Package: tbepreport (via r-universe)

October 5, 2026

Title Compile and Visualize Ecosystem Report Card Indicators for Tampa Bay

Version 1.0.0

Description Functions to calculate, combine, and visualize water quality, sediment, fish/wildlife, and habitat indicators as bay segment ``report card'' scores for the Tampa Bay Estuary Program.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.2), tbeptools

Imports dplyr, ggplot2, ggrepel, here, jsonlite, plotly, purrr, rlang, scales, sf, stats, tibble, tidyr, tools, utils

LazyData true

LazyDataCompression xz

URL <https://github.com/tbep-tech/tbepreport>

BugReports https://github.com/tbep-tech/tbepreport/issues

Suggests knitr, mockery, patchwork, quarto, testthat (>= 3.0.0)

SystemRequirements GDAL (>= 2.0.0)

Remotes tbep-tech/tbeptools

Config/testthat/edition 3

VignetteBuilder quarto

Config/pak/sysreqs libabsl-dev libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev libfribidi-dev libgdal-dev gdal-bin libgeos-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libuv1-dev libwebp-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

Repository https://fawda123.r-universe.dev

Date/Publication 2026-08-21 12:17:40 UTC

RemoteUrl https://github.com/tbep-tech/tbepreport

RemoteRef HEAD

RemoteSha c09fdfe065672fbddd51c544bd3f13714908384e

Contents

anlz_category . . . 2
anlz_fw_nonnative_abundance . . . 3
anlz_fw_nonnative_obs . . . 4
anlz_fw_nonnative_richness . . . 5
anlz_fw_tbni . . . 6
anlz_hab_seagrass_coverage . . . 7
anlz_hab_seagrass_transect . . . 8
anlz_indicators . . . 9
anlz_score . . . 10
anlz_sed_peltel . . . 11
anlz_sed_tbbi . . . 12
anlz_wq_attain . . . 13
anlz_wq_fib . . . 13
anlz_wq_load . . . 14
anlz_wq_thresh . . . 15
anlz_wq_tidalcreeks . . . 16
aoi_shp . . . 17
bayseg_shp . . . 18
plot_category . . . 19
plot_score . . . 20
plot_trend . . . 21
sgsegest . . . 23
util_fw_fetch_fim . . . 24
util_fw_fetch_nas . . . 24
util_fw_nonnative_areas . . . 25
util_fw_nonnative_long . . . 26
util_outcome . . . 27
util_percentile . . . 29
util_rdataload . . . 30

Index **31**

`anlz_category` *Combine indicators into a single score by bay segment and year*

Description

Combine indicators into a single score by bay segment and year

Usage

```
anlz_category(
  ...,
  bay_segments = c("OTB", "HB", "MTB", "LTB"),
  yr_min = 2000,
  wt = NULL
)
```

Arguments

`...`	named data.frames, passed to `anlz_indicators`
`bay_segments`	chr vector of bay segments to include, defaults to `c('OTB', 'HB', 'MTB', 'LTB')`
`yr_min`	integer, minimum year to include, defaults to `2000`
`wt`	named numeric vector of weights, keyed by the `indicator` values produced by `anlz_indicators` (e.g. `c(chla_thresh = 2, fib = 1)` to weight an indicator, or `c(wq = 2, sed = 1)` when combining category scores via `anlz_score`). Indicators not named in `wt` (including all of them when `wt = NULL`) get a weight of `1`, so the default is a plain unweighted mean.

Details

Stacks `...` with `anlz_indicators`, then averages `outcome` across whatever indicators have data for a given bay segment/year. See `n_indicator` in the output for how many contributed. This same averaging step produces a category score when `...` are raw indicator outputs, and the final bay segment score when `...` are the four category scores (see `anlz_score`).

Value

A data.frame with columns `bay_segment`, `yr`, `outcome` (0-1, 1 = best), `n_indicator` (number of indicators averaged for that segment/year), and one additional column per indicator (named from `...` or `anlz_indicators`'s `indicator` column) holding that indicator's own outcome, `NA` where a segment/year has no data for it

Examples

```
anlz_category(
  wq_attain = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.8),
  fib = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.4)
)
```

---

anlz_fw_nonnative_abundance

Non-native species abundance outcome by bay segment and year

Description

Non-native species abundance outcome by bay segment and year

Usage

```
anlz_fw_nonnative_abundance(obs)
```

Arguments

`obs`	output of `anlz_fw_nonnative_obs`

Details

Counts occurrences per bay segment/year, normalizes by segment area (plus an AOI-wide `"All"` total normalized by total AOI area, from `util_fw_nonnative_areas`), converts to a percentile with `util_percentile`, then reshapes to long format and reverses to a 0-1 outcome with `util_fw_nonnative_long` (higher abundance is worse).

Value

A data.frame with columns `yr`, `bay_segment` (abbreviated), `value` (0-100 percentile of area-normalized abundance), and `outcome` (0-1, 1 = best)

Examples

```
## Not run:
obs <- anlz_fw_nonnative_obs()
anlz_fw_nonnative_abundance(obs)

## End(Not run)
```

---

anlz_fw_nonnative_obs *Non-native species occurrence points, clipped and assigned to a bay segment*

Description

Non-native species occurrence points, clipped and assigned to a bay segment

Usage

```
anlz_fw_nonnative_obs(
 huc8_list = c("03100101", "03100201", "03100202", "03100203", "03100204", "03100205",
    "03100206", "03100207", "03100208"),
  fim_url =
   "https://github.com/kflahertywalia/tb_fim_nonnatives/raw/refs/heads/main/Output/tb_fim_inv.RData"
  min_yr = 2000
)
```

Arguments

`huc8_list`	chr vector of HUC8 codes, passed to `util_fw_fetch_nas`
`fim_url`	chr string, FIM data URL, passed to `util_fw_fetch_fim`
`min_yr`	integer, earliest year to include, defaults to `2000`

Details

Combines util_fw_fetch_nas and util_fw_fetch_fim, filters to yr >= min_yr and non-missing coordinates, then clips to the aoi_shp boundary and spatially joins each point to a bay_segment using bayseg_shp's BAY_SEG_GP grouping. This is the shared, network-heavy step behind anlz_fw_nonnative_abundance and anlz_fw_nonnative_richness, computed once and passed to both rather than letting each refetch.

Value

A data.frame with columns scientificName, commonName, group, year, lon, lat, source, and bay_segment (full segment name, e.g. "Old Tampa Bay"). Each row is one occurrence point that fell within the AOI and a bay segment

Examples

```
## Not run:
anlz_fw_nonnative_obs()

## End(Not run)
```

---

anlz_fw_nonnative_richness

Non-native species richness outcome by bay segment and year

Description

Non-native species richness outcome by bay segment and year

Usage

```
anlz_fw_nonnative_richness(obs)
```

Arguments

obs	output of anlz_fw_nonnative_obs

Details

Counts unique scientificName values per bay segment/year, normalizes by segment area (plus an AOI-wide "All" total normalized by total AOI area, from util_fw_nonnative_areas), converts to a percentile with util_percentile, then reshapes to long format and reverses to a 0-1 outcome with util_fw_nonnative_long (higher richness is worse).

Value

A data.frame with columns yr, bay_segment (abbreviated), value (0-100 percentile of area-normalized richness), and outcome (0-1, 1 = best)

Examples

```
## Not run:
obs <- anlz_fw_nonnative_obs()
anlz_fw_nonnative_richness(obs)

## End(Not run)
```

`anlz_fw_tbni` *Tampa Bay Nekton Index (TBNI) outcome by bay segment and year*

Description

Tampa Bay Nekton Index (TBNI) outcome by bay segment and year

Usage

```
anlz_fw_tbni(fimdata, from = c(32, 46))
```

Arguments

`fimdata`	data.frame of raw fisheries independent monitoring data, e.g. `tbeptools::fimdata`
`from`	numeric vector of length 2, the `c(min, max)` range of `Segment_TBNI` passed to `util_outcome` as its `from` argument. Defaults to `c(32, 46)`, TBNI's own grade breakpoints. Pass `c(0, 100)` to revert to a plain linear rescale over the full score range with no clamping.

Details

Uses `anlz_tbniscr` and `anlz_tbniave` to score each bay segment/year 0-100, then converts to a 0-1 outcome with `util_outcome` (`type = "continuous"`). With the default `from = c(32, 46)`, matching TBNI's own grade breakpoints (On Alert below 32, Caution from 32 to 46, Stay the Course above 46), scores below 32 give an outcome of 0, scores above 46 give an outcome of 1, and scores in between are linearly rescaled.

Value

A data.frame with columns `bay_segment`, `yr`, `Segment_TBNI` (the raw 0-100 score), `Action` (management action category from `anlz_tbniave`), and `outcome` (0-1, 1 = best)

Examples

```
anlz_fw_tbni(tbeptools::fimdata)
```

anlz_hab_seagrass_coverage

Seagrass coverage outcome by bay segment and year

Description

Seagrass coverage outcome by bay segment and year

Usage

```
anlz_hab_seagrass_coverage(
  sgsegest,
  yr_max = max(sgsegest$year),
  smooth = "ramp",
  pct = 0.1
)
```

Arguments

sgsegest	data.frame of seagrass coverage estimates by bay segment and year, e.g. sgsegest
yr_max	integer, the last year to carry estimates forward to, defaults to the last year sgsegest actually has an estimate for (max(sgsegest$year))
smooth	passed to util_outcome. One of "ramp" (the default), "logistic", or "none". "ramp" gives an outcome of 1 for any acreage at or above a segment's target, decaying toward 0 the further short of it a segment falls.
pct	numeric, passed to util_outcome as the fraction of each acreage target used for the transition's steepness when smooth is "ramp" or "logistic". Defaults to 0.1 (10% of the target).

Details

Seagrass coverage maps are not produced every year. They're flown approximately biennially (sgsegest has estimates for most even years). To get a value for every calendar year, this fills the gaps by carrying the most recent actual estimate (and the outcome derived from it) **forward** via complete + fill. For example, a year with no survey repeats the prior survey year's acreage and outcome unchanged, until the next actual survey year updates it. This means acres/outcome in a non-survey year are not a new estimate, just a repeat of the last known one.

Coverage is compared against a fixed per-segment acreage target with util_outcome (type = "threshold", op = ">="). By default (smooth = "ramp"), meeting or exceeding the target gives an outcome of 1, decaying toward 0 the further short of it a segment falls. Targets (acres):

Old Tampa Bay 11,100

Hillsborough Bay 1,751

Middle Tampa Bay 9,400

Lower Tampa Bay 7,400

Boca Ciega Bay 8,800

Terra Ceia Bay 1,100

Manatee River 449

These targets are derived from the baywide 40,000-acre seagrass coverage target apportioned to each segment's share of total bay area.

Value

A data.frame with columns `bay_segment` (abbreviated, e.g. `"OTB"`), `yr`, `acres` (carried forward in non-survey years), and `outcome` (0-1, 1 = best, also carried forward in non-survey years)

Examples

```
anlz_hab_seagrass_coverage(sgsegest)
```

`anlz_hab_seagrass_transect` *Seagrass transect frequency of occurrence outcome by bay segment and year*

Description

Seagrass transect frequency of occurrence outcome by bay segment and year

Usage

```
anlz_hab_seagrass_transect(transect)
```

Arguments

`transect`	data.frame of raw seagrass transect data, e.g. `tbeptools::transect`

Details

Uses `anlz_transectocc` and `anlz_transectave` to estimate frequency of seagrass occurrence (`foest`, 0-100) at each bay segment/year, drops the bay-wide `"All"` aggregate row, then converts to a 0-1 outcome with `util_outcome` (`type = "continuous"`, `from = c(0, 100)`).

Value

A data.frame with columns `bay_segment`, `yr`, `foest` (frequency of occurrence estimate, 0-100), and `outcome` (0-1, 1 = best)

Examples

```
anlz_hab_seagrass_transect(tbeptools::transect)
```

anlz_indicators *Stack indicator outcomes into one long-format table*

Description

Stack indicator outcomes into one long-format table

Usage

```
anlz_indicators(..., bay_segments = c("OTB", "HB", "MTB", "LTB"))
```

Arguments

`...`	named data.frames, each with `bay_segment`, `yr`, and `outcome` columns. A data.frame may already have its own `indicator` column if it bundles more than one indicator (e.g. `anlz_wq_load`'s output already has `indicator` = `"tn_load"`/`"tnhy_load"`). Otherwise every row is labeled with that argument's name.
`bay_segments`	chr vector of bay segments to keep, defaults to `c('OTB', 'HB', 'MTB', 'LTB')`

Details

Stacks (`bind_rows`) rather than joins the inputs, since bay segment/year coverage differs by indicator. A segment/ year missing from one indicator's source data simply has no row for it, rather than an `NA`-filled one. Rows outside `bay_segments` or with an `NA` outcome are dropped.

This same stacking step is used at every level of the indicator hierarchy, once per category to build a category's indicator table from its raw `anlz_*` indicator outputs, and again inside `anlz_score` (via `anlz_category`) to stack the four category scores into one table before the final average. A category score is just another single-indicator input.

Value

A data.frame with columns `bay_segment`, `yr`, `indicator`, and `outcome` (0-1, 1 = best)

Examples

```
anlz_indicators(
  wq_attain = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.8),
  load = data.frame(bay_segment = 'OTB', yr = 2020,
    indicator = c('tn_load', 'tnhy_load'), outcome = c(1, 0))
)
```

anlz_score *Overall bay segment score across all four indicator categories*

Description

Overall bay segment score across all four indicator categories

Usage

```
anlz_score(wqoverall, sedoverall, fwoverall, haboverall, wt = NULL)
```

Arguments

wqoverall	output of anlz_category for water quality
sedoverall	output of anlz_category for sediment
fwoverall	output of anlz_category for fish/wildlife
haboverall	output of anlz_category for habitat
wt	named numeric vector of category weights, e.g. c(wq = 2, sed = 1, fw = 1, hab = 1). Categories not named in wt (including all of them when wt = NULL) get a weight of 1, so the default is a plain unweighted mean across the four categories, passed to anlz_category.

Details

A thin, semantically-named wrapper around anlz_category. A category score is just another single-indicator input, so combining the four category scores into one bay segment score uses exactly the same stacking (anlz_indicators) and weighted-averaging logic that combines individual indicators into a category score.

Value

A data.frame with columns bay_segment, yr, outcome (0-1, 1 = best, the overall Bay Segment Score), n_indicator (number of categories with a score for that segment/year), and one additional column per category (wq, sed, fw, hab) holding that category's own score, NA where a segment/year has no score for it

Examples

```
wqoverall <- data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.8)
sedoverall <- data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.6)
fwoverall <- data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.7)
haboverall <- data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.5)
anlz_score(wqoverall, sedoverall, fwoverall, haboverall)
```

anlz_sed_peltel — *Sediment quality (PEL/TEL) outcome by bay segment and year*

Description

Sediment quality (PEL/TEL) outcome by bay segment and year

Usage

```
anlz_sed_peltel(sedimentdata, yrs)
```

Arguments

sedimentdata	data.frame of raw sediment monitoring data, e.g. tbeptools::sedimentdata
yrs	integer vector of years to assess, passed to anlz_sedimentpelave one year at a time

Details

Grades each bay segment/year by its average sediment contamination score (ave, from anlz_sedimentpelave) into A-F, kept here as grd for reference, but the outcome itself comes directly from the continuous ave score via util_outcome (type = "continuous", reverse = TRUE) rather than from grd. Because the grade breakpoints are geometrically spaced (each roughly 2.7-4x the last: 0.00756, 0.02052, 0.08567, 0.28026), ave is log-transformed first so the outcome varies smoothly across grades B-D instead of being compressed. from spans the A/B breakpoint to the D/F breakpoint (on the log scale), so that ave at or below the A/B breakpoint gives an outcome of 1 and at or above the D/F breakpoint gives an outcome of 0.

Value

A data.frame with columns yr, bay_segment, ave (average sediment contamination score), grd (letter grade A-F, for reference only, not used to compute outcome), and outcome (0-1, 1 = best)

Examples

```
anlz_sed_peltel(tbeptools::sedimentdata, yrs = 2020)
```

anlz_sed_tbbi *Benthic index (TBBI) outcome by bay segment and year*

Description

Benthic index (TBBI) outcome by bay segment and year

Usage

```
anlz_sed_tbbi(benthicdata)
```

Arguments

benthicdata	raw benthic monitoring data, e.g. `tbeptools::benthicdata`

Details

Uses `anlz_tbbiscr` to get station-level TBBI scores (0-100), filtered to the same stations `anlz_tbbimed` uses for its bay segment grades (`FundingProject == "TBEP"`, `ProgramID == 4`, and `TBBI` between 0 and 100, for the 7 bay segments `anlz_tbbimed` scores), then takes the median station score for each bay segment/year. `outcome` comes directly from that continuous median via `util_outcome` (`type = "continuous"`, `from = c(73, 87)`), TBBI's own grade breakpoints (Degraded below 73, Intermediate 73-87, Healthy above 87). This is the same clamped-breakpoint-window treatment used for the Nekton Index's `from = c(32, 46)`.

This bypasses `anlz_tbbimed` entirely, which instead grades a bay segment/year Poor/Fair/Good from the *proportion* of its stations falling in each of the Degraded/Intermediate/Healthy categories. This is a compound rule on those proportions, not a discretized version of a single continuous statistic, so there's no direct continuous equivalent of it the way there is for TBNI's or PEL/TEL's breakpoints. Scoring the median of the raw station values instead is a related but distinct measure. It can disagree with `anlz_tbbimed`'s category in some edge cases (e.g. a bay segment split between many Healthy and a few very Degraded stations can have a middling median while still tripping the proportion rule's `Degraded >= 0.2` condition for `"Poor"`).

Value

A data.frame with columns `yr`, `bay_segment`, `TBBI` (median station-level score for that bay segment/year), and `outcome` (0-1, 1 = best)

Examples

```
anlz_sed_tbbi(tbeptools::benthicdata)
```

anlz_wq_attain — *Water quality attainment outcome by bay segment and year*

Description

Water quality attainment outcome by bay segment and year

Usage

```
anlz_wq_attain(epcdata)
```

Arguments

epcdata	data.frame of raw water quality data, e.g. `tbeptools::epcdata`

Details

Combines chlorophyll and light attenuation attainment of management targets (from `anlz_attain`) into a single continuous outcome. The two sub-scores are summed (each 0-3 assessing magnitude and duration of exceedance), then converted with `util_outcome` (`type = "continuous"`, `reverse = TRUE`) so a lower combined sub-score (better attainment) gives a higher outcome.

Value

A data.frame with columns `bay_segment`, `yr`, `totsum` (the summed chlorophyll/light attenuation sub-score), and `outcome` (0-1, 1 = best)

Examples

```
anlz_wq_attain(tbeptools::epcdata)
```

anlz_wq_fib — *Fecal indicator bacteria (FIB) outcome by bay segment and year*

Description

Fecal indicator bacteria (FIB) outcome by bay segment and year

Usage

```
anlz_wq_fib(enterodata)
```

Arguments

enterodata	data.frame of raw enterococcus monitoring data, e.g. `tbeptools::enterodata`

Details

Uses anlz_fibmatrix to get, for each bay segment/year, exceed_rate, a continuous, one-sided 90\ confidence estimate of the true exceedance rate (0-1, lower is better), and converts it to a 0-1 outcome with util_outcome (type = "continuous", from = c(0, 1), reverse = TRUE), i.e. outcome = 1 - exceed_rate. This is a continuous analog of the A-E letter grade anlz_fibmatrix() also returns (cat), which is itself a discretized version of exceed_rate. See anlz_fibmatrix for details.

Value

A data.frame with columns bay_segment, yr, and outcome (0-1, 1 = best)

Examples

```
anlz_wq_fib(tbeptools::enterodata)
```

---

anlz_wq_load *Nutrient loading outcome by bay segment, year, and indicator*

Description

Nutrient loading outcome by bay segment, year, and indicator

Usage

```
anlz_wq_load(totanndat, smooth = "logistic", pct = 0.1)
```

Arguments

totanndat	data.frame of total annual loading estimates, with columns year, bay_segment (full segment names, e.g. "Old Tampa Bay"), tn_load, and tnhy, e.g. as returned by util_rdataload() on the load-estimates data
smooth	passed to util_outcome. One of "logistic" (the default), "ramp", or "none". By default, use a smooth logistic transition centered at each threshold instead of a hard 0/1 cutoff.
pct	numeric, passed to util_outcome as the fraction of each threshold used for the transition's steepness when smooth is "logistic" or "ramp". Defaults to 0.1 (10% of the threshold).

Details

Compares two loading measures against fixed bay-segment thresholds, using util_outcome with type = "threshold". By default (smooth = "logistic"), a value below its threshold approaches an outcome of 1 and at or above approaches 0, with a smooth logistic transition between them.

- tn_load: total nitrogen load

- tnhy_load: tnhy, the total nitrogen load normalized by hydrologic load (i.e. TN load per unit of hydrologic load, a hydrologically-normalized loading rate). This is not the raw hydrologic (freshwater inflow) load itself. The raw hy_load column in totanndat is not used here.

Returned in long format with one row per indicator, ready to stack into anlz_indicators.

Value

A data.frame with columns bay_segment (abbreviated, e.g. "OTB"), yr, indicator ("tn_load" or "tnhy_load", see Details for what "tnhy_load" measures), and outcome (0-1, 1 = best)

Examples

```
## Not run:
totanndat <- util_rdataload(
  "https://github.com/tbep-tech/load-estimates/raw/main/data/totanndat.RData"
)
anlz_wq_load(totanndat)

## End(Not run)
```

---

anlz_wq_thresh — *Water quality threshold attainment outcome by bay segment, year, and variable*

Description

Water quality threshold attainment outcome by bay segment, year, and variable

Usage

```
anlz_wq_thresh(epcdata, smooth = "logistic", pct = 0.1)
```

Arguments

epcdata	data.frame of raw water quality data, e.g. tbeptools::epcdata
smooth	passed to util_outcome. One of "logistic" (the default), "ramp", or "none". By default, use a smooth logistic transition centered at each bay-segment's threshold instead of a hard 0/1 cutoff.
pct	numeric, passed to util_outcome as the fraction of each threshold used for the transition's steepness when smooth is "logistic" or "ramp". Defaults to 0.1 (10% of the threshold).

Details

Compares annual mean chlorophyll and light attenuation values against the bay-segment-specific thresholds in targets, using util_outcome with type = "threshold". By default (smooth = "logistic"), a value below its threshold approaches an outcome of 1 and at or above approaches 0, with a smooth logistic transition between them. Only mean_chla and mean_la are scored (tbeptools::targets has no threshold for the third variable anlz_avedat returns, mean_sdm).

Value

A data.frame with columns `bay_segment`, `yr`, `indicator` (`"chla_thresh"` or `"la_thresh"`), and `outcome` (0-1, 1 = best)

Examples

```
anlz_wq_thresh(tbeptools::epcdata)
```

---

anlz_wq_tidalcreeks *Tidal creek condition outcome by bay segment and year*

Description

Tidal creek condition outcome by bay segment and year

Usage

```
anlz_wq_tidalcreeks(tidalcreeks, iwrraw, yrs)
```

Arguments

`tidalcreeks`	sf object of tidal creek assessment segments, e.g. `tbeptools::tidalcreeks`
`iwrraw`	data.frame of raw IWR water quality data for tidal creeks, e.g. `tbeptools::iwrraw`
`yrs`	integer vector of years to assess, passed to `anlz_tdlcrk`

Details

Each tidal creek is assigned to the bay segment subwatershed (`tbsegshed`) it overlaps most (by length, for creeks spanning more than one subwatershed).

`anlz_tdlcrk` computes each creek's condition category (Prioritize/Investigate/Caution/Monitor) from year counts out of a rolling 10-year window. The category itself comes from a compound rule (the worst grade present usually wins, with several count-based exceptions that downgrade it, e.g. a single bad year surrounded by mostly good years). Rather than use that category, this converts each creek's own 4 counts directly into a continuous 0-1 score. This is a count-weighted average across an ordinal scale for each category (`Monitor = 1`, `Caution = 2/3`, `Investigate = 1/3`, `Prioritize = 0`), i.e. what fraction of the 10-year window fell in each grade. Creeks with a "No Data" assessment are dropped.

Each creek's continuous score is then averaged to a bay-segment/year outcome, weighted by each creek's physical length so longer creek segments contribute proportionally more.

Value

A data.frame with columns `bay_segment`, `yr`, `outcome` (0-1, 1 = best), and `n_assessed` (number of creek records feeding that segment/year's outcome)

Examples

```
## Not run:
anlz_wq_tidalcreeks(tbeptools::tidalcreeks, tbeptools::iwrraw, yrs = 2015:2020)

## End(Not run)
```

---

aoi_shp *Area of interest boundary for non-native species occurrence data*

Description

Area of interest boundary for non-native species occurrence data

Usage

```
aoi_shp
```

Format

An `sf` object (EPSG:4326, WGS 84) with 1 row and 3 columns:

BAY_SEG chr, boundary name

area_ac numeric, area in acres

geometry polygon geometry

Details

The Tampa Bay area of interest polygon used to clip non-native species occurrence points to the area covered by the report card (see `anlz_fw_nonnative_obs()`). Copied from the tbep-invasives repo (`input_data/shp/TBEP_AOI_4326.shp`).

See `data-raw/aoi_shp.R` for construction.

Examples

```
aoi_shp
```

bayseg_shp *Bay segment boundaries for non-native species occurrence data*

Description

Bay segment boundaries for non-native species occurrence data

Usage

```
bayseg_shp
```

Format

An `sf` object (EPSG:4326, WGS 84) with 7 rows and 5 columns:

BAY_SEG chr, raw bay segment name (7 unique values, e.g. "Boca Ciega Bay")

BAY_SEG_GP chr, grouped bay segment name (5 unique values). The 7 raw segments grouped into "Old Tampa Bay", "Hillsborough Bay", "Middle Tampa Bay", "Lower Tampa Bay", and "Remainder Lower Tampa Bay" (Boca Ciega Bay, Manatee River, and Terra Ceia Bay dissolved together)

BAY_GP_ABB chr, abbreviation of `BAY_SEG_GP` (OTB, HB, MTB, LTB, RALTB)

area_ac numeric, area of the raw `BAY_SEG` polygon in acres

geometry polygon geometry

Details

Bay segment boundaries used to spatially join non-native species occurrence points to a bay segment (see `anlz_fw_nonnative_obs()`). Copied from the tbep-invasives repo (`input_data/shp/TBEP_Bay_Segments_4326.shp`) so segment grouping and areas match its report cards exactly. Note `area_ac` is a static attribute of each raw polygon. Functions using this data recompute area from the geometry instead (grouped by `BAY_SEG_GP`), which matches this static value closely but is used for consistency with the source pipeline's own area calculation.

See `data-raw/bayseg_shp.R` for construction.

Examples

```
bayseg_shp
```

`plot_category` *Sunburst plot of indicator outcomes for one category, bay segment, and year*

Description

Sunburst plot of indicator outcomes for one category, bay segment, and year

Usage

```
plot_category(
  ...,
  bay_segment,
  yr,
  bay_segments = c("OTB", "HB", "MTB", "LTB"),
  yr_min = 2000,
  wt = NULL
)
```

Arguments

`...`	named data.frames, passed to `anlz_category`. See its documentation (and `anlz_indicators`) for the expected shape
`bay_segment`	chr string, single bay segment to plot, must be one of `bay_segments`
`yr`	integer, single year to plot
`bay_segments`	chr vector of bay segments to include, passed to `anlz_category`, defaults to `c('OTB', 'HB', 'MTB', 'LTB')`
`yr_min`	integer, minimum year to include, passed to `anlz_category`, defaults to `2000`
`wt`	named numeric vector of indicator weights, passed to `anlz_category`

Details

Calls `anlz_category` on `...` to get both the per-indicator outcomes (one wide column per indicator) and the overall category score for `bay_segment/yr`, then plots a two-ring sunburst using a colored center for the category's overall `outcome` and an outer ring with one equal-size wedge per indicator colored by that indicator's own outcome. Both rings use the same continuous red/yellow/green outcome scale, and an indicator missing for `bay_segment/yr` still gets an equal-size wedge, rendered gray rather than dropped. Thematically identical to `plot_score`, which extends this same two-ring layout with a third ring combining several categories.

Value

A `plotly` htmlwidget

Examples

```
plot_category(
  wq_attain = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.8),
  fib = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.4),
  bay_segment = 'OTB', yr = 2020
)
```

---

plot_score — *Sunburst plot of indicator, category, and overall outcomes for one bay segment and year*

Description

Sunburst plot of indicator, category, and overall outcomes for one bay segment and year

Usage

```
plot_score(
  wqoverall,
  sedoverall,
  fwoverall,
  haboverall,
  bay_segment,
  yr,
  wt = NULL
)
```

Arguments

wqoverall	output of anlz_category for water quality
sedoverall	output of anlz_category for sediment
fwoverall	output of anlz_category for fish/wildlife
haboverall	output of anlz_category for habitat
bay_segment	chr string, single bay segment to plot
yr	integer, single year to plot
wt	named numeric vector of category weights, passed to anlz_score

Details

Calls anlz_score on the four category data.frames to get the overall bay segment score and each category's own score for bay_segment/yr, then plots a three-ring sunburst using a colored center for the overall score, a middle ring with one equal-size wedge per category (wq, sed, fw, hab), and an outer ring with the indicator columns of each of wqoverall, sedoverall, fwoverall, haboverall split evenly within their own category's wedge, regardless of how many indicators another category has. Every ring is colored on the same continuous red/yellow/green outcome scale used by plot_category, with missing indicators/categories rendered gray rather than dropped.

Value

A `plotly` htmlwidget

Examples

```
wqoverall <- anlz_category(
  wq_attain = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.8)
)
sedoverall <- anlz_category(
  sed_tbbi = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.6)
)
fwoverall <- anlz_category(
  tbni = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.7)
)
haboverall <- anlz_category(
  seagrass = data.frame(bay_segment = 'OTB', yr = 2020, outcome = 0.5)
)
plot_score(wqoverall, sedoverall, fwoverall, haboverall, bay_segment = 'OTB', yr = 2020)
```

---

plot_trend — *Faceted trend plot of indicator, category, and overall outcomes over time*

Description

Faceted trend plot of indicator, category, and overall outcomes over time

Usage

```
plot_trend(
  wqoverall,
  sedoverall,
  fwoverall,
  haboverall,
  bay_segment,
  yr_range = NULL,
  wt = NULL,
  facets = c("Overall", "wq", "sed", "fw", "hab"),
  text = c("labels", "legend", "none")
)
```

Arguments

`wqoverall`	output of `anlz_category` for water quality
`sedoverall`	output of `anlz_category` for sediment
`fwoverall`	output of `anlz_category` for fish/wildlife
`haboverall`	output of `anlz_category` for habitat

bay_segment	chr string, single bay segment to plot
yr_range	numeric vector of length 2, c(min, max) years to plot (inclusive). Defaults to NULL, plotting every year available in wqoverall, sedoverall, fwoverall, and haboverall.
wt	named numeric vector of category weights, passed to anlz_score
facets	chr vector, one or more of "Overall", "wq", "sed", "fw", "hab" indicating which facet(s) to plot. Defaults to all five.
text	chr string, one of "labels" (the default), "legend", or "none", controlling how series are identified. "labels" draws a direct label on each line at its right-most value. "legend" instead draws a conventional ggplot legend for the color scale, with "Score" always listed first. "none" draws neither. When text = "labels", the extra right-margin/x-axis space reserved for labels is included, otherwise it's removed. "labels" is preferred when more than one facet is shown, whereas "legend" works better when facets selects only one or two facets.

Details

Calls anlz_score on the four category data.frames to get the overall bay segment score and each category's own score across all years for bay_segment, then plots up to five stacked facets (one column, top to bottom, or a subset via facets): an "Overall" facet with the bay segment score and one colored line per category (wq, sed, fw, hab), followed by one facet per category with that category's own score and one colored line per indicator column of wqoverall, sedoverall, fwoverall, haboverall. In every facet, the score itself is drawn as a thick dark "Score" line and each component (category or indicator) is a thinner colored line, identified either by a direct label or a conventional legend depending on text.

Value

A ggplot object

Examples

```
wqoverall <- anlz_category(
  wq_attain = data.frame(bay_segment = 'OTB', yr = 2018:2020, outcome = c(0.6, 0.7, 0.8))
)
sedoverall <- anlz_category(
  sed_tbbi = data.frame(bay_segment = 'OTB', yr = 2018:2020, outcome = c(0.5, 0.6, 0.6))
)
fwoverall <- anlz_category(
  tbni = data.frame(bay_segment = 'OTB', yr = 2018:2020, outcome = c(0.7, 0.65, 0.7))
)
haboverall <- anlz_category(
  seagrass = data.frame(bay_segment = 'OTB', yr = 2018:2020, outcome = c(0.4, 0.5, 0.5))
)
plot_trend(wqoverall, sedoverall, fwoverall, haboverall, bay_segment = 'OTB')

# a single facet, with no series identification at all
plot_trend(wqoverall, sedoverall, fwoverall, haboverall, bay_segment = 'OTB',
```

```
            facets = 'Overall', text = 'none')

# a conventional legend instead of direct labels
plot_trend(wqoverall, sedoverall, fwoverall, haboverall, bay_segment = 'OTB',
            facets = 'Overall', text = 'legend')
```

`sgsegest` *Seagrass coverage estimates by bay segment and year*

Description

Seagrass coverage estimates by bay segment and year

Usage

```
sgsegest
```

Format

A `data.frame` with 126 rows and 3 columns:

segment Factor with 7 levels, the bay segment name (e.g. "Old Tampa Bay")

acres numeric, estimated seagrass coverage in acres

year numeric, year of the estimate

Details

Seagrass coverage estimates by bay segment, sourced from the seagrass-analysis pipeline. Used in `anlz_hab_seagrass_coverage()` to compare estimated coverage against bay segment targets.

See `data-raw/sgsegest.R` for construction.

Examples

```
sgsegest
```

util_fw_fetch_fim — *Fetch the Tampa Bay FIM species dataset*

Description

Fetch the Tampa Bay FIM species dataset

Usage

```
util_fw_fetch_fim(
  fim_url =
   "https://github.com/kflahertywalia/tb_fim_nonnatives/raw/refs/heads/main/Output/tb_fim_inv.RData'
)
```

Arguments

`fim_url`	chr string, the URL of the FIM species RData file, defaults to the tb_fim_nonnatives repo's output

Details

Downloads and parses the FIM RData file directly (not via `util_rdataload`). Only `Fish`/`Turtle` taxa are present in this dataset. They're recoded to `Fishes`/ `Reptiles` to match `util_fw_fetch_nas`'s group names. Year is recovered from the `Reference` field.

Value

A data.frame with columns `scientificName`, `commonName`, `group`, `year`, `lon`, `lat`, and `source` (always `"FIM"`)

Examples

```
## Not run:
util_fw_fetch_fim()

## End(Not run)
```

util_fw_fetch_nas — *Fetch USGS NAS non-native species occurrences for a set of HUC8 watersheds*

Description

Fetch USGS NAS non-native species occurrences for a set of HUC8 watersheds

Usage

```
util_fw_fetch_nas(
 huc8_list = c("03100101", "03100201", "03100202", "03100203", "03100204", "03100205",
    "03100206", "03100207", "03100208")
)
```

Arguments

huc8_list	chr vector of HUC8 watershed codes to query, defaults to the 9 codes covering the Tampa Bay area (from tbep-invasives' `config/settings.yaml`)

Details

Queries the public USGS Nonindigenous Aquatic Species (NAS) occurrence API once per HUC8 code and combines the results. Dashed NAS taxonomic group names (e.g. `"Amphibians-Frogs"`) are collapsed to their parent group (matching `tbep-invasives`' `download_invasives.py`), and only `Amphibians`, `Crustaceans`, `Fishes`, `Mammals`, `Mollusks`, `Plants`, and `Reptiles` are kept.

Value

A data.frame with columns `scientificName`, `commonName`, `group`, `year`, `lon`, `lat`, and `source` (always `"NAS"`)

Examples

```
## Not run:
util_fw_fetch_nas()

## End(Not run)
```

---

util_fw_nonnative_areas

Bay segment and AOI areas for non-native species density calculations

---

Description

Bay segment and AOI areas for non-native species density calculations

Usage

```
util_fw_nonnative_areas()
```

Details

Recomputes area from geometry (reprojected to EPSG:3086) rather than using `bayseg_shp`'s static `area_ac` attribute, matching the `tbep-invasives` Python pipeline's own area calculation method. `bayseg_shp`'s 7 raw segments are dissolved into 5 groups by `BAY_SEG_GP` before computing area (i.e., BCB, TCB, MR to RALTB).

Value

A named list with `bayseg_area_sqmi` (a named numeric vector of area in square miles, one per `BAY_SEG_GP` group) and `aoi_area_sqmi` (a single number, the total `aoi_shp` area in square miles)

Examples

```
util_fw_nonnative_areas()
```

---

util_fw_nonnative_long

Reshape a non-native metric's wide year-by-segment table to long format

---

Description

Reshape a non-native metric's wide year-by-segment table to long format

Usage

```
util_fw_nonnative_long(df)
```

Arguments

`df`	data.frame with a `year` column, one column per bay segment full name (e.g. `"Old Tampa Bay"`), and an `"All"` column

Details

Drops the `"All"` column (only needed upstream so `util_percentile`'s mean/sd includes it), abbreviates bay segment names, and reverses the percentile to a 0-1 outcome where `1` = best, since higher non-native abundance/richness is worse.

Value

A data.frame with columns `yr`, `bay_segment` (abbreviated, e.g. `"OTB"`, `"RALTB"` for Remainder Lower Tampa Bay), `value` (the input percentile, 0-100), and `outcome` (`1 - value / 100`, 0-1, 1 = best)

Examples

```
df <- data.frame(
  year = 2020,
  `Old Tampa Bay` = 80, `Hillsborough Bay` = 20, `Middle Tampa Bay` = 50,
  `Lower Tampa Bay` = 10, `Remainder Lower Tampa Bay` = 30, All = 40,
  check.names = FALSE
)
util_fw_nonnative_long(df)
```

util_outcome *Convert raw indicator values to a 0-1 outcome score*

Description

Convert raw indicator values to a 0-1 outcome score

Usage

```
util_outcome(
  x,
  type = c("continuous", "threshold", "category"),
  from = NULL,
  reverse = FALSE,
  thresh = NULL,
  op = "<",
  smooth = "logistic",
  pct = 0.1,
  levels = NULL
)
```

Arguments

x	vector of raw values to convert
type	chr string indicating how to interpret x: "continuous" rescales a continuous value, "threshold" compares against a cutoff, "category" maps discrete categories to fixed outcome values
from	numeric vector of length 2, the c(min, max) range of x to rescale from when type = "continuous". Defaults to range(x, na.rm = TRUE).
reverse	logical, if TRUE the outcome is flipped (1 - outcome) so a higher raw value in x indicates a worse outcome.
thresh	numeric, the cutoff value to compare x against when type = "threshold"
op	chr string, one of "<", "<=", ">", ">=". Defines the condition under which x attains an outcome of 1 when type = "threshold"
smooth	for type = "threshold", one of "logistic" (the default), "ramp", or "none". See Details.
pct	numeric, the fraction of abs(thresh) used as the steepness of the "logistic"/"ramp" transition when smooth is one of those. Smaller values give a sharper transition. Defaults to 0.1 (10% of thresh).
levels	named numeric vector mapping each category value in x to a fixed outcome when type = "category", e.g. c(Poor = 0, Fair = 0.5, Good = 1). Values of x not found in levels return NA.

Details

This is meant as the single point of modification for how a raw indicator value becomes a 0-1 outcome, since that conversion is expected to change as individual indicators are refined (e.g. moving a category-based indicator to use its underlying continuous value instead, or a hard threshold to a smooth one) without needing to change every `anlz_*` function that calls it.

continuous: `x` is linearly rescaled from `from` to `c(0, 1)` with `rescale`, then clamped to `c(0, 1)`. Values of `x` outside `from` are pinned to `0` or `1` rather than extrapolated. This lets `from` act as a transition window narrower than the full range of `x` (e.g. TBNI's 32-46 breakpoints within its 0-100 score).

threshold: `x` is compared to `thresh` using `op`, in one of three ways selected by `smooth`:

- `"logistic"` (the default): a smooth logistic transition centered at `thresh`, exactly `0.5` at `thresh` and approaching `0`/`1` on either side, in the direction `op` would have used (e.g. `op = "<"` still means lower `x` is better).
- `"ramp"`: `1` once `x` reaches the "good" side of `thresh` (as defined by `op`). This is not just `0.5` there like `"logistic"`. It decays exponentially toward `0` the further `x` is on the "bad" side. Use this when meeting or beating a target should already be full credit rather than half credit (e.g. an acreage target that's fine to exceed by any amount).
- `"none"`: a hard `0`/`1` cutoff, no transition.

For `"logistic"` and `"ramp"`, the steepness of the transition is a percentage (`pct`) of `thresh` rather than a fixed absolute value, so it stays meaningful across indicators/thresholds with very different units and magnitudes (e.g. a threshold of 1 vs. a threshold in the hundreds).

category: `x` is mapped to an outcome via `levels`.

In all cases, `reverse = TRUE` flips the result so a higher raw value in `x` corresponds to a lower (worse) outcome.

Value

A numeric vector the same length as `x`, with values from 0 to 1 (1 = best).

Examples

```
# continuous, e.g. TBNI scores from 0-100
util_outcome(c(20, 46, 90), type = 'continuous', from = c(0, 100))

# continuous with a narrower transition window, e.g. TBNI's 32-46
# breakpoints. Values outside the window are clamped to 0/1
util_outcome(c(20, 32, 39, 46, 60), type = 'continuous', from = c(32, 46))

# threshold, e.g. chlorophyll attainment (lower is better). Smooth by
# default, a logistic transition rather than a hard cutoff
util_outcome(c(5, 10, 15), type = 'threshold', thresh = 10)

# a larger pct widens the transition (steepness relative to thresh)
util_outcome(12, type = 'threshold', thresh = 10, pct = 0.1)
util_outcome(12, type = 'threshold', thresh = 10, pct = 0.2)

# smooth = "none" instead gives a hard 0/1 cutoff
```

```
util_outcome(c(5, 15), type = 'threshold', thresh = 10, op = '<', smooth = 'none')

# smooth = "ramp": meeting/beating a target (op = ">=") is full credit,
# decaying toward 0 the further short of it a value falls
util_outcome(c(90, 95, 100, 105), type = 'threshold', thresh = 100, op = '>=', smooth = 'ramp')

# category, e.g. FIB grades
util_outcome(c('A', 'C', 'E'), type = 'category',
  levels = c(A = 1, B = 0.75, C = 0.5, D = 0.25, E = 0))
```

`util_percentile` *Convert a wide data frame of values to percentiles by mean and standard deviation*

Description

Convert a wide data frame of values to percentiles by mean and standard deviation

Usage

```
util_percentile(df, id_cols = "year")
```

Arguments

`df`	data.frame with one or more id columns (see `id_cols`) and the remaining columns holding numeric values to convert
`id_cols`	chr vector of column names to carry through unchanged, defaults to `"year"`

Details

Computes a single mean and standard deviation across every value in every non-id column (not per-column), then converts each value to a percentile via `pnorm` of its z-score. This matches the `tbep-invasives` Python pipeline's `as_percentile_by_mean_std()` exactly, including using one global mean/sd for the whole table rather than one per column. If the standard deviation is `0` or undefined (e.g. a single-row input), every value is given a percentile of `50`.

Value

A data.frame the same shape as `df`, with non-id columns converted to percentiles from `0` to `100`

Examples

```
df <- data.frame(year = 2020:2022, OTB = c(10, 20, 30), HB = c(5, 15, 25))
util_percentile(df)
```

util_rdataload *Load an RData file from a URL*

Description

Load an RData file from a URL

Usage

```
util_rdataload(dataurl = NULL)
```

Arguments

dataurl	chr string of the URL for the RData file, expected to contain a single object named the same as the file (minus the .RData extension)

Details

Tries to load the file directly from dataurl. If that fails (e.g. the host doesn't support url() connections), falls back to downloading the file to a temporary location first, then loading it from there. The downloaded file is removed via on.exit() once the function returns, whether the fallback succeeds or fails.

Value

The R object stored in the RData file.

Examples

```
## Not run:
totanndat <- util_rdataload("https://github.com/tbep-tech/load-estimates/raw/main/data/totanndat.RData")

## End(Not run)
```

Index

∗ **datasets**
aoi_shp, 17
bayseg_shp, 18
sgsegest, 23

anlz_category, 2, *9, 10*, *19–21*
anlz_fw_nonnative_abundance, 3, *5*
anlz_fw_nonnative_obs, *3*, 4, *5*
anlz_fw_nonnative_richness, 5, *5*
anlz_fw_tbni, 6
anlz_hab_seagrass_coverage, 7
anlz_hab_seagrass_transect, 8
anlz_indicators, *3*, 9, *10*, *15*, *19*
anlz_score, *3*, *9*, 10, *20*, *22*
anlz_sed_peltel, 11
anlz_sed_tbbi, 12
anlz_wq_attain, 13
anlz_wq_fib, 13
anlz_wq_load, *9*, 14
anlz_wq_thresh, 15
anlz_wq_tidalcreeks, 16
aoi_shp, *5*, 17, *26*

bayseg_shp, *5*, 18, *25*
bind_rows, *9*

complete, *7*

fill, *7*

plot_category, 19, *20*
plot_score, *19*, 20
plot_trend, 21
pnorm, *29*

rescale, *28*

sgsegest, *7*, 23

util_fw_fetch_fim, *4, 5*, 24
util_fw_fetch_nas, *4, 5*, 24, *24*
util_fw_nonnative_areas, *4, 5*, 25
util_fw_nonnative_long, *4, 5*, 26
util_outcome, *6–8*, *11–15*, 27
util_percentile, *4, 5*, *26*, 29
util_rdataload, *24*, 30