Lab session 1

Aim of this practical session:

In this first practical we are going

- To explore graphically some environmental data sets.
- To explore different estimation techniques for censored data (Section 1).
- Design a monitoring network using the GRTS algorithm (Section 2).

1 Part 1: Limits of detection

Data reported at a limit of detection represent a common problem in environmental studies. For this problem, measurements on ammonia from a stretch of river will be studied, with 31% censored data. The data are available in the file `SiteSoar.csv`.

The variables in the dataset are as follows:

Variable	Meaning
`year`	Year of observation
`month`	Month of observation
`day`	Day of observation (within month)
`doy`	Day of observation (from start of year)
`Ammonia`	Ammonia level
`censored`	Censoring indicator (`FALSE`: no censoring; `TRUE`: censored at limit of detection)

Here we will consider three different methods of dealing with censored data (i.e., three strategies for replacement). These are:

- Kaplan-Meier estimates,
- Maximum Likelihood Estimators (MLEs), and
- Regression on Order Statistics (ROS). Note: ROS computes a linear regression for data (or their logs) versus their normal scores (from a Normal probability plot).

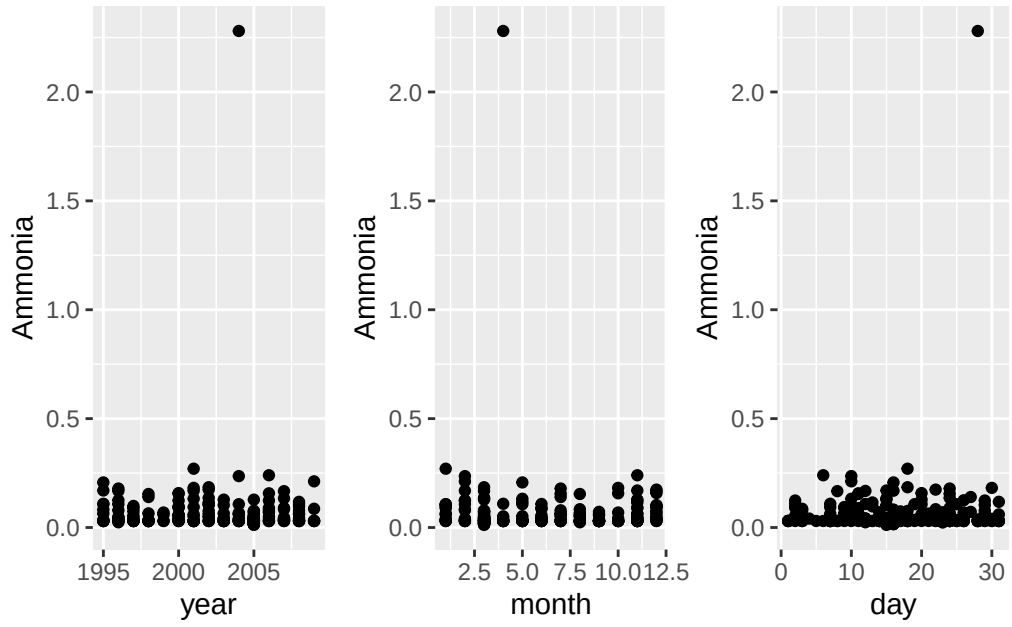
```
# Load required R packages:

library(NADA)       # for analyzing censored observations
library(ggplot2)    # For visualizing our data
library(patchwork)  # For plotting multiple ggplot objects together

# Read in data (making sure to set the working directory to the
# appropriate location):
Soar <- read.csv("datasets/SiteSoar.csv", header = TRUE)

# Examine structure of the dataset:
str(Soar)
```

```
'data.frame':   224 obs. of  6 variables:
 $ year    : int  1995 1995 1995 1996 2004 2003 2002 2002 2002 2000 ...
 $ month   : int  11 9 4 3 3 7 12 11 7 11 ...
 $ day     : int  21 30 24 20 24 24 22 13 25 2 ...
 $ doy     : int  326 274 115 80 84 206 357 318 207 307 ...
 $ Ammonia : num  0.064 0.03 0.03 0.049 0.03 0.03 0.174 0.115 0.03 0.091 ...
 $ censored: logi  FALSE TRUE TRUE FALSE TRUE TRUE ...
```

1.1 Visualizing our data

Lets create some exploratory plots to visualize the relationship between Ammonia and time.

```
ggplot(data = Soar,aes(y = Ammonia, x = year))+geom_point()+
  ggplot(data = Soar,aes(y = Ammonia, x = month))+geom_point()+
  ggplot(data = Soar,aes(y = Ammonia, x = day))+geom_point()
```



Question

What do you notice about the relationship between Ammonia and time? Do you notice any patterns or *unusual* observations?

See Solution

We would like to focus here on the relationship between Ammonia and time. The key thing that we notice is the single point that has a high value of Ammonia. This seems like it could be an error rather than a true value (possibly a decimal point in the wrong place). We can also see this by looking at the dataset — type `View(Soar)`.

We could justify removing the outlier, since its value is an order of magnitude larger than the other values, so is probably an error. We could alternatively justify not removing the outlier, since we would need to first consult a subject-matter expert before concluding that it was an unlikely value of Ammonia. (It's up to you which you choose, as long as you justify it. I might remove this value.)

There are no obvious trends or seasonal patterns in the data, from these plots.

Task 1

By looking at the exploratory plots there seems to be an outlier given by an Ammonia value that is larger than 2. Suppose we want to remove this from any further analysis (provided we have reasonable justification to do it). Remove the outlier from the dataset, you can use the `filter()` function from the `dplyr` package to achieve this.
Take hint
We can use the `filter` function from `dplyr` to subset our data according to a logical statement. Load the `dplyr` library and type `?filter` for further details.
Click here to see the solution

```
library(dplyr)

Soar <- Soar %>% filter(Ammonia <2)
```

Lets look into some further statistics using the `pctCen` and `censummary` functions from `NADA`:

```
# Further summary statistics:
pctCen(Soar$Ammonia, Soar$censored)     ## percent of censored data
```

```
[1] 31.39013
```

```
censummary(Soar$Ammonia, Soar$censored) ## like summary cmd but for censored data
```

```
all:
        n     n.cen   pct.cen       min       max
223.00000  70.00000  31.39013   0.01200   0.27000

limits:
  limit   n uncen   pexceed
1  0.00   0    22 1.0000000
2  0.03  70   131 0.5874439
```

What does this tell us? You can ignore the limit of `0.00`, since the function adds this by default if there is only one positive limit in the dataset — you can see that `n` takes the value 0 here, meaning that there are no data points censored at 0.

In the first line, `uncen` tells us that there are 22 data points that have values between 0 and the LoD of 0.03, so this emphasises that just because we have a LoD in the data, that doesn't mean that all data points will be censored here, since different instruments (that have different LoDs) may have been used to generate the same dataset.

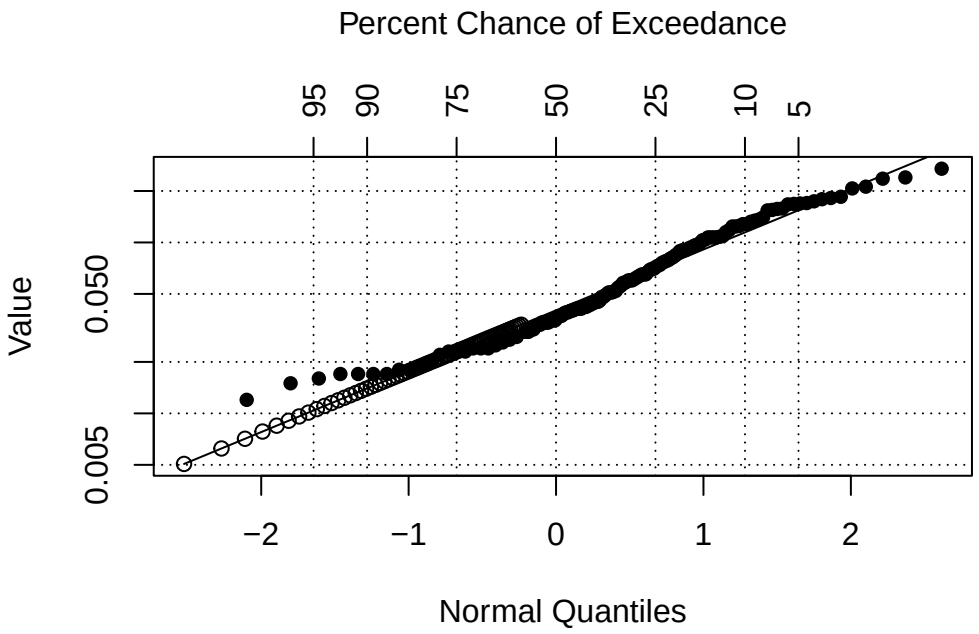
In the second line, we see that there are 70 data points censored at the LoD of 0.03. There are 131 data points that take values above 0.03. `pexceed` tells us that the probability of exceeding the LoD is $131/(131+70+22) = 0.589$.

1.2 Dealing with censored observations

1.2.1 ROS

First, we will implement Regression on Order Statistics (ROS) using the `cenros` function

```
## 1. ROS:
ROS <- cenros(Soar$Ammonia, Soar$censored)
plot(ROS, plot.censored = TRUE) ## plots the modelled censored
```



Interpretation of plot: Here, do the filled black circles generally follow a straight line? Yes, so the model seems reasonable to use here.

The imputed values are the empty black circles, and these seem reasonable — we have already seen that the model seems to be appropriate, so this should be the case.

```
summary(ROS) ## more info about the ROS regression
```

```
Call:
lm(formula = obs.transformed ~ pp.nq, na.action = na.action)

Residuals:
     Min       1Q   Median       3Q      Max
-0.23643 -0.09932 -0.00931  0.08775  0.51484

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.21742    0.01090 -295.29   <2e-16 ***
pp.nq        0.81982    0.01175   69.75   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1194 on 151 degrees of freedom
Multiple R-squared:  0.9699,    Adjusted R-squared:  0.9697
F-statistic:  4866 on 1 and 151 DF,  p-value: < 2.2e-16
```

pp.nq should be statistically significant — it is ($p < 0.05$).

```
print(ROS)    ## prints a simple summary of the ROS model.
```

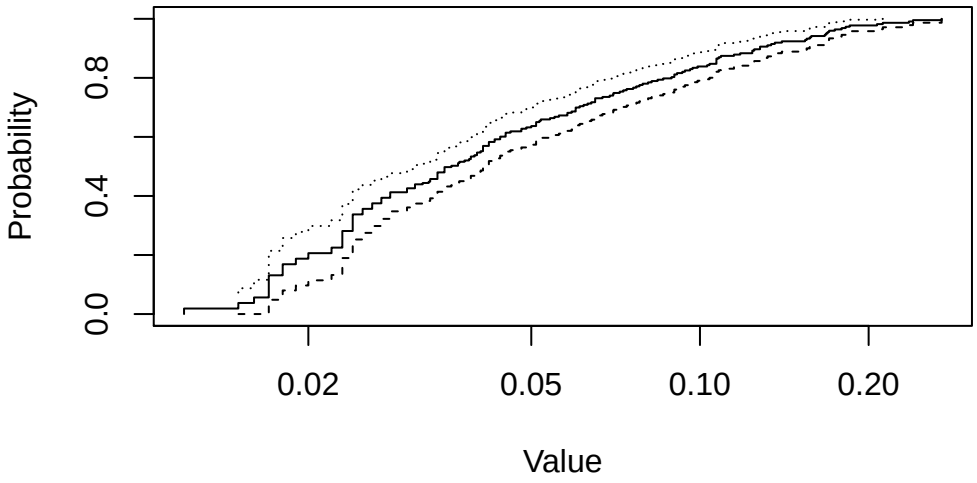
```
           n        n.cen       median         mean           sd
223.00000000  70.00000000   0.03600000   0.05577297   0.04960383
```

This is what we really care about here — this represents the summary of the censored data, with mean 0.056 and standard deviation 0.049. We'll use this to generate our imputed values later.

1.2.2 Kaplan-Meier

Now lets look into the Kaplan-Meier

```
## 2. Kaplan-Meier:
KM <- cenfit(Soar$Ammonia, Soar$censored)  ## constructs a Kaplan-Meier model
plot(KM)    ## survival function plot
```



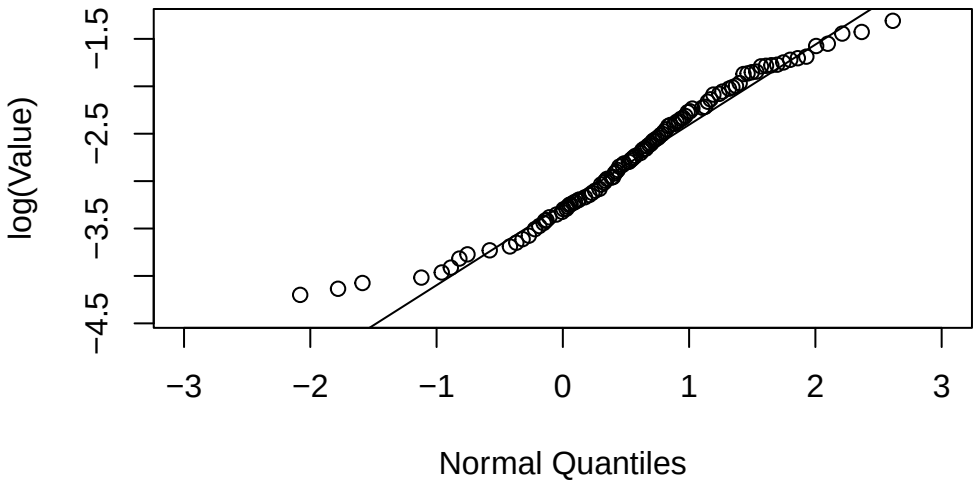
On the x-axis we have the Concentration values and on the y-axis we have the proportion of observations $\leq$ to each concentration. Don't worry too much about the interpretation of this plot. It tells us about the probability of the data being at most a certain value (using the information that we have from the censored and uncensored data). What we are actually interested is in the following output:

```
print(KM)
```

```
           n        n.cen       median         mean           sd
223.00000000  70.00000000   0.03600000   0.05590147   0.04967880
```

1.2.3 MLE

```
## 3. MLE
MLE <- cenmle(Soar$Ammonia, Soar$censored) ## constructs a Maximum Likelihood model
plot(MLE)
```



Question

What can you tell about the plot above?
See Solution
Most of the points follow the fitted line, so this model appears appropriate. (We should not worry too much about the few points that lie far from the line — this doesn't mean that our model is not appropriate.)

Lets look into some summaries for the model

```
summary(MLE)
```

```
             Value Std. Error      z       p
(Intercept) -3.252     0.0612 -53.16 0.00000
Log(scale)  -0.167     0.0604  -2.77 0.00559

Scale = 0.846

Log Normal distribution
Loglik(model)= 189.9   Loglik(intercept only)= 189.9
Loglik-r:  0

Number of Newton-Raphson Iterations: 5
n = 223
```

```
print(MLE)
```

```
           n        n.cen       median         mean           sd
223.00000000  70.00000000   0.03871202   0.05536429   0.05660577
```

1.3 Imputation

First, lets compare the estimated mean and sd of each method. We can use the `censtats` function to achieve this:

```
censtats(Soar$Ammonia, Soar$censored)
```

```
        n     n.cen   pct.cen
223.00000  70.00000  31.39013

        median       mean         sd
K-M 0.03600000 0.05590147 0.04967880
ROS 0.03600000 0.05577297 0.04960383
MLE 0.03871202 0.05536429 0.05660577
```

Now, lets create a function that draws a value between 0 and a given *LoD* based on a Normal(μ, σ) density.

```
fx_lod = function(lod,mean,sd) {
    repeat {
      x <- rnorm(1, mean, sd) # generate a value from N(mu,sigma)
      if (x >= 0 && x <= lod) # repeat unless the generated value is >=0 and <LoD
        return(x)
    }
}
```

For example taking the mean and sd from the ROS output and a $LoD = 0.3$ we have

```
fx_lod(0.3,mean(ROS),sd(ROS))
```

```
[1] 0.1469796
```

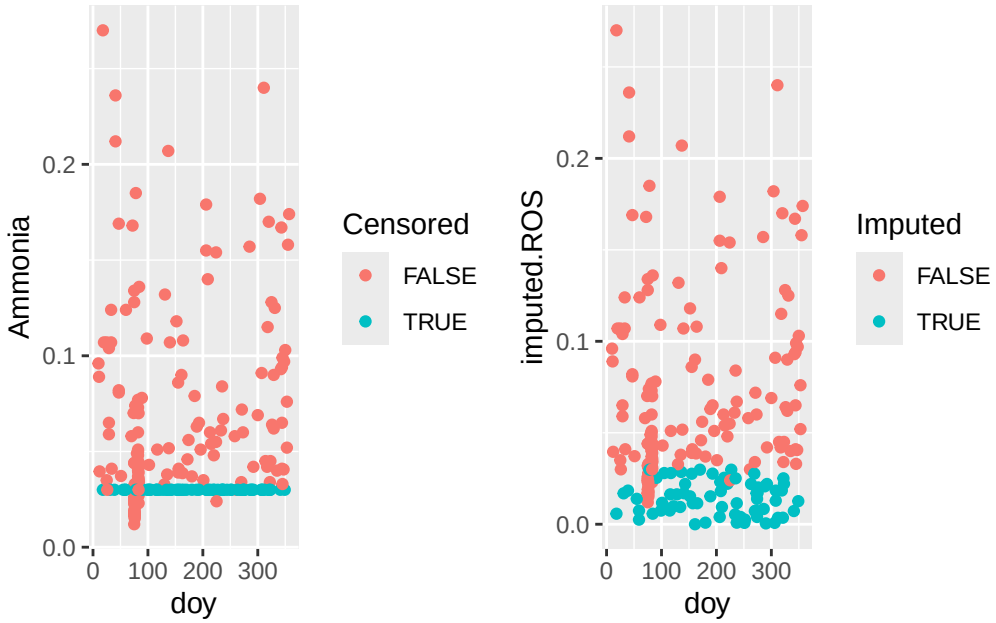
Now we can input the censored values by applying the custom-built `fx_lod()` function using the estimated mean and variance from the ROS as follows:

```
Soar$imputed.ROS <- ifelse(
  Soar$censored == F,
  Soar$Ammonia,  # Keep original if not censored
  # otherwise apply the fx_lod function for each censored observation
  sapply(Soar$Ammonia[Soar$censored], fx_lod, mean = mean(ROS), sd = sd(ROS))
)
```

Lets visualize our results. We can plot the original and imputed Ammonia values against the day of the year as follows:

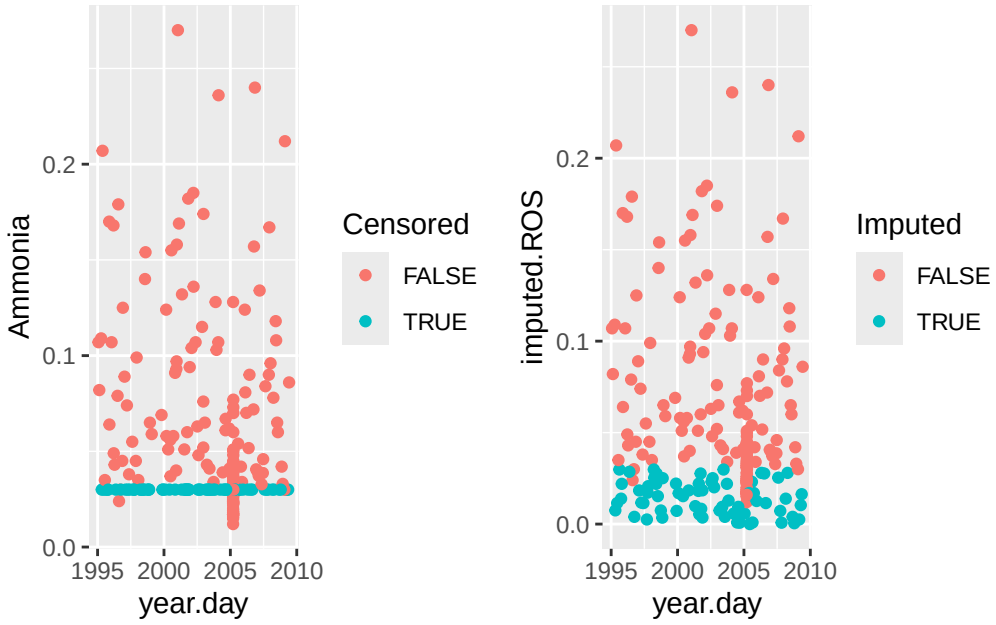
```
ggplot(data=Soar,aes(y=Ammonia,x=doy,color=censored))+
  geom_point() +
```

```
  scale_color_discrete(name="Censored")+
ggplot(data=Soar,aes(y=imputed.ROS,x=doy,color=censored))+
  geom_point() + scale_color_discrete(name="Imputed")
```



Instead, we can create a decimal year or fractional year timestamp by combining the year with the day of year:

```
Soar$year.day <- Soar$year + Soar$doy / 366

ggplot(data=Soar,aes(y=Ammonia,x=year.day,color=censored))+
  geom_point() +
  scale_color_discrete(name="Censored")+
ggplot(data=Soar,aes(y=imputed.ROS,x=year.day,color=censored))+
  geom_point() + scale_color_discrete(name="Imputed")
```



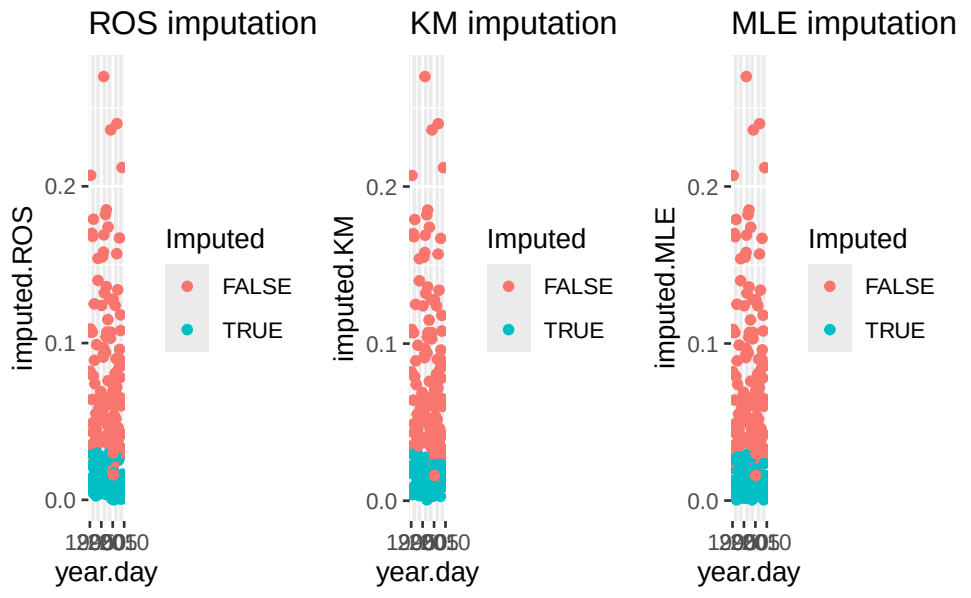
Task 2

1. Add to other columns to the `Soar` data set names `Soar$imputed.KM` and `Soar$imputed.MLE` containing the imputed values for KM and MLE respectively.

2. Create three plots that compare the imputed ammonia values against the decimal year for each method. Discuss how changing the approach taken affects the imputed values.

Click here to see the solution

```r
Soar$imputed.KM <- ifelse(
  Soar$censored == F,
  Soar$Ammonia,  # Keep original if not censored
  # otherwise apply the fx_lod function for each censored observation
  sapply(Soar$Ammonia[Soar$censored], fx_lod, mean = mean(KM), sd = sd(KM))
)

Soar$imputed.MLE <- ifelse(
  Soar$censored == F,
  Soar$Ammonia,  # Keep original if not censored
  # otherwise apply the fx_lod function for each censored observation
  sapply(Soar$Ammonia[Soar$censored], fx_lod, mean = mean(MLE), sd = sd(MLE))
)

ggplot(data=Soar,aes(y=imputed.ROS,x=year.day,color=censored))+
  geom_point() +
  scale_color_discrete(name="Imputed")+
  ggtitle("ROS imputation") +
ggplot(data=Soar,aes(y=imputed.KM,x=year.day,color=censored))+
  geom_point() +
  scale_color_discrete(name="Imputed")+
  ggtitle("KM imputation") +
ggplot(data=Soar,aes(y=imputed.MLE,x=year.day,color=censored))+
  geom_point() +
  scale_color_discrete(name="Imputed")+
  ggtitle("MLE imputation")
```



2 Part 2: Designing a monitoring network

Spatial sampling design is critical for ensuring monitoring data are representative and cost-effective. This practical introduces **generalized random-tessellation stratified (GRTS) sampling** using the `spsurvey` package in R. GRTS provides spatially balanced samples—

avoiding clustering while ensuring geographic coverage—and is widely used in environmental monitoring programs. We will design a monitoring network for lakes in the Northeastern US. The `NE_Lakes` dataset contains the spatial information of 195 lakes in the Northeastern United States which are summarised in the following table:

Variable	Meaning
AREA	Lake area in hectares.
AREA_CAT	Lake area categories based on a hectare cutoff.
ELEV	Elevation in meters.
ELEV_CAT	Elevation categories based on a meter cutoff.
geometry	POINT geometry using the NAD83 / Conus Albers coordinate reference system (EPSG: 5070)

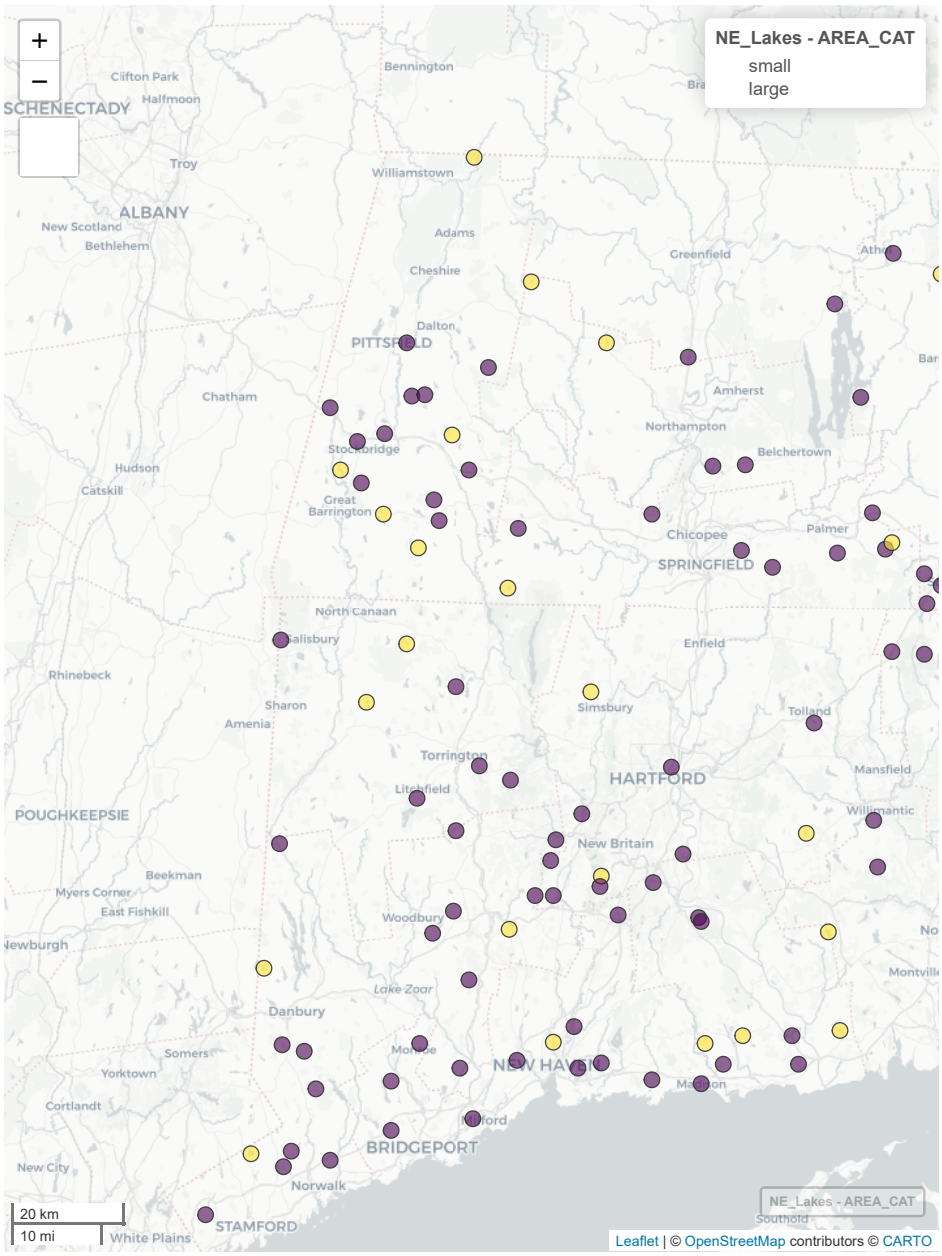
We begin by first loading the data set contained within the `spsurvey` package, additionally we will load the `mapview` and `sf` packages for visualizing our data.

```
library(spsurvey)
library(mapview)
data("NE_Lakes")
```

The `NE_Lakes` data is a Simple Features (`sf`) object containing the spatial information for the point-location of 195 lakes in the Northeastern United States. We can use the `mapview` function to visualise the distribution of lakes and colored them by their size as follow:

```
mapview(NE_Lakes,zcol="AREA_CAT")
```

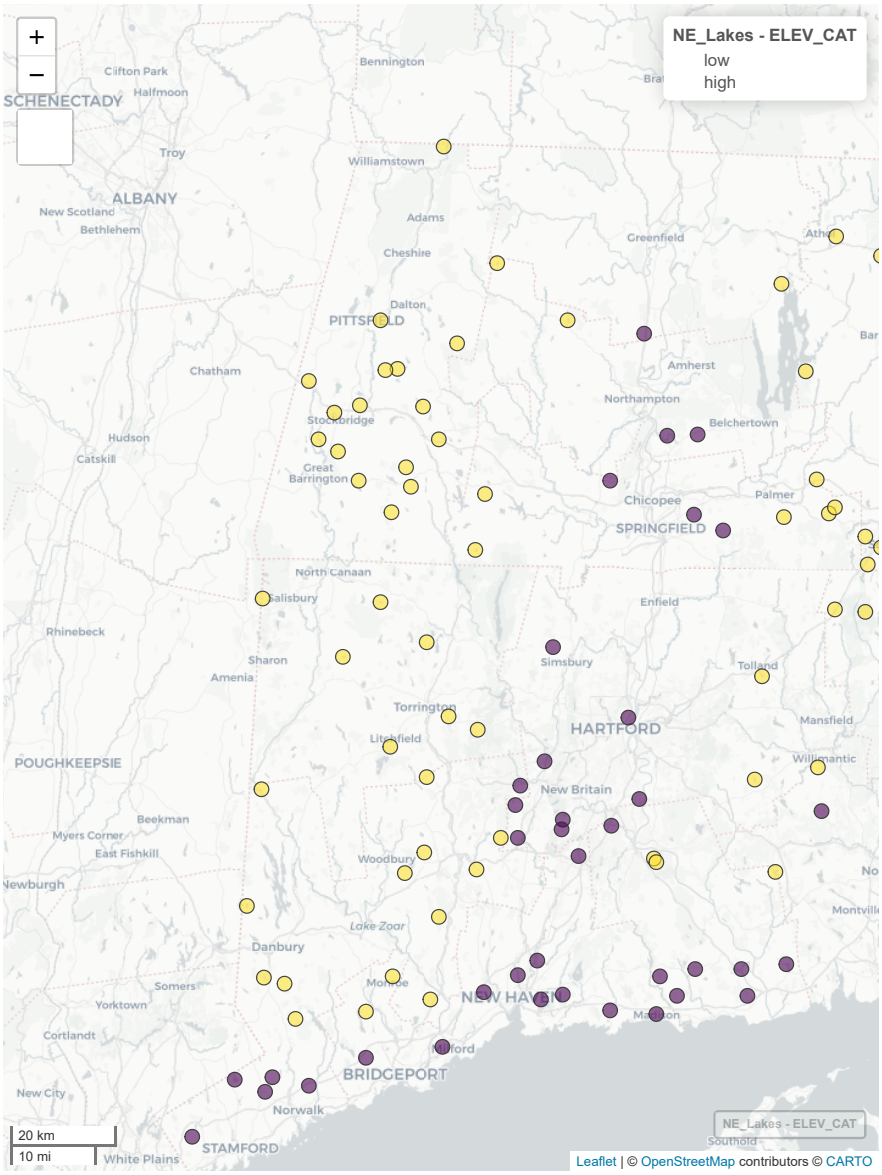
`file:///C:/Users/admin/AppData/Local/Temp/RtmpwxbGHw/file217c3d54135e/widget217c5ad0723b.html`



Task 3

In the previous plot we have shown the spatial distribution of lakes colored by their size. Create a map that shows the lake distribution colored by low and high elevation levels. Do you see any patterns?
Click here to see the solution

```
mapview(NE_Lakes,zcol="ELEV_CAT")
```



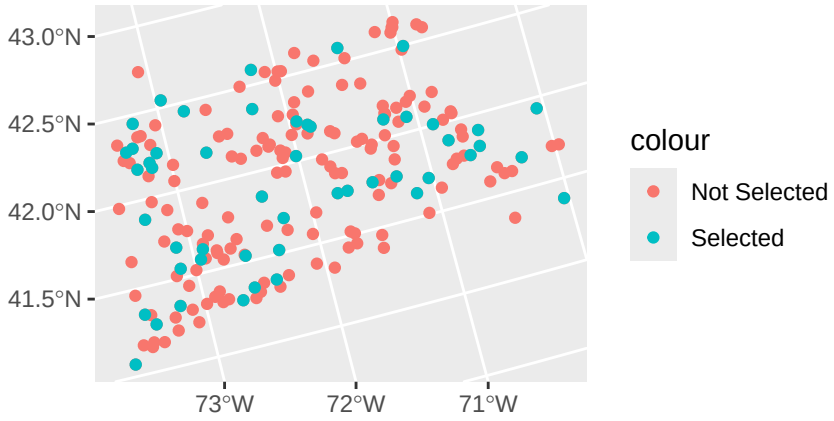
2.1 Independent Random Sampling

Suppose we want to select a random sample of 50 lakes using independent random sampling without replacement. To do so we can use the `irs` function from `spsurvey` to draw a random sample of size 50 with equal probabilities as follows:

```
eqprob_irs <- irs(NE_Lakes, n_base = 50)
```

To visualize the selected sites (lakes) we can use the `sf` library and `ggplot` packages:

```
ggplot() +
  geom_sf(data=NE_Lakes,aes(color="Not Selected")) +
  geom_sf(data=eqprob_irs$sites_base,aes(color="Selected"))
```



2.2 GRTS with equal inclusion probabilities

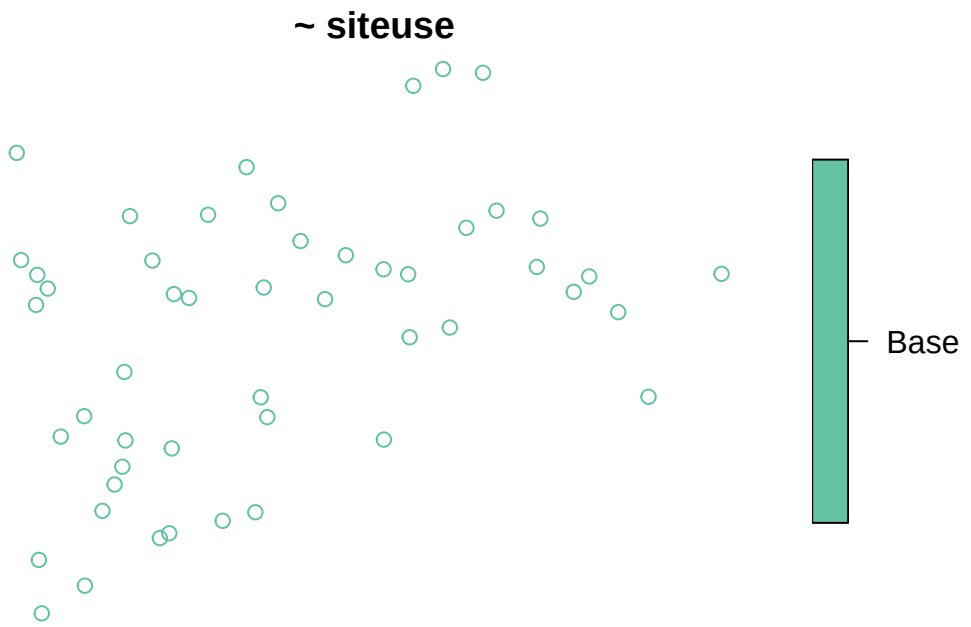
Now, we will implement the GRTS algorithm using the `grts()` function to select a spatially balanced sample of size 50 where each lake has an equal inclusion probability,

```
eqprob <- grts(NE_Lakes, n_base = 50)
```

You can either, `mapview` (interactive map), `ggplot()` (static map) or the R base `plot()` functions to visualize the selected site for monitoring.

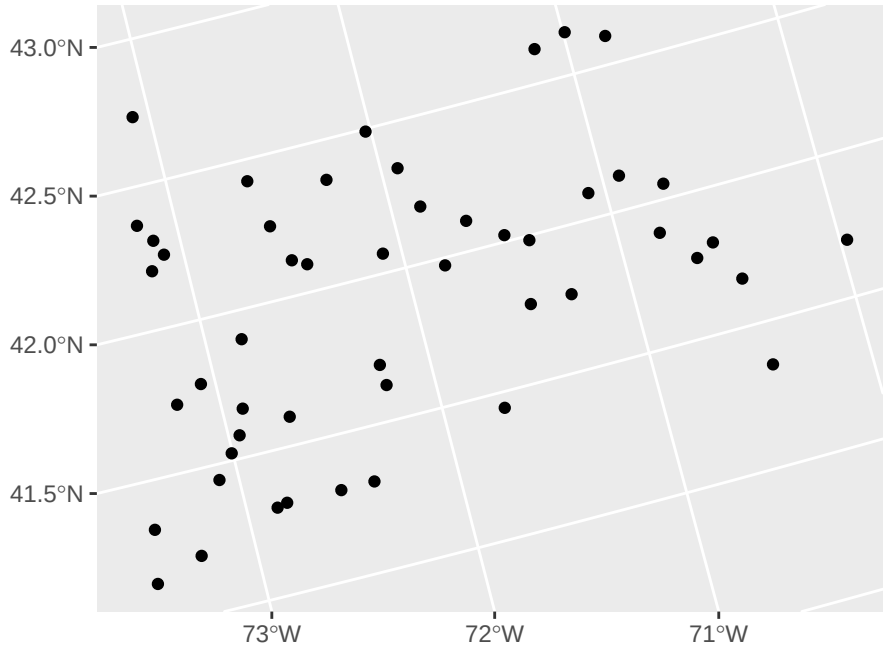
2.3 Base R `plot()`

```
plot(eqprob)
```



2.4 ggplot and sf

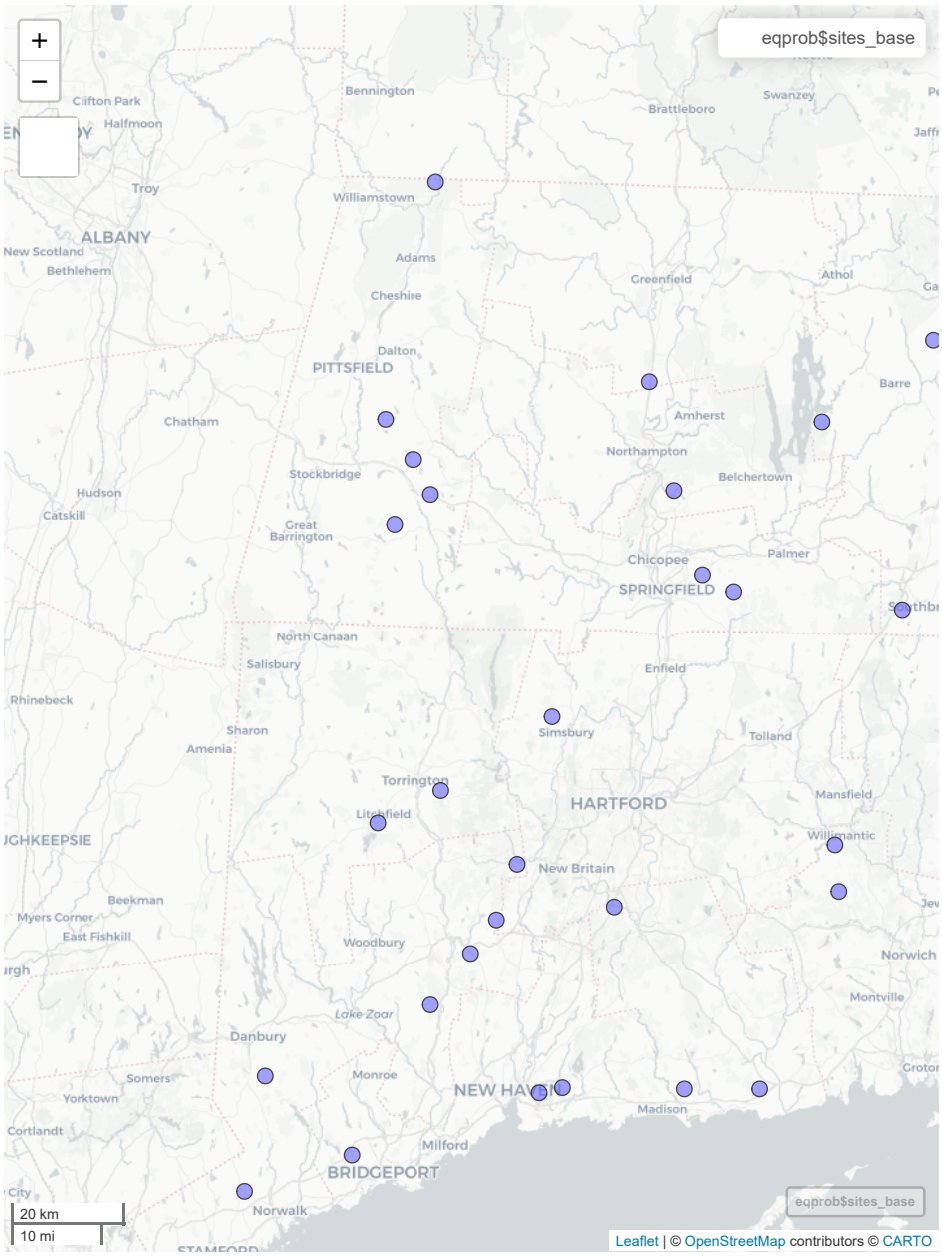
```
ggplot() +
  geom_sf(data=eqprob$sites_base)
```



2.5 mapview

```
mapview(eqprob$sites_base)
```

file:///C:/Users/admin/AppData/Local/Temp/RtmpwxbGHw/file217c76d33f23/widget217c3777223.html s



2.6 GRTS with stratified sampling

Instead of sampling from the entire sampling area simultaneously, we can apply the GRTS algorithm for a given strata and select samples from each stratum independently of other strata.

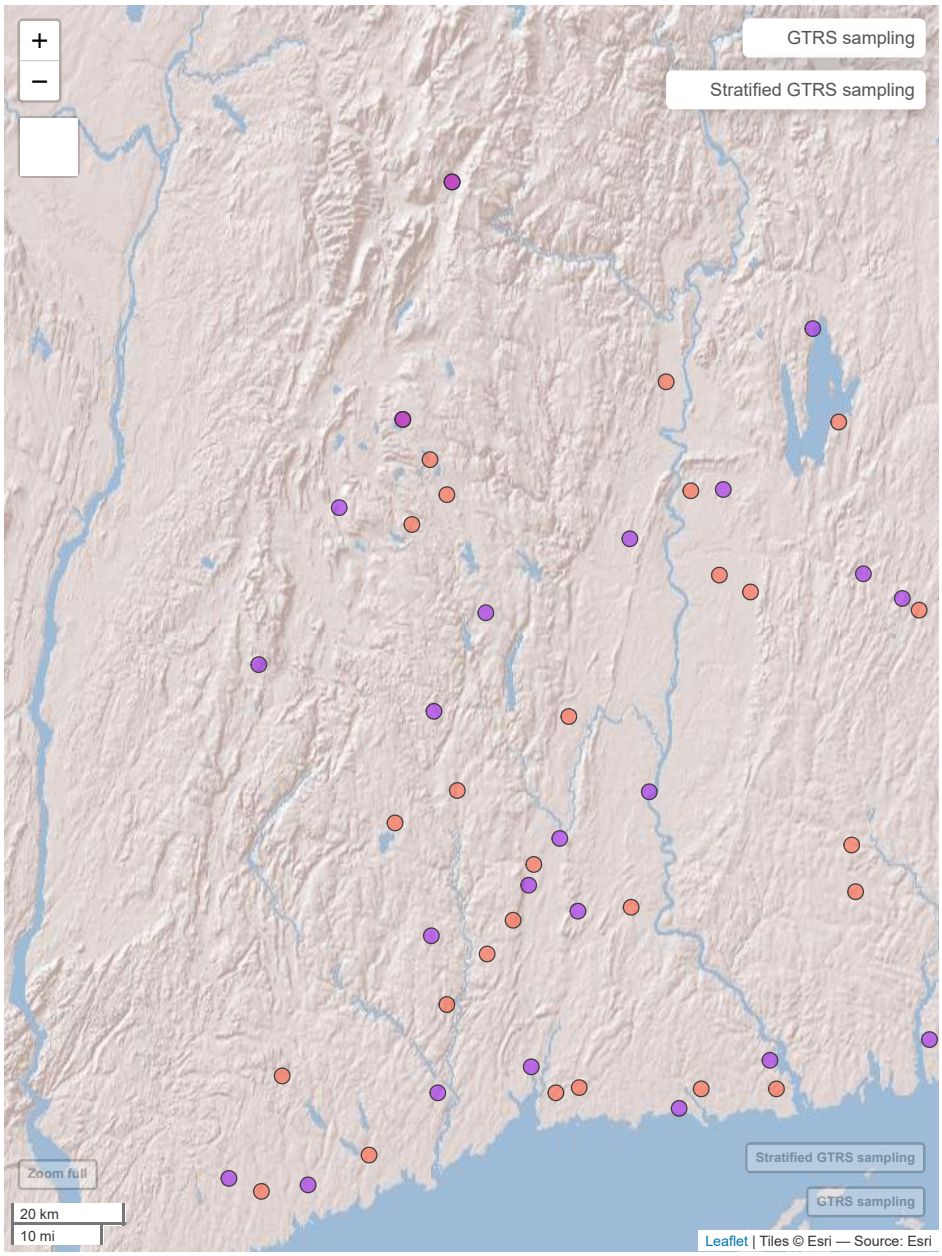
In this example we will obtain a GRTS sample stratified by the lake elevation categories where all lake within a stratum have equal inclusion probabilities:

```
n_strata <- c(low = 35, high = 15)

eqprob_strat <- grts(NE_Lakes, n_base = n_strata,
                     stratum_var = "ELEV_CAT")
```

Here,`n_strata` specifies the stratum-specific sample sizes (35 for low elevation category and 15 for the high elevation category). Notice that the names in `n_strata` (low and high) need to match the names of the stratification variable (`"ELEV_CAT"`) in the `NE_Lakes` data. The the `grts` function receives as arguments the `n_strata` vector and name of the column in the data that represents the stratification variable via the `stratum_var` argument.

```
mapview(eqprob$sites_base,
        map.types = c("Esri.WorldShadedRelief"),
        col.regions = "tomato",
        layer.name="GTRS sampling")+
mapview(eqprob_strat$sites_base,
        map.types = c("Esri.WorldShadedRelief"),
        col.regions = "purple",
        layer.name="Stratified GTRS sampling")
```

```
file:///C:/Users/admin/AppData/Local/Temp/RtmpwxbGHw/file217c7b2147ba/widget217c3b4f362.html s
```



2.7 GRTS with unequal inclusion probabilities

Sometimes we don't want inclusion probabilities to be equal for all sites. For example, we may want larger lakes to be sampled more frequently than smaller lakes based on attributes like surface area.

The `caty_n` and `caty_var` arguments in the `grts` functions allows us to select a GRTS sample with unequal inclusion probabilities according to a particular category e.g., lake area.

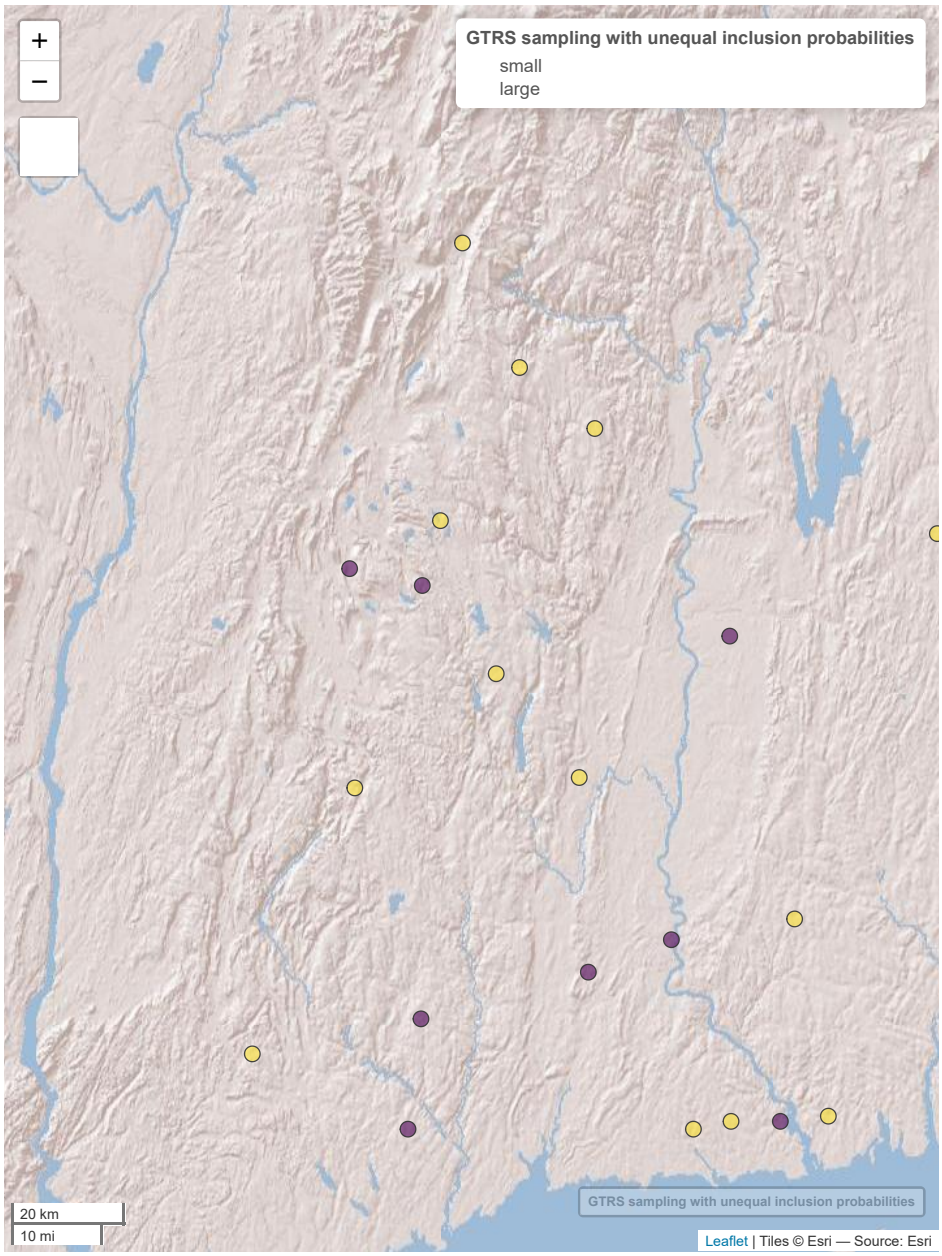
```
caty_n <- c(small = 10, large = 40)
uneqprob <- grts(NE_Lakes, n_base = 50, caty_n = caty_n, caty_var = "AREA_CAT")
```

The `cat_n` vector specifies the within-level sample sizes. This gets passed on to `grts` via the `caty_n` argument. The names in `caty_n` must match the different levels of categorical variable which in the data (specified via the `caty_var` argument).

The map below shows a sample size of 40 for large lakes and sample size of 10 small lakes.

```
mapview(uneqprob$sites_base,
        zcol="AREA_CAT",
        map.types = c("Esri.WorldShadedRelief"),
        layer.name="GTRS sampling with unequal inclusion probabilities")
```

```
file:///C:/Users/admin/AppData/Local/Temp/RtmpwxbGHw/file217c1f92278f/widget217c615a39a4.html
```



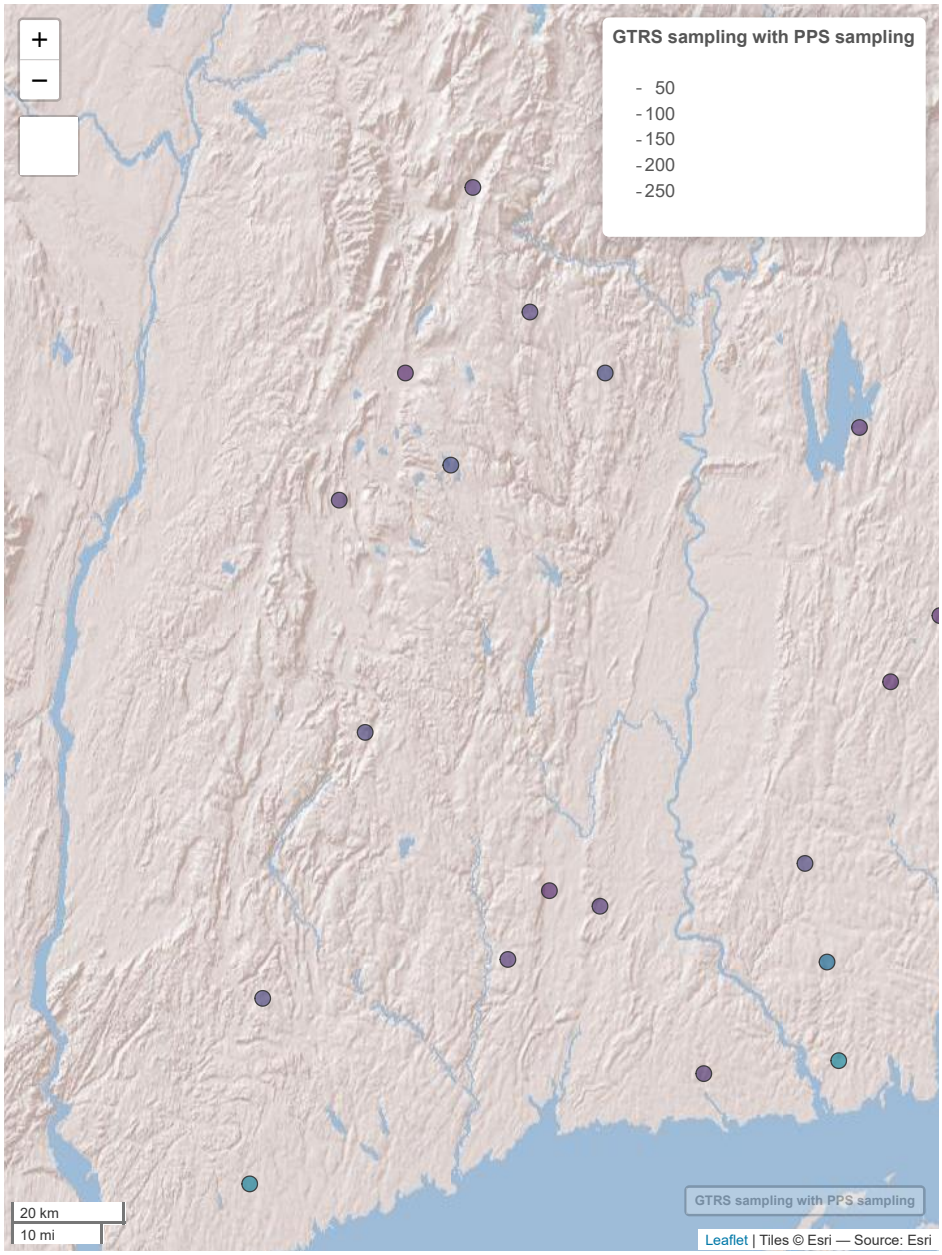
We can also implement probability proportional to size (PPS) sampling, where each lake's selection probability is directly proportional to its surface area. This avoids arbitrary size

categories and ensures larger lakes—which often have greater ecological and socioeconomic importance—are more likely to be selected. We can conduct PPS sampling by specifying the lake's areas as an `aux_var` variable in the `grts()` function:

```
propprob <- grts(NE_Lakes, n_base = 50, aux_var = "AREA")
```

```
mapview(propprob$sites_base,
                zcol="AREA",
        map.types = c("Esri.WorldShadedRelief"),
        layer.name="GTRS sampling with PPS sampling")
```

```
file:///C:/Users/admin/AppData/Local/Temp/RtmpwxbGHw/file217c45be52c4/widget217c739661.html sc
```



2.8 Additional features in spsurvey

Other useful features that have been implemented on the grts function are (see Dumelle et al. (2023) for a more comprehensive description of the package):

- *Legacy sites* - allows sites selected from a previous sampling scheme to be selected in a new sample (grts(NE_Lakes, n_base = 50,legacy_sites = NE_Lakes_Legacy)). This is often used to study or monitor the behavior of the sites in the network through time.
- *Minimum distance selection* -Sometimes the selected sites are too close to each other. We can set a minimum distance between sites by setting the mindis argument to a particular distance determined by the data CRS (e.g., grts(NE_Lakes, n_base = 50, mindis = 1600))
- *Replacement sites* - it is common that once a network has been designed the data at some of the selected in the sample not able to been able to collected at the site (e.g, due terrain contraints or landowner permission). We can then use a nearest neighbor approach to selects replacement sites according to the distance between GRTS-sampled site and all other sites in the sampling frame that are not part of the GRTS sample. E.g., to select a GRTS sample of size 50 with two nearest neighbor replacement we can run eqprob_nn <- grts(NE_Lakes, n_base = 50, n_near = 2).

2.9 Assessing spatial balance

A practical way to measure spatial balance was developed by Stevens and Olsen (2004) using Voronoi polygons. In this approach, each sampled site defines a region containing all locations closer to it than to any other sampled site. For a spatially balanced design, the total inclusion probability of all sites within each Voronoi polygon is expected to be 1. Deviation from this ideal can be quantified using a loss metric based on these polygon totals. One common choice is Pielou's evenness index (PEI), which assesses how uniformly the inclusion probability is distributed across the sample sites. Pielou's evenness index (PEI) is defined as:

$$\text{PEI} = 1 + \frac{\sum_{i=1}^{n} \frac{v_i}{n} \ln(v_i/n)}{\ln(n)},$$

where n is the sample size. PEI is bounded between zero and one. A PEI of zero indicates perfect spatial balance. As PEI increases, the spatial balance worsens. The sp_balance() function which receives as arguments the (i) design sites and (ii) the sampling frame (note that if stratified sampling is being compared you also need to supply the name of the stratified variable in your data via the stratum_var arugment).

The following code compares the GRTS with equal inclusion probabilities again the SRS:

```
sp_balance(eqprob$sites_base, NE_Lakes)
```

```
  stratum metric      value
1    None pielou 0.02918495
```

```
sp_balance(eqprob_irs$sites_base, NE_Lakes)
```

```
  stratum metric      value
1    None pielou 0.03701446
```

Question

Which design has better spatial balance?

- (A) IRS
- (B) GRTS

So far we have applied the GRTS algorithm to point reference data. However, this methodology can also be applied on linear and areal data in similar fashion – the only difference being the geometry type of the `sf` object used as argument. In the next exercise, you will be tasked with designing a river network using the GRTS algorithm for a section of the Illinois River in Arkansas and Oklahoma

Task 4

The `Illinois_River` data in `spsurvey` contains the spatial information of 244 segments of the Illinois River in Arkansas and Oklahoma. The data can be accessed with `data(Illinois_River)`. Use the `grts` to:

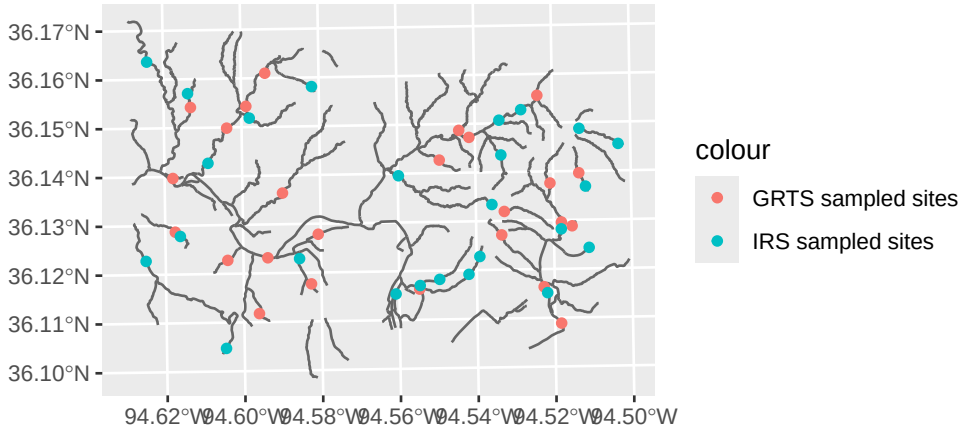
1. Design a monitoring network with $n = 25$ sampling points using *independent random sampling*.
2. Design a monitoring network with $n = 25$ sampling points using *GRTS sampling* with equal inclusion probabilities.
3. Plot both sampling designs using `ggplot2`, coloring the selected sites according to the sampling method.

Finally, compare the spatial balance of the two approaches. Which method provides better spatial coverage across the river network?
Take hint
You can add multiple `geom_sf()` layers to a ggplot object, e.g.,

```
ggplot()+
geom_sf(data=layer_1)+
geom_sf(data=leayer_2) +...
```

Click here to see the solution

```
data(Illinois_River)

irs_linear <- irs(Illinois_River, n_base = 25)
eqprob_linear <- grts(Illinois_River, n_base = 25)

ggplot()+
  geom_sf(data=Illinois_River,color="gray40")+
  geom_sf(data=eqprob_linear$sites_base,aes(color="GRTS sampled sites"))+
  geom_sf(data=irs_linear$sites_base,aes(color="IRS sampled sites"))
```




```
sp_balance(irs_linear$sites_base, Illinois_River)
```

```
  stratum metric      value
1    None pielou 0.04648646
```

```
sp_balance(eqprob_linear$sites_base, Illinois_River)
```

```
  stratum metric      value
1    None pielou 0.03126968
```

```
# GRTS PEI is smaller and thus provided a better spatially balanced design
```

Dumelle, Michael, Tom Kincaid, Anthony R. Olsen, and Marc Weber. 2023. "**Spsurvey**: Spatial Sampling Design and Analysis in *R*." *Journal of Statistical Software* 105 (3). https://doi.org/10.18637/jss.v105.i03.

Stevens, Don L, and Anthony R Olsen. 2004. "Spatially Balanced Sampling of Natural Resources." *Journal of the American Statistical Association* 99 (465): 262–78. https://doi.org/10.1198/016214504000000250.