

Creating maps in R - basics

Heidi Rautiainen

2024-12-20

Objectives

- Become familiar with R packages for processing and simple visualization of vector and raster data
- Perform common data transformation operations
- Create static maps (with ggplot2) and interactive maps (with mapview)

Contents

- Explore vector dataset
- Reference systems and transformations
- Customize maps (ggplot2)
- Explore raster dataset
- Save your map

Data used in this exercise: * Downloaded using “geodata”-package (exercise 1) * “create-maps/input_data”

Data availability

- using “rnatural-earth”-package. The “geodata”-package also provide open access data: GADM (<https://gadm.org>) country borders, landcover data from ESA WorldCover etc.
- raster and vector data downloaded from SLU server gis.slu.se/gisdata, and available from the GET download service at <http://maps.slu.se>. See GIS support at SLU for more information.

Prerequisites and setup

- Install packages
- Set up exercise structure: input and output folders

Exercise 1: Create study area map using “ggplot”

All datasets are found in the “data”-folder.

Here, we use the package ‘rnatural-earth’ to download countries in a ESRI shapefile format. These are available for download here: <https://www.naturalearthdata.com/downloads/10m-cultural-vectors/10m-admin-0-countries/>

Coordinate reference systems (CRS) CRS provide a standardized way of describing locations to describe geographic data. The CRS that is chosen depends on when the data was collected, the purpose of the data, etc. It is necessary to transform vector and raster data to a common CRS when data with different CRS are combined.

Setting CRS to SWEREF99 (Sweden). CRS can be referenced by its: 1) EPSG code, CRS(“+init=epsg:3006”) (see <http://www.epsg-registry.org/> and <http://spatialreference.org/>) or by 2) proj4string; “+proj=utm +zone=33 +ellps=GRS80 +towgs84=0,0,0,0,0,0 +units=m +no_defs”

There are two general options: 1) unprojected (a.k.a. Geographic): Latitude/Longitude for referencing location on the ellipsoid Earth (wgs84), and
2) projected: Easting/Northing for referencing location on 2D representations of Earth (the creation of maps) e.g., SWEREF99.

More reading: <https://www.nceas.ucsb.edu/sites/default/files/2020-04/OverviewCoordinateReferenceSystems.pdf>

Load vector data

```
## load country boarders -----
swe_no_sweref <- input_vector %>%
  grep(pattern = "swe_no", value = T) %>%
  sf:::st_read() %>%
  st_as_sf()

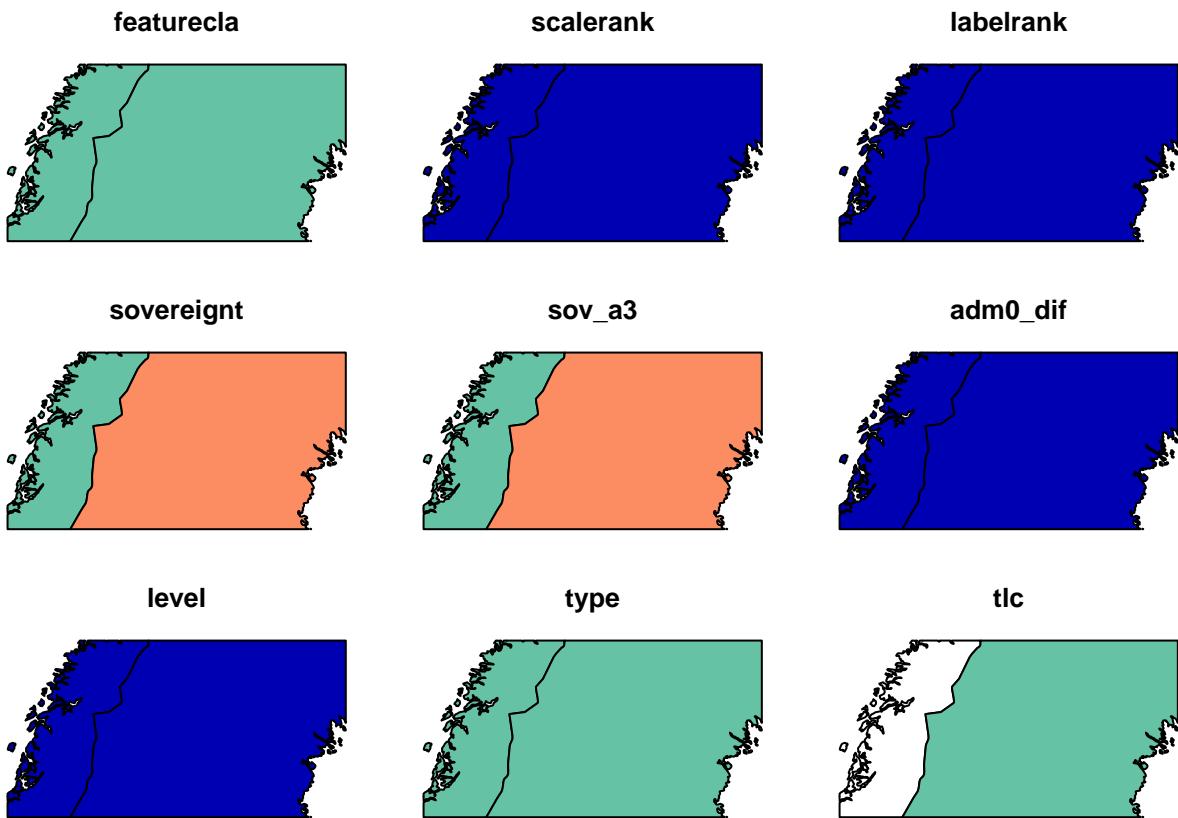
## Reading layer 'swe_no' from data source
##   '/Users/hirn0001/Library/CloudStorage/OneDrive-Sverigeslantbruksuniversitet/Undervisning/Teaching'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 2 features and 168 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:  xmin: -362704.5 ymin: -6097519 xmax: 1114145 ymax: 8972413
## Projected CRS: SWEREF99 TM

st_crs(swe_no_sweref)$proj4string

## [1] "+proj=utm +zone=33 +ellps=GRS80 +towgs84=0,0,0,0,0,0 +units=m +no_defs"

# Create a bounding box as an sf object for layers ----
b_box <- c(xmin = 350432, ymin = 7179284,
            xmax = 857780, ymax = 7444230)

# Now intersect the full world's coastline with the bounding box
suppressWarnings({
  sweden.c <- st_crop(swe_no_sweref, b_box)
  plot(sweden.c)
})
```



```
### water data ----
```

```
lakes <- input_vector %>%
  grep(pattern = "lakes", value = T) %>%
  sf::st_read()
```

```
## Reading layer 'ne_10m_lakes' from data source
##   '/Users/hirn0001/Library/CloudStorage/OneDrive-Sverigeslantbruksuniversitet/Undervisning/Teaching ...
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1355 features and 41 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -165.9656 ymin: -50.66967 xmax: 177.1544 ymax: 81.95521
## Geodetic CRS:  WGS 84
```

```
class(lakes)
```

```
## [1] "sf"           "data.frame"
```

```
st_crs(lakes)
```

```
## Coordinate Reference System:
##   User input: WGS 84
##   wkt:
##   GEOGCRS["WGS 84",
##         DATUM["World Geodetic System 1984",
```

```

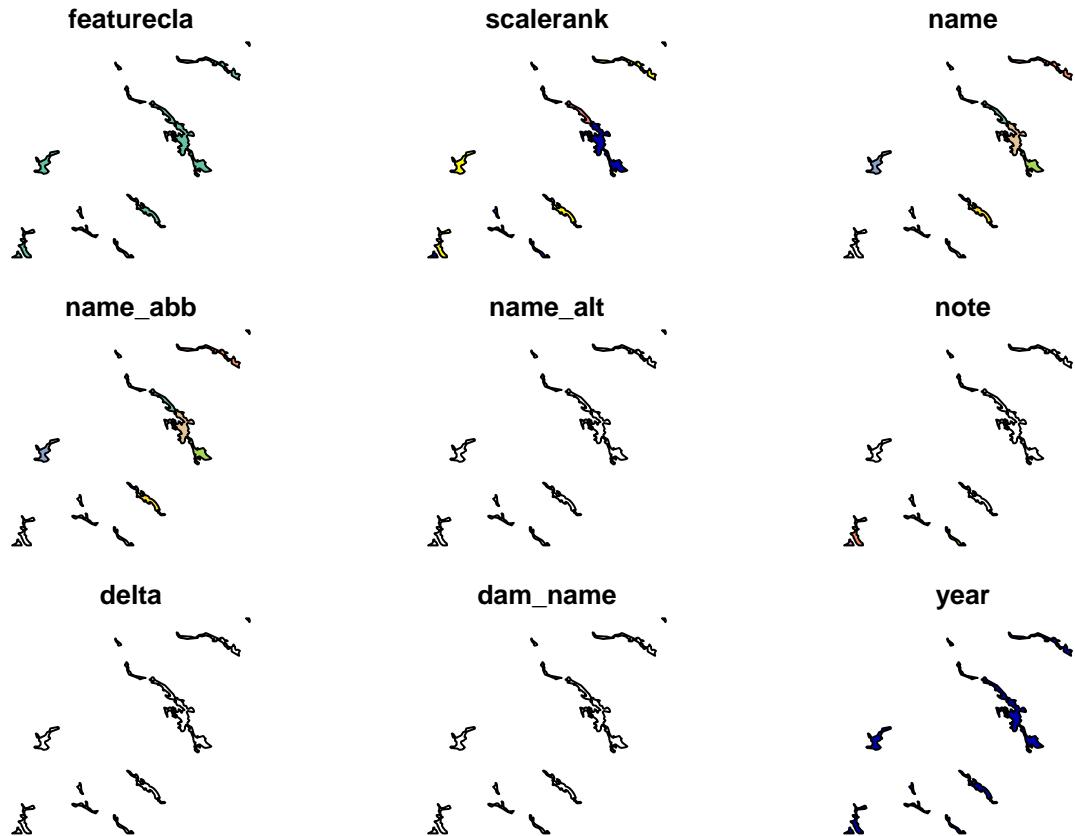
##           ELLIPSOID["WGS 84",6378137,298.257223563,
##           LENGTHUNIT["metre",1]],
##           PRIMEM["Greenwich",0,
##           ANGLEUNIT["degree",0.0174532925199433]],
##           CS[ellipsoidal,2],
##           AXIS["latitude",north,
##           ORDER[1],
##           ANGLEUNIT["degree",0.0174532925199433]],
##           AXIS["longitude",east,
##           ORDER[2],
##           ANGLEUNIT["degree",0.0174532925199433]],
##           ID["EPSG",4326]

# Transform or convert coordinates of simple feature (sf)
lakes <- st_transform(lakes, crs = 3006)
suppressWarnings({

# crop an sf object to a specific rectangle (here based on extent of "sweden.c"-shp)
# sf_use_s2(FALSE) # use if errors occur below
lakes <- st_crop(lakes, sweden.c)

class(lakes)
plot(lakes)
})

```



Add points of interest (cities)

Create xy points with labels. Here creating sf-object and setting the crs.

```
# Create points of interest (coordinates from google maps for demonstration)
places <- data.frame(ID= c("Arjeplog", "Arvidsjaur", "Sorsele"),
                      y = c(7328407.945, 7280856.153, 7270326.684),
                      x = c(630040.101, 693262.414, 617477.638)) %>%
  st_as_sf(coords = c("x", "y"), # create sf object
            crs=st_crs(sweden.c)) # set same crs as for sweden.c
class(places)

## [1] "sf"           "data.frame"

# st_crs(places) # check projection
```

Make plot using ggplot

```
library(ggspatial)

ggplot() +

  # plot sweden and lakes
  geom_sf(aes(geometry = sweden.c$geometry)) + # geom_sf() for vector data

  geom_sf(data = sweden.c,
          aes(geometry = sweden.c$geometry),
          fill= "antiquewhite")+

  geom_sf(data = lakes,
          fill="lightblue") +

  # layer_spatial(data=elevation) +

  # add cities
  geom_sf(data = places) +

  # add city names
  geom_sf_text(data = places, aes(label = ID),
               vjust=-0.5)+

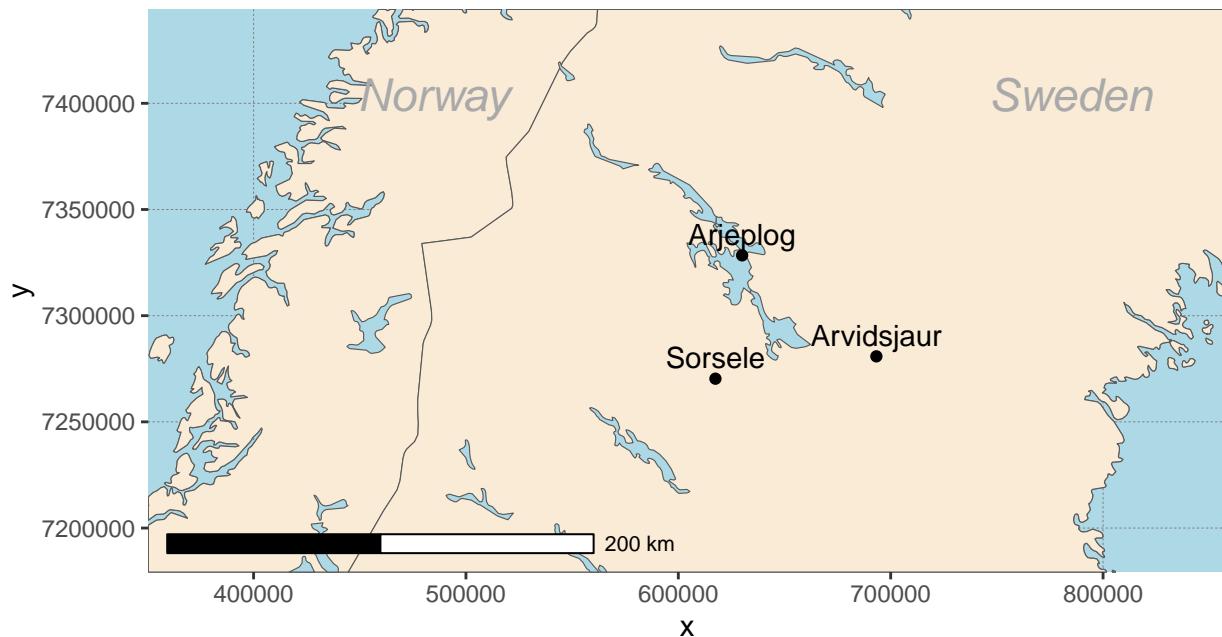
  # Add text on map
  annotate(geom = "text",
           x = 484609, y = 7404230,
           label = "Norway",
           fontface = "italic",
           color = "darkgray",
           size = 6) +

  annotate(geom = "text",
           x = 784609, y = 7404230,
```

```

    label = "Sweden",
    fontface = "italic",
    color = "darkgray",
    size = 6) +
    # Add north arrow
  annotation_north_arrow(location = "bl",
    which_north = "true",
    pad_x = unit(0.5, "cm"), # horizontal align
    pad_y = unit(8.5, "cm"), # vertical align
    style = north_arrow_fancy_orienteering) +
  #add annotation scale
  annotation_scale(location = "bl",
    width_hint = 0.5) +
  # keep new projection for ggplot
  coord_sf(crs = st_crs(3006),
  datum = sf::st_crs(3006),
  expand = FALSE) +
  # add grid lines
  theme(panel.grid.major = element_line(color = gray(.5),
    linetype = "dashed",
    linewidth = 0.1),
  panel.background = element_rect(fill = "lightblue"))

```



```

#### for removing or adding titles
# theme(axis.title.x = element_blank(),
#       axis.title.y = element_blank()) +
#       xlab("Longitude") + ylab("Latitude")

ggsave("output/study_area.png", width = 8, height = 6, dpi = 400)

```

Optional: Create a polygon and add to map

Exercise 2: Maps in R

Pre-define crs using ESPG-code or proj4string by ESPG: CRS("+init=epsg:3006") proj4string: "+proj=utm +zone=33 +ellps=GRS80 +units=m +no_defs"

```
# projection to use
crs_to_use <- "+proj=utm +zone=33 +ellps=GRS80 +units=m +no_defs" #proj4string
```

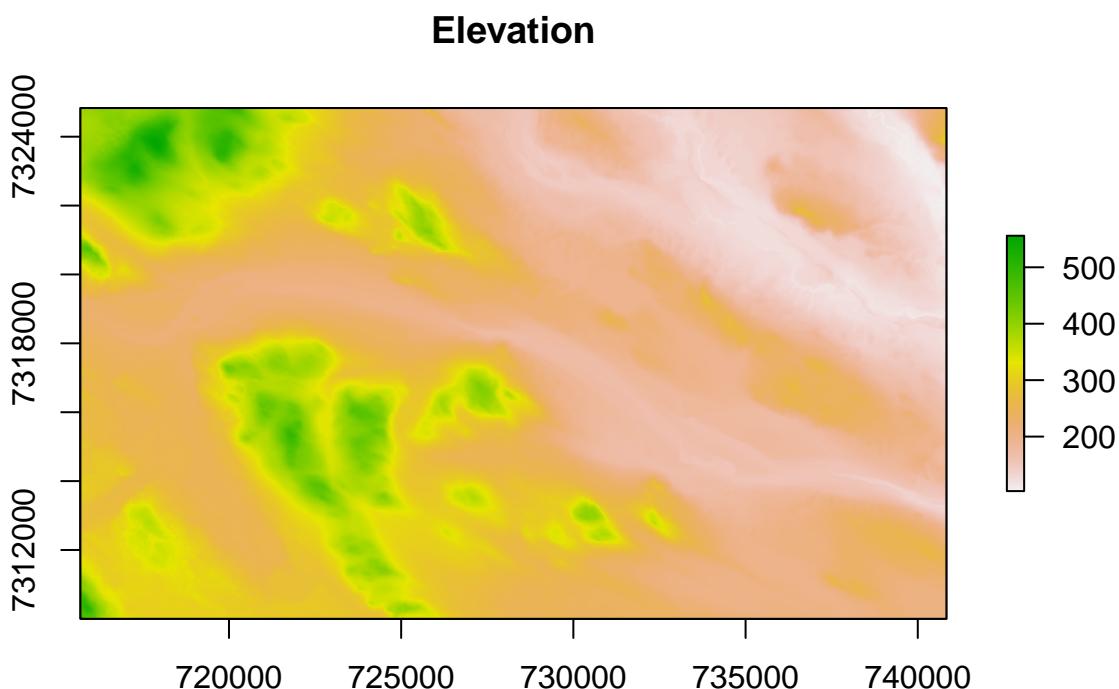
Load background data

Now let's start by loading the maps using the "raster"-package (another option: "terra"-package using rast()-function)

```
## [1] "input_data/raster/PV_2021_cc.tif"
## [2] "input_data/raster/PV_2021_dem.tif"
## [3] "input_data/raster/PV_2021_houses.tif"
## [4] "input_data/raster/PV_2021_lichen.tif"
## [5] "input_data/raster/PV_2021_nmd.tif"
## [6] "input_data/raster/PV_2021_roads.tif"
## [7] "input_data/raster/PV_2021_slope.tif"
## [8] "input_data/raster/PV_2021_TRI.tif"

## [1] "Clear_cuts_2021"           "PV_2021_dem"
## [3] "prox10_houses_stakke"     "lichenmap_stakke_original"
## [5] "nmd10_stakke_ungeneralized" "prox10_stakke_resample"
## [7] "slope10m_QGIS"            "ruggedness10_QGIS"

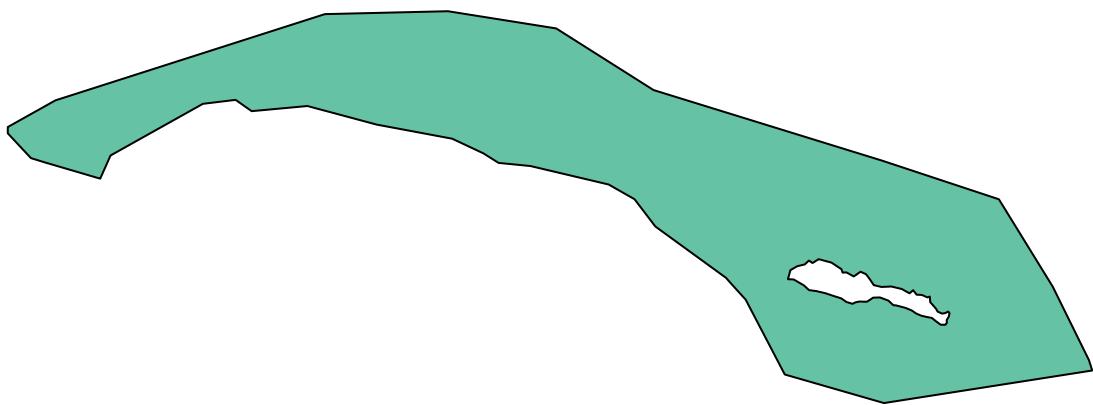
## [1] "+proj=utm +zone=33 +ellps=GRS80 +units=m +no_defs"
```



```
## Reading layer 'spatial_files_availability_PV_winter_2021_shorter_modified_new' from data source '/Us
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1 feature and 3 fields
## Geometry type: POLYGON
## Dimension:      XY
## Bounding box:  xmin: 716285.4 ymin: 7313268 xmax: 739293.8 ymax: 7321582
## Projected CRS: SWEREF99 TM

## [1] "+proj=utm +zone=33 +ellps=GRS80 +units=m +no_defs"
```

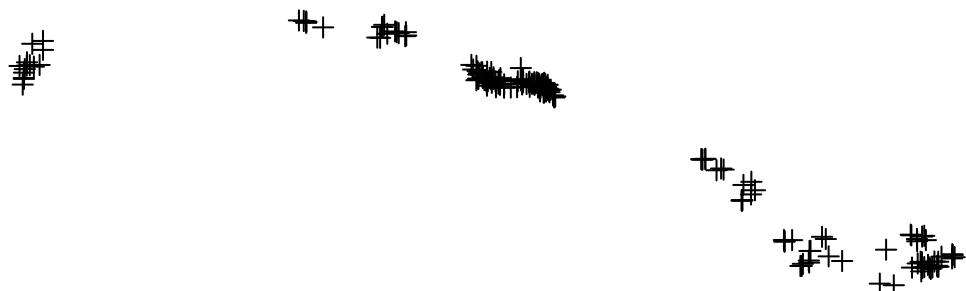
Home range



Add animal data (GPS points)

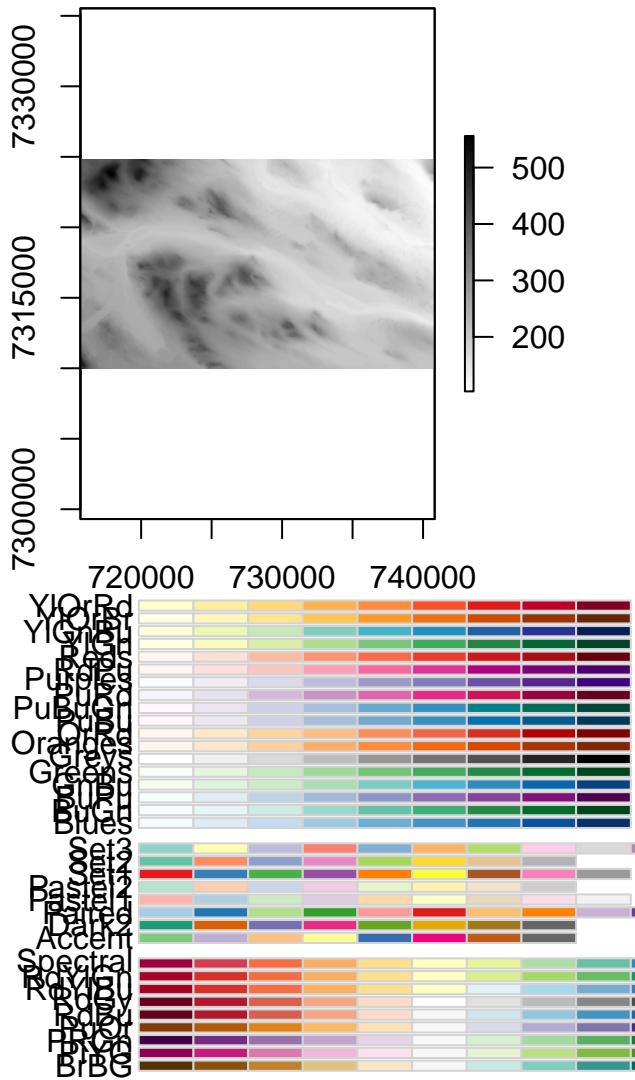
Load prepared reindeer data

Animal position data (one ID)

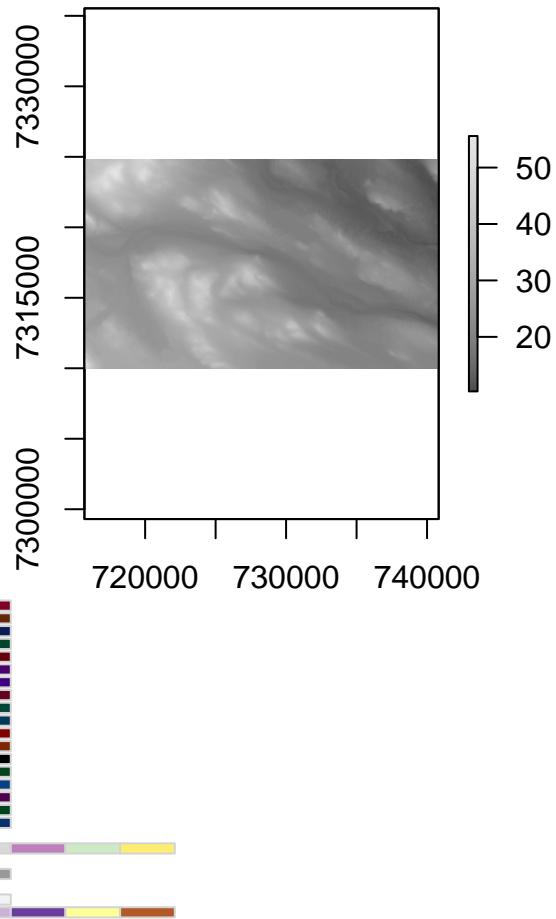


Explore and select colors

Elevation: inverted

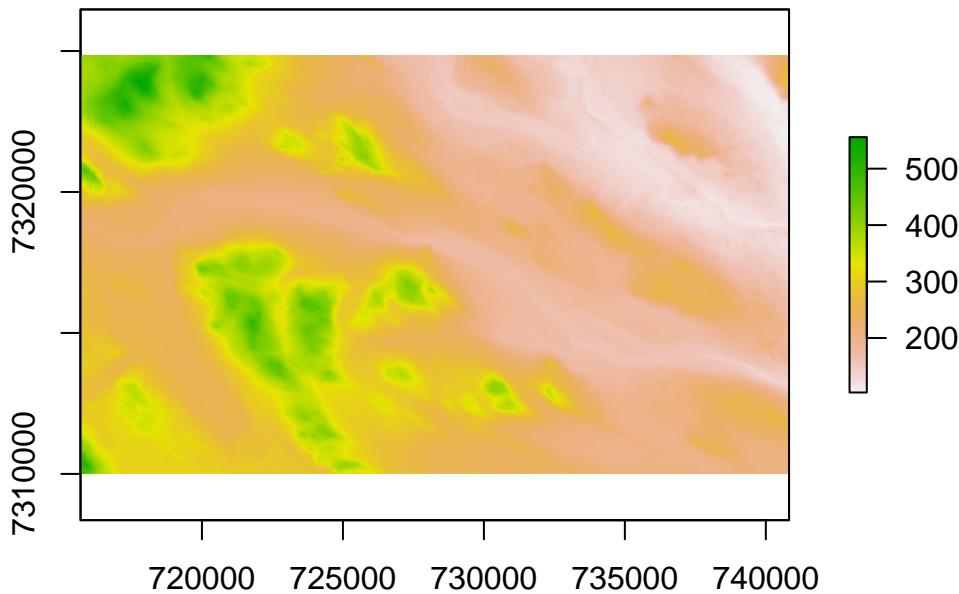


Elevation: grey scale colors



```
## [1] "#00A600" "#01A600" "#03A700" "#04A700" "#05A800" "#07A800"
```

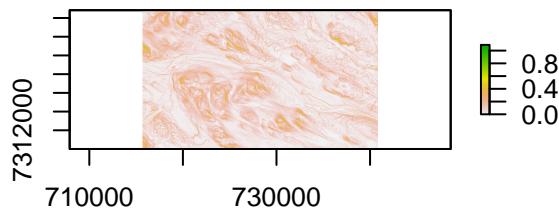
Elevation



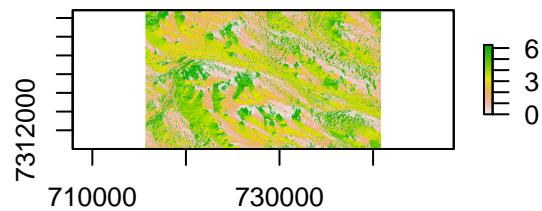
Prepare hillshade background map

Calculate slope and aspect from elevation (dem)

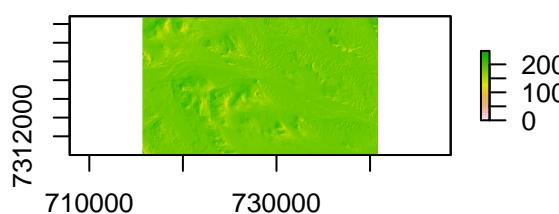
Slope



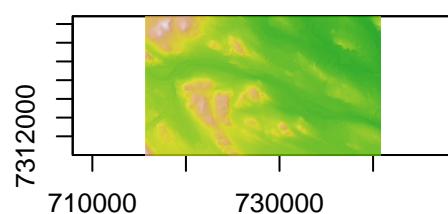
Aspect



Hillshade



Hillshade and overlay DEM



Plot and save all together

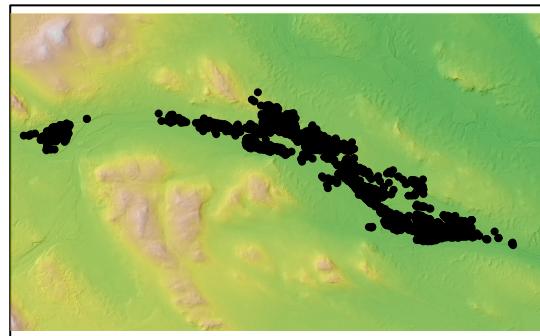
```
## Warning in plot.sf(HR_area, col = NA, border = 1, add = T): ignoring all but
```

```
## the first attribute
```

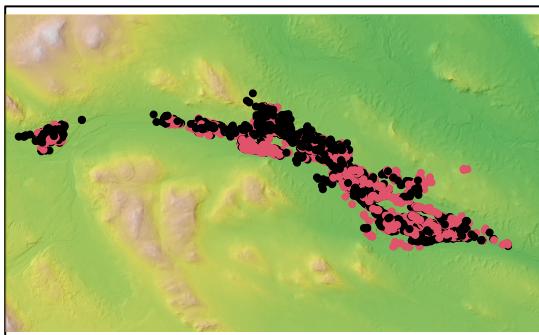
A Plot fed individuals



B Plot control individuals



C Plot all individuals



D Plot home range

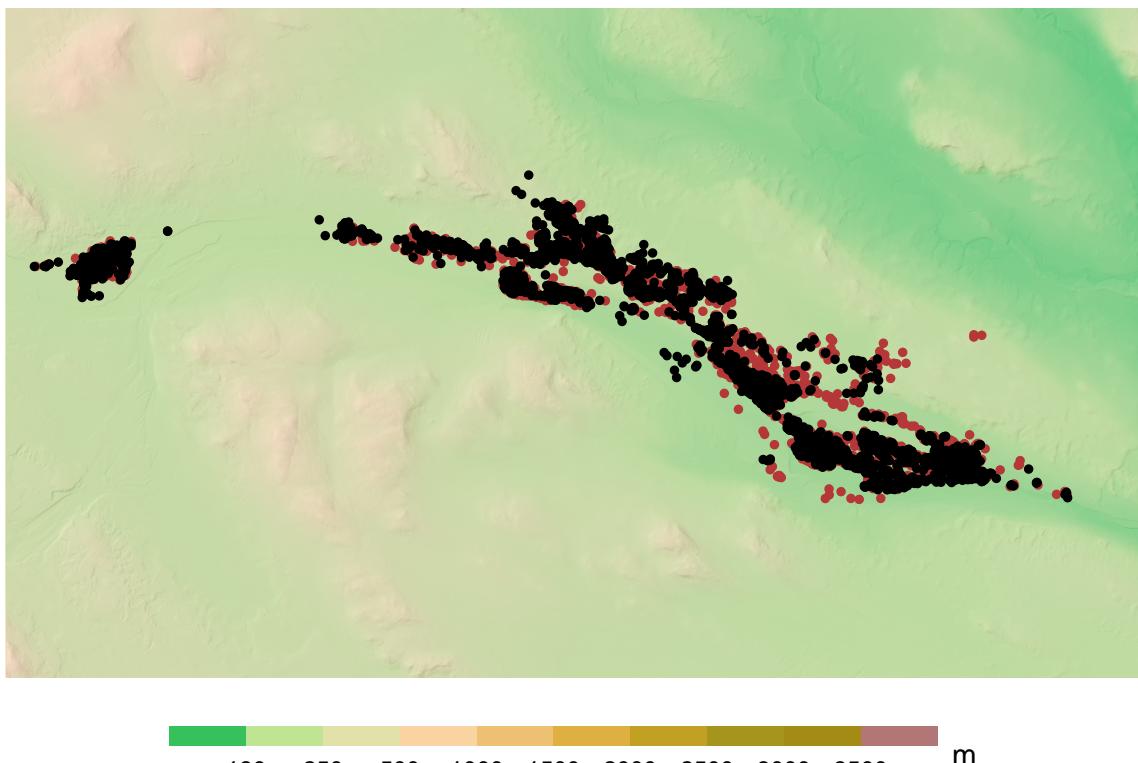


Saving it as ggplot

Some more packages required.. Credits to: <https://dominicroye.github.io/blog/hillshade-effect/index.html>

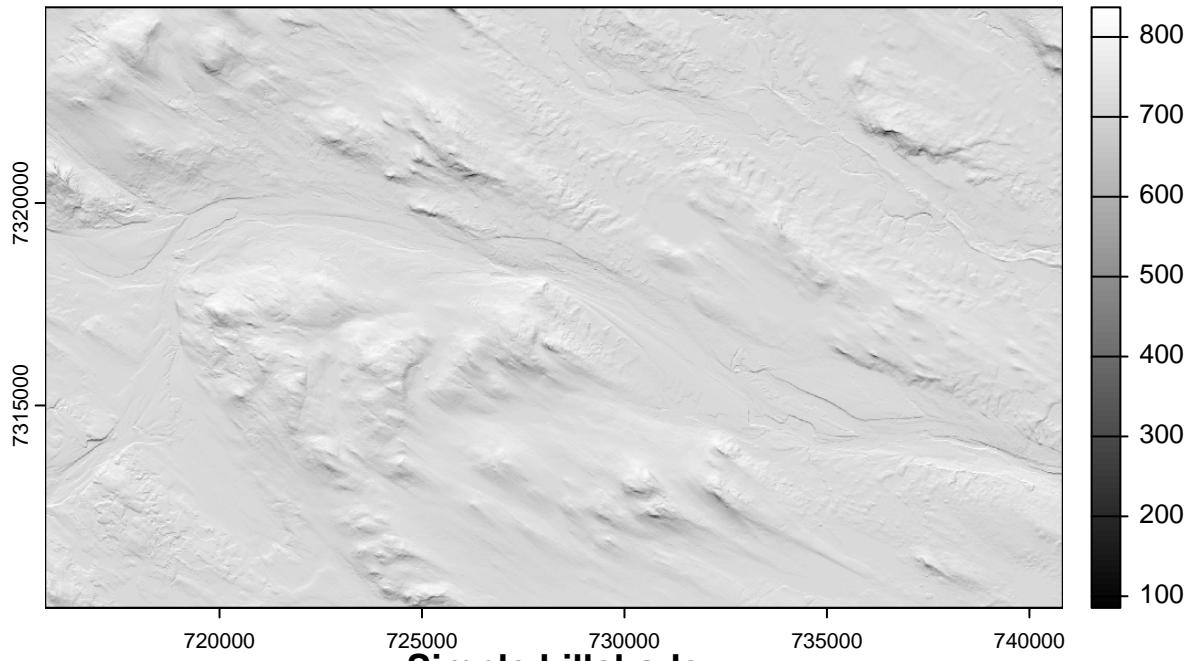
```
##  
## Attaching package: 'tidyterra'  
  
## The following object is masked from 'package:raster':  
##  
##     select  
  
## The following object is masked from 'package:stats':  
##  
##     filter  
  
## class      : Extent  
## xmin       : 715694  
## xmax       : 740824  
## ymin       : 7310002  
## ymax       : 7324832  
  
## [1] "SpatialPoints"  
## attr(,"package")  
## [1] "sp"
```

Simple hillshade using ggplot2

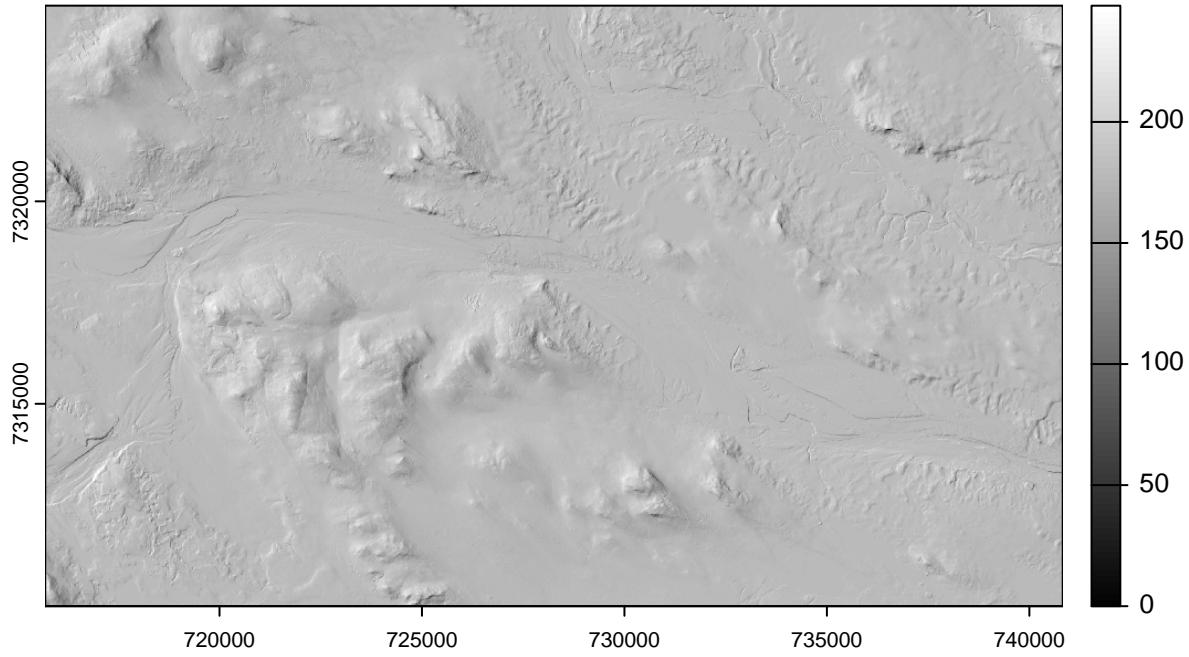


More examples

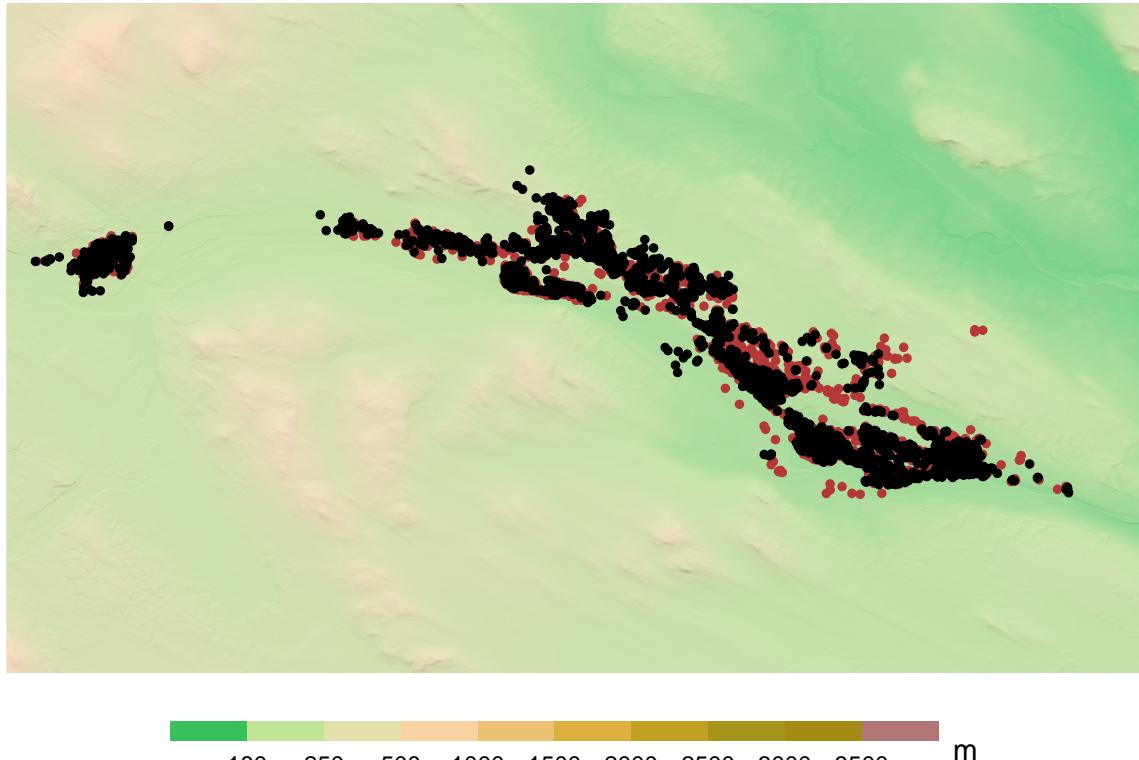
Multidirectional hillshade



Simple hillshade



Multidirectional hillshade using ggplot2



Example of creating multiplots ggplots

```
##  
## Attaching package: 'ggpubr'  
  
## The following object is masked from 'package:terra':  
##  
##     rotate  
  
## The following object is masked from 'package:raster':  
##  
##     rotate  
  
## Error in loadNamespace(x): there is no package called 'gpabr'  
  
## Error in gridExtra::arrangeGrob(p, top = top, bottom = bottom, left = left, : object 'fig' not found  
  
## Error in grid.draw(plot): object 'fig2' not found
```