

Appendix S1: case study of grasslands in Redwood County, MN code

This document lays out the R code for all the simulations, analysis, and figures for the “Case study of grasslands in Redwood, MN” which accompanies “Apps help bridge restoration science and practice” by: Sperry, Shaw, and Sullivan (submitted to Restoration Ecology 2019). Code was created by Katie Sperry; please contact katieprosperry@gmail.com with any questions.

```
rm(list=ls())
knitr::opts_chunk$set(tidy=TRUE)
library(raster)
library(rgdal)
library(spatial)
library(sp)
library(sf)
library(dplyr)
library(purrr)
library(tidyr)
library(glue)
library(reshape2)
library(igraph)
library(dismo)
library(leaflet)
library(ggplot2)
library(wesanderson)
library(knitr)
```

Data

```
# county polygon shapefile data
counties <- rgdal::readOGR(dsn = ".", layer = "mn_county_boundaries_500")

## OGR data source with driver: ESRI Shapefile
## Source: "/Users/katherinesperry/Dropbox/RENEW/Analysis/data", layer: "mn_county_boundaries_500"
## with 117 features
## It has 12 fields

counties_latlon <- sp::spTransform(counties, sp::CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"))
utm15nCRS <- crs(counties_latlon)
redwood <- subset(counties_latlon, counties_latlon$CTY_NAME == "Redwood")

# Redwood's grassland centroid coordinate data
centroids_csv <- read.csv("~/Desktop/Connectivity Project/redwood_5acres.csv")
centroids_sp <- SpatialPointsDataFrame(coords = centroids_csv[, 3:4], data = centroids_csv,
  proj4string = utm15nCRS)
centroids_sp <- sp::spTransform(centroids_sp, sp::CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"))
```

Define connectivity metrics function for later analysis

```
# takes in a distance matrix for all pairs of grasslands and sorts it to
# include only pairs with distances of 1,000m or less. Then use igraph
# package to create a network and extract connectivity metrics.
metricsUtility <- function(distanceMatrix) {
  df <- data.frame(as.matrix(distanceMatrix))
  names(df) <- as.factor(as.character(seq(1, nrow(df))))
  df$id <- as.factor(as.character(seq(1, nrow(df))))
  df_melt <- reshape2::melt(df, id = "id")
  threshold <- 1000
  units(threshold) <- "m"
  g <- igraph::graph.edgelist(as.matrix(subset(df_melt, value <= threshold)[,
    1:2]), directed = FALSE)
  g <- igraph::simplify(g)
  no_grasslands <- nrow(df)
  no_comp <- igraph::count_components(g, mode = c("weak", "strong"))
  comp <- igraph::components(g, mode = c("weak", "strong"))
  maxcomp <- max(comp$size)
  avgsz <- base::mean(comp$size)
  isolated <- length(comp$size[comp$size == 1])
  transitivity <- mean(igraph::transitivity(g, type = "local", na.rm = TRUE))
  degree <- mean(igraph::degree(g, v = V(g), loop = FALSE))
  between <- mean(igraph::betweenness(g, v = V(g), normalized = TRUE))
  close <- mean(igraph::closeness(g, v = V(g), normalized = TRUE))
  network_results <- as.vector(c(no_grasslands, no_comp, round(avgsz, digits = 5),
    maxcomp, isolated, round(transitivity, digits = 5), round(degree, digits = 5),
    round(between, digits = 5), round(close, digits = 5)))
  return(network_results)
}
```

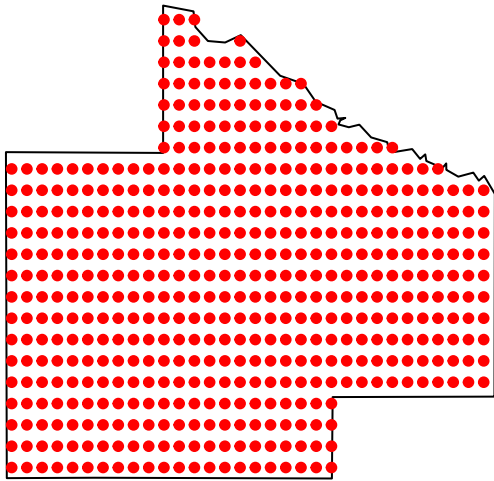
Analysis

1. Find the baseline (current) connectivity for Redwood County.

```
# convert grassland centroid coordinates to an sf (spatial) object
grasses_geo <- sf::st_as_sf(centroids_sp, coords = c("Longitude", "Latitude"),
  crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
# make a distance matrix between all pairs of centroids
dist_matrix <- sf::st_distance(grasses_geo, grasses_geo, tolerance = 2000)
dist_matrix[upper.tri(dist_matrix)] <- NA
# calculate metrics with the metrics function defined above
metricsOriginal <- metricsUtility(dist_matrix)
metrics <- matrix(metricsOriginal, ncol = 9, byrow = TRUE)
colnames(metrics) <- c("Original.no.grasslands", "Original.no.comps", "Original.avg.size",
  "Original.max.size", "Original.no.isolated", "Original.transitivity", "Original.degree",
  "Original.betweenness", "Original.closeness")
```

Objective A: determine optimal areas to put a new prairie to maximize connectivity

```
# Create regularly spaced points in redwood county
random_points <- sp::spsample(redwood, n = 500, "regular")
# transform points to match redwood grassland's data type
random_trans <- sp::spTransform(random_points, sp::CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs"))
plot(redwood)
plot(random_trans, add = TRUE, col = "red", pch = 20)
```



Assess how each generated point would change connectivity.

```
# create an empty matrix for storing results
resultsA <- matrix(ncol = 12, nrow = 0)
# iterate through the 500 newly made points, adding each to the existing
# network and calculating the new connectivity metrics
for (p in 1:length(random_trans)) {
  newPoint <- random_trans[p]
  newGrasses <- rbind.SpatialPoints(centroids_sp, newPoint)
  grasses_geo <- sf::st_as_sf(newGrasses, coords = c("Latitude", "Longitude"),
    crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
  dist_matrix <- sf::st_distance(grasses_geo, grasses_geo)
  dist_matrix[upper.tri(dist_matrix)] <- NA
  resultsPoint <- metricsUtility(dist_matrix)
  lng <- newPoint@coords[, "x1"]
  lat <- newPoint@coords[, "x2"]
  newrow <- c(p, lat, lng, resultsPoint[1], resultsPoint[2], resultsPoint[3],
    resultsPoint[4], resultsPoint[5], resultsPoint[6], resultsPoint[7],
    resultsPoint[8], resultsPoint[9])
  resultsA <- rbind(resultsA, newrow)
}
```

Explore results for objective A

```
resultsA <- as.data.frame(resultsA)
names(resultsA) <- c("ID", "Latitude", "Longitude", "No.grasslands", "No.comps",
  "Avg.comp.size", "Max.comp.size", "No.isolated", "Transitivity", "Degree",
  "Betweenness", "Closeness")
# head(resultsA)
summary(resultsA)
# add baseline metrics and calculate the change in metrics for each network
resultsA <- cbind(resultsA, metrics)
resultsA$Relative.no.comps <- resultsA$No.comps - resultsA$Original.no.comps
resultsA$Relative.avg.size <- resultsA$Avg.comp.size - resultsA$Original.avg.size
resultsA$Relative.max.size <- resultsA$Max.comp.size - resultsA$Original.max.size
resultsA$Relative.no.isolated <- resultsA$No.isolated - resultsA$Original.no.isolated
resultsA$Relative.transitivity <- resultsA$Transitivity - resultsA$Original.transitivity
resultsA$Relative.degree <- resultsA$Degree - resultsA$Original.degree
resultsA$Relative.betweenness <- resultsA$Betweenness - resultsA$Original.betweenness
resultsA$Relative.closeness <- resultsA$Closeness - resultsA$Original.closeness
# write.csv(resultsA, file = 'resultsA')
```

4. Objective B: determine where losing existing grasslands would most decrease connectivity

Assess how losing each existing point would change connectivity.

```
# create an empty matrix for storing results
resultsB <- matrix(ncol = 12, nrow = 0)
# iterate through all existing points, removing each from the existing
# network one by one and calculating the new connectivity metrics
for (r in 1:length(centroids_sp)) {
  grasses_sub1 <- centroids_sp[-r, ]
  grasses_geoB <- sf::st_as_sf(grasses_sub1, coords = c("Latitude", "Longitude"),
    crs = "+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
  dist_matrixB <- sf::st_distance(grasses_geoB, grasses_geoB)
  upper.tri(dist_matrixB)
  dist_matrixB[upper.tri(dist_matrixB)] <- NA
  resultsPointB <- metricsUtility(dist_matrixB)
  lng <- centroids_sp[r, ]@coords[, "Longitude"]
  lat <- centroids_sp[r, ]@coords[, "Latitude"]
  newrowB <- c(r, lat, lng, resultsPointB[1], resultsPointB[2], resultsPointB[3],
    resultsPointB[4], resultsPointB[5], resultsPointB[6], resultsPointB[7],
    resultsPointB[8], resultsPointB[9])
  resultsB <- rbind(resultsB, newrowB)
}
```

Explore results for Objective B

```
resultsB <- as.data.frame(resultsB)
names(resultsB) <- c("ID", "Latitude", "Longitude", "No.grasslands", "No.comps",
  "Avg.comp.size", "Max.comp.size", "No.isolated", "Transitivity", "MaxDegree",
  "MaxBetweenness", "MaxCloseness")
# Add current metrics and calculate the change in metrics for each network
resultsB <- cbind(resultsB, metrics)
```

```

resultsB$Relative.no.comps <- resultsB$No.comps - resultsB$Original.no.comps
resultsB$Relative.avg.size <- resultsB$Avg.comp.size - resultsB$Original.avg.size
resultsB$Relative.max.size <- resultsB$Max.comp.size - resultsB$Original.max.size
resultsB$Relative.no.isolated <- resultsB$No.isolated - resultsB$Original.no.isolated
resultsB$Relative.transitivity <- resultsB$Transitivity - resultsB$Original.transitivity
resultsB$Relative.max.degree <- resultsB$MaxDegree - resultsB$Original.max.degree
resultsB$Relative.max.betweenness <- resultsB$MaxBetweenness - resultsB$Original.max.betweenness
resultsB$Relative.max.closeness <- resultsB$MaxCloseness - resultsB$Original.max.closeness
# write.csv(resultsB, 'resultsB')

```

Vizualising results

```

# Fortify redwood shape in order to use ggplot2 for visuals
myDFR <- fortify(redwood)
# make centroid coordinates into a dataframe for ggplot2 visuals
mRG <- data.frame(centroids_sp@coords)

```

Vizualising objective A: Find the top 5% of all metric's outcomes from addition simulations and plot together

```

# subset the top 5% of each metric - 0.05 or 0.95 is used depending on the
# optimal direction i.e. if it is better to increase or decrease the metric
top_nocomps <- resultsA[which(resultsA$Relative.no.comps < quantile(resultsA$Relative.no.comps,
  0.05)), ]
top_degree <- resultsA[which(resultsA$Relative.degree > quantile(resultsA$Relative.degree,
  0.95)), ]
top_transitivity <- resultsA[which(resultsA$Relative.transitivity > quantile(resultsA$Relative.transitivity,
  0.95)), ]
top_maxsize <- resultsA[which(resultsA$Relative.max.size > quantile(resultsA$Relative.max.size,
  0.95)), ]
top_maxbet <- resultsA[which(resultsA$Relative.maxbetweenness > quantile(resultsA$Relative.maxbetweenness,
  0.95)), ]
# summarize these the subsetted top 5% of each metric
sum_top_comps <- summary(top_nocomps$Relative.no.comps)
sum_top_degree <- summary(top_degree$Relative.degree)
sum_top_trans <- summary(top_transitivity$Relative.transitivity)
sum_top_maxsize <- summary(top_maxsize$Relative.max.size)
sum_top_maxbet <- summary(top_maxbet$Relative.maxbetweenness)
# calculate the cutoff i.e. the value that metrics must be greater or less
# than to be plotted
cut_nocompsA <- quantile(resultsA$Relative.no.comps, 0.05)
cut_degreeA <- quantile(resultsA$Relative.degree, 0.95)
cut_transitivityA <- quantile(resultsA$Relative.transitivity, 0.95)
cut_maxsizeA <- quantile(resultsA$Relative.max.size, 0.975)
cut_maxbetA <- quantile(resultsA$Relative.maxbetweenness, 0.975)
Cutoff <- c(cut_nocompsA, cut_degreeA, cut_transitivityA, cut_maxsizeA, cut_maxbetA)
# present summary statistics and cutoffs for the top 5% of all metrics
summarydfA <- as.data.frame(rbind(sum_top_comps, sum_top_degree, sum_top_trans,
  sum_top_maxsize, sum_top_maxbet))
summarydfA <- cbind(summarydfA, Cutoff)

```

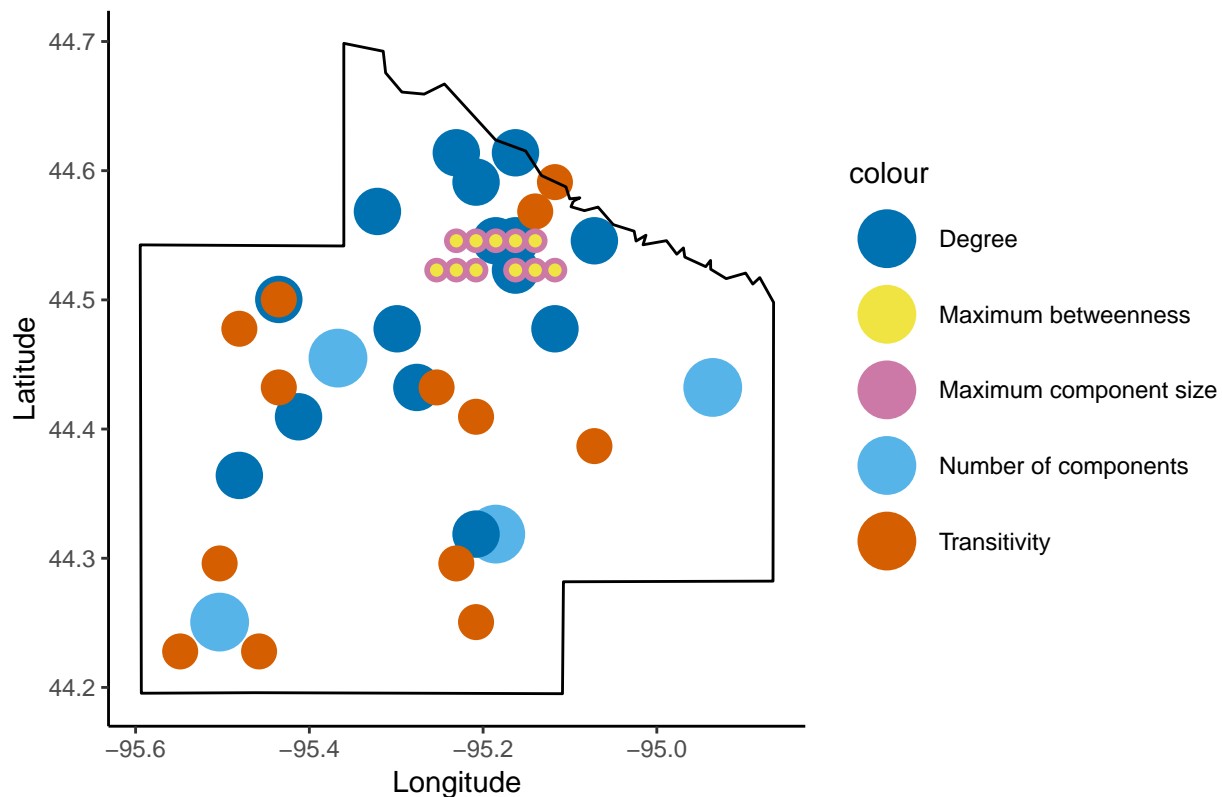
```
rownames(summarydfA) <- c("No. components", "Degree", "Transitivity", "Max component size",
  "Max betweenness")
summarydfA[, -1] <- round(summarydfA[, -1], 3)
kable(summarydfA)
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Cutoff
No. components	-2.00000	-2.000	-2.000	-2.000	-2.000	-2.000	-1.000
Degree	0.00667	0.007	0.007	0.007	0.008	0.010	0.005
Transitivity	0.00231	0.003	0.003	0.003	0.003	0.004	0.002
Max component size	1.00000	1.000	1.000	2.182	4.000	5.000	0.000
Max betweenness	0.00003	0.000	0.000	0.000	0.000	0.000	0.000

*# Make the composite figure of the locations that result in the top 5% of
changes - the locations that most maximize #connectivity when added to
the network.*

```
CompositeTop <- ggplot(data = myDFR, aes(x = long, y = lat, group = group)) +
  geom_polygon(fill = "white") + geom_point(data = top_nocomps, aes(x = Longitude,
    y = Latitude, color = "Number of components"), size = 10, shape = 16, inherit.aes = FALSE) +
  geom_point(data = top_degree, aes(x = Longitude, y = Latitude, color = "Degree"),
    size = 8, shape = 16, inherit.aes = FALSE) + geom_point(data = top_transitivity,
    aes(x = Longitude, y = Latitude, color = "Transitivity"), size = 6, shape = 16,
    inherit.aes = FALSE) + geom_point(data = top_maxsize, aes(x = Longitude,
    y = Latitude, color = "Maximum component size"), size = 4, shape = 16, inherit.aes = FALSE) +
  geom_point(data = top_maxbet, aes(x = Longitude, y = Latitude, color = "Maximum betweenness"),
    size = 2, shape = 16, inherit.aes = FALSE) + labs(title = "Restoration locations resulting in o
    y = "Latitude", x = "Longitude") + scale_color_manual(values = c(`Number of components` = "#56B4E9",
    Degree = "#0072B2", Transitivity = "#D55E00", `Maximum component size` = "#CC79A7",
    `Maximum betweenness` = "#F0E442")) + geom_path() + theme_classic()
CompositeTop
```

Restoration locations resulting in optimal outcomes for all metrics



Vizualising bjective B: Find the bottom 5% of all metric's outcomes from loss simulations and plot together

```
# subset the bottom 5% of each metric - 0.05 or 0.95 is used depending on
# the worse direction i.e. if it is worse to increase or decrease the metric
bot_nocomps <- resultsB[which(resultsB$Relative.no.comps > quantile(resultsB$Relative.no.comps,
  0.95)), ]
bot_degree <- resultsB[which(resultsB$Relative.max.degree < quantile(resultsB$Relative.max.degree,
  0.05)), ]
bot_transitivity <- resultsB[which(resultsB$Relative.transitivity < quantile(resultsB$Relative.transitivity,
  0.05)), ]
bot_maxsize <- resultsB[which(resultsB$Relative.max.size < quantile(resultsB$Relative.max.size,
  0.05)), ]
bot_maxbet <- resultsB[which(resultsB$Relative.max.betweenness < quantile(resultsB$Relative.max.betweenness,
  0.05)), ]
# summarize these the subsetting top 5% of each metric
sum_bot_comps <- summary(bot_nocomps$Relative.no.comps)
sum_bot_degree <- summary(bot_degree$Relative.max.degree)
sum_bot_degree[is.null(sum_bot_degree)] <- NA
sum_bot_trans <- summary(bot_transitivity$Relative.transitivity)
sum_bot_maxsize <- summary(bot_maxsize$Relative.max.size)
sum_bot_maxbet <- summary(bot_maxbet$Relative.max.betweenness)
# present summary statistics and cutoffs for the top 5% of all metrics
cut_nocompsB <- quantile(resultsB$Relative.no.comps, 0.95)
cut_degreeB <- NA
```

```

cut_transitivityB <- quantile(resultsB$Relative.transitivity, 0.05)
cut_maxsizeB <- quantile(resultsB$Relative.max.size, 0.05)
cut_maxbetB <- quantile(resultsB$Relative.max.betweenness, 0.05)
Cutoff <- c(cut_nocompsB, cut_degreeB, cut_transitivityB, cut_maxsizeB, cut_maxbetB)
# Summary statistics for the bottom 5% of all metrics
summarydfB <- as.data.frame(rbind(sum_bot_comps, sum_bot_degree, sum_bot_trans,
  sum_bot_maxsize, sum_bot_maxbet))
rownames(summarydfB) <- c("No. components", "Degree", "Transitivity", "Max component size",
  "Max betweenness")
summarydfB <- cbind(summarydfB, Cutoff)
summarydfB[, -1] <- round(summarydfB[, -1], 3)
kable(summarydfB)

```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Cutoff
No. components	2.00000	2.000	2.000	2.000	2.000	2.000	1.000
Degree	NA	NA	NA	NaN	NA	NA	NA
Transitivity	-0.00338	-0.003	-0.002	-0.002	-0.002	-0.002	-0.002
Max component size	-15.00000	-2.000	-1.000	-2.472	-1.000	-1.000	0.000
Max betweenness	-0.00042	0.000	0.000	0.000	0.000	0.000	0.000

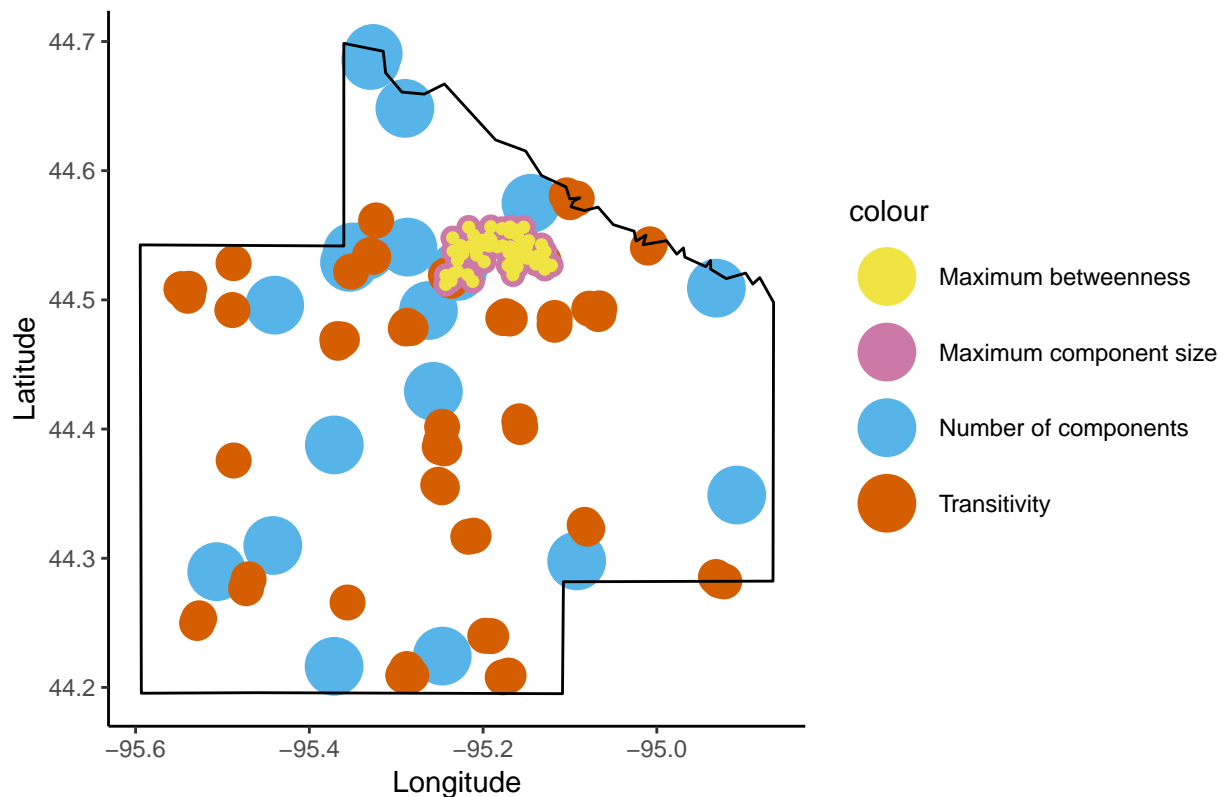
*# Make the composite figure of the locations that result in the bottom 5% of
changes - the locations that most decrease #connectivity when removed to
the network. - don't include degree because no removals decreased maximum
#degree.*

```

CompositeBot <- ggplot(data = myDFR, aes(x = long, y = lat, group = group)) +
  geom_polygon(fill = "white") + geom_point(data = bot_nocomps, aes(x = Longitude,
  y = Latitude, color = "Number of components"), size = 10, shape = 16, inherit.aes = FALSE) +
  geom_point(data = bot_transitivity, aes(x = Longitude, y = Latitude, color = "Transitivity"),
  size = 6, shape = 16, inherit.aes = FALSE) + geom_point(data = bot_maxsize,
  aes(x = Longitude, y = Latitude, color = "Maximum component size"), size = 4,
  shape = 16, inherit.aes = FALSE) + geom_point(data = bot_maxbet, aes(x = Longitude,
  y = Latitude, color = "Maximum betweenness"), size = 2, shape = 16, inherit.aes = FALSE) +
  labs(title = "Habitat loss locations resulting in the worst outcomes for all metrics",
  y = "Latitude", x = "Longitude") + scale_color_manual(values = c(`Number of components` = "#56B4E9",
  Degree = "#0072B2", Transitivity = "#D55E00", `Maximum component size` = "#CC79A7",
  `Maximum betweenness` = "#F0E442")) + geom_path() + theme_classic()
CompositeBot

```


Habitat loss locations resulting in the worst outcomes for all metrics



```
# Make a size-scaled legend to match both figures
df <- matrix(c(2, 4, 6, 8, 10, 2, 4, 6, 8, 10, "Number of components", "Degree",
  "Transitivity", "Maximum component size", "Maximum betweenness"), ncol = 3,
  nrow = 5) %>% `colnames<-`(c("X", "Y", "Metric"))

df <- as.data.frame(df)
df$Metric <- factor(df$Metric, levels = c("Number of components", "Degree",
  "Transitivity", "Maximum component size", "Maximum betweenness"))

legend <- ggplot(df, (aes(x = X, y = Y))) + geom_point(aes(size = Metric, color = Metric)) +
  scale_color_manual(values = c(`Number of components` = "#56B4E9", Degree = "#0072B2",
    Transitivity = "#D55E00", `Maximum component size` = "#CC79A7", `Maximum betweenness` = "#F0E444")) +
  scale_size_manual(values = c(`Number of components` = 10, Degree = 8, Transitivity = 6,
    `Maximum component size` = 4, `Maximum betweenness` = 2)) + theme_classic()

legend
```

