

# Ecosystem Service Change & Hotspot Extraction

vv1.0.0

Jerónimo Rodríguez Escobar

2025-12-31 12:49 Pacific Standard Time

## Table of contents

|   |           |
|---|-----------|
| <b>1 Overview</b>   | <b>1</b>  |
| <b>2 Setup and Libraries</b>  | <b>2</b>  |
| <b>3 Metadata Banner</b>  | <b>2</b>  |
| <b>4 Add regional attributes to Grid (countries &amp; biomes)</b>               | <b>2</b>  |
| <b>5 Produce signed change bars</b>   | <b>3</b>  |
| <b>6 Load cached long table (plt_long) for hotspots/KS</b>                      | <b>6</b>  |
| <b>7 Hotspot extraction workflow (global + subregional)</b>                     | <b>6</b>  |
| 7.1 Hotspot rules & export configuration . . . . .                              | 6         |
| 7.2 Validate hotspot configuration . . . . .                                    | 7         |
| 7.3 Export hotspot layers . . . . .   | 8         |
| <b>8 Trimmed change bar plots</b>   | <b>8</b>  |
| 8.1 TODO: Hotspot vs non-hotspot contribution bars . . . . .                    | 8         |
| 8.2 Signed means (alt variants: keep0 vs drop0; dashed = global mean) . . . . . | 9         |
| <b>9 Hotspot violin plots</b>   | <b>19</b> |

## 1 Overview

Prepares the processed 10 km grid, derives hotspot flags, and produces the bar/violin plots that summarize signed and absolute change by group (region, income, biome). This version uses the corrected 10 km equal-area (AOO land-only) grid. Key steps: load helpers and paths, attach grouping attributes, pivot change fields to long form, define hotspots using the configured thresholds, export hotspot layers, and write the latest plots to outputs/plots/latest/.

## 2 Setup and Libraries

Load all core packages (tidyverse, spatial, plotting helpers) and initialize reproducibility settings. This chunk also sources `R(paths.R)` and runs `devtools::load_all()` so downstream chunks can reuse helper functions.

## 3 Metadata Banner

Published: 2025-12-31 13:09 PST

### Note

#### Run metadata

- Analysis version: v1.0.0
- Rendered: 2025-12-31 13:09 PST
- Git: feat/ks-analysis @ 718025e
- Data root: /home/jeronimo/data/global\_ncp

## 4 Add regional attributes to Grid (countries & biomes)

Attach country and biome IDs to the 10 km grid (`sf_f`) so downstream aggregation/hotspot steps can group by region. This chunk only runs if the processed GPKG is missing.

The input 10 km grid (`sf_f`) is enriched with country and WWF biome IDs to allow aggregation and comparisons of change by subregions (World Bank region, income group, continent, UN region, and biome). We use a point-on-surface join to avoid sliver/overlap issues, keep only the needed fields, and write a single enriched GPKG for downstream grouping, hotspot extraction, and plotting.

### Tip

#### What this step does

- Reads country and biome layers from `vectors/`.
- Joins them to the 10 km grid via point-on-surface.
- Writes `processed/10k_change_calc.gpkg` for downstream analysis (pivoting, hotspots, plots).  
**Inputs:** `sf_f` grid; `vectors/cartographic_ee_ee_r264_correspondence.gpkg`; `vectors/Biome.gpkg`  
**Output:** `processed/10k_change_calc.gpkg` (grid + regional attributes)

**Why point-on-surface?** It's robust for odd cell shapes and avoids polygon–polygon sliver issues.

```
attach-region-attrs: existing 10k_change_calc.gpkg is non-spatial; skipping join.
```

```
Skipping join: /home/jeronimo/data/global_ncp/processed/10k_change_calc.gpkg already exists.
```

## 5 Produce signed change bars

Bars are computed directly from `processed/change_calc.gpkg`. We output two variants—zeros kept vs zeros dropped—both with trimmed tails and a dashed global mean. Pivoted `plt_long` is only necessary for hotspots/KS. Trim/no-trim diagnostics and sign-flip tests live in `analysis/hotspot_sandbox.qmd`.

What this chunk does:

- Read the processed GPKG and normalize IDs (`fid`, `c_fid`), dropping any `c_fid.x/y` artifacts.
- Build a service mapping (`svc_map`) so raw column bases (with/without `_mean`) map to the canonical services; construct `col_pct`/`col_abs` names.
- `agg_by_group()`: intersect requested groupings with available columns; for each grouping  $\times$  service, pick the pct/abs change column, handle Inf pct values (`handle_inf`), optionally drop zeros (`drop_zero`), trim extremes at `cut_q`, then compute mean change by group. Returns tidy rows with `grouping`, `group`, `service`, `metric`, `mean_chg`.
- Run `agg_by_group()` two ways: trimmed keep0/drop0  $\rightarrow$  `signed_keep0`, `signed_drop0`.
- A later chunk computes global reference means; plotting is in its own chunk so you can skip it when debugging.

```
services <- svc_order
groupings <- c("income_grp", "region_wb", "continent", "region_un", "WWF_biome")
cut_q <- 0.999
handle_inf <- "na" # options: "na", "cap"
verbose <- TRUE
# helper for lightweight progress messages

vmsg <- function(...) if (isTRUE(verbose)) message(...)

gpkg <- file.path(Sys.getenv("GLOBAL_NCP_DATA"), "processed", "10k_change_calc.gpkg")
stopifnot(file.exists(gpkg))

vmsg("Reading: ", pkg, " [layer: 10k_change_calc]")
sf_f <- sf::st_read(gpkg, layer = "10k_change_calc", quiet = TRUE)
sf_f <- sf::st_drop_geometry(sf_f)
if (!"fid" %in% names(sf_f)) sf_f$fid <- seq_len(nrow(sf_f))
if (!"c_fid" %in% names(sf_f)) {
  if ("c_fid.x" %in% names(sf_f)) sf_f <- dplyr::rename(sf_f, c_fid = c_fid.x)
  else if ("c_fid.y" %in% names(sf_f)) sf_f <- dplyr::rename(sf_f, c_fid = c_fid.y)
  else if ("id" %in% names(sf_f)) sf_f <- dplyr::rename(sf_f, c_fid = id)
}
sf_f <- dplyr::select(sf_f, -dplyr::any_of(c("c_fid.x", "c_fid.y")))

# build mapping; include both raw base and no_mean base, with ready-to-use columns
chg_cols <- grep("_abs|pct|chg$", names(sf_f), value = TRUE)
vmsg("Found ", length(chg_cols), " change columns; mapping services...")
services_raw <- unique(sub("_abs|pct|chg$", "", chg_cols))      # as in file (may include _mean)
services_clean <- stringr::str_remove(services_raw, "_mean$")
```

```

services_lower <- tolower(services_clean)
svc_map <- tibble::tibble(
  col_base = c(services_raw, services_clean),
  canonical = dplyr::recode(c(services_lower, services_lower),
                            !!!canonical_lookup, .default = c(services_clean, services_clean)))
) |>
  dplyr::distinct() |>
  dplyr::mutate(col_pct = paste0(col_base, "_pct_chg"),
                col_abs = paste0(col_base, "_abs_chg"))

agg_by_group <- function(df, services, groupings, cut_q = 0.999,
                         handle_inf = c("na", "cap"), drop_zero = FALSE) {
  handle_inf <- match.arg(handle_inf)
  groupings <- intersect(groupings, names(df))
  vmsg("Groupings available: ", paste(groupings, collapse = ", "))
  out <- list()
  for (g in groupings) {
    vmsg(" Grouping: ", g)
    for (svc in services) {
      map_rows <- dplyr::filter(svc_map, canonical == svc)
      if (!nrow(map_rows)) next
      cols_pct <- map_rows$col_pct[map_rows$col_pct %in% names(df)]
      cols_abs <- map_rows$col_abs[map_rows$col_abs %in% names(df)]
      if (!length(cols_pct) && !length(cols_abs)) next

      add_one <- function(col, metric) {
        v <- df[[col]]
        if (metric == "pct" && any(is.infinite(v))) {
          if (handle_inf == "na") v[is.infinite(v)] <- NA_real_
        }
        if (isTRUE(drop_zero)) v[v == 0] <- NA_real_
        cap <- stats::quantile(abs(v), cut_q, na.rm = TRUE)
        v_trim <- pmax(pmin(v, cap), -cap)
        tibble::tibble(
          service = svc,
          group = df[[g]],
          metric = metric,
          val = v_trim
        )
      }
    }

    rows <- list()
    if (length(cols_pct)) rows[[length(rows)+1]] <- add_one(cols_pct[1], "pct")
    if (length(cols_abs)) rows[[length(rows)+1]] <- add_one(cols_abs[1], "abs")
    rows <- dplyr::bind_rows(rows)
    if (!nrow(rows)) next
    rows <- rows |>
  }
}

```

```

    dplyr::filter(!is.na(group)) |>
    dplyr::group_by(service, metric, group) |>
    dplyr::summarise(mean_chg = mean(val, na.rm = TRUE), .groups = "drop") |>
    dplyr::mutate(grouping = g)
    out[[length(out)+1]] <- rows
  }
}
dplyr::bind_rows(out)
}

signed_keep0 <- agg_by_group(sf_f, services, groupings, cut_q = cut_q,
                             handle_inf = handle_inf, drop_zero = FALSE)
signed_drop0 <- agg_by_group(sf_f, services, groupings, cut_q = cut_q,
                             handle_inf = handle_inf, drop_zero = TRUE)
print(dplyr::count(signed_keep0, grouping, metric, service))

```

```

# A tibble: 80 x 4
  grouping metric service      n
  <chr>    <chr>  <chr>     <int>
1 WWF_biome abs   C_Risk       16
2 WWF_biome abs   C_Risk_Red_Ratio 16
3 WWF_biome abs   N_Ret_Ratio   16
4 WWF_biome abs   N_export     16
5 WWF_biome abs   Nature_Access 16
6 WWF_biome abs   Pollination   16
7 WWF_biome abs   Sed_Ret_Ratio 16
8 WWF_biome abs   Sed_export    16
9 WWF_biome pct   C_Risk       16
10 WWF_biome pct  C_Risk_Red_Ratio 16
# i 70 more rows

```

```
print(dplyr::count(signed_drop0, grouping, metric, service))
```

```

# A tibble: 80 x 4
  grouping metric service      n
  <chr>    <chr>  <chr>     <int>
1 WWF_biome abs   C_Risk       16
2 WWF_biome abs   C_Risk_Red_Ratio 16
3 WWF_biome abs   N_Ret_Ratio   16
4 WWF_biome abs   N_export     16
5 WWF_biome abs   Nature_Access 16
6 WWF_biome abs   Pollination   16
7 WWF_biome abs   Sed_Ret_Ratio 16
8 WWF_biome abs   Sed_export    16
9 WWF_biome pct   C_Risk       16
10 WWF_biome pct  C_Risk_Red_Ratio 16
# i 70 more rows

```

## 6 Load cached long table (plt\_long) for hotspots/KS

Hotspot export and KS still expect plt\_long. To avoid the heavy pivot here, we load the cached table from outputs/tables/plt\_long.rds. If it's missing, run scratch/pivot\_long.qmd (or scratch/pivot\_long.R) to regenerate it.

## 7 Hotspot extraction workflow (global + subregional)

We identify per-service hotspots using a 5% percentile rule and **direction vectors**: - **loss\_services** (e.g., Nature\_Access, Pollination, N/Sed\_Ret\_Ratio, C\_Risk\_Red\_Ratio) → we flag the **lowest** values; - **gain\_services** (Sed\_export, N\_export, C\_Risk) → we flag the **highest** values.

We run this **once globally** and then **once per subregion** (World Bank region, income group, continent, UN region, WWF biome). For each run and for each metric (absolute and percent change) we write a compact **GPKG** containing only the hotspot cells, plus a CSV index:

- Output root: processed/hotspots/
  - abs/global/hotspots\_global\_abs.gpkg
  - pct/global/hotspots\_global\_pct.gpkg
  - abs/<group\_col>/hotspots\_<group\_col>\_<group\_val>\_abs.gpkg
  - pct/<group\_col>/hotspots\_<group\_col>\_<group\_val>\_pct.gpkg
- Index: processed/hotspots/\_hotspots\_index.csv (columns: scope, group\_col, group\_val, metric, n\_hot, gpkg).

These files are meant for QGIS/QA and downstream stats (e.g., KS) without recomputing hotspots.

### 7.1 Hotspot rules & export configuration

The analysis uses a single, central configuration so the hotspot rules are consistent everywhere:

Thresholding: we flag hotspots using the top/bottom tails of the distribution per service. Here we use a percentile cutoff (e.g., 5%) rather than a fixed count.

Direction of concern: services in loss are “worse when they go down” (we keep the lowest tail); services in gain are “worse when they go up” (we keep the highest tail).

Combos (optional): grouped service sets that we count per cell for quick composite summaries.

Export switches: choose whether to write GPKGs and/or the CSV index.

```
HOTS_CFG <- list(  
  analysis_name = "global_NCP_hotspots",  
  pct_cutoff = 0.05,  
  threshold_mode = "percent",  
  rule_mode = "vectors",  
  loss = c("Nature_Access", "Pollination", "N_Ret_Ratio", "Sed_Ret_Ratio", "C_Risk_Red_Ratio"),  
  gain = c("Sed_export", "N_export", "C_Risk"),
```

```

combos = list(
  deg_combo = c("Nature_Access", "Pollination", "N_export", "Sed_export", "C_Risk"),
  rec_combo = c("Nature_Access", "Pollination", "N_Ret_Ratio", "Sed_Ret_Ratio", "C_Risk_Red_Ratio"),
),
# centralize the grouping columns here
groupings = c("income_grp", "region_wb", "continent", "region_un", "WWF_biome"),
# IO
write_layers = TRUE,
write_index = TRUE,
out_dir = file.path(data_dir(), "processed", "hotspots")
)

```

::: callout-note  
**\*\*Hotspot configuration\*\***

- Cutoff: 5% (percent)
- Rule mode: vectors
- Loss services: Nature\_Access, Pollination, N\_Ret\_Ratio, Sed\_Ret\_Ratio, C\_Risk\_Red\_Ratio
- Gain services: Sed\_export, N\_export, C\_Risk
- Combos: **\*\*deg\_combo\*\***: Nature\_Access, Pollination, N\_export, Sed\_export, C\_Risk  
**\*\*rec\_combo\*\***: N\_Ret\_Ratio, Sed\_Ret\_Ratio, C\_Risk\_Red\_Ratio
- ..
- Groupings: income\_grp, region\_wb, continent, region\_un, WWF\_biome
- ...- Write layers: TRUE | Write index: TRUE
- Output dir: `/home/jeronimo/data/global\_ncp/processed/hotspots`

:::

## 7.2 Validate hotspot configuration

```

# stopifnot(exists("plt_long"))
# svc_all <- unique(plt_long$service)
# if (length(intersect(HOTS_CFG$loss, HOTS_CFG$gain)) > 0) {
#   stop("A service appears in BOTH `loss` and `gain`. Fix HOTS_CFG.")
# }
# miss_loss <- setdiff(HOTS_CFG$loss, svc_all)
# miss_gain <- setdiff(HOTS_CFG$gain, svc_all)
# if (length(miss_loss) > 0 || length(miss_gain) > 0) {
#   warning("Services in HOTS_CFG not found in `plt_long$service`:\n",
#         if (length(miss_loss)) paste0(" - missing loss: ", paste(miss_loss, collapse=", ")),
#         if (length(miss_gain)) paste0(" - missing gain: ", paste(miss_gain, collapse=", ")))
# }

```

## 7.3 Export hotspot layers

### Note

#### **Hotspot export module**

- Computes hotspot cells once (global + by subregion) for ABS and PCT change.
- Writes compact GPKGs for mapping/QA and maintains `_hotspots_index.csv`.
- Prereqs: `plt_long` in memory, `HOTS_CFG` defined (loss/gain/combos/etc.).

## 8 Trimmed change bar plots

### Note

#### **How to read these bars**

- Each bar shows the **trimmed mean absolute change** ( $|\Delta|$ ) per service within each group; facet axes are free.
- Bars are **always positive** by design: height = **magnitude of change**, not direction.
- Direction-of-concern used elsewhere in the analysis:
  - Worse when `up Sed_export, N_export, C_Risk`.
  - Worse when `down Nature_Access, Pollination, N_Ret_Ratio, Sed_Ret_Ratio, C_Risk_Red_Ratio`.
- These bars answer “**where is change largest?**”. See hotspot maps/violins for **up vs. down** patterns.

Short answer: your current barplots use all grid cells (the full 10-km population), not just hotspots. They summarize trimmed means per subregion/global from `plt_long` via `aggregate_change_simple()`, with `cut_q=0.999` to cap outliers and (optionally) `drop_zeros=TRUE`. That’s why every bar is positive—those bars are the magnitude of change, not the direction.

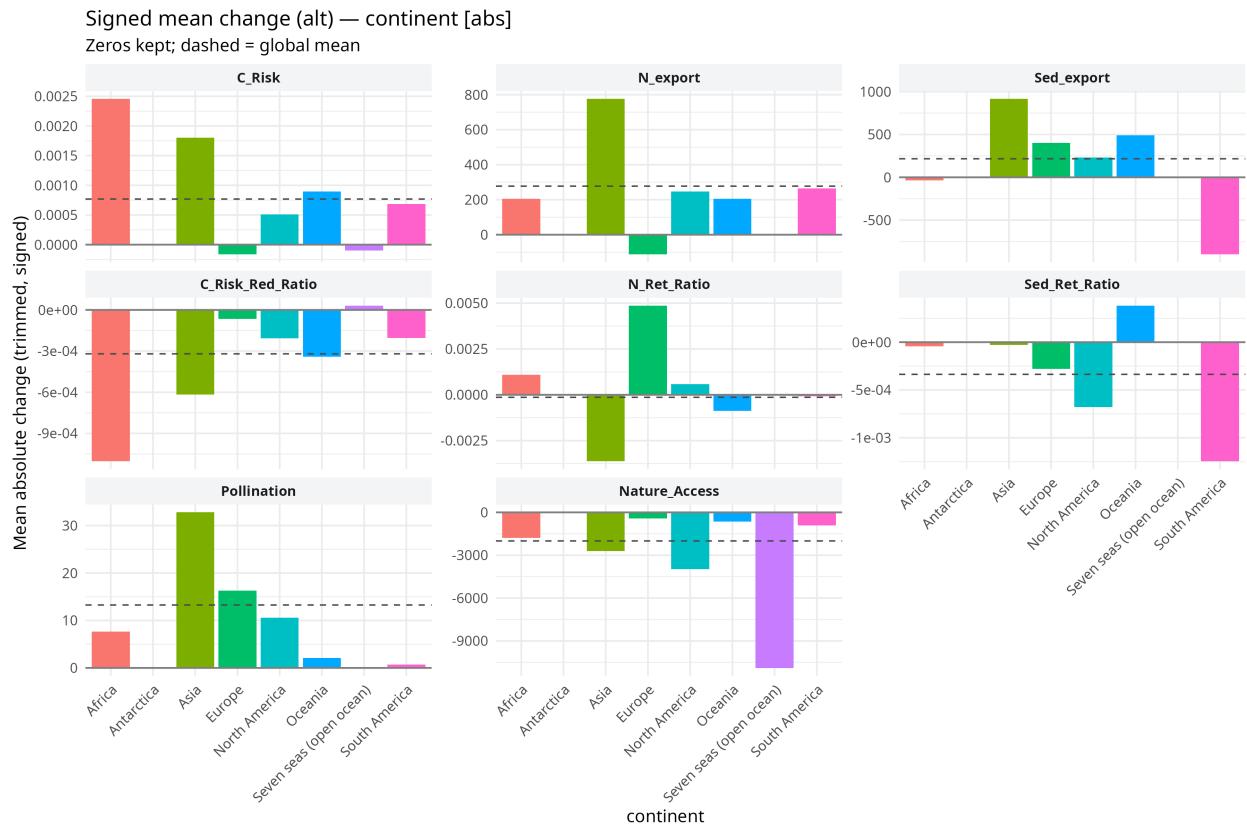
Outputs: PNGs land in `outputs/plots/{abs|pct}/<group_col>/bars_*.png` plus a flat copy in `outputs/plots/latest/bars/` for embedding here. The legacy global-only single-bar chart was removed to keep the gallery focused on grouped views.

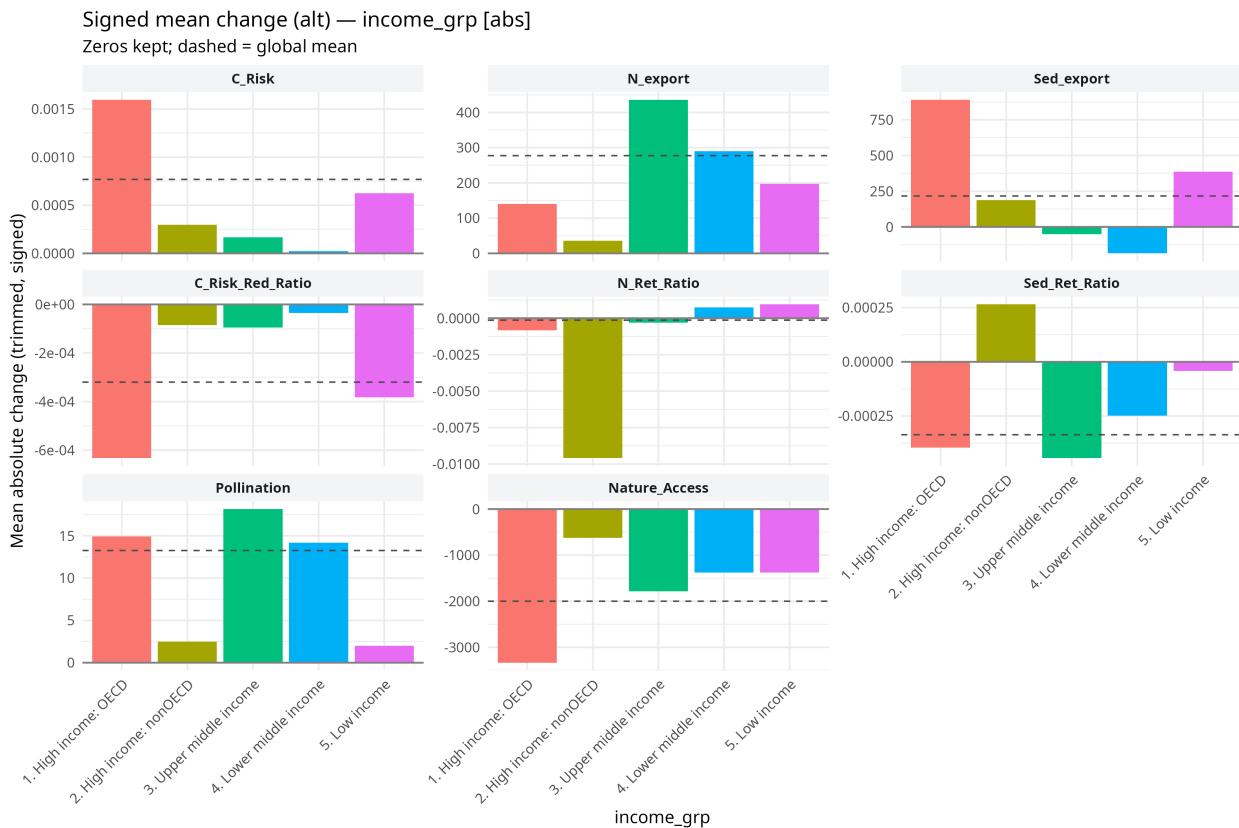
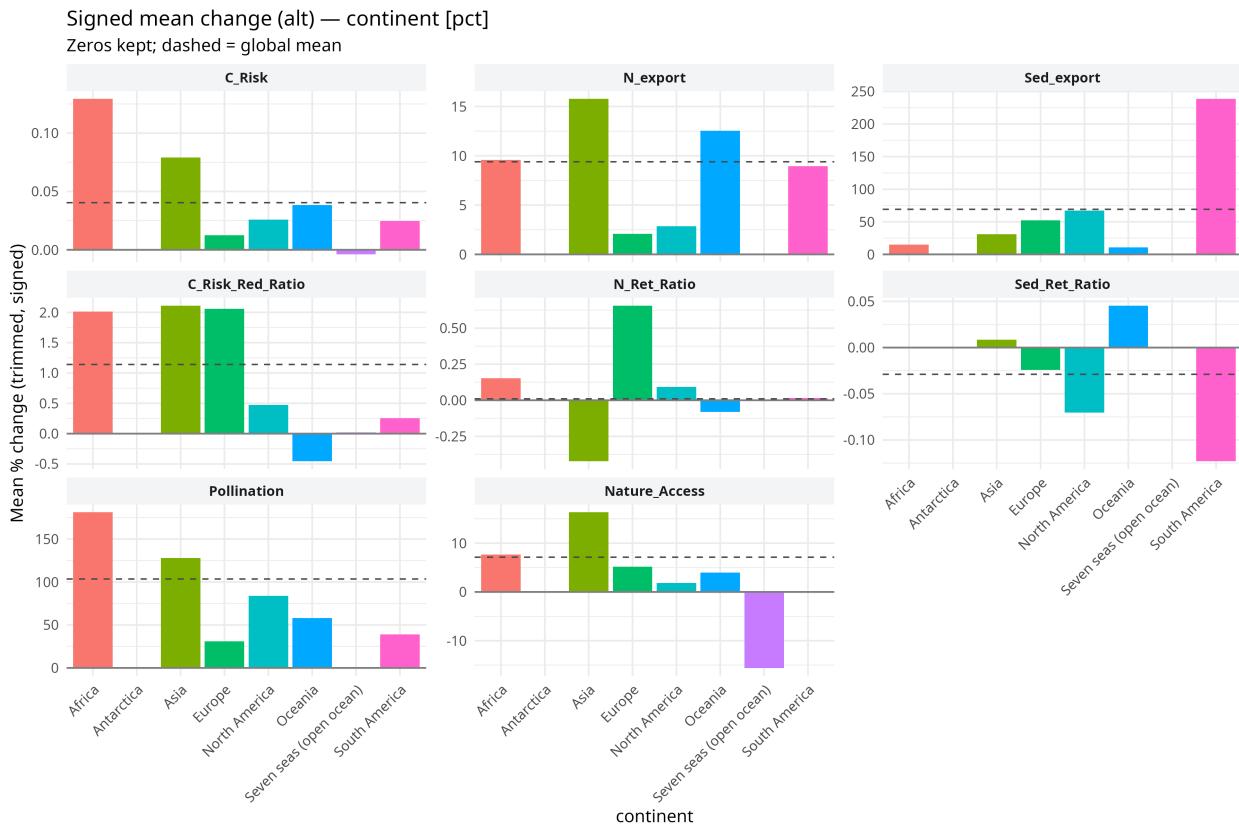
Generate trimmed-mean bar plots (abs & pct) for each grouping. Plots save to `outputs/plots/...` and are embedded below.

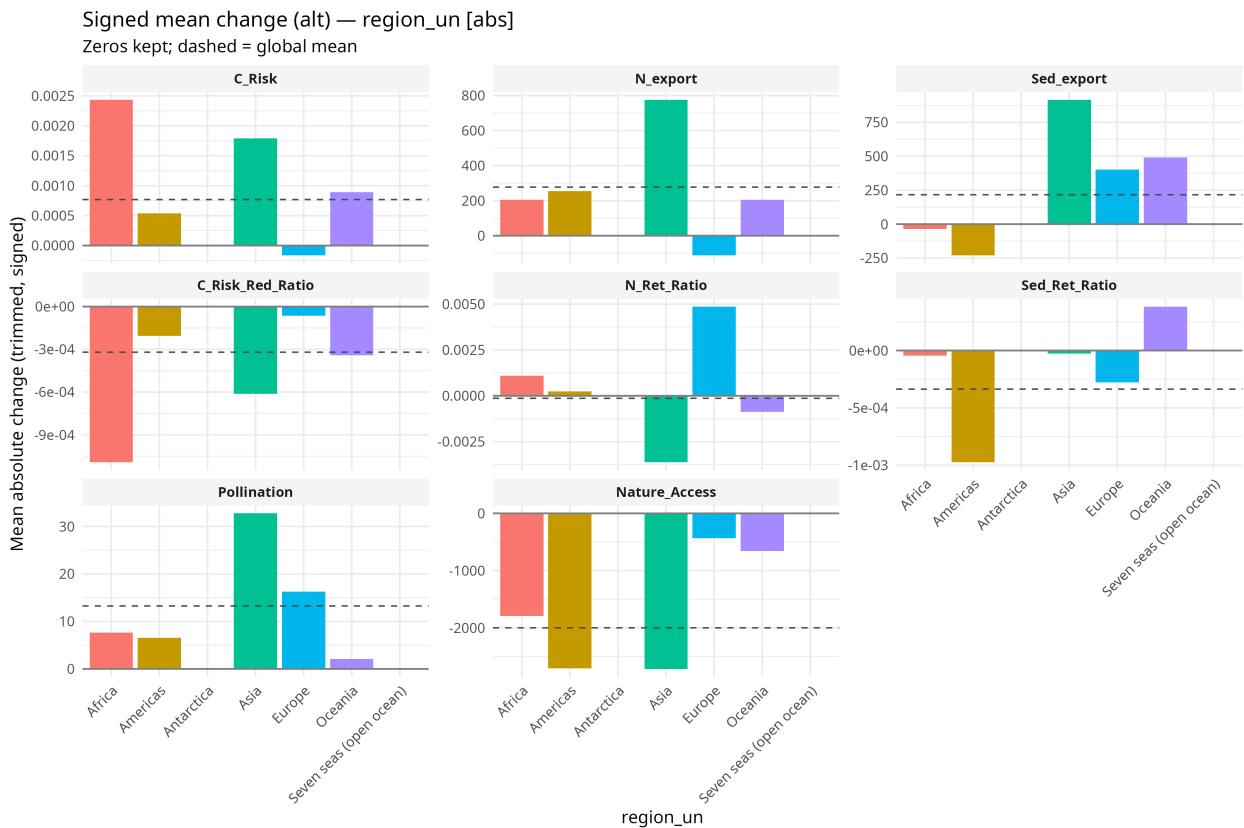
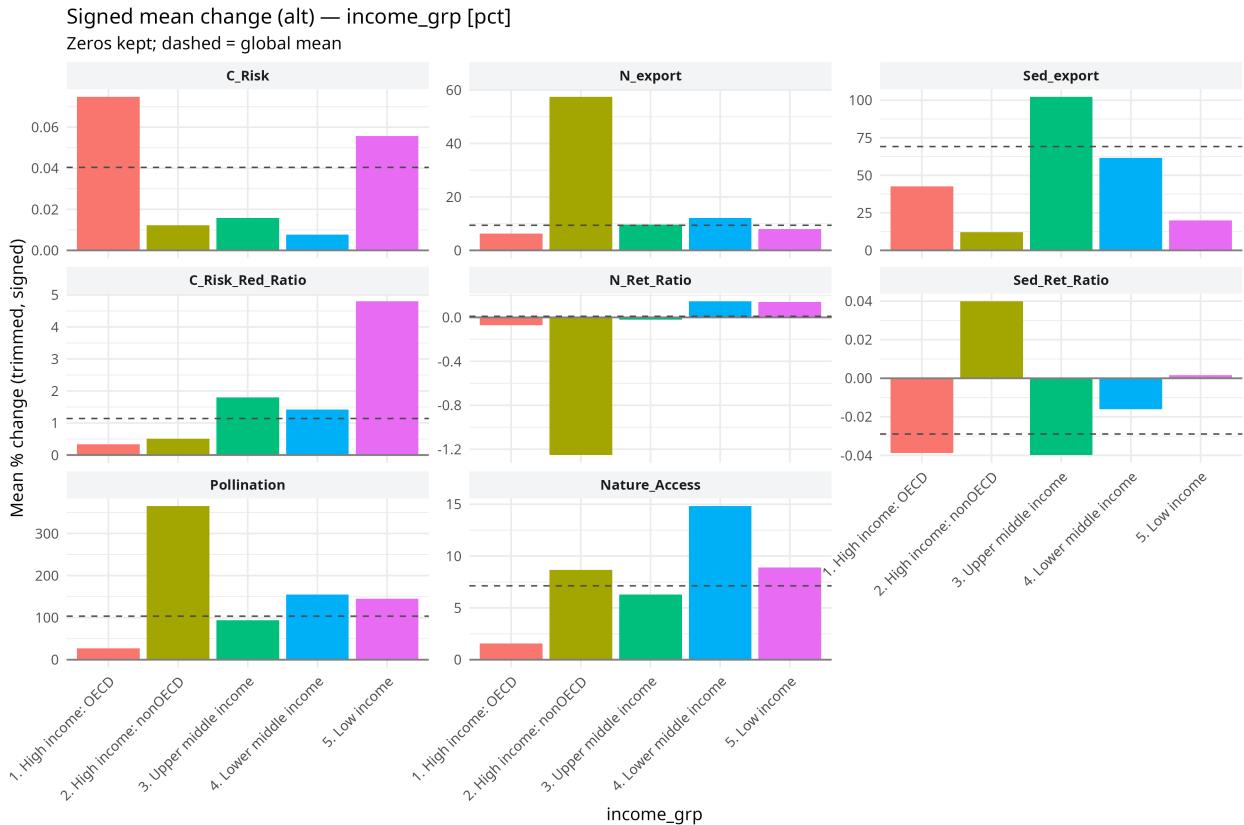
### 8.1 TODO: Hotspot vs non-hotspot contribution bars

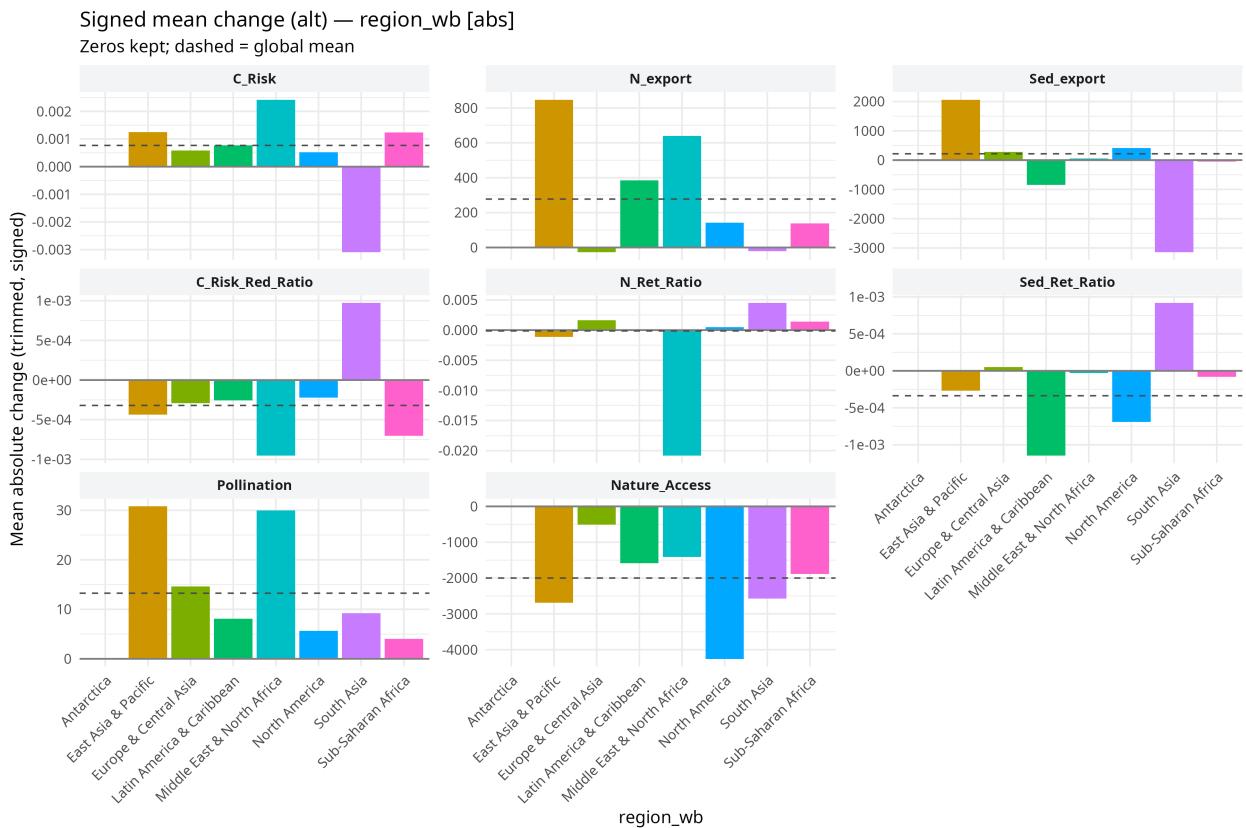
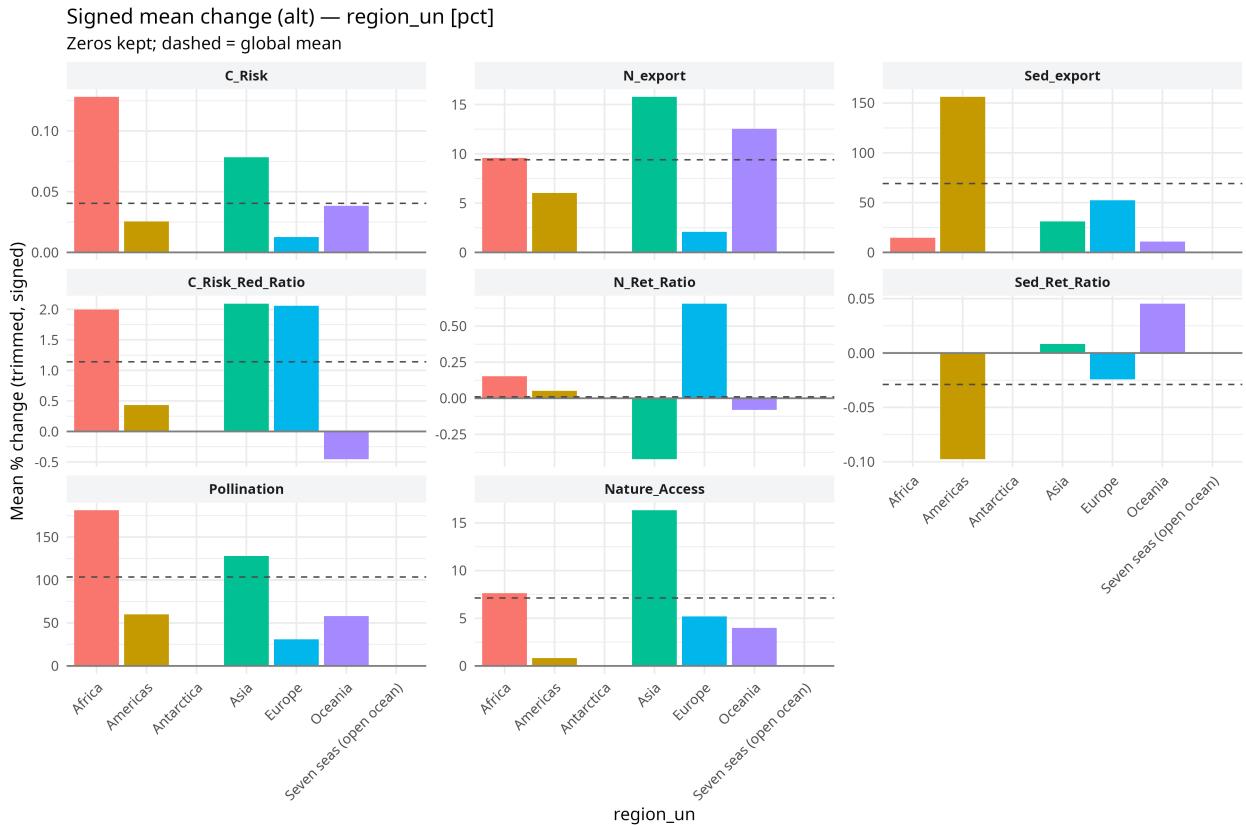
Future idea: stack bars to show how much of each group’s total signed change comes from hotspot cells vs the complement. Steps: compute group means separately for hotspot and non-hotspot subsets (using the same 5% rules), then plot stacked bars (hotspot share + non-hotspot share) per service/group. Keep both absolute and percent-change variants; avoid overwriting existing plots (write to `outputs/plots/signed_hotshare/`).

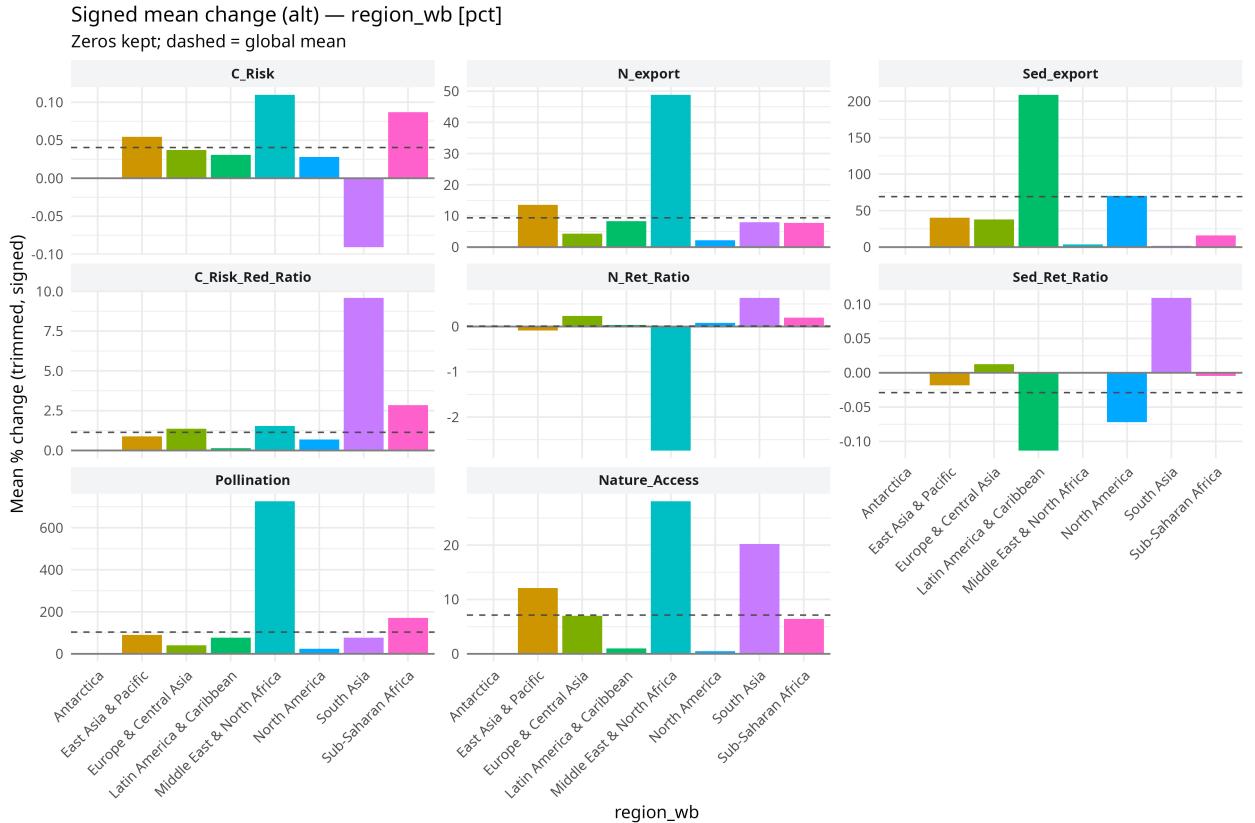
## 8.2 Signed means (alt variants: keep0 vs drop0; dashed = global mean)

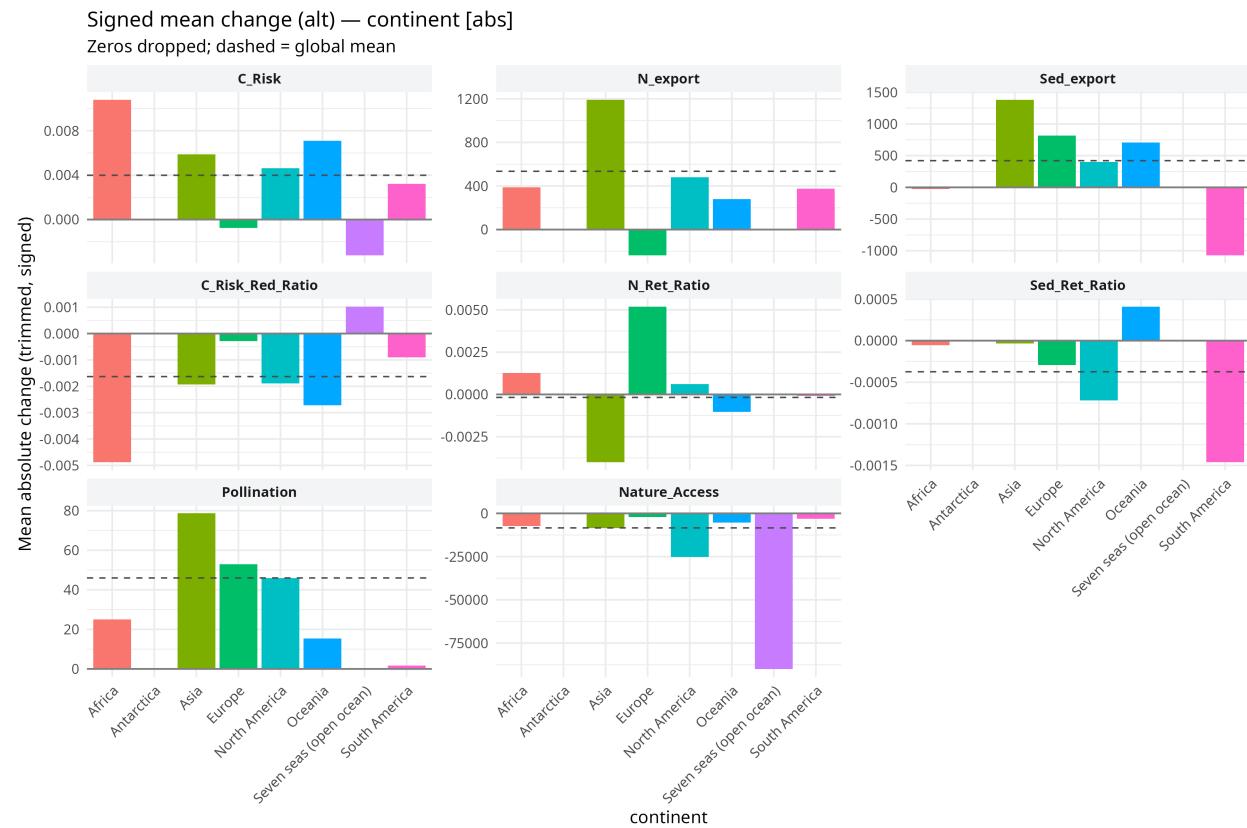


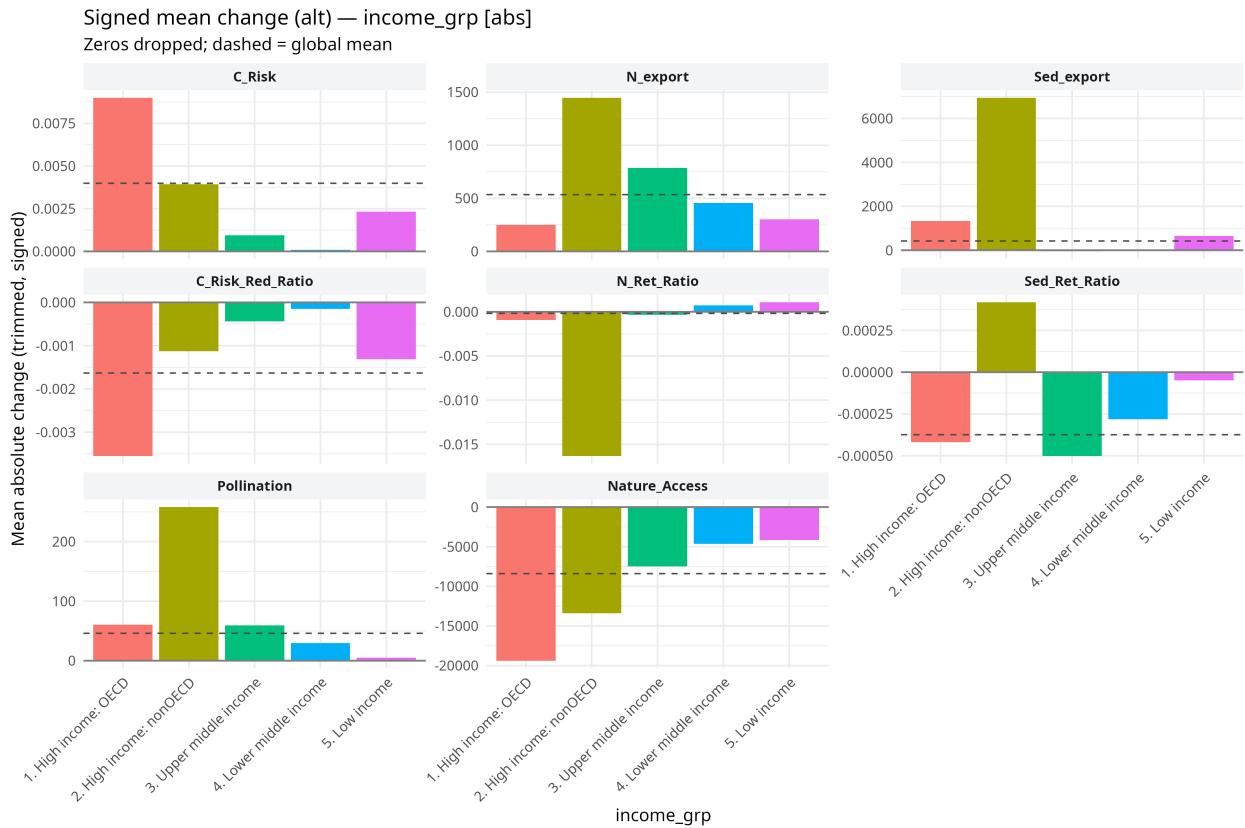
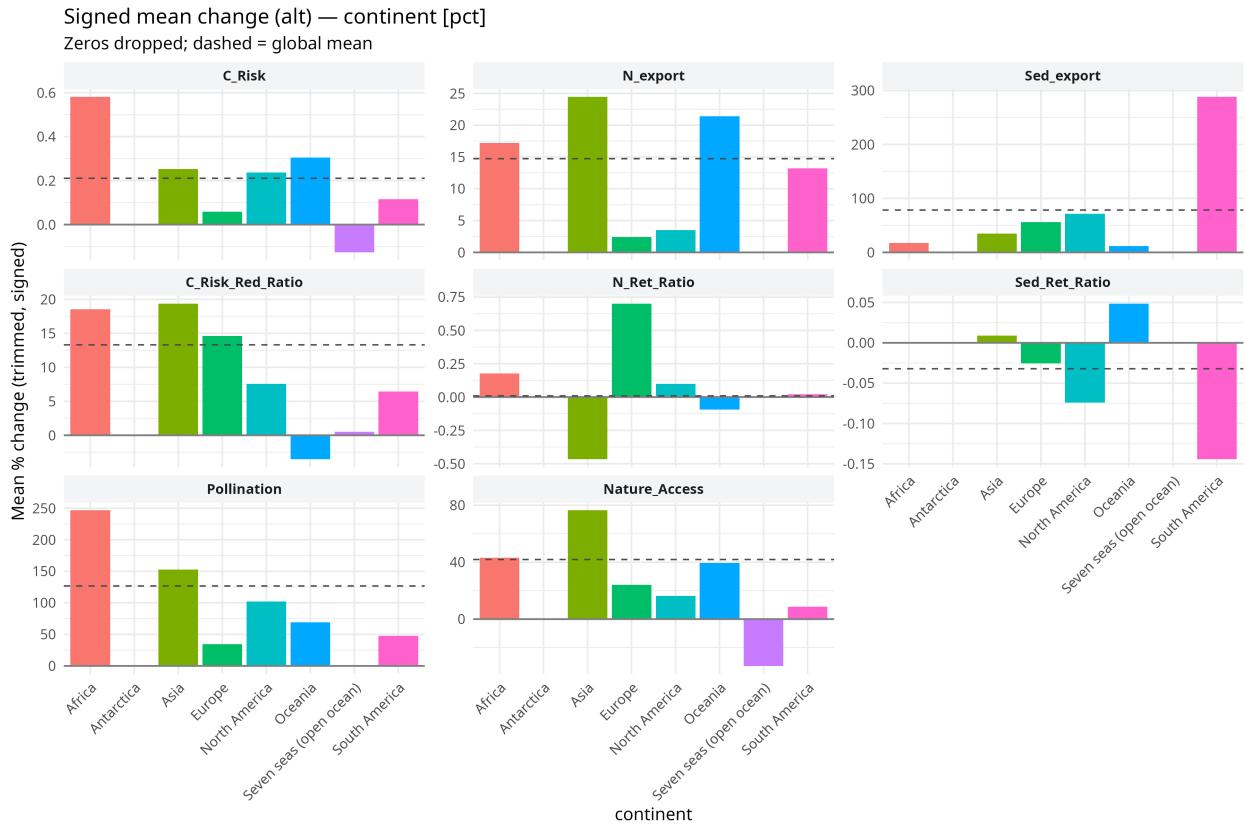


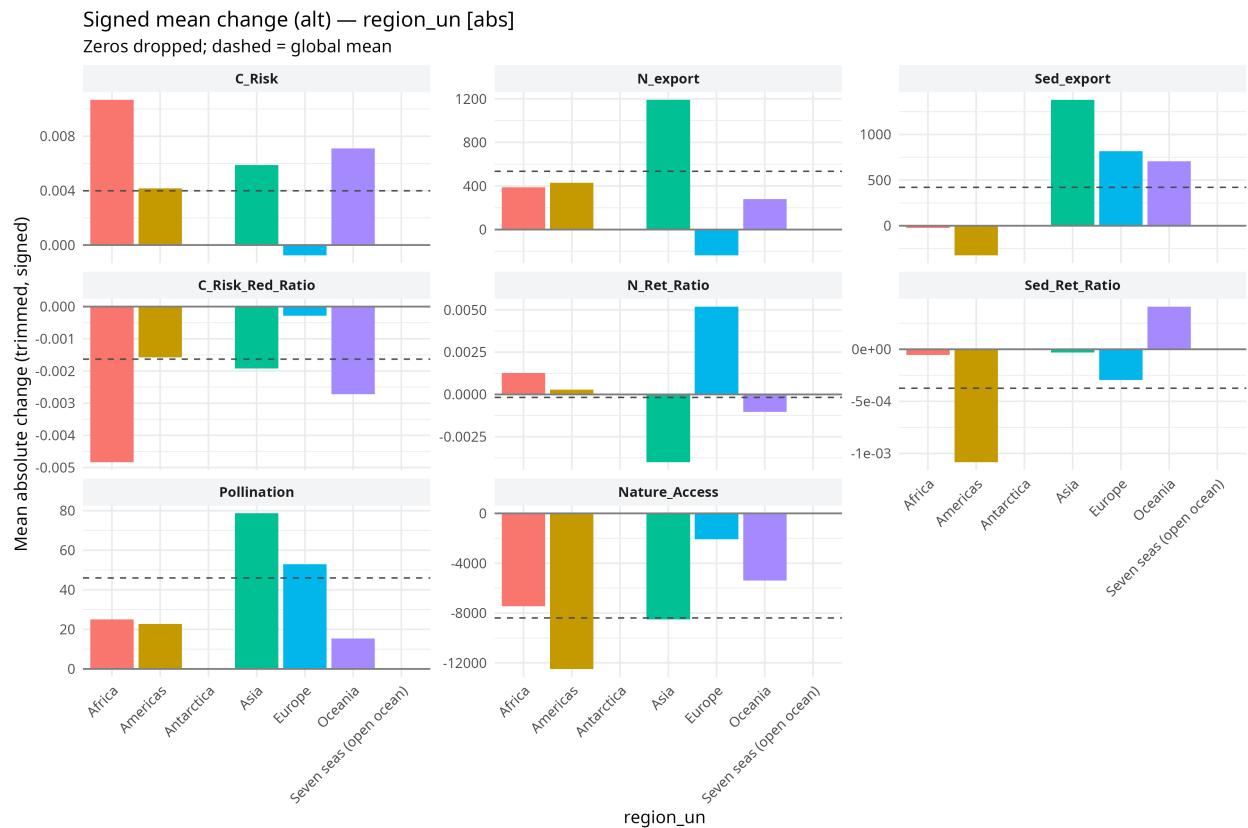
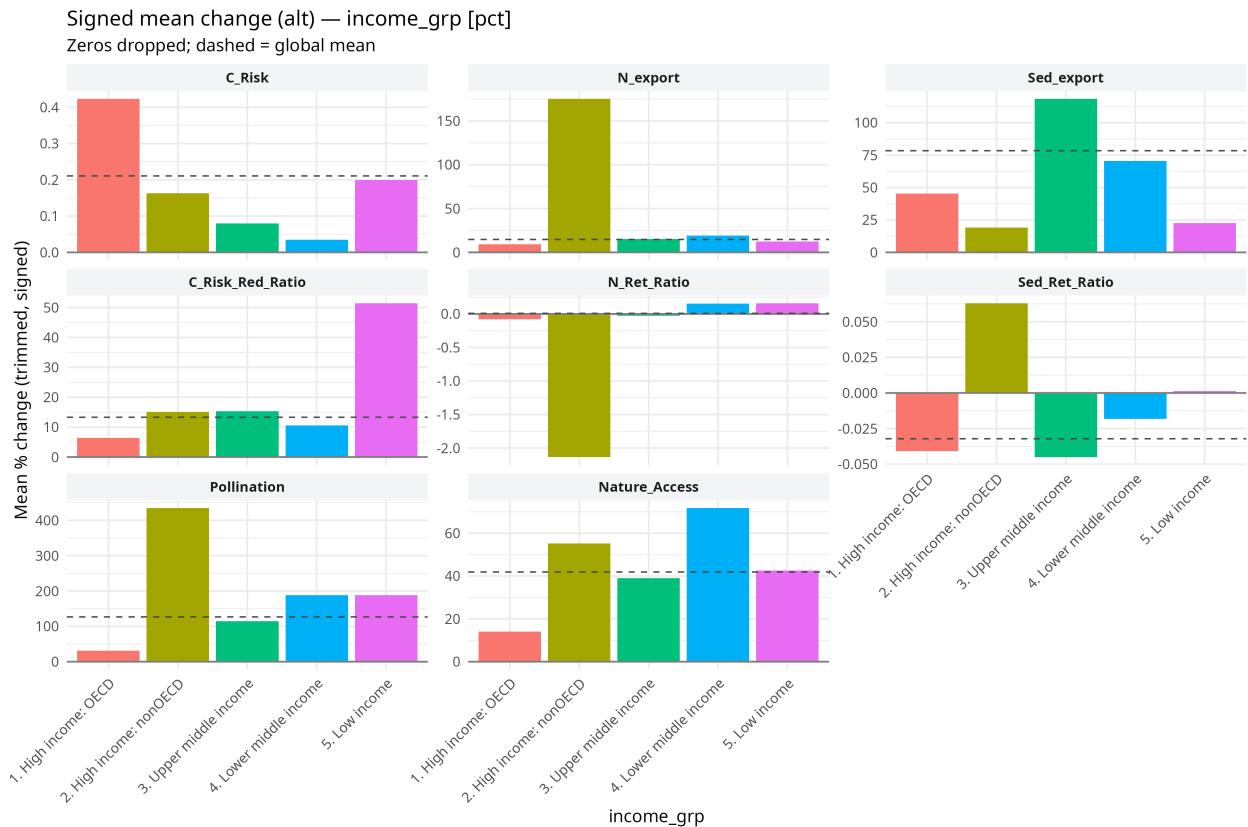


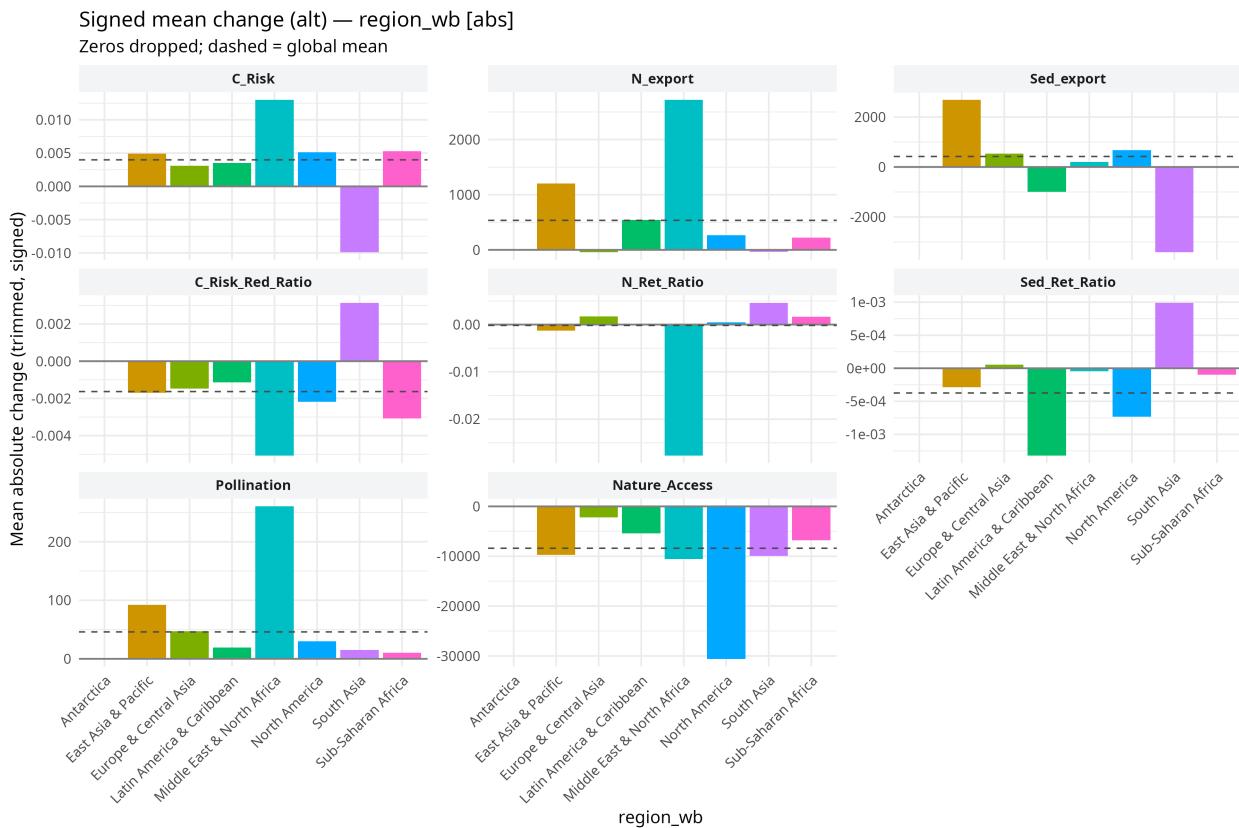
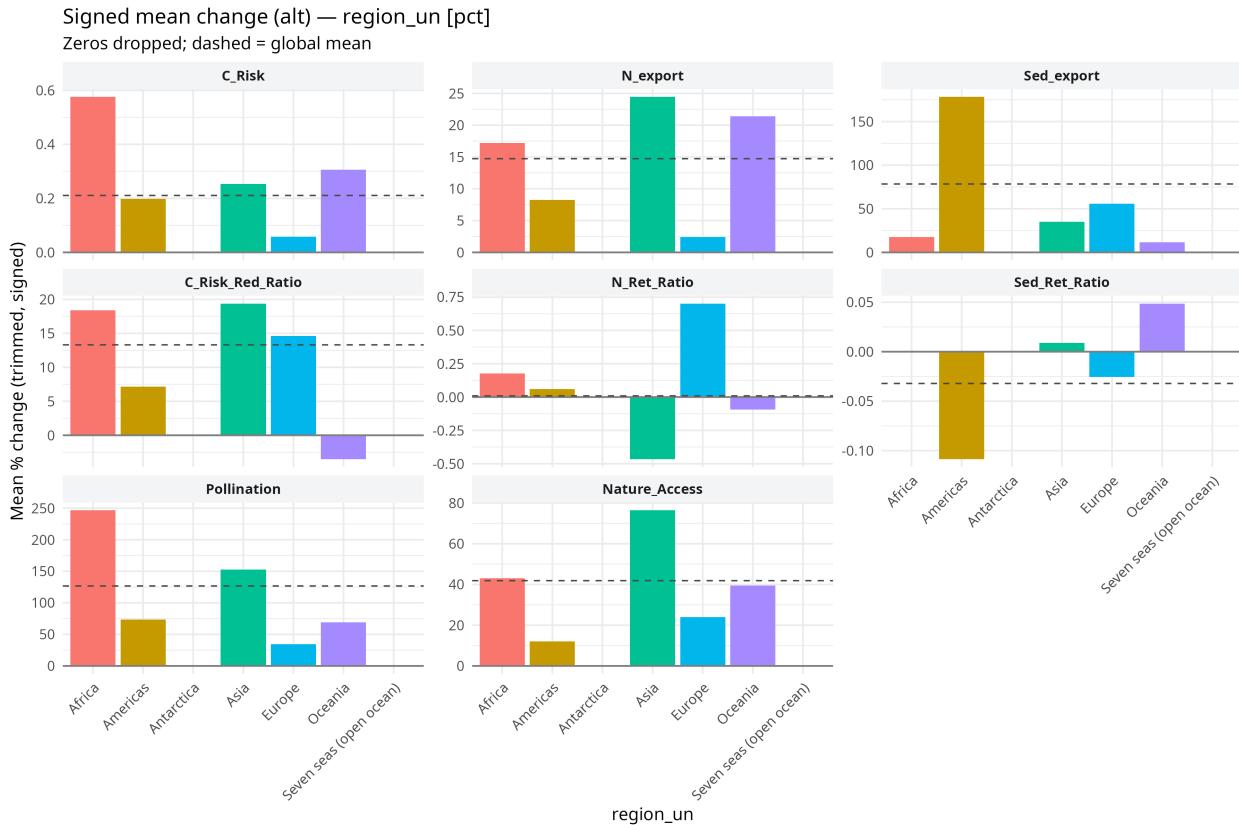


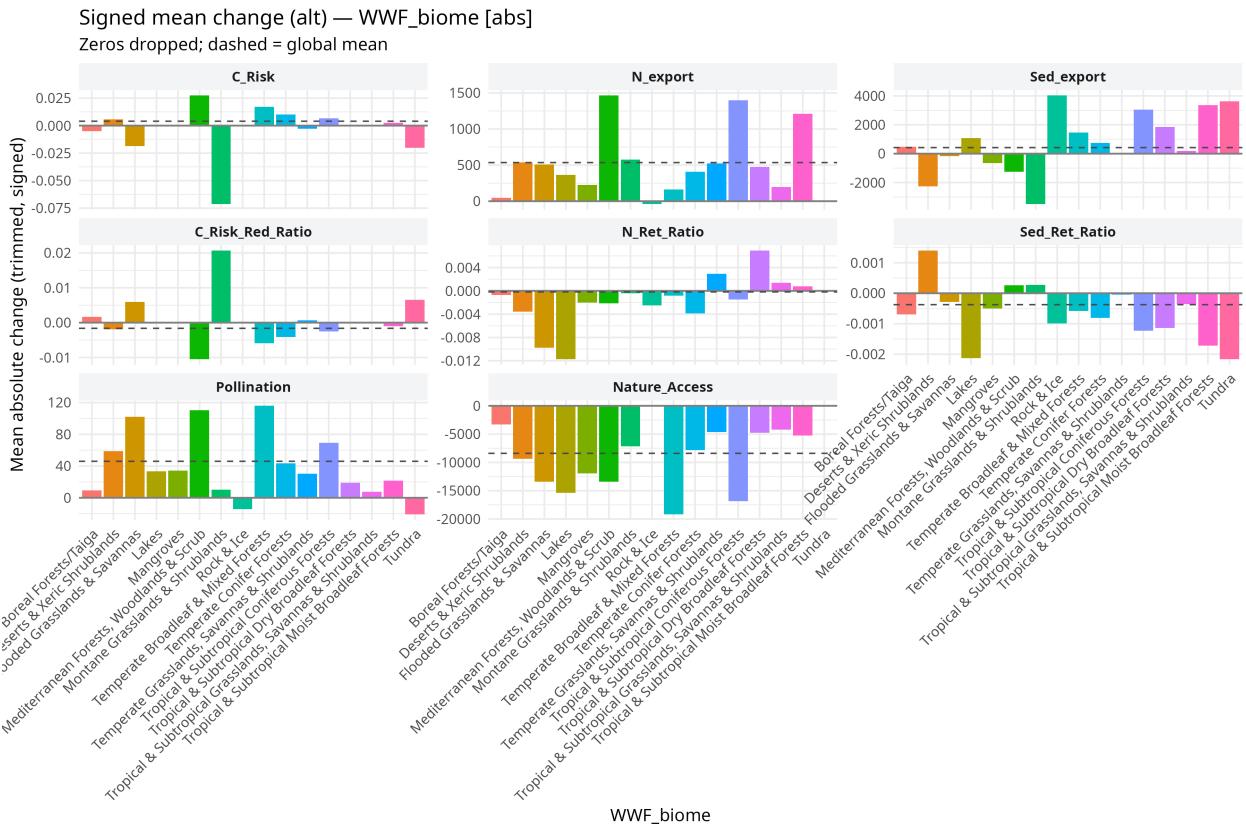
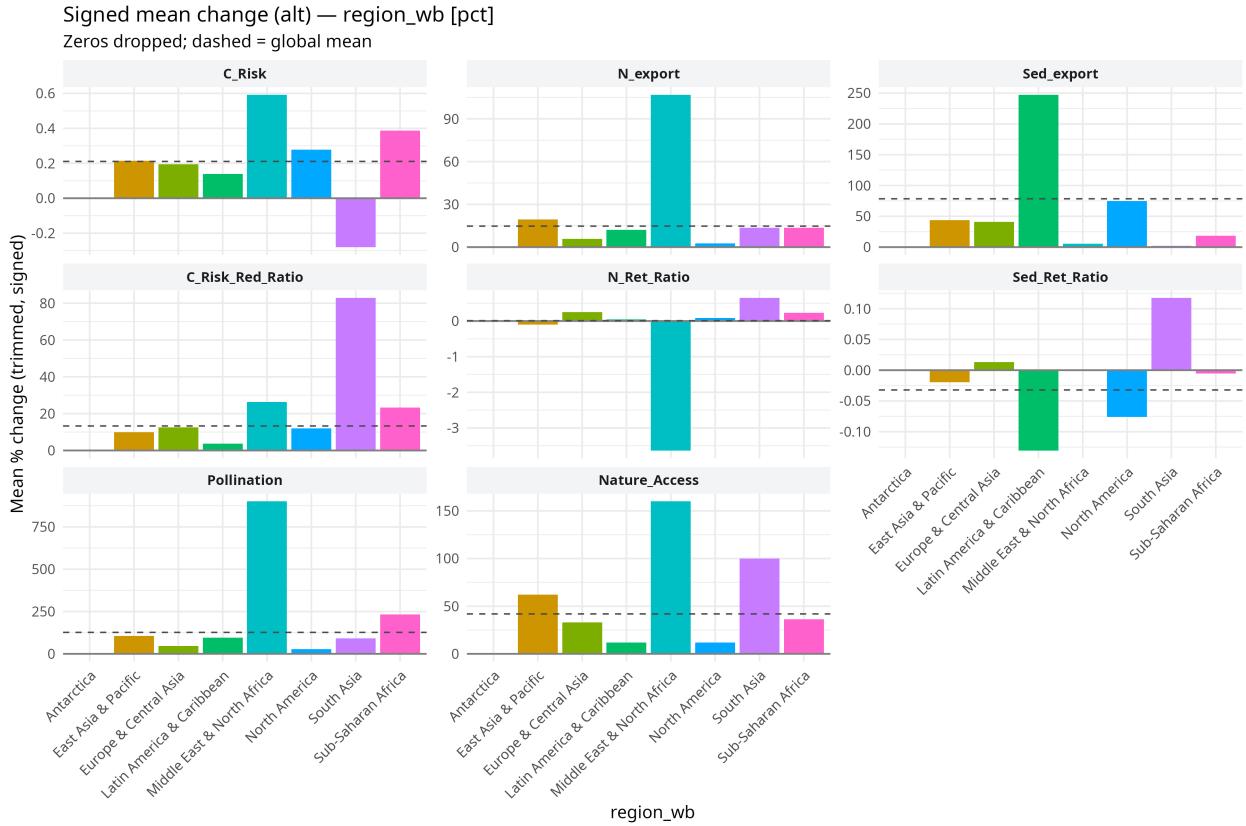


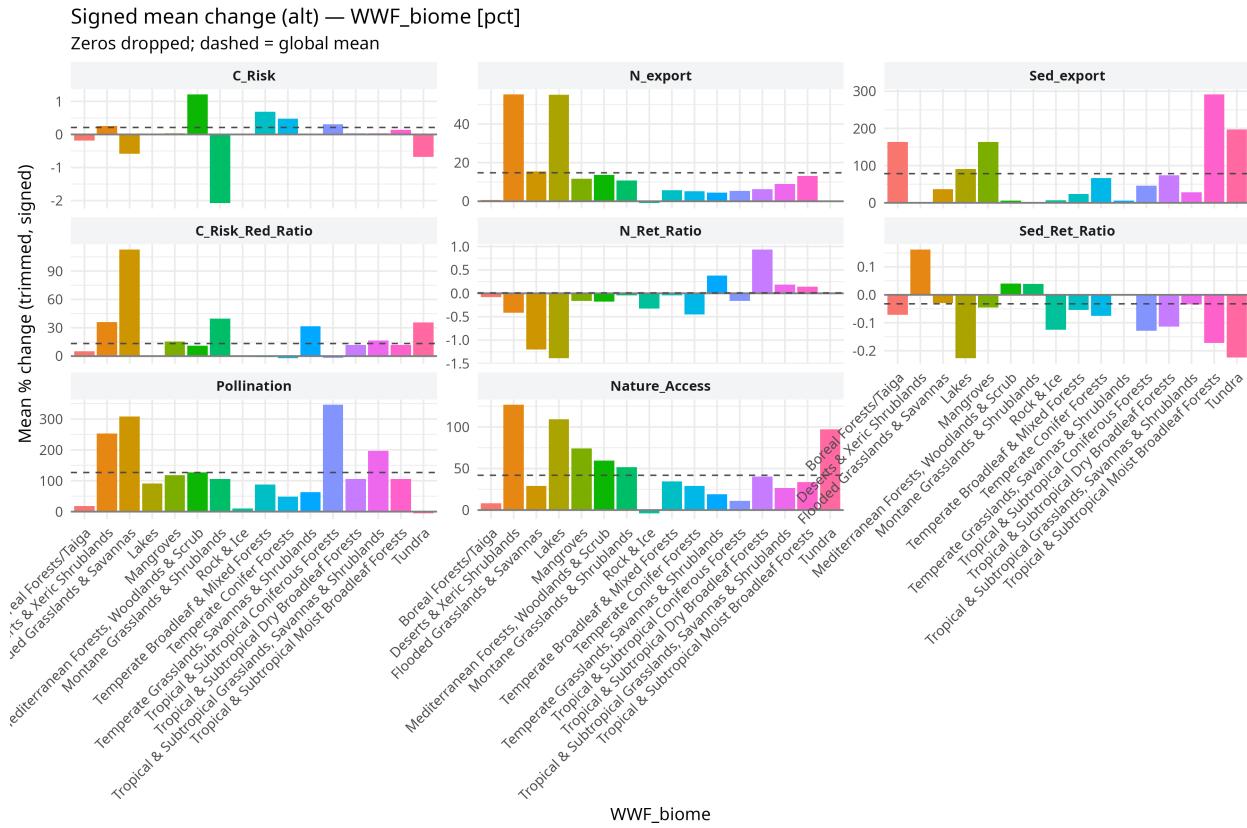










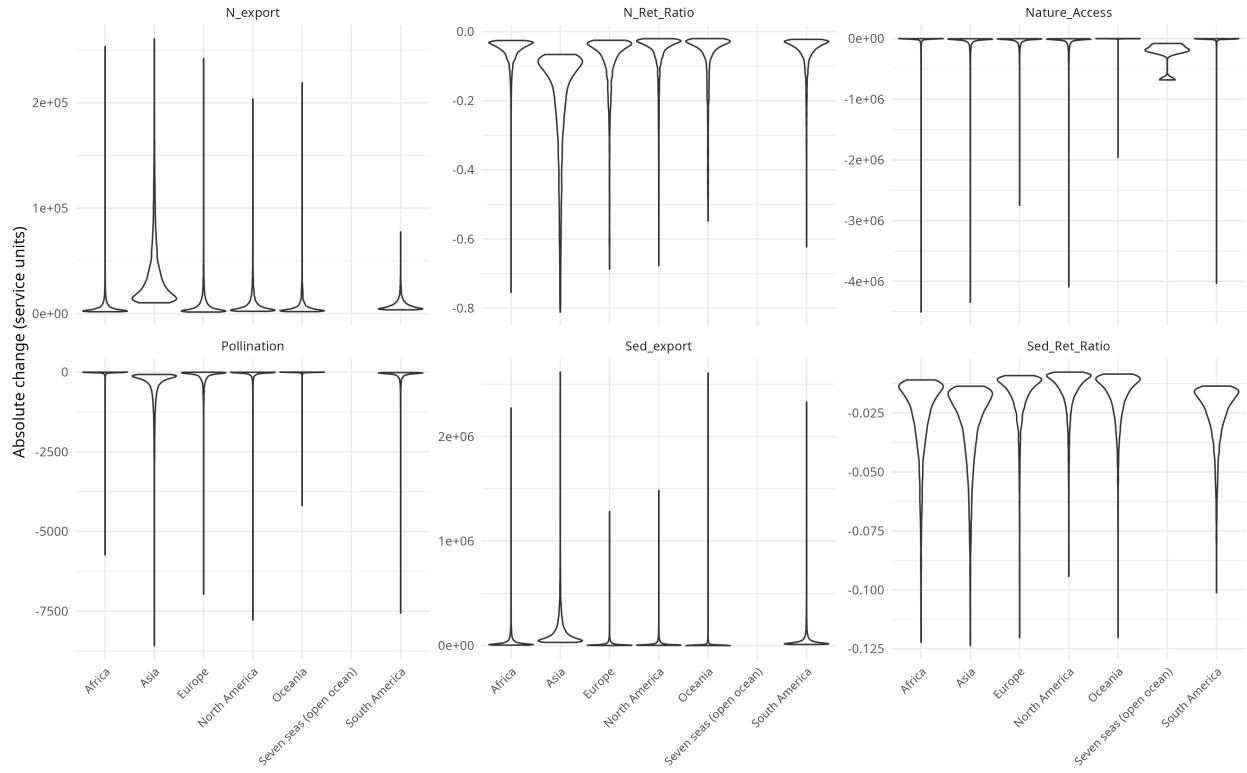


## 9 Hotspot violin plots

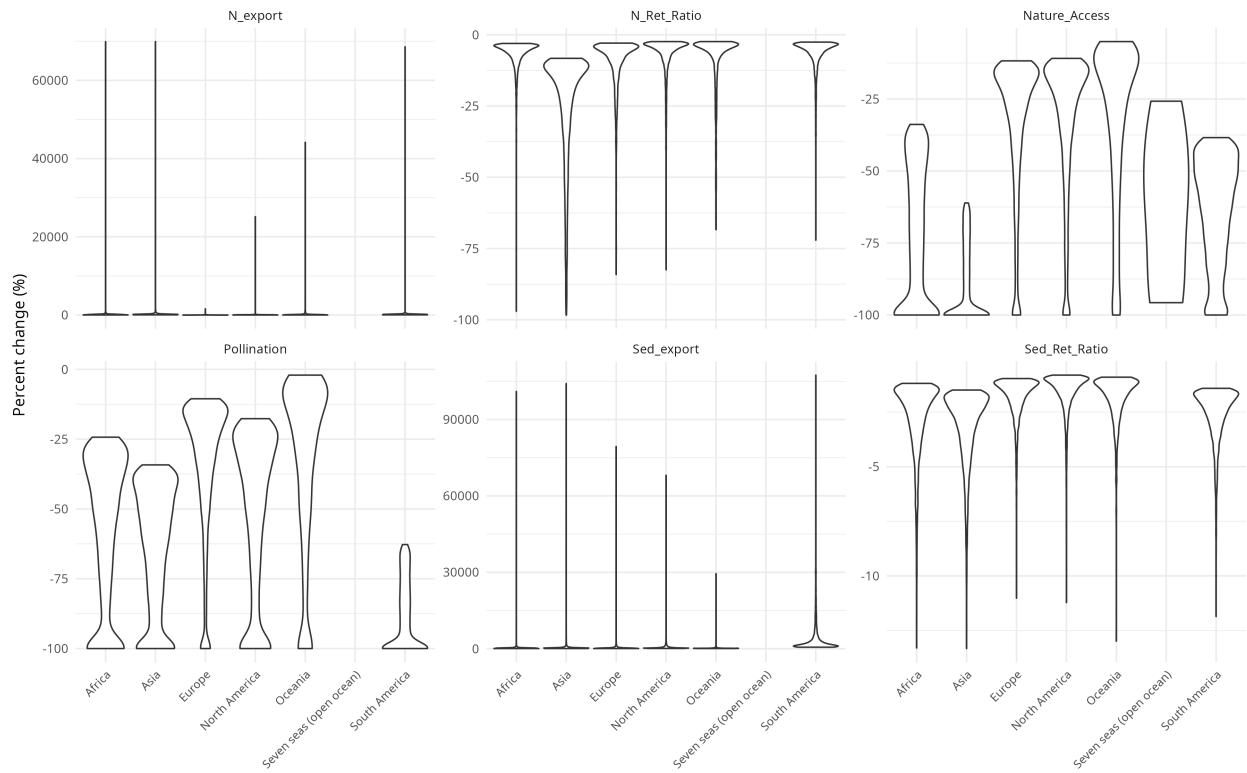
Summarize hotspot distributions by group using saved PNGs; skip computation if group columns are absent. Helper `run_hotspot_violins_by()` now lives in R/hotspot\_violins.R, so we just call it from this report.

Outputs: PNGs are written to `outputs/plots/{abs|pct}/{group_col}/violins_*.png` and mirrored into `outputs/plots/latest/violins/` for embedding.

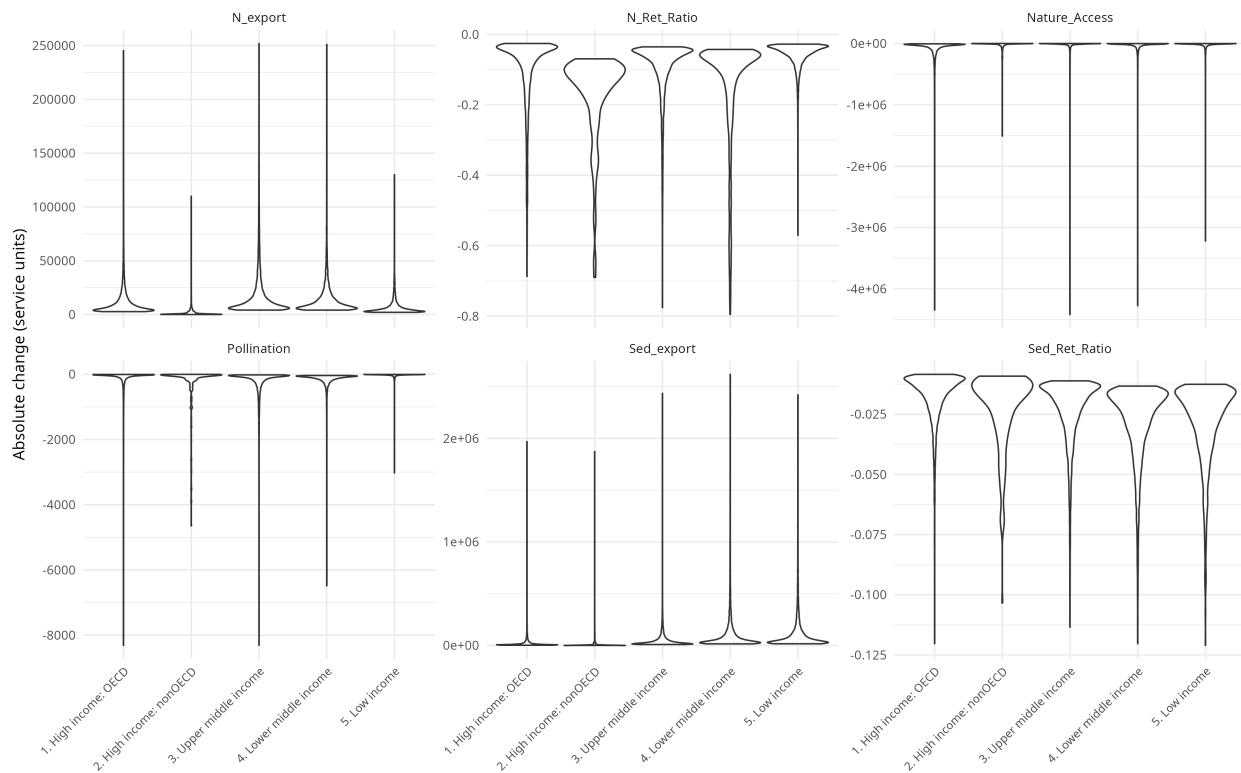
Absolute change in hotspots by continent  
Violins only · NA/0 removed · per-service 99.9% trim



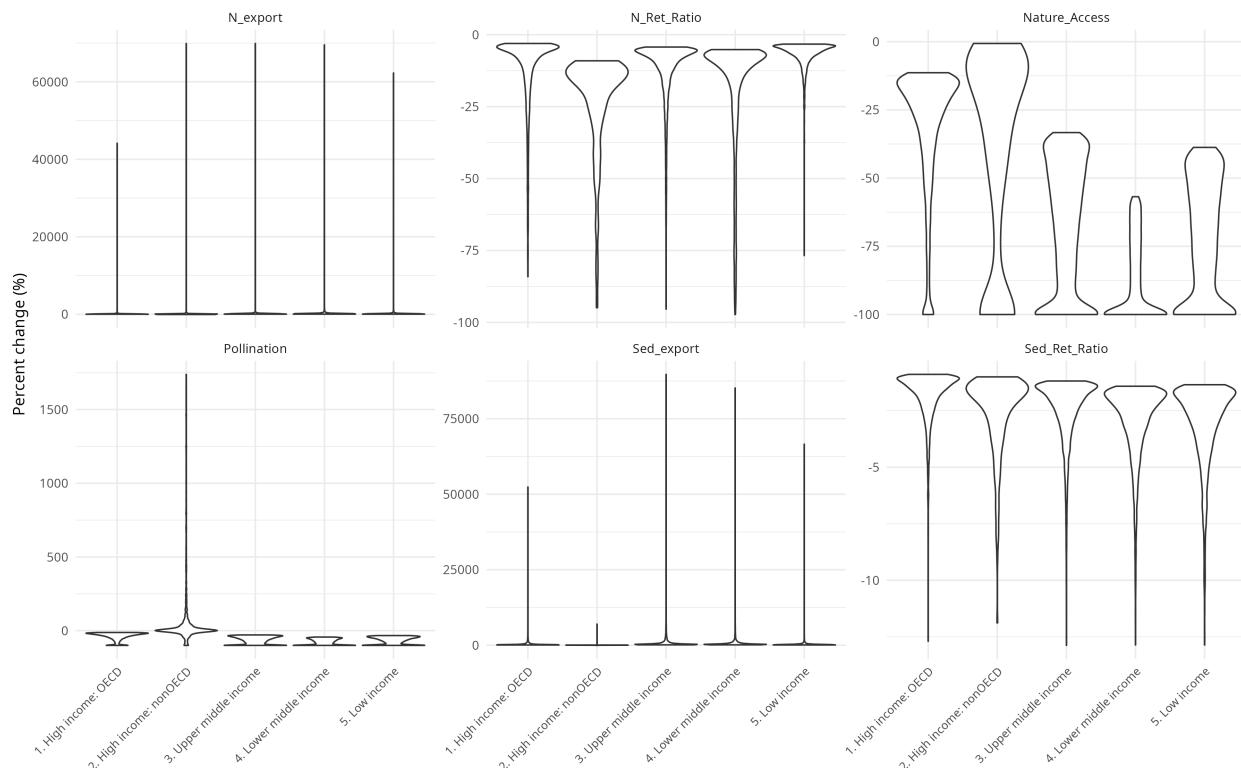
Percent change in hotspots by continent  
Violins only · NA/0 removed · per-service 99.9% trim



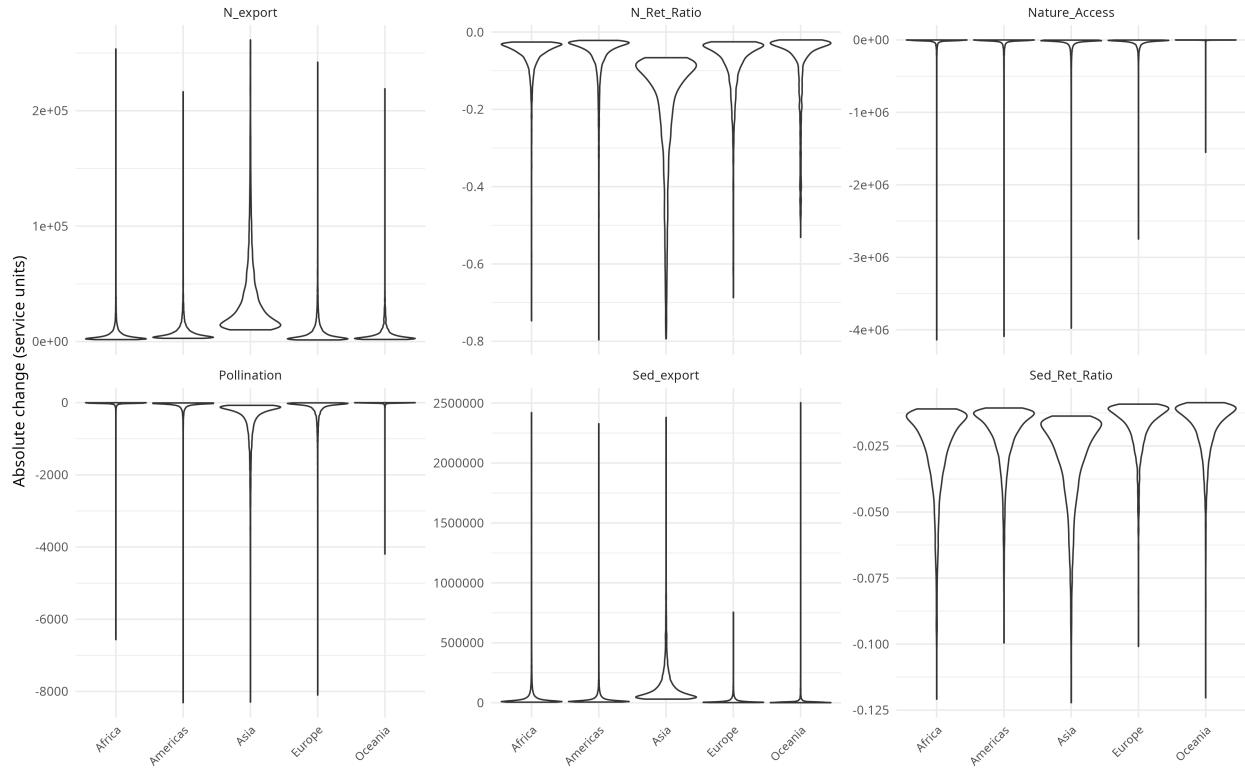
Absolute change in hotspots by income\_grp  
Violins only · NA/0 removed · per-service 99.9% trim



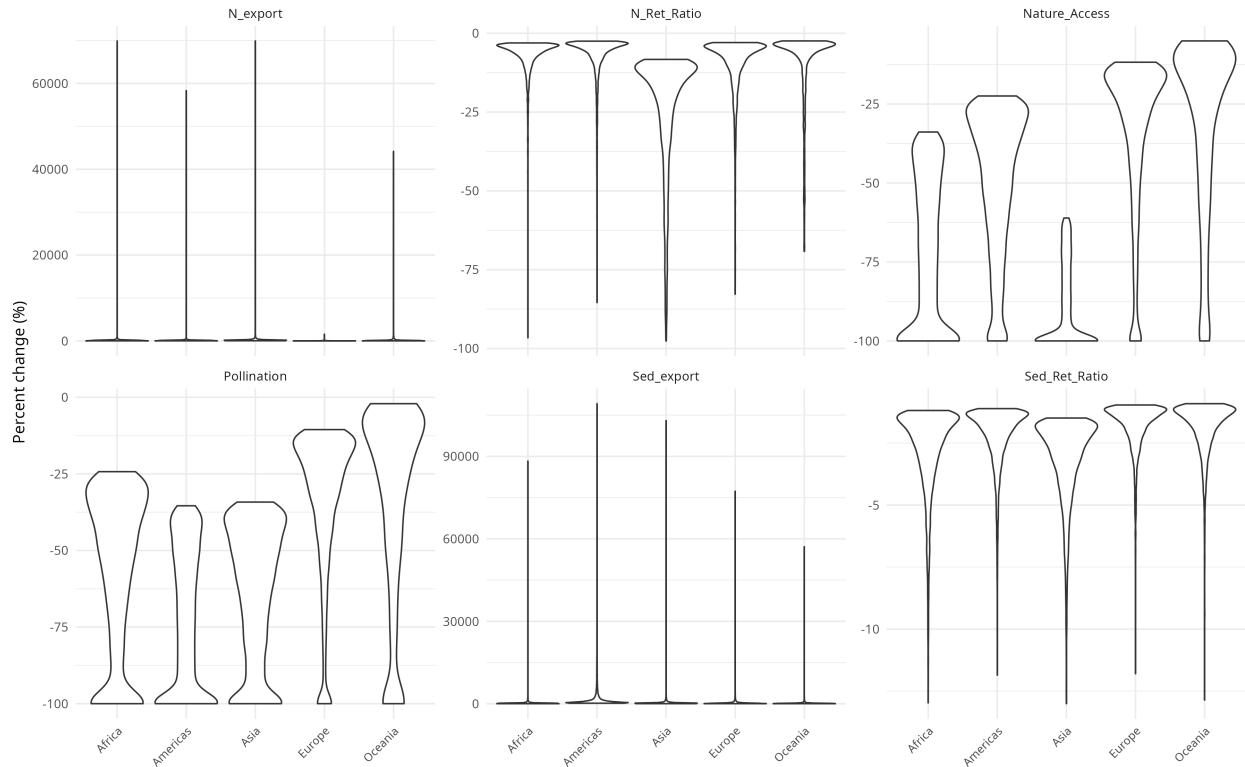
Percent change in hotspots by income\_grp  
Violins only · NA/0 removed · per-service 99.9% trim



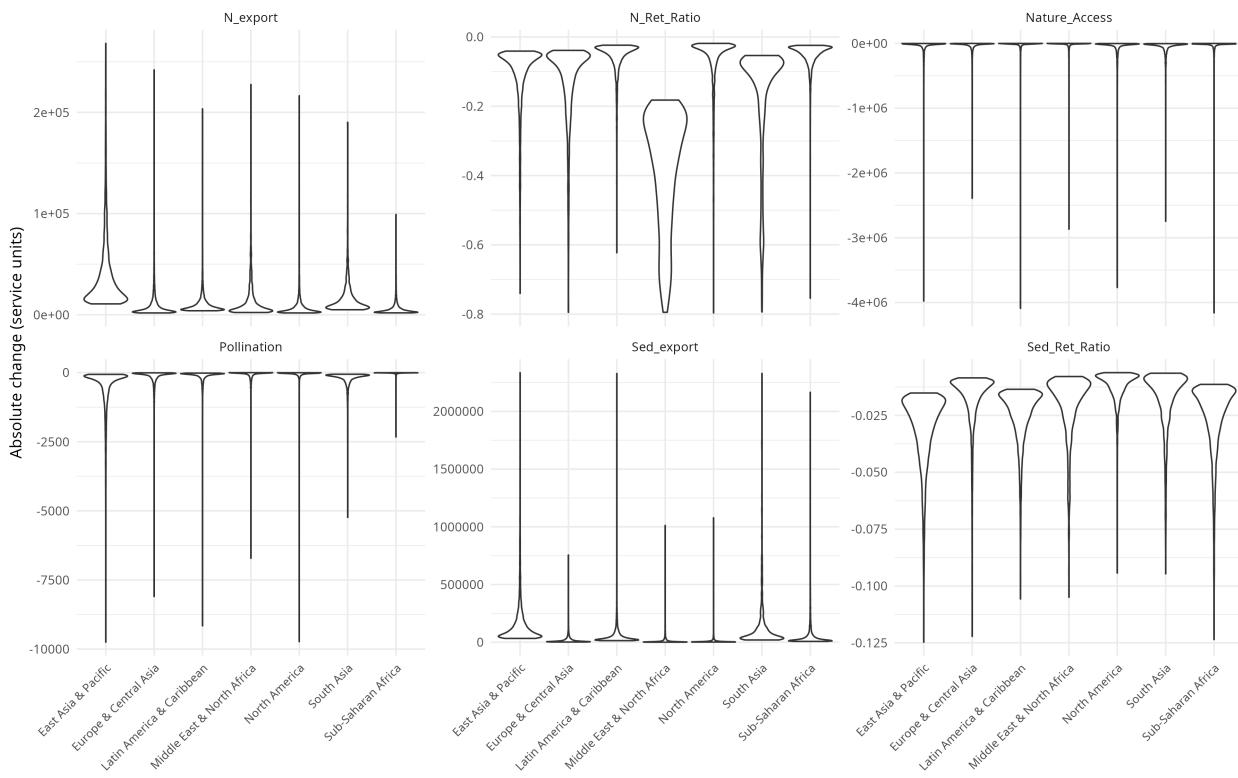
Absolute change in hotspots by region\_un  
Violins only · NA/0 removed · per-service 99.9% trim



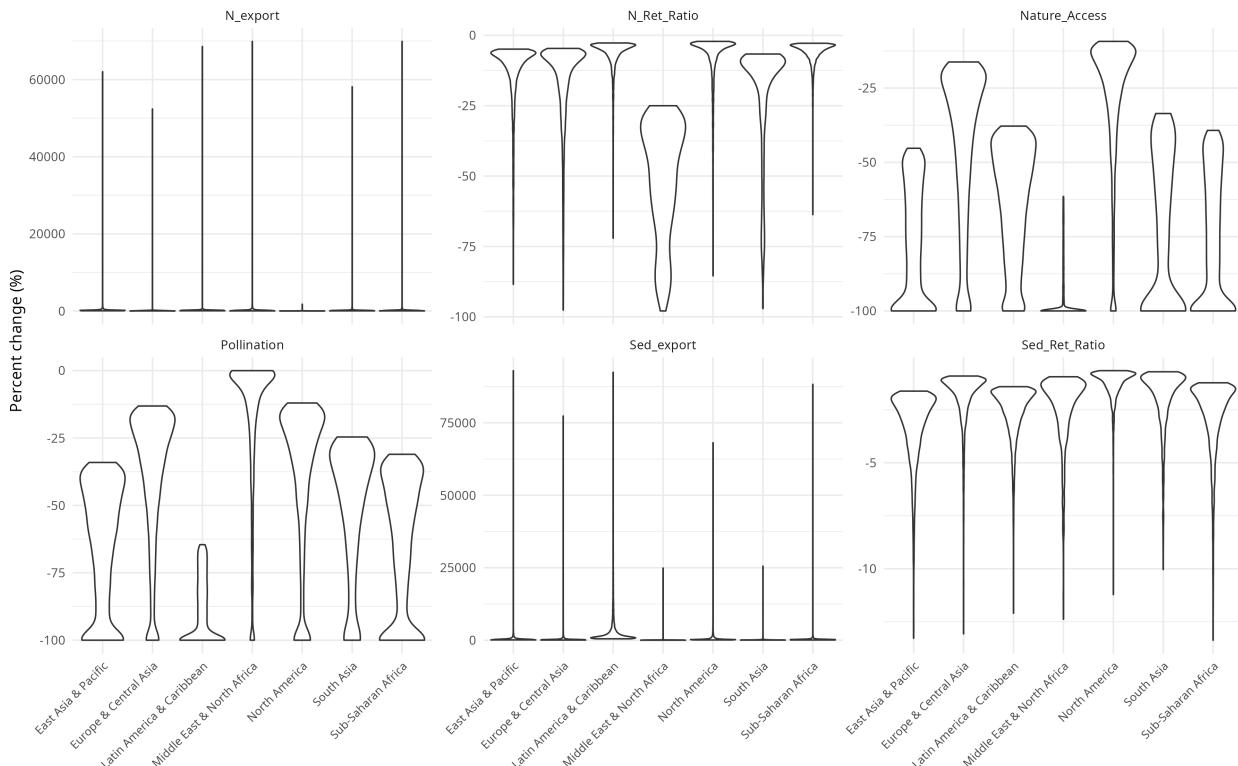
Percent change in hotspots by region\_un  
Violins only · NA/0 removed · per-service 99.9% trim



Absolute change in hotspots by region\_wb  
Violins only · NA/0 removed · per-service 99.9% trim

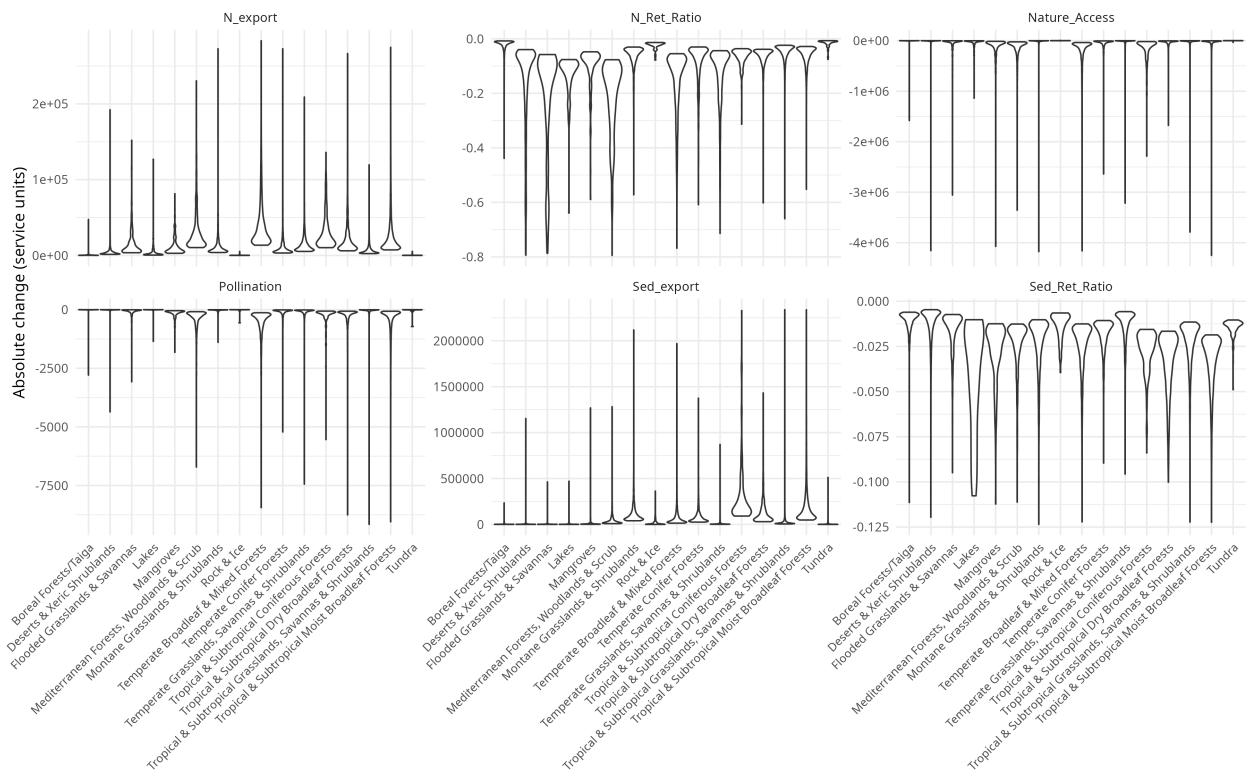


Percent change in hotspots by region\_wb  
Violins only · NA/0 removed · per-service 99.9% trim



### Absolute change in hotspots by WWF\_biome

Violins only · NA/0 removed · per-service 99.9% trim



### Percent change in hotspots by WWF\_biome

Violins only · NA/0 removed · per-service 99.9% trim

