Analysis of a GRTS Survey Design for a Linear Resource

Thomas Kincaid

August 16, 2016

Contents

1 Preliminaries

This document presents analysis of a GRTS survey design for a linear resource. The linear resource used in the analysis is streams in the Upper Wabash basin in Indiana. The analysis will include calculation of three types of population estimates: (1) estimation of proportion and size (length of streams) for site evaluation status categorical variables; (2) estimation of proportion and size for stream 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 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 IN_streams. The nrow function is used to determine the number of rows in the IN_streams 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 IN_streams 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(IN_streams)
> nr <- nrow(IN_streams)
>
```

Display the initial six lines in the data file.

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

```
      siteID  xcoord   ycoord       wgt Strahler_Cat           Status    TNT
1 INRB98-001 7574790 12556023 180.49965          1st Landowner_Denial Target
2 INRB98-002 7490591 12580092 180.49965          1st          Sampled Target
3 INRB98-003 7500191 12545177  57.70535          2nd          Sampled Target
4 INRB98-004 7543103 12557747  26.40031          4th Landowner_Denial Target
5 INRB98-005 7459317 12689535  29.59298          3rd          Sampled Target
6 INRB98-006 7515604 12649037  57.70535          2nd Physical_Barrier Target
  IBI_Score   IBI_Status QHEI_Score  QHEI_Status
1        NA         <NA>         NA         <NA>
2        50 Not_Impaired         48     Impaired
3        22     Impaired         65 Not_Impaired
4        NA         <NA>         NA         <NA>
5        38 Not_Impaired         31     Impaired
6        NA         <NA>         NA         <NA>
```

```
>
```

Display the final six lines in the data file.

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

```
        siteID  xcoord   ycoord       wgt Strahler_Cat           Status
95  INRB98-095 7503526 12628573  57.70535          2nd Landowner_Denial
96  INRB98-096 7496050 12662272 180.49965          1st        NonTarget
97  INRB98-097 7483750 12664829  29.59298          3rd   Chemistry_Only
98  INRB98-098 7496653 12634435 180.49965          1st        NonTarget
99  INRB98-099 7443579 12609765  26.40031          4th          Sampled
100 INRB98-100 7445529 12651391  26.40031          4th   Chemistry_Only
          TNT IBI_Score   IBI_Status QHEI_Score  QHEI_Status
95     Target        NA         <NA>         NA         <NA>
96  NonTarget        NA         <NA>         NA         <NA>
97     Target        NA         <NA>         NA         <NA>
98  NonTarget        NA         <NA>         NA         <NA>
99     Target        48 Not_Impaired         78 Not_Impaired
100    Target        NA         <NA>         NA         <NA>
```

```
>
```

The sample of streams in Indiana is displayed in Figure 1. The sample sites for each Strahler order are displayed using a unique color.

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 linear resource like streams, size refers to the length of streams 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 streams into seven evaluation categories and (2) TNT, which classifies streams 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.

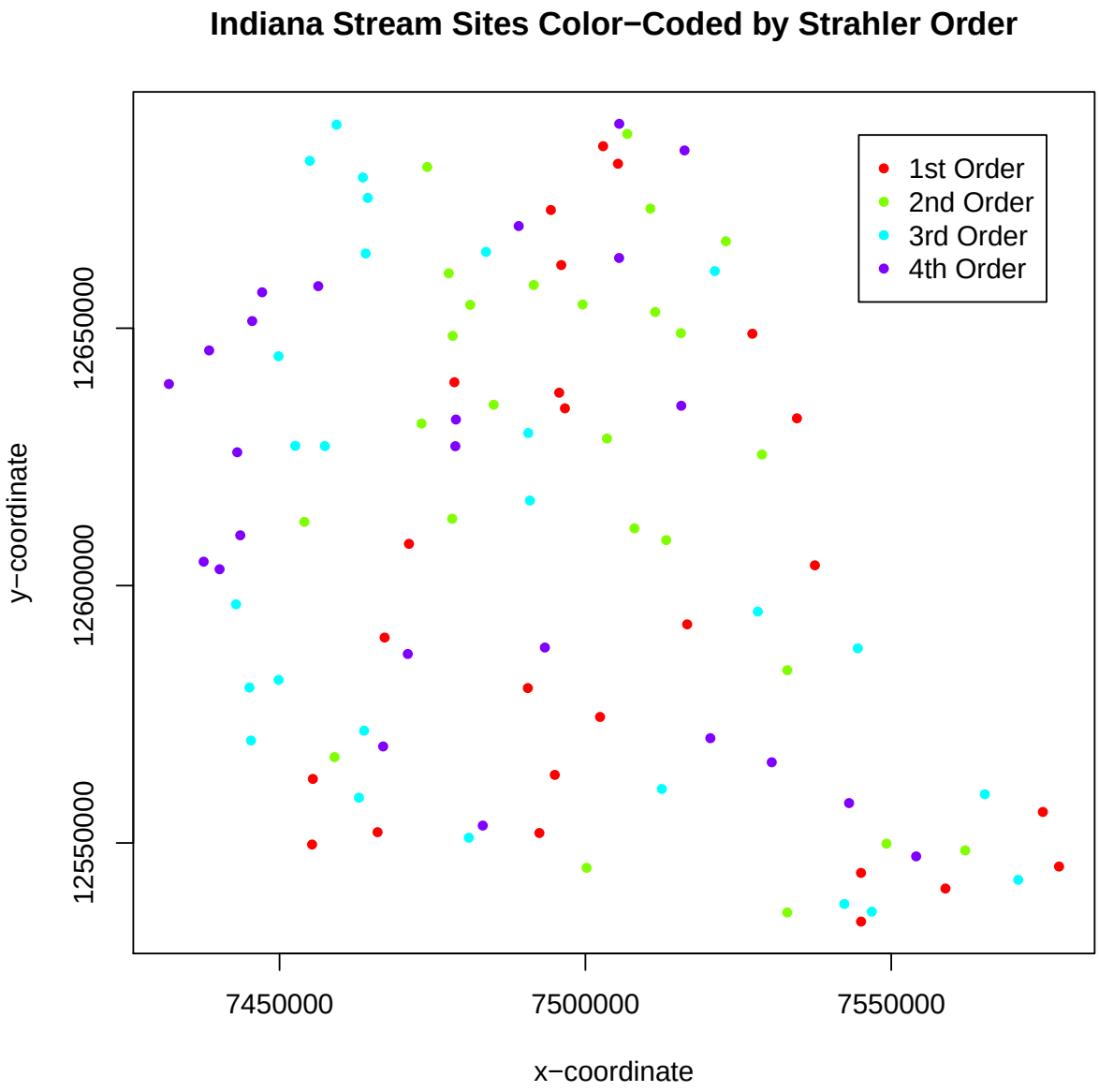

Figure 1: Location of stream sample sites in Indiana color-coded by Strahler order.

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

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

    Chemistry_Only   Landowner_Denial          NonTarget   Physical_Barrier
                14                 19                  9                  7
           Sampled Target_Not_Sampled            Unknown                Sum
                48                  2                  1                100

> addmargins(table(IN_streams$TNT))

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

NonTarget    Target       Sum
       10        90       100
```

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 IN_streams 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 IN_streams 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) Upper_Wabash - which will be used to calculate estimates for all of the sample sites combined, and (3) Strahler_Order - which will be used to calculate estimates for each Strahler order individually. The Strahler_order variable in the IN_streams data frame is assigned to the Strahler_Order 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 IN_streams 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 IN_streams 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=IN_streams$siteID,
+                     Use=rep(TRUE, nr))
```

Create the subpop data frame.

```
> subpop <- data.frame(siteID=IN_streams$siteID,
+                      Upper_Wabash=rep("Upper Wabash", nr),
+                      Strahler_Order=IN_streams$Strahler_Cat)
```

Create the design data frame.

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

Create the data.cat data frame.

```
> data.cat <- data.frame(siteID=IN_streams$siteID,
+                        Status=IN_streams$Status,
+                        Target_NonTarget=IN_streams$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(1:8, 32:34),])
```

```
           Type Subpopulation        Indicator           Category NResp
1  Upper_Wabash  Upper Wabash           Status     Chemistry_Only    14
2  Upper_Wabash  Upper Wabash           Status   Landowner_Denial    19
3  Upper_Wabash  Upper Wabash           Status          NonTarget     9
4  Upper_Wabash  Upper Wabash           Status   Physical_Barrier     7
5  Upper_Wabash  Upper Wabash           Status            Sampled    48
6  Upper_Wabash  Upper Wabash           Status Target_Not_Sampled     2
7  Upper_Wabash  Upper Wabash           Status            Unknown     1
8  Upper_Wabash  Upper Wabash           Status              Total   100
32 Upper_Wabash  Upper Wabash Target_NonTarget          NonTarget    10
33 Upper_Wabash  Upper Wabash Target_NonTarget             Target    90
34 Upper_Wabash  Upper Wabash Target_NonTarget              Total   100
    Estimate.P StdError.P  LCB95Pct.P  UCB95Pct.P Estimate.U StdError.U
1    6.5597397  1.6598843   3.3064261   9.8130532  482.67548  110.03523
2   17.8769326  3.7404140  10.5458559  25.2080092 1315.41150  285.35722
3   22.0775177  5.0281966  12.2224335  31.9326019 1624.49685  423.20639
4    5.5434713  2.4060864   0.8276286  10.2593140  407.89693  179.10621
5   46.4405214  5.0106571  36.6198139  56.2612289 3417.16319  427.23184
6    1.1430273  0.7450965   0.0000000   2.6033896   84.10566   54.27639
7    0.3587901  0.2951899   0.0000000   0.9373516   26.40031   21.63337
8  100.0000000  0.0000000 100.0000000 100.0000000 7358.14992  536.14393
32  22.4363077  5.0285302  12.5805696  32.2920459 1650.89716  423.75896
33  77.5636923  5.0285302  67.7079541  87.4194304 5707.25276  460.82638
34 100.0000000  0.0000000 100.0000000 100.0000000 7358.14992  536.14393
   LCB95Pct.U UCB95Pct.U
1   267.01038  698.34058
```

```
2   756.12163 1874.70137
3   795.02756 2453.96614
4    56.85522  758.93864
5  2579.80417 4254.52221
6     0.00000  190.48543
7     0.00000   68.80094
8  6307.32713 8408.97271
32  820.34487 2481.44945
33 4804.04965 6610.45587
34 6307.32713 8408.97271


>
```

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", sep=",",
+             row.names=FALSE)
```

4 Analysis of stream condition variables

The second analysis that will be examined is estimating resource proportion and size for stream condition variables. Two stream condition variables will be examined: (1) IBI_Status, which classifies streams by IBI (index of biotic integrity) status categories and (2) QHEI_Status, which classifies streams by QHEI (qualitative habitat evaluation index) status categories. The table and addmargins functions are used to create tables displaying the count for each level of the two stream condition variables.

```
> addmargins(table(IN_streams$IBI_Status))
```

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

```
    Impaired Not_Impaired          Sum
          12           36           48
```

```
> addmargins(table(IN_streams$QHEI_Status))
```

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

```
    Impaired Not_Impaired          Sum
          14           34           48
```

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 stream condition variables: IBI_Status and QHEI_Status. Variables IBI_Status and QHEI_Status in the IN_streams data frame are assigned to IBI_Status and QHEI_Status, respectively.

Create the sites data frame.

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

Create the data.cat data frame.

```
> data.cat <- data.frame(siteID=IN_streams$siteID,
+                        IBI_Status=IN_streams$IBI_Status,
+                        QHEI_Status=IN_streams$QHEI_Status)
```

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

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

Print the stream condition estimates for all sites combined.

```
> # Print the condition estimates for all basins combined
> print(Condition_Estimates[c(1:3, 16:18),])
```

```
           Type Subpopulation   Indicator     Category NResp Estimate.P
1  Upper_Wabash  Upper Wabash  IBI_Status     Impaired    12   27.66052
2  Upper_Wabash  Upper Wabash  IBI_Status Not_Impaired    36   72.33948
3  Upper_Wabash  Upper Wabash  IBI_Status        Total    48  100.00000
16 Upper_Wabash  Upper Wabash QHEI_Status     Impaired    14   40.90216
17 Upper_Wabash  Upper Wabash QHEI_Status Not_Impaired    34   59.09784
18 Upper_Wabash  Upper Wabash QHEI_Status        Total    48  100.00000
   StdError.P LCB95Pct.P UCB95Pct.P Estimate.U StdError.U LCB95Pct.U UCB95Pct.U
1    6.611920   14.70139   40.61964    945.205   247.3122   460.4819   1429.928
2    6.611920   59.38036   85.29861   2471.958   345.9131  1793.9811   3149.935
```

```
3    0.000000  100.00000  100.00000   3417.163   362.5003  2706.6756   4127.651
16   8.383366   24.47106   57.33325   1397.694   357.9031   696.2163   2099.171
17   8.383366   42.66675   75.52894   2019.470   305.3225  1421.0486   2617.891
18   0.000000  100.00000  100.00000   3417.163   362.5003  2706.6756   4127.651

>
```

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 stream 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., length of streams) 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 IBI_status variable, population "Upper_Wabash" and subpopulation "Upper Wabash" is 3,417 kilometers(rounded to a whole number). The corresponding frame size value is 7,358 kilometers. 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 asumption 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., Upper_Wabash and Strahler_Order for this analysis. The sum function applied to framesize is assigned to the Upper_Wabash entry in the popsize list. Recall that the Strahler order population type contains subpopulations, i.e., Strahler order categories. 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 Strahler_Order entry in the popsize list.

Assign frame size values.

```
> framesize <- c("1"=4514.450, "2"=1443.260, "3"=740.146, "4"=660.294)
```

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

```
> Condition_Estimates_popsize <- cat.analysis(sites, subpop, design, data.cat,
+   popsize=list(Upper_Wabash=sum(framesize),
+                Strahler_Order=as.list(framesize)))
```

Print the stream condition estimates for all sites combined.

```
> # Print the stream condition estimates for all sites combined
> print(Condition_Estimates_popsize[c(1:3, 16:18),])
```

```
           Type Subpopulation   Indicator     Category NResp Estimate.P
1  Upper_Wabash  Upper Wabash  IBI_Status     Impaired    12   27.66052
2  Upper_Wabash  Upper Wabash  IBI_Status Not_Impaired    36   72.33948
3  Upper_Wabash  Upper Wabash  IBI_Status        Total    48  100.00000
16 Upper_Wabash  Upper Wabash QHEI_Status     Impaired    14   40.90216
17 Upper_Wabash  Upper Wabash QHEI_Status Not_Impaired    34   59.09784
18 Upper_Wabash  Upper Wabash QHEI_Status        Total    48  100.00000
   StdError.P LCB95Pct.P UCB95Pct.P Estimate.U StdError.U LCB95Pct.U UCB95Pct.U
1    6.611920   14.70139   40.61964   2035.302   486.5150   1081.750   2988.854
2    6.611920   59.38036   85.29861   5322.848   486.5150   4369.296   6276.400
3          NA         NA         NA   7358.150         NA         NA         NA
16   8.383366   24.47106   57.33325   3009.642   616.8607   1800.618   4218.667
17   8.383366   42.66675   75.52894   4348.508   616.8607   3139.483   5557.532
18         NA         NA         NA   7358.150         NA         NA         NA

>
```

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

```
> write.csv(Condition_Estimates_popsize, file="Condition_Estimates_popsize.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) IBI_Score - IBI score and (2) QHEI_Score - QHEI score. The summary function is used to summarize the data structure of the two quantitative variables.

```
> summary(IN_streams$IBI_Score)
```

```
Summarize the data structure of the IBI score variable:
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00   31.50   36.00   36.12   42.00   54.00      52
```

```
> summary(IN_streams$QHEI_Score)
```

```
Summarize the data structure of the QHEI score variable:
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  25.00   47.75   60.00   59.65   71.25   87.00      52
```

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 stream condition variables can be reused for this analysis. The data.cont data frome contains the two quantitative variables: IBI_Score and QHEI_Score, which contain the numeric scores for the IBI and QHEI variables, respectively. Variables IBI_Score and QHEI_Score in the IN_streams data frame are assigned to IBI_Score and QHEI_Score, respectively. The popsize argument is included in the call to cont.analysis.

Create the data.cont data frame.

```
> data.cont <- data.frame(siteID=IN_streams$siteID,
+                         IBI_Score=IN_streams$IBI_Score,
+                         QHEI_Score=IN_streams$QHEI_Score)
```

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(Upper_Wabash=sum(framesize),
+                Strahler_Order=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.

Produce a PDF file containing plots of the CDF estimates.

```
> cont.cdfplot("CDF_Estimates.pdf", CDF_Estimates$CDF)
>
```

Print the percentile estimates for IBI score for all sites combined.

```
> # Print the percentile estimates for IBI score for all sites combined
> print(CDF_Estimates$Pct[1:10,])
```

```
           Type Subpopulation Indicator      Statistic NResp  Estimate
1  Upper_Wabash  Upper Wabash IBI_Score           5Pct     1   0.00000
2  Upper_Wabash  Upper Wabash IBI_Score          10Pct     2  23.39923
3  Upper_Wabash  Upper Wabash IBI_Score          25Pct     8  28.73106
4  Upper_Wabash  Upper Wabash IBI_Score          50Pct    23  34.24697
5  Upper_Wabash  Upper Wabash IBI_Score          75Pct    31  39.58683
6  Upper_Wabash  Upper Wabash IBI_Score          90Pct    41  44.24131
7  Upper_Wabash  Upper Wabash IBI_Score          95Pct    44  48.88966
8  Upper_Wabash  Upper Wabash IBI_Score           Mean    48  34.19264
9  Upper_Wabash  Upper Wabash IBI_Score       Variance    48 112.13090
10 Upper_Wabash  Upper Wabash IBI_Score Std. Deviation    48  10.58919
           StdError  LCB95Pct  UCB95Pct
1                    0.000000  24.63962
2                    0.000000  26.64929
3                   24.221557  32.17595
4                   31.384275  37.06088
5                   35.911571  43.88564
6                   40.800963  51.47035
7                   41.691545  54.00000
```

```
8   1.7410238506777 30.780300  37.60499
9  45.0419816500115 23.850234 200.41156
10 2.12679116946548  6.420754  14.75762

>
```

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.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 for the four Strahler order categories. The cont.cdftest function will test all possible pairs of Strahler order categories. Arguments to cont.cdftest are the same as arguments to cont.analysis. Since we are interested only in testing among Strahler order categories, the subpop data frame is subsetted to include only the siteID and Strahler_Order variables. Note that the popsize argument was modified from prior examples to include only the entry for Strahler_Order.

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

During execution of the program, a warning message was generated.  The warning
message is stored in a data frame named 'warn.df'.  Enter the following command
to view the warning message: warnprnt()
```

The print function is used to display results for IBI score of the statistical tests for difference between CDFs for Strahler order categories. 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
> # Strahler order categories for IBI score
> print(CDF_Tests, digits=2)
```

```
             Type Subpopulation_1 Subpopulation_2  Indicator Wald_F
1  Strahler_Order             1st             2nd  IBI_Score  0.350
2  Strahler_Order             1st             3rd  IBI_Score  0.314
3  Strahler_Order             1st             4th  IBI_Score  3.535
4  Strahler_Order             2nd             3rd  IBI_Score  0.065
5  Strahler_Order             2nd             4th  IBI_Score  3.554
6  Strahler_Order             3rd             4th  IBI_Score  2.670
7  Strahler_Order             1st             2nd QHEI_Score  0.989
8  Strahler_Order             1st             3rd QHEI_Score  1.633
9  Strahler_Order             1st             4th QHEI_Score  5.631
10 Strahler_Order             2nd             3rd QHEI_Score  0.406
11 Strahler_Order             2nd             4th QHEI_Score  3.510
12 Strahler_Order             3rd             4th QHEI_Score  1.968
   Degrees_of_Freedom_1 Degrees_of_Freedom_2 p_Value
1                     2                   21   0.709
2                     2                   23   0.733
3                     2                   17   0.052
4                     2                   25   0.938
5                     2                   19   0.049
6                     2                   21   0.093
7                     2                   21   0.389
8                     2                   23   0.217
9                     2                   17   0.013
10                    2                   25   0.671
11                    2                   19   0.050
12                    2                   21   0.165


>
```

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)
>
```