Stream Ecology in R: an introductory GIS workshop

Alex Franzen

8 June 2022

In this lesson you will learn about some basic concepts in stream ecology, such as calculating river distance and conducting a mantel test. These are useful for calculating how connected river systems are and testing for an isolation-by-distance effect. We'll go into more detail as we get further along in the lesson, but this is the gist of what we are going to do. For the most part, all you need to do is run the lines of code that I have written, but for some thing you will need to edit lines of code for it to run on your computer. Lines of code that will need to be edited by you will have a "#" in front of them and an instruction on what to do. If you have any questions, please email me at ajfranzen@ou.edu or feel free to use the resources of the internet if you are able to find a different solution. The beauty of R is that it is open source and continually evolving, so there are almost always multiple paths to accomplish what you are trying to do.

Part 1: Introduction to mapping spatial data

```
## This first part will be about learning to map spatial data using R. Usually, GIS analyses and mapping

## First, let's get an idea of what our study area looks like by mapping the a river basin is southeast

## Spatial data can be stored in a number of different formats, but one of the most common is called a

## First things first though, we will need to install and load the following packages that we will be u

## To install a package, you can use this generic format: install.packages('package.name')

## These are the packages you will need to install (if you have not previously). Un-comment the next fe

#install.packages('sf')
#install.packages('sp')
#install.packages('rgdal')
#install.packages('ggplot2')
#install.packages('dplyr')
#install.packages('maps')
#install.packages('ggspatial')

## Once you install all of the packages, you need to load them into the workspace. To load a package, e

library(sf)
```

```
## Warning: package 'sf' was built under R version 4.0.5
```

```
## Linking to GEOS 3.9.1, GDAL 3.4.0, PROJ 8.1.1; sf_use_s2() is TRUE
```

```
library(sp)
library(rgdal)
```

```
## Please note that rgdal will be retired by the end of 2023,
## plan transition to sf/stars/terra functions using GDAL and PROJ
## at your earliest convenience.
##
## rgdal: version: 1.5-27, (SVN revision 1148)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 3.2.1, released 2020/12/29
## Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/gdal
## GDAL binary built with GEOS: TRUE
## Loaded PROJ runtime: Rel. 7.2.1, January 1st, 2021, [PJ_VERSION: 721]
## Path to PROJ shared files: /Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/proj
## PROJ CDN enabled: FALSE
## Linking to sp version:1.4-5
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,
## use options("rgdal_show_exportToProj4_warnings"="none") before loading sp or rgdal.
## Overwritten PROJ_LIB was /Library/Frameworks/R.framework/Versions/4.0/Resources/library/rgdal/proj
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(maps)

## To load our shape files, we are going to use the st_read() function from the sf package.

ARW<-st_read("/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/WBDHU2.shp") # this reads in the
```

```
## Reading layer 'WBDHU2' from data source
##   '/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/WBDHU2.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1 feature and 15 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -106.5998 ymin: 31.20832 xmax: -90.143 ymax: 39.38325
## Geodetic CRS:  NAD83
```

```r
## Now that we have our shapefile loaded, let's plot our area. To give ourselves some reference, we are

states<-st_as_sf(map("state", plot = FALSE, fill = TRUE)) # this uses the "maps" package to make a map
head(states)
```

```
## Simple feature collection with 6 features and 1 field
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -124.3834 ymin: 30.24071 xmax: -71.78015 ymax: 42.04937
## Geodetic CRS:  WGS 84
##            ID                           geom
## 1     alabama MULTIPOLYGON (((-87.46201 3...
## 2     arizona MULTIPOLYGON (((-114.6374 3...
## 3    arkansas MULTIPOLYGON (((-94.05103 3...
## 4  california MULTIPOLYGON (((-120.006 42...
## 5    colorado MULTIPOLYGON (((-102.0552 4...
## 6 connecticut MULTIPOLYGON (((-73.49902 4...
```

```r
sf::sf_use_s2(FALSE)
```

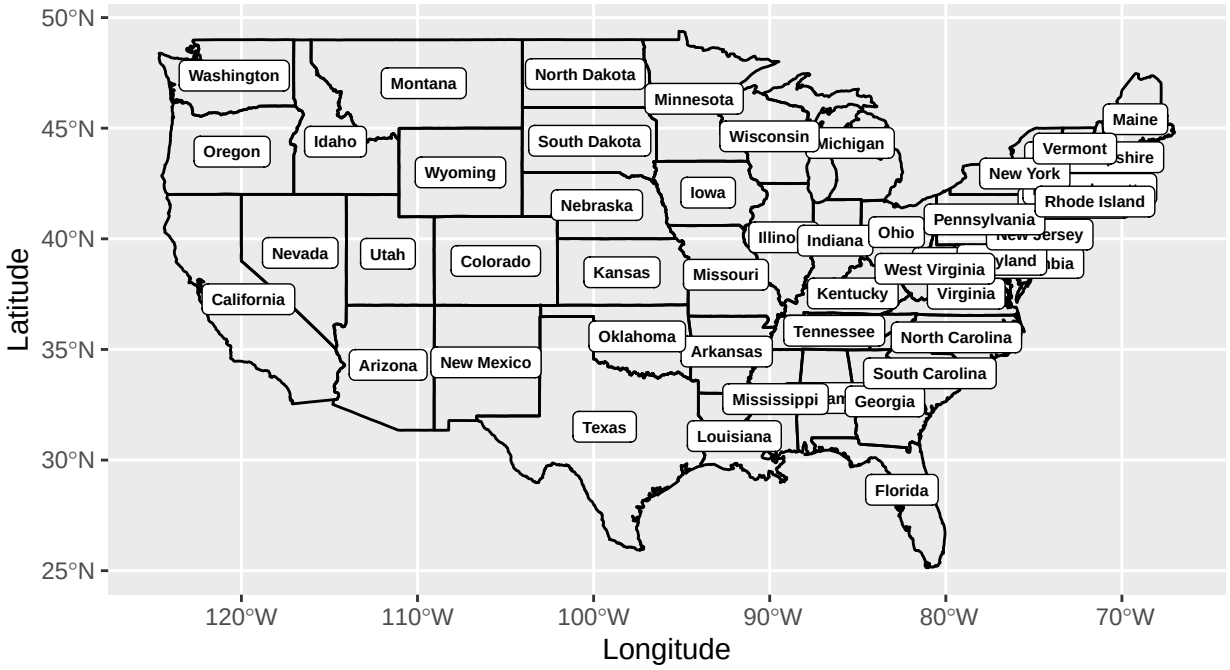
```
## Spherical geometry (s2) switched off
```

```r
states <- cbind(states, st_coordinates(st_centroid(states)))
```

```
## Warning in st_centroid.sf(states): st_centroid assumes attributes are constant
## over geometries of x
```

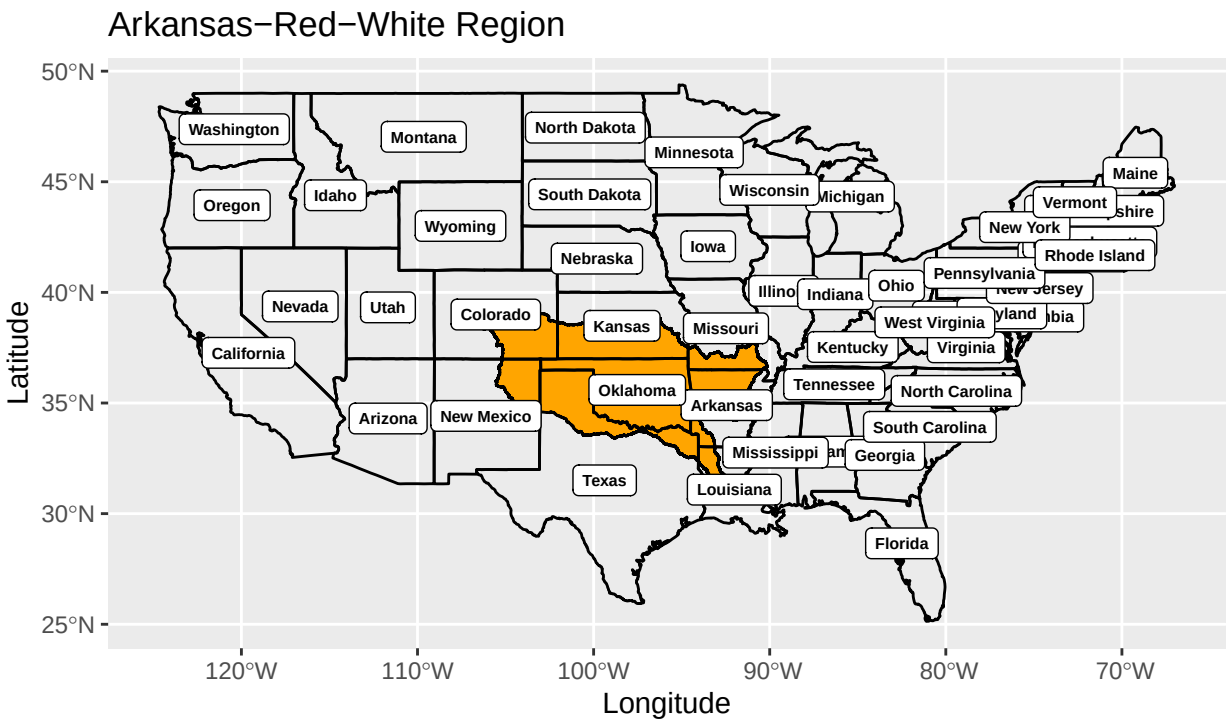
```
## Warning in st_centroid.sfc(st_geometry(x), of_largest_polygon =
## of_largest_polygon): st_centroid does not give correct centroids for longitude/
## latitude data
```

```r
library(tools) # this is part of the ggplot2 package, so we didn't need to install it.
states$ID <- toTitleCase(states$ID)

US<-ggplot() + geom_sf(data = states, color = "black", fill = NA) + geom_label(data = states, aes(X, Y,
US
```




```
## We can now plot our HUC region

region<-ggplot() + geom_sf(data = ARW, color = "black", fill = "orange") + ggtitle("Arkansas-Red-White

## Let's put the two together!

region + geom_sf(data = states, color = "black", fill = NA) + geom_label(data = states, aes(X, Y, label
```

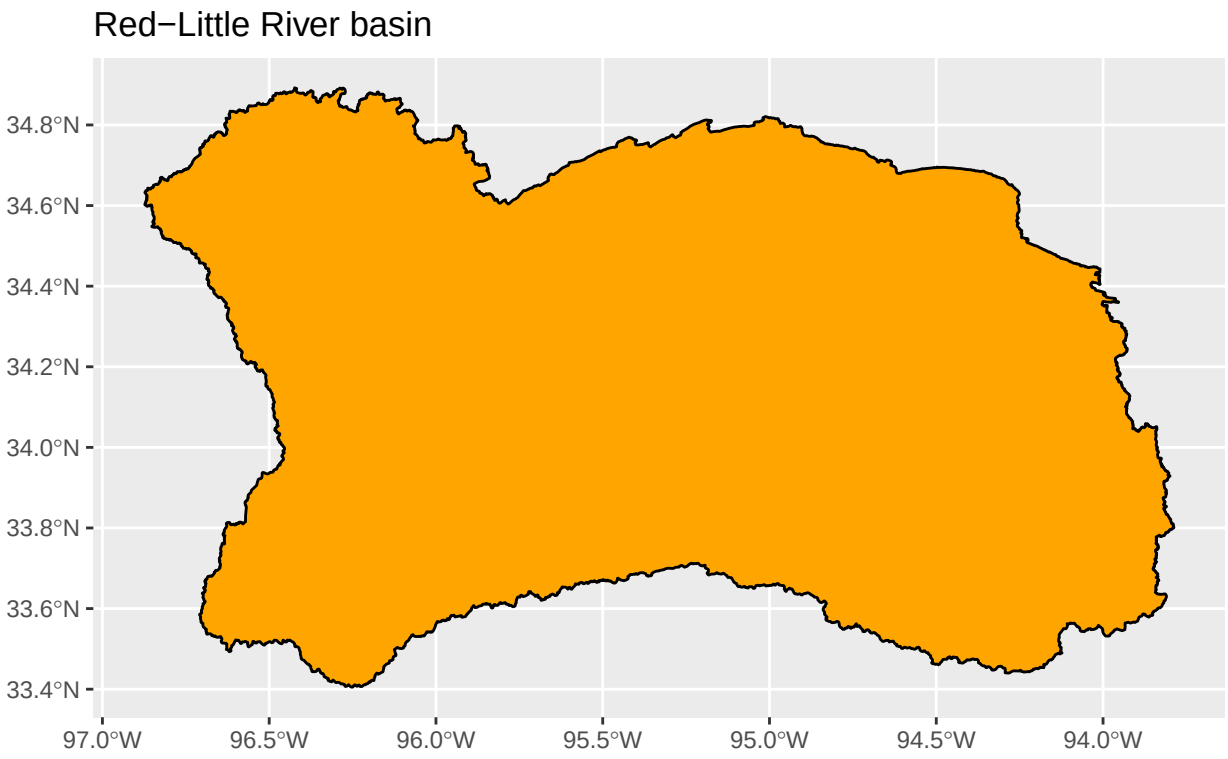
```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```




```
## Ok, now let's map the HUC 6 level.

RedLittle<-st_read("/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/WBDHU6.shp")
```

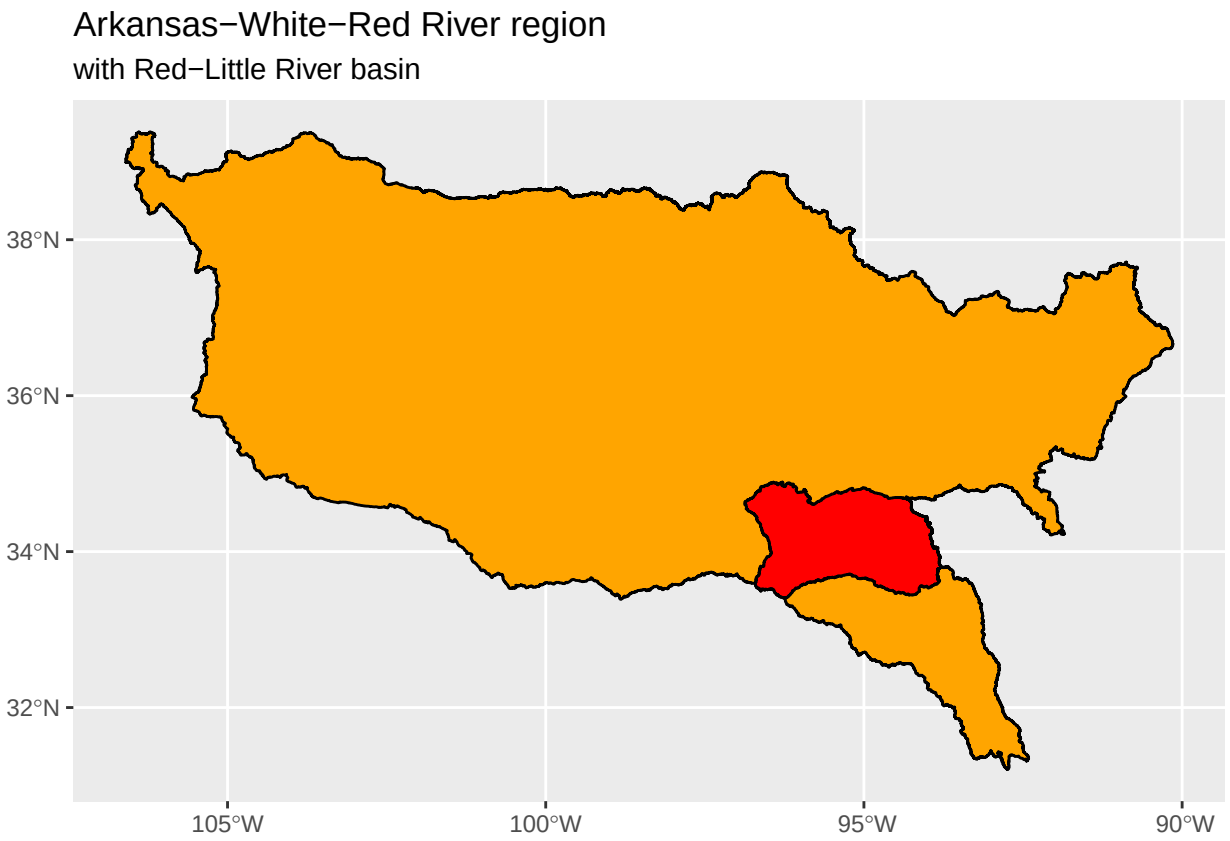
```
## Reading layer 'WBDHU6' from data source
##   '/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/WBDHU6.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1 feature and 15 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -96.87279 ymin: 33.40484 xmax: -93.78743 ymax: 34.89228
## Geodetic CRS:  NAD83
```

```
basin<-ggplot() + geom_sf(data = RedLittle, color = "black", fill = "orange") + ggtitle("Red-Little Riv
basin
```




```
## Let's overlay our HUC 6 layer on top of the HUC 2 layer to show how each HUC is nested within one an

region + geom_sf(data = RedLittle, color = "black", fill = "red") + ggtitle("Arkansas-White-Red River r
```

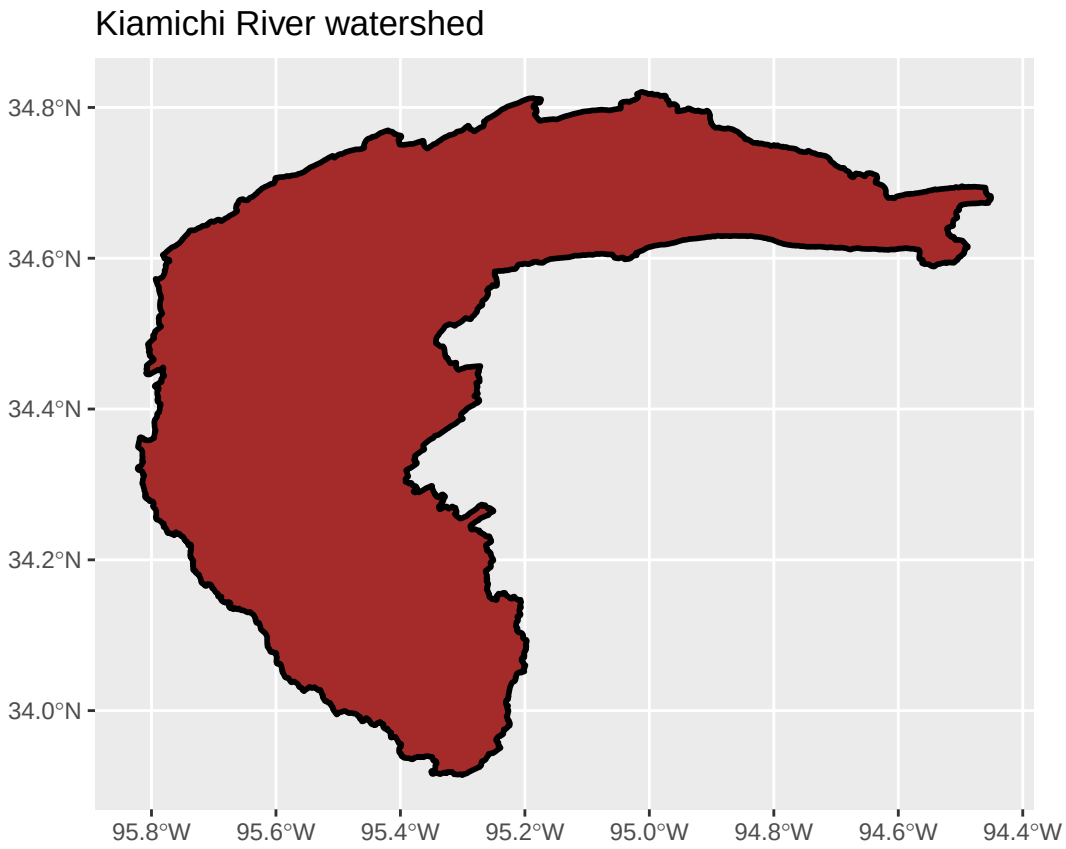
```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```




```
## Finally, let's map the Kiamichi River basin (watershed)

KiamichiHUC8<-st_read("/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/WBDHU8.shp") # this read
```

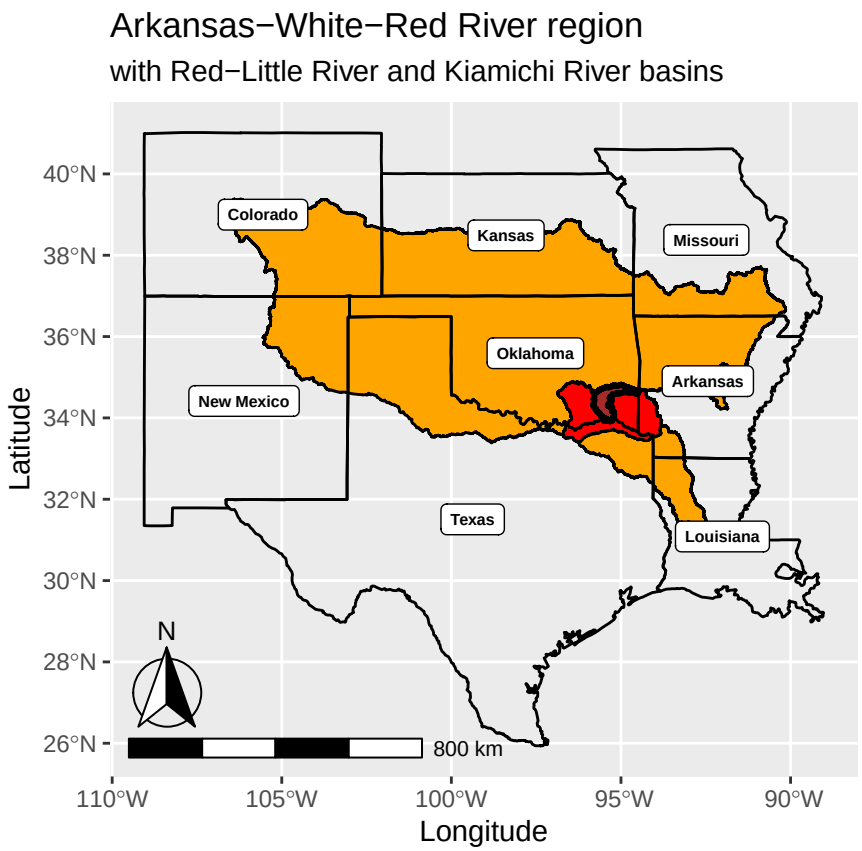
```
## Reading layer 'WBDHU8' from data source
##   '/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/WBDHU8.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1 feature and 15 fields
## Geometry type: POLYGON
## Dimension:     XY
## Bounding box:  xmin: -95.82215 ymin: 33.91447 xmax: -94.45091 ymax: 34.82064
## Geodetic CRS:  NAD83
```

```
watershed<-ggplot() + geom_sf(data = KiamichiHUC8, size = 1, color = "black", fill = "brown") + ggtitle
watershed
```




```
## we can see all of our layers put together, but let's zoom in on just the south-central US.

south.central.us<-states[c(3,5,15,17,24,30,35,42),]

library(ggspatial)

region + geom_sf(data = RedLittle, color = "black", fill = "red") + geom_sf(data = KiamichiHUC8, size =
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one

## Scale on map varies by more than 10%, scale bar may be inaccurate
```

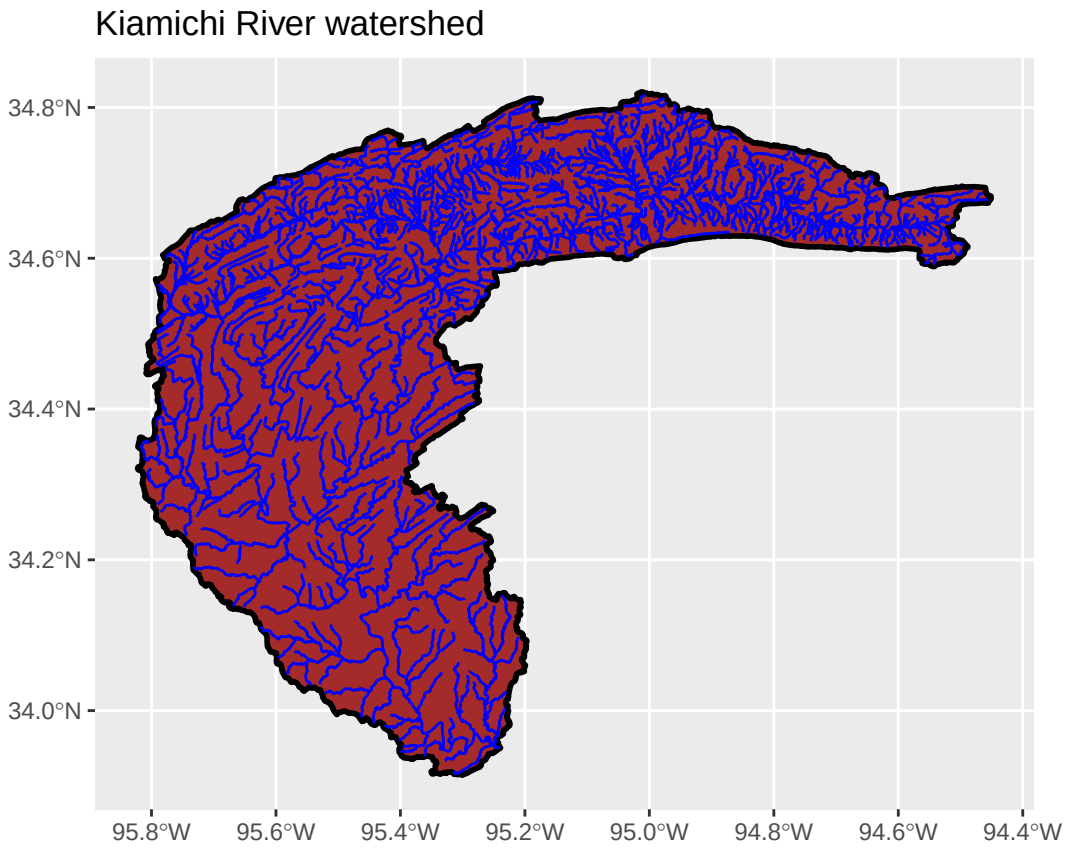



```
## That's enough comparing areas, let's add some data to our HUC 8 layer!

## First, let's read in and plot the NHD Flowlines shapefile. This is basically all of the "streams" co

Kiamichi_NHDflowlines<-st_read("/Users/alexfranzen/unionidae/CODE_workshop/Kiamichi_NHDFlowlines/Kiamic
```

```
## Reading layer ‘Kiamichi_NHDFlowlines’ from data source
##   ‘/Users/alexfranzen/unionidae/CODE_workshop/Kiamichi_NHDFlowlines/Kiamichi_NHDFlowlines.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 2502 features and 149 fields
## Geometry type: LINESTRING
## Dimension:     XY
## Bounding box:  xmin: -95.81634 ymin: 33.91637 xmax: -94.4542 ymax: 34.81648
## Geodetic CRS:  NAD83
```

```
watershed + geom_sf(data = Kiamichi_NHDflowlines, color = "blue") + coord_sf()
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

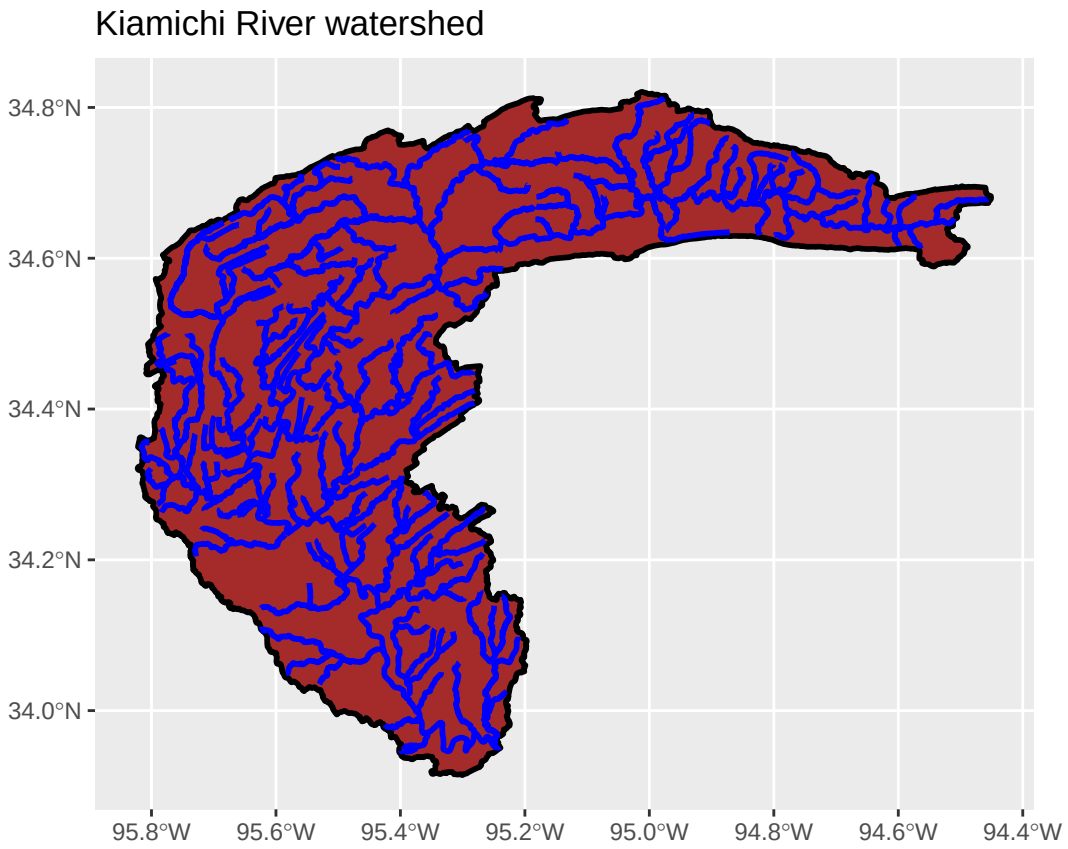



```
## There's a lot going on in that map, so I've simplified the file. Let's read in and map the edited fi

Kiamichi_NHDflowlines_edited<-st_read("/Users/alexfranzen/unionidae/CODE_workshop/Kiamichi_NHDFlowlines
```

```
## Reading layer ‘Kiamichi_NHDFlowlines_edited’ from data source
##   ‘/Users/alexfranzen/unionidae/CODE_workshop/Kiamichi_NHDFlowlines_edited/Kiamichi_NHDFlowlines_edi
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 161 features and 6 fields
## Geometry type: MULTILINESTRING
## Dimension:     XY
## Bounding box:  xmin: -95.81634 ymin: 33.94534 xmax: -94.4542 ymax: 34.81252
## Geodetic CRS:  NAD83
```

```
watershed + geom_sf(data = Kiamichi_NHDflowlines_edited, color = "blue", size = 1) + coord_sf()
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

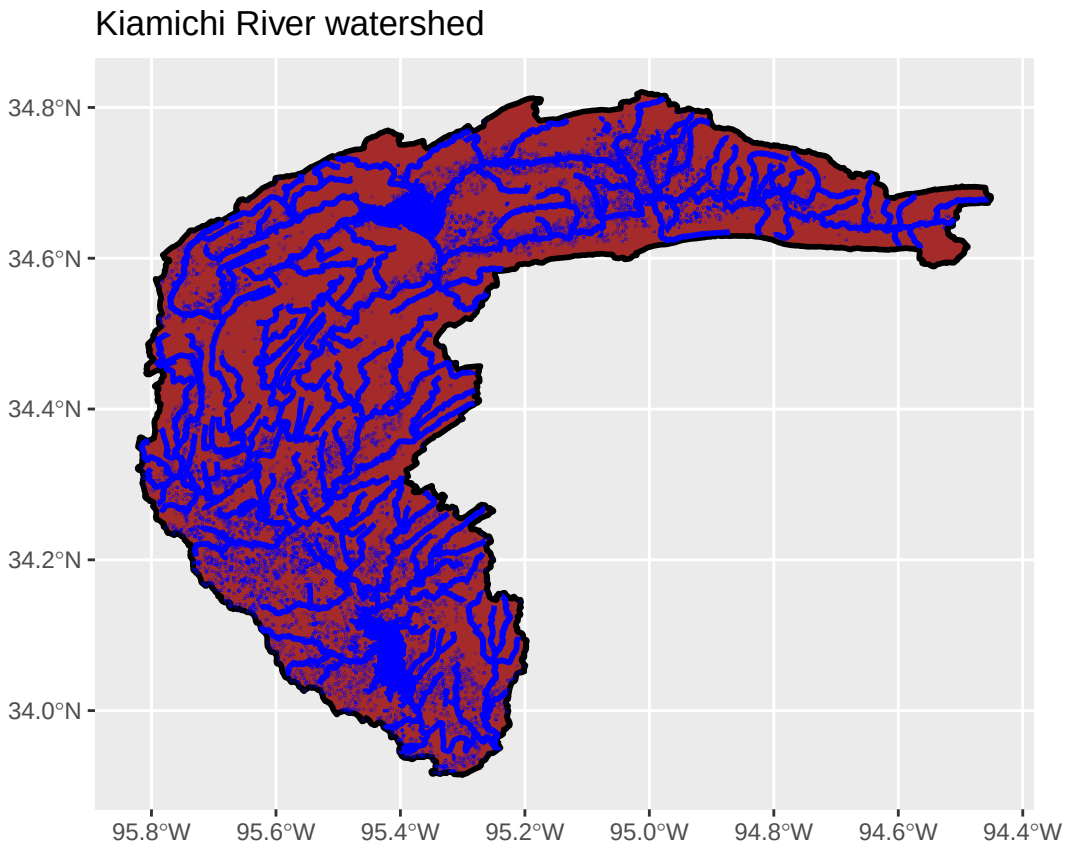



```
## Now, let's add any waterbodies that are not "flowing". On the Kiamichi River, there are two major im

Kiamichi_waterbodies<-st_read("/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/NHDWaterbody.shp
```

```
## Reading layer 'NHDWaterbody' from data source
##   '/Users/alexfranzen/unionidae/CODE_workshop/KiamichiHUCs/NHDWaterbody.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 7272 features and 14 fields
## Geometry type: POLYGON
## Dimension:     XYZ
## Bounding box:  xmin: -95.8148 ymin: 33.91612 xmax: -94.51243 ymax: 34.79488
## z_range:       zmin: 0 zmax: 0
## Geodetic CRS:  NAD83
```

```
## Let's map the HUC 8 area, flowlines, and waterbodies.

watershed + geom_sf(data = Kiamichi_NHDflowlines_edited, color = "blue", size = 1) + geom_sf(data = Kia
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```



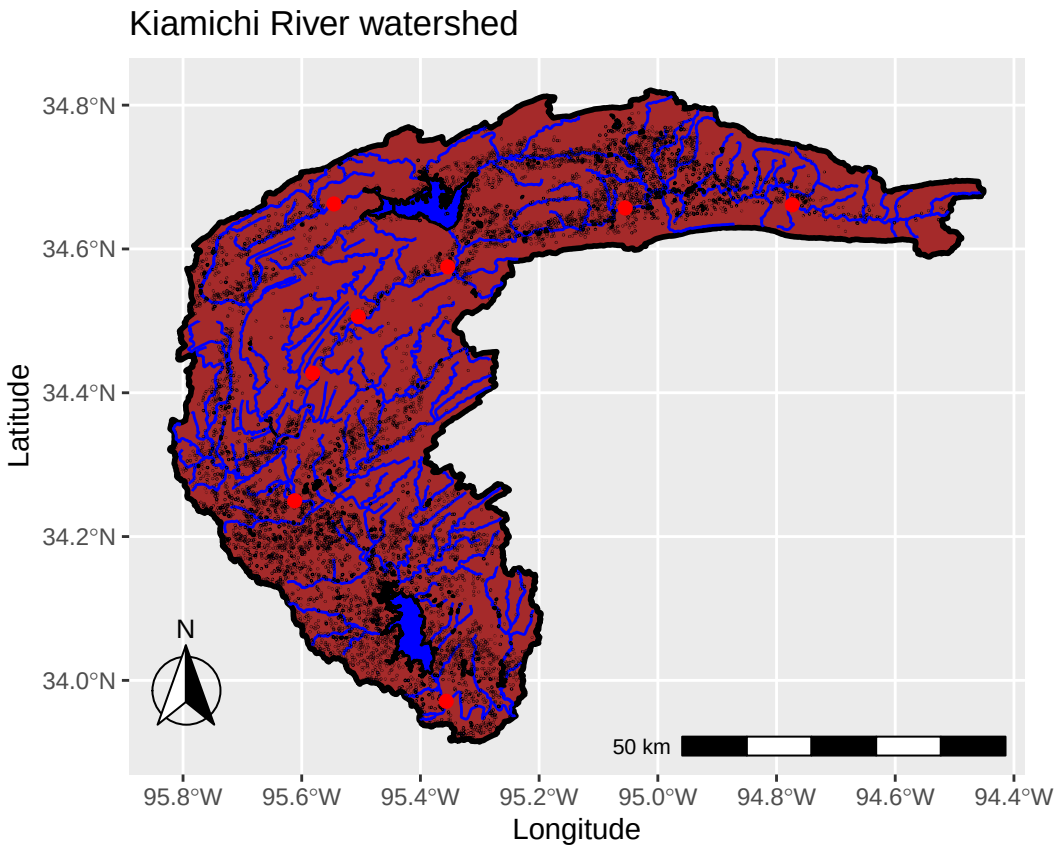

```
## There's a lot of water in this basin!!

## We'll use this data set in the next section, but let's plot some points on to our map. Load the "mus

mussel_beds<-st_read("/Users/alexfranzen/unionidae/CODE_workshop/mussel_beds/mussel_beds.shp")
```

```
## Reading layer ‘mussel_beds’ from data source
##   ‘/Users/alexfranzen/unionidae/CODE_workshop/mussel_beds/mussel_beds.shp’
##   using driver ‘ESRI Shapefile’
## Simple feature collection with 8 features and 3 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -10643460 ymin: 4024903 xmax: -10550140 ymax: 4118124
## Projected CRS: WGS 84 / Pseudo-Mercator
```

```
## Let's plot the whole thing!

watershed + geom_sf(data = Kiamichi_NHDflowlines_edited, color = "blue") + geom_sf(data = Kiamichi_wate
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```

```
## Warning: Ignoring unknown parameters: KiamichiHUCs
```




```
## What a pretty map!
```

Part 2: Calculating river distance between a group of points

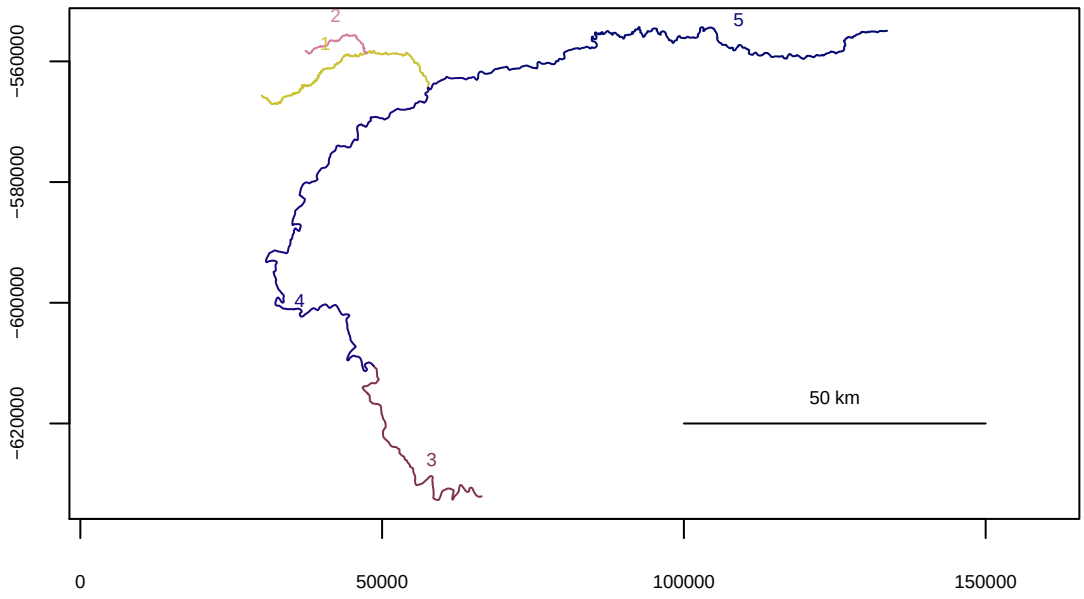
```
### Now that we have done some basic map making, let's actually run some spatial analyses. For this nex

### If you need help fixing an issue, you might be able to find it in this documentation: https://cran.

### We'll be using some of the package we loaded before, but for this part we will need to load an addi

## remove the comments from the next line of code
#install.packages('riverdist')
library(riverdist)
```

```
## Warning: package 'riverdist' was built under R version 4.0.5
```

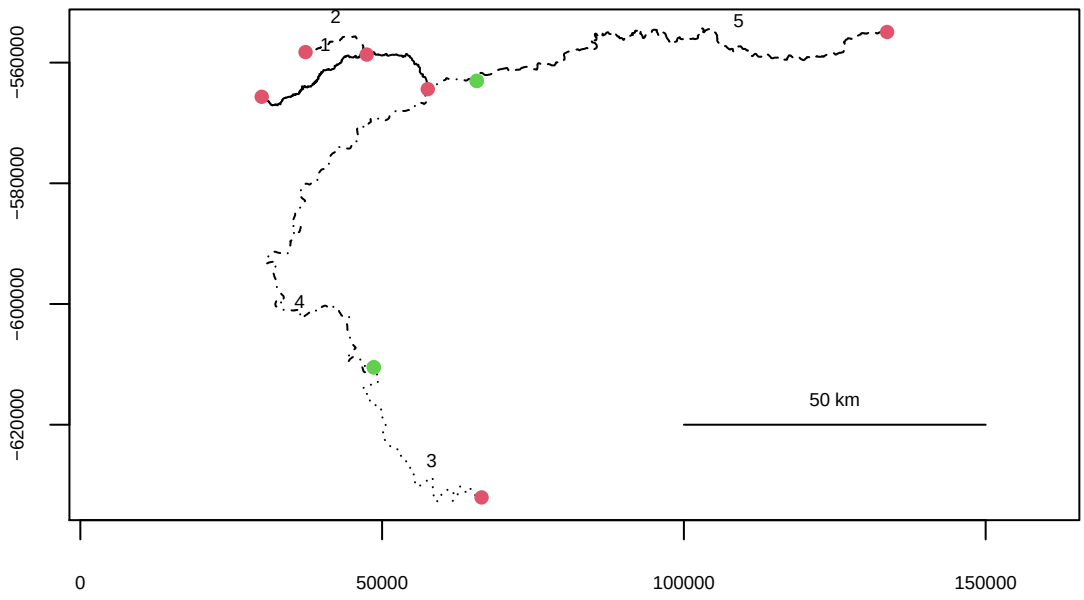
```
## First we need to make our network of streams. For simplicity, I've made a shapefile of only the Kiam

Kiamichi<-line2network(path="/Users/alexfranzen/unionidae/CODE_workshop/Kiamichi_River/Kiamichi_River.sh
```

```
##
##  Units: m
```

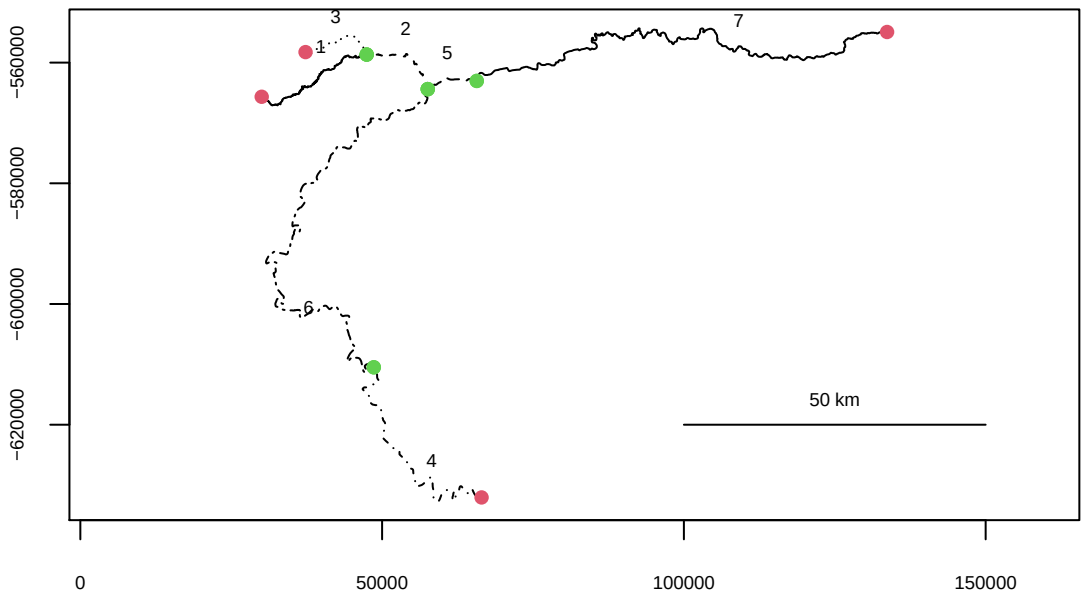
```
plot(Kiamichi) # let's plot our network.
```




```
topologydots(rivers = Kiamichi) # For our route and distance calculations to work, the topologies must
```




```
## Since connectedness is not detected for the Jack Fork Creek tributary, topologydots() shows the endp

Kiamichi <- splitsegments(rivers=Kiamichi) # we can use the splitsegments function to fix our Kiamichi

## Note: any point data already using the input river network must be re-transformed to river coordinat

topologydots(rivers = Kiamichi)
```

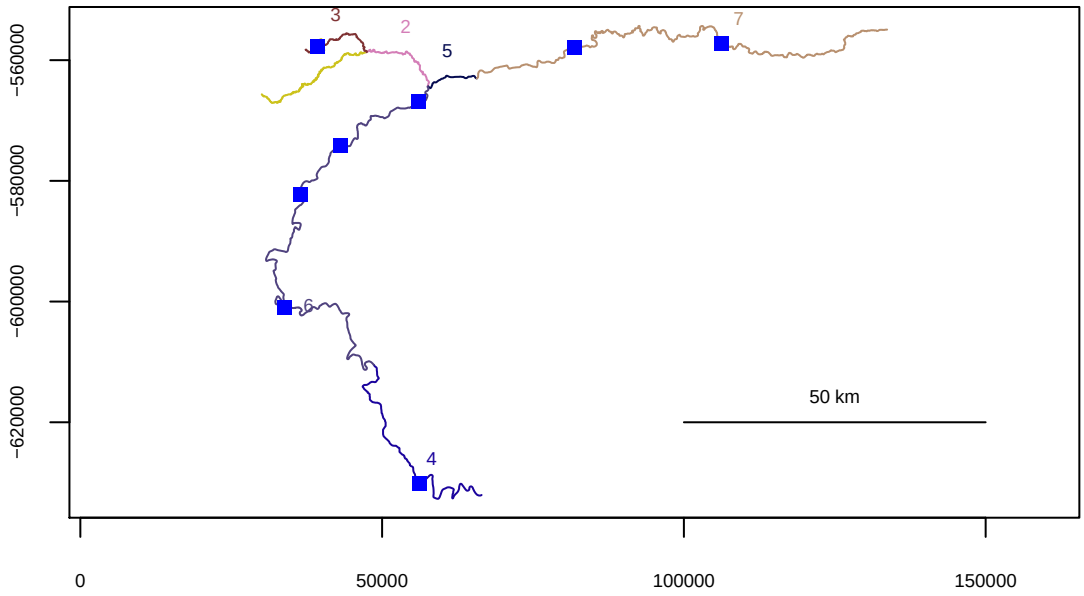



```
## Now we are going to work with our mussel beds data. In my lab, we study the ecology and conservation

## The US Interior Highlands mussel fauna has high species richness (S = 63) and endemicity (14%; Haag

mussel_beds<-pointshp2segvert(path = "/Users/alexfranzen/unionidae/CODE_workshop/mussel_beds/mussel_beds
```

```
## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS =
## dumpSRS, : Discarded ellps WGS 84 in Proj4 definition: +proj=merc +a=6378137
## +b=6378137 +lat_ts=0 +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null
## +wktext +no_defs

## Warning in OGRSpatialRef(dsn, layer, morphFromESRI = morphFromESRI, dumpSRS =
## dumpSRS, : Discarded datum WGS_1984 in Proj4 definition: +proj=merc +a=6378137
## +b=6378137 +lat_ts=0 +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m +nadgrids=@null
## +wktext +no_defs

## Warning in showSRID(wkt2, "PROJ"): Discarded ellps WGS 84 in Proj4 definition:
## +proj=merc +a=6378137 +b=6378137 +lat_ts=0 +lon_0=0 +x_0=0 +y_0=0 +k=1 +units=m
## +nadgrids=@null +wktext +no_defs +type=crs

## Warning in showSRID(wkt2, "PROJ"): Discarded datum World Geodetic System 1984 in
## Proj4 definition

## OGR data source with driver: ESRI Shapefile
```

```
## Source: "/Users/alexfranzen/unionidae/CODE_workshop/mussel_beds/mussel_beds.shp", layer: "mussel_beds
## with 8 features
## It has 3 fields

## Warning in sp::proj4string(shp): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html

## Warning in sp::proj4string(rivers$sp): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html

##
##  Point projection detected as different from river network.  Re-projecting points before snapping to

## Warning in sp::proj4string(rivers$sp): CRS object has comment, which is lost in output; in tests, see
## https://cran.r-project.org/web/packages/sp/vignettes/CRS_warnings.html
```

```r
## Let's plot our points on our river network.

plot(Kiamichi)
points(mussel_beds$lat_n, mussel_beds$long_w, pch=16, col="red")
riverpoints(seg = mussel_beds$seg, vert = mussel_beds$vert, rivers = Kiamichi, pch = 15, col = "blue")
```



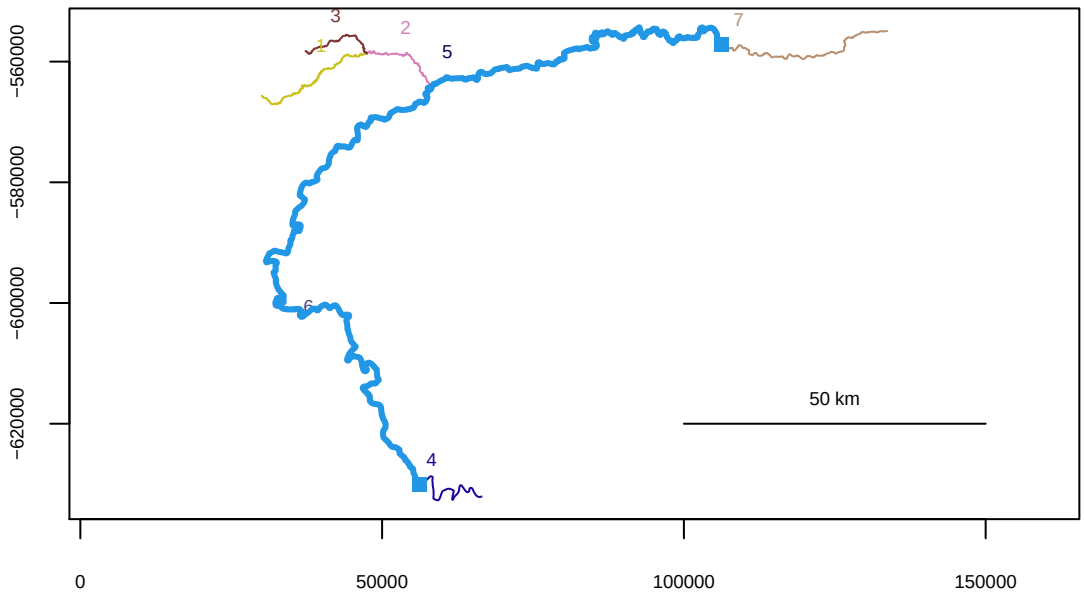
```
## This function is used to verify that there is a route in our network between a set of points.

detectroute(start = 3, end = 4, rivers = Kiamichi)
```

```
## [1] 3 2 6 4
```

```
## We can see there is a viable route between segments 3 and 4.

# Let's take a quick look at our data so we can enter specific information to calculate distance.
print(mussel_beds)
```

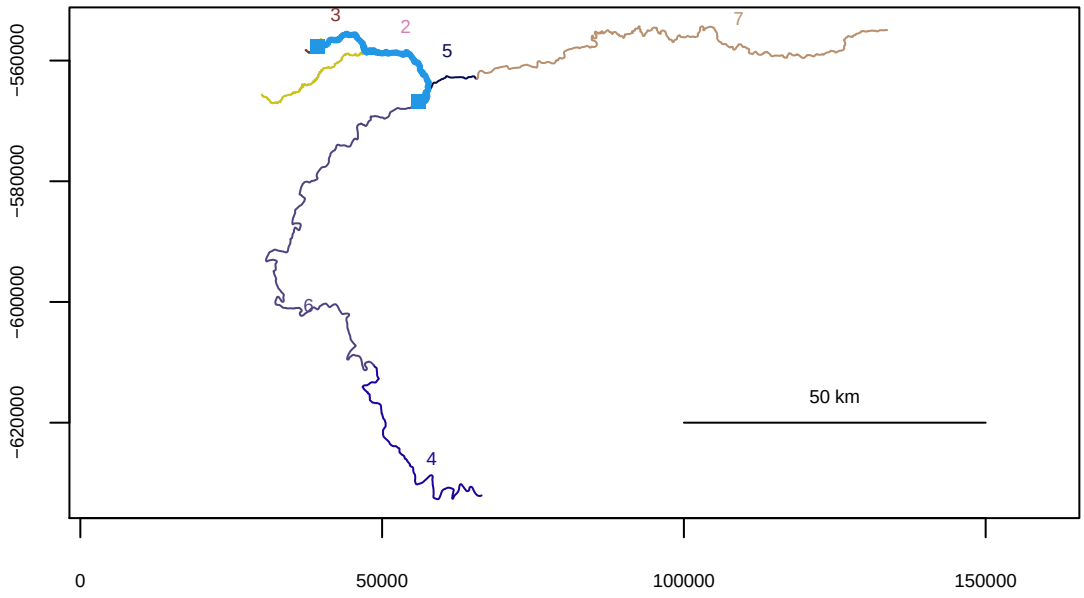
```
##   seg vert   snapdist    name    lat_n    long_w
## 1   3   77   5.798554      JF 34.66260 -95.54569
## 2   7  939  36.247346    Muse 34.66137 -94.77353
## 3   7 1593  32.175109      K2 34.65696 -95.05496
## 4   6   31  57.972980  Tammys 34.57511 -95.35410
## 5   6  135  43.701615      KS 34.50582 -95.50471
## 6   6  306 132.139594      K7 34.42736 -95.58160
## 7   6  566   8.662969 Antlers 34.24978 -95.61181
## 8   4  286  60.565834     K12 33.97096 -95.35661
```

```
## Now lets calculate distance between two points. I chose the "Muse" site and the "K12" site. To calcu

riverdistance(startseg = 7, startvert = 939, endseg = 4, endvert = 286 , rivers = Kiamichi, map = TRUE)
```



```
## [1] 199914.6
```

```
## Nice! We can see that the distance between Muse and K12 is 199914.6 meters! We can also get a visual

riverdistance(startseg = 3, startvert = 77, endseg = 6, endvert = 31 , rivers = Kiamichi, map = TRUE)
```




```
## [1] 33975.56
```

```
## The distance from JF to Tammys is 33975.56 meters!

## Instead of doing each calculation one at a time, we can calculate a distance matrix. This is a pairw

distmatrix<-riverdistancemat(seg = mussel_beds$seg, vert = mussel_beds$vert, rivers = Kiamichi) # calcu
siteID<-c("JF", "Muse", "K2", "Tammys", "KS", "K7", "Antlers", "K12") # list of site names
colnames(distmatrix)<-siteID # add column names to our matrix
rownames(distmatrix)<-siteID # add row names to our matrix
print(distmatrix)
```

```
##                JF      Muse        K2    Tammys        KS        K7   Antlers
## JF           0.00 103349.55  60523.97  33975.56  53060.67  65436.27  97600.85
## Muse    103349.55      0.00  42825.59  77876.65  96961.76 109337.36 141501.94
## K2       60523.97  42825.59      0.00  35051.06  54136.17  66511.78  98676.35
## Tammys   33975.56  77876.65  35051.06      0.00  19085.11  31460.71  63625.29
## KS       53060.67  96961.76  54136.17  19085.11      0.00  12375.61  44540.18
```

```
## K7       65436.27 109337.36  66511.78  31460.71  12375.61      0.00  32164.58
## Antlers  97600.85 141501.94  98676.35  63625.29  44540.18  32164.58      0.00
## K12     156013.48 199914.57 157088.98 122037.92 102952.81  90577.21  58412.63
##               K12
## JF      156013.48
## Muse    199914.57
## K2      157088.98
## Tammys  122037.92
## KS      102952.81
## K7       90577.21
## Antlers  58412.63
## K12          0.00
```

```
## Let's quickly convert our calculations from meters to kilometers

distmatrix<-t(distmatrix/1000)
print(distmatrix)
```

```
##                JF      Muse        K2    Tammys        KS        K7   Antlers
## JF        0.00000 103.34955  60.52397  33.97556  53.06067  65.43627  97.60085
## Muse    103.34955   0.00000  42.82559  77.87665  96.96176 109.33736 141.50194
## K2       60.52397  42.82559   0.00000  35.05106  54.13617  66.51178  98.67635
## Tammys   33.97556  77.87665  35.05106   0.00000  19.08511  31.46071  63.62529
## KS       53.06067  96.96176  54.13617  19.08511   0.00000  12.37561  44.54018
## K7       65.43627 109.33736  66.51178  31.46071  12.37561   0.00000  32.16458
## Antlers  97.60085 141.50194  98.67635  63.62529  44.54018  32.16458   0.00000
## K12     156.01348 199.91457 157.08898 122.03792 102.95281  90.57721  58.41263
##               K12
## JF      156.01348
## Muse    199.91457
## K2      157.08898
## Tammys  122.03792
## KS      102.95281
## K7       90.57721
## Antlers  58.41263
## K12       0.00000
```

```
## We can also convert km to mi.

distmatrix<-t(distmatrix*0.62137)
print(distmatrix)
```

```
##                JF      Muse       K2   Tammys       KS       K7  Antlers
## JF       0.00000  64.21831 37.60778 21.11139 32.97031 40.66014 60.64624
## Muse    64.21831   0.00000 26.61054 48.39021 60.24913 67.93896 87.92506
## K2      37.60778  26.61054  0.00000 21.77968 33.63859 41.32842 61.31453
## Tammys  21.11139  48.39021 21.77968  0.00000 11.85891 19.54874 39.53485
## KS      32.97031  60.24913 33.63859 11.85891  0.00000  7.68983 27.67593
## K7      40.66014  67.93896 41.32842 19.54874  7.68983  0.00000 19.98610
## Antlers 60.64624  87.92506 61.31453 39.53485 27.67593 19.98610  0.00000
## K12     96.94210 124.22092 97.61038 75.83070 63.97179 56.28196 36.29586
##               K12
## JF       96.94210
```

```
## Muse    124.22092
## K2       97.61038
## Tammys   75.83070
## KS       63.97179
## K7       56.28196
## Antlers  36.29586
## K12       0.00000
```

```
## Sweet, now we know how far we would need to travel the river to get between our mussel beds.
```

Part 3: Mantel Test

```
### For this final part, we are going to implement a Mantel test on our river distance matrix and some

## You will need to install the following packages:

#install.packages('hierfstat')
#install.packages('ade4')

library(hierfstat)
library(ade4)
library(ggplot2) # load ggplot again, just in case.

## First, we need to read in the genetic data for a species of mussel, *Fusconaia flava*. In this examp

Kiamichi_flava_FST_pairwise<-read.csv("Kiamichi_flava_pairwise_Fst.csv", header=FALSE) # reads in the c
colnames(Kiamichi_flava_FST_pairwise)<-siteID # add column names to our matrix
rownames(Kiamichi_flava_FST_pairwise)<-siteID # add row names to our matrix
print(Kiamichi_flava_FST_pairwise)
```

```
##            JF  Muse    K2 Tammys    KS    K7 Antlers K12
## JF      0.000    NA    NA     NA    NA    NA       NA  NA
## Muse    0.100 0.000    NA     NA    NA    NA       NA  NA
## K2      0.043 0.022 0.000     NA    NA    NA       NA  NA
## Tammys  0.060 0.031 0.014  0.000    NA    NA       NA  NA
## KS      0.072 0.026 0.020  0.019 0.000    NA       NA  NA
## K7      0.074 0.038 0.032  0.021 0.090 0.000       NA  NA
## Antlers 0.085 0.057 0.042  0.027 0.023 0.019       NA  NA
## K12     0.130 0.098 0.064  0.043 0.042 0.037    0.043   0
```

```
## time for the Mantel test

## Here, we run a Mantel test with 5000 permutations for randomization test on Fst values.

distmatrix<-as.dist(distmatrix) # we need to coerce our matrix to a dist vector
print(distmatrix)
```

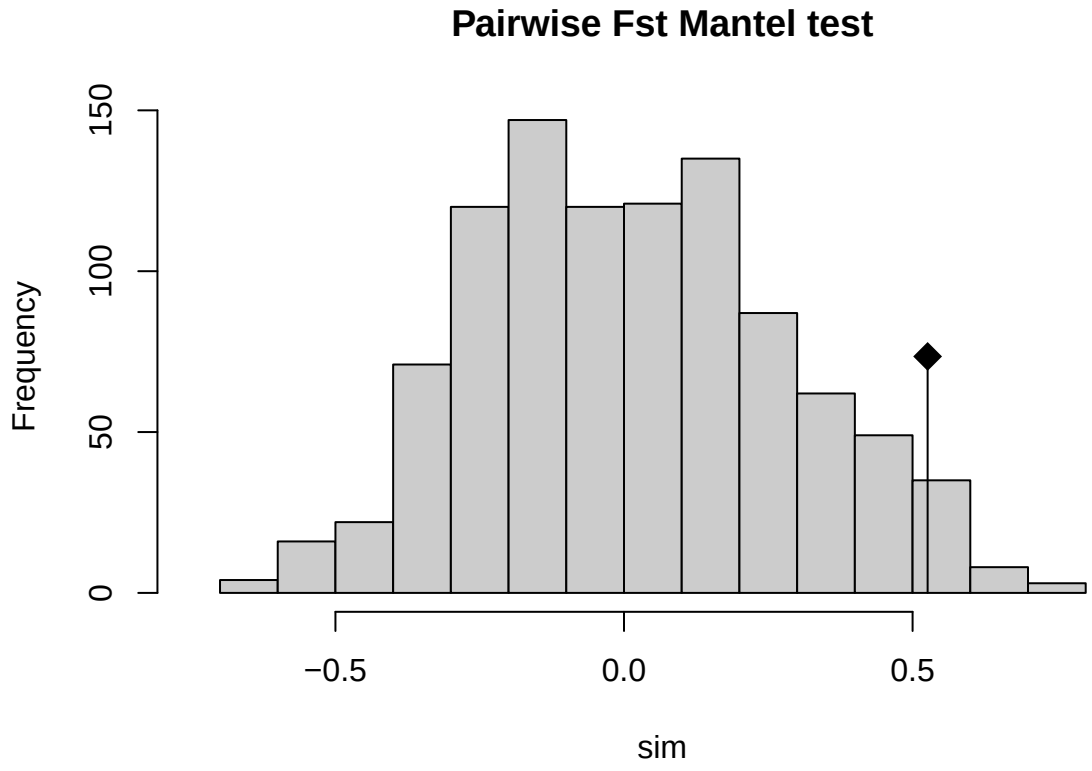
```
##                JF      Muse        K2    Tammys        KS        K7   Antlers
## Muse     64.21831
## K2       37.60778  26.61054
```

```
## Tammys   21.11139  48.39021  21.77968
## KS       32.97031  60.24913  33.63859  11.85891
## K7       40.66014  67.93896  41.32842  19.54874   7.68983
## Antlers  60.64624  87.92506  61.31453  39.53485  27.67593  19.98610
## K12      96.94210 124.22092  97.61038  75.83070  63.97179  56.28196  36.29586
```

```
Kiamichi_flava_FST_pairwise<-as.dist(Kiamichi_flava_FST_pairwise) # we need to coerce our matrix to a d
print(Kiamichi_flava_FST_pairwise)
```

```
##            JF  Muse    K2 Tammys    KS    K7 Antlers
## Muse    0.100
## K2      0.043 0.022
## Tammys  0.060 0.031 0.014
## KS      0.072 0.026 0.020  0.019
## K7      0.074 0.038 0.032  0.021 0.090
## Antlers 0.085 0.057 0.042  0.027 0.023 0.019
## K12     0.130 0.098 0.064  0.043 0.042 0.037   0.043
```

```
isobydist<-mantel.randtest(Kiamichi_flava_FST_pairwise, distmatrix, nrepet = 1000) # testing for isolat
plot(isobydist, main = "Pairwise Fst Mantel test")
```




```
isobydist
```

```
## Monte-Carlo test
```

```
## Call: mantel.randtest(m1 = Kiamichi_flava_FST_pairwise, m2 = distmatrix,
##     nrepet = 1000)
##
## Observation: 0.5260979
##
## Based on 1000 replicates
## Simulated p-value: 0.03496503
## Alternative hypothesis: greater
##
##     Std.Obs Expectation    Variance
##  1.91422402  0.01249181  0.07199044
```

```
## Looks like we have isolation by distance occurring. That means that the further populations are from
```