Skagit Creel Analysis

Thomas Buehrens, Kale Bentley, Andrew Fowler & Amy Edwards

Contents

This document was generated on 01/21/2021.

Purpose

The purpose of this document is to record the steps and code necessary to reproduce the Skagit steelhead fishery creel analysis for 2021.

Requirements

All analyses require R software **(link)** (v3.4.3) for data retrieval, data processing, and summarizing model results, and Stan software **(link)** for Hamiltonian Monte Carlo (HMC) simulation. For Stan to work, rtools must also be installed: **(link)**.

Functions

We also need a couple of helper functions which we will load from the functions folder, which we will load using sapply

```
wd_functions<-"functions"
sapply(FUN = source, paste(wd_functions, list.files(wd_functions), sep="/"))
```

Packages

In addition to purrr, we also need a few packages that are not included with the base installation of R, so we begin by installing them (if necessary) and then loading them.

```
#=============================================
# Load packages, install and load if not already
#=============================================
using("tidyverse",
      "rstan",
      "RColorBrewer",
      "readxl",
      "readr",
      #"ggplot2",
      "tinytex",
      "here",
      "plyr",
      "lubridate",
      "devtools",
      "xlsx",
      "cowplot",
      "ggpubr",
      "chron",
      "suncalc",
      "shinystan",
      "loo",
      "data.table",
      "RColorBrewer",
      "reshape2",
      "MASS",
      "timeDate"
      )
```

User inputs

```
#=====================================================
# Specify relative working directories for sub-folders
#=====================================================
wd_LUTs <-"lookup tables"       # Location of look-up tables (maybe could be merged with data files??)
wd_data   <-"data"              # Location where data files are stored
wd_source_files<-"source files" # Location of source file (code working "behind the scenes")
```

```
wd_models  <-"models"           # Location of model files
wd_outputs <-"results"          # Location of saved output (summary figures/tables and model results)

#=====================================================
# Specify names of .csv data files
#=====================================================
effort_file_name <-   "03_Effort_dat - 2019_Skagit_creel_JSH_thru_4-30-19.csv"
interview_file_name <-"03_Interview-dat_2019-Skagit_JSH_thru_4-30-2019.csv"
effort_xwalk_filename<-"02_Crosswalk_Table_for_Index_TieIn_Sections_2019-01-10.csv"
river_loc_filename<-"02_River.Locations_2019-01-07.csv"
creel_models_filename<-"02_Creel_Models_2021-01-20.csv"

#=====================================================
# Denote data of interest (used to filter data below)
#=====================================================
# Specify filter type(s) to extract data by (Enter "Y" or "N")
  by.Year<-      "N" # If "Y", will filter by full calendar year(s) (Jan. 1 - Dec. 31)
  by.YearGroup<- "N" # If "Y", will filter by a "Year Group", which go from May 1st Yr1 - April 30 Yr2
  by.Season<-    "N" # If "Y", will filter by "season", which is either Summer (May 1st - Oct 31st) or
  by.StreamName<-"Y" # If "Y", will filter by stream name
  by.Date<-      "N" # If "Y", will filter by a date range

# Specify date ranges for "Year Groups" and "Seasons"
  YearBegin<-  121 # day of year a "YearGroup" begins (FYI - 121 = May 1st in a non-leap year)
  summerBegin<-121
  summerEnd<-  304 # FYI - 304 = Oct. 31st (in a non-leap year)
  winterBegin<-305
  winterEnd<-  120

# Specify filter unit(s)
  YearGroup.of.Interest<- c("2017-2018")
  Season.of.Interest<-    c("Winter")
  Year.of.Interest<-      c("2017")
  StreamName.of.Interest<-c("Skagit")
  Begin.Date<-            c("2016-05-01") #Format must be "yyyy-mm-dd"
  End.Date<-              c("2017-03-31") #Format must be "yyyy-mm-dd"

#=====================================================
# Denote catch group of interest (species_origin_fate)
#=====================================================
  catch.group.of.interest<-c("SH_W_R")

#=====================================================
# Identify dates when fishery was closed by section
#=====================================================
  total.closed.dates<-0 # Total number of dates that at least one section of the river was closed (i.e.

# NOTE: if "total.closed.dates" >0, use the following format to enter closure dates and section, where
      # the first column is the list of individual dates (by row) the fishery was closed date
      # the number of additional columns equals the number of "final" sections based on "final.effort.s
      # the enter the following values below each section:
      # Enter 1 if the section was open and enter 0 if the section was closed
```

```
                            #    Date    , Section-1, Section-2
    closed.Dates.Sections<-c("2019-02-11",     0,         0 )
```

Data Preparation

```
# Load LUTs
  source(paste0(wd_source_files, "/Load_LUTs.R"))

# Load creel data and format
  source(paste0(wd_source_files, "/Import_Skagit_Creel_Data_and_Format.R"))

# Extract data of interest and format
  ## add code that shows options for filtering data by date/year/season/location
  source(paste0(wd_source_files, "/05_Extract_Data_of_Interest_and_Calculate_Fields_2019-04-08.R"))

#Run source summary file
  ## add code that shows options for "catch groups"
  source(paste0(wd_source_files, "/06_Summarize_Effort_and_Catch_Data_for_TimeSeries_Model_2019-04-23.R

##KB note: I will work on updating the code in the "05" and "06" file at some point soon; also, creatin
```

Run Analysis

```
#message=FALSE, warning=FALSE
#=======
#note for editing: any ner priors need to go here, also in "prepare data" and in "summarize inputs"
#=======

# Denote whether you want to run a new model or load "saved" results from a previous model run
  model_source<-c("load_saved")  #enter either "run_new" or "load_saved"

# Assign a "Model_Run" number (if model_source == run_new, results will be saved to a new sub-folder; i
  Model_Run<-8 #Enter numeric number (NOTE: be careful not to over-write previous models runs by enteri

# Denote which creel model you want to run
  creel_models[,1:3] #model summary table
  model_number<-c(3) # 1 and 2 are the 6-3-2019 uncorr and corr

# Specify time period to stratify data by - day vs. week
  model_period<-c("day") #enter "day" or "week"

# Specify parameter values for model priors
  value_cauchyDF_sigma_eps_C = 1 # the hyperhyper scale (degrees of freedom) parameter in the hyperprio
  value_cauchyDF_sigma_eps_E = 1 # the hyperhyper scale (degrees of freedom) parameter in the hyperprio
  value_cauchyDF_sigma_r_E = 1  # the hyperhyper scale (degrees of freedom) parameter in the hyperprior
  value_cauchyDF_sigma_r_C = 1  # the hyperhyper scale (degrees of freedom) parameter in the hyperprior
  value_normal_sigma_omega_C_0 = 1  #the SD hyperparameter in the prior distribution omega_C_0; normal
  value_normal_sigma_omega_E_0 =  3 # the SD hyperparameter in the prior distribution omega_E_0; normal
```

```
  value_lognormal_sigma_b = 1 # the SD hyperparameter in the prior distribution b; default = 1
  value_normal_sigma_B1 = 5 # the SD hyperparameter in the prior distribution B1; default = 5
  value_normal_mu_mu_C = log(0.02) # the mean hyperparameter in the prior distribution mu_C; median (lo
  value_normal_sigma_mu_C = 1.5 # the SD hyperparameter in the prior distribution mu_C; normal sd (log-
  value_normal_mu_mu_E = log(15) # the mean hyperparameter in the prior distribution mu_E; median effor
  value_normal_sigma_mu_E = 2  # the SD hyperparameter in the prior distribution mu_E; normal sd (log-s
  value_betashape_phi_E_scaled = 1 # the rate (alpha) and shape (beta) hyperparameters in phi_E_scaled;
  value_betashape_phi_C_scaled = 1 # the rate (alpha) and shape (beta) hyperparameters in phi_C_scaled;
  value_cauchyDF_sigma_mu_C = 1       # the hyperhyper SD parameter in the hyperprior distribution sigm
  value_cauchyDF_sigma_mu_E = 1       # the hyperhyper SD parameter in the hyperprior distribution sigm

# Specific Stan model arguments
  n_chain<-4          # set the number of Markov chains. The default is 4.
  n_iter<-2000         # set the number of iterations for each chain (including warmup). The default is
  n_cores<-4          # set the number of cores to use when executing the chains in parallel. The defaul
  n_warmup<-1000      # set the length of warm-up (aka burn-in) iterations per chain.  The default is n_
  n_thin<-1           # set the thinning rate (aka, the period for saving samples). The default is 1, wh
  adapt_delta<-0.9  # set adapt delta, which is the target average proposal acceptance probability duri
  max_treedepth<-10 # set the max tree depth; default is 8; NOTE: this sets the max depth of tree used

# Create sub-folders for output (if they don't already exist)
    source(paste0(wd_source_files, "/Create_output_subfolder.R"), print.eval = TRUE)

# Run source code to prepare data for model
    source(paste0(wd_source_files, "/Prepare_Data_For_Model.R "))

# Run source code to generate creel estimates
  source(paste0(wd_source_files, "/RunNew_or_LoadSaved_Creel_Model.R"))

# Generate summaries of model inputs and outputs
  if(model_source == "run_new"){  source(paste0(wd_source_files, "/Summarize_Model_Inputs_and_Outputs.R
```

Summarize Results

```
#convergence diagnostics
  launch_diagnostics<-c("No") #Enter "Yes" to launch ShinyShin diagnostics
  if(launch_diagnostics=="Yes"){launch_shinystan(output$res_stan)}

# generate plots and tables of creel estimates
  source(paste0(wd_source_files, "/Generate_Summaries_of_Creel_Estimates.R"))

# KB note: update so table/plots of results are shown in PDF document
```

Reproducing this pdf or html page

In order to reproduce this pdf or html page you need to have a LaTex application installed. Running this snippet of code will automatically install tinytex on your machine so you can render pdfs and html:

```
#tinytex::install_tinytex()
```

Results: Summary of Effort, CPUE, and Catch

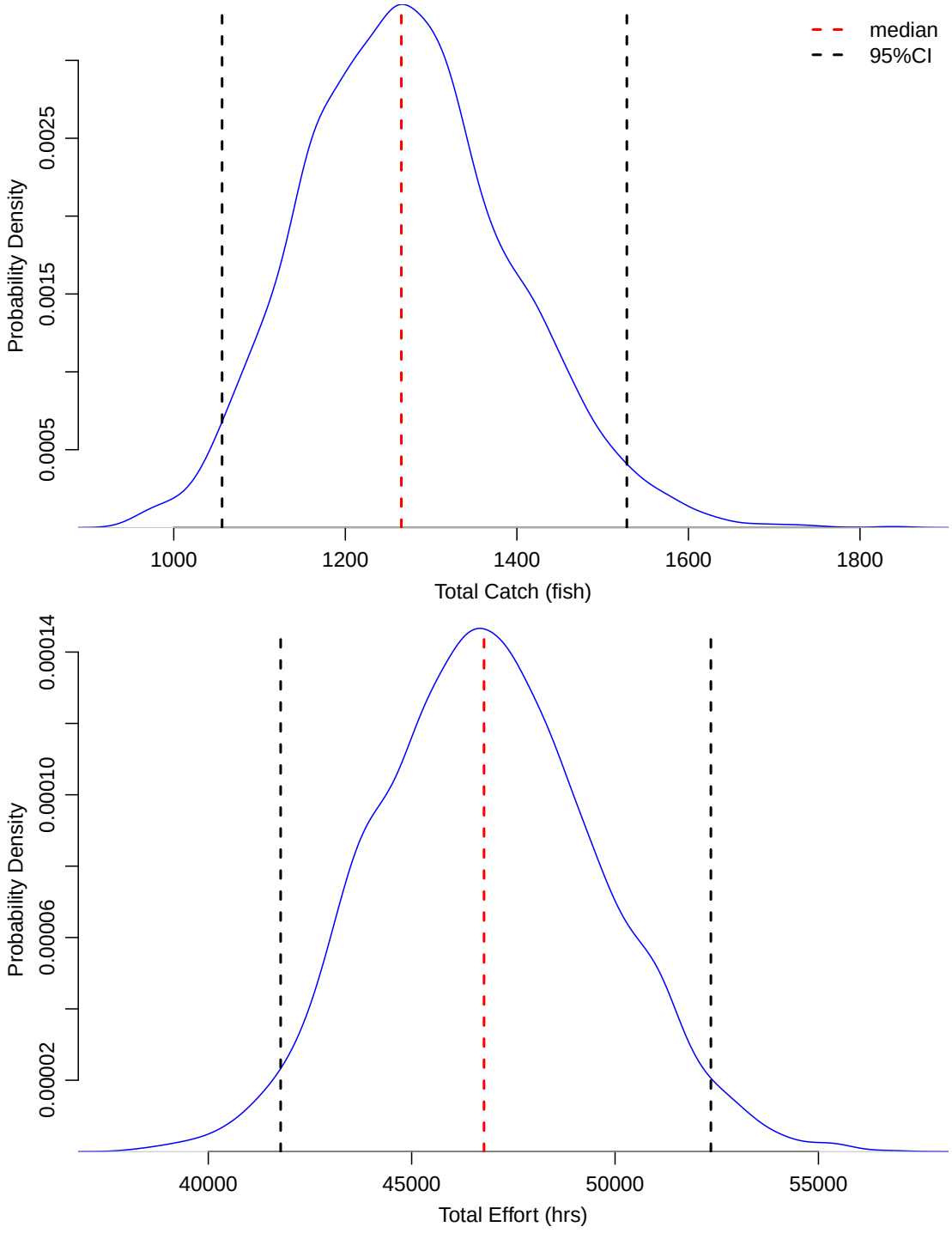

Figure 1: Total Catch and Total Effort

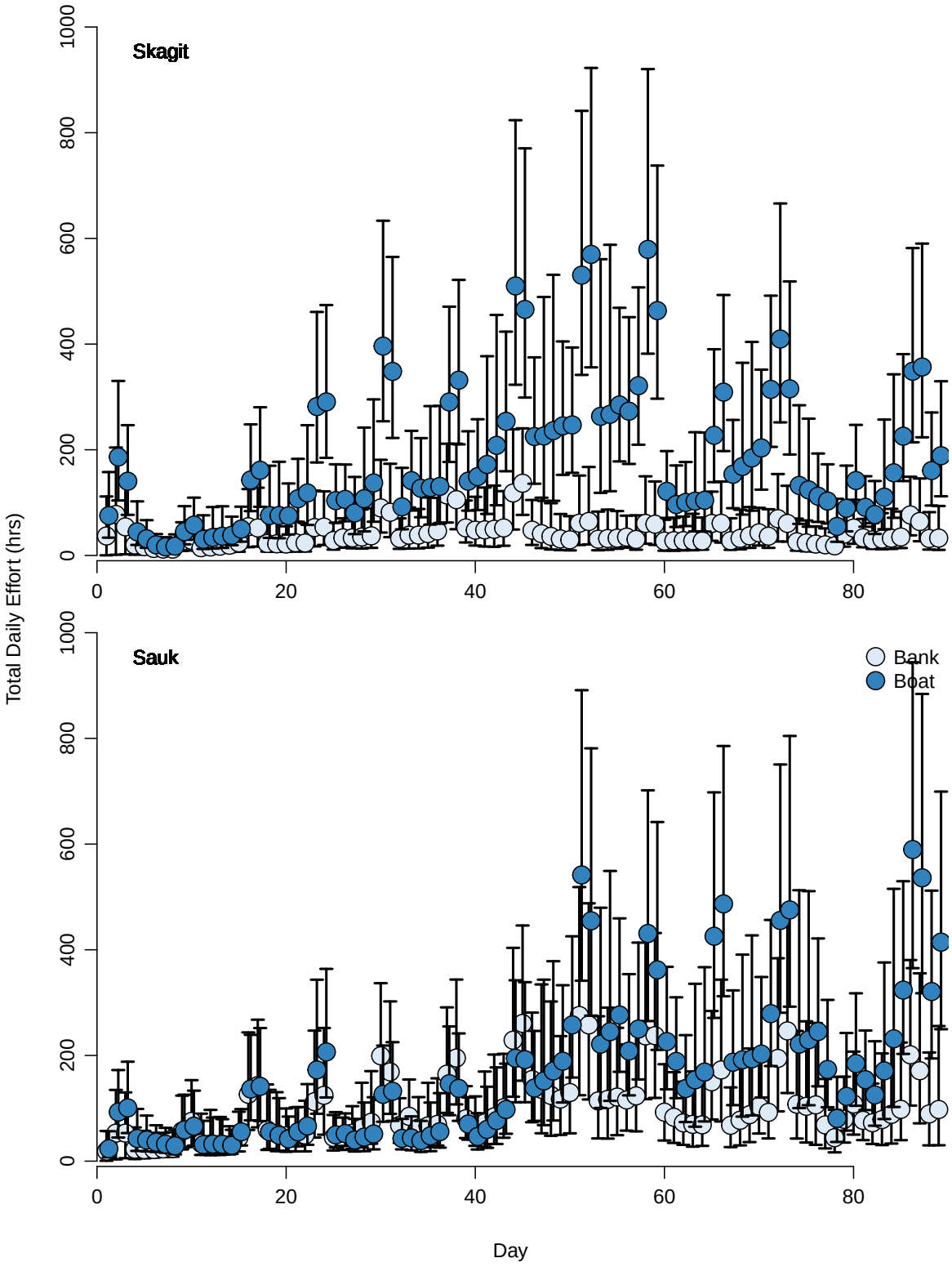

Figure 2: Daily Effort

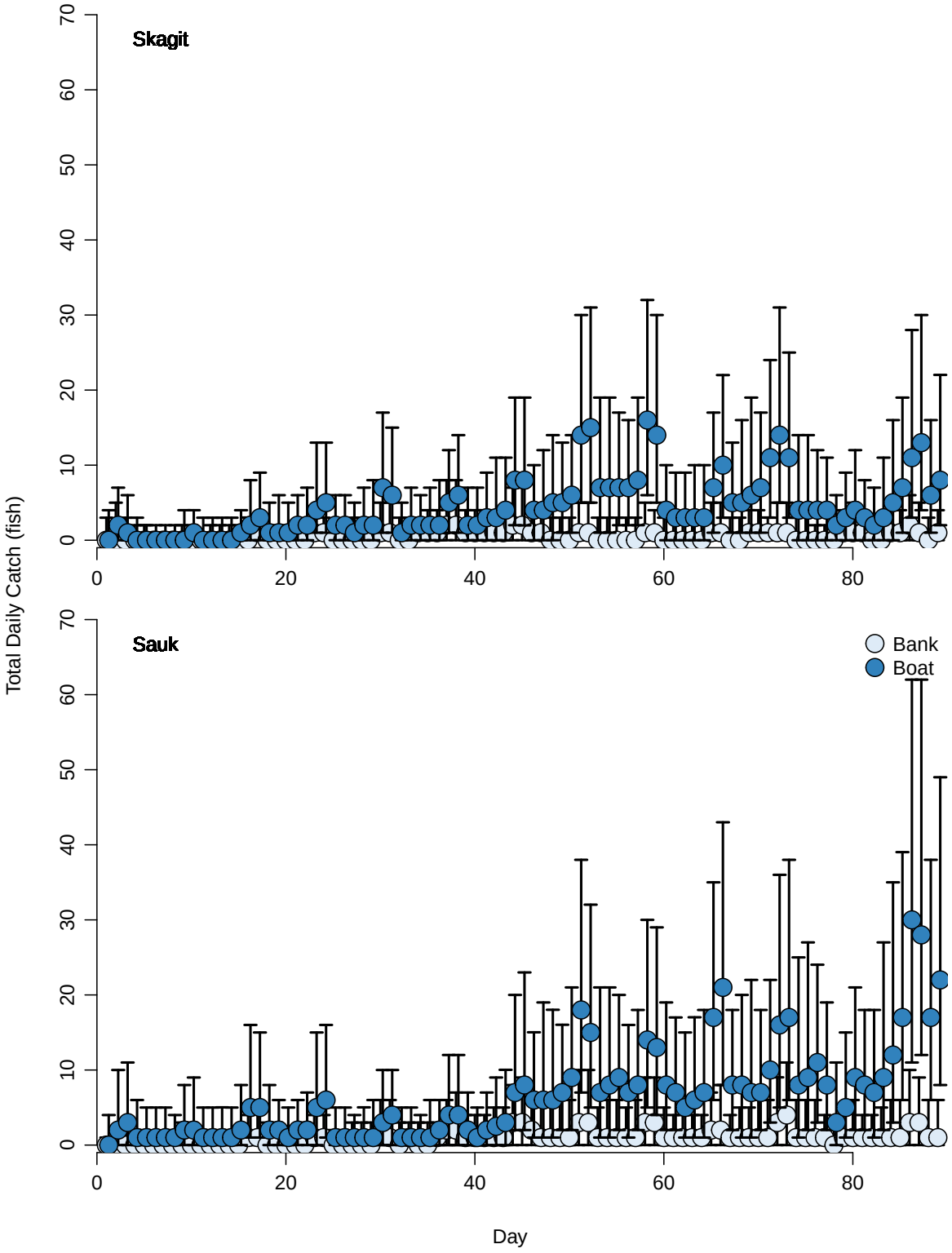

Figure 3: Daily Catch

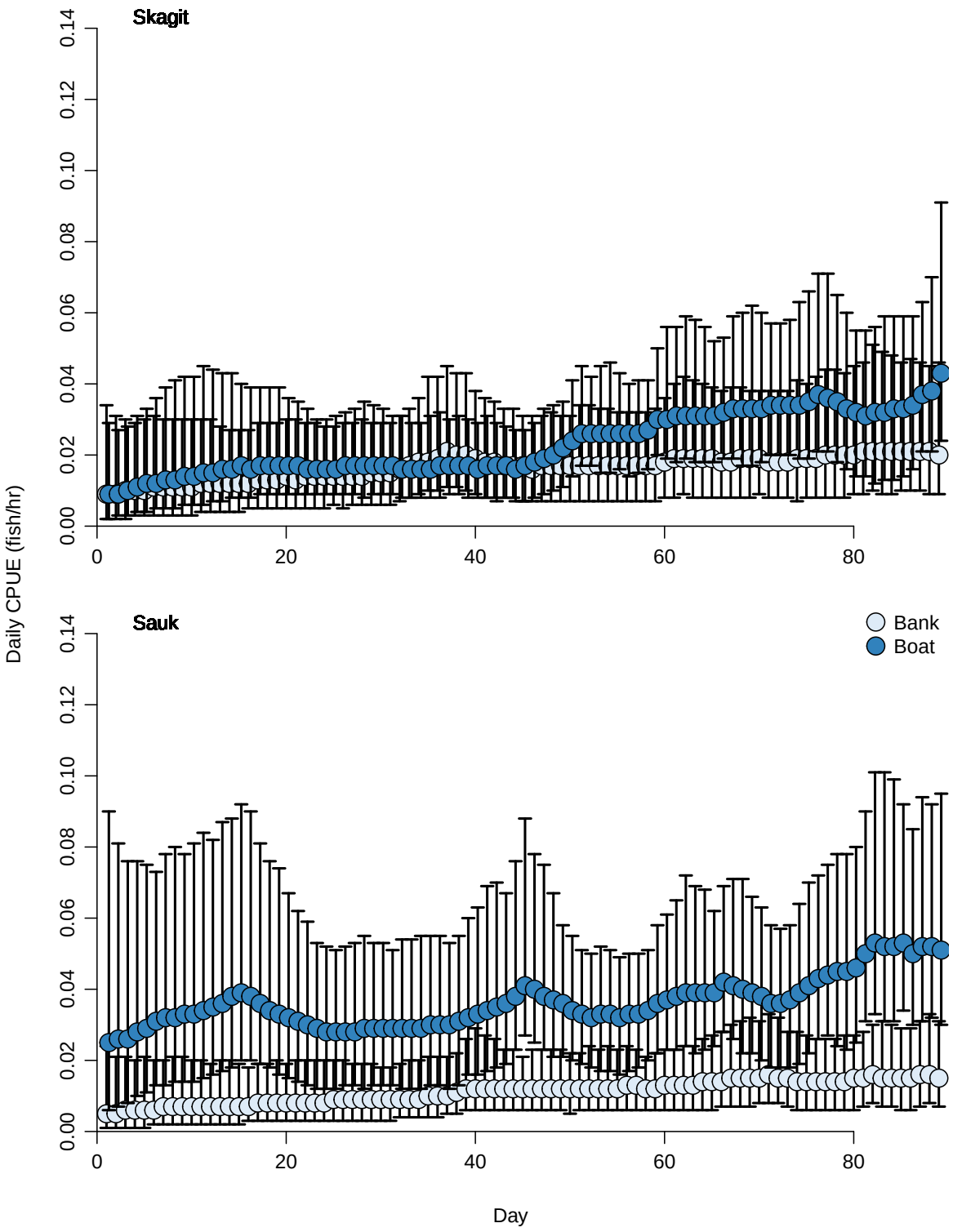

Figure 4: Daily CPUE