UDWR Creel Survey - Stratified Random Sampling Design

Introduction

This vignette is designed to provide a brief overview of one method to design a creel survey using stratified random sampling. Examples are provided to stratify by time of day, day of week, and multiple access points.

Required Packages for this Vignette

Functions used in this chapter require loading the packages shown below.

```
library(dplyr)      ## for data manipulation
library(tidyr)      ## for data manipulation
library(lubridate)  ## for dates
library(magrittr)   ## for %<>%
library(suncalc)    ## for retrieving sunrise and sunset times
library(ggplot2)    ## for visualizations
```

Build Sampling Calendar

Before we can conduct random sampling, we need to build the sampling calendar which we will randomly draw samples from. This design builds a calendar that classifies holidays as weekends to accurately stratify by day of week. Day length (i.e., sunrise, sunset, and the time between) is also calculated for each day in the sampling calendar.

1. Create a vector of possible sampling dates

```
## Define start and end dates
start_date <- "2026/03/01"
end_date <- "2026/10/31"
## Create vector based on start and end dates
date_vec <- seq.Date(as.Date(start_date), as.Date(end_date), "days")

## Load function to create a calendar data frame that can be merged with creel data files
source("scripts/calendar.R")

## Build calendar from start and end dates to assign DOW and holidays
cal <- create_calendar(start_date, end_date) %>%
  ## Add week number
  mutate(week = isoweek(date))

## Display first six rows for example
head(cal)
```

```
##         date year month day      wday dow week
## 1 2026-03-01 2026     3   1    Sunday  we    9
## 2 2026-03-02 2026     3   2    Monday  wd   10
## 3 2026-03-03 2026     3   3   Tuesday  wd   10
## 4 2026-03-04 2026     3   4 Wednesday  wd   10
## 5 2026-03-05 2026     3   5  Thursday  wd   10
## 6 2026-03-06 2026     3   6    Friday  wd   10
```

2. Find the sunrise and sunset times for a given location for each date in the vector defined above.

```
## Define daily sunrise and sunset times
#---adjust the latitude and longitude to reflect your sampling location
dat <- getSunlightTimes(date_vec, lat = 41.186126, lon = -111.381330,
                        keep = c("sunrise", "sunset"), tz = "America/Denver") %>%
  ## Add month variable
  mutate(month = month(date))

## Display first six rows for example
head(dat)
```

```
##         date      lat       lon             sunrise              sunset month
## 1 2026-03-01 41.18613 -111.3813 2026-03-01 07:01:24 2026-03-01 18:17:17     3
## 2 2026-03-02 41.18613 -111.3813 2026-03-02 06:59:50 2026-03-02 18:18:26     3
## 3 2026-03-03 41.18613 -111.3813 2026-03-03 06:58:15 2026-03-03 18:19:35     3
## 4 2026-03-04 41.18613 -111.3813 2026-03-04 06:56:40 2026-03-04 18:20:43     3
## 5 2026-03-05 41.18613 -111.3813 2026-03-05 06:55:04 2026-03-05 18:21:51     3
## 6 2026-03-06 41.18613 -111.3813 2026-03-06 06:53:28 2026-03-06 18:22:59     3
```

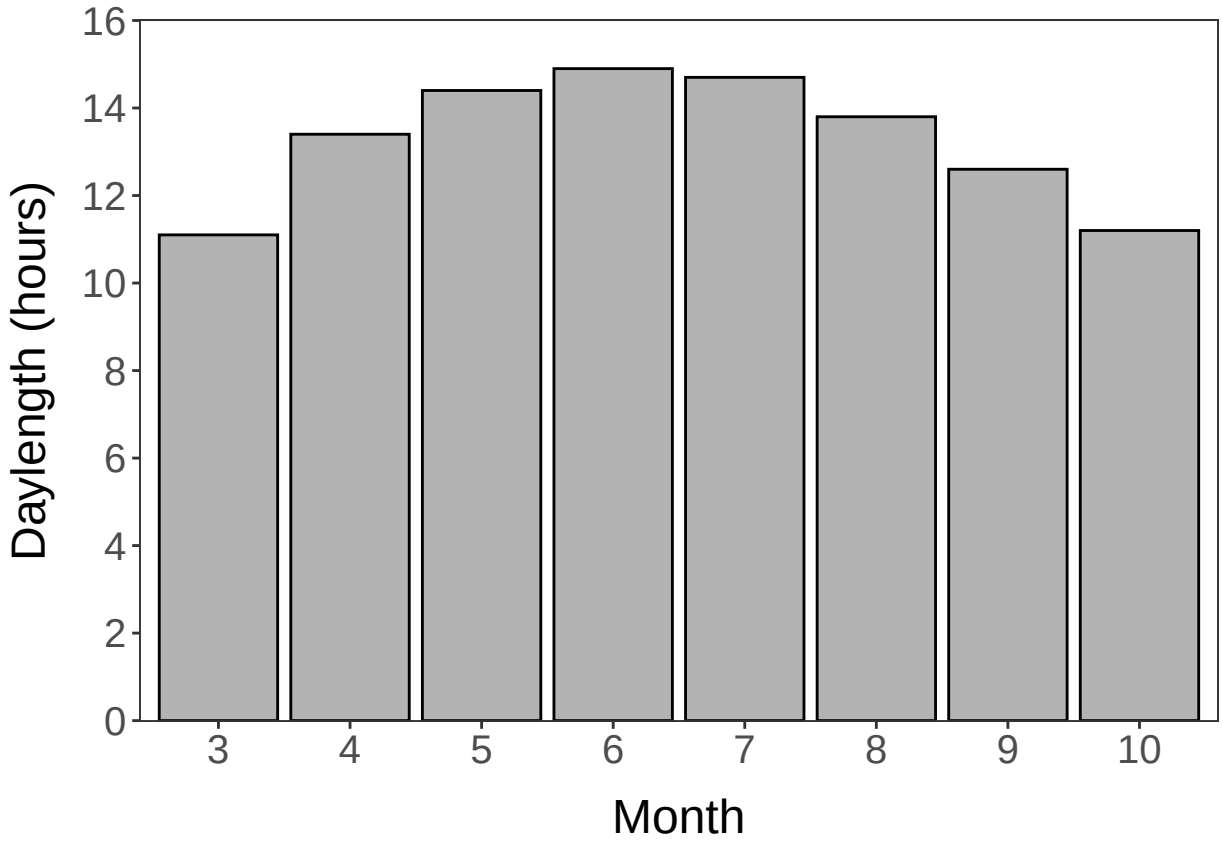
Examine Day Length During Sampling Period

A common stratum is time of day (e.g., AM and PM). The rationale for this stratum is explained in the monitoring plan but the figures below provide a visual representation of why this stratum improves feasibility of a creel design.

```
## Summarize the calendar
dat_sum <- dat %>% group_by(month) %>%
  ## find latest sunrise and earliest sunset in each month
  summarize(max_sunrise = max(sunrise),
            min_sunset = min(sunset)) %>%
  ## build sunrise and sunset variables with common date to calculate length of day
  mutate(sunrise_hour = hour(max_sunrise),
         sunrise_minute = minute(max_sunrise),
         sunrise_time = as.POSIXct(paste0(Sys.Date(), " ", sunrise_hour, ":",
                                          sunrise_minute), format = "%Y-%m-%d %H:%M"),
         sunset_hour = hour(min_sunset),
         sunset_minute = minute(min_sunset),
         sunset_time = as.POSIXct(paste0(Sys.Date(), " ", sunset_hour, ":",
                                         sunset_minute), format = "%Y-%m-%d %H:%M")) %>%
  ## calculate length of day in each month
  mutate(lod = round(sunset_time-sunrise_time, 1))
```

No Time of Day Stratum

```
## plot
ggplot(dat_sum, aes(x = month, y = lod)) +
  geom_bar(stat = "identity", fill = "gray70", color = "black") +
  scale_x_continuous(limits = c(2.5, 10.5), breaks = seq(1, 12, 1), expand = c(0, 0.1)) +
  scale_y_continuous(limits = c(0, 16), breaks = seq(0, 16, 2), expand = c(0, 0.02)) +
  labs(y = "Daylength (hours)", x = "Month") +
  theme_bw() +
  theme(axis.title.x = element_text(size = 18, margin = margin(10, 0, 0, 0)),
        axis.title.y = element_text(size = 18, margin = margin(0, 10, 0, 0)),
        axis.text.x = element_text(size = 15),
        axis.text.y = element_text(size = 15),
        panel.grid = element_blank(),
        plot.margin = margin(10, 10, 5, 5))
```

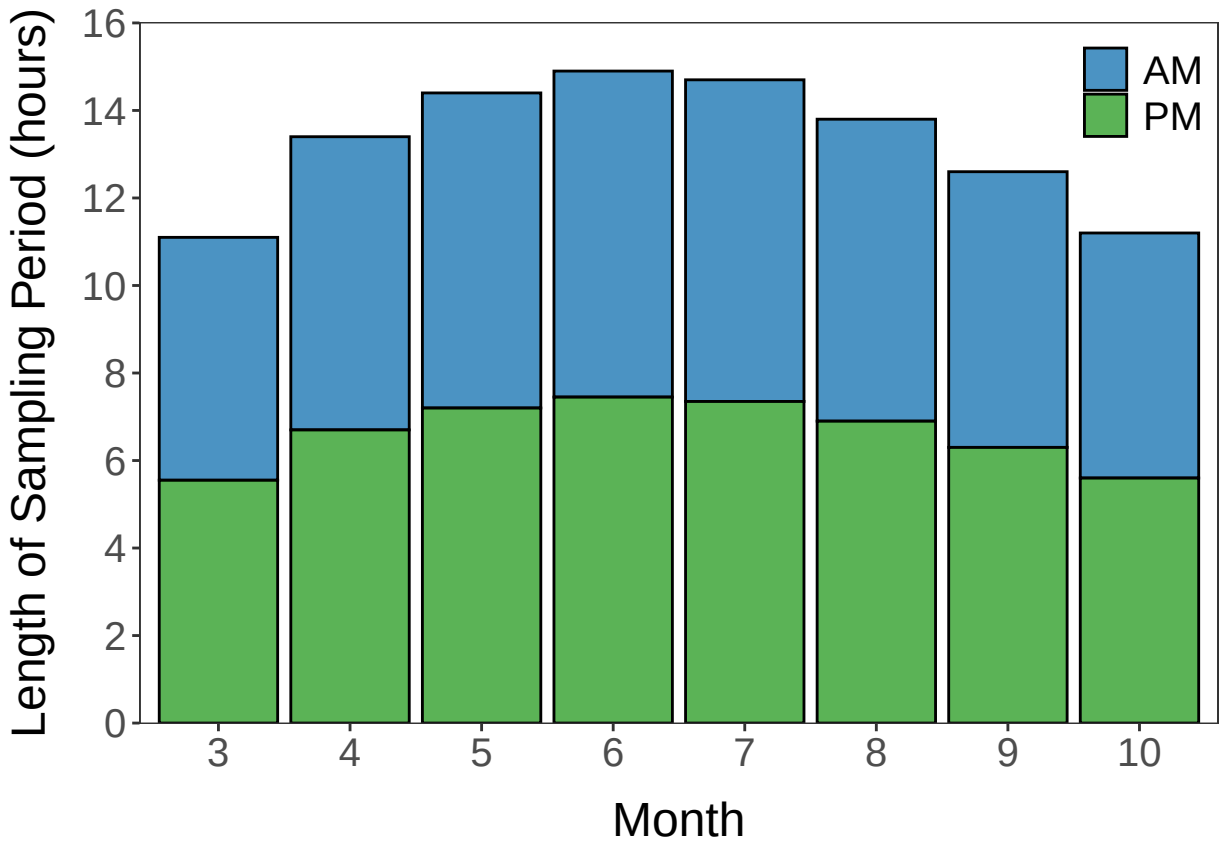


Add Time of Day Stratum

```
dat_sum_strat <- dat_sum %>%
  mutate(losp = lod/2) %>%    # length of day is divided in half to calculate AM and PM
  group_by_all() %>%
  expand(shift = c("AM", "PM")) %>%
```

```
  ungroup()

ggplot(dat_sum_strat, aes(x = month, y = losp, group = shift, fill = shift)) +
  geom_bar(stat = "identity", color = "black", alpha = 0.8) +
  scale_x_continuous(limits = c(2.5, 10.5), breaks = seq(1, 12, 1), expand = c(0, 0.1)) +
  scale_y_continuous(limits = c(0, 16), breaks = seq(0, 16, 2), expand = c(0, 0.02)) +
  scale_fill_manual(values = c("#1f78b4", "#33a02c")) +
  labs(y = "Length of Sampling Period (hours)", x = "Month") +
  theme_bw() +
  theme(axis.title.x = element_text(size = 18, margin = margin(10, 0, 0, 0)),
        axis.title.y = element_text(size = 18, margin = margin(0, 10, 0, 0)),
        axis.text.x = element_text(size = 15),
        axis.text.y = element_text(size = 15),
        legend.position = "inside",
        legend.position.inside = c(0.93, 0.9),
        legend.background = element_rect(fill = "transparent"),
        legend.title = element_blank(),
        legend.text = element_text(size = 15),
        panel.grid = element_blank(),
        plot.margin = margin(10, 10, 5, 5))
```



Create Randomized Sampling Calendar

Example 1: No Stratum

```
## set seed to reproduce randomized results
set.seed(65775744)

## randomly select 16 days
cal_rand <- cal %>%
  # number of days
  slice_sample(n = 16) %>%
  ungroup() %>%
  arrange(month, day)
```

date	year	month	day	wday	dow	week
2026-03-20	2026	3	20	Friday	wd	12
2026-03-31	2026	3	31	Tuesday	wd	14
2026-04-04	2026	4	4	Saturday	we	14
2026-04-12	2026	4	12	Sunday	we	15
2026-04-23	2026	4	23	Thursday	wd	17
2026-04-28	2026	4	28	Tuesday	wd	18
2026-05-11	2026	5	11	Monday	wd	20
2026-05-30	2026	5	30	Saturday	we	22
2026-06-01	2026	6	1	Monday	wd	23
2026-06-09	2026	6	9	Tuesday	wd	24
2026-06-12	2026	6	12	Friday	wd	24
2026-06-16	2026	6	16	Tuesday	wd	25
2026-06-27	2026	6	27	Saturday	we	26
2026-07-29	2026	7	29	Wednesday	wd	31
2026-07-31	2026	7	31	Friday	wd	31
2026-09-16	2026	9	16	Wednesday	wd	38

Example 2: Stratified by Time of Day (TOD) and Day of Week (DOW)

```
## set seed to reproduce randomized results
set.seed(254929024)

## define strata
tod_levels <- c("AM", "PM")

## randomly select two weekdays per week and randomly assign a stratification
## (e.g., AM vs PM) with equal probability
cal_wd_one <- cal %>% filter(dow == "wd") %>%
  # group by a variable to ensure equal distribution within the levels of
  # selected variable. In this case, week was selected.
  group_by(week) %>%
  # number of days
  slice_sample(n = 2) %>%    # change value here to alter the number of days sampled
  ungroup() %>%
```

```r
  # randomly assign strata
  mutate(shift = sample(rep(tod_levels, each = ceiling(n()/length(tod_levels))),
                        size = n(), replace = FALSE))

## randomly select six weekend days per month and randomly assign a stratification
## (e.g., AM vs PM) with equal probability
cal_we_one <- cal %>% filter(dow == "we") %>%
  # number of days
  slice_sample(n = 6) %>%     # change value here to alter the number of days sampled
  ungroup() %>%
  # randomly assign strata
  mutate(shift = sample(rep(tod_levels, each = ceiling(n()/length(tod_levels))),
                        size = n(), replace = FALSE))

## Combine randomized calendar
cal_rand_one <- bind_rows(cal_wd_one, cal_we_one) %>%
  arrange(month, day)
```

date	year	month	day	wday	dow	week	shift
2026-03-02	2026	3	2	Monday	wd	10	PM
2026-03-04	2026	3	4	Wednesday	wd	10	PM
2026-03-10	2026	3	10	Tuesday	wd	11	AM
2026-03-12	2026	3	12	Thursday	wd	11	PM
2026-03-17	2026	3	17	Tuesday	wd	12	PM
2026-03-19	2026	3	19	Thursday	wd	12	AM
2026-03-23	2026	3	23	Monday	wd	13	AM
2026-03-27	2026	3	27	Friday	wd	13	PM
2026-03-31	2026	3	31	Tuesday	wd	14	AM
2026-04-03	2026	4	3	Friday	wd	14	AM
2026-04-06	2026	4	6	Monday	wd	15	PM
2026-04-07	2026	4	7	Tuesday	wd	15	PM
2026-04-14	2026	4	14	Tuesday	wd	16	PM
2026-04-15	2026	4	15	Wednesday	wd	16	PM
2026-04-21	2026	4	21	Tuesday	wd	17	PM
2026-04-22	2026	4	22	Wednesday	wd	17	AM
2026-04-25	2026	4	25	Saturday	we	17	PM
2026-04-27	2026	4	27	Monday	wd	18	AM
2026-04-28	2026	4	28	Tuesday	wd	18	PM
2026-05-06	2026	5	6	Wednesday	wd	19	PM

Example 3: Stratified by Time of Day (TOD), Day of Week (DOW), and Access Sites

```r
## set seed to reproduce randomized results
set.seed(32104934)

## define strata
tod_levels <- c("AM", "PM")
site_levels <- c("A", "B", "C")
# create a data frame with all possible combinations of strata
```

```r
all_levels <- expand_grid(tod_levels, site_levels)

## define effort
n_per_week_wd <- 3       # change value here to alter the number of days sampled
n_per_month_we <- 4      # change value here to alter the number of days sampled

## define number of weeks and months in survey period
n_week <- cal %>% filter(dow == "wd") %>% distinct(week) %>% nrow()
n_month <- cal %>% filter(dow == "we") %>% distinct(month) %>% nrow()

## randomly select n weekdays per week and randomly assign both strata
## with equal probability
cal_wd_mult <- cal %>% filter(dow == "wd") %>%
  # group by a variable to ensure equal distribution within the levels
  # of selected variable
  group_by(week) %>%
  # number of days
  slice_sample(n = n_per_week_wd) %>%
  ungroup()

# randomly assign strata (e.g., time of day and access point)
rand_wd_mult <- bind_rows(replicate(n_week, all_levels %>%
                                      sample_n(n_per_week_wd, replace = FALSE),
                                    simplify = F)) %>%
  rename(shift = 1, site = 2)
  # There is a chance that the randomized component of the code could lead to a
  # slight difference between the number of rows that are selected in the calendar
  # above and the number of rows in the random strata data frame build here. This
  # would lead to an error here. In that case, adding the line of code below will
  # remove the error and ensure the two data frames match.
  #
  ### %>% slice(1:nrow(cal_wd_mult))

# combine random dates with randomized strata
cal_wd_mult %<>% bind_cols(rand_wd_mult)

## randomly select n weekend days per month and randomly assign both strata
## with equal probability
cal_we_mult <- cal %>% filter(dow == "we") %>%
  # group by a variable to ensure equal distribution within the levels of
  # selected variable
  group_by(month) %>%
  # number of days
  slice_sample(n = n_per_month_we) %>%
  ungroup()

# randomly assign strata (e.g., time of day and access point)
rand_we_mult <- bind_rows(replicate(n_month, all_levels %>%
                                      sample_n(n_per_month_we),
                                    simplify = F)) %>%
  rename(shift = 1, site = 2)
  # There is a chance that the randomized component of the code could lead to a
  # slight difference between the number of rows that are selected in the calendar
```

```r
  # above and the number of rows in the random strata data frame build here. This
  # would lead to an error here. In that case, adding the line of code below will
  # remove the error and ensure the two data frames match.
  #
  ### %>% slice(1:nrow(cal_we_mult))

# combine random dates with randomized strata
cal_we_mult %<>% bind_cols(rand_we_mult)

## Combine randomized calendar
cal_rand_mult <- bind_rows(cal_wd_mult, cal_we_mult) %>%
  arrange(month, day)
```

date	year	month	day	wday	dow	week	shift	site
2026-03-01	2026	3	1	Sunday	we	9	AM	B
2026-03-02	2026	3	2	Monday	wd	10	PM	A
2026-03-03	2026	3	3	Tuesday	wd	10	PM	C
2026-03-06	2026	3	6	Friday	wd	10	PM	B
2026-03-09	2026	3	9	Monday	wd	11	AM	C
2026-03-12	2026	3	12	Thursday	wd	11	PM	A
2026-03-13	2026	3	13	Friday	wd	11	PM	C
2026-03-14	2026	3	14	Saturday	we	11	AM	A
2026-03-16	2026	3	16	Monday	wd	12	AM	C
2026-03-18	2026	3	18	Wednesday	wd	12	PM	C
2026-03-20	2026	3	20	Friday	wd	12	AM	B
2026-03-21	2026	3	21	Saturday	we	12	PM	C
2026-03-24	2026	3	24	Tuesday	wd	13	PM	A
2026-03-25	2026	3	25	Wednesday	wd	13	AM	B
2026-03-26	2026	3	26	Thursday	wd	13	AM	A
2026-03-29	2026	3	29	Sunday	we	13	AM	C
2026-03-31	2026	3	31	Tuesday	wd	14	PM	C
2026-04-01	2026	4	1	Wednesday	wd	14	AM	C
2026-04-03	2026	4	3	Friday	wd	14	PM	B
2026-04-05	2026	4	5	Sunday	we	14	PM	A