EnvironmentalStatisticsIRP

Getting Started

Always have to start with loading in a few libraries I'm sure I'll need

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   4.0.0     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(ggplot2)
```

Getting the Data

Need to read in 2 critical things: my actual data, as well as my shapefiles for PNF. This data is in a very raw state, so it will take quite a bit of modification to get it into my actual database.

First, I'll read in my raw data, clean it, and make it into a single database. This is a lot of data and will require a lot of cleaning to get into a single database.

```
# Reading in the CSVs. There were multiple files, due to the size of the database being difficult to do

RawBradshaw1 <- read_csv("RawData/RawPNFBradshawData.csv")
```

```
## Rows: 32907 Columns: 38
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): ProtocolType, ProtocolName, EventType, EventName, FormName, Date, ...
## dbl  (9): Slope, Aspect, Elevation, Transect, SampleNumber, Element, nValue,...
## lgl (12): SubElement, cParameter, cParameter2, cParameter3, cValue, cValueCo...
##
## 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.
```

```
RawBradshaw2 <- read_csv("RawData/RawPNFBradshawData2.csv")
```

```
## Rows: 23673 Columns: 38
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): ProtocolType, ProtocolName, EventType, EventName, FormName, Date, ...
## dbl  (9): Slope, Aspect, Elevation, Transect, SampleNumber, Element, nValue,...
## lgl (12): SubElement, cParameter, cParameter2, cParameter3, cValue, cValueCo...
##
## 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.
```

```
RawChinoValley1 <- read_csv("RawData/RawPNFChinoValleyData.csv")
```

```
## Rows: 43557 Columns: 38
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): ProtocolType, ProtocolName, EventType, EventName, FormName, Date, ...
## dbl  (9): Slope, Aspect, Elevation, Transect, SampleNumber, Element, nValue,...
## lgl (12): SubElement, cParameter, cParameter2, cParameter3, cValue, cValueCo...
##
## 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.
```

```
RawChinoValley2 <- read_csv("RawData/RawPNFChinoValleyData3.csv")
```

```
## Rows: 50598 Columns: 38
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): ProtocolType, ProtocolName, EventType, EventName, FormName, Date, ...
## dbl  (9): Slope, Aspect, Elevation, Transect, SampleNumber, Element, nValue,...
## lgl (12): SubElement, cParameter, cParameter2, cParameter3, cValue, cValueCo...
##
## 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.
```

```
RawVerde1 <- read_csv("RawData/RawPNFVerdeData.csv")
```

```
## Rows: 33078 Columns: 38
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): ProtocolType, ProtocolName, EventType, EventName, FormName, Date, ...
## dbl  (9): Slope, Aspect, Elevation, Transect, SampleNumber, Element, nValue,...
## lgl (12): SubElement, cParameter, cParameter2, cParameter3, cValue, cValueCo...
##
## 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.
```

```
RawVerde2 <- read_csv("RawData/RawPNFVerdeData2.csv")
```

```
## Rows: 29910 Columns: 38
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): ProtocolType, ProtocolName, EventType, EventName, FormName, Date, ...
## dbl  (9): Slope, Aspect, Elevation, Transect, SampleNumber, Element, nValue,...
## lgl (12): SubElement, cParameter, cParameter2, cParameter3, cValue, cValueCo...
##
## 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.
```

```
#From here, I'm going to treat to A) have only the columns I need (site name, location, species, and ra

CleanPNF <- function(df) {
  CleanData <- df %>%
    select(Date, Ancestry, SiteID, SpeciesSymbol, SpeciesName, CommonName, SampleNumber, nValue, DDLat,
    rename(Rank = nValue,
           Lat = DDLat,
           Lon = DDLong)
  return(CleanData)
}

Bradshaw1Refined <- CleanPNF(RawBradshaw1)
Bradshaw2Refined <- CleanPNF(RawBradshaw2)
ChinoValley1Refined <- CleanPNF(RawChinoValley1)
ChinoValley2Refined <- CleanPNF(RawChinoValley2)
Verde1Refined <- CleanPNF(RawVerde1)
Verde2Refined <- CleanPNF(RawVerde2)


#Combine the main group into one dataframe

PNFFirst <- Bradshaw1Refined %>%
  full_join(Bradshaw2Refined) %>%
  full_join(ChinoValley1Refined) %>%
  full_join(ChinoValley2Refined) %>%
  full_join(Verde1Refined) %>%
  full_join(Verde2Refined)
```

```
## Joining with `by = join_by(Date, Ancestry, SiteID, SpeciesSymbol, SpeciesName,
## CommonName, SampleNumber, Rank, Lat, Lon)`
## Joining with `by = join_by(Date, Ancestry, SiteID, SpeciesSymbol, SpeciesName,
## CommonName, SampleNumber, Rank, Lat, Lon)`
## Joining with `by = join_by(Date, Ancestry, SiteID, SpeciesSymbol, SpeciesName,
## CommonName, SampleNumber, Rank, Lat, Lon)`
## Joining with `by = join_by(Date, Ancestry, SiteID, SpeciesSymbol, SpeciesName,
## CommonName, SampleNumber, Rank, Lat, Lon)`
## Joining with `by = join_by(Date, Ancestry, SiteID, SpeciesSymbol, SpeciesName,
## CommonName, SampleNumber, Rank, Lat, Lon)`
```

Getting Composition

That's my first dataframe, so now I'm going to narrow this data down to the point where I can get the composition for each species on each site. To do this, I will count each rank each species got per site (How many times was BOCU ranked 1 on X site in Y event?). This will involve creating a new dataframe, as the per sample data will become irrelevant.

```r
#Create a count of how many times each Rank was assigned per species

PNFSecond <- PNFFirst %>%
  group_by(Ancestry, SiteID, Lat, Lon, Date, SpeciesSymbol, Rank) %>%
  summarize(RankCount = n())
```

```
## `summarise()` has grouped output by 'Ancestry', 'SiteID', 'Lat', 'Lon', 'Date',
## 'SpeciesSymbol'. You can override using the `.groups` argument.
```

```r
#Assign true values to the Ranks

PNFSecond <- PNFSecond %>%
  mutate(Rank = case_when(
    Rank == 1 ~ 7,
    Rank == 2 ~ 2,
    Rank == 3 ~ 1,
    TRUE ~ Rank
  ))

#Change Date to an actual date data

PNFSecond$Date <- as.POSIXct(PNFSecond$Date, format = "%m/%d/%Y")

#Only select most recent event per site

PNFSecond <- PNFSecond %>%
  group_by(Ancestry, SiteID) %>%
  slice_max(Date, n = 1) %>%
  ungroup()

PNFSecond
```

```
## # A tibble: 14,571 x 8
##    Ancestry SiteID   Lat   Lon Date                SpeciesSymbol  Rank RankCount
##    <chr>    <chr>  <dbl> <dbl> <dttm>              <chr>         <dbl>     <int>
##  1 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 2FA               7         1
##  2 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 2FA               2         1
##  3 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 2FA               1        21
##  4 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 AGAVE             7         1
##  5 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 AGAVE             2         1
##  6 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 AGAVE             1         2
##  7 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 ARPU5             2         1
##  8 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 ARTE3             7        13
##  9 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 ARTE3             2        13
## 10 USFS > ~ 03-09~  34.5 -112. 2024-11-14 00:00:00 ARTE3             1        10
## # i 14,561 more rows
```

```r
#Determine how many frames were read per event (Composition is determined by taking the total value of

PNFThird <- PNFSecond %>%
  group_by(Ancestry, SiteID) %>%
  summarize(SiteTotalFrames = sum(RankCount)/3) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'Ancestry'. You can override using the
## `.groups` argument.
```

```
PNFThird
```

```
## # A tibble: 351 x 3
##    Ancestry                                                SiteID SiteTotalFrames
##    <chr>                                                   <chr>            <dbl>
##  1 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             100
##  2 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             100
##  3 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             188
##  4 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             160
##  5 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             100
##  6 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~              14
##  7 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             200
##  8 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~              99
##  9 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             100
## 10 USFS > Region 03 > Prescott National Forest > Bradsha~ 03-09~             100
## # i 341 more rows
```

```
#Get the Rank*RankCount value

PNFPreComposition <- PNFSecond %>%
  group_by(Ancestry, SiteID, Lon, Lat, Date, SpeciesSymbol) %>%
  summarize(PreComp = Rank*RankCount)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## `summarise()` has grouped output by 'Ancestry', 'SiteID', 'Lon', 'Lat', 'Date',
## 'SpeciesSymbol'. You can override using the `.groups` argument.
```

```
#Connect PNFThird to PNFPreComposition so that PNFPreComposition has the SiteTotalFrames

PNFPreComposition <- full_join(PNFPreComposition, PNFThird, join_by(Ancestry, SiteID))

PNFPreComposition
```

```
## # A tibble: 14,571 x 8
## # Groups:   Ancestry, SiteID, Lon, Lat, Date, SpeciesSymbol [6,370]
##    Ancestry         SiteID   Lon   Lat Date                SpeciesSymbol PreComp
##    <chr>            <chr>  <dbl> <dbl> <dttm>              <chr>           <dbl>
##  1 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 2FA                 7
##  2 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 2FA                 2
##  3 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 2FA                21
##  4 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 AGAVE               7
##  5 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 AGAVE               2
##  6 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 AGAVE                2
##  7 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARPU5                2
##  8 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARTE3               91
##  9 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARTE3               26
## 10 USFS > Region 0~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARTE3               10
## # i 14,561 more rows
## # i 1 more variable: SiteTotalFrames <dbl>
```

```
#Get actual Composition

PNFComposition <- PNFPreComposition %>%
  group_by(Ancestry, SiteID, Lon, Lat, Date, SpeciesSymbol, SiteTotalFrames) %>%
  summarize(Composition = sum(PreComp))
```

```
## `summarise()` has grouped output by 'Ancestry', 'SiteID', 'Lon', 'Lat', 'Date',
## 'SpeciesSymbol'. You can override using the `.groups` argument.
```

```
PNFComposition$Composition <- 10*PNFComposition$Composition/PNFComposition$SiteTotalFrames

PNFComposition
```

```
## # A tibble: 6,370 x 8
## # Groups:   Ancestry, SiteID, Lon, Lat, Date, SpeciesSymbol [6,370]
##    Ancestry SiteID   Lon   Lat Date                SpeciesSymbol SiteTotalFrames
##    <chr>    <chr>  <dbl> <dbl> <dttm>              <chr>                   <dbl>
##  1 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 2FA                       100
##  2 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 AGAVE                     100
##  3 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARPU5                     100
##  4 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARTE3                     100
##  5 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 BOCU                      100
##  6 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 BOHI2                     100
##  7 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 CAREX                     100
##  8 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 CEMO2                     100
##  9 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ELEL5                     100
## 10 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ERIN                      100
## # i 6,360 more rows
## # i 1 more variable: Composition <dbl>
```

Here I'm going to split columns into allotment and pasture as well as only including the most recent reading at each site and in each pasture (I want to project composition across a whole pasture, if there are multiple compositions per pasture, that will complicate things)

```
#Create the new columns with the string split functions

PNFComposition <- PNFComposition %>%
  mutate(Allotment = str_split_fixed(Ancestry, " > ", 6) [, 5]) %>%
  mutate(Pasture = str_split_fixed(Ancestry, " > ", 6) [, 6])

#Filter out unnecessary sites and events, I want only the most recent reading per pasture, as well as t

PNFComposition <- PNFComposition %>%
  filter(!is.na(Lon))
```

```
#Select only the most recent event per pasture

PNFComposition <- PNFComposition %>%
  group_by(Pasture) %>%
  filter(Date == max(Date)) %>%
  ungroup()

PNFComposition
```

```
## # A tibble: 3,540 x 10
##    Ancestry SiteID   Lon   Lat Date                SpeciesSymbol SiteTotalFrames
##    <chr>    <chr>  <dbl> <dbl> <dttm>              <chr>                   <dbl>
##  1 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 2FA                       100
##  2 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 AGAVE                     100
##  3 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARPU5                     100
##  4 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ARTE3                     100
##  5 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 BOCU                      100
##  6 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 BOHI2                     100
##  7 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 CAREX                     100
##  8 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 CEMO2                     100
##  9 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ELEL5                     100
## 10 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 ERIN                      100
## # i 3,530 more rows
## # i 3 more variables: Composition <dbl>, Allotment <chr>, Pasture <chr>
```

Very Important: Choose Your Species

This will be one final modification to the dataframe, where we select what species we want to visualize across the map. I am going to sample blue grama, whose code is BOGR2.

```
#Create dataframe that only has the target species involved

SelectedSpComp <- PNFComposition %>%
  group_by(Pasture) %>%
  filter(SpeciesSymbol == "BOCU") %>% #Can swap out the species code with any desired species code for
  ungroup()

SelectedSpComp
```

```
## # A tibble: 155 x 10
##    Ancestry SiteID   Lon   Lat Date                SpeciesSymbol SiteTotalFrames
##    <chr>    <chr>  <dbl> <dbl> <dttm>              <chr>                   <dbl>
##  1 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 BOCU                      100
##  2 USFS > ~ 03-09~ -112.  34.4 2024-11-14 00:00:00 BOCU                      100
##  3 USFS > ~ 03-09~ -112.  34.5 2024-11-14 00:00:00 BOCU                      100
##  4 USFS > ~ 03-09~ -112.  34.5 2021-09-23 00:00:00 BOCU                      200
##  5 USFS > ~ 03-09~ -112.  34.3 2023-11-14 00:00:00 BOCU                       99
##  6 USFS > ~ 03-09~ -112.  34.3 2017-09-19 00:00:00 BOCU                      194
##  7 USFS > ~ 03-09~ -112.  34.3 2024-10-30 00:00:00 BOCU                      100
##  8 USFS > ~ 03-09~ -112.  34.4 2023-11-14 00:00:00 BOCU                      100
##  9 USFS > ~ 03-09~ -113.  34.4 2021-09-15 00:00:00 BOCU                      198
## 10 USFS > ~ 03-09~ -112.  34.4 2024-11-05 00:00:00 BOCU                       94
## # i 145 more rows
## # i 3 more variables: Composition <dbl>, Allotment <chr>, Pasture <chr>
```

Making the Map

Now I need to enter my shapefiles, which will give me the pasture boundaries for Prescott National Forest.

```
library(sf)
```

```
## Linking to GEOS 3.13.0, GDAL 3.10.1, PROJ 9.5.1; sf_use_s2() is TRUE
```

```
library(ggmap)
```

```
## i Google's Terms of Service: <https://mapsplatform.google.com>
##   Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service>
##   OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

```
PastureBoundaries <- st_read("Shapefiles/PNFPastureBoundaries.shp")
```

```
## Reading layer `PNFPastureBoundaries' from data source
##   `C:\Users\caleb\Documents\School\STAT574EEnvironmentalStatistics\EnviornmentalStatisticsIRP\Shapef
##   using driver `ESRI Shapefile'
## Simple feature collection with 608 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 312242.2 ymin: 3777051 xmax: 430480.1 ymax: 3892366
## Projected CRS: NAD83 / UTM zone 12N
```

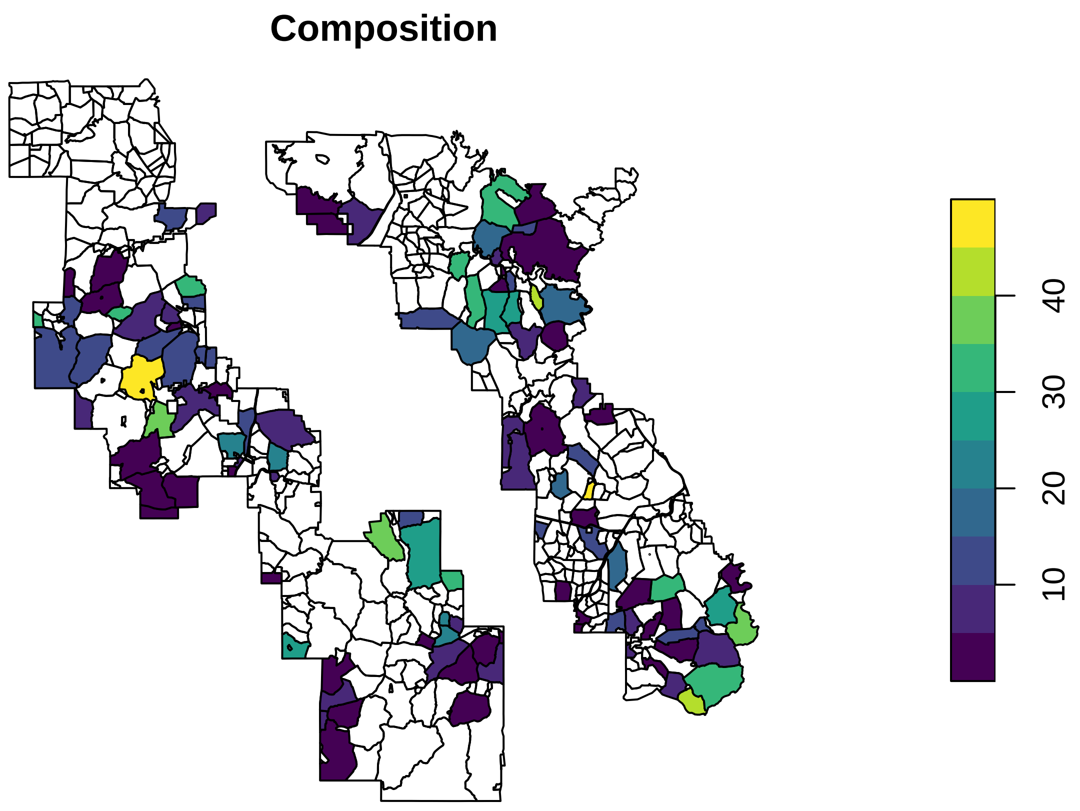
Finally, I will do some final tweaking to my code and eventually, spit out the map that is needed.

```
#First remove the spaces (not sure this is needed, but spaces in column names can sometimes create head

SelectedSpComp$Allotment <- gsub(" ", "", SelectedSpComp$Allotment)

#Remove the word allotment to match the PastureBoundaries naming convention

SelectedSpComp$Allotment <- gsub("Allotment", "", SelectedSpComp$Allotment)

#Do the same process for the pasture column

SelectedSpComp$Pasture <- gsub(" ", "", SelectedSpComp$Pasture)
SelectedSpComp$Pasture <- gsub("Pasture", "", SelectedSpComp$Pasture)

#Combine into single column called AP (allotment_pasture)

SelectedSpComp$AP <- paste0(SelectedSpComp$Allotment, "_", SelectedSpComp$Pasture)

#Do the same process for PastureBoundaries
```

```r
PastureBoundaries$UNIT_NAME <- gsub(" ", "", PastureBoundaries$UNIT_NAME)
PastureBoundaries$SUB_NAME <- gsub(" ", "", PastureBoundaries$SUB_NAME)
PastureBoundaries$AP <- paste0(PastureBoundaries$UNIT_NAME, "_", PastureBoundaries$SUB_NAME)

#Merge but keep all rows from PastureBoundaries (all.x = T). This created 2 extra rows, so there is lik

MergedDatas <- merge(PastureBoundaries, SelectedSpComp, all.x = T)

#Make spatial

MergedDatas_sf <- st_as_sf(MergedDatas, coords = c("lon", "lat"))
MergedDatas_sf <- st_transform(MergedDatas_sf, crs = 4326)

#Plot based on composition

library(viridis)
```

```
## Loading required package: viridisLite
```

```r
plot(MergedDatas_sf["Composition"],
     pal = viridis)
```



A composition map that should be able to be modified and show different compositions depending on the SelectedSpComp function, where we select the species we would like to have and put into the function