

Analysis of a GRTS Survey Design for a Finite Resource

Thomas Kincaid

August 16, 2016

Contents

1 Preliminaries	1
2 Load the survey design and analytical variables data set	2
3 Analysis of site status evaluation variables	5
4 Analysis of lake condition variables	8
5 Analysis of lake condition variables correcting for population size	10
6 Analysis of quantitative variables	12

1 Preliminaries

This document presents analysis of a GRTS survey design for a finite resource. The finite resource used in the analysis is small lakes in Florida. The analysis will include calculation of three types of population estimates: (1) estimation of proportion and size (number of lakes) for site evaluation status categorical variables; (2) estimation of proportion and size for lake condition categorical variables; and (3) estimation of the cumulative distribution function (CDF) and percentiles for quantitative variables. Testing for difference between CDFs from subpopulations also will be presented.

The initial step is to use the library function to load the spsurvey package. After the package is loaded, a message is printed to the R console indicating that the spsurvey package was loaded successfully.

Load the spsurvey package

```
> # Load the spsurvey package
> library(spsurvey)
>
```

Version 3.2 of the spsurvey package was loaded successfully.

2 Load the survey design and analytical variables data set

The original Florida small lakes data file contains more than 3,800 records and 29 basins. To produce a more manageable number of records, only six basins were retained in the data that will be analyzed, which produced a file containing 930 records.

The next step is to load the data set, which includes both survey design variables and analytical variables. The data function is used to load the data set and assign it to a data frame named FL_lakes. The nrow function is used to determine the number of rows in the FL_lakes data frame, and the resulting value is assigned to an object named nr. Finally, the initial six lines and the final six lines in the FL_lakes data frame are printed using the head and tail functions, respectively.

Load the survey design and analytical variables data set

```
> # Load the data set and determine the number of rows in the data frame
> data(FL_lakes)
> nr <- nrow(FL_lakes)
>
```

Display the initial six lines in the data file.

```
> # Display the initial six lines in the data file
> head(FL_lakes)
```

	siteID	xcoord	ycoord	wgt	Basin	Status	TNT
1	FLW03414-0014	8635535	12860896	5.369048	NWFWMD-1	Sampled	Target
2	FLW03414-0046	8636136	12886783	5.369048	NWFWMD-1	Physical_Barrier	Target
3	FLW03414-0062	8617834	12869126	5.369048	NWFWMD-1	NonTarget	NonTarget
4	FLW03414-0078	8673500	12883071	5.369048	NWFWMD-1	Physical_Barrier	Target
5	FLW03414-0086	8631884	12816428	5.369048	NWFWMD-1	NonTarget	NonTarget
6	FLW03414-0118	8607699	12856644	5.369048	NWFWMD-1	NonTarget	NonTarget
	pH_Cat	Coliform_Cat	Oxygen	Turbidity			
1	(0,6]	(0,5]	9.9	0.4			
2	<NA>	<NA>	NA	NA			

```

3  <NA>          <NA>      NA      NA
4  <NA>          <NA>      NA      NA
5  <NA>          <NA>      NA      NA
6  <NA>          <NA>      NA      NA

```

```
>
```

Display the final six lines in the data file.

```

> # Display the final six lines in the data file
> tail(FL_lakes)

```

	siteID	xcoord	ycoord	wgt	Basin	Status	TNT
925	FLW03414-3878	8880656	12694963	4.80791	SWFWMD-4	Dry	Target
926	FLW03414-3886	8892406	12732977	4.80791	SWFWMD-4	Sampled	Target
927	FLW03414-3894	8836528	12723056	4.80791	SWFWMD-4	Dry	Target
928	FLW03414-3918	8923107	12725502	4.80791	SWFWMD-4	Landowner_Denial	Target
929	FLW03414-3926	8861298	12715824	4.80791	SWFWMD-4	Dry	Target
930	FLW03414-3950	8888601	12715641	4.80791	SWFWMD-4	NonTarget	NonTarget

	pH_Cat	Coliform_Cat	Oxygen	Turbidity
925	<NA>	<NA>	NA	NA
926	(6,8]	(5,50]	1.98	8.2
927	<NA>	<NA>	NA	NA
928	<NA>	<NA>	NA	NA
929	<NA>	<NA>	NA	NA
930	<NA>	<NA>	NA	NA

```
>
```

The sample of small lakes in Florida is displayed in Figure 1. The sample sites for each basin are displayed using a unique color.

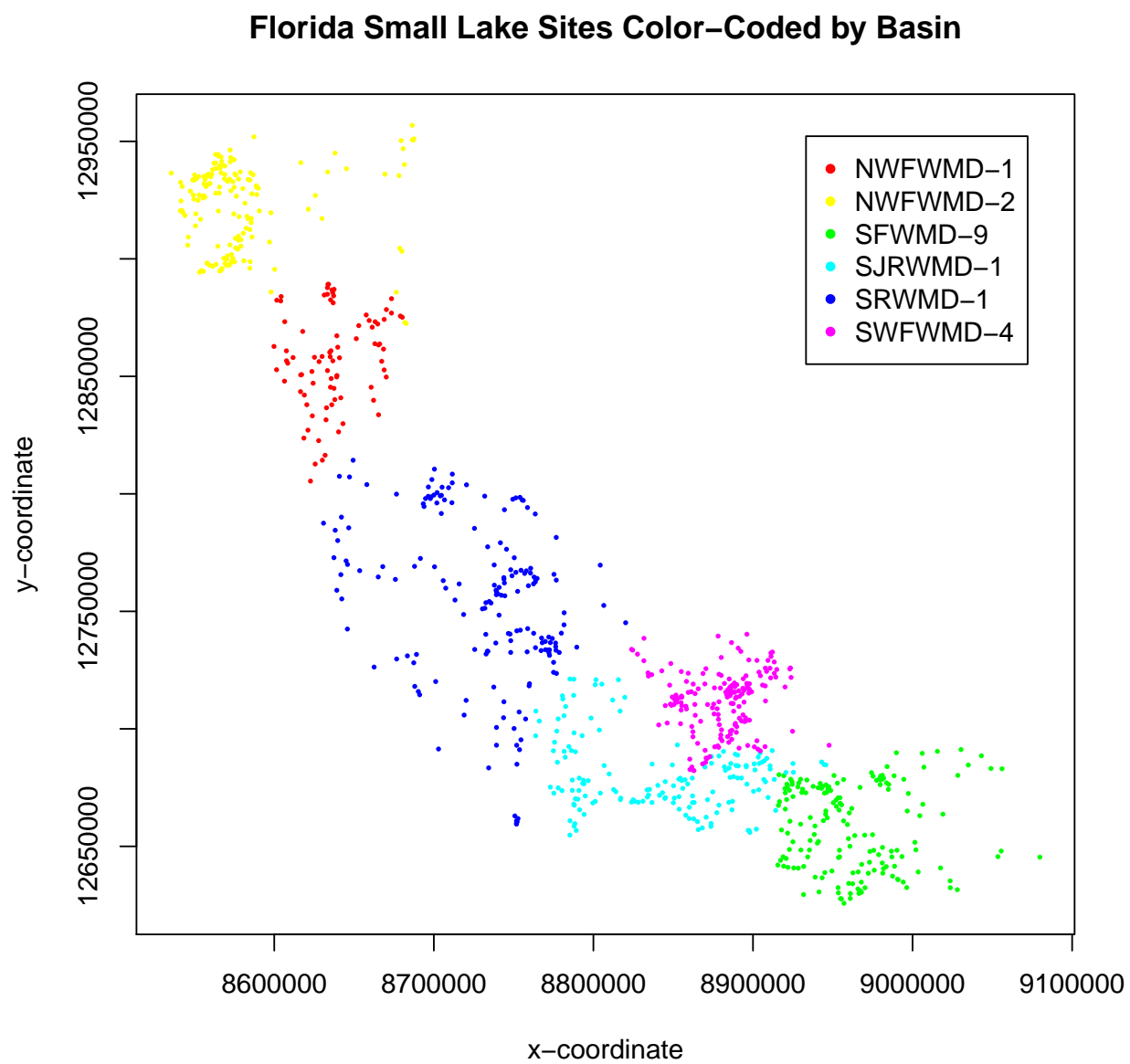


Figure 1: Location of small lake sample sites in Florida color-coded by basin.

3 Analysis of site status evaluation variables

The first analysis that will be examined is calculation of extent estimates for site status evaluation variables. Extent is measured both by the proportion of the resource in status evaluation categories and by size of the resource in each category. For a finite resource like lakes, size refers to the number of lakes in a category. For calculating extent estimates (and for all of the analyses we will consider), the survey design weights are incorporated into the calculation process. Weights used in the analyses were modified from the original survey design weights to ensure that the weights sum to the known size of the resource. Further information regarding weight adjustment is provided in the help page for the `adjwgt` (weight adjustment) function. Two site status variables will be examined: (1) `status`, which classifies lakes into six evaluation categories and (2) `TNT`, which classifies lakes as either "Target" or "NonTarget". The `table` and `addmargins` functions are used to create tables displaying the count for each code (level) of the two status variables.

```
> addmargins(table(FL_lakes$Status))
```

A table displaying the number of values for each level of the status variable follows:

	Dry	Landowner_Denial	NonTarget
	223	119	317
Otherwise_Unsampleable		Physical_Barrier	Sampled
	1	99	171
Sum			
	930		

```
> addmargins(table(FL_lakes$TNT))
```

A table displaying the number of values for each level of the TNT variable follows:

NonTarget	Target	Sum
317	613	930

The `cat.analysis` function in the `spsurvey` package will be used to calculate extent estimates. Four data frames constitute the primary input to the `cat.analysis` function. The first column (variable) in the four data frames provides the unique identifier (site ID) for each sample site and is used to connect records among the data frames. The `siteID` variable in the `FL_lakes` data frame is assigned to the `siteID` variable in the data frames. The four data frames that will be created are named as follows: `sites`, `subpop`, `design`, and `data.cat`. The `sites` data frame identifies sites to use in the analysis and contains two variables: (1) `siteID` - site ID

values and (2) Use - a logical vector indicating which sites to use in the analysis. The rep (repeat) function is used to assign the value TRUE to each element of the Use variable. Recall that nr is an object containing the number of rows in the FL_lakes data frame. The subpop data frame defines populations and, optionally, subpopulations for which estimates are desired. Unlike the sites and design data frames, the subpop data frame can contain an arbitrary number of columns. The first variable in the subpop data frame identifies site ID values and each subsequent variable identifies a type of population, where the variable name is used to identify type. A type variable identifies each site with a character value. If the number of unique values for a type variable is greater than one, then the set of values represent subpopulations of that type. When a type variable consists of a single unique value, then the type does not contain subpopulations. For this analysis, the subpop data frame contains three variables: (1) siteID - site ID values, (2) CombinedBasins - which will be used to calculate estimates for all of the basins combined, and (3) Basin - which will be used to calculate estimates for each basin individually. The basin variable in the FL_lakes data frame is assigned to the Basin variable in the subpop data frame. The design data frame consists of survey design variables. For the analysis under consideration, the design data frame contains the following variables: (1) siteID - site ID values; (2) wgt - final, adjusted, survey design weights; (3) xcoord - x-coordinates for location; and (4) ycoord - y-coordinates for location. The wgt, xcoord, and ycoord variables in the design data frame are assigned values using variables with the same names in the FL_lakes data frame. Like the subpop data frame, the data.cat data frame can contain an arbitrary number of columns. The first variable in the data.cat data frame identifies site ID values and each subsequent variable identifies a response variable. The two response variables are Status and Target_NonTarget, which are assigned the status and TNT variables, respectively, in the FL_lakes data frame. Missing data (NA) is allowed for the response variables, which are the only variables in the input data frames for which NA values are allowed.

Create the sites data frame.

```
> sites <- data.frame(siteID=FL_lakes$siteID,
+                      Use=rep(TRUE, nr))
```

Create the subpop data frame.

```
> subpop <- data.frame(siteID=FL_lakes$siteID,
+                      CombinedBasins=rep("All Basins", nr),
+                      Basin=FL_lakes$Basin)
```

Create the design data frame.

```
> design <- data.frame(siteID=FL_lakes$siteID,
+                      wgt=FL_lakes$wgt,
+                      xcoord=FL_lakes$xcoord,
+                      ycoord=FL_lakes$ycoord)
```

Create the data.cat data frame.

```
> data.cat <- data.frame(siteID=FL_lakes$siteID,
+                         Status=FL_lakes$Status,
+                         Target_NonTarget=FL_lakes$TNT)
```

Use the cat.analysis function to calculate extent estimates for the site status evaluation variables.

```
> # Calculate extent estimates for the site status evaluation variables
> Extent_Estimates <- cat.analysis(sites, subpop, design, data.cat)
>
```

The extent estimates for all basins combined are displayed using the print function. The object produced by cat.analysis is a data frame containing thirteen columns. The first five columns identify the population (Type), subpopulation (Subpopulation), response variable (Indicator), levels of the response variable (Category), and number of values in a category (NResp). A category labeled "Total" is included for each combination of population, subpopulation, and response variable. The next four columns in the data frame provide results for the proportion estimates: the proportion estimate (Estimate.P), standard error of the estimate (StdError.P), lower confidence bound (LCB95Pct.P), and upper confidence bound (UCB95Pct.P). Argument conf for cat.analysis allows control of the confidence bound level. The default value for conf is 95, hence the column names for confidence bounds contain the value 95. Supplying a different value to the conf argument will be reflected in the confidence bound names. Confidence bounds are obtained using the standard error and the Normal distribution multiplier corresponding to the confidence level. The final four columns in the data frame provide results for the size (units) estimates: the units estimate (Estimate.U), standard error of the estimate (StdError.U), lower confidence bound (LCB95Pct.U), and upper confidence bound (UCB95Pct.U). Note that the size estimate for the Total category will be equal to the sum of the survey design weights.

```
> # Print the extent estimates for all basins combined
> print(Extent_Estimates[c(1:7, 45:47),])
```

	Type	Subpopulation	Indicator	Category	NResp
1	CombinedBasins	All Basins	Status	Dry	223
2	CombinedBasins	All Basins	Status	Landowner_Denial	119
3	CombinedBasins	All Basins	Status	NonTarget	317
4	CombinedBasins	All Basins	Status	Otherwise_Unsampleable	1
5	CombinedBasins	All Basins	Status	Physical_Barrier	99
6	CombinedBasins	All Basins	Status	Sampled	171
7	CombinedBasins	All Basins	Status	Total	930
45	CombinedBasins	All Basins	Target_NonTarget	NonTarget	317

	CombinedBasins	All Basins	Target_NonTarget		Target	613
	CombinedBasins	All Basins	Target_NonTarget		Total	930
	Estimate.P	StdError.P	LCB95Pct.P	UCB95Pct.P	Estimate.U	StdError.U
1	23.01117939	0.97789814	21.094534	24.9278245	1184.155291	50.188531
2	13.32737468	0.99049216	11.386046	15.2687037	685.826701	50.967147
3	36.91250997	1.15995817	34.639034	39.1859862	1899.517763	60.260564
4	0.09422536	0.08497475	0.000000	0.2607728	4.848837	4.372792
5	8.47917794	0.71507723	7.077652	9.8807036	436.338497	36.766620
6	18.17553265	1.03356643	16.149780	20.2012856	935.312910	53.169549
7	100.00000000	0.00000000	100.000000	100.0000000	5146.000000	9.275053
45	36.91250997	1.15995817	34.639034	39.1859862	1899.517763	60.260564
46	63.08749003	1.15995817	60.814014	65.3609663	3246.482237	59.166424
47	100.00000000	0.00000000	100.000000	100.0000000	5146.000000	9.275053
	LCB95Pct.U	UCB95Pct.U				
1	1085.7876	1282.52300				
2	585.9329	785.72047				
3	1781.4092	2017.62630				
4	0.0000	13.41935				
5	364.2772	508.39975				
6	831.1025	1039.52331				
7	5127.8212	5164.17877				
45	1781.4092	2017.62630				
46	3130.5182	3362.44630				
47	5127.8212	5164.17877				

>

The write.csv function is used to store the extent estimates as a comma-separated value (csv) file. Files in csv format can be read by programs such as Microsoft Excel.

```
> write.csv(Extent_Estimates, file="Extent_Estimates.csv")
```

4 Analysis of lake condition variables

The second analysis that will be examined is estimating resource proportion and size for lake condition variables. Two lake condition variables will be examined: (1) pH_cat, which classifies lakes by categories of pH value and (2) coliform_cat, which classifies lakes by categories of fecal coliform count. The table and addmargins functions are used to create tables displaying the count for each level of the two lake condition variables.

```
> addmargins(table(FL_lakes$pH_Cat))
```


A table displaying the number of values for each level of the pH category variable follows:

(0,6]	(6,8]	(8,14]	Sum
78	82	11	171

```
> addmargins(table(FL_lakes$Coliform_Cat))
```

A table displaying the number of values for each level of the fecal coliform category variable follows:

(0,5]	(5,50]	(50,500]	(500,5e+03]	Sum
97	40	31	2	170

As for extent estimates, the `cat.analysis` function will be used to calculate condition estimates. The sites data frame for this analysis differs from the one used to calculate extent estimates. The `Use` logical variables in `sites` is set equal to the value "Sampled", so that only sampled sites are used in the analysis. The `subpop` and `design` data frames created in the prior analysis can be reused for this analysis. The `data.cat` data frame contains the two lake condition variables: `pHCat` and `ColiformCat`. Variables `pH_cat` and `coliform_cat` in the `FL_lakes` data frame are assigned to `pHCat` and `ColiformCat`, respectively.

Create the sites data frame.

```
> sites <- data.frame(siteID=FL_lakes$siteID,
+                      Use=FL_lakes$Status == "Sampled")
```

Create the `data.cat` data frame.

```
> data.cat <- data.frame(siteID=FL_lakes$siteID,
+                         pHCat=FL_lakes$pH_Cat,
+                         ColiformCat=FL_lakes$Coliform_Cat)
```

Use the `cat.analysis` function to calculate estimates for the lake condition variables.

```
> # Calculate estimates for the categorical variables
> Condition_Estimates <- cat.analysis(sites, subpop, design, data.cat)
>
```

Print the lake condition estimates for all basins combined.

```
> # Print the condition estimates for all basins combined
> print(Condition_Estimates[c(1:4, 28:32),])
```

	Type	Subpopulation	Indicator	Category	NResp	Estimate.P	
1	CombinedBasins	All Basins	pHCat	(0,6]	78	42.915056	
2	CombinedBasins	All Basins	pHCat	(6,8]	82	50.396558	
3	CombinedBasins	All Basins	pHCat	(8,14]	11	6.688386	
4	CombinedBasins	All Basins	pHCat	Total	171	100.000000	
28	CombinedBasins	All Basins	ColiformCat	(0,5]	97	55.986933	
29	CombinedBasins	All Basins	ColiformCat	(5,50]	40	24.108155	
30	CombinedBasins	All Basins	ColiformCat	(50,500]	31	18.521502	
31	CombinedBasins	All Basins	ColiformCat	(500,5e+03]	2	1.383410	
32	CombinedBasins	All Basins	ColiformCat	Total	170	100.000000	
	StdError.P	LCB95Pct.P	UCB95Pct.P	Estimate.U	StdError.U	LCB95Pct.U	UCB95Pct.U
1	2.8530505	37.323179	48.506932	401.39005	26.886965	348.69257	454.08754
2	3.0180108	44.481366	56.311751	471.36552	28.637754	415.23655	527.49448
3	1.5603961	3.630066	9.746706	62.55734	14.557867	34.02444	91.09023
4	0.0000000	100.000000	100.000000	935.31291	7.447521	920.71604	949.90978
28	2.8761564	50.349770	61.624096	519.19950	26.470305	467.31866	571.08035
29	3.0417644	18.146407	30.069904	223.56900	28.568114	167.57652	279.56147
30	2.4596628	13.700651	23.342352	171.76069	22.738993	127.19309	216.32830
31	0.8268103	0.000000	3.003929	12.82917	7.673900	0.00000	27.86974
32	0.0000000	100.000000	100.000000	927.35836	7.435967	912.78414	941.93259

>

Use the write.csv function to write the condition estimates as a csv file.

```
> write.csv(Condition_Estimates, file="Condition_Estimates.csv")
```

5 Analysis of lake condition variables correcting for population size

The frame is a data structure containing spatial location data in addition to other attributes regarding a resource of interest and is used to create a survey design. A frame often takes the form of a shapefile. The frame can be used to obtain size values (e.g., number of lakes) for the populations and subpopulations examined in an analysis. Examination of the Estimate.U column in the Condition_Estimates data frame produced by cat.analysis reveals that the estimated Total value for both condition variables and each combination of population value and subpopulation value does not sum to the corresponding frame size value. For example, the Total entry in the Estimate.U column for the pHcat variable, population "CombinedBasins" and subpopulation "All Basins" is 935 (rounded to a whole number). This value is an estimate of the size of the sampled resource. The corresponding frame size value is 5,146. The popsize (population size) argument to cat.analysis provides a mechanism for forcing the size estimates to sum to a desired value, e.g., the frame size

value. Note that including popsize as an argument results in assigning the popsize value to the Total category of the size estimates. Use of the popsize argument assumes that sites which were evaluated but not sampled were missing at random. The missing at random assumption may not be a valid assumption, e.g., sites for which access was denied by the landowner may not be the same as sites that were sampled. For the current analysis, we will assume that the assumption is valid. As a first step for use of the popsize argument, the c (combine) function is used to create a named vector of frame size values for each basin. Output from the c function is assigned to an object named framesize. The popsize argument is a list, which is a particular type of R object. The popsize list must include an entry for each population type included in the subpop data frame, i.e., CombinedBasins and Basin for this analysis. The sum function applied to framesize is assigned to the CombinedBasins entry in the popsize list. Recall that the basin population type contains subpopulations, i.e., basins. When a population type contains subpopulations, the entry in the popsize list also is a list. The as.list function is applied to framesize, and the result is assigned to the Basin entry in the popsize list.

Assign frame size values.

```
> framesize <- c("NFWMD-1"=451, "NFWMD-2"=394, "SFWMD-9"=834, "SJRWMD-1"=1216,
+               "SRWMD-1"=1400, "SWFWMD-4"=851)
```

Use the cat.analysis function to calculate estimates for the lake condition variables.

```
> Condition_Estimates_popsiz <- cat.analysis(sites, subpop, design, data.cat,
+      popsize=list(CombinedBasins=sum(framesize),
+                  Basin=as.list(framesize)))
```

Print the lake condition estimates for all basins combined.

```
> # Print the lake condition estimates for all basins combined
> print(Condition_Estimates_popsiz[c(1:4, 28:32),])
```

	Type	Subpopulation	Indicator	Category	NResp	Estimate.P	
1	CombinedBasins	All Basins	pHCat	(0,6]	78	42.915056	
2	CombinedBasins	All Basins	pHCat	(6,8]	82	50.396558	
3	CombinedBasins	All Basins	pHCat	(8,14]	11	6.688386	
4	CombinedBasins	All Basins	pHCat	Total	171	100.000000	
28	CombinedBasins	All Basins	ColiformCat	(0,5]	97	55.986933	
29	CombinedBasins	All Basins	ColiformCat	(5,50]	40	24.108155	
30	CombinedBasins	All Basins	ColiformCat	(50,500]	31	18.521502	
31	CombinedBasins	All Basins	ColiformCat	(500,5e+03]	2	1.383410	
32	CombinedBasins	All Basins	ColiformCat	Total	170	100.000000	
	StdError.P	LCB95Pct.P	UCB95Pct.P	Estimate.U	StdError.U	LCB95Pct.U	UCB95Pct.U
1	2.8530505	37.323179	48.506932	2208.40876	146.81798	1920.6508	2496.1667

2	3.0180108	44.481366	56.311751	2593.40689	155.30684	2289.0111	2897.8027
3	1.5603961	3.630066	9.746706	344.18435	80.29798	186.8032	501.5655
4	NA	NA	NA	5146.00000	NA	NA	NA
28	2.8761564	50.349770	61.624096	2881.08756	148.00701	2590.9992	3171.1760
29	3.0417644	18.146407	30.069904	1240.60567	156.52920	933.8141	1547.3973
30	2.4596628	13.700651	23.342352	953.11648	126.57425	705.0355	1201.1974
31	0.8268103	0.000000	3.003929	71.19029	42.54766	0.0000	154.5822
32	NA	NA	NA	5146.00000	NA	NA	NA

>

Use the write.csv function to write the condition estimates as a csv file.

```
> write.csv(Condition_Estimates_popsiz, file="Condition_Estimates_popsiz.csv")
```

6 Analysis of quantitative variables

The third analysis that will be examined is estimating the CDF and percentiles for quantitative variables. Two quantitative variables will be examined: (1) oxygen - dissolved oxygen value and (2) turbidity - turbidity value. The summary function is used to summarize the data structure of the two quantitative variables.

```
> summary(FL_lakes$Oxygen)
```

Summarize the data structure of the dissolved oxygen variable:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
0.830	4.880	6.870	6.468	8.310	12.480	759

```
> summary(FL_lakes$Turbidity)
```

Summarize the data structure of the turbidity variable:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
0.150	1.100	1.700	8.055	3.800	400.000	759

The cont.analysis function will be used to calculate estimates for quantitative variables. Input to the cont.analysis function is the same as input for the cat.analysis function except that the data frame containing response variables is named cont.data rather than cat.data. The sites, subpop, and design data frames created in the analysis of lake condition variables

can be reused for this analysis. The `data.cont` data frame contains the two quantitative variables: `DissolvedOxygen` and `Turbidity`. Variables `oxygen` and `turbidity` in the `FL_lakes` data frame are assigned to `DissolvedOxygen` and `Turbidity`, respectively. The `popsiz` argument is included in the call to `cont.analysis`.

Create the `data.cont` data frame.

```
> data.cont <- data.frame(siteID=FL_lakes$siteID,  
+                          DissolvedOxygen=FL_lakes$Oxygen,  
+                          Turbidity=FL_lakes$Turbidity)
```

Use the `cont.analysis` function to calculate CDF and percentile estimates for the quantitative variables.

```
> CDF_Estimates <- cont.analysis(sites, subpop, design, data.cont,  
+   popsize=list(CombinedBasins=sum(framesize),  
+               Basin=as.list(framesize)))
```

The object produced by `cont.analysis` is a list containing two objects: (1) `CDF`, a data frame containing the CDF estimates and (2) `Pct`, a data frame containing percentile estimates plus estimates of population values for mean, variance, and standard deviation. Format for the CDF data frame is analogous to the data frame produced by `cat.analysis`. For the CDF data frame, however, the fourth column is labeled `Value` and contains the value at which the CDF was evaluated. Unlike the data frames produced by the other analysis functions we have examined, the `Pct` data frame contains only nine columns since there is a single set of estimates rather than two sets of estimates. In addition, the fourth column is labeled `Statistic` and identifies either a percentile or the mean, variance, or standard deviation. Finally, since percentile estimates are obtained by inverting the CDF estimate, the percentile estimates do not have a standard error value associated with them.

Use the `write.csv` function to write the CDF estimates as a csv file.

```
> write.csv(CDF_Estimates$CDF, file="CDF_Estimates.csv")
```

The `cont.cdfplot` function in `spsurvey` can be used to produce a PDF file containing plots of the CDF estimates. The primary arguments to `cont.cdfplot` are a character string containing a name for the PDF file and the CDF data frame in the `CDF_Estimates` object. In addition, we make use of the `logx` argument to `cont.cdfplot`, which controls whether the CDF estimate is displayed using a logarithmic scale for the x-axis. The `logx` argument accepts two values: (1) `""`, do not use a logarithmic scale and (2) `"x"` - use a logarithmic scale. For this analysis, dissolved oxygen is displayed using the original response scale and turbidity is displayed using a logarithmic scale.

Produce a PDF file containing plots of the CDF estimates.

```
> cont.cdfplot("CDF_Estimates.pdf", CDF_Estimates$CDF, logx=c("", "x"))
>
```

Print the percentile estimates for dissolved oxygen for all basins combined.

```
> # Print the percentile estimates for dissolved oxygen for all basins combined
> print(CDF_Estimates$Pct[1:10,])
```

	Type	Subpopulation	Indicator	Statistic	NResp	Estimate
1	CombinedBasins	All Basins	DissolvedOxygen	5Pct	8	1.578342
2	CombinedBasins	All Basins	DissolvedOxygen	10Pct	17	2.285793
3	CombinedBasins	All Basins	DissolvedOxygen	25Pct	42	4.624982
4	CombinedBasins	All Basins	DissolvedOxygen	50Pct	83	6.809475
5	CombinedBasins	All Basins	DissolvedOxygen	75Pct	129	8.333775
6	CombinedBasins	All Basins	DissolvedOxygen	90Pct	153	9.428672
7	CombinedBasins	All Basins	DissolvedOxygen	95Pct	163	9.996570
8	CombinedBasins	All Basins	DissolvedOxygen	Mean	171	6.477253
9	CombinedBasins	All Basins	DissolvedOxygen	Variance	171	6.442747
10	CombinedBasins	All Basins	DissolvedOxygen	Std. Deviation	171	2.538257
		StdError	LCB95Pct	UCB95Pct		
1			0.9546438	2.003976		
2			1.7532592	3.384501		
3			4.1087503	5.506396		
4			6.5621691	7.142007		
5			7.9711324	8.553456		
6			9.0237184	9.884125		
7			9.7570792	10.457057		
8	0.148905597115604	6.1854029	6.769102			
9	0.561664353995088	5.3419051	7.543589			
10	0.110639786234289	2.3214067	2.755107			

```
>
```

Use the write.csv function to write the percentile estimates as a csv file.

```
> write.csv(CDF_Estimates$Pct, file="Percentile_Estimates.csv")
```

The cont.cdfctest function in spsurvey can be used to test for statistical difference between the CDFs from subpopulations. For this analysis we will test for statistical difference between the CDFs from the six basins. The cont.cdfctest function will test all possible pairs of basins. Arguments to cont.cdfctest are the same as arguments to cont.analysis. Since we are interested only in testing among basins, the subpop data frame is subsetting to include only the siteID and Basin variables. Note that the popsize argument was modified from prior examples to include only the entry for Basin.

```
> CDF_Tests <- cont.cdfctest(sites, subpop[,c(1,3)], design, data.cont,
+   popsize=list(Basin=as.list(framesize)))
```

The print function is used to display results for dissolved oxygen of the statistical tests for difference between CDFs for basins. The object produced by cont.cdfctest is a data frame containing eight columns. The first column (Type) identifies the population. The second and third columns (Subpopulation_1 and Subpopulation_2) identify the subpopulations. The fourth column (Indicator) identifies the response variable. Column five contains values of the test statistic. Six test statistics are available, and the default statistic is an F-distribution version of the Wald statistic, which is identified in the data frame as "Wald-F". The default statistic is used in this analysis. For further information about the test statistics see the help file for the cdf.test function in spsurvey, which includes a reference for the test for differences in CDFs. Columns six and seven (Degrees_of_Freedom_1 and Degrees_of_Freedom_2) provide the numerator and denominator degrees of freedom for the Wald test. The final column (p_Value) provides the p-value for the test.

```
> # Print results of the statistical tests for difference between CDFs from
> # basins for dissolved oxygen
> print(CDF_Tests, digits=3)
```

	Type	Subpopulation_1	Subpopulation_2	Indicator	Wald_F
1	Basin	NFWWMD-1	NFWWMD-2	DissolvedOxygen	3.1442
2	Basin	NFWWMD-1	SFWMD-9	DissolvedOxygen	4.4795
3	Basin	NFWWMD-1	SJRWMD-1	DissolvedOxygen	20.2917
4	Basin	NFWWMD-1	SRWMD-1	DissolvedOxygen	0.3048
5	Basin	NFWWMD-1	SWFWMD-4	DissolvedOxygen	10.6685
6	Basin	NFWWMD-2	SFWMD-9	DissolvedOxygen	2.6095
7	Basin	NFWWMD-2	SJRWMD-1	DissolvedOxygen	6.1606
8	Basin	NFWWMD-2	SRWMD-1	DissolvedOxygen	2.8194
9	Basin	NFWWMD-2	SWFWMD-4	DissolvedOxygen	3.8223
10	Basin	SFWMD-9	SJRWMD-1	DissolvedOxygen	12.7598
11	Basin	SFWMD-9	SRWMD-1	DissolvedOxygen	6.0877
12	Basin	SFWMD-9	SWFWMD-4	DissolvedOxygen	14.1179
13	Basin	SJRWMD-1	SRWMD-1	DissolvedOxygen	16.9733
14	Basin	SJRWMD-1	SWFWMD-4	DissolvedOxygen	5.2374
15	Basin	SRWMD-1	SWFWMD-4	DissolvedOxygen	6.4086
16	Basin	NFWWMD-1	NFWWMD-2	Turbidity	0.5751
17	Basin	NFWWMD-1	SFWMD-9	Turbidity	1.5886
18	Basin	NFWWMD-1	SJRWMD-1	Turbidity	1.1966
19	Basin	NFWWMD-1	SRWMD-1	Turbidity	1.8996
20	Basin	NFWWMD-1	SWFWMD-4	Turbidity	11.3469
21	Basin	NFWWMD-2	SFWMD-9	Turbidity	0.2456
22	Basin	NFWWMD-2	SJRWMD-1	Turbidity	0.2944
23	Basin	NFWWMD-2	SRWMD-1	Turbidity	0.4627

24	Basin	NFWMD-2	SWFWMD-4	Turbidity	11.0052
25	Basin	SFWMD-9	SJRWMD-1	Turbidity	0.3688
26	Basin	SFWMD-9	SRWMD-1	Turbidity	0.0753
27	Basin	SFWMD-9	SWFWMD-4	Turbidity	13.5140
28	Basin	SJRWMD-1	SRWMD-1	Turbidity	0.6625
29	Basin	SJRWMD-1	SWFWMD-4	Turbidity	17.2017
30	Basin	SRWMD-1	SWFWMD-4	Turbidity	9.7487

	Degrees_of_Freedom_1	Degrees_of_Freedom_2	p_Value
1	2	55	5.09e-02
2	2	57	1.56e-02
3	2	57	2.21e-07
4	2	54	7.39e-01
5	2	51	1.35e-04
6	2	55	8.27e-02
7	2	55	3.85e-03
8	2	52	6.88e-02
9	2	49	2.87e-02
10	2	57	2.63e-05
11	2	54	4.13e-03
12	2	51	1.32e-05
13	2	54	1.91e-06
14	2	51	8.54e-03
15	2	48	3.41e-03
16	2	55	5.66e-01
17	2	57	2.13e-01
18	2	57	3.10e-01
19	2	54	1.59e-01
20	2	51	8.39e-05
21	2	55	7.83e-01
22	2	55	7.46e-01
23	2	52	6.32e-01
24	2	49	1.13e-04
25	2	57	6.93e-01
26	2	54	9.28e-01
27	2	51	1.95e-05
28	2	54	5.20e-01
29	2	51	1.95e-06
30	2	48	2.80e-04

>

Use the write.csv function to write CDF test results as a csv file.

```
> # Write CDF test results as a csv file
> write.csv(CDF_Tests, file="CDF_Tests.csv", row.names=FALSE)
```


>