Analysis of a GRTS Survey Design for a Finite Resource

Thomas Kincaid

February 3, 2012

Contents

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 2.2 of the spsurvey package was loaded successfully.
```

2 Read the survey design and analytical variables data file

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 read the data file, which includes both survey design variables and analytical variables. The read.delim function is used to read the tab-delimited file 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.

Read the survey design and analytical variables data file

```
> # Read the data file and determine the number of rows in the file
> FL_lakes <- read.delim("FL_lakes.tab")
> 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 5.37 NWFWMD-1          Sampled    Target
2 FLW03414-0046 8636136 12886783 5.37 NWFWMD-1 Physical Barrier    Target
3 FLW03414-0062 8617834 12869126 5.37 NWFWMD-1        NonTarget NonTarget
4 FLW03414-0078 8673500 12883071 5.37 NWFWMD-1 Physical Barrier    Target
5 FLW03414-0086 8631884 12816428 5.37 NWFWMD-1        NonTarget NonTarget
6 FLW03414-0118 8607699 12856644 5.37 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 4.81 SWFWMD-4              Dry    Target
926 FLW03414-3886 8892406 12732977 4.81 SWFWMD-4          Sampled    Target
927 FLW03414-3894 8836528 12723056 4.81 SWFWMD-4              Dry    Target
928 FLW03414-3918 8923107 12725502 4.81 SWFWMD-4 Landowner Denial    Target
929 FLW03414-3926 8861298 12715824 4.81 SWFWMD-4              Dry    Target
930 FLW03414-3950 8888601 12715641 4.81 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. First, the levels function is used to extract the set of basin names, and the result is assigned to object basin. Next, the rainbow function is called to select a set of six colors, and the result is assigned to object cols. The plot function is then used to produce the basic figure, but plotting of sample points is suppressed. The for function is used to loop through the set of six basins and plot color-coded points for each basin using the points function. Finally, the legend function is used to add a legend to the figure, and the title function is used to create a figure title.

```
> basins <- levels(FL_lakes$basin)
> cols <- rainbow(6)
> plot(FL_lakes$xcoord, FL_lakes$ycoord, type="n", xlab="x-coordinate",
+      ylab="y-coordinate")
> for(i in 1:6) {
```

```
+    ind <- FL_lakes$basin == basins[i]
+    points(FL_lakes$xcoord[ind], FL_lakes$ycoord[ind], pch=20, cex=0.4,
+           col=cols[i])
+    }
> legend(x="topright", inset=0.05, legend=basins, pch=20, cex=1, col=cols)
> title("Plot of Florida Small Lake Sites 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. 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
```

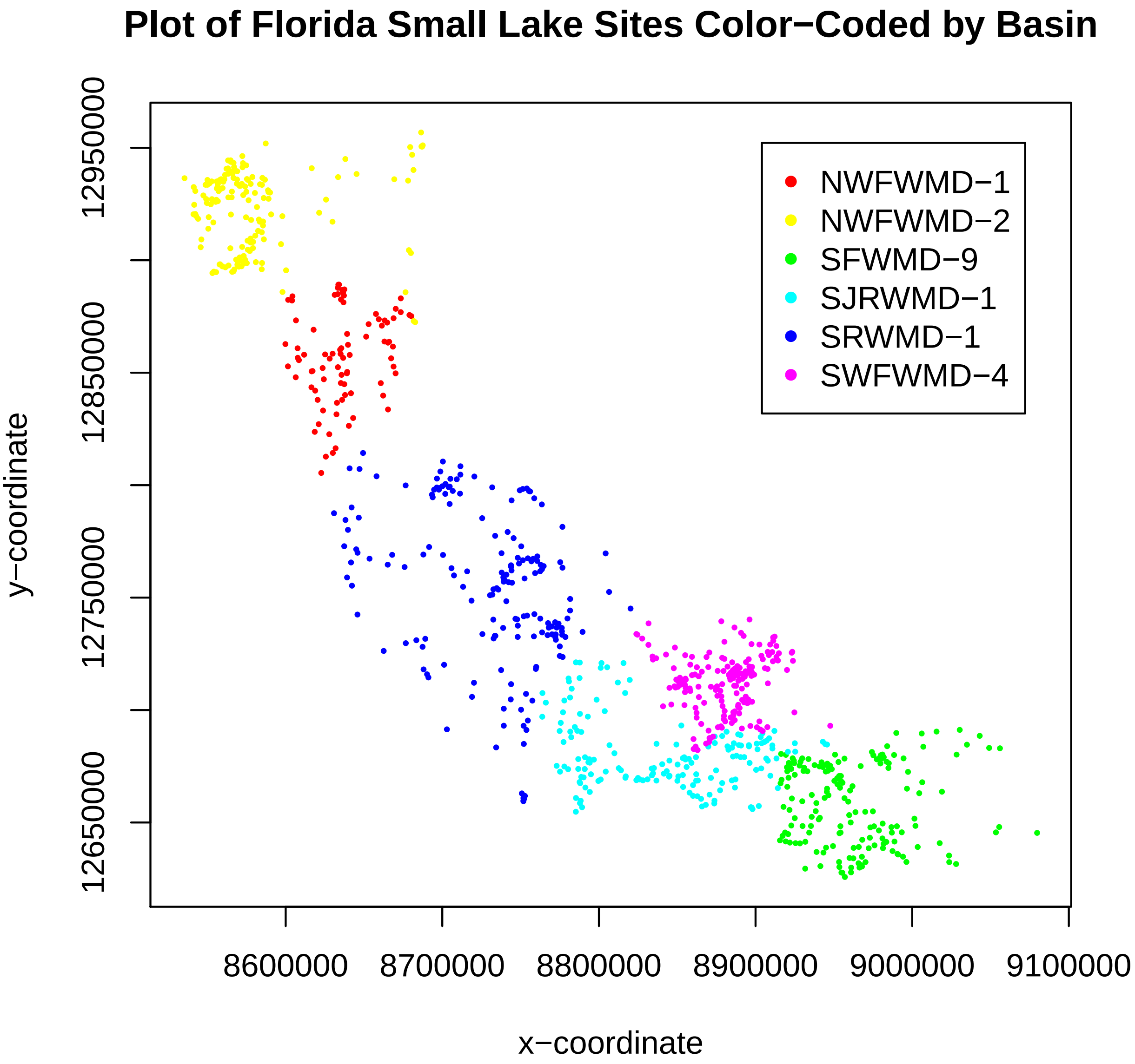

Figure 1: Florida Small Lake Sample Sites.

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).

```
> # 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
46 CombinedBasins    All Basins Target_NonTarget                 Target  613
47 CombinedBasins    All Basins Target_NonTarget                  Total  930
   Estimate.P StdError.P LCB95Pct.P UCB95Pct.P Estimate.U StdError.U LCB95Pct.U
1     23.0112      0.978      21.09     24.928    1184.16      50.20       1086
2     13.3274      0.991      11.39     15.269     685.83      50.98        586
3     36.9125      1.161      34.64     39.188    1899.52      60.32       1781
4      0.0942      0.085       0.00      0.261       4.85       4.37          0
5      8.4792      0.715       7.08      9.880     436.34      36.74        364
6     18.1755      1.034      16.15     20.203     935.31      53.21        831
7    100.0000      0.000     100.00    100.000    5146.00       9.28       5128
45    36.9125      1.161      34.64     39.188    1899.52      60.32       1781
46    63.0875      1.161      60.81     65.363    3246.48      59.23       3130
47   100.0000      0.000     100.00    100.000    5146.00       9.28       5128
   UCB95Pct.U
1      1282.6
2       785.8
3      2017.8
4        13.4
5       508.3
6      1039.6
7      5164.2
45     2017.8
46     3362.6
47     5164.2

>
```

The write.table 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.table(Extent_Estimates, file="Extent_Estimates.csv", sep=",",
+             row.names=FALSE)
```

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      42.92
2  CombinedBasins    All Basins       pHCat       (6,8]    82      50.40
3  CombinedBasins    All Basins       pHCat      (8,14]    11       6.69
4  CombinedBasins    All Basins       pHCat       Total   171     100.00
28 CombinedBasins    All Basins ColiformCat       (0,5]    97      55.99
29 CombinedBasins    All Basins ColiformCat      (5,50]    40      24.11
30 CombinedBasins    All Basins ColiformCat    (50,500]    31      18.52
31 CombinedBasins    All Basins ColiformCat (500,5e+03]     2       1.38
32 CombinedBasins    All Basins ColiformCat       Total   170     100.00
   StdError.P LCB95Pct.P UCB95Pct.P Estimate.U StdError.U LCB95Pct.U UCB95Pct.U
1       2.859      37.31      48.52      401.4      26.95      348.6      454.2
2       3.020      44.48      56.32      471.4      28.64      415.2      527.5
3       1.571       3.61       9.77       62.6      14.65       33.8       91.3
4       0.000     100.00     100.00      935.3       7.42      920.8      949.9
28      2.879      50.34      61.63      519.2      26.50      467.3      571.1
29      3.041      18.15      30.07      223.6      28.55      167.6      279.5
30      2.453      13.71      23.33      171.8      22.68      127.3      216.2
31      0.826       0.00       3.00       12.8       7.67        0.0       27.9
32      0.000     100.00     100.00      927.4       7.41      912.8      941.9

>
```

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

```
> write.table(Condition_Estimates, file="Condition_Estimates.csv", sep=",",
+             row.names=FALSE)
```

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 Estimates.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). The corresponding frame size value is 5,146. The popsize (population size) argument to cat.analysis provides a mechanism for forcing the Total category to equal a desired value. First, 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("NWFWMD-1"=451, "NWFWMD-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_popsize <- 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_popsize[c(1:4, 28:32),])
```

```
             Type Subpopulation   Indicator    Category NResp Estimate.P
1  CombinedBasins    All Basins       pHCat       (0,6]    78      42.92
2  CombinedBasins    All Basins       pHCat       (6,8]    82      50.40
3  CombinedBasins    All Basins       pHCat      (8,14]    11       6.69
4  CombinedBasins    All Basins       pHCat       Total   171     100.00
28 CombinedBasins    All Basins ColiformCat       (0,5]    97      55.99
29 CombinedBasins    All Basins ColiformCat      (5,50]    40      24.11
30 CombinedBasins    All Basins ColiformCat    (50,500]    31      18.52
31 CombinedBasins    All Basins ColiformCat (500,5e+03]     2       1.38
32 CombinedBasins    All Basins ColiformCat       Total   170     100.00
   StdError.P LCB95Pct.P UCB95Pct.P Estimate.U StdError.U LCB95Pct.U UCB95Pct.U
1       2.859      37.31      48.52     2208.4      147.1       1920       2497
2       3.020      44.48      56.32     2593.4      155.4       2289       2898
3       1.571       3.61       9.77      344.2       80.8        186        503
4          NA         NA         NA     5146.0         NA         NA         NA
```

```
28      2.879      50.34      61.63     2881.1      148.1       2591       3171
29      3.041      18.15      30.07     1240.6      156.5        934       1547
30      2.453      13.71      23.33      953.1      126.2        706       1200
31      0.826       0.00       3.00       71.2       42.5          0        154
32         NA         NA         NA     5146.0         NA         NA         NA

>
```

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

```
> write.table(Condition_Estimates_popsize, file="Condition_Estimates_popsize.csv",
+             sep=",", row.names=FALSE)
```

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 
      1       5       7       6       8      12     759 

> summary(FL_lakes$turbidity)

Summarize the data structure of the turbidity variable:

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
      0       1       2       8       4     400     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 popsize 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.table function to write the CDF estimates as a csv file.

```
> write.table(CDF_Estimates$CDF, file="CDF_Estimates.csv", sep=",",
+             row.names=FALSE)
```

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     1.58
2  CombinedBasins    All Basins DissolvedOxygen          10Pct    17     2.29
3  CombinedBasins    All Basins DissolvedOxygen          25Pct    42     4.62
4  CombinedBasins    All Basins DissolvedOxygen          50Pct    83     6.81
5  CombinedBasins    All Basins DissolvedOxygen          75Pct   129     8.33
6  CombinedBasins    All Basins DissolvedOxygen          90Pct   153     9.43
7  CombinedBasins    All Basins DissolvedOxygen          95Pct   163    10.00
21 CombinedBasins    All Basins DissolvedOxygen           Mean   171     6.48
31 CombinedBasins    All Basins DissolvedOxygen       Variance   171     6.44
41 CombinedBasins    All Basins DissolvedOxygen Std. Deviation   171     2.54
            StdError LCB95Pct UCB95Pct
1                       0.955     2.00
2                       1.753     3.38
3                       4.109     5.51
4                       6.562     7.14
5                       7.971     8.55
6                       9.022     9.89
7                       9.755    10.46
21 0.148923428773653    6.185     6.77
31 0.562255761580149    5.341     7.54
41 0.110756285008553    2.321     2.76

>
```

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

```
> write.table(CDF_Estimates$Pct, file="Percentile_Estimates.csv", sep=",",
+             row.names=FALSE)
```

The cont.cdftest 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.cdftest function will test all possible pairs of basins. Arguments to cont.cdftest are the same as arguments to cont.analysis. Since we are interested only in testing among basins, the subpop data frame is subsetted 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.cdftest(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.cdftest 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[1:15,])
```

```
    Type Subpopulation_1 Subpopulation_2       Indicator Wald_F
1  Basin        NWFWMD-1        NWFWMD-2 DissolvedOxygen  3.125
2  Basin        NWFWMD-1         SFWMD-9 DissolvedOxygen  4.487
3  Basin        NWFWMD-1        SJRWMD-1 DissolvedOxygen 20.300
4  Basin        NWFWMD-1         SRWMD-1 DissolvedOxygen  0.306
5  Basin        NWFWMD-1        SWFWMD-4 DissolvedOxygen 10.675
6  Basin        NWFWMD-2         SFWMD-9 DissolvedOxygen  2.619
7  Basin        NWFWMD-2        SJRWMD-1 DissolvedOxygen  6.072
8  Basin        NWFWMD-2         SRWMD-1 DissolvedOxygen  2.789
9  Basin        NWFWMD-2        SWFWMD-4 DissolvedOxygen  3.835
10 Basin         SFWMD-9        SJRWMD-1 DissolvedOxygen 12.829
11 Basin         SFWMD-9         SRWMD-1 DissolvedOxygen  6.079
12 Basin         SFWMD-9        SWFWMD-4 DissolvedOxygen 14.091
13 Basin        SJRWMD-1         SRWMD-1 DissolvedOxygen 16.915
14 Basin        SJRWMD-1        SWFWMD-4 DissolvedOxygen  5.246
15 Basin         SRWMD-1        SWFWMD-4 DissolvedOxygen  6.398
   Degrees_of_Freedom_1 Degrees_of_Freedom_2  p_Value
1                     2                   55 5.18e-02
2                     2                   57 1.55e-02
3                     2                   57 2.20e-07
4                     2                   54 7.38e-01
5                     2                   51 1.34e-04
6                     2                   55 8.19e-02
7                     2                   55 4.14e-03
8                     2                   52 7.07e-02
9                     2                   49 2.84e-02
10                    2                   57 2.51e-05
```

```
11                         2                       54 4.16e-03
12                         2                       51 1.34e-05
13                         2                       54 1.98e-06
14                         2                       51 8.48e-03
15                         2                       48 3.44e-03

>
```

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

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