

# pannotator examples

Nunzio Knerr      Robert Godfree

2024-11-29

## Table of contents

<b>1 Data analysis for pannotator paper</b>	<b>2</b>
1.1 Load libraries . . . . .	2
1.2 Read species richness, spinifex cover and tree health .rds files . . . . .	2
1.3 Add plot and frame numbers to each .rds file . . . . .	3
1.4 Map the location of data points in each file . . . . .	4
<b>2 Analysis of species distribution and richness data</b>	<b>6</b>
2.1 Clean up the species data (df_species) for mapping and analysis . . . . .	6
2.2 Map the species distributions (generate image for Figure 5a in paper) . . . . .	9
2.3 Make a 25 m buffer around each plot, create a polygon, segment it and calculate species diversity in each segment . . . . .	10
2.4 Make buffer around each point and create polygon(s) . . . . .	11
2.5 Import the .kml file for segmenting the polygon(s) and cut the polygon(s) . . . . .	11
2.6 Calculate the species richness in each of the new polygons . . . . .	13
2.7 Plot the centroids of each polygon with species richness of the polygon (image for Figure 5f) . . . . .	14
<b>3 Analysis of spinifex data</b>	<b>15</b>
3.1 Check the data and correct outliers . . . . .	15
3.2 Map the total cover of live and dead cover within each image frame (images for Figures 6c and 6d) . . . . .	16
3.3 Map the cover of different spinifex size classes within each image frame (image for Figure 6e) . . . . .	19
3.4 Re-name spinifex species/size data for size class analysis . . . . .	19
3.5 Calculate and map the cover and distribution of < 50cm and fragmentary (old > 1m) size classes (image for Figure 6e) . . . . .	20

3.6	Create boxplot of size class vs pre-drought and post-drought percentage cover with kruskal-wallis test (image for Figure 6f) . . . . .	21
<b>4</b>	<b>Analysis of desert oak crown health data</b>	<b>23</b>
4.1	Import the lookup csv containing size classes and match to size class data in the df_oak object . . . . .	23
4.2	Calculate crown health score and compare crown health across size classes (create image for Figure 7f) . . . . .	24
4.3	Map burnt and unburnt tree crowns (image for Figure 7e) . . . . .	26
4.4	Map the distribution of tree size classes (image for Figure 7c) . . . . .	27
4.5	Plot the crown health of mature and early mature size classes (Figure 7d) . .	28

# 1 Data analysis for pannotator paper

## 1.1 Load libraries

This code loads in the libraries needed.

```

1 library(dplyr)
2 library(ggplot2)
3 library(mapview)
4 library(RColorBrewer)
5 library(rcompanion)
6 library(sf)
7 library(stringr)
8 library(webshot)
```

## 1.2 Read species richness, spinifex cover and tree health .rds files

This code reads in the .Rds files containing species, spinifex cover and tree crown data.

```

1 # Read in the .Rds files containing species, spinifex cover and tree
  ↳ crown data
2
3 # If you have exported an .Rds file from PannotatoR
4 df_species <- readRDS("Data_files/df_species.Rds")
5
6 # If you have exported an .csv file from PannotatoR
```

```

7 df_spinifex <- read.csv(file = "Data_files/df_spinifex.csv", header =
8   TRUE, sep = ",")
9 df_oak <- readRDS("Data_files/oak_data_user1.Rds")

```

### 1.3 Add plot and frame numbers to each .rds file

Now we add the plot and information to each frame in the .rds file. This is useful code if you have multiple plots or subplots in a single .rds file. In this example we have only one plot (plot\_06\_new). A .csv file linking plot ID to frame or .kmz file ID is required.

```

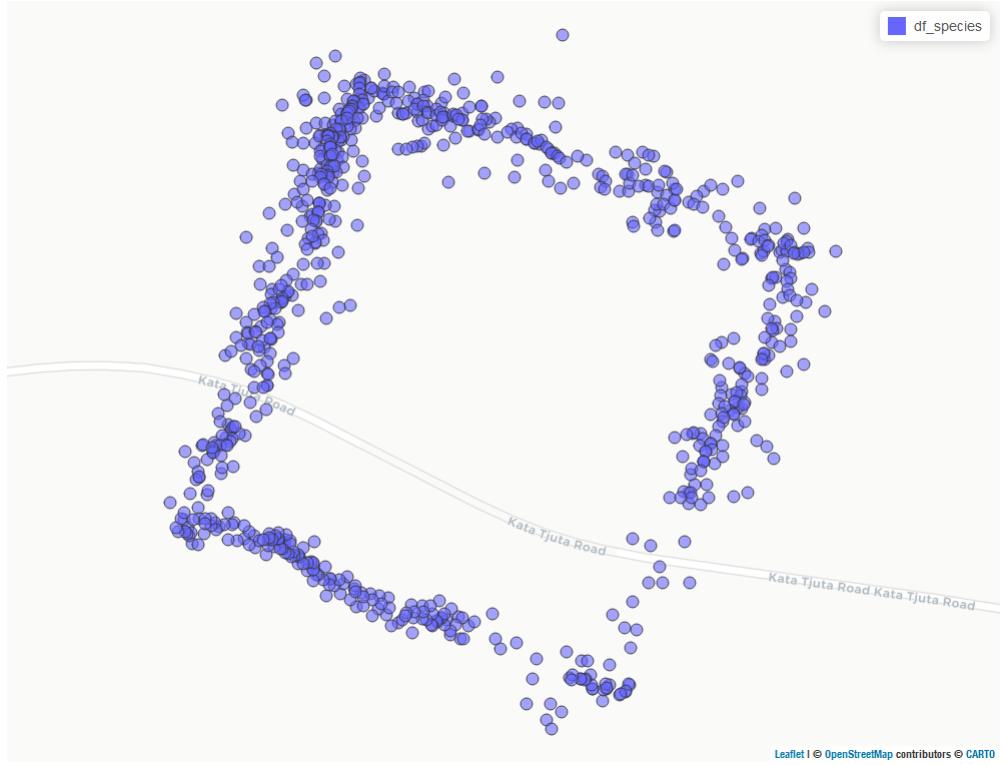
1 # Now we add the plot and information to each frame in the .rds file.
2   ↵ This is useful code if you have multiple plots or subplots in a
3   ↵ single .rds file. In this example we have only one plot
4   ↵ (plot_06_new). A .csv file linking plot ID to frame or .kmz file
5   ↵ ID is required
6
7
8 # Import the csv file
9 df_plot_number <- read.csv(file = "Data_files/Plot_names.csv")
10
11 # This function adds a plot_kmz column, fills it with a substring from
12   ↵ the imagefile column, add a frame column with a substring from the
13   ↵ imagefile column, and then match the plot number to the plot_kmz
14   ↵ in the df_plot_number dataframe.
15
16 add_plot_kmz <- function(df) {
17   df$plot_kmz <- substr(df$imagefile,1,24)
18   df$frame <- substring(df$imagefile,1,37 )
19   df$plot_name_new <- df_plot_number$plot_name_new[match(df$plot_kmz,
20     ↵ df_plot_number$plot)]
21   return(df)
22 }
23
24
25 df_species <- add_plot_kmz(df_species)
26 df_spinifex <- add_plot_kmz(df_spinifex)
27 df_oak <- add_plot_kmz(df_oak)

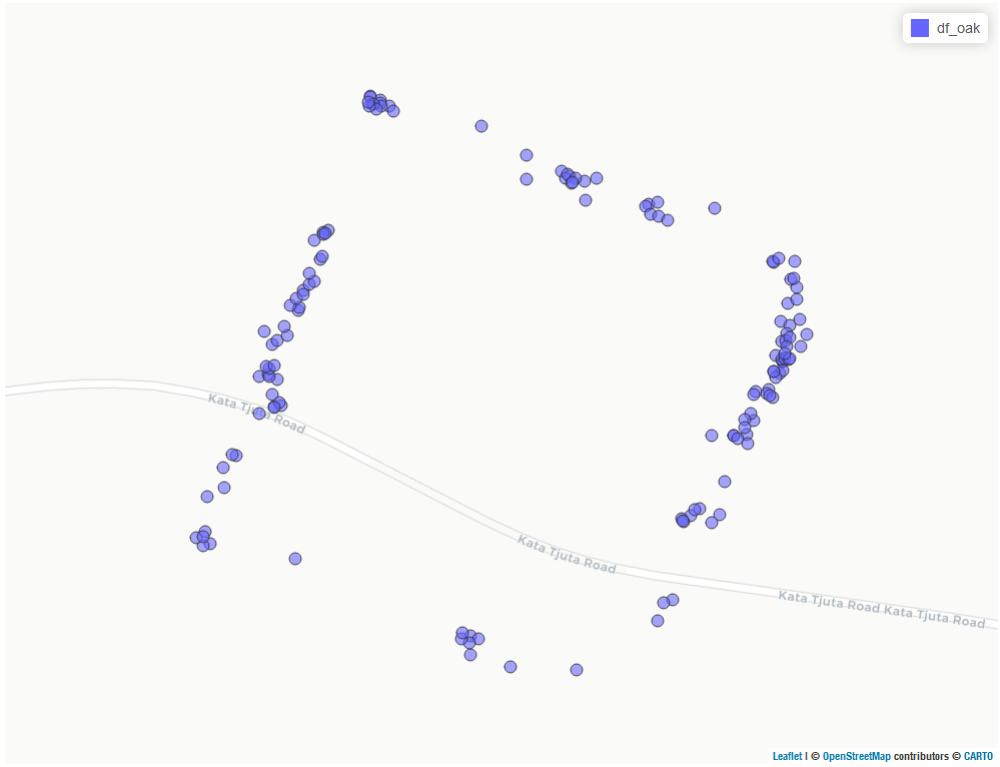
```

## 1.4 Map the location of data points in each file

This code maps the points for a quick look at the data.

```
1 # Using the geometry field
2 df_species <- st_as_sf(df_species, wkt = "geometry", crs = 4326)
3 s1 <- mapview(df_species)
4
5 mapshot(s1, file = "Data_files/species_map.png")
6
7 df_spinifex <- st_as_sf(df_spinifex, wkt = "geometry", crs = 4326)
8 s2 <- mapview(df_spinifex)
9
10 mapshot(s2, file = "Data_files/spinifex_map.png")
11
12 df_oak <- st_as_sf(df_oak, wkt = "geometry", crs = 4326)
13 s3 <- mapview(df_oak)
14
15 mapshot(s3, file = "Data_files/desertoak_map.png")
16
17 if (knitr:::is_latex_output()) {
18   # This part runs only for PDF output
19 } else if (knitr:::is_html_output()) {
20   # This part runs only for HTML output
21   s1
22   s2
23   s3
24 } else {
25   # This part runs if neither PDF nor HTML is the target format
26   s1
27   s2
28   s3
29 }
```





## 2 Analysis of species distribution and richness data

### 2.1 Clean up the species data (df\_species) for mapping and analysis

```

1 # View(df_species)
2 # Create a new species variable
3 df_species$Species_new <- NA
4
5 # If variable dd3 = -999, then add species (mulga information),
#   otherwise keep the species listed in dd3. This annotation was
#   used to quantify mulga cover and also to add additional species #
6 df_species$Species_new <- ifelse(df_species$dd3 == -999,
#   df_species$species, df_species$dd3)
7
8
9 # For simplicity of data analysis we must now rename the species in
#   the species_new variable

```

```

10
11 df_species$Species_new[df_species$Species_new == 'Acacia_aneura
12   ↵ (mulga)'] <- 'Acacia_aneura'
13
14 df_species$Species_new[df_species$Species_new == 'Acacia_ligulata
15   ↵ (dune wattle)'] <- 'Acacia_ligulata'
16
17 df_species$Species_new[df_species$Species_new ==
18   ↵ 'Allocasuarina_decaisneana (oak)'] <- 'Allocasuarina_decaisneana'
19
20 df_species$Species_new[df_species$Species_new == 'Burnt Mulga Mature
21   ↵ (3-6m)'] <- 'Acacia_aneura'
22
23 df_species$Species_new[df_species$Species_new == 'Burnt Mulga Juvenile
24   ↵ (1-3m)'] <- 'Acacia_aneura'
25
26 df_species$Species_new[df_species$Species_new ==
27   ↵ 'Codonocarpus_cotinifolius (poplar)']
28   ↵ <- 'Codonocarpus_cotinifolius'
29
30 df_species$Species_new[df_species$Species_new ==
31   ↵ 'Grevillea_eriostachya (honey grevillea)']
32   ↵ <- 'Grevillea_eriostachya'
33
34 df_species$Species_new[df_species$Species_new ==
35   ↵ 'Grevillea_stenobotrys (sandhill grevillea)']
36   ↵ <- 'Grevillea_stenobotrys'
37
38 df_species$Species_new[df_species$Species_new == 'Gyrostemon_ramulosus
39   ↵ (camel poison)'] <- 'Gyrostemon_ramulosus'
40
41 df_species$Species_new[df_species$Species_new == 'Mulga Juvenile
42   ↵ (1-3m)'] <- 'Acacia_aneura'
43
44 df_species$Species_new[df_species$Species_new == 'Mulga Mature
45   ↵ (3-6m)'] <- 'Acacia_aneura'

```

```

36
37 df_species$Species_new[df_species$Species_new == 'Mulga Seedling
38   ↵ (<1m)'] <- 'Acacia_aneura'
39
40 df_species$Species_new[df_species$Species_new == 'Santalum_lanceolatum
41   ↵ (sandalwood)'] <- 'Santalum_lanceolatum'
42
43 df_species$Species_new[df_species$Species_new == 'Themedea_australis
44   ↵ (kangaroo grass)'] <- 'Crotalaria_cunninghamii'
45
46 df_species$Species_new[df_species$Species_new == 'Themedea_avenaceae
47   ↵ (oat grass)'] <- 'Crotalaria_cunninghamii'
48
49 df_species$Species_new[df_species$Species_new == 'Acacia_kempeana
50   ↵ (granite wattle)'] <- 'Acacia_kempeana'
51
52 df_species$Species_new[df_species$Species_new == 'Corymbia_opaca
53   ↵ (desert bloodwood)'] <- 'Corymbia_opaca'
54
55 df_species$Species_new[df_species$Species_new ==
56   ↵ 'Eucalyptus_gamophylla (blue mallee)'] <- 'Eucalyptus_gamophylla'
57
58 # We now Clean up a few more cases. We don't want any data points that
59   ↵ contain no species data, unknown species, or species with unclear
60   ↵ taxonomy (e.g., Hakea spp.)
61
62 df_species <- subset(df_species, Species_new != "No Mulga present")
63 df_species <- subset(df_species, Species_new !=
64   ↵ "Re_Check_Unknown_Shrub")
65 df_species <- subset(df_species, Species_new != "Hakea_sp")
66 df_species <- subset(df_species, Species_new != "Acacia_sp")
67 df_species <- subset(df_species, Species_new != "Gyrostemon_sp.")
68
69 # Remove species and location duplicates
70 duplicated_rows <- duplicated(df_species[c("Species_new",
71   ↵ "geometry")])
72 df_species<- df_species[!duplicated_rows, ]
73
74 # We must now jitter the geometries so that we can see the data easier
75   ↵ in mapview. Many of the locations are very close together
76 df_species_jittered <- st_jitter(df_species, amount = 0.00001)

```

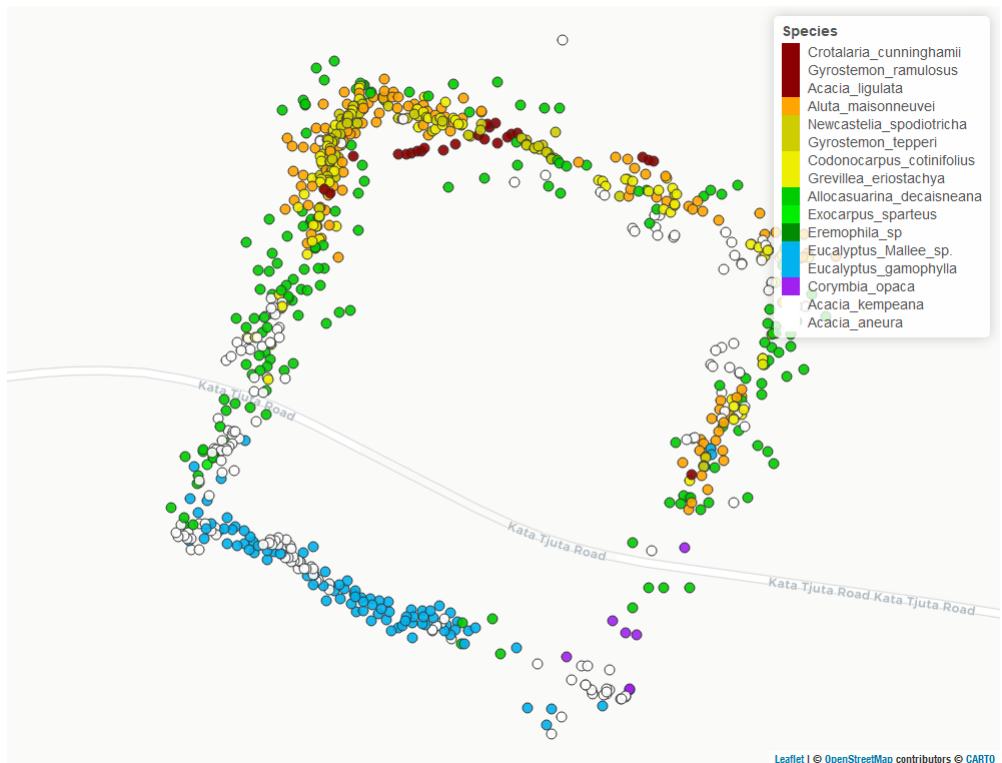
## 2.2 Map the species distributions (generate image for Figure 5a in paper)

This is useful code for plotting species distributions in mapview, including setting colour schemes within a user-specified species order.

```
1 # This code generates Figure 5A
2
3 # Confirm the data
4 #mapview(df_species_jittered, zcol = "Species_new", cex = 8, alpha =
4   ↵ 0.9)
5
6 # Here I reorder the species from sand dune crest species to swale
6   ↵ species (approximately)
7
8 df_species_jittered$speciesReorder <-
8   ↵ factor(df_species_jittered$Species_new,
8   ↵ c('Crotalaria_cunninghamii','Gyrostemon_ramulosus',
8   ↵ 'Acacia_ligulata', 'Aluta_maisonneuvei',
8   ↵ 'Newcastelia_spodiotoricha',
8   ↵ 'Gyrostemon_tepperi','Codonocarpus_cotinifolius',
8   ↵ 'Grevillea_eriostachya', 'Allocasuarina_decaisneana',
8   ↵ 'Exocarpus_sparteus', 'Eremophila_sp', 'Eucalyptus_Mallee_sp.',
8   ↵ 'Eucalyptus_gamophylla', 'Corymbia_opaca', 'Acacia_kempeana'
8   ↵ , 'Acacia_anera'))
9
10 # Here I colour species groups according to their typical ecological
10   ↵ position; e.g., sand dune crest species in red, swale species in
10   ↵ white
11 p <- mapview(df_species_jittered, zcol = "speciesReorder", cex = 5,
11   ↵ alpha = 1, alpha.regions = 0.9, layer.name = "Species",
11   ↵ col.regions = c("red4","red4", 'red4', "orange",
11   ↵ 'yellow3','yellow3',
11   ↵ 'yellow2','yellow2','green3','green2','green4', 'deepskyblue2',
11   ↵ 'deepskyblue2','purple','white','white'),
11   ↵ na.rm = TRUE)
12
13 mapshot(p, file = "Data_files/Fig_5a.png")
14
15
16
17 if (knitr:::is_latex_output()) {
17   # This part runs only for PDF output
```

```

19 } else if (knitr:::is_html_output()) {
20   # This part runs only for HTML output
21   p
22 } else {
23   # This part runs if neither PDF nor HTML is the target format
24   p
25 }
```



## 2.3 Make a 25 m buffer around each plot, create a polygon, segment it and calculate species diversity in each segment

This is useful code to add a buffer around each data point in the dataframe, combined these together if overlapping into a series of polygons. The intent is to then split up the polygons according to another spatial dataframe, in this case a .kml containing lines that divide the west side of the transect into 50 m segments with approximately homogeneous vegetation.

## 2.4 Make buffer around each point and create polygon(s)

```
1 # Add a spatial buffer of 25m around each point
2 buffer <- st_buffer(df_species_jittered, dist = 50)
3
4 # Combine the resulting buffers into a single polygon
5 polygon <- st_union(buffer)
6
7 # mapview(polygon) # we can see that the buffers have been merged into
    ↵ a series of polygons.
8
9 # Export the resulting polygon as a .kml file
10 st_write(polygon, "Data_files/buffer_plot_polygon.kml", append= FALSE,
    ↵ driver = "KML")
```

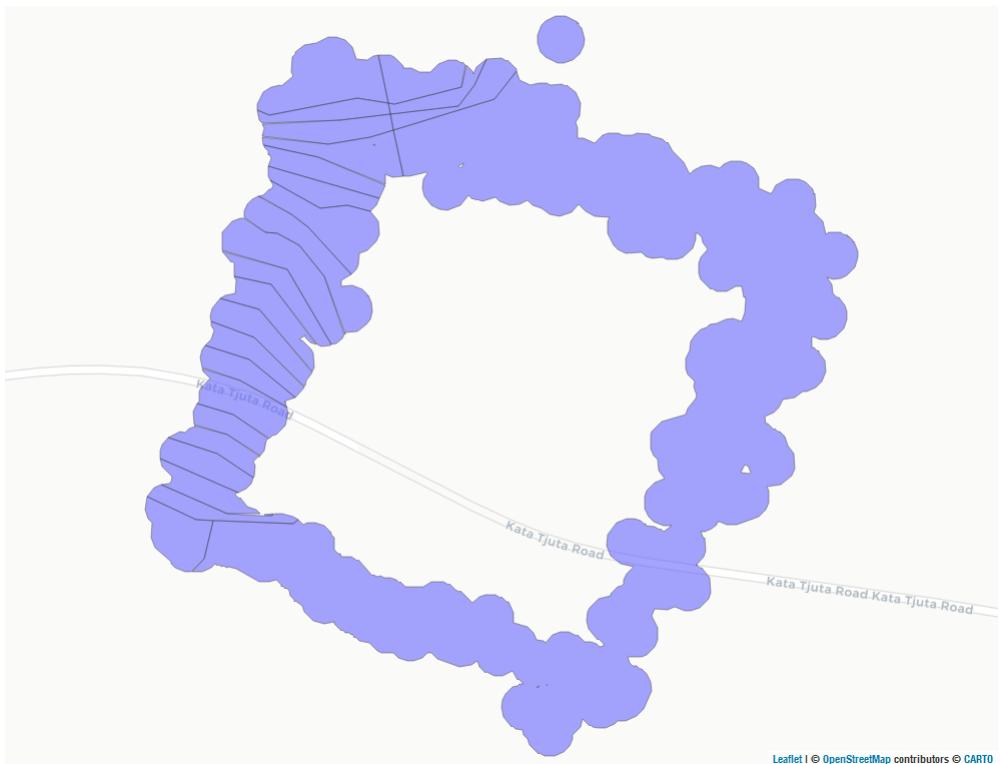
## 2.5 Import the .kml file for segmenting the polygon(s) and cut the polygon(s)

```
1 # This is.kml file that contains lines that cut the westerly side of
    ↵ the study transect into a series of segments roughly 50 m apart.
    ↵ Each segment was oriented to contain homogenous vegetation. It was
    ↵ drawn in Google Earth.
2
3 segmentation_lines <- st_read("Data_files/polygon_cuts.kml")
4
5 # Map the lines
6 # mapview(segmentation_lines)
7
8 # Before we cut the polygons using the segmentation lines we
    ↵ mustrepair the geometry of the polygon and lines data for any
    ↵ errors
9 polygon_valid <- st_make_valid(polygon)
10 # mapview(polygon_valid)
11
12 lines_valid <- st_make_valid(segmentation_lines)
13 # mapview(lines_valid)
14
15 # Now cut the polygon using the segmentation lines
```

```

16 cut_polygon <- polygon_valid %>%
17   lwgeom::st_split(lines_valid) %>%
18   st_collection_extract("POLYGON")
19
20 p <- mapview(cut_polygon)
21
22 mapshot(p, file = "Data_files/cut_polygon.png")
23
24 if (knitr:::is_latex_output()) {
25   # This part runs only for PDF output
26 } else if (knitr:::is_html_output()) {
27   # This part runs only for HTML output
28   p
29 } else {
30   # This part runs if neither PDF nor HTML is the target format
31   p
32 }

```



## 2.6 Calculate the species richness in each of the new polygons

```
1 # Now we will calculate the number of species in each polygon
2   ↵ segment.We are interested in polygons with IDs.
3
4 # The cut_polygon object is an 'sfc' object and we want to make it an
5   ↵ 'sf' object
6 class(cut_polygon)
7
8
9 # Create a new data frame
10 df <- data.frame(id = 1:length(cut_polygon))
11
12 # Add the sfc_POLYGON object as a geometry column
13 df$geometry <- cut_polygon
14
15 # Convert the data frame to an sf object
16 cut_polygon_sf <- st_as_sf(df)
17
18 # Now an sf object with 72 polygons.
19 # str(cut_polygon_sf)
20
21 st_write(cut_polygon_sf, "Data_files/cut_polygon_plot_sf.kml", append=
22   ↵ FALSE, driver = "KML")
23
24 # Now determine which polygon each point falls into
25 result <- st_join(df_species_jittered, cut_polygon_sf)
26
27 # Convert the polygon ID to character
28 result$id <- as.character(result$id)
29
30 # Now we calculate the species diversity of each polygon
31
32 species_div <- result %>%
33   group_by(id) %>%
34   summarise(num_species = n_distinct(speciesReorder))
35
36 species_div
```

## 2.7 Plot the centroids of each polygon with species richness of the polygon (image for Figure 5f)

```
1 # To enable plotting we calculate the centroids of each polygon
2 average_geometry <- st_centroid(species_div$geometry)
3 species_div_centroids <- st_set_geometry(species_div,
4   ↵ average_geometry)
5
6 # mapview(species_div_centroids)
7
8 # We can select the centroids along the west side of the transect
9   ↵ using this code if we want, default is to plot all
10
11 # id_values <- c(14:23, 25:38, 2:7, 9)
12 # Species_div_centroids_subset <-
13   ↵ species_div_centroids[species_div_centroids$id %in% id_values, ]
14
15 # Rename
16 species_div_centroids_subset <- species_div_centroids
17
18 # Removed the very large unsegmented polygon
19 species_div_centroids_subset <-
20   ↵ species_div_centroids_subset[species_div_centroids_subset$num_species
21   ↵ < 7, ]
22
23 # Check the data
24 # mapview(species_div_centroids_subset)
25
26 mapviewOptions(basemaps = c("Esri.WorldImagery"),
27                 raster.palette = grey.colors,
28                 vector.palette = colorRampPalette(c("white", "red3")),
29                 na.color = "gray",
30                 layers.control.pos = "topright")
31
32 q <- mapview(species_div_centroids_subset, zcol = "num_species",
33   ↵ layer.name = "Richness", cex = 8, alpha = 0.9, alpha.regions =
34   ↵ 0.9, na.rm = TRUE)
35
36 mapshot(q, file = "Data_files/Fig_5f.png")
```

```

32 if (knitr:::is_latex_output()) {
33   # This part runs only for PDF output
34 } else if (knitr:::is_html_output()) {
35   # This part runs only for HTML output
36   q
37 } else {
38   # This part runs if neither PDF nor HTML is the target format
39   q
40 }

```



## 3 Analysis of spinifex data

### 3.1 Check the data and correct outliers

Here we are interested in analysing two variables: the % cover of live and dead material within the 10 m diameter plots.

```

1 # Ensure variables are numeric
2 df_spinifex$live_cover<- as.numeric(df_spinifex$live_cover)
3 df_spinifex$dead_cover<- as.numeric(df_spinifex$dead_cover)
4
5 # Check the range of the data fields of interest. Percentage cover is
6 #   the % cover within the 10 m diameter plots, impact score is based
7 #   on the percentage of dead versus live material.
8
9 # Within range
10 summary(df_spinifex$live_cover)
11 # Within range
12 summary(df_spinifex$dead_cover)
13
14 # Generate an object with total live and dead cover in each frame
15 frame_sum <- df_spinifex %>%
16   group_by(frame) %>%
17   summarise(across(c(dead_cover, live_cover), sum, .names = "{.col}"))
18
19 # Calculate the total spinifex cover variable
20 frame_sum$total_cover <- frame_sum$dead_cover+ frame_sum$live_cover
21
22 frame_sum

```

### 3.2 Map the total cover of live and dead cover within each image frame (images for Figures 6c and 6d)

```

1 # Rename the object
2 data <- frame_sum
3
4 # Set the mapview options for live cover
5 mapviewOptions(basemaps = c("Esri.WorldImagery"),
6                 raster.palette = grey.colors,
7                 vector.palette = colorRampPalette(c("transparent",
8                                           "green", 'darkgreen')),
9                 na.color = "gray",
10                layers.control.pos = "topright")
11
12 # Plot of live cover
13 p <- mapview(data, zcol = "live_cover", layer.name = "Cover(%)", cex =
14   10, alpha = 0.9, alpha.regions = 0.9, na.rm = TRUE)

```

```

13
14 #p
15
16 mapshot(p, file = "Data_files/Fig_6c.png")
17
18 # Set the mapview options for dead cover
19 mapviewOptions(basemaps = c("Esri.WorldImagery"),
20                 raster.palette = grey.colors,
21                 vector.palette = colorRampPalette(c("white", "red",
22                                         "darkred")),
22                 na.color = "gray",
23                 layers.control.pos = "topright")
24
25 # Plot of dead cover
26 q <- mapview(data, zcol = "dead_cover", layer.name = "Cover(%)", cex =
27   ↪ 10, alpha = 0.9, alpha.regions = 0.9, na.rm = TRUE)
28
29 mapshot(q, file = "Data_files/Fig_6d.png")
30
31 #q
32
33 if (knitr:::is_latex_output()) {
34   # This part runs only for PDF output
35 } else if (knitr:::is_html_output()) {
36   # This part runs only for HTML output
37   p
38
39   q
40 } else {
41   # This part runs if neither PDF nor HTML is the target format
42   p
43
44   q
45 }
```



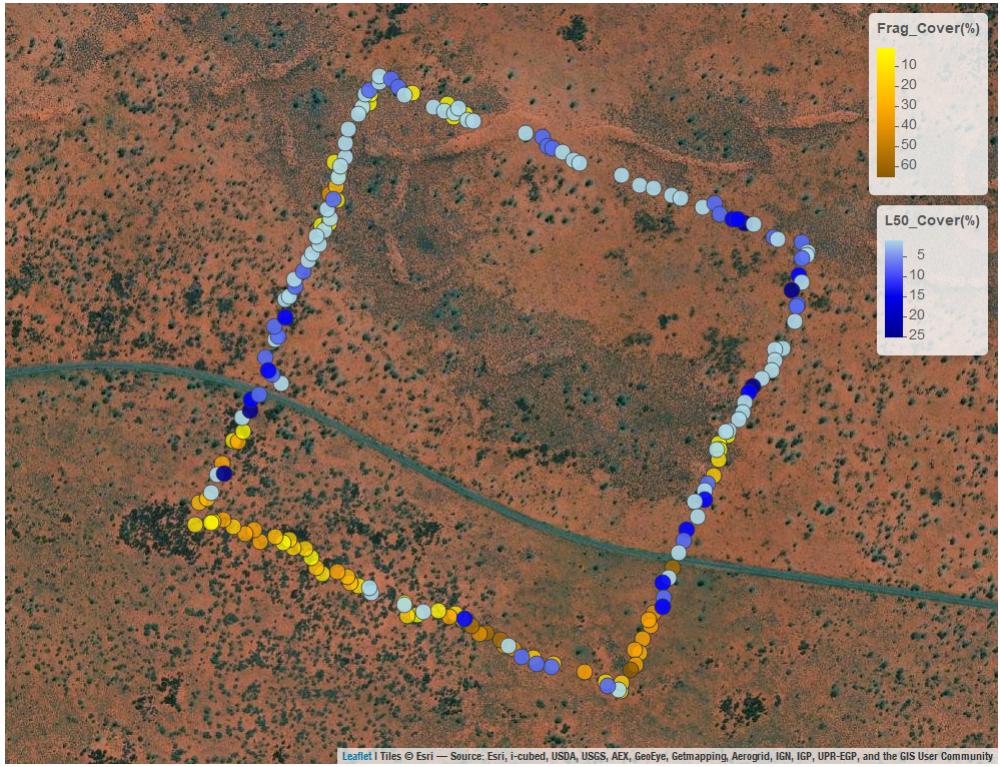
### 3.3 Map the cover of different spinifex size classes within each image frame (image for Figure 6e)

### 3.4 Re-name spinifex species/size data for size class analysis

```
1 df_spinifex$species[df_spinifex$species == 'No Triodia Present'] <-
2   ↪ 'Absent'
3
4 df_spinifex$species[df_spinifex$species == 'Triodia pungens >100cm
5   ↪ old fragmentary'] <- 'frag'
6
7 df_spinifex$species[df_spinifex$species == 'Triodia sp. >100cm old
8   ↪ fragmentary'] <- 'frag'
9
10 df_spinifex$species[df_spinifex$species == 'Triodia pungens < 50 cm']
11   ↪ <- '<50'
12
13 df_spinifex$species[df_spinifex$species == 'Triodia sp. 50-<100 cm']
14   ↪ <- '50_100'
15
16 df_spinifex$species[df_spinifex$species == 'Triodia sp. 100-<200 cm']
17   ↪ <- '100_200'
18
19 df_spinifex$species[df_spinifex$species == 'Triodia pungens 200-<400
20   ↪ cm'] <- '200_400'
21
22 df_spinifex$species[df_spinifex$species == 'Triodia sp. < 50 cm'] <-
23   ↪ '<50'
```

### 3.5 Calculate and map the cover and distribution of < 50cm and fragmentary (old > 1m) size classes (image for Figure 6e)

```
1 # Subset spinifex data for only < 50 cm size class
2 Less50cm <- subset(df_spinifex, df_spinifex$species == "<50")
3
4 # Subset spinifex data for only old fragmentary size class
5 frag <- subset(df_spinifex, df_spinifex$species == "frag")
6
7 # Jitter the data so that they can be seen more easily in the map
8 jittered_frag <- st_jitter(frag, amount = 0.0001)
9 jittered_Less50cm <- st_jitter(Less50cm, amount = 0.0001)
10
11 # Map the variables together - Figure 6e
12 p <- mapview(jittered_frag, zcol = "percentage_cover", layer.name =
  ↪ "Frag_Cover(%)", cex = 8, alpha = 0.9, alpha.regions = 0.75,
  ↪ col.regions = colorRampPalette(c("yellow", "orange", 'orange4')), 
  ↪ na.rm = TRUE) + mapview(jittered_Less50cm, zcol =
  ↪ "percentage_cover", cex = 8, alpha = 0.75, alpha.regions = 0.9,
  ↪ col.regions = colorRampPalette(c("lightblue", "blue",
  ↪ 'darkblue')),layer.name = "L50_Cover(%)", na.rm = TRUE)
13
14 mapshot(p, file = "Data_files/Fig_6e.png")
15
16 if (knitr:::is_latex_output()) {
17   # This part runs only for PDF output
18 } else if (knitr:::is_html_output()) {
19   # This part runs only for HTML output
20   p
21 } else {
22   # This part runs if neither PDF nor HTML is the target format
23   p
24 }
```



### 3.6 Create boxplot of size class vs pre-drought and post-drought percentage cover with kruskal-wallis test (image for Figure 6f)

```

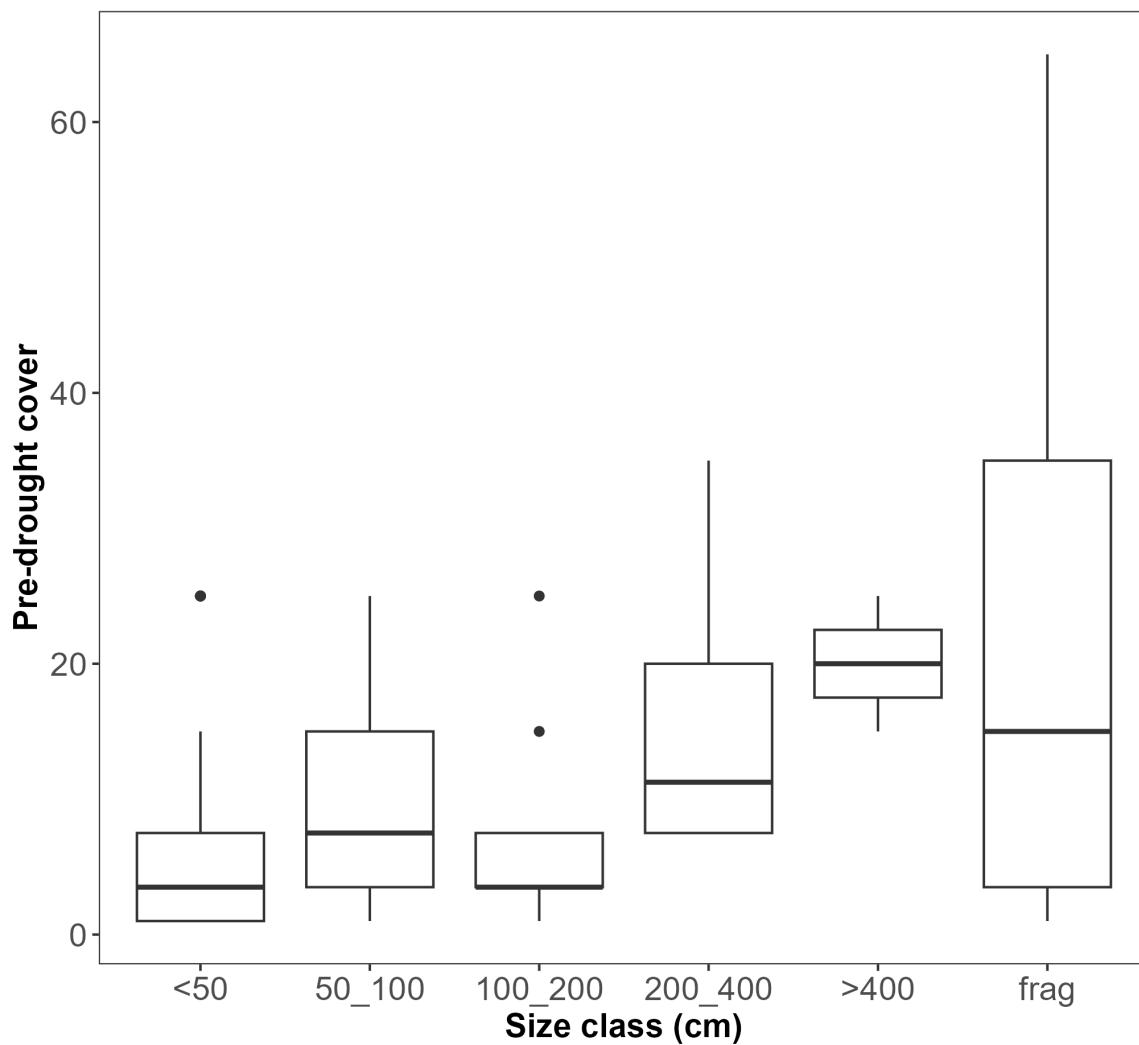
1 df_spinifex_full_data <- df_spinifex
2
3 df_spinifex <- df_spinifex %>%
4   filter(percentage_cover > 0)
5
6 df_spinifex <- df_spinifex[df_spinifex$species != 'Absent', ]
7 df_spinifex <- df_spinifex[df_spinifex$species != 'NA', ]
8
9 df_spinifex$speciesReorder <- factor(df_spinifex$species, c("<50",
10   "50_100", "100_200", "200_400", ">400", "frag"))
11
12 p <- ggplot(df_spinifex, aes(x = speciesReorder, y =
13   percentage_cover)) +
  geom_boxplot() + theme_bw() + theme(panel.grid.major =
    element_blank(),
    panel.grid.minor = element_blank()) +

```

```

14     theme(axis.text.x = element_text(size = 14),
15            axis.text.y = element_text(size = 14))
16
17 p <- p + ggtitle("") +
18   xlab("Size class (cm)") + ylab("Pre-drought cover") +
19   theme(axis.text=element_text(size=12),
20         axis.title=element_text(size=14,face="bold"))
21 ggsave("Data_files/Fig_6f.png", width = 6.81, height = 6.45, dpi =
22   ↪ 300)
23 # Test for nonparametric relationship between percentage cover and
24   ↪ size class
25 kruskal.test(percentage_cover ~ speciesReorder, data = df_spinifex)

```



## 4 Analysis of desert oak crown health data

### 4.1 Import the lookup csv containing size classes and match to size class data in the df\_oak object

```

1 # The original df_oak data have been imported in code above
2
3 # Import the match table
4 df_oak_class <- read.csv("Data_files/Oak_sizes_reclass.csv")

```

```

5   # We must create new variables to allow the match
6   df_oak$Taxon <- NA
7   df_oak$Size_class <- NA
8   df_oak$Burnt <- NA
9   df_oak$Count <- NA
10
11
12 # We must remove whitespace and other incompatibilities
13
14 df_oak$species <- str_replace_all(df_oak$species, "<.*?>", "")
15 df_oak_class$species <- str_replace_all(df_oak_class$species, "<.*?>",
16   " ")
17
18 df_oak$species <- str_to_lower(df_oak$species)
19 df_oak$species <- str_replace_all(df_oak$species, "\\s+", "")
20
21 df_oak_class$species <- str_to_lower(df_oak_class$species)
22 df_oak_class$species <- str_replace_all(df_oak_class$species, "\\s+",
23   " ")
24
25 # Now merge Taxon, Burnt, Size_class, Species
26
27 df_oak$Taxon <- df_oak_class$Taxon[match(df_oak$species,
28   df_oak_class$species)]
29
30 df_oak$Burnt <- df_oak_class$Burnt[match(df_oak$species,
31   df_oak_class$species)]
32
33 df_oak$Count <- df_oak_class$Count[match(df_oak$species,
34   df_oak_class$species)]
35
36 df_oak$Size_class <- df_oak_class$Size_class[match(df_oak$species,
37   df_oak_class$species)]

```

## 4.2 Calculate crown health score and compare crown health across size classes (create image for Figure 7f)

```

1 # Ensure that the variables are numeric
2 df_oak$dd1 <- as.numeric(df_oak$dd1)

```

```

3 df_oak$dd2 <- as.numeric(df_oak$dd2)
4 df_oak$dd3 <- as.numeric(df_oak$dd3)

5
6 # Calculate the crown score (HC = AC × CE × CD in paper)
7 df_oak$Crown_score <- df_oak$dd1/100 * df_oak$dd2/100 * df_oak$dd3/100

8
9 # Reorder the classes in order of size
10 df_oak$Size_class_Reorder <- factor(df_oak$Size_class, c("Seedling",
11   "Juvenile", "Early_Pole", "Late_Pole", "Early_Mature", "Mature"))

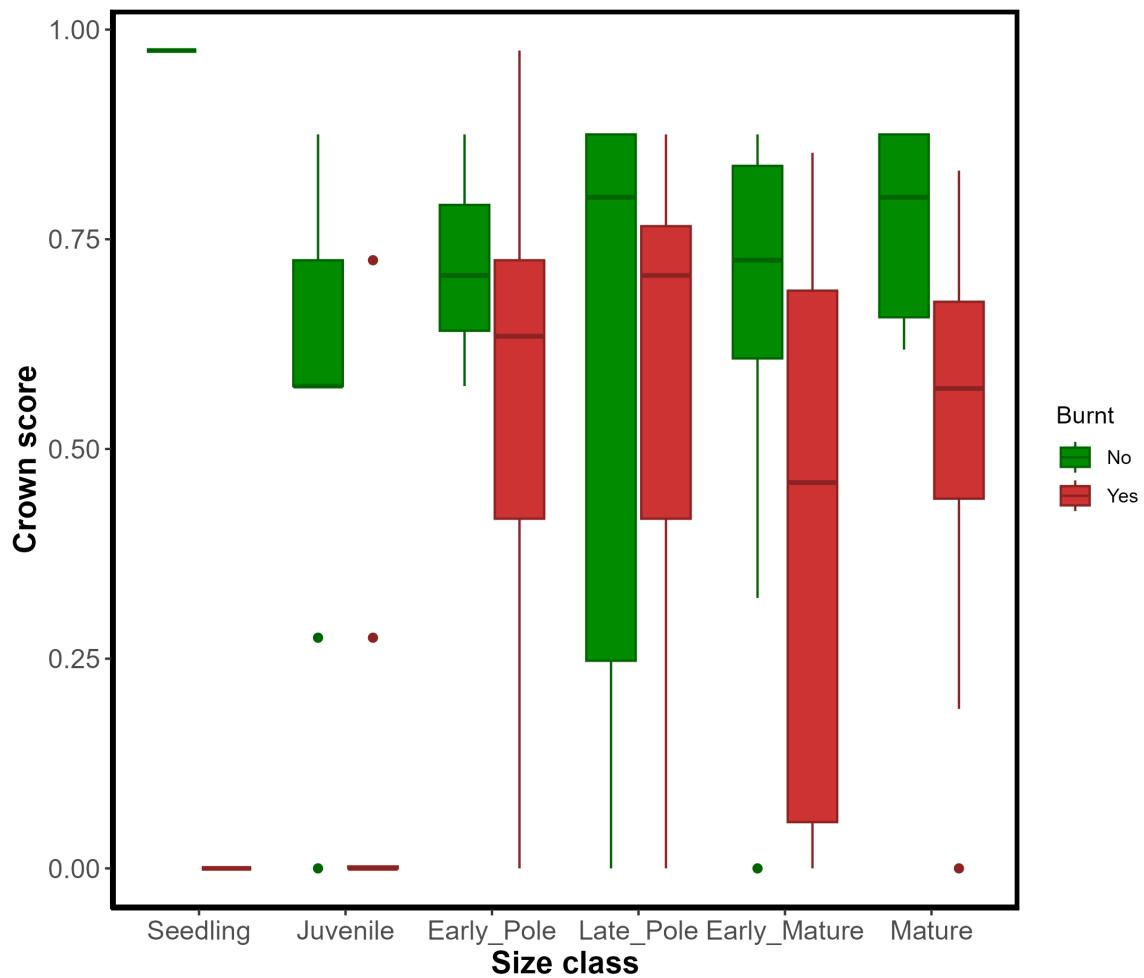
12 # Boxplot of size class and burnt vs crown score
13 p <- ggplot(df_oak, aes(x = Size_class_Reorder, y = Crown_score, z =
14   Burnt, fill = Burnt, color = Burnt)) +
15   geom_boxplot() + theme_bw() + theme(panel.grid.major =
16     element_blank(),
17     panel.grid.minor = element_blank()) +
18   scale_fill_manual(values=c("green4","brown3")) +
19   scale_color_manual(values=c("darkgreen","brown4")) +
20   theme_classic() +
21   theme(panel.border = element_rect(color = "black", fill = NA,
22     size = 2))

23 p <- p + ggtitle("") +
24   xlab("Size class") + ylab("Crown score") +
25   theme(axis.text=element_text(size=12),
26     axis.title=element_text(size=14,face="bold"))

27 ggsave("Data_files/Fig_7f.png", width = 7.25, height = 6.45, dpi =
28   300)

29 # Test for nonparametric relationship between crown score and burnt +
30   size class
31 scheirerRayHare(Crown_score ~ Burnt + Size_class_Reorder, data =
32   df_oak)

```



### 4.3 Map burnt and unburnt tree crowns (image for Figure 7e)

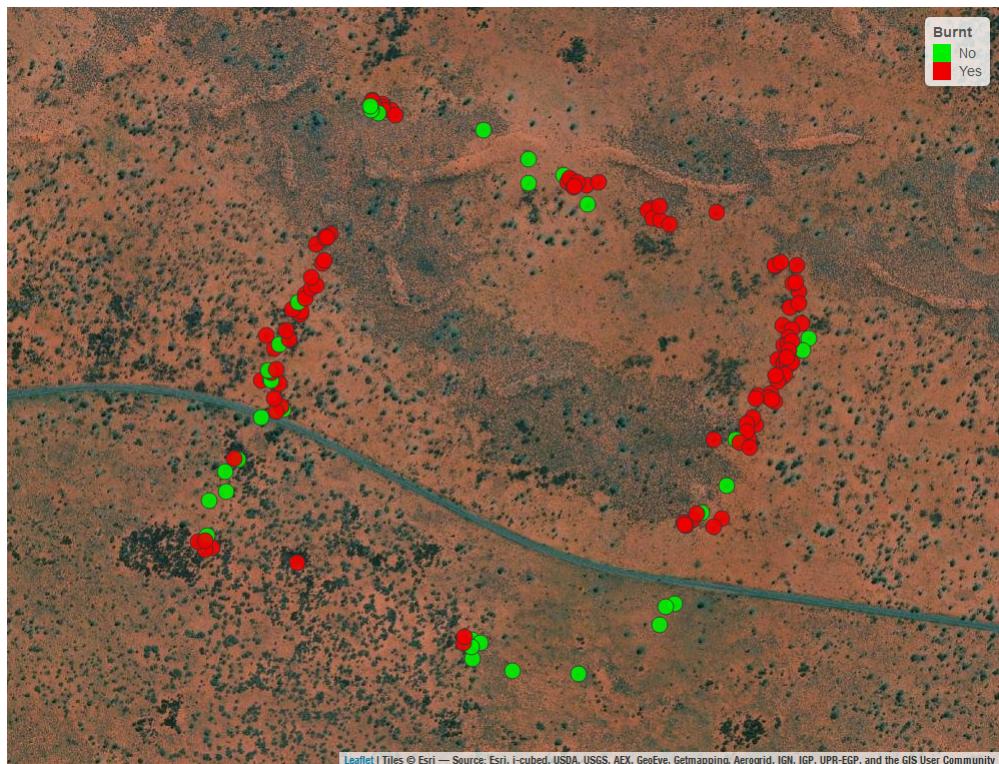
```

1 # Map burnt and unburnt areas
2 p <- mapview(df_oak, zcol = "Burnt", cex = 8, alpha = 0.9,
   ↪ alpha.regions = 0.9, layer.name = "Burnt", col.regions =
   ↪ c("green2", "red2"), na.rm = TRUE)
3
4 mapshot(p, file = "Data_files/Fig_7e.png")
5
6 if (knitr:::is_latex_output()) {
7   # This part runs only for PDF output
8 } else if (knitr:::is_html_output()) {

```

```

9  # This part runs only for HTML output
10 p
11 } else {
12   # This part runs if neither PDF nor HTML is the target format
13   p
14 }
```



#### 4.4 Map the distribution of tree size classes (image for Figure 7c)

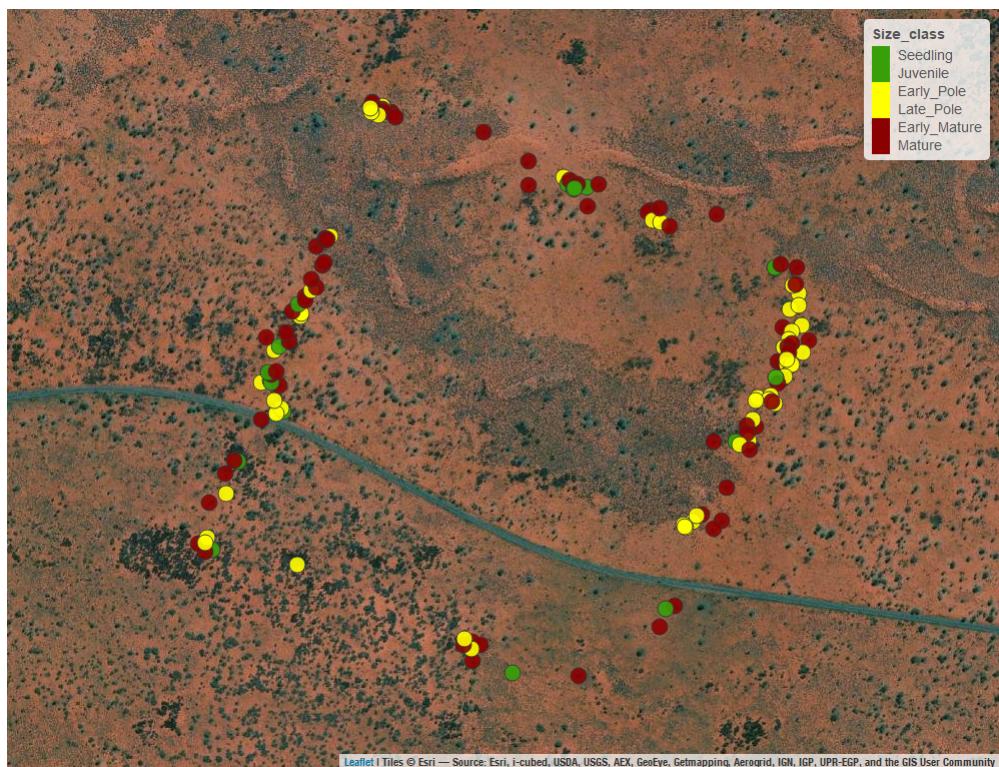
```

1 # Map size classes
2 p <- mapview(df_oak, zcol = "Size_class_Reorder", cex = 8, alpha =
+   0.9, alpha.regions = 0.9, layer.name = "Size_class", col.regions =
+   c("#389f0a", "#389f0a","yellow", "yellow", "red4", "red4"),na.rm =
+   TRUE)
3
4 mapshot(p, file = "Data_files/Fig_7c.png")
5
6 if (knitr:::is_latex_output()) {
```

```

7 # This part runs only for PDF output
8 } else if (knitr:::is_html_output()) {
9 # This part runs only for HTML output
10 p
11 } else {
12 # This part runs if neither PDF nor HTML is the target format
13 p
14 }

```



## 4.5 Plot the crown health of mature and early mature size classes (Figure 7d)

```

1 # Select only mature and early mature size classes
2
3 Mature <- subset(df_oak, df_oak$Size_class == "Mature" |
4   ↪ df_oak$Size_class == "Early_mature")

```

```

5 mapviewOptions(basemaps = c("Esri.WorldImagery"),
6                 raster.palette = grey.colors,
7                 vector.palette = colorRampPalette(c("red", "yellow",
8                                         "green2")),
9                 na.color = "gray",
10                layers.control.pos = "topright")
11
12 p <- mapview(df_oak, zcol = "Crown_score", layer.name = "Crown_score",
13               cex = 8, alpha = 0.9, alpha.regions = 0.9, na.rm = TRUE)
14
15 if (knitr:::is_latex_output()) {
16   # This part runs only for PDF output
17 } else if (knitr:::is_html_output()) {
18   # This part runs only for HTML output
19   p
20 } else {
21   # This part runs if neither PDF nor HTML is the target format
22   p
23 }
```

