OTNuEC_data_and_calibration

November 25, 2025

1 0) Environment

1.1 0.0) Packages

[49]:
```
options(warn = -1)

knitr::opts_chunk$set(
  warning = FALSE,
  message = FALSE
)

quiet <- function(x) suppressWarnings(suppressMessages(x))

suppressPackageStartupMessages({
  library(sf); library(terra); library(dplyr); library(purrr);library(sidrar);
 ↪library(geobr); library(ggplot2); library(osrm)
})
```

1.2 0.1) Internal data paths

[50]:
```
DATA <- list(
  viirs_blackmarble_2016 = "data_viirs/viirs_blackmarble_br_2016.tif",
  etopo1_elevation       = "ETOPO1_Ice_g_geotiff.tif",
  erosao_2020_ibge       = "Brasil_vulnerabilidade_2020/
 ↪BR_vulnerabilidade_solos_erosao_2020.tif",

  solos_wrb = c(
    "soils_brazil_wrb_wgs84.shp",
    "soils_brazil_wrb_wgs84.dbf",
    "soils_brazil_wrb_wgs84.shx",
    "soils_brazil_wrb_wgs84.prj",
    "soils_brazil_wrb_wgs84.cst"
  ),

  mapbiomas_10m_2023 = "data/mapbiomas_near_1000m.
 ↪tif"#"mapbiomas_10m_collection2_integration_v1-classification_2023.tif"
)
```

```r
# BC250
bc250_layers <- c(
  "rod_trecho_rodoviario_l",
  "fer_trecho_ferroviario_l",
  "hdv_trecho_hidroviario_l",
  "hdv_complexo_portuario_p",
  "hdv_eclusa_p",
  "tra_ponte_l",
  "tra_tunel_l",
  "tra_travessia_l",
  "enc_trecho_energia_l",
  "enc_hidreletrica_p",
  "enc_termeletrica_p",
  "enc_subest_transm_distrib_energia_eletrica_p"
)

bc250_ext <- c(".shp", ".dbf", ".shx", ".prj")

DATA$bc250 <- as.vector(outer(
  file.path("bc250 completo", bc250_layers),
  bc250_ext,
  paste0
))
```

1.3 0.2) Check datasets

[51]:
```r
check_dataset <- function(label, paths) {
  ok <- file.exists(paths)
  if (all(ok)) message("[OK] ", label)
  else {
    message("[FALTA] ", label)
    walk(paths[!ok], ~ message("   - ", .x))
  }
}

walk(names(DATA), DATA, check_dataset)
```

1.4 0.3) Sidrar API data paths

[52]:
```r
# PIB municipal (mil R$ correntes) e POP (pessoas)
pib_sidra <- sidrar::get_sidra(5938, variable = 37, period = "last",
geo = "City", format = 3) |>
transmute(
cod_mun = as.integer(`Município (Código)`),
pib_mil_rs = suppressWarnings(as.numeric(Valor))
)
```

```
pop_est <- sidrar::get_sidra(6579, variable = 9324, period = "2025",
geo = "City", format = 3) |>
transmute(
cod_mun = as.integer(`Município (Código)`),
populacao = suppressWarnings(as.numeric(Valor))
)
```

```
Considering all categories once 'classific' was set to 'all' (default)

Considering all categories once 'classific' was set to 'all' (default)
```

1.5 0.4) Geobr data paths

[53]:
```
# Brasil continental (aprox.)
clip_mainland_ll <- function(x) {
  sf::st_crop(
    x,
    xmin = -75,  # oeste
    xmax = -34,  # leste (corta Fernando de Noronha, ~ -32.4)
    ymin = -35,  # sul
    ymax =   6   # norte
  )
}
micro_br <- geobr::read_micro_region(
  year = 2013,
  simplified = TRUE
) |>
  clip_mainland_ll()   # recorte Brasil continental


#micro_br <- geobr::read_micro_region(
#  year = 2013,
#  simplified = TRUE
#)

ggplot() +
  geom_sf(
    data = micro_br,
    fill = NA,
    color = "grey40",
    linewidth = 0.2
  ) +
  coord_sf(expand = FALSE) +
  theme_void() +
  labs(title = "Microrregiões do Brasil (IBGE / geobr)")
```

```
Using year/date 2013
```

[54]:
```
str(micro_br)
summary(micro_br)
```

```
Classes 'sf' and 'data.frame':  557 obs. of  6 variables:
 $ code_state  : num  11 11 11 11 11 11 11 11 12 12 …
 $ abbrev_state: chr  "RO" "RO" "RO" "RO" …
 $ name_state  : chr  "Rondônia" "Rondônia" "Rondônia" "Rondônia" …
 $ code_micro  : num  11005 11003 11006 11008 11002 …
 $ name_micro  : chr  "Alvorada D'oeste" "Ariquemes" "Cacoal" "Colorado Do
Oeste" …
 $ geom        :sfc_GEOMETRY of length 557; first list element: List of 1
  ..$ : num [1:1168, 1:2] -63.6 -63.6 -63.6 -63.6 -63.6 …
  ..- attr(*, "class")= chr [1:3] "XY" "POLYGON" "sfg"
 - attr(*, "sf_column")= chr "geom"
 - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA
  ..- attr(*, "names")= chr [1:5] "code_state" "abbrev_state" "name_state"
"code_micro" …
```

```
   code_state    abbrev_state        name_state          code_micro
 Min.   :11.00   Length:557         Length:557         Min.   :11001
 1st Qu.:24.00   Class :character   Class :character   1st Qu.:24007
 Median :31.00   Mode  :character   Mode  :character   Median :31028
 Mean   :31.31                                         Mean   :31329
 3rd Qu.:41.00                                         3rd Qu.:41007
 Max.   :53.00                                         Max.   :53001
  name_micro                    geom
 Length:557         MULTIPOLYGON : 26
 Class :character   POLYGON      :531
 Mode  :character   epsg:4674    :  0
```

```
                    +proj=long…:  0
```

2 1) Nodes Positions

[55]:
```r
# ------------------------------------------------------------
# 1) Ler raster VIIRS
# ------------------------------------------------------------
ntl <- terra::rast(DATA$viirs_blackmarble_2016)
names(ntl) <- "value"

# ------------------------------------------------------------
# 2) Função minimalista: ponto mais iluminado por microrregião
# ------------------------------------------------------------
build_nodes_simple <- function(regions_sf, ntl_rast) {

  # garantir mesmo CRS
  regions_ll <- st_transform(regions_sf, st_crs(ntl_rast))

  # extrair valores do raster dentro de cada polígono
  vals <- terra::extract(ntl_rast, terra::vect(regions_ll), cells = TRUE)

  best <- vals |>
    dplyr::group_by(ID) |>
    dplyr::filter(!is.na(value)) |>
    dplyr::slice_max(order_by = value, n = 1, with_ties = FALSE) |>
    dplyr::ungroup()

  xy <- terra::xyFromCell(ntl_rast, best$cell)

  sf::st_as_sf(
    data.frame(
      id = regions_ll$code_micro[best$ID],
      lon = xy[,1],
      lat = xy[,2]
    ),
    coords = c("lon", "lat"),
    crs = st_crs(ntl_rast)
  )
}

# ------------------------------------------------------------
# 3) Construir os nós
# ------------------------------------------------------------
nodes_micro <- build_nodes_simple(micro_br, ntl)
```

```r
# ------------------------------------------------------------
# 4) Plot leve
# ------------------------------------------------------------
ggplot() +
  geom_sf(data = micro_br, fill = NA, color = "grey70", linewidth = 0.2) +
  geom_sf(data = nodes_micro, color = "#2E7D32", size = 1) +
  theme_void() +
  labs(title = "Nós = pixel mais iluminado por microrregião")
```

3 2) Contiguity

[56]:
```r
build_edges_contiguity <- function(regions_sf, nodes_sf,
                                   crs_metric = 5880,
                                   snap_size = 5,
                                   buffer_dist = 5) {

  # detectar automaticamente a coluna de ID
  id_col_regions <- "code_micro"
  id_col_nodes   <- "id"

  # transformar para métrico
  reg <- regions_sf |>
    st_transform(crs_metric) |>
    st_set_precision(1e6) |>
    lwgeom::st_snap_to_grid(size = snap_size) |>
    st_buffer(0) |>
    st_make_valid()

  # ordenar nós pela ordem das regiões
```

```
  nodes_m <- nodes_sf |>
    st_transform(crs_metric) |>
    arrange(match(.data[[id_col_nodes]], reg[[id_col_regions]]))

  stopifnot(nrow(nodes_m) == nrow(reg))

  coords <- st_coordinates(nodes_m)

  buf <- st_buffer(reg, buffer_dist)

  edge_pairs <- purrr::imap_dfr(
    st_intersects(buf, buf),
    ~ tibble::tibble(i = .y, j = .x)
  ) |>
    filter(i != j) |>
    mutate(
      a = pmin(i, j),
      b = pmax(i, j)
    ) |>
    distinct(a, b) |>
    transmute(i = a, j = b)

  edge_lines <- purrr::map2(
    edge_pairs$i,
    edge_pairs$j,
    ~ st_linestring(rbind(coords[.x, ], coords[.y, ]))
  )

  st_sf(
    tibble::tibble(
      id_i = reg[[id_col_regions]][edge_pairs$i],
      id_j = reg[[id_col_regions]][edge_pairs$j],
      geometry = st_sfc(edge_lines, crs = crs_metric)
    )
  )
}
```

[57]:
```
edges_micro <- build_edges_contiguity(micro_br, nodes_micro)

ggplot() +
  geom_sf(data = micro_br, fill = NA, color = "grey80", linewidth = 0.2) +
  geom_sf(data = edges_micro, color = "#1E88E5", linewidth = 0.15) +
  geom_sf(data = nodes_micro, color = "#2E7D32", size = 1) +
  theme_void()
```

4 3) Contiguity with PA

4.1 3.1) Simple PA

[58]:
```
read_protected_areas_simple <- function(crs_target = 5880) {

  uc <- geobr::read_conservation_units() |>
    st_make_valid() |>
    st_transform(crs_target)

  ti <- geobr::read_indigenous_land() |>
    st_make_valid() |>
    st_transform(crs_target)

  bind_rows(
    uc |> mutate(src = "UC"),
    ti |> mutate(src = "TI")
  ) |>
    st_make_valid()
}


filter_edges_by_pa <- function(edges_sf, protected_sf, thr = 0.95) {

  # criar um id explícito para cada aresta
  edges_sf <- edges_sf |>
    dplyr::mutate(edge_id = dplyr::row_number())

  inter <- suppressWarnings(sf::st_intersection(edges_sf, protected_sf))
```

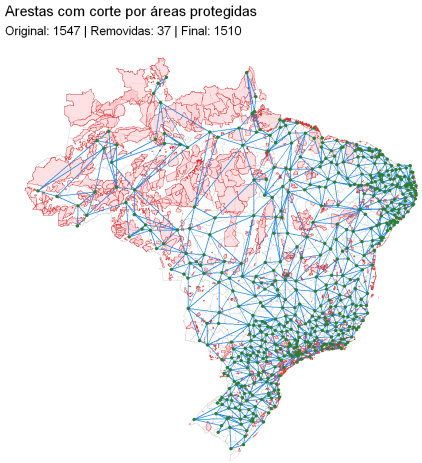
```r
  total_len <- as.numeric(sf::st_length(edges_sf))

  protected_len <- numeric(nrow(edges_sf))

  if (nrow(inter) > 0) {
    # mesmo id das arestas originais
    id    <- inter$edge_id
    len   <- as.numeric(sf::st_length(inter))

    # soma do comprimento protegido por aresta
    agg <- tapply(len, id, sum)

    protected_len[as.integer(names(agg))] <- agg
  }

  ratio <- protected_len / total_len
  keep  <- ratio < thr

  list(
    edges_clean = dplyr::select(edges_sf[keep, ], -edge_id),
    removed_idx = !keep,
    n_removed   = sum(!keep),
    n_original  = nrow(edges_sf)
  )
}




plot_edges_pa_filter <- function(regions_sf, nodes_sf,
                                 edges_before, edges_after,
                                 prot_sf = NULL,
                                 title = "Arestas com corte por áreas␣
 ↪protegidas") {

  p <- ggplot() +
    geom_sf(data = regions_sf,
            fill = NA, color = "grey80", linewidth = 0.2)

  if (!is.null(prot_sf)) {
    p <- p +
      geom_sf(data = prot_sf,
              fill = "#FFCDD2", color = "#D32F2F",
              alpha = 0.6, linewidth = 0.2)
  }
```

```
  p +
    geom_sf(data = edges_before, color = "#9E9E9E",
            linewidth = 0.25, alpha = .4) +
    geom_sf(data = edges_after,  color = "#1E88E5",
            linewidth = 0.35) +
    geom_sf(data = nodes_sf,     color = "#2E7D32",
            size = 1) +
    theme_void() +
    labs(
      title = title,
      subtitle = paste0(
        "Original: ", nrow(edges_before),
        " | Removidas: ", nrow(edges_before) - nrow(edges_after),
        " | Final: ", nrow(edges_after)
      )
    )
}
```

[59]:
```
# 1) Ler PAs
prot <- read_protected_areas_simple(crs_target = st_crs(edges_micro))

mask_reg <- micro_br |>
  sf::st_union() |>
  sf::st_transform(sf::st_crs(prot))

prot <- suppressWarnings(
  sf::st_intersection(prot, mask_reg)
)

# 2) Filtrar arestas
pa_clean <- filter_edges_by_pa(edges_sf = edges_micro,
                               protected_sf = prot,
                               thr = 0.75)   # thr representa  a porcentagem da␣
 ↪fronteira que está em PA


edges_clean <- pa_clean$edges_clean

cat("Original:", pa_clean$n_original,
    "| Removidas:", pa_clean$n_removed,
    "| Final:", nrow(edges_clean), "\n")

# 3) Plot
plot_edges_pa_filter(
  regions_sf   = micro_br,
  nodes_sf     = nodes_micro,
  edges_before = edges_micro,
```

```
  edges_after  = edges_clean,
  prot_sf      = prot
)
```

```
Using year/date 201909

Using year/date 201907


Original: 1547 | Removidas: 37 | Final: 1510
```



4.2 3.2) PA including forests

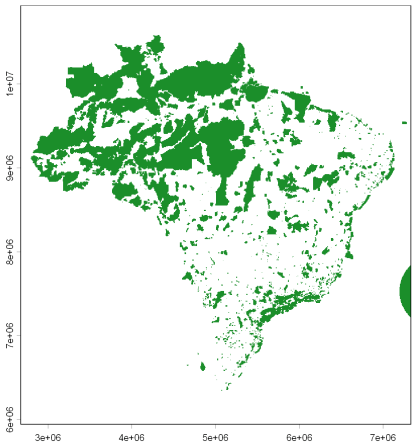
[60]:
```
library(terra); library(sf); library(dplyr)

crs_target    <- st_crs(edges_micro)
forest_codes  <- c(3L, 4L, 5L)
buf_m         <- 3200   # buffer em metros
min_patch_km2 <- 10     # tamanho mínimo da mancha de floresta (ajuste)

# 1) Raster binário
r <- rast("data/mapbiomas_near_1000m.tif")
r[r == 0] <- NA
r_bin <- classify(r, rbind(
  cbind(forest_codes, forest_codes, 1),
  c(-Inf, min(forest_codes) - 0.001, 0),
  c(max(forest_codes) + 0.001, Inf, 0)
))

# 2) Projeta para métrico
tmp <- tempfile(fileext = ".tif")
```

```
r_bin_m <- project(r_bin, crs_target$wkt, method = "near",
                   filename = tmp, overwrite = TRUE)

# 3) Remove manchas pequenas
cell_area_km2 <- prod(res(r_bin_m)) / 1e6
min_cells     <- ceiling(min_patch_km2 / cell_area_km2)

cl   <- patches(ifel(r_bin_m == 1, 1, NA), directions = 8, zeroAsNA = TRUE)
ftbl <- as.data.frame(freq(cl, digits = 0)) |> filter(!is.na(value))

big_ids <- ftbl$value[ftbl$count >= min_cells]

if (length(big_ids) == 0) {
message("Nenhum patch >= min_patch_km2; mantendo floresta original.")
r_forest_clean <- ifel(r_bin_m == 1, 1, NA)
} else {
r_forest_clean <- ifel(cl %in% big_ids, 1, NA)
}


# 4) Buffer só da floresta (não considera 0 como "dentro")
dist_forest  <- distance(r_forest_clean)
r_forest_buf <- ifel(dist_forest <= buf_m, 1, NA)

# 5) PAs no mesmo buffer e na mesma grade
pa <- read_protected_areas_simple(crs_target = crs_target) |>
st_make_valid() |>
st_buffer(buf_m)

if (nrow(pa) == 0) {
pa_r <- setValues(r_forest_buf, NA)
} else {
pa_r <- rasterize(vect(pa), r_forest_buf,
              field = 1, background = NA, touches = TRUE)
}

# garante numérico
pa_r <- as.numeric(pa_r)

# 6) União PA + floresta buffered (cover = pega o 1 do primeiro não-NA)
# máscara Brasil
mask_reg <- micro_br |>
  st_union() |>
  st_transform(crs_target) |>
  st_make_valid()
mask_v <- vect(mask_reg)
```

```r
# recorta raster combinado
pa_forest_r  <- cover(pa_r, r_forest_buf)
pa_forest_sf <- as.polygons(pa_forest_r, dissolve = TRUE) |>
  st_as_sf() |>
  st_transform(crs_target) |>
  mutate(src = "PA2")

saveRDS(pa_forest_sf, "data/pa2_pa_floresta.rds")
```

```
Nenhum patch >= min_patch_km2; mantendo floresta original.

Using year/date 201909

Using year/date 201907
```

[61]:
```r
plot(pa_forest_r, col = c(NA, "#1B8E2A"), legend = FALSE)
```



[62]:
```r
pa_forest_clean <- filter_edges_by_pa(
  edges_sf     = edges_micro,
  protected_sf = pa_forest_sf,
  thr          = 0.75  # ajuste se quiser mais/menos estrito
)
edges_clean <- pa_forest_clean$edges_clean
cat("Original:", pa_forest_clean$n_original,
    "| Removidas:", pa_forest_clean$n_removed,
    "| Final:", nrow(edges_clean), "\n")
```

```
Original: 1547 | Removidas: 106 | Final: 1441
```

[63]:
```
plot_edges_pa_filter(
  regions_sf    = micro_br,
  nodes_sf      = nodes_micro,
  edges_before = edges_micro,
  edges_after   = edges_clean,
  prot_sf       = pa_forest_sf
)
```

5 4) Creating feasible edges

[64]:
```
# file: R/edge_stat_from_roads_dijkstra_fix.R
suppressPackageStartupMessages({
  library(sf)
  library(dplyr)
  library(lwgeom)
  library(sfnetworks)
  library(tidygraph)
  library(tibble)
})

# --- 1) Sanitizador: deixa só LINESTRING válida e mensurável ---
clean_lines_for_network <- function(x) {
  x |>
    sf::st_zm(drop = TRUE, what = "ZM") |>              # tirar Z/M (por quê: sfnetwork usa XY)
    sf::st_make_valid() |>                              # corrigir geometrias inválidas
    sf::st_collection_extract("LINESTRING", warn = FALSE) |>  # remover não-linhas
```

```
    sf::st_cast("LINESTRING", warn = FALSE) |>          # explodir MULTI em␣
 ↪simples
    dplyr::filter(!sf::st_is_empty(geometry)) |>
    dplyr::mutate(.len = suppressWarnings(as.numeric(sf::st_length(geometry))))␣
 ↪|>
    dplyr::filter(is.finite(.len) & .len > 0) |>
    dplyr::select(-.len)
}

# --- 2) Endpoints seguros para LINESTRING ---
.edge_endpoints <- function(lines_sf) {
  lines_ok <- lines_sf |>
    sf::st_cast("LINESTRING", warn = FALSE) |>
    dplyr::filter(!sf::st_is_empty(geometry))
  starts <- lwgeom::st_startpoint(lines_ok$geometry)
  ends   <- lwgeom::st_endpoint(lines_ok$geometry)
  tibble(edge_id = seq_len(nrow(lines_ok)),
         from_geom = starts,
         to_geom   = ends) |>
    sf::st_as_sf(crs = sf::st_crs(lines_ok))
}
```

[65]:
```
read_bc250_roads_from_DATA <- function(data_bc250,
                                       crs_target) {

  shp_paths <- data_bc250[grepl("\\.shp$", data_bc250)]

  rod_path <- shp_paths[basename(shp_paths) == "rod_trecho_rodoviario_l.shp"]
  if (!length(rod_path))
    stop("Shapefile 'rod_trecho_rodoviario_l.shp' não encontrado em DATA$bc250")

  terra::vect(rod_path[1]) |>
    sf::st_as_sf() |>
    sf::st_make_valid() |>
    sf::st_transform(crs_target)
}
roads_bc250 <- read_bc250_roads_from_DATA(
  DATA$bc250,
  crs_target = st_crs(edges_clean)
)

roads_bc250 <- roads_bc250 |>
  mutate(
    # 1) Pavimentação (você já tem algo assim)
    pav_pav = if_else(
      grepl("Sem revestimento", revestimen, ignore.case = TRUE),
      0, 1
```

```
    ),

    # 2) Nº de faixas (já é numérico: nrfaixas)

    # 3) Nº de pistas (simples x dupla)
    pistas_duplas = if_else(nrpistas >= 2, 1, 0),

    # 4) Concessão
    conc_via = if_else(concession == "Sim", 1, 0),

    # 5) Jurisdição "alta hierarquia"
    jur_fed_est = case_when(
      jurisdicao %in% c("Federal", "Estadual/Distrital") ~ 1,
      jurisdicao == "Municipal"                          ~ 0,
      TRUE                                               ~ NA_real_
    ),

    # 6) Operacional / tráfego permanente
    oper_ativa   = if_else(operaciona == "Sim", 1, 0),
    traf_perman  = if_else(trafego == "Permanente", 1, 0)
  )
edges_stats <- edges_clean


# --- 3) DROP-IN: igual à versão Dijkstra, mas com limpeza antes do grafo ---
edge_stat_from_roads <- function(edges_sf,
                                 roads_sf,
                                 var,
                                 buf_km = 25,        # mantido por␣
 ↪compatibilidade (ignorado)
                                 crs_metric = 5880) {
  if (!var %in% names(roads_sf)) {
    stop(sprintf("Coluna '%s' inexistente em roads_sf.", var))
  }

  # Projeção
  edges_m <- edges_sf |>
    sf::st_transform(crs_metric) |>
    dplyr::mutate(edge_id = dplyr::row_number())

  roads_m_raw <- roads_sf |>
    sf::st_transform(crs_metric)

  # **PASSO CRÍTICO**: limpar geometrias de rodovias (conserta o erro do print)
  roads_m <- clean_lines_for_network(roads_m_raw)
  if (nrow(roads_m) == 0) stop("roads_sf ficou vazio após limpeza; verifique a␣
 ↪camada.")
```

```r
# Grafo
net <- sfnetworks::as_sfnetwork(roads_m, directed = FALSE)
net <- tidygraph::activate(net, "edges") |>
  dplyr::mutate(
    len_m = as.numeric(sf::st_length(geometry)),
    w     = len_m  # custo = distância; troque por tempo se desejar
  )

# Endpoints das suas arestas e snap para nós
ep <- .edge_endpoints(edges_m)
g_nodes <- sf::st_as_sf(net, "nodes")
from_id <- sf::st_nearest_feature(ep$from_geom, g_nodes)
to_id   <- sf::st_nearest_feature(ep$to_geom,   g_nodes)

# Saída
col_name <- paste0("stat_", var)
edges_m[[col_name]] <- NA_real_

net_edges_sf <- sf::st_as_sf(net, "edges")
if (!var %in% names(net_edges_sf)) {
  stop(sprintf("Coluna '%s' não foi preservada na rede (cheque o␣
↪pré-processamento).", var))
}

# Dijkstra por aresta i→j
for (i in seq_len(nrow(edges_m))) {
  fi <- from_id[i]; ti <- to_id[i]
  if (!is.finite(fi) || !is.finite(ti) || fi == ti) next

  paths <- tryCatch(
    sfnetworks::st_network_paths(net, from = fi, to = ti, weights = "w", type␣
↪= "shortest"),
    error = function(e) NULL
  )
  if (is.null(paths) || length(paths$edge_paths) == 0 ||␣
↪length(paths$edge_paths[[1]]) == 0) next

  idx <- as.integer(paths$edge_paths[[1]])
  segs <- net_edges_sf[idx, , drop = FALSE] |>
    dplyr::mutate(
      len_km = as.numeric(sf::st_length(geometry)) / 1000,
      val    = .data[[var]]
    )

  if (!any(is.finite(segs$val))) next
  den <- sum(segs$len_km, na.rm = TRUE)
```

```
    if (!is.finite(den) || den <= 0) next

    edges_m[[col_name]][i] <- sum(segs$len_km * segs$val, na.rm = TRUE) / den
  }

  sf::st_transform(edges_m, sf::st_crs(edges_sf))
}



for (v in c("pav_pav", "nrfaixas", "pistas_duplas",
            "conc_via", "jur_fed_est", "oper_ativa", "traf_perman")) {

  edges_stats <- edge_stat_from_roads(
    edges_sf = edges_stats,   # reutiliza o que já tem
    roads_sf = roads_bc250,
    var      = v,
    buf_km   = 25
  )
}
edges_stats |>
  st_drop_geometry() |>
  summarise(
    across(starts_with("stat_"),
           list(
             n_non_na = ~sum(!is.na(.x)),
             mean     = ~mean(.x, na.rm = TRUE),
             sd       = ~sd(.x,   na.rm = TRUE)
           ),
           .names = "{.col}_{.fn}")
  )
```

A tibble: 1 × 21

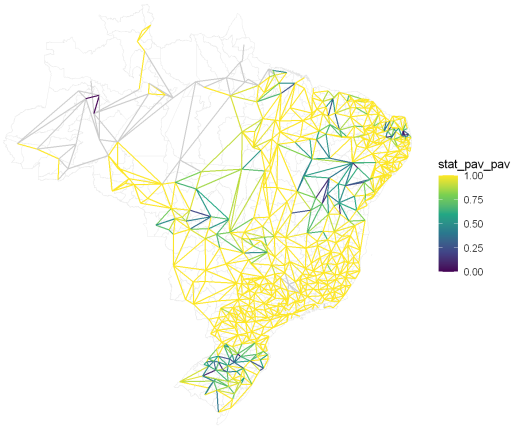
stat_pav_pav_n_non_na <int>	stat_pav_pav_mean <dbl>	stat_pav_pav_sd <dbl>	stat_nrfaixas_n_non <int>
1382	0.9254838	0.1776586	1379

[66]:
```
plot_edge_stat <- function(edges_sf,
                           regions_sf,
                           var,
                           title    = NULL,
                           subtitle = NULL,
                           limits   = NULL,
                           palette  = "viridis",
                           na.value = "grey80") {

  if (is.null(title)) {
    title <- paste("Rede -", var)
```

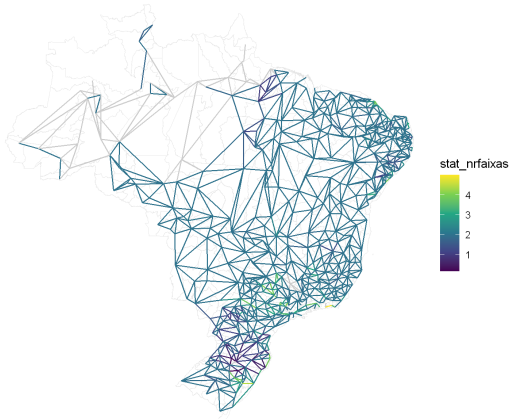
```
  }

  ggplot() +
    geom_sf(data = regions_sf,
            fill = NA, color = "grey90", linewidth = 0.2) +
    geom_sf(data = edges_sf,
            aes_string(color = var),
            linewidth = 0.6) +
    scale_color_viridis_c(
      option  = palette,
      na.value = na.value,
      limits   = limits
    ) +
    theme_void() +
    labs(
      title    = title,
      subtitle = subtitle,
      color    = var
    )
}
```

[67]:
```
vars_to_plot <- c(
  "stat_pav_pav",
  "stat_nrfaixas",
  "stat_pistas_duplas",
  "stat_conc_via",
  "stat_jur_fed_est",
  "stat_oper_ativa",
  "stat_traf_perman"
)

plots <- lapply(vars_to_plot, function(v) {
  plot_edge_stat(edges_stats, micro_br, var = v)
})

# Para ver um:
plots[]
```

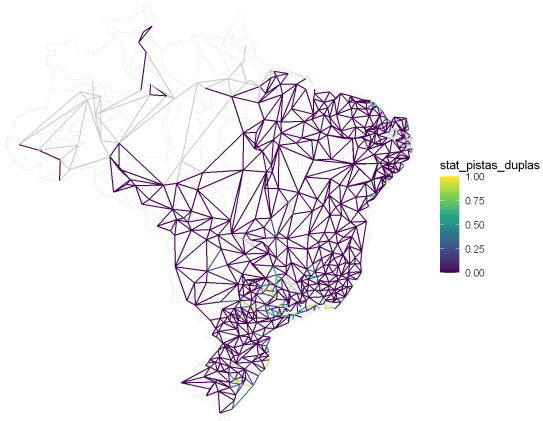
Rede - stat_pav_pav

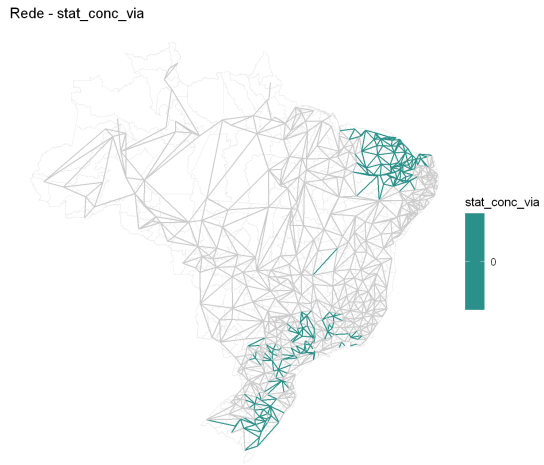


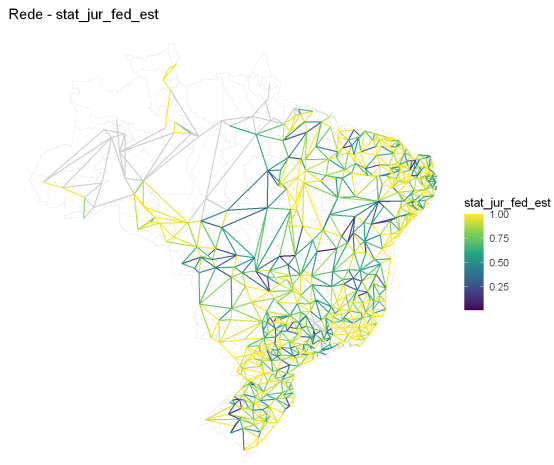
Rede - stat_nrfaixas



Rede - stat_pistas_duplas




Rede - stat_conc_via
stat_conc_via
0


Rede - stat_jur_fed_est
stat_jur_fed_est
1.00
0.75
0.50
0.25

[[1]]

[[2]]

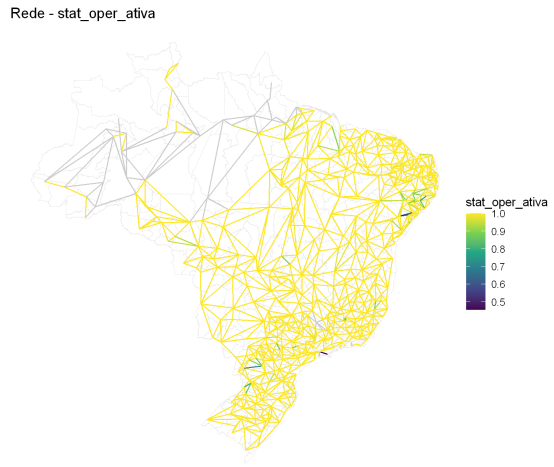
[[3]]

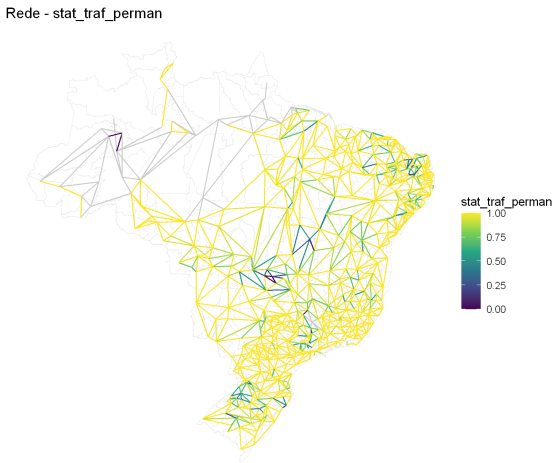
[[4]]

[[5]]

[[6]]

[[7]]





6 5) The Speed Challenge

6.1 5.1) Google API

[68]:
```
library(httr2)
library(jsonlite)
library(dplyr)

key <- "MY_KEY"

gmaps_speed_live <- function(origin, destination, api_key) {
```

```
  url <- "https://maps.googleapis.com/maps/api/directions/json"

  resp <- request(url) |>
    req_url_query(
      origin      = origin,
      destination = destination,
      mode        = "driving",
      key         = api_key
    ) |>
    req_perform() |>
    resp_body_json()

  leg <- resp$routes[[1]]$legs[[1]]

  dist_km <- leg$distance$value / 1000
  dur_h   <- leg$duration$value / 3600

  tibble(
    origin,
    destination,
    dist_km,
    dur_h,
    speed_kmh = dist_km / dur_h
  )
}
```

[69]:
```
#gmaps_speed_live("Sao Paulo", "Campinas", key)
```

6.2 5.2) Calibrating my infrastructure measure with roadspeed

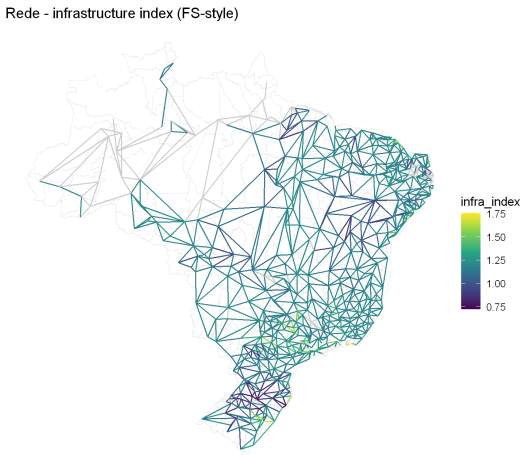
I construct an infrastructure quality index Ijk as a function of simple road characteristics from BC250, such as pavement type and number of lanes. The functional form is parameterized, and its coefficients are calibrated using a representative subsample of Brazilian links, for which I estimate the mapping from these basic BC250 attributes to effective infrastructure quality. The estimated coefficients are then applied to all BC250 links in Brazil to obtain the nationwide infrastructure measure.

7 6) Infrastructure Measure following FS

[70]:
```
# 6) Infrastructure measure (FS-style)

# Parâmetros tirados/derivados do FS OG
chi_lane   <- 0.355   # efeito de nº de faixas (aprox. da conta deles)
chi_use    <- 1.07    # custo maior em via não nacional
chi_paved  <- 1.35    # custo maior em estrada não pavimentada
chi_median <- 1.05    # custo maior sem canteiro/dupla pista
```

```r
I_jk <- edges_stats |>
  mutate(
    # 1) Médias ao longo do caminho (já estão calculadas via Dijkstra)
    share_paved   = stat_pav_pav,        # fração do caminho pavimentado
    lanes_mean    = if_else(is.na(stat_nrfaixas) | stat_nrfaixas < 1,
                            1, stat_nrfaixas),  # nº médio de faixas (mínimo 1)
    share_nat     = stat_jur_fed_est,    # fração em via fed/estadual (proxy
 ↪"national")
    share_median  = stat_pistas_duplas,  # fração em pista dupla (proxy de
 ↪median)

    # 2) Multiplicadores de custo em relação a uma referência "boa"
    mult_lane   = lanes_mean^(-chi_lane),
    mult_use    = chi_use^(1 - share_nat),
    mult_paved  = chi_paved^(1 - share_paved),
    mult_median = chi_median^(1 - share_median),

    # 3) Custo relativo de transporte pela qualidade da infraestrutura
    infra_cost_mult = mult_lane * mult_use * mult_paved * mult_median,

    # 4) Medida de infraestrutura: quanto maior, melhor a infraestrutura
    infra_index = 1 / infra_cost_mult
  )
```

[71]:
```r
plot_edge_stat(
  edges_sf  = I_jk,
  regions_sf = micro_br,
  var       = "infra_index",
  title     = "Rede - infrastructure index (FS-style)"
)
```



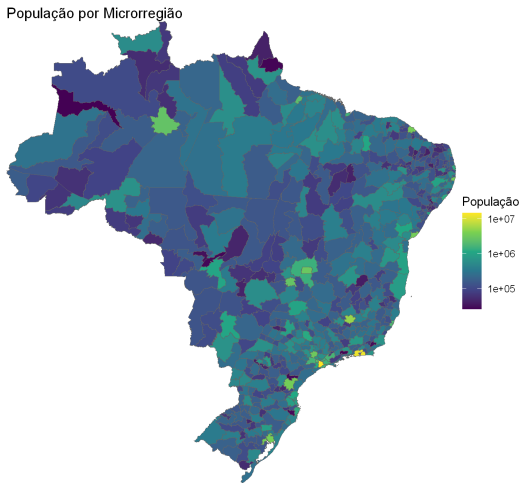
8 7) Nodes Statistics

8.1 7.1) Population and GDP from SIDRA

[72]:
```r
# ------------------------------------------------------------
# 7.1.1 Population (SIDRA 6579)
# ------------------------------------------------------------

POP_YEAR <- "2025"

pop_raw <- sidrar::get_sidra(
  6579,
  variable = 9324,
  period = POP_YEAR,
  geo = "City",
  format = 3
)

# Detect column names robustly (handles accents, name changes)
col_muni <- names(pop_raw)[
  grepl("Munic", names(pop_raw), ignore.case = TRUE) &
  grepl("C[oó]d", names(pop_raw), ignore.case = TRUE)
][1]

pop <- pop_raw |>
  transmute(
    cod_mun = suppressWarnings(as.integer(readr::parse_number(.
 ↪data[[col_muni]]))),
    populacao = suppressWarnings(as.numeric(Valor))
  )

lookup <- geobr::lookup_muni("all") |> select(code_muni, code_micro,␣
 ↪abbrev_state)

pop_micro <- pop |>
  left_join(lookup, by = c("cod_mun" = "code_muni")) |>
  group_by(code_micro, abbrev_state) |>
  summarise(populacao = sum(populacao, na.rm = TRUE), .groups = "drop")

micro_pop <- micro_br |>
  left_join(pop_micro, by = c("code_micro", "abbrev_state"))

ggplot(micro_pop) +
  geom_sf(aes(fill = populacao), color = "grey40", size = 0.05) +
  scale_fill_viridis_c(trans = "log10", na.value = "grey90") +
  coord_sf(expand = FALSE) +
  theme_void() +
```

```
  labs(fill = "População", title = "População por Microrregião")
```

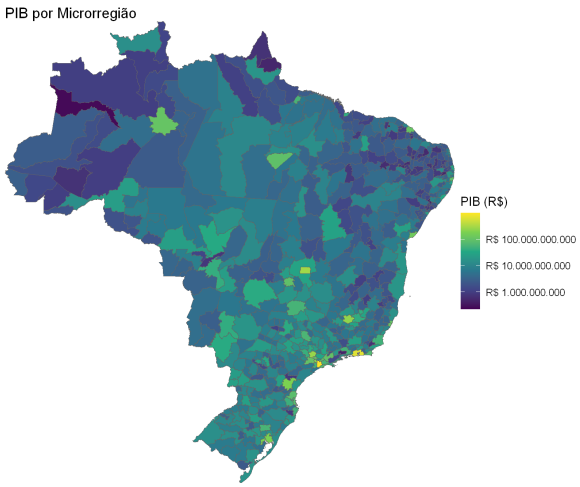
```
Considering all categories once 'classific' was set to 'all' (default)
```



[73]:
```
# ------------------------------------------------------------
# 7.1.2 GDP (SIDRA 5938)
# ------------------------------------------------------------

PIB_VAR <- 37

pib_raw <- sidrar::get_sidra(
  5938,
  variable = PIB_VAR,
  period = "last",
  geo = "City",
  format = 3
)

col_muni <- names(pib_raw)[
  grepl("Munic", names(pib_raw), ignore.case = TRUE) &
  grepl("C[oó]d", names(pib_raw), ignore.case = TRUE)
][1]

pib <- pib_raw |>
  transmute(
    cod_mun = suppressWarnings(as.integer(readr::parse_number(.
  ↪data[[col_muni]]))),
    pib_mil_rs = suppressWarnings(as.numeric(Valor))
  )
```

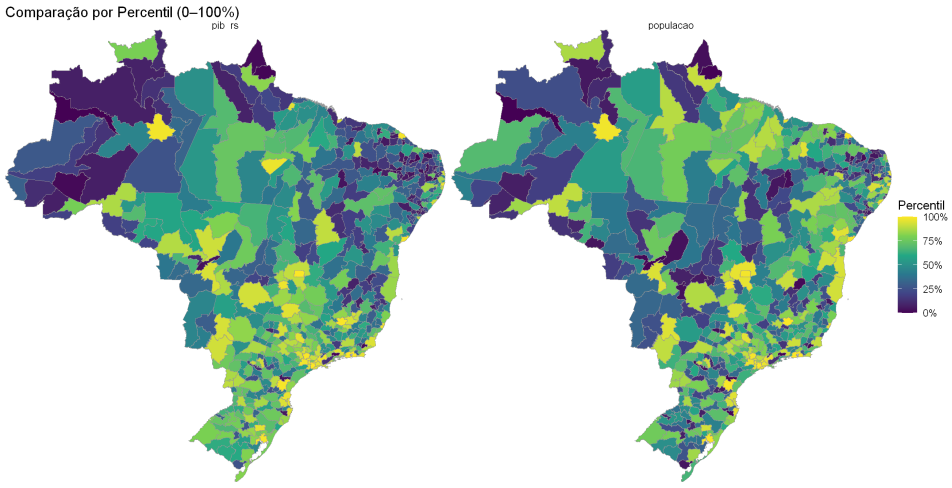
```
pib_micro <- pib |>
  left_join(lookup, by = c("cod_mun" = "code_muni")) |>
  group_by(code_micro, abbrev_state) |>
  summarise(pib_rs = sum(pib_mil_rs * 1000, na.rm = TRUE), .groups = "drop")

micro_pib <- micro_br |>
  left_join(pib_micro, by = c("code_micro", "abbrev_state"))

ggplot(micro_pib) +
  geom_sf(aes(fill = pib_rs), color = "grey40", size = 0.05) +
  scale_fill_viridis_c(
    trans = "log10",
    na.value = "grey90",
    labels = scales::label_number(big.mark = ".", decimal.mark = ",", prefix =␣
 ↪"R$ ")
  ) +
  coord_sf(expand = FALSE) +
  theme_void() +
  labs(fill = "PIB (R$)", title = "PIB por Microrregião")
```

```
Considering all categories once 'classific' was set to 'all' (default)
```



[74]:
```
library(tidyr)

# ------------------------------------------------------------
# 7.1.3 Join population and GDP + percentile maps
# ------------------------------------------------------------

micro_all <- micro_br |>
  left_join(pop_micro, by = c("code_micro", "abbrev_state")) |>
```

```r
  left_join(pib_micro, by = c("code_micro", "abbrev_state"))

mp_long <- micro_all |>
  select(code_micro, populacao, pib_rs) |>
  pivot_longer(
    c(populacao, pib_rs),
    names_to = "var",
    values_to = "val"
  )

mp_rel <- mp_long |>
  group_by(var) |>
  mutate(val_pct = percent_rank(val)) |>
  ungroup()

ggplot(mp_rel) +
  geom_sf(aes(fill = val_pct), color = "grey60", size = 0.06) +
  facet_wrap(~ var, nrow = 1) +
  scale_fill_viridis_c(
    limits = c(0,1),
    labels = scales::label_percent(accuracy = 1),
    name = "Percentil"
  ) +
  coord_sf(expand = FALSE) +
  theme_void() +
  labs(title = "Comparação por Percentil (0-100%)")
```



8.2 7.2) Population and GDP divided by sectors

8.2.1 7.2.1) SIDRA GDP per sector

[75]:
```
baixar_pib_setor <- function(var_cod, nome_var) {
  sidrar::get_sidra(
    x = 5938,
    variable = var_cod,
    geo = "City",
    period = "last",
    format = 3
  ) |>
    transmute(
      cod_mun = as.integer(`Município (Código)`),
      !!nome_var := as.numeric(Valor)
    )
}
```

[76]:
```
vars <- list(
  pib_total = 37,
  pib_agro  = 513,
  pib_imp   = 514,
  pib_serv  = 6575,
  pib_ind   = 543
)

pib_mun <- purrr::map2_dfr(
  vars,
  names(vars),
  baixar_pib_setor
) |>
  group_by(cod_mun) |>
  summarise(across(everything(), ~ first(na.omit(.x))), .groups = "drop")
```

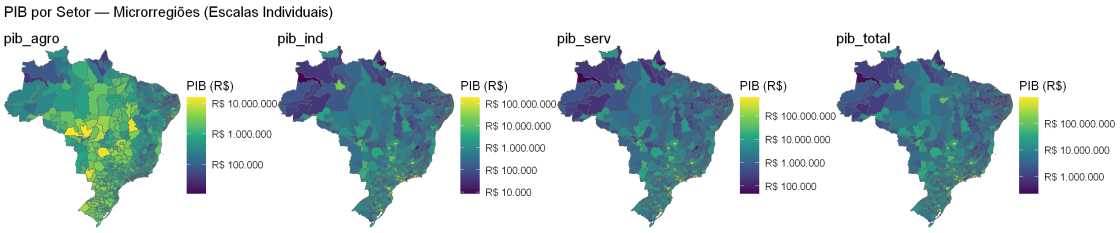
```
Considering all categories once 'classific' was set to 'all' (default)

Considering all categories once 'classific' was set to 'all' (default)

Considering all categories once 'classific' was set to 'all' (default)

Considering all categories once 'classific' was set to 'all' (default)

Considering all categories once 'classific' was set to 'all' (default)
```

[77]:
```
pib_micro <- pib_mun |>
  left_join(lookup, by = c("cod_mun" = "code_muni")) |>
  group_by(code_micro, abbrev_state) |>
```

```
  summarise(across(starts_with("pib_"), ~ sum(.x, na.rm = TRUE)), .groups =␣
 ↪"drop")
```

[78]:
```
micro_pib_setores <- micro_br |>
  left_join(pib_micro, by = c("code_micro", "abbrev_state"))
```

[79]:
```
library(ggplot2)
library(patchwork)

plot_pib_setor <- function(df, var) {
  ggplot(df) +
    geom_sf(aes(fill = .data[[var]]), color = "grey40", size = 0.05) +
    scale_fill_viridis_c(
      trans = "log10",
      na.value = "grey90",
      #breaks = c(1e3, 1e4, 1e5, 1e6, 1e7, 1e8),
      labels = scales::label_number(big.mark = ".", decimal.mark = ",", prefix␣
 ↪= "R$ ")#,scale_cut = scales::cut_short_scale())
    ) +
    coord_sf(expand = FALSE) +
    theme_void() +
    labs(
      title = var,
      fill = "PIB (R$)"
    )
}

p1 <- plot_pib_setor(micro_pib_setores, "pib_agro")
#p2 <- plot_pib_setor(micro_pib_setores, "pib_imp")
p3 <- plot_pib_setor(micro_pib_setores, "pib_ind")
p4 <- plot_pib_setor(micro_pib_setores, "pib_serv")
p5 <- plot_pib_setor(micro_pib_setores, "pib_total")

(p1 | p3 | p4 | p5) +
  plot_annotation(
    title = "PIB por Setor - Microrregiões (Escalas Individuais)"
  )
```



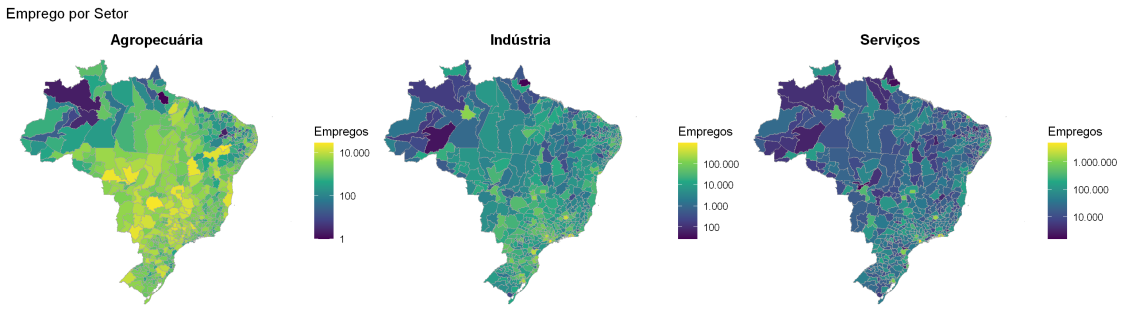
8.2.2 7.2.2) RAIS population per sector

[80]:
```
library(readr)
library(dplyr)
library(tidyr)
library(geobr)
library(ggplot2)
library(scales)
library(stringr)
library(stringi)
library(patchwork)

# 1) Ler e preparar RAIS
colunas <- c(
  "municipio",
  "setor1_extrativa",
  "setor2_transformacao",
  "setor3_utilidade_publica",
  "setor4_construcao",
  "setor5_comercio",
  "setor6_servicos",
  "setor7_administracao",
  "setor8_agro_pesca",
  "setor9_ignorado",
  "total"
)

emprego_raw <- read_csv2(
  "consulta9293258.csv",
  skip = 1,
```

```
  col_names = colunas,
  locale = locale(encoding = "latin1"),
  na = c("", "*", "-")
) |>
  mutate(across(-municipio, parse_number))

norm_nome <- function(x) {
  x |>
    stringi::stri_trans_general("Latin-ASCII") |>
    tolower() |>
    str_remove("^[a-z]{2}-") |>
    str_replace_all("[^a-z ]", " ") |>
    str_squish()
}

lookup <- geobr::lookup_muni("all") |>
  transmute(
    code_muni, code_micro, abbrev_state,
    nome_chave = norm_nome(name_muni)
  )

emprego_geo <- emprego_raw |>
  mutate(
    agro = setor8_agro_pesca,
    industria = setor1_extrativa + setor2_transformacao +
                setor3_utilidade_publica + setor4_construcao,
    servicos = setor5_comercio + setor6_servicos + setor7_administracao,
    nome_chave = norm_nome(municipio)
  ) |>
  left_join(lookup, by = "nome_chave")

emprego_micro <- emprego_geo |>
  group_by(code_micro, abbrev_state) |>
  summarise(across(c(agro, industria, servicos), sum, na.rm = TRUE), .groups =␣
 ↪"drop")

micro_emprego <- geobr::read_micro_region(year = 2013) |>
  left_join(emprego_micro, by = c("code_micro", "abbrev_state"))

# 2) Plots com escalas independentes
plot_emprego_setor <- function(df, var, titulo) {
  ggplot(df, aes(fill = .data[[var]])) +
    geom_sf(color = "grey65", size = 0.05) +
    scale_fill_viridis_c(
      trans = "log10",
      labels = label_number(big.mark = ".", decimal.mark = ","),
      name = "Empregos"
```

```
    ) +
    labs(title = titulo) +
    theme_void() +
    theme(
      plot.title = element_text(hjust = 0.5, face = "bold"),
      legend.position = "right"
    )
}

p_agro <- plot_emprego_setor(micro_emprego, "agro", "Agropecuária")
p_ind  <- plot_emprego_setor(micro_emprego, "industria", "Indústria")
p_serv <- plot_emprego_setor(micro_emprego, "servicos", "Serviços")

(p_agro | p_ind | p_serv) +
  plot_layout(guides = "keep") +
  plot_annotation(title = "Emprego por Setor")
```

```
  Using "','" as decimal and
"'.'" as grouping mark. Use `read_delim()` for more control.

Rows: 5577 Columns: 11
  Column specification

Delimiter: ";"
chr (11): municipio, setor1_extrativa, setor2_transformacao,
setor3_utilidad…

  Use `spec()` to retrieve the full column specification for this
data.
  Specify the column types or set `show_col_types = FALSE` to quiet
this message.
Using year/date 2013
```



8.2.3 7.2.3) Censo population and gdp per sector

[81]:
```
#TBD
```

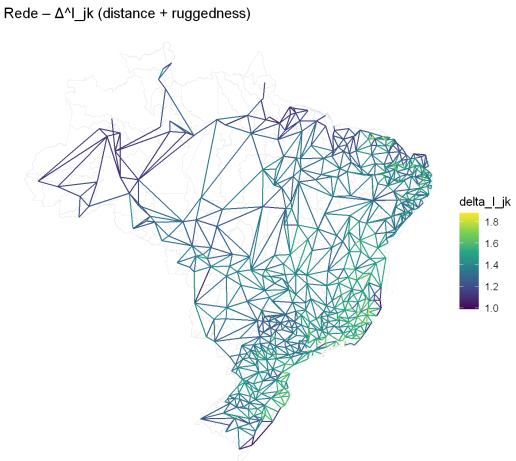
9 8) Cost of building infrastructure

9.1 8.1) delta FS

[82]:
```
# 6) Delta^I_jk - distance + ruggedness (FS/Collier)

library(terra)
library(purrr)

# 6.1 Ruggedness ao longo de cada aresta (ETOPO1)
ETOPO_PATH <- "ETOPO1_Ice_g_geotiff.tif"  # ajuste se estiver em outro lugar
etopo <- terra::rast(ETOPO_PATH)
if (is.na(terra::crs(etopo))) terra::crs(etopo) <- "EPSG:4326"

compute_ruggedness <- function(edges_sf, etopo_rast) {
  edges_ll <- sf::st_transform(edges_sf, crs = terra::crs(etopo_rast))
  rugged_fun <- function(g) {
    pts  <- sf::st_line_sample(g, density = 50) |> sf::st_coordinates()
    elev <- terra::extract(etopo_rast, pts[, c("X","Y")])[,1]
    if (sum(!is.na(elev)) < 3) return(NA_real_)
    sd(diff(elev), na.rm = TRUE)
  }
  dplyr::mutate(edges_sf,
                rugged = purrr::map_dbl(edges_ll$geometry, rugged_fun))
}

rugged <- compute_ruggedness(I_jk, etopo)

# 6.2 Delta^I_jk usando exatamente os coeficientes do FS
compute_delta_geo <- function(edges_sf,
                              rugged_col = "rugged",
                              crs_metric = 5880) {

  edges_m <- sf::st_transform(edges_sf, crs_metric)
  dist_km <- as.numeric(sf::st_length(edges_m)) / 1000
  rugged  <- edges_sf[[rugged_col]]

  delta0 <- 1
  delta1 <- -0.11   # dummy dist>50 km
  delta2 <-  0.12   # ln(ruggedness)
```

```
  delta_geo <- exp(
    log(delta0) +
      delta1 * (dist_km > 50) +
      delta2 * log(pmax(rugged, 1e-4))
  )

  dplyr::mutate(edges_sf,
                dist_km    = dist_km,
                delta_I_jk = delta_geo)
}

delta_I_jk <- compute_delta_geo(rugged)
```

[83]:
```
plot_edge_stat(
  edges_sf  = delta_I_jk,
  regions_sf = micro_br,
  var       = "delta_I_jk",
  title     = "Rede - Δ^I_jk (distance + ruggedness)"
)
```



9.2 8.2) new delta (TBD)

[84]:
```
#library(sf)
#library(dplyr)
#library(terra)

#rescale01 <- function(x) {
#  x <- as.numeric(x)
#  x_ok <- x[is.finite(x)]
```

```
#  if (!length(x_ok)) return(rep(0, length(x)))
#  rng <- range(x_ok, na.rm = TRUE)
#  if (diff(rng) == 0) return(rep(0, length(x)))
#  (x - rng[1]) / diff(rng)
#}

#build_env_nodes <- function(nodes_sf, DATA,
#                            crs_ll = 4326,
#                            soil_col = "soil_class") {
#
#  nodes_ll <- st_transform(nodes_sf, crs_ll)

  ## 1) Erosão IBGE (2.5 GB, mas só 1 extract em pontos)
#  eros_r <- rast(DATA$erosao_2020_ibge)
#  if (is.na(crs(eros_r))) crs(eros_r) <- sprintf("EPSG:%d", crs_ll)
#  eros_vals <- terra::extract(eros_r, vect(nodes_ll))[,1]
#  eros_vals[is.na(eros_vals)] <- median(eros_vals, na.rm = TRUE)
#  eros_idx <- rescale01(eros_vals)

  ## 2) Solo Embrapa WRB -> índice de fragilidade
#  soils <- st_read(DATA$solos_wrb[1], quiet = TRUE) |>
#    st_make_valid() |>
#    st_transform(crs_ll)

#  soil_join <- st_join(nodes_ll, soils)
#  soil_raw <- as.factor(soil_join[[soil_col]])
#  soil_num <- as.numeric(soil_raw)
#  soil_num[is.na(soil_num)] <- median(soil_num, na.rm = TRUE)
#  soil_frag <- rescale01(soil_num)

  ## 3) MapBiomas 10m -> raster agregado mais leve + share de floresta
#  mb <- rast(DATA$mapbiomas_10m_2023)
#  if (is.na(crs(mb))) crs(mb) <- sprintf("EPSG:%d", crs_ll)

#  forest_codes <- c(3L, 4L, 5L)
#  mb_forest <- mb %in% forest_codes

  # 10m -> ~1km (ajuste o factor se quiser mais fino/grosso)
#  fact <- 100L
#  mb_forest_agg <- terra::aggregate(mb_forest, fact = fact, fun = mean, na.rm
  ↪= TRUE)

#  forest_share <- terra::extract(mb_forest_agg, vect(nodes_ll))[,1]
#  forest_share[is.na(forest_share)] <- 0

#  env_nodes <- nodes_sf |>
#    st_drop_geometry() |>
```

```
#    transmute(
#      id,
#      eros_idx     = eros_idx,
#      soil_frag    = soil_frag,
#      forest_share = pmin(pmax(forest_share, 0), 1)
#    )

#  env_nodes
#}

# --- só roda UMA vez; depois é só ler o RDS ---
#env_nodes <- build_env_nodes(nodes_micro, DATA)
#dir.create("data", showWarnings = FALSE)
#saveRDS(env_nodes, "data/env_nodes.rds")
```

[85]:
```
#env_nodes <- readRDS("data/env_nodes.rds")

# média i-j
#env_i <- env_nodes |> rename(id_i = id,
#                             eros_i = eros_idx,
#                             soil_i = soil_frag,
#                             forest_i = forest_share)
#env_j <- env_nodes |> rename(id_j = id,
#                             eros_j = eros_idx,
#                             soil_j = soil_frag,
#                             forest_j = forest_share)

#edges_env <- I_jk |>
#  left_join(env_i, by = "id_i") |>
#  left_join(env_j, by = "id_j") |>
#  mutate(
#    eros_mean   = rowMeans(cbind(eros_i, eros_j), na.rm = TRUE),
#    soil_mean   = rowMeans(cbind(soil_i, soil_j), na.rm = TRUE),
#    forest_mean = rowMeans(cbind(forest_i, forest_j), na.rm = TRUE)
#  )

# distância + rugged (que já é rápido no seu código atual)
#edges_m  <- st_transform(edges_env, 5880)
#dist_km  <- as.numeric(st_length(edges_m)) / 1000
#rugged_m <- edges_env$rugged_mean   # se você já tem, ou use sua versão atual

#delta0      <- 1
#delta1_dist <- -0.11
#delta2_rug  <-  0.12
#theta_eros   <- 0.20
#theta_soil   <- 0.15
#theta_forest <- 0.25
```

```
#delta_I_jk_pedro <- exp(
#  log(delta0) +
#    delta1_dist * (dist_km > 50) +
#    delta2_rug  * log(pmax(rugged_m, 1e-4)) +
#    theta_eros   * rescale01(eros_mean) +
#    theta_soil   * rescale01(soil_mean) +
#    theta_forest * forest_mean
#)

#delta_I_jk_pedro <- edges_env |>
#  mutate(
#    dist_km         = dist_km,
#    delta_I_jk_pedro = delta_I_jk_pedro
#  )
```

[86]:
```
#plot_edge_stat(
#  edges_sf  = delta_I_jk_pedro,
#  regions_sf = micro_br,
#  var       = "delta_I_jk",
#  title     = "Rede - Δ^I_jk (distance + ruggedness)"
#)
```

10 9) Creating Matrices and exporting to matlab

[87]:
```
# ------------------------------------------------------------
# 9) Matrizes + export + plots (hidro + aeroportos, sem mults)
# ------------------------------------------------------------
# Fontes fixas:
#   - Solo: soils_brazil_wrb_wgs84.shp
#   - Hidrovias: bc250 completo/hdv_trecho_hidroviario_l.shp
#   - Eclusas: bc250 completo/hdv_eclusa_p.shp
#   - Portos: bc250 completo/hdv_complexo_portuario_p.shp
#   - Barragens: bc250 completo/hid_barragem_p.shp
#   - Aeroportos: bc250 completo/aer_complexo_aeroportuario_p.shp (sem pistas)

suppressPackageStartupMessages({
  library(dplyr); library(sf); library(terra); library(R.matlab)
  library(lwgeom)
  library(ggplot2); library(patchwork); library(scales); library(grid)
})
OUT_DIR <- "output"; crs_ll <- 4326; crs_m <- 5880

# ---------------------------------------------------------------------
# Helpers
# ---------------------------------------------------------------------
```

```r
assert_cols <- function(x, cols, nm) {
  miss <- setdiff(cols, names(x))
  if (length(miss)) stop(nm, ": faltam colunas: ", paste(miss, collapse = ", "))
}
rescale01 <- function(x) {
  x <- as.numeric(x); ok <- is.finite(x)
  if (!any(ok)) return(rep(0, length(x)))
  r <- range(x[ok], na.rm = TRUE); if (diff(r) == 0) return(rep(0, length(x)))
  (x - r[1]) / diff(r)
}
fill_mat <- function(df, value_col, ids) {
  m <- matrix(0, length(ids), length(ids), dimnames = list(ids, ids))
  ii <- match(df$id_i, ids); jj <- match(df$id_j, ids); vals <- df[[value_col]]
  ok <- is.finite(ii) & is.finite(jj) & !is.na(vals)
  for (k in which(ok)) {
    i <- ii[k]; j <- jj[k]; v <- vals[k]
    m[i,j] <- v; m[j,i] <- v
  }
  diag(m) <- 0
  m
}
avg_edge <- function(varname, nodes_sf, edges_sf) {
  df_nodes <- st_drop_geometry(nodes_sf)
  assert_cols(df_nodes, c("code_micro", varname), "avg_edge")
  vals  <- df_nodes[[varname]]
  idmap <- df_nodes$code_micro
  vi <- vals[match(edges_sf$id_i, idmap)]
  vj <- vals[match(edges_sf$id_j, idmap)]
  rowMeans(cbind(vi, vj), na.rm = TRUE)
}
norm_txt <- function(x) {
  x <- tolower(iconv(x, to = "ASCII//TRANSLIT"))
  trimws(x)
}

# ---------------------------------------------------------------------
# Bases obrigatórias
# ---------------------------------------------------------------------
assert_cols(nodes_micro, c("geometry"), "nodes_micro")
assert_cols(I_jk,       c("id_i","id_j","geometry"), "I_jk")
if (!"code_micro" %in% names(nodes_micro)) {
  if ("id" %in% names(nodes_micro)) nodes_micro$code_micro <- nodes_micro$id
  else stop("nodes_micro precisa de code_micro ou id.")
}

# ---------------------------------------------------------------------
# Coordenadas / nós
```

```r
# ------------------------------------------------------------------
nodes_ll <- nodes_micro |> st_transform(crs_ll)
coords   <- st_coordinates(nodes_ll)
nodes_ll$lon <- coords[,1]
nodes_ll$lat <- coords[,2]

# ------------------------------------------------------------------
# Ruggedness (ETOPO TRI por n\u00f3)
# ------------------------------------------------------------------
tryCatch({
  etopo_path <- if (exists("DATA") && "etopo1_elevation" %in% names(DATA))
    DATA$etopo1_elevation else "ETOPO1_Ice_g_geotiff.tif"
  etopo_r <- rast(etopo_path)
  if (is.na(crs(etopo_r))) crs(etopo_r) <- sprintf("EPSG:%d", crs_ll)
  tri_rugged <- terra::terrain(etopo_r, v = "TRI", neighbors = 8)
  rugged_node <- terra::extract(tri_rugged, terra::vect(nodes_ll))[,1]
  rugged_node[is.na(rugged_node)] <- median(rugged_node, na.rm = TRUE)
  nodes_ll$rugged_tri <- rugged_node
  message("\u2714 Ruggedness (ETOPO TRI) por n\u00f3")
}, error = function(e) message("[ruggedness] ", e$message))

# ------------------------------------------------------------------
# Solo (WRB)
# ------------------------------------------------------------------
tryCatch({
  soils <- st_read("soils_brazil_wrb_wgs84.shp", quiet = TRUE) |>
    st_make_valid() |>
    st_transform(crs_ll)
  soil_join <- st_join(nodes_ll, soils)
  nodes_ll$soil_cat_idx <- rescale01(as.numeric(factor(soil_join$symbol_wrb)))
  message("  Solo: soils_brazil_wrb_wgs84.shp (indice categ. 0-1)")
}, error = function(e) message("[solo] ", e$message))

# ------------------------------------------------------------------
# Eclusas, Portos, Barragens (nós = contagem por microrregião)
# ------------------------------------------------------------------
tryCatch({
  locks_sf <- st_read("bc250 completo/hdv_eclusa_p.shp", quiet = TRUE) |>
    st_transform(crs_ll)
  locks_by_micro <- st_join(st_transform(micro_br, crs_ll), locks_sf, join =␣
 ↪st_intersects) |>
    st_drop_geometry() |>
    count(code_micro, name = "n_locks")
  nodes_ll$n_locks <- locks_by_micro$n_locks[match(nodes_ll$code_micro,␣
 ↪locks_by_micro$code_micro)]
  nodes_ll$n_locks[is.na(nodes_ll$n_locks)] <- 0
  message("  Eclusas: contagem por microrregião")
```

```
}, error = function(e) message("[eclusas] ", e$message))

tryCatch({
  ports_sf <- st_read("bc250 completo/hdv_complexo_portuario_p.shp", quiet =␣
 ↪TRUE) |>
    st_transform(crs_ll)
  ports_by_micro <- st_join(st_transform(micro_br, crs_ll), ports_sf, join =␣
 ↪st_intersects) |>
    st_drop_geometry() |>
    count(code_micro, name = "n_ports")
  nodes_ll$n_ports <- ports_by_micro$n_ports[match(nodes_ll$code_micro,␣
 ↪ports_by_micro$code_micro)]
  nodes_ll$n_ports[is.na(nodes_ll$n_ports)] <- 0
  message("  Portos: contagem por microrregião")
}, error = function(e) message("[portos] ", e$message))

tryCatch({
  dams_sf <- st_read("bc250 completo/hid_barragem_p.shp", quiet = TRUE) |>
    st_transform(crs_ll)
  dams_by_micro <- st_join(st_transform(micro_br, crs_ll), dams_sf, join =␣
 ↪st_intersects) |>
    st_drop_geometry() |>
    count(code_micro, name = "n_dams")
  nodes_ll$n_dams <- dams_by_micro$n_dams[match(nodes_ll$code_micro,␣
 ↪dams_by_micro$code_micro)]
  nodes_ll$n_dams[is.na(nodes_ll$n_dams)] <- 0
  message("  Barragens: contagem por microrregião")
}, error = function(e) message("[barragens] ", e$message))

# ---------------------------------------------------------------------
# Aeroportos (nós: dummy > 0)
# ---------------------------------------------------------------------
tryCatch({
  aero_sf <- st_read("bc250 completo/aer_complexo_aeroportuario_p.shp", quiet =␣
 ↪TRUE) |>
    st_make_valid() |>
    st_transform(crs_ll)

  coords         <- st_coordinates(aero_sf)
  coords_rounded <- round(coords, 5)
  aero_sf$xy_round <- paste(coords_rounded[,1], coords_rounded[,2])
  aero_sf <- aero_sf[!duplicated(aero_sf$xy_round), ]

  reg_aer   <- st_transform(micro_br, crs_ll)
  aer_micro <- st_join(aero_sf, reg_aer["code_micro"], left = FALSE)
```

```
  aer_counts <- aer_micro |>
    st_drop_geometry() |>
    count(code_micro, name = "n_airports")

  nodes_ll$n_airports <- aer_counts$n_airports[match(nodes_ll$code_micro,␣
 ↪aer_counts$code_micro)]
  nodes_ll$n_airports[is.na(nodes_ll$n_airports)] <- 0
  nodes_ll$n_airport_dummy <- as.numeric(nodes_ll$n_airports > 0)

  message("  Aeroportos: complexos dedup + intersect")
}, error = function(e) message("[aeroportos] ", e$message))

# ---------------------------------------------------------------------
# Tabela de nós
# ---------------------------------------------------------------------
assert_cols(micro_all,   c("code_micro","abbrev_state","populacao","pib_rs"),␣
 ↪"micro_all")
assert_cols(pib_micro,   c("code_micro","abbrev_state"),                    ␣
 ↪"pib_micro")
assert_cols(micro_emprego,␣
 ↪c("code_micro","abbrev_state","agro","industria","servicos"),␣
 ↪"micro_emprego")

ids_micro <- sort(unique(c(I_jk$id_i, I_jk$id_j)))
J <- length(ids_micro)

nodes_tbl <- nodes_ll |>
  st_drop_geometry() |>
  left_join(micro_all |> st_drop_geometry() |>
              select(code_micro, abbrev_state, populacao, pib_rs),
            by = "code_micro") |>
  left_join(pib_micro, by = c("code_micro","abbrev_state")) |>
  left_join(micro_emprego |> st_drop_geometry() |>
              select(code_micro, abbrev_state, agro, industria, servicos),
            by = c("code_micro","abbrev_state")) |>
  mutate(
    pop_total = populacao,
    pib_total = coalesce(pib_total, pib_rs),
    pop_agro  = agro,
    pop_ind   = industria,
    pop_serv  = servicos
  ) |>
  filter(code_micro %in% ids_micro) |>
  arrange(match(code_micro, ids_micro))

df_nodes <- nodes_tbl |>
  transmute(
```

```
    id  = as.numeric(code_micro),
    pop = as.numeric(pop_total),
    pib = as.numeric(pib_total),
    lon = as.numeric(lon),
    lat = as.numeric(lat)
  ) |>
  as.matrix()

pib_sector_cols <- setdiff(grep("^pib_", names(nodes_tbl), value = TRUE),␣
 ↪"pib_total")
pib_sector_mat  <- if (length(pib_sector_cols))
  as.matrix(nodes_tbl[, pib_sector_cols, drop = FALSE]) else NULL

pop_sector_cols <- intersect(c("pop_agro","pop_ind","pop_serv"),␣
 ↪names(nodes_tbl))
pop_sector_mat  <- if (length(pop_sector_cols))
  as.matrix(nodes_tbl[, pop_sector_cols, drop = FALSE]) else NULL

node_airports <- if ("n_airport_dummy" %in% names(nodes_tbl))
  as.numeric(nodes_tbl$n_airport_dummy) else NULL

# ---------------------------------------------------------------------
# Arestas / matrizes
# ---------------------------------------------------------------------
edges_sf <- I_jk |>
  st_transform(crs_m) |>
  mutate(dist_km = as.numeric(st_length(geometry)) / 1000)

if ("share_nat" %in% names(edges_sf))
  edges_sf$share_fed_state <- edges_sf$share_nat

# Hidrovias como ARESTA: km de hidrovias do BC250 "próximas" à aresta
tryCatch({

  # carregar hidrovias operacionais
  hydro_m <- st_read("bc250 completo/hdv_trecho_hidroviario_l.shp", quiet =␣
 ↪TRUE) |>
    st_make_valid() |>
    st_transform(crs_m) |>
    mutate(
      oper_norm = norm_txt(operaciona),
      sit_norm  = norm_txt(situacaofi)
    ) |>
    filter(
      oper_norm %in% c("sim", "operacional"),
      sit_norm  %in% c("construida","em operacao")
    ) |>
```

```
    dplyr::select(geometry)

  # buffer em torno de cada aresta (corredor de navegação)
  buf_dist <- 20000  # 20 km

  edges_buf <- st_buffer(
    edges_sf |> dplyr::select(id_i, id_j, geometry),
    dist = buf_dist
  )

  # interseção: hidrovias que passam dentro do corredor da aresta
  inter <- suppressWarnings(
    st_intersection(edges_buf, hydro_m)
  )

  if (nrow(inter) == 0) {
    edges_sf$waterway_km <- 0
  } else {
    inter$len_km <- as.numeric(st_length(inter)) / 1000  # km

    water_by_edge <- inter |>
      st_drop_geometry() |>
      group_by(id_i, id_j) |>
      summarise(waterway_km = sum(len_km, na.rm = TRUE), .groups = "drop")

    key_edges <- paste(edges_sf$id_i, edges_sf$id_j)
    key_water <- paste(water_by_edge$id_i, water_by_edge$id_j)
    m <- match(key_edges, key_water)

    edges_sf$waterway_km <- 0
    ok <- !is.na(m)
    edges_sf$waterway_km[ok] <- water_by_edge$waterway_km[m[ok]]
  }

  # dummy só pra facilitar plot / matrizes binárias
  edges_sf$waterway_dummy <- as.integer(edges_sf$waterway_km > 0)

  message(
    "  Hidrovias (aresta): km de hidrovias em buffer de ",
    buf_dist/1000, " km das arestas"
  )

}, error = function(e) {
  message("[hidrovias-aresta] ", e$message)
  edges_sf$waterway_km    <- 0
  edges_sf$waterway_dummy <- 0
})
```

```r
# Atributos de aresta vindos dos nós
if ("rugged_tri" %in% names(nodes_ll))
  edges_sf$rugged_mean <- avg_edge("rugged_tri", nodes_ll, I_jk)
if ("soil_cat_idx" %in% names(nodes_ll))
  edges_sf$soil_cat_idx <- avg_edge("soil_cat_idx", nodes_ll, I_jk)
if ("n_ports" %in% names(nodes_ll))
  edges_sf$n_ports <- avg_edge("n_ports", nodes_ll, I_jk)
if ("n_locks" %in% names(nodes_ll))
  edges_sf$n_locks <- avg_edge("n_locks", nodes_ll, I_jk)
if ("n_dams" %in% names(nodes_ll))
  edges_sf$n_dams <- avg_edge("n_dams", nodes_ll, I_jk)

edge_comp_cols <- c(
  "share_paved","lanes_mean","share_fed_state","share_median",
  "infra_index","dist_km","rugged_mean",
  "soil_cat_idx","waterway_km",
  "n_ports","n_locks","n_dams"
)
edge_comp_cols <- edge_comp_cols[edge_comp_cols %in% names(edges_sf)]

edges_list <- edges_sf |>
  st_drop_geometry() |>
  select(id_i, id_j, all_of(edge_comp_cols)) |>
  distinct()

adj_mat   <- edges_list |> mutate(val = 1) |> fill_mat("val", ids_micro)
if (!"infra_index" %in% names(edges_list))
  stop("Faltou infra_index em I_jk.")
infra_mat <- fill_mat(edges_list, "infra_index", ids_micro)
dist_mat  <- fill_mat(edges_list, "dist_km", ids_micro)

# Aglomeracao por aresta: populacao de i, j e vizinhos
pop_vec <- setNames(as.numeric(nodes_tbl$pop_total), ids_micro)
pop_vec[is.na(pop_vec)] <- 0

neighbors_of <- function(node_id) {
  idx <- match(node_id, ids_micro)
  if (is.na(idx)) return(integer(0))
  ids_micro[which(adj_mat[idx, ] > 0)]
}

edge_agglom_pop <- vapply(
  seq_len(nrow(edges_list)),
  function(k) {
    i_id <- edges_list$id_i[k]; j_id <- edges_list$id_j[k]
    neigh_i <- neighbors_of(i_id)
```

```
    neigh_j <- neighbors_of(j_id)
    nodes_use <- unique(c(i_id, j_id, neigh_i, neigh_j))
    sum(pop_vec[match(nodes_use, ids_micro)], na.rm = TRUE)
  },
  numeric(1)
)

edges_list$edge_agglom_pop <- edge_agglom_pop
edge_comp_cols <- unique(c(edge_comp_cols, "edge_agglom_pop"))

edge_keys_list <- paste(edges_list$id_i, edges_list$id_j)
edge_keys_sf   <- paste(edges_sf$id_i, edges_sf$id_j)
edges_sf$edge_agglom_pop <- edge_agglom_pop[match(edge_keys_sf, edge_keys_list)]

extra_cols <- setdiff(edge_comp_cols, c("infra_index","dist_km"))
extra_mats <- lapply(extra_cols, function(v) fill_mat(edges_list, v, ids_micro))
names(extra_mats) <- extra_cols

rugged_mat <- if ("rugged_mean" %in% names(extra_mats))
  extra_mats[["rugged_mean"]] else NULL
edge_agglom_mat <- if ("edge_agglom_pop" %in% names(extra_mats))
  extra_mats[["edge_agglom_pop"]] else NULL

find_first_existing <- function(cands, pool) cands[cands %in% pool][1]
A_solo      <- if (!is.na(find_first_existing(c("soil_cat_idx"),␣
 ↪names(extra_mats))))
  extra_mats[[find_first_existing(c("soil_cat_idx"), names(extra_mats))]] else␣
 ↪NULL
A_hidro     <- if (!is.na(find_first_existing(c("waterway_km"),␣
 ↪names(extra_mats))))
  extra_mats[[find_first_existing(c("waterway_km"), names(extra_mats))]] else␣
 ↪NULL
A_portos    <- if (!is.na(find_first_existing(c("n_ports"), names(extra_mats))))
  extra_mats[[find_first_existing(c("n_ports"), names(extra_mats))]] else NULL
A_eclusas   <- if (!is.na(find_first_existing(c("n_locks"), names(extra_mats))))
  extra_mats[[find_first_existing(c("n_locks"), names(extra_mats))]] else NULL
A_barragens <- if (!is.na(find_first_existing(c("n_dams"), names(extra_mats))))
  extra_mats[[find_first_existing(c("n_dams"), names(extra_mats))]] else NULL

emis <- rep(0, J)

# ---------------------------------------------------------------------
# Export .mat
# ---------------------------------------------------------------------
if (!dir.exists(OUT_DIR)) dir.create(OUT_DIR, recursive = TRUE)
mat_path <- file.path(OUT_DIR, "graph_BR.mat")
```

```
export_list <- list(
  con      = mat_path,
  A        = adj_mat,
  I        = infra_mat,
  df_nodes = df_nodes,
  ids      = as.numeric(ids_micro),
  J        = as.double(J),
  dist_mat = dist_mat,
  emis     = emis
)
if (!is.null(rugged_mat)) {
  export_list$rugged_mat <- rugged_mat
}
if (!is.null(edge_agglom_mat)) {
  export_list$edge_agglom_mat <- edge_agglom_mat
}
if (length(extra_mats)) {
  export_list$edge_mats       <- extra_mats
  export_list$edge_mats_names <- names(extra_mats)
}
if (!is.null(A_solo))      export_list$A_solo      <- A_solo
if (!is.null(A_hidro))     export_list$A_hidro     <- A_hidro
if (!is.null(A_portos))    export_list$A_portos    <- A_portos
if (!is.null(A_eclusas))   export_list$A_eclusas   <- A_eclusas
if (!is.null(A_barragens)) export_list$A_barragens <- A_barragens

export_list$node_pop_total <- as.numeric(nodes_tbl$pop_total)
export_list$node_pib_total <- as.numeric(nodes_tbl$pib_total)

if (!is.null(pib_sector_mat)) {
  export_list$node_pib_sector       <- pib_sector_mat
  export_list$node_pib_sector_names <- pib_sector_cols
}
if (!is.null(pop_sector_mat)) {
  export_list$node_pop_sector       <- pop_sector_mat
  export_list$node_pop_sector_names <- pop_sector_cols
}
if (!is.null(node_airports))
  export_list$node_airports <- node_airports

do.call(writeMat, export_list)
cat("OK: exportado", mat_path, "| J =", J,
    "| #arestas =", sum(adj_mat[upper.tri(adj_mat)]), "\n")

# ---------------------------------------------------------------------
# Plots
# ---------------------------------------------------------------------
```

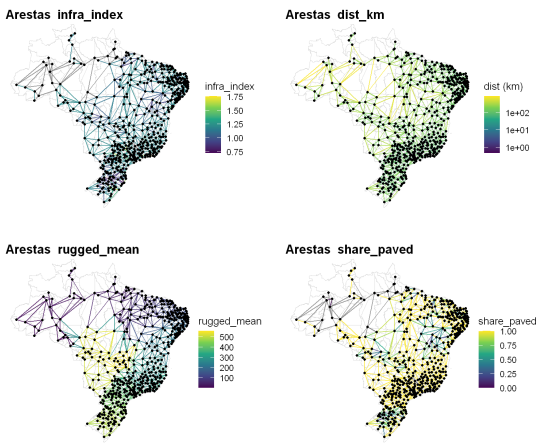
```r
edges_plot_ll <- edges_sf |> st_transform(crs_ll)
nodes_plot    <- nodes_tbl |> as.data.frame()

bbox_br <- list(xlim = c(-75, -34), ylim = c(-35, 6))
leg_small <- guides(color = guide_colorbar(title.position = "top",
                                           barheight = unit(18,"mm"),
                                           barwidth  = unit(5,"mm")))
map_theme <- theme_void(base_size = 10) +
  theme(legend.position = "right",
        legend.title    = element_text(size = 9),
        legend.text     = element_text(size = 8),
        plot.title      = element_text(face = "bold", size = 11, hjust = 0),
        plot.margin     = margin(4,4,4,4,"mm"))

edge_plot <- function(var, title, trans = NULL, name = var) {

  # CASO ESPECIAL: hidrovias
  if (var == "waterway_km") {
    e <- edges_plot_ll

    if (!"waterway_km" %in% names(e)) {
      stop("edges_plot_ll não tem coluna 'waterway_km'.")
    }

    # arestas que cruzam algum trecho hidroviário
    e$has_hidro <- e$waterway_km > 0

    return(
      ggplot() +
        geom_sf(data = micro_br, fill = NA, color = "grey85", linewidth = 0.2) +
        # rede completa em cinza claro
        geom_sf(data = e, color = "grey90", linewidth = 0.20, alpha = 0.4) +
        # apenas arestas com hidrovia em azul
        geom_sf(data = e[e$has_hidro, ], color = "#1565C0",
                linewidth = 0.55, alpha = 0.9) +
        geom_point(data = nodes_plot, aes(x = lon, y = lat),
                   color = "black", size = 0.8) +
        coord_sf(xlim = bbox_br$xlim, ylim = bbox_br$ylim, expand = FALSE) +
        map_theme + labs(title = title)
    )
  }

  # CASO GERAL
  sc <- if (is.null(trans)) scale_color_viridis_c(name = name)
        else                scale_color_viridis_c(trans = trans, name = name)

  ggplot() +
```

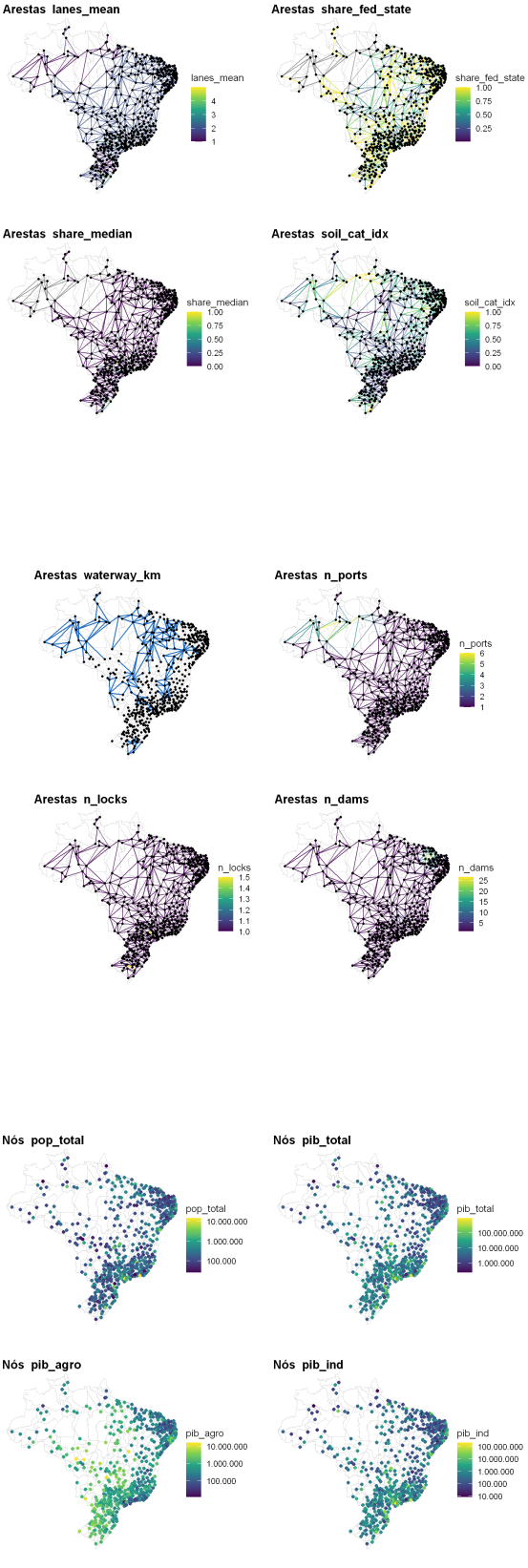
```
    geom_sf(data = micro_br, fill = NA, color = "grey85", linewidth = 0.2) +
    geom_sf(data = edges_plot_ll, aes(color = .data[[var]]),
            linewidth = 0.35, alpha = 0.8) +
    geom_point(data = nodes_plot, aes(x = lon, y = lat),
               color = "black", size = 0.8) +
    sc + leg_small +
    coord_sf(xlim = bbox_br$xlim, ylim = bbox_br$ylim, expand = FALSE) +
    map_theme + labs(title = title)
}

node_plot <- function(var, title, trans = "log10", name = var) {
  if (var == "n_airport_dummy") {
    cols <- c("0" = "#cccccc", "1" = "#1f77b4")
    nodes_plot$.__tmp_air__ <- factor(ifelse(nodes_plot[[var]] > 0, 1, 0))
    return(
      ggplot() +
        geom_sf(data = micro_br, fill = NA, color = "grey85", linewidth = 0.2) +
        geom_point(data = nodes_plot,
                   aes(x = lon, y = lat, color = .__tmp_air__),
                   size = 1.4) +
        scale_color_manual(values = cols, name = name,
                           labels = c("0" = "sem aeroporto", "1" = "tem␣
 ↪aeroporto")) +
        leg_small +
        coord_sf(xlim = bbox_br$xlim, ylim = bbox_br$ylim, expand = FALSE) +
        map_theme + labs(title = title)
    )
  }

  sc <- if (is.null(trans))
    scale_color_viridis_c(labels = label_number(big.mark=".", decimal.
 ↪mark=","), name = name)
  else
    scale_color_viridis_c(trans = trans,
                          labels = label_number(big.mark=".", decimal.mark=","),
                          name = name)

  ggplot() +
    geom_sf(data = micro_br, fill = NA, color = "grey85", linewidth = 0.2) +
    geom_point(data = nodes_plot,
               aes(x = lon, y = lat, color = .data[[var]]),
               size = 1.2) +
    sc + leg_small +
    coord_sf(xlim = bbox_br$xlim, ylim = bbox_br$ylim, expand = FALSE) +
    map_theme + labs(title = title)
}
```

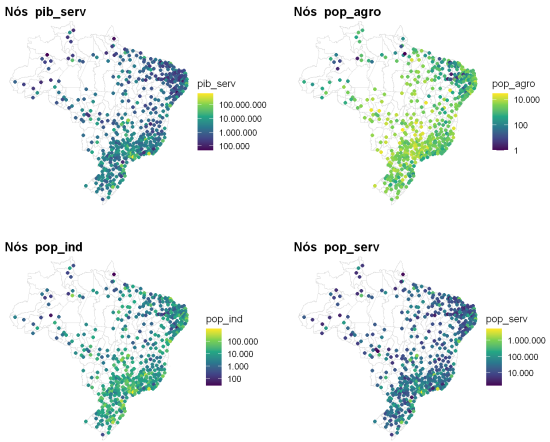
```r
view_plots <- function(plots, per_page = 4, ncol = 2) {
  n <- length(plots); if (!n) return(invisible(NULL))
  idx <- split(seq_len(n), ceiling(seq_len(n) / per_page))
  for (g in idx) print(wrap_plots(plots[g], ncol = ncol))
  invisible(NULL)
}

plots_edges_main <- lapply(
  intersect(c("infra_index","dist_km","rugged_mean"), names(edges_plot_ll)),
  function(v) edge_plot(
    v,
    paste("Arestas ", v),
    if (v == "dist_km") "log10" else NULL,
    if (v == "dist_km") "dist (km)" else v
  )
)

plots_edges_bc <- lapply(
  intersect(c("share_paved","lanes_mean","share_fed_state","share_median"),␣
 ↪names(edges_plot_ll)),
  function(v) edge_plot(v, paste("Arestas ", v))
)

plots_edges_env <- lapply(
  intersect(c("soil_cat_idx","waterway_km","n_ports","n_locks","n_dams"),
            names(edges_plot_ll)),
  function(v) edge_plot(v, paste("Arestas ", v))
)

plots_nodes_tot <- lapply(
  intersect(c("pop_total","pib_total"), names(nodes_plot)),
  function(v) node_plot(v, paste("Nós ", v))
)

plots_nodes_pib <- lapply(
  intersect(c("pib_agro","pib_ind","pib_serv"), names(nodes_plot)),
  function(v) node_plot(v, paste("Nós ", v))
)

plots_nodes_pop <- lapply(
  intersect(c("pop_agro","pop_ind","pop_serv"), names(nodes_plot)),
  function(v) node_plot(v, paste("Nós ", v))
)

plots_nodes_air <- if ("n_airport_dummy" %in% names(nodes_plot)) {
  list(node_plot("n_airport_dummy", "Nós aeroportos (dummy)",
                 trans = NULL, name = "dummy aeroportos"))
```

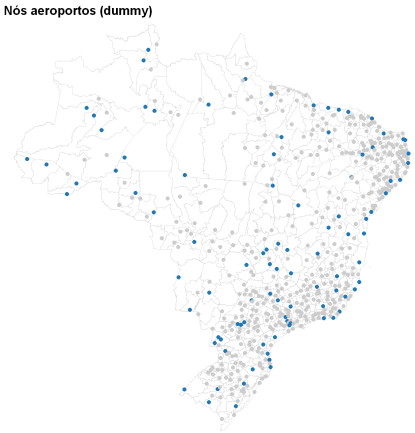
```
} else list()

message(">>> Plots principais")
view_plots(
  c(plots_edges_main, plots_edges_bc, plots_edges_env,
    plots_nodes_tot, plots_nodes_pib, plots_nodes_pop, plots_nodes_air),
  per_page = 4, ncol = 2
)
```

```
  Ruggedness (ETOPO TRI) por nó

  Solo: soils_brazil_wrb_wgs84.shp (indice categ. 0-1)

  Eclusas: contagem por microrregião

  Portos: contagem por microrregião

  Barragens: contagem por microrregião

  Aeroportos: complexos dedup + intersect

  Hidrovias (aresta): km de hidrovias em buffer de 20 km das arestas

OK: exportado output/graph_BR.mat | J = 555 | #arestas = 1441

>>> Plots principais
```




Arestas lanes_mean
lanes_mean
4
3
2
1
Arestas share_fed_state
share_fed_state
1.00
0.75
0.50
0.25
Arestas share_median
share_median
1.00
0.75
0.50
0.25
0.00
Arestas soil_cat_idx
soil_cat_idx
1.00
0.75
0.50
0.25
0.00
Arestas waterway_km
Arestas n_ports
n_ports
6
5
4
3
2
1
Arestas n_locks
n_locks
1.5
1.4
1.3
1.2
1.1
1.0
Arestas n_dams
n_dams
25
20
15
10
5
Nós pop_total
pop_total
10.000.000
1.000.000
100.000
Nós pib_total
pib_total
100.000.000
10.000.000
1.000.000
Nós pib_agro
pib_agro
10.000.000
1.000.000
100.000
Nós pib_ind
pib_ind
100.000.000
10.000.000
1.000.000
100.000
10.000





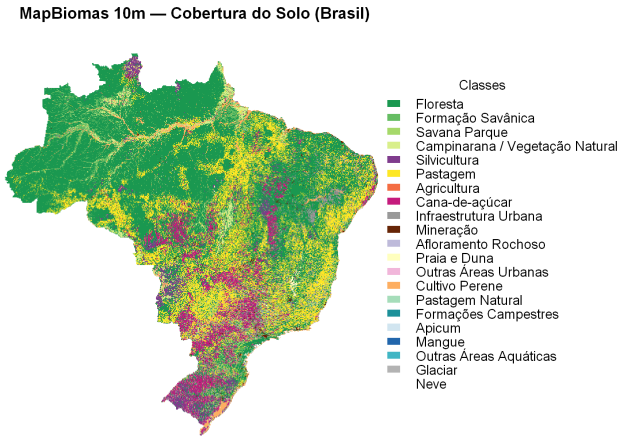
11 10) Missing Data

11.1 10.1) Mapbiomas Coverage

[88]:
```
plot_mapbiomas_10m <- function(tif,
                               aoi_path = NULL,
                               title = "MapBiomas 10m - Cobertura do Solo␣
↪(Brasil)") {

 r <- rast(tif)

 # Recorte opcional
```

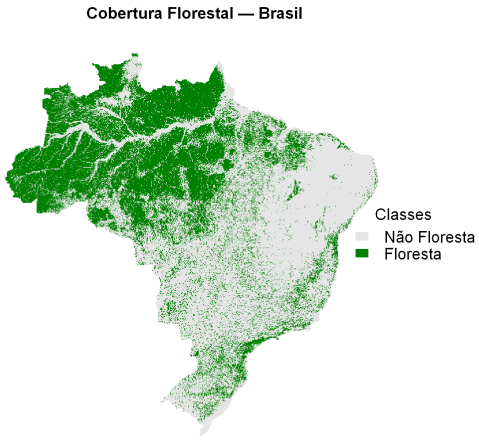
```r
if (!is.null(aoi_path) && file.exists(aoi_path)) {
  r <- crop(r, vect(aoi_path), mask = TRUE)
}

# === CORRIGIR FUNDO: 0 → NA ===
vals_raw <- unique(r[])
if (0 %in% vals_raw) {
  r[r == 0] <- NA
}

NAflag(r) <- NA

# Tabela oficial
mapbiomas_classes <- tibble::tribble(
  ~value, ~label, ~color,
   3, "Floresta", "#1A9850",
   4, "Formação Savânica", "#66BD63",
   5, "Savana Parque", "#A6D96A",
   6, "Campinarana / Vegetação Natural", "#D9EF8B",
   9, "Silvicultura", "#7F3C8D",
  11, "Pastagem", "#FDE725",
  12, "Agricultura", "#F46D43",
  13, "Agricultura Irrigada", "#F6C141",
  14, "Mosaico Agricultura e Pasto", "#BF812D",
  15, "Cana-de-açúcar", "#C51B7D",
  19, "Infraestrutura Urbana", "#999999",
  21, "Mineração", "#662506",
  22, "Outras Áreas Não Vegetadas", "#E6E6E6",
  23, "Afloramento Rochoso", "#BEBADA",
  24, "Praia e Duna", "#FFFFBF",
  25, "Outras Áreas Urbanas", "#F1B6DA",
  26, "Área Degradada", "#FDE0EF",
  27, "Solo Exposto", "#FDD0A2",
  29, "Cultivo Perene", "#FDAE61",
  30, "Pastagem Natural", "#A6DDBA",
  31, "Formações Campestres", "#1C9099",
  32, "Apicum", "#D1E5F0",
  33, "Mangue", "#2166AC",
  34, "Água", "#08306B",
  36, "Outras Áreas Aquáticas", "#41B6C4",
  49, "Glaciar", "#B3B3B3",
  50, "Neve", "#FFFFFF"
)

# Quais valores aparecem
vals <- sort(unique(na.omit(r[])))
df <- mapbiomas_classes[mapbiomas_classes$value %in% vals, ]
```

```
  col_vec <- df$color
  names(col_vec) <- df$value

  oldpar <- par(no.readonly = TRUE)
  on.exit(par(oldpar))

  par(bg = "white", mar = c(3, 3, 2, 12), xaxs = "i", yaxs = "i")

  plot(
    r,
    col = col_vec[match(vals, df$value)],
    legend = FALSE,
    axes = FALSE,
    box = FALSE,
    colNA = "white",
    main = title
  )

  legend(
    "right",
    legend = df$label,
    fill = df$color,
    border = NA,
    cex = 1.0,
    bty = "n",
    xpd = TRUE,
    inset = c(-0.45, 0),
    title = "Classes"
  )
}
plot_mapbiomas_10m(
  "C:/Users/Pedro/Desktop/OTNuEC/R/data/mapbiomas_near_1000m.tif"
)
```



[89]:
```r
plot_floresta_10m <- function(tif,
                              aoi_path = NULL,
                              title = "Cobertura Florestal - Brasil") {

  r <- rast(tif)

  # Recorte opcional
  if (!is.null(aoi_path) && file.exists(aoi_path)) {
    r <- crop(r, vect(aoi_path), mask = TRUE)
  }

  # Corrigir fundo: 0 → NA (fora do Brasil)
  if (0 %in% unique(r[])) {
    r[r == 0] <- NA
  }
  NAflag(r) <- NA

  # === Classes de floresta MapBiomas 10m ===
  # Seu dataset tem floresta apenas no código 3
  floresta_code <- 3

  # Criar raster com 2 classes:
  # NA  → fundo branco (fora BR)
  # 0   → não floresta
  # 1   → floresta
  r2 <- classify(r, rbind(
    c(floresta_code, floresta_code, 1),     # floresta
    c(-Inf, floresta_code - 0.001, 0),      # não floresta
    c(floresta_code + 0.001, Inf, 0)
  ))
```

```r
  # Paleta:
  cols <- c(
    "0" = "#E5E5E5",    # cinza claro
    "1" = "#008000"     # verde forte
  )

  oldpar <- par(no.readonly = TRUE)
  on.exit(par(oldpar))

  par(bg = "white", mar = c(3,3,2,8), xaxs = "i", yaxs = "i")

  plot(
    r2,
    col = cols,
    legend = FALSE,
    axes = FALSE,
    box = FALSE,
    colNA = "white",      # FUNDO BRANCO REAL
    main = title
  )

  legend(
    "right",
    legend = c("Não Floresta", "Floresta"),
    fill = cols,
    border = NA,
    cex = 1.2,
    bty = "n",
    xpd = TRUE,
    inset = c(-0.15, 0),
    title = "Classes"
  )

  invisible(r2)
}
plot_floresta_10m(
  "C:/Users/Pedro/Desktop/OTNuEC/R/data/mapbiomas_near_1000m.tif"
)
```



[90]:
```r
library(terra)
library(dplyr)

resumo_mapbiomas <- function(tif_path) {
  cat("\n==== Lendo raster ====\n")
  r <- rast(tif_path)

  cat("\n==== Metadados ====\n")
  print(r)

  # ---- Valores únicos ----
  vals <- unique(r[])
  vals <- sort(na.omit(as.numeric(vals)))

  cat("\n==== Valores únicos encontrados ====\n")
  print(vals)

  # ---- Pixel resolution ----
  res_m <- res(r)[1]                # tamanho do pixel em metros
  pixel_area_km2 <- (res_m^2) / 1e6

  # ---- Contagem de pixels ----
  cat("\n==== Contagem de pixels por classe ====\n")
  tab <- freq(r, digits = 0)
  tab <- tab[!is.na(tab$value), ]  # remove NA
  print(tab)

  # ---- Área por classe ----
  tab <- tab %>%
    mutate(
```

```
      area_km2 = count * pixel_area_km2,
      perc = 100 * area_km2 / sum(area_km2)
    ) %>%
    arrange(desc(area_km2))

  cat("\n==== Área por classe (km²) ====\n")
  print(tab)

  return(tab)
}
resumo <- resumo_mapbiomas(
  "C:/Users/Pedro/Desktop/OTNuEC/R/data/mapbiomas_near_1000m.tif"
)
```

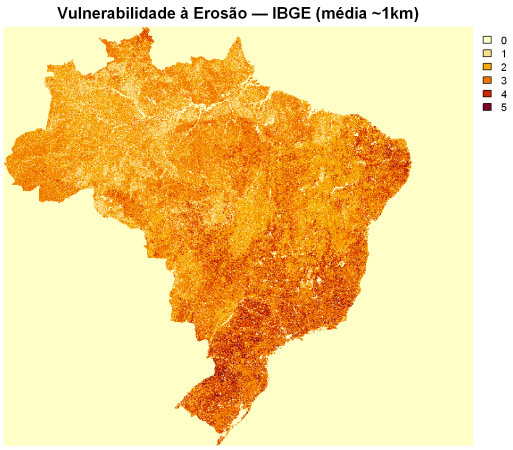
```
==== Lendo raster ====

==== Metadados ====
class       : SpatRaster
size        : 4757, 4530, 1  (nrow, ncol, nlyr)
resolution  : 0.009235355, 0.008996598  (x, y)
extent      : -74.8974, -33.06124, -34.85203, 7.94479  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : mapbiomas_near_1000m.tif
name        : mapbiomas_10m_collection2_integration_v1-classification_2023
min value   :                                                              0
max value   :                                                             50

==== Valores únicos encontrados ====
 [1]  0  3  4  5  6  9 11 12 15 19 21 23 24 25 29 30 31 32 33 36 49 50

==== Contagem de pixels por classe ====
   layer value    count
1      1     0 12950531
2      1     3  3504377
3      1     4  1073736
4      1     5    10059
5      1     6   347095
6      1     9   114506
7      1    11   254644
8      1    12   257428
9      1    15  1650241
10     1    19   600767
11     1    21   429846
12     1    23     4845
13     1    24    42215
14     1    25    49936
15     1    29    18655
```

```
16     1    30     3618
17     1    31      665
18     1    32      533
19     1    33   210844
20     1    36    15270
21     1    49     6720
22     1    50     2679

==== Área por classe (km²) ====
   layer value    count     area_km2         perc
1      1     0 12950531 1.104574e-03 60.097474571
2      1     3  3504377 2.988946e-04 16.262206364
3      1    15  1650241 1.407520e-04  7.658011593
4      1     4  1073736 9.158086e-05  4.982716304
5      1    19   600767 5.124049e-05  2.787884103
6      1    21   429846 3.666233e-05  1.994718136
7      1     6   347095 2.960435e-05  1.610708699
8      1    12   257428 2.195649e-05  1.194605278
9      1    11   254644 2.171904e-05  1.181686011
10     1    33   210844 1.798326e-05  0.978430300
11     1     9   114506 9.766422e-06  0.531369827
12     1    25    49936 4.259131e-06  0.231730073
13     1    24    42215 3.600593e-06  0.195900453
14     1    29    18655 1.591118e-06  0.086569299
15     1    36    15270 1.302406e-06  0.070861066
16     1     5    10059 8.579501e-07  0.046679205
17     1    49     6720 5.731608e-07  0.031184438
18     1    23     4845 4.132387e-07  0.022483423
19     1    30     3618 3.085857e-07  0.016789479
20     1    50     2679 2.284967e-07  0.012432010
21     1    31      665 5.671904e-08  0.003085960
22     1    32      533 4.546052e-08  0.002473409
```

11.2 10.2) IBGE vulnerability

[91]:
```r
library(terra)
plot_erosao <- function(path) {
    r <- rast(path)

    pal <- hcl.colors(30, "YlOrRd", rev = TRUE)

    plot(
        r,
        col = pal,
        main = "Vulnerabilidade à Erosão - IBGE (média ~1km)",
        axes = FALSE,
        box  = FALSE
```

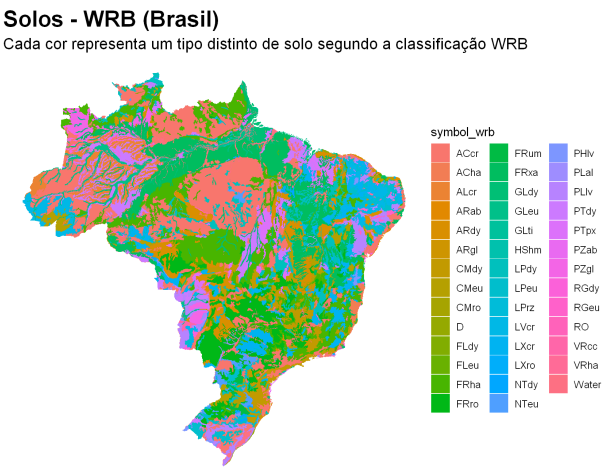
```
    )
}

plot_erosao("data/erosao_mean_1km.tif")
```



11.3 10.3) Embrapa soil

[92]:
```
library(sf)
library(ggplot2)
library(dplyr)

plot_solos_wrb <- function(shp_path) {

  # 1. Ler shapefile
  soils <- sf::st_read(shp_path, quiet = TRUE)

  # 2. Remover coluna de geometria para inspecionar atributos
  attrs <- soils |> st_drop_geometry()

  # 3. Identificar colunas categóricas (factor ou character)
  cat_cols <- names(attrs)[sapply(attrs, \(x) is.factor(x) || is.character(x))]

  if (length(cat_cols) == 0)
    stop("Nenhuma coluna categórica encontrada no shapefile!")

  # Usar a primeira coluna categórica
  col_to_plot <- cat_cols[1]

  # 4. Converter para fator
  soils[[col_to_plot]] <- factor(soils[[col_to_plot]])
```

```r
  # 5. Criar paleta
  n <- nlevels(soils[[col_to_plot]])
  pal <- scales::hue_pal()(n)

  # 6. Plot bonito estilo MapBiomas
  ggplot(soils) +
    geom_sf(aes(fill = .data[[col_to_plot]]), color = NA) +
    scale_fill_manual(values = pal, name = col_to_plot) +
    labs(
      title = "Solos - WRB (Brasil)",
      subtitle = "Cada cor representa um tipo distinto de solo segundo a␣
 ↪classificação WRB"
    ) +
    theme_void() +
    theme(
      legend.position = "right",
      plot.title = element_text(size = 20, face = "bold"),
      plot.subtitle = element_text(size = 14)
    )
}
shp_path <- "soils_brazil_wrb_wgs84.shp"
plot_solos_wrb(shp_path)
```



12 11) Appendix

12.1 11.1) Mapbiomas compact

Sanity on full data

[93]:
```r
# file: R/sanity_mapbiomas_fix.R
suppressPackageStartupMessages(library(terra))

# --- helpers (por que: compatibilidade e precisão métrica) ---
is_ll <- function(x) isTRUE(tryCatch(is.lonlat(x), error = function(e) NA))

dtype_terra <- function(x) {
  # terra >= 1.5 usa `datatype()`. Fallback tenta `dataType()`.
  out <- tryCatch(datatype(x), error = function(e) NA_character_)
  if (is.na(out)) out <- tryCatch(dataType(x), error = function(e)␣
 ↪NA_character_)
  out
}

epsg_code <- function(r) {
  d <- tryCatch(crs(r, describe = TRUE), error = function(e) NULL)
  if (is.null(d)) return(NA_character_)
  if (!is.null(d$code)) return(as.character(d$code))
  NA_character_
}

res_meters <- function(r) {
  rs <- res(r)
  if (!is_ll(r)) return(setNames(rs, c("x_m","y_m")))
  yc  <- (ymin(r) + ymax(r)) / 2
  x_m <- rs[1] * 111320 * cos(yc * pi/180)
  y_m <- rs[2] * 111132
  setNames(c(x_m, y_m), c("x_m","y_m"))
}

pretty_bytes <- function(b) {
  if (is.na(b)) return(NA_character_)
  u <- c("B","KB","MB","GB","TB"); i <- 1
  while (b >= 1024 && i < length(u)) { b <- b/1024; i <- i+1 }
  sprintf("%.3f %s", b, u[i])
}

categorical_guess <- function(dtype, sample_vals) {
  sample_vals <- sample_vals[!is.na(sample_vals)]
  is_int_dtype <- grepl("^(INT|BYTE)", dtype, ignore.case = TRUE)
  all_integer  <- length(sample_vals) > 0 &&
    all(abs(sample_vals - round(sample_vals)) < .Machine$double.eps)
  nuniq <- length(unique(sample_vals))
  list(is_categorical = (is_int_dtype || (all_integer && nuniq <= 512)),
       unique_in_sample = nuniq)
}
```

```r
suggest_output_dtype <- function(vals, categorical) {
  vals <- vals[!is.na(vals)]
  if (!length(vals)) return(if (categorical) "INT2U" else "FLT4S")
  vmin <- suppressWarnings(min(vals))
  vmax <- suppressWarnings(max(vals))
  if (categorical) {
    if (vmin >= 0 && vmax <= 255)   return("INT1U")
    if (vmin >= 0 && vmax <= 65535) return("INT2U")
    return("INT4U")
  }
  "FLT4S"
}

agg_table_for_targets <- function(r, targets_m) {
  rm <- res_meters(r)
  fx <- pmax(1L, as.integer(round(targets_m / rm["x_m"])))
  fy <- pmax(1L, as.integer(round(targets_m / rm["y_m"])))
  data.frame(
    alvo_m   = as.integer(targets_m),
    fator_x  = as.integer(fx),
    fator_y  = as.integer(fy),
    fator    = as.integer(pmax(fx, fy)),
    ncol_out = as.integer(ceiling(ncol(r) / fx)),
    nrow_out = as.integer(ceiling(nrow(r) / fy)),
    outsize_x_percent = round(100/fx, 3),
    outsize_y_percent = round(100/fy, 3),
    stringsAsFactors = FALSE
  )
}

# --- SANITY principal ---
#' Diagnóstico rápido de um raster grande (sem varrer tudo)
#' @param path caminho do GeoTIFF
#' @param sample_n número de pixels para amostrar (rápido)
#' @param targets_m resoluções alvo em metros (vetor)
#' @param seed semente para reprodutibilidade
#' @return lista invisível com todos os resultados
raster_sanity <- function(path,
                          sample_n = 200000L,
                          targets_m = c(300L, 500L, 1000L),
                          seed = 1L) {
  stopifnot(file.exists(path))
  r <- rast(path)

  finfo <- file.info(path)
  rm <- res_meters(r)
  dtype <- dtype_terra(r)
```

```
epsg  <- epsg_code(r)

# amostra rápida (1ª banda)
set.seed(seed)
v <- tryCatch({
  vv <- spatSample(r, size = sample_n, method = "random",
                   values = TRUE, na.rm = FALSE, warn = FALSE)
  # pega a primeira coluna de valores, qualquer que seja o nome
  as.numeric(vv[[1]])
}, error = function(e) numeric())

na_pct   <- if (length(v)) round(mean(is.na(v))*100, 3) else NA_real_
v_no_na  <- v[!is.na(v)]
top_tbl  <- if (length(v_no_na)) sort(table(v_no_na), decreasing = TRUE) else␣
↪integer()
guess    <- categorical_guess(dtype, v)
outdtype <- suggest_output_dtype(v, guess$is_categorical)
aggs     <- agg_table_for_targets(r, unique(as.integer(targets_m)))

cat("\n===== SANITY =====\n")
cat("Arquivo: ", normalizePath(path, winslash="/"), "\n", sep = "")
cat("Tamanho: ", pretty_bytes(finfo$size), "\n", sep = "")
cat(sprintf("Camadas: %d | Dims: %d x %d (%.1f M células)\n",
            nlyr(r), ncol(r), nrow(r), ncell(r)/1e6))
cat(sprintf("CRS definido: %s | EPSG: %s | lon/lat: %s\n",
            !is.na(crs(r)), ifelse(is.na(epsg), "NA", epsg), is_ll(r)))
cat(sprintf("Res (CRS): %s\n", paste(signif(res(r), 8), collapse=" x ")))
cat(sprintf("Res aprox (m): x=%.2f  y=%.2f | lat_centro=%.5f\n",
            rm["x_m"], rm["y_m"], (ymin(r)+ymax(r))/2))
cat(sprintf("Datatype: %s | Tem levels?: %s\n",
            ifelse(is.na(dtype), "NA", dtype), any(lengths(levels(r)) > 0)))
if (length(v)) {
  cat(sprintf("Amostra: %d valores | NA: %.3f%% | min/max(amostra)= %s /␣
↪%s\n",
              length(v), na_pct,
              ifelse(length(v_no_na), as.character(min(v_no_na)), "NA"),
              ifelse(length(v_no_na), as.character(max(v_no_na)), "NA")))
  cat(sprintf("Categórico (guess): %s | únicos na amostra: %d\n",
              guess$is_categorical, guess$unique_in_sample))
  cat("\nTop 15 códigos (amostra):\n"); print(head(top_tbl, 15))
  cat(sprintf("\nSugestão datatype saída: %s (use LZW + TILED)\n", outdtype))
} else {
  cat("Amostra indisponível (seguindo só com metadados).\n")
}
cat("\nAlvos e fatores sugeridos:\n"); print(aggs)
cat("Obs: use -outsize X% Y% do GDAL com outsize_x_percent /␣
↪outsize_y_percent.\n")
```

```
  gt <- Sys.which("gdal_translate"); gw <- Sys.which("gdalwarp")
  cat(sprintf("\nGDAL: gdal_translate=%s | gdalwarp=%s\n",
              ifelse(nzchar(gt),"SIM","NÃO"), ifelse(nzchar(gw),"SIM","NÃO")))
  cat("==================\n\n")

  invisible(list(
    path = normalizePath(path, winslash="/"),
    size_bytes = finfo$size,
    dims = c(ncol=ncol(r), nrow=nrow(r), nlyr=nlyr(r), ncells=ncell(r)),
    crs = as.character(crs(r)),
    epsg = epsg,
    lonlat = is_ll(r),
    res_crs = res(r),
    res_m   = unname(rm),
    dtype   = dtype,
    sample  = list(n=length(v), na_pct=na_pct,
                   min=if(length(v_no_na)) min(v_no_na) else NA_real_,
                   max=if(length(v_no_na)) max(v_no_na) else NA_real_,
                   top=top_tbl),
    categorical_guess = guess,
    out_dtype_suggest = outdtype,
    targets = aggs,
    gdal = list(gdal_translate = nzchar(gt), gdalwarp = nzchar(gw))
  ))
}

# ---- exemplo de uso ----
 sanity <- raster_sanity(
   "mapbiomas_10m_collection2_integration_v1-classification_2023.tif",
   sample_n = 200000L,
   targets_m = c(300L, 500L, 1000L)
 )
```

```
===== SANITY =====
Arquivo: C:/Users/Pedro/Desktop/OTNuEC/R/mapbiomas_10m_collection2_integration_v
1-classification_2023.tif
Tamanho: 5.122 GB
Camadas: 1 | Dims: 465718 x 476412 (221873.6 M células)
CRS definido: TRUE | EPSG: 4326 | lon/lat: TRUE
Res (CRS): 8.9831528e-05 x 8.9831528e-05
Res aprox (m): x=9.73  y=9.98 | lat_centro=-13.45362
Datatype: INT1U | Tem levels?: TRUE
Amostra: 200000 valores | NA: 0.000% | min/max(amostra)= 0 / 50
Categórico (guess): TRUE | únicos na amostra: 22

Top 15 códigos (amostra):
v_no_na
```

```
     0      3     15      4     19     21      6     12     11     33      9
118144  33925  15501  10382   5513   3984   3302   2418   2406   1931   1037
    25     24     29     36
   513    355    173    152

Sugestão datatype saída: INT1U (use LZW + TILED)

Alvos e fatores sugeridos:
  alvo_m fator_x fator_y fator ncol_out nrow_out outsize_x_percent
1    300      31      30    31    15024    15881             3.226
2    500      51      50    51     9132     9529             1.961
3   1000     103     100   103     4522     4765             0.971
  outsize_y_percent
1             3.333
2             2.000
3             1.000
Obs: use -outsize X% Y% do GDAL com outsize_x_percent / outsize_y_percent.

GDAL: gdal_translate=NÃO | gdalwarp=NÃO
====================
```

Compressing the data:

[94]:
```
# file: R/compactar_mapbiomas_hyperfast_fix2.R
#suppressPackageStartupMessages(library(terra))

# ---- utils ----
#is_ll <- function(x) isTRUE(tryCatch(is.lonlat(x), error = function(e) NA))
#deg_from_m <- function(r, target_m) {
#  yc <- (ymin(r) + ymax(r)) / 2
#  c(deg_x = target_m / (111320 * cos(yc * pi/180)),
#    deg_y = target_m / 111132)
#}
#make_template <- function(r, deg_x, deg_y) {
#  ex <- ext(r)
#  ncol_out <- as.integer(ceiling((xmax(ex) - xmin(ex)) / deg_x))
#  nrow_out <- as.integer(ceiling((ymax(ex) - ymin(ex)) / deg_y))
#  rast(xmin = xmin(ex), xmax = xmax(ex), ymin = ymin(ex), ymax = ymax(ex),
#       ncol = ncol_out, nrow = nrow_out, crs = crs(r))
#}
#locate_gdal <- function() {
#  first_ok <- function(v) { v <- v[file.exists(v) & nzchar(v)]; if (length(v))␣
 ↪normalizePath(v[1], winslash="/") else "" }
#  t_candidates <- c(Sys.which("gdal_translate"),
#                    Sys.glob("C:/OSGeo4W*/bin/gdal_translate.exe"),
#                    Sys.glob("C:/Program Files/QGIS*/bin/gdal_translate.exe"))
#  a_candidates <- c(Sys.which("gdaladdo"),
```

```
#                    Sys.glob("C:/OSGeo4W*/bin/gdaladdo.exe"),
#                    Sys.glob("C:/Program Files/QGIS*/bin/gdaladdo.exe"))
#  list(translate = first_ok(t_candidates), addo = first_ok(a_candidates))
#}
#wopt_int2 <- list(
#  datatype = "INT2U",  # por quê: seus códigos >255
#  gdal = c("COMPRESS=LZW","PREDICTOR=2","TILED=YES",
#           "BLOCKXSIZE=512","BLOCKYSIZE=512","BIGTIFF=IF_SAFER")
#)

# ---- principal ----
#compactar_fast <- function(in_path, out_path, target_m = 1000L, aoi_path = "")␣
 ↪{
#  stopifnot(file.exists(in_path))
#  dir.create(dirname(out_path), recursive = TRUE, showWarnings = FALSE)
#  terraOptions(memfrac = 0.9, todisk = TRUE)

#  r0 <- rast(in_path)
#  if (!is_ll(r0)) stop("Este runner assume WGS84 (graus).")
#  if (nzchar(aoi_path) && file.exists(aoi_path)) r0 <- crop(r0,␣
 ↪vect(aoi_path), mask = TRUE)

#  td <- deg_from_m(r0, target_m)
#  ex <- ext(r0)
#  ncol_out <- ceiling((xmax(ex) - xmin(ex)) / td["deg_x"])
#  nrow_out <- ceiling((ymax(ex) - ymin(ex)) / td["deg_y"])
#  outsize_x <- (ncol_out / ncol(r0)) * 100
#  outsize_y <- (nrow_out / nrow(r0)) * 100

#  g <- locate_gdal()
#  if (nzchar(g$translate)) {
#    message(sprintf("GDAL: gdal_translate (nearest) → outsize %.6f%% x %.
 ↪6f%%", outsize_x, outsize_y))
#    if (nzchar(g$addo)) {
#      ovs <- c(2,4,8,16,32,64,128,256)
#      try(system2(g$addo, c("-r","nearest", shQuote(in_path), paste(ovs)),␣
 ↪stdout = "", stderr = ""), silent = TRUE)
#    }
#    args <- c("-r","nearest",
#              "-outsize", sprintf("%.6f%%", outsize_x), sprintf("%.6f%%",␣
 ↪outsize_y),
#              "-co","COMPRESS=LZW","-co","TILED=YES","-co","BIGTIFF=IF_SAFER",
#              shQuote(in_path), shQuote(out_path))
#    st <- system2(g$translate, args, stdout = "", stderr = "")
#    if (!identical(st, 0L)) stop("gdal_translate falhou.")
#    return(normalizePath(out_path, winslash="/"))
#  }
```

```
#  message("GDAL não encontrado: usando terra::resample('near')")
#  tmpl <- make_template(r0, td["deg_x"], td["deg_y"])
#  # IMPORTANTE: escrever apenas uma vez (evita 'source and target filename␣
 ↪cannot be the same')
#  resample(r0, tmpl, method = "near",
#           filename = out_path, overwrite = TRUE, wopt = wopt_int2)
#  normalizePath(out_path, winslash="/")
#}

# ---- exemplo ----
# out <- compactar_fast(
#   in_path  = "C:/Users/Pedro/Desktop/OTNuEC/R/
 ↪mapbiomas_10m_collection2_integration_v1-classification_2023.tif",
#   out_path = "data/mapbiomas_near_1000m.tif",
#   target_m = 1000L
# )
# cat("Saída:", out, "\n")
```

12.2 11.2) Erosao IBGE compact

[95]:
```
#library(terra)
#compactar_erosao <- function(in_path, out_path, alvo_m = 1000) {
#
#    r <- rast(in_path)
#
#    # Supondo que está em graus (WGS84) como no IBGE
#    if (is.lonlat(r)) {
#        # converter graus para metros
#        yc <- (ymin(r) + ymax(r))/2
#        px_m <- 111320 * cos(yc*pi/180)
#    } else {
#        px_m <- res(r)[1]
#    }
#
#    fact <- max(1, round(alvo_m / px_m))

#    cat("Res original (m):", px_m, "| Fator:", fact, "\n")

#    r_agg <- aggregate(
#        r,
#        fact = fact,
#        fun = mean,
#        na.rm = TRUE,
#        filename = out_path,
#        overwrite = TRUE,
#        wopt = list(
```

```
#             datatype = "FLT4S",
#             gdal = c("COMPRESS=LZW", "PREDICTOR=2", "TILED=YES")
#         )
#     )
#
#     return(r_agg)
#}

#eros_simplif <- compactar_erosao(
#     "Brasil_vulnerabilidade_2020/BR_vulnerabilidade_solos_erosao_2020.tif",
#     "data/erosao_mean_1km.tif",
#     alvo_m = 1000
#)
```

[96]:
```
sanity_raster <- function(path, sample_n = 100000, seed = 123) {
    r <- rast(path)

    cat("\n===== SANITY -", basename(path), "=====\n")
    cat("Tamanho:", round(file.info(path)$size/1024^2, 2), "MB\n")
    cat("Dim:", ncol(r), "x", nrow(r), "pixels\n")
    cat("Res:", paste(res(r), collapse=" x "), "(graus)\n")

    # resolução aproximada em metros
    if (is.lonlat(r)) {
        yc <- (ymin(r) + ymax(r))/2
        res_m <- c(
            res(r)[1] * 111320 * cos(yc*pi/180),
            res(r)[2] * 111132
        )
    } else {
        res_m <- res(r)
    }

    cat("Res (m):", round(res_m, 2), "\n")
    cat("CRS:", crs(r), "\n")
    cat("Extent:", as.vector(ext(r)), "\n\n")

    # Amostra ------------------------------------------------
    set.seed(seed)
    v <- spatSample(r, size = sample_n, method = "random", values = TRUE)[,1]

    cat("Amostra:", length(v), "pixels\n")
    cat(sprintf("NA: %.3f%%\n", mean(is.na(v)) * 100))

    v2 <- v[!is.na(v)]
    cat("Min-Max:", min(v2), "→", max(v2), "\n")
    cat("Média:", mean(v2), "\n")
```

```r
    cat("Desvio-padrão:", sd(v2), "\n")

    # Histograma textual
    h <- hist(v2, plot = FALSE, breaks = 20)
    cat("\nHistograma (20 classes):\n")
    print(data.frame(mid = h$mids, freq = h$counts))

    cat("\n============================\n\n")
}
sanity_raster("data/erosao_mean_1km.tif")
```

```
===== SANITY - erosao_mean_1km.tif =====
Tamanho: 3159.8 MB
Dim: 167557 x 144805 pixels
Res: 0.000269494585235856 x 0.000269494585235856 (graus)
Res (m): 29.08 29.95
CRS: GEOGCRS["WGS 84",
    ENSEMBLE["World Geodetic System 1984 ensemble",
        MEMBER["World Geodetic System 1984 (Transit)"],
        MEMBER["World Geodetic System 1984 (G730)"],
        MEMBER["World Geodetic System 1984 (G873)"],
        MEMBER["World Geodetic System 1984 (G1150)"],
        MEMBER["World Geodetic System 1984 (G1674)"],
        MEMBER["World Geodetic System 1984 (G1762)"],
        MEMBER["World Geodetic System 1984 (G2139)"],
        MEMBER["World Geodetic System 1984 (G2296)"],
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]],
        ENSEMBLEACCURACY[2.0]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    USAGE[
        SCOPE["Horizontal component of 3D system."],
        AREA["World."],
        BBOX[-90,-180,90,180]],
    ID["EPSG",4326]]
Extent: -73.99109 -28.83538 -33.75177 5.272392

Amostra: 100000 pixels
NA: 0.000%
```

```
Min-Max: 0 → 5
Média: 0.97177
Desvio-padrão: 1.345151

Histograma (20 classes):
   mid  freq
1  0.1 61294
2  0.3     0
3  0.5     0
4  0.7     0
5  0.9  4535
6  1.1     0
7  1.3     0
8  1.5     0
9  1.7     0
10 1.9 15311
11 2.1     0
12 2.3     0
13 2.5     0
14 2.7     0
15 2.9 14308
16 3.1     0
17 3.3     0
18 3.5     0
19 3.7     0
20 3.9  3664
21 4.1     0
22 4.3     0
23 4.5     0
24 4.7     0
25 4.9   888

============================
```