z at the start of each wall's horizontal grid line.
omit_wall	Optional side ("north", "south", "east", "west") to leave out so an outside camera can see in.
width, height, samples	Passed to rayrender::render_scene() .
...	Further arguments to rayrender::render_scene() .

Value

The output file path, invisibly.

render_scene_report	<i>Render a scene report to Word, PDF and/or HTML</i>
---------------------	---

Description

Renders an R Markdown scene report to one or more output formats. By default all three formats are produced so the investigator can pick the one their agency expects (Word for editing, PDF for submission, HTML for quick review). Every output ends with [report_footer\(\)](#).

Usage

```
render_scene_report(
  input = NULL,
  output_dir,
  formats = c("docx", "pdf", "html"),
  params = list(),
  quiet = TRUE
)
```

Arguments

input	Path to an .Rmd file. Defaults to a fresh copy of the packaged scene-report template written to output_dir as scene-report.Rmd (overwritten on every run). To customize the template, copy it with scene_report_skeleton() under another name, edit it, and pass that path as input.
output_dir	Directory to write outputs to. Created if missing. There is no default: choose the case folder explicitly, for example "case-2026-000123".
formats	Character vector, any of "docx", "pdf", "html".
params	Named list passed to the document as params. The packaged template understands: case_id, agency, investigator, scene_date, units, narrative (markdown text), points (tibble from the coords_*() functions), shooting (a list with optional impact, from impact_angle() , and trajectory, from trajectory_2pt()), evidence (an evidence_log()) and death_scene (a death_scene_record()). Sections whose object is missing print a placeholder line.
quiet	Logical. Suppress pandoc/knitr output.

Value

Invisibly, a named character vector of output file paths.

See Also

[report_footer\(\)](#), [scene_report_skeleton\(\)](#)

report_footer	<i>Provenance footer for generated reports</i>
---------------	--

Description

Returns a single short sentence stating that the document was generated with forensicR, including the package version, R version and timestamp. Every report template appends this line at the bottom; it is deliberately small so it does not compete with the report content or the agency header.

Usage

```
report_footer(time = Sys.time(), tz = "")
```

Arguments

time	POSIXct. Timestamp to print. Defaults to <code>Sys.time()</code> .
tz	Character. Time zone for the printed timestamp. Defaults to the session time zone.

Value

A length-one character vector.

Examples

```
report_footer()
```

room_rect	<i>Build a rectangular room outline for scene plots</i>
-----------	---

Description

Returns a closed polyline in the format accepted by the `walls` argument of `plot_scene()` and `plot_trajectories()`: columns `x`, `y`, `group` and `type`. Combine several with `rbind()` and add openings with `opening()`.

Usage

```
room_rect(x0, y0, width, depth, id = "room")
```

Arguments

x0, y0	Coordinates of the south-west corner.
width, depth	Extent along <code>x</code> (east) and <code>y</code> (north).
id	Group label.

Value

A tibble with columns x, y, group, type.

Examples

```
walls <- rbind(room_rect(0, 0, 6, 5),
  opening(2.5, 0, 3.5, 0, type = "door"))
```

scene_3d_widget	<i>Interactive 3D scene widget (rgl / WebGL)</i>
-----------------	--

Description

Same schematic model as [render_scene_3d\(\)](#), as an interactive widget the reader can rotate, zoom and walk around in an HTML report. Walls are semi-transparent so the inside stays visible from any angle.

Usage

```
scene_3d_widget(
  walls,
  points = NULL,
  trajectories = NULL,
  furniture = NULL,
  furniture_alpha = NULL,
  wall_height = 2.5,
  back = 3,
  cone = TRUE,
  grid = 1,
  grid_on = c("floor", "walls"),
  grid_strength = 0.6,
  grid_labels = TRUE,
  wall_alpha = 0.35,
  n_cone = 24,
  width = 800,
  height = 600
)
```

Arguments

walls	Walls tibble; see room_rect() and opening() .
points	Optional scene points tibble (z and type used).
trajectories	Optional trajectory or list of them.
furniture	Optional furniture tibble; see furniture() . Objects are drawn as semi-transparent boxes (or cylinders) with a label on top.

furniture_alpha	Optional opacity overriding every object's own.
wall_height	Ceiling height.
back	Length of trajectory rods behind the defect.
cone	Logical. Draw the uncertainty cone for each trajectory.
grid	Grid spacing, scene units. 0 disables all grid lines.
grid_on	Where to draw grid lines: any of "floor" and "walls" (wall grids carry the z axis: verticals at each x/y tick and horizontals at each height tick). Use "none" for no lines but keep labels.
grid_strength	Number in 0 to 1 controlling how strongly grid lines stand out (line darkness and thickness). Wall lines are always drawn fainter than floor lines.
grid_labels	Logical. Draw the numbers: x/y along the floor edges and z at the start of each wall's horizontal grid line.
wall_alpha	Wall transparency (0 to 1).
n_cone	Number of lines used to sketch each uncertainty cone.
width, height	Widget size in pixels.

Value

An rglwidget htmlwidget.

scene_report_skeleton *Copy the packaged scene report template*

Description

Copy the packaged scene report template

Usage

```
scene_report_skeleton(path, overwrite = FALSE)
```

Arguments

path	Destination path for the .Rmd file. No default: name the file explicitly.
overwrite	Logical. Overwrite an existing file?

Value

The destination path, invisibly.

trajectory_2pt	<i>Trajectory through two points on the bullet path</i>
----------------	---

Description

Two defects (entry and exit of a wall, or defects in two surfaces), or a defect and a probe tip. Positional uncertainty in each coordinate is propagated to the direction.

Usage

```
trajectory_2pt(p1, p2, se_position = 0, id = "T1")
```

Arguments

p1, p2	Numeric length-3 c(x, y, z). p1 is closer to the shooter, p2 further along the flight. The trajectory is anchored at p2 (the impact); p1 is kept as \$p1 and becomes the joint when the object is used as a later segment of a trajectory_path() .
se_position	Standard error of each coordinate, scene units.
id	Character label.

Value

An object of class trajectory with extra elements length (distance between the two points) and p1.

Examples

```
t <- trajectory_2pt(c(0, 0, 1.2), c(0, 2, 1.0), se_position = 0.01)
t$vertical_deg
```

trajectory_from_angles	<i>Trajectory from angles measured at the defect (rod, laser or protractor)</i>
------------------------	---

Description

The most common field method: a trajectory rod or laser is placed through the defect(s) and its vertical angle (angle finder / inclinometer) and horizontal angle (protractor, or azimuth from a total station) are recorded. Angular uncertainty of about 5 degrees is commonly cited for rod-based work; adjust to what your method validation supports.

Usage

```
trajectory_from_angles(
  anchor,
  azimuth_deg,
  vertical_deg,
  se_azimuth = 5,
  se_vertical = 5,
  id = "T1"
)
```

Arguments

anchor	Numeric length-3 c(x, y, z) of the defect in scene frame.
azimuth_deg	Direction of <i>flight</i> , degrees clockwise from north.
vertical_deg	Degrees above horizontal; negative = downward flight.
se_azimuth, se_vertical	Standard errors of the two angles, degrees.
id	Character label.

Value

An object of class trajectory.

Examples

```
t <- trajectory_from_angles(c(4.8, 0, 1.35), azimuth_deg = 350, vertical_deg = -8)
t
t$project(3) # point 3 m back toward the shooter
```

trajectory_from_defect

Trajectory from a single defect's ellipse on a vertical surface

Description

Combines the impact angle from the ellipse ([impact_angle\(\)](#)) with the orientation of the ellipse's major axis on the surface and a directionality indicator (lead-in mark, pinch point, or exit beveling) to obtain a full 3D trajectory. Use only when a rod cannot be placed.

Usage

```
trajectory_from_defect(
  anchor,
  width,
  length,
  major_axis_deg,
```

```

    wall_azimuth_deg,
    came_from = c("above", "below", "left", "right"),
    se = 0.5,
    se_orientation = 5,
    id = "T1"
)

```

Arguments

anchor	Numeric length-3 c(x, y, z) of the defect.
width, length	Ellipse axes.
major_axis_deg	Orientation of the major axis on the surface as seen by an observer on the shooter's side facing the wall: 0 = vertical, 90 = horizontal, measured clockwise from 12 o'clock (range 0 to 180).
wall_azimuth_deg	Azimuth of the wall's outward normal, i.e. the direction the wall <i>faces</i> (toward the shooter), degrees clockwise from north. A wall on the north side of a room faces south: 180.
came_from	Which end of the major axis the bullet came from, as seen by that observer: "above", "below", "left" or "right".
se	Measurement SE of the ellipse axes.
se_orientation	SE of major_axis_deg, degrees.
id	Character label.

Value

An object of class trajectory.

Examples

```

# North wall, bullet came from above-right, 30 deg to the surface
trajectory_from_defect(c(2, 5, 1.4), width = 6, length = 12,
                      major_axis_deg = 45, wall_azimuth_deg = 180,
                      came_from = "above")

```

trajectory_obstructions

Which objects does a back-projected trajectory pass through?

Description

Samples each trajectory backwards from its defect and reports the distance intervals inside every object. A hit means the object is an intermediate target or an obstruction that must be reconciled with the evidence (a defect in it, or a bullet path that cannot be right).

Usage

```
trajectory_obstructions(trajectories, f, back = 6, step = 0.01)
```

Arguments

trajectories	A trajectory or list of them.
f	A furniture tibble.
back	Distance to trace back from the defect.
step	Sampling step along the line.

Value

A tibble with trajectory, object, from, to (distances back from the defect) and top_height. Zero rows if nothing is hit.

trajectory_path	<i>Build a segmented trajectory through intermediate targets</i>
-----------------	--

Description

Segments are ordered from the shooter's side to the terminal defect. Each segment is a trajectory whose anchor is the defect at the *end* of that segment (the entry into the intermediate target for segment 1, the terminal defect for the last). Joints are the exit points from each intermediate target; when a following segment was measured by two points its first point is used as the joint automatically, otherwise the previous segment's anchor is used (thin-target approximation, reported as such).

Usage

```
trajectory_path(..., joints = NULL, id = "P1")
```

Arguments

...	trajectory objects in order, or a single list of them.
joints	Optional list of c(x, y, z) exit points, one per joint (length(segments) - 1). NULL entries fall back to the defaults.
id	Path label.

Value

An object of class trajectory_path.

Examples

```
s1 <- trajectory_from_angles(c(4.0, 2.6, 0.5), 69, -30, id = "into table")
s2 <- trajectory_2pt(c(4.25, 3.1, 0.45), c(4.9, 5.0, 0.2), se_position = 0.01, id = "to wall")
p <- trajectory_path(s1, s2, id = "Shot 4")
p
path_deflections(p)
```

trajectory_table	<i>Summarize one or more trajectories as a table</i>
------------------	--

Description

Summarize one or more trajectories as a table

Usage

```
trajectory_table(trajectories)
```

Arguments

trajectories A trajectory, a trajectory_path, or a list of them.

Value

A tibble with one row per trajectory.

wall_from_room	<i>Wall segment of a rectangular room, ordered as seen from inside</i>
----------------	--

Description

Convenience for `plot_wall_elevation()`: returns the end points of one side of a room built with `room_rect()`, ordered so that the first point is on the viewer's *left* when standing inside the room facing that wall.

Usage

```
wall_from_room(walls, side = c("north", "south", "east", "west"), id = "room")
```

Arguments

walls A tibble from `room_rect()` (optionally with openings bound).
side "north", "south", "east" or "west".
id Group id of the room rectangle in walls. Default "room".

Value

Numeric length-4 c(x1, y1, x2, y2).

Index

along_wall, 3
assumed_segment, 3

coords_baseline, 4
coords_polar, 5
coords_triangulation, 6

death_scene_record, 7
death_scene_record(), 23

evidence_log, 8
evidence_log(), 23

file_hash, 9
furniture, 9
furniture(), 3, 11, 12, 18, 22, 25
furniture_catalog, 11
furniture_catalog(), 9, 10
furniture_footprint, 11
furniture_from_corners, 12
furniture_table, 12

impact_angle, 13
impact_angle(), 23, 28
intersect_trajectories, 13
intersect_trajectories(), 19

log_custody, 14
log_item, 15
log_item(), 14

opening, 16
opening(), 18, 19, 22, 24, 25
origin_zone, 16

path_deflections, 17
plot_scene, 18
plot_scene(), 24
plot_trajectories, 19
plot_trajectories(), 24
plot_wall_elevation, 20

plot_wall_elevation(), 31

rayrender::render_scene(), 22
render_scene_3d, 21
render_scene_3d(), 25
render_scene_report, 23
report_footer, 24
report_footer(), 23
room_rect, 24
room_rect(), 3, 18, 19, 22, 25, 31

scene_3d_widget, 25
scene_report_skeleton, 26
scene_report_skeleton(), 23

trajectory_2pt, 27
trajectory_2pt(), 23
trajectory_from_angles, 27
trajectory_from_defect, 28
trajectory_obstructions, 29
trajectory_path, 30
trajectory_path(), 27
trajectory_table, 31

wall_from_room, 31
wall_from_room(), 3, 20