Data Preparation

2025-11-21

Here we do some basic data processing and prepare the spatial layers.

Table of contents

Load packages

```r
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.2
v ggplot2   4.0.0     v tibble    3.3.0
```

```
v lubridate 1.9.4     v tidyr     1.3.1
v purrr     1.1.0
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```r
library(sf)
```

```
Linking to GEOS 3.13.0, GDAL 3.8.5, PROJ 9.5.1; sf_use_s2() is TRUE
```

```r
library(terra)
```

```
terra 1.8.50

Attaching package: 'terra'

The following object is masked from 'package:tidyr':

    extract
```

```r
library(amt)
```

```
Attaching package: 'amt'

The following object is masked from 'package:stats':

    filter
```

Load data

```r
dingo_data <- read_csv("data/tanami_collars.csv") %>% filter(x > 0)
```

```
Rows: 61359 Columns: 53
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (13): dogname, sex, mine_away, gmt_date, gmt_time, cst_date, cst_month_...
dbl  (38): x, y, dog_id, attempt_id, fix_id, cstmonth, cstmonth_cnt, csthour...
```

```
time  (2): cst_time, timemaxwindgist

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

```r
dingo_data_sf <- st_as_sf(dingo_data, coords = c("x", "y"), crs = 4326)
```

Load shapefiles

```r
artificial_food <- st_read("spatial_layers/artificial_food/Art_Food_W84.shp")
```

```
Reading layer `Art_Food_W84' from data source
  `/Users/scottforrest/Library/CloudStorage/OneDrive-QueenslandUniversityofTechnology/PhD -
  using driver `ESRI Shapefile'
Simple feature collection with 3 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 129.6909 ymin: -20.53831 xmax: 130.3135 ymax: -19.98017
Geodetic CRS:  WGS 84
```

```r
artificial_water <- st_read("spatial_layers/artificial_water/ArtWater_W84v2.shp")
```

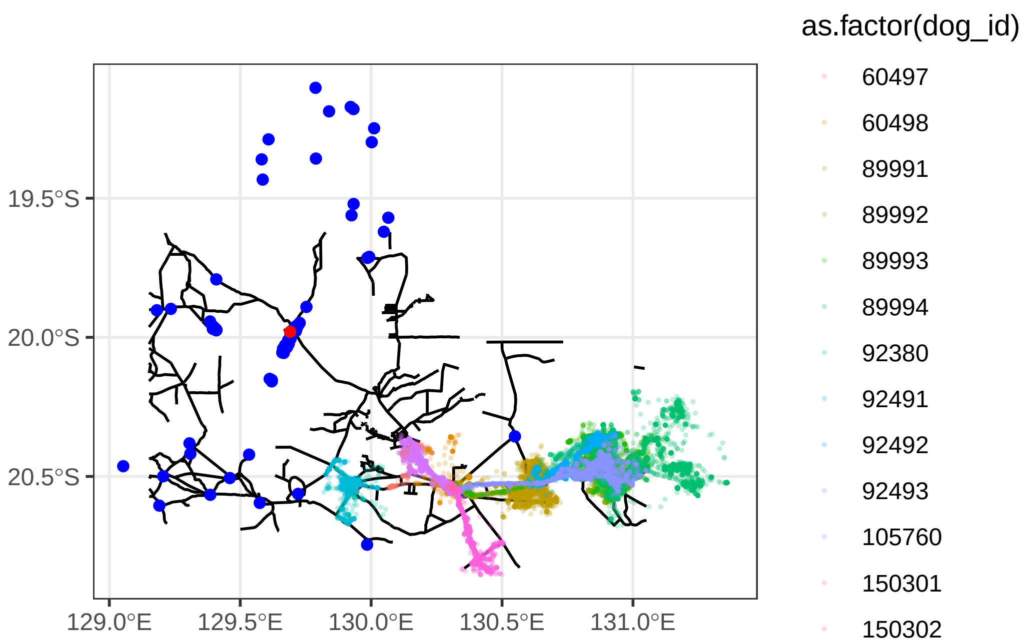
```
Reading layer `ArtWater_W84v2' from data source
  `/Users/scottforrest/Library/CloudStorage/OneDrive-QueenslandUniversityofTechnology/PhD -
  using driver `ESRI Shapefile'
Simple feature collection with 91 features and 16 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 129.0532 ymin: -20.74527 xmax: 130.9083 ymax: -19.10321
Geodetic CRS:  WGS 84
```

```r
roads <- st_read("spatial_layers/roads/RM_rds_trks_clip.shp") %>%
  st_transform(crs = st_crs(artificial_food))
```

```
Reading layer `RM_rds_trks_clip' from data source
  `/Users/scottforrest/Library/CloudStorage/OneDrive-QueenslandUniversityofTechnology/PhD -
  using driver `ESRI Shapefile'
Simple feature collection with 1004 features and 1 field
Geometry type: LINESTRING
Dimension:     XY
Bounding box:  xmin: 515871.5 ymin: 7696050 xmax: 713746.3 ymax: 7829996
Projected CRS: GDA94 / MGA zone 52
```

Plot the shapefiles

```
ggplot() +
  geom_sf(data = roads, color = "black") +
  geom_sf(data = artificial_water, color = "blue") +
  geom_sf(data = artificial_food, color = "red") +
  geom_sf(data = dingo_data_sf, aes(color = as.factor(dog_id)), size = 0.25, alpha = 0.25) +
  theme_bw()
```



Create an AOI for downloading DEM

We can go to this website https://elevation.fsdf.org.au/ to download a DEM for our area of interest (AOI). To do this, we first need to create a rectangle JSON that defines the AOI based on the GPS locations of the dingoes. We will add a buffer around the min and max coordinates to ensure we capture the full area.

In this chunk we create a function that can take a GPS dataset, pull out the location columns to find a max and min in both lat and long (or easting and northing), add a buffer, and create a rectangle polygon shapefile and geojson file.

```
# Function to create rectangle from coordinate bounds
create_rectangle_shapefile <- function(df, x_col = "x", y_col = "y",
                                       buffer = 0.05, crs = 4326,
                                       output_path = "rectangle") {
```

```
  # Calculate bounds with buffer
  min_x <- min(df[[x_col]], na.rm = TRUE) - buffer
  max_x <- max(df[[x_col]], na.rm = TRUE) + buffer
  min_y <- min(df[[y_col]], na.rm = TRUE) - buffer
  max_y <- max(df[[y_col]], na.rm = TRUE) + buffer

  # Create rectangle coordinates (clockwise from bottom-left)
  rectangle_coords <- matrix(c(
    min_x, min_y,  # bottom-left
    max_x, min_y,  # bottom-right
    max_x, max_y,  # top-right
    min_x, max_y,  # top-left
    min_x, min_y   # close the polygon
  ), ncol = 2, byrow = TRUE)

  # Create polygon geometry
  rectangle_poly <- st_polygon(list(rectangle_coords))

  # Create sf object
  rectangle_sf <- st_sf(
    id = 1,
    area = st_area(rectangle_poly),
    geometry = st_sfc(rectangle_poly, crs = crs)
  )

  rectangle_projected <- st_transform(rectangle_sf, crs = 4326)

  # Export as shapefile
  st_write(rectangle_projected, paste0(output_path, ".shp"),
           delete_dsn = TRUE, quiet = TRUE)

  # Also export as GeoJSON (alternative format)
  st_write(rectangle_projected, paste0(output_path, ".geojson"),
           delete_dsn = TRUE, quiet = TRUE)

  # Print summary
  cat("Rectangle created with bounds:\n")
  cat("X range:", min_x, "to", max_x, "\n")
  cat("Y range:", min_y, "to", max_y, "\n")
  cat("Files saved:", paste0(output_path, ".shp"), "and", paste0(output_path, ".geojson"), "

  return(rectangle_projected)
}
```

Use function to create shapefile and geojson

```
# Create the rectangle shapefile
dingo_extent <- create_rectangle_shapefile(
  df = dingo_data,
  x_col = "x",              # column name for easting/longitude
  y_col = "y",              # column name for northing/latitude
  buffer = 0.1,            # buffer distance in coordinate units (degrees or m)
  crs = 4326,               # coordinate reference system of GPS locations (will output in 43
  output_path = "spatial_layers/extent/dingo_extent" # without extension
)
```

```
Rectangle created with bounds:
X range: 129.7193 to 131.4602
Y range: -20.95433 to -20.09433
Files saved: spatial_layers/extent/dingo_extent.shp and spatial_layers/extent/dingo_extent.g
```

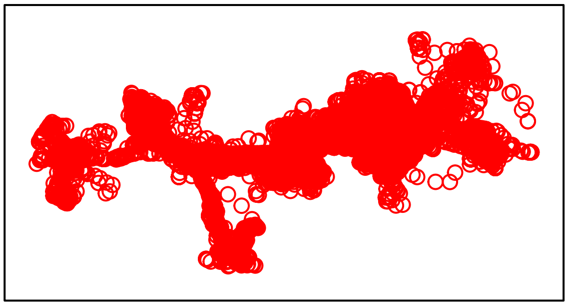
```
# View the result
print(dingo_extent)
```

```
Simple feature collection with 1 feature and 2 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 129.7194 ymin: -20.95433 xmax: 131.4602 ymax: -20.09433
Geodetic CRS:  WGS 84
  id     area                       geometry
1  1 1.497153 POLYGON ((129.7193 -20.9543...
```

```
# Create a sf object
dingo_extent_sf <- st_geometry(dingo_extent)

# Optional: Add your original points to the plot
plot(dingo_extent_sf, main = "Rectangle with Original Points")
plot(st_geometry(dingo_data_sf), add = TRUE, col = "red")
```

Rectangle with Original Points



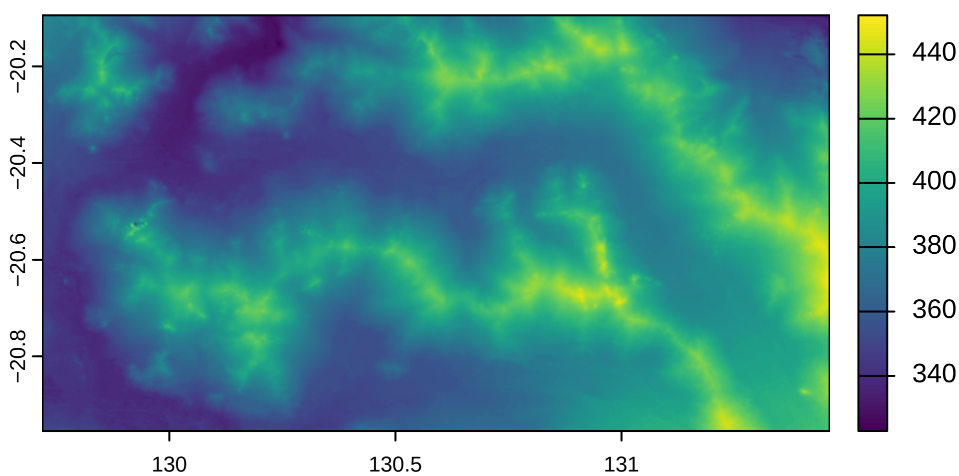
Check DEM

We downloaded the DEM from Geoscience Australia, which is a 1 second resolution DEM for Australia. We will use this to create slope and roughness layers. We will .

```
# read in the DEM
dem <- terra::rast("spatial_layers/topography/Hydro_Enforced_1_Second_DEM.tif") # hydro-enfo
dem
```

```
class       : SpatRaster
dimensions  : 3096, 6267, 1  (nrow, ncol, nlyr)
resolution  : 0.0002777778, 0.0002777778  (x, y)
extent      : 129.7193, 131.4601, -20.95431, -20.09431  (xmin, xmax, ymin, ymax)
coord. ref. : lon/lat WGS 84 (EPSG:4326)
source      : Hydro_Enforced_1_Second_DEM.tif
name        : Hydro_Enforced_1_Second_DEM
```

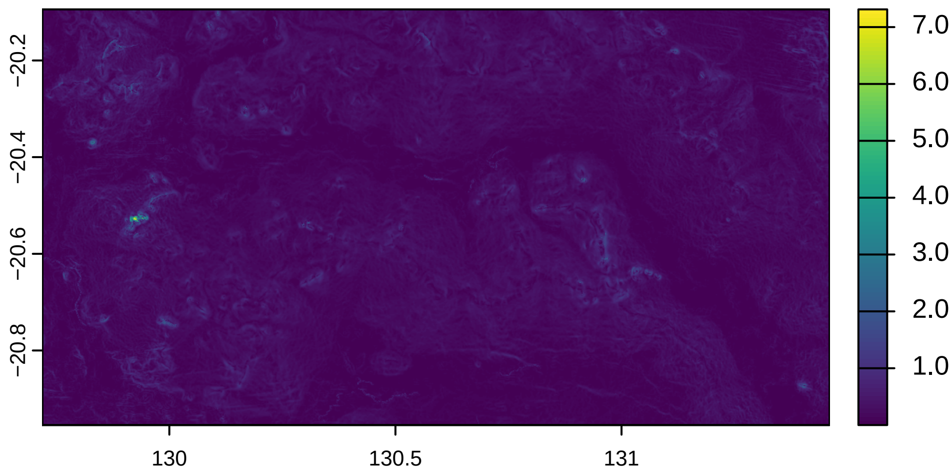
```
plot(dem)
```



Create slope layer

```r
# # create slope layer
# slope <- terra::terrain(dem, v = "slope", unit = "degrees", neighbors = 8)
# plot(slope)
# # save the layer
# terra::writeRaster(slope, "spatial_layers/topography/slope.tif", overwrite = T)

# load slope
slope <- terra::rast("spatial_layers/topography/slope.tif")
plot(slope)
```



Clip the spatial layers to the dingo extent

```r
# clip artificial food layer
artificial_food_clipped <- st_intersection(artificial_food, dingo_extent_sf)
```

```
Warning: attribute variables are assumed to be spatially constant throughout
all geometries
```

```r
# clip artificial water layer
artificial_water_clipped <- st_intersection(artificial_water, dingo_extent_sf)
```

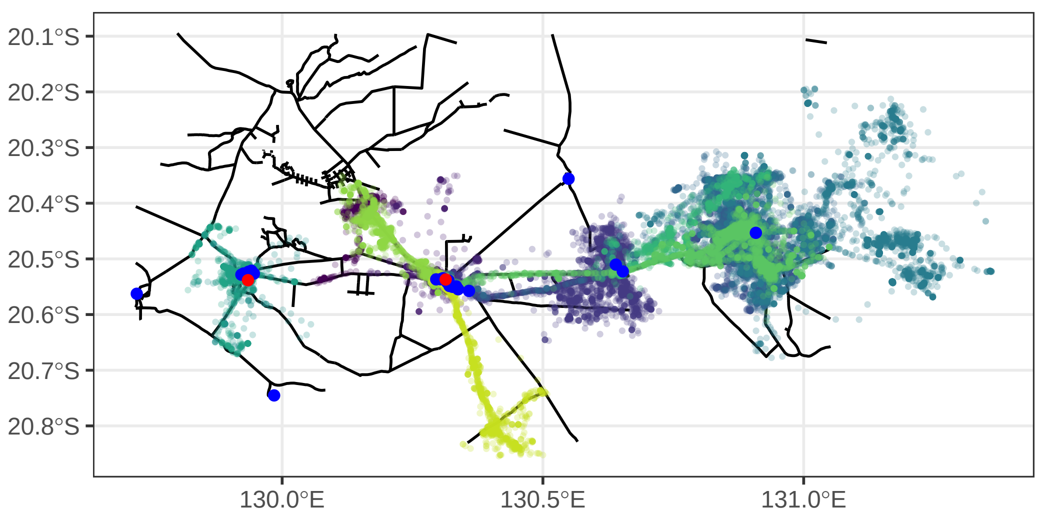
```
Warning: attribute variables are assumed to be spatially constant throughout
all geometries
```

```r
# clip the roads layer
roads_clipped <- st_intersection(roads, dingo_extent_sf)
```

```
Warning: attribute variables are assumed to be spatially constant throughout
all geometries
```

Plot the clipped layers

```r
ggplot() +
  geom_sf(data = roads_clipped, color = "black") +
  geom_sf(data = dingo_data_sf, aes(color = as.factor(dog_id)), size = 0.5, alpha = 0.25) +
  geom_sf(data = artificial_water_clipped, color = "blue") +
  geom_sf(data = artificial_food_clipped, color = "red") +
  scale_colour_viridis_d("Dingo ID") +
  theme_bw() +
  theme(legend.position = "none")
```




```r
ggsave("figures/gps_locations_vector_objects.png", width = 150, height = 100, units = "mm",
```

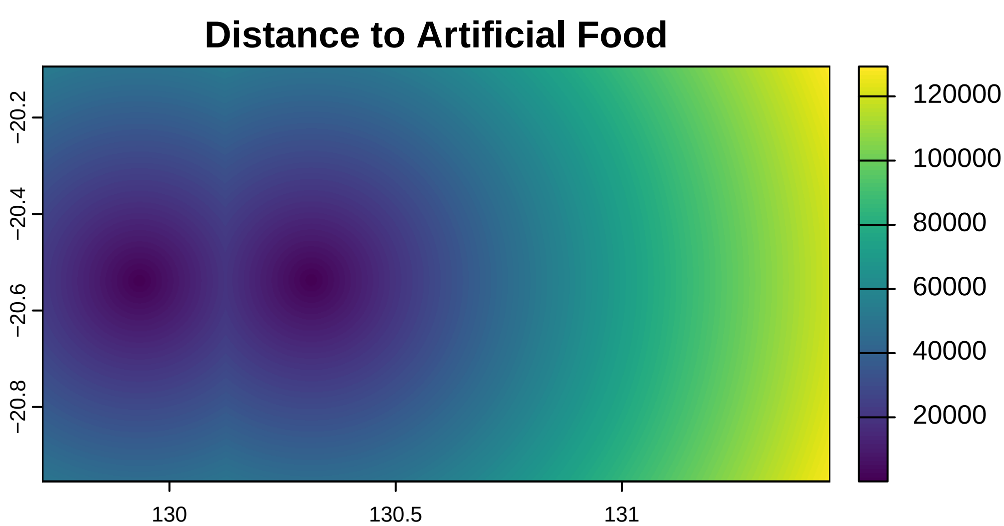
Create distance covariates

```r
# # create distance to artificial food layer
# artificial_food_raster <- terra::rasterize(vect(artificial_food_clipped), dem, field = 1,
# dist_artificial_food <- terra::distance(artificial_food_raster)
```

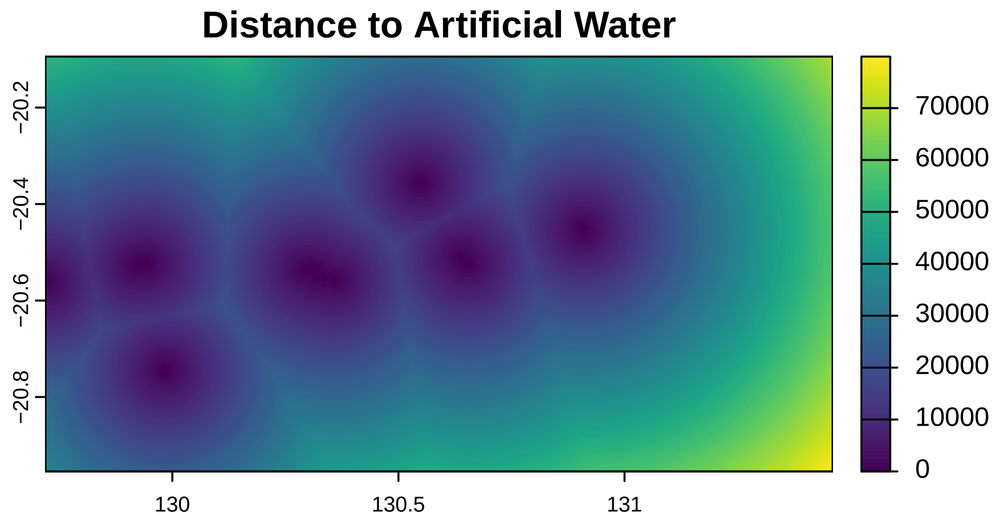
```
# plot(dist_artificial_food, main = "Distance to Artificial Food")
# terra::writeRaster(dist_artificial_food, "spatial_layers/artificial_food/dist_artificial_f
#
# # create distance to artificial water layer
# artificial_water_raster <- terra::rasterize(vect(artificial_water_clipped), dem, field = 1
# dist_artificial_water <- terra::distance(artificial_water_raster)
# plot(dist_artificial_water, main = "Distance to Artificial Water")
# terra::writeRaster(dist_artificial_water, "spatial_layers/artificial_water/dist_artificial
#
# # create distance to roads layer
# roads_raster <- terra::rasterize(vect(roads_clipped), dem, field = 1, background = NA)
# dist_roads <- terra::distance(roads_raster)
# plot(dist_roads, main = "Distance to Roads")
# terra::writeRaster(dist_roads, "spatial_layers/roads/dist_roads.tif", overwrite = T)
```

Load distance layers

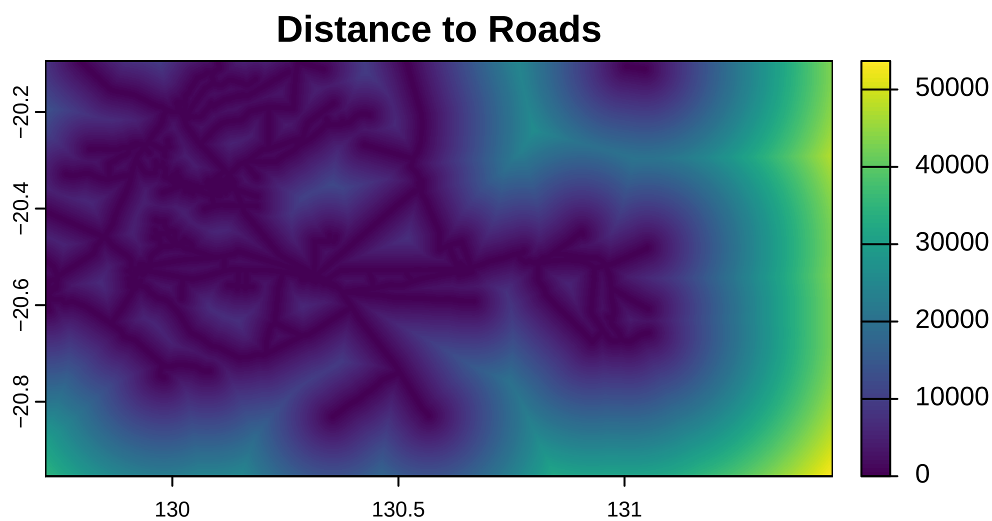
```
# load distance to artificial food layer
dist_artificial_food <- terra::rast("spatial_layers/artificial_food/dist_artificial_food.tif
plot(dist_artificial_food, main = "Distance to Artificial Food")
```




```
# load distance to artificial water layer
dist_artificial_water <- terra::rast("spatial_layers/artificial_water/dist_artificial_water.
plot(dist_artificial_water, main = "Distance to Artificial Water")
```




```r
# load distance to roads layer
dist_roads <- terra::rast("spatial_layers/roads/dist_roads.tif")
plot(dist_roads, main = "Distance to Roads")
```



References