Package 'streetlamp'

November 6, 2026

Title Panel Econometrics for the Effect of Police Stop and Search on Recorded Crime

Version 0.2.0

Description Builds an area-by-month panel of recorded crime in England and Wales from the data.police.uk bulk archive <https://data.police.uk/data/archive/>, attaches stop and search intensity as a treatment variable, and provides a panel-econometrics toolkit for evaluating policing interventions: two-way fixed effects, event studies, difference-in-differences with staggered adoption, synthetic control, spatial spillover and displacement models, and stop-crime elasticities by offence type. Area boundaries are read from the Office for National Statistics Open Geography Portal <https://geoportal.statistics.gov.uk/>, Census 2021 population from the 'Nomis' API <https://www.nomisweb.co.uk/api/v01/help>, and notes of re-supplied and missing force data from the data.police.uk change log <https://data.police.uk/changelog/>. Every panel carries a contract recording provenance, coverage, geography vintage and treatment definition; every estimator names its identifying assumption and ships with a placebo or pre-trend diagnostic.

License MIT + file LICENSE

URL https://github.com/BlackThrive/streetlamp, https://blackthrive.github.io/streetlamp/

BugReports https://github.com/BlackThrive/streetlamp/issues

Depends R (>= 4.1.0)

Imports cli, did, digest, dplyr, fixest, generics, ggplot2 (>= 3.5.0), grDevices, grid, httr2, rlang (>= 1.1.0), sf, spdep, stats, tibble, tidyr, tools, utils, withr

Suggests didimputation, httptest2, knitr, rmarkdown, spelling, strucchange, Synth, testthat (>= 3.0.0), vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

Config/Needs/website rmarkdown

Encoding UTF-8

Language en-GB

RoxygenNote 7.3.3

NeedsCompilation no

Author Mustapha Wasseja [aut, cre],
Souci Frissa [aut],
Sarah Hamed [aut],
Black Thrive Global [cph, fnd]

Maintainer Mustapha Wasseja <muswaseja@gmail.com>

Repository CRAN

Date/Publication 2026-11-06 17:10:02 UTC

Contents

lamp_adjacency . . . 3
lamp_adjacency_from_nb . . . 4
lamp_allocation . . . 5
lamp_archive_download . . . 6
lamp_archive_index . . . 8
lamp_archive_register . . . 9
lamp_archive_snapshot . . . 10
lamp_area_lookup . . . 11
lamp_bank_holidays . . . 11
lamp_boundaries . . . 12
lamp_cache_dir . . . 13
lamp_changelog . . . 14
lamp_colours . . . 15
lamp_contract . . . 16
lamp_coverage . . . 17
lamp_crimes_prevented . . . 18
lamp_crime_types . . . 19
lamp_deprivation . . . 20
lamp_design_audit . . . 21
lamp_detect_surges . . . 22
lamp_did_staggered . . . 23
lamp_displacement_quotient . . . 25
lamp_effect_summary . . . 27
lamp_elasticity . . . 28
lamp_elasticity_robustness . . . 30
lamp_event_study . . . 31
lamp_forces . . . 33
lamp_list_versions . . . 34
lamp_lsoa_lookup . . . 35
lamp_lsoa_vintage . . . 36
lamp_map . . . 37
lamp_outcome_types . . . 38

lamp_panel . . . 39
lamp_placebo . . . 41
lamp_population . . . 43
lamp_pretrends . . . 44
lamp_read_crime . . . 45
lamp_read_outcomes . . . 46
lamp_read_stop_counts . . . 48
lamp_report . . . 49
lamp_revintage . . . 50
lamp_s60_legislation . . . 51
lamp_sample_panel . . . 52
lamp_select_version . . . 53
lamp_shocks . . . 54
lamp_simulate . . . 54
lamp_simulation_truth . . . 57
lamp_spillover . . . 58
lamp_synth . . . 59
lamp_table . . . 61
lamp_theme . . . 62
lamp_treatment . . . 63
lamp_trend_sensitivity . . . 64
lamp_twfe . . . 65
lamp_version_diff . . . 67
tidy.lamp_estimate . . . 68

Index **69**

`lamp_adjacency` *Spatial adjacency for a set of areas*

Description

Builds a neighbour list and row-standardised spatial weights for area polygons with `spdep`, for the spillover estimators and for Moran's I diagnostics. Queen contiguity (any shared point) is the default; rook contiguity needs a shared edge; `knn` links each area to its `k` nearest neighbours by centroid and symmetrises the result. Areas with no neighbours (islands) are allowed and counted.

Usage

```
lamp_adjacency(boundaries, method = c("queen", "rook", "knn"), k = 6L)
```

Arguments

`boundaries`	An `sf` polygon layer with an `area` column, for example from `lamp_boundaries()`.
`method`	`"queen"` (default), `"rook"` or `"knn"`.
`k`	Number of neighbours for `method = "knn"`.

Value

A list of class `lamp_adjacency` with `areas` (identifiers in order), `nb` (an spdep neighbour list), `listw` (row-standardised weights, `style = "W"`), `method`, `k`, `n_islands`, `boundary_vintage` (taken from the `vintage` attribute of `boundaries` when present) and `created`.

See Also

Other panel: `lamp_adjacency_from_nb()`, `lamp_boundaries()`, `lamp_changelog()`, `lamp_coverage()`, `lamp_panel()`, `lamp_population()`, `lamp_revintage()`

Examples

```
squares <- sf::st_sf(
  area = c("a", "b", "c"),
  geometry = sf::st_sfc(
    sf::st_polygon(list(rbind(c(0, 0), c(1, 0), c(1, 1), c(0, 1), c(0, 0)))),
    sf::st_polygon(list(rbind(c(1, 0), c(2, 0), c(2, 1), c(1, 1), c(1, 0)))),
    sf::st_polygon(list(rbind(c(5, 5), c(6, 5), c(6, 6), c(5, 6), c(5, 5)))),
    crs = 27700
  )
)
adj <- lamp_adjacency(squares)
adj
adj$nb
```

---

lamp_adjacency_from_nb

Build an adjacency object from a neighbour list

Description

Wraps an existing spdep neighbour list, for example the lattice built by `lamp_simulate()`, in the object the spillover estimators expect.

Usage

```
lamp_adjacency_from_nb(nb, areas, method = "supplied")
```

Arguments

`nb`	An spdep neighbour list.
`areas`	Area identifiers in the same order as `nb`.
`method`	Label recorded in the object.

Value

A `lamp_adjacency` object.

See Also

Other panel: lamp_adjacency(), lamp_boundaries(), lamp_changelog(), lamp_coverage(), lamp_panel(), lamp_population(), lamp_revintage()

Examples

```
sim <- lamp_simulate(n_areas = 16, n_months = 12, design = "spillover", seed = 1)
truth <- attr(sim, "truth")
lamp_adjacency_from_nb(truth$nb, truth$lattice$area)
```

---

lamp_allocation *How stop and search activity follows recorded crime*

Description

Regresses searches on the crime recorded in previous months, with area and month fixed effects. This is not an effect of crime on searching in any causal sense; it is a description of how activity is allocated, and it is here because without it the other estimators cannot be read honestly.

Usage

```
lamp_allocation(
  panel,
  stops = "stops",
  crime = "crime_total",
  crime_lags = 1:3,
  cluster = c("area", "force"),
  family = c("poisson", "ols_log")
)
```

Arguments

panel	A lamp_panel.
stops	The activity column, default "stops".
crime	The crime column, default "crime_total".
crime_lags	Lags of crime to include, default 1 to 3.
cluster	"area" (default) or "force".
family	"poisson" (default) or "ols_log".

Value

A lamp_estimate of class lamp_allocation, with one coefficient per lag and diagnostics$total for their sum. The identifying assumption field states plainly that the result is descriptive.

Why this matters for every other estimate

If police search more where crime has just risen, then searching is a response to crime as well as a possible cause of it. A regression of crime on searches then mixes the two directions: the deterrent effect pushes the coefficient down, the allocation response pushes it up, and the estimate is the net of them. A positive coefficient here is the size of the channel that has to be argued away before a stop-crime elasticity or a difference-in-differences estimate can be called causal. A coefficient near zero does not establish exogeneity or rule out other allocation channels.

See Also

Other estimators: `lamp_crimes_prevented()`, `lamp_did_staggered()`, `lamp_displacement_quotient()`, `lamp_elasticity()`, `lamp_event_study()`, `lamp_spillover()`, `lamp_synth()`, `lamp_twfe()`, `tidy.lamp_estimate()`

Examples

```
sim <- lamp_simulate(n_areas = 24, n_months = 30, design = "continuous", seed = 1)
lamp_allocation(sim, crime_lags = 1:2)
```

---

lamp_archive_download *Download force-month files from the data.police.uk archive*

Description

Fetches the street, outcomes and stop-and-search CSV files for chosen months and forces from one or more archive snapshots, without downloading the whole zip: the zip's table of contents is read with an HTTP byte-range request and each wanted file is then fetched by range and inflated locally. Files already present in the cache with a matching checksum are skipped. Each fetched file is recorded with its CRC32 from the archive, a SHA-256 of its content and the time of download, which is what the panel contract reports as provenance.

Usage

```
lamp_archive_download(
  archives = "latest",
  months = NULL,
  forces = NULL,
  file_types = c("street", "outcomes", "stop-and-search"),
  dir = lamp_cache_dir(),
  index = NULL,
  relist = FALSE
)
```

Arguments

archives	Archive identifiers such as `c("2026-06", "2026-07")`, or `"latest"` (the default) for the newest snapshot in `lamp_archive_index()`.
months	Months to fetch as `"YYYY-MM"` strings or Dates; `NULL` for every month in the archive.
forces	police.uk force identifiers such as `"west-yorkshire"`; `NULL` for every force. See `lamp_forces()`.
file_types	Which of `"street"`, `"outcomes"` and `"stop-and-search"` to fetch.
dir	Cache directory; see `lamp_cache_dir()`.
index	An archive index from `lamp_archive_index()`, to avoid fetching it again.
relist	Read the archive's table of contents again even if it has been listed before.

Value

A `lamp_snapshot` (see `lamp_archive_snapshot()`) for `dir`, invisibly.

See Also

Other ingest: `lamp_archive_index()`, `lamp_archive_register()`, `lamp_archive_snapshot()`, `lamp_list_versions()`, `lamp_lsoa_vintage()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`, `lamp_version_diff()`

Examples

```
# Needs network access. Fetches the latest month for one force into a
# temporary directory, and prints a message instead of failing when offline.
cache <- tempfile("streetlamp-")
snap <- tryCatch(
  {
    idx <- lamp_archive_index(dir = cache)
    lamp_archive_download(
      months = format(max(idx$to), "%Y-%m"), forces = "dyfed-powys",
      dir = cache, index = idx
    )
  },
  streetlamp_error_network = function(e) message(conditionMessage(e))
)
snap
```

`lamp_archive_index` *Index of the data.police.uk archive*

Description

Lists the monthly archive snapshots published at https://data.police.uk/data/archive/: one zip per month, each holding the street, outcomes and stop-and-search CSV files for every force and, since the May 2017 snapshot, a rolling window of the latest 36 months (earlier snapshots hold every month back to December 2010). The index is fetched once and cached; pass `refresh = TRUE` to fetch it again. Without network access a cached index is returned with a warning, and an error of class `streetlamp_error_network` is raised if there is no cache.

Usage

```
lamp_archive_index(refresh = FALSE, dir = lamp_cache_dir())
```

Arguments

`refresh`	Fetch the page again even if a cached copy exists.
`dir`	Cache directory; see `lamp_cache_dir()`.

Value

A tibble of class `lamp_archive_index` with one row per snapshot: `archive` (identifier such as `"2026-07"`), `label`, `url`, `from` and `to` (first and last data month in the zip), `n_months`, `size_gb` (as shown on the page) and `md5` (the published checksum of the zip). The attribute `fetched_at` records when the page was read.

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_register()`, `lamp_archive_snapshot()`, `lamp_list_versions()`, `lamp_lsoa_vintage()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`, `lamp_version_diff()`

Examples

```
# Needs network access. The index is cached in a temporary directory, and
# the example prints a message instead of failing when offline.
idx <- tryCatch(
  lamp_archive_index(dir = tempfile("streetlamp-")),
  streetlamp_error_network = function(e) message(conditionMessage(e))
)
if (!is.null(idx)) tail(idx)
```

`lamp_archive_register` *Register a locally downloaded archive zip*

Description

Makes a complete archive zip that was downloaded by other means (for example in a browser) available to `lamp_archive_snapshot()` without copying it: the zip is listed and every member becomes readable in place.

Usage

```
lamp_archive_register(path, dir = lamp_cache_dir(), md5 = NA_character_)
```

Arguments

`path`	Path to a zip named like the archive it came from, for example `2026-07.zip`; the name gives the archive identifier.
`dir`	Cache directory; see `lamp_cache_dir()`.
`md5`	Optional published checksum to record with the archive.

Value

The archive identifier, invisibly.

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_index()`, `lamp_archive_snapshot()`, `lamp_list_versions()`, `lamp_lsoa_vintage()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`, `lamp_version_diff()`

Examples

```
cache <- tempfile("streetlamp-cache-")
zip <- system.file("extdata", "archive", "2026-07.zip", package = "streetlamp")
lamp_archive_register(zip, dir = cache)
lamp_archive_snapshot(cache)
```

`lamp_archive_snapshot` *Snapshot of the archive files held locally*

Description

Describes what the cache directory holds: which archive snapshots have been listed, and for every force-month file in them whether its content is available locally, either because it was fetched by `lamp_archive_download()` or because the whole zip was registered with `lamp_archive_register()`. The snapshot is the starting point for `lamp_list_versions()` and the readers.

Usage

```
lamp_archive_snapshot(dir = lamp_cache_dir())
```

Arguments

`dir`	Cache directory; see `lamp_cache_dir()`.

Value

A list of class `lamp_snapshot` with elements `dir`, `archives` (one row per archive: `archive`, `source`, `kind`, `size`, `etag`, `last_modified`, `md5`, `n_members`, `listed_at`) and `members` (one row per file: `archive`, `member`, `month`, `force_id`, `file_type`, `crc32`, `csize`, `usize`, `available`, `path`, `sha256`, `fetched_at`).

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_index()`, `lamp_archive_register()`, `lamp_list_versions()`, `lamp_lsoa_vintage()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`, `lamp_version_diff()`

Examples

```
cache <- tempfile("streetlamp-cache-")
zips <- list.files(
  system.file("extdata", "archive", package = "streetlamp"),
  pattern = "zip$", full.names = TRUE
)
for (z in zips) lamp_archive_register(z, dir = cache)
snap <- lamp_archive_snapshot(cache)
snap
table(snap$members$force_id, snap$members$file_type)
```

lamp_area_lookup *Area hierarchy for 2021 LSOAs*

Description

One row per 2021 Lower layer Super Output Area in England and Wales with its 2021 Middle layer Super Output Area, 2022 local authority district, police force area and the police.uk force identifier. This is how `lamp_panel()` reaches the `msoa21`, `lad` and `pfa` levels and how it knows which areas belong to a force.

Usage

```
lamp_area_lookup()
```

Details

Source: Office for National Statistics lookups licensed under the Open Government Licence v3.0; see `inst/NOTES/data_sources.md` for the products and retrieval dates.

Value

A tibble with columns `lsoa21`, `lsoa21_name`, `msoa21`, `msoa21_name`, `lad22`, `lad22_name`, `pfa`, `pfa_name` and `force_id`.

See Also

Other bundled data: `lamp_bank_holidays()`, `lamp_crime_types()`, `lamp_deprivation()`, `lamp_forces()`, `lamp_lsoa_lookup()`, `lamp_outcome_types()`, `lamp_sample_panel()`, `lamp_shocks()`

Examples

```
lk <- lamp_area_lookup()
table(lk$force_id[lk$pfa_name == "West Yorkshire"])
length(unique(lk$msoa21))
```

lamp_bank_holidays *Bank holidays in England and Wales*

Description

Bank and public holidays in England and Wales from 2010 onwards, for seasonality controls. Rows from 2012 onwards are taken verbatim from the GOV.UK bank holidays feed (the current feed for 2019 onwards and archived copies of the same feed for 2012 to 2018); 2010 and 2011 are transcribed from the archived Directgov bank holidays page. Apostrophes are normalised to ASCII.

Usage

```
lamp_bank_holidays()
```

Value

A tibble with columns `date` (`Date`), `title`, `notes` (for example `Substitute day`), `bunting` (logical, as published; `NA` for the transcribed 2010 and 2011 rows), `source` and `source_url`.

See Also

Other bundled data: `lamp_area_lookup()`, `lamp_crime_types()`, `lamp_deprivation()`, `lamp_forces()`, `lamp_lsoa_lookup()`, `lamp_outcome_types()`, `lamp_sample_panel()`, `lamp_shocks()`

Examples

```
bh <- lamp_bank_holidays()
range(bh$date)
bh[format(bh$date, "%Y") == "2023", c("date", "title")]
```

---

lamp_boundaries *ONS digital boundaries as an sf layer*

Description

Fetches the generalised, clipped (BGC, 20 m) boundaries for a geography from the ONS Open Geography Portal, in British National Grid (EPSG:27700), and caches them. Products: 2021 LSOAs, 2011 LSOAs, 2021 MSOAs, December 2022 local authority districts (England and Wales only) and December 2023 police force areas. The generalised boundaries suit adjacency and mapping; a search made within a few metres of a boundary can be assigned to the neighbouring area, so supply full-resolution boundaries yourself for point-in-polygon work that needs that precision.

Usage

```
lamp_boundaries(
  type = c("lsoa21", "lsoa11", "msoa21", "lad", "pfa"),
  dir = lamp_cache_dir(),
  refresh = FALSE
)
```

Arguments

`type`	One of `"lsoa21"` (default), `"lsoa11"`, `"msoa21"`, `"lad"` or `"pfa"`.
`dir`	Cache directory; see `lamp_cache_dir()`.
`refresh`	Fetch again even if a cached copy exists.

Details

Without network access a cached copy is returned, and an error of class `streetlamp_error_network` is raised if there is none. A first fetch of the 2021 LSOA layer reads about 50 MB in 18 pages.

Source: Office for National Statistics licensed under the Open Government Licence v3.0; contains OS data, Crown copyright and database right 2024.

Value

An `sf` object with columns `area` (code), `name` and `geometry` (multipolygons, EPSG:27700), sorted by `area`, with attributes `vintage` (for example `"lsoa21-bgc-v5"`), `product` (the ONS title and item identifier) and `retrieved`.

See Also

Other panel: `lamp_adjacency()`, `lamp_adjacency_from_nb()`, `lamp_changelog()`, `lamp_coverage()`, `lamp_panel()`, `lamp_population()`, `lamp_revintage()`

Examples

```
# Needs network access. The boundaries are cached in a temporary directory,
# and the example prints a message instead of failing when offline.
pfa <- tryCatch(
  lamp_boundaries("pfa", dir = tempfile("streetlamp-")),
  streetlamp_error_network = function(e) message(conditionMessage(e))
)
if (!is.null(pfa)) {
  plot(sf::st_geometry(pfa))
  lamp_adjacency(pfa)
}
```

---

lamp_cache_dir *Location of the streetlamp cache*

Description

Downloaded archive files, boundaries and lookups are cached on disk so that each is fetched once. By default the cache is a directory inside the session's temporary directory: it is deleted when R closes, and nothing is written to your file space.

Usage

```
lamp_cache_dir(create = TRUE)

lamp_cache_clear(subdir = NULL)
```

Arguments

create	Create the directory if it does not exist? Default TRUE.
subdir	Optional sub-directory to clear instead of the whole cache, for example "archive".

Details

To keep downloads between sessions, which is worth doing before building a national panel of several gigabytes, choose a directory and set it with the option streetlamp.cache_dir or the environment variable STREETLAMP_CACHE_DIR (the option wins). The conventional place is the user cache directory, for example in .Rprofile:

```
options(streetlamp.cache_dir = tools::R_user_dir("streetlamp", "cache"))
```

lamp_cache_clear() deletes the cache, or one sub-directory of it.

Value

The cache directory path, invisibly for lamp_cache_clear().

Examples

```
# Point the cache at a temporary directory for the duration of the example
old <- options(streetlamp.cache_dir = tempfile("streetlamp-cache-"))
lamp_cache_dir()
lamp_cache_clear()
options(old)
```

---

lamp_changelog *The data.police.uk changelog as a table of refresh and gap notes*

Description

Fetches and parses the publisher's changelog, which records when a force re-supplied months ("data refresh") or failed to supply a month ("not provided"). lamp_coverage() and lamp_panel() use it to mark force-months as refreshed and to annotate gaps. The page is fetched once and cached; pass refresh = TRUE to fetch it again.

Usage

```
lamp_changelog(refresh = FALSE, dir = lamp_cache_dir())
```

Arguments

refresh	Fetch the page again even if a cached copy exists.
dir	Cache directory; see lamp_cache_dir().

Value

A tibble with columns `entry_month` (the changelog heading, a Date), `force_name`, `force_id`, `file_type` (`street`, `outcomes`, `stop-and-search`, a semicolon-separated combination, or `all`), `action` (`refresh` or `not_provided`), `from`, `to` (Dates, `NA` when the entry covers all months) and `text` (the entry as published). Entries that do not follow the refresh or gap patterns are kept with `action = NA`.

See Also

Other panel: `lamp_adjacency()`, `lamp_adjacency_from_nb()`, `lamp_boundaries()`, `lamp_coverage()`, `lamp_panel()`, `lamp_population()`, `lamp_revintage()`

Examples

```
# Needs network access. The page is cached in a temporary directory, and
# the example prints a message instead of failing when offline.
cl <- tryCatch(
  lamp_changelog(dir = tempfile("streetlamp-")),
  streetlamp_error_network = function(e) message(conditionMessage(e))
)
if (!is.null(cl)) cl[cl$force_id == "west-yorkshire", ]
```

---

lamp_colours *The colours streetlamp draws with*

Description

A named vector of the colours every `plot()` method in the package uses, so that figures drawn outside the package can match them. The series colours were chosen and checked as a set for separation under the common forms of colour vision deficiency; the coverage statuses are drawn from the same set so that a status never impersonates a series.

Usage

```
lamp_colours()
```

Value

A named character vector of hex colours. `series` is the main data colour, `contrast` the second series where a plot has two, `accent` the colour for the one mark the reader is meant to compare against (the actual estimate in a placebo distribution), and `neutral` the fill for marks that carry no identity. The `ink`, `secondary`, `muted`, `grid`, `baseline`, `surface` and `shade` entries are the text and chrome colours. The remaining entries are named for coverage statuses.

See Also

Other presentation: `lamp_map()`, `lamp_table()`, `lamp_theme()`

Examples

```
lamp_colours()[c("series", "contrast", "accent")]
```

`lamp_contract` *The panel contract*

Description

Every table streetlamp derives from the archive carries a contract as an attribute: the archive snapshots and versions it was built from, the coverage status of every force-month, the LSOA vintage, the crime scope, the stop-count definition and the treatment definition. `lamp_contract()` returns it; the contract survives subsetting with `[` and dplyr verbs.

Usage

```
lamp_contract(x)
```

Arguments

`x`	A table produced by a `lamp_read_*()` function or by `lamp_panel()`, or a contract.

Value

A list of class `lamp_contract` with elements `source`, `snapshots`, `versions`, `coverage`, `geography`, `population`, `crime_scope`, `stops`, `treatment` and `created`. Elements not yet applicable are `NULL`.

Examples

```
cache <- tempfile("streetlamp-cache-")
zip <- system.file("extdata", "archive", "2026-07.zip", package = "streetlamp")
lamp_archive_register(zip, dir = cache)
crime <- lamp_read_crime(cache, forces = "dyfed-powys")
contract <- lamp_contract(crime)
contract
contract$coverage
```

lamp_coverage *Coverage audit: which force-months exist, and can be compared*

Description

`lamp_coverage()` returns the force by month by file-type grid behind a panel or a records table, with one status per cell (`submitted`, `missing`, `not_read`, `refreshed`, `partial_suspected`), the record count, the archive version used, any changelog note, and a `mismatch` flag for months where a stop-and-search file exists without a street file or the reverse. Its `plot()` method draws the grid as a heat map.

Usage

```
lamp_coverage(x, changelog = NULL)

lamp_coverage_compare(coverage, period_a, period_b, file_type = "street")
```

Arguments

`x`	A `lamp_panel` from `lamp_panel()` or records from `lamp_read_crime()`, `lamp_read_outcomes()` or `lamp_read_stop_counts()`.
`changelog`	Optional `lamp_changelog()` table; entries saying a force re-supplied months mark those months `refreshed`.
`coverage`	A `lamp_coverage` table.
`period_a, period_b`	Two periods, each a length-two vector of months (`"YYYY-MM"` or Dates) giving the first and last month inclusive.
`file_type`	Which file type to compare (default `"street"`).

Details

`lamp_coverage_compare()` summarises comparability per force between two periods: the share of months with usable files, and whether every month in both periods was submitted without refreshes, partial submissions or mismatches.

Value

`lamp_coverage()` returns a tibble of class `lamp_coverage` with columns `force_id`, `month`, `file_type`, `status`, `n_records`, `archive`, `n_versions`, `versions_differ`, `note` and `mismatch`.

`lamp_coverage_compare()` returns a tibble with one row per force: `n_months_a`, `n_usable_a`, `n_months_b`, `n_usable_b` (months with status `submitted` or `refreshed`), `n_partial_a`, `n_partial_b`, `n_refreshed_a`, `n_refreshed_b`, `n_mismatch_a`, `n_mismatch_b` and `comparable` (`TRUE` when every month in both periods is `submitted`, with no mismatch).

See Also

Other panel: lamp_adjacency(), lamp_adjacency_from_nb(), lamp_boundaries(), lamp_changelog(), lamp_panel(), lamp_population(), lamp_revintage()

Examples

```
panel <- lamp_sample_panel()
cov <- lamp_coverage(panel)
cov
table(cov$file_type, cov$status)
# Dyfed-Powys stopped submitting stop-and-search files in December 2025
cov[cov$mismatch %in% TRUE, c("force_id", "month", "file_type", "status")]
lamp_coverage_compare(cov, c("2024-08", "2025-07"), c("2025-08", "2026-07"),
  file_type = "stop-and-search"
)
```

---

lamp_crimes_prevented *Convert an estimate into crimes prevented per thousand searches*

Description

Takes an elasticity or a treatment effect and expresses it as the number of crimes prevented per thousand searches at the sample mean, which is the form in which the question is usually asked. The conversion is arithmetic, not evidence: it inherits every assumption of the estimate it is given, and adds more of its own. The assumption chain is returned with the number and printed with it.

Usage

```
lamp_crimes_prevented(
  estimate,
  panel,
  per_stops = 1000,
  stops_added = NULL,
  outcome_column = NULL,
  n_boot = 2000,
  seed = NULL
)
```

Arguments

estimate	A lamp_estimate from lamp_elasticity(), lamp_twfe(), lamp_did_staggered() or lamp_event_study().
panel	The panel the estimate was built from, for the sample means.
per_stops	Searches to express the result per, default 1000.
stops_added	For a treatment effect: the searches the intervention added per area-month. Required unless the estimate is an elasticity.

outcome_column	The crime column to take the mean of; defaults to the estimate's outcome.
n_boot	Retained for compatibility. Intervals now transform the coefficient interval directly and require no simulation.
seed	Retained for compatibility; the conversion is deterministic.

Details

For an elasticity e, a proportional change in searches dS/S changes crime by e * dS/S, so at mean monthly crime C and mean searches S the crimes prevented by an extra per_stops searches are -e * C * per_stops / S. The elasticity estimators use log(1 + stops), so their derivative has denominator S + 1. The count-mean model uses C in the numerator; the legacy log-one-plus outcome approximation uses C + 1. For a treatment effect in log points b, the effect on crime is C * (exp(b) - 1), and it is divided by the searches the intervention actually added, which you must supply as stops_added because a binary treatment does not say how much searching it involved.

Value

A list of class lamp_crimes_prevented with crimes_prevented, conf_low, conf_high, the inputs used, and assumptions, a character vector naming every step from the estimate to the number.

What this number does not include

It counts recorded crime only, so crimes not reported to the police are invisible to it, and any change in recording practice is counted as a change in crime. It assumes the effect is linear in searches over the range considered, and that the average effect applies at the margin, which is the opposite of what diminishing returns implies. It says nothing about the costs of searching, which fall on the people searched.

See Also

Other estimators: lamp_allocation(), lamp_did_staggered(), lamp_displacement_quotient(), lamp_elasticity(), lamp_event_study(), lamp_spillover(), lamp_synth(), lamp_twfe(), tidy.lamp_estimate()

Examples

```
sim <- lamp_simulate(n_areas = 30, n_months = 30, design = "continuous", seed = 1)
el <- lamp_elasticity(sim, "crime_total", lags = 0:1)
lamp_crimes_prevented(el, sim)
```

---

lamp_crime_types *Crime type classification used by data.police.uk*

Description

data.police.uk assigns every record in the street-level files to one of fourteen categories: anti-social behaviour plus thirteen crime types. This table gives the labels exactly as they appear in the Crime type column of the archive CSV files, a snake_case key used for panel columns, the slug used by the police.uk API, and two package-defined groupings.

Usage

```
lamp_crime_types()
```

Details

Anti-social behaviour (ASB) is not a crime: ASB incidents carry no Crime ID and no outcome, so `is_asb` is `TRUE` and `in_crime_total` is `FALSE` for that row and `lamp_panel()` keeps ASB in its own series.

The category set was verified against the police.uk API and archive files on 2026-09-16. It has been stable since the May 2013 data, when data.police.uk introduced bicycle theft and theft from the person (previously within other theft), split public disorder and weapons into possession of weapons and public order, and renamed violent crime to violence and sexual offences. Files for September 2011 to April 2013 use eleven categories and files for December 2010 to August 2011 only six, with a very broad other crime. `since` gives the first data month of each current category; `lamp_read_crime()` maps legacy labels where the mapping is one to one and keeps `Public disorder and weapons` as its own level.

`group` follows the Home Office offence groups with the six theft categories combined; `broad_group` collapses further into `violent`, `acquisitive`, `drugs`, `damage`, `other` and `asb`. Both groupings are the package's own and are documented here so that results by offence group can be reproduced.

Value

A tibble with one row per category and columns `crime_type`, `key`, `api_slug`, `is_asb`, `in_crime_total`, `group` and `broad_group`.

See Also

`lamp_outcome_types()`

Other bundled data: `lamp_area_lookup()`, `lamp_bank_holidays()`, `lamp_deprivation()`, `lamp_forces()`, `lamp_lsoa_lookup()`, `lamp_outcome_types()`, `lamp_sample_panel()`, `lamp_shocks()`

Examples

```
lamp_crime_types()
```

---

lamp_deprivation *Small-area deprivation indices for England and Wales*

Description

The English Indices of Deprivation 2025 (IoD2025, 33,755 LSOAs) and the Welsh Index of Multiple Deprivation 2025 (WIMD2025, 1,917 LSOAs), both on 2021 LSOAs, as a compact table for covariates and heterogeneity analysis. The two indices are separate national rankings built from different indicators, so ranks and deciles are comparable only within a country; `rank / n_lsoas` gives a within-country percentile. Domain deciles cover the seven domains the two indices share; Wales's community safety domain is held in `decile_crime` and its physical environment domain in `decile_environment`. England's domain deciles are as published; Wales's are formed within Wales from the published domain ranks, with the same thresholds as its published overall decile.

Usage

```
lamp_deprivation()
```

Details

Sources: MHCLG, English indices of deprivation 2025, File 7 (published 30 October 2025, corrected 19 November 2025); Welsh Government, WIMD 2025 index and domain ranks by small area (published 27 November 2025). Both Open Government Licence v3.0; retrieved 2026-09-16.

Value

A tibble with columns `lsoa21`, `country`, `index`, `score` (IMD score, England only), `rank`, `decile` (1 is most deprived), `decile_income`, `decile_employment`, `decile_education`, `decile_health`, `decile_crime`, `decile_housing`, `decile_environment` and `n_lsoas`.

See Also

Other bundled data: `lamp_area_lookup()`, `lamp_bank_holidays()`, `lamp_crime_types()`, `lamp_forces()`, `lamp_lsoa_lookup()`, `lamp_outcome_types()`, `lamp_sample_panel()`, `lamp_shocks()`

Examples

```
dep <- lamp_deprivation()
table(dep$country, dep$decile)[, 1:3]
```

---

lamp_design_audit *Audit the support and assumptions of an intervention design*

Description

Reports missing submissions, pre/post support and detection-sensitive outcomes. A clean data audit does not establish causal identification; intervention assignment, contemporaneous changes and spillovers still require substantive evidence.

Usage

```
lamp_design_audit(
  panel,
  treatment = NULL,
  outcomes = "crime_total",
  min_pre = 6L,
  min_post = 6L
)
```

Arguments

panel	A lamp_panel.
treatment	Optional dated lamp_treatment() object.
outcomes	Outcome columns under consideration.
min_pre, min_post	Minimum observed months per treated area before and after adoption for the support audit.

Value

A list with issues (check, status, detail), area_support, and summary. Statuses describe data support or required arguments, not certification of causality.

See Also

Other diagnostics: lamp_effect_summary(), lamp_elasticity_robustness(), lamp_placebo(), lamp_pretrends(), lamp_trend_sensitivity()

Examples

```
sim <- lamp_simulate(design = "event", seed = 1)
lamp_design_audit(sim)
```

---

lamp_detect_surges *Detect surges in stop and search activity*

Description

Flags area-months in which searches rise far above the area's recent level, and returns them as a candidate staggered-adoption table. Two methods are offered: a threshold of k standard deviations above the area's trailing twelve-month mean, and a structural break test from strucchange.

Usage

```
lamp_detect_surges(
  panel,
  method = c("threshold", "breakpoints"),
  k = 2,
  min_months = 12L,
  min_stops = 3L
)
```

Arguments

panel	A lamp_panel with a stops column.
method	"threshold" (default) or "breakpoints".
k	Threshold in standard deviations above the trailing mean.
min_months	Months of history required before an area can be flagged.
min_stops	Minimum searches in the month, so that tiny areas do not surge from one search to three.

Value

A tibble of class lamp_surges with one row per area: area, adoption_month (the first flagged month, NA if none), n_flagged, trailing_mean, trailing_sd, peak_stops and peak_ratio (peak over trailing mean). Pass it to lamp_treatment() as adoption for a staggered design. The attribute flags holds every flagged area-month.

Endogeneity

A surge detected from the stop series is not an experiment. Police deploy searches where crime has risen, so an area's first surge month is correlated with its recent crime history and with anything else that prompted the deployment. Treating a detected surge as exogenous will mistake the response for the cause. Use this to explore where activity changed and to pick candidate cases for a design with an external source of variation, not as the primary design; and read lamp_allocation() alongside any estimate that uses it.

See Also

Other treatment: lamp_treatment()

Examples

```
sim <- lamp_simulate(n_areas = 16, n_months = 30, design = "event", seed = 3)
surges <- lamp_detect_surges(sim, min_stops = 5)
head(surges)
```

---

lamp_did_staggered *Difference-in-differences with staggered adoption*

Description

Estimates the effect of an intervention that different areas adopt at different dates, using an estimator that compares each adopting cohort with areas that have not yet adopted rather than with areas already treated. Plain two-way fixed effects does the latter and can return an estimate outside the range of every area's true effect; this function exists so that staggered designs are not analysed that way.

Usage

```
lamp_did_staggered(
  panel,
  outcome = "crime_total",
  treatment,
  estimator = c("callaway_santanna", "sun_abraham", "imputation"),
  control_group = c("never_treated", "not_yet_treated"),
  cluster = c("area", "force"),
  family = c("ols_log", "ols_ihs", "identity"),
  window = c(-12, 12)
)
```

Arguments

panel	A lamp_panel from lamp_panel().
outcome	The outcome column, for example "crime_total" or a crime type key from lamp_crime_types().
treatment	A lamp_treatment() object of type staggered (or event, which is a single cohort).
estimator	"callaway_santanna", "sun_abraham" or "imputation".
control_group	"never_treated" (default) or "not_yet_treated".
cluster	"area" or "force". Callaway–Sant'Anna uses the multiplier bootstrap for force clustering (the backend default of 1000 draws); set the R random seed for reproduction. Sun–Abraham uses clustered regression covariance. Force clustering is not implemented for the imputation wrapper and is rejected.
family	Scale for the outcome: "ols_log" (default, log of one plus the outcome), "ols_ihs" or "identity".
window	Relative months to report in the dynamic aggregation.

Value

A lamp_estimate of class lamp_did_staggered. coefficients holds the dynamic (event-study) aggregation with a rel_time column; diagnostics$overall is the single summary effect with its standard error; diagnostics$group_time holds the group-time effects for the Callaway and Sant'Anna backend; and diagnostics$pretrend_p is that backend's pre-test.

Identifying assumption

Parallel trends holds for each adopting cohort against the chosen control group, from the period before that cohort adopts onward, and there is no anticipation: the outcome does not move before adoption in response to it. With control_group = "never_treated" the comparison is against areas that never adopt, which must exist; with "not_yet_treated" it is against areas that adopt later, which uses more data but assumes their later adoption carries no information about their current path.

Backends

- `callaway_santanna` (default) calls `did::att_gt()` and `did::aggte()`, giving one effect per cohort and period and the aggregations of them.
- `sun_abraham` uses `fixest::sunab()`, an interaction-weighted estimator fitted in one regression.
- `imputation` uses the `didimputation` package, which fits the untreated potential outcome and imputes: efficient, but it needs that package, which is in Suggests.

Counts are modelled on a transformed scale by all three backends, which is why the function warns when a count outcome meets `ols_log` or `ols_ihs`: the coefficients are effects on that scale.

References

Callaway, B. and Sant'Anna, P. H. C. (2021). Difference-in-differences with multiple time periods. Journal of Econometrics 225(2), 200-230.

Sun, L. and Abraham, S. (2021). Estimating dynamic treatment effects in event studies with heterogeneous treatment effects. Journal of Econometrics 225(2), 175-199.

Borusyak, K., Jaravel, X. and Spiess, J. (2024). Revisiting event-study designs: robust and efficient estimation. Review of Economic Studies 91(6).

See Also

Other estimators: `lamp_allocation()`, `lamp_crimes_prevented()`, `lamp_displacement_quotient()`, `lamp_elasticity()`, `lamp_event_study()`, `lamp_spillover()`, `lamp_synth()`, `lamp_twfe()`, `tidy.lamp_estimate()`

Examples

```
# small: most of this example's cost is loading the did package, which
# whichever example reaches it first has to pay
sim <- lamp_simulate(n_areas = 24, n_months = 18, design = "staggered", effect = -0.3, seed = 1)
ad <- attr(sim, "truth")$adoption
tr <- lamp_treatment(sim, "staggered", adoption = ad[!is.na(ad$adoption_month), ])
fit <- lamp_did_staggered(sim, "crime_total", tr)
fit
fit$diagnostics$overall
```

---

lamp_displacement_quotient

Weighted displacement quotient

Description

Measures whether crime prevented in a treated area reappeared in the areas around it. The quotient compares the change in a buffer of surrounding areas with the change in control areas, relative to the change in the treated areas, following Bowers and Johnson (2003).

Usage

```
lamp_displacement_quotient(
  panel,
  outcome = "crime_total",
  treated_areas,
  buffer_areas,
  control_areas,
  pre,
  post,
  n_boot = 1000,
  seed = NULL
)
```

Arguments

panel	A lamp_panel.
outcome	The outcome column.
treated_areas, buffer_areas, control_areas	Area identifiers.
pre, post	Two-element vectors giving the first and last month of the before and after periods.
n_boot	Bootstrap replications over areas, default 1000.
seed	Random seed.

Details

The statistic is WDQ = ((B1/C1) - (B0/C0)) / ((A1/C1) - (A0/C0)), where A, B and C are total crime in the treated, buffer and control areas and the subscripts are the pre and post periods. Read it as:

- negative: crime moved into the buffer, so the gain in the treated area is partly or wholly displacement;
- between 0 and 1: the buffer improved too, a diffusion of benefit smaller than the treated area's gain;
- above 1: the buffer improved more than the treated area.

The quotient is a descriptive ratio, not a causal estimate: it assumes the control areas show what would have happened everywhere, and it is unstable when the treated-area change is close to zero, which the bootstrap interval will show as a very wide range.

Value

A list of class lamp_wdq with wdq, conf_low, conf_high, the underlying totals, success (the treated area's change net of controls) and interpretation.

References

Bowers, K. J. and Johnson, S. D. (2003). Measuring the geographical displacement and diffusion of benefit effects of crime prevention activity. Journal of Quantitative Criminology 19(3), 275-301.

See Also

Other estimators: lamp_allocation(), lamp_crimes_prevented(), lamp_did_staggered(), lamp_elasticity(), lamp_event_study(), lamp_spillover(), lamp_synth(), lamp_twfe(), tidy.lamp_estimate()

Examples

```
sim <- lamp_simulate(
  n_areas = 36, n_months = 20, design = "spillover",
  effect = -0.4, spillover_effect = 0.2, seed = 2
)
truth <- attr(sim, "truth")
areas <- truth$lattice$area
lamp_displacement_quotient(
  sim, "crime_total",
  treated_areas = truth$treated_areas[1:6],
  buffer_areas = truth$treated_areas[7:12],
  control_areas = setdiff(areas, truth$treated_areas),
  pre = c("2018-01", "2018-06"), post = c("2018-12", "2019-05"), n_boot = 50
)
```

---

lamp_effect_summary *Summarise effects using their joint covariance*

Description

Computes a linear combination with the full covariance matrix. For staggered estimates, the default is the backend's overall ATT and its own weighting, rather than an equally weighted mean of dynamic estimates.

Usage

```
lamp_effect_summary(estimate, terms = NULL, weights = NULL, level = 0.95)
```

Arguments

estimate	A lamp_estimate.
terms	Coefficient names; NULL selects all elasticity lags, post-event coefficients for an event study, or the treatment coefficient.
weights	Numeric weights in the order of terms. Defaults to one for elasticity lags and equal averaging weights otherwise. Supplied weights are used as supplied, without normalisation.
level	Confidence level between zero and one.

Value

A one-row tibble with estimate, standard error, interval, confidence level and the aggregation definition. Unsupported joint covariance is an error, rather than an independence approximation.

See Also

Other diagnostics: lamp_design_audit(), lamp_elasticity_robustness(), lamp_placebo(), lamp_pretrends(), lamp_trend_sensitivity()

Examples

```
sim <- lamp_simulate(design = "continuous", seed = 1)
lamp_effect_summary(lamp_elasticity(sim, lags = 0:1))
```

---

lamp_elasticity *Elasticity of crime with respect to stop and search*

Description

Estimates how recorded crime responds to the intensity of searching, as a distributed lag of log-one-plus stops with area and month fixed effects. Three methods are offered, differing in how they handle the fact that areas move together.

Usage

```
lamp_elasticity(
  panel,
  outcome = "crime_total",
  stops = "stops",
  lags = 0:3,
  method = c("fe", "cce_mg", "cce_pooled"),
  cluster = c("area", "force"),
  min_months = 12L,
  family = NULL
)
```

Arguments

panel	A lamp_panel from lamp_panel().
outcome	The outcome column, for example "crime_total" or a crime type key from lamp_crime_types().
stops	The stop intensity column, default "stops".
lags	Lags of log stops to include, default 0 to 3.
method	"fe", "cce_mg" or "cce_pooled".
cluster	"area" (default) or "force".
min_months	Months an area needs before it enters a mean-group regression.
family	NULL selects "poisson" for fe and "ols_log" for the CCE methods. Poisson pseudo-likelihood models the conditional count mean, including zeros, with clustered inference. CCE currently supports only "ols_log". Specify "ols_log" to reproduce the former FE scale.

Value

A `lamp_estimate` of class `lamp_elasticity`. Coefficients are the lag-by-lag elasticities; `diagnostics$long_run` is their sum with a standard error, `diagnostics$cd_test` is Pesaran's test, and `diagnostics$area_coefficients` holds the per-area slopes for the mean group estimator. The CD test sums over every pair of areas, so above two thousand areas it is computed on a random sample of them, drawn with a fixed seed: `cd_test$sampled` says whether that happened and `cd_test$n_areas_used` how many areas went in.

What an elasticity here is and is not

With `family = "poisson"`, a coefficient describes the log conditional mean of the count per unit of `log(1 + stops)`. Its elasticity with respect to stops at intensity S is `coefficient * S / (1 + S)`. With `family = "ols_log"`, it describes `log(1 + observed crime)` instead; this is a different estimand, particularly when counts are small. Neither is automatically the effect of a decision to search more. Police send officers where crime is rising, so the association runs in both directions; `lamp_allocation()` measures that reverse channel and should be reported alongside. Read the elasticity as a description of the joint movement unless the variation in searching has an argued external source.

Methods

- `fe`: area and month fixed effects only.
- `cce_mg`: the common correlated effects mean group estimator. The cross-sectional averages of the outcome and the regressors are added to each area's own time-series regression, and the area coefficients are averaged; the standard error is the spread of those coefficients, which makes no assumption that areas share a slope.
- `cce_pooled`: one pooled regression with the same averages added, which is more precise if the slope really is common.

Pesaran's CD test is an auxiliary descriptive diagnostic in every case. It uses OLS residuals of log-one-plus counts with area effects only, including when the primary estimator is Poisson, because month dummies would remove the common factor by construction and make the statistic negative whatever the data looked like. A large statistic flags shared movements; it does not by itself prove that clustered intervals are invalid.

References

Pesaran, M. H. (2006). Estimation and inference in large heterogeneous panels with a multifactor error structure. Econometrica 74(4), 967-1012.

Chudik, A. and Pesaran, M. H. (2015). Common correlated effects estimation of heterogeneous dynamic panel data models with weakly exogenous regressors. Journal of Econometrics 188(2), 393-420.

See Also

Other estimators: `lamp_allocation()`, `lamp_crimes_prevented()`, `lamp_did_staggered()`, `lamp_displacement_quot`
`lamp_event_study()`, `lamp_spillover()`, `lamp_synth()`, `lamp_twfe()`, `tidy.lamp_estimate()`

Examples

```
sim <- lamp_simulate(n_areas = 30, n_months = 36, design = "continuous", seed = 1)
fit <- lamp_elasticity(sim, "crime_total", lags = 0:1)
fit
fit$diagnostics$cd_test$statistic
```

`lamp_elasticity_robustness` *Evaluate an explicit elasticity specification grid*

Description

Runs every requested outcome, lag window and family, retaining failures and optional leave-one-area-out estimates. Register the primary outcome and specification before examining the grid. These are dependent, exploratory checks; selecting a favourable row does not establish a cause. The OLS and Poisson rows target different quantities and should not be interpreted as estimates of one common elasticity.

Usage

```
lamp_elasticity_robustness(
  panel,
  outcomes = "crime_total",
  lags = list(current = 0L, short = 0:1, distributed = 0:3),
  families = c("poisson", "ols_log"),
  cluster = c("area", "force"),
  leave_one_out = FALSE
)
```

Arguments

`panel`	A `lamp_panel`.
`outcomes`	Outcome column names.
`lags`	A named list of non-negative integer lag vectors.
`families`	Character vector containing `"poisson"`, `"ols_log"`, or both.
`cluster`	`"area"` or `"force"`.
`leave_one_out`	Also fit each specification after omitting each area. This can be expensive on large panels.

Value

A tibble with every requested specification, omitted area, status, error message, lag-sum estimate and interval, and observed sample sizes.

See Also

Other diagnostics: lamp_design_audit(), lamp_effect_summary(), lamp_placebo(), lamp_pretrends(), lamp_trend_sensitivity()

Examples

```
sim <- lamp_simulate(design = "continuous", n_areas = 12, seed = 1)
lamp_elasticity_robustness(sim, lags = list(current = 0, short = 0:1))
```

`lamp_event_study` *Event study around a dated intervention*

Description

Estimates one coefficient per month relative to the intervention, so that the path of the outcome before and after the event is visible rather than summarised in a single number. The months before the event are the pre-trend test: if the outcome was already moving, the parallel-trends assumption behind any difference-in-differences estimate is in doubt.

Usage

```
lamp_event_study(
  panel,
  outcome = "crime_total",
  event,
  window = c(-12, 12),
  reference = -1,
  cluster = c("area", "force"),
  family = c("poisson", "negbin", "ols_log", "ols_ihs"),
  controls = NULL,
  staggered_ok = FALSE
)
```

Arguments

`panel`	A `lamp_panel` from lamp_panel().
`outcome`	The outcome column, for example `"crime_total"` or a crime type key from lamp_crime_types().
`event`	A lamp_treatment() object of type `event` or `binary`, or the name of a panel column holding relative time.
`window`	The first and last relative month to estimate, default `c(-12, 12)`. Months outside it are pooled into endpoint bins so that they still contribute to the fixed effects.
`reference`	The omitted relative month, default `-1` (the month before the event).
`cluster`	`"area"` (default) or `"force"`.

family	"poisson" (default, a Poisson pseudo-likelihood suited to counts), "negbin", "ols_log" or "ols_ihs".
controls	Optional panel columns to include as covariates.
staggered_ok	With several adoption dates, delegate to lamp_did_staggered() instead of stopping.

Value

A lamp_estimate of class lamp_event_study. Its coefficients carry a rel_time column, and diagnostics$pretrend_p holds the joint test that every pre-period coefficient is zero.

Identifying assumption

In the absence of the event, treated areas would have followed the same path as controls, net of area and month fixed effects. The pre-period coefficients test a necessary consequence of this, not the assumption itself: passing the test does not establish parallel trends, and with few areas the test has little power to detect a trend that matters. See lamp_pretrends().

Staggered adoption

When areas adopt at different dates, relative-time dummies in a two-way fixed effects regression use already-treated areas as controls and are biased. This function never does that: with more than one adoption date it stops and explains, or, with staggered_ok = TRUE, hands the work to lamp_did_staggered() and says so. The result is then a lamp_did_staggered estimate, not an event study.

See Also

Other estimators: lamp_allocation(), lamp_crimes_prevented(), lamp_did_staggered(), lamp_displacement_quot
lamp_elasticity(), lamp_spillover(), lamp_synth(), lamp_twfe(), tidy.lamp_estimate()

Examples

```
sim <- lamp_simulate(n_areas = 24, n_months = 24, design = "event", effect = -0.25, seed = 4)
truth <- attr(sim, "truth")
# half the areas are treated; the rest are the comparison group
tr <- lamp_treatment(
  sim, "event",
  date = truth$event_date, scope = truth$treated_areas, window = c(-6, 6)
)
es <- lamp_event_study(sim, "crime_total", tr, window = c(-6, 6))
es
plot(es)
```

`lamp_forces` *Police forces in the data.police.uk archive*

Description

The 45 forces whose files appear in the archive: the 43 territorial forces of England and Wales, the Police Service of Northern Ireland and British Transport Police, with the identifier used in archive file names, the force name as published by the police.uk API, and the ONS police force area code (December 2025) where one exists.

Usage

```
lamp_forces()
```

Details

British Transport Police has no police force area; its records carry `British Transport Police` in the `Falls within` column rather than the territorial force in whose area the offence occurred. Greater Manchester Police has submitted no data since 2019 and is absent from recent archive snapshots.

Sources: https://data.police.uk/api/forces and the ONS Police Force Areas (December 2025) Names and Codes in the United Kingdom, both retrieved 2026-09-16; Open Government Licence v3.0.

Value

A tibble with columns `force_id`, `name`, `pfa_code`, `pfa_name`, `country` (`england`, `wales`, `northern_ireland` or `NA`) and `notes`.

See Also

Other bundled data: `lamp_area_lookup()`, `lamp_bank_holidays()`, `lamp_crime_types()`, `lamp_deprivation()`, `lamp_lsoa_lookup()`, `lamp_outcome_types()`, `lamp_sample_panel()`, `lamp_shocks()`

Examples

```
lamp_forces()
```

`lamp_list_versions` *List the archive versions of every force-month file*

Description

List the archive versions of every force-month file

Usage

```
lamp_list_versions(snapshot, file_types = NULL)
```

Arguments

`snapshot`	A `lamp_snapshot` from `lamp_archive_snapshot()`.
`file_types`	Restrict to some of `"street"`, `"outcomes"` and `"stop-and-search"`; `NULL` for all.

Value

A tibble of class `lamp_versions` with one row per force, month and file type: `n_versions`, `archives` (list-column of archive identifiers, oldest first), `crc32` (list-column, aligned with `archives`), `differs` (`TRUE` when the versions do not all share one checksum), `usize_min`, `usize_max`, and `n_available` (versions readable locally).

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_index()`, `lamp_archive_register()`, `lamp_archive_snapshot()`, `lamp_lsoa_vintage()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`, `lamp_version_diff()`

Examples

```
cache <- tempfile("streetlamp-cache-")
zips <- list.files(
  system.file("extdata", "archive", package = "streetlamp"),
  pattern = "zip$", full.names = TRUE
)
for (z in zips) lamp_archive_register(z, dir = cache)
v <- lamp_list_versions(lamp_archive_snapshot(cache))
v[v$n_versions > 1, c("force_id", "month", "file_type", "n_versions", "differs")]
```

`lamp_lsoa_lookup` *ONS LSOA code lists and the 2011 to 2021 lookup*

Description

The Office for National Statistics lookup between 2011 and 2021 Lower layer Super Output Areas (LSOAs) in England and Wales, with the ONS change indicator, the 2022 local authority district, and a flag marking the single 2021 LSOA that the ONS best-fit lookup assigns to each 2011 LSOA. `lamp_lsoa_codes()` returns the complete code list for either vintage, which is what `lamp_lsoa_vintage()` matches against.

Usage

```
lamp_lsoa_lookup()

lamp_lsoa_codes(vintage = c("lsoa21", "lsoa11"))
```

Arguments

`vintage`	Which code list: `"lsoa21"` (default) or `"lsoa11"`.

Details

`lamp_lsoa_lookup()` is the ONS exact-fit lookup (version 3): one row per 2011 to 2021 pair, so a 2011 LSOA that was split appears on several rows and a merged 2021 LSOA appears on several rows. `change` is the ONS change indicator: `U` unchanged, `S` split, `M` merged, `X` complex. `best_fit` is `TRUE` on the row chosen by the ONS best-fit lookup (version 2), which assigns every 2011 LSOA to exactly one 2021 LSOA; re-vintaging counts with the `best_fit` rows is deterministic but leaves the 2021 LSOAs that no 2011 LSOA best-fits to without data.

Source: Office for National Statistics licensed under the Open Government Licence v3.0. Retrieved 2026-09-16; see `inst/NOTES/data_sources.md`.

Value

`lamp_lsoa_lookup()` returns a tibble with columns `lsoa11`, `lsoa11_name`, `lsoa21`, `lsoa21_name`, `change` (factor), `lad22`, `lad22_name` and `best_fit`. `lamp_lsoa_codes()` returns a sorted character vector of codes.

See Also

Other bundled data: `lamp_area_lookup()`, `lamp_bank_holidays()`, `lamp_crime_types()`, `lamp_deprivation()`, `lamp_forces()`, `lamp_outcome_types()`, `lamp_sample_panel()`, `lamp_shocks()`

Examples

```
length(lamp_lsoa_codes("lsoa21"))
length(lamp_lsoa_codes("lsoa11"))
table(lamp_lsoa_lookup()$change)
```

`lamp_lsoa_vintage` *Detect whether LSOA codes are 2011 or 2021 vintage*

Description

Matches codes against the bundled ONS code lists (`lamp_lsoa_codes()`). Codes that exist in only one vintage decide the answer; when a file has none of those (possible for very small forces), the publication month decides, because data.police.uk moved to 2021 codes with the June 2023 data.

Usage

```
lamp_lsoa_vintage(x, month = NULL)
```

Arguments

`x`	A character vector of LSOA codes, or a records table from `lamp_read_crime()` or `lamp_read_outcomes()` (which is assessed one file at a time).
`month`	Optional month (Date or `"YYYY-MM"`) used only when the codes are ambiguous and `x` is a character vector.

Value

A tibble with one row per file (or one row for a vector) and columns `vintage` (`"lsoa11"`, `"lsoa21"`, `"mixed"` or `"ambiguous"`), `basis` (`"codes"` or `"month"`), `n_codes`, `n_only_2011`, `n_only_2021`, `n_both` and `n_unknown`, preceded by `archive`, `file`, `force_id` and `month` for records.

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_index()`, `lamp_archive_register()`, `lamp_archive_snapshot()`, `lamp_list_versions()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`, `lamp_version_diff()`

Examples

```
lamp_lsoa_vintage(c("E01000001", "E01000002", "E01035000"))
lamp_lsoa_vintage(c("E01000001", "E01000002"), month = "2024-01")
```

`lamp_map` *Map a panel column, an area-level estimate or a synthetic control's donors*

Description

Draws a choropleth of the areas in a panel. The geometry comes from `lamp_boundaries()` or `lamp_sample_boundaries()`; the fill comes from a panel column, from the per-area slopes of a mean-group elasticity, or from the donor weights of a synthetic control.

Usage

```
lamp_map(x, ...)

## S3 method for class 'lamp_panel'
lamp_map(
  x,
  column = "crime_total",
  boundaries,
  month = NULL,
  statistic = c("mean", "total", "rate"),
  per = 1000,
  border = NA,
  extent = c("panel", "all"),
  ...
)

## S3 method for class 'lamp_elasticity'
lamp_map(x, boundaries, border = NA, extent = c("panel", "all"), ...)

## S3 method for class 'lamp_synth'
lamp_map(x, boundaries, border = NA, extent = c("panel", "all"), ...)

## S3 method for class 'lamp_estimate'
lamp_map(x, ...)
```

Arguments

`x`	A `lamp_panel`, a `lamp_elasticity` fitted with `method = "cce_mg"`, or a `lamp_synth` fit.
`...`	Passed to the method.
`column`	The panel column to map.
`boundaries`	An `sf` object with an `area` column, from `lamp_boundaries()` or `lamp_sample_boundaries()`.
`month`	A single month to map, as a date or `"YYYY-MM"`. `NULL`, the default, aggregates every month in the panel.

statistic	How to combine months when month is NULL: the area's "mean" per month, its "total" over the period, or "rate", the total per per residents per month.
per	Denominator for statistic = "rate".
border	Colour for area borders, or NA for none. At lower-layer geography the borders swamp the fill, so the default is none.
extent	"panel", the default, draws only the areas the data covers. "all" keeps every boundary supplied, and the caption then counts the areas outside the panel separately from those whose data is missing.

Value

A ggplot.

Missing is not zero

An area whose value cannot be computed, because every month of it was a force-month with no file or because it is absent from the panel, is filled with the missing colour from lamp_colours() and given its own legend entry. It is never shaded as a low value. The caption counts those areas, and counts areas present in one of the panel and the boundaries but not the other, so that a map with a hole in it says why.

See Also

Other presentation: lamp_colours(), lamp_table(), lamp_theme()

Examples

```
panel <- lamp_sample_panel()
boundaries <- lamp_sample_boundaries()
# one local authority, so that the example draws quickly; a whole force is
# a few seconds and a national panel a few minutes
leeds <- boundaries$area[grepl("^Leeds", boundaries$name)]
lamp_map(panel[panel$area %in% leeds, ], "stop_rate", boundaries)
```

---

lamp_outcome_types *Outcome classification used by data.police.uk*

Description

The outcome categories that appear in the Outcome type column of the outcomes files and the Last outcome category column of the street files, with the police.uk API code and a package-defined grouping into six classes.

Usage

```
lamp_outcome_types()
```

Details

The grouping is the package's own:

- `charged_or_summonsed`: the suspect was charged or summonsed, including every subsequent court outcome supplied by the Ministry of Justice and offences taken into consideration (`Suspect charged as part of another case`).
- `out_of_court`: cautions, drugs possession warnings, penalty notices and community (local) resolutions.
- `no_suspect`: investigation complete with no suspect identified.
- `evidential_difficulties`: a suspect was identified but the police were unable to prosecute.
- `other`: action not in the public interest or taken by another body.
- `unknown`: still under investigation or no status update available.

`is_court_outcome` marks the categories that come from court records rather than from the police. data.police.uk reports that court outcomes from June 2019 onwards are unavailable, so these categories are sparse after that date and `charged_or_summonsed` should be read as the charge stage.

The category list was verified against the police.uk API documentation and against archive files on 2026-09-16. `outcome_type` is the string written in the archive CSV files; where the API documentation names a category differently (`Offender given penalty notice` appears there as `Offender given a penalty notice`) the API name is kept in `api_name` and `lamp_read_outcomes()` accepts either.

Value

A tibble with columns `outcome_type`, `api_name`, `code`, `group` (factor with the six levels above) and `is_court_outcome`.

See Also

`lamp_crime_types()`

Other bundled data: `lamp_area_lookup()`, `lamp_bank_holidays()`, `lamp_crime_types()`, `lamp_deprivation()`, `lamp_forces()`, `lamp_lsoa_lookup()`, `lamp_sample_panel()`, `lamp_shocks()`

Examples

```
lamp_outcome_types()
```

`lamp_panel` *Build an area-by-month panel of crime and stops*

Description

Turns crime records into a balanced area-by-month panel with one column per crime type, a crime total, an anti-social behaviour series, stop counts, population and a stop rate, and attaches a full panel contract. The panel covers every area in the police force areas of the forces present in `crime` (or the areas you name) and every month in the records.

Usage

```
lamp_panel(
  crime,
  outcomes = NULL,
  stops = NULL,
  area = c("lsoa21", "lsoa11", "msoa21", "lad", "pfa"),
  population = NULL,
  covariates = NULL,
  include_asb = FALSE,
  attribution = c("falls_within", "reported_by"),
  areas = NULL,
  adjacency = NULL,
  changelog = NULL
)
```

Arguments

crime	Records from lamp_read_crime().
outcomes	Optional records from lamp_read_outcomes(); adds outcomes_total and one column per outcome group, counted by the month the outcome was recorded.
stops	Optional stop counts from lamp_read_stop_counts() at the panel's area level (or at lsoa21 level, which is aggregated), or a user-supplied tibble with columns area, month, stops and optionally stops_s60 and coverage_status, which is tagged user_supplied in the contract and never zero-filled.
area	The area level: "lsoa21" (default), "lsoa11", "msoa21", "lad" (2022 local authority districts) or "pfa" (police force areas).
population	Optional population by area: the output of lamp_population() or a tibble with columns area and population. Needed for stop_rate (stops per 1,000 residents).
covariates	Optional tibble with an area column (and optionally month) whose other columns are joined to the panel.
include_asb	Include anti-social behaviour in crime_total?
attribution	Which column names the force a record belongs to: "falls_within" (default) or "reported_by". Areas are always placed by location; attribution only affects the diagnostics on records located outside their force's area and the pfa level, where records are counted by the attributed force.
areas	Optional character vector restricting the panel to these areas (identifiers at the chosen level).
adjacency	Optional lamp_adjacency() object for the panel's areas, stored in the contract for spatial estimators.
changelog	Optional lamp_changelog() table used to mark force-months that the publisher's changelog says were refreshed.

Details

Records are placed by their LSOA code. Files published with 2011 codes are re-vintaged to 2021 with the bundled ONS lookup (or 2021 to 2011 when `area = "lsoa11"`), and the contract's `geography$revintage` element records how many records and areas were unchanged, split, merged or complex. Coarser areas (`msoa21`, `lad`, `pfa`) are reached through `lamp_area_lookup()`.

"Force did not submit" is never a zero: counts are zero-filled only in force-months whose street file was submitted; force-months with no file are `NA` and carry `coverage_status = "missing"`. Force-months whose record count is suspiciously low are kept but flagged `partial_suspected`; see `lamp_coverage()`.

Anti-social behaviour is not a crime: it goes to the `asb` column and is excluded from `crime_total` unless `include_asb = TRUE`. The crime type columns are the snake_case keys of `lamp_crime_types()`; the legacy `public_disorder_and_weapons` column appears only when records from before May 2013 are present.

Value

A tibble of class `lamp_panel` with columns `area`, `month`, `force_id`, one column per crime type, `crime_total`, `asb`, `stops`, `stops_s60`, `population`, `stop_rate`, `coverage_status`, `stops_status`, any outcome and covariate columns, and a contract retrievable with `lamp_contract()`.

See Also

Other panel: `lamp_adjacency()`, `lamp_adjacency_from_nb()`, `lamp_boundaries()`, `lamp_changelog()`, `lamp_coverage()`, `lamp_population()`, `lamp_revintage()`

Examples

```
cache <- tempfile("streetlamp-cache-")
zip <- system.file("extdata", "archive", "2026-07.zip", package = "streetlamp")
lamp_archive_register(zip, dir = cache)
crime <- lamp_read_crime(cache, forces = "dyfed-powys", months = "2026-07")
panel <- lamp_panel(crime, area = "lad")
panel
lamp_contract(panel)$coverage

# The bundled sample is a full two-force LSOA panel over 24 months
lamp_sample_panel()
```

`lamp_placebo` *Placebo tests for a streetlamp estimate*

Description

Re-estimates the same specification under conditions in which the true effect is zero, and compares the real estimate with that distribution. An estimate that looks the same when the treatment is moved to the wrong date, the wrong areas, or an outcome the intervention cannot plausibly affect, is measuring something other than the intervention.

Usage

```
lamp_placebo(
  estimate,
  type = c("time", "space", "outcome"),
  n = 200,
  outcome = "bicycle_theft",
  seed = NULL
)
```

Arguments

estimate	A lamp_estimate.
type	"time", "space" or "outcome".
n	Number of placebo draws for type = "space".
outcome	Placebo outcome for type = "outcome", default "bicycle_theft".
seed	Random seed for type = "space".

Details

Types:

- time: the event is moved back to each month that leaves a full pre-period, and only pre-period data is used, so nothing real happens at the fake date.
- space: the same number of areas is drawn at random, keeping the real dates, and the treatment is reassigned to them.
- outcome: the same specification is run on a crime type the intervention is not expected to move (bicycle theft by default), which tests for anything that shifts recorded crime generally, such as a recording change.

The reported p value is the share of placebo estimates at least as large in absolute value as the real one. It is a randomisation p value, not a test of a specific null, and with few areas it is coarse.

Value

A list of class lamp_placebo with type, actual (the real estimate), distribution (a tibble of placebo estimates), p_value, n_valid and interpretation.

See Also

Other diagnostics: lamp_design_audit(), lamp_effect_summary(), lamp_elasticity_robustness(), lamp_pretrends(), lamp_trend_sensitivity()

Examples

```
sim <- lamp_simulate(n_areas = 20, n_months = 24, design = "event", effect = -0.3, seed = 6)
truth <- attr(sim, "truth")
tr <- lamp_treatment(sim, "event", date = truth$event_date, scope = truth$treated_areas)
fit <- lamp_twfe(sim, "crime_total", tr)
lamp_placebo(fit, type = "space", n = 20, seed = 1)
```

`lamp_population` *Census 2021 usual resident population by area*

Description

Fetches the Census 2021 usual resident population of every 2021 LSOA in England and Wales from the NOMIS API (table TS001, dataset `NM_2021_1`), caches it, and returns it at the requested area level. Coarser levels are sums of LSOA values through `lamp_area_lookup()`; because ONS applies cell-key perturbation to small-area counts, these sums differ from the published MSOA and district totals by a few persons. 2011 LSOAs receive the population of the 2021 LSOAs that map to them: unchanged and split LSOAs pass their value to their single parent, and a 2021 LSOA formed by a merger shares its value equally among its parents, so `lsoa11` values are approximate.

Usage

```
lamp_population(
  area = c("lsoa21", "msoa21", "lad", "pfa", "lsoa11"),
  dir = lamp_cache_dir(),
  refresh = FALSE
)
```

Arguments

`area`	The area level: `"lsoa21"` (default), `"msoa21"`, `"lad"`, `"pfa"` or `"lsoa11"`.
`dir`	Cache directory; see `lamp_cache_dir()`.
`refresh`	Fetch from NOMIS again even if a cached copy exists.

Details

Without network access a cached download is returned, and an error of class `streetlamp_error_network` is raised if there is none.

Value

A tibble with columns `area`, `name` (`NA` for aggregated levels) and `population`, with attributes `area` (the level) and `source` (a list naming the NOMIS dataset, the census, the geography type, the retrieval time and the aggregation rule) that `lamp_panel()` copies into the contract.

See Also

Other panel: `lamp_adjacency()`, `lamp_adjacency_from_nb()`, `lamp_boundaries()`, `lamp_changelog()`, `lamp_coverage()`, `lamp_panel()`, `lamp_revintage()`

Examples

```
# Needs network access. The download is cached in a temporary directory,
# and the example prints a message instead of failing when offline.
pop <- tryCatch(
  lamp_population("pfa", dir = tempfile("streetlamp-")),
  streetlamp_error_network = function(e) message(conditionMessage(e))
)
if (!is.null(pop)) head(pop)
```

`lamp_pretrends` *Test and interpret pre-trends*

Description

Reports the joint test that every pre-period coefficient of an event study is zero, and answers the question the test cannot: how large a pre-trend could have been present without this test detecting it, and how much bias that undetected pre-trend would put into the post-period estimates.

Usage

```
lamp_pretrends(estimate, power = c(0.5, 0.8), level = 0.05)
```

Arguments

`estimate`	A `lamp_estimate` from `lamp_event_study()`.
`power`	Detection probabilities to report, default 0.5 and 0.8.
`level`	Significance level of the pre-trend test, default 0.05.

Details

The power calculation follows the argument in Roth (2022) and is implemented here rather than taken from another package. Under a linear violation of parallel trends with slope `s` per month, the expected pre-period coefficients are `s` times their relative time (measured from the reference month), so the Wald statistic is non-central chi-square with non-centrality `s^2 * t' V^-1 t`, where `t` is that vector of relative times and `V` the clustered covariance of the pre-period coefficients. Inverting this gives the slope the test detects with a given probability. The bias column shows what that slope would add to each post-period coefficient if it continued, which is the quantity that matters: a test that passes while remaining blind to a trend big enough to explain the result is not reassurance.

Value

A list of class `lamp_pretrends` with `test` (statistic, degrees of freedom and p value), `coefficients` (the pre-period coefficients), `power` (a tibble of `power`, `slope` in log points per month, and `bias_mean_post`, the bias such a slope would add to the mean post-period coefficient) and `interpretation`.

References

Roth, J. (2022). Pretest with caution: event-study estimates after testing for parallel trends. American Economic Review: Insights 4(3), 305-322.

See Also

Other diagnostics: `lamp_design_audit()`, `lamp_effect_summary()`, `lamp_elasticity_robustness()`, `lamp_placebo()`, `lamp_trend_sensitivity()`

Examples

```
sim <- lamp_simulate(n_areas = 24, n_months = 24, design = "event", effect = -0.25, seed = 5)
truth <- attr(sim, "truth")
tr <- lamp_treatment(
  sim, "event",
  date = truth$event_date, scope = truth$treated_areas, window = c(-6, 6)
)
es <- lamp_event_study(sim, "crime_total", tr, window = c(-6, 6))
pt <- lamp_pretrends(es)
pt
```

`lamp_read_crime` *Read street-level crime records from the archive*

Description

Reads the selected `street` files into one table with a fixed snake_case schema, harmonises crime type labels across the three category sets that data.police.uk has used since December 2010, flags anti-social behaviour, detects the LSOA vintage of every file, and attaches a panel contract.

Usage

```
lamp_read_crime(
  dir = lamp_cache_dir(),
  versions = NULL,
  months = NULL,
  forces = NULL,
  fetch = FALSE
)
```

Arguments

`dir`	A cache directory (see `lamp_cache_dir()`) or a `lamp_snapshot` from `lamp_archive_snapshot()`.
`versions`	A `lamp_selection` from `lamp_select_version()`; by default the newest archive version of each file is used.
`months, forces`	Optional filters, as in `lamp_archive_download()`.

`fetch` Fetch selected files that are listed in the snapshot but not held locally (needs network access). With the default `FALSE` only the files held locally are read, with a message saying how many were skipped, so a reader never starts a large download on its own.

Details

Crime types are returned as a factor whose levels are the fourteen current categories plus `Public disorder and weapons`, the legacy category that was split in May 2013 and cannot be mapped forward. `Violent crime`, the pre-2013 name of `Violence and sexual offences`, is renamed. The raw label is kept in `crime_type_raw`, and the contract's `crime_scope$category_sets` records which set each file uses (`six` for December 2010 to August 2011, `eleven` to April 2013, `fourteen` since May 2013).

Anti-social behaviour rows have no Crime ID and no outcome; they are kept with `is_asb = TRUE` so that `lamp_panel()` can hold them in a separate series. Rows without coordinates or LSOA code are kept as `NA`.

Value

A tibble of class `lamp_records` with columns `crime_id`, `month` (Date, first of month), `reported_by`, `falls_within`, `force_id` (from the file name), `longitude`, `latitude`, `location`, `lsoa_code`, `lsoa_name`, `lsoa_vintage`, `crime_type` (factor), `crime_type_raw`, `is_asb`, `last_outcome_category`, `context`, `archive` and `file`, and a contract retrievable with `lamp_contract()` whose `coverage` element marks force-months with no file as `missing`.

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_index()`, `lamp_archive_register()`, `lamp_archive_snapshot()`, `lamp_list_versions()`, `lamp_lsoa_vintage()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`, `lamp_version_diff()`

Examples

```
cache <- tempfile("streetlamp-cache-")
zip <- system.file("extdata", "archive", "2026-07.zip", package = "streetlamp")
lamp_archive_register(zip, dir = cache)
crime <- lamp_read_crime(cache, forces = "dyfed-powys")
crime
table(crime$crime_type)
lamp_contract(crime)$coverage
```

---

lamp_read_outcomes *Read outcome records from the archive*

Description

Reads the selected `outcomes` files into one table with a fixed schema, classifies outcomes with `lamp_outcome_types()`, detects the LSOA vintage of every file, and attaches a panel contract. An outcomes file for a month lists the outcomes recorded in that month, whose crimes may date from earlier months, and a crime can have several outcomes.

Usage

```
lamp_read_outcomes(
  dir = lamp_cache_dir(),
  versions = NULL,
  months = NULL,
  forces = NULL,
  fetch = FALSE
)
```

Arguments

dir	A cache directory (see lamp_cache_dir()) or a lamp_snapshot from lamp_archive_snapshot().
versions	A lamp_selection from lamp_select_version(); by default the newest archive version of each file is used.
months, forces	Optional filters, as in lamp_archive_download().
fetch	Fetch selected files that are listed in the snapshot but not held locally (needs network access). With the default FALSE only the files held locally are read, with a message saying how many were skipped, so a reader never starts a large download on its own.

Value

A tibble of class lamp_records with columns crime_id, month, reported_by, falls_within, force_id, longitude, latitude, location, lsoa_code, lsoa_name, lsoa_vintage, outcome_type (factor), outcome_type_raw, outcome_group (factor with the six groups of lamp_outcome_types()), archive and file, plus a contract.

See Also

Other ingest: lamp_archive_download(), lamp_archive_index(), lamp_archive_register(), lamp_archive_snapshot(), lamp_list_versions(), lamp_lsoa_vintage(), lamp_read_crime(), lamp_read_stop_counts(), lamp_s60_legislation(), lamp_select_version(), lamp_version_diff()

Examples

```
cache <- tempfile("streetlamp-cache-")
zip <- system.file("extdata", "archive", "2026-07.zip", package = "streetlamp")
lamp_archive_register(zip, dir = cache)
outcomes <- lamp_read_outcomes(cache, forces = "dyfed-powys", months = "2026-07")
table(outcomes$outcome_group)
```

`lamp_read_stop_counts` *Count stops by area and month*

Description

Reads the selected `stop-and-search` files and counts searches by area and month, with a Section 60 flag. Only the `Date`, `Latitude`, `Longitude` and `Legislation` columns are read; the package never parses the ethnicity, object, outcome or other fields. Area assignment is by point-in-polygon in British National Grid (EPSG:27700) against boundaries you supply; without boundaries, counts are per force (the force that submitted the file). Stops are dated by the month of the file they were published in: the publisher stores local time as UTC, so a handful of searches made after 23:00 on the first night of a summer-time month carry the previous month's date, and the count of such rows is reported per file as `n_outside_month`.

Usage

```
lamp_read_stop_counts(
  dir = lamp_cache_dir(),
  versions = NULL,
  area = "force",
  boundaries = NULL,
  months = NULL,
  forces = NULL,
  fetch = FALSE
)
```

Arguments

`dir`	A cache directory (see `lamp_cache_dir()`) or a `lamp_snapshot` from `lamp_archive_snapshot()`.
`versions`	A `lamp_selection` from `lamp_select_version()`; by default the newest archive version of each file is used.
`area`	`"force"` (default) for force-month counts, or `"lsoa21"`, `"lsoa11"` or another label describing the polygons in `boundaries`.
`boundaries`	An `sf` polygon layer with an `area` column giving the area identifier; required unless `area = "force"`. `lamp_boundaries()` provides ONS LSOA boundaries.
`months, forces`	Optional filters, as in `lamp_archive_download()`.
`fetch`	Fetch selected files that are listed in the snapshot but not held locally (needs network access). With the default `FALSE` only the files held locally are read, with a message saying how many were skipped, so a reader never starts a large download on its own.

Value

A tibble of class `lamp_records` with one row per area and month: `area`, `month`, `force_id` (the submitting force), `stops`, `stops_s60` and `archive`; for `area = "force"` also `stops_no_location` (searches without coordinates, which cannot be placed in an area) and `stops_no_legislation`.

The contract's stops element records the origin (streetlamp), the definition (raw count), the area type and per-file diagnostics, and its coverage element marks force-months with no stop-and-search file as missing.

See Also

Other ingest: lamp_archive_download(), lamp_archive_index(), lamp_archive_register(), lamp_archive_snapshot(), lamp_list_versions(), lamp_lsoa_vintage(), lamp_read_crime(), lamp_read_outcomes(), lamp_s60_legislation(), lamp_select_version(), lamp_version_diff()

Examples

```
cache <- tempfile("streetlamp-cache-")
zip <- system.file("extdata", "archive", "2026-07.zip", package = "streetlamp")
lamp_archive_register(zip, dir = cache)
stops <- lamp_read_stop_counts(cache, forces = c("west-yorkshire", "dyfed-powys"))
stops
lamp_contract(stops)$coverage
```

---

lamp_report *Build a report on a panel and the estimates made from it*

Description

Renders a document that puts a set of estimates next to the things that decide whether they mean anything: the coverage of the panel they came from, the identifying assumption of each estimator, the pre-trend and placebo diagnostics, and the allocation model. The report is deliberately hard to read as a simple verdict, because the underlying question does not have one.

Usage

```
lamp_report(
  panel,
  estimates,
  file = NULL,
  title = "streetlamp report",
  diagnostics = TRUE,
  figures = NULL,
  quiet = FALSE
)
```

Arguments

panel	A lamp_panel.
estimates	A named list of lamp_estimate objects, or a single estimate. Names become section headings.
file	Output file. The extension decides the format: .html or .md. NULL writes an HTML file to a temporary location.

title	Report title.
diagnostics	Include the pre-trend and placebo sections where the estimates support them.
figures	Draw each estimate's plot() as a PNG into a _figures directory beside the report and include it. NULL, the default, draws figures for an HTML report (where they are also embedded in the page, so the file stands alone) and not for a Markdown one. A figure that cannot be drawn is skipped, never an error.
quiet	Suppress rendering output.

Value

The path to the written file, invisibly.

Examples

```
sim <- lamp_simulate(n_areas = 20, n_months = 20, design = "event", seed = 1)
truth <- attr(sim, "truth")
tr <- lamp_treatment(sim, "event", date = truth$event_date, scope = truth$treated_areas)
out <- lamp_report(
  sim,
  list("Two-way fixed effects" = lamp_twfe(sim, "crime_total", tr)),
  file = tempfile(fileext = ".md"), quiet = TRUE
)
file.exists(out)
```

---

lamp_revintage *Re-vintage LSOA codes between 2011 and 2021*

Description

Converts a vector of LSOA codes from one vintage to the other with the bundled ONS lookup (lamp_lsoa_lookup()) and reports how many areas were unchanged, split, merged or complex. From 2011 to 2021 each code maps to its best-fit 2021 LSOA. From 2021 to 2011 unchanged and split LSOAs map to their single parent; a 2021 LSOA formed by a merger maps to the first of its parents in code order, so lsoa21 panels are the recommended target.

Usage

```
lamp_revintage(codes, from = c("lsoa11", "lsoa21"), to = c("lsoa21", "lsoa11"))
```

Arguments

codes	A character vector of LSOA codes.
from, to	The vintages, "lsoa11" or "lsoa21".

Value

A list with codes (the mapped vector, NA where a code is not in the lookup) and summary (counts of records and distinct areas by change type and the share of areas that were split or merged).

See Also

Other panel: lamp_adjacency(), lamp_adjacency_from_nb(), lamp_boundaries(), lamp_changelog(), lamp_coverage(), lamp_panel(), lamp_population()

Examples

```
r <- lamp_revintage(c("E01000001", "E01000002", "E01033768"), from = "lsoa11", to = "lsoa21")
r$codes
r$summary$share_split_or_merged
```

---

lamp_s60_legislation *Legislation label for Section 60 searches*

Description

The exact string that data.police.uk uses in the `Legislation` column of stop-and-search files for searches under section 60 of the Criminal Justice and Public Order Act 1994 (verified 2026-09-16).

Usage

```
lamp_s60_legislation()
```

Value

A string.

See Also

Other ingest: lamp_archive_download(), lamp_archive_index(), lamp_archive_register(), lamp_archive_snapshot(), lamp_list_versions(), lamp_lsoa_vintage(), lamp_read_crime(), lamp_read_outcomes(), lamp_read_stop_counts(), lamp_select_version(), lamp_version_diff()

Examples

```
lamp_s60_legislation()
```

lamp_sample_panel *The bundled sample panel and boundaries*

Description

A ready-made lamp_panel() for West Yorkshire and Dyfed-Powys: every 2021 LSOA in the two police force areas over the 24 most recent complete months in the July 2026 archive snapshot (August 2024 to July 2026), with crime by type, anti-social behaviour, stops and Section 60 stops assigned by location, Census 2021 population, stop rate, and deprivation deciles as covariates. Its contract records the archive versions, coverage, geography and adjacency it was built with. lamp_sample_boundaries() returns the matching ONS generalised boundaries, simplified to 50 m for bundling, as an sf layer in British National Grid.

Usage

```
lamp_sample_panel()

lamp_sample_boundaries()
```

Details

The panel is aggregated counts, not records, and is meant for examples, tests and the vignettes. Data: data.police.uk, Office for National Statistics and NOMIS, Open Government Licence v3.0; see inst/extdata/README.md.

Value

lamp_sample_panel() returns a lamp_panel; lamp_sample_boundaries() an sf object with columns area, name and geometry.

See Also

Other bundled data: lamp_area_lookup(), lamp_bank_holidays(), lamp_crime_types(), lamp_deprivation(), lamp_forces(), lamp_lsoa_lookup(), lamp_outcome_types(), lamp_shocks()

Examples

```
panel <- lamp_sample_panel()
panel
lamp_coverage(panel)
bnd <- lamp_sample_boundaries()
nrow(bnd)
```

`lamp_select_version` *Select one archive version per force-month file*

Description

Select one archive version per force-month file

Usage

```
lamp_select_version(versions, rule = c("latest", "earliest"), prefer = NULL)
```

Arguments

`versions`	Output of `lamp_list_versions()`.
`rule`	`"latest"` (default) takes the newest archive snapshot holding the file, on the grounds that later snapshots carry the most complete outcome information and any re-supplied data; `"earliest"` takes the oldest, which is the version closest to first publication.
`prefer`	Optional archive identifiers to use whenever they hold the file, before applying `rule` to the rest.

Value

A tibble of class `lamp_selection` with one row per force, month and file type: the chosen `archive`, its `crc32` and `usize`, `n_versions`, `differs`, `alternatives` (list-column of tibbles naming the other archives with their checksums and sizes) and `rule`.

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_index()`, `lamp_archive_register()`, `lamp_archive_snapshot()`, `lamp_list_versions()`, `lamp_lsoa_vintage()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_version_diff()`

Examples

```
cache <- tempfile("streetlamp-cache-")
zips <- list.files(
  system.file("extdata", "archive", package = "streetlamp"),
  pattern = "zip$", full.names = TRUE
)
for (z in zips) lamp_archive_register(z, dir = cache)
v <- lamp_list_versions(lamp_archive_snapshot(cache))
sel <- lamp_select_version(v, rule = "latest")
sel[, c("force_id", "month", "file_type", "archive", "n_versions")]
```

lamp_shocks *National policy shocks affecting stop and search or recorded crime*

Description

A curated table of dated national events for use as default intervention definitions in `lamp_treatment()`: the Best Use of Stop and Search Scheme, the 2019 and later changes to Section 60 authorisation conditions, COVID-19 lockdown periods, changes to crime recording practice, and other verified policy changes. Every row carries the source that establishes its dates; see `inst/NOTES/data_sources.md` in the package sources for the verification record.

Usage

```
lamp_shocks()
```

Value

A tibble with columns `name` (identifier), `label`, `kind` (`stop_search_policy`, `crime_policy`, `covid`, `recording_practice` or `disorder`), `start` and `end` (`Date`; `end` is `NA` for point events or regimes still in force), `scope` (`national`, `england`, `wales` or `forces`), `forces` (semicolon-separated police.uk force identifiers when `scope` is `forces`, otherwise `NA`), `source_url`, `notes` and `verified` (`Date` the source was checked).

See Also

Other bundled data: `lamp_area_lookup()`, `lamp_bank_holidays()`, `lamp_crime_types()`, `lamp_deprivation()`, `lamp_forces()`, `lamp_lsoa_lookup()`, `lamp_outcome_types()`, `lamp_sample_panel()`

Examples

```
shocks <- lamp_shocks()
shocks[shocks$kind == "stop_search_policy", c("name", "start", "end")]
```

lamp_simulate *Simulate a panel with a known treatment effect*

Description

Generates an area-by-month panel from a known data-generating process, for testing estimators and for the simulation evidence in the vignettes. Crime counts are Poisson draws around area and month fixed effects, and the treatment multiplies the mean of the affected crime types by `exp(effect)`, so `effect` is a log point effect: -0.1 is a 9.5 percent reduction.

Usage

```
lamp_simulate(
  n_areas = 36L,
  n_months = 24L,
  design = c("staggered", "event", "continuous", "spillover"),
  effect = -0.15,
  spillover_effect = 0.05,
  adoption = NULL,
  missing = 0,
  base_rate = 8,
  population = 1500,
  seed = NULL,
  cohort_effects = NULL,
  dynamic_effect = NULL,
  trend_violation = 0,
  dispersion = Inf,
  missing_mechanism = c("random", "high_outcome"),
  effect_scale = c("log", "additive")
)
```

Arguments

`n_areas`	Number of areas.
`n_months`	Number of months.
`design`	One of `"staggered"`, `"event"`, `"continuous"` or `"spillover"`.
`effect`	True treatment effect in log points on the affected crime types (for `continuous`, the elasticity of crime with respect to `log(stops + 1)`).
`spillover_effect`	True neighbour effect in log points, used by the `spillover` design.
`adoption`	Adoption months for the `staggered` design, as month indices within the panel; `NULL` picks three evenly spaced cohorts.
`missing`	Share of force-months with no street file (0 to 1).
`base_rate`	Expected monthly count per area before fixed effects.
`population`	Population per area, used for `stop_rate`.
`seed`	Random seed.
`cohort_effects`	Optional named numeric vector, with adoption month indices as names (e.g. `c("9" = -0.4, "13" = -0.1, "17" = 0.2)`). Replaces `effect` for those cohorts; requires the staggered design and exactly one value for every adopting cohort.
`dynamic_effect`	Optional finite numeric multipliers for relative months 0, 1, ...; the last multiplier persists thereafter. For example `c(0, 0.5, 1)` gives a delayed, growing effect. Binary designs only.
`trend_violation`	Differential monthly trend for ever-treated areas, in the same units as `effect`. Zero preserves the identifying assumption.

`dispersion` Negative-binomial size per crime type; `Inf` (default) gives Poisson counts. Smaller positive values give more overdispersion.

`missing_mechanism` `"random"` or `"high_outcome"`. The latter makes force-months with larger untreated expected counts more likely to be missing, as a stress test rather than an ignorable missingness claim.

`effect_scale` `"log"` (default) multiplies affected count means. `"additive"` adds `effect` counts to the combined affected outcome and uses an additive area/time baseline, so count-level parallel trends hold when `trend_violation = 0`. Non-positive type means are rejected.

Details

Designs:

- `event`: one date; the areas in the treated half are treated from that month onward. Suits `lamp_event_study()`.
- `staggered`: areas adopt in cohorts spread across the middle of the period, with never-treated areas kept as controls. Plain two-way fixed effects are biased here, which is what `lamp_did_staggered()` is for.
- `continuous`: every area has a stop intensity that varies over time, and the outcome responds to `log(stops + 1)` with elasticity `effect`.
- `spillover`: as `event`, and in addition the mean treatment of an area's first-order lattice neighbours shifts its outcome by `spillover_effect`, so an estimator that omits the neighbour term is biased.

Areas sit on a square lattice and belong to two forces, so that force-month missingness, clustering and adjacency are all exercisable. A share `missing` of force-months has no street file: those rows carry `NA` counts and `coverage_status = "missing"`, exactly as a real panel does. Bicycle theft is never affected by treatment, which makes it the default placebo outcome in `lamp_placebo()`.

Value

A `lamp_panel` with the usual columns plus `treat` (the treatment variable) and, for the event and staggered designs, `rel_time` and `cohort`. Its contract carries the treatment definition, and `attr(x, "truth")` holds the true `effect`, the implied effect on the crime total (`effect_crime_total`, smaller in size than `effect` because bicycle theft is unaffected), `spillover_effect`, `design`, the affected crime types, the treated areas, the event date, the adoption table and the neighbour list.

See Also

Other simulation: `lamp_simulation_truth()`

Examples

```
sim <- lamp_simulate(n_areas = 36, n_months = 24, design = "event", effect = -0.2)
sim
attr(sim, "truth")$effect
table(sim$coverage_status)
```

`lamp_simulation_truth` *Recover exact expected outcomes from a simulated panel*

Description

Keeps count-level effects and log mean ratios distinct. This function deliberately does not treat a log mean as the expectation of log counts.

Usage

```
lamp_simulation_truth(
  panel,
  outcome = c("affected_crime", "crime_total"),
  scale = c("identity", "log_mean")
)
```

Arguments

`panel`	A panel returned by `lamp_simulate()`.
`outcome`	`"affected_crime"` or `"crime_total"`.
`scale`	`"identity"` for expected count differences or `"log_mean"` for log ratios of expected counts.

Value

A tibble of area-month counterfactual means and exact effects, including cohort, relative time and observed coverage status. For spillover designs the effect includes the simulated neighbour exposure.

See Also

Other simulation: `lamp_simulate()`

Examples

```
sim <- lamp_simulate(seed = 1)
head(lamp_simulation_truth(sim))
```

`lamp_spillover` *Own-area and neighbour effects estimated together*

Description

Adds the treatment of an area's neighbours to the specification, so that the effect on the area itself and the effect on its neighbours are estimated in one model. This is what separates a reduction in crime from its displacement next door: if searching an area pushes offending into the surrounding areas, the own-area coefficient is negative and the neighbour coefficient positive, and the net effect is their sum.

Usage

```
lamp_spillover(
  panel,
  outcome = "crime_total",
  treatment,
  adjacency = NULL,
  rings = 1:2,
  cluster = c("area", "force"),
  family = c("poisson", "negbin", "ols_log", "ols_ihs"),
  controls = NULL
)
```

Arguments

`panel`	A `lamp_panel` from `lamp_panel()`.
`outcome`	The outcome column, for example `"crime_total"` or a crime type key from `lamp_crime_types()`.
`treatment`	A `lamp_treatment()` object, or the name of a panel column.
`adjacency`	A `lamp_adjacency()` object; taken from the panel's contract when not given.
`rings`	Contiguity rings to include, default first and second order.
`cluster`	`"area"` (default) or `"force"`.
`family`	`"poisson"` (default, a Poisson pseudo-likelihood suited to counts), `"negbin"`, `"ols_log"` or `"ols_ihs"`.
`controls`	Optional panel columns to include as covariates.

Value

A `lamp_estimate` of class `lamp_spillover`, whose coefficients hold the own-area effect and one neighbour effect per ring, and whose `diagnostics$net` gives the implied net effect (own plus neighbours) with a standard error from the full covariance matrix.

Identifying assumption

Parallel trends as in two-way fixed effects, and in addition that the treatment reaches an area only through its own treatment and that of the rings included. If spillovers extend beyond the last ring, the areas used as controls are themselves affected, and every coefficient is biased toward zero.

See Also

Other estimators: lamp_allocation(), lamp_crimes_prevented(), lamp_did_staggered(), lamp_displacement_quot
lamp_elasticity(), lamp_event_study(), lamp_synth(), lamp_twfe(), tidy.lamp_estimate()

Examples

```
sim <- lamp_simulate(
  n_areas = 49, n_months = 20, design = "spillover",
  effect = -0.3, spillover_effect = 0.15, seed = 1
)
adj <- lamp_adjacency_from_nb(attr(sim, "truth")$nb, attr(sim, "truth")$lattice$area)
truth <- attr(sim, "truth")
tr <- lamp_treatment(sim, "event", date = truth$event_date, scope = truth$treated_areas)
fit <- lamp_spillover(sim, "crime_total", tr, adjacency = adj, rings = 1)
fit
```

---

lamp_synth *Synthetic control for a single treated area or force*

Description

Builds a weighted average of untreated donor units whose pre-period path matches the treated unit's, and reads the gap between them afterwards as the effect. Inference is by permutation: the same procedure is run pretending each donor was treated, and the treated unit's post-period fit is compared with that distribution.

Usage

```
lamp_synth(
  panel,
  outcome = "crime_total",
  treated_unit,
  treatment_start,
  donors = NULL,
  method = c("internal", "synth", "gsynth"),
  placebo = TRUE,
  min_pre = 6L
)
```

Arguments

panel	A lamp_panel.
outcome	The outcome column.
treated_unit	The area that was treated.
treatment_start	The first treated month.
donors	Candidate donor areas; NULL uses every other area that has a complete series.
method	"internal" (default), "synth" or "gsynth".
placebo	Run in-space placebos over the donors for the permutation p value.
min_pre	Months of pre-period required.

Value

A list of class lamp_synth with weights, path (a tibble of month, observed, synthetic and gap), effect (mean post-period gap), rmspe_pre, rmspe_post, rmspe_ratio, p_value (the share of donors with a ratio at least as large), placebos and assumption.

Identifying assumption

The weighted donors reproduce what the treated unit would have done without the intervention. This is credible only when the pre-period fit is close over a long window, no donor was itself affected by the intervention, and nothing else happened to the treated unit at the same time. A poor pre-period fit invalidates the comparison, which is why the pre-period root mean squared error is reported first.

References

Abadie, A., Diamond, A. and Hainmueller, J. (2010). Synthetic control methods for comparative case studies. Journal of the American Statistical Association 105(490), 493-505.

Xu, Y. (2017). Generalized synthetic control method. Political Analysis 25(1), 57-76.

See Also

Other estimators: lamp_allocation(), lamp_crimes_prevented(), lamp_did_staggered(), lamp_displacement_quot
lamp_elasticity(), lamp_event_study(), lamp_spillover(), lamp_twfe(), tidy.lamp_estimate()

Examples

```
sim <- lamp_simulate(n_areas = 20, n_months = 30, design = "event", effect = -0.4, seed = 1)
truth <- attr(sim, "truth")
sc <- lamp_synth(
  sim, "crime_total",
  treated_unit = truth$treated_areas[1],
  treatment_start = truth$event_date,
  donors = setdiff(unique(sim$area), truth$treated_areas),
  placebo = FALSE
)
sc
```

`lamp_table` *A presentation-ready table from a streetlamp object*

Description

Formats an estimate, a diagnostic or a coverage audit as a tibble of character columns with readable headers, rounded figures, confidence intervals in brackets and p values as `<0.001` where they are that small. The result is meant to be passed to `knitr::kable()` or a similar table renderer; the unrounded numbers remain available from `tidy()` and from the object itself.

Usage

```
lamp_table(x, ...)

## S3 method for class 'lamp_estimate'
lamp_table(x, digits = 3, ...)

## S3 method for class 'lamp_event_study'
lamp_table(x, digits = 3, ...)

## S3 method for class 'lamp_did_staggered'
lamp_table(x, digits = 3, ...)

## S3 method for class 'lamp_pretrends'
lamp_table(x, digits = 3, ...)

## S3 method for class 'lamp_synth'
lamp_table(x, digits = 3, ...)

## S3 method for class 'lamp_coverage'
lamp_table(x, ...)
```

Arguments

`x`	An estimate from any of the package's estimators, a `lamp_pretrends()` result, a `lamp_synth()` fit, or a `lamp_coverage()` audit.
`...`	Unused.
`digits`	Significant figures for estimates and intervals.

Value

A tibble of character columns. For estimates: the term, the estimate, its standard error, the 95 percent confidence interval and the p value. Event studies and staggered fits show months relative to the event in place of the term. For a pre-trend diagnostic: the power table. For a synthetic control: donor weights. For a coverage audit: counts of force-months by file type and status.

See Also

Other presentation: `lamp_colours()`, `lamp_map()`, `lamp_theme()`

Examples

```
sim <- lamp_simulate(n_areas = 24, n_months = 24, design = "event", effect = -0.2, seed = 2)
truth <- attr(sim, "truth")
tr <- lamp_treatment(sim, "event", date = truth$event_date, scope = truth$treated_areas)
lamp_table(lamp_twfe(sim, "crime_total", tr))
lamp_table(lamp_coverage(sim))
```

`lamp_theme` *The ggplot2 theme streetlamp plots use*

Description

A quiet theme built on `ggplot2::theme_minimal()`: recessive hairline grid, no minor grid or tick marks, left-aligned titles, muted axis text, and the legend above the plot. Every `plot()` method in the package applies it, and it can be added to any other ggplot.

Usage

```
lamp_theme(base_size = 11)
```

Arguments

`base_size`	Base font size in points.

Value

A ggplot2 theme object.

See Also

Other presentation: `lamp_colours()`, `lamp_map()`, `lamp_table()`

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(colour = lamp_colours()[["series"]]) +
  labs(title = "Weight and fuel economy") +
  lamp_theme()
```

`lamp_treatment` *Define the treatment*

Description

Turns a panel into a treatment definition that the estimators can use: a continuous stop intensity, a binary intervention in named areas and months, staggered adoption dates by area, or a dated event applied to every area. The result records how the treatment was built, is stored in the contract of every estimate made with it, and is printed by the estimators alongside their identifying assumption.

Usage

```
lamp_treatment(
  panel,
  type = c("continuous", "binary", "staggered", "event"),
  measure = c("stop_rate", "stops", "stops_per_100_crimes"),
  transform = c("identity", "log", "ihs"),
  areas = NULL,
  window = NULL,
  adoption = NULL,
  event = NULL,
  date = NULL,
  scope = NULL
)
```

Arguments

`panel`	A `lamp_panel` from `lamp_panel()`.
`type`	`"continuous"`, `"binary"`, `"staggered"` or `"event"`.
`measure, transform`	For `continuous`: which intensity and which transformation.
`areas`	For `binary`: the treated areas.
`window`	For `binary`: the first and last treated month. For `event`: the event window in months, default `c(-12, 12)`.
`adoption`	For `staggered`: a table with columns `area` and `adoption_month`.
`event`	For `event`: a name in `lamp_shocks()`, or `NULL` when `date` is given.
`date`	For `event`: the event month, overriding the shocks table.
`scope`	For `event`: areas the event applies to; `NULL` for all.

Details

Types:

- `continuous`: `measure` is `"stops"`, `"stop_rate"` (per 1,000 residents, the default) or `"stops_per_100_crimes"` (searches per 100 crimes recorded in the area's previous twelve months, which removes the mechanical dependence on population). `transform` is `"identity"`, `"log"` (log of one plus the measure) or `"ihs"` (inverse hyperbolic sine, which handles zeros).
- `binary`: `areas` are treated during `window`, a pair of months, and untreated outside it; every other area is a control.
- `staggered`: `adoption` is a table of `area` and `adoption_month`. Areas absent from it are never treated. `rel_time` (months since adoption) and `cohort` are added.
- `event`: a dated shock applied to every area in scope, from `lamp_shocks()` by `event` name or from `date`, with an event `window` in months around it. `rel_time` is months since the event.

A continuous treatment is a dose, not an experiment: the allocation of searches to areas responds to crime, so estimates from it describe an association unless the variation is argued to be exogenous. See `lamp_allocation()`.

Value

A `lamp_treatment` object: a list with `type`, `column` (the treatment variable's name), `data` (area, month, the treatment column and, where applicable, `rel_time` and `cohort`), `details`, `n_treated_areas`, `n_control_areas` and `created`.

See Also

Other treatment: `lamp_detect_surges()`

Examples

```
sim <- lamp_simulate(n_areas = 16, n_months = 18, design = "event", seed = 1)

# a dated event, twelve months either side
lamp_treatment(sim, "event", date = "2018-10", window = c(-6, 6))

# stop intensity as a continuous dose
lamp_treatment(sim, "continuous", measure = "stop_rate", transform = "ihs")

# a surge in named areas
lamp_treatment(sim, "binary", areas = unique(sim$area)[1:4], window = c("2018-06", "2018-12"))
```

---

lamp_trend_sensitivity

Sensitivity to a specified differential linear trend

Description

Adjusts an event-study average for user-specified untreated trend slopes. Each slope is an assumption in the model's outcome units per month, not a trend estimated and treated as known. Intervals reflect sampling uncertainty conditional on each slope. This is a transparent linear bias calculation, not a general robust confidence procedure or a test of parallel trends.

Usage

```
lamp_trend_sensitivity(
  estimate,
  slopes,
  periods = NULL,
  reference = -1L,
  level = 0.95
)
```

Arguments

estimate	An estimate from lamp_event_study().
slopes	Finite numeric differential trend assumptions.
periods	Post-event relative months to average; defaults to available post-event months.
reference	Reference relative month; must match the fitted model.
level	Confidence level.

Value

A tibble with the assumed slope, implied bias and adjusted estimate and interval for each scenario.

See Also

Other diagnostics: lamp_design_audit(), lamp_effect_summary(), lamp_elasticity_robustness(), lamp_placebo(), lamp_pretrends()

Examples

```
sim <- lamp_simulate(design = "event", seed = 1)
truth <- attr(sim, "truth")
tr <- lamp_treatment(sim, "event", date = truth$event_date, scope = truth$treated_areas)
fit <- lamp_event_study(sim, "crime_total", tr, window = c(-4, 4))
lamp_trend_sensitivity(fit, c(-0.01, 0, 0.01))
```

---

lamp_twfe *Two-way fixed effects estimate of the effect of stops on crime*

Description

Fits the outcome on the treatment with area and month fixed effects, so that the effect is identified from changes within an area over time, net of anything common to all areas in a month. Standard errors are clustered at the level given in cluster.

Usage

```
lamp_twfe(
  panel,
  outcome = "crime_total",
  treatment,
  controls = NULL,
  cluster = c("area", "force"),
  family = c("poisson", "negbin", "ols_log", "ols_ihs"),
  force_month = FALSE
)
```

Arguments

panel	A lamp_panel from lamp_panel().
outcome	The outcome column, for example "crime_total" or a crime type key from lamp_crime_types().
treatment	A lamp_treatment() object, or the name of a panel column.
controls	Optional panel columns to include as covariates.
cluster	"area" (default) or "force".
family	"poisson" (default, a Poisson pseudo-likelihood suited to counts), "negbin", "ols_log" or "ols_ihs".
force_month	Add force-by-month fixed effects, which absorb anything that moved a whole force in a month (a recording change, a force-wide operation) at the cost of identifying only from within-force variation.

Value

A lamp_estimate of class lamp_twfe: coefficients with clustered standard errors and 95 percent intervals, the identifying assumption, the exclusion table and a diagnostics list holding the dispersion statistic and, when the panel carries adjacency, Moran's I of the residuals.

Identifying assumption

Treated and control areas would have followed parallel paths in the outcome had the treatment not changed, once area and month fixed effects are removed. With a continuous treatment this also requires that the intensity of searching is unrelated to what else was changing in the area, which police allocation makes doubtful: see lamp_allocation().

When this estimator is the wrong one

With staggered adoption, two-way fixed effects compares later-treated areas against already-treated ones, and the estimate can lie outside the range of every area's true effect. Use lamp_did_staggered() instead; this function warns when the treatment has more than one adoption date.

See Also

Other estimators: lamp_allocation(), lamp_crimes_prevented(), lamp_did_staggered(), lamp_displacement_quot
lamp_elasticity(), lamp_event_study(), lamp_spillover(), lamp_synth(), tidy.lamp_estimate()

Examples

```
sim <- lamp_simulate(n_areas = 24, n_months = 24, design = "event", effect = -0.2, seed = 2)
truth <- attr(sim, "truth")
tr <- lamp_treatment(sim, "event", date = truth$event_date, scope = truth$treated_areas)
fit <- lamp_twfe(sim, "crime_total", tr)
fit
tidy(fit)
```

---

lamp_version_diff *Compare the archive versions of one force-month file*

Description

Reads every available version of a file and reports what differs: the number of rows, and for street and outcomes files the crime identifiers present in one version but not another. Street files routinely differ between snapshots only in their `Last outcome category` column, which is why row and identifier counts, not checksums, decide whether a force re-supplied a month.

Usage

```
lamp_version_diff(
  snapshot,
  force,
  month,
  file_type = c("street", "outcomes", "stop-and-search"),
  fetch = FALSE
)
```

Arguments

`snapshot`	A `lamp_snapshot` from `lamp_archive_snapshot()`.
`force`	A police.uk force identifier.
`month`	A month as `"YYYY-MM"` or a Date.
`file_type`	One of `"street"`, `"outcomes"`, `"stop-and-search"`.
`fetch`	Fetch versions that are listed but not held locally (needs network access).

Value

A tibble with one row per version: `archive`, `n_rows`, `n_ids` (distinct crime identifiers; `NA` for stop-and-search files), `ids_not_in_others` (identifiers absent from every other version) and `crc32`. The attribute `outcome_changes` gives, for street files, the number of shared crimes whose last outcome category differs between the oldest and newest versions.

See Also

Other ingest: `lamp_archive_download()`, `lamp_archive_index()`, `lamp_archive_register()`, `lamp_archive_snapshot()`, `lamp_list_versions()`, `lamp_lsoa_vintage()`, `lamp_read_crime()`, `lamp_read_outcomes()`, `lamp_read_stop_counts()`, `lamp_s60_legislation()`, `lamp_select_version()`

Examples

```
cache <- tempfile("streetlamp-cache-")
zips <- list.files(
  system.file("extdata", "archive", package = "streetlamp"),
  pattern = "zip$", full.names = TRUE
)
for (z in zips) lamp_archive_register(z, dir = cache)
snap <- lamp_archive_snapshot(cache)
lamp_version_diff(snap, "west-yorkshire", "2026-05", "outcomes")
```

---

tidy.lamp_estimate *Tidy a streetlamp estimate*

Description

Tidy a streetlamp estimate

Usage

```
## S3 method for class 'lamp_estimate'
tidy(x, ...)
```

Arguments

x	A lamp_estimate.
...	Unused.

Value

A tibble with columns term, estimate, std_error, statistic, p_value, conf_low and conf_high.

See Also

Other estimators: lamp_allocation(), lamp_crimes_prevented(), lamp_did_staggered(), lamp_displacement_quot
lamp_elasticity(), lamp_event_study(), lamp_spillover(), lamp_synth(), lamp_twfe()

Index

∗ **bundled data**
  lamp_area_lookup, 11
  lamp_bank_holidays, 11
  lamp_crime_types, 19
  lamp_deprivation, 20
  lamp_forces, 33
  lamp_lsoa_lookup, 35
  lamp_outcome_types, 38
  lamp_sample_panel, 52
  lamp_shocks, 54
∗ **cache**
  lamp_cache_dir, 13
∗ **contract**
  lamp_contract, 16
∗ **diagnostics**
  lamp_design_audit, 21
  lamp_effect_summary, 27
  lamp_elasticity_robustness, 30
  lamp_placebo, 41
  lamp_pretrends, 44
  lamp_trend_sensitivity, 64
∗ **estimators**
  lamp_allocation, 5
  lamp_crimes_prevented, 18
  lamp_did_staggered, 23
  lamp_displacement_quotient, 25
  lamp_elasticity, 28
  lamp_event_study, 31
  lamp_spillover, 58
  lamp_synth, 59
  lamp_twfe, 65
  tidy.lamp_estimate, 68
∗ **ingest**
  lamp_archive_download, 6
  lamp_archive_index, 8
  lamp_archive_register, 9
  lamp_archive_snapshot, 10
  lamp_list_versions, 34
  lamp_lsoa_vintage, 36
  lamp_read_crime, 45
  lamp_read_outcomes, 46
  lamp_read_stop_counts, 48
  lamp_s60_legislation, 51
  lamp_select_version, 53
  lamp_version_diff, 67
∗ **panel**
  lamp_adjacency, 3
  lamp_adjacency_from_nb, 4
  lamp_boundaries, 12
  lamp_changelog, 14
  lamp_coverage, 17
  lamp_panel, 39
  lamp_population, 43
  lamp_revintage, 50
∗ **presentation**
  lamp_colours, 15
  lamp_map, 37
  lamp_table, 61
  lamp_theme, 62
∗ **reporting**
  lamp_report, 49
∗ **simulation**
  lamp_simulate, 54
  lamp_simulation_truth, 57
∗ **treatment**
  lamp_detect_surges, 22
  lamp_treatment, 63

ggplot2::theme_minimal(), *62*

lamp_adjacency, 3, *5*, *13*, *15*, *18*, *41*, *43*, *51*
lamp_adjacency(), *40*, *58*
lamp_adjacency_from_nb, *4*, 4, *13*, *15*, *18*, *41*, *43*, *51*
lamp_allocation, 5, *19*, *25*, *27*, *29*, *32*, *59*, *60*, *66*, *68*
lamp_archive_download, 6, *8–10*, *34*, *36*, *46*, *47*, *49*, *51*, *53*, *67*
lamp_archive_download(), *10*, *45*, *47*, *48*

lamp_archive_index, *7*, 8, *9, 10*, *34*, *36*, *46, 47*, *49*, *51*, *53*, *67*
lamp_archive_index(), *7*
lamp_archive_register, *7, 8*, 9, *10*, *34*, *36*, *46, 47*, *49*, *51*, *53*, *67*
lamp_archive_register(), *10*
lamp_archive_snapshot, *7–9*, 10, *34*, *36*, *46, 47*, *49*, *51*, *53*, *67*
lamp_archive_snapshot(), *7*, *9*, *34*, *45*, *47, 48*, *67*
lamp_area_lookup, 11, *12*, *20, 21*, *33*, *35*, *39*, *52*, *54*
lamp_area_lookup(), *41*, *43*
lamp_bank_holidays, *11*, 11, *20, 21*, *33*, *35*, *39*, *52*, *54*
lamp_boundaries, *4, 5*, 12, *15*, *18*, *41*, *43*, *51*
lamp_boundaries(), *37*
lamp_cache_clear *(*lamp_cache_dir*)*, 13
lamp_cache_dir, 13
lamp_cache_dir(), *7–10*, *12*, *14*, *43*, *45*, *47, 48*
lamp_changelog, *4, 5*, *13*, 14, *18*, *41*, *43*, *51*
lamp_changelog(), *17*, *40*
lamp_colours, 15, *38*, *62*
lamp_colours(), *38*
lamp_contract, 16
lamp_contract(), *41*, *46*
lamp_coverage, *4, 5*, *13*, *15*, 17, *41*, *43*, *51*
lamp_coverage(), *14*, *41*, *61*
lamp_coverage_compare *(*lamp_coverage*)*, 17
lamp_crime_types, *11, 12*, 19, *21*, *33*, *35*, *39*, *52*, *54*
lamp_crime_types(), *24*, *28*, *31*, *39*, *41*, *58*, *66*
lamp_crimes_prevented, *6*, 18, *25*, *27*, *29*, *32*, *59, 60*, *66*, *68*
lamp_deprivation, *11, 12*, *20*, 20, *33*, *35*, *39*, *52*, *54*
lamp_design_audit, 21, *28*, *31*, *42*, *45*, *65*
lamp_detect_surges, 22, *64*
lamp_did_staggered, *6*, *19*, 23, *27*, *29*, *32*, *59, 60*, *66*, *68*
lamp_did_staggered(), *18*, *32*
lamp_displacement_quotient, *6*, *19*, *25*, 25, *29*, *32*, *59, 60*, *66*, *68*
lamp_effect_summary, *22*, 27, *31*, *42*, *45*, *65*
lamp_elasticity, *6*, *19*, *25*, *27*, 28, *32*, *59, 60*, *66*, *68*
lamp_elasticity(), *18*
lamp_elasticity_robustness, *22*, *28*, 30, *42*, *45*, *65*
lamp_event_study, *6*, *19*, *25*, *27*, *29*, 31, *59, 60*, *66*, *68*
lamp_event_study(), *18*, *44*, *56*, *65*
lamp_forces, *11, 12*, *20, 21*, 33, *35*, *39*, *52*, *54*
lamp_forces(), *7*
lamp_list_versions, *7–10*, 34, *36*, *46, 47*, *49*, *51*, *53*, *67*
lamp_list_versions(), *10*, *53*
lamp_lsoa_codes *(*lamp_lsoa_lookup*)*, 35
lamp_lsoa_codes(), *36*
lamp_lsoa_lookup, *11, 12*, *20, 21*, *33*, 35, *39*, *52*, *54*
lamp_lsoa_lookup(), *50*
lamp_lsoa_vintage, *7–10*, *34*, 36, *46, 47*, *49*, *51*, *53*, *67*
lamp_map, *15*, 37, *62*
lamp_outcome_types, *11, 12*, *20, 21*, *33*, *35*, 38, *52*, *54*
lamp_outcome_types(), *20*, *46, 47*
lamp_panel, *4, 5*, *13*, *15*, *18*, 39, *43*, *51*
lamp_panel(), *11*, *14*, *16, 17*, *24*, *28*, *31*, *43*, *52*, *58*, *63*, *66*
lamp_placebo, *22*, *28*, *31*, 41, *45*, *65*
lamp_placebo(), *56*
lamp_population, *4, 5*, *13*, *15*, *18*, *41*, 43, *51*
lamp_pretrends, *22*, *28*, *31*, *42*, 44, *65*
lamp_pretrends(), *32*, *61*
lamp_read_crime, *7–10*, *34*, *36*, 45, *47*, *49*, *51*, *53*, *67*
lamp_read_crime(), *17*, *20*, *36*, *40*
lamp_read_outcomes, *7–10*, *34*, *36*, *46*, 46, *49*, *51*, *53*, *67*
lamp_read_outcomes(), *17*, *36*, *39, 40*
lamp_read_stop_counts, *7–10*, *34*, *36*, *46, 47*, 48, *51*, *53*, *67*
lamp_read_stop_counts(), *17*, *40*
lamp_report, 49
lamp_revintage, *4, 5*, *13*, *15*, *18*, *41*, *43*, 50
lamp_s60_legislation, *7–10*, *34*, *36*, *46, 47*, *49*, 51, *53*, *67*
lamp_sample_boundaries *(*lamp_sample_panel*)*, 52
lamp_sample_boundaries(), *37*
lamp_sample_panel, *11, 12*, *20, 21*, *33*, *35*,

39, 52, *54*
lamp_select_version, *7–10*, *34*, *36*, *46, 47*, *49*, *51*, 53, *67*
lamp_select_version(), *45*, *47, 48*
lamp_shocks, *11, 12*, *20, 21*, *33*, *35*, *39*, *52*, 54
lamp_shocks(), *63, 64*
lamp_simulate, 54, *57*
lamp_simulate(), *4*, *57*
lamp_simulation_truth, *56*, 57
lamp_spillover, *6*, *19*, *25*, *27*, *29*, *32*, 58, *60*, *66*, *68*
lamp_synth, *6*, *19*, *25*, *27*, *29*, *32*, *59*, 59, *66*, *68*
lamp_synth(), *61*
lamp_table, *15*, *38*, 61, *62*
lamp_theme, *15*, *38*, *62*, 62
lamp_treatment, *23*, 63
lamp_treatment(), *22–24*, *31*, *58*, *66*
lamp_trend_sensitivity, *22*, *28*, *31*, *42*, *45*, 64
lamp_twfe, *6*, *19*, *25*, *27*, *29*, *32*, *59, 60*, 65, *68*
lamp_twfe(), *18*
lamp_version_diff, *7–10*, *34*, *36*, *46, 47*, *49*, *51*, *53*, 67

tidy(), *61*
tidy.lamp_estimate, *6*, *19*, *25*, *27*, *29*, *32*, *59, 60*, *66*, 68