

Northern Great Plains Network

National Park Service



Data Analysis and Reporting

Stream Channel Monitoring

Please cite this as:

Thornbrugh, D. J., and J. S. Mills. 2024. Data analysis and reporting: stream channel monitoring. Northern Great Plains Network Standard Operation Procedure NPS/NGPN/SOP. National Park Service, Rapid City, SD.

Summary

This standard operating procedure (SOP) describes data analysis and reporting procedures for NGPN Stream Channel Monitoring (Thornbrugh et al. 2019). The procedures are adapted from the Southeast Coast Network (SECN) SOP Wadeable Stream Data Processing and Analysis—Version 1.0. and 2.2.14 (McDonald et al. 2018, 2022), SECN SOP Wadeable Stream Data Entry Final—Version 2.2.13 (McDonald 2022), SECN SOP Reporting_Final – Version 2.2.15, and from the Water Quality Monitoring Protocol for Wadeable Streams and Rivers in the Northern Great Plains Network—Version 1.0: SOP 6 Data Analysis and Reporting (Wilson and Wilson 2014).

Revision Log

Previous Version	Revision Date	Author	Changes Made	Reason for Change	New Version
1.0	12/20/2023	G Galles	Addition of data entry and QA	Providing guidance on additional aspects of data reporting	1.1
		A.Rosse	Incorporation of updated SECN procedures, add example figures and tables	Needs more detail	1.1

Updated Methods

- Detailed instructions for Data Entry and Reporting added, including where metrics are found and how they are calculated.
- List of Tables and Maps for reporting are added.

Introduction

NGPN will apply standard data analysis processes to characterize reach status, detect spatial or temporal changes, and detect trends in NGPN stream channels. The results of these analyses will be routinely published.

Procedures

Data Entry

Data collected at each park are tabulated physically on datasheet templates while in the field. These datasheet templates are as follows: Form 1 – Reach Characterization, Form 2 – Standard Transect Assessments, Form 3 – Detailed Cross-Sections, Form 4 – Reach Map, Form 5 – Post-Survey Report Form 6 – Photograph Record Sheet, and Form 7 – Discharge Measurements. After all data is collected on the physical forms, the original is scanned at the office, and entered into the data folder for each PARK and YEAR on the N drive. Data is then entered into computer-based reporting templates, generally .xlsx, that follow a similar structure and tabulate metrics such as averages, medians, minimums, and maximums of datasets. Data is checked as below in the QA section, then scanned again to the N drive. Once all data is complete, a copy of that year's data is zipped in a folder called PARK SCM 001 YEAR Final Data.zip and uploaded to IRMA.

After initial data collection is complete, copy the appropriate reporting templates, rename as below and enter the data.

Form 1 – Reach Characterization: Copy Form_1_Reporting_Template_vers_1.x_DATE.xlsx. Paste and rename as PARK_SCM_001_Form_1_Reporting_Data_DATE.xlsx.

- Entry.SurveyConditions: Most fields should be present on the Form 1 datasheet. The northing and easting should be a downstream point, preferably at T1 and entered in Web Mercator. Elevation should be specified as ellipsoidal (if collected via FieldMaps) or orthometric. The permanent average wetted width and permanent reach length should be the same every year after the site is established. The precipitation fields at the bottom should be from the [RAWS USA Climate Archive State Selection Map \(dri.edu\)](https://www.dri.edu/raus) and the site at or nearest the park selected.
- Entry.GCU: Enter Geomorphic Channel Units section from Form 1.
- Entry.PebbleCount: Enter Pebble Count data from Form 1.

- Entry.LWD: Enter Large Woody debris data here. Will also need average channel-full width and reach length.
- Entry.Densimeter_Bearing: Enter canopy cover and bearing here.
- Entry.Notes: Enter notes from Form 1 here.

Form 2 – Standard Transect Assessments: Copy Form_2_Reporting_Template_vers_1.x_DATE.xlsx. Paste and rename as PARK_SCM_001_Form_2_Reporting_Data_DATE.xlsx.

- All data entry is on Entry.Data and Entry.Notes tabs.

Form 3- Detailed Cross Sections: cross-check locations against PARK_PARK_PARK_Rebar_coordinates_DATE.xlsx and PARK_SCM_YEAR_allpoints_for_Ipad.xlsx compilation of transect locations and benchmark locations in Web Mercator. Note what side of the bank the primary benchmark is on. The cross-sections will be adjusted if working from the right bank instead of the left bank which is typical.

Forms 4-6: are not “entered”, merely scanned, and used for reference for future field work as a reference for creating maps.

Form 7- Discharge measurements can be entered into the Discharge_Reporting_Template_YEAR_.xlsx template. If referencing a gage to compare calculated and measured discharge, look up the associated gage on the USGS NWIS site at the same day and time.

Reporting templates are saved by 4 letter park code (e.g. THRO for Theodore Roosevelt National Park), data type code (e.g. SCM for Stream Channel Monitoring), form number, and date entered. An example of this would read: THRO_SCM_Form_1_20221014. The individual who entered the data initials and scans the sheets for later comparison. These physical forms are saved in the NGPN office.

Quality Analysis

Quality analysis (QA) is performed by a different team member who will compare the physical copy with the entered data on the accompanying reporting template to assure all metrics are correctly represented. If any reported data needs to be amended, the reporting template is copied, and all changes are made in red for comparison. The amended copy is saved similarly but with date and initials of the QA. After quality is assured, the form is then re-scanned and uploaded and the physical copy saved. Quality analysis also ensures the normality of the data, which prevents errant data from being included into any parametric testing.

Reach Level Measurements:

- Mean bank angle is calculated in the Form 2 Datasheet, Reporting Bank Table tab.
- Erosion is calculated by summing the number of transects with erosion and by using the purple box on the Entry.Data tab that calculates lineal bank feet of erosion and the percentage of bank experiencing erosion.
- Mean bank heights are calculated from the Reporting Bank Table tab.
- Bank Vegetation Cover and Bank Stability Index are also taken from the Reporting Bank Table tab.
- Cross-sectional area: This is calculated both for mean bankfull area and for mean channel-full area. After the cross-sectional data is collected on the total station (See SCM_SOP_UsingATotalStationtoConductAStreamSurvey), copy the job from the data collector to the laptop. Use Trimble Business Center to look at the data in Trimble and correct any errors. Export the data using the fields 'point', 'northing', 'easting', 'elevation', and 'code' from Trimble Business Center to excel. Paste into an .xlsx notebook. Name the first tab PARKYEARallprofiles. First, add a transect tab to separate the transects and longitudinal profiles. It is easier to deal with only one profile at a time. Next, within each tab order the points from left to right (or downstream to upstream if working on the longitudinal profile). Create and fill in the 'Point-C' field next to the point field so that it is ordered by transect number, dash then the point number which ascends as the points increase from left to right. The leftmost point should be TX-1 and the rightmost point if there are 100 points would be TX-100. These numbers may not be the same as the point field. See Figure 2 for an example, point-C field is in magenta box.

A	B	C	D	E	F	G	H	I	J	K
Point	Point-C	Northing	Easting	Ele_Ortho	Position	Transect	distancebetween	distance	Code	Note
T1-1	T1-1	175369.778	350053.491	1396.33	LBMOC	T1	0	0		
T1-2	T1-2	175369.121	350054.398	1396.08	LBMOC	T1	1.119954463	1.119954		
T1LBM	T1-3	175368.138	350055.631	1395.661	LOC	T1	1.576888709	2.696843	LBM	
T1-3	T1-4	175367.362	350056.871	1395.264	LOC	T1	1.46279732	4.15964		
T1-4	T1-5	175367.073	350057.278	1394.726	LOC	T1	0.49916931	4.65881		
T1-5	T1-6	175366.763	350057.605	1394.297	LOC	T1	0.450587394	5.109397		

Figure 2. Fields in MORU2023allprofiles.xlsx, T1 tab.

If the total station was set up over the benchmark, the benchmark point will have to be added into the correct position in the left to right order. T1-1, T1-2, etc. The blue box shows where the T1LBM point was inserted into the correct order in Figure 2. Note the Point-C field must remain in order from T1-1 to T1-6. Transects should be labeled T1(most downstream), T6 (midpoint), and T11(most upstream), with T1, T6, and T11 as the defaults unless the cross-sections were done at different transect numbers. LONG should be used for to label the water surface points used in the longitudinal profile.

Next, the ‘position’ and ‘note’ field will be created in the new workbook. The ‘code’ field has all the coded data including position: out of channel (OC), in-channel (CH) and beyond the benchmark (BMOC) for left (L) and right (R) sides as well as the special diagnostic points: benchmark (BM), channelfull (CF), floodplain margin (FPM), bankfull (BF), base of bank (BB), and water’s edge (EW). We will split this information into two fields by having the position data in the position field and the special diagnostic points in the code field. Every point should have a position, only the special diagnostic points have a code. Table 1 shows the positions and code that should be used.

Table 1. Position and Code for Detailed Transect points.

Position	Code
LBMOC	
LOC	LBM
LOC	LCF
LOC	LFPM
LOC	LBF
CH	LBB
CH	LEW
CH	REW
CH	RBB
ROC	RBF
ROC	RFPM
ROC	RCF
ROC	RBM
RBMOC	

If the points are in the correct order from left to right along the stream cross-section, or in the water surface (LONG) profile from the bottom of the reach (ws01) to the top of the reach (ws11), then two fields can be created in each tab, ‘distance_between’ and ‘distance’. The ‘distance_between’ formula is $\sqrt{(\text{point_2}_{\text{easting}} - \text{point_1}_{\text{easting}})^2 + (\text{point_2}_{\text{northing}} - \text{point_1}_{\text{northing}})^2}$. The ‘distance’ field refers to the distance from the left most point. T11-1’s distance as the leftmost point will be zero. T11-2, the next point to the right, equals the ‘distance_between’ of T11-2 + the previous ‘distance’ of T11-1. Continue filling out these fields for every point in the list.

From there, the points can be plotted and the diagnostic points visually checked to ensure that they are in the right order and position. Points can be recoded after examining graphically in excel and corrected (assigning a different point a code for instance) as in Figure 3.

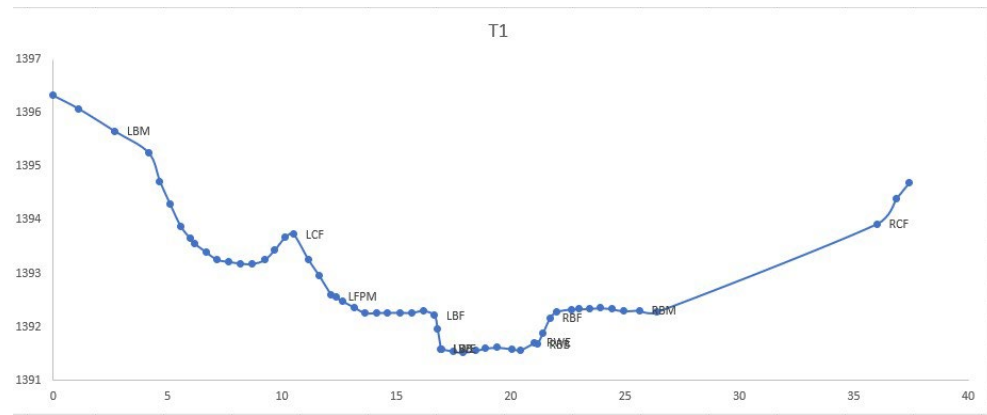


Figure 3. Profile of T1, MORU 2023 with diagnostic points labeled, distance(m) on the x -axis and elevation on the y-axis.

Then a .csv can be created from each transect's tab and put in a .csv named PARKYEARtx.csv which can then be run in R to generate some presentation quality graphs for the annual report and the table needed for the PARK_SCM_001_Detailed_Transect_YEAR_Date.xlsx reporting template. A copy of that R code follows in Appendix 1 and was written by SECN and minimally edited by NGPN. Figure 4 shows the detailed transect graph that can be generated in R.

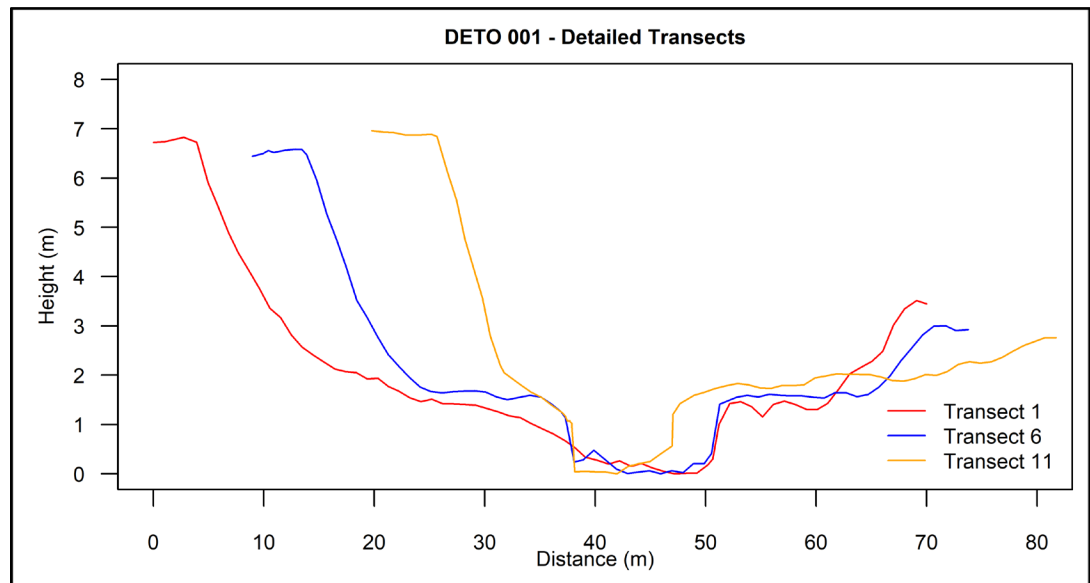


Figure 4. Detailed transects cross sections, plotting height above thalweg vs distance from the left benchmark (LBM). Axes were adjusted so that left bank-full begins at the same relative position for all three transects.

Figure 5 shows the longitudinal profile. The cross-sectional area tables are also an output of the R code.

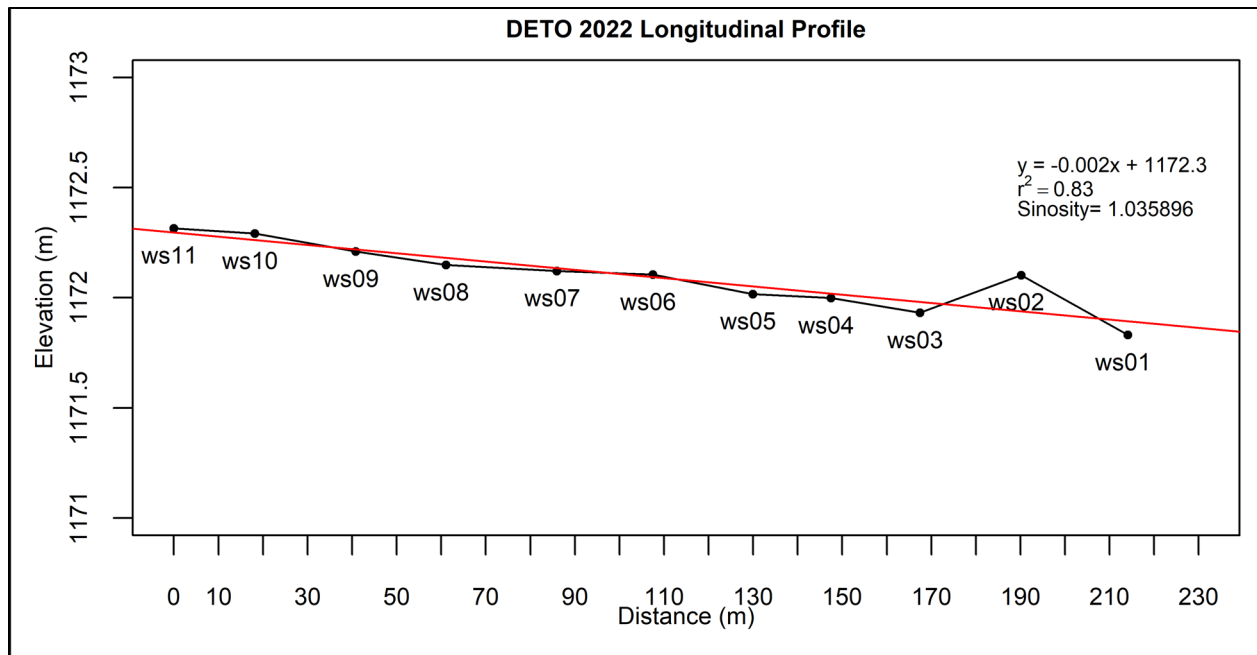


Figure 5. Elevational profile of DETO SCM 001. Slope is -.2%.

- Max channel migration: If multiple years of data are available, calculate the distances for each year from LBM or primary benchmark (left benchmark) to LBF (left bankfull) and LBM to RBF (right bank full). One can also calculate the distances from the primary benchmark to the left channel full (LCF) and right channel full (RCF) points. The table will be presented as movement in meters per year.

Maps can be included in the annual reporting. One map should show the reach area with the 11 transects and permanent benchmarks (Figure 6). Direction of flow should be indicated as well as major changes in elevation, abrupt changes in GCUs, or important channel features.



Figure 6. Layout of sampled reach of DETO stream channel monitoring site DETO SCM 001. Transects number 1-11 running from down to upstream.

Physical habitat measurements:

Status

Summary statistics of reach-averaged measurements are used to characterize the reach-scale status of the surveyed reaches. These include tables with descriptive statistics listing the minimum, maximum, and mean of stream channel sample metrics, as well as derived habitat measures in Table 2.

- Geomorphic channel units are summarized from Form 1, Reporting.GCU_Table4 tab.
- Median Particle Size is derived from Form 1, Reporting.PebbleCount_Table 5 tab.
- Large Woody Debris is derived from Form 1, Reporting.LWD_Table 6 and 7 tab.
- Particle Size and Embeddedness is from Form 2, Sediment tab.
- Percentage Fine Substrate is derived from Form 2, Sediment tab.
- Canopy Cover is derived from Form 1, Reporting.Canopy_Table 8 tab.

Table 2. Tables in Annual Report

Tables in Annual Report	
Executive Summary Table	Upland Land Use, Land Development Change, Mean Bank Angle, Erosion, Mean Bank Heights, Mean Bankfull Width, Bank Vegetation Cover, Bank Stability Index, Max Distance of Bank Migration, Cross-sectional Area, median particle size of Representative Riffle, Percentage fine substrate, Geomorphic units, and Canopy Cover
Table 1	Benchmark Locations
Table 2	Stream lengths of PARK SCM 001 watershed
Table 3	Watershed Characteristics
Table 4	Flow Characteristics derived from StreamStats
Table 5	National Land Cover Data
Table 6	Detailed transects measurements from Total Station Data
Table 7	Thalwegs
Table 8	Widths
Table 9	Bank Stability Index Scoring categories
Table 10	Reach scale bank characteristics
Table 11	Transect scale bank characteristics
Table 12	Transect scale geomorphic channel units
Table 13	Calculated geomorphic channel metrics
Table 14	Sediment diameters from pebble count
Table 15	Dominant particle size and percent fines in reach
Table 16	Canopy Cover in Reach

Reporting

NGPN will publish approved data and periodic reports following current NPS IMD reporting requirements (Gallo 2018). Most reports will be published and available to the public in the NPS Natural Resource Publications Series (NRPS; <https://www.nps.gov/im/publication-series.htm>). Some results will also be submitted as articles to Park Science, another NPS journal, or to peer-reviewed, scientific journals. Regardless of the outlet, all reports and publications will adhere to NPS peer-review and publication standards and will be catalogued in the NPS Data Store (<https://irma.nps.gov/DataStore/>).

NGPN staff may also present data, analytical methods, and results at NGPN Technical Committee meetings, or at regional and national conferences.

Annual Data Summaries

NGPN will publish brief summaries of all approved data within a year of sampling, either as resource briefs for routine updates or through the NRPS Natural Resource Data series for notable or unusual results. These park-by-park data summaries will supplement published data but will not analyze trends or include significant interpretive discussion. In years that NGPN publishes trend reports, data summaries may be omitted if the trend reports include the same summary data.

Change Detection and Trend Reporting

Change Detection

Change detection metrics will be compiled after enough data has been collected to allow for change over time to be adequately assessed and reported. This will typically occur after several cycles of data have been collected, which occurs every 3 years. Change detection tests for significant change (in means or variance) in surveyed variables between two sampling intervals (i.e., from time 1 to time 2 as well as from time 1 to time n). Changes in channel widths, bank heights, and bed sediment between sampling intervals at each reach are evaluated using a paired t-test if the variable is normally distributed, the nonparametric Wilcoxon signed rank test if it is non-normal, or a Fisher exact test (for proportion data). Like the Student's paired t-test, the Wilcoxon signed rank test determines whether the means of two paired samples are significantly different while not requiring the data to be normally distributed (Crawley 2005). Any statistically significant changes will be further evaluated to determine if it is an ecologically significant change.

The F-test (which assumes normality) or Levene's test (which does not require normality) can be used to compare the variance in thalweg position and depth between sampling intervals. Thalweg depth among other variable can fluctuate dramatically over the course of a year based on flow, so interannual changes which can be large in addition to intra-annual changes which can be large must be estimated. In general, the difference in variance is more informative for habitat quality than the difference in means. Due to limited data, change detection may take many years of data collection and nonparametric methods to detect changes will be preferred. Previously collected data from Rust 2006 which was meant to establish some baseline data for NGPN streams, or NGPN SCCRM 2018, (unpublished) could be used if data collection and methods are similar.

Trends

Trend analysis will use linear mixed models, linear or multiple regression, or preferably the Mann-Kendall test at the reach and transect scale. For non-normal data, trend analysis will use a nonparametric technique such as the Mann-Kendall test. The Mann-Kendall test uses Kendall's Tau to determine if there is a statistically significant monotonic (upward or downward) trend in the data (Helsel and Hirsch 2002). Outside data such as from Rust 2006 or NGPN SCRRM 2018, unpublished could be used if data collection and methods are very similar.

Groups of three or more reaches of similar stream size may be compared using linear mixed models or the regional Kendall test to find similar trends within groups of sites. Due to the limited number of stream channel sites, and only one site per park, cross-park comparisons might not be useful. Rust 2006 conducted baseline habitat data at NGPN parks with many metrics that are similar or identical to the ones she collected. She divided the streams into large rivers or small streams. Any comparison should take place within these groups and possibly considering other groupings such as substrate and ecoregion which vary widely across NGPN.

Table 3. Large Rivers in NGPN

Large Rivers	Dominant Substrate	Ecoregion Level 3
Niobrara River at NIOB	Bedrock	Nebraska Sand Hills
Knife River at KNRI	Sand	Northwestern Great Plains
Little Missouri R. at THRO	Sand	Northwestern Great Plains
Belle Fourche at DETO	Cobble	Middle Rockies
Laramie River at FOLA	Gravel	Western High Plains
Bow Creek at MNRR- not measured by Rust 2006	Sand	Western Corn Belt Plains

Table 4. Streams in NGPN

Small Streams	Dominant Substrate	Ecoregion Level 3
Niobrara River at AGFO	Sand	Western High Plains
Highland Creek at WICA	Gravel	Middle Rockies
Grizzly Bear Creek at MORU	Gravel/Bedrock	Middle Rockies

If trends are detected in stream data, land cover, climate, hydrology, and other potential influences, these trends can be explored and incorporated into mixed effect and regression models as covariates. For example, extended drought, flooding, or dam behavior may have influenced the reach.

Trend Reports

After four sampling cycles (or twelve years), and every three sampling cycles thereafter, NGPN will publish trend reports for each park through the NRPS Natural Resource Report series (per schedule in Gallo (2018)). These reports show the overall status and changes or trends detected in selected parameters and may make recommendations for assessment thresholds or management actions. These analyses will incorporate other information (e.g., climate data, streamflow) known to influence stream channel characteristics in the Northern Great Plains. Reports will highlight any detected trends to help park managers and other stakeholders understand them.

Interim or out-of-cycle reports may be published if new management practices are implemented or a resource is believed to be changing rapidly.

Other Publications

Routine analyses will be supplemented with periodic, in-depth evaluation of all stream channel characteristics and water quality data in concert with information from other protocols (e.g. NGPN water quality, climate data) or other sources (parks, partners, etc.). These synthesis reports will seek to understand the reasons behind the patterns of status and trends of stream channel attributes, usually within a single park or a group of similar parks (e.g., small parks in the Black Hills).

They will report results of statistical techniques incorporating data from other vital signs monitoring (e.g. water quality/stream flow) to discern relationships among the monitored parameters. When such combined analyses are not possible, trends or patterns discerned from analyzing data from other vital signs or from other sources will be used to generate hypotheses regarding the reasons for observed

changes in stream channel characteristics. For example, land use change and precipitation patterns in the watershed could be incorporated into a statistical model evaluating trends in stream channel characteristics, water quality, stream flow, and climate variables (Wilson et al. 2014, Nelson 2018).

Literature Cited

- Crawley, M. J. 2005. *Statistics: an introduction using R*. John Wiley & Sons, West Sussex, England.
- Gallo, K. April 23, 2018. Data Analysis and Reporting Requirements [Memorandum]. Inventory and Monitoring Division, National Park Service, Fort Collins, Colorado. Available at: <https://irma.nps.gov/DataStore/Reference/Profile/2252934>
- Helsel, D.R. and R. M. Hirsch. 2002. *Statistical Methods in Water Resources Techniques of Water Resources Investigations*, Book 4, chapter A3. U.S. Geological Survey.
- Kaufmann, P.R., P. Levine, E.G. Robinson, C. Seeliger, and D.V. Peck. 1999. Quantifying physical habitat in wadeable streams. EPA620/R-99/003. Environmental Monitoring and Assessment Program, U.S. Environmental Protection Agency, Corvallis, OR.
- McDonald, J. M., E. N. Starkey, and M. B. Gregory. 2018. Wadeable stream data processing and analysis—Version 1.0. Southeast Coast Network Standard Operating Procedure NPS/SECN/SOP—2.2.14. National Park Service, Athens, Georgia.
- Nelson, L. 2018. EnvironmentalSettingToolkit: Station Finder and Data Requestor for ACIS climate data in support of the IMD Environmental Setting protocol. R package version 2.0.
- R Core Team. 2022. R: A language and environment for statistical computing. version 4.2.1.. <http://www.R-project.org>. Vienna, Austria.
- Rust JD. 2006. Establishing baseline data for aquatic resources in national parks of the Northern Great Plains Network.
- Thornbrugh, D. J., I. W. Ashton, A. R. Jarding, J. S. Mills, and K. Paintner-Green. 2019. Protocol implementation plan for stream channel monitoring in the Northern Great Plains Network: narrative version 1.0. Natural Resource Report NPS/NGPN/NRR—2019/1918. National Park Service, Fort Collins, Colorado.
- Wilson, M. H., B. L. Rowe, R. A. Gitzen, S. K. Wilson, and K. J. Paintner-Green. 2014. Water quality monitoring protocol for wadeable streams and rivers in the Northern Great Plains Network: Narrative version 1.0. Natural Resources Report NPS/NGPN/NRR—2014/868. National Park Service, Fort Collins, Colorado. Available at: <https://irma.nps.gov/DataStore/Reference/Profile/2216799>. (accessed 02/24/2017).

Appendix A: R code for Detailed Transects Cross-sections

```
install.packages("dplyr")
library(dplyr)
park<-"MORU"
site<-"001"
day<-"20231010"
side<-c("L","L","L")
#input median bankfull height
# this is used to calculate bankfull dimensions
# set working directory
setwd("C:/Users/aninerosse/OneDrive - DOI/stream channel/2023/MORU/MORUprofiles")
t1<-read.csv("MORU2023t1.csv",header=TRUE) #transect1
t6<-read.csv("MORU2023t6.csv", header=TRUE) #transect6
t11<-read.csv("MORU2023t11.csv", header=TRUE) #transect11
#calculate distance to benchmark
d2bm<-function(transect)
{
  #Determine northing, easting, elevation of river left benchmark
  bm<-data.frame (
    Northing=transect[which(transect[, 'Code']=="LBM"), 'Northing'],
    Easting=transect[which(transect[, 'Code']=="LBM"), 'Easting'],
    Elevation=transect[which(transect[, 'Code']=="LBM"), 'Elevation'])

  #Calculate the distance to the river left benchmark for the points to the left of the benchmark
  LoBM<-transect[which(transect[, 'Position']=="LBMOC"),]
  LoBM[, 'd2BM']<-sqrt((LoBM[, 'Northing']-bm[, 'Northing'])^2+(LoBM[, 'Easting']-
bm[, 'Easting'])^2)

  #Calculate the distance to the river left benchmark for the points to the right of the benchmark
  RoBM<-transect[which(transect[, 'Position']!="LBMOC"),]
  RoBM[, 'd2BM']<-sqrt((RoBM[, 'Northing']-bm[, 'Northing'])^2+(RoBM[, 'Easting']-
bm[, 'Easting'])^2)

  #combine the points
  rbind(LoBM,RoBM)
}
# Add a column with distance to benchmark
transect.1<-d2bm(t1)
transect.6<-d2bm(t6)
transect.11<-d2bm(t11)

#Remove 'Position' column
transect.1$Position<-NULL
transect.6$Position<-NULL
transect.11$Position<-NULL

# Convert d2BM and elevation to Distances (from furthest left point and Heights(above thalweg)
```

```

transect.1$Distance<-abs(min(transect.1$d2BM))+transect.1$d2BM
transect.1$Height<-transect.1$Elevation-min(transect.1$Elevation)

transect.6$Distance<-abs(min(transect.6$d2BM))+transect.6$d2BM
transect.6$Height<-transect.6$Elevation-min(transect.6$Elevation)

transect.11$Distance<-abs(min(transect.11$d2BM))+transect.11$d2BM
transect.11$Height<-transect.11$Elevation-min(transect.11$Elevation)
# Determine Thalweg elevation, and the position and elevation of the channel full and bankfull bank
tops

bank.position<-function(transect)
{
  thalweg.y<-min(transect[, 'Height'])
  Lcf.x<-transect[which(transect[, 'Code']=="LCF"), 'Distance']
  Lcf.y<-transect[which(transect[, 'Code']=="LCF"), 'Height']
  Rcf.x<-transect[which(transect[, 'Code']=="RCF"), 'Distance']
  Rcf.y<-transect[which(transect[, 'Code']=="RCF"), 'Height']
  Lbf.x<-transect[which(transect[, 'Code']=="LBF"), 'Distance']
  Lbf.y<-transect[which(transect[, 'Code']=="LBF"), 'Height']
  Rbf.x<-transect[which(transect[, 'Code']=="RBF"), 'Distance']
  Rbf.y<-transect[which(transect[, 'Code']=="RBF"), 'Height']

  data.frame(
    thalweg=thalweg.y,
    LCFx=Lcf.x,
    LCFy=Lcf.y,
    RCFx=Rcf.x,
    RCFy=Rcf.y,
    LBFx=if(length(Lbf.x)==0){9999}else{Lbf.x},
    LBFy=if(length(Lbf.y)==0){9999}else{Lbf.y},
    RBFx=if(length(Rbf.x)==0){9999}else{Rbf.x},
    RBFy=if(length(Rbf.y)==0){9999}else{Rbf.y})
}
t1.bp<-bank.position(transect.1)
t6.bp<-bank.position(transect.6)
t11.bp<-bank.position(transect.11)

# Subset transect points based on
# elevation (below minimum Cf elevation) &
# d2BM (right of LCf & left of RCf)

Cf.XS <- function(transect,bp)
{
  Cf.channel <- subset( transect, transect[, 'Height'] <=
    min(bp[,c('LCFy', 'RCFy')]) &
    transect[, 'Distance'] >=bp[, 'LCFx'] &
    transect[, 'Distance'] <=bp[, 'RCFx'])
}

```

```
# Check to see if the left-most point Height is equal to Cf Height # if it is not, then an unknown
point needs to be determined
```

```
Cf.channel <- if(Cf.channel[1,'Height'] < max(Cf.channel[, 'Height']))
```

```
{
# Calculate the (x,y) of last (potentially unknown) point
# Determine Channel-full Height and x,y of right-most point on Cf XS
Cf.y <- max(Cf.channel[, 'Height']) # Channel-full Height
CfL.x      <- Cf.channel[1, 'Distance']
CfL.y <- Cf.channel[1, 'Height']
```

```
# Extract the x,y of the point that is left of the left most point (LoL)
```

```
LoL <- transect[which(transect[, 'Distance'] == CfL.x) - 1,]
```

```
# Difference in Height to unknown point
```

```
D2U.y <- LoL[, 'Height'] - Cf.y # LoL Height minus Cf Height
```

```
# Difference in Height to left-most point
```

```
D2L.y <- LoL[, 'Height'] - CfL.y
```

```
# Distance between LoL and left-most point
```

```
D2L.x <- abs(LoL[, 'Distance'] - CfL.x)
```

```
# Distance between RoR and unknown point
```

```
D2U.x <- D2U.y * tan(atan(D2L.x / D2L.y))
```

```
# Distance for unknown point
```

```
Lx.u <- LoL[, 'Distance'] + D2U.x
```

```
unL <- data.frame( Name = 9999,
```

```
Code = "unL",
```

```
Northing = 9999,
```

```
Easting = 9999,
```

```
Elevation = 9999,
```

```
d2BM = 9999,
```

```
Distance = Lx.u,
```

```
Height = Cf.y)
```

```
data_bind_rowsL <- bind_rows(unL, Cf.channel)
```

```
}
```

```
else {Cf.channel}
```

```
# Check to see if the right-most point Height is equal to Cf Height
```

```
# if it is not, then an unknown point needs to be determined
```

```
Cf.channel <- if(Cf.channel[nrow(Cf.channel), 'Height'] <
max(Cf.channel[, 'Height']))
```

```
{
```

```
# Calculate the (x,y) of last (potentially unknown) point
```

```
# Determine Channel-full Height and x,y of right-most point on Cf XS
```

```
Cf.y <- max(Cf.channel[, 'Height']) # Channel-full Height
```

```
CfR.x      <- Cf.channel[nrow(Cf.channel), 'Distance']
```

```
CfR.y <- Cf.channel[nrow(Cf.channel), 'Height']
```

```
# Extract the x,y of the point that is right of the right most point (RoR)
```

```
RoR <- transect[which(transect[, 'Distance'] == CfR.x) + 1,]
```

```

# Difference in Height to unknown point
D2U.y <- RoR[, 'Height'] - Cf.y # RoR Height minus Cf Height

# Difference in Height to right-most point
D2R.y <- RoR[, 'Height'] - CfR.y

# Distance between RoR and right-most point
D2R.x <- abs(RoR[, 'Distance'] - CfR.x)

# Distance between RoR and unknown point
D2U.x <- D2U.y*tan(atan(D2R.x/D2R.y))

# Distance for unknown point
Rx.u <- RoR[, 'Distance'] - D2U.x

unR <- data.frame(
  Name = 9999,
  Code = "unR",
  Northing = 9999,
  Easting = 9999,
  Elevation = 9999,
  d2BM = 9999,
  Distance = Rx.u,
  Height = Cf.y)

data_bind_rowsR<-bind_rows(Cf.channel,unR)
}
else {Cf.channel}
}
t1.Cf.xs <-Cf.XS(transect.1, t1.bp)
t6.Cf.xs<-Cf.XS(transect.6, t6.bp)
t11.Cf.xs<-Cf.XS(transect.11, t11.bp)

aBf.depth<-.44 # median bankfull ht. from standard transects

# Extract bankfull XS points
Bf.XS <- function(transect, bp)
{
  Bf.y <- if(min(bp[,c('LBFy', 'RBFy')])<9999)
  {
    min(bp[,c('LBFy', 'RBFy')])
  } else {bp[, 'thalweg'] + aBf.depth}

  Bf.channel <- subset(transect, transect[, 'Height'] <= Bf.y      &
    transect[, 'Distance'] > bp[, 'LCFx'] &
    transect[, 'Distance'] < bp[, 'RCFx'])

```

```
# Check to see if the left-most point Height is equal to Bf Height # if it is not, then an unknown
point needs to be determined
```

```
Bf.channel <- if(Bf.channel[1,'Height'] < Bf.y)
```

```
{
```

```
# Calculate the (x,y) of last (potentially unknown) point
```

```
# Determine x,y of right-most point on Bf XS
```

```
BfL.x <- Bf.channel[1,'Distance']
```

```
BfL.y <- Bf.channel[1,'Height']
```

```
# Extract the x,y of the point that is left of the left most point (LoL)
```

```
LoL <- transect[which(transect[, 'Distance'] == BfL.x) - 1,]
```

```
# Difference in Height to unknown point
```

```
D2U.y <- LoL[, 'Height'] - Bf.y # LoL Height minus Bf Height
```

```
# Difference in Height to left-most point
```

```
D2L.y <- LoL[, 'Height'] - BfL.y
```

```
# Distance between LoL and left-most point
```

```
D2L.x <- abs(LoL[, 'Distance'] - BfL.x)
```

```
# Distance between RoR and unknown point
```

```
D2U.x <- D2U.y * tan(atan(D2L.x / D2L.y))
```

```
# Distance for unknown point
```

```
Lx.u <- LoL[, 'Distance'] + D2U.x
```

```
unL <- data.frame( Name = 9999, Code = "unL",
```

```
  Northing = 9999,
```

```
  Easting = 9999,
```

```
  Elevation = 9999,
```

```
  d2BM = 9999,
```

```
  Distance = Lx.u,
```

```
  Height = Bf.y)
```

```
data_bind_rowsBFL <- bind_rows(unL, Bf.channel)
```

```
}
```

```
else {Bf.channel}
```

```
# Check to see if the right-most point Height is equal to Bf Height
```

```
# if it is not, then an unknown point needs to be determined
```

```
Bf.channel <- if(Bf.channel[nrow(Bf.channel), 'Height'] < Bf.y)
```

```
{
```

```
# Calculate the (x,y) of last (potentially unknown) point
```

```
# Determine x,y of right-most point on Bf XS
```

```
BfR.x <- Bf.channel[nrow(Bf.channel), 'Distance']
```

```
BfR.y <- Bf.channel[nrow(Bf.channel), 'Height']
```

```
# Extract the x,y of the point that is right of the right most point (RoR)
```

```

RoR <- transect[which(transect[, 'Distance'] == BfR.x) + 1,]

# Difference in Height to unknown point
D2U.y <- RoR[, 'Height'] - Bf.y
# RoR Height minus Bf Height
# Difference in Height to right-most point

D2R.y <- RoR[, 'Height'] - BfR.y

# Distance between RoR and right-most point
D2R.x <- abs(RoR[, 'Distance'] - BfR.x)

# Distance between RoR and unknown point
D2U.x <- D2U.y * tan(atan(D2R.x / D2R.y))

# Distance for unknown point
Rx.u <- RoR[, 'Distance'] - D2U.x

unR <- data.frame( Name = 9999,
                  Code = "unR",
                  Northing = 9999,
                  Easting = 9999,
                  Elevation = 9999,
                  d2BM = 9999,
                  Distance = Rx.u,
                  Height = Bf.y)

data_bind_rowsBFR <- bind_rows(Bf.channel, unR)
}
else { Bf.channel }
}

t1.Bf.xs <- Bf.XS(transect.1, t1.bp)
t6.Bf.xs <- Bf.XS(transect.6, t6.bp)
t11.Bf.xs <- Bf.XS(transect.11, t11.bp)

#####
##### CALCULATE XS AREA, WETTED PERIMETER, and HYDRAULIC RADIUS #####
### ## #

#####
### BE SURE TO CHECK CF and BF transects so no extra points were included!!!!

# FUNCTION - CALCULATE AREAS and PERIMETER
xs.calculate <- function(xs)

```

```

{
  n <- nrow(xs)-1

  a.1 <- numeric(n)
  a.2 <- numeric(n)

  perimeter <- numeric(n)
  depth <- numeric(n)

  for (i in 1:n)
  {
    if(xs[i,'Height'] > xs[i+1,'Height'])
    {
      a.1[i] <- (max(xs[, 'Height']) - xs[i, 'Height']) *
        (abs(xs[i, 'Distance'] - xs[i+1, 'Distance']))
      a.2[i] <- (abs(xs[i, 'Height'] - xs[i+1, 'Height']) *
        (abs(xs[i, 'Distance'] - xs[i+1, 'Distance'])))/2
      perimeter[i] <- sqrt(abs(xs[i, 'Height'] - xs[i+1, 'Height'])^2 + abs(xs[i, 'Distance'] -
xs[i+1, 'Distance'])^2)
      #depth[i] <- max(xs[, 'Height']) - xs[i, 'Height']
    } else {a.1[i] <- (max(xs[, 'Height']) - xs[i+1, 'Height']) *
      (abs(xs[i, 'Distance'] - xs[i+1, 'Distance']))
      a.2[i] <- (abs(xs[i, 'Height'] - xs[i+1, 'Height']) *
        (abs(xs[i, 'Distance'] - xs[i+1, 'Distance'])))/2
      perimeter[i] <- sqrt(abs(xs[i, 'Height'] - xs[i+1, 'Height'])^2
        + abs(xs[i, 'Distance'] - xs[i+1, 'Distance'])^2)
      #depth[i] <- max(xs[, 'Height']) - xs[i, 'Height']
    }
  }

  xs.area <- sum(a.1) + sum(a.2)
  xs.perimeter <- sum(perimeter)
  xs.hydraulic.radius <- xs.area/xs.perimeter
  xs.width <- max(xs[, 'Distance']) - min(xs[, 'Distance'])
  xs.depth <- max(xs[, 'Height']) - min(xs[, 'Height'])
  #xs.depth.avg <- sum(depth)/n
  xs.wd.ratio <- xs.width/xs.depth

  data.frame( Area = xs.area,
    Perimeter = xs.perimeter,
    Hydraulic_Radius = xs.hydraulic.radius,
    Width = xs.width,
    Depth = xs.depth,
    #Avg_Depth = xs.depth.avg,
    #Area_Avg = xs.width * xs.depth.avg,
    WD_ratio = xs.wd.ratio)
}

# Function to calculate XS metrics and combine
xs.combined <- function(bf, cf)

```

```

{
  Bf <- xs.calculate(bf)
  Cf <- xs.calculate(cf)
  ratio <- Cf/Bf

  output <- rbind(Bf, Cf, ratio)
  rownames(output) <- c("Bankfull", "Channel-full", "Ratio")
  return(output)
}

write.csv(xs.combined(t1.Bf.xs,t1.Cf.xs),"T1_calculated.csv")
write.csv(xs.combined(t6.Bf.xs,t6.Cf.xs), "T6_calculated.csv")
write.csv(xs.combined(t11.Bf.xs,t11.Cf.xs), "T11_calculated.csv")

#####
##### PLOT Individual Cross-Sections

# Determine axis' range
x.max <- max(c(transect.1$Distance, transect.6$Distance, transect.11$Distance))
y.max <- max(c(transect.1$Height, transect.6$Height, transect.11$Height))

# Specify axis labels (Will be dependent on maxima)
x.axis <- seq(0,90,10)
x.labels <- c(0,10,20,30,40,50,60,70,80,90)

y.axis <- seq(0,9,1)
y.labels <- c(0,1,2,3,4,5,6,7,8,9)

# Create objects with diagnostic points
pts4plots <- function(transect, Cf, Bf)
{
  data.frame(
    LBMx = transect[which(transect[, 'Code'] == "LBM"), 'Distance'],
    LBMy = transect[which(transect[, 'Code'] == "LBM"), 'Height'],
    RBMx = transect[which(transect[, 'Code'] == "RBM"), 'Distance'],
    RBMy = transect[which(transect[, 'Code'] == "RBM"), 'Height'],
    thalx = Cf[which(Cf[, 'Height'] == 0), 'Distance'],
    thaly = 0,
    LCfx = transect[which(transect[, 'Code'] == "LCF"), 'Distance'],
    LCfy = transect[which(transect[, 'Code'] == "LCF"), 'Height'],
    RCfx = transect[which(transect[, 'Code'] == "RCF"), 'Distance'],
    RCfy = transect[which(transect[, 'Code'] == "RCF"), 'Height'],
    LBfx = if(Bf[1, 'Code'] == "unL"){Bf[1, 'Distance']}
    } else {transect[which(transect[, 'Code'] == "LBF"), 'Distance']},
    LBfy = if(Bf[1, 'Code'] == "unL"){Bf[1, 'Height']}
    } else {transect[which(transect[, 'Code'] == "LBF"), 'Height']},
    RBfx = if(Bf[nrow(Bf), 'Code'] == "unR"){Bf[nrow(Bf), 'Distance']}
    } else {transect[which(transect[, 'Code'] == "RBF"), 'Distance']},
    RBfy = if(Bf[nrow(Bf), 'Code'] == "unR"){Bf[nrow(Bf), 'Height']}
  )
}

```

```

    } else {transect[which(transect[, 'Code'] == "RBF"), 'Height']}))
  }

t1.p4p <- pts4plots(transect.1, t1.Cf.xs, t1.Bf.xs)
t6.p4p <- pts4plots(transect.6, t6.Cf.xs, t6.Bf.xs)
t11.p4p <- pts4plots(transect.11, t11.Cf.xs, t11.Bf.xs)

# PLOT Detailed Transect 1
png(filename = paste(park, site, day, "T1_plot.png", sep = "_"),
     units = "in",
     width = 6.5,
     height = 3.5,
     res = 600)
par(mai=c(0.6,0.64,0.33,0.1))
plot( transect.1[, 'Distance'], transect.1[, 'Height'], pch = 20,
     cex = 0.75,
     cex.lab = 0.75, xlim = c(0, x.max),
     ylim = c(0, y.max), xaxt = "n",
     yaxt = "n", ann = FALSE)
lines(transect.1[, 'Distance'], transect.1[, 'Height'])

axis(1, at = x.axis, labels = x.labels, cex.axis = 0.75)
mtext("Distance (m)", 1, line = 1.5, cex = 0.75)

axis(2, at = y.axis, labels = y.labels, las=2, cex.axis = 0.75)
mtext("Height (m)", 2, line = 1.75, cex = 0.75)

x.Cf <- c(t1.Cf.xs[, 'Distance'], sort(t1.Cf.xs[, 'Distance'], decreasing = TRUE))
y.Cf <- c(t1.Cf.xs[, 'Height'], rep(max(t1.Cf.xs[, 'Height']), nrow(t1.Cf.xs)))
polygon(x.Cf, y.Cf, col= rgb(0,0,255,50,maxColorValue=255))

x.Bf <- c(t1.Bf.xs[, 'Distance'], sort(t1.Bf.xs[, 'Distance'], decreasing = TRUE))
y.Bf <- c(t1.Bf.xs[, 'Height'], rep(max(t1.Bf.xs[, 'Height']), nrow(t1.Bf.xs)))
polygon(x.Bf, y.Bf, col= rgb(0,0,255,100,maxColorValue=255))

points(t1.p4p[, 'LBMx'], t1.p4p[, 'LBMy'], pch = 24, bg = "RED")
points(t1.p4p[, 'RBMx'], t1.p4p[, 'RBMy'], pch = 24, bg = "RED")
points(t1.p4p[, 'thalx'], t1.p4p[, 'thaly'], pch = 21, bg = "SKYBLUE")
points(t1.p4p[, 'LCfx'], t1.p4p[, 'LCfy'], pch = 21, bg = "ORANGE")
points(t1.p4p[, 'RCfx'], t1.p4p[, 'RCfy'], pch = 21, bg = "ORANGE")
points(t1.p4p[, 'LBfx'], t1.p4p[, 'LBfy'], pch = 21, bg = "GREEN")
points(t1.p4p[, 'RBfx'], t1.p4p[, 'RBfy'], pch = 21, bg = "GREEN")

title(paste(park, site, "- Detailed Transect 1", sep = " "), cex.main
     = 0.75)

legend("bottomright", inset = 0.01,

```

```

c("Benchmark", "Thalweg", "Channel-full", "Bankfull"), pch = c(24, 21, 21, 21),
pt.bg = c("RED", "SKYBLUE", "ORANGE", "GREEN"),
cex = 0.75, bty = "n")

box(which = "outer")
dev.off()

# PLOT Detailed Transect 3
png(filename = paste(park, site, day, "T6_plot.png", sep = "_"), units = "in",
width = 6.5,
height = 3.5,
res = 600)
par(mai=c(0.6,0.64,0.33,0.1))
plot( transect.6[, 'Distance'], transect.6[, 'Height'], pch = 20,
cex = 0.75,
cex.lab = 0.75, xlim = c(0, x.max),
ylim = c(0, y.max), xaxt = "n",
yaxt = "n", ann = FALSE)
lines(transect.6[, 'Distance'], transect.6[, 'Height'])

axis(1, at = x.axis, labels = x.labels, cex.axis = 0.75)
mtext("Distance (m)", 1, line = 1.5, cex = 0.75)

axis(2, at = y.axis, labels = y.labels, las=2, cex.axis = 0.75)
mtext("Height (m)", 2, line = 1.75, cex = 0.75)

x.Cf <- c(t6.Cf.xs[, 'Distance'], sort(t6.Cf.xs[, 'Distance'], decreasing = TRUE))
y.Cf <- c(t6.Cf.xs[, 'Height'], rep(max(t6.Cf.xs[, 'Height']), nrow(t6.Cf.xs)))
polygon(x.Cf, y.Cf, col= rgb(0,0,255,50,maxColorValue=255))

x.Bf <- c(t6.Bf.xs[, 'Distance'], sort(t6.Bf.xs[, 'Distance'], decreasing = TRUE))
y.Bf <- c(t6.Bf.xs[, 'Height'], rep(max(t6.Bf.xs[, 'Height']), nrow(t6.Bf.xs)))
polygon(x.Bf, y.Bf, col= rgb(0,0,255,100,maxColorValue=255))

points(t6.p4p[, 'LBMx'], t6.p4p[, 'LBMy'], pch = 24, bg = "RED")
points(t6.p4p[, 'RBMx'], t6.p4p[, 'RBMy'], pch = 24, bg = "RED")
points(t6.p4p[, 'thalx'], t6.p4p[, 'thaly'], pch = 21, bg = "SKYBLUE")
points(t6.p4p[, 'LCfx'], t6.p4p[, 'LCfy'], pch = 21, bg = "ORANGE")
points(t6.p4p[, 'RCfx'], t6.p4p[, 'RCfy'], pch = 21, bg = "ORANGE")
points(t6.p4p[, 'LBfx'], t6.p4p[, 'LBfy'], pch = 21, bg = "GREEN")
points(t6.p4p[, 'RBfx'], t6.p4p[, 'RBfy'], pch = 21, bg = "GREEN")

title(paste(park, site, "- Detailed Transect 3", sep = " "), cex.main
= 0.75)

legend("bottomright", inset = 0.01,
c("Benchmark", "Thalweg", "Channel-full", "Bankfull"), pch = c(24, 21, 21, 21),
pt.bg = c("RED", "SKYBLUE", "ORANGE", "GREEN"),

```

```

    cex = 0.75, bty = "n")

box(which = "outer")
dev.off()

# PLOT Detailed Transect 11
png(filename = paste(park, site, day, "T11_plot.png", sep = "_"), units = "in",
    width = 6.5,
    height = 3.5,
    res = 600)

par(mai=c(0.6,0.64,0.33,0.1))

plot( transect.11[, 'Distance'], transect.11[, 'Height'], pch = 20,
    cex = 0.75,
    cex.lab = 0.75, xlim = c(0, x.max),
    ylim = c(0, y.max), xaxt = "n",
    yaxt = "n", ann = FALSE)
lines(transect.11[, 'Distance'], transect.11[, 'Height'])

axis(1, at = x.axis, labels = x.labels, cex.axis = 0.75)
mtext("Distance (m)", 1, line = 1.5, cex = 0.75)

axis(2, at = y.axis, labels = y.labels, las=2, cex.axis = 0.75)
mtext("Height (m)", 2, line = 1.75, cex = 0.75)

x.Cf <- c(t11.Cf.xs[, 'Distance'], sort(t11.Cf.xs[, 'Distance'], decreasing = TRUE))
y.Cf <- c(t11.Cf.xs[, 'Height'], rep(max(t11.Cf.xs[, 'Height']), nrow(t11.Cf.xs)))
polygon(x.Cf, y.Cf, col= rgb(0,0,255,50,maxColorValue=255))

x.Bf <- c(t11.Bf.xs[, 'Distance'], sort(t11.Bf.xs[, 'Distance'], decreasing = TRUE))
y.Bf <- c(t11.Bf.xs[, 'Height'], rep(max(t11.Bf.xs[, 'Height']), nrow(t11.Bf.xs)))
polygon(x.Bf, y.Bf, col= rgb(0,0,255,100,maxColorValue=255))

points(t11.p4p[, 'LBMx'], t11.p4p[, 'LBMy'], pch = 24, bg = "RED")
points(t11.p4p[, 'RBMx'], t11.p4p[, 'RBMy'], pch = 24, bg = "RED")
points(t11.p4p[, 'thalx'], t11.p4p[, 'thaly'], pch = 21, bg = "SKYBLUE")
points(t11.p4p[, 'LCfx'], t11.p4p[, 'LCfy'], pch = 21, bg = "ORANGE")
points(t11.p4p[, 'RCfx'], t11.p4p[, 'RCfy'], pch = 21, bg = "ORANGE")
points(t11.p4p[, 'LBfx'], t11.p4p[, 'LBfy'], pch = 21, bg = "GREEN")
points(t11.p4p[, 'RBfx'], t11.p4p[, 'RBfy'], pch = 21, bg = "GREEN")

title(paste(park, site, "- Detailed Transect 11", sep = " "), cex.main
    = 0.75)

legend("bottomright",

    inset = 0.01,

```

```

c("Benchmark", "Thalweg", "Channel-full", "Bankfull"), pch = c(24, 21, 21, 21),
pt.bg = c("RED", "SKYBLUE", "ORANGE", "GREEN"),
cex = 0.75, bty = "n")

box(which = "outer")
dev.off() #####
##### PLOT Combined Cross-Sections

# Determine the max distance from origin for LCF
t1.lcf <- transect.1[which(transect.1[, 'Code'] == 'LBF'), 'Distance']
t6.lcf <- transect.6[which(transect.6[, 'Code'] == 'LBF'), 'Distance']
t11.lcf <- transect.11[which(transect.11[, 'Code'] == 'LBF'), 'Distance']
lcf.max <- max(t1.lcf, t6.lcf, t11.lcf)

# Determine the LCF correction factor
# ATR: I changed LCF to LBF so banks would line up better
t1.correction <- lcf.max - t1.lcf
t6.correction <- lcf.max - t6.lcf
t11.correction <- lcf.max - t11.lcf

# Standardize the LCF locations
transect.1[, 'new_Dist'] <- transect.1[, 'Distance'] + t1.correction
transect.6[, 'new_Dist'] <- transect.6[, 'Distance'] + t6.correction
transect.11[, 'new_Dist'] <- transect.11[, 'Distance'] + t11.correction

# Define axis' range (standardize across sites in report)
x.max <- 50
y.max <- 10

# Specify axis labels (Will be dependent on maxima)
x.axis <- seq(0, 50, 10)
x.labels <- c(0, 10, 20, 30, 40, 50)

y.axis <- seq(0, 10, 1)
y.labels <- c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

# PLOT All Transects
png(filename = paste(park, site, "_", day, "_all_plot.png", sep = ""),
     units = "in",

     width = 6.5,
     height = 3.5,
     res = 600)
par(mai = c(0.6, 0.64, 0.33, 0.1))
plot( transect.1[, 'new_Dist'], transect.1[, 'Height'], type = "n",
      cex = 0.75,
      cex.lab = 0.75, xlim = c(0, x.max),
      ylim = c(0, y.max), xaxt = "n",
      yaxt = "n", ann = FALSE)

```

```

lines(transect.1[, 'new_Dist'], transect.1[, 'Height'], col = 'red')

lines(transect.6[, 'new_Dist'], transect.6[, 'Height'], col = 'blue')

lines(transect.11[, 'new_Dist'], transect.11[, 'Height'], col = 'orange')

axis(1, at = x.axis, labels = x.labels, cex.axis = 0.75)
mtext("Distance (m)", 1, line = 1.5, cex = 0.75)

axis(2, at = y.axis, labels = y.labels, las=2, cex.axis = 0.75)
mtext("Height (m)", 2, line = 1.75, cex = 0.75)

title(paste(park, site, "- Detailed Transects", sep = " "), cex.main
      = 0.75)

legend("bottomright", inset = 0.01,
      c("Transect 1", "Transect 6", "Transect 11"), lty = c(1, 1, 1),
      col = c("red", "blue", "orange"), cex = 0.75,
      bty = "n")
box(which = "outer")
dev.off()

```

Appendix B: R code for Longitudinal Profiles

```
# Calculate Channel Slope and Sinuosity
setwd("C:/Users/aninerosse/OneDrive - DOI/stream channel/2023/MORU/MORUprofiles")
# Identify PARK and Sites
#park <- c("MORU")
#site <- c("001")
#day <- c("2023_10_16")
long <- read.csv("MORU_SCM_001_2023_LONG.csv")
#dbp<-c(rep(0,11))
#for(i in 1:11)
#{
# dbp[i]<-sqrt((long[i+1,'northing']-long[i,'northing'])^2+(long[i+1,'easting']-long[i,'easting'])^2)]
#}
#long$DbP<-dbp
# calculate distance along profile
#distance<-c(rep(0,11))
#for(i in 2:11)
#{distance[i]<-long[i,'DbP']+distance[i-1]}
#}
#long$distance<-distance
#Determine height of point above profile minimum elevation
#height<-c(rep(0,11))
#for(i in 1:11)
#{
# height[i]<-long[i,'elevation']-min(long[, 'elevation'])
#}
#long$height<-height
#determine profile trend
trend<-lm(long$elevation~long$distance_m)
r2.round<-round(summary(trend)$r.squared,2)
intercept<-coef(trend)[1]
slope<-coef(trend)[2]
equation<-paste("y = ",round(slope,4),"x + ",round(intercept,2),sep= "")
r2.label<-bquote(r^2 ==.(r2.round))

#Specify axis labels (dependent on maxima)
x.axis<-seq(0,180,10)
y.axis<-seq(1390,1396,1)

#Plot Longitudinal Profile
png(filename = "MORU 2023 Longitudinal Profile.png",
     units = "in",
     width = 6.5,
     height = 3.5,
     res = 600)
par(mai = c(.7, .64, .33, .1))
```

```

plot(
  long[, 'distance_m'],
  long[, 'elevation'],
  pch=20,
  cex=.75,
  cex.lab=.75,
  xaxt="n",
  yaxt="n",
  xlim=c(0,180),
  ylim=c(1390,1396),

  ann=FALSE)

title("MORU 2023 Longitudinal Profile",cex.main=.75)
axis(1,at=x.axis,labels=x.axis,cex.axis=.75)
mtext("Distance (m)",1,line=1.5,cex=.75)
text(long[, 'distance_m']-1, long[, 'elevation'], labels=long[, 'description'], cex=.75, pos=1)
axis(2, at=y.axis, labels=y.axis, las=0, cex.axis=.75)
mtext("Elevation (m)",2,line=1.9,cex=.75)
lines(long[, 'distance_m'],long[, 'elevation'])
abline(trend, col="red")

text(130,1395,r2.label,pos=4,cex=.65)
text(130,1394.6,equation, pos=4, cex=.65)
text(130, 1394.2, "Sinosity= 1.033482" , pos=4, cex=.65)

box(which="outer")
dev.off()

# Calculate Sinuosity
# Determine distance between upstream and downstream points
dt2b <- sqrt((long[1,'northing']-long[nrow(long),'northing'])^2+(long[1,'easting']-
long[nrow(long),'easting'])^2)

sinu <- long[1,'distance_m']/dt2b
dt2b
sinu
trend

```