```
# ============================================
# MISO × Counties intersection → CSV
# ============================================
# Output columns:
#   fips_code, county, state,
#   area_county_km2, area_in_miso_km2,
#   share_in_miso, percent_in_miso
#
# Usage:
#   - Option A (URLs): just run as-is; it will fetch MISO via FeatureServer
#   - Option B (local files): set paths below for MISO and/or Counties
#
install.packages(c("sf","dplyr","tigris","janitor","stringr","lwgeom","readr"))
suppressPackageStartupMessages({
  library(sf)
  library(dplyr)
  library(tigris)
  library(janitor)
  library(stringr)
  library(lwgeom)
  library(readr)
})

options(tigris_use_cache = TRUE)

# ----------------------------
# Config: choose your inputs
# ----------------------------
miso_local_path    <- "MISO.json"       # e.g., "MISO.geojson" or "MISO.json"
counties_local_path <- "Counties.geojson"  # e.g., "Counties.geojson"

# ArcGIS FeatureServer query for MISO only (RTO Regions, Layer 256 on EIA Atlas)
miso_query_url <- paste0(
  "https://services7.arcgis.com/FGr1D95XCGALKXqM/arcgis/rest/services/",
  "RTO_Regions/FeatureServer/256/query",
  "?where=", utils::URLencode("RTO_ISO = 'MISO'"),
  "&outFields=*",
  "&outSR=4326",
  "&f=geojson"
)

# Output path
out_csv <- "miso_county_overlap_with_areas_calculated.csv"
```

```
# ----------------------------
# 1) Load MISO geometry
# ----------------------------
read_miso <- function() {
 if (!is.na(miso_local_path) && file.exists(miso_local_path)) {
  message("Reading MISO from local file: ", miso_local_path)
  miso <- st_read(miso_local_path, quiet = TRUE)
 } else {
  message("Reading MISO from FeatureServer (filtered to RTO_ISO = 'MISO')...")
  miso <- tryCatch(
   st_read(miso_query_url, quiet = TRUE),
   error = function(e) stop("Could not read MISO from URL. Set miso_local_path to a local
GeoJSON. ", e$message)
  )
 }

 miso <- miso %>% clean_names()
 # Handle field variants (sometimes rto_iso vs rto_iso_name)
 if ("rto_iso" %in% names(miso)) {
  miso <- dplyr::filter(miso, rto_iso == "MISO")
 } else if ("rto_iso_name" %in% names(miso)) {
  miso <- dplyr::filter(miso, str_detect(rto_iso_name, "(?i)\\bmiso\\b"))
 }
 if (nrow(miso) == 0) stop("No MISO features found.")

 # Valid + dissolve to a single geometry
 miso |> st_make_valid() |> st_union()
}

# ----------------------------
# 2) Load counties
# ----------------------------
read_counties <- function() {
 if (!is.na(counties_local_path) && file.exists(counties_local_path)) {
  message("Reading counties from local file: ", counties_local_path)
  ct <- st_read(counties_local_path, quiet = TRUE) |>
   clean_names()
  # Standardize essential fields: geoid, name, stusps
  if (!"geoid" %in% names(ct) && "GEOID" %in% names(ct)) {
   ct <- ct |> rename(geoid = GEOID)
  }
  if (!"name" %in% names(ct) && "NAME" %in% names(ct)) {
   ct <- ct |> rename(name = NAME)
  }
```

```
   if (!"stusps" %in% names(ct) && "STUSPS" %in% names(ct)) {
    ct <- ct |> rename(stusps = STUSPS)
   }
   ct
  } else {
   message("Downloading counties via tigris (2023 cartographic boundaries)...")
   tigris::counties(cb = TRUE, year = 2023, class = "sf") |>
    clean_names()
  }
}

miso    <- read_miso()
counties <- read_counties()

# Drop AK, HI, PR (lower 48 focus)
if ("statefp" %in% names(counties)) {
  counties <- counties |> filter(!statefp %in% c("02","15","72"))
} else if ("stusps" %in% names(counties)) {
  counties <- counties |> filter(!stusps %in% c("AK","HI","PR"))
}

# ----------------------------
# 3) Project to equal-area CRS
# ----------------------------
aea_crs <- 5070  # NAD83 / Conus Albers
counties_aea <- st_transform(counties, aea_crs) |> st_make_valid()
miso_aea    <- st_transform(miso,    aea_crs) |> st_make_valid()

# ----------------------------
# 4) Areas & intersection
# ----------------------------
# County total area (m^2)
counties_aea$area_total_m2 <- as.numeric(st_area(counties_aea))

# Intersections (counties ∩ MISO)
message("Computing intersections...")
inter <- suppressWarnings(st_intersection(counties_aea, miso_aea))
if (nrow(inter) == 0) stop("No county overlaps with MISO were found.")

inter$area_in_m2 <- as.numeric(st_area(inter))

# Determine key fields
id_field <- if ("geoid" %in% names(counties_aea)) "geoid" else if ("GEOID" %in%
names(counties_aea)) "GEOID" else NA
```

```
nm_field <- if ("name"  %in% names(counties_aea)) "name"  else if ("NAME"  %in%
names(counties_aea)) "NAME"  else NA
st_field <- if ("stusps"%in% names(counties_aea)) "stusps" else if ("STUSPS"%in%
names(counties_aea)) "STUSPS" else NA

if (is.na(id_field) || is.na(nm_field) || is.na(st_field)) {
  stop("Could not find expected county fields (GEOID/geoid, NAME/name,
STUSPS/stusps).")
}

# Sum overlap area by county
area_in_by_geoid <- inter |>
  st_drop_geometry() |>
  group_by(.data[[id_field]]) |>
  summarise(area_in_m2 = sum(area_in_m2, na.rm = TRUE), .groups = "drop") |>
  rename(geoid = !!id_field)

# ----------------------------
# 5) Build base table (out_raw)
# ----------------------------
out_raw <- counties_aea |>
  st_drop_geometry() |>
  transmute(
    geoid = .data[[id_field]],
    county = .data[[nm_field]],
    state  = .data[[st_field]],
    area_county_m2 = area_total_m2
  ) |>
  left_join(area_in_by_geoid, by = "geoid") |>
  mutate(
    area_in_m2       = coalesce(area_in_m2, 0),
    area_county_km2   = area_county_m2 / 1e6,
    area_in_miso_km2  = area_in_m2 / 1e6,
    share_in_miso     = ifelse(area_county_m2 > 0, area_in_m2 / area_county_m2, 0),
    percent_in_miso   = round(share_in_miso * 100, 4),
    fips_code         = sprintf("%05s", geoid),
    state_up          = toupper(state)
  ) |>
  # Keep only counties with some overlap
  filter(
    area_in_miso_km2 > 0,
    # Remove specific TX & MS counties regardless of area
    !(
      (state_up == "TX" & county %in% c(
```

```
      "Dallas","Ellis","Kaufman","Rockwall",
      "Collin","Colin","Denton","Tarrant",
      "Johnson","Henderson"
    )) |
    (state_up == "MS" & county %in% c(
      "Chickasaw","Monroe","Lee","Pontotoc","Clay"
    ))
  )
)

# ----------------------------
# 6) Split: IL vs non-IL, apply 10 km² only to non-IL
# ----------------------------

# Non-Illinois counties: apply 10 km² threshold
out_non_il <- out_raw |>
  filter(state_up != "IL", area_in_miso_km2 >= 10)

# All Illinois counties from the counties shapefile (even if no overlap)
il_all <- counties_aea |>
  st_drop_geometry() |>
  filter(toupper(.data[[st_field]]) == "IL") |>
  transmute(
    fips_code       = sprintf("%05s", .data[[id_field]]),
    county          = .data[[nm_field]],
    state           = "IL",
    area_county_km2  = as.numeric(area_total_m2) / 1e6
  )

# Illinois counties that *do* have overlap (from out_raw)
il_metrics <- out_raw |>
  filter(state_up == "IL") |>
  select(
    fips_code,
    area_in_miso_km2,
    share_in_miso,
    percent_in_miso
  )

# Left-join metrics onto the full IL county list:
# - Existing IL with overlap get their values
# - IL without overlap get NA for area_in_miso_km2, share_in_miso, percent_in_miso
out_il_complete <- il_all |>
  left_join(il_metrics, by = "fips_code")
```

```
# ----------------------------
# 7) Rejoin IL + non-IL for final output
# ----------------------------
out <- bind_rows(
  out_il_complete,
  out_non_il |>
    select(
      fips_code,
      county,
      state = state_up,
      area_county_km2,
      area_in_miso_km2,
      share_in_miso,
      percent_in_miso
    )
) |>
  arrange(desc(percent_in_miso), state, county)

# ----------------------------
# 8) Total MISO area (km^2) just for reference
# ----------------------------
miso_total_area_km2 <- as.numeric(st_area(miso_aea)) / 1e6
message(sprintf("Total MISO area (km^2): %.2f", miso_total_area_km2))

# ----------------------------
# 9) Write CSV
# ----------------------------
write_csv(out, out_csv)
message("Wrote: ", out_csv, " with ", nrow(out), " counties")
```