UDWR Creel Survey

Run Creel Analysis

Author: Taylor R. Stewart
Date Written: 2025-Mar-13
Last Modified: 2025-Mar-13

Introduction

Creel surveys are a method used in fisheries management to collect data on recreational fishing activities, including catch rates, species composition, fishing effort, and angler demographics. These surveys typically involve interviews with anglers at fishing sites, where they provide information about their catch, effort, and fishing experience. Some surveys also include direct biological sampling of harvested fish to assess size, age, and health. The collected data is then analyzed to estimate total catch, fishing pressure, and trends over time, which helps in making informed decisions about fishery regulations, stocking programs, and conservation efforts. Statistical techniques are often used to extrapolate results from sampled anglers to the broader fishing population.

The DWR commissioned the development of a creel analysis in the late 1990s. This analysis was written in SAS and used until 2024, when it was converted to R, an open-source programming language widely used in data science. This vignette is intended to provide informational guidance for DWR biologists on the division's creel analysis and to minimize potential issues.

This vignette is broken down into three chapters:

1. Data Structure
2. R Scripts
3. Creel Analysis

Chapter 1

Data Structure

The format and accuracy of your creel data are pivotal. R is case-sensitive, and the analysis is designed to recognize variables based on a predefined data template. Any changes to this data structure will result in errors.

Additional variables can be added to the data template, but the following key variables must be present, and their column headers must remain unchanged:

- "year" - the year the surveys were conducted.
- "month" - the month the surveys were conducted.
- "dow" - the day of the week the survey was conducted.
- "tod" - the time-of-day stratum (if stratified).
- "site" - the site where the survey was conducted.
- "effort" - the total number of hours fished by the interviewee or interview group.
- "catch" - the total number of fish caught by the interviewee or interview group.

- "harvest" - the total number of fish harvested by the interviewee or interview group.
- "losp" - the length of the survey period.
- "totcnt" - instantaneous or progressive counts of anglers during the survey period.

Chapter 2

R Scripts

A suite of R scripts is provided for the creel analyses. A description of each script is outlined below:

1. **calendar.R** – A helper script that loads the *create_calendar()* function. This function builds a calendar based on the start and end dates entered, automatically assigning holidays as weekends. This script must be loaded prior to running any statistical analyses.
2. **creel-run.R** – The main script used to perform creel analyses. It loads all prerequisite functions, allows the user to define key dates, strata, variables, and species, runs all statistical analyses, and saves the output.
3. **data-filter.R** – A helper script that checks for proper date formatting, filters data based on start and end dates, and produces error messages to help identify potential issues (*e.g.,* a holiday listed as a weekday). This is part of the QA/QC process.
4. **date-check.R** – A helper script that checks for consistency in dates and date formats between contact and count data files. This is part of the QA/QC process.
5. **example_visualizations.R** – A script that provides code to generate example figures based on the creel analysis output.
6. **freq.R** – A helper script that loads the *freq()* function. This function allows for the frequency (*i.e.,* counts) and percentage to be calculated based on user-defined groups.
7. **roving.R** – This script contains the code to perform statistical analyses and loads the *roving1a()*, *roving1b()*, *roving1c()*, and *roving1d()* functions. No changes should be made to this script without a high level of expertise in R.
8. **stratified-sample-design.R** – This script should be run during the creel design phase to randomly assign strata and build a sampling calendar.
9. **summary-stats.R** – A helper script that loads the *creel_summary_stats()* function. This function allows for summary statistics (e.g., mean, median, min, max) to be calculated based on user-defined variables and groups.

Chapter 3

Creel Analysis

This chapter includes the code from the *creel_run.R* script, along with additional explanations. Each R code chunk is broken down into sections for easy reference, and R code chunks are distinguished by a gray background. A basic understanding of R is expected to follow this vignette chapter. If needed, numerous free introductory courses are available online (*e.g.,* datacamp.com).

3.1 REQUIRED PACKAGES FOR THIS ANALYSIS

An R package is a standardized and well-defined collection of code, data, and documentation designed for easy sharing and reuse, extending the core functionality of R. A package needs to be installed only once (or updated when a new version is available) but must be loaded in each R session where its functions will be used. Packages are loaded into the current R session by specifying the package name within *library()*. For example, the following code loads the *readxl*, *dplyr*, *tidyr*, *lubridate*, and *magrittr* packages into the current R session, making their functions available for use.

If R is closed and then re-opened (*i.e.,* a new R session), then all needed packages must be re-loaded with *library()* to make the functionality in the packages available for use during that R session.

```
library(readxl)     ## for loading Excel files
library(dplyr)      ## for data manipulation (see ??tidyverse for additional details)
library(tidyr)      ## for data manipulation (see ??tidyverse for additional details)
library(lubridate)  ## for date formatting
library(magrittr)   ## for %<>%
```

3.2 IMPORT DATA

This section imports the creel data to be analyzed. The data will be loaded and stored in the R environment. All three 'standard' files used by the division are imported using the *read_excel()* function. This process assumes that the files are saved as Excel files (.xlsx). If they are saved in a different format (*e.g.,* comma-separated values [CSV]), you will need to modify the code to use the appropriate function to match the file type or extension. An example of code for a CSV file is below:

```
dat_contact <- read.csv("data/test-ec-contact.csv")
```

NOTE: The species composition file is not required for the analyses below but is imported for posterity and future development of the creel analysis code.

```
## Contact
dat_contact <- read_excel("data/test-ec-contact.xlsx", sheet = 1)
## Counts
dat_count <- read_excel("data/test-ec-count.xlsx", sheet = 1)
## Species Composition
dat_sppcomp <- read_excel("data/test-ec-sppcomp.xlsx", sheet = 1)
```

3.3 DEFINE DATA OF INTEREST

This section allows the user to define key elements of the creel analysis. These elements are used for data filtering, QA/QC, and analysis setup. Consistent formatting and accuracy are critical here as any discrepancies between the data structure and R objects saved here will result in errors. If discrepancies do exist, they will be flagged in Sections 3.4 and 3.6, which will potentially result in the user needing to make adjustments to this section.

```
## Define start and end dates
start_date <- c("2016-04-01")    # Format must be "yyyy-mm-dd"
end_date <- c("2016-09-30")      # Format must be "yyyy-mm-dd"
##--- accurate dates are important to not over inflate the potential number of sampling dates

## Define waterbody name
waterbody_interest <- c("East Canyon Res")
##--- if multiple systems combine with c() e.g., c("East Canyon Res", "Lost Creek Res)
```

3.4 DATA FILTERING

This section is the first step in the QA/QC process. The code below runs the data-filter.R script, and messages will appear in the console indicating whether it was successful. If unsuccessful, carefully review the message and make the necessary adjustments. The most common errors involve improper date formatting in either the data or Section 3.3. Not every discrepancy is obvious and may require significant time to identify.

```
## Run data filtering by dates and waterbody name
source("scripts/data-filter.R")
```

```
## [1] "Contact data filtering successful."
## [1] "Count data filtering successful."
## [1] "Species composition data filtering successful."
```

```
  #--- pay close attention to any errors and address accordingly
```

3.5 LOAD FUNCTIONS FOR ANALYSIS

This section loads all necessary helper functions for the analysis. No changes should be made.

```
## Build calendar from start and end dates to assign DOW and holidays
source("scripts/calendar.R")
## QA/QC
source("scripts/date-check.R")  ## must run calendar.R first to load create_calendar()
## Roving Analyses
source("scripts/roving.R")
## Frequency
source("scripts/freq.R")
## Summary Statistics
source("scripts/summary-stats.R")
```

3.6 QA/QC

This section is the second step in the QA/QC process. The code identifies discrepancies in date formats between the count and contact files, as well as errors in day-of-week entries. The reference calendar is built based on the start and end dates entered in Section 3.3; therefore, accurate dates are essential. After running the code, open the *qaqc* data frame saved in the environment to check for any identified 'flags.' Blank cells (*i.e.,* NA) indicate no errors were found, which is a good outcome.

```
## Check for consistency between data files and if dates and DOW were assigned correctly
qaqc <- check(data1 = dat_contact, data2 = dat_count)
```

3.7 DEFINE STARTIFICATION OPTION

This section defines the grouping variables used in all future analyses. The code currently includes four predefined options. Users must modify the number at the end of the code chunk to match their creel design. If you have a stratified design that does not fit within these four categories, significant modifications to the *roving.R* script will be required. Consulting someone familiar with the code is recommended in such cases.

```r
## Stratification options to define stratum. This will determine how creel data are summarized.
### option 1
### 1. by Month and DOW
### 2. by Month
### 3. Overall

### option 2
### 1. by Month, DOW, and TOD
### 2. by Month and DOW
### 3. by Month
### 4. Overall

### option 3
### 1. by Site, Month, and DOW
### 2. by Site and Month
### 3. by Site
### 4. Overall

### option 4
### 1. by Site, Month, DOW, and TOD
### 2. by Site, Month, and DOW
### 3. by Site and Month
### 4. by Site
### 5. Overall

## Define stratification option
opt <- 2
```

3.8 DEFINE GROUPING VARIABLES BASED ON STRATIFICATION

This section builds on Section 3.7, and no changes should be made unless your creel design does not fit within the four categories defined above. However, the vector of grouping variables created in this code chunk must match the column headers in both the count and contact data files.

```r
if(opt == 1 | opt == 2) {
  grp_var_vec <- c("month", "day", "dow", "tod")
  ## Remove 'day' by position in vector above. Adjust accordingly.
  grp_var_vec2 <- grp_var_vec[-2]

} else if(opt == 3 | opt == 4) {
  grp_var_vec <- c("month", "day", "dow", "tod", "site")
  ## Remove 'day' by position in vector above. Adjust accordingly.
  grp_var_vec2 <- grp_var_vec[-2]
}
```

3.9 CALCULATE OVERALL CATCH CREEL STATISTICS

Now that the data has been checked and variables defined, we can run the statistical analyses. The first function summarizes catch statistics for the contact dataset. No changes should be made to the function unless variable names or column headers have been modified.

Pay close attention to the annotated remarks in the code chunk below.

```
## Statistics for contact data set
## Grouping variables should match the variables listed in choice one of the stratification option, plu
roving1a(data = dat_contact, grp_var = grp_var_vec, effort = "effort", catch = "catch")
##--- IGNORE WARNING ABOUT USING EXTERNAL VECTORS. A suitable alternative is not available at the time
##--- output is saved as "contact_sum" in environment
##--- NOTE: if "tot_anglers" = 1, no variance components can be calculated and will be stated as NaN
```

This function summarizes catch statistics for the count dataset. No changes should be made unless variable names or column headers have been modified.

```
## Statistics for count data set
## Grouping variables should match the variables listed in choice one of the stratification option, plu
roving1b(data = dat_count, grp_var = grp_var_vec, count = "totcnt", sample_time = "losp")
##--- output is saved as "count_sum" in environment
##--- NOTE: if "tot_anglers" = 1, variance components will be zero
```

This function calculates catch statistics based on stratification. No changes should be made unless variable names or column headers have been modified.

```
## Statistics by stratum
## Grouping variables should match the variables listed in choice one of the stratification option.
roving1c(grp_var = grp_var_vec2)
##--- output is saved as "stratum_sum" in environment
##--- NOTE: if "n" = 1, variance components will be zero or NA
```

This function summarizes catch statistics based on stratification. No changes should be made unless variable names or column headers have been modified.

```
## Summarize by options defined above
roving1d(option = opt)
##--- outputs are saved as multiple objects in environment based on grouping option defined.
##--- each object will be prefixed with "stat".
```

3.10 SAVE STATISTICAL OUTPUT

This section saves the catch statistic summaries as individual CSV files in the ‘tables’ directory. No changes should be made unless variable names or column headers have been modified.

```
## Export Overall CATCH
if(opt == 1 | opt == 3) {
  write.csv(statdow, "tables/catch_overall_dow.csv", row.names = FALSE)
  write.csv(statmon, "tables/catch_overall_mon.csv", row.names = FALSE)
  write.csv(statall, "tables/catch_overall_all.csv", row.names = FALSE)
} else if(opt == 2 | opt == 4) {
  write.csv(stattod, "tables/catch_overall_tod.csv", row.names = FALSE)
  write.csv(statdow, "tables/catch_overall_dow.csv", row.names = FALSE)
  write.csv(statmon, "tables/catch_overall_mon.csv", row.names = FALSE)
  write.csv(statall, "tables/catch_overall_all.csv", row.names = FALSE)
}
```

3.11 CALCULATE OVERALL HARVEST CREEL STATISTICS

The next four code chunks follow the same structure as Section 3.9, but the response variable is changed from ‘catch’ to ‘harvest’ to calculate the harvest statistics. No changes should be made unless variable names or column headers have been modified.

```
## Statistics for contact data set
## Grouping variables should match the variables listed in choice one of the stratification option, plu
roving1a(data = dat_contact, grp_var = grp_var_vec, effort = "effort", catch = "harvest")
##--- output is saved as "contact_sum" in environment
##--- NOTE: if "tot_anglers" = 1, no variance components can be calculated and will be stated as NaN
```

```
## Statistics for count data set
## Grouping variables should match the variables listed in choice one of the stratification option, plu
roving1b(data = dat_count, grp_var = grp_var_vec, count = "totcnt", sample_time = "losp")
##--- output is saved as "count_sum" in environment
##--- NOTE: if "tot_anglers" = 1, variance components will be zero
```

```
## Statistics by stratum
## Grouping variables should match the variables listed in choice one of the stratification option.
roving1c(grp_var = grp_var_vec2)
##--- output is saved as "stratum_sum" in environment
##--- NOTE: if "n" = 1, variance components will be zero or NA
```

```
## Summarize by options defined above
roving1d(option = opt)
##--- outputs are saved as multiple objects in environment based on grouping option defined.
##--- each object will be prefixed with "stat"
```

3.12 SAVE STATISTICAL OUTPUT

This section saves the harvest statistic summaries as individual CSV files in the ‘tables’ directory. No changes should be made unless variable names or column headers have been modified.

```
## Export Overall HARVEST
if(opt == 1 | opt == 3) {
  write.csv(statdow, "tables/harvest_overall_dow.csv", row.names = FALSE)
  write.csv(statmon, "tables/harvest_overall_mon.csv", row.names = FALSE)
  write.csv(statall, "tables/harvest_overall_all.csv", row.names = FALSE)
} else if(opt == 2 | opt == 4) {
  write.csv(stattod, "tables/harvest_overall_tod.csv", row.names = FALSE)
  write.csv(statdow, "tables/harvest_overall_dow.csv", row.names = FALSE)
  write.csv(statmon, "tables/harvest_overall_mon.csv", row.names = FALSE)
  write.csv(statall, "tables/harvest_overall_all.csv", row.names = FALSE)
}
```

3.13 CALCULATE SHORE CATCH STATISTIC

This section filters the data to include only shore anglers. This is for example purposes only, and the user can select any variable or angling group of interest. Changes must be made within the *filter()* function to reflect the variable and factor level of interest.

```r
## Create data frame with shore anglers
dat_shore <- dat_contact %>% filter(method == "s") ## case sensitive!!
```

3.14 CALCULATE SHORE CATCH CREEL STATISTICS

The next four code chunks follow the same structure as Section 3.9, but the data source is changed to calculate the catch statistics for shore anglers only. Users can modify the name of the data frame in the *roving1a()* function to match the data of interest from Section 3.13. Do not change the data frame in *roving1b()*, as we still want the total count to reflect all anglers.

```r
## Statistics for contact data set
## Grouping variables should match the variables listed in choice one of the stratification option, plu
roving1a(data = dat_shore, grp_var = grp_var_vec, effort = "effort", catch = "catch")

## Statistics for count data set
## Grouping variables should match the variables listed in choice one of the stratification option, plu
roving1b(data = dat_count, grp_var = grp_var_vec, count = "totcnt", sample_time = "losp")

## Statistics by stratum
## Grouping variables should match the variables listed in choice one of the stratification option.
roving1c(grp_var = grp_var_vec2)

## Summarize by options defined above
roving1d(option = opt)
```

3.15 SAVE STATISTICAL OUTPUT

This section saves the shore angler catch statistic summaries as individual CSV files in the 'tables' directory. No changes should be made unless variable names or column headers have been modified.

```r
## Export shore CATCH
if(opt == 1 | opt == 3) {
  write.csv(statdow, "tables/catch_shore_dow.csv", row.names = FALSE)
  write.csv(statmon, "tables/catch_shore_mon.csv", row.names = FALSE)
  write.csv(statall, "tables/catch_shore_all.csv", row.names = FALSE)
} else if(opt == 2 | opt == 4) {
  write.csv(stattod, "tables/catch_shore_tod.csv", row.names = FALSE)
  write.csv(statdow, "tables/catch_shore_dow.csv", row.names = FALSE)
  write.csv(statmon, "tables/catch_shore_mon.csv", row.names = FALSE)
  write.csv(statall, "tables/catch_shore_all.csv", row.names = FALSE)
}
```

The code can be repeated to calculate harvest statistics for the filtered data, as done in Section 3.11.

3.16 CALCULATE CATCH CREEL STATISTICS BY SPECIES

The next section follows the same structure as Section 3.9, but the catch variable is modified to calculate the catch statistics for a single species. This process is looped to analyze multiple species and append the summaries into one file containing all species.

The typical format adopted by DWR biologists is to name the column header for the catch of a species with 'c_' followed by the species abbreviation. Users can modify the list of species abbreviations to match the column names in the contact data file in the code below. The species abbreviations used by each region vary, so users should pay close attention to the line of code below.

```r
## use the species abbreviations in the headers (e.g., catch of rainbow trout is labeled as 'c_rbt' so
spp_vec <- c("rbt", "bnt", "smb")
```

The next code chunks calculate the catch statistics for each species listed in the vector above. The *lapply()* function selects each species abbreviation in *spp_vec* and creates the catch variable name for that species in *roving1a()* (*e.g.,* c_rbt). No changes should be made to the code below unless the contact column headers do not follow the typical data structure, or the default stratification options do not fit your creel design.

```r
## run loop over vector of species above
lapply(spp_vec, function(spp) {
  ## Statistics for contact data set
  ## Grouping variables should match the variables listed in choice one of the stratification option, p
  roving1a(data = dat_contact, grp_var = grp_var_vec, effort = "effort", catch = paste0("c_", spp))

  ## Statistics for count data set
  ## Grouping variables should match the variables listed in choice one of the stratification option, p
  roving1b(data = dat_count, grp_var = grp_var_vec, count = "totcnt", sample_time = "losp")

  ## Statistics by stratum
  ## Grouping variables should match the variables listed in choice one of the stratification option.
  roving1c(grp_var = grp_var_vec2)

  ## Summarize by options defined above
  roving1d(option = opt)


  ## Export species CATCH by each stratification level
  if(opt == 1 | opt == 3) {
    if(exists("statdow_spp") == FALSE) {
      statdow_spp <<- statdow %<>% mutate(Species = spp) %>%
        select(Waterbody, Species, everything())
    } else {
      statdow %<>% mutate(Species = spp)
      statdow_spp <<- rbind(statdow_spp, statdow)
    }
    if(exists("statmon_spp") == FALSE) {
      statmon_spp <<- statmon %<>% mutate(Species = spp) %>%
        select(Waterbody, Species, everything())
    } else {
      statmon %<>% mutate(Species = spp)
      statmon_spp <<- rbind(statmon_spp, statmon)
    }
    if(exists("statall_spp") == FALSE) {
      statall_spp <<- statall %<>% mutate(Species = spp) %>%
        select(Waterbody, Species, everything())
    } else {
      statall %<>% mutate(Species = spp)
      statall_spp <<- rbind(statall_spp, statall)
    }
  } else if(opt == 2 | opt == 4) {
    if(exists("stattod_spp") == FALSE) {
      stattod_spp <<- stattod %<>% mutate(Species = spp) %>%
        select(Waterbody, Species, everything())
    } else {
      stattod %<>% mutate(Species = spp)
      stattod_spp <<- rbind(stattod_spp, stattod)
    }
    if(exists("statdow_spp") == FALSE) {
      statdow_spp <<- statdow %<>% mutate(Species = spp) %>%
        select(Waterbody, Species, everything())
    } else {
      statdow %<>% mutate(Species = spp)
      statdow_spp <<- rbind(statdow_spp, statdow)
    }
    if(exists("statmon_spp") == FALSE) {
      statmon_spp <<- statmon %<>% mutate(Species = spp) %>%
        select(Waterbody, Species, everything())
    } else {
      statmon %<>% mutate(Species = spp)
      statmon_spp <<- rbind(statmon_spp, statmon)
    }
    if(exists("statall_spp") == FALSE) {
      statall_spp <<- statall %<>% mutate(Species = spp) %>%
        select(Waterbody, Species, everything())
    } else {
      statall %<>% mutate(Species = spp)
      statall_spp <<- rbind(statall_spp, statall)
    }
  }
})
```

CAUTION: Re-running the loop above will only append new data to the previous data (no overwriting). Run the line of code below to remove the previous data from the environment (the hashtag will need to be removed).

```
# rm(stattod_spp, statdow_spp, statmon_spp, statall_spp)
```

3.17 SAVE STATISTICAL OUTPUT

This section saves the catch statistic summaries by species as individual CSV files in the ‘tables’ directory. No changes should be made unless variable names or column headers have been modified.

```
## Save catch by species files
if(opt == 1 | opt == 3) {
  write.csv(statdow_spp, paste0("tables/catch_by_species_dow.csv"), row.names = FALSE)
  write.csv(statmon_spp, paste0("tables/catch_by_species_mon.csv"), row.names = FALSE)
  write.csv(statall_spp, paste0("tables/catch_by_species_all.csv"), row.names = FALSE)
} else if(opt == 2 | opt == 4) {
  write.csv(stattod_spp, paste0("tables/catch_by_species_tod.csv"), row.names = FALSE)
  write.csv(statdow_spp, paste0("tables/catch_by_species_dow.csv"), row.names = FALSE)
  write.csv(statmon_spp, paste0("tables/catch_by_species_mon.csv"), row.names = FALSE)
  write.csv(statall_spp, paste0("tables/catch_by_species_all.csv"), row.names = FALSE)
}
rm(stattod_spp, statdow_spp, statmon_spp, statall_spp)
```

The code can be repeated to calculate harvest statistics by species, as done in Section 3.11, except the column headers will be labeled as ‘h_’ for harvest.