LTDSMethod

Max Dolton Jones

2023-01-05

Introduction to LTDSMethod

The LTDSMethod package was designed for several main tasks:

- Designing line transect distance sampling pilot surveys
- Using pilot survey methodology to inform full LTDS surveys
- Estimate the density of a species within a study area
- Creating a summary of survey effort and results

More specifically, the package was initially intended as a tool to aid monitoring efforts for translocated gopher tortoises throughout Florida. As a result, some of the functions are highly specific to gopher tortoise biology and conservation. However, LTDSMethod may be applicable for other studies/projects trying to implement LTDS methods.

The LTDSMethod package has a complimentary Standard Operating Procedure (SOP), which can be referenced to help understand background, terminology and access some results behind survey design decisions.

Working example

```
library(LTDSMethod)

#Set a seed for reproducible results
set.seed(1235)
```

Before sampling

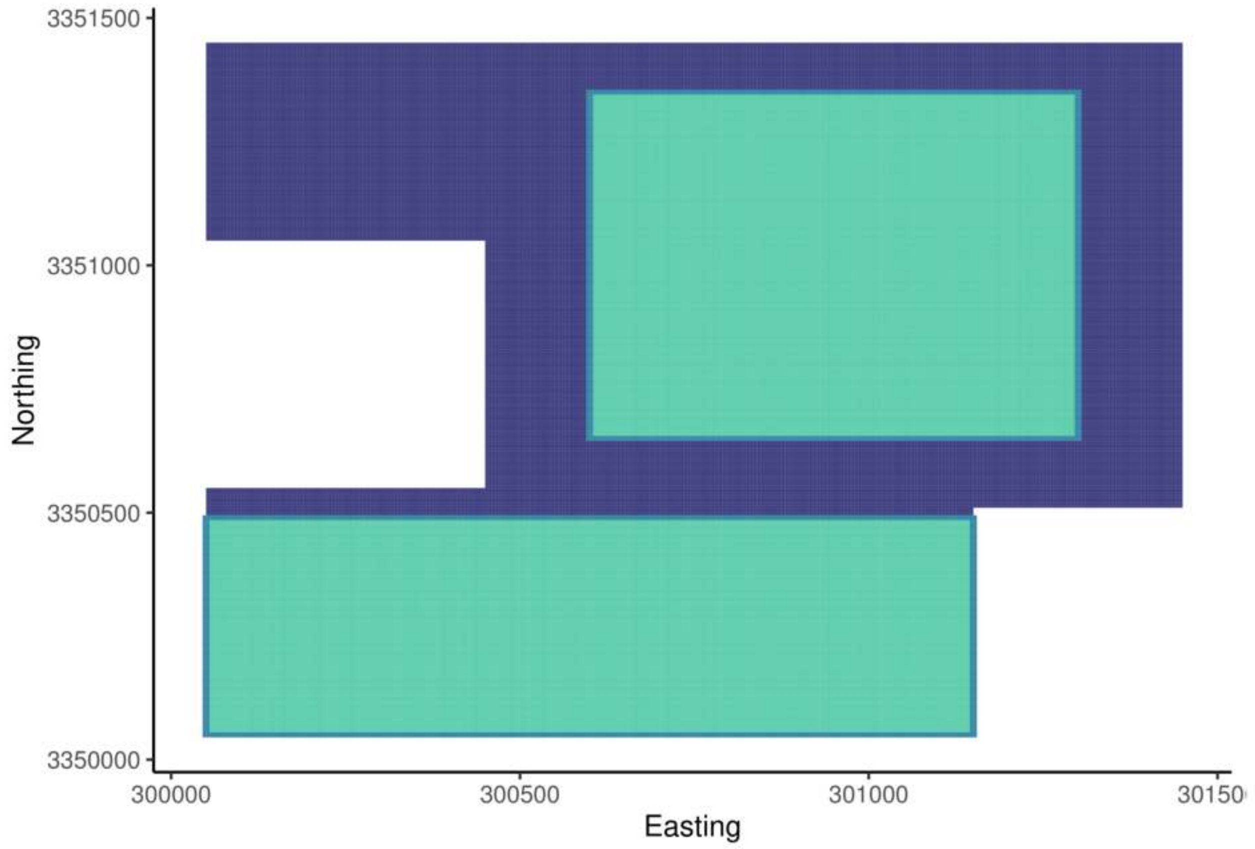
Let's remind ourselves what is needed to work through the LTDSMethod package. This information can be found on page XXX of the SOP:

1. Shapefile of recipient site (only permitted, habitat area)
2. Shapefile of additional acreage (only permitted, habitat area that is not already included in the original shapefile)
3. The acreage of the recipient site (only permitted, habitat area which is to be surveyed)
4. UTM zone of recipient site location (either 16 or 17, see Figure 3 of SOP)
5. Location of soft-release enclosures at recipient site (either embedded into shapefiles, or on a separate map)

Let's start with number 1. It would be easy to simply load in a shapefile for this section. Here, we are going to create our own shapefile and raster. Specifically, we will create an irregular-shaped site (not square), with 100% available habitat for tortoises, that also has two 120 acre soft-release pens.

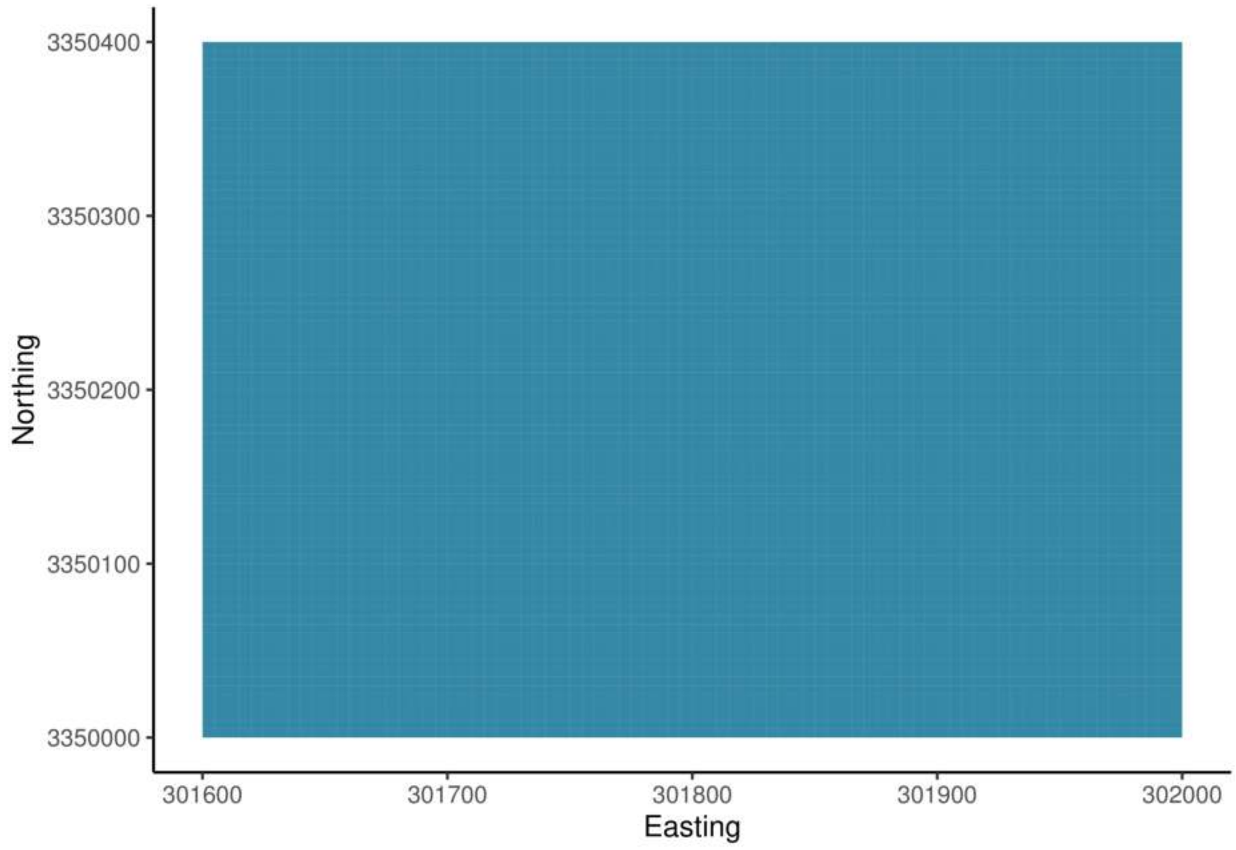
```
#400 acre irregular site, 100% habitat, 2x120 acre pens

#Site size is length in meters of one square edge
site.size <- 1500
#Pixel size will be set to 5m per pixel
pixel.size <- 5
#We can create a raster based on those set sizes. Projection is for Florida.
new.crs <- "+proj=utm +zone=17 +datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0"
site <- raster(nrows=site.size/pixel.size, ncol=site.size/pixel.size, crs=new.crs)
#Set smallest Easting and Northing for site
xmin <- 300000
ymin <- 3350000
#Set the extent based on coordinates and size
extent(site) <- c(xmin, xmin+site.size, ymin, ymin+site.size)
#For now, set all pixel values to 0
values(site) <- 0
site
#> class       : RasterLayer
#> dimensions  : 300, 300, 90000  (nrow, ncol, ncell)
#> resolution  : 5, 5  (x, y)
#> extent      : 3e+05, 301500, 3350000, 3351500  (xmin, xmax, ymin, ymax)
#> crs         : +proj=utm +zone=17 +datum=NAD83 +units=m +no_defs
#> source      : memory
#> names       : layer
#> values      : 0, 0  (min, max)

#We want to create a site that is more irregular than just a square:
area.1 <- extent(matrix(c(300050, 3350050, 301450, 3351450), nrow=2))
site[area.1][site[area.1] == 0] <- 1
area.2 <- extent(matrix(c(300050, 3350550, 300450, 3351050), nrow=2))
site[area.2][site[area.2] == 1] <- 0
area.3 <- extent(matrix(c(301150, 3350050, 301450, 3350510), nrow=2))
site[area.3][site[area.3] == 1] <- 0
#We can use the below code to check the acreage of the site
sum <- sum(site@data@values, na.rm = TRUE)
(sqrt(sum)*5)^2/4047
#> [1] 400.7907

#We need to add our soft-release pens - just to help with the vignette example
pen.area <- site

pen.1 <- extent(matrix(c(300050, 3350050, 301150, 3350490), nrow=2))
pen.area[pen.1][pen.area[pen.1] == 1] <- 3
pen.2 <- extent(matrix(c(300600, 3350650, 301300, 3351350), nrow=2))
pen.area[pen.2][pen.area[pen.2] == 1] <- 3

#Remove rest of habitat to set boundaries
pen.area <- reclassify(pen.area, cbind(1, NA))
pen.area <- reclassify(pen.area, cbind(0, NA))
```

```r
#Add fence to pen area to site
edge <- boundaries(pen.area, classes=TRUE, directions=8)
edge <- reclassify(edge, cbind(1, 2))
edge <- reclassify(edge, cbind(0, NA))
#Merge all three together to make site
new.site <- merge(edge, pen.area, site)
site.rast <- reclassify(new.site, cbind(0, NA))

#Convert and store shapefiles to crop rasters later
site.shp <- rasterToPolygons(site.rast)

#Plot raster to visualize site
temp<-as.data.frame(site.rast, xy = T)
temp <- temp %>% na.omit()
ggplot(temp) +
  geom_tile(aes(x = x, y = y, fill = layer)) +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```



We have now created a recipient site to work with. The second item on our list is any additional acreage to be added to the site. We will only work with the site we have made above moving forward, but we can create additional acreage to test package functionality later on.

```
#Create another site, just to test the first function.
site.size <- 400  #Site size is length in meters of one square edge
pixel.size <- 5
new.crs <- "+proj=utm +zone=17 +datum=NAD83 +units=m +no_defs +ellps=GRS80 +towgs84=0,0,0"
site.2 <- raster(nrows=site.size/pixel.size, ncol=site.size/pixel.size, crs=new.crs)
xmin <- 301600
ymin <- 3350000
extent(site.2) <- c(xmin, xmin+site.size, ymin, ymin+site.size)
values(site.2) <- 0
site.2
#> class      : RasterLayer
#> dimensions : 80, 80, 6400  (nrow, ncol, ncell)
#> resolution : 5, 5  (x, y)
#> extent     : 301600, 302000, 3350000, 3350400  (xmin, xmax, ymin, ymax)
#> crs        : +proj=utm +zone=17 +datum=NAD83 +units=m +no_defs
#> source     : memory
#> names      : layer
#> values     : 0, 0  (min, max)
#Convert and store shapefiles to crop rasters later
site.shp.2 <- rasterToPolygons(site.2)

temp.2<-as.data.frame(site.2, xy = T)
temp.2 <- temp.2 %>% na.omit()
ggplot(temp.2) +
  geom_tile(aes(x = x, y = y, fill = layer)) +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```



The last three requirements can be conveniently parsed from the first site we created. We have already checked the site acreage, but we do this again below. We also set the projection system, but this can easily be verified.

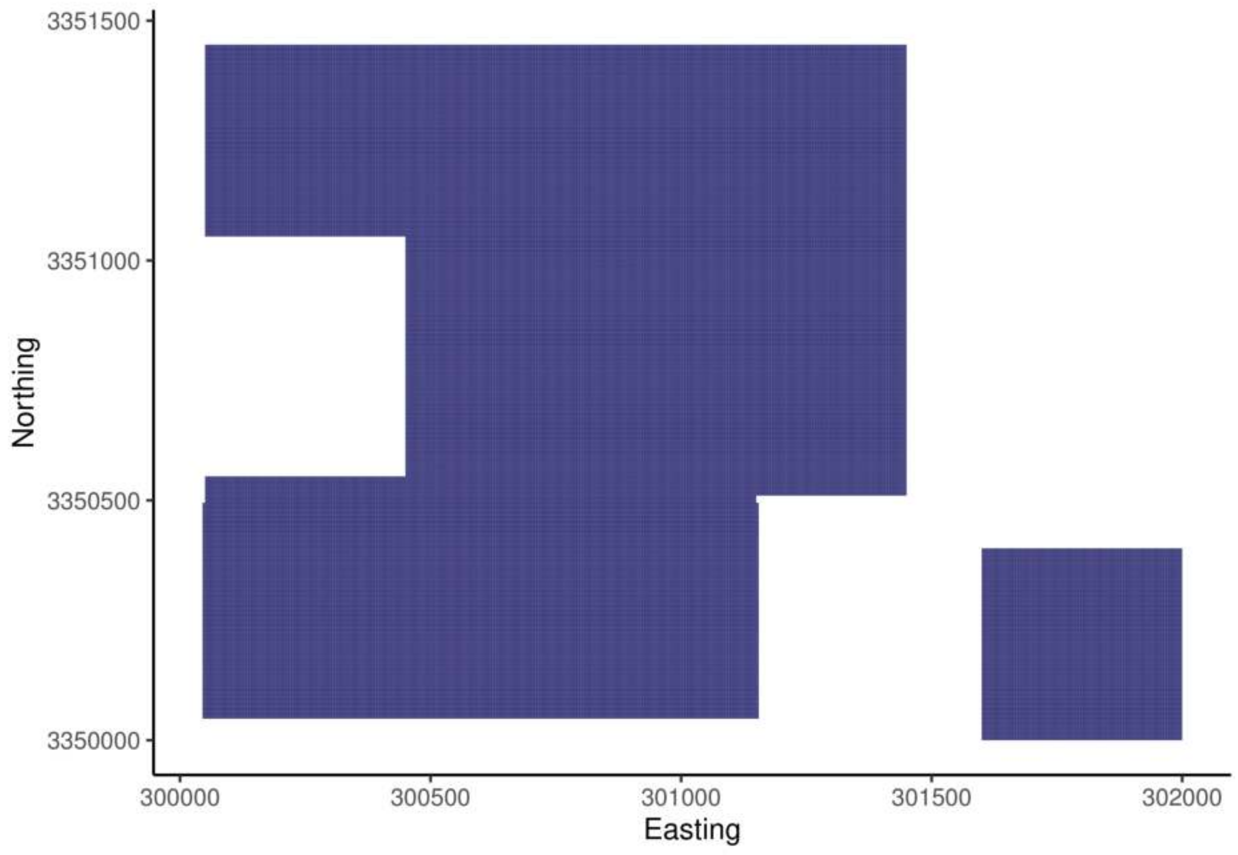
```
#Calculate site acreage
sum <- sum(site@data@values, na.rm = TRUE)
(sqrt(sum)*5)^2/4047
#> [1] 400.7907

#Check site CRS
site.rast
#> class      : RasterLayer
#> dimensions : 300, 300, 90000  (nrow, ncol, ncell)
#> resolution : 5, 5  (x, y)
#> extent     : 3e+05, 301500, 3350000, 3351500  (xmin, xmax, ymin, ymax)
#> crs        : +proj=utm +zone=17 +datum=NAD83 +units=m +no_defs
#> source     : memory
#> names      : layer
#> values     : 1, 3  (min, max)
```

Function:merging shapefiles - *site_merge*

We are not going to use our first function, as outlined in the SOP (Figure 10). The *site_merge* function will combine the two shapefiles we have just created.
For this function to run, we need to provide specific information. Firstly, we need to provide the two

shapefiles that need to be merged together (site.shp and site.shp.2). Then we need to specify the UTM zone (17 in this case). Next, we need to let the function know if we would like the merged site to be plotted upon merging, and also if this shapefile needs to be saved to the working directory (TRUE[T]/FALSE[F]).

```
new_site <- site_merge(shape1 = site.shp, shape2 = site.shp.2, zone = 17,
                       plot = F, save = F)

ggplot() +
  geom_polygon(data = new_site, aes(x = long, y = lat, group = group,
                                    fill = "light blue")) +
  scale_fill_viridis_d(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
#> Regions defined for each Polygons
```

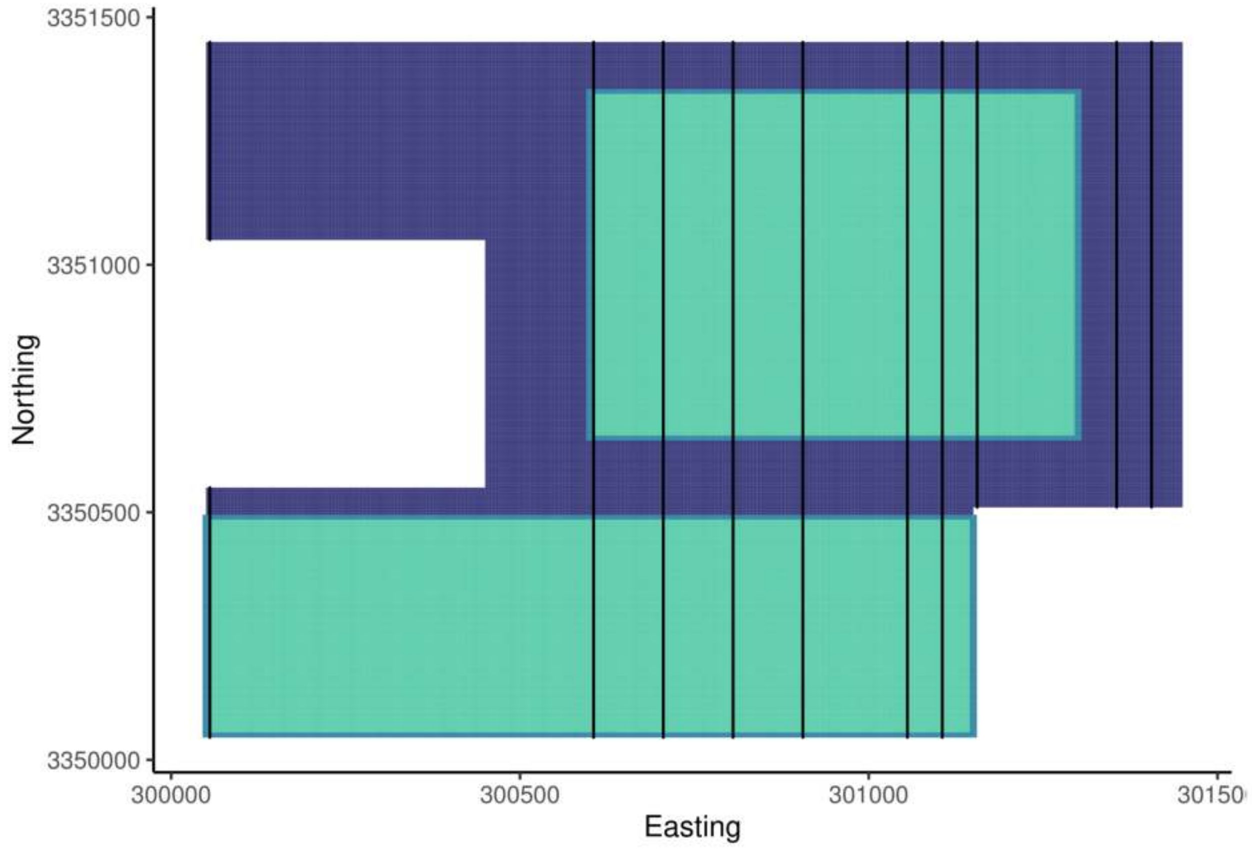


Function:placing transect lines - *tran_place*

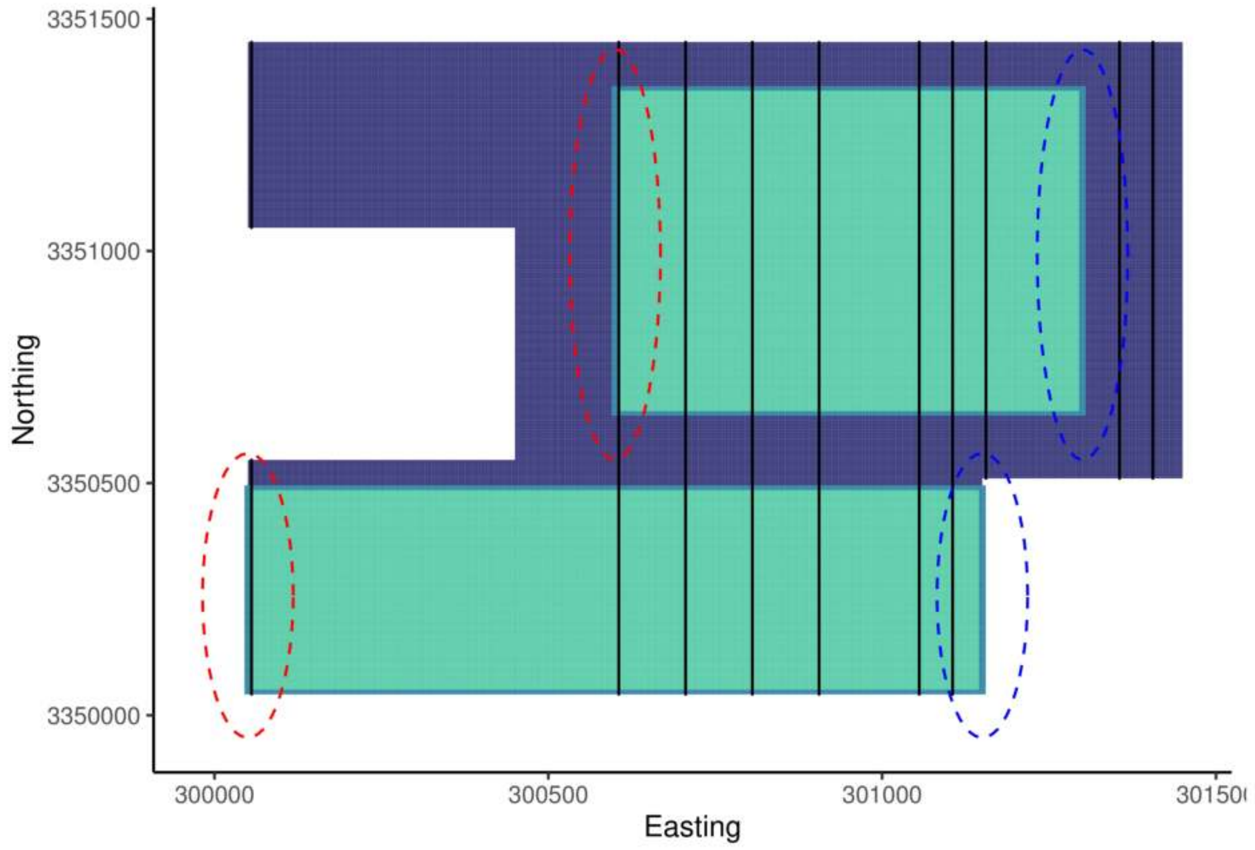
We are going to use the next function, *tran_place*, to delineate transect lines across our recipient site depicting the lines to be walked for the pilot survey. We need to provide the function with some key information, depending on the size of the area to be surveyed (number 3 of checklist). We have a 400 acre site, so we can refer to figure 4 of the SOP to know what characteristics are needed. For this size, we need 50% of the overall transects to be surveyed for the pilot survey, with a distance of 50m between adjacent transects.

Importantly, this is where we decide which direction the transect lines will be running. The options are either North-South ("N-S"), East-West ("E-W"), or Northeast-Southwest ("NE-SW"). In an attempt to troubleshoot potential problems, we will run the function using a "N-S" orientation. Similar to the last function, we need to provide the site shapefile, UTM zone and if we want the lines to be plotted and saved to the working directory.

```r
#The tran_place function produces a SpatialLines object
pilot.lines <- tran_place(tran.dist = 50, pil.perc = 0.5, direction = "N-S",
                          site.poly = site.shp, zone = 17, plot = F, save = F)
#We can review information about the SpatialLines object by running:
pilot.lines
#> class       : SpatialLines
#> features    : 10
#> extent      : 300055.5, 301405.5, 3350045, 3351450  (xmin, xmax, ymin, ymax)
#> crs         : +proj=utm +zone=17 +ellps=WGS84 +units=m +no_defs

#Plot the raster and lines to assess position relative to pens
temp<-as.data.frame(site.rast, xy = T)
temp <- temp %>% na.omit()
#Need to convert transect lines for plotting in ggplot
data <- data.frame(lines=1:length(pilot.lines))
p.lines.new <- SpatialLinesDataFrame(pilot.lines, data, match.ID = FALSE)
ggplot() +
  geom_tile(data = temp, aes(x = x, y = y, fill = layer)) +
  geom_polygon(data = p.lines.new, aes(x = long, y = lat, group = group),
               color = "black") +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```



We now have the first potential pilot lines that we could survey. However, when choosing transect direction, it is extremely important to consider the position of pen edges in relation to the transect lines. In our example, consider the following locations specifically:



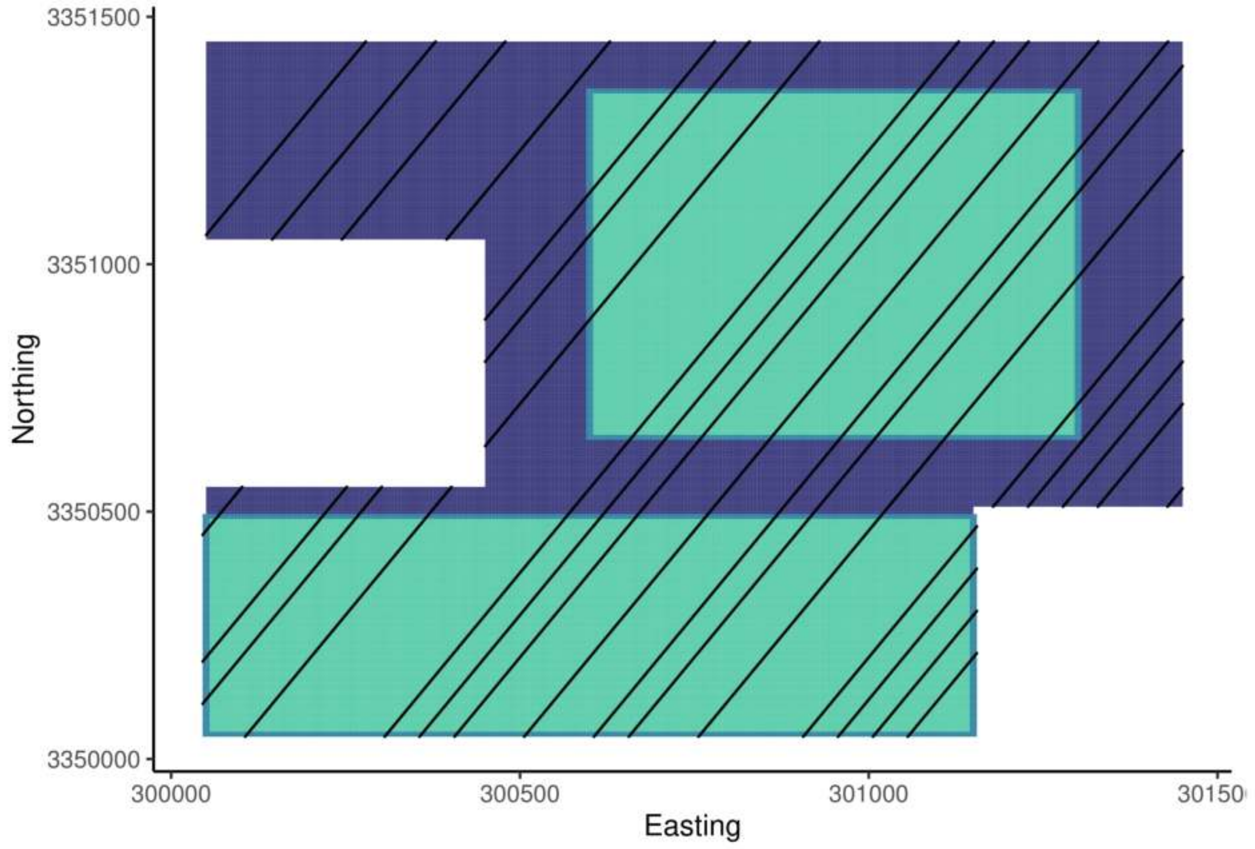
The dashed red lines show locations where the transect lines run directly on top of pen edge locations. Where this occurs, we are likely to overestimate the true density of tortoises. This is because translocated gopher tortoises are likely to cluster around these pen edges, because they typically offer built-up sandy habitat (perfect for burrowing), but also limit outward movement - which is typical for translocated reptiles. As a result of the transect line running along one of these highly clustered edges, the encounter rate of a survey is inflated and ultimately the estimates are highly overestimated.

On the other hand, the blue dashed lines show where a pen edge has been completely missed by the placement of transect lines. We have also found that underestimations of density can occur due to an uneven sampling across pen edges, particularly where highly clustered pen edges are missed entirely.

Due to the poor placement of transect lines shown in the above figure, we will now try running the *tran_place* function again using a NE-SW direction. If we tried an E-W direction, we could easily encounter the same issue as N-S, due to the high linearity of our simulated study site. Let's see what the NE-SW looks like:

```
#Now trying the same parameters, but with NE-SW
pilot.lines <- tran_place(tran.dist = 50, pil.perc = 0.5, direction = "NE-SW",
                          site.poly = site.shp, zone = 17, plot = F, save = F)
#We can review information about the SpatialLines object by running:
pilot.lines
#> class       : SpatialLines
#> features    : 19
#> extent      : 300045, 301450, 3350045, 3351450  (xmin, xmax, ymin, ymax)
#> crs         : +proj=utm +zone=17 +ellps=WGS84 +units=m +no_defs

#Plot the raster and lines to assess position relative to pens
temp<-as.data.frame(site.rast, xy = T)
```

```
temp <- temp %>% na.omit()
#Need to convert transect lines for plotting in ggplot
data <- data.frame(lines=1:length(pilot.lines))
p.lines.new <- SpatialLinesDataFrame(pilot.lines, data, match.ID = FALSE)
ggplot() +
  geom_tile(data = temp, aes(x = x, y = y, fill = layer)) +
  geom_polygon(data = p.lines.new, aes(x = long, y = lat, group = group),
               color = "black") +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```



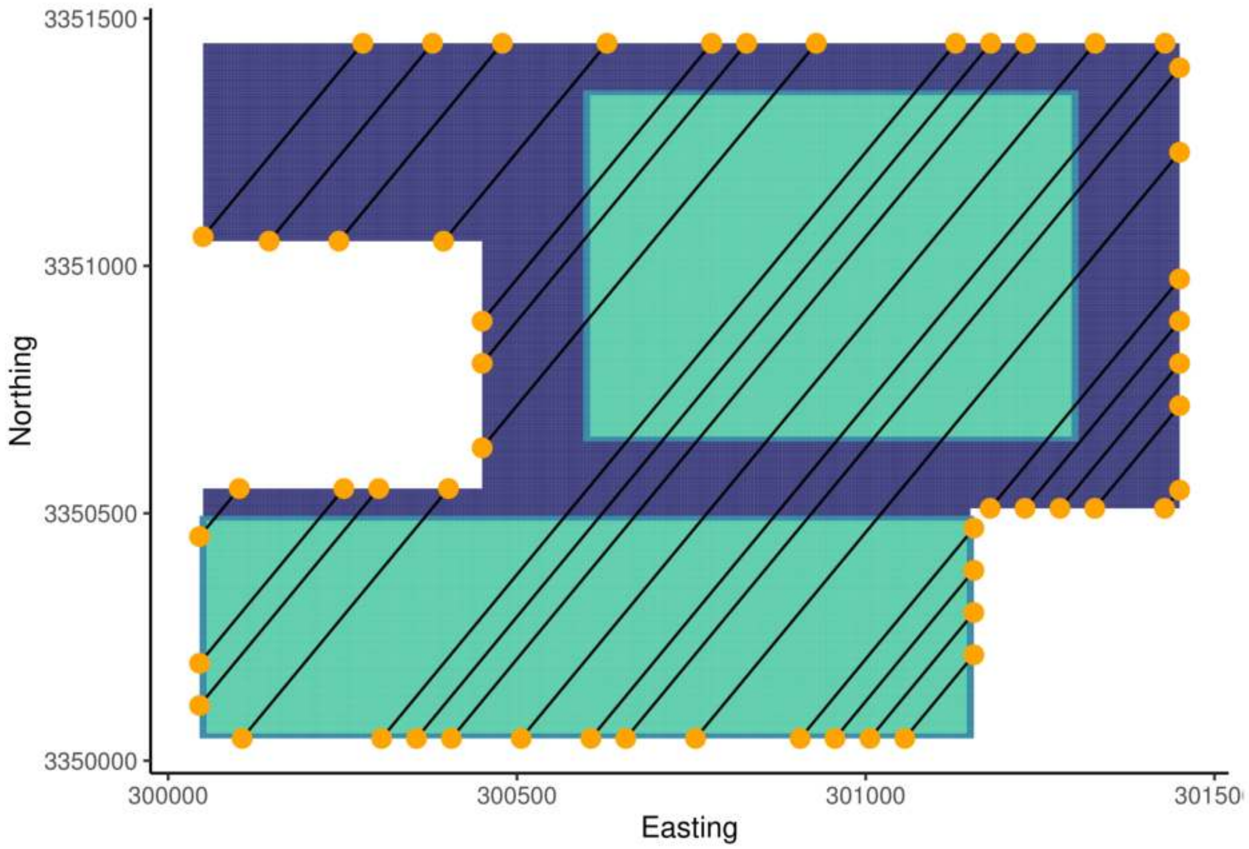
These new pilot lines give a better distribution of transects across pen edges. So we will be moving forward using these pilot transect lines.

Function:getting start and end locations - *start_end*

Now we have the transect lines needed for our pilot surveys, we are going to extract the start and end locations of each line. These will be exported to a GPS device so that we can adhere to line trajectories in the field.

For the start_end function to work, we need to provide the pilot lines that we created using the *tran_place* function, and the utm zone as before. By setting save = T, this will save the points to the working directory.

```r
se.pts <- start_end(lines = pilot.lines, zone = 17, save = F, plot = F)

#Change to dataframe for plotting
se_loc <- data.frame(se.pts)
#We can also check the coordinated by plotting against the site and transects
ggplot() +
  geom_tile(data = temp, aes(x = x, y = y, fill = layer)) +
  geom_polygon(data = p.lines.new, aes(x = long, y = lat, group = group),
               color = "black") +
  geom_point(data = se_loc, aes(x = x, y = y), size = 3, color = "orange") +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```



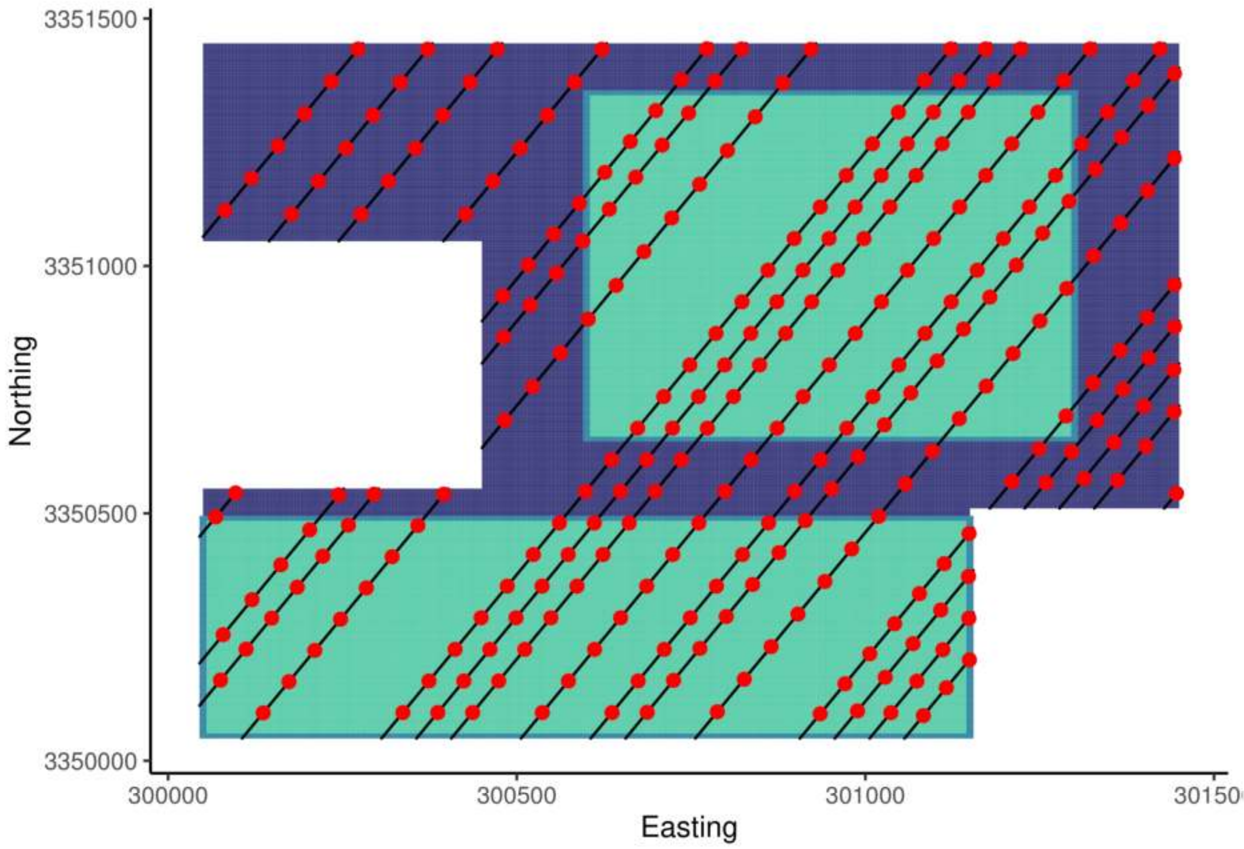
The orange points now delineate either where a transect line should start or end during a survey. Observers should survey from one orange point, to the opposite orange point which is connected via the blak trnsect line.

Function:delineate locations for vegetation sampling - *veg_plot*

While doing the surveys, collecting vegetation obstruction data can be extremely important for accurate tortoise density estimation. In the next function, we are going to determine where these vegetation plots are going to be conducted.

The locations are determined in a very similar manner to the *start_end* function where we need to provide the pilot lines. However, we need to also provide the set distance in meters between vegetation points.

```
veg.loc <- veg_plot(lines = pilot.lines, intv = 75, save = F, plot = F)

#Change to dataframe for plotting
veg_loc <- data.frame(veg.loc)
#We can also check the coordinated by plotting against the site and transects
ggplot() +
  geom_tile(data = temp, aes(x = x, y = y, fill = layer)) +
  geom_polygon(data = p.lines.new, aes(x = long, y = lat, group = group),
               color = "black") +
#  geom_point(data = se_loc, aes(x = x, y = y), size = 3, color = "orange") +
  geom_point(data = veg_loc, aes(x = x, y = y), size = 2, color = "red") +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```



The red points now show where observers will stop and collect vegetation obstruction data. Consistent with our function, these are set at 75m apart.

After pilot

Function:calculate pilot effort - *samp_effort*

Once the pilot survey has been conducted, as outlined in the SOP, we can take the start and end locations for all surveyed lines and calculate the effort. Our *start_end* function creates a csv file with start and end locations for each transect line. This mirrors the expected dataframe from an actual field survey. The order of the columns is the most important for the function. See the SOP for information on how to properly format the coordinate data during and following an LTDS survey.

To calculate the effort (distance in meters), we provide the start and end locations of each transect line walked during the pilot survey. As before, we also need the utm zone. We also have the option to plot the transect lines. Since we are using lines that we created, we will not re-plot the lines, however, this is this could give us the opportunity to assess if there are any errors with the data collection or function. For example, if the lines do not look anything like what was actually walked (besides being perfectly straight), then there is likely an issue.

```
#The csv file will be in the working directory
start.end <- read_csv("./start_end_coord.csv")
#> Rows: 27 Columns: 5
#> -- Column specification --------------------------------------------------------
#> Delimiter: ","
#> dbl (5): Start_x, Start_y, End_x, End_y, Transect_ID
#>
#> 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.

effort <- samp_effort(points = start.end, zone = 17, plot = F)

#View resulting effort - this is another place to check if the distance is
#corresponding to the actual effort in the field.
effort
#> [1] 19868.67
```

As we can see from above, the effort for the pilot survey was 19869m (~19.9km).

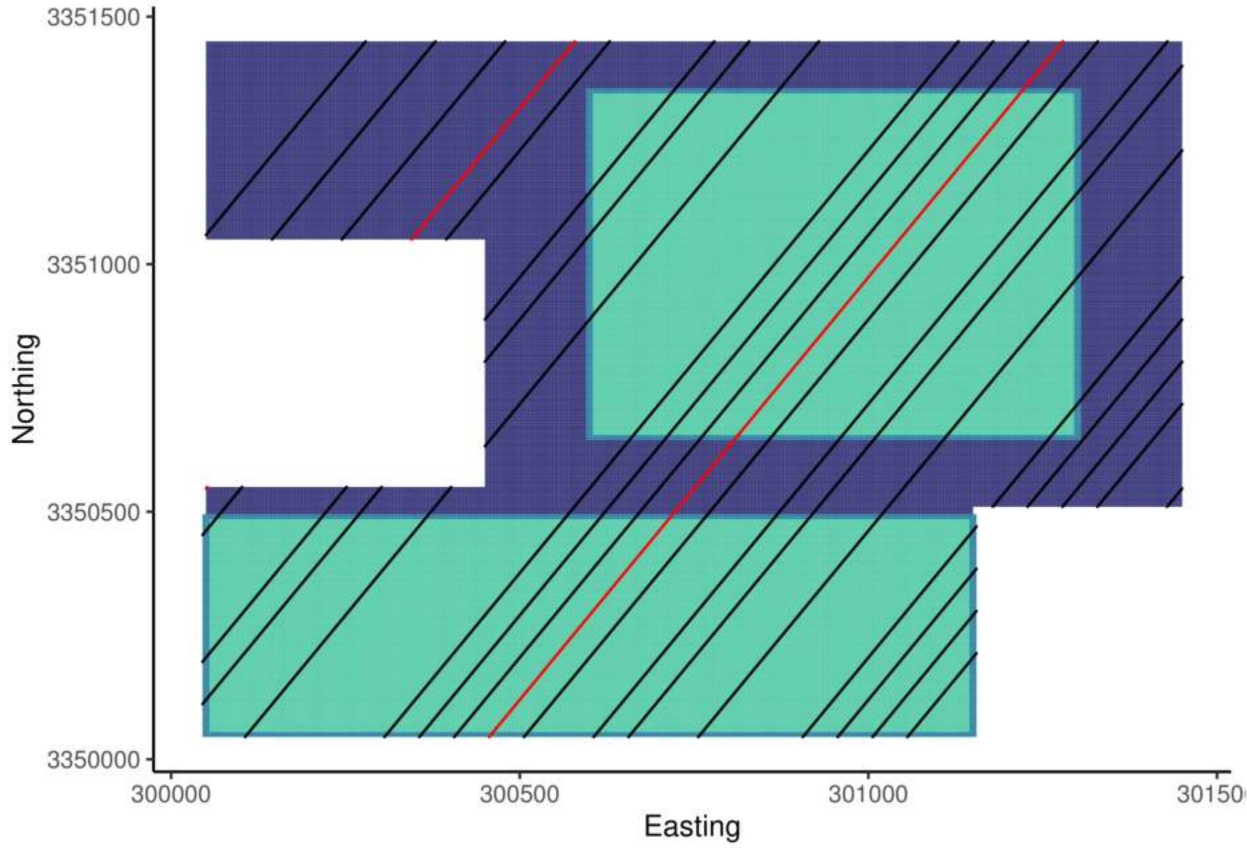
Function:determine lines for full LTDS survey - *samp_full*

Now we have the distance covered during the pilot survey, we can calculate the remaining effort needed to attain our target CV (default = 17%). The following function has a similar functionality to the *tran_place* function used at the beginning of the tutorial. However, we do not want to duplicate effort, so we now build upon the pilot survey to add additional transect lines.

There are several bits of information that we need to provide to get the function to work. Firstly, we need the effort from the *samp_effort* function. We need to state the direction of the transects (which needs to match the direction of the pilot survey), which in this case is NE-SW. Next, we need the number of occupied burrows detected during the pilot survey. If there were a lot of occupied burrows discovered, we can pull this information from the data. However, for the purpose of this vignette, we will set the number of occupied burrows (nburr) to 100.We then need to provide the desired CV from the survey. For our tutorial, we will aim for a CV of 17%. As with the other functions, we lastly need to provide the utm zone.

```
full.ltds <- samp_full(effort = effort, site.poly = site.shp, direction = "NE-SW",
                       nburr = 100, cv = 0.17, zone = 17, plot = F, save = F)
```

```
#> [1] "CV = 17.32%"

data <- data.frame(lines=1:length(full.ltds))
full.lines <- SpatialLinesDataFrame(full.ltds, data, match.ID = FALSE)
ggplot() +
  geom_tile(data = temp, aes(x = x, y = y, fill = layer)) +
  geom_polygon(data = p.lines.new, aes(x = long, y = lat, group = group),
               color = "black") +
  geom_polygon(data = full.lines, aes(x = long, y = lat, group = group),
               color = "red") +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```

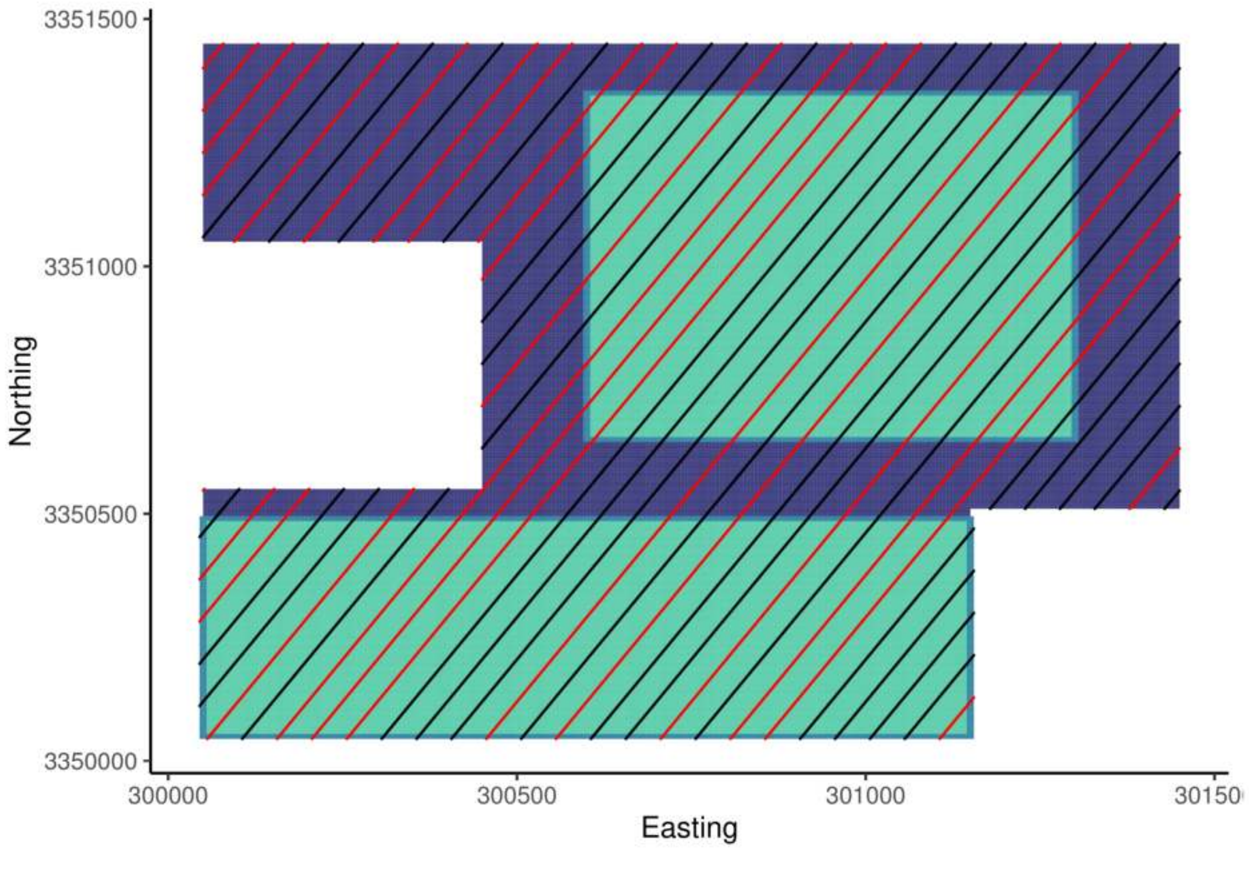


By assigning 100 discovered, occupied burrows we reached a CV of 17.32%. The function has delineated new lines (red), that need to be surveyed to attain a CV under 17%. Now, let's test the function again, but with 120 burrows:

```
full.ltds <- samp_full(effort = effort, site.poly = site.shp, direction = "NE-SW",
                       nburr = 120, cv = 0.17, zone = 17, plot = F, save = F)
#> [1] "Target CV attained. CV = 15.81%"
```

By changing the nburr to 120, the object is no longer created. In the console, we can see the statement "Target CV attained". This means that by finding 120 burrows with the current amount of pilot effort, we have already achieved a CV that is lower than our target. In this case, the CV is 15.81%. Now, let's try the opposite, what if we only detected 50 occupied burrows?

```
full.ltds <- samp_full(effort = effort, site.poly = site.shp, direction = "NE-SW",
                       nburr = 50, cv = 0.17, zone = 17, plot = F, save = F)
#> [1] "CV = 24.49%"

data <- data.frame(lines=1:length(full.ltds))
full.lines <- SpatialLinesDataFrame(full.ltds, data, match.ID = FALSE)
ggplot() +
  geom_tile(data = temp, aes(x = x, y = y, fill = layer)) +
  geom_polygon(data = p.lines.new, aes(x = long, y = lat, group = group),
               color = "black") +
  geom_polygon(data = full.lines, aes(x = long, y = lat, group = group),
               color = "red") +
  scale_fill_viridis_c(option = "mako", begin = 0.3, end = 0.8) +
  xlab("Easting") +
  ylab("Northing") +
  theme_classic() +
  theme(legend.position = "none")
```



With this much lower encounter rate, our CV was much higher at 24.49%. We can see from the accompanying plot that we would now need to survey all possible transects to achieve a more adequate CV.

After full LTDS

We now presume that the full LTDS survey has been conducted. This survey will ultimately give us a dataframe with several important variables. Import the data by:

```
full.data <- read_csv("./Data/LTDS_example_data.csv")
#> Rows: 263 Columns: 6
#> -- Column specification --------------------------------------------------------
#> Delimiter: ","
#> chr (3): Occupied, Burrow_condition, Commensals
#> dbl (3): Transect_ID, Burrow_width, Distance
#>
#> 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.

#We can also summarize the data
summary(full.data)
#>   Transect_ID     Burrow_width      Distance          Occupied
#>  Min.   : 1.00   Min.   : 50.5   Min.   : 0.02674   Length:263
#>  1st Qu.: 4.50   1st Qu.:149.1   1st Qu.: 1.71504   Class :character
#>  Median :12.00   Median :241.0   Median : 3.72894   Mode  :character
#>  Mean   :10.98   Mean   :210.0   Mean   : 4.36837
#>  3rd Qu.:16.00   3rd Qu.:279.6   3rd Qu.: 6.48641
#>  Max.   :22.00   Max.   :307.9   Max.   :14.83788
#>  Burrow_condition    Commensals
#>  Length:263         Length:263
#>  Class :character   Class :character
#>  Mode  :character   Mode  :character
#>
#>
#>
```

Function:truncate data - *LTDS_crop*

So that we can use the data to estimate the density of tortoises at the site, we are going to slightly format the input. Specifically, we are going to truncate the data so that any burrows found over half the distance between the lines are removed; and we are also going to remove the top distance outliers (5%).

The function we are using simply needs two bits of information. We need the data from the full survey, and we also need to set the desired truncation distance - this will always be half the distance between transects. For our tutorial, that is 25m.

```
data.new <- LTDS_crop(ltds_data = full.data, trunc = 25)

#We can summarize the data again for comparison.
summary(data.new)
#>   Transect_ID     Burrow_width      Distance          Occupied
#>  Min.   : 1.00   Min.   : 50.5   Min.   : 0.02674   Length:249
#>  1st Qu.: 5.00   1st Qu.:149.6   1st Qu.: 1.64439   Class :character
#>  Median :12.00   Median :246.1   Median : 3.51532   Mode  :character
#>  Mean   :10.96   Mean   :211.6   Mean   : 3.94488
#>  3rd Qu.:16.00   3rd Qu.:279.8   3rd Qu.: 6.06748
#>  Max.   :22.00   Max.   :307.9   Max.   :10.20368
#>  Burrow_condition    Commensals
#>  Length:249         Length:249
#>  Class :character   Class :character
#>  Mode  :character   Mode  :character
#>
#>
#>
```

Function:estimate the density - *dens_est*

We are finally at the penultimate function of the LTDSMethod package. We are now ready to calculate the density of tortoises at the recipient site.

For the function to work and estimate the density, we need to provide three main bits of information. We need the cropped ltds data from *LTDS_crop*, the effort resulting from *samp_effort*, and lastly we need the total area of the recipient site in acres.

```
tort.dens <- dens_est(ltds_data = data.new, effort = effort, area = 400)
#> Defining model
#> Building model
#> Setting data and initial values
#> Running calculate on model
#>   [Note] Any error reports that follow may simply reflect missing values in model variables.
#> Checking model sizes and dimensions
#>   [Note] This model is not fully initialized. This is not an error.
#>          To see which variables are not initialized, use model$initializeInfo().
#>          For more information on model initialization, see help(modelInitialization).
#> Checking model calculations
#> [Note] NAs were detected in model variables: dist.a, logProb_dist.a, sigma, logProb_sigma, sigma2, o
#> Compiling
#>   [Note] This may take a minute.
#>   [Note] Use 'showCompilerOutput = TRUE' to see C++ compilation details.
#> Running chain 1 ...
#> |-------------|-------------|-------------|-------------|
#> |-------------------------------------------------------|
#> Running chain 2 ...
#> |-------------|-------------|-------------|-------------|
#> |-------------------------------------------------------|
#> Running chain 3 ...
#> |-------------|-------------|-------------|-------------|
#> |-------------------------------------------------------|

#Let's look at the resulting density:
tort.dens
#> [1] 1.678087

#This means that our total number of tortoises estimated at the site is:
tort.dens*400
#> [1] 671.2349
```

Our final density estimate at the site is 1.68 tortoises per acre, which is approximately 671 tortoises at our site. This is fabricated data to fit in with the vignette training, so the accuracy of this estimation is unknown. See the SOP for simulation results.

Function:produce LTDS final report - *LTDS_summary*

Now we can produce our final report. There is slightly more manual input for the report, since we wanted to make this adaptable to changes in data and reuslts which cannot be captured by the LTDSMethod package. The parameters of the function and report are as follows:

```
#effort: the effort from the full LTDS survey. We will use the pilot effort that
#we calculated earlier for this example.
effort <- effort
#nburr: the total number of burrows found during the survey. We can axtract this
#from our example dataset, which will likely be applicable for future use.
nburr <- length(full.data$Burrow_width)
#burr.w.mn: the mean width of detected burrows.
burr.w.mn <- round(mean(full.data$Burrow_width), digits = 2)
#burr.w.min: the minimum width of detected burrows.
burr.w.min <- round(min(full.data$Burrow_width), digits = 2)
#burr.w.max: the maximum width of detected burrows.
burr.w.max <- round(max(full.data$Burrow_width), digits = 2)
#comm.comm: the name of the most common commensal species detected. We can
#extract the number of detections for the full dataset by:
table(full.data$Commensals)
#>
#>                    Burrowing owl Eastern diamondback rattlesnake
#>                                1                               1
#>              Florida gopher frog                   Florida mouse
#>                                3                               5
#>                      Indigo snake
#>                                1
#This shows that the most common species was the Florida mouse
comm.comm <- "Florida mouse"
#occ: the number of occupied burrows discovered.
#unocc: the number of unoccupied burrows discovered.
#unk: the number of unknown burrows discovered.
#We can use the same method as above to pull these numbers out of the data
table(full.data$Occupied)
#>
#>      No Unknown     Yes
#>     131      25     107
occ <- 107
unocc <- 131
unk <- 25
#site.name: the name of the recipient site. We can set this to whatever we want.
site.name <- "Tortoise Wonderland"
#tort.dens: the estimated tortoise density at the site. We calculated this earlier.
tort.dens <- round(tort.dens, digits = 2)
#tort.pop: the estimated tortoise population at the site. We calculated this earlier.
tort.pop <- round(tort.dens*400)
```

The above information is used for the text part of the documents, however, any figures produced in the report will pull data directly from the working directory.

We can now input all of these into our function. As a warning, if you have previously run this function, and viewed the resulting pdf, the function will not run if you still have the pdf open.

```
LTDS_summary(effort = effort, nburr = nburr, burr.w.mn = burr.w.mn,
             burr.w.min = burr.w.min, burr.w.max = burr.w.max, comm.comm = comm.comm,
             occ = occ, unocc = unocc, unk = unk, site.name = site.name,
             tort.dens = tort.dens, tort.pop = tort.pop)
```

End of vignette.

Final LTDS Report

2023-01-05

LTDS Summary - Tortoise Wonderland

Final report for line transect distance sampling survey with burrow scoping.

We walked a total of **19868.67** m during our full LTDS survey, where we found a total of **263** burrows. We calculated a density of **1.66** tortoises/acre, resulting in approximately **664** tortoise at the recipient site. We calculated a mean burrow width of **210.02** mm (range = **50.5 - 307.94** mm).

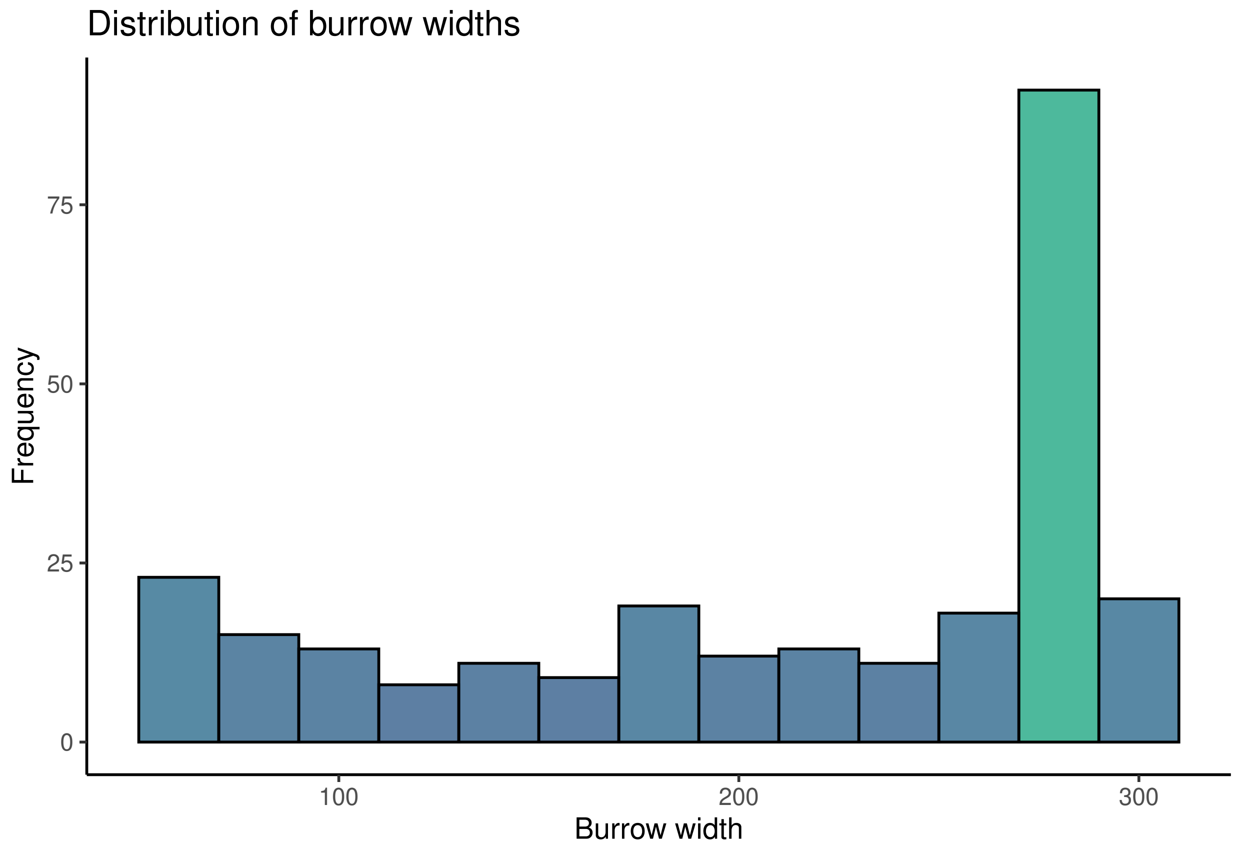


Figure 1. Histogram representing the distribution of burrow width measurements from detected burrows during the full LTDS survey. Data includes measurements from all burrow conditions and occupancy status.

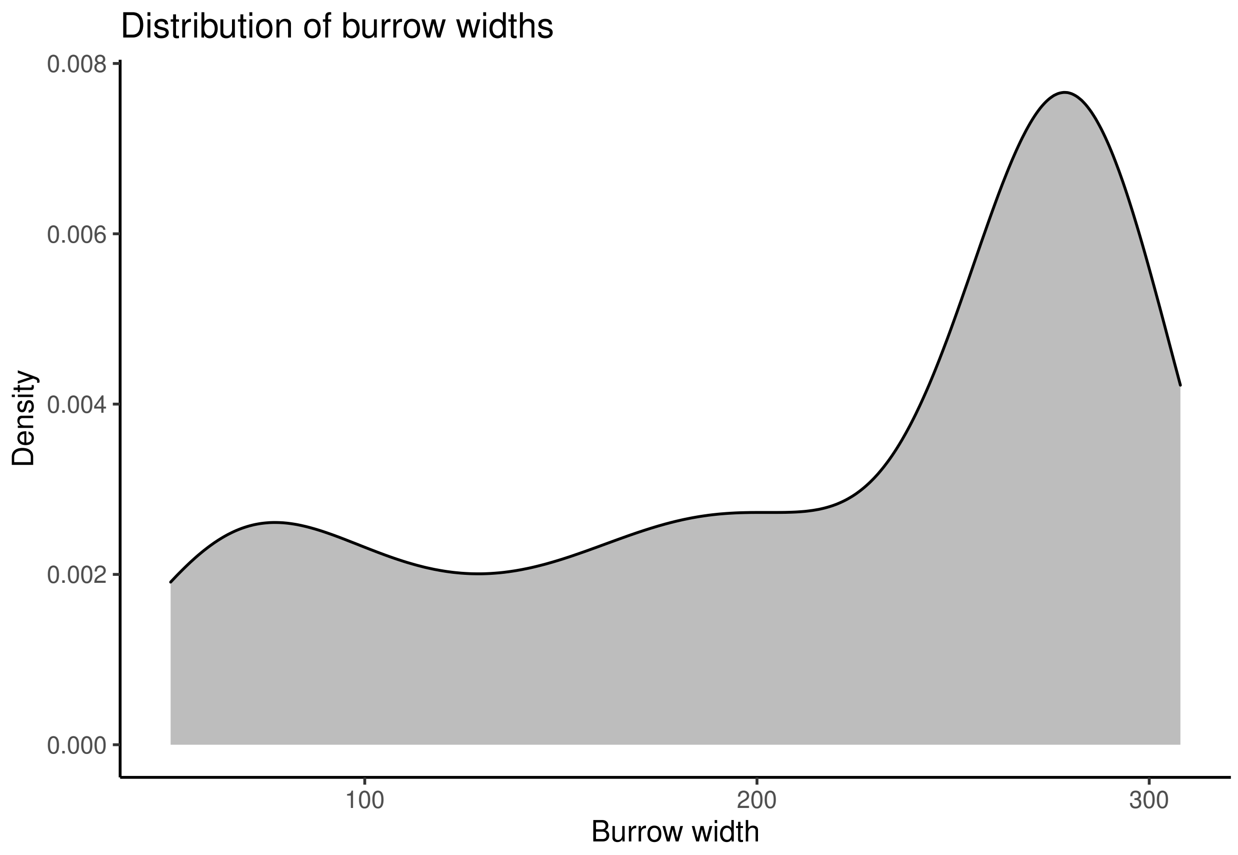


Figure 2. Density plot representing the distribution of burrow width measurements from detected burrows during the full LTDS survey. Data includes measurements from all burrow conditions and occupancy status.

We found **107 occupied** burrows, **131 unoccupied** burrows, and **25 unknowns**.

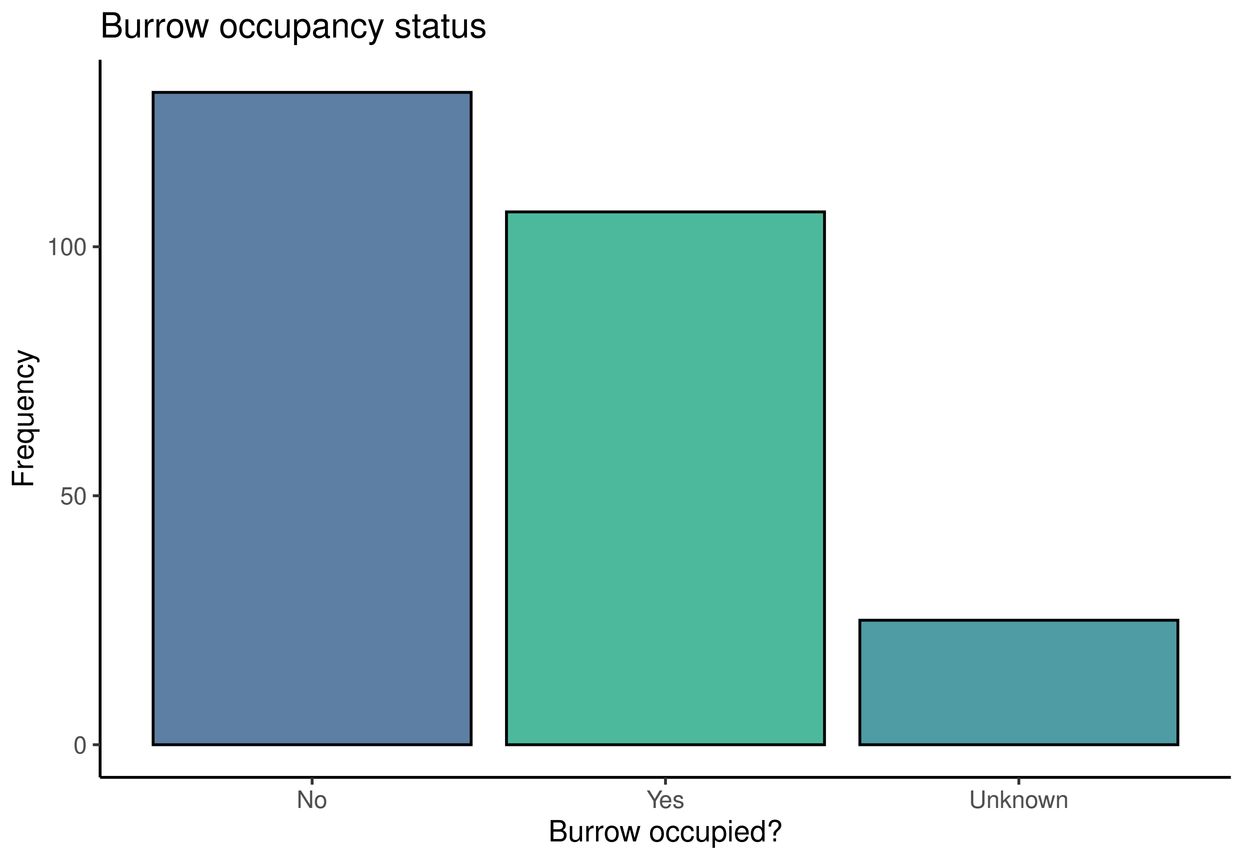


Figure 3. Histogram showing the occupancy status from detected burrows during the full LTDS survey.

The most common commensal species discovered in burrows during scoping was the **Florida mouse**. A breakdown of the commensal species detection can be seen in the following table and figure:

Table 1. Name and total number of commensal species found inside gopher tortoise burrows during full LTDS survey.

Commensal species discovered

Species	No.found
Indigo snake	1
Burrowing owl	1
Eastern diamondback rattlesnake	1
Florida gopher frog	3
Florida mouse	5

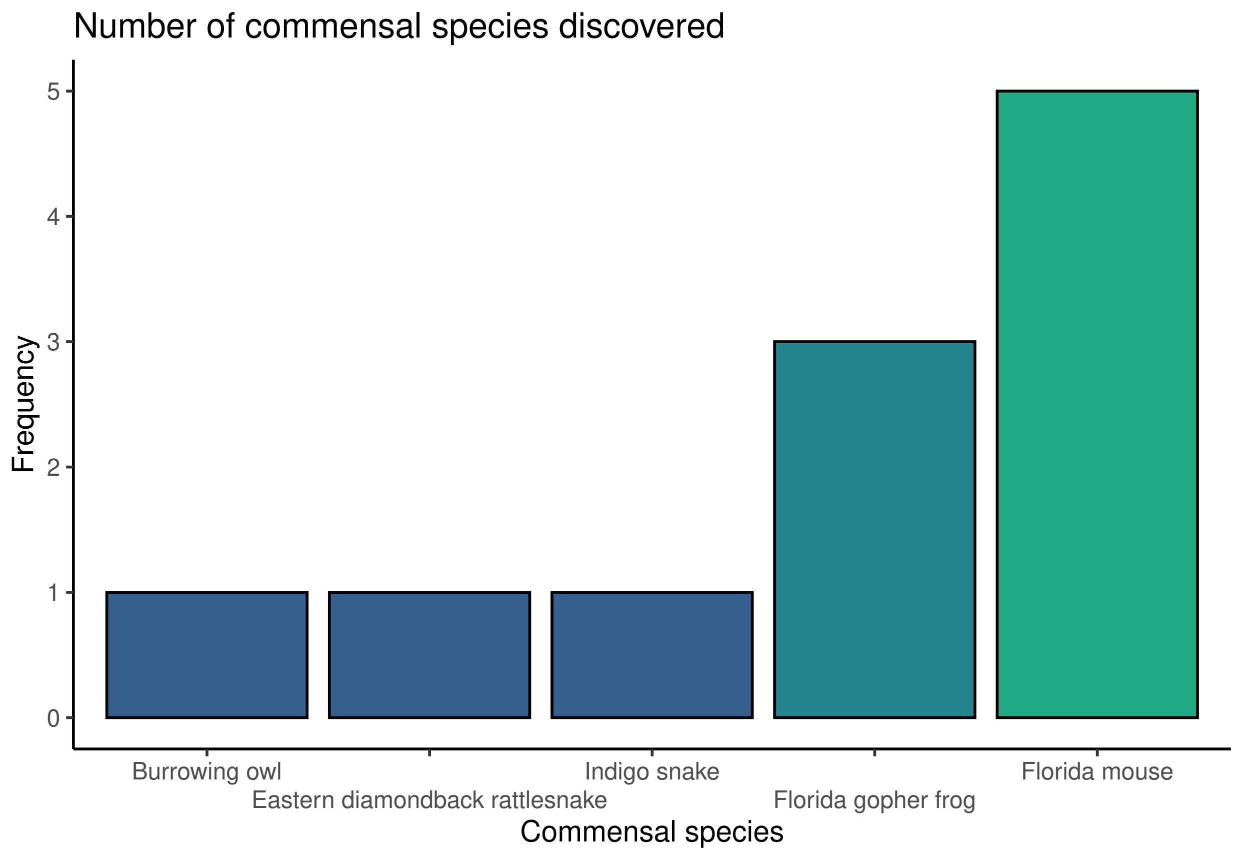


Figure 4. Histogram showing the unmber of commensal species found within detected burrows during the full LTDS survey.

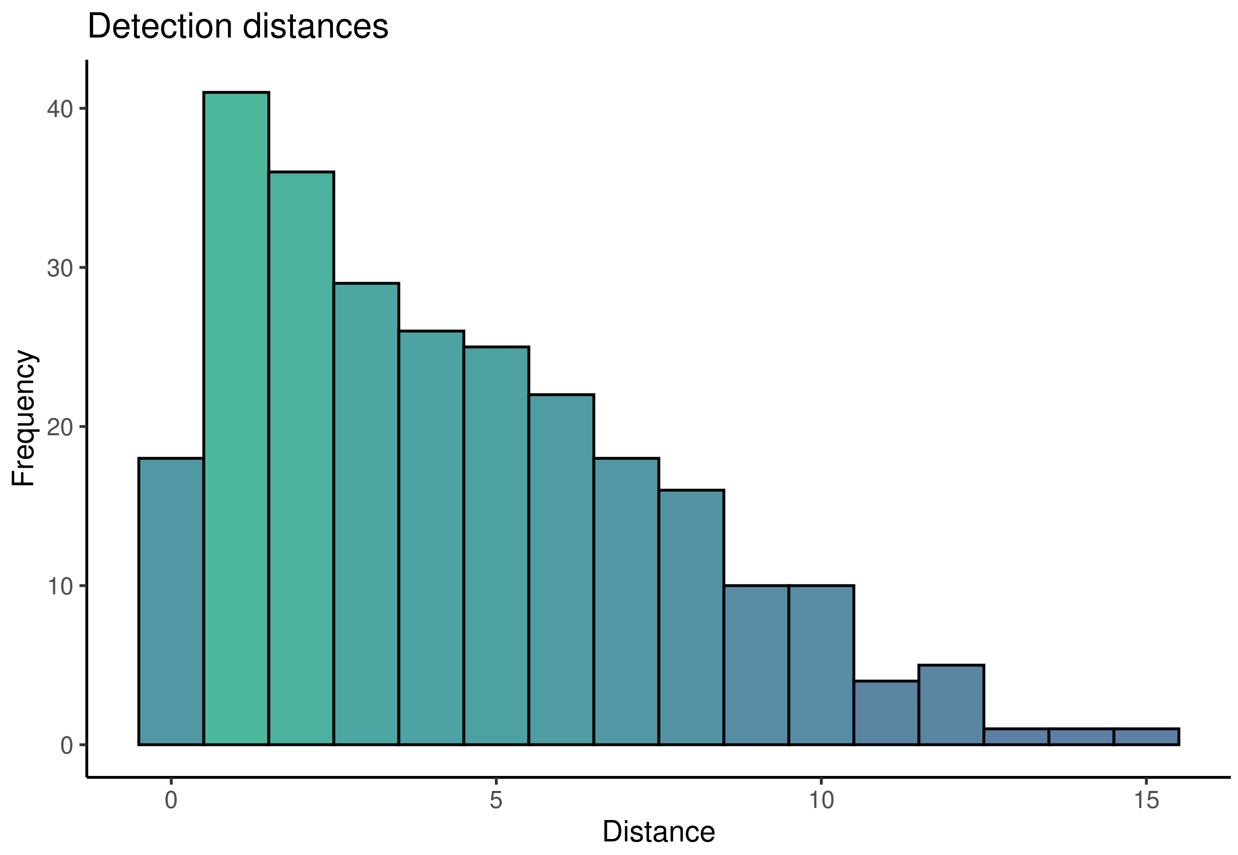


Figure 5. Histogram showing the perpendicular distance from transect lines of all detected burrows during the full LTDS survey.

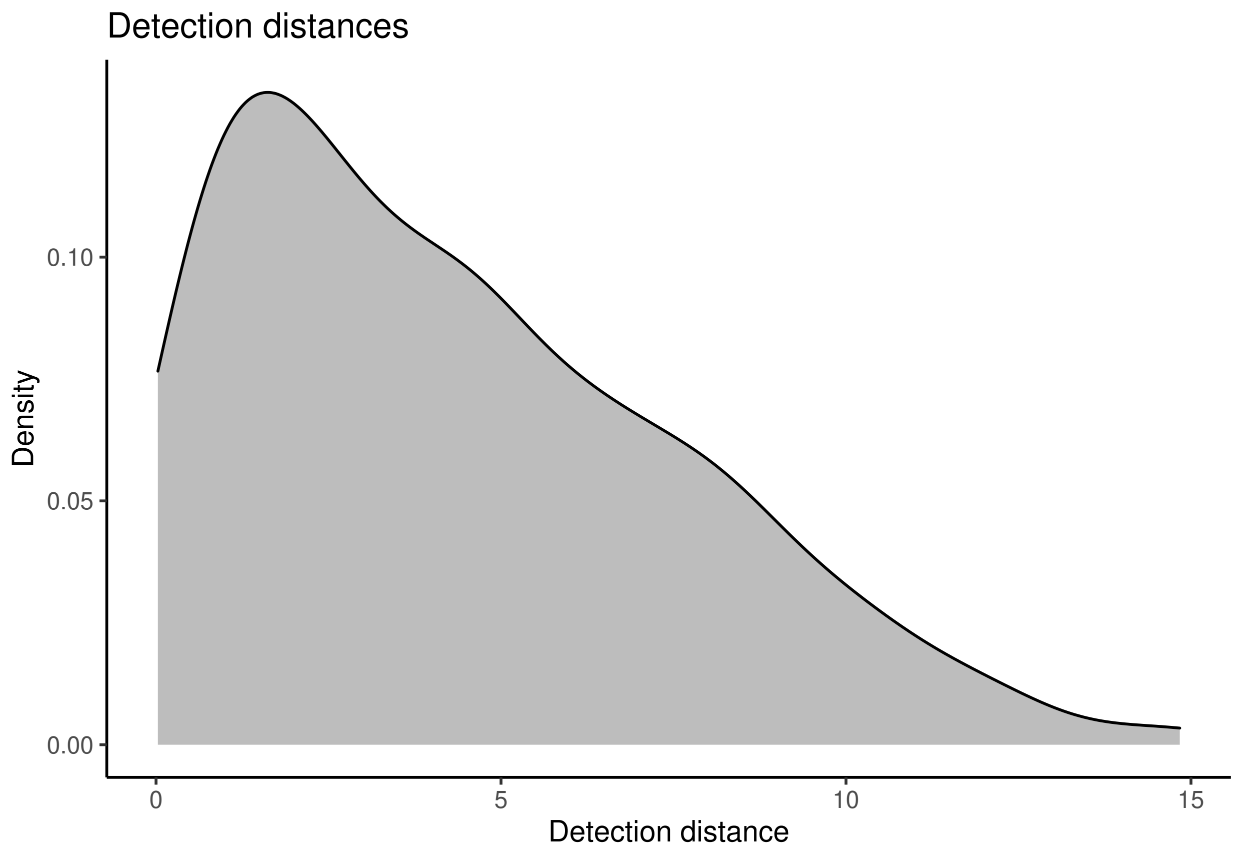


Figure 6. Density plot showing the perpendicular distance from transect lines of all detected burrows during the full LTDS survey.

End of report.