Week 6, Lecture 12
Advanced statistical methods, part II: Structural equation modeling, social network analysis, and geospatial analyses

Richard E.W. Berl

Spring 2019

Social network analysis

Let's use another network that will hopefully be a bit more workable: a data set of food web relationships in Ancestral Pueblo sites in southwestern Colorado in and around the Ute Mountain Reservation: https://www.sciencedirect.com/science/article/pii/S0305440317300377#appsec1

The source paper is:

Crabtree, S. A., Vaughn, L. J., & Crabtree, N. T. (2017). Reconstructing Ancestral Pueblo food webs in the southwestern United States. Journal of Archaeological Science, 81, 116-127. doi: 10.1016/j.jas.2017.03.005

In Appendix A: Supplementary data, click "Download spreadsheet (329KB)" to get the data and put it in your `/data` directory. If you can't access the data because of journal restrictions, you can get it here.

If we take a look at the Excel file, we can see there are separate sheets for the node list (1) and edge lists (2). In the edge list sheet, there are different lists for each archaeological site, and a combined list. For this exercise, we'll stick to the data from the Albert Porter Pueblo Food Web because it's the smallest. So, we specify that when we read in the sheet.

```
library(readxl)
library(igraph)
```

```
##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
foodN = as.data.frame(read_xlsx("./data/1-s2.0-S0305440317300377-mmc1.xlsx", sheet=1))
foodE.AP = as.data.frame(read_xlsx("./data/1-s2.0-S0305440317300377-mmc1.xlsx", sheet=2,
                                   range="G2:H1398"))
```

```
head(foodN)
```

```
##   ID              Species Name
## 1  1       A boreal chorus frog
```

```
## 2  2 A hammond's spadefoot toad
## 3  3    A northern leopard frog
## 4  4         A red-spotted toad
## 5  5      A rocky mountain toad
## 6  6    A utah tiger salamander
##                                                                     Source
## 1 http://www.reptilesofaz.org/Turtle-Amphibs-Subpages/h-p-triseriata.html
## 2 http://www.reptilesofaz.org/Turtle-Amphibs-Subpages/h-s-bombifrons.html
## 3    http://www.reptilesofaz.org/Turtle-Amphibs-Subpages/h-l-pipiens.html
## 4  http://www.reptilesofaz.org/Turtle-Amphibs-Subpages/h-a-punctatus.html
## 5          http://www.californiaherps.com/frogs/pages/b.w.woodhousii.html
## 6  http://www.reptilesofaz.org/Turtle-Amphibs-Subpages/h-a-mavortium.html
##
## 1
## 2 nocturnal beetles, crickets, grasshoppers, and other small arthropods. As mentioned, tadpoles are 
## 3
## 4
## 5
## 6                               Larvae and branchiates feed on a wide variety of invertebrates. Cannabi
```

```r
str(foodN)
```

```
## 'data.frame':    334 obs. of  4 variables:
##  $ ID          : num  1 2 3 4 5 6 319 7 8 9 ...
##  $ Species Name: chr  "A boreal chorus frog" "A hammond's spadefoot toad" "A northern leopard frog" 
##  $ Source      : chr  "http://www.reptilesofaz.org/Turtle-Amphibs-Subpages/h-p-triseriata.html" "http
##  $ Diet        : chr  "flies, springtails, spiders, snails, and ants" "nocturnal beetles, crickets, 
```

```r
head(foodE.AP)
```

```
##   Consumer Resource
## 1        3       79
## 2        3       82
## 3        3       84
## 4        4       79
## 5        4       82
## 6        4       84
```

```r
str(foodE.AP)
```

```
## 'data.frame':    1396 obs. of  2 variables:
##  $ Consumer: num  3 3 3 4 4 4 5 5 5 6 ...
##  $ Resource: num  79 82 84 79 82 84 79 82 84 79 ...
```

Species are noted by a letter to represent their class, then their common name. Let's split those apart so we can use the class as a variable later.

```r
foodN$Class = substr(foodN$Species, 1, 1)
foodN$Species = substr(foodN$Species, 3, nchar(foodN$Species))
```

Dince we subset the edge list to the Albert Porter site, let's make sure we only have the nodes that are present in that site.

```r
foodN.AP = foodN[foodN$ID %in% unique(unname(unlist(foodE.AP))),]
```

And just to make it more intuitive (in my mind, at least) let's make the direction of the edges flow from the resource to the consumer.

```r
foodE.AP = foodE.AP[,c(2,1)]
```

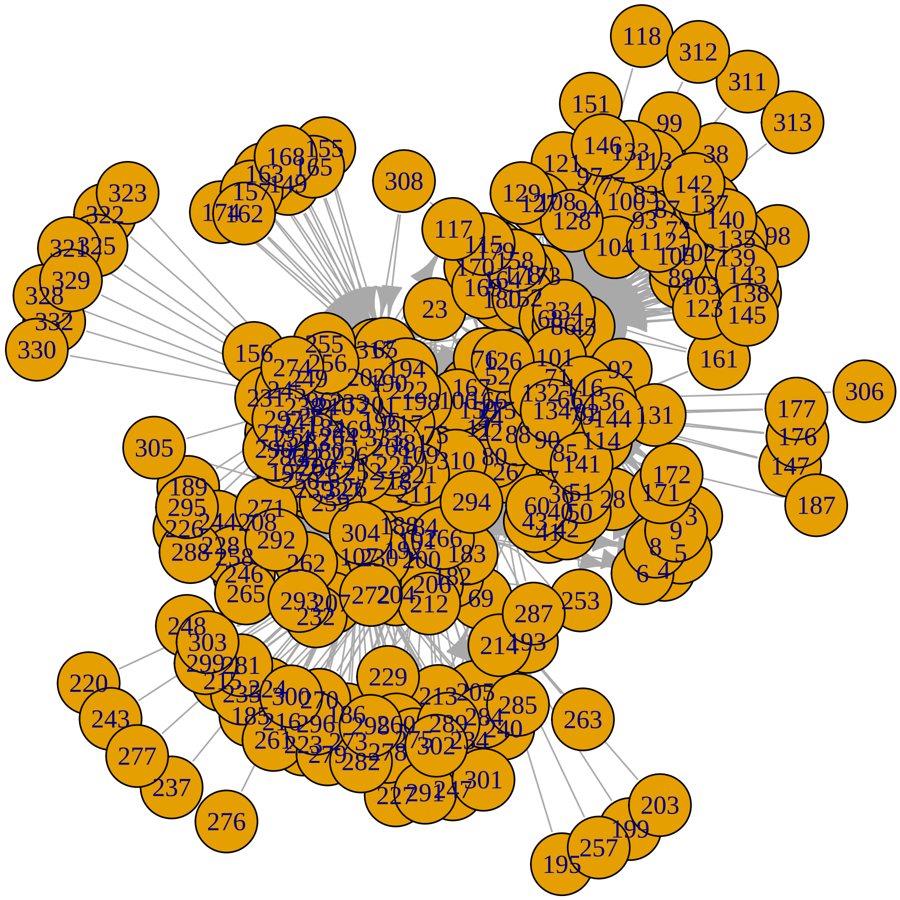
Now let's make it into a network, specifying the edge list and node list.

```
foodNet = graph_from_data_frame(foodE.AP, vertices=foodN.AP)
foodNet
```

```
## IGRAPH eeab8cd DN-- 263 1396 --
## + attr: name (v/c), Species Name (v/c), Source (v/c), Diet (v/c),
## | Class (v/c), Species (v/c)
## + edges from eeab8cd (vertex names):
##  [1] 79 ->3  82 ->3  84 ->3  79 ->4  82 ->4  84 ->4  79 ->5  82 ->5
##  [9] 84 ->5  79 ->6  82 ->6  84 ->6  75 ->7  88 ->7  109->7  66 ->7
## [17] 79 ->7  80 ->7  82 ->7  84 ->7  79 ->8  82 ->8  84 ->8  79 ->9
## [25] 82 ->9  84 ->9  75 ->11 109->11 66 ->11 76 ->11 69 ->11 79 ->11
## [33] 80 ->11 82 ->11 84 ->11 86 ->11 107->11 68 ->11 65 ->11 310->11
## [41] 333->11 317->11 334->11 75 ->22 109->22 66 ->22 76 ->22 69 ->22
## [49] 79 ->22 80 ->22 82 ->22 84 ->22 86 ->22 107->22 68 ->22 65 ->22
## + ... omitted several edges
```

And take a look by plotting it:

```
plot(foodNet)
```



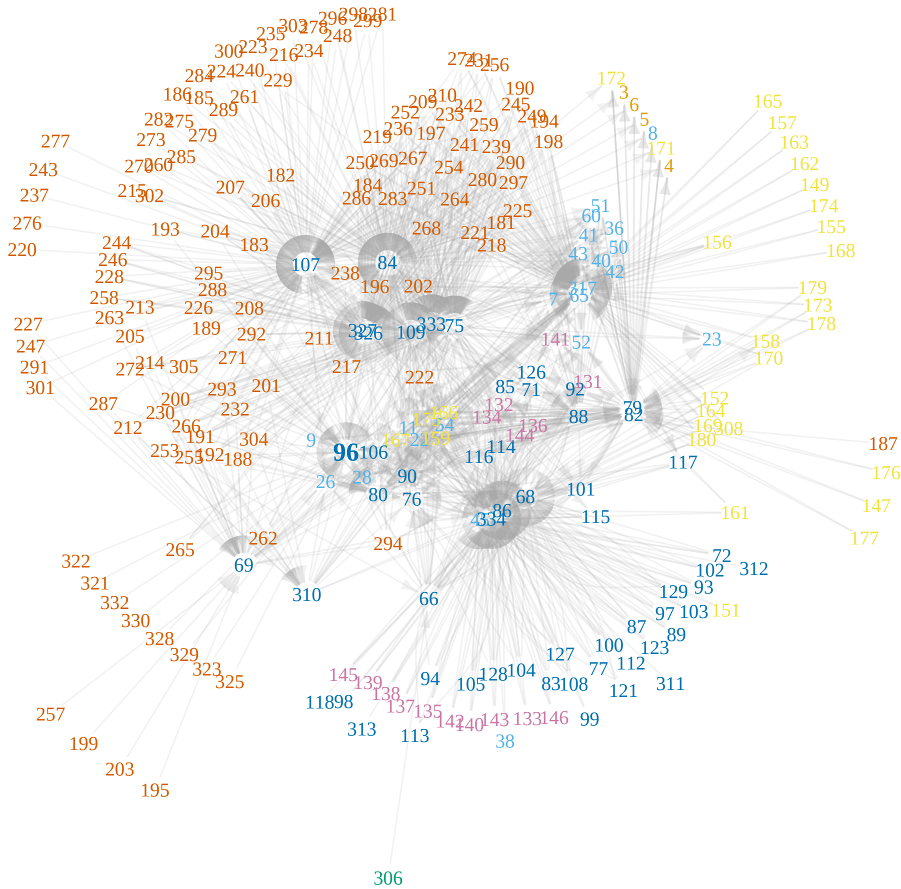
Kind of unintelligible given the default formatting, but we can tell there are some outer isolated nodes.

We can take a look at all the options available to tweak our plot:

```
?igraph::igraph.plotting
```

And make some changes to the node ("vertex") labels and edges, color labels by class, make edges partially transparent, and try a new layout to try to make things more visible:

```
plot(foodNet,
     vertex.size=6, vertex.shape="none",
     vertex.label.cex=ifelse(V(foodNet)$Species == "human", 1, 0.75),
     vertex.label.font=ifelse(V(foodNet)$Species == "human", 2, 1),
     vertex.label.color=as.numeric(as.factor(V(foodNet)$Class)),
     edge.arrow.size=0.5, edge.color=adjustcolor("darkgrey", alpha.f=0.15),
     layout=layout_with_kk)
```



Great! Still a little dense, but I think it's about as good as it can get without a lot more manual tweaking. I also enlarged and bolded the label for node 96, which we can see on the node list corresponds to humans. It looks like a lot of things are utilizing humans as a resource, but not so much the other way around.

But now we can dive into more of the details of our network.

Descriptive statistics

There isn't much hypothesis testing that I've encountered in social network analysis; most of the analysis is simply used to characterize the network as a whole, compare nodes with one another, and compare statistics with other networks.

A variety of descriptive measures are listed below, many of them taken from the excellent tutorial located here, where you can find some additional information on them: https://kateto.net/netscix2016.html

Edge density

Ratio of the number of edges to all possible edges

```
edge_density(foodNet, loops=F)
```

```
## [1] 0.02025948
```

Reciprocity

Proportion of mutual connections

```
reciprocity(foodNet, ignore.loops=T)
```

```
## [1] 0.02578797
```

```
dyad_census(foodNet)
```

```
## $mut
## [1] 18
##
## $asym
## [1] 1360
##
## $null
## [1] 33075
```

Transitivity

Probability of three vertices being connected

```
transitivity(foodNet, type="global")
```

```
## [1] 0.04461827
```

```
triad_census(foodNet)
```

```
##  [1] 2706972  224161   10957    6911   30896   15397     776     440
##  [9]     812       2      53       0       0      34       0       0
```

```
?triad_census
```

Diameter and distance

Longest chain of connections

```
diameter(foodNet, directed=T)
```

```
## [1] 6
```

```
get_diameter(foodNet, directed=T)
```

```
## + 7/263 vertices, named, from eeab8cd:
## [1] 322 96  334 175 317 86  26
```

Bottle gourd -> human -> dog -> tick -> sage grouse -> coyote -> snowshoe hare

```
mean_distance(foodNet, directed=T)
```

```
## [1] 2.602155
```

Shortest distance from humans to pinyon jay

```
shortest_paths(foodNet, from="96", to="43", output="both")
```

```
## $vpath
## $vpath[[1]]
## + 5/263 vertices, named, from eeab8cd:
```

```
## [1] 96  334 175 79  43
##
##
## $epath
## $epath[[1]]
## + 4/1396 edges from eeab8cd (vertex names):
## [1] 96 ->334 334->175 175->79  79 ->43
##
##
## $predecessors
## NULL
##
## $inbound_edges
## NULL
```

Human -> dog -> tick -> centipede -> pinyon jay

Centrality

Degree

Number of connections

```
degree(foodNet, v="96", mode="out")
```

```
## 96
##  1
```

```
degree(foodNet, v="96", mode="in")
```

```
## 96
## 49
```

```
ego(foodNet, nodes="96", mode="out")
```

```
## [[1]]
## + 2/263 vertices, named, from eeab8cd:
## [1] 96  334
```

Domestic dog

```
ego(foodNet, nodes="96", mode="in")
```

```
## [[1]]
## + 50/263 vertices, named, from eeab8cd:
##  [1] 96  45  317 65  66  68  69  333 75  76  79  80  82  84  86  88  107
## [18] 109 310 326 327 188 189 196 322 201 202 332 208 325 217 222 226 232
## [35] 238 323 255 321 262 271 288 328 292 293 294 295 329 330 304 305
```

Closeness

Inverse distance to all other nodes

```
closeness(foodNet, vids="96", mode="all")
```

```
##          96
## 0.002109705
```

```
closeness(foodNet, mode="all")[which.max(closeness(foodNet, mode="all"))]
```

```
##          84
## 0.002298851
```

Colorado chipmunk

Betweenness

Number of paths passing through a node

```r
betweenness(foodNet, v="96", directed=T)
```

```
##       96
## 438.0387
```

```r
betweenness(foodNet, directed=T)[which.max(betweenness(foodNet, directed=T))]
```

```
##       84
## 2895.905
```

Eigenvector

Values of the first eigenvector

```r
eigen_centrality(foodNet, directed=T)$vector[names(eigen_centrality(foodNet, directed=T)$vector) == "96
```

```
##       96
## 0.511795
```

```r
eigen_centrality(foodNet, directed=T)$vector[which.max(eigen_centrality(foodNet,
                                                                         directed=T)$vector)]
```

```
## 334
##   1
```

Domestic dog

Hub

```r
hub_score(foodNet)$vector[names(hub_score(foodNet)$vector) == "96"]
```

```
##         96
## 0.02091609
```

```r
hub_score(foodNet)$vector[which.max(hub_score(foodNet)$vector)]
```

```
## 222
##   1
```

Maize

Authority

```r
authority_score(foodNet)$vector[names(authority_score(foodNet)$vector) == "96"]
```

```
##        96
## 0.2598049
```

```r
authority_score(foodNet)$vector[which.max(authority_score(foodNet)$vector)]
```

```
## 107
##   1
```

Mule deer

Community detection

Community detection, essentially a way of finding clusters in our network, is another common type of analysis. There are a variety of methods available in `igraph`, described here:

```r
?igraph::communities
```

None of them can deal with directed networks, like we have here, so we need to convert it to an undirected network.

```
foodNetU = as.undirected(foodNet, mode="mutual")
```

Now we can try two different options that are both good to try for different uses:

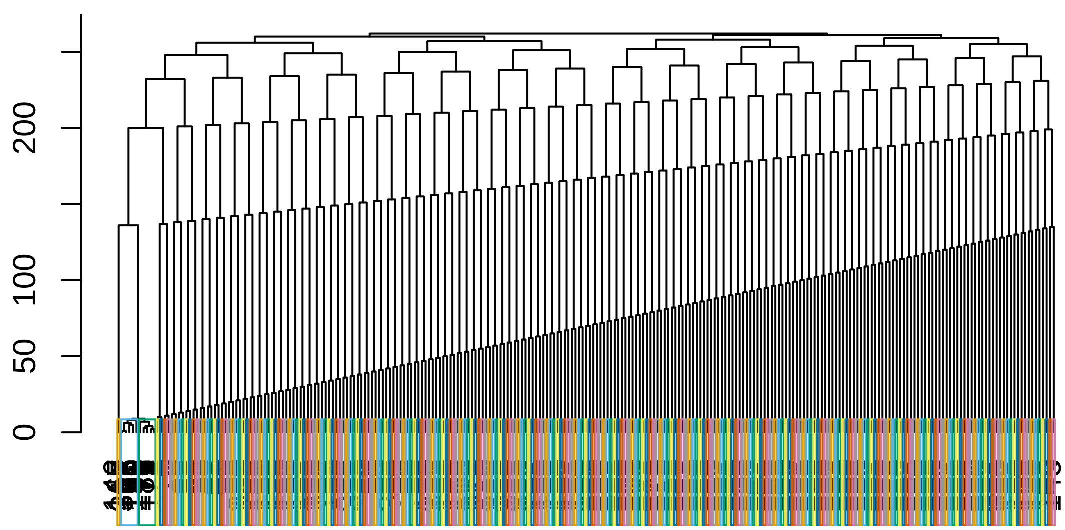
By greedy optimization of modularity

Fast, but potentially less accurate

```
foodNetCFG = cluster_fast_greedy(foodNetU)
foodNetCFG
```

```
## IGRAPH clustering fast greedy, groups: 255, mod: 0.17
## + groups:
##   $`1`
##   [1] "317" "65"  "167" "159" "166"
##
##   $`2`
##   [1] "175" "171" "172" "79"  "82"
##
##   $`3`
##   [1] "3"
##
##   $`4`
##   + ... omitted several groups/vertices
```

```
dendPlot(foodNetCFG, mode="hclust")
```



We can see that there are two real "communities" found here, shown as groups 1 and 2 above, and that almost all other nodes cluster by themselves in increasing distance from those communities.

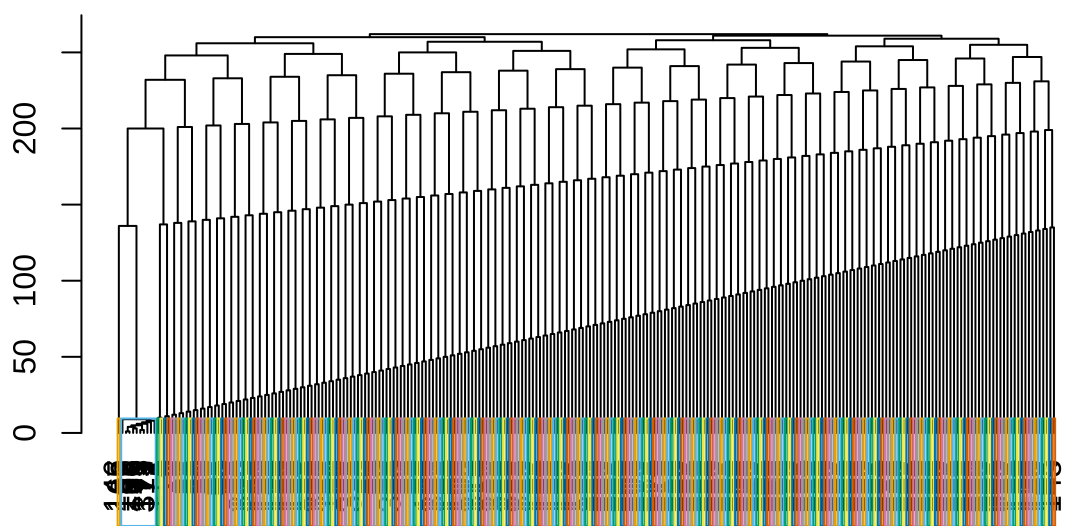
By edge betweenness

Slow with larger data sets, potentially more accurate

```
foodNetCB = cluster_edge_betweenness(foodNetU)
foodNetCB
```

```
## IGRAPH clustering edge betweenness, groups: 254, mod: 0
## + groups:
##   $`1`
##   [1] "3"
##
##   $`2`
##   [1] "4"
##
##   $`3`
##   [1] "5"
##
##   $`4`
##   + ... omitted several groups/vertices
```

```
dendPlot(foodNetCB, mode="hclust")
```



Roughly the same result, except we now have one community…

```
foodNetCB[20]
```

```
## $`20`
##  [1] "317" "65"  "175" "167" "159" "171" "166" "172" "79"  "82"
```

... that is the same as the other two communities found by the previous method, merged.

So, it seems that these nodes (mainly birds, arthropod parasites, and a couple mice) form a cohesive group in the food web, with everything else relatively isolated.

Geospatial analyses

There is so much that one can do with geospatial data that this will only be able to act as a cursory overview. For more details, you can consult some of the resources available.

I'll (briefly) cover three types of data that can be plotted spatially: points, polygons, and rasters.

Points

For this, let's use some data from the Global Biodiversity Information Facility, which is a worldwide database of species occurrences. We'll subset the data we work with here to plants in Ethiopia with non-missing geospatial data: https://www.gbif.org/occurrence/search?country=ET&has_coordinate=true&has_geospatial_issue=false&taxon_key=6

Click "DOWNLOAD" on the top, then wait for it to prepare the data and download the CSV file by clicking "CSV" and "DOWNLOAD." However, at least for the file I got, it turns out it's not comma separated, but tab separated, so change its extension to ".txt" and place it in your `/data` folder.

In this lecture, I download files manually and import them into R to show you how to do it the hard and messy way. However, many databases now, especially those with open GIS data, have their own R packages to load the data directly into R with reproducible code. The GBIF has one, `rgbif`, and you can learn more about how to use it from the rOpenSci tutorial: https://ropensci.org/tutorials/rgbif_tutorial/

```r
install.packages("rgbif")
```

However, here, we go ahead and load in the text file we loaded, using `readr`:

```r
library(readr)
```

```r
ethPlants = as.data.frame(read_tsv("./data/0009408-190415153152247.txt"))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   gbifID = col_double(),
##   decimalLatitude = col_double(),
##   decimalLongitude = col_double(),
##   coordinateUncertaintyInMeters = col_double(),
##   coordinatePrecision = col_logical(),
##   elevation = col_double(),
##   elevationAccuracy = col_double(),
##   depth = col_logical(),
##   depthAccuracy = col_logical(),
##   eventDate = col_datetime(format = ""),
##   day = col_double(),
##   month = col_double(),
##   year = col_double(),
##   taxonKey = col_double(),
##   speciesKey = col_double(),
```

```
##   dateIdentified = col_datetime(format = ""),
##   establishmentMeans = col_logical(),
##   lastInterpreted = col_datetime(format = "")
## )

## See spec(...) for full column specifications.

## Warning: 10236 parsing failures.
##   row                col           expected  actual                                 file
##  1603 locality           delimiter or quote         './data/0009408-190415153152247.txt'
##  1603 locality           delimiter or quote T       './data/0009408-190415153152247.txt'
##  4352 establishmentMeans 1/0/T/F/TRUE/FALSE MANAGED './data/0009408-190415153152247.txt'
## 15262 locality           delimiter or quote ,       './data/0009408-190415153152247.txt'
## 15262 locality           delimiter or quote o       './data/0009408-190415153152247.txt'
## ..... .................. .................. ....... ....................................
## See problems(...) for more details.
```

There are quite a few errors, but we're going to ignore them here and remove the last row that has all of the 10,000 messed up values in it. We're still left with 79,632 entries to work with.

```
ethPlants = ethPlants[-79633,]
head(ethPlants)
```

```
##       gbifID                           datasetKey
## 1   24720713 85771146-f762-11e1-a439-00145eb45e9a
## 2   35322954 84806e86-f762-11e1-a439-00145eb45e9a
## 3 1139762077 15f819bd-6612-4447-854b-14d12ee1022d
## 4 1139766810 15f819bd-6612-4447-854b-14d12ee1022d
## 5 1139786802 15f819bd-6612-4447-854b-14d12ee1022d
## 6 1139793104 15f819bd-6612-4447-854b-14d12ee1022d
##                                                         occurrenceID
## 1                                                               <NA>
## 2                                                               <NA>
## 3      http://data.biodiversitydata.nl/naturalis/specimen/L.2907355
## 4      http://data.biodiversitydata.nl/naturalis/specimen/L.2996350
## 5 http://data.biodiversitydata.nl/naturalis/specimen/L%20%200537678
## 6      http://data.biodiversitydata.nl/naturalis/specimen/L.3001151
##   kingdom       phylum         class       order        family
## 1 Plantae Tracheophyta Magnoliopsida     Apiales      Apiaceae
## 2 Plantae Tracheophyta Magnoliopsida  Sapindales Anacardiaceae
## 3 Plantae Tracheophyta Magnoliopsida Gentianales     Rubiaceae
## 4 Plantae Tracheophyta Magnoliopsida   Asterales Campanulaceae
## 5 Plantae Tracheophyta Magnoliopsida Boraginales  Boraginaceae
## 6 Plantae Tracheophyta Magnoliopsida   Asterales Campanulaceae
##          genus                 species infraspecificEpithet taxonRank
## 1 Macroselinum Macroselinum latifolium                 <NA>   SPECIES
## 2  Antrocaryon   Antrocaryon micraster                 <NA>   SPECIES
## 3       Galium      Galium aparinoides                 <NA>   SPECIES
## 4    Campanula        Campanula edulis                 <NA>   SPECIES
## 5  Cynoglossum Cynoglossum amplifolium                 <NA>   SPECIES
## 6      Lobelia          Lobelia exilis                 <NA>   SPECIES
##                                       scientificName countryCode
## 1              Peucedanum latifolium (M.Bieb.) DC.          ET
## 2       Antrocaryon micraster A.Chev. & Guillaumin          ET
## 3                         Galium aparinoides Forssk.          ET
## 4 Campanula rigidipila Steud. & Hochst. ex A.Rich.          ET
```

```
## 5          Cynoglossum amplifolium Hochst. ex DC.          ET
## 6               Lobelia exilis Hochst. ex A.Rich.          ET
##                                locality
## 1               am Nordabfall des Buahit
## 2                     Omo - member A1
## 3          ...Abyssinicum. Montis Silke.
## 4                     Montes Scholoda.
## 5 ad Endschedcap versus Schoata ad rivos.
## 6           ...Abyssinicum. Prope Gafta.
##                       publishingOrgKey decimalLatitude decimalLongitude
## 1 57254bd0-8256-11d8-b7ed-b8a03c50a862        13.00000         38.00000
## 2 c96dd940-165d-11da-a5ec-b8a03c50a862         5.43000         36.36000
## 3 396d5f30-dea9-11db-8ab4-b8a03c50a862        13.36667         38.28333
## 4 396d5f30-dea9-11db-8ab4-b8a03c50a862        14.18333         38.90000
## 5 396d5f30-dea9-11db-8ab4-b8a03c50a862        13.10000         38.08333
## 6 396d5f30-dea9-11db-8ab4-b8a03c50a862        14.25000         39.00000
##   coordinateUncertaintyInMeters coordinatePrecision elevation
## 1                            NA                  NA      3200
## 2                            NA                  NA        NA
## 3                            NA                  NA        NA
## 4                            NA                  NA        NA
## 5                            NA                  NA        NA
## 6                            NA                  NA        NA
##   elevationAccuracy depth depthAccuracy  eventDate day month year taxonKey
## 1                 0    NA            NA 1966-01-01  NA    NA 1966  3629762
## 2                NA    NA            NA       <NA>  NA    NA   NA  3660638
## 3                NA    NA            NA 1840-02-13  13     2 1840  2914306
## 4                NA    NA            NA 1837-06-07   7     6 1837  5411431
## 5                NA    NA            NA 1838-08-01   1     8 1838  7295212
## 6                NA    NA            NA 1838-09-13  13     9 1838  5408491
##   speciesKey      basisOfRecord institutionCode
## 1    3632734 PRESERVED_SPECIMEN             STU
## 2    3660638    FOSSIL_SPECIMEN            PBDB
## 3    2914306 PRESERVED_SPECIMEN            <NA>
## 4    5411424 PRESERVED_SPECIMEN            <NA>
## 5    7295212 PRESERVED_SPECIMEN            <NA>
## 6    5408491 PRESERVED_SPECIMEN            <NA>
##                                         collectionCode catalogNumber
## 1 Staatliches Museum für Naturkunde Stuttgart, Herbarium  Main-1-17720
## 2                                                22030        216748
## 3                                               Botany     L.2907355
## 4                                               Botany     L.2996350
## 5                                               Botany    L  0537678
## 6                                               Botany     L.3001151
##   recordNumber identifiedBy dateIdentified      license
## 1         <NA>         <NA>           <NA> CC_BY_NC_4_0
## 2         <NA>         <NA>           <NA>    CC_BY_4_0
## 3         <NA> Seegeler CJP           <NA>      CC0_1_0
## 4         <NA>         <NA>           <NA>      CC0_1_0
## 5         <NA>  Veldkamp JF     2005-09-01      CC0_1_0
## 6         <NA>         <NA>           <NA>      CC0_1_0
##                  rightsHolder    recordedBy typeStatus
## 1                        <NA> Sebald, Oskar       <NA>
## 2                        <NA>          <NA>       <NA>
```

```
## 3 Naturalis Biodiversity Center  Schimper GHW    SYNTYPE
## 4 Naturalis Biodiversity Center  Schimper GHW    ISOTYPE
## 5 Naturalis Biodiversity Center  Schimper GHW    ISOTYPE
## 6 Naturalis Biodiversity Center  Schimper GHW    ISOTYPE
##   establishmentMeans     lastInterpreted  mediaType
## 1                 NA 2018-12-23 17:14:12       <NA>
## 2                 NA 2018-12-23 19:22:43       <NA>
## 3                 NA 2019-04-05 03:25:42 STILLIMAGE
## 4                 NA 2019-04-05 03:26:18 STILLIMAGE
## 5                 NA 2019-04-05 03:03:03 STILLIMAGE
## 6                 NA 2019-04-05 03:26:19 STILLIMAGE
##                                                   issue
## 1 GEODETIC_DATUM_ASSUMED_WGS84;PRESUMED_SWAPPED_COORDINATE
## 2                             GEODETIC_DATUM_ASSUMED_WGS84
## 3                                                    <NA>
## 4                                                    <NA>
## 5                                                    <NA>
## 6                                                    <NA>
```

```
tail(ethPlants)
```

```
##           gbifID                           datasetKey
## 79627 1320939042 821cc27a-e3bb-4bc5-ac34-89ada245069d
## 79628 1320951083 821cc27a-e3bb-4bc5-ac34-89ada245069d
## 79629 1320956209 821cc27a-e3bb-4bc5-ac34-89ada245069d
## 79630 1320966452 821cc27a-e3bb-4bc5-ac34-89ada245069d
## 79631 1320992490 821cc27a-e3bb-4bc5-ac34-89ada245069d
## 79632 1320993394 821cc27a-e3bb-4bc5-ac34-89ada245069d
##                                                            occurrenceID
## 79627 http://n2t.net/ark:/65665/3a5464821-91ff-4a92-b25d-bd506bc92802
## 79628 http://n2t.net/ark:/65665/3a5cf2e63-b4dc-4ec5-afe7-bb8800bb6e52
## 79629 http://n2t.net/ark:/65665/3a609668c-d3bf-4808-bf3e-41309117adfb
## 79630 http://n2t.net/ark:/65665/3a67b448a-3aa5-4fbf-ae26-9416db526593
## 79631 http://n2t.net/ark:/65665/3a7a20692-34da-418c-abe5-5640d9f4d04a
## 79632 http://n2t.net/ark:/65665/3a7ab73ee-5523-4e01-95f3-8ab0f02e4a36
##       kingdom       phylum          class        order          family
## 79627 Plantae Tracheophyta Polypodiopsida Polypodiales  Dryopteridaceae
## 79628 Plantae Tracheophyta Polypodiopsida Polypodiales Thelypteridaceae
## 79629 Plantae Tracheophyta Polypodiopsida Polypodiales Thelypteridaceae
## 79630 Plantae Tracheophyta Polypodiopsida Polypodiales Thelypteridaceae
## 79631 Plantae Tracheophyta     Liliopsida  Alismatales Hydrocharitaceae
## 79632 Plantae Tracheophyta  Magnoliopsida  Celastrales     Celastraceae
##             genus               species infraspecificEpithet taxonRank
## 79627  Dryopteris Dryopteris kilemensis                 <NA>   SPECIES
## 79628 Amauropelta  Amauropelta bergiana                 <NA>   SPECIES
## 79629  Christella Christella gueinziana                 <NA>   SPECIES
## 79630 Amauropelta  Amauropelta bergiana                 <NA>   SPECIES
## 79631       Najas       Najas pectinata                 <NA>   SPECIES
## 79632 Gymnosporia   Gymnosporia serrata                 <NA>   SPECIES
##                               scientificName countryCode
## 79627   Dryopteris kilemensis (Kuhn) Kuntze          ET
## 79628 Thelypteris bergiana (Schltdl.) Ching          ET
## 79629  Christella gueinziana (Mett.) Holtt.          ET
## 79630 Thelypteris bergiana (Schltdl.) Ching          ET
## 79631                Najas pectinata Magnus          ET
```

```
## 79632      Catha edulis (Vahl) S.Endlicher         ET
##
## 79627 Kaffa. Zona a sud di Bonga. nella foresta umida de montana a crica un'or camino da Muti lungo
## 79628                   Ethiopia, 38 km. so. along Omonadda road after turnoff of Jimma Road at Littl
## 79629                                                              Ethiopia and Eritrea, IL _ 10 km S
## 79630                        Ethiopia: Kaffa Prov., 38 km. S along Omonadda Road after tunroff of Ji
## 79631                                                                                              La
## 79632
##                          publishingOrgKey decimalLatitude
## 79627 bc092ff0-02e4-11dc-991f-b8a03c50a862          7.1200
## 79628 bc092ff0-02e4-11dc-991f-b8a03c50a862          7.5200
## 79629 bc092ff0-02e4-11dc-991f-b8a03c50a862          8.0467
## 79630 bc092ff0-02e4-11dc-991f-b8a03c50a862          7.5200
## 79631 bc092ff0-02e4-11dc-991f-b8a03c50a862          9.4000
## 79632 bc092ff0-02e4-11dc-991f-b8a03c50a862          9.4000
##       decimalLongitude coordinateUncertaintyInMeters coordinatePrecision
## 79627          36.2000                            NA                  NA
## 79628          37.3800                            NA                  NA
## 79629          35.5233                            NA                  NA
## 79630          37.3800                            NA                  NA
## 79631          42.1167                            NA                  NA
## 79632          42.0200                            NA                  NA
##       elevation elevationAccuracy depth depthAccuracy  eventDate day month
## 79627      2000                NA    NA            NA 1967-01-01   1     1
## 79628      2460                NA    NA            NA 1962-01-10  10     1
## 79629      1950                NA    NA            NA 2005-08-30  30     8
## 79630        NA                NA    NA            NA 1962-01-10  10     1
## 79631        NA                NA    NA            NA 1965-01-17  17     1
## 79632      2000                NA    NA            NA 1965-01-10  10     1
##       year taxonKey speciesKey      basisOfRecord institutionCode
## 79627 1967  5655957    5655957 PRESERVED_SPECIMEN              US
## 79628 1962  5666647    4081215 PRESERVED_SPECIMEN              US
## 79629 2005  7586005    7586005 PRESERVED_SPECIMEN              US
## 79630 1962  5666647    4081215 PRESERVED_SPECIMEN              US
## 79631 1965  5329459    5329459 PRESERVED_SPECIMEN              US
## 79632 1965  3169158    7265714 PRESERVED_SPECIMEN              US
##       collectionCode catalogNumber recordNumber identifiedBy
## 79627         Botany       2858418         7109         <NA>
## 79628         Botany       2426205         7973         <NA>
## 79629         Botany       3547322        11977         <NA>
## 79630         Botany       2426206         7973         <NA>
## 79631         Botany       2598653         3606         <NA>
## 79632         Botany       2598620         3605         <NA>
##       dateIdentified license rightsHolder
## 79627           <NA> CC0_1_0         <NA>
## 79628           <NA> CC0_1_0         <NA>
## 79629           <NA> CC0_1_0         <NA>
## 79630           <NA> CC0_1_0         <NA>
## 79631           <NA> CC0_1_0         <NA>
## 79632           <NA> CC0_1_0         <NA>
##                                  recordedBy typeStatus
## 79627                  R. E. Pichi-Sermolli       <NA>
## 79628                           F. G. Meyer       <NA>
## 79629 I. Friis, G. S. McKee, A. Hailu & B. Yitbarek       <NA>
```

```
## 79630                                F. G. Meyer       <NA>
## 79631                                  W. Burger       <NA>
## 79632                                  W. Burger       <NA>
##       establishmentMeans     lastInterpreted  mediaType
## 79627                 NA 2018-12-06 01:24:35 STILLIMAGE
## 79628                 NA 2018-12-06 01:24:41 STILLIMAGE
## 79629                 NA 2018-12-06 01:24:44 STILLIMAGE
## 79630                 NA 2018-12-06 01:24:49 STILLIMAGE
## 79631                 NA 2018-12-06 01:25:03       <NA>
## 79632                 NA 2018-12-06 01:25:03       <NA>
##                            issue
## 79627 GEODETIC_DATUM_ASSUMED_WGS84
## 79628 GEODETIC_DATUM_ASSUMED_WGS84
## 79629 GEODETIC_DATUM_ASSUMED_WGS84
## 79630 GEODETIC_DATUM_ASSUMED_WGS84
## 79631 GEODETIC_DATUM_ASSUMED_WGS84
## 79632 GEODETIC_DATUM_ASSUMED_WGS84
```

```
str(ethPlants, give.attr=F)
```

```
## 'data.frame':    79632 obs. of  45 variables:
##  $ gbifID                       : num  2.47e+07 3.53e+07 1.14e+09 1.14e+09 1.14e+09 ...
##  $ datasetKey                   : chr  "85771146-f762-11e1-a439-00145eb45e9a" "84806e86-f762-11e1-a4
##  $ occurrenceID                 : chr  NA NA "http://data.biodiversitydata.nl/naturalis/specimen/L.2
##  $ kingdom                      : chr  "Plantae" "Plantae" "Plantae" "Plantae" ...
##  $ phylum                       : chr  "Tracheophyta" "Tracheophyta" "Tracheophyta" "Tracheophyta" .
##  $ class                        : chr  "Magnoliopsida" "Magnoliopsida" "Magnoliopsida" "Magnoliopsid
##  $ order                        : chr  "Apiales" "Sapindales" "Gentianales" "Asterales" ...
##  $ family                       : chr  "Apiaceae" "Anacardiaceae" "Rubiaceae" "Campanulaceae" ...
##  $ genus                        : chr  "Macroselinum" "Antrocaryon" "Galium" "Campanula" ...
##  $ species                      : chr  "Macroselinum latifolium" "Antrocaryon micraster" "Galium apa
##  $ infraspecificEpithet         : chr  NA NA NA NA ...
##  $ taxonRank                    : chr  "SPECIES" "SPECIES" "SPECIES" "SPECIES" ...
##  $ scientificName               : chr  "Peucedanum latifolium (M.Bieb.) DC." "Antrocaryon micraster
##  $ countryCode                  : chr  "ET" "ET" "ET" "ET" ...
##  $ locality                     : chr  "am Nordabfall des Buahit" "Omo - member A1" "...Abyssinicum.
##  $ publishingOrgKey             : chr  "57254bd0-8256-11d8-b7ed-b8a03c50a862" "c96dd940-165d-11da-a5
##  $ decimalLatitude              : num  13 5.43 13.37 14.18 13.1 ...
##  $ decimalLongitude             : num  38 36.4 38.3 38.9 38.1 ...
##  $ coordinateUncertaintyInMeters: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ coordinatePrecision          : logi  NA NA NA NA NA NA ...
##  $ elevation                    : num  3200 NA NA NA NA NA NA NA NA NA ...
##  $ elevationAccuracy            : num  0 NA NA NA NA NA NA NA NA NA ...
##  $ depth                        : logi  NA NA NA NA NA NA ...
##  $ depthAccuracy                : logi  NA NA NA NA NA NA ...
##  $ eventDate                    : POSIXct, format: "1966-01-01" NA ...
##  $ day                          : num  NA NA 13 7 1 13 4 26 29 NA ...
##  $ month                        : num  NA NA 2 6 8 9 10 10 12 NA ...
##  $ year                         : num  1966 NA 1840 1837 1838 ...
##  $ taxonKey                     : num  3629762 3660638 2914306 5411431 7295212 ...
##  $ speciesKey                   : num  3632734 3660638 2914306 5411424 7295212 ...
##  $ basisOfRecord                : chr  "PRESERVED_SPECIMEN" "FOSSIL_SPECIMEN" "PRESERVED_SPECIMEN" "
##  $ institutionCode              : chr  "STU" "PBDB" NA NA ...
##  $ collectionCode               : chr  "Staatliches Museum für Naturkunde Stuttgart, Herbarium" "220
##  $ catalogNumber                : chr  "Main-1-17720" "216748" "L.2907355" "L.2996350" ...
```

```
##  $ recordNumber                  : chr  NA NA NA NA ...
##  $ identifiedBy                  : chr  NA NA "Seegeler CJP" NA ...
##  $ dateIdentified                : POSIXct, format: NA NA ...
##  $ license                       : chr  "CC_BY_NC_4_0" "CC_BY_4_0" "CC0_1_0" "CC0_1_0" ...
##  $ rightsHolder                  : chr  NA NA "Naturalis Biodiversity Center" "Naturalis Biodiversity
##  $ recordedBy                    : chr  "Sebald, Oskar" NA "Schimper GHW" "Schimper GHW" ...
##  $ typeStatus                    : chr  NA NA "SYNTYPE" "ISOTYPE" ...
##  $ establishmentMeans            : logi  NA NA NA NA NA NA ...
##  $ lastInterpreted               : POSIXct, format: "2018-12-23 17:14:12" "2018-12-23 19:22:43" ...
##  $ mediaType                     : chr  NA NA "STILLIMAGE" "STILLIMAGE" ...
##  $ issue                         : chr  "GEODETIC_DATUM_ASSUMED_WGS84;PRESUMED_SWAPPED_COORDINATE" "G
```

We can see how many distinct classes of plants are represented in Ethiopia.

```
unique(ethPlants$class)
```

```
##  [1] "Magnoliopsida"     "Liliopsida"        "Pinopsida"
##  [4] "Polypodiopsida"    "Lycopodiopsida"    NA
##  [7] "Bryopsida"         "Marattiopsida"     "Equisetopsida"
## [10] "Psilotopsida"      "Anthocerotopsida"  "Jungermanniopsida"
## [13] "Florideophyceae"   "Marchantiopsida"
```

13, not including `NA`s. Let's remove those.

```
ethPlants = ethPlants[!is.na(ethPlants$class),]
```

Now let's load up `ggplot2` so we can plot some maps. There are lots of other packages that can handle geospatial mapping in R (and you can see some of them on the CRAN task view: https://cran.r-project.org/web/views/Spatial.html), including `sp` and `maps`. I prefer to use methods that are compatible with `ggplot2`, because it makes it easy to combine different data sources just by adding another layer. There are some quirks to navigate, however, and we'll encounter a few.

Note: Keep an eye on the `sf` package (for "simple features"), which is moving toward a common, open format for spatial data and is now supported in `ggplot2` with `geom_sf`: https://ggplot2.tidyverse.org/reference/ggsf.html

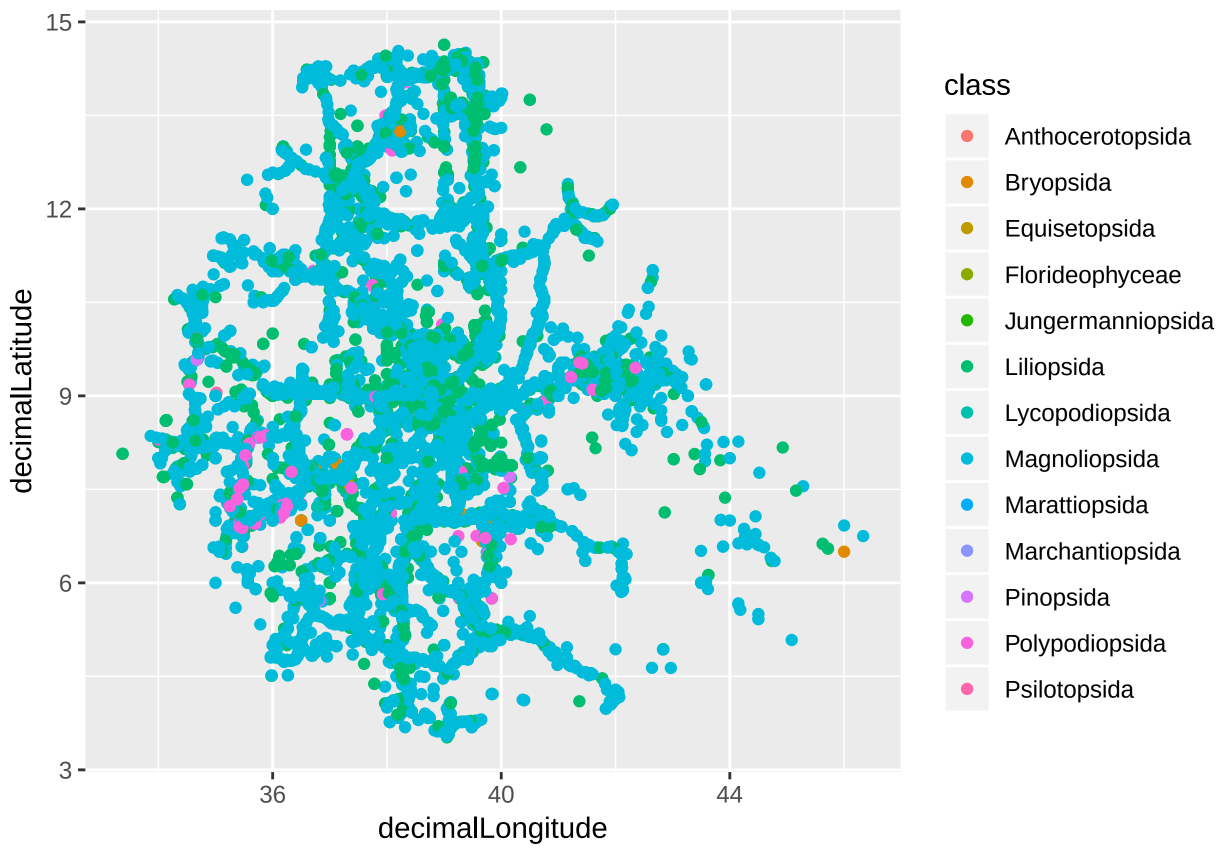
```
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(ggthemes)
```

Let's go ahead and do a quick plot of the points, colored by class, using their latitude and longitude as Y and X values, respectively.

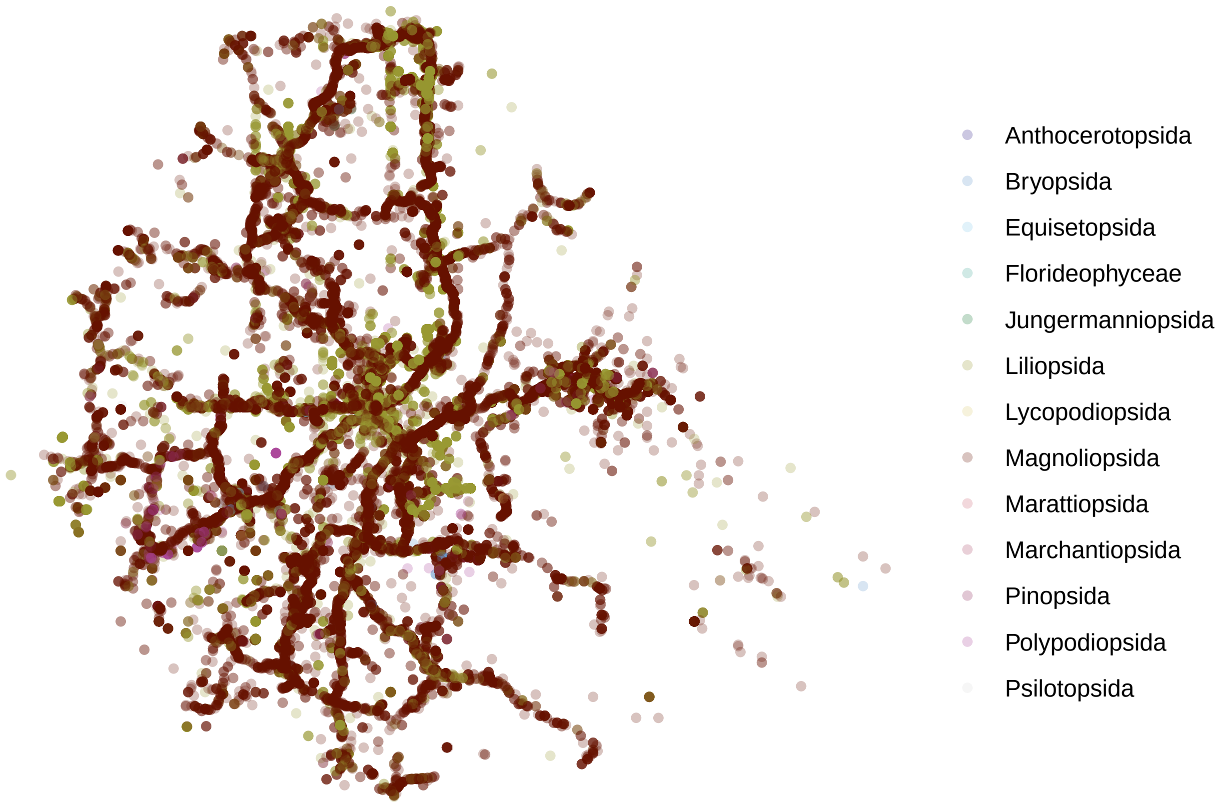
```
ggplot(ethPlants, aes(x=decimalLongitude, y=decimalLatitude)) +
  geom_point(aes(color=class))
```



Well, there's a ton of overlapping points, so it's hard to tell what's going on, but we can at least clearly see the outline of Ethiopia, so we know it's plotting correctly.

Let's simplify a bit, because I prefer minimal and clean spatial plots. We also critically add the `coord_cartesian()` layer, which will maintain the correct ratio for our axes.

```
ggplot(ethPlants, aes(x=decimalLongitude, y=decimalLatitude)) +
  geom_point(aes(color=class), shape=16, alpha=0.25) +
  scale_color_manual(values=c(ptol_pal()(12), "#DDDDDD")) +
  guides(color=guide_legend(title=NULL)) +
  coord_cartesian() +
  theme_void()
```



Shapefiles

Shapefiles are vector polygons (meaning they're represented as mathematical shapes and look good at any resolution) which usually represent spatial lines or boundaries.

A resource called Natural Earth has public domain shapefiles and rasters for areas across the world. Some areas (particularly outside the U.S. and Canada) are not represented at all resolutions, but it's good enough for most purposes.

We're going to download two "cultural" shapefiles (opposed to physical features or rasters) at 1:10m resolution: https://www.naturalearthdata.com/downloads/10m-cultural-vectors/

First, get "Admin 0 – Boundary Lines" by clicking "Download land boundaries."

Then, get "Admin 1 – States, Provinces" by clicking "Download boundary lines."

Place the contents of both ZIP files in your `/data` directory. You need all the files, not just the one with the ".shp" extension.

We then need to install the `rgdal` package (the R version of commonly used geospatial library GDAL) to read in our shapefiles with the `readOGR()` function.

```
install.packages("rgdal")
```

```
library(rgdal)
```

```
## Loading required package: sp
```

```
## rgdal: version: 1.4-3, (SVN revision 828)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
```

```
##  Loaded GDAL runtime: GDAL 2.2.3, released 2017/11/20
##  Path to GDAL shared files: C:/Program Files/R/R-3.6.0/library/rgdal/gdal
##  GDAL binary built with GEOS: TRUE
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: C:/Program Files/R/R-3.6.0/library/rgdal/proj
##  Linking to sp version: 1.3-1
```

```
?readOGR
```

```
neCountries = readOGR(dsn="./data", layer="ne_10m_admin_0_countries")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\MEGAsync\CSU\Courses\NR 592 (R Seminar)\Site\lectures\data", layer: "ne_10m_admin_0_coun
## with 255 features
## It has 94 fields
## Integer64 fields read as strings:  POP_EST NE_ID
```

```
neProvinces = readOGR(dsn="./data", layer="ne_10m_admin_1_states_provinces_lines")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\MEGAsync\CSU\Courses\NR 592 (R Seminar)\Site\lectures\data", layer: "ne_10m_admin_1_state
## with 10114 features
## It has 21 fields
```

The format is a little strange, in that the `dsn` argument is the path to the file, and the `layer` argument is the name of the files (without any extension). You'll notice a shapefile actually consists of the ".shp" file plus a bunch of other files with the same name that hold additional information. `readOGR()` wants all of them, so you just tell it the name that all the files share and it will load everything.

We then subset both spatial objects to Ethiopia (we don't need the whole world on our plot). We can find what variable we should use (because they're essentially data frames with other data included) by using `names()` first.

```
names(neCountries)
```

```
##  [1] "featurecla" "scalerank"  "LABELRANK"  "SOVEREIGNT" "SOV_A3"
##  [6] "ADM0_DIF"   "LEVEL"      "TYPE"       "ADMIN"      "ADM0_A3"
## [11] "GEOU_DIF"   "GEOUNIT"    "GU_A3"      "SU_DIF"     "SUBUNIT"
## [16] "SU_A3"      "BRK_DIFF"   "NAME"       "NAME_LONG"  "BRK_A3"
## [21] "BRK_NAME"   "BRK_GROUP"  "ABBREV"     "POSTAL"     "FORMAL_EN"
## [26] "FORMAL_FR"  "NAME_CIAWF" "NOTE_ADM0"  "NOTE_BRK"   "NAME_SORT"
## [31] "NAME_ALT"   "MAPCOLOR7"  "MAPCOLOR8"  "MAPCOLOR9"  "MAPCOLOR13"
## [36] "POP_EST"    "POP_RANK"   "GDP_MD_EST" "POP_YEAR"   "LASTCENSUS"
## [41] "GDP_YEAR"   "ECONOMY"    "INCOME_GRP" "WIKIPEDIA"  "FIPS_10_"
## [46] "ISO_A2"     "ISO_A3"     "ISO_A3_EH"  "ISO_N3"     "UN_A3"
## [51] "WB_A2"      "WB_A3"      "WOE_ID"     "WOE_ID_EH"  "WOE_NOTE"
## [56] "ADM0_A3_IS" "ADM0_A3_US" "ADM0_A3_UN" "ADM0_A3_WB" "CONTINENT"
## [61] "REGION_UN"  "SUBREGION"  "REGION_WB"  "NAME_LEN"   "LONG_LEN"
## [66] "ABBREV_LEN" "TINY"       "HOMEPART"   "MIN_ZOOM"   "MIN_LABEL"
## [71] "MAX_LABEL"  "NE_ID"      "WIKIDATAID" "NAME_AR"    "NAME_BN"
## [76] "NAME_DE"    "NAME_EN"    "NAME_ES"    "NAME_FR"    "NAME_EL"
## [81] "NAME_HI"    "NAME_HU"    "NAME_ID"    "NAME_IT"    "NAME_JA"
## [86] "NAME_KO"    "NAME_NL"    "NAME_PL"    "NAME_PT"    "NAME_RU"
## [91] "NAME_SV"    "NAME_TR"    "NAME_VI"    "NAME_ZH"
```

```
names(neProvinces)
```

```
##  [1] "featurecla" "name"       "adm0_a3"    "adm0_name"  "shape_leng"
##  [6] "mapcolor13" "mapcolor9"  "sov_a3"     "name_l"     "name_r"
```

```
## [11] "name_alt_l" "name_alt_r" "name_loc_l" "name_loc_r" "name_len_l"
## [16] "name_len_r" "note"       "type"       "min_zoom"   "min_label"
## [21] "scalerank"
```

```
neCountETH = neCountries[neCountries$NAME == "Ethiopia",]
neProvETH = neProvinces[neProvinces$adm0_name == "Ethiopia",]
```

And we can now remove the original objects from memory in R. This can be important, especially in geospatial work, because the files tend to be large and R requires that all of its objects are stored in memory. Your computer only has so much memory, and R has a limit to the amount of memory it can utilize, so freeing up space is important to keep R from freezing up.

```
rm(neCountries)
rm(neProvinces)
```

If you find you're having memory issues even after removing extraneous objects, you can run the `gc()` function (for "garbage collection") to free up any other memory that's tied up in R.

```
?gc
```

Taking a break to save your post-processed geospatial objects using `saveRDS()` for later loading (and keeping all of that separate in a "data prep" script) can also save you the hassle of processing them every time.

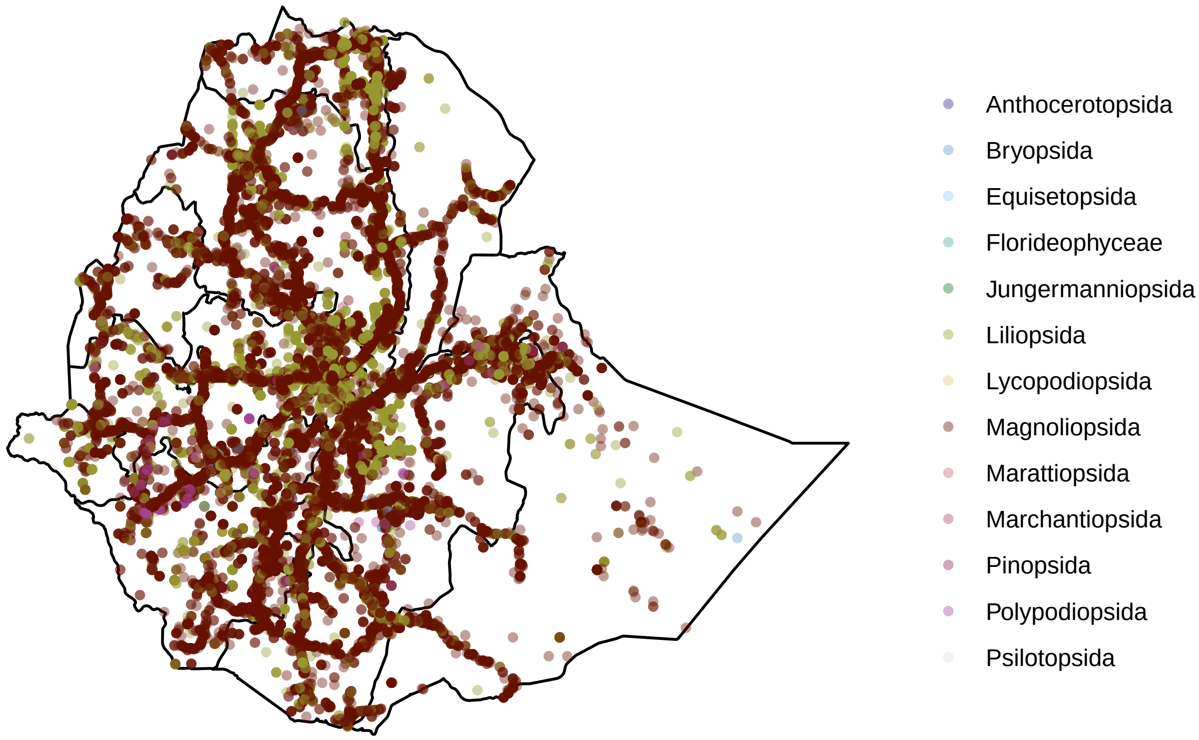
Now, to plot shapefiles in `ggplot()` we have two options, `geom_polygon()` (for closed shapes) and `geom_path()` (for lines). We downloaded lines for our administrative boundaries, so we use `geom_path()` to add them to our previous plot.

```
?geom_polygon
?geom_path
```

Note: It's important to include `group=group` in your aesthetics so the lines stay grouped into the intended shapes.

```
ggplot(ethPlants, aes(x=decimalLongitude, y=decimalLatitude)) +
  geom_path(data=neCountETH, aes(x=long, y=lat, group=group),
              color = "#000000", size=0.5, lty="solid") +
  geom_path(data=neProvETH, aes(x=long, y=lat, group=group),
            color = "#000000", size=0.5, lty="solid") +
  geom_point(aes(color=class), shape=16, alpha=0.4) +
  scale_color_manual(values=c(ptol_pal()(12), "#DDDDDD")) +
  guides(color=guide_legend(title=NULL)) +
  coord_cartesian(xlim=c(32.7, 48.3), ylim=c(2.9, 15.3)) +
  theme_void()
```

```
## Regions defined for each Polygons
```



Now, because I have a suspicion about those long continuous lines of points, let's find a shapefile of Ethiopian roads to plot on top.

We can get one from the World Food Programme: https://geonode.wfp.org/layers/ogcserver.gis.wfp.org%3Ageonode%3Aeth_trs_roads_osm

Click "Download Layer," choose "Zipped Shapefile," and place the unzipped files in the `/data` directory, as usual.

We then load it in the same way, and subset it to only show the main roads rather than every road in the country.

```
roadsETH = readOGR(dsn="./data",layer="eth_trs_roads_osm")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\MEGAsync\CSU\Courses\NR 592 (R Seminar)\Site\lectures\data", layer: "eth_trs_roads_osm"
## with 67559 features
## It has 20 fields
```

```
names(roadsETH)
```

```
##  [1] "id"         "osm_id"     "sourceid"   "notes"      "onme"
##  [6] "rtenme"     "ntlclass"   "fclass"     "numlanes"   "srftpe"
## [11] "srfcond"    "isseasonal" "curntprac"  "gnralspeed" "rdwidthm"
## [16] "status"     "iselevated" "iso3"       "country"    "last_updat"
```

```
unique(roadsETH$ntlclass)
```

```
##  [1] tertiary       track          secondary_link tertiary_link
##  [5] trunk_link     primary_link   road           motorway_link
##  [9] unclassified   trunk          primary        secondary
```

```
## [13] motorway
## 13 Levels: motorway motorway_link primary primary_link road ... unclassified
```

```
roadsETH = roadsETH[roadsETH$ntlclass == "primary" |
                    roadsETH$ntlclass == "secondary" |
                    roadsETH$ntlclass == "tertiary",]
```

Then we can plot the roads atop the points, as intended:

```
ggplot(ethPlants, aes(x=decimalLongitude, y=decimalLatitude)) +
  geom_path(data=neCountETH, aes(x=long, y=lat, group=group),
            color = "#000000", size=0.5, lty="solid") +
  geom_path(data=neProvETH, aes(x=long, y=lat, group=group),
           color = "#000000", size=0.5, lty="solid") +
  geom_point(aes(color=class), shape=16, alpha=0.4) +
  geom_path(data=roadsETH, aes(x=long, y=lat, group=group),
           color = "gray", size=0.2, lty="solid") +
  scale_color_manual(values=c(ptol_pal()(12), "#DDDDDD")) +
  guides(color=guide_legend(title=NULL)) +
  coord_cartesian(xlim=c(32.7, 48.3), ylim=c(2.9, 15.3)) +
  theme_void()
```

```
## Regions defined for each Polygons
```

And we see that they match up perfectly with the long lines of points, showing that most sampling has taken place on roadsides. If we were using these data for a project, it would be good to consider what influences that biased sampling may have on the data and our analyses.

Rasters

Rasters are static image files, usually used as backgrounds or "basemaps" for geospatial work. GeoTIFFs are commonly used for this purpose (though there are other formats), which are TIFF image files with geographic coordinates embedded in them.

We can get a variety of different rasters with varying color palettes from Natural Earth: https://www.naturalearthdata.com/downloads/50m-raster-data/

In this case, we'll use "Natural Earth 1" at 1:50m resolution: https://www.naturalearthdata.com/downloads/50m-raster-data/50m-natural-earth-1/

Under "Natural Earth I with Shaded Relief", click "Download small size" and place the files in your `/data` directory.

To import raster files, we need the `raster` package.

```
install.packages("raster")
```

```
library(raster)
```

As we'll see, this raster file is actually composed of three layers, each of which represents one of the Red, Green, and Blue color channels. Normally we would use the `raster()` function to load it in, but since we know it's a "multi-band" raster, we need to use the `stack()` or `brick()` function. There are minor differences between the two; we'll use `brick()` here.

```
ne1 = brick("./data/NE1_50M_SR.tif")
```

Then we want to crop the raster to a bounding box around the rough extent of Ethiopia (again, because we don't need to represent the whole world and it would take a lot longer to plot). We've used these points as our axis limits previously. There are different tools you can use to find a bounding box, including this one and this one.

We then remove the full raster object and convert our raster shape to a data frame, because `ggplot2` can't work with raster files directly (another function to accomplish this conversion is `fortify()`).

```
ne1ETH = crop(ne1, c(32.7, 48.3, 2.9, 15.3))
rm(ne1)

ne1ETHdf = as.data.frame(ne1ETH, xy=T)
```

```
head(ne1ETHdf)
```

```
##          x        y NE1_50M_SR.1 NE1_50M_SR.2 NE1_50M_SR.3
## 1 32.71667 15.28333          231          225          193
## 2 32.75000 15.28333          229          224          192
## 3 32.78333 15.28333          229          219          193
## 4 32.81667 15.28333          228          216          193
## 5 32.85000 15.28333          225          210          189
## 6 32.88333 15.28333          226          216          189
```

```
str(ne1ETHdf)
```

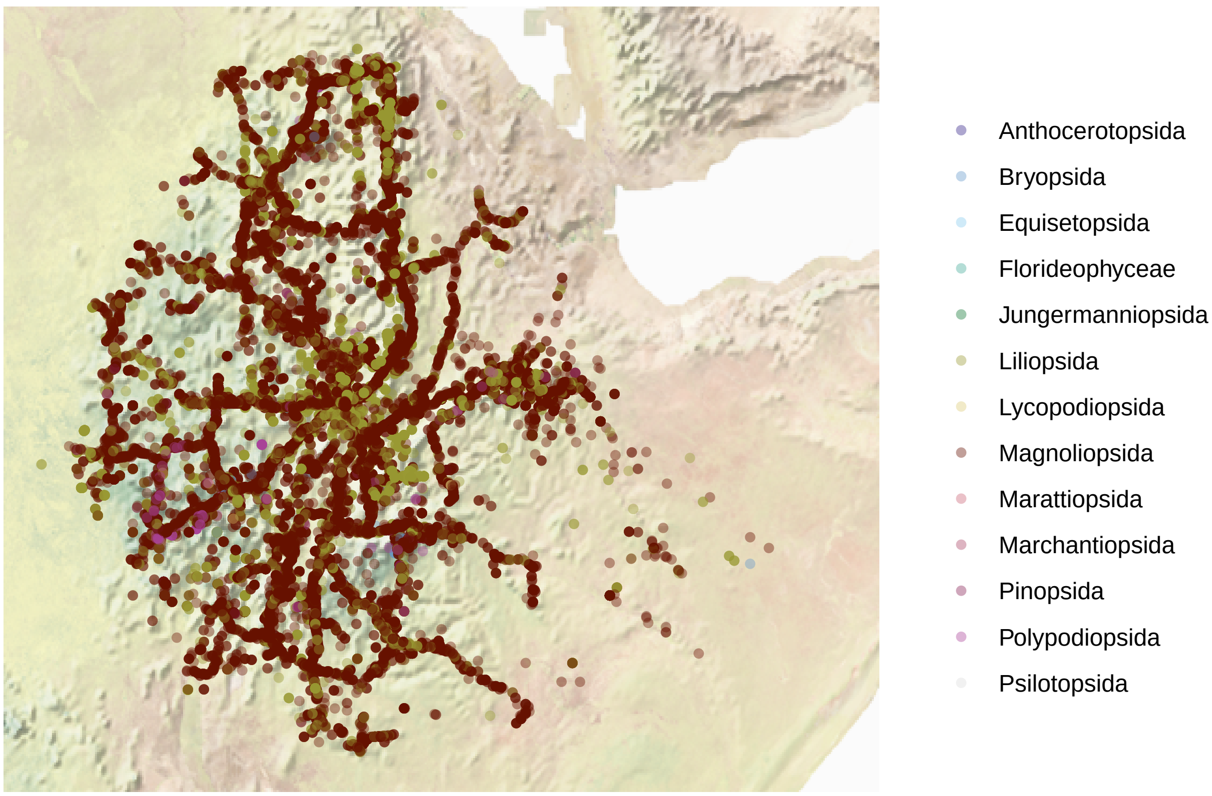
```
## 'data.frame':    174096 obs. of  5 variables:
##  $ x           : num  32.7 32.8 32.8 32.8 32.9 ...
##  $ y           : num  15.3 15.3 15.3 15.3 15.3 ...
##  $ NE1_50M_SR.1: int  231 229 229 228 225 226 227 229 233 232 ...
##  $ NE1_50M_SR.2: int  225 224 219 216 210 216 220 220 225 219 ...
##  $ NE1_50M_SR.3: int  193 192 193 193 189 189 190 193 196 195 ...
```

Remember how our raster file had three layers for different color channels? Now it's time to convert those to hexadecimal values, so we can plot the correct colors:

```
ne1ETHdf$color=rgb(ne1ETHdf$NE1_50M_SR.1, ne1ETHdf$NE1_50M_SR.2,
                   ne1ETHdf$NE1_50M_SR.3, maxColorValue=255)
```

And we plot it (underneath our points):

```
# Warning: Long!
ggplot(ethPlants, aes(x=decimalLongitude, y=decimalLatitude)) +
  geom_raster(data=ne1ETHdf, aes(x=x, y=y), fill=ne1ETHdf$color) +
  geom_point(aes(color=class), shape=16, alpha=0.4) +
  scale_color_manual(values=c(ptol_pal()(12), "#DDDDDD")) +
  guides(color=guide_legend(title=NULL)) +
  coord_cartesian(xlim=c(32.7, 48.3), ylim=c(2.9, 15.3)) +
  theme_void()
```



Looks good. For a finished plot we might want a higher resolution raster, but it would take longer to process and plot. ggplot2, for whatever reason, really does not do well with raster files and takes a lot of time and memory plotting them. If you have to do a lot of raster work, you might be better off checking out another plotting package.

Point-in-polygon

There are a variety of other tasks we can do with geospatial data aside from just visualizing it. Maybe we want to find which points are within a particular polygon's boundaries? This is called, appropriately, "point-in-polygon" analysis.

We'll need to go back to Natural Earth to get the actual shapes for the provinces, instead of just the lines: https://www.naturalearthdata.com/downloads/10m-cultural-vectors/

This time, click "Download states and provinces" and put the contents in `/data`.

```
neProvShape = readOGR(dsn="./data",layer="ne_10m_admin_1_states_provinces")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:\MEGAsync\CSU\Courses\NR 592 (R Seminar)\Site\lectures\data", layer: "ne_10m_admin_1_stat
## with 4594 features
## It has 83 fields
## Integer64 fields read as strings:  ne_id
```

```
names(neProvShape)
```

```
##  [1] "featurecla" "scalerank"  "adm1_code"  "diss_me"    "iso_3166_2"
##  [6] "wikipedia"  "iso_a2"     "adm0_sr"    "name"       "name_alt"
## [11] "name_local" "type"       "type_en"    "code_local" "code_hasc"
## [16] "note"       "hasc_maybe" "region"     "region_cod" "provnum_ne"
## [21] "gadm_level" "check_me"   "datarank"   "abbrev"     "postal"
## [26] "area_sqkm"  "sameascity" "labelrank"  "name_len"   "mapcolor9"
## [31] "mapcolor13" "fips"       "fips_alt"   "woe_id"     "woe_label"
## [36] "woe_name"   "latitude"   "longitude"  "sov_a3"     "adm0_a3"
## [41] "adm0_label" "admin"      "geonunit"   "gu_a3"      "gn_id"
## [46] "gn_name"    "gns_id"     "gns_name"   "gn_level"   "gn_region"
## [51] "gn_a1_code" "region_sub" "sub_code"   "gns_level"  "gns_lang"
## [56] "gns_adm1"   "gns_region" "min_label"  "max_label"  "min_zoom"
## [61] "wikidataid" "name_ar"    "name_bn"    "name_de"    "name_en"
## [66] "name_es"    "name_fr"    "name_el"    "name_hi"    "name_hu"
## [71] "name_id"    "name_it"    "name_ja"    "name_ko"    "name_nl"
## [76] "name_pl"    "name_pt"    "name_ru"    "name_sv"    "name_tr"
## [81] "name_vi"    "name_zh"    "ne_id"
```

```
neProvShapeETH = neProvShape[neProvShape$adm0_a3 == "ETH",]
rm(neProvShape)
```

We'll use the `over()` function from the `sp` package to do this.

```
?sp::over
```

The operation itself is a little tricky. We need our plant occurrence points (currently a data frame) to be a `SpatialPointsDataFrame` object, with coordinates and a map projection (which we haven't covered, but is important in geospatial tasks). We add these with `coordinates()` and `proj4string()`, respectively.

```
ethPlantsSP = ethPlants[!is.na(ethPlants$decimalLongitude) &
                          !is.na(ethPlants$decimalLatitude),]
coordinates(ethPlantsSP) = ~ decimalLongitude + decimalLatitude
proj4string(ethPlantsSP) = CRS(proj4string(neProvShapeETH))
class(ethPlantsSP)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

Then we can see which plants fall within a specific province of Ethiopia (here we'll use the border region of Gambela). To get the actual subset of those points, though, we then have to subset the original plant data frame by the rows of the returned version of our shapefile that are not `NA`. It's unintuitive but, if you do it this way, it should work out fine.

```
names(neProvShapeETH)
```

```
##  [1] "featurecla" "scalerank"  "adm1_code"  "diss_me"    "iso_3166_2"
##  [6] "wikipedia"  "iso_a2"     "adm0_sr"    "name"       "name_alt"
## [11] "name_local" "type"       "type_en"    "code_local" "code_hasc"
## [16] "note"       "hasc_maybe" "region"     "region_cod" "provnum_ne"
## [21] "gadm_level" "check_me"   "datarank"   "abbrev"     "postal"
## [26] "area_sqkm"  "sameascity" "labelrank"  "name_len"   "mapcolor9"
## [31] "mapcolor13" "fips"       "fips_alt"   "woe_id"     "woe_label"
## [36] "woe_name"   "latitude"   "longitude"  "sov_a3"     "adm0_a3"
## [41] "adm0_label" "admin"      "geonunit"   "gu_a3"      "gn_id"
## [46] "gn_name"    "gns_id"     "gns_name"   "gn_level"   "gn_region"
## [51] "gn_a1_code" "region_sub" "sub_code"   "gns_level"  "gns_lang"
## [56] "gns_adm1"   "gns_region" "min_label"  "max_label"  "min_zoom"
## [61] "wikidataid" "name_ar"    "name_bn"    "name_de"    "name_en"
## [66] "name_es"    "name_fr"    "name_el"    "name_hi"    "name_hu"
## [71] "name_id"    "name_it"    "name_ja"    "name_ko"    "name_nl"
## [76] "name_pl"    "name_pt"    "name_ru"    "name_sv"    "name_tr"
## [81] "name_vi"    "name_zh"    "ne_id"
```

```
ethPlantsInGambela = over(ethPlantsSP,
                          neProvShapeETH[neProvShapeETH$name == "Gambela Peoples",])
ethPlantsInGambela = ethPlants[!is.na(ethPlantsInGambela$name),]

nrow(ethPlants)
```

```
## [1] 79611
```

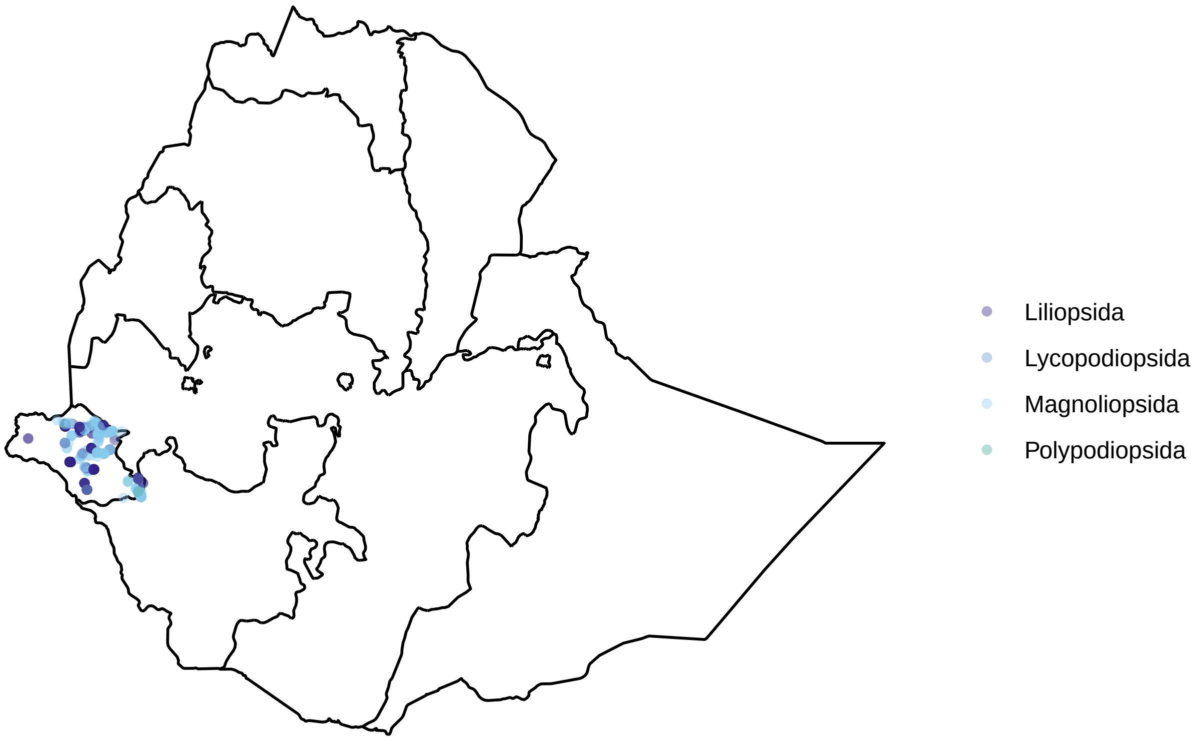
```
nrow(ethPlantsInGambela)
```

```
## [1] 1260
```

Let's plot the resulting set of points to do a visual check.

```
ggplot(ethPlantsInGambela, aes(x=decimalLongitude, y=decimalLatitude)) +
  geom_path(data=neCountETH, aes(x=long, y=lat, group=group),
             color = "#000000", size=0.5, lty="solid") +
  geom_path(data=neProvETH, aes(x=long, y=lat, group=group),
            color = "#000000", size=0.5, lty="solid") +
  geom_point(aes(color=class), shape=16, alpha=0.4) +
  scale_color_manual(values=c(ptol_pal()(12), "#DDDDDD")) +
  guides(color=guide_legend(title=NULL)) +
  coord_cartesian(xlim=c(32.7, 48.3), ylim=c(2.9, 15.3)) +
  theme_void()
```

```
## Regions defined for each Polygons
```



Yep! All in Gambela, which is that far western region. Looks good. We could then use this method do some statistical comparisons of species occurrences or community composition in other provinces, if we wished.

Geodesic distance

We've dealt with distance matrices before, and one very easy-to-understand application is to calculate a distance matrix of geographic points, which is the literal geographic distance between those points. This isn't quite as simple as taking Euclidean distances of lat-long coordinates, however, because we need to account for the curvature and shape of the Earth. Luckily, there are a number of packages that can do it for us. The one we'll use here is `geosphere`, and the function is `distm()`.

```
install.packages("geosphere")
```

```
library(geosphere)
```

```
?geosphere::distm
```

Then it's as simple as feeding in our longitudes and latitudes (in that order). We also subset the plants data frame to ocurrences with species data (in anticipation of our next step).

```
gambelaDist = distm(ethPlantsInGambela[!is.na(ethPlantsInGambela$species),
                                       c("decimalLongitude","decimalLatitude")])
gambelaDist = as.dist(gambelaDist)
as.matrix(gambelaDist)[1:10,1:10]
```

```
##            1          2          3          4          5          6
## 1      0.000  11114.199 138245.650 138245.650 137256.689 137256.689
## 2  11114.199      0.000 129734.612 129734.612 128664.742 128664.742
```

```
## 3  138245.650 129734.612      0.000      0.000   1836.288   1836.288
## 4  138245.650 129734.612      0.000      0.000   1836.288   1836.288
## 5  137256.689 128664.742   1836.288   1836.288      0.000      0.000
## 6  137256.689 128664.742   1836.288   1836.288      0.000      0.000
## 7  137256.689 128664.742   1836.288   1836.288      0.000      0.000
## 8  137256.689 128664.742   1836.288   1836.288      0.000      0.000
## 9    5827.921   5578.401 134672.717 134672.717 133632.147 133632.147
## 10  96138.339  89225.707  46988.377  46988.377  46664.003  46664.003
##             7          8          9       10
## 1  137256.689 137256.689   5827.921 96138.34
## 2  128664.742 128664.742   5578.401 89225.71
## 3    1836.288   1836.288 134672.717 46988.38
## 4    1836.288   1836.288 134672.717 46988.38
## 5       0.000      0.000 133632.147 46664.00
## 6       0.000      0.000 133632.147 46664.00
## 7       0.000      0.000 133632.147 46664.00
## 8       0.000      0.000 133632.147 46664.00
## 9  133632.147 133632.147      0.000 93507.68
## 10  46664.003  46664.003  93507.684     0.00
```

We get a very big distance matrix, because it calculated pairwise distances between every combination of points in Gambela (the unit of the values shown is meters).

Mantel tests

When you have geographic distances, one common analysis is to compare that matrix to a distance matrix for another variable that those points share. This is usually done to check for spatial autocorrelation, or how much points are similar simply due to their being geographically close to one another (and affected by common conditions or processes that cause that similarity).

One option that we have data for here is to compute taxonomic distance between species. We can then see if the taxonomic distance is correlated with the geographic distance. An interesting question!

We'll need a couple functions in the `vegan` package.

```
library(vegan)
```

```
## Loading required package: permute

##
## Attaching package: 'permute'

## The following object is masked from 'package:igraph':
##
##     permute

## Loading required package: lattice

## This is vegan 2.5-4

##
## Attaching package: 'vegan'

## The following object is masked from 'package:igraph':
##
##     diversity
```

We determine the taxonomic distance (number of steps through a taxonomic tree) between each pair of species…

```
gambelaTaxa = ethPlantsInGambela[!duplicated(ethPlantsInGambela$species) &
                                   !is.na(ethPlantsInGambela$species),9:5]
rownames(gambelaTaxa) = ethPlantsInGambela[!duplicated(ethPlantsInGambela$species) &
                                   !is.na(ethPlantsInGambela$species),]$species
gambelaTaxa[1:10,]
```

```
##                               genus       family        order
## Celtis philippensis          Celtis  Cannabaceae      Rosales
## Strychnos mitis           Strychnos  Loganiaceae  Gentianales
## Raphionacme splendens   Raphionacme  Apocynaceae  Gentianales
## Ethulia conyzoides          Ethulia   Asteraceae    Asterales
## Acmella caulirhiza          Acmella   Asteraceae    Asterales
## Bidens biternata             Bidens   Asteraceae    Asterales
## Tridax procumbens            Tridax   Asteraceae    Asterales
## Baccharoides calvoana Baccharoides   Asteraceae    Asterales
## Colocasia esculenta       Colocasia      Araceae  Alismatales
## Asplenium aethiopicum     Asplenium Aspleniaceae Polypodiales
##                               class       phylum
## Celtis philippensis    Magnoliopsida Tracheophyta
## Strychnos mitis        Magnoliopsida Tracheophyta
## Raphionacme splendens  Magnoliopsida Tracheophyta
## Ethulia conyzoides     Magnoliopsida Tracheophyta
## Acmella caulirhiza     Magnoliopsida Tracheophyta
## Bidens biternata       Magnoliopsida Tracheophyta
## Tridax procumbens      Magnoliopsida Tracheophyta
## Baccharoides calvoana  Magnoliopsida Tracheophyta
## Colocasia esculenta       Liliopsida Tracheophyta
## Asplenium aethiopicum Polypodiopsida Tracheophyta
```

```
gambelaTaxaDist = taxa2dist(gambelaTaxa)
gambelaTaxaDist = as.matrix(gambelaTaxaDist)
gambelaTaxaDist[1:10,1:10]
```

```
##                       Celtis philippensis Strychnos mitis
## Celtis philippensis                     0              80
## Strychnos mitis                        80               0
## Raphionacme splendens                  80              60
## Ethulia conyzoides                     80              80
## Acmella caulirhiza                     80              80
## Bidens biternata                       80              80
## Tridax procumbens                      80              80
## Baccharoides calvoana                  80              80
## Colocasia esculenta                   100             100
## Asplenium aethiopicum                 100             100
##                       Raphionacme splendens Ethulia conyzoides
## Celtis philippensis                      80                 80
## Strychnos mitis                          60                 80
## Raphionacme splendens                     0                 80
## Ethulia conyzoides                       80                  0
## Acmella caulirhiza                       80                 40
## Bidens biternata                         80                 40
## Tridax procumbens                        80                 40
## Baccharoides calvoana                    80                 40
## Colocasia esculenta                     100                100
## Asplenium aethiopicum                   100                100
```

```
##                         Acmella caulirhiza Bidens biternata
## Celtis philippensis                     80               80
## Strychnos mitis                         80               80
## Raphionacme splendens                   80               80
## Ethulia conyzoides                      40               40
## Acmella caulirhiza                       0               40
## Bidens biternata                        40                0
## Tridax procumbens                       40               40
## Baccharoides calvoana                   40               40
## Colocasia esculenta                    100              100
## Asplenium aethiopicum                  100              100
##                         Tridax procumbens Baccharoides calvoana
## Celtis philippensis                    80                    80
## Strychnos mitis                        80                    80
## Raphionacme splendens                  80                    80
## Ethulia conyzoides                     40                    40
## Acmella caulirhiza                     40                    40
## Bidens biternata                       40                    40
## Tridax procumbens                       0                    40
## Baccharoides calvoana                  40                     0
## Colocasia esculenta                   100                   100
## Asplenium aethiopicum                 100                   100
##                         Colocasia esculenta Asplenium aethiopicum
## Celtis philippensis                     100                   100
## Strychnos mitis                         100                   100
## Raphionacme splendens                   100                   100
## Ethulia conyzoides                      100                   100
## Acmella caulirhiza                      100                   100
## Bidens biternata                        100                   100
## Tridax procumbens                       100                   100
## Baccharoides calvoana                   100                   100
## Colocasia esculenta                       0                   100
## Asplenium aethiopicum                   100                     0
```

And then we manually construct a matrix with a row and column for each row in our data, and fill it with the appropriate taxonomic distances using nested `for()` loops.

```
ethPlantsInGambela = ethPlantsInGambela[!is.na(ethPlantsInGambela$species),]
gambelaTaxaDistByRow = matrix(data=0,
                              nrow=nrow(ethPlantsInGambela),
                              ncol=nrow(ethPlantsInGambela))
for (i in 1:nrow(ethPlantsInGambela)) {
  for (j in 1:nrow(ethPlantsInGambela)) {
    gambelaTaxaDistByRow[i,j] = gambelaTaxaDist[rownames(as.matrix(gambelaTaxaDist)) ==
                                                  ethPlantsInGambela[i, "species"],
                                                colnames(as.matrix(gambelaTaxaDist)) ==
                                                  ethPlantsInGambela[j, "species"]]
  }
}
gambelaTaxaDistByRow = as.dist(gambelaTaxaDistByRow)
as.matrix(gambelaTaxaDistByRow)[1:10,1:10]
```

```
##     1   2   3   4   5   6   7   8   9  10
## 1   0  80  80  80  80  80  80  80  80 100
## 2  80   0  60  60  80  80  80  80  80 100
```

```
## 3    80   60    0    0   80   80   80   80   80  100
## 4    80   60    0    0   80   80   80   80   80  100
## 5    80   80   80   80    0   40   40   40   40  100
## 6    80   80   80   80   40    0   40   40   40  100
## 7    80   80   80   80   40   40    0   40   40  100
## 8    80   80   80   80   40   40   40    0   40  100
## 9    80   80   80   80   40   40   40   40    0  100
## 10  100  100  100  100  100  100  100  100  100    0
```

And finally we can run our test, which is called a Mantel test:

```
?vegan::mantel
```

```
mantel(gambelaDist, gambelaTaxaDistByRow)
```

```
##
## Mantel statistic based on Pearson's product-moment correlation
##
## Call:
## mantel(xdis = gambelaDist, ydis = gambelaTaxaDistByRow)
##
## Mantel statistic r: 0.1598
##       Significance: 0.001
##
## Upper quantiles of permutations (null model):
##    90%    95%  97.5%    99%
## 0.0237 0.0290 0.0359 0.0439
## Permutation: free
## Number of permutations: 999
```

We can say with good confidence that the taxonomic distance between occurrences is independent of geographic distance.

Geocoding

Another fun thing to do is to turn text descriptions of locations (like you would type into Google Maps) into sets of latitude and longitude coordinates.

One way to do this is to use the `geocode()` function in the `ggmap` package. Unfortunately, Google now requires you to sign up for an API key, which I won't get into here, but it can be very useful to turn something like open survey responses on locations into points you can easily plot on a map (or export to KML format for Google Earth).

```
install.packages("ggmap")
```

```
?ggmap::geocode
```

(pdf / Rmd)