Creel Survey - Stratified Random Sampling

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 <- "2023/03/01"
end_date <- "2023/10/31"
## 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 2023-03-01 2023     3   1 Wednesday  wd    9
## 2 2023-03-02 2023     3   2  Thursday  wd    9
## 3 2023-03-03 2023     3   3    Friday  wd    9
## 4 2023-03-04 2023     3   4  Saturday  we    9
## 5 2023-03-05 2023     3   5    Sunday  we    9
## 6 2023-03-06 2023     3   6    Monday  wd   10
```

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

```r
## Define daily sunrise and sunset times
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 2023-03-01 41.18613 -111.3813 2023-03-01 07:01:44 2023-03-01 18:17:01     3
## 2 2023-03-02 41.18613 -111.3813 2023-03-02 07:00:11 2023-03-02 18:18:11     3
## 3 2023-03-03 41.18613 -111.3813 2023-03-03 06:58:36 2023-03-03 18:19:20     3
## 4 2023-03-04 41.18613 -111.3813 2023-03-04 06:57:01 2023-03-04 18:20:28     3
## 5 2023-03-05 41.18613 -111.3813 2023-03-05 06:55:26 2023-03-05 18:21:36     3
## 6 2023-03-06 41.18613 -111.3813 2023-03-06 06:53:49 2023-03-06 18:22:44     3
```

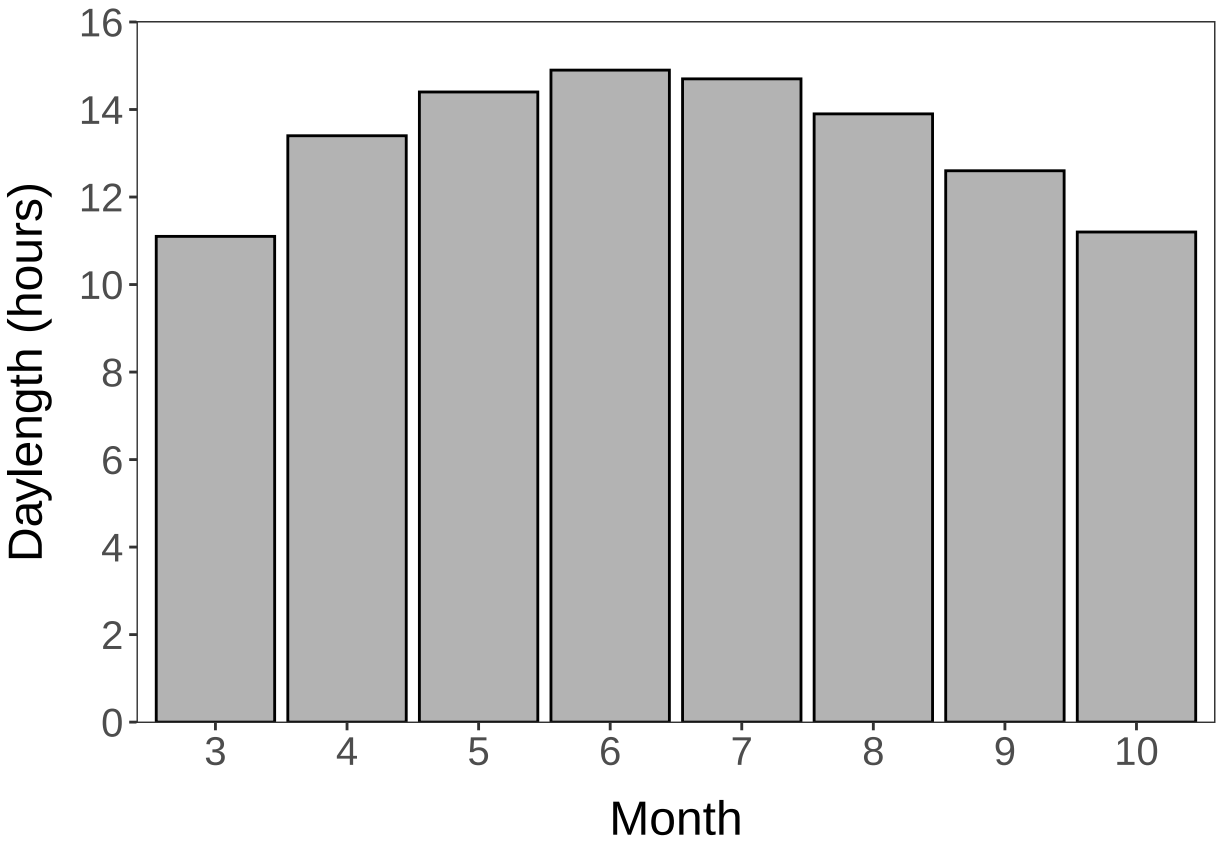
Examine Day Length During Sampling Period

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.

```r
## 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

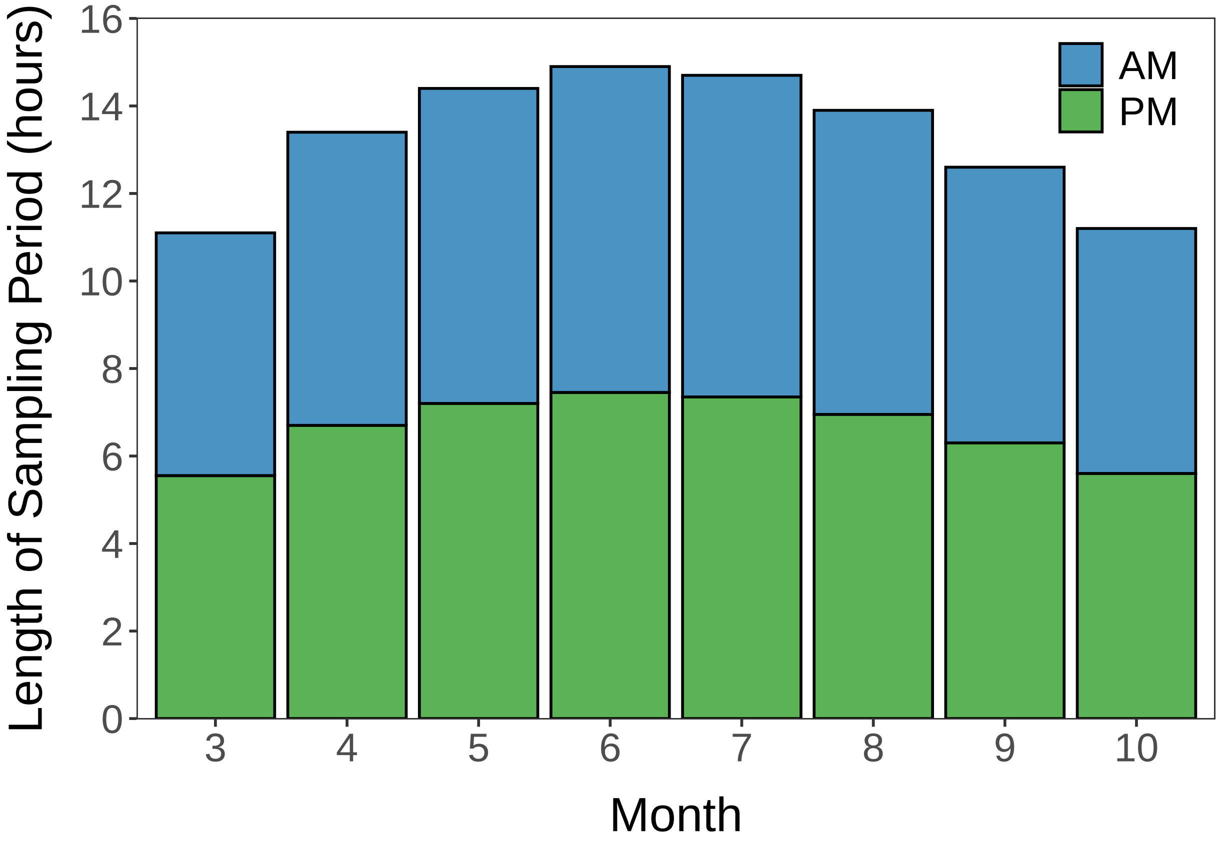
```
## plot
ggplot(dat_sum, aes(x = month, y = lod)) +
  geom_bar(stat = "identity", fill = "gray70", color = "black") +
  scale_x_continuous(limits = c(2.5, 10.5), breaks = seq(1, 12, 1), expand = c(0, 0.1)) +
  scale_y_continuous(limits = c(0, 16), breaks = seq(0, 16, 2), expand = c(0, 0.02)) +
  labs(y = "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) %>%
  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(0.91, 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

```r
## Restrict calendar to four weeks to simplify examples
cal <- cal %>%
  filter(week %in% 14:17)
```

Example 1: Stratified by Time of Day (TOD) and Day of Week (DOW)

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

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
  group_by(week) %>%
  # number of days
  slice_sample(n = 2) %>%
  ungroup() %>%
  # 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) %>%
  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
2023-04-03	2023	4	3	Monday	wd	14	PM
2023-04-05	2023	4	5	Wednesday	wd	14	AM
2023-04-08	2023	4	8	Saturday	we	14	AM
2023-04-09	2023	4	9	Sunday	we	14	PM
2023-04-11	2023	4	11	Tuesday	wd	15	AM
2023-04-13	2023	4	13	Thursday	wd	15	AM
2023-04-15	2023	4	15	Saturday	we	15	AM
2023-04-18	2023	4	18	Tuesday	wd	16	PM

date	year	month	day	wday	dow	week	shift
2023-04-20	2023	4	20	Thursday	wd	16	PM
2023-04-22	2023	4	22	Saturday	we	16	PM
2023-04-23	2023	4	23	Sunday	we	16	AM
2023-04-24	2023	4	24	Monday	wd	17	AM
2023-04-28	2023	4	28	Friday	wd	17	PM
2023-04-30	2023	4	30	Sunday	we	17	PM

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

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

site_levels <- c("A", "B", "C")

## randomly select two 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 = 2) %>%
  ungroup() %>%
  # randomly assign strata (e.g., time of day and access point)
  mutate(shift = sample(rep(tod_levels, each = ceiling(n()/length(tod_levels))),
                        size = n(), replace = FALSE),
         site = sample(rep(site_levels, each = ceiling(n()/length(site_levels))),
                       size = n(), replace = FALSE))

## randomly select six weekend days per month and randomly assign both strata with
## equal probability
cal_we_mult <- cal %>% filter(dow == "we") %>%
  # number of days
  slice_sample(n = 6) %>%
  ungroup() %>%
  # randomly assign strata (e.g., time of day and access point)
  mutate(shift = sample(rep(tod_levels, each = ceiling(n()/length(tod_levels))),
                        size = n(), replace = FALSE),
         site = sample(rep(site_levels, each = ceiling(n()/length(site_levels))),
                       size = n(), replace = FALSE))

## 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
2023-04-03	2023	4	3	Monday	wd	14	AM	C

date	year	month	day	wday	dow	week	shift	site
2023-04-07	2023	4	7	Friday	wd	14	PM	B
2023-04-11	2023	4	11	Tuesday	wd	15	PM	B
2023-04-14	2023	4	14	Friday	wd	15	AM	A
2023-04-15	2023	4	15	Saturday	we	15	AM	A
2023-04-16	2023	4	16	Sunday	we	15	PM	C
2023-04-17	2023	4	17	Monday	wd	16	PM	A
2023-04-20	2023	4	20	Thursday	wd	16	AM	A
2023-04-22	2023	4	22	Saturday	we	16	AM	C
2023-04-23	2023	4	23	Sunday	we	16	PM	A
2023-04-24	2023	4	24	Monday	wd	17	AM	B
2023-04-27	2023	4	27	Thursday	wd	17	PM	C
2023-04-29	2023	4	29	Saturday	we	17	AM	B
2023-04-30	2023	4	30	Sunday	we	17	PM	B